Universidad Rey Juan Carlos

Escuela Superior de Ciencias Experimentales y Tecnología

INGENIERÍA ELECTRÓNICA INDUSTRIAL Y AUTOMÁTICA

Trabajo Fin de Grado

PLATAFORMA PARA SITUACIONES DE EMERGENCIA CON DETECCIÓN DE OBJETOS, NAVEGACIÓN AUTÓNOMA Y ADQUISICIÓN DE PARÁMETROS AMBIENTALES EN ROS.

Autor: Silvia Iniesta Tejero

Tutora: María Cristina Rodríguez Sánchez

Cotutor: Juan Jesús Roldán Gómez

Curso académico 2022/23

Grado en Ingeniería Electrónica Industrial y Automática.

Trabajo Fin de Grado

El presente trabajo, titulado **PLATAFORMA PARA SITUACIONES DE EMERGENCIA CON DETECCIÓN DE OBJETOS, NAVEGACIÓN AUTÓNOMA Y ADQUISICIÓN DE PARÁMETROS AMBIENTALES EN ROS.**, constituye la memoria correspondiente a la asignatura Trabajo de Fin de Grado que presenta **Silvia Iniesta Tejero** como parte de su formación para aspirar al Título de Grado en Ingeniería Electrónica Industrial y Automática. Este trabajo ha sido realizado en la **Escuela Superior de Ciencias Experimentales y Tecnología** de la Universidad Rey Juan Carlos en el **Departamento de Tecnología Electrónica** bajo la dirección de **María Cristina Rodríguez Sánchez** y **Juan Jesús Roldán Gómez**.

Móstoles, a 6 de julio de 2023

Agradecimientos

Me gustaría comenzar expresando mi profundo agradecimiento a mi tutora, la Dra. Mª Cristina Rodríguez Sánchez, por su confianza y apoyo incondicional desde el inicio de este proyecto. Su experiencia y orientación han sido fundamentales para el desarrollo y éxito de este trabajo. También quiero agradecer a mi cotutor, Juan Jesús Roldán Gómez, por compartir su conocimiento y experiencia en el campo de la robótica y la programación, lo cual ha enriquecido enormemente mi formación.

Además, quiero expresar mi gratitud a los profesionales del cuerpo de bomberos, en especial a Fernando Horta Velasco (Jefe Supervisor Bomberos CAM), por su valiosa colaboración y disposición a compartir su experiencia y conocimientos en situaciones de emergencia. Su aporte ha sido fundamental para comprender las necesidades reales de este ámbito y orientar el desarrollo del proyecto de manera efectiva.

Agradezco sinceramente a mi familia y amigos por su invaluable apoyo y constante aliento a lo largo de mi formación. Su respaldo incondicional ha sido un motor de motivación para enfrentar los desafíos y perseguir mis metas. Quiero expresar mi profunda gratitud a mis padres y a mi hermano, quienes han sido pilares fundamentales en mi camino, brindándome un sólido fundamento y siendo un ejemplo de tenacidad y dedicación. También quiero agradecer de manera especial a Nuria y a mi padre, que han sido referentes esenciales en mi vida

Por último, mi agradecimiento a todos los profesores que han formado parte de mi educación como ingeniera. Gracias por su dedicación, por transmitir su pasión por la enseñanza y por brindarme una formación de calidad. Su compromiso y entusiasmo han sido una inspiración para mí. Quiero hacer una mención especial a Emma Pérez Martín, quien ha sido una fuente constante de apoyo y motivación a lo largo de este camino. Su amistad y compañerismo han sido imprescindibles, gracias por cada momento compartido y por el mutuo crecimiento que hemos experimentado juntas.

A todas las personas mencionadas, mi más sincero agradecimiento por su contribución incalculable y por haber sido parte de mi trayectoria académica y personal. Sin su apoyo y colaboración, este trabajo no habría sido posible. Estoy profundamente agradecida y valoro enormemente su influencia en mi desarrollo profesional y personal.

Resumen

En una situación de emergencia en la que se pierden vidas se evidencia la necesidad de contar con medios tecnológicos que puedan contribuir a mitigar el accidente. Incendios, explosiones o fugas de agentes tóxicos son eventos en los que contar con información previa del entorno resulta crucial para que el equipo de emergencias pueda actuar de manera efectiva y segura. Es por ello que el desarrollo de tecnologías que brinden ayuda en estas situaciones puede significar la diferencia entre la vida y la muerte, reduciendo los daños y el peligro asociado.

Este Trabajo de Fin de Grado tiene como objetivo principal ampliar las capacidades de un sistema existente, mediante la creación de una nueva plataforma que incorpore un sistema de detección de objetos y la capacidad de navegación autónoma en el entorno. Además, se agregarán nuevos sensores con el fin de enriquecer el conocimiento del entorno por parte de la plataforma.

Para lograr estas implementaciones, se requerirá un rediseño de la plataforma existente, adaptándola a ROS (Robot Operating System) un nuevo software robótico. La integración con ROS pone a disposición algoritmos de navegación y permite el desarrollo de las nuevas funcionalidades del robot. Esto implica la revisión y actualización de la arquitectura del sistema, la integración de nuevos módulos y la implementación de algoritmos de detección y navegación.

El objetivo final es mejorar significativamente la capacidad de respuesta ante emergencias, brindando al equipo de emergencias una plataforma tecnológica más robusta y avanzada que les permita obtener información precisa del entorno y tomar decisiones fundamentadas. Esto contribuirá a una actuación más efectiva y segura durante las situaciones críticas, salvaguardando vidas y minimizando los riesgos involucrados.

Este trabajo se enmarca dentro de una de las líneas de investigación de la cátedra SMARTE2, la cual se centra en el desarrollo de aplicaciones de ingeniería en colaboración con los cuerpos de emergencias de Madrid 112, la Asociación Profesional de Bomberos y la Jefatura de Bomberos de Alcorcón. El objetivo principal es poder integrar este trabajo en gemelos digitales como parte del proyecto GUIDE2FR, liderado por la cátedra.

Índice general

Índice de figuras

Acrónimos y abreviaturas

5G: Fifth Generation Internet (Internet de quinta generación)

IOT: Internet of Things (Internet de las cosas)

PSD: Power Spectral Density (Densidad Espectral de Potencia)

2D: Two Dimensions (Dos dimensiones)

3D: Three Dimensions (Tres dimensiones)

RGT-V: Radar Gas sensors Thermal Cameras - Vision

LIDAR: Light Detection and Ranging (Detección y Rastreo de Luz)

UWB: Ultra Wide Band (Banda Ultra Ancha)

ROS: Robot Operating System (Sistema Operativo de Robots)

SLAM: Simultaneous Localization and Mapping (Localización y Mapeo Simultáneos)

BBDD: Database (Base de Datos)

CSV: Comma-Separated Values (Valores Separados por Comas)

YOLO: You Only Look Once (Solo Miras una Vez)

DNS: Domain Name System (Sistema de Nombres de Dominio)

CNN: Convolutional Neural Network (Red Neuronal Convolucional)

IDE: Integrated Development Environment (Entorno de Desarrollo Integrado)

PWM: Pulse Width Modulation (Modulación por Ancho de Pulso)

SRAM: Static Random Access Memory (Memoria de Acceso Aleatorio Estática)

EEPROM: Electrically Erasable Programmable Read-Only Memory (Memoria de Solo Lectura Programable y Borrable Eléctricamente)

USB: Universal Serial Bus (Bus Universal en Serie)

I2C: Inter-Integrated Circuit (Circuito Interintegrado)

SCL: Serial Clock (Reloj en Serie)

SDA: Serial Data (Datos en Serie)

HDMI: High Definition Multimedia Interface (Interfaz Multimedia de Alta Definición)

GPIO: General Purpose Input/Output (Entrada/Salida de Propósito General)

GPS: Global Positioning System (Sistema de Posicionamiento Global)

TCP: Transmission Control Protocol (Protocolo de Control de Transmisión)

IP: Internet Protocol (Protocolo de Internet)

UDP: User Datagram Protocol (Protocolo de Datagramas de Usuario)

SPI: Serial Peripheral Interface (Interfaz Periférica en Serie)

TVOC: Total Volatile Organic Compounds (Compuestos Orgánicos Volátiles Totales)

eCO2: Equivalent Carbon Dioxide (Dióxido de Carbono Equivalente)

H2: Hydrogen (Hidrógeno)

DC: Direct Current (Corriente Continua)

SSH: Secure Shell (Shell Seguro)

Capítulo 1

Introducción

1.1 Presentación

1.1.1 Situación actual del entorno derivado de una emergencia

En el mundo actual, la tecnología va más rápido que el tiempo. Los cambios tecnológicos avanzan centrándose en un usuario común y en un servicio dedicado al día a día. Sin embargo, cuando sucede un accidente o una situación de peligro en la que se pierden vidas o salud, se reconoce fácilmente la falta de medios que existe en ciertos sectores. Esta demanda de mejora y la integración de la tecnología en estas situaciones ha sido una de las motivaciones principales para llevar a cabo este proyecto.

En el pasado año 2021 se produjeron aproximadamente 17.400 incendios o explosiones en viviendas en España, en los cuales más de 200 personas perdieron la vida, siendo la causa principal del fallecimiento la intoxicación por humo y/o gases tóxicos producidos por el fuego [9]. La concentración de gases en los incendios en interiores es fundamental, el humo está a una temperatura muy elevada, si los gases se encuentran en un estado de inflamabilidad puede provocar que el fuego se siga extendiendo por el entorno. De esta idea nace el concepto de "plano neutro", el plano neutro diferencia la zona en la que se acumula el humo y los gases de lo denominado zona limpia. Los gases y el humo ascienden formando una densa capa en la parte superior de la habitación cuando no existe salida (ventana, puerta...). Este aspecto favorece la creación de una plataforma terrestre como herramienta de asistencia en situaciones de emergencia, ya que se moverá en un entorno de menor temperatura, más limpio y menos perjudicial para la toma de datos.

En una situación de explosión, la capacidad de prevención es mayor, ya que la mayoría de las veces que surgen estos incidentes se debe a una previa fuga de gases, por ejemplo, una explosión de butano. Las explosiones ocurren cuando se forma una mezcla de gases con oxígeno, una mínima chispa puede producir una explosión o un incendio. Por ello, si se nota una fuerte presencia de gas natural en una sala, es importante tomar las precauciones necesarias (abrir ventanas, cortar el foco del gas, apagar luces...) o llamar al equipo de emergencias, que tomará las medidas de concentración de estos gases. El trabajo desarrollado pretende poder cubrir esta función midiendo los valores de concentración de gases, la temperatura y la humedad de la sala.

Es evidente la importancia que se debe prestar a prevenir estas situaciones de emergencia y a fortalecer los equipos de actuación tanto a nivel de personal como con medios tecnológicos.

Para ilustrar debidamente el entorno producido en una situación de emergencia se ha realizado un estudio con el equipo de bomberos. En este estudio han participado los Parques de Bomberos de los

municipios de Alcorcón y de Leganés. Se ha realizado un formulario con varias preguntas acerca de los tiempos de operación, los parámetros más relevantes y la inversión en el sector. Han participado ocho bomberos y se han alcanzado las siguientes conclusiones:

1. La información previa de la situación de emergencia es muy importante, destacando la presencia de víctimas, la temperatura y la concentración de componentes tóxicos. Además, los obstáculos tales como muebles, objetos o puertas cerradas adquieren relevancia en la intervención, ya que pueden ralentizar la operación.

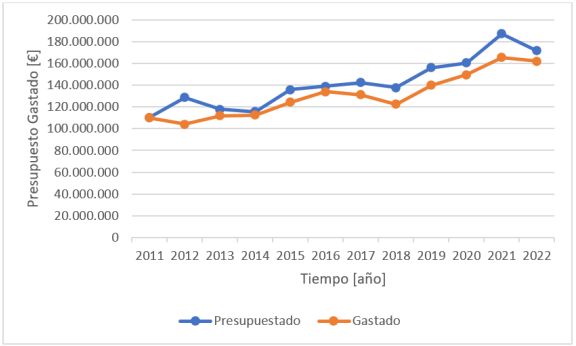
2. En cuanto al tiempo de intervención en este tipo de emergencias se establece una media de 45 minutos. Sin tener en cuentas los aproximadamente 10 minutos que trascurren desde que llega el aviso hasta que el cuerpo de bomberos accede a la ubicación.

3. La mayoría manifiestan su disconformidad con la inversión realizada en la investigación en este sector. Presentan la necesidad de invertir en el desarrollo de nuevas tecnologías que ayuden a los profesionales a mejorar su respuesta ante las situaciones de emergencia. Además, piden que se realice mayor labor de divulgación para mejorar la actuación de los civiles en caso de un escenario de peligro.

Para ilustrar debidamente el último punto, se presenta un estudio acerca del presupuesto invertido y consumido por este sector en la Comunidad de Madrid. En la figura 1.1a se exponen los datos oficiales proporcionados por la propia comunidad autónoma del presupuesto invertido y el desembolso por parte del sector de bomberos [10]. En la segunda figura (figura 1.1b) se ha realizado una gráfica que resume los datos recogidos en la tabla para ilustrar la mínima diferencia entre las dos variables.

Año	Presupuestado	Gastado	Porcentaje
2011	110.273.117	110.003.017	99,76%
2012	128.754.239	104.250.658	80,97%
2013	117.626.492	111.994.642	95,21%
2014	115.546.350	112.440.772	97,31%
2015	135.837.837	124.189.930	91,43%
2016	138.894.715	133.861.276	96,38%
2017	142.320.471	131.137.000	92,14%
2018	137.855.719	122.453.774	88,83%
2019	156.145.800	139.929.794	89,61%
2020	160.273.695	149.408.711	93,22%
2021	187.109.486	165.409.183	88,40%
2022	171.556.000	161.892.000	94,37%

(a) Estudio sobre la inversión en el sector de bomberos de la Comunidad de Madrid.



(b) Tabla del presupuesto invertido y lo gastado por el sector de bomberos en los últimos 10 años.

Figura 1.1: Representación del presupuesto invertido y lo gastado por el sector de bomberos en los últimos 10 años.

A partir de la encuesta realizada y las ilustraciones anteriores, se aprecia claramente que los recursos requeridos por el cuerpo de bomberos son superiores a lo invertido en este sector, ya que todos los años se ha superado el 85 % de desembolso del presupuesto. También es apreciable en las declaraciones de los profesionales ante la encuesta presentada.

Por ello, uno de los objetivos de este trabajo es brindar una herramienta de bajo coste al equipo de bomberos que les permita actuar de forma más adecuada sobre una situación de emergencia, ofreciendo la mayor cantidad de información posible en el tiempo previo a la intervención.

1.1.2 Contexto de la robótica

Para comenzar, es necesario establecer qué es la robótica. La robótica se conoce como la ciencia que estudia el diseño y construcción de máquinas capaces de desempeñar las tareas del ser humano mediante procesos mecanizados y programados [11]. Joseph L. Jones en el artículo "Mobile Robots: Inspiration to implementation"describió la robótica de la siguiente forma:

> *"La robótica consiste en el diseño de sistemas. Actuadores de locomoción, manipuladores, sistemas de control, sensores, fuentes de energía, software de calidad–todos estos subsistemas tienen que ser diseñados para trabajar conjuntamente en la consecución de la tarea del robot."*

Por su parte, García-Prieto Cuesta considera al robot como una máquina, provista de cierta complejidad tanto en sus componentes como en su diseño o en su comportamiento, y que manipula información acerca de su entorno para así interactuar con él [12].

Quizás ambos conceptos conformen la definición más certera y coincidente con la plataforma que se va a desarrollar en este proyecto, resaltando la importancia de que el robot actúe acorde a los parámetros recibidos por el entorno.

En el siglo XXI se ha producido la conocida Revolución Digital. Este suceso ha marcado el desarrollo de todas las tecnologías, y como no, de la robótica, permitiendo integrar las innovaciones de la informática en la robótica tradicional. Además, cabe destacar gran avance que ha sufrido la inteligencia y la visión artificial, lo que ha producido un efecto directo en el crecimiento de la robótica en los sectores industriales y sociales. Actualmente, las principales líneas de trabajo de la robótica son [13]:

- Aprendizaje de nuevas tareas: gracias al 'machine learning' y a la inteligencia artificial, los robots son capaces de realizar actividades cada vez más complejas, esto implica la innovación en los protocolos de comunicación entre dispositivos.
- Aplicaciones sociales: los robots, ya no se limitan a trabajar en fábricas, el gran avance en la ingeniería robótica está abriendo un amplio abanico de trabajo en los sectores sociales. Desde drones utilizados por los equipos de seguridad del estado, hasta dispositivos complejos como el famoso asistente quirúrgico "Da Vinci". No solo evoluciona la ingeniería que hay detrás de estos dispositivos, también lo hace la confianza de las personas en dichos equipos, este factor es fundamental para la incorporación de sistemas robóticos en los sectores sociales.

Este crecimiento es claramente exponencial, el progreso que se va a llevar a cabo en los próximos años es inimaginable, sin embargo, se predice que con la red de 5G se incremente la capacidad de trabajo de los robots y la comunicación derivada de las tecnologías IoT.

La sinergia entre la inteligencia artificial y la robótica tiene grandes ventajas, entre ellas destaca la consecución de los Objetivos de Desarrollo Sostenible. Se están desarrollando sistemas robóticos que buscan brindar asistencia en áreas afectadas por la pobreza en diversos campos: atención médica, educación, producción, seguridad... En el informe "Fourth Industrial Revolution for Earth" de Naciones Unidas, se plantean propuestas para solucionar los principales retos que presenta el planeta hoy en día, desde el uso de recursos naturales hasta la biodiversidad y las situaciones catastróficas [14].

Como se ha mencionado anteriormente, la visión artificial está jugando un papel muy importante en el desarrollo de la robótica. Las líneas principales de aplicación se basaban en el uso de cámaras y redes neuronales para los procesos de control de calidad, en los que se ha alcanzado una precisión mayor a la del ojo humano. La visión artificial se enfoca principalmente en aspectos cuantitativos, mientras que los seres humanos pueden cometer errores debido al cansancio, el estrés y otros factores psicológicos.

1.2 Estado del arte

En la sección 1.1.2 se ha detallado la importancia que tiene la robótica en la actualidad y la que tendrá en los próximos años. Esta oportunidad no puede pasar desapercibida para intentar mitigar los riesgos a los que se puede exponer el ser humano, como catástrofes naturales, incendios, terremotos... Por ello, en este apartado se exponen los trabajos punteros en asistencia en situaciones de emergencia, que han servido de inspiración para desarrollar este agente autónomo.

En la actualidad hay numerosos proyectos de robótica cuyo objetivo es ser de ayuda en situaciones de emergencia. Uno de los países punteros en el desarrollo de esta tecnología es Japón, dada la naturaleza y las condiciones geográficas de la nación, sus investigaciones se centran en paliar los efectos de los terremotos, de esta necesidad nació el sistema Quince.

Quince [15] es un sistema que responde a situaciones de emergencias con un alto nivel de incertidumbre, en el que puede variar el entorno de forma impredecible. Mecánicamente, este robot está equipado con un sistema de orugas que rodean el cuerpo y permiten un buen agarre a la superficie. Uno de los objetivos principales de Quince es la capacidad de adaptación a las situaciones más extremas, se ha probado su versatilidad en escenarios de nieve y está preparado para operar tras terremotos. El sistema de navegación se rige por los datos recogidos del conjunto de sensores que lo constituye: acelerómetros, giroscopio, sensor de corriente, sensor PSD y una cámara.

Mediante los acelerómetros y el giroscopio, se lleva a cabo el control mecánico, determinando así la velocidad y la dirección, también se apoya en el sensor PSD que es el encargado de detectar obstáculos. Dejando a un lado la transversalidad en relación con los terrenos que tiene este robot, otro de los objetivos es ofrecer información acerca del entorno en el que ha sucedido la tragedia. Por ello, la cámara juega un rol muy importante, ya que será la fuente de información que permita al usuario tener el conocimiento del contexto en el que se va a adentrar. Otra función que puede desempeñar Quince es la elaboración de un mapa de entorno mediante la incorporación de un escáner láser. Esta capacidad le permite generar mapas incluso en entornos con condiciones de iluminación limitada. Este agente se ha utilizado en operaciones reales y su desempeño ha sido un éxito. La figura 1.2 muestra como es este robot.

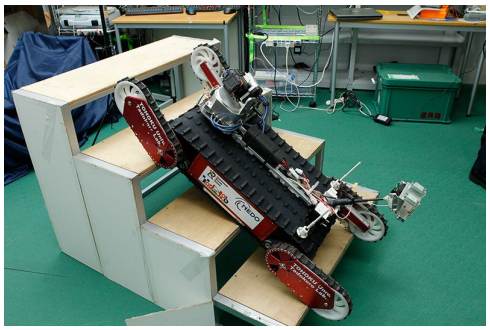


Figura 1.2: Agente de asistencia en terrenos de condiciones adversas Quince

En Europa también se están desarrollando prototipos centrados en facilitar las operaciones a los equipos de emergencia. En concreto, SmokeBot [16] nace de la necesidad de ofrecer soluciones en entornos de visibilidad reducida. Tiene como cometido, proporcionar información en operaciones de gestión posteriores a incendios en entornos cerrados, como túneles. Cuenta con un módulo llamado RGT-V, formado por una cámara radar 3D, una cámara térmica estéreo y sensores de detección de compuestos de gran ancho de banda. Además, este dispositivo permitiría mediciones LIDAR, que consisten en datos topográficos para elaborar mapas y modelos de terreno. En este caso, la funcionalidad del agente consiste en proporcionar

información de calidad en entornos de baja visibilidad. Por ello, se destina gran arte de su presupuesto en el desarrollo de los componentes que proporcionan la información y no tanto a la versatilidad del robot en diferentes terrenos.

Se encuentra en fase experimental, pero su proyección en un futuro es muy prometedora. La arquitectura de este robot se puede ver en la siguiente imagen (figura 1.3):

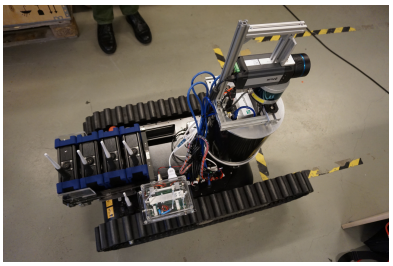


Figura 1.3: Agente de asistencia en entornos de baja visibilidad SmokeBot

Cabe destacar otro trabajo de mayor complejidad y más avanzado desarrollado por la compañía francesa "Shark Robotics" [17]. Se trata de Colossus, un robot diseñado íntegramente para trabajar con el equipo de bomberos y actuar en situaciones de emergencia como una herramienta más durante la operación. Son múltiples las funcionalidades de este equipo, es capaz de operar de forma autónoma durante 12 horas, puede lidiar con pesos de hasta una tonelada y trabajar en condiciones de 800º C. Además, es capaz de interpretar órdenes y transmitir información a una distancia de hasta 1,2 kilómetros.

Colossus [18] está equipado con ocho módulos, entre los que destacan un ventilador de gran caudal, un porta camillas, y un equipo de extinción de fuego. Su peso es de 525 Kg, su movimiento se rige por dos motores de 4000 watios y su velocidad de actuación es de hasta 7 Km/h. Como todos los robots citados anteriormente, cuenta con equipo de 6 cámaras con tecnología térmica que permiten al equipo de bomberos tener una visión completa de la situación de emergencia.

Este equipo se ha probado en situaciones reales, entre las que se resalta la operación de extinción del incendio de la Catedral de Notre Dame. Su actuación fue un éxito, sobresale la importancia de estos equipos en operaciones extensas en las que los profesionales alcanzan situaciones límite tanto físicamente como psicológicamente. En la figura 1.4 se muestra una imagen del agente Colossus.

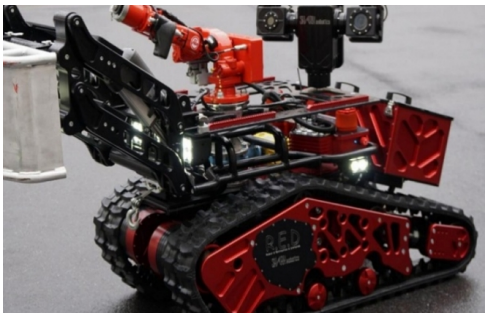


Figura 1.4: Agente autónomo de asistencia en incendios Colossus

Desde la Universidad Rey Juan Carlos se han elaborado varios proyectos siguiendo esta línea de trabajo. El desarrollo de un sistema de navegación en interiores que a su vez pueda llevar a cabo una monitorización de parámetros se ha trabajado anteriormente por Diego Salas Prieto y Noelia Fernández Talavera.

El primer proyecto [19] constaba del desarrollo de un sistema de localización en interiores basado en tecnología UWB, además este equipo estaba equipado con la capacidad de recopilar datos de sensores, que luego se almacenaban en una base de datos. El sistema creado por Diego Salas Prieto era capaz de detectar obstáculos mediante un sistema de ultrasonidos. Las limitaciones que presentaba este agente eran principalmente de eficiencia, las funciones del robot se veían afectadas a medida que disminuía la batería de los motores, además algunas funcionalidades presentaban anomalías en su ejecución.

Por ello, se decidió trabajar en un equipo que resolviera dichas dificultades y en la integración de otros módulos. Este segundo proyecto [20] se incorporó un módulo de control de motores que actuara de forma más precisa, se mejoró el hardware del equipo, añadiendo sensores medioambientales, se aumentó la vida útil del sistema y se desarrolló una plataforma para visualizar los parámetros recogidos por el robot. Además, se mejoró el sistema de control manual que permitía al usuario dirigir el robot de forma controlada y se creó un entorno de simulación en Unity.

Las limitaciones que presenta este trabajo residen en la navegación autónoma que se quiere implementar en este proyecto, y su integración en otros entornos. La programación estaba limitada estructuralmente, ya que se desarrolló principalmente en Python. Por ello, en este proyecto se integra la plataforma en un nuevo sistema basado en ROS [21], se incorpora un paquete de navegación autónoma, y se integran nuevos sensores para desarrollar estas funcionalidades.

Además, dadas las tecnologías que se han abierto paso en estos últimos años, y tras el estudio de otros equipos diseñados para casos similares, se ha estimado como imprescindible la creación de un módulo de visión artificial que permita al agente autónomo proporcionar más información al equipo de seguridad que dirija la operación.

1.3 Motivación

Como se ha citado anteriormente, este trabajo nace de la necesidad de mejorar las capacidades y las funcionalidades de un sistema de navegación dirigido manualmente, y de suministrar una herramienta más completa al equipo de bomberos, atendiendo a las necesidades actuales presentadas en 1.1.1. La capacidad de proporcionar información acerca de las variables del entorno, permite al equipo de bomberos tomar decisiones más precisas, asegurando la eficacia y la seguridad en las intervenciones [22]. En este proyecto se va a integrar un módulo de visión artificial que permita al equipo de bomberos obtener imágenes previas a su intervención.

Por otro lado, la integración de la plataforma en un software más completo pretende favorecer el entorno de simulación y así poder incorporarla en un gemelo digital siguiendo las líneas de trabajo de la cátedra SMARTE2 [23] de la Universidad Rey Juan Carlos. Se integrará dentro del proyecto TD-GUIDE2FR [24].

Finalmente, una de las razones por las que se lleva a cabo este proyecto, es la necesidad de dotar de autonomía al robot, de esta forma los agentes no tendrán que dirigir continuamente al robot sino que podrán centrarse en los datos recogidos por la plataforma.

1.3.1 Integración en ROS

La integración del sistema anterior en ROS era imprescindible dadas las prestaciones que ofrece este software [25]:

Infraestructura de comunicación y herramientas software.
ROS utiliza un middleware, es decir, un software que proporciona la infraestructura de comunicaciones

para conectar componentes y procesos. Este middleware permite intercambiar datos de forma consciente a través del mismo canal. Esto simplifica la implementación de sistemas de computación distribuidos, habilitando así la posibilidad de ejecutar diferentes procesos desde un mismo dispositivo o varios. Además, su sistema de comunicaciones está fuertemente tipado, impidiendo cualquier intrusión en los datos compartidos.

Herramientas de alto nivel para implementación de aplicaciones robóticas.
Este proyecto pretende concebirse como una herramienta para situaciones de emergencia, por lo que debe poder adaptarse a circunstancias imprevistas. Esto exige que el dispositivo sea capaz de percibir los datos del entorno y convertirlos en información útil que le permita comprender el contexto en el que opera. ROS cuenta con multitud de paquetes que proporcionan algoritmos específicos según las necesidades del entorno. Facilitando la integración de la plataforma en diferentes escenarios.

Localización y elaboración de mapas para robots móviles
ROS permite la localización y la elaboración de mapas para robots en espacios interiores. Esta característica es muy distintiva, porque simplifica al incorporar bibliotecas específicas para llevar a cabo esta función. Esta tarea precisa de herramientas como láseres que proporcionen una nube de puntos para posteriormente poder elaborar el mapa, pero todas las herramientas software las proporciona ROS. Incorpora el algoritmo SLAM, que permite la localización y el mapeo de forma simultánea.

Planificación de trayectorias y navegación para robots móviles.
El apartado principal que ha motivado la integración del anterior sistema en ROS es precisamente la posibilidad de desarrollar un sistema de navegación autónoma. ROS ofrece un paquete de navegación "Navigation Stack" que facilita la generación de trayectorias evitando obstáculos. Se apoya en los datos de odometría, velocidad y orientación y produce comandos de velocidad como respuesta.

1.3.2 Módulo de Visión Artificial

Siendo una de las principales motivaciones poder ofrecer más información al equipo de bomberos, se ha decidido implementar un módulo de visión artificial.

La vista es el sentido que permite a los seres humanos percibir y comprender el entorno, la visión artificial pretende cumplir la misma función entendiendo y analizando una imagen. Se trata de simular esa fuente de información mediante programación y electrónica. El suministro de datos para las aplicaciones de visión artificial es proporcionado por sensores capaces de adquirir imágenes de una escena tridimensional. En este caso se utiliza una cámara webcam, aunque en otras aplicaciones puede obtenerse la imagen de satélites artificiales o incluso de láseres. El propósito de la visión artificial es desarrollar algoritmos programables para poder extraer la mayor información posible de una imagen.

El entorno se compone de 3 dimensiones, la mayoría de cámaras obtienen una imagen en 2D, esto puede generar una gran pérdida de información. Sin embargo, el uso de la imagen en 3 dimensiones, que puede captar otro tipo de dispositivos, provoca que el análisis sea más complejo, y a veces, menos intuitivo. Por ello, se ha optado por el uso de una cámara.

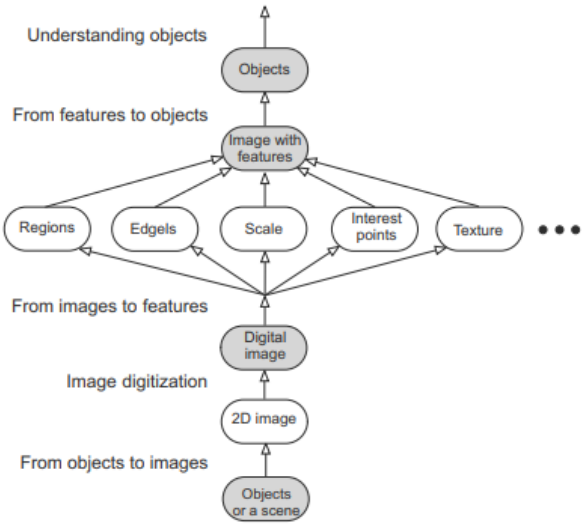


Figura 1.5: Los 4 niveles de representación de imagen adecuados para los problemas de análisis [1]

En la figura 1.5 se encuentran los 4 niveles de representación que conforman el análisis de imagen en la detección y clasificación de objetos. El proceso de análisis de imagen implica un sistema de representación robusto y fiable. El nivel más bajo hace referencia a la obtención de la imagen, es decir, el paso de objeto a imagen, sería el realizado por los sensores de visión. Un segundo paso, sería digitalizar la imagen, llevando a cabo un muestreo en el que se convierte en una matriz de píxeles que representan los puntos de la imagen. Posteriormente, se produce una cuantificación, que consiste en dar valor a esos píxeles mediante una cantidad de bits y finalmente, el almacenamiento, se guardan los valores en una estructura digital que permite ser analizada. El tercer nivel es el reconocimiento de regiones, bordes, escalas, puntos de interés y demás características. Y en último nivel analizan los resultados del nivel anterior y se clasifican los objetos.

En este proyecto es fundamental equipar al robot con un módulo de visión artificial ya que se obtiene un gran aporte de datos previos a la actuación del bombero. Este módulo es capaz de reconocer víctimas, mobiliario que puede obstaculizar la operación e incluso localizar bocas de incendios, por lo que puede resultar de gran utilidad en las intervenciones.

Se destaca esta innovación, que abre una nueva línea de trabajo en la plataforma e incluye los conceptos punteros que se están incorporando en las aplicaciones robóticas actuales.

1.4 Estructura de la memoria.

Esta memoria se divide en seis capítulos. En el primer capítulo se ha realizado una breve introducción para contextualizar el proyecto. Seguidamente, en el segundo capítulo, se exponen los objetivos que se pretenden abordar en este trabajo. En el tercer capítulo se describen las herramientas hardware y software utilizadas, así como la metodología que guía el proyecto. Hay un cuarto capítulo dedicado al diseño y a la implementación de cada módulo que conforma la plataforma. Finalmente, en el capítulo 5 se incluyen las pruebas realizadas y los resultados obtenidos de las mismas, y en sexto capítulo las conclusiones obtenidas de dichas pruebas y las futuras líneas de trabajo que ofrece este proyecto.

Capítulo 2

Objetivos

Una vez aclaradas las motivaciones que han llevado a realizar esta nueva plataforma, y tras revisar los diferentes modelos que existen en la actualidad, se procede a describir los objetivos planteados para llevar a cabo una plataforma eficiente, segura, de bajo coste y escalable.

2.1 Objetivos Generales

El objetivo de este proyecto es desarrollar una plataforma de navegación autónoma que brinde asistencia al equipo de bomberos durante situaciones de emergencia en entornos interiores. El propósito principal de este trabajo es proporcionar una solución efectiva y eficiente para abordar estos desafíos.

En primer lugar, para acometer este objetivo principal se ha de tener en cuenta el desarrollo de un sistema de navegación autónoma que permita al robot moverse de forma independiente en entornos interiores, evitando obstáculos y siguiendo rutas seguras. En segundo lugar, será necesario integrar nuevos sensores en el robot que mejoren su interpretación del entorno. Por último, se ha considerado imprescindible implementar una herramienta de visión artificial, que permita la detección de víctimas y objetos del entorno. Esto ayudará al equipo de bomberos a tener una visión precisa de la situación y tomar decisiones más seguras. Asimismo, se busca establecer una comunicación efectiva entre el robot y el equipo de bomberos. Esto implicará desarrollar un sistema de comunicación que permita intercambiar información en tiempo real y coordinar las acciones de manera eficiente.

Al cumplir estos objetivos, se espera lograr una solución sólida y efectiva para asistir al equipo de bomberos en situaciones de emergencia en interiores, mejorando su capacidad de respuesta y maximizando la seguridad de las víctimas y de los profesionales.

2.2 Objetivos Específicos

Acorde al objetivo general descrito, se han definido los siguientes objetivos específicos para cumplirlo:

1. Análisis de los requisitos de la nueva plataforma basándose en las limitaciones del antiguo agente y las tecnologías emergentes.
2. Formación en previas y nuevas plataformas para el desarrollo de los módulos principales del agente y sus correspondientes mejoras.

3. Despliegue de la infraestructura requerida para el desarrollo de la nueva plataforma.
4. Migración del desarrollo del robot a ROS (Robot Operating System) para facilitar su integración en gemelos digitales.
5. Incorporación de un sistema de navegación autónoma basado en el paquete de navegación "Navigation Stack"de ROS.
6. Integración de sistemas de visión artificial y redes neuronales que permitan la detección de víctimas y el reconocimiento de objetos.
7. Evaluación y pruebas de la nueva plataforma diseñada.
8. Documentación y recopilación de los resultados obtenidos.

Capítulo 3

Metodología y herramientas

Tras la motivación vista en el capítulo (1.3) y los objetivos planteados en la sección 2, se va a realizar una plataforma en la que se integrarán tres módulos: navegación, monitorización de parámetros y visión artificial. El escenario en el que debe operar esta plataforma se basa en una situación de emergencias ocurrida en un espacio interior, como una vivienda o una nave industrial. La condición de peligro se puede presentar como un incendio, una explosión o una fuga de gases en las que puede haber víctimas y el equipo de bomberos deba actuar.

La arquitectura de estos módulos se presenta en la figura 3.1.

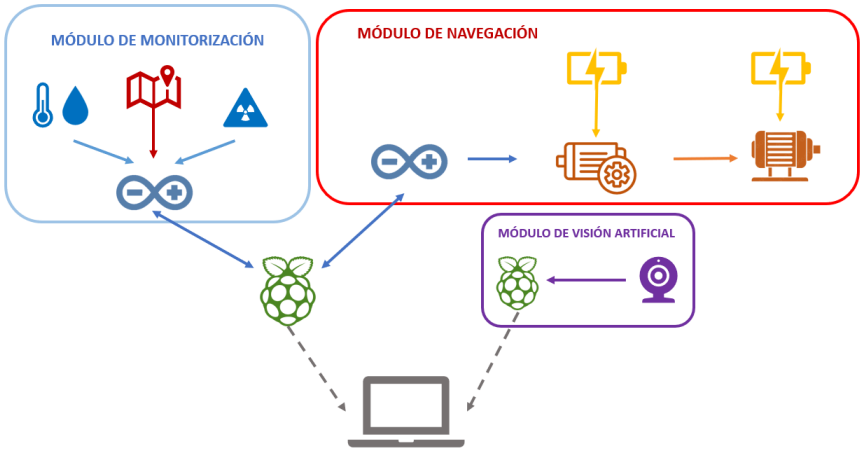


Figura 3.1: Arquitectura conceptual de la plataforma desarrollada.

3.1 Fases del proyecto

En la figura anterior, se observan los tres módulos que constituyen la plataforma. El primer módulo, el de monitorización de parámetros, consta de un sistema de balizas representado por el símbolo del mapa, y dos sensores, uno que toma la medida de temperatura y humedad, y otro que mide la concentración de varios compuestos, estos datos son recogidos por una placa Arduino One. El segundo módulo está formado por dos baterías que alimentan los motores y el controlador de motores, ambos elementos incluidos en esta sección, y una segunda placa Arduino que proporciona indicaciones al controlador. Tanto el módulo de navegación como el módulo de monitorización se comunican con una segunda Raspberry Pi para el

intercambio de información. Finalmente, el módulo de visión artificial, está formado por una Raspberry Pi y una cámara webcam.

Una vez presentada la motivación en el apartado 1.3 y los objetivos vistos en la sección 2 se plantea el desarrollo de una plataforma que siga la estructura de la figura 3.1. Para ello se dividirá el proyecto en las siguientes fases:

1. Fase 1: Puesta en marcha del agente anterior (4 semanas).
2. Fase 2: Implementación de la infraestructura base del proyecto (5 semanas).
3. Fase 3: Integración del agente en ROS y rediseño de la programación (8 semanas).
4. Fase 4: Desarrollo del módulo de visión artificial (3 semanas).
5. Fase 5: Creación del sistema de navegación (8 semanas).
6. Fase 6: Pruebas y documentación (6 semanas).

Fase 1: Puesta en marcha del modelo previo

Se toma como base el robot utilizado en los trabajos anteriores descritos en la sección 1.2. Para comprender las necesidades y el funcionamiento del antiguo agente autónomo resulta esencial restaurar y poner a punto dicho equipo. Esta tarea se divide en 3 subtareas:

- Subtarea 1.1: estudio de componentes hardware y análisis de los códigos realizados para el funcionamiento del antiguo agente. Finaliza con la comprensión de las funcionalidades que integra el robot.
- Subtarea 1.2: puesta en marcha del sistema. Se llevó a cabo la restauración de todas las partes que conforman el agente, para probar y finalizar la comprensión del funcionamiento del equipo.
- Subtarea 1.3: análisis de las posibles mejoras y viabilidad de las propuestas planteadas.

Los resultados esperados de esta fase son poner en funcionamiento el robot anterior y definir las líneas de trabajo del nuevo proyecto. En esta fase se abarcan los siguientes objetivos:

1. Análisis de los requisitos de la nueva plataforma basándose en las limitaciones del antiguo agente y las tecnologías emergentes.
2. Formación en previas y nuevas plataformas para el desarrollo de los módulos principales del agente y sus correspondientes mejoras.

Fase 2: Desarrollo de infraestructura

Uno de los aspectos fundamentales a la hora de llevar a cabo este proyecto es el diseño de la base sobre la que se asentaran los tres módulos que forman la plataforma. Para realizar este proceso se han llevado a cabo los siguientes pasos:

- Subtarea 2.1: estudio de las necesidades del sistema, centrándose en la selección de la versión de ROS a implementar y en las dependencias que requiere esa versión.
- Subtarea 2.2: instalación y configuración de un entorno Ubuntu 18.04 en las placas de desarrollo Raspberry Pi y en el ordenador que manejará el usuario, en este caso se instalará una máquina virtual. Se procede a configurar una versión de escritorio en los equipos para facilitar el desarrollo de los siguientes procesos.

- Subtarea 2.3: Instalación de ROS y paquetes necesarios para la implementación de los módulos de monitorización de parámetros y navegación. Una vez instalado se establece el sistema de comunicaciones a través de los archivos de configuración incluidos en Ubuntu 18.04.

El objetivo que se pretende alcanzar en esta fase es:

- Despliegue de la infraestructura requerida para el desarrollo de la nueva plataforma.

Fase 3: Integración del agente en ROS

Uno de los principales cometidos del proyecto es la integración de las antiguas funciones del agente en el entorno de ROS. Para cumplir este objetivo se ha trabajado sobre los siguientes puntos:

- Subtarea 3.1: Es imprescindible una formación en ROS, ya que los conocimientos adquiridos a lo largo del grado no abarcaban las funcionalidades que se pretenden implementar con esta herramienta.
- Subtarea 3.2: Estudio previo y diseño de la integración. En este punto se trabaja sobre la estructura de nodos que se va a desarrollar para poder realizar la integración de las funcionalidades anteriores.
- Subtarea 3.3: Integración del control manual. Este paso contempla la programación que se ha de desarrollar para poder tener un control manual sobre la navegación de la plataforma.
- Subtarea 3.4: Integración de la monitorización de parámetros y de la localización. En este apartado se realiza el código que permitirá la trasmisión de información entre el robot y el ordenador. Se enviarán los datos de posición y los parámetros ambientales que recibe la plataforma de los diferentes sensores.

Esta fase cubre el objetivo:

Migración del desarrollo del robot a ROS para facilitar su integración en gemelos digitales.

Fase 4: Módulo de visión artificial.

El desarrollo de un módulo de visión artificial comprende otro de los principales objetivos que se pretenden abarcar en este trabajo. Para poder llevar a cabo esta mejora se ha dividido su desarrollo en las siguientes tareas:

- Subtarea 4.1: Fase de estudio previo. Se realizará una investigación sobre las redes neuronales desarrolladas hoy en día, un análisis de los posibles modelos preentrenados que se ajustan a los requisitos de la aplicación y finalmente selección del modelo que mejor se adapte a las necesidades del sistema.
- Subtarea 4.2: Adaptación de la red neuronal para el análisis de vídeo. Se ha de diseñar el código de tal forma que no se limite a analizar una sola imagen, permitiendo así el análisis de vídeo.
- Subtarea 4.3: Rediseño del código para analizar en tiempo real la imagen recibida por una cámara. En esta tarea se pretende que la entrada a la red neuronal no sea un archivo de vídeo, sino que directamente se analice la imagen captada por una cámara.

El objetivo que se quiere alcanzar es:

- La integración de sistemas de visión artificial y redes neuronales que permitan la detección de víctimas y el reconocimiento de objetos.

Fase 5: Creación del sistema de navegación autónoma.
Para poder ofrecer una herramienta útil y válida para situaciones de emergencia, es esencial incorporar la navegación autónoma en la plataforma. Para su consecución se ha dividido este apartado en varias subtareas:

- Subtarea 5.1: Estudio del funcionamiento del paquete de navegación ofrecido por ROS.
- Subtarea 5.2: Diseño de nodos requeridos por el paquete de navegación.
- Subtarea 5.3: Generación de los ficheros de configuración y ajuste de parámetros para el funcionamiento del paquete.
- Subtarea 5.4: Lanzamiento de "Navigation Stack".

Se pretende alcanzar el objetivo:

- Incorporación de sistemas de navegación autónoma basados en el paquete de navegación "Navigation Stack"de ROS.

Fase 6: Pruebas y documentación
Tras la consecución de las tareas centradas en el desarrollo de la plataforma, se requiere una fase de pruebas y documentación.

- Subtarea 6.1: Desarrollo las pruebas referentes a los distintos módulos.
- Subtarea 6.2: Documentación de los resultados obtenidos y del diseño de la plataforma.

Finalmente, se alcanzan los objetivos:

1. Evaluación y pruebas de la nueva plataforma diseñada.
2. Documentación y estudio de los resultados obtenidos.

Para ilustrar el proceso de trabajo de este proyecto se ha diseñado un diagrama de GANT en el que se incluyen las tareas descritas anteriormente y el tiempo que ha tomado realizarlas en semanas, este diagrama se muestra en el **apéndice B**. El proyecto comenzó en diciembre de 2022 y ha finalizado en julio de 2023.

3.2 Herramientas.

Tras la presentación de las fases del proyecto, se procede a describir las herramientas necesarias para incorporar las funcionalidades de la plataforma.

3.2.1 Software de desarrollo robótico

ROS (Robot Operating System) [21] ha sido la herramienta esencial de este trabajo. Es un meta sistema operativo, es decir, proporciona una tercera capa adicional de abstracción que permite gestionar múltiples entornos operativos y recursos. Además, es de código abierto, esto fomenta la investigación y participación en la robótica, permitiendo reciclar y compartir códigos en su comunidad, y proporciona innumerables librerías accesibles para todos los usuarios.

Su programación se rige por un sistema de nodos. Un nodo es una estructura formada por un archivo ejecutable que cumple una función dentro de un algoritmo diseñado por el usuario. Estos nodos se comunican por tópicos, que representarían canales de comunicación por los que se intercambian la información en forma de mensajes. El encargado de hacer que estos nodos se encuentren unos a otros dentro de un sistema es el máster, permite que se compartan los mensajes y que se invoquen los servicios.

ROS utiliza un sistema basado en el patrón de diseño publicador/suscriptor. En este sistema, los nodos pueden publicar y/o suscribirse a un tópico, lo que les permite enviar o recibir mensajes a través de ese canal de comunicación, la figura 3.2 representa la arquitectura básica de ROS. Los nodos se conectan entre ellos directamente, el máster al igual que el servidor DNS solo proporciona la información de búsqueda. El protocolo comúnmente utilizado para la comunicación de los nodos es TCP/IP. Esta estructura facilita la autonomía e independencia de los módulos programados, dando como resultado un sistema escalable, flexible y modulable.

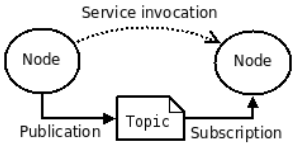


Figura 3.2: Estructura de comunicación de ROS.

Las ventajas que proporciona trabajar en un entorno de ROS recaen en la modularidad, los recursos a disposición del programador, la interoperabilidad y, sobre todo, la arquitectura distribuida. La posibilidad de dividir las funcionalidades del equipo facilita la reutilización de código, participando así en una comunidad de creadores que permite el intercambio de conocimientos. Por otro lado, ROS pone a disposición de los usuarios una amplia variedad de recursos, lo que proporciona una gran flexibilidad en la integración de componentes de hardware. Esto se ve potenciado por la adaptabilidad que ofrece ROS, permitiendo la integración de robots, sensores y actuadores de diferentes tipos. Finalmente, la gran distinción de este entorno es la arquitectura distribuida, ROS permite la comunicación y el intercambio de información entre nodos distribuidos de diferentes dispositivos, esto da la posibilidad de desarrollar sistemas robóticos complejos y proporciona escalabilidad al sistema. Y es importante destacar, que mediante ROS se puede introducir el sistema de navegación autónoma, gracias al paquete Navigation Stack.

Otro de los aspectos que ha favorecido la selección de esta herramienta es la facilidad que ofrece en la integración en entornos de simulación como Unity o Gazebo y en gemelos digitales, facilitando su futura integración en los proyectos llevados a cabo en la cátedra SMARTE2 [23] de la Universidad Rey Juan Carlos.

3.2.2 Red Neuronal

Haciendo referencia a los objetivos de la sección 2, el módulo de visión es un apartado fundamental del proyecto. En el estudio previo realizado para elegir el tipo de red neuronal que ofreciera un mejor rendimiento, se decidió utilizar la red neuronal YOLO.

Una red neuronal YOLO es un modelo basado en redes neuronales CNN, consta de varias capas convolucionales conectadas entre sí. La primera capa es la responsable de tomar la imagen, posteriormente se realiza el análisis extrayendo las principales características del input, tales como bordes, formas y texturas. Una vez se ha completado el análisis se procede a la clasificación, en este apartado se caracterizan los objetos de la imagen y se realiza el encuadre de dichos objetos.

Este tipo de red se diseñó como solución a la problemática del análisis de imagen en tiempo real. La arquitectura de YOLO hace que la imagen sea procesada solo una vez, lo que le aporta agilidad y velocidad, sin devaluar la precisión de la detección. Esto se debe a la completa conexión entre el gran número de capas de la red.

Otra ventaja que ofrece este algoritmo es la detección de múltiples objetos, no depende de regiones de interés, lo que le proporciona velocidad y precisión, ya que la detección se lleva a cabo simultáneamente. Por otro lado, YOLO cuenta con contexto espacial, una característica fundamental que la distingue de otras redes neuronales. YOLO es capaz de identificar la relación espacial entre los objetos, es decir, no los percibe como entidades independientes, mejorando de nuevo la precisión y proporcionando coherencia a sus resultados.

YOLO cuenta con una capacidad de generalización superior a la de otros algoritmos, generaliza objetos en diferentes tamaños, formas, orientación e incluso condiciones lumínicas. Esta cualidad es fundamental cuando se trata de implementar esta tecnología en diferentes entornos.

Teniendo en cuenta las características presentadas, YOLO se ha considerado la mejor opción para llevar a cabo el módulo de visión artificial.

Se ha decidido utilizar YOLOv3 para este proyecto, ya que dentro de la familia YOLO mejoró la precisión y la capacidad de detectar objetos frente a versiones anteriores. La arquitectura incorpora bloques residuales que ayudan a mitigar la degradación de rendimiento que se presentaba anteriormente a medida que aumentaba la profundidad de la red.

3.2.3 Microcontrolador

Una de las herramientas hardware utilizadas en el proyecto es Arduino [26]. Se decide utilizar dos placas Arduino para aumentar la modularidad de la plataforma, una de ellas se utilizará para el bloque de navegación y la segunda para el bloque de monitorización de parámetros. Las prestaciones que se exigían para la selección de la tarjeta de desarrollo eran las siguientes:

- Un entorno de desarrollo integrado compatible con las librerías de los sensores utilizados.
- Comunicación I2C.
- Canales PWM.
- Tecnología de bajo coste.

Después de revisar y evaluar las características requeridas, se ha determinado que solo se considerarán los modelos desarrollados por Arduino debido a la necesidad de utilizar las librerías específicas de los sensores, las cuales están diseñadas para funcionar con el entorno de desarrollo integrado (IDE) de Arduino.

El uso de un modelo que no esté desarrollado por Arduino implicaría enfrentar dificultades en la compatibilidad de las librerías y podría resultar en problemas de funcionamiento. Por lo tanto, con el objetivo de garantizar una integración fluida y un rendimiento óptimo, se ha decidido descartar cualquier opción que no cumpla con este requisito.

Arduino es una plataforma de desarrollo software y hardware de código abierto, consiste en una placa programable y un entorno de desarrollo integrado (IDE). Se basa en programación C/C++ y pone a disposición del usuario múltiples recursos y bibliotecas que simplifican la interacción con los componentes electrónicos. Por ello se ha realizado un estudio comparativo de tres modelos Arduino que podrían integrarse en la plataforma:

	Arduino Uno R3	Arduino Uno Wifi Rev2	Arduino Nano 33 IoT
Procesador	ATmega328	ATmega4809	SAMD21 Cortex-M0
Frecuencia de reloj	16 MHz	16 MHz	48 MHz
Memoria RAM	6 Kb	6 Kb	32 Kb
WiFi	No	Sí	Sí
Bluetooth	No	Sí	Sí
Pines GPIO	14	14	14
Canales PWM	6	5	11
Serial	1	1	1
I2C	Sí	Sí	Sí
Precio	20 €	47 €	25 €
IDE	Sí	Sí	Sí

Tabla 3.1: Tabla comparativa para la selección del microcontrolador.

En vista de crear una plataforma accesible y de bajo coste, la opción más acorde es Arduino R3. Aunque los otros modelos son superiores en cuanto a prestaciones, el precio de Arduino Wifi Rev2 es muy superior al del primer modelo. Por otro lado, se descartó el uso de Arduino Nano 33 IoT debido a la disponibilidad del componente. Sin embargo, puede ser una opción para futuras líneas de trabajo en cuanto a la incorporación de un sistema IoT en la plataforma.

La placa de Arduino Uno R3 cumple con el cometido que se pretende alcanzar en este proyecto, opera de transmisor de información entre los distintos periféricos del sistema. Por ello, se ha decidido utilizar esta herramienta, ofrece accesibilidad, siendo una plataforma de bajo coste y ampliamente disponible. Además, ofrece un amplio soporte por su comunidad. Cuenta con un prototipado rápido debido a su disponibilidad de componentes compatibles. En la figura 3.3 se presenta el modelo escogido.

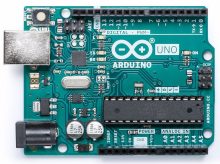


Figura 3.3: Arduino One R3

3.2.4 Procesamiento

Dentro de la arquitectura hardware se necesita un componente que actúe de procesador, se podría denominar el cerebro de la plataforma. Para ello se ha escogido una tarjeta de desarrollo que debía cumplir una serie de características:

1. Era necesario que soportase la instalación de ROS.
2. Puertos de conexión USB.
3. Capacidad de conexión Wifi y Ethernet.
4. Fluidez en su funcionamiento
5. Disponibilidad

Para seleccionar el componente que mejor se ajustaba a las especificaciones anteriores y que ofrecía mejores prestaciones, se ha realizado una comparativa entre los modelos de tres marcas distintas. A continuación, se muestra una tabla con los datos recogidos:

	Raspberry Pi 4	**Orange Pi PCH3**	**Banana Pi M64**
Procesador	Quad-Core ARMCortex-A72	Quad-core Cortex-A7	Quad-Core ARMCortex-A53
Frecuencia de reloj	1,5 GHz	1,2 GHz	1,2 GHz
Memoria RAM	4GB	1 GB	2GB
WiFi	Integrado	Integrado	Integrado
Bluetooth	Bluetooth 5.0	No integrado	Bluetooth 4.0
Pines GPIO	40	40	40
USB	4	3	2
Ethernet	Integrado	Integrado	Integrado
UART	2	1	1
Comunicación I2C	2	1	2
Precio	90 €	60 €	65 €

Tabla 3.2: Tabla comparativa para la selección del componente de procesamiento

Tras una revisión de las principales características que ofrece cada modelo, se ha decidido utilizar una Raspberry Pi4 como la que aparece en la figura 3.4, ya que permite la instalación de ROS de forma sencilla y la velocidad del procesador es superior a la de los demás modelos. Además, estaba disponible para este proyecto y la arquitectura física de la plataforma permitía su fácil incorporación.

Figura 3.4: Raspberry Pi 4 [2]

3.2.5 Sistema de posicionamiento

La localización del agente autónomo se hace mediante un sistema de posicionamiento en tiempo real, en este caso se ha decidido utilizar la tecnología de balizas de que ofrece Pozyx [27]. Este sistema se basa en señales de radiofrecuencia emitidas desde las balizas fijas, dispuestas en las paredes de la estancia y recibidas por la baliza del dispositivo (etiqueta), colocada en el objeto a posicionar.

Para el caso que se presenta se utilizará la técnica de trilateración (figura 3.5, que consiste en determinar la posición del objeto a partir de la distancia a varios puntos de referencia conocidos. Las medidas se obtienen basándose en el tiempo de vuelo de una señal, desde la baliza de posición conocida hasta la baliza que porta la plataforma.

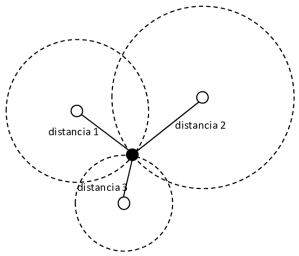


Figura 3.5: Técnica de trilateración [3]

Se realizó un estudio previo para decidir cuál era el sistema de posicionamiento que mejor se adaptaba a las necesidades del proyecto, se estudiaron las características de 3 sistemas. En primer lugar, el uso del robot se centra en situaciones de emergencia en espacios interiores, por lo que un sistema de localización por GPS quedó claramente descartado.

En segundo lugar, se planteó la opción de utilizar sensores de inercia, este sistema, aunque es viable para este proyecto, evidencia algunas desventajas. Como premisa, un sensor de inercia mide la aceleración y velocidad angular del dispositivo al que está conectado, para ello cuenta con un acelerómetro, un giroscopio y, a veces, un magnetómetro.

El acelerómetro mide la aceleración en los 3 ejes, la aceleración angular se obtiene utilizando el giroscopio, y con el magnetómetro es posible calcular la orientación del objeto respecto al campo magnético de la Tierra, obteniendo así una referencia de la dirección absoluta. Con todo ello, se podría contar con la posición, la velocidad y la orientación, sin embargo, este sistema presenta las siguientes desventajas:

- Se requiere una calibración muy precisa. Esto puede ser un proceso complejo en el que tengan que intervenir equipos de alta capacidad y muy especializados.
- Para una aplicación de alta precisión se necesitan acelerómetros de alta calidad cuyo precio no respetaba la idea de crear un sistema de bajo coste.
- Estos sensores suelen presentar errores acumulativos debido a la deriva causada por imprecisiones de velocidad y aceleración. Con el tiempo estos errores afectan negativamente a la exactitud del sistema.
- En entornos dinámicos como edificios de varias plantas, los sensores de inercia obtienen unas medidas inexactas debido a los cambios de aceleración y velocidad.

Finalmente, se determinó que la tecnología que mejor se adaptaba al caso de operación sería el uso de balizas, las principales ventajas que aporta este sistema son:

- No contemplan error acumulativo en las medidas, ofreciendo mayor precisión a lo largo de un tiempo prolongado.
- La precisión puede llegar a los 10 cm de error como máximo, mientras que los sensores de inercia pueden llegar a cometer hasta varios metros de error.
- El alcance que ofrece el sistema de balizas supera con creces el alcance de los sensores inerciales, llegando a los 70 m.
- Se podrían llegar a localizar varios objetos con la tecnología de balizas, esto abre un gran abanico de posibilidades para futuras aplicaciones.

3.2.6 Comunicaciones

Tras describir las principales herramientas que se han utilizado en este proyecto, se procede a describir los diferentes protocolos de comunicación que han permitido el intercambio de información entre los componentes de la plataforma.

Para la comunicación entre nodos que se desarrolla en ROS se emplea el protocolo TCP/IP. Consta de dos protocolos principales, TCP (Protocolo de Control de Trasmisión) y el protocolo IP (Protocolo de Internet). Las capas que definen la estructura de TCP/IP se muestran en la figura 3.6:

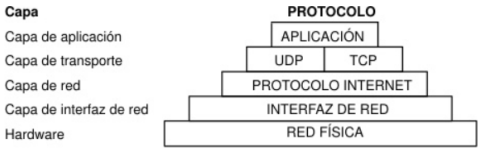


Figura 3.6: Conjunto de protocolos TCP/IP [4]

En la capa de aplicación, los programas envían mensajes a TCP (Trasmission Control Protocol). Este protocolo recibe los datos enviados y lo divide en paquetes, se añade una dirección de destino y se remiten a la capa de red. La capa de red transforma el paquete en un datagrama de IP, se añade la cabecera y la cola del datagrama, y se decide cuál es el receptor de dichos datos (directamente a un destino o a una pasarela). Seguidamente, pasa el datagrama a la capa de interfaz de red. La capa de interfaz de red recibe los datagramas y los trasmite como tramas a través del hardware de red específico, en este proyecto, una red Ethernet. En la figura 3.7 se muestra visualmente como se trasmite la información por las diferentes capas.

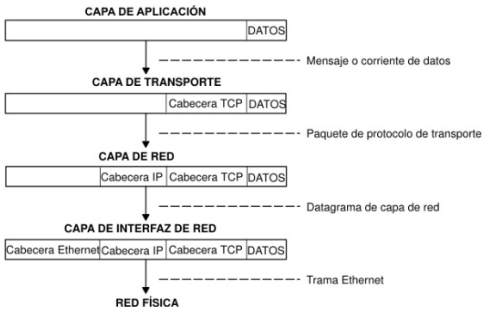


Figura 3.7: Modelo de trasmisión de la trama de datos a través de TCP/IP [4]

Otro de los protocolos utilizados en esta plataforma es el protocolo I2C (Inter-Integrated-Circuit), permite que varios circuitos integrados digitales se comuniquen con los controladores. En este caso, los periféricos que miden las variables ambientales y los sensores ultrasonidos utilizan este protocolo para hacer llegar los datos a la placa Arduino One. Se utiliza en comunicaciones de corta distancia, se trata de una comunicación maestro-esclavo y requiere dos señales para intercambiar información, como se puede ver en la figura 3.8. A diferencia de SPI, al utilizar I2C se podría integrar más de un controlador en el sistema de comunicación.

Cada bus I2C está formado por dos señales: SCL y SDA. SCL (Serial Clock) es la señal de reloj producida por el controlador, y SDA (Serial Data), la señal de datos. Los mensajes se dividen en dos tipos de tramas: una trama de datos con los mensajes de 8 bits que se intercambian el maestro y el esclavo, y una trama de direcciones, en la que el maestro indica a qué periférico se envía el mensaje. Los datos se disponen en la línea SDA, el tiempo que trascurre entre cada flanco de reloj y la lectura o escritura de los datos está definido por los dispositivos del bus.

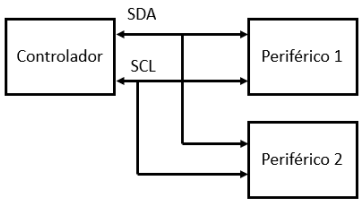


Figura 3.8: Esquema de la comunicación entre el controlador y los periféricos por protocolo I2C.

3.3 Metodología

Para lograr los objetivos descritos en el capítulo 2, se ha aplicado una metodología cíclica en cada prueba. Se ha seguido la siguiente estructura en cada fase del proyecto:

Figura 3.9: Diagrama de la metodología utilizada a lo largo del proyecto.

La estructura seguida en los procesos de cada bloque funcional se ha desarrollado en varias etapas. En primer lugar, se llevó a cabo una fase de formación o estudio previo, donde se analizaron detalladamente los requisitos específicos de cada parte del proyecto y adquisición de los conocimientos necesarios para llevar a cabo el módulo. Esto permitió comprender en profundidad las necesidades de cada módulo.

En segundo lugar, se procedió a evaluar los requisitos identificados y se estudió el diseño e integración de cada componente en la plataforma. Se realizaron análisis de viabilidad y se tomaron decisiones fundamentadas para garantizar una implementación exitosa.

En tercer lugar, se realizó la implementación de cada proceso, siguiendo las pautas establecidas durante la etapa de diseño. Se llevaron a cabo las tareas necesarias para poner en marcha cada módulo, incluyendo la configuración de parámetros, la programación de algoritmos y la interacción con los dispositivos correspondientes.

Posteriormente, se procedió a realizar simulaciones para verificar el correcto funcionamiento de cada módulo. Estas simulaciones permitieron evaluar el desempeño del sistema en diferentes escenarios y detectar posibles problemas o áreas de mejora.

Finalmente, se llevaron a cabo pruebas para evaluar el rendimiento de cada módulo y se documentaron los resultados obtenidos. Se registraron datos y se compararon con los objetivos establecidos.

Esta estructura en etapas ha permitido abordar de manera sistemática cada bloque funcional, asegurando un desarrollo ordenado y una evaluación precisa de los resultados obtenidos en cada fase del proyecto.

Capítulo 4

Diseño e implementación.

4.1 Arquitectura

Tras la presentación de las metodologías y herramientas utilizadas para llevar a cabo este proyecto, se procede a describir la forma en la que se han desarrollado las tareas descritas en la sección 3.1.

El proyecto se divide en 2 grandes bloques: el bloque software y el bloque hardware. Dado que se trata de un proyecto en el que se combinan diversas disciplinas de la ingeniería (electrónica, programación, inteligencia artificial, mecánica...), podemos distinguir la parte enfocada a la elaboración de código y al diseño computacional, y, por otro lado, los componentes electrónicos, los periféricos y la parte física del sistema robot.

4.1.1 Arquitectura Hardware

En este apartado se desarrolla lo referente a la estructura y organización de los componentes físicos que constituyen el sistema robótico. A continuación, se presenta el esquema principal del bloque hardware:

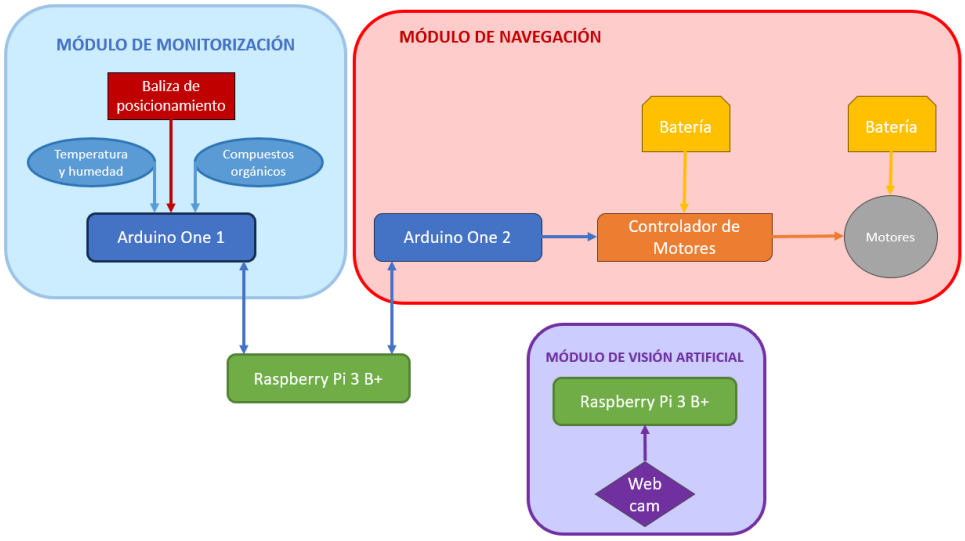


Figura 4.1: Arquitectura global Hardware

En la figura 4.1, se observan una representación de los componentes que conforman cada módulo de la plataforma.

Diseño hardware del módulo de monitorización de parámetros.

Tras la selección de la tarjeta de desarrollo, se procede a describir qué periféricos se han conectado a ella para obtener los datos de monitorización de parámetros ambientales. Se han escogido los siguientes sensores:

- Sensor HTU21D-F (figura 4.2a): este sensor toma el valor de temperatura y humedad relativa del ambiente, el envío de datos se hace mediante el protocolo I2C, que proporciona velocidad y precisión en la trasmisión de los valores.

  1. La temperatura es la magnitud física que indica la energía interna de un cuerpo, de un objeto o del medio ambiente en general. La temperatura ideal en reposo para el ser humano ronda los 20º C, por encima de 38º C puede tener efectos negativos en la salud, desde deshidratación y golpes de calor hasta arritmias o muerte. Además, la temperatura es uno de los datos más importantes en el desarrollo de un incendio.
  2. La humedad relativa mide la relación entre la cantidad de vapor de agua real que contiene el aire y la cantidad que necesitaría para saturarse a idéntica temperatura. Uno de los mecanismos de regulación térmica es la evaporación del sudor, por lo que cuanto más humedad haya en el ambiente menor será la capacidad de refrigeración de las personas implicadas. Por ello, es importante medir la humedad relativa [28]. El nivel ideal de humedad para el confort del ser humano se encuentra en el rango de 40 % a 60 %.

- Para la medida de compuestos ambientales se ha escogido el sensor Adafruit SGP30 TVOC/eCO2 Gas Sensor (figura 4.2b), este sensor permite cuantificar los siguientes parámetros:

  1. Los compuestos orgánicos volátiles totales (TVOC), se trata de hidrocarburos presentes en el ambiente en estado gaseoso. Se puede considerar como COV aquel compuesto que a 20º C tenga una presión de vapor de 0.01 kPa o más, o una volatilidad equivalente en las condiciones particulares de su uso [29]. Estos compuestos pueden clasificarse, según su peligrosidad, en tres principales grupos:
     - Compuestos extremadamente peligrosos, como el benceno o el cloruro de vinilo.
     - Compuestos de clase A, hacen referencia a los compuestos que pueden causar daños significativos al medio ambiente.
     - Compuestos de clase B, son los que tienen menor impacto en el medio ambiente, en este grupo se ubicarían la acetona y el etanol.
  2. El nivel de dióxido de carbono equivalente (eCO2). Es primordial conocer el valor de este parámetro, su presencia puede llegar a producir asfixia por emplazamiento de oxígeno si su concentración hace que el nivel de oxígeno este por debajo del 20 %. Además, puede producir, mareos, somnolencia y problemas respiratorios.
  3. La concentración de hidrógeno (H2) presente es otro de los parámetros a tener en cuenta, su importancia reside en el amplio rango de inflamabilidad (entre el 4 % y el 74 % de concentración en el aire) y la poca energía que requiere para iniciar una combustión [30]. Otra de las razones por las que se requiere evaluar el nivel de hidrógeno reside en los efectos que puede tener su inhalación en la salud. Si existe una gran concentración de este compuesto se pueden experimentar mareos, dolores de cabeza, pitidos en los oídos, somnolencia, vómitos y en casos muy extremos podría causar inconsciencia e incluso la muerte.
  4. Otro de los compuestos presentes que se ha decidido monitorizar es el etanol. Concentraciones excesivas de este compuesto pueden producir irritación del tracto respiratorio superior e irritación ocular.

(a) Sensor de temperatura y humedad Adafruit HTU21D-F. [31]

(b) Sensor de COV Adafruit SGP30 [5]

Figura 4.2: Sensores utilizados para la medida de parámetros ambientales. [5]

En este módulo también se desarrolla el posicionamiento del agente mediante el sistema de balizas Pozyx. La plataforma cuenta con un juego de 5 balizas, 4 de ellas deberán colocarse en las paredes del espacio donde se encuentra el sujeto y la quinta se denominará etiqueta, es la que porta el objeto a localizar, en este caso el robot. Se alimentan a través del puerto USB a 3,3 V, está compuesto por un microcontrolador STM34F4 y un chip DECAWAVE DW 1000, este chip emite la onda de alta frecuencia que sirve para localizar la etiqueta. Su compatibilidad con Arduino ha sido clave para la incorporación de esta tecnología en el sistema.

Por último, en este módulo se han integrado cinco sensores ultrasonidos. Estos sensores forman parte del bloque de monitorización de parámetros, aunque tendrán gran implicación en el sistema de navegación autónoma que se pretende incorporar. Este sensor envía los datos mediante el protocolo I2C, además es compatible con las placas Arduino, incluye una librería que facilita su incorporación en la plataforma. La tensión de alimentación entra en un rango de 3,3 V a 5,5 V, y su corriente de funcionamiento es de 20 mA. La integración de los nuevos sensores ultrasonidos permite localizar cualquier obstáculo que rodee al robot gracias a su disposición. En la figura 4.3, se presenta el sensor descrito.

Figura 4.3: Sensor ultrasonidos Gravity SEN0304 DF Robots.

Diseño hardware del módulo de navegación.
El módulo de navegación debe permitir el movimiento del agente teniendo en cuenta sus dimensiones y los periféricos instalados. Para ello se ha precisado de los siguientes componentes:

- El componente principal de un módulo de navegación es el motor, en este caso se utilizan dos motores GM25-370 DC. Es micro motorreductor de imán permanente, su alimentación es a 12 V.
- El motor está alimentado por tres pilas dispuestas en serie recargables de Ión-Litio de 3,7 V cada una, y 12.800 mAh. Estas pilas a su vez proporcionan tensión a una de las placas Arduino One.
- Para coordinar los motores se dispone de un controlador de motores SKU DRI0002. Utiliza un chip de alta potencia que proporciona el accionamiento directo de los motores de forma bidireccional. Su arquitectura le permite alcanzar altos niveles de tensión, ya que cuenta con un disipador térmico y con diodos de alta velocidad para su protección. El voltaje de entrada permitido es de 6 V a 12 V, funciona a 2 A y dispone de un modulador de ancho de pulso (PWM) que permite llevar un control

de la energía que se trasmite. De esta forma los motores van sincronizados independientemente de la alimentación.

- El controlador se alimenta con un "power bank", una batería de litio de 5 V con 10.800 mAh de capacidad. La segunda placa Arduino One y la Raspberry Pi también obtienen la alimentación de este dispositivo.

Esta arquitectura apenas ha sufrido cambios en comparación con el agente anterior. Se han realizado algunas actualizaciones para asegurar su funcionamiento, incluyendo la sustitución de un conector de motor que presentaba problemas. Además, se ha verificado la configuración y el correcto funcionamiento de todas las conexiones.

Diseño hardware del módulo de visión artificial.
En vista de aumentar la información que le llega al equipo de bomberos, se ha implementado un módulo de visión artificial que complemente la percepción del entorno. Para ello se han utilizado dos componentes: una placa Raspberry Pi y una cámara webcam.

La Raspberry Pi es la encargada de ejecutar el código que analiza la imagen recibida por la cámara, y posteriormente, trasmitir el análisis a través de SSH a la máquina virtual. La plataforma pretende así que a medida que el agente pase por la sala vaya enviando la imagen con la detección de objetos al ordenador que porta el usuario final, así podrán preparar al equipo de forma más específica para la operación.

El modelo de cámara utilizado es LifeCam HD-3000 - Webcam, a continuación se muestra una tabla con las especificaciones técnicas:

Especificaciones Técnicas	
Relación de aspecto	16:9
Sensor CMOS	1MP
Resolución máxima	1280x720 píxeles
Velocidad máxima	30 fps
Zoom digital	4x
Interfaz	USB 2.0
Alimentación	1,1 V

Figura 4.4: Tabla características técnicas de la cámara.

Figura 4.5: Cámara LifeCam HD-3000 Webcam [6]

Esta cámara (figura 4.6) cuenta con un soporte moldeable que permite un posicionamiento sencillo en el robot, está dispuesta de tal forma que detecte los objetos, personas o animales que se posicionen frente a él en su recorrido. Además, la incorporación del conector USB permite integrarla fácilmente en la Raspberry Pi.

4.1.2 Arquitectura Software

Una vez realizada la descripción de los componentes hardware que incluye este equipo, se procede a describir la parte software. En esta sección se pretende reflejar la estructura y el diseño de los diferentes módulos que conforman la arquitectura software del sistema.

A continuación, se muestra la arquitectura de las aplicaciones software utilizadas en el programa:

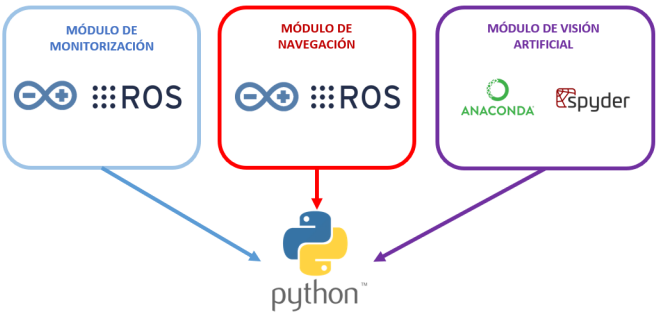


Figura 4.6: Arquitectura Software

En los bloques de navegación y monitorización de parámetros se ha utilizado ROS como herramienta principal, el lenguaje de programación en este entorno ha sido Python. Sin embargo, los programas desarrollados en el IDE de Arduino se han escrito siguiendo el lenguaje C/C++ propio de Arduino. Por otro lado, para la aplicación de visión artificial se ha utilizado el entorno de Anaconda, en concreto, el software Spyder, cuyo lenguaje de programación es Python.

El módulo de navegación tiene como objetivo direccionar y controlar el movimiento del robot en el espacio, el módulo de monitorización pretende poder almacenar los datos obtenidos y mostrarlos al usuario, y por último, el módulo de visión desarrolla la detección de objetos y personas de un vídeo mediante una red neuronal.

Diseño software del módulo de monitorización y posicionamiento.
En lo relativo al diseño software de este apartado, la estructura consiste en dos nodos, un publicador y un suscriptor. El tópico por el que se trasmiten los mensajes es "valores" y el tipo de mensaje creado es Float32MultiArray permitiendo así mandar una estructura con todos los datos recibidos por el Arduino. A continuación, en la figura 4.7, se muestra un diagrama de los nodos que rigen este módulo:

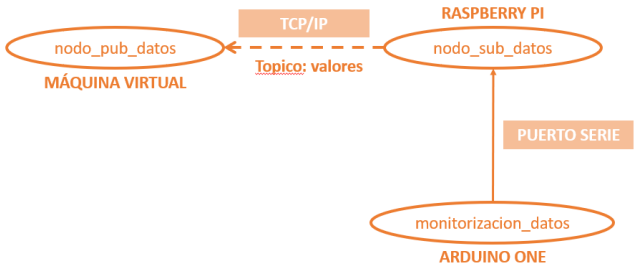


Figura 4.7: Estructura de nodos para el módulo de monitorización de parámetros y posicionamiento

El proceso consiste en 3 etapas principales:

1. Adquisición de datos por los sensores: en este primer paso los sensores envían los valores medidos a la placa Arduino, a través de los pines SDA y SCL, que son los que cuentan con el protocolo de comunicación I2C. Arduino ejecuta un código en el que se crea una trama de datos que posteriormente se enviará a la Raspberry Pi por el puerto serie. La trama de datos es la siguiente:

Figura 4.8: Estructura de la trama de datos general

La figura 4.8 representa como se crea la trama de datos que luego será analizada por la Raspberry Pi. La cadena comienza con un "@" que marca el inicio de trama, en segundo lugar, el símbolo "#" separa los datos de los diferentes sensores, le acompaña un número que indica la posición de ese sensor en la trama de datos y seguidamente el valor que ha adquirido dicho sensor.

El orden en el que se envían los parámetros es el presentado en la figura 4.9:

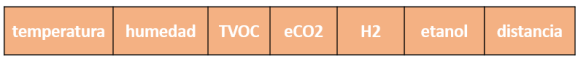


Figura 4.9: Trama de datos general

2. Recepción de la trama de datos en Raspberry Pi. La Raspberry Pi ejecuta el nodo "nodo_pub_datos.py", es decir, el nodo publicador que se encargará de mandar la información al ordenador mediante el protocolo TCP/IP por el tópico "valores".

3. Recepción de datos por máquina virtual. El nodo suscriptor al tópico "valores" se ejecuta en la máquina virtual que controla el usuario. Este nodo es el encargado de recopilar la información y mostrarla al usuario por pantalla en tiempo real. Además, desde este código se vuelcan los datos a un archivo CSV, en el que no solo aparecen los valores sino el momento exacto en el que se han recibido.

Diseño software del módulo de navegación

En este bloque se ha desarrollado el software necesario para la integración en ROS de las funciones de navegación anteriores y la creación de un sistema de navegación autónoma.

Para el primer modo de funcionamiento, se describe un nuevo diseño basado en ROS, conformado por dos nodos, un publicador y un suscriptor. El sistema de nodos resultante se presenta en la ilustración 4.10:

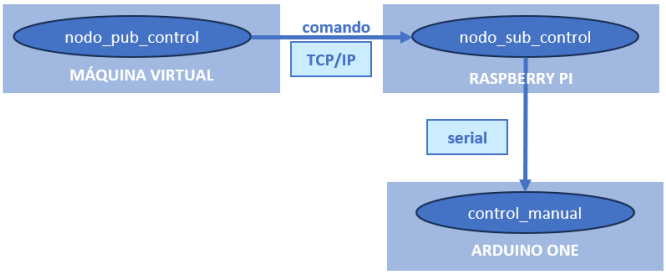


Figura 4.10: Arquitectura software del control manual.

El nodo publicador tiene como objetivo recibir por teclado el comando de dirección y trasmitir este dato al nodo suscriptor de la Raspberry Pi. Al tratarse de ROS el protocolo de comunicación utilizado es TCP/IP, esto exige una previa configuración de red robusta, en la que se establezca la dirección IP del dispositivo que actúa de máster y la dirección IP del equipo que actúa de esclavo (de suscriptor). El mensaje se envía por el tópico "comando" y el formato es String, esto facilita la trasmisión por puerto serie que se debe hacer posteriormente desde la Raspberry Pi al Arduino. El nodo suscriptor, nodo_sub_control.py, se aloja en la Raspberry Pi. Este nodo debe trasmitir el comando a través del puerto serie a Arduino.

Una vez recibido el comando en formato String por la tarjeta de desarrollo Arduino One, se ejecuta el programa que debe interpretar el comando y enviar al controlador de motores las directrices necesarias para desarrollar el movimiento correspondiente.

Por otro lado, en esta plataforma se ha incorporado un nuevo modo de funcionamiento, la navegación autónoma. Se ha implementado un paquete de navegación incluido en ROS, "Navigation Stack". Este paquete de navegación sigue la siguiente arquitectura de nodos (figura 4.11):

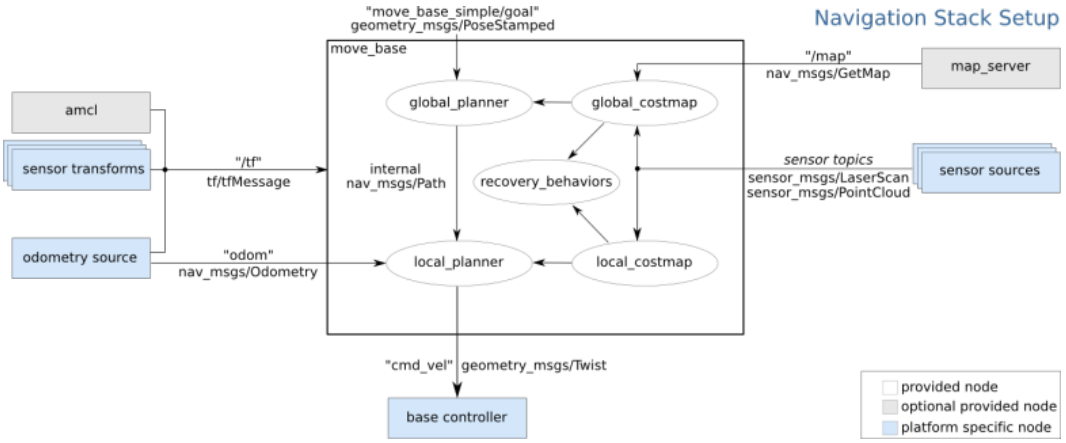


Figura 4.11: Arquitectura del paquete de navegación de ROS. [7]

"Navigation Stack" es un conjunto de paquetes y algoritmos que permiten a un robot moverse de forma autónoma en su entorno. Proporciona las herramientas y funcionalidades para la planificación de rutas, localización y control de movimiento.

A partir de sensores o láseres, y odometría, estima la posición del robot y la orientación en el entorno. Una vez que el robot tiene información sobre su posición, utiliza algoritmos de planificación de rutas para seleccionar la mejor trayectoria desde su ubicación hasta la posición objetivo. Tras la planificación de la ruta, la plataforma utiliza algoritmos de control para seguir esa ruta y realizar los movimientos necesarios para llegar al destino. Estos algoritmos ajustan la velocidad y la dirección para mantener al robot en la trayectoria y evitar obstáculos.

Los nodos de odometría, sensores ultrasonidos y fuentes de sensores, debe crearlos el usuario y proporcionar los debidos datos que se enviarán por los tópicos especificados. Además, se deben elaborar los archivos de configuración de parámetros que determinan el funcionamiento del sistema. Los tópicos que participan en el paquete de navegación son:

1. odom: este tópico permite enviar los mensajes relacionados con el movimiento del robot (velocidad linear, velocidad angular, distancia...).

2. tf: por este tópico se envía esta transformación de distancias entre los marcos de referencia del sistema.

3. scan: a través este tópico se envían los valores de los ultrasonidos, así el robot puede tener el conocimiento de lo que le rodea y realizar una ruta acorde evitando posibles obstáculos.

La arquitectura final que conforma la navegación autónoma es la siguiente (ilustración 4.12):

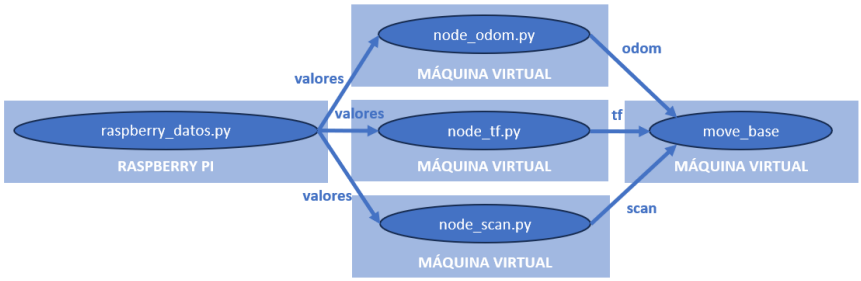


Figura 4.12: Arquitectura software del modo de navegación autónoma.

Diseño software del módulo de visión artificial.

Por último, se procede a describir el diseño software con el que cuenta el módulo de visión artificial. Para llevar a cabo este módulo, se ha utilizado el entorno de desarrollo "Anaconda", que incluye programas como "Spyder" para el desarrollo de aplicaciones de visión artificial. El código de este módulo se ha trabajado en esta aplicación con las librerías "NumPy" y "OpenCv". Estas dos librerías cuentan con las herramientas necesarias para llevar a cabo la función deseada en este proyecto. En el **anexo A.5** se presentan las funciones utilizadas procedentes de estas dos librerías. Como se ha descrito anteriormente en el apartado 3.2.2, se ha utilizado un modelo preentrenado regido por una red neuronal YOLO. Este modelo se ha adaptado para el análisis de vídeo en tiempo real por una webcam. El modelo está compuesto por los siguientes archivos:

1. yolov3.cfg: este archivo lleva la configuración de la red neuronal.

2. yolov3.txt: en este archivo se guardan las clases de objetos que detecta la red neuronal.

3. yolov3.weigths: recoge los pesos de la red neuronal, son los parámetros que te permiten ajustar la fuerza de la conexión entre las neuronas de la red. Por lo tanto, cada conexión dentro de la red neuronal tiene asignado un peso que determina la importancia de esa neurona dentro de la red. Además, se utilizan para calcular la salida de cada neurona según las entradas definidas. En la fase de entrenamiento de la red los pesos se autoajustan para minimizar la diferencia entre la salida de la neurona y la salida esperada.

4. yolov3_opencv_webcam.py: este el código principal del diseño. Se cargan el resto de los archivos involucrados en la red neuronal y se adjudica el input, que en este caso es la imagen proporcionada por la webcam. Este código incluye la programación necesaria para analizar vídeo y dibujar los marcos alrededor del objeto detectado, y simultáneamente mostrar el resultado por pantalla.

Estos archivos están disponibles en el repositorio de GitHub presentado en el **apéndice A.6**.

4.2 Diseño e implementación de los módulos que conforman la plataforma.

Una vez introducidos los tres bloques principales de la plataforma, se procede a describir como se ha diseñado cada bloque atendiendo a los requisitos del proyecto, y su posterior implementación siguiendo las fases especificadas en el apartado 3.1.

4.2.1 Diseño e implementación del módulo de monitorización de parámetros y localización.

Como se ha explicado en apartados anteriores, este módulo debe proporcionar al usuario los datos que describen la calidad ambiental del espacio de emergencias. Tras describir los componentes, hardware y software en el apartado 4.1, se procede a explicar cómo se ha implementado este bloque. El bloque consta de la estructura presentada en la figura 4.13:

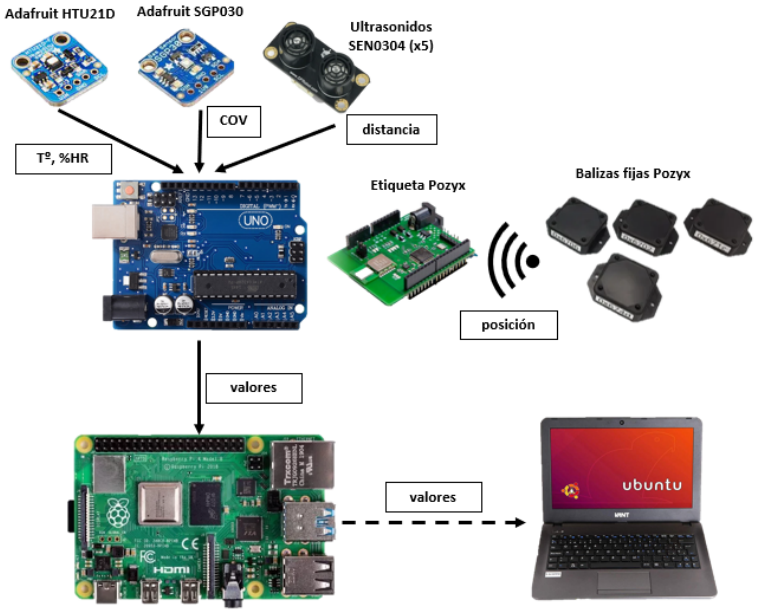


Figura 4.13: Arquitectura del módulo de monitorización de parámetro y localización.

En la figura anterior, se presentan los diferentes componentes que forman este módulo. Los sensores que se ven en la parte superior izquierda reciben los parámetros ambientales (temperatura, humedad relativa, compuestos orgánicos volátiles), junto a ellos se puede ver el sensor ultrasonidos. En este proyecto se han integrado 4 ultrasonidos nuevos a la plataforma. Para ello se ha rediseñado el código localización_us.ino donde solo se contemplaba la comunicación con ultrasonidos por protocolo I2C. Para la incorporación de estos nuevos sensores ha sido necesario un proceso de direccionamiento y la creación de nuevas funciones que recojan los datos de distancia y los agreguen a la cadena de datos enviada por puerto serie a la Raspberry Pi. El conexionado de los componentes es el siguiente:

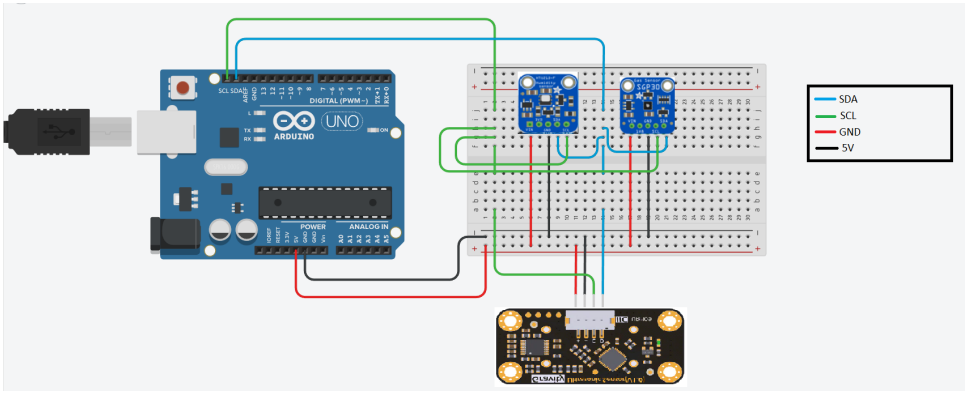


Figura 4.14: Conexionado de los sensores a la placa de Arduino.

En la figura 4.14 no se han incluido las resistencias para presentar una imagen clara de las cuatro señales principales que rigen este circuito. El diseño se ha realizado utilizando Tinkercad [32]. Tampoco se incluyen los cinco sensores ultrasonidos por la misma razón, la conexión es idéntica al ultrasonido expuesto en la imagen.

Al incorporar múltiples sensores de ultrasonidos al sistema, es necesario abordar la configuración y lectura de cada sensor de manera individualizada. Para ello, se han implementado funciones específicas que permiten gestionar eficientemente estos sensores.

- La función "configureSensor()" se encarga de establecer la configuración necesaria para cada sensor. Esto incluye configurar los sensores en modo pasivo y establecer el rango de medición adecuado para cada uno. De esta manera, se asegura que los sensores estén preparados para realizar mediciones precisas.

- La función "readDistanceSingular()" se utiliza para medir la distancia de un sensor de ultrasonidos específico identificado por su dirección. Esta función activa el sensor, realiza la medición y almacena el valor obtenido. Cada sensor individual puede ser accedido de forma independiente para obtener mediciones precisas y actualizadas.

- La función "readDistances()" se encarga de leer las distancias de todos los sensores de ultrasonidos presentes en el sistema. Esta función ofrece escalabilidad al proyecto, ya que permite agregar un número variable de sensores sin tener que modificar el código en cada caso. Simplemente, se deben agregar las direcciones de los sensores adicionales y la función se encargará de leer las distancias de forma simultánea.

Estas funciones aseguran un adecuado funcionamiento del sistema, proporcionando mediciones precisas y una gestión eficiente de los datos obtenidos. El código de estas funciones se encuentra en el archivo "localizacion_us.ino" accesible mediante el enlace al repositorio Github del **anexo A.6**.

Para el posicionamiento de la plataforma se utiliza el sistema de balizas Pozyx descrito en la sección 3.2, se requiere una disposición específica de las balizas. El objetivo es proporcionar un sistema de localización preciso y confiable para la plataforma. Para este proceso es necesario colocar las balizas siguiendo la estructura que se muestra a continuación en la figura 4.15:

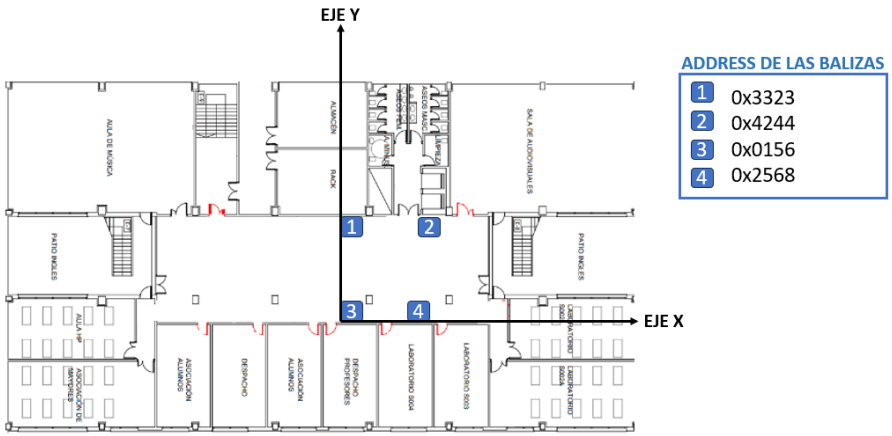


Figura 4.15: Posicionamiento de las balizas fijas en el hall de Laboratorios III (Universidad Rey Juan Carlos).

La disposición de las balizas consiste en colocar cuatro balizas fijas en las paredes opuestas del entorno, distribuyéndolas en pares. Se recomienda ubicar las balizas de cada par a diferentes alturas para obtener una referencia espacial clara y mejorar la precisión del sistema. La figura 4.15 muestra un ejemplo de la disposición en el hall de Laboratorios III de la Universidad Rey Juan Carlos.

Además de las balizas fijas, se utiliza una quinta conocida como la etiqueta. Esta baliza está acoplada al robot y se mueve con él, permitiendo conocer la posición del agente en tiempo real. La información de posición generada se transmite a través del puerto serie a la Raspberry Pi. El código "localización_us.ino" se encarga de configurar y recopilar los datos de posición de las balizas. Este código es fundamental para obtener información precisa sobre la ubicación del robot en el entorno.

En conjunto, el sistema de posicionamiento por balizas Pozyx proporciona una solución confiable y precisa para determinar la posición en tiempo real del robot en el entorno de trabajo. La correcta disposición de las balizas, con especial atención a las alturas y la configuración adecuada del código, son elementos esenciales para lograr una localización precisa y confiable del agente. Con esta configuración, el robot puede moverse de manera autónoma y realizar tareas específicas basadas en su posición en el entorno.

En la Raspberry Pi se ejecuta el código "nodo_pub_datos.py", en este código se instancia el nodo publicador que publica los datos recibidos en el tópico "valores". Para ello debe reconocer la cadena recibida en formato String y transformarla a formato Float32MultiArray. Este nodo sigue el siguiente diagrama de flujo (figura 4.16):

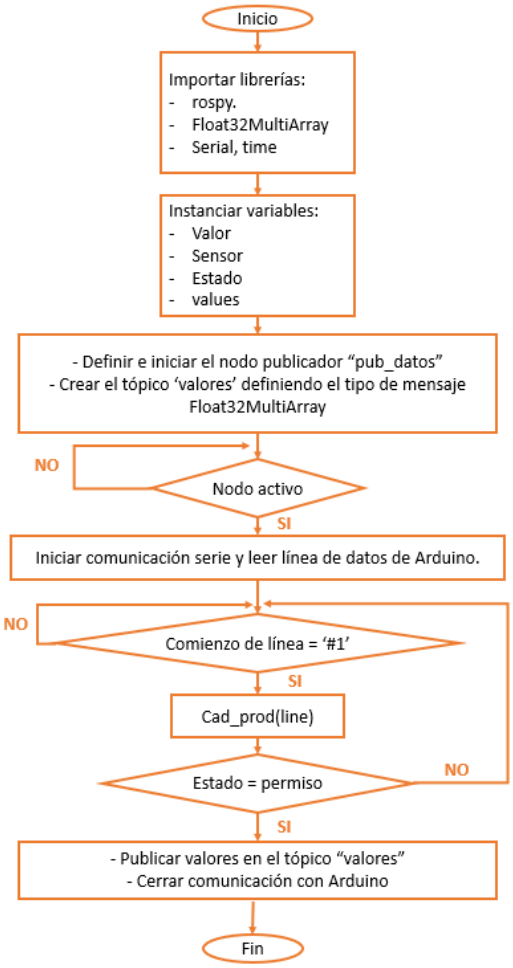


Figura 4.16: Diagrama de flujo del programa principal de nodo_pub_datos.py

El programa comienza con la importación de las librerías necesarias y la declaración de las variables que se utilizarán de forma global a lo largo del código. En segundo lugar, se define y se inicia el nodo publicador. Este nodo recibe por puerto serie los datos procesados por Arduino One y los publica en el tópico "valores" en un mensaje tipo Float32MultiArray.

Si el nodo está activo se inicia la comunicación por puerto serie con Arduino y leemos la línea de datos. Atendiendo a la estructura presentada en la figura 4.9 la cadena debe empezar por "#1" para poder procesar todos los datos de los sensores, si es así se ejecuta la función "cad_prod" teniendo como variable de entrada la cadena leída por puerto serie. Esta función divide los datos recibidos según el ID del sensor en 2 variables: "sensor" y "valor", con ello se crea una nueva cadena compuesta solo por los valores. Esta será la que se publique en el tópico "valores" siempre que la variable "estado" sea igual a "permiso" esto asegurará que la cadena se ha formado correctamente. Finalmente, se cierra la comunicación con Arduino.

El nodo suscriptor al tópico "valores" está alojado en la máquina virtual del ordenador. La información se envía mediante el protocolo TCP/IP atendiendo a las comunicaciones de ROS. El nodo suscriptor, "nodo_sub_datos.py". En la figura 4.17 se muestra el diagrama de flujo que describe el funcionamiento del código.

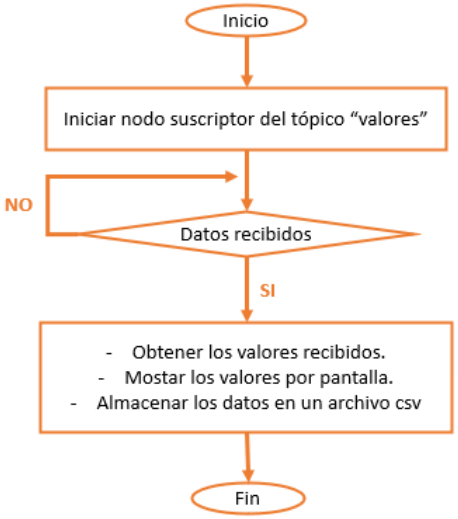


Figura 4.17: Diagrama de flujo del programa principal de nodo_sub_datos.py

Cuando el suscriptor recibe un mensaje por el tópico, se imprimen por pantalla los valores medidos y la variable correspondiente al valor, de esta forma el usuario obtiene los parámetros ambientales del entorno en tiempo real y los visualiza en la pantalla de la máquina virtual. Además, este código vuelca los datos a un archivo CSV que permite visualizarlos posteriormente en una base de datos. De esta manera se asegura un acceso a los datos en cualquier momento, ya que se incluye la hora en la que se han recibido esos datos.

4.2.2 Diseño e implementación del módulo de navegación.

Atendiendo a los objetivos descritos en la sección 2, se pretende incorporar dos modos de funcionamiento en ROS: un modo manual y un modo de navegación autónoma. Para estas dos funcionalidades se comparten los mismos elementos, a continuación, se muestra una representación del bloque de navegación:

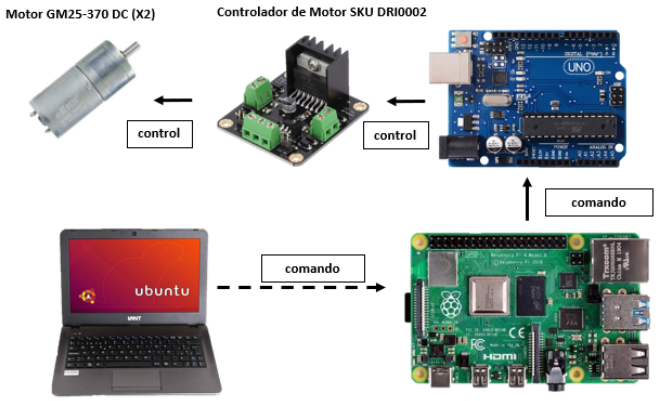


Figura 4.18: Arquitectura del módulo de navegación.

En la figura 4.18 se muestran los dispositivos que conforman este bloque: el ordenador con la máquina virtual Ubuntu 18.04, la Raspberry Pi 4, una de las placas Arduino One, el controlador de motores y dos motores GM25-370.

En el diseño del módulo de navegación se desarrolla la integración del agente en ROS y rediseño de la programación, y el lanzamiento del paquete de navegación (correspondientes a las fases 2 y 5 del apartado 3.1). Previamente, se ha realizado una formación en ROS para acometer los objetivos correspondientes a estas fases.

Como primer paso, se integró el control manual en una estructura de nodos en ROS. Se rediseñó la programación para establecer el código de recepción de comandos en la máquina virtual y el código de comunicación con el controlador de motores en la Raspberry Pi. El nodo publicador, "nodo_pub_control.py" sigue el siguiente diagrama de flujo:

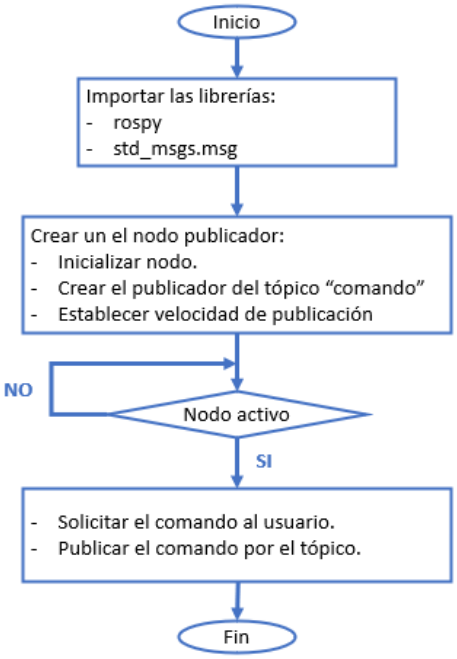


Figura 4.19: Diagrama de flujo del programa principal de nodo_pub_comando.py

Para el funcionamiento de este modo, el usuario debe introducir por el teclado la dirección a la que quiere dirigir al robot atendiendo a la siguiente tabla (figura 4.20):

Letra	Comando
W	Adelante
S	Atrás
D	Derecha
A	Izquierda
E	Aproximación Derecha
Q	Aproximación Izquierda
C	Aproximación trasera derecha
Z	Aproximación trasera izquierda
T	Turbo
P	Parada

Figura 4.20: Tabla de comandos del control manual.

El código nodo_pub_control.py recibe un comando por teclado y lo publica en el tópico "comando". El comando es recibido por el nodo suscriptor nodo_sub_control.py, que a su vez, enviará dicha variable a la placa Arduino por puerto serie. Finalmente, la tarjeta de desarrollo interpretará el comando y actuará sobre el controlador de motores para activar los motores según la dirección proporcionada. A continuación, se muestra el diagrama de flujo correspondiente al nodo suscriptor:

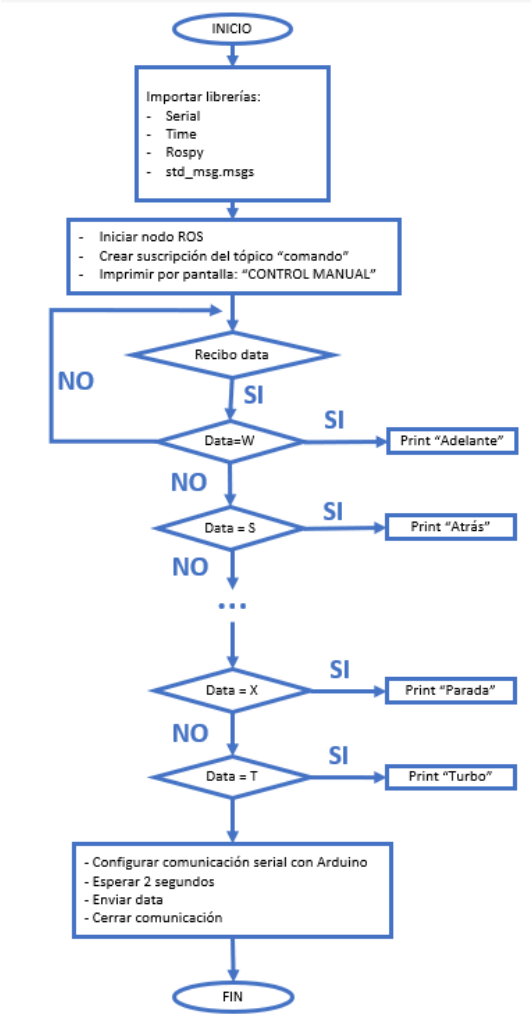


Figura 4.21: Diagrama de flujo del programa principal de nodo_sub_control.py

El segundo modo de funcionamiento otorga la autonomía a la plataforma para dirigir su movimiento hasta una posición señalada por el usuario. En este caso, las decisiones de ruta residen en la plataforma, no en el control manual. Como se ha descrito en la sección 4.1.2 se va a utilizar un paquete de navegación proporcionado por ROS, siguiendo la estructura de nodos representada en la figura 4.12. Se han creado tres nodos acorde a los tres tópicos que deben proporcionar la información al nodo principal move_base.

El primero de ellos enviará los datos de odometría a través del tópico "odom", y será suscriptor del tópico "valores" para obtener la posición, se presenta el diagrama de flujo de la figura 4.22:

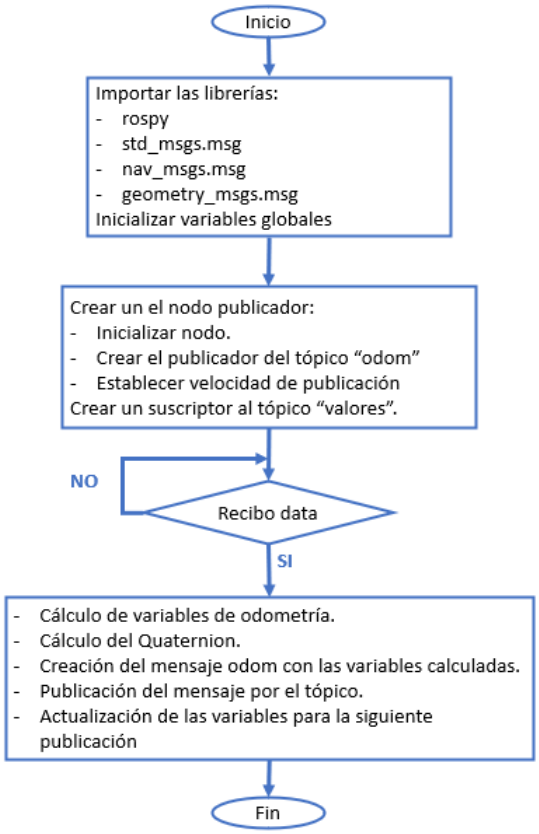


Figura 4.22: Diagrama de flujo del programa principal de nodo_pub_odom.py

Se deben importar las librerías correspondientes para poder crear el mensaje que se va a publicar. Este nodo recibe la información por el tópico "valores" que se publica desde la Raspberry Pi. Las variables que se publican por este tópico están relacionadas con la posición, la velocidad y el cuaternión. Un cuaternión de movimiento es una representación matemática que describe la orientación y rotación de un objeto en un espacio tridimensional, se utiliza comúnmente para representar la orientación de un robot en un sistema de coordenadas tridimensional. Consta de cuatro componentes: un componente escalar y tres componentes vectoriales. La parte escalar representa la rotación alrededor del eje de referencia, mientras que los componentes vectoriales representan los ejes de rotación (x, y, z) y el ángulo de rotación alrededor de cada eje. El cálculo de las variables de odometría se encuentra en la **sección del anexo A.2**. Una vez se ha creado el mensaje se publicará por "odom".

En segundo lugar, se debe crear un nodo que trasmita la información referente a los posibles obstáculos. Este nodo utiliza la información captada por los sensores ultrasonidos y la envía por el tópico "scan", además debe especificar las características técnicas que tiene el sensor utilizado, como el rango mínimo

y máximo, el tiempo de escaneo... etc. Este nodo funciona acorde al diagrama de flujo presentado en la figura 4.23:

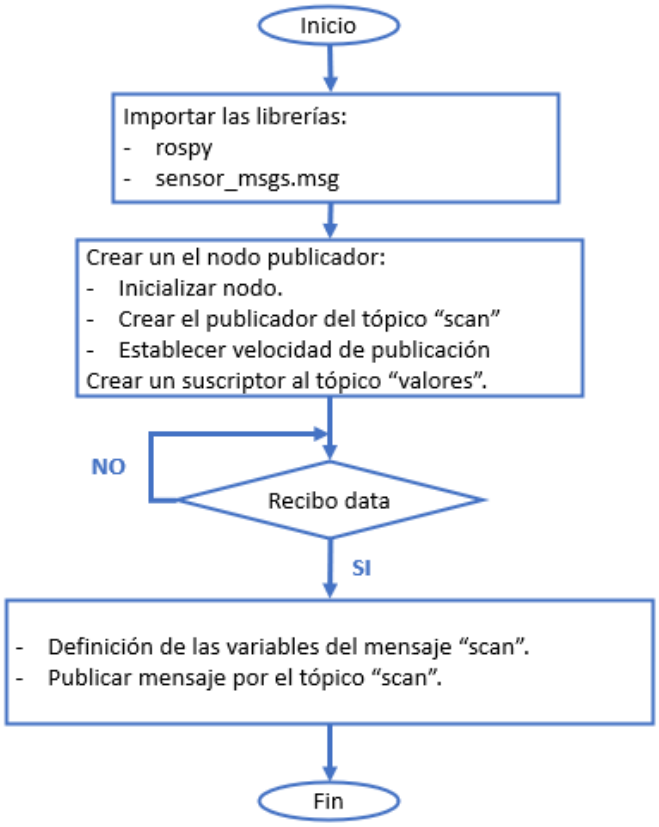


Figura 4.23: Diagrama de flujo del programa principal de nodo_pub_scan.py

El nodo llamado "nodo_pub_odom.py" sigue una estructura de publicador y suscriptor en el contexto del sistema. Este nodo actúa como suscriptor del tópico llamado "valores", desde donde obtiene la información proveniente de los sensores de ultrasonidos. Además, actúa como publicador en el tópico "scan".

El propósito de este nodo es procesar los datos de los sensores de ultrasonidos y proporcionarlos al sistema de navegación "move_base". El tópico "scan" es utilizado por el sistema de navegación para generar trayectorias y tomar decisiones en función de los obstáculos detectados por los sensores.

En el **apéndice A.2** se detallan y describen las variables y parámetros asociados a este tópico. Estas variables incluyen información como la distancia a los obstáculos detectados por los sensores de ultrasonidos, la dirección y la intensidad de las lecturas, entre otros.

El último nodo implementado es denominado "nodo_pub_tf.py", el cual tiene como objetivo publicar las transformaciones de distancias para los marcos de referencia del sistema. Este nodo se encarga de mantener actualizadas las transformaciones entre diferentes marcos de referencia, lo cual es esencial para asegurar la consistencia y precisión de los datos utilizados por los diferentes componentes de la plataforma. El código del nodo sigue el flujo descrito en el diagrama de flujo mostrado en la figura 4.24. Este diagrama de flujo es una representación visual de las diferentes etapas y decisiones que el nodo lleva a cabo durante su ejecución.

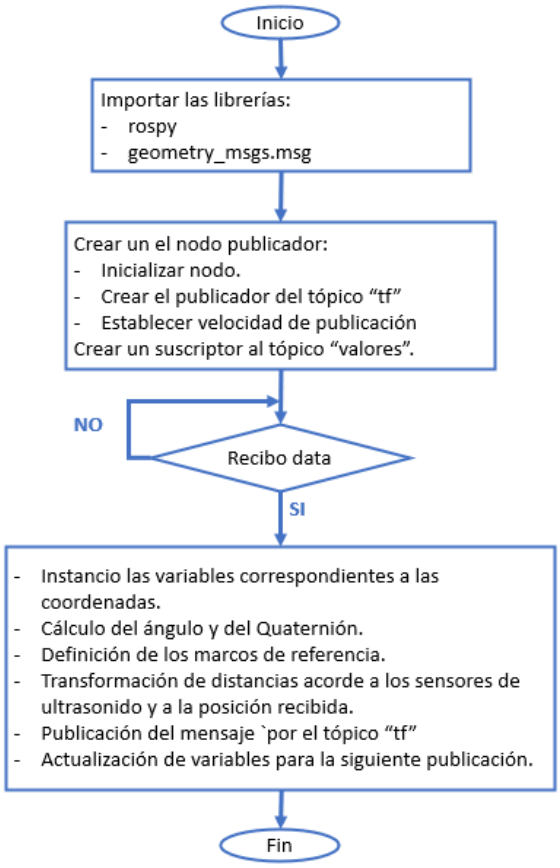


Figura 4.24: Diagrama de flujo del programa principal de nodo_pub_tf.py

El nodo "nodo_pub_tf.py" define dos relaciones clave entre marcos de referencia. La primera establece la posición del robot en relación con el mapa global, mientras que la segunda relaciona los datos de los sensores con la distancia real al obstáculo. Estas transformaciones son críticas para el funcionamiento del paquete de navegación.

Todos los campos de los mensajes publicados en los tópicos "odom", "scan" y "tf" se encuentran en el **anexo A.2**.

Finalmente, se deben crear los archivos de configuración del paquete "Navigation Stack". Estos archivos definen parámetros relacionados con la frecuencia de funcionamiento, los marcos de referencia, tolerancias, etc.

Para lanzar el paquete de navegación se han creado unos archivos que ejecutan los nodos necesarios para el funcionamiento de "Navigation Stack", esto facilita su implementación y economiza el uso de terminales necesarias para visualizar los datos. Estos archivos son: "my_configuration_robot.launch" y "move_base.launch", disponibles en el repositorio de GitHub (**apéndice A.6**).

4.2.3 Diseño e implementación del módulo de visión artificial.

En el apartado 3.1 se puede ver como el punto 4 hace referencia a la creación del módulo de visión artificial, esta es la fase que se aborda en este bloque. Para esta labor, ha sido imprescindible una formación en inteligencia y visión artificial, aunque los conocimientos básicos fueron aprendidos a lo largo de la asignatura de Visión Artificial con la que cuenta el grado. Se hizo un estudio de las redes neuronales que existían para determinar cuál era la más adecuada según la aplicación que se iba a agregar a la

plataforma. Se decidió que YOLO cumplía los requisitos estimados, ya que permite un análisis en tiempo real, ganando en velocidad a otras redes convencionales.

Una vez tomada esta decisión se realizó una investigación para escoger un modelo preentrenado que se ajustase lo máximo posible a las necesidades del proyecto. En este caso, atendiendo a las demandas de los participantes de la encuesta, era fundamental que incluyese la detección de personas y de animales, ya que son los segundos en ser rescatados en las operaciones de emergencia. Otras clases que resultaron de interés y favorecieron la elección del modelo utilizado [33] fueron: boca de incendios, mobiliario y señales, aunque este modelo es capaz de detectar muchos otros objetos.

El propósito de este módulo es ofrecer información en tiempo real. Por ello, fue necesario crear un código que analizara vídeo en vez de imagen, este código corresponde al archivo "yolo_opencv_webcam.py". En la siguiente ilustración se muestra su funcionamiento a través de un diagrama de flujo:

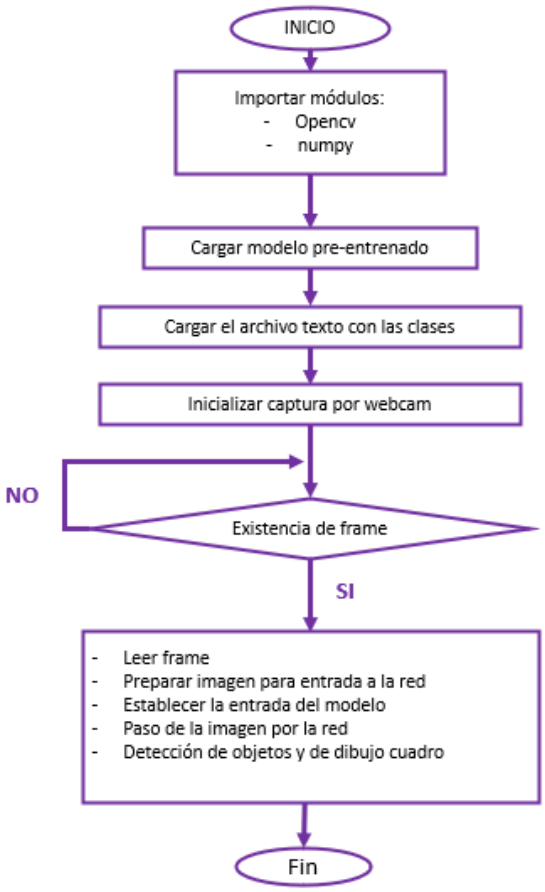


Figura 4.25: Diagrama de flujo del código que rige el comportamiento del módulo de visión artificial.

Para el desarrollo del código es necesario importar las bibliotecas OpenCv [34] y NumPy [35], que incluyen los comandos principales utilizados. Se carga el modelo preentrenado importando los archivos con los pesos y con la configuración de la red neuronal. Seguidamente, se accede también al archivo de clases que cuenta con los nombres que aparecen por pantalla una vez detectado el objeto.

Ahora, se inicia la grabación por webcam y se ejecuta un bucle en el que se analiza uno a uno los frames que toma la cámara. En este bucle, primero se adecua el frame para la entrada a la red neuronal, ajustando la escala, el código de colores, entre otros. Se hace pasar el frame por el modelo, primeramente se obtienen las capas de la red no conectada, estas capas son las que dan los resultados de la detección

de objetos, seguidamente obtenemos como resultado una serie de tensores que contienen la detección de objetos desarrollada por la red.

Una vez obtenido el resultado anterior, se recogen los datos de las puntuaciones obtenidas para la detección de objetos, es decir, la probabilidad de que un objeto pertenezca a una clase. Se asigna la clase con la probabilidad más alta, si esta probabilidad es suficientemente alta se procede a dibujar el marco del objeto detectado en el frame. Esto se hace a partir de las coordenadas del cuadro que devuelve la red neuronal. Finalmente, se nombrará ese marco con la clase de mayor probabilidad determinada anteriormente.

En el **anexo A.5** se describen de forma detallada las funciones utilizas y el desarrollo del código.

Capítulo 5

Resultados

En este apartado se describe la última fase del proyecto, exponiéndose los resultados obtenidos del trabajo realizado a partir del diseño y la puesta en marcha de los módulos que conforman la plataforma.

5.1 Resultados del módulo de monitorización de parámetros ambientales.

Dentro de la integración de la plataforma en ROS, distinguimos dos módulos principales. El módulo de monitorización de parámetros tiene como objetivo ofrecer información de las variables ambientales en tiempo real al equipo de bomberos. Su diseño en ROS ha permitido su futura integración en gemelos digitales.

Para visualizar el comportamiento de los nodos en tiempo real se ha utilizado la herramienta "Rqt". A partir de ella, se comprueba el resultado de la integración en ROS de este módulo. Esta herramienta muestra los nodos que están actuando en ese preciso instante y por qué tópicos se está trasmitiendo información. En la figura 5.1 se muestra el resultado:

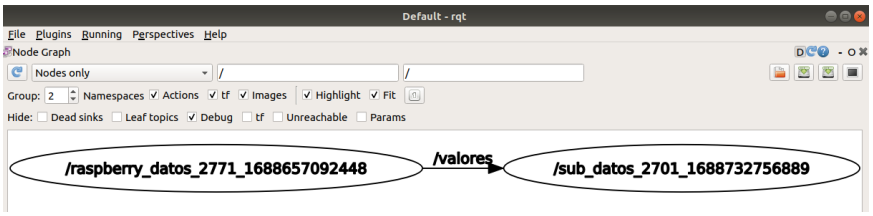


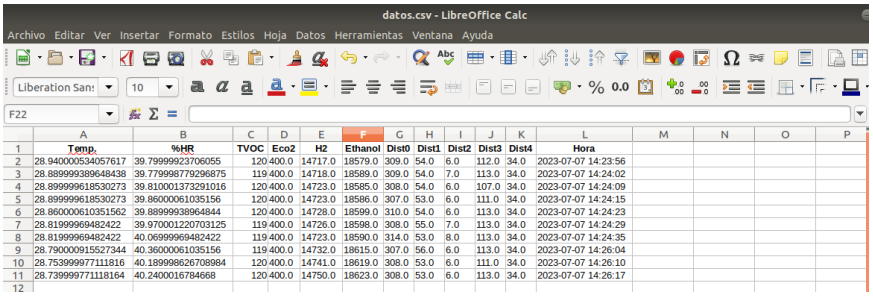
Figura 5.1: Resultado de la integración en ROS del bloque de monitorización de parámetros.

De esta forma, se alcanza parte del objetivo descrito en la sección 2 referente a la integración en ROS. Se ha integrado el funcionamiento de la monitorización de parámetros en ROS. Una vez se ha comprobado el funcionamiento de esta estructura, se procede a la realización de pruebas. Para este apartado se han realizado dos pruebas. La primera de ellas se realiza con el robot parado, de esta forma se prueba la lectura de los sensores y la función implementada para mostrar los datos por pantalla al usuario.

```
('Temp: ', 30.700000762939453)
('HR: ', 27.780000686645508)
('TVOC: ', 0.0)
('eCO2: ', 400.0)
('H2: ', 14516.0)
('Ethanol: ', 18466.0)
('Dist 1: ', 18.0)
('Dist 2: ', 106.0)
('Dist 3: ', 13.0)
('Dist 4: ', 20.0)
('Dist 5: ', -1.0)
```

Figura 5.2: Visualización de los parámetros ambientales en tiempo real por la pantalla del usuario.

Como se puede observar en la figura 5.2 el usuario puede ver las medidas tomadas por todos los sensores en tiempo real. Además, se puede probar la exportación de los datos a un archivo CSV en tiempo real, a continuación, en la figura 5.3 se observa el resultado de esta nueva funcionalidad.



	A	B	C	D	E	F	G	H	I	J	K	L
1	Temp.	%HR	TVOC	Eco2	H2	Ethanol	Dist0	Dist1	Dist2	Dist3	Dist4	Hora
2	28.940000534057617	39.79999923706055	120	400.0	14717.0	18579.0	309.0	54.0	6.0	112.0	34.0	2023-07-07 14:23:56
3	28.8899993896484 38	39.779998779296875	119	400.0	14718.0	18589.0	309.0	54.0	7.0	113.0	34.0	2023-07-07 14:24:02
4	28.899999618530273	39.810001373291016	120	400.0	14723.0	18585.0	308.0	54.0	6.0	107.0	34.0	2023-07-07 14:24:09
5	28.899999618530273	39.86000061035156	120	400.0	14723.0	18586.0	307.0	53.0	6.0	111.0	34.0	2023-07-07 14:24:15
6	28.860000610351562	39.88999938964844	120	400.0	14728.0	18599.0	310.0	54.0	6.0	113.0	34.0	2023-07-07 14:24:23
7	28.81999969482422	39.97000122070312 5	119	400.0	14726.0	18598.0	308.0	55.0	7.0	113.0	34.0	2023-07-07 14:24:29
8	28.81999969482422	40.06999969482422	119	400.0	14723.0	18590.0	314.0	53.0	8.0	113.0	34.0	2023-07-07 14:24:35
9	28.790000915527344	40.36000061035156	119	400.0	14732.0	18615.0	307.0	56.0	6.0	113.0	34.0	2023-07-07 14:26:04
10	28.753999977111816	40.18999862670898 4	120	400.0	14741.0	18619.0	308.0	53.0	6.0	111.0	34.0	2023-07-07 14:26:10
11	28.739999771118164	40.2400016784668	120	400.0	14750.0	18623.0	308.0	53.0	6.0	113.0	34.0	2023-07-07 14:26:17
12												

Figura 5.3: Resultado de la exportación en tiempo real de los datos obtenidos por la plataforma a un archivo CSV.

El escenario fue un entorno cerrado, sin ventilación a la hora y fecha indicada en la tabla del archivo. Por lo tanto, los resultados obtenidos pueden justificar el correcto funcionamiento de este módulo.

En segundo lugar, se realizó la misma prueba, pero en diferentes condiciones ambientales, de esta forma se podía probar si los sensores estaban correctamente calibrados y adaptados para funcionar en distintos espacios. Esta segunda prueba se ha realizado en un espacio con ventilación Gracias a esta nueva implementación se pueden desarrollar estudios sobre el comportamiento del entorno una vez terminada la operación, siendo el archivo accesible en cualquier momento. Por otro lado, en el terminal que controla el usuario en tiempo real aparecen los siguientes datos a medida que la plataforma avanza (figura 5.4):

	A	B	D	E	F	G	H	I	J	K	L
1	Temp.	%HR	Eco2	H2	Ethanol	Dist0	Dist1	Dist2	Dist3	Dist4	Hora
2	29.969999313354492	31.190000534057617	320.0	14912.0	18897.0	20.0	10.0	120.0	18.0	28.0	2023-07-02 11:27:13
3	29.979999542236328	32.040000915527344	321.0	14890.0	18888.0	61.0	10.0	122.0	21.0	25.0	2023-07-02 11:27:19
4	30.030000686645508	31.31999969482422	315.0	14951.0	18909.0	50.0	10.0	129.0	21.0	20.0	2023-07-02 11:27:26
5	30.030000686645508	30.59000015258789	315.0	14968.0	18932.0	45.0	10.0	135.0	20.0	16.0	2023-07-02 11:27:32
6	30.049999237060547	30.360000610351562	315.0	14988.0	18945.0	40.0	10.0	140.0	20.0	13.0	2023-07-02 11:27:38

Figura 5.4: Visualización de los parámetros ambientales en tiempo real por la pantalla del usuario.

Además de ofrecer una mejora sustancial a la herramienta al aumentar la cantidad de datos que proporciona la plataforma, este proyecto permite ampliar su aplicación en nuevos escenarios, en estrecha colaboración con los equipos de emergencia nacionales.

5.2 Resultados del módulo de navegación.

Otro de los módulos que se han diseñado para su integración en ROS es el módulo de navegación. Como se ha citado en secciones anteriores, este bloque consta de dos modos de funcionamiento: el modo manual y el modo de navegación autónoma.

En cuanto al módulo manual, se ha conseguido que la plataforma obedezca a las órdenes dadas por el usuario a través de una estructura de nodos basada en ROS. De nuevo, se ha hecho uso de la herramienta "Rqt" para probar la conexión entre los nodos por el tópico "comando".

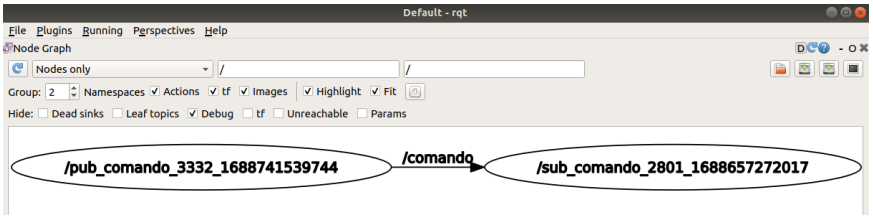


Figura 5.5: Resultado de la integración en ROS del módulo de navegación en control manual.

Como se puede ver en la figura 5.5 la integración del control manual se ha desarrollado correctamente. Se observa la comunicación entre el nodo publicador y el nodo suscriptor del tópico "comando".

Para demostrar que los motores atienden a las órdenes se realiza una segunda prueba en la que se ejecutarán los nodos de monitorización y los de control manual para mostrar el avance de la plataforma por en la dirección establecida por el comando.

En la siguiente imagen, se presenta la ruta que seguirá el robot:

Figura 5.6: Ruta sobre plano para la prueba del control manual.

Para el recorrido se ha ejecutado el nodo publicador de comandos en el ordenador, a continuación se muestran los comandos utilizados:

```
siniesta2019@Ubuntu18:~/catkin_ws$ rosrun control_manual keyboard_ordenador.py
Introduce un comando de movimiento: W
Introduce un comando de movimiento: A
Introduce un comando de movimiento: A
Introduce un comando de movimiento: X
Introduce un comando de movimiento:
```

Figura 5.7: Comandos enviados desde nodo_pub_control para la ejecución de la ruta.

Mediante una conexión SSH se abre una terminal en la Raspberry Pi, y se ejecuta el nodo suscriptor del tópico “comando”. Además, para la posición de las balizas se debe ejecutar el nodo nodo_pub_datos para obtener las coordenadas del robot y probar su recorrido. En la siguiente imagen se muestra el resultado de los comandos recibidos por la Raspberry Pi:

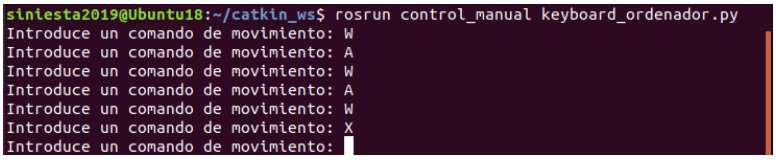


Figura 5.8: Comandos recibidos por la Raspberry Pi.

A continuación se muestra la posición recogida por las balizas:

	A	B	C	D	E
1	coorx	coory	time		
2	171.0	154.0	2023-07-07 11:19:02		
3	212.0	155.0	2023-07-07 18:19:09		
4	284.0	156.0	2023-07-07 18:19:15		
5	318.0	193.0	2023-07-07 18:19:23		
6	381.0	196.0	2023-07-07 18:19:31		
7	375.0	210.0	2023-07-07 18:19:38		
8	258.0	207.0	2023-07-07 18:19:45		
9	191.0	209.0	2023-07-07 18:19:53		
10	161.0	209.0	2023-07-07 18:20:01		

Figura 5.9: Tabla de posiciones en formato CSV.

Las posiciones recogidas se ajustan a la ruta dirigida por los comandos publicados por el nodo del ordenador. Por lo que este módulo funciona correctamente y su integración en ROS es óptima.

Por otro lado, se ha configurado el paquete de navegación que integra ROS ("Navigation Stack"), para la consecución de una navegación autónoma. Gracias a la incorporación de los nuevos sensores ultrasonidos es posible que la plataforma detecte obstáculos y los evite para llegar a la posición objetivo. Como se ha descrito en la sección 4.2.2, el sistema lo conforman tres nodos publicadores en los tópicos “odom”, “tf” y “scan”, y un nodo generado por el paquete de navegación “move_base”. Para obtener la posición de la plataforma y las medidas tomadas por los sensores ultrasonidos se reutiliza el nodo publicador del bloque de monitorización “nodo_pub_valores”. Los pasos que se han de seguir para poner en funcionamiento este paquete de navegación se encuentra en el **anexo A**.

Una vez se ha lanzado el paquete de navegación, es posible monitorizar su comportamiento con la herramienta vista anteriormente (“Rqt”).

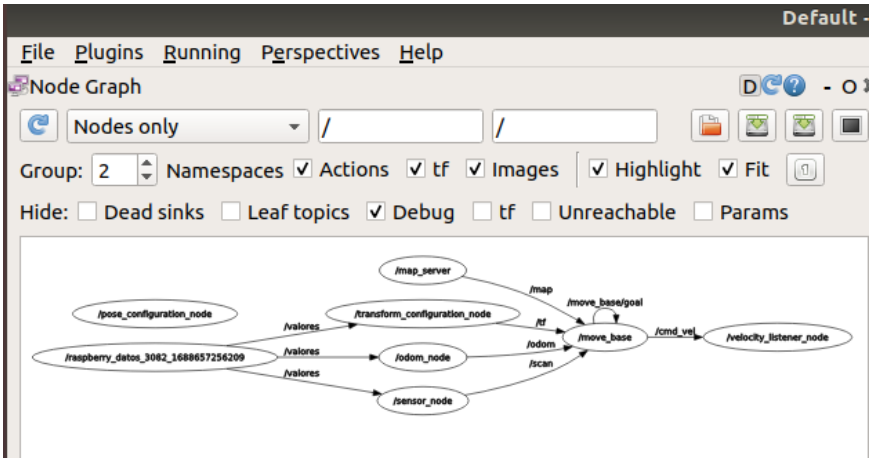


Figura 5.10: Trayectoria del Robot visualizada desde el entorno RViz.

Atendiendo a lo mostrado en la figura 5.10 queda probado el lanzamiento del paquete de navegación autónoma, abordando el objetivo principal del proyecto definido en el capítulo 2.

Una vez se ha logrado lanzar el paquete de navegación autónoma, se procede a monitorizar su comportamiento mediante la herramienta "RViz". Esta herramienta permite la visualización 3D de un entorno de desarrollo de ROS. Proporciona la interfaz gráfica que permite visualizar y analizar datos generados por los componentes del sistema robótico. Ofrece una representación de trayectorias, mapas, datos de odometría... etc.

Con RViz se ha obtenido la siguiente imagen:

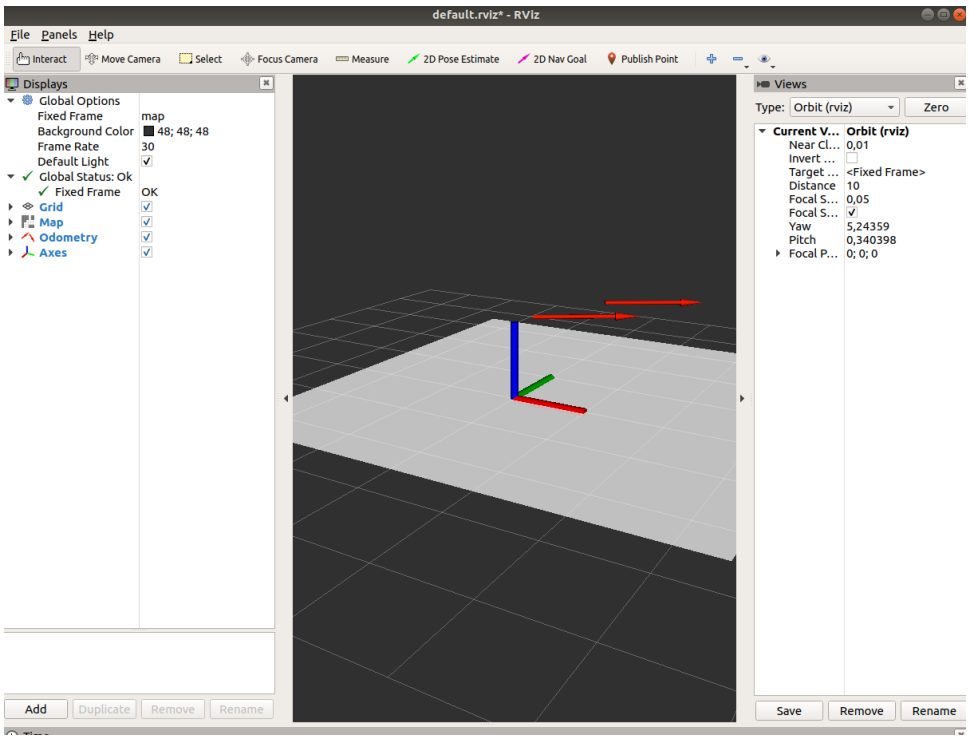


Figura 5.11: Trayectoria del Robot visualizada desde el entorno RViz.

En la figura 5.11 se puede ver el resultado obtenido con la herramienta RViz. Se muestra el mapa, que en este caso se ha presentado un mapa en blanco. Y la trayectoria del robot marcada por los datos de odometría publicados.

Los resultados se han obtenido a partir de una configuración inicial primaria, ya que se trata de un robot no registrado comercialmente. Por esta razón, se han asignado unos parámetros de prueba que proporcionan unos resultados preliminares.

Además, se quería probar el funcionamiento de reconocimiento del entorno mediante el uso de sensores ultrasonidos. A pesar de que este sistema es viable, su funcionamiento es muy primitivo, en un entorno real resultaría más adecuado el uso de tecnología láser.

5.3 Resultados del módulo de visión artificial.

La incorporación de un sistema de visión artificial ha implicado un gran avance en la plataforma, ya que le permite abrir nuevas líneas de desarrollo y estar en línea con las novedades que incorpora la inteligencia artificial.

En la encuesta realizada en el apartado 1.1.1 los participantes daban gran importancia a la presencia de víctimas en las situaciones de emergencia. Por ello, se propuso como objetivo la incorporación de un módulo de visión artificial que fuese capaz de detectar objetos y posibles víctimas.

Como primer paso, se hizo una prueba inicial para comprobar el funcionamiento del modelo con imágenes propias. Se analizó su respuesta ante imágenes de vehículos (figura 5.12).

(a) Imagen original de calle con vehículos.

(b) Imagen pasada por la red neuronal.

Figura 5.12: Detección de vehículos estacionados en una calle.

En la ilustración 5.12 se observa que tras el paso de la imagen por la red neuronal se pueden ver los vehículos detectados con su correspondiente clase y su marco. De esta forma, aseguramos que el archivo del que toma los nombres de las clases es correcto. La plataforma está diseñada para actuar en interiores. Sin embargo, este módulo podría incorporarse en otros dispositivos, permitiendo tener una visión preliminar sobre el volumen de vehículos estacionados en la ubicación del suceso. Esto podría ayudarles a los bomberos establecer una ruta más rápida hasta el lugar de la emergencia.

La información más relevante para los bomberos es la detección de víctimas, por ello, el segundo experimento realizado fue con múltiples imágenes de personas en las que variaba la posición, la expresión y la etnia. A continuación, se muestra el resultado obtenido.

Figura 5.13: Resultado de la detección de personas por el modelo utilizado.

En la ilustración 5.13 se observa como la red neuronal realiza una detección exitosa para todas las imágenes. Este resultado fue clave para la incorporación del módulo de visión artificial en la plataforma.

Una vez se ha comprobado que la red detecta con éxito personas con diferentes rasgos. Se procede a comprobar como actúa el modelo ante personas en diferentes posturas. De esta forma podríamos comprobar su capacidad de detectar posibles víctimas que hayan quedado inconscientes en el lugar del accidente.

Figura 5.14: Resultado de la detección de personas por el modelo utilizado. [8]

En la figura 5.14 se observa como detecta perfectamente la presencia de una persona y los objetos alrededor que pueden influir en su rescate. Por lo que la herramienta brinda muy buenos resultados para la aplicación que se quería desarrollar.

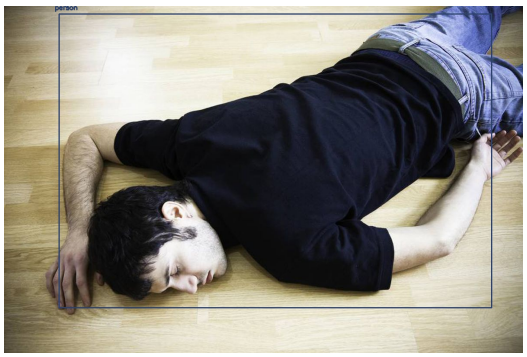
Para el desarrollo del módulo se requería la detección en tiempo real, primeramente se adaptó el código para poder incorporar esta funcionalidad. Esta prueba se realizó pasando por la red neuronal un vídeo de un anuncio. Para la integración de este bloque se presentan dos opciones:

1. Incorporar el módulo en la misma Raspberry Pi que utiliza el resto de la plataforma.
2. Diseñar el sistema en otra Raspberry Pi y anexionarla a la plataforma.

Las dos opciones son válidas, sin embargo, con la segunda opción se pretende tener una visión más amplia dirigida al desarrollo de futuras aplicaciones que se detallarán más adelante en el apartado 6.2. Además, optimiza el desarrollo de las demás funciones de la plataforma integradas en la primera Raspberry Pi.

A continuación, se muestran las imágenes obtenidas con la red neuronal incorporada en el módulo (figura 5.15).

(a) Imagen original de posible víctima

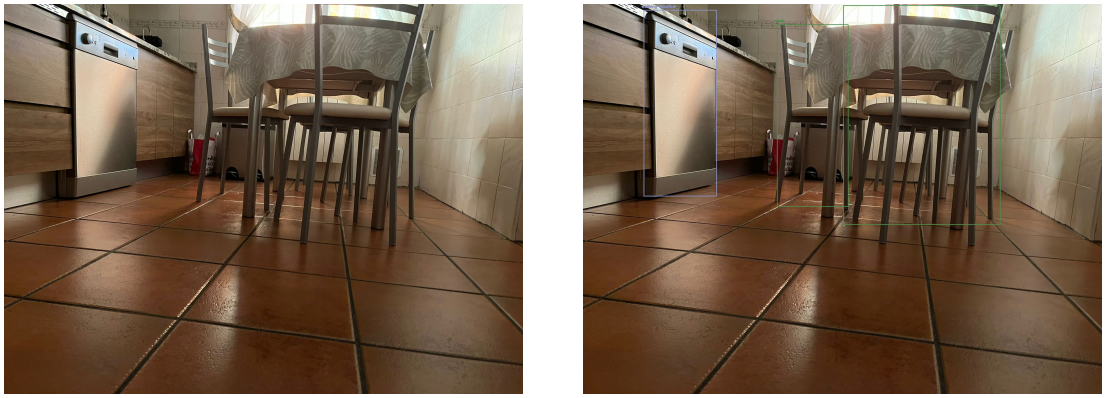


(b) Imagen pasada por la red neuronal.

Figura 5.15: Detección de posible víctima como resultado del módulo de visión artificial.

La primera figura presenta la imagen original tomada, cuando esa imagen pasa por la red neuronal obtenemos como resultado la figura 5.15b.

Finalmente, en los obstáculos que pueden aparecer en el recorrido de la intervención que realizan los bomberos también cobran relevancia, por ello se han tenido en cuenta en esta plataforma. El resultado es el mostrado en la figura 5.16.



(a) Imagen original obstáculos.

(b) Imagen pasada por la red neuronal.

Figura 5.16: Detección del mobiliario presente en un entorno cerrado.

A pesar de tratarse de una cámara webcam, la red neuronal responde correctamente ante condiciones de luminosidad baja, ya que se trata de una red neuronal muy potente.

Además, no solo ha reconocido posibles obstáculos como la silla. También ha detectado el lavavajillas, que es una fuente de calor que puede influir en el desarrollo de un incendio.

Las pruebas realizadas en los diferentes módulos confirman la eficacia de cada componente. Aunque los resultados obtenidos en el módulo de navegación son preliminares, se ha demostrado la viabilidad de adaptar el paquete de navegación proporcionado por ROS a un modelo no comercializado. Además, la integración de nuevos sensores en la plataforma de monitorización ha mejorado significativamente el reconocimiento del entorno. Por último, la exitosa incorporación de un sistema de visión artificial capaz de reconocer objetos y posibles víctimas ha supuesto un gran avance para la plataforma.

Esta plataforma ha sido equipada con un sistema de navegación autónoma, un módulo de visión artificial y una nueva función que permite exportar los datos de los sensores a un archivo accesible en cualquier momento. Estas nuevas características acercan la plataforma a los modelos descritos en el estado del arte, ya que comparte similitudes con los tres modelos mencionados. Sin embargo, la capacidad de acceder a los datos en cualquier momento y la integración de una red neuronal YOLO en la cámara son funciones exclusivas de esta plataforma en particular.

Capítulo 6

Conclusiones y líneas futuras

Una vez presentados los resultados del proyecto se procede a exponer las conclusiones obtenidas y se proponen futuras líneas de investigación que se deriven de este trabajo. Este trabajo nace de la motivación de ofrecer una herramienta completa al equipo de bomberos, siguiendo las líneas de trabajo de la robótica actual. Se ha propuesto una solución a la falta de información visual en situaciones de emergencia dadas en interiores, y se han integrado sus funcionalidades en un sistema robusto que ha permitido incorporar la navegación autónoma a la plataforma. Se han realizado pruebas con imágenes y vídeos tomados por la webcam desde el robot, y se ha monitorizado mediante el simulador RViz y la herramienta Rqt, el comportamiento del sistema de navegación autónoma.

6.1 Conclusiones

Para el cumplimiento del objetivo referente a una mejora del sistema hardware que permita posteriormente dotar de autonomía a la plataforma, se han añadido nuevos componentes hardware que proporcionen más información acerca del entorno del robot. Los cinco sensores ultrasonidos añadidos han sido fundamentales para la detección de obstáculos y la creación de trayectorias del sistema de navegación autónoma. Además, se ha habilitado la exportación de información en tiempo real a un archivo CSV, de esta forma los datos son accesibles en cualquier momento y permiten un estudio posterior a la intervención.

En segundo lugar, se ha integrado el funcionamiento del agente en una plataforma de diseño robótico. Este logro es fundamental, ya que permite desarrollar aplicaciones más robustas que trabajen en tiempo real y facilita la incorporación de la plataforma en entornos de simulación como Unity o Gazebo. La integración en ROS, era necesaria para el desarrollo de un sistema de navegación autónoma que permita al robot crear sus propias trayectorias para llegar a un objetivo marcado.

Para llevar a cabo el objetivo relativo a dotar de autonomía a la plataforma y tras mejorar las antiguas funcionalidades del agente integrándolas en ROS, se ha implementado el sistema de navegación autónoma en el mismo software. Esto ofrece versatilidad y velocidad al agente, ya que pasa a ser un sistema independiente que no necesita que un usuario le marque la trayectoria de forma manual, sino que se le proporcione una meta y el robot cree su propia ruta evitando obstáculos hacia ella. Gracias al cumplimiento de este objetivo, se puede agilizar el trabajo de los profesionales, ya que no será necesario que un operario dirija manualmente al robot y este pendiente de que este no colisione, sino que podrá mandarle un punto objetivo y simplemente visualizar su recorrido por pantalla.

Por último, y como nueva línea de investigación, se pretendía desarrollar un sistema que ofreciese más información del entorno al equipo de bomberos. Por consiguiente, se ha implantado en la plataforma un

módulo de visión artificial, otro gran avance que abre una nueva línea de investigación en el proyecto. Esta novedad enriquece la información proporcionada al equipo de bomberos, no solo trasmite la imagen en vivo del entorno, sino que cuenta con un sistema de detección que permite saber si hay víctimas, tanto humanas como animales, y los obstáculos que se puede encontrar el bombero en su incursión, como sillas, mesas y demás mobiliario. Además, detecta ventanas, este dato puede ayudar a los bomberos para buscar caminos de fuga de humos. Permite así a la plataforma estar en las líneas de trabajo actuales de la robótica.

6.2 Líneas futuras

En este Trabajo de Fin de Grado, se ha proporcionado una nueva línea de trabajo en la plataforma de asistencia para situaciones de emergencia dadas en interiores. El sistema de navegación autónoma incorporado tiene algunas limitaciones en su comportamiento debido al uso de sensores ultrasonidos para la detección de obstáculos. Por otra parte, el sistema de visión artificial está comprometido por las condiciones de visibilidad que ofrezca el entorno.

Por lo descrito anteriormente, se propone el diseño de un nuevo sistema para la navegación autónoma que este conformado por un sensor láser, además este componente, permitiría la creación de mapas de entorno que incluye ROS. Esta mejora, perfeccionaría el sistema de navegación autónoma y sería de gran utilidad en las intervenciones en interiores como viviendas o naves industriales. Otro aspecto que mejoraría el funcionamiento de la plataforma, es la instalación de todo el desarrollo en un robot "TurtleBot". Este robot tiene una alta compatibilidad con ROS y el ajuste de parámetros necesario para la navegación autónoma es más accesible.

Por otro lado, a pesar de que la herramienta RViz permite visualizar el comportamiento de la plataforma en su actuación, sería interesante proporcionar un entorno de simulación mediante herramientas como Unity o Gazebo. Gracias a la herramienta ROS, la actuación del agente real y el entorno simulado es idéntica, por su compatibilidad con las herramientas de simulación mencionadas. De esta forma, la representación sería más clara y visual, facilitando la comprensión del funcionamiento de la plataforma.

En cuanto a la información ofrecida por el sistema de visión artificial, las condiciones de baja visibilidad pueden comprometer su funcionamiento, por ello, se propone adaptar el sistema para poder usar una cámara térmica que permita detectar objetos y posibles focos de incendios. Para este sistema, podría implementarse un mecanismo que moviera la cámara para ofrecer una visión de 360º del entorno.

Bibliografía

[1] Milán Sonka, Vaclav Hlavac, Roger Boyle. Image Processing, Analysis, and Machine Vision, 4th ed. Disponible en: https://kgut.ac.ir/useruploads/1550563201478ety.pdf

[2] Raspberry Pi. Raspberry Pi: Computing for everybody. Disponible en: https://www.raspberrypi.com/

[3] Accesories, M. Diagrama de la técnica de trilateración para el posicionamiento en interiores. Disponible en: https://www.researchgate.net/figure/Trilateracion-La-distancia-a-tres-balizas-nodos-blancos-permite-a-un-sensor-nodo_fig1_228705728

[4] IBM. Protocolos TCP/IP. Disponible en: https://www.ibm.com/docs/es/aix/7.1?topic=protocol-tcpip-protocols

[5] Mouser. Sensor Adafruit SGP30. Disponible en: https://www.mouser.es/ProductDetail/Adafruit/3709?qs=rrS6PyfT74crPlmcf95bYg%3D%3D

[6] Accesories, M. Cámara LifeCam HD-3000 Webcam. Disponible en: https://www.microsoft.com/en/accessories/products/webcams/lifecam-hd-3000?activetab=pivot:overviewtab

[7] ROS. Arquitectura del paquete de navegación de ROS. Disponible en: http://wiki.ros.org/navigation/Tutorials/RobotSetu

[8] Auxilios, P. Representación de una personas incosnciente para las pruebas de la detección de objetos. Disponible en: https://primerosauxilios.org.es/desmayo

[9] Fundación Mapfre. Informe víctimas de incendios en España en 2021. Disponible en: https://documentacion.fundacionmapfre.org/documentacion/publico/es/media/group/1118213.do

[10] Comunidad de Madrid. Bomberos - Presupuestos abiertos de la Comunidad de Madrid. Disponible en: https://presupuestosabiertos.madrid.es/es/programas/13610/bomberos#view=economic&year=2023

[11] Adriana Margarita Porcelli, Universidad de LujÃ¡n. Influencia de machine learning en la robótica. Disponible en: https://www.scielo.org.mx/scielo.php?script=sci_arttext&pid=S2448-51362020000300049

[12] García Prieto Cuesta. J. Â¿Qué es un robot?- García- Prieto Cuesta, J. (2018). Disponible en: javascript:void(0)

[13] Mobile World Capital. Los últimos pasos de la robótica en España y en todo el mundo. Disponible en: https://mobileworldcapital.com/avances-robotica-espana/

[14] Naciones Unidas. Robots al rescate: Uso de la tecnología para mitigar los efectos de los desastres naturales. Disponible en: https://www.un.org/es/robots-al-rescate-uso-de-la-tecnolog~A\protect\discretionary{\char\hyphenchar\font}{}{}a-para-mitigar-los-efectos-de-los-desastres-naturales

[15] Eric Rohmer, Tomoaki Yoshida, Kazunori 0hno, Keiji Nagatani, Satoshi Tadokoro, Eiji Konayagi . Robot Quince. Disponible en: https://www.jstage.jst.go.jp/article/jsmeicam/2010.5/0/2010.5_225/_pdf

[16] SmokeBot . Platafomra SmokeBot. Disponible en: https://www.smokebot.eu/

[17] . Shark Robotics. Disponible en: https://www.shark-robotics.com/

[18] Shark Robots. Agente Colossus. Disponible en: https://www.shark-robotics.com/robots

[19] Diego Salas Prieto. Implementación de tecnología UWB para localización y guiado de interiores. B.m., 2019. Universidad Rey Juan Carlos. Disponible en: https://burjcdigital.urjc.es/handle/10115/18048

[20] Noelia Fernández Talavera. Sistema de navegación autónomo en entornos reales y simulados para situaciones de emergencia. Disponible en: https://burjcdigital.urjc.es/handle/10115/18048

[21] ROS. ROS (Robot Operating System. Disponible en: https://www.ros.org/

[22] M. Cristina Rodríguez-Sánchez, Luis Fernández-Jiménez, Antonio R. Jiménez, Joaquín Vaquero, Susana Borromeo, Jose L. Lázaro-Galilea. Help Responder-System for security of First Responder Intervetions. Disponible en: https://www.mdpi.com/1424-8220/21/8/2614

[23] Catedra SMARTE2. Cátedra SMARTE2 Universidad Rey Juan Carlos. Disponible en: https://smarte2urjc.es/

[24] Cátedra SMARTE2. Poryecto GUIDE2FR. Disponible en: https://guide2fr-cda.senialab.com/accounts/login/?next=/

[25] Grupo de Trabajo de Innovación de la Asociación Española de Robótica y Automatización (AER). Uso de ROS. Disponible en: https://www.aer-automation.com/wp-content/uploads/2022/03/ROS_articuloAER.pdf

[26] Arduino Tecnology. Arduino. Disponible en: https://www.arduino.cc/

[27] Pozyx. Pozyx: Unleash the power of real-time location solutions. Disponible en: https://www.pozyx.io/

[28] Martín, C. A. V. Incendio en túneles y Galerías. Criterios de Intervención. Disponible en: https://www.aptb.org/

[29] Ministerio para la transición ecológica y el reto demográfico de España. Compuestos Orgánicos Volátiles. Disponible en: https://www.miteco.gob.es/es/calidad-y-evaluacion-ambiental/temas/atmosfera-y-calidad-del-aire/emisiones/act-emis/compuestos_organicos_volatiles.aspx

[30] Centro Nacional del Hidrógeno. Composición y características del hidrógeno molecular. Disponible en: https://www.cnh2.es/seguridad/

[31] Mouser. Sensor Adafruit HTU21D-F. Disponible en: https://www.mouser.es/new/adafruit/adafruit-htu21d-f-sensor/

[32] Tinkercad. Diseño y simulación de circuitos electrónicos. Disponible en: https://www.tinkercad.com/

[33] Arun Ponnusamy. Modelo pre-entrenado de red neuronal YOLO para la detección y clasificación de objetos. Disponible en: https://towardsdatascience.com/yolo-object-detection-with-opencv-and-python-21e50ac599e9

[34] Opencv. Disponible en: https://opencv.org/

[35] Numpy. Disponible en: https://numpy.org/

Apéndice A

Anexo I

A.1 Cálculo de las variables de odometría para la navegación autónoma.

Para el cálculo de las variables de odometría, se presenta la siguiente figura que ilustra el escenario de las variables tomadas (figura A.1):

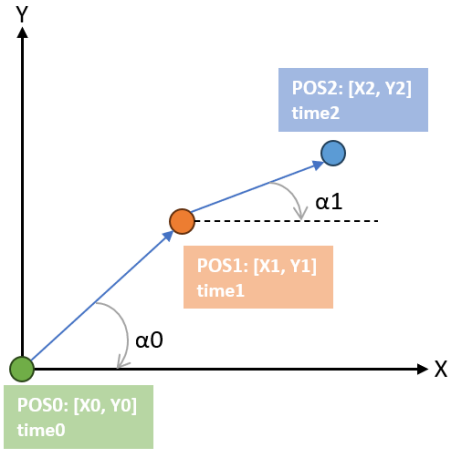


Figura A.1: Visualización de los parámetros ambientales en tiempo real por la pantalla del usuario.

El eje z no se ha tenido en cuenta, ya que movimiento del robot no varía en ese plano. Las variables a calcular son:

1. Ángulo.
2. Distancia.
3. Intervalo de tiempo.
4. Velocidad.
5. Velocidad Angular.

Ecuación cálculo del ángulo:

$$\alpha = \arctan\left(\frac{X1 - X0}{Y1 - Y0}\right) \tag{A.1}$$

Ecuación cálculo de la distancia:

$$distancia = \sqrt{(X0 - X1)^2 + (Y0 - Y1)^2} \tag{A.2}$$

Ecuación para el cálculo del incremento de tiempo:

$$deltatime = time1 - time0 \tag{A.3}$$

Ecuación para el cálculo de velocidad:

$$velocidad = \left(\frac{distancia}{deltatime}\right) \tag{A.4}$$

Ecuación para el cálculo de la velocidad angular:

$$velocidadangular = \left(\frac{\alpha 1 - \alpha 0}{deltatime}\right) \tag{A.5}$$

A.2 Parámetros de los tópicos de navegación autónoma.

Como se ha descrito a lo largo del proyecto, el paquete de navegación autónoma se rige por tres tópicos:

1. Odometría: odom
2. Fuente de sensores: scan
3. Transformación de distancias: tf

Cada tópico tiene sus propias variables que deben incluirse en el mensaje de los mismos. En primer lugar, el tópico de odometría consta de los siguientes:

Variable del mensaje	Variables adjudicadas	Función
odom.header.stamp	time	Encabezado del mensaje
odom.header.frame_id	base_link	Marcode referencia global
odom.pose.pose	x, y, z, Quaternion	Campos de posición y orientación
odom.child_frame_id	base_link	Marco de referencia asociado al robot
ódom.twist.twist.twist	velocidad, velocidad angular	Campos de velocidad linear y angular

Figura A.2: Tabla de los campos del mensaje correspondiente al tópico odom

En segundo lugar, el tópico de transformación de distancia entre los sensores y la plataforma contempla los siguientes campos:

Variable del mensaje	Variables adjudicadas	Función
transform.header.stamp	Time.now()	Encabezado del mensaje
transform.header.frame_id	base_link	Marco de referencia padre
transform.child_frame_id	sensor_frame	Marco de referencia de hijo
transform.translation.x	0,15 cm	Traslación en eje x
transform.translation.y	0	Traslación en eje y
transform.translation.z	0	Traslación en eje z
transform.rotation.x	0	Rotación de la transformación en x
transform.rotation.y	0	Rotación de la transformación en y
transform.rotation.z	0	Rotación de la transformación en z
transform.rotation.w	1	Rotación de la transformación en w

Figura A.3: Tabla de los campos del mensaje correspondiente al tópico transformación de distancias.

La variable referente a la rotación del cuaternión en w, representa la rotación en torno al eje imaginario.

En tercer lugar, el tópico de transformación de distancia entre el robot y el mapa. Pretende ajustar la relación entre la posición del robot y el eje de coordenadas que establece el mapa. Sus campos son idénticos a los presentados en la tabla anterior, sin embargo, en este caso varían las variables asignadas. Está formado por los siguientes campos:

Variable del mensaje	Variables adjudicadas	Función
transform2.header.stamp	Time.now()	Encabezado del mensaje
transform2.header.frame_id	map	Marco de referencia padre
transform2.child_frame_id	base_link	Marco de referencia hijo
transform2.translation.x	x0	Traslación en eje x
transform2.translation.y	y0	Traslación en eje y
transform2.translation.z	z0	Traslación en eje z
transform2.rotation.x	quat[0]	Rotación de la transformación en x
transform2.rotation.y	quat[1]	Rotación de la transformación en y
transform2.rotation.z	quat[2]	Rotación de la transformación en z
transform2.rotation.w	quat[3]	Rotación de la transformación en w

Figura A.4: Tabla de los campos del mensaje correspondiente al tópico transformación de distancias.

Y por último, el tópico por el que se publica la información relativa al entorno del robot. El tópico "scan", recoge la información de los sensores ultrasonidos y de su escaneó. Este tópico normalmente se basa en el uso de un escáner láser, aunque en este caso se pretendía comprobar su funcionamiento con los sensores de distancia. Consta de los campos:

Variable del mensaje	Variables adjudicadas	Función
scan.header.frame_id	sensor_frame	Encabezado con el marco de referencia
scan.angle_min	180	Valor mínimo del ángulo de exposición
scan.angle_max	180	Valor máximo del ángulo de exposición
scan.angle_increment	45	Incremento angular entre las medidas
scan.scan_time	5	Tiempo total de excaneo completo
scan.time_increment	1	Incremento de tiempo entre medidas
scan.range_min	0.02	Valor mínimo de medida del sensor
scan.range_max	150	Valor máximo de medida del sensor
scan.ranges	valores[9] ... valores[13]	Valores de los sensores ultrasonidos

Figura A.5: Tabla de los campos del mensaje correspondiente al tópico referente a la información de los sensores.

A.3 Presupuesto

Una de las principales motivaciones ha sido crear una herramienta accesible, debido a la poca inversión recibida por el sector de bomberos, según la encuesta realizada en 1.1.1. En esta sección se presenta el presupuesto necesario para la creación de la plataforma (A.6).

Componente	Precio/Ud	Unidades	Distribuidor	Precio Total
Chasis	63,53 €	1	Mouser	63,53 €
Motor		2		
Interruptor ON/OFF	2,50 €	1	Amitronica	2,50 €
Power bank	24,74 €	1	RS	24,74 €
Pilas recargables	5,37 €	4	Amitronica	21,48 €
Controlador de motores	14,44 €	1	Mouser	14,44 €
Placa Arduino One R3	24,72 €	2	Mouser	49,44 €
Raspberry Pi 4	97,27 €	1	Mouser	194,54 €
Sensor de temperatura y humedad	13,51 €	1	Mouser	13,51 €
Sensor HVOC	24,40 €	1	Mouser	24,40 €
Sensor Ultrasonidos	10,93 €	6	Mouser	65,58 €
Cables	8,18 €	1	Mouser	8,18 €
Soporte para sensores	4,20 €	6	URJC	25,20 €
Separadores	5,51 €	1	Mouser	5,51 €
Soporte para pilas	6,16 €	1	Mouser	6,19 €
Tarjeta SD	7,05 €	1	Mouser	7,05 €
Adaptador de tarjeta SD	6,34 €	1	Mouser	6,34 €
Sensor de Energía	10,89 €	1	Mouser	10,89 €
Pozyx-Creator Kit Lite	847,00 €	1	Pozyx	847,00 €
Placas de metal	7,50 €	2	Mouser	15,00 €
Resistencias	3,01 €	2	Mouser	6,02 €
Cámara	27,99 €	1	Amazon	27,99 €
Total				**1.439,53 €**

Figura A.6: Tabla de costes de la plataforma.

Este precio, en comparación con otros agentes, el modelo resultante entraría en la categoría de bajo coste. Los robots mencionados en el estado del arte tienen un precio de hasta 225.000$.

A.4 Lanzamiento del paquete de navegación.

Tras exponer en la sección 4.2.2 como se ha realizado el diseño y la implementación del sistema de navegación autónoma. En este apartado se describe de forma detallada como poner en funcionamiento la herramienta.

Como primera instancia, se deberá disponer de un ordenador con un entorno Ubuntu 18.04 y deberá tener conexión a Internet. El siguiente paso es abrir una terminal en Ubuntu y establecer conexión por SSH con la Raspberry Pi (figura A.7).

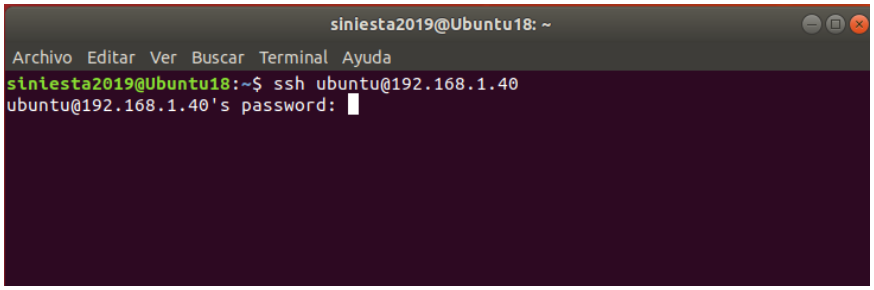


Figura A.7: Puesta en marcha del sistema de navegación autónoma: Paso 1.

Mediante el comando “ifconfig” se obtiene la dirección IP del ordenador, esta dirección deberá incluirse en el archivo “/etc/hosts” de la Raspberry Pi para poder compartir información entre los dispositivos.

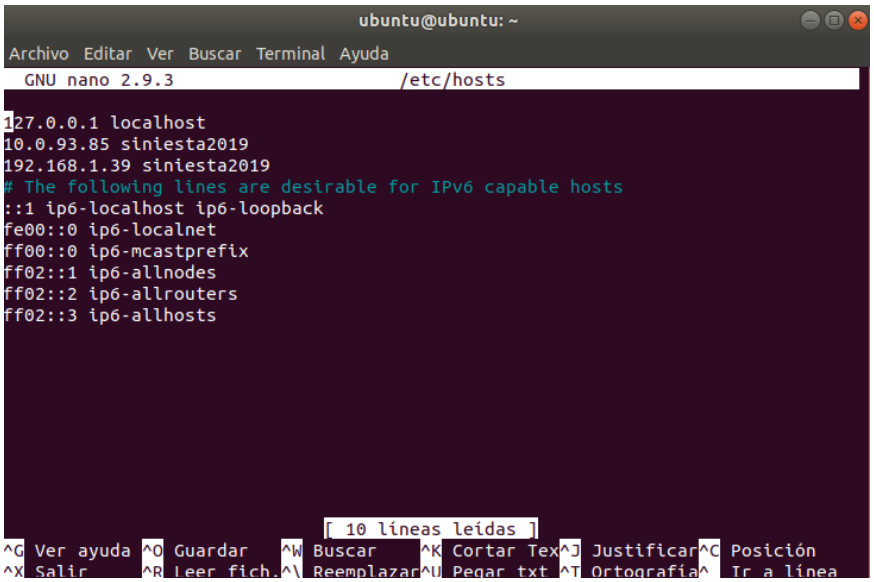


Figura A.8: Puesta en marcha del sistema de navegación autónoma: Paso 2.

Ahora, es necesario indicar al ordenador la dirección IP del dispositivo que lanzará "roscore", y exportar la propia dirección IP del dispositivo en el que se va a utilizar ROS. Este segundo paso (figura A.9), se debe hacer tanto en la Raspberry Pi como en el ordenador.

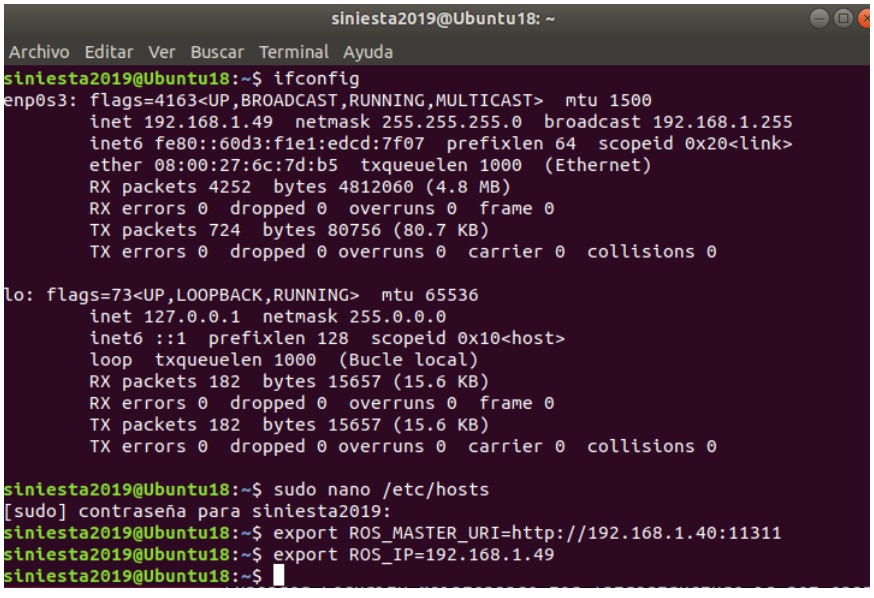


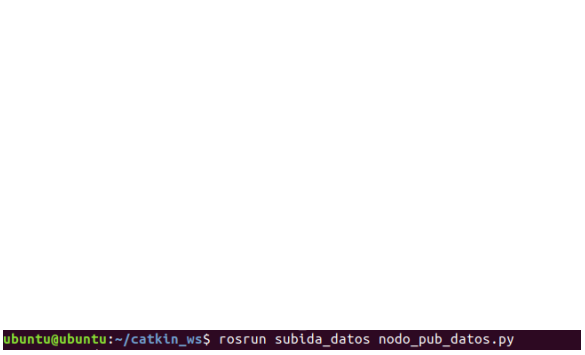
Figura A.9: Puesta en marcha del sistema de navegación autónoma: Paso 3.

Una vez se ha configurado la red de ROS, se procede a lanzar el roscore en la Raspberry Pi mediante el comando "roscore" en una nueva terminal. A continuación, se accede a la carpeta de trabajo de ROS y se compila el directorio "catkin_make", como se puede ver en la ilustración A.10.

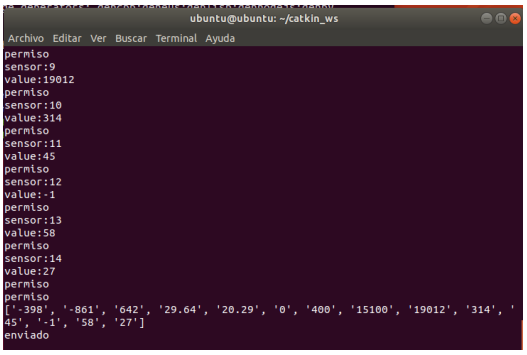
```
siniesta2019@Ubuntu18:~$ cd catkin_ws
siniesta2019@Ubuntu18:~/catkin_ws$ catkin_make
Base path: /home/siniesta2019/catkin_ws
Source space: /home/siniesta2019/catkin_ws/src
Build space: /home/siniesta2019/catkin_ws/build
Devel space: /home/siniesta2019/catkin_ws/devel
Install space: /home/siniesta2019/catkin_ws/install
####
#### Running command: "make cmake_check_build_system" in "/home/siniesta2019/cat
kin_ws/build"
####
####
#### Running command: "make -j4 -l4" in "/home/siniesta2019/catkin_ws/build"
####
[ 50%] Built target tf_broadcaster
[100%] Built target tf_listener
siniesta2019@Ubuntu18:~/catkin_ws$
```

Figura A.10: Puesta en marcha del sistema de navegación autónoma: Paso 4.

Una vez realizado este último paso en los dos dispositivos, se pueden ejecutar los programas de ROS que se precisen. Para el paquete de navegación es necesario que el nodo nodo_pub_datos.py envíe los valores de posición y las medidas de los sensores ultrasonidos, el primer paso se presenta en la figura A.11a. Para comprobar que los datos se están enviando correctamente por el tópico valores, el código imprime por pantalla la cadena de datos que se publica por el tópico, como se puede ver en la figura A.11b.



(a) Lanzar el nodo que publica los datos por el tópico "valores".



(b) Datos mostrados por pantalla en tiempo real.

Figura A.11: Puesta en marcha del sistema de navegación autónoma: Paso 5.

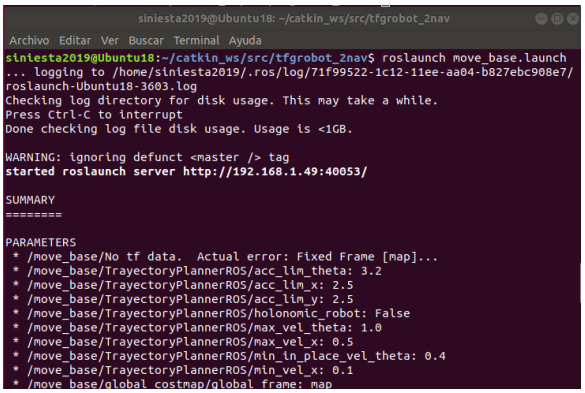
El siguiente paso consiste en el propio lanzamiento del paquete de navegación, para ello será necesario abrir dos terminales en el ordenador. En ambas, se accede al directorio en el que se ubican los archivos de lanzamiento.

```
siniesta2019@Ubuntu18:~/catkin_ws$ cd src/tfgrobot_2nav
siniesta2019@Ubuntu18:~/catkin_ws/src/tfgrobot_2nav$
```

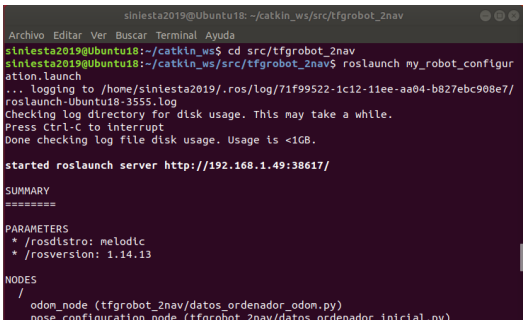
Figura A.12: Puesta en marcha del sistema de navegación autónoma: Paso 6.

En la figura A.12, se presenta el acceso al directorio "tfgrobot_2nav".

Ahora debemos lanzar los dos archivos que inician el sistema de navegación autónoma: "my_robot_configuration.launch" y "move_base.launch". Se ejecutará uno en cada terminal mediante el comando "roslaunch".



(a) Lanzamiento de move_base.launch



(b) Lanzamiento de my_robot_configuration.launch.

Figura A.13: Puesta en marcha del sistema de navegación autónoma: Paso 7.

Una vez lanzado el paquete de navegación autónoma, es posible supervisar su comportamiento con las herramientas “Rqt” y “RViz”.

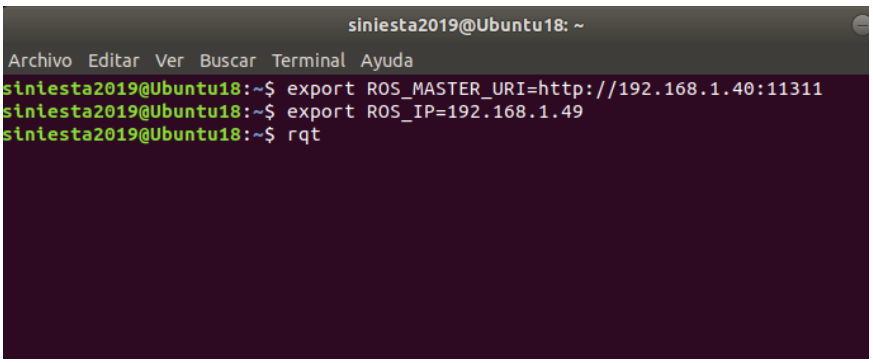


Figura A.14: Puesta en marcha del sistema de navegación autónoma: Paso 8.

A.5 Funciones implementadas en el módulo de visión artificial

En esta sección del apéndice se van a describir las funciones que participan en el código “yolo_opencv_webcam.py”. Este código rige el comportamiento de la red neuronal.

La primera función es:

```
model = cv2.dnn.readNetFromDarknet('yolov3.cfg', 'yolov3.weights')
```

Esta función carga un modelo preentrenado en formato Darknet en OpenCV utilizando la función readNetFromDarknet(). El modelo preentrenado utilizado en este caso es YOLOv3 (You Only Look Once). La configuración de la red se especifica en el archivo “yolov3.cfg” y los pesos del modelo se almacenan en el archivo “yolov3.weights”. Devuelve un objeto “model” que se utilizará posteriormente para realizar la detección de objetos.

Otra de las funciones recogidas en este código es la siguiente:

```
cap = cv2.VideoCapture(0)
```

Esta función se utiliza para crear un objeto de captura de vídeo en OpenCV, que representa la conexión con un dispositivo de captura de vídeo, en este caso, la webcam utilizada.

En el código mencionado, "cap" es el nombre del objeto de captura de vídeo que se crea utilizando cv2.VideoCapture(0). El parámetro 0 especifica el índice del dispositivo de captura de vídeo que se va a utilizar. En este caso, se utiliza 0 para indicar el primer dispositivo de captura de vídeo disponible.

El siguiente paso es proceder a la captura de frames del vídeo grabado, para ello, se inicia un bucle que empieza con la siguiente función:

```
ret, frame = cap.read()
```

Este comando se utiliza en OpenCV para leer un fotograma del vídeo capturado por la cámara webcam.

En el código mencionado, "cap" es el objeto de captura de vídeo creado previamente utilizando cv2.VideoCapture(0). La función cap.read() se utiliza para leer el siguiente fotograma del vídeo. La función devuelve dos valores:

- ret: Un valor booleano que indica si la lectura del fotograma fue exitosa o no. Si se pudo leer el fotograma correctamente, ret será True. En caso contrario, será False.
- frame: El fotograma de vídeo leído, representado como una matriz NumPy. Este fotograma se puede procesar y mostrar posteriormente.

La variable ret determina cuando acaba el bucle. Aunque, al tratarse de una grabación de cámara, la detección no parará hasta que de forma manual decidas parar la grabación.

El siguiente paso es preparar la imagen para introducirla en la red neuronal.

```
blob = cv2.dnn.blobFromImage(frame, 1/255, (416, 416), swapRB=True, crop=False)
model.setInput(blob)
```

La función cv2.dnn.blobFromImage() se utiliza para generar un "blob" de imagen a partir del fotograma capturado. Un blob es una representación especial de una imagen para el procesamiento en una red neuronal.

Los parámetros utilizados en cv2.dnn.blobFromImage() son:

- frame: El frame de vídeo capturado por la cámara web.
- 1/255: Factor de escala para normalizar los valores de píxeles de la imagen. Dividir por 255 normaliza los valores de píxeles en el rango de 0 a 1.
- (416, 416): Tamaño de la imagen de entrada esperado por la red neuronal. El tamaño de (416, 416) es comúnmente utilizado por el modelo YOLOv3.
- swapRB=True: Indica si se debe intercambiar los canales de color de la imagen de BGR (azul-verde-rojo) a RGB (rojo-verde-azul). En este caso, se establece en True para realizar el intercambio.
- crop=False: Indica si se debe recortar la imagen para que se ajuste al tamaño especificado. En este caso, se establece en False para mantener la relación de aspecto original de la imagen.

Una vez se ha preparado la imagen, es el momento de hacerla pasar por el modelo utilizado.

```
output_layers = model.getUnconnectedOutLayersNames()
outputs = model.forward(output_layers)
```

En las líneas de código mencionadas, se obtienen las salidas de las capas no conectadas del modelo de red neuronal YOLOv3.

La función model.getUnconnectedOutLayersNames() se utiliza para obtener los nombres de las capas no conectadas del modelo. En el caso de YOLOv3, estas capas contienen la información de detección de objetos.

La función model.forward(output_layers) realiza el paso hacia adelante (forward pass) a través del modelo utilizando las capas no conectadas especificadas en output_layers. Esto calcula las salidas de estas capas y devuelve los resultados en forma de una lista de tensores.

Los resultados se almacenan en la variable outputs. Cada tensor en la lista outputs contiene las detecciones de objetos para una de las capas no conectadas.

Estos resultados se utilizan posteriormente para realizar la detección de objetos y el dibujo de las cajas delimitadoras en el frame de vídeo.

Ahora, se da el paso principal del código, la detección de objetos y dibujo del marco.

```
for output in outputs:
    for detection in output:
        scores = detection[5:]
        class_id = np.argmax(scores)
        fiabilidad = scores[class_id]
        if fiabilidad > 0.5:
            centro_x = int(detection[0] * frame.shape[1])
            centro_y = int(detection[1] * frame.shape[0])
            width = int(detection[2] * frame.shape[1])
            height = int(detection[3] * frame.shape[0])
            x = int(centro_x - width/2)
            y = int(centro_y - height/2)
            cv2.rectangle(frame, (x, y), (x + width, y + height), (0, 255, 0), 2)
            etiqueta = f'{clases[class_id]}: {fiabilidad:.2f}'
            cv2.putText(frame, etiqueta, (x, y - 10), cv2.FONT_HERSHEY_SIMPLEX,
            0.5, (0, 255, 0), 2)
```

Se itera sobre cada una de las salidas (output) de la lista outputs. Para cada detección en la salida actual (detection), se extraen la puntuación de confianza (scores). Estas puntuaciones representan la confianza de que un objeto pertenezca a cada clase posible. Utilizando np.argmax(scores), se encuentra el índice de la puntuación más alta, que corresponde a la clase con mayor confianza y se guarda en la variable fiabilidad.

Se verifica si la fiabilidad es mayor que 0.5 para filtrar las detecciones menos confiables. Se calculan las coordenadas de la caja delimitadora utilizando las dimensiones del fotograma (frame.shape), el centro de la detección y su ancho y alto normalizados.

Se dibuja un rectángulo en el fotograma utilizando cv2.rectangle(), especificando las coordenadas de la esquina superior izquierda y la esquina inferior derecha de la caja, así como el color y el grosor del borde. Se agrega un texto en la parte superior de la caja utilizando cv2.putText(), mostrando la clase predicha y la confianza correspondiente.

Este proceso se repite para cada detección en todas las salidas del modelo.

Finalmente mediante la función:

```
cv2.imshow('Webcam', frame)
```

La función cv2.imshow('Webcam', frame) se utiliza para mostrar el frame de vídeo procesado en una ventana con el título "Webcam".

A.6 Github

La recopilación de todos los códigos que conforman este Trabajo de Fin de Grado, se encuentra en un repositorio de GitHub accesible a tarvés del siguiente enlace:

https://github.com/silviainiesta/TFG-Silvia-Iniesta-Tejero/tree/main

Está organizado en un sistema de carpetas. La carpeta de control manual incluye los códigos que rigen la navegación manual de la plataforma.

En segundo lugar, la carpeta de Monitorización de parámetros incluye los códigos que permiten recoger y enviar los datos de los sensores y de las balizas.

En tercer lugar, la carpeta Navegación Autónoma presenta los códigos necesarios para poner en marcha el robot en modo autónomo.

Y por último, se incluyen los códigos Arduino del agente autónomo.

Apéndice B

Cronograma del proceso de trabajo.

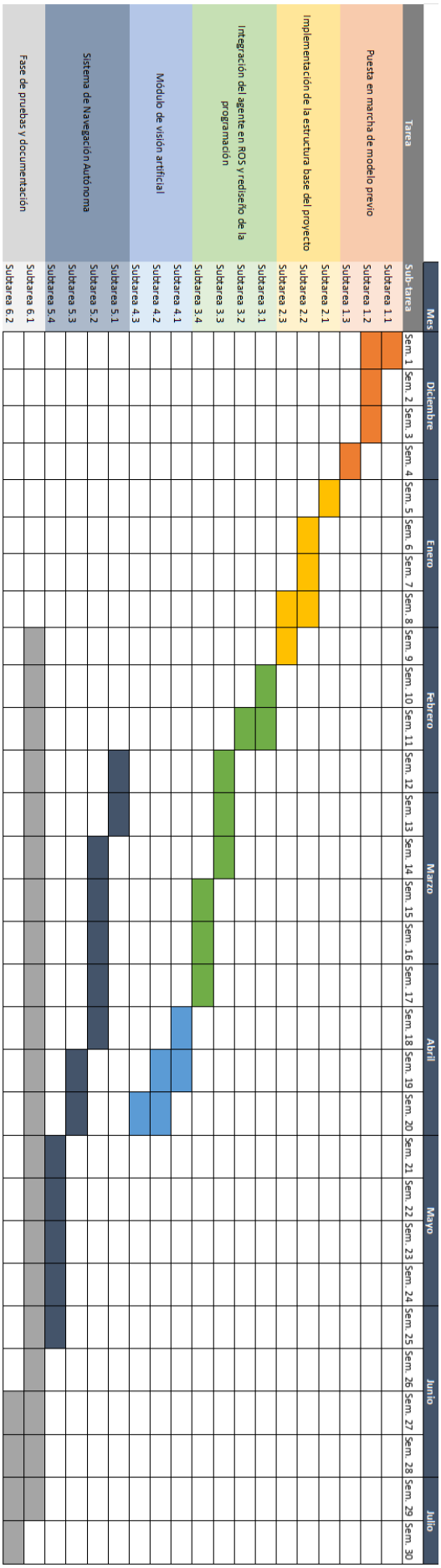


Figura B.1: Diagrama de GANT de las fases del proyecto.

Universidad
Rey Juan Carlos