



GRADO EN INGENIERÍA DE ROBÓTICA SOFTWARE

Escuela de Ingeniería de Fuenlabrada

Curso académico 2023-2024

Trabajo Fin de Grado

Programación de flujo de datos en
multirobótica con ROS2 y Zenoh-Flow

Tutor: Julio Vega Pérez

Autor: Unai Sanz Conejo



Este trabajo se distribuye bajo los términos de la licencia internacional CC BY-NC-SA International License (Creative Commons AttributionNonCommercial-ShareAlike 4.0). Usted es libre de *(a) compartir*: copiar y redistribuir el material en cualquier medio o formato; y *(b) adaptar*: remezclar, transformar y crear a partir del material. El licenciador no puede revocar estas libertades mientras cumpla con los términos de la licencia:

- *Atribución.* Usted debe dar crédito de manera adecuada, brindar un enlace a la licencia, e indicar si se han realizado cambios. Puede hacerlo en cualquier forma razonable, pero no de forma tal que sugiera que usted o su uso tienen el apoyo de la licenciatante.
- *No comercial.* Usted no puede hacer uso del material con propósitos comerciales.
- *Compartir igual.* Si remezcla, transforma o crea a partir del material, debe distribuir su contribución bajo la misma licencia del original.

Agradecimientos

Quisiera expresar mi más sincero agradecimiento a todas las personas que contribuyeron a la realización de este trabajo. En primer lugar, agradezco al equipo de ZettaScale por darme la oportunidad de realizar mis prácticas con ellos y de aprender incontables aspectos acerca de las telecomunicaciones y por su persistente ayuda. También agradezco a mi tutor de TFG por su orientación constante y su gran paciencia a lo largo de este proceso.

Además estoy profundamente agradecido a mis compañeros de clase por sus ideas y debates constructivos, que han enriquecido enormemente mi investigación.

No puedo dejar de agradecer a mis padres, hermana, amigos cercanos y pareja por sus ideas, consejos y apoyo moral e incondicional.

Por último, pero no menos importante, quiero expresar mi gratitud a todas las fuentes y recursos que consulté durante la elaboración de este trabajo, así como a cualquier institución o persona que haya contribuido de alguna manera, aunque indirecta, a este proyecto.

Sin el apoyo de todas estas personas y entidades, este trabajo no habría sido posible. Gracias de todo corazón.

*A mi abuelo;
que estaría sumamente orgulloso de mi trabajo.*

Madrid, xx de xxxxxxx de 2024

Unai Sanz

Resumen

Escribe aquí el resumen del trabajo. Un primer párrafo para dar contexto sobre la temática que rodea al trabajo.

Un segundo párrafo concretando el contexto del problema abordado.

En el tercer párrafo, comenta cómo has resuelto la problemática descrita en el anterior párrafo.

Por último, en este cuarto párrafo, describe cómo han ido los experimentos.

Acrónimos

API *Application Programming Interface*

DDS *Data Distribution Service*

GNU *GNU's Not Unix (acrónimo recursivo)*

HSV *Hue, Saturation, Value ('espacio de color')*

IMU *Inertial Measurement Unit*

JAXA *Japan Aerospace Exploration Agency*

LED *Light Emitting Diode*

LIDAR *Light Detection and Ranging / Laser Imaging Detection and Ranging*

LTS *Long-Term Support*

NASA *National Aeronautics and Space Administration*

OOP *Object Oriented Programming*

OSRF *Open Source Robotics Foundation (Open Robotics)*

PDCA *Plan, Do, Check, Act ('ciclo de desarrollo')*

RAM *Random Access Memory*

RGBD *Red, Green, Blue, Depth ('color y profundidad')*

RMW *ROS Middleware Interface*

ROS *Robotic Operating System*

SDF *Simulation Description Format*

STEM *Ciencia, Tecnología, Ingeniería y Matemáticas*

UPC *Universidad Politécnica de Cataluña*

URDF *Unified Robotics Description Format*

URJC *Universidad Rey Juan Carlos*

XML *eXtensible Markup Language*

Índice general

| | |
|---|-----------|
| 1. Introducción | 1 |
| 1.1. La robótica | 1 |
| 1.2. La robótica en la ciencia | 1 |
| 1.3. La robótica móvil | 3 |
| 1.4. La robótica educativa | 4 |
| 1.5. La robótica de bajo coste | 7 |
| 1.6. La robótica colaborativa | 9 |
| 1.7. Flujos de datos en robótica | 10 |
| 2. Objetivos | 13 |
| 2.1. Descripción del problema | 13 |
| 2.1.1. El problema de la congestión de red | 13 |
| 2.1.2. El problema del escalón de aprendizaje en robótica | 14 |
| 2.1.3. Planteamiento de solución | 14 |
| 2.2. Requisitos | 15 |
| 2.3. Competencias | 15 |
| 2.4. Metodología | 16 |
| 2.5. Plan de trabajo | 17 |
| 3. Plataforma de desarrollo | 19 |
| 3.1. Hardware | 19 |
| 3.1.1. Turtlebot 2 | 19 |
| 3.1.2. Turtlebot 4 | 21 |
| 3.1.3. Ordenadores de a bordo | 22 |
| 3.1.4. Ordenador principal | 23 |
| 3.1.5. Router | 23 |
| 3.2. Software | 23 |
| 3.2.1. Sistema Operativo | 24 |
| 3.2.2. Lenguajes de programación | 24 |

| | |
|---|-----------|
| 3.2.3. Middleware Robótico | 25 |
| 3.2.4. Simulación | 25 |
| 3.2.5. Protocolos de comunicación | 26 |
| 3.2.6. Visión Artificial | 28 |
| 3.2.7. Software de navegación | 29 |
| 3.2.8. Software matemático | 29 |
| 3.2.9. Visualización de datos | 29 |
| 4. Arquitectura Software con Zenoh y ROS | 31 |
| 4.1. Snippets | 31 |
| 4.2. Verbatim | 32 |
| 4.3. Ecuaciones | 32 |
| 4.4. Tablas o cuadros | 33 |
| 5. Experimentos | 34 |
| 5.1. Snippets | 34 |
| 5.2. Verbatim | 34 |
| 5.3. Ecuaciones | 35 |
| 5.4. Tablas o cuadros | 35 |
| Bibliografía | 37 |

Índice de figuras

| | | |
|-------|---|----|
| 1.1. | Rover Perseverance y helicóptero Ingenuity de la NASA en Marte. | 2 |
| 1.2. | Robots Spot (izq.) y Atlas (dcha.). | 4 |
| 1.3. | Arduino. | 6 |
| 1.4. | Código de Arduino en Scratch. | 6 |
| 1.5. | Arquitectura de ROS y ROS2. | 7 |
| 1.6. | Sora-Q, robot lunar <i>low-cost</i> | 8 |
| 1.7. | Múltiples robots durante una operación de búsqueda y rescate. | 10 |
| 1.8. | Robots educativos Turtlebot 2 (arriba) y Turtlebot 4 (abajo). | 11 |
| 1.9. | Flujo de datos de un robot para la detección y seguimiento de códigos QR. | 12 |
| 1.10. | Comparación de un flujo de datos con los componentes de un robot. . . | 12 |
| 2.1. | Esquema del desarrollo software iterativo (fuente propia) | 17 |
| 3.1. | Modelos del robot Turtlebot (OSRF) | 20 |
| 3.2. | Base Kobuki (ROS) | 20 |
| 3.3. | Turtlebot 2 (OSRF) | 21 |
| 3.4. | Turtlebot 4 Lite –izquierda– y Standard –derecha– (OSRF) | 22 |
| 3.5. | HP ProBook 450 G6 (You) | 22 |
| 3.6. | Raspberry Pi 4, modelo B (Team) | 23 |
| 3.7. | Router ASUS, modelo ROG Rapture GT-AXE16000 (Ludlow) | 24 |
| 3.8. | Gazebo simulando un robot Turtlebot 3 en un mundo virtual (Yuhong) | 26 |
| 3.9. | Rendimiento de distintos protocolos en múltiples máquinas (Eclipse Foundation). | 27 |
| 3.10. | Espacios de color HSV (BuckyBall) | 28 |
| 3.11. | Representación de imagen RGB como matriz (Wiki300) | 29 |
| 3.12. | <i>Sliders</i> de un filtro de color con OpenCV (fuente propia). | 30 |

Listado de códigos

| | |
|--|----|
| 3.1. Función para calcular transformadas | 30 |
| 4.1. Función para buscar elementos 3D en la imagen | 31 |
| 4.2. Cómo usar un Slider | 32 |
| 5.1. Función para buscar elementos 3D en la imagen | 34 |
| 5.2. Cómo usar un Slider | 35 |

Listado de ecuaciones

| | |
|---|----|
| 4.1. Ejemplo de ecuación con fracciones | 32 |
| 4.2. Ejemplo de ecuación con array y letras y símbolos especiales | 32 |
| 5.1. Ejemplo de ecuación con fracciones | 35 |
| 5.2. Ejemplo de ecuación con array y letras y símbolos especiales | 35 |

Índice de cuadros

| | |
|--|----|
| 4.1. Parámetros intrínsecos de la cámara | 33 |
| 5.1. Parámetros intrínsecos de la cámara | 36 |

Capítulo 1

Introducción

El éxito es la suma de pequeños esfuerzos repetidos día tras día.

Robert Collier

1.1. La robótica

La robótica es un campo multidisciplinario que se concentra en el diseño, construcción, programación y operación de robots. Estos dispositivos electromecánicos, con frecuencia modelados antropomórficamente o zoomórficamente, están destinados a realizar tareas de manera autónoma o semiautónoma en una variedad de entornos, para lo que disponen de sensores que les proveen de información del medio que les rodea, de cierta capacidad de cómputo para tomar decisiones acerca de estos datos recolectados y de actuadores que les permiten interactuar con el mismo y llevar a cabo dichas decisiones, por lo que sus capacidades y limitaciones están determinadas por su *hardware*, y su inteligencia reside en su *software*.

Este gran campo de estudio e investigación ha experimentado un rápido crecimiento y expansión desde sus inicios en la década de 1950, abarcando una amplia gama de aplicaciones en la industria, la medicina, el entretenimiento y la exploración espacial, entre muchas otras, y se encuentra en constante evolución, desempeñando un papel cada vez más importante en nuestra sociedad moderna, gracias en gran medida a los incesantes avances en tecnologías como la inteligencia artificial, los sensores, los procesadores y los actuadores.

1.2. La robótica en la ciencia

La robótica ha conformado un factor crucial en la exploración espacial, y esta ha impulsado la creación de nuevas tecnologías que han sido desarrolladas exclusivamente

para ciertas misiones espaciales pero que luego se han aplicado en la Tierra, mejorando la calidad de vida de las personas. Ejemplos destacados incluyen los sistemas de purificación de agua, los tejidos avanzados como la viscoelástica, los pañales y los dispositivos de imágenes médicas como la resonancia magnética, que han proporcionado acceso a las personas respectivamente a agua potable en regiones remotas, a colchones y almohadas que promueven un mejor descanso, a mayores facilidades en cuanto al cuidado de los niños y mayores y a diagnósticos médicos precisos sin radiación nociva, lo cual ha salvado innumerables vidas. De esta manera, la investigación espacial no solo expande nuestro conocimiento del universo, sino que también beneficia directamente a la humanidad en la Tierra.

En cuanto al papel de la robótica en este campo, podemos destacar ejemplos como los resistentes robots enviados a diferentes planetas y lunas del sistema solar en busca de datos científicos. Numerosas misiones científicas lo evidencian, como la misión *Mars 2020* de la NASA¹, cuyos robots se ven ilustrados en la Figura 1.1, tomada por uno de los robots, el rover Perseverance, que depositó con éxito al segundo robot sobre la superficie marciana, el helicóptero Ingenuity, que aparece más al fondo en la imagen, y que ayudó al rover a llevar a cabo su exploración y toma de muestras. Este helicóptero estaba diseñado para realizar cinco vuelos durante un mes a modo de demostrador tecnológico, pero su misión pudo alargarse hasta casi los tres años, momento en el que termina debido a la rotura de una de sus palas, sumando entonces un total de 72 vuelos.



Figura 1.1: Rover Perseverance y helicóptero Ingenuity de la NASA en Marte.

Para el éxito de esta misión fue clave la investigación en múltiples campos de la robótica, como la robótica móvil y todos los campos que esta conlleva, como pueden

¹<https://www.nasa.gov/>

ser la visión artificial, la navegación o la localización; áreas clave que no solo están redefiniendo los límites de la tecnología, sino que también tienen un gran impacto en cómo usamos la tecnología en nuestra vida diaria, como ha sucedido, por ejemplo, con las aspiradoras robóticas o la conducción autónoma, que inevitablemente ya forman parte de nuestra sociedad.

1.3. La robótica móvil

La robótica móvil ha emergido como un campo multidisciplinario que fusiona la ingeniería, la inteligencia artificial y múltiples ramas de la robótica y la mecatrónica para crear sistemas capaces de moverse y operar en entornos dinámicos, aprovechando áreas como la robótica de campo, la creación de mapas, la localización y la navegación con ayuda de otros como la visión artificial o la manipulación de objetos.

Desde sus inicios, ha sido impulsada por los avances tecnológicos, permitiendo su aplicación en una amplia gama de campos, desde la exploración espacial y submarina, hasta la logística industrial y la atención médica, siendo ya parte indispensable de nuestras vidas y mejorando la calidad de las mismas. En este contexto, la investigación en robótica móvil se centra en desarrollar sistemas autónomos capaces de navegar de manera segura y eficiente en entornos conocidos o desconocidos, adaptarse a cambios imprevistos y realizar tareas complejas de manera autónoma.

Un ejemplo representativo de este tipo de robots se puede ver en la Figura 1.2, donde se pueden observar dos de los robots más desarrollados en el ámbito móvil, ambos de la empresa Boston Dynamics², que han demostrado una gran versatilidad en una variedad de entornos para ejecutar una amplia variedad de tareas, desde abrir puertas, pasando por transportar cargas de peso o realizar trabajos manuales, hasta incluso seguir rutinas de deportivas variadas, que en muchos casos iguala o incluso supera la de los humanos.

En concreto, la localización de los robots juega un papel importante en la robótica móvil debido a su indispensable necesidad a la hora de navegar por el entorno, normalmente tomando puntos de referencia gracias a los sensores y creando un modelo probabilístico de la posición del robot basado en estos datos.

Por su parte, para el correcto funcionamiento de la navegación, es indispensable conocer la posición del robot durante el movimiento del mismo, para tener la capacidad de sortear obstáculos y poder desplazar el robot al objetivo.

²<https://bostondynamics.com/>



Figura 1.2: Robots Spot (izq.) y Atlas (dcha.).

La robótica móvil también forma parte de campos más grandes y complejos e incluso sienta las bases de algunos de ellos, como sucede con la multirobótica, que representa un paso adelante en la complejidad y la escala de los sistemas robóticos individuales, y permite realizar tareas que un solo robot no es capaz de hacer, o realizarlas mucho mas rápidamente o eficientemente. Como ejemplo de esta ventaja, son notables ciertos trabajos realizados sobre localización con múltiples robots, como en el artículo (16), en el que se ponen a prueba las mismas técnicas utilizadas para un solo robot y evalúan su viabilidad en un entorno unidimensional.

También se han realizado trabajos enfocados a entornos tridimensionales, como se observa en el artículo (8), en el cuál se logra una localización basada en sensores, teniendo en cuenta la posición de los demás robots y aumentando de este modo la precisión de la localización del propio robot en cuestión.

Además, este campo supone una gran oportunidad para el proceso de enseñanza-aprendizaje, como se verá en la Sección 1.4, ya que juega un papel crucial en el desarrollo de habilidades tecnológicas a la vez que en la motivación de los estudiantes, los cuales obtienen una gran sensación de realización y entusiasmo por aprender, al poder visualizar los resultados en movimiento de manera autónoma.

1.4. La robótica educativa

Debido a la creciente participación de los robots móviles en nuestras vidas diarias, como se ha hecho notar en el caso de las aspiradoras robóticas, la relevancia de la robótica en el ámbito educativo ha ido ganando terreno en los últimos años, tanto en la comunidad europea como en otros muchos países. Esto ha dado lugar a un aumento en la implementación de programas educativos las cuales incluyen actividades prácticas de

robótica en escuelas de educación primaria y secundaria, promoviendo la creatividad, el pensamiento crítico y las habilidades tecnológicas entre los estudiantes, además de fomentar el trabajo en equipo y la resolución de problemas complejos, preparándolos de este modo para futuros grados o carreras relacionadas con la tecnología, cuya demandada aumenta año tras año.

En el caso de España, desde la década de los 90, se han implementado programas piloto y competiciones robóticas, como el programa Robolot³ (1992), desarrollado por la UPC, las Olimpiadas de Informática⁴ (1993), que incorporaron desafíos relacionados con la programación de robots, así como la RoboCupJunior⁵ (2000), ofreciendo a los estudiantes la oportunidad de diseñar, construir y programar robots para competir en diferentes categorías, o la competición Robocampeones⁶ (2016), por la URJC y en la que cada año participan más institutos.

Desde alrededor de 2014, dependiendo de la comunidad autónoma de España, se han introducido programas y asignaturas que incluyen la robótica como parte esencial del plan de estudios, en concreto en la Comunidad de Madrid, se implantaron asignaturas relacionadas con la robótica en el itinerario formativo en la educación entre 2014 y 2015. Concretamente en la Educación Primaria en 2014, según el Decreto (2), se añadió la asignatura llamada *Tecnología y recursos digitales para la mejora del aprendizaje*, en cuyo anexo III se describen los contenidos, que incluyen la programación en *Scratch*⁷. Más tarde, en 2015, conforme al (1), se añadió una asignatura correspondiente al periodo de Educación Secundaria con el nombre *Tecnología, Programación y Robótica*, en cuyo anexo III se establecen los contenidos, también basados en *Scratch*, *Arduino*⁸ y la impresión 3D. Toda esta información se puede ver resumida en el apartado 2.12 del artículo publicado por el (10).

Al ser necesario un contexto simple y barato para la introducción de la robótica, en educación se buscan herramientas como las placas *Arduino* (Figura 1.3), que presentan una amplia compatibilidad con distintos sensores y actuadores y brinda un entorno sencillo para aquellos que se están introduciendo en este campo y plataformas como *Scratch* para simplificar la programación gracias a su interfaz de bloques y la asociación de ideas a colores, como se muestra en la Figura 1.4.

Las placas utilizadas en el ámbito educativo, mencionadas anteriormente, resultan

³<https://www.roboLOT.online/>

⁴<https://olimpiada-informatica.org/>

⁵<https://junior.robocup.org/>

⁶<https://sites.google.com/view/robocampeonesfuenlabrada/>

⁷<https://scratch.mit.edu/>

⁸<https://www.arduino.cc/>



Figura 1.3: Arduino.



Figura 1.4: Código de Arduino en Scratch.

ideales para estos propósitos debido a su coste y simplicidad, sin embargo, también imponen ciertas limitaciones en las capacidades del robot y en la posibilidad de añadir *hardware* externo más complejo y potente, como cámaras o LIDARs, siendo estos últimos sensores dedicados a la medición de distancias la emisión y recepción de un láser, o actuadores como motores más potentes que a menudo requieren de mayor alimentación eléctrica de la que estas placas pueden brindar. Esto genera trabas a la propia originalidad y aprendizaje de los estudiantes, restringiendo así su creatividad, innovación y potencial de creación, una vez se han obtenido unos conocimientos básicos.

Es entonces cuando se encuentra el desafío de dar el siguiente paso: la programación de robots utilizando ROS⁹, cuyas siglas significan, en inglés, sistema operativo de robots, pero que nada tiene que ver con ningún sistema operativo, siendo por el contrario el *middleware* estándar por excelencia en robótica, encargado de abstraer del hardware al programador, y permitiéndolo, por tanto, un desarrollo más ameno y compatible con una amplia gama de robots, cuya arquitectura interna puede verse apreciada en la Figura 1.5. Este proceso implica un considerable escalón de aprendizaje, ya que no solo se debe dominar un lenguaje de programación más complejo, sino que también se debe comprender el entorno que rodea a esta plataforma, en el que se incluyen campos de la robótica como son las comunicaciones, la arquitectura *software*, la programación modular y orientada a objetos, algoritmos y estructuras de datos, entre otros muchos, y que suelen conllevar decenas de asignaturas con identidad propia en cualquier grado de universidad. La teoría de la existencia de una brecha educativa

⁹<https://www.ros.org/>

en este ámbito se ve respaldada por trabajos como la tesis doctoral de (17), en cuyas secciones A.3 y A.4, se analiza esta misma perspectiva y se propone una solución respectivamente.

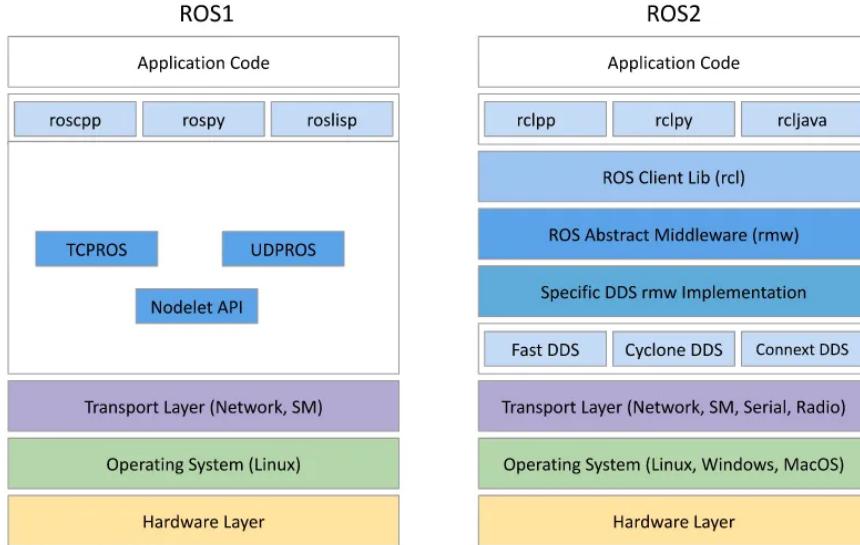


Figura 1.5: Arquitectura de ROS y ROS2.

Por este motivo, resulta evidente la necesidad de un paso intermedio que pueda actuar como puente entre estos dos niveles de aprendizaje, el cual podría ser incorporado, por ejemplo, en el programa educativo de la etapa de Educación Secundaria o Bachillerato. Este nivel intermedio facilitaría la transición entre esta etapa educativa y la universitaria, concretamente dentro del ámbito científico-tecnológico.

1.5. La robótica de bajo coste

La robótica de bajo coste se refiere al desarrollo e implementación de sistemas robóticos, como los descritos anteriormente, utilizando componentes y recursos económicos, con el objetivo de hacer la tecnología robótica más accesible y asequible para una amplia gama de aplicaciones y usuarios. Este enfoque busca reducir los costes asociados con la construcción y operación de robots, empleando materiales económicos, hardware de bajo coste y técnicas de fabricación eficientes.

En el contexto de la robótica móvil, los sistemas de bajo coste pueden ofrecer soluciones viables para aplicaciones con presupuestos limitados o despliegues a gran escala, abarcando un papel crucial en áreas como la educación, la investigación académica, la asistencia social y la exploración de entornos remotos o peligrosos. Además de su utili-

dad práctica, la robótica de bajo coste también promueve la innovación y el desarrollo de nuevas tecnologías al proporcionar una plataforma accesible para la experimentación y la creatividad abierta a una amplia comunidad.

Un ejemplo destacado de este tipo de robótica es el robot Sora-Q mostrado en la Figura 1.6, enviado a la Luna recientemente por la JAXA¹⁰ y desarrollado por la juguetería japonesa Takara Tomy¹¹, y que tras completar su misión, fue comercializado por 150€, hito que ilustra cómo la tecnología robótica puede volverse accesible para un público más amplio, incluso después de su participación en misiones espaciales.



Figura 1.6: Sora-Q, robot lunar *low-cost*.

De entre las distintas áreas en que se aplica este enfoque, cabe destacar la robótica educativa, que suele basarse en este tipo de sistemas de bajo coste, ya que las instituciones educativas enfrentan limitaciones presupuestarias que dificultan la adquisición de sistemas más costosos, por el presupuesto limitado y por el gran número de alumnos, a los que no podrían proveer de sistemas de este calibre de otra manera; o no al menos de forma individual.

La robótica de bajo coste no solo ha democratizado el acceso a la tecnología robótica, sino que también ha revolucionado la forma en que se enseña la robótica en las escuelas. Este enfoque económico ha permitido a las instituciones educativas superar las limitaciones presupuestarias y proporcionar a un mayor número de estudiantes la oportunidad de involucrarse en actividades prácticas de robótica, allanando el camino para que los estudiantes se sumerjan en áreas más avanzadas de este área, como pueden ser la robótica móvil o campos estrechamente relacionados.

¹⁰<https://global.jaxa.jp/>

¹¹<https://www.takaratomy.co.jp/english/>

1.6. La robótica colaborativa

La multirobótica es un campo de investigación que estudia y desarrolla sistemas robóticos compuestos por múltiples robots que trabajan en conjunto para realizar una variedad tareas complejas, que ha sido ampliamente estudiada en artículos como (18), en los que se relatan todos los aspectos de la misma, poniendo en contexto este novedoso campo.

Estos sistemas pueden dividir sus tareas, como la exploración de entornos desconocidos o la búsqueda y rescate en áreas de difícil acceso, por lo que la colaboración entre ellos es de vital importancia, e implica aspectos como el establecimiento de comunicaciones para compartirse información y entender de un mejor modo el mundo y contexto que les rodea, la creación de mapas del entorno para poder localizarse y navegar por el mismo de manera controlada o la manipulación de objetos, muchas veces necesaria para completar el objetivo propuesto para estos sistemas. Todo ello puede verse descrito en el trabajo de (12), en el que se relatan con más detalle los avances logrados en estos aspectos.

La coordinación entre estos sistemas puede suponer la diferencia entre el éxito o el fracaso de su misión, por lo que también es de suma importancia, y por ello se han realizado múltiples trabajos acerca de este tema. En estos términos, los equipos de robots pueden operar eficientemente asignando roles y responsabilidades como expone el artículo de (3), en el que se desarrolla un sistema de control con la menor centralización posible para estudiar la cooperación multirobot en el proyecto MARTHA¹².

A pesar de intentar crear sistemas con la mayor descentralización posible, el trabajo anterior sigue siendo un sistema centralizado, donde un robot puede asumir roles específicos. La tendencia actual se inclina hacia sistemas directamente o casi totalmente descentralizados, donde la coordinación y la optimización son fundamentales, como se hace notar en el trabajo de (14), en el que todos los robots actualizan su propio mapa local con la información de los demás, sin que ninguno de ellos adquiera un mayor protagonismo o importancia.

Gracias a este tipo de trabajos, se ha conseguido mejorar la eficiencia y la robustez de los sistemas robóticos, y la multirobótica se ha convertido en un campo importante de investigación. Los principios y problemas técnicos en este campo se exploran en diversos contextos, como se ilustra en la Figura 1.7, en la cual, la flota de robots está realizando una tarea de búsqueda y rescate, mediante la división de un área, probablemente

¹²MARTHA: European ESPRIT III Project No 6668 “Multiple Autonomous Robots for Transport and Handling Applications”

desconocida, actualizando sus mapas, como se ha explicado anteriormente.



Figura 1.7: Múltiples robots durante una operación de búsqueda y rescate.

En el ámbito educativo, la multirobótica ofrece una oportunidad única para involucrar a los estudiantes en actividades prácticas y colaborativas. Al trabajar con sistemas multirobot, los estudiantes no solo adquieren conocimientos sobre programación, control y mecánica de robots, sino que también exploran conceptos como son las telecomunicaciones entre robots, la coordinación, la planificación y asignación de tareas con o sin prioridades, la localización y navegación conjunta, como se ha explicado en los trabajos citados en la Sección 1.3, así como la seguridad que se requiere para evitar colisiones entre ellos, adquiriendo a su vez habilidades útiles para el trabajo en equipo. Además, la multirobótica proporciona un entorno de aprendizaje dinámico y estimulante que despierta aún más la curiosidad y la creatividad de los estudiantes, preparándolos para enfrentar los desafíos del mundo tecnológico en constante evolución.

Un ejemplo representativo de los robots educativos, en este caso del Laboratorio de Robótica de la URJC, aparece en la Figura 1.8, en la que se pueden diferenciar hasta dos modelos distintos de robots: Turtlebot 2, en la parte superior de la imagen; y Turtlebot 4, más modernos, en la parte inferior de la misma.

1.7. Flujos de datos en robótica

Las mencionadas telecomunicaciones entre robots son fundamentales en la multirobótica, ya que garantizan una comunicación rápida, óptima, eficiente y ordenada entre los distintos dispositivos, necesaria para un correcto desempeño de la funcionalidad en cuestión. Sin embargo, este proceso puede enfrentarse a desafíos, como la congestión de la red, y la consecuente pérdida de mensajes, que pueden ser críticos para el correcto funcionamiento de los robots. Por este motivo, resulta crucial gestionar



Figura 1.8: Robots educativos Turtlebot 2 (arriba) y Turtlebot 4 (abajo).

cuidadosamente la cantidad de robots y mensajes generados, intentando minimizarlos para optimizar el rendimiento del sistema y evitando de esta manera el problema conocido como *cuello de botella*, siendo este el objetivo principal del estudio de los flujos de datos en Robótica.

Un flujo de datos consiste en un grafo dirigido de los datos que fluyen entre operaciones. Mantener un flujo de datos correcto es fundamental para solventar los problemas de telecomunicaciones mencionados en el desarrollo de sistemas de multirobótica. Además, simplifica el desarrollo del software al proporcionar una clara visión de la dirección, el origen y el destino de los datos en cada momento. Esto permite dividir el programa en partes claramente diferenciadas, normalmente llamadas nodos, modularizándolo y dando lugar a la división del problema último en varios problemas más simples y fáciles de atajar, creando un paradigma que modela el programa como un flujo de datos. Como resultado, el desarrollo se vuelve un proceso más sostenible y escalable, y por tanto, más fácil de llevar a cabo por los estudiantes.

Este proceso se puede ver ilustrado en la Figura 1.9, donde se ilustra el flujo de datos, de manera simplificada, de un robot programado para detectar y seguir códigos QR¹³, en la que se puede observar cómo los datos siguen un esquema de nodos dirigido y, en este caso unidireccional, pasando de su origen en el robot a su procesamiento en otra máquina y acabando en su posterior vuelta al robot en forma de órdenes de movimiento, pudiendo saber en todo momento en qué proceso se encuentran dichos datos.

Este proceso es equiparable a la forma de programación de un robot reactivo, ya que,

¹³https://github.com/USanz/follow_beacon

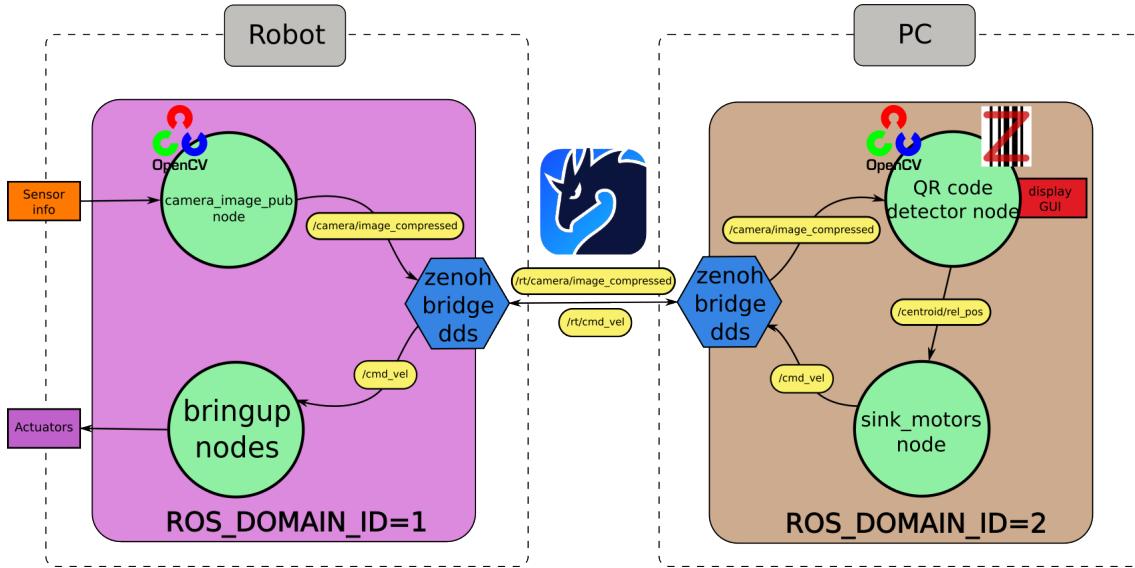


Figura 1.9: Flujo de datos de un robot para la detección y seguimiento de códigos QR.

como se observa en la Figura 1.10, en los flujos de datos existen tres tipos esenciales de nodos: unos en los que se originan los datos, otros donde se computan, y otros donde los datos llegan a su final y que encuentran correspondencia en los nodos encargados de la percepción de un robot, del cómputo de estos datos obtenidos y de los nodos encargados de la actuación del robot, respectivamente¹⁴. Además, este tipo de programación de los robots de manera reactiva es el más simple y el primero que se suele aprender, por lo que genera una sinergia con el área educativa en este ámbito.

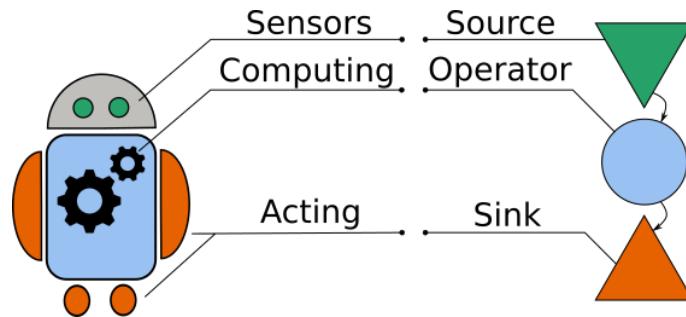


Figura 1.10: Comparación de un flujo de datos con los componentes de un robot.

El próximo capítulo detalla los objetivos planteados y logrados en este trabajo, ofreciendo una perspectiva más completa de su finalidad.

¹⁴[Ponencia de la conferencia ROSCON Madrid de 2023] <https://www.youtube.com/watch?v=ZgFHCvEFU0I>

Capítulo 2

Objetivos

Dame seis horas para talar un árbol y pasaré las primeras cuatro afilando el hacha

Abraham Lincoln

Una vez establecido el marco contextual de este proyecto, se procederá a presentar una descripción del problema a abordar, así como el proceso creativo e intelectual que guiará el desarrollo del mismo, lo que incluirá los requisitos del proyecto, la metodología empleada y el plan de trabajo detallado.

2.1. Descripción del problema

Este proyecto surge como respuesta a la escasa investigación sobre los flujos de datos empleados en conjunto con ROS2, ofreciendo a su vez un entorno propicio para la creación sencilla de distintas aplicaciones robóticas basadas en dichos flujos de datos, que pueden replicar de manera versátil y sencilla los comportamientos reactivos de los robots, eludiendo la complejidad inherente de este *middleware* robótico y dando lugar, por tanto, a un nuevo entorno de programación más simple, a la vez que solucionando varios problemas expuestos a continuación.

2.1.1. El problema de la congestión de red

Con el uso de ROS2, que por defecto funciona sobre el protocolo DDS, existe un problema de congestión de red: los nodos de ROS2 generan una gran cantidad de mensajes de manera iterativa, que a su vez van a través de DDS, que es un protocolo de comunicaciones que también origina muchos de mensajes de *Discovery*, lo que en conjunto conlleva a la congestión de la red, y dificulta de esta manera la programación de aplicaciones multirobot.

Una solución muy sencilla a este problema, consistirá en cambiar el protocolo que lo provoca por otro con mejores prestaciones. En nuestro caso se utilizará un protocolo llamado Zenoh, como ya se expondrá más adelante, en el Capítulo 3.

2.1.2. El problema del escalón de aprendizaje en robótica

La educación en robótica se basa en robots de bajo coste, como se expuso en la Sección 1.4, lo que suele limitar la capacidad del robot en cuestión y merma la cantidad de *hardware* externo compatible, así como su calidad, y consecuentemente limita la creatividad y el aprendizaje de los estudiantes. Además, siendo ROS el estándar en robótica para la programación de robots, se genera un escalón de aprendizaje, previamente expuesto en la sección referenciada, debido al gran cambio desde la programación de placas como Arduino, a la utilización tanto de *hardware*, como de *software* mucho más complejos.

Esto conlleva a una gran diferencia entre la robótica que se enseña en las etapas de la educación primaria y secundaria respecto a la etapa universitaria, y es debido precisamente, a la complejidad de código y enseñanza de ROS, para los cuales, se requiere incluso de varias asignaturas en esta última etapa. Es por este motivo, que este trabajo pretende incorporar un paso intermedio en la enseñanza, simplificando el desarrollo del software en ROS2, y dando la posibilidad de crear aplicaciones robóticas más complejas de una manera más simple para robots más completos, y que a su vez permanecen dentro de la categoría de robots de bajo coste, siendo asequibles para instituciones como colegios o institutos.

2.1.3. Planteamiento de solución

La simplicidad del código de ROS2, se conseguirá gracias al uso de un *framework* llamado Zenoh-Flow, que funciona sobre el protocolo mencionado en la Sección 2.1.1, y el cuál le da nombre. Este *framework* está pensado para la programación de flujos de datos, por lo que se requerirá la previa definición de un flujo de datos a seguir por parte de los nodos, los cuales activarán su iteración al momento de recibir un dato, y generarán en su conjunto, un flujo de datos transmitiendo los datos de nodo a nodo únicamente cuando es necesario, cualidad que reducirá en gran cantidad los mensajes generados.

La implementación conjunta con ROS2 será posible gracias a que Zenoh-Flow permite serializar los datos que se quieren enviar, y existe un *bridge* que los traduce de Zenoh a DDS para que los nodos de ROS2 puedan entender dicha información. Es por este motivo, que se serializarán los mensajes de la misma manera que se hace

internamente en ROS2, y así se podrá seguir utilizando nodos de ROS2 existentes, como pueden ser los relativos a la navegación, lo que evitara la complejidad de su programación.

Este trabajo pretende, por tanto, solucionar tanto el problema del escalón de aprendizaje, suponiendo un paso intermedio en la enseñanza de la robótica, como el problema de la congestión de red generada por DDS, suponiendo una posible solución a la misma.

2.2. Requisitos

Para solucionar los problemas descritos, este trabajo deberá cumplir los siguientes requisitos:

1. Se utilizará *GNU/Linux*, con la distribución *Ubuntu 22.04 LTS* como sistema operativo en todos los *hardwares*.
2. El sistema deberá desarrollar alguna forma de programación de flujos de datos con ROS2.
3. El entorno de programación debe brindar la posibilidad de funcionar en conjunto con nodos de ROS2, permitiendo la comunicación con los mismos mediante *topics*.
4. Los *softwares* utilizados deben ser compatibles para funcionar correctamente en conjunto.
5. Las aplicaciones demostrativas que se desarrolle deben ser fácilmente reproducibles y desplegables tanto en un entorno simulado como en un ambiente educativo real o de laboratorio.
6. El desarrollo del *software* debe ser lo suficientemente sencillo para poder ser llevado a cabo por alumnos preuniversitarios.
7. El *hardware* utilizado debe ser suficientemente económico para ser adquirido por organismos educativos.

2.3. Competencias

Las competencias generales que se cumplen con la realización de este trabajo de fin de grado, según la guía docente de la asignatura son las siguientes:

1. *CB2.* Que los estudiantes sepan aplicar sus conocimientos a su trabajo o vocación de una forma profesional y posean las competencias que suelen demostrarse por medio de la elaboración y defensa de argumentos y la resolución de problemas dentro de su área de estudio. Esta competencia se cumple con la realización de la parte del *software* de este trabajo, en la que se aplican distintos conocimientos adquiridos durante el grado.
2. *CB4.* Que los estudiantes puedan transmitir información, ideas, problemas y soluciones a un público tanto especializado como no especializado. Esta competencia se adquiere al detallar todo el complejo proceso consecuente a este trabajo de manera clara y comprensible en el presente documento.
3. *CB5.* Que los estudiantes hayan desarrollado aquellas habilidades de aprendizaje necesarias para emprender estudios posteriores con un alto grado de autonomía. Esta competencia queda cumplida al adquirir los conocimientos suficientes para el desarrollo de este trabajo de manera completamente autónoma, a base de distintas pruebas y consultas en distintas fuentes: publicaciones científicas, foros de desarrollo en la web, etc.

La competencia específica *CE28* de la asignatura detalla lo siguiente: Desarrollo de las capacidades adecuadas para realizar un ejercicio original individual (o excepcionalmente colectivo), presentarlo y defenderlo ante un tribunal universitario, consistente en un proyecto en el ámbito de las tecnologías específicas del campo de la Robótica de naturaleza profesional en el que se sinteticen e integren las competencias adquiridas en las enseñanzas. Esta última competencia se cumple con la creación de este proyecto, memoria y documentación, y su defensa ante un tribunal.

2.4. Metodología

La metodología utilizada sigue pautas de investigación sobre el estado del arte previo al trabajo, y posteriormente sobre el *software* utilizado, siempre evaluando de antemano la compatibilidad con el *hardware* disponible, así como realizando pruebas pertinentes sobre su correcto funcionamiento en los distintos entornos, incluyendo la simulación y el laboratorio.

En relación con el desarrollo del *software* demostrativo se siguió un ciclo de desarrollo *software* iterativo, que consiste en la planificación del *software*, el desarrollo del mismo, su consecuente revisión mediante pruebas y su corrección, todo ello de manera periódica, generando en cada una de las iteraciones un resultado ejecutable mejor

que el anterior, hasta conseguir al final una versión completamente funcional, como se ve reflejado en la Figura 2.1. Este proceso de desarrollo puede verse alineado con los principios de mejora continua del ciclo de desarrollo PDCA (*Plan, Do, Check, Act*).

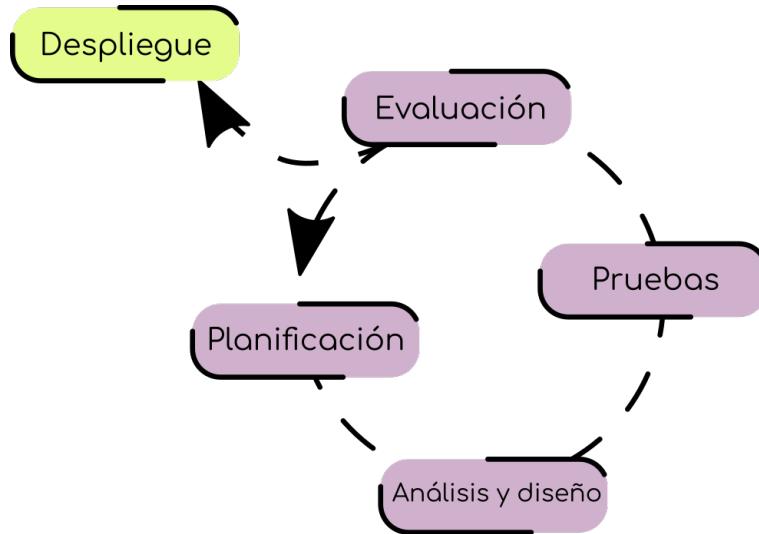


Figura 2.1: Esquema del desarrollo software iterativo (fuente propia)

Este desarrollo implicó pruebas periódicas en simulación con el fin de identificar errores y perfeccionar los valores de los parámetros, para posteriormente evaluar su funcionamiento en un entorno real, concretamente en el Laboratorio de Robótica del Aulario 3 de la Universidad Rey Juan Carlos.

2.5. Plan de trabajo

El desarrollo del proyecto ha comprendido varias etapas, que incluyen la investigación del *software* a utilizar, la investigación del estado del arte, la implementación de una arquitectura *software* funcional en los distintos entornos y el desarrollo del *software* demostrativo o de ejemplo, comprendiendo un periodo de tiempo superior a un año, comenzando en febrero de 2023 y finalizando en junio de 2024.

1. *Investigación del software a utilizar.* Periodo de febrero a mayo de 2023 durante las prácticas de empresa, en las que estuve aprendiendo el funcionamiento de softwares como Zenoh, Zenoh-Flow, Zenoh-Bridge-DDS, CycloneDDS, y acerca de las telecomunicaciones entre robots, 35 horas semanales durante 4 meses.
2. *Investigación del estado del arte.* Periodo de junio a agosto de 2023, en el que se investigó acerca de los trabajos previos relacionados, y sobre la viabilidad y

compatibilidad del proyecto.

3. *Implementación de una arquitectura software funcional en simulación.* Periodo de junio a agosto de 2023, en el que se consiguió un correcto funcionamiento del *software* en simulación.
4. *Desarrollo de software demostrativo.* Periodo de agosto a noviembre de 2023 en el que se migró el *software* desarrollado durante las prácticas a versiones posteriores.
5. *Implementación de una arquitectura software funcional en un entorno real.* Periodo de noviembre de 2023 a enero de 2024 en el que se consiguió un correcto funcionamiento del *software* en el laboratorio.
6. *Pruebas del software desarrollado en el laboratorio.* Periodo de noviembre de 2023 a abril de 2024 en el que se realizaron las pruebas y cambios necesarios para un correcto funcionamiento del *software* demostrativo en el entorno real del laboratorio.
7. *Escritura de la memoria.* Periodo de mayo a junio de 2024 en el que se elaboró el presente documento, así como la presentación para su defensa.

Durante los periodos de desarrollo de este proyecto fuera de las prácticas de empresa, se dedicaban aproximadamente de 30 a 40 horas semanales, manteniendo reuniones con el tutor, que generalmente se llevaban a cabo semanalmente, y ocasionalmente cada dos semanas.

Todo el proceso de trabajo se ha ido alojando en un repositorio público de GitHub¹. Asimismo, el trabajo diario se ha ido documentando detalladamente en la Wiki² de dicho repositorio, haciendo las veces de cuaderno de bitácora, donde quedan reflejados todos los contratiempos, soluciones y pruebas realizados.

¹<https://github.com/RoboticsURJC/tfg-unai>

²<https://github.com/RoboticsURJC/tfg-unai/wiki>

Capítulo 3

Plataforma de desarrollo

Un buen artesano no se separa nunca de sus herramientas, las conoce y las elige con sabiduría.

Leonardo da Vinci

En este capítulo describe la infraestructura, tanto de hardware como de software, utilizada como base para el desarrollo y ejecución de los sistemas robóticos que se explicarán más adelante.

3.1. Hardware

Como se ha detallado en el Capítulo 1, el *hardware* constituye la infraestructura fundamental de los robots, definiendo sus capacidades operativas. Estas capacidades están intrínsecamente ligadas a los componentes físicos del robot, lo que implica que la disponibilidad y calidad del *hardware* son críticas para ejecutar cualquier tarea. Por ejemplo, la ausencia de un sistema de agarre adecuado o la limitación en su fuerza o precisión pueden comprometer la ejecución correcta de una tarea de *pick and place*.

En nuestro caso, hemos utilizado plataformas robóticas como los famosos robots de la familia *Turtlebot*, en concreto los modelos 2 y 4 lite, que se pueden apreciar en la Figura 3.1, debido a su naturaleza móvil y barata, y a su disponibilidad en el laboratorio.

3.1.1. Turtlebot 2

En concreto, el robot *Turtlebot 2* se fundamenta en la base *kobuki*, que dispone de los sensores necesarios para una navegación robusta, como puede ser la IMU o Unidad de Medida Inercial, para las correcciones en los datos de odometría y un *bumper* o sensor de colisiones. La base *kobuki* también dispone de una batería que dota al robot de

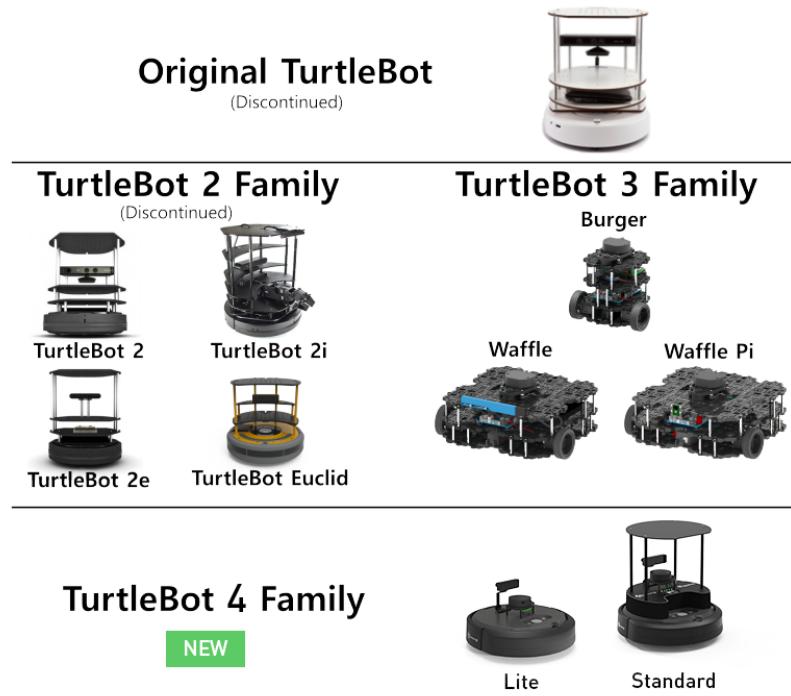


Figura 3.1: Modelos del robot Turtlebot (OSRF)

cierta autonomía. Asimismo, dispone de actuadores que le permiten dicha navegación, como son los motores y otros que le permiten transmitir información de manera visual o auditiva, como los LEDs y el *buzzer* con capacidad de reproducir varios sonidos.



Figura 3.2: Base Kobuki (ROS)

A parte de los componentes de la base, el robot cuenta con una estructura de madera soportada por barras de aluminio que permite apoyar un ordenador portátil, a modo de ordenador de a bordo, sensores externos, cuyos modelos pueden variar dependiendo del robot utilizado en cada momento, ya que se disponen de varias configuraciones de los mismos en el laboratorio, sin afectar a la compatibilidad. Dichos sensores son: un

LIDAR para la detección de obstáculos (modelos RPLIDAR¹ A1 o A2), y una cámara RGBD, de color RGB y profundidad, (modelos Orbbec² Astra o Asus³ Xtion). Este último sensor aporta un gran potencial debido a su versatilidad en cuanto a la detección se refiere. Se puede ver una imagen del modelo en la Figura 3.3.



Figura 3.3: Turtlebot 2 (OSRF)

3.1.2. Turtlebot 4

Por su parte, el robot *Turtlebot 4 lite* utilizado, es bastante parecido al anterior en cuanto a los sensores de la base se refiere, correspondiendo esta vez con el modelo Create 3 de iRobot⁴, que a efectos prácticos es una aspiradora robótica sin aspiradora ni escobillas que las caracterizan, otorgándoles la capacidad de limpiar.

Este robot también añade un sensor LIDAR y una cámara RGBD que en el caso de nuestro laboratorio los modelos *RPLIDAR-S2* y *Oak-D-Pro* respectivamente, sustituyendo a los modelos originales *RPLIDAR-A1* y *Oak-D-Lite*. A diferencia de su versión estándar y del anterior robot, este no tiene una estructura de soporte, por lo que el

¹https://www.slamtec.ai/product/slamtec-rplidar-s2/?gad_source=1&gclid=CjwKCAjwrIixBhBbEiwACEqDJdjBNB-VeyXLm1hx033F5wKfJLRu6KRyC1aa4NfaTWvre8dR0scc8xoCMq8QAvD_BwE

²<https://www.orbbec.com/>

³<https://www.asus.com/es/>

⁴https://www.irobot.es/es_ES/roomba.html?gad_source=1&gclid=CjwKCAjwrIixBhBbEiwACEqDJcM6GM0s9Ew0S98K9IJv0aBGm30dR6D1pk5T09dpqwcvNVTaYr7ZTBoC3HgQAvD_BwE

ordenador de a bordo es una Raspberry Pi⁵ 4 Model B, que en nuestro caso tiene 8Gb de RAM, aunque en el modelo original es de 4Gb. Se puede ver una imagen del modelo en la Figura 3.4.



Figura 3.4: Turtlebot 4 Lite –izquierda– y Standard –derecha– (OSRF)

3.1.3. Ordenadores de a bordo

Como ya se ha presentado en la subsección 3.1.1, para comandar ordenes al robot, se necesita un ordenador de a bordo. En nuestro caso se ha utilizado un ordenador personal portátil, de la marca HP⁶, del modelo *ProBook 450 G6*.



Figura 3.5: HP ProBook 450 G6 (You)

Para algunas pruebas también se añadió una Raspberry Pi Model B de 4Gb de RAM, como ordenador de a bordo de un robot *Turtlebot 2*.

⁵<https://www.raspberrypi.com/>

⁶<https://www.hp.com/es-es/home.html>



Figura 3.6: Raspberry Pi 4, modelo B (Team)

3.1.4. Ordenador principal

Para llevar a cabo este trabajo se precisó de la utilización de un *hardware* con la capacidad de desarrollar el *software*, así como de ejecutar la simulación necesaria para la realización de las pruebas. Este ordenador también se corresponde con el portátil HP mencionado en la Sección 3.1.3.

En cuanto a las pruebas realizadas con robots reales, el uso de este mismo componente a modo de ordenador principal o centralizado también fue necesario. Este ordenador portátil es el *hardware* en el que se albergó toda la capacidad de cómputo acerca de la toma de decisiones, así como el lugar de ejecución del *software* utilizado para la ejecución de la aplicación robótica creada.

3.1.5. Router

La comunicación entre los distintos robots es una parte indispensable de este trabajo, por lo que surge la necesidad de añadir un *hardware* con la capacidad de comunicar múltiples robots, en nuestro caso un router de la marca ASUS, Modelo ROG Rapture GT-AXE16000⁷ sin necesidad de conexión a internet, ya que se utiliza únicamente para la comunicación local entre los distintos robots.

3.2. Software

Por otro lado, el *software* desempeña un papel crucial al convertir las capacidades del *hardware* en acciones concretas. Es el encargado de emitir las instrucciones necesarias para que el robot, en conjunto, funcione como un sistema inteligente capaz de

⁷<https://rog.asus.com/es/networking/rog-rapture-gt-axe16000-model/>



Figura 3.7: Router ASUS, modelo ROG Rapture GT-AXE16000 (Ludlow)

llevar a cabo la tarea definida. En ausencia de una programación inteligente, el robot permanecerá como un dispositivo inerte, con un potencial latente pero incapaz de realizar tareas de manera autónoma.

A continuación se explicarán los distintos *softwares* utilizados, desde el sistema operativo, pasando por el lenguaje de programación principal, hasta el *middleware* robótico.

3.2.1. Sistema Operativo

El sistema operativo utilizado es Ubuntu en su versión 22.04 LTS (Long Term Support), lo que implica su soporte a largo plazo. Además, es una distribución de GNU/Linux, basada en Debian GNU/Linux, que se basa en el *software* libre y de código abierto, lo que permite una mayor participación de cualquier persona en su desarrollo, aumentando su robustez y mejorando su soporte.

Esta elección se fundamenta principalmente en la facilidad de uso y programación con el mismo, así como la compatibilidad con el resto de programas utilizados. La versión del sistema operativo utilizada, coincide con la más nueva de soporte a largo plazo en la fecha de realización de este proyecto.

3.2.2. Lenguajes de programación

En este caso, el lenguaje de programación utilizado para el desarrollo del *software* de este trabajo, es Python, ya que debido a su sencillez de programación y a su legibilidad de código, da lugar a un buen entorno para el desarrollo de *software* educativo, permitiendo una gran facilidad a la hora de aprender, además de ser de código abierto, con las respectivas ventajas que esto supone, explicadas en la Sección 3.2.1.

Python es un lenguaje de programación interpretado, lo que evita tener que ser

compilado, que suele conllevar un proceso tedioso para programadores principiantes. Además se define como lenguaje de alto nivel, y que soporta la programación orientada a objetos (OOP) lo que también permite su escala en complejidad, si así se requiriese.

Este lenguaje ha tomado un gran impulso en los últimos años debido a su popularidad, aunque cabe destacar que es un lenguaje sumamente lento en comparación a otros más simples y generalmente de más bajo nivel como C. En concreto utilizaremos la versión 3 de este lenguaje debido a su compatibilidad y a su relevancia actual.

Este lenguaje de programación será usado directamente a la hora de programar nuestro propio *software*, pero también serán utilizados otros lenguajes indirectamente, por aplicaciones o nodos externos, como C++ en algunos nodos de ROS2 utilizados o Rust en los programas relacionados con Zenoh.

3.2.3. Middleware Robótico

ROS2 es un *middleware* robótico, creado para abstraer al programador del *hardware* del robot, permitiendo una mayor atención en el desarrollo *software* del mismo. Por todo ello, simplifica este proceso, sirviendo de herramienta tanto para la programación o desarrollo, como para el uso o la ejecución del *software* en cuestión.

En este *middleware* se divide el código en nodos ejecutables de manera iterativa a una determinada frecuencia, que pueden comunicarse a través del protocolo DDS, generalmente con la utilización de *topics*, ya que dicho protocolo basa su funcionamiento en un modelo de publicador/suscriptor. La librería cliente de ROS2 permite programar los nodos en Python o C++, aunque la amplia comunidad ha desarrollado extraoficialmente el equivalente en otros lenguajes como Java, C, Rust o incluso Java o Android, entre otros.

3.2.4. Simulación

En cuanto a la simulación, el *software* utilizado es Gazebo, un simulador que permite la visualización de los robots y de los entornos, gracias a su previa descripción, generalmente mediante un archivo con formato XML (eXtensible Markup Language) o derivado, que pueden ser URDF⁸ (Unified Robotics Description Format) y SDF⁹ (Simulation Description Format) respectivamente. Puede verse un ejemplo de simulación en la Figura 3.8.

⁸<https://wiki.ros.org/urdf>

⁹<http://sdfformat.org/>

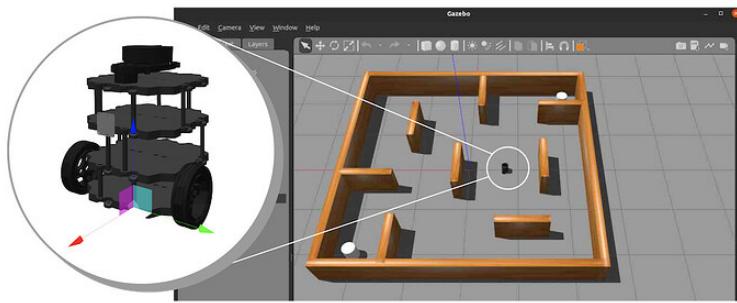


Figura 3.8: Gazebo simulando un robot Turtlebot 3 en un mundo virtual (Yuhong)

Este simulador también permite la visualización de ciertos datos de sus sensores si así se requiere, o de las partes o componentes de un robot gracias al archivo de descripción del mismo, mencionado en el párrafo anterior. Además permite el uso de *plugins* que pueden añadir las funcionalidades que sean necesarias.

3.2.5. Protocolos de comunicación

Una de las empresas más importantes en este ámbito es ZettaScale Technology¹⁰, que ha tomado mucha relevancia en los últimos años, ya que es responsable de una de las implementaciones de *middleware* de telecomunicaciones de DDS utilizado en ROS2, llamado CycloneDDS¹¹ y son los desarrolladores de un reciente protocolo de comunicaciones llamado Zenoh¹², por sus siglas en inglés, Zero Overhead Network Protocol, que ya cuenta con importantes clientes como la NASA, debido a las prestaciones que este ofrece, superando en la mayoría de casos a los protocolos de su misma índole y que ya ha sido oficialmente seleccionado como el próximo RMW de ROS2¹³. El protocolo Zenoh, está basado en un modelo de publicador/suscriptor, así como DDS, por lo que hacen una buena sinergia.

Además de esto poseen un potente *framework* para la programación de flujos de datos aún en desarrollo, llamado Zenoh-Flow¹⁴, así como un puente llamado Zenoh-Bridge-DDS¹⁵ que hace las veces de traductor entre los protocolos DDS y Zenoh, lo que permite la comunicación entre estos flujos de datos con nodos de ROS2, haciendo posible la creación de aplicaciones robóticas en forma de flujos de datos, utilizando nodos de ROS2 existentes, y fusionando de esta manera ambos campos.

¹⁰<https://www.zettascale.tech/>

¹¹<https://cyclonedds.io/>

¹²<https://zenoh.io/>

¹³<https://discourse.ros.org/t/ros-2-alternative-middleware-report/33771>

¹⁴<https://zenoh.io/blog/2023-02-10-zenoh-flow/>

¹⁵<https://github.com/eclipse-zenoh/zenoh-plugin-dds>

Es notable destacar el gran soporte que brinda esta empresa, que ayuda en gran medida a resolver los problemas o dudas que puedan surgir durante el desarrollo o despliegue de las aplicaciones.

El protocolo de comunicaciones utilizado en el *software* desarrollado en este trabajo es Zenoh, que reduce en gran medida los bytes de la cabecera de los mensajes, y ofrece mejores prestaciones en comparación con otros protocolos de su misma índole, como se puede apreciar en la gráfica de la Figura 3.9.

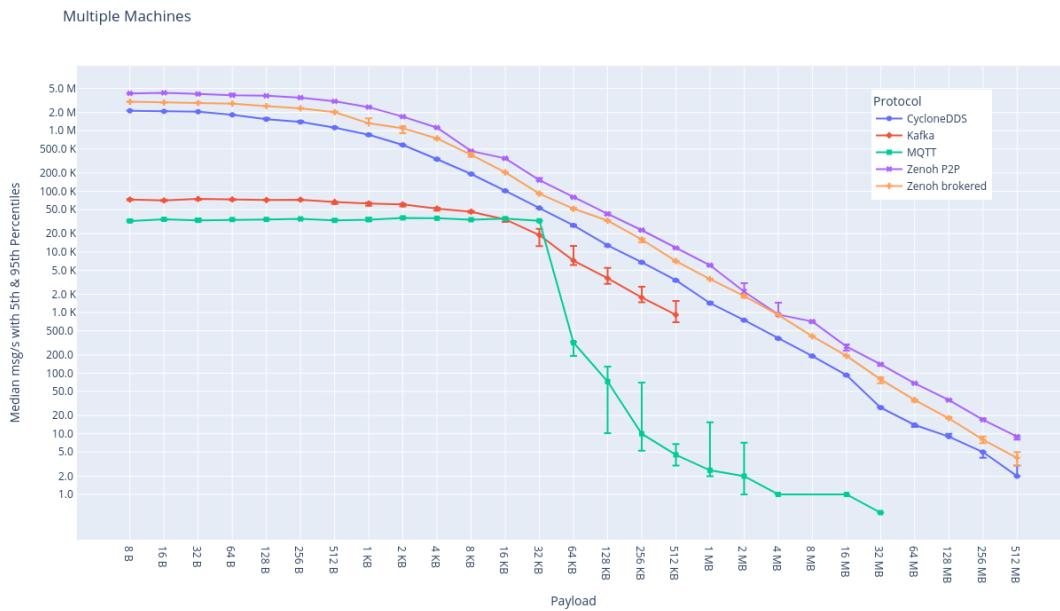


Figura 3.9: Rendimiento de protocolos en varias máquinas (Eclipse Foundation).

En nuestro caso utilizaremos Zenoh-Flow para la programación de flujos de datos, ya que tiene una API (Application Programming Interface) en Python, lo que permite programarlos de manera sencilla, como se ha explicado en la Sección 3.2.2.

También se utilizará DDS indirectamente en los nodos de ROS2 de los que se hará uso, para los que utilizaremos la RMW de CycloneDDS, ya que es el que mejor funciona con el software utilizado.

Además enlazaremos dichos nodos de ROS2 con nuestro flujo de datos mediante el uso del Zenoh-Bridge-DDS, que nos permitirá traducir los mensajes de un protocolo a otro en ambos sentidos, ya que Zenoh-Flow utiliza Zenoh para las comunicaciones, mientras que ROS2 se comunica mediante DDS.

3.2.6. Visión Artificial

Para la detección de objetos a partir de imágenes, ya sean generadas por una cámara o creadas a partir de los datos de otros sensores, se ha utilizado una reconocida librería de procesamiento de imágenes llamada OpenCV, disponible tanto en Python como en C++.

Este software permite distintos cambios de espacios de color, entre los que se encuentran desde el típico formato RGB, pasando por el formato GreyScale o escala de grises, que permite ahorrar memoria, o detectar bordes, hasta el formato HSV (Hue, Saturation, Value), que permite, entre otros muchos usos, trabajar con los colores en distintas intensidades de luz brindando mejores resultados que con otros espacios de color. Este espacio de color se ve representado en la Figura 3.10.

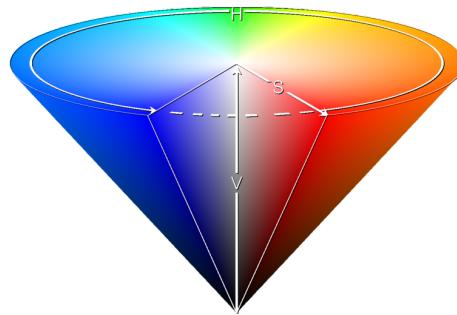


Figura 3.10: Espacios de color HSV (BuckyBall)

También permite distintos tipos de detecciones como pueden ser la detección de rostros, bordes o contornos, o la detección de objetos mediante aprendizaje profundo. En nuestro caso se han utilizado la detección de objetos mediante el uso de un filtro de color aplicado a la imagen obtenida desde la cámara, y la detección de bordes en conjunto con la posterior detección de círculos, en una imagen generada a partir de los datos posicionales de un sensor LIDAR, como ya se explicará más adelante.

Esta librería también se apoya en otra, llamada Numpy, encargada de la representación matemática eficiente de matrices, así como de operaciones matemáticas entre ellas, ya que las imágenes a color, generalmente son matrices tridimensionales, que albergan tres canales (RGB o HSV), cuyas columnas tienen el tamaño de la longitud vertical de la imagen en píxeles, así como el tamaño de las filas corresponde con el ancho de la imagen, como se puede ver en la Figura 3.11.

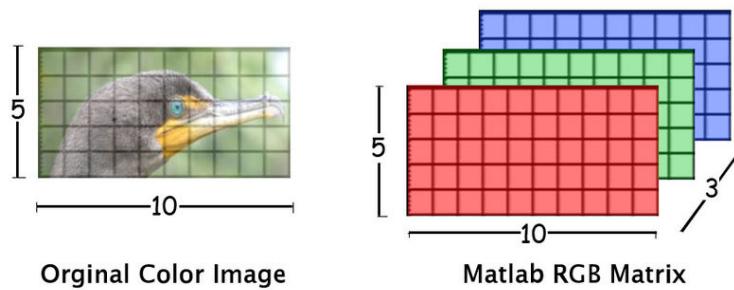


Figura 3.11: Representación de imagen RGB como matriz (Wiki300)

3.2.7. Software de navegación

El *software* de navegación utilizado para trasladar los robots de una posición a otra en el espacio consiste en los paquetes del conocido *stack* de Nav2¹⁶, que brinda herramientas que solucionan la localización de manera probabilística mediante el algoritmo de Montecarlo y una navegación bastante robusta y precisa del robot.

3.2.8. Software matemático

Para suplir la necesidad de realizar operaciones matemáticas de todo tipo, como pueden ser las transformaciones entre espacios de coordenadas o entre ejes de coordenadas, operaciones simples o complejas con grandes cantidades de números, como matrices o imágenes o cualquier otro tipo de operaciones, se han utilizado tanto la librería Numpy como Math, dependiendo del tipo de operación, ya que ambas son muy eficientes.

Para operaciones de transformadas (TFs) de ROS2 entre dos *frames*, utilizaremos las propias herramientas de ROS2, aunque no podremos utilizar cualquiera, ya que la mayoría sufren la necesidad de correr en un nodo de ROS2, y no funcionan fuera del mismo, por lo que concretamente utilizaremos la función del Código 3.1, por lo que se necesitará haber creado una suscripción previamente al *topic* de TFs del robot en cuestión.

3.2.9. Visualización de datos

Para representar de manera simplificada y visual todos los datos, tanto de los sensores como de los actuadores de los robots, se utiliza RViz2¹⁷, una herramienta integrada en ROS2, que permite visualizar desde imágenes procedentes de las cámaras,

¹⁶<https://navigation.ros.org/>

¹⁷<https://github.com/ros2/rviz>

```

from builtin_interfaces.msg import Duration
import tf2_ros, rclpy

buffer_core = tf2_ros.BufferCore(Duration(sec=1, nanosec=0))
buffer_core.lookup_transform_core(
    frame_id_1, frame_id_2, rclpy.time.Time()
)

```

Código 3.1: Función para calcular transformadas (TFs)

como el propio modelo del robot en movimiento o la posición probabilística del mismo derivada de un algoritmo de localización como el mencionado en la Sección 3.2.7.

Además, a la hora de hacer pruebas, también se han usado otras herramientas como la misma librería de OpenCV, para crear ventanas que muestran una imagen con *sliders* que permiten modificar variables, generando de esta manera una mayor interactividad y fluidez a la hora de encontrar los valores óptimos de ciertos parámetros, como pueden ser los valores máximos y mínimos de un filtro de color, permitiendo ver el resultado en la imagen de manera dinámica, como se puede observar en la Figura 3.12.

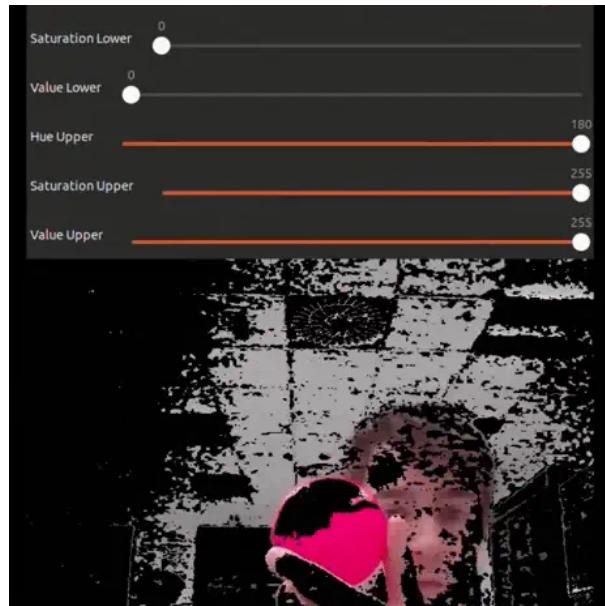


Figura 3.12: *Sliders* de un filtro de color con OpenCV (fuente propia).

Asimismo, se ha utilizado software externo, como PlotJuggler¹⁸, para mostrar gráficas sobre datos provenientes de *topics* en directo.

¹⁸<https://plotjuggler.io/>

Capítulo 4

Arquitectura Software con Zenoh y ROS

Quizás algún fragmento de libro inspirador...

Autor, Título

Escribe aquí un párrafo explicando brevemente lo que vas a contar en este capítulo. En este capítulo (y quizás alguno más) es donde, por fin, describes detalladamente qué has hecho y qué experimentos has llevado a cabo para validar tus desarrollos.

4.1. Snippets

Puede resultar interesante, para clarificar la descripción, mostrar fragmentos de código (o *snippets*) ilustrativos. En el Código 5.1 vemos un ejemplo escrito en C++.

```
void Memory::hypothesizeParallelograms () {
    for(it1 = this->controller->segmentMemory.begin(); it1++) {
        squareFound = false; it2 = it1; it2++;
        while ((it2 != this->controller->segmentMemory.end()) && (!squareFound))
        {
            if (geometry::haveACommonVertex((*it1), (*it2), &square)) {
                dist1 = geometry::distanceBetweenPoints3D ((*it1).start, (*it1).end);
                dist2 = geometry::distanceBetweenPoints3D ((*it2).start, (*it2).end);
            }
        // [...]
```

Código 4.1: Función para buscar elementos 3D en la imagen

En el Código 5.2 vemos un ejemplo escrito en Python.

```

def mostrarValores():
    print (w1.get(), w2.get())

master = Tk()
w1 = Scale(master, from_=0, to=42)
w1.pack()
w2 = Scale(master, from_=0, to=200, orient=HORIZONTAL)
w2.pack()
Button(master, text='Show', command=mostrarValores).pack()

mainloop()

```

Código 4.2: Cómo usar un Slider

4.2. Verbatim

Para mencionar identificadores usados en el código —como nombres de funciones o variables— en el texto, usa el entorno literal o verbatim `hypothesizeParallelograms()`. También se puede usar este entorno para varias líneas, como se ve a continuación:

```

void Memory::hypothesizeParallelograms () {
    // add your code here
}

```

4.3. Ecuaciones

Si necesitas insertar alguna ecuación, puedes hacerlo. Al igual que las figuras, no te olvides de referenciarlas. A continuación se exponen algunas ecuaciones de ejemplo: Ecuación 5.1 y Ecuación 5.2.

$$H = 1 - \frac{\sum_{i=0}^N \frac{(d_{js} + d_{je})}{2}}{M} \quad (4.1)$$

Ecuación 4.1: Ejemplo de ecuación con fracciones

$$v(\text{entrada}) = \begin{cases} 0 & \text{if } \epsilon_t < 0,1 \\ K_p \cdot (T_t - T) & \text{if } 0,1 \leq \epsilon_t < M_t \\ K_p \cdot M_t & \text{if } M_t < \epsilon_t \end{cases} \quad (4.2)$$

Ecuación 4.2: Ejemplo de ecuación con array y letras y símbolos especiales

4.4. Tablas o cuadros

Si necesitas insertar una tabla, hazlo dignamente usando las propias tablas de L^AT_EX, no usando pantallazos e insertándolas como figuras... En el Cuadro 5.1 vemos un ejemplo.

| Parámetros | Valores |
|-----------------------|-------------------------------|
| Tipo de sensor | Sony IMX219PQ[7] CMOS 8-Mpx |
| Tamaño del sensor | 3.674 x 2.760 mm (1/4"format) |
| Número de pixels | 3280 x 2464 (active pixels) |
| Tamaño de pixel | 1.12 x 1.12 um |
| Lente | f=3.04 mm, f/2.0 |
| Ángulo de visión | 62.2 x 48.8 degrees |
| Lente SLR equivalente | 29 mm |

Cuadro 4.1: Parámetros intrínsecos de la cámara

Capítulo 5

Experimentos

Quizás algún fragmento de libro inspirador...

Autor, Título

Escribe aquí un párrafo explicando brevemente lo que vas a contar en este capítulo. En este capítulo (y quizás alguno más) es donde, por fin, describes detalladamente qué has hecho y qué experimentos has llevado a cabo para validar tus desarrollos.

5.1. Snippets

Puede resultar interesante, para clarificar la descripción, mostrar fragmentos de código (o *snippets*) ilustrativos. En el Código 5.1 vemos un ejemplo escrito en C++.

```
void Memory::hypothesizeParallelograms () {
    for(it1 = this->controller->segmentMemory.begin(); it1++) {
        squareFound = false; it2 = it1; it2++;
        while ((it2 != this->controller->segmentMemory.end()) && (!squareFound))
            {
                if (geometry::haveACommonVertex((*it1), (*it2), &square)) {
                    dist1 = geometry::distanceBetweenPoints3D ((*it1).start, (*it1).end);
                    dist2 = geometry::distanceBetweenPoints3D ((*it2).start, (*it2).end);
                }
            // [...]
```

Código 5.1: Función para buscar elementos 3D en la imagen

En el Código 5.2 vemos un ejemplo escrito en Python.

5.2. Verbatim

Para mencionar identificadores usados en el código —como nombres de funciones o variables— en el texto, usa el entorno literal o verbatim `hypothesizeParallelograms()`. También se puede usar este entorno para varias líneas, como se ve a continuación:

```

def mostrarValores():
    print (w1.get(), w2.get())

master = Tk()
w1 = Scale(master, from_=0, to=42)
w1.pack()
w2 = Scale(master, from_=0, to=200, orient=HORIZONTAL)
w2.pack()
Button(master, text='Show', command=mostrarValores).pack()

mainloop()

```

Código 5.2: Cómo usar un Slider

```

void Memory::hypothesizeParallelograms () {
    // add your code here
}

```

5.3. Ecuaciones

Si necesitas insertar alguna ecuación, puedes hacerlo. Al igual que las figuras, no te olvides de referenciarlas. A continuación se exponen algunas ecuaciones de ejemplo: Ecuación 5.1 y Ecuación 5.2.

$$H = 1 - \frac{\sum_{i=0}^N \frac{(\frac{d_{js} + d_{je}}{2})}{N}}{M} \quad (5.1)$$

Ecuación 5.1: Ejemplo de ecuación con fracciones

$$v(\text{entrada}) = \begin{cases} 0 & \text{if } \epsilon_t < 0,1 \\ K_p \cdot (T_t - T) & \text{if } 0,1 \leq \epsilon_t < M_t \\ K_p \cdot M_t & \text{if } M_t < \epsilon_t \end{cases} \quad (5.2)$$

Ecuación 5.2: Ejemplo de ecuación con array y letras y símbolos especiales

5.4. Tablas o cuadros

Si necesitas insertar una tabla, hazlo dignamente usando las propias tablas de L^AT_EX, no usando pantallazos e insertándolas como figuras... En el Cuadro 5.1 vemos un ejemplo.

| Parámetros | Valores |
|-----------------------|-------------------------------|
| Tipo de sensor | Sony IMX219PQ[7] CMOS 8-Mpx |
| Tamaño del sensor | 3.674 x 2.760 mm (1/4"format) |
| Número de pixels | 3280 x 2464 (active pixels) |
| Tamaño de pixel | 1.12 x 1.12 um |
| Lente | f=3.04 mm, f/2.0 |
| Ángulo de visión | 62.2 x 48.8 degrees |
| Lente SLR equivalente | 29 mm |

Cuadro 5.1: Parámetros intrínsecos de la cámara

Bibliografía

- [1] Decreto 48/2015, de 14 de mayo, por el que se establece para la comunidad de madrid el currículo de la educación secundaria obligatoria. Boletín Oficial de la Comunidad de Madrid, 118, de 20 de mayo de 2015.
- [2] Decreto 89/2014, de 24 de julio, por el que se establece para la comunidad de madrid el currículo de la educación primaria. Boletín Oficial de la Comunidad de Madrid, 175, de 25 de Julio de 2014.
- [3] Alami, R., F. S. H. M. I. F. y. R. F. (1998). Multi-robot cooperation in the martha project. *IEEE Robotics Automation Magazine*, 5(1):36–47.
- [4] Arai, T. y Parker, L. (2003). Editorial: Advances in multi-robot systems.
[BuckyBall] BuckyBall. File: hsv cone.png.
- [6] Chaimowicz, L., S. T. K. V. y. C. M. (2001). An architecture for tightly coupled multi-robot cooperation. In *Proceedings 2001 ICRA. IEEE International Conference on Robotics and Automation (Cat. No.01CH37164)*, volume 3, pages 2992–2997 vol.3.
- [Eclipse Foundation] Eclipse Foundation, Z. Comparing the performance of zenoh, mqtt, kafka, and dds.
- [8] Fox, D., B. W. K. H. y. T. S. (2000). A probabilistic approach to collaborative multi-robot localization. *Autonomous Robots*, 8(3):325–344.
- [Ludlow] Ludlow, D. Asus rog rapture gt-axe1600 review.
- [10] Ministerio de Educación, Formación Profesional y Deportes, I. N. d. T. E. y. d. F. d. P. (2018). Programación, robótica y pensamiento computacional en el aula. situación en españa y propuesta normativa.
- [OSRF] OSRF. Turtlebot.
- [12] Parker, L. E. (2003). Current research in multirobot systems. *Artificial Life and Robotics*, 7(1):1–5.

- [ROS] ROS, O. Kobuki.
- [14] Sheng, W., Y. Q. T. J. y. X. N. (2006). Distributed multi-robot coordination in area exploration. *Robotics and Autonomous Systems*, 54(12):945–955.
- [Team] Team, B. File:raspberry pi 4 b.png.
- [16] Trawny, N., R. S. I. y. G. G. B. (2009). Cooperative multi-robot localization under communication constraints. In *2009 IEEE International Conference on Robotics and Automation*, pages 4394–4400.
- [17] Vega, J. M. (2018). *Educational framework using robots with vision for constructivist teaching Robotics to pre-university students*. Doctoral thesis on computer science and artificial intelligence, University of Alicante.
- [18] Verma, J. K. y Ranga, V. (2021). Multi-robot coordination analysis, taxonomy, challenges and future scope. *Journal of Intelligent & Robotic Systems*, 102(1):10.
- [Wiki300] Wiki300. File:matl.jpg. Page Version ID: 449974156.
- [You] You, H. S. Hp probook 440 g6 laptop.
- [Yuhong] Yuhong, L. New ros online courses: Gazebo simulator, intermediate ros2, tf ros2. Section: Training & Education.