Exploración robótica colaborativa de interiores

Ricardo Ercoli, Fausto Navadian, Joaquin Urrisa, Pablo Monzón, Facundo Benavides
Universidad de la República, Montevideo, Uruguay
Email: {ricardo.ercoli, fausto.navadian, joaquin.urrisa, monzon, fbenavid}@fing.edu.uy

***Resumen*—La exploración robótica colaborativa se basa en que múltiples robots colaboren con el objetivo de descubrir un entorno desconocido. Este trabajo presenta la implementación de una flota colaborativa de robots diseñados para realizar mapeo 2D en interiores de manera autónoma. Sus principales contribuciones son: i) una solución original al problema de distribución de múltiples tareas a múltiples robots implementada como una versión distribuida del mecanismo de subastas; y 2) la liberación del código a través de un repositorio público.**

***Palabras Claves*—Robótica autónoma móvil, ROS2, Exploración colaborativa, mapeo 2D.**

I. Introducción

La exploración robótica colaborativa se basa en que múltiples robots trabajen en conjunto con el objetivo de descubrir un entorno desconocido. Estas soluciones ofrecen ventajas tales como una mayor eficiencia en la exploración, reducción del tiempo de exploración y una mejor cobertura del área de interés. Además, si la colaboración entre robots es descentralizada, se eliminan puntos de falla individuales y se mejora la robustez del sistema [1]. Este trabajo presenta la implementación de una flota colaborativa de robots diseñados para realizar mapeo 2D en interiores de manera autónoma. Dentro de la solución se destaca la implementación de una propuesta original para realizar la distribución de tareas basada en subastas de forma distribuida. Primeramente, se implementó una flota robótica con las capacidades sensoriales y de cómputo necesarias para la misión. Partiendo de Robotito [2], un plataforma robótica educativa, se le agregaron una placa controladora Odroid N2 [3], un lidar RPLIDAR A1 [4] e interfaces de comunicación inalámbrica para que cada unidad pudiera realizar mapeo 2D de su entorno cercano, intercambiar información (mapas y posiciones) y coordinar acciones. La implementación se desarrolló utilizando el framework ROS2 [5] y se encuentra disponible en [6]. Finalmente, se realizaron pruebas en entornos simulados para evaluar su desempeño y una prueba de concepto en un entorno real para probar su factibilidad de uso práctico.

II. Hardware

En esta sección, se introduce el robot omnidireccional construido, Rogel. Como se ve en la Fig.1, se descompone el sistema en tres componentes: la Plataforma Motora heredada de Robotito, la Forza Power Bank [7] encargada de alimentar todo el sistema y el sub sistema de Cómputo y Sensado. La plataforma motora se encarga del control de los motores, del cálculo de odometría y también de generar una comunicación con el sistema de cómputo y sensado para obtener comandos de velocidad y reportar la información de odometría. El sub sistema de Cómputo y Sensado se encarga de la lógica de alto nivel: SLAM, planificación de rutas, selección de objetivos y distribución de tareas. En la Fig.2 se ve una imagen del sistema robótico implementado. La plataforma motora se ubica en el nivel más cercano al suelo; siguiendo se encuentra la batería y en los últimos dos niveles se tiene el cómputo y el sensado, respectivamente.

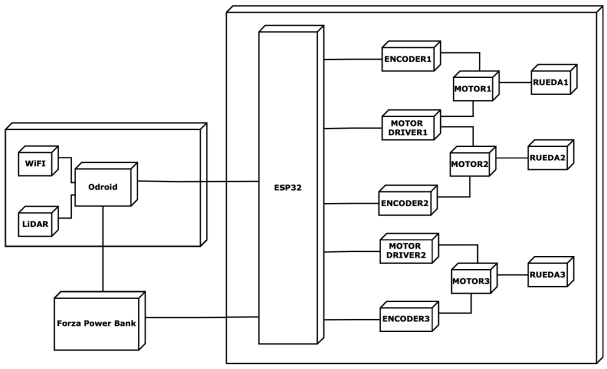


Figura 1: Arquitectura: Plataforma motora (derecha), Cómputo y sensado (izquierda-arriba), Power bank (izquierda-abajo).

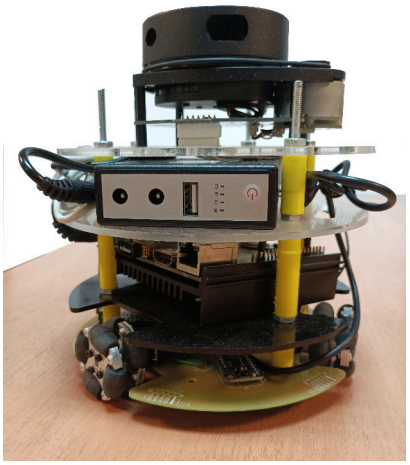


Figura 2: Plataforma robótica: Rogel

III. Firmware

Esta sección introduce el firmware desarrollado para llevar a cabo el control del movimiento del robot, el cálculo de odometría y un protocolo de comunicación serial para establecer el intercambio de información con la plataforma de cómputo central. En la Fig.3 se observan los módulos de firmware que componen la solución. La implementación sigue un diseño en capas de módulos que proveen servicios fomentando el

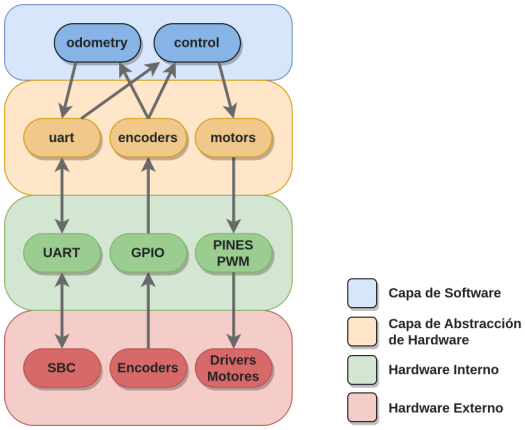


Figura 3: Firmware. Capas y servicios.

reuso y facilitando el mantenimiento y la extensión de la solución a futuro. En alto nivel, el módulo *UART* implementa un protocolo de comunicación que permite el intercambio de comandos de velocidad y odometría entre el sub sistema de cómputo y la plataforma motora. El módulo *control* regula la velocidad de las ruedas y el módulo *odometry* calcula la odometría en función del desplazamiento de cada una de las ruedas. El resto de los módulos implementan la comunicación con su contraparte de hardware y fueron tomados de la plataforma Robotito.

IV. Software

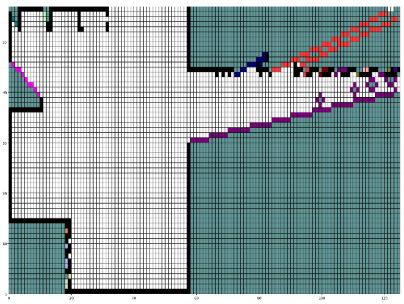
Esta sección proporciona una descripción de cada uno de los paquetes utilizados y de los algoritmos de agrupación de fronteras y división de tareas desarrollados.

IV-A. Paquetes externos

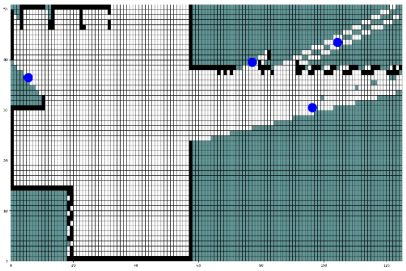
Para interactuar con el *LiDAR* se utilizó el paquete *rplidar_ros2* [8], el cual maneja los drivers del sensor y publica los datos sensoriales en el tópico */scan*. Para la realización de SLAM se utilizó un *fork* de *slam_toolbox* [9] que permite la construcción de mapas de forma colaborativa. La representación del entorno se basa en una grilla de ocupación probabilística. En la Fig.4a se observa la representación parcial de un entorno. Finalmente, se utiliza el *stack* de navegación *nav2* con el propósito de implementar la navegación autónoma entre dos puntos: configuración actual y final (asociada a la tarea asignada). Estos tres paquetes habilitan a la flota a realizar mapeo 2D de forma colaborativa a partir de datos sensoriales locales y a moverse autónomamente hacia los objetivos asignados.

IV-B. Identificación de tareas

Para que la exploración sea autónoma se utiliza la noción de punto de frontera definida en [10], [11] como heurística para la identificación de tareas u objetivos. En la Fig.4a se puede observar en colores las fronteras calculadas. Asimismo, con el fin de disminuir la probabilidad de que dos o más robots vayan hacia fronteras cercanas, se implementó un algoritmo de agrupamiento de fronteras contiguas. Luego, se filtran aquellos grupos con menos de N fronteras y se subdividen los que



(a) Grilla de ocupación. Tipos de celda: desconocida (gris), ocupada (negro), libre (blanco) y frontera (rojo, azul, rosa y violeta).



(b) Centroides. Cada punto azul corresponde a un agrupamiento de puntos de frontera.

Figura 4: Resultados del algoritmo de agrupación de fronteras.

tengan más de M fronteras siendo N y M son parámetros configurables del sistema. Finalmente, como se aprecia en la Fig.4b, el algoritmo publica los centroides de cada grupo.

IV-C. Método de subasta y asignación de tareas

La Sección IV-B introdujo el concepto de puntos fronteras y la idea de agruparlos para generar los objetivos a distribuir entre los robots durante la exploración. En un entorno multi-robot es importante distribuir dichos objetivos de forma de minimizar el tiempo total de la misión. Para ello, se implementa un método original basado en subastas distribuidas donde el subastador puede variar con el tiempo. El proceso de subasta comienza con la designación del subastador (denominado *broker*). El mismo envía la lista de objetivos a cada uno de los robots que participan en la subasta (denominados *bidders*) y espera un periodo a recibir sus ofertas. Recibidas las mismas, calcula la asignación óptima y envía a cada *bidder* un objetivo. La designación del *broker* depende de un factor impredecible como la disponibilidad. De este modo, el primer robot que queda disponible toma el rol, mitigando el problema del punto de falla único del sistema. Como el proceso es asíncrono, no todos los robots podrán participar de la subasta, ya que algunos pueden estar llevando adelante alguna tarea. La Fig.5 muestra la máquina de estados que determina las tareas que debe realizar cada robot, logrando la descentralización del subastador. El estado inicial es **IDLE**. Es el único estado que requiere *input* humano, y su única función es esperar por la señal para que la flota comience a operar. De esta forma se puede corroborar la correcta inicialización de cada robot antes de comenzar a explorar, especialmente útil al momento de

desarrollar. Una vez recibida la señal de inicio, se pasa al es-

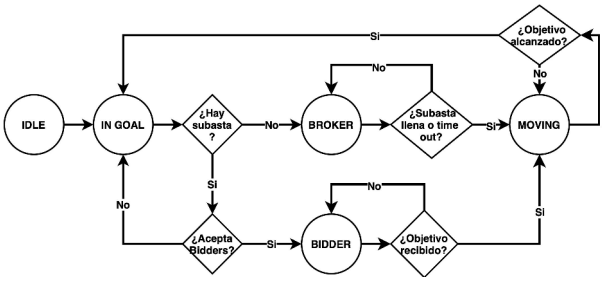


Figura 5: Máquina de estados de cada robot.

tado **IN GOAL**. Dicho estado representa que el robot alcanzó el último objetivo recibido y está disponible para recibir un nuevo objetivo de exploración. Aquí es donde se determina el rol del robot en la próxima subasta. Existen tres posibles escenarios: no hay una subasta activa, hay una subasta activa que acepta *bidders* y por último, existe una subasta activa pero ya no acepta *bidders*. En el primer caso, el robot asume el rol de *broker* para comenzar una nueva subasta y pasa al estado con el mismo nombre. Análogamente, para el segundo caso, el robot pasa al estado **BIDDER**, comunicándole previamente al *broker* su participación en la subasta. Por último, en el tercer caso, el robot realiza una espera activa (haciendo *polling* al estado de la subasta) hasta la finalización de la subasta, en cuyo momento se repite el proceso de selección de *broker*. La Fig.6 ilustra un escenario típico para una flota de 3 robots, con todos los robots participando de la misma subasta. El robot que se identifique con el estado **BROKER** es el encargado de llevar a cabo la subasta. Primeramente, espera cierto tiempo a que otros robots se sumen a la subasta. Es interesante observar que entre mayor sea el tiempo de espera, mayor es la probabilidad de sumar robots a cada subasta y con ello lograr una mayor coordinación (mejor distribución de tareas). Por el contrario, un tiempo excesivamente grande podría ocasionar esperas innecesarias. Durante los experimentos se utilizó un periodo de $6s$, definido a partir del desempeño observado en algunas ejecuciones de prueba. Finalizada dicha espera, se dejan de aceptar participantes (lo cual se comunica cambiando el valor del tópico */broker_id* a un valor negativo), el *broker* calcula las fronteras y las envía a los *bidders*. Luego, el *broker* recibe los rankings de las fronteras de cada participante y da por finalizada la subasta, enviando un 0 al tópico */broker_id*. Por último, asigna y envía una frontera a cada *bidder* como próximo objetivo a explorar. Cabe destacar que el *broker* también participa de la subasta, asignándose un objetivo. Para el caso de un *bidder*, la primera tarea es comunicarle al *broker* su intención de participar en la subasta. Para eso, envía su identificador al *broker* a través de un servicio de este, quedando a la espera de recibir las fronteras objetivos, las cuales evalúa, ordena en un ranking basado en una función de costos basada en la distancia euclídea (debido a bajo costo computacional), y devuelve al *broker*. Finalmente, espera la frontera hacia la cual debe navegar en el siguiente estado. Finalizada la subasta, tanto el *broker* como los *bidders* pasan al estado **MOVING**, en el cual se toma la nueva frontera asignada como objetivo de exploración y se navega hacia ella.

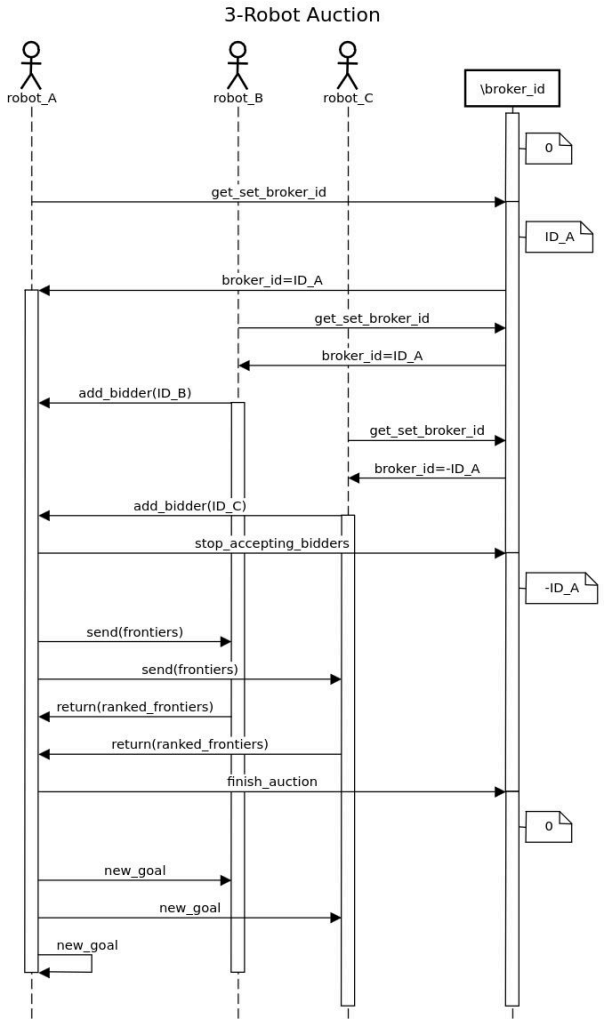


Figura 6: Subasta entre 3 robots. A es *broker*, B y C son *bidders*. El tópico */broker_id* se usa para comunicar el identificador del *broker*.

Al alcanzar el objetivo o detectar algún error en la navegación, se pasa al siguiente estado (**IN GOAL**) para organizar una nueva subasta.

V. Resultados

V-A. Cantidad de Robots

En esta sección se presentan los resultados obtenidos en un entorno simulado cuando se varía el tamaño de la flota. El objetivo principal de esta sección es verificar que la implementación propuesta reduce los tiempos de generación de mapas a medida que se aumenta el tamaño de la flota. Para ello, se varió la cantidad de robots que componen la flota (1-4) y se realizaron 10 mapas con cada una de estas flotas. Los resultados se registran en el Cuadro I, mientras la gráfica de la Fig.7 presenta los valores promedios. Los resultados muestran que el aumento del tamaño de la flota reduce el tiempo de exploración promedio en forma decreciente, sugiriendo que para cada entorno podría haber un número máximo óptimo (ganancia nula) por encima de la cual el agregado de un nuevo robot degradaría el tiempo de exploración (ganancia negativa).

Cantidad de robots	Tiempo mínimo (s)	Tiempo máximo (s)	Tiempo promedio (s)
1	220	416	325
2	216	310	248
3	152	231	196
4	147	213	183

Cuadro I: Tiempo de mapeo.Velocidad individual igual a $0{,}5\ m/s$. Área igual a $85\ m^2$.

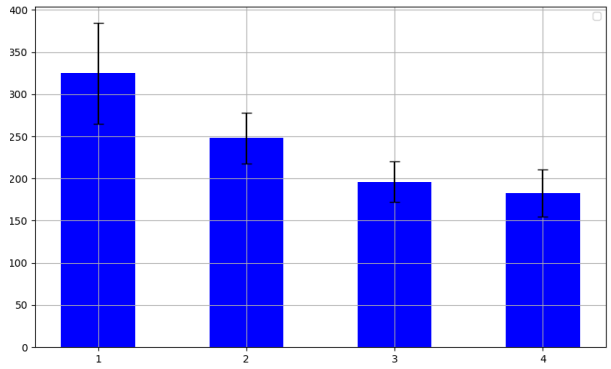


Figura 7: Tiempo de exploración promedio variando el tamaño de la flota (abscisas: #robots; ordenadas: tiempo (s).

Este mismo fenómeno se documenta en [12], incluso allí se considera que es probable que para cierto tamaño de flota, el tiempo empeore. Esto se debe a que al aumentar la cantidad de robots se pueden producir colisiones, atascamientos e incluso, los algoritmos de distribución de tareas deben computar mayor cantidad de información.

V-B. Mapping

Debido a que, en último término, el objetivo individual de cada robot es mapear el entorno que lo rodea, esta sección se propone validar y cuantificar la precisión del mapa obtenido al término de una exploración en un entorno real. Se construyó un mapa de la planta baja del Instituto de Computación [13] para el cual se cuenta con mapas de referencia. Con el fin de cuantificar los errores del mapa generado, se compararon distintas medidas en ambos mapas, las cuales se muestran en la Fig.8 y presentan en el Cuadro II. Teniendo en cuenta todas las medidas, se obtuvo un error cuadrático medio de $RMSE = 7{,}22\ cm$. Este error representa un 14 % y 0.3 % de la menor distancia medida (47 cm) y la mayor distancia medida (22.2 m), respectivamente. Se puede concluir que el error porcentual disminuye a medida que aumentan las distancias.

Número medida	Medición (cm)	Mapa (cm)	Diferencia porcentual (%)
1	151	160	6
2	2220	2220	0
3	282	290	3
4	179	190	6
5	177	190	7
6	47	50	6
7	181	190	5

Cuadro II: Comparación entre las medidas de referencia del plano y las obtenidas del mapa generado.

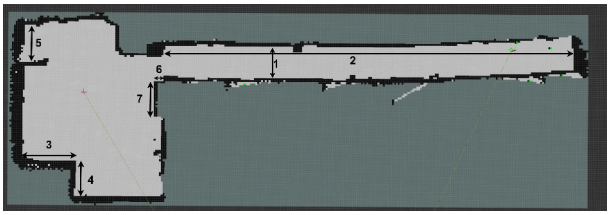


Figura 8: Mapa generado con referencia de medidas reales.

VI. Conclusiones y Trabajo futuro

A partir de una plataforma robótica de bajo costo se ensambló y programó una flota de robots omnidireccionales con capacidades de exploración autónoma probada en entornos simulados y reales. La combinación de paquetes importados y código propio demostró un desempeño satisfactorio donde se destacan desde funcionalidades básicas como una navegación segura hasta la precisión del mapa final y la distribución de tareas para realizar la exploración de forma colaborativa y eficiente. En relación a la distribución de tareas se destaca la implementación de un mecanismo de subasta distribuida que -hasta donde sabemos- es original y aporta robustez sin pérdida de eficacia respecto a las versiones centralizadas. Como posibles direcciones de trabajo a futuro se visualiza la ampliación de la flota y realización de pruebas en entornos reales más extensos, variando el tiempo de espera para habilitar las subastas, así como la utilización de otras representaciones del entorno (p.e. diagramas de Voronoi).

Referencias

[1] Y. Tian, Y. Chang, F. Herrera Arias, C. Nieto-Granda, J. P. How, and L. Carlone, "Kimera-multi: Robust, distributed, dense metric-semantic slam for multi-robot systems," *IEEE Transactions on Robotics*, vol. 38, no. 4, pp. 2022–2038, 2022.

[2] CICEA. (2023) Programando robots jugando con el entorno. UdelaR. [En linea]. Disponible en: http://www.robotito.cicea.ei.udelar.edu.uy/node/10

[3] (2024) Odroid-n2/n2plus. ODROID Wiki. [En linea]. Disponible en: wiki.odroid.com/odroid-n2/odroid-n2

[4] (2024) Rplidar a1. SLAMTEC. [En linea]. Disponible en: www.slamtec.ai/product/slamtec-rplidar-a1

[5] S. Macenski, T. Foote, B. Gerkey, C. Lalancette, and W. Woodall, "Robot operating system 2: Design, architecture, and uses in the wild," *Science Robotics*, vol. 7, no. 66, 2022.

[6] R. Ercoli, F. Navadian, and J. Urrisa. (2023) Icre-ros2. Facultad de Ingeniería, UdelaR. [En linea]. Disponible en: gitlab.fing.edu.uy/fausto.navadian/icre-ros2

[7] (2024) Forza power bank. Forza Power Technologies. [En linea]. Disponible en: www.forzaups.com/es/productos/interna/DC-140USB-esp/

[8] SLAMTEC. (2024) Slamtec lidar ros2 package. Shanghai Slamtec Co. [En linea]. Disponible en: https://github.com/Slamtec/rplidar_ros/tree/ros2

[9] A. Athukorala. (2024) Slam toolbox for lifelong mapping and localization in potentially massive maps with ros. [En linea]. Disponible en: https://github.com/acachathuranga/slam_toolbox/tree/multirobot_ros2

[10] B. Yamauchi, "A frontier-based approach for autonomous exploration," in *IEEE CIRA'97.*, 1997, pp. 146–151.

[11] ——, "Frontier-based exploration using multiple robots," in *Proceedings of the Second International Conference on Autonomous Agents*, ser. AGENTS '98, 1998, p. 47–53.

[12] M. V. Díaz, S. Robaudo, M. Marzoa, and F. Benavides, "Cooperative indoor exploration on affordable robots," in *Brazils's LARS, SBR and WRE*, 2022, pp. 235–240.

[13] (2024) Instituto de computación, facultad de ingeniería, udelar, uruguay. Google Maps. [En linea]. Disponible en: https://maps.app.goo.gl/8NDBVzFGeNKUeziD6