Software de Control de Misión para Operaciones Robóticas de Largo Plazo basado en OpenMCT y ROS2

Bruno De Cruz[1], Claus Rosito[2],
[1] Ingeniería en Inteligencia Artificial, y [2]Laboratorio de Inteligencia Artificial y Robótica.
Universidad de San Andrés
Email: linar@udesa.edu.ar

***Resumen*—Se presenta una solución integral de software capaz de gestionar la operación de misiones robóticas de largo plazo, tomando como caso de estudio una embarcación autónoma para monitoreo de espejos de agua. Esta solución facilita el control remoto, la planificación de misiones, el monitoreo en tiempo real, la interacción con la carga útil, la actualización de software/firmware, el almacenamiento y análisis de datos, y la supervisión de la performance del sistema, basándose en OpenMCT y ROS2.**

I. Introducción

ROS2 ha demostrado ser confiable para experimentos de robótica de corta duración [1], pero su integración en operaciones continuas 24/7 representa un desafío considerable. Aunque existen enfoques parciales en código abierto para abordar esta cuestión, aún no se ha desarrollado una solución integral que cumpla con los rigurosos requisitos de robustez y continuidad operativa necesarios para aplicaciones productivas a largo plazo. En particular, el presente trabajo se basa en un caso de uso derivado de misiones de largo plazo de monitoreo de espejos de agua y aplicaciones de robótica acuática por medio de catamaranes autónomos [2].

II. Diseño

El diseño se centró a partir de un conjunto de requerimientos que se fijaron teniendo en cuenta la operación continua de misiones robóticas de largo plazo. El sistema requerido debe estar equipado con una interfaz intuitiva que habilite al operador a planificar y gestionar misiones, tanto retrospectivas como futuras, facilitando la emisión de comandos en tiempo real y la generación de informes personalizables sobre el estado de la embarcación. Es imprescindible que el sistema permita la interacción con la carga útil a bordo, como sistemas de cámaras, y garantice el registro continuo de datos de telemetría, su almacenamiento y disponibilidad para análisis posteriores. Además, debe operar de manera ininterrumpida (24/7) en un servidor, asegurar la accesibilidad remota y establecer comunicación con la embarcación a través de un puerto serie conectado a un radio modem, incorporando capacidades para monitorear la performance de sus componentes y alertar sobre posibles incidencias a los operadores fuera de sitio. Todo esto no es posible utilizando software estándard de ROS2 como por ejemplo rviz2 [1].

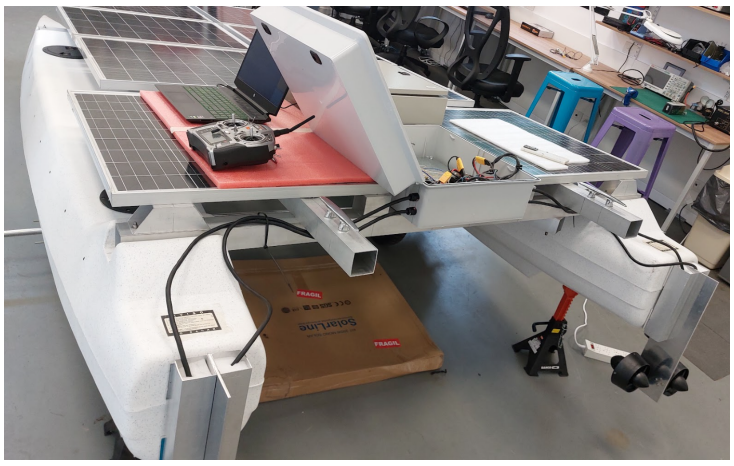

Figura 1. Sistema bajo prueba.

A partir de estos requerimientos se realizó una prospección de soluciones libres y comerciales disponibles, y se realizó una matriz de verificación de requerimientos (ver tabla II). Como conclusión de esta prospección se determinó que ninguna solución actualmente disponible verifique todos los requerimientos, pero se identificó la solución de código abierto más cercana a verificar nuestros requerimientos para proceder a adaptarla y modificarla de modo que pase a verificar todos los requerimientos planteados. Se descartaron versiones con licencias comerciales debido al elevado costo.

Open Mission Control Technologies (Open MCT) [3] es un marco de control de misiones de código abierto que permite la visualización de datos en dispositivos de escritorio y móviles. Desarrollado por el Centro de Investigación Ames de la NASA, Open MCT se utiliza para el análisis, la visualización, la operación y el soporte de misiones espaciales. Su sistema de plugins extensible facilita la integración con sistemas terrestres existentes y la adaptación a múltiples misiones, así como a aplicaciones no espaciales. Este último caso de uso es el que se tomó como base para el presente desarrollo y adaptación, contemplando también el entorno en el que correrá y desarrollando los componentes necesarios para una fluida comunicación con el hardware del robot mediante nodos de ROS2 corriendo a bordo de la computadora de la embarcación y en microcontroladores dedicados a controlar los motores por medio de micro-ROS [4]. Para la implementación se siguieron prácticas de arquitectura de alta disponibilidad como micro-servicios, contenedores y herramientas de orquestación.

III. Simulación

Para comenzar el desarrollo se utilizó como robot un TurtleBot 3 simulado en Gazebo [10], una plataforma compatible con ROS, para garantizar la estabilidad y reducir la complejidad de las pruebas iniciales. Se siguieron los pasos de la documentación oficial de TurtleBot 3 para configurar el entorno virtual, que proporcionó una representación precisa del mundo real.

En este entorno simulado, se llevaron a cabo pruebas iniciales y exploraciones del sistema, refinando funcionalidades básicas como el control de movimiento y la adquisición de datos de sensores mediante ROS y los nodos de relay y la integración con OpenMCT en desarrollo. Esto permitió identificar desafíos y limitaciones tempranas para su mejora continua.

Conforme el desarrollo progresaba, se trasladó a un entorno de trabajo con OpenManipulator [11] simulado, un manipulador robótico compatible con ROS. Esto habilitó tareas más complejas,

Funcionalidad	OpenMCT	QGround Control [5]	Mission Planner [6]	Movai [7]	OpenC3 [8]	YAMCS [9]
Planificar Misión	X	✓	✓	X	X	X
Misiones pasadas y futuras	X	✓	✓	X	✓	✓
Comandos en tiempo real	X	✓	✓	X	✓	✓
Generar reportes	✓	X	!	X	X	X
Interactuar con cámaras	✓	!	!	X	X	✓
Registro de telemetría	!	!	✓	X	✓	✓
Correr 24/7	✓	✓	✓	✓	✓	✓
Ser accesible remotamente	✓	✓	✓	✓	✓	✓
Comunicarse por medio de un puerto	✓	✓	✓	✓	✓	✓
Monitorear performance de componentes	!	!	✓	X	✓	✓
Alertar en caso de fallas	✓	✓	✓	X	✓	✓
Posibilidad de configuración del sistema	✓	✓	✓	✓	✓	✓
Agnóstico al hardware	✓	✓	!	✓	X	✓
Código abierto	✓	✓	✓	X	✓	✓

Cuadro I
COMPARACIÓN DE FUNCIONALIDADES.

como la simulación de recepción de imágenes de una cámara a través de ROS 2, esencial para implementar funcionalidades avanzadas como el procesamiento de imágenes y la interacción autónoma con el entorno.

IV. DESARROLLO

El desarrollo comenzó con un diagrama del flujo de información, estableciendo un protocolo de comunicación bidireccional entre el sistema y el robot. Se desarrolló un node que se suscribe a los topics del robot y los los transmitía vía websocket y un servicio que procesa el flujo de datos a medida que son recibidos, haciéndolos disponibles dentro del OpenMCT.

En paralelo, se ha desarrollado un sistema de alertas en tiempo real que evalúa condiciones específicas en los mensajes recibidos, como valores inferiores a 0. Estas alertas, tanto auditivas como visuales en color rojo, proporcionan una respuesta proactiva ante posibles fallos, asegurando así la integridad y el rendimiento del sistema.

Además, se ha implementado un proceso para generar reportes automáticos. Un botón en la interfaz de usuario permite acceder a los datos almacenados en el servidor de historial, obteniendo valores máximos, mínimos, promedios de la telemetría y tiempos de error en un archivo CSV, facilitando el análisis y la toma de decisiones.

Para simplificar la interacción con el sistema, se ha desarrollado una consola desplegable que permite controlar el movimiento del robot en la simulación de TurtleBot3 mediante teclas WASD, proporcionando una experiencia intuitiva y eficiente.

Otro aspecto crucial es la geolocalización, para la cual se ha integrado un mapa base de OpenStreetMap en el sistema. Esto permite visualizar la ubicación geoespacial de los elementos involucrados en las operaciones del robot, recibiendo información de ROS 2 a través de websockets.

Finalmente, la representación visual del entorno circundante del robot se ha logrado recibiendo imágenes en formato RGB8 y aplicando técnicas de procesamiento de imágenes para optimizar la calidad visual en la simulación de TurtleBot3.

V. RESULTADOS EXPERIMENTALES

A la fecha presente se realizaron varios ensayos navegando un catamarán autónomo (ver Fig. 1) y se verificó la eficacia de tres funcionalidades específicas: recepción de datos, generación de alertas y generación de informes. Se midieron el consumo (corriente y tensión) de los motores de la embarcación, la temperatura y la humedad, acelerando y desacelerando para observar las distintas variaciones en los datos reportados (ver Fig. 2 y Fig. 3).

Se pudo confirmar que estas tres funcionalidades operaban correctamente y presentaban variaciones con cierto retraso, el cual fue subsanado durante el transcurso de la prueba mediante ciertas optimizaciones. Se logró visualizar la información en tiempo real, recibir las alertas pertinentes en el momento indicado y los reportes funcionaron de manera correcta, mostrando los datos esperados, cuyos valores fueron validados por instrumental de laboratorio.

VI. CONCLUSIONES Y TRABAJO FUTURO

Se desarrolló una solución de control y monitoreo de robots móviles basado en ROS2 y OpenMCT que viene a cubrir un nicho en cuanto a misiones robóticas de largo plazo. La solución es modular y permite seguir variables de interés, generar alertas, y enviar comandos, siguiendo prácticas de diseño para alta disponibilidad. El sistema ofrece la posibilidad de capturar, reportar y analizar diversos datos de manera efectiva e intuitiva, facilitando la optimización, mejoras y detección de fallos en sistemas subyacentes. Se validó su funcionamiento con un robot acuático y se generaron reportes que fueron consistentes con mediciones por medio de instrumental de laboratorio. Estos resultados brinda un futuro prometedor para potenciales aplicaciones en proyectos de robótica basados en ROS2.

Actualmente se está trabajando en la modularización en microservicios utilizando contenedores para mejorar la portabilidad y modularidad, en simplificar la parametrización de los nodos, tópicos y alertas, y en validar la alta disponibilidad. Como trabajo futuro queda validar la operación del sistema en casos donde la conección es inestable o de bajo ancho de banda, poner a prueba las funcionalidades restantes y contrastar su aplicabilidad en otros campos, como el terrestre o el aéreo, y en casos de uso multirobot.

REFERENCIAS

[1] T. R. . D. Team, "Ros 2 documentation," https://docs.ros2.org/, 2024.
[2] LINAR, "Catamarán guazú: Catamarán autónomo para monitoreo de espejos de agua y aplicaciones de robótica acuática," *Jornadas Argentinas de Robótica*, 2025.
[3] N. S. Catalog, "Open mission control technologies (open mct)(arc-15256-1d)," 2024. [Online]. Available: https://software.nasa.gov/
[4] T. micro ROS Development Team, "micro-ros," https://github.com/micro-ROS/, 2024.
[5] Q. Developers, "Qgroundcontrol," https://qgroundcontrol.com/, 2024.
[6] M. P. Developers, "Mission planner," https://ardupilot.org/planner/, 2024.
[7] M. AI, "Move ai," https://www.mov.ai/, 2024.
[8] O. Developers, "Openc3 - open control component compiler," https://openc3.org/, 2024.
[9] Y. Developers, "Yet another mission control system (yamcs)," https://yamas.github.io/, 2024.
[10] R. O. S. (ROS), "Gazebo robot simulator," 2024. [Online]. Available: http://gazebosim.org/
[11] O. Community, "Openmanipulator," https://www.openmanipulator.com/, 2024.

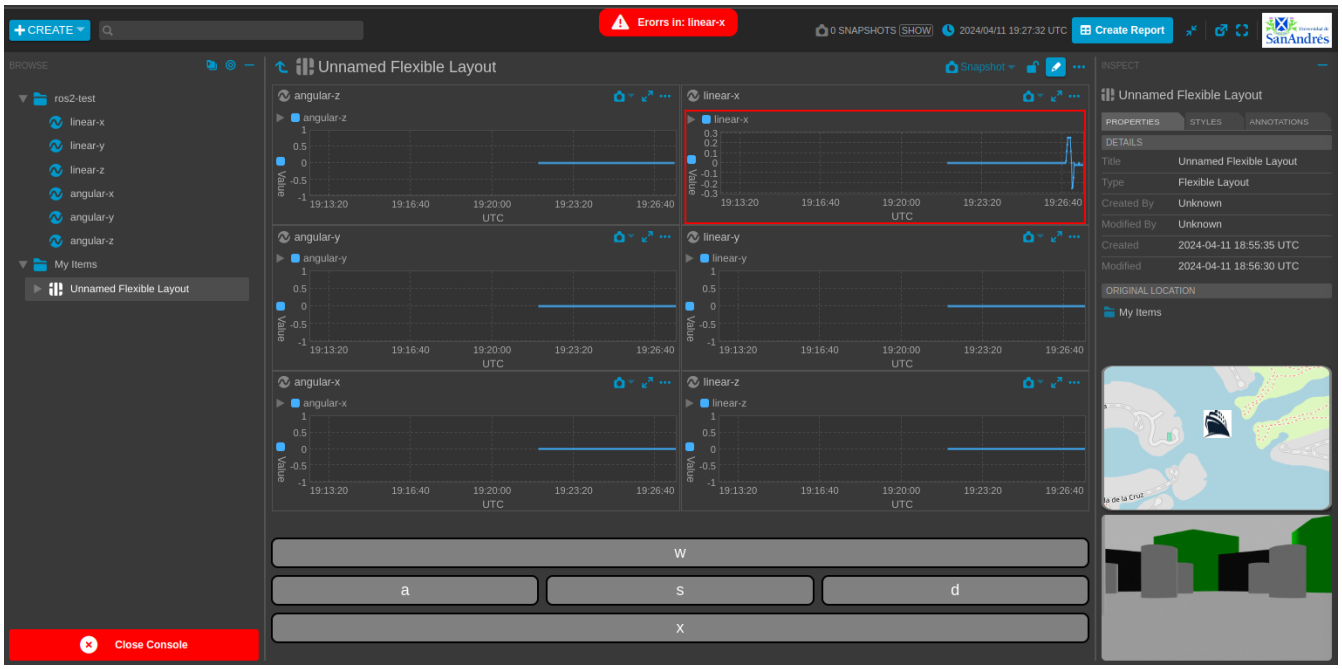

Figura 2. Vista de la Interfaz en la última versión probada en simulación.

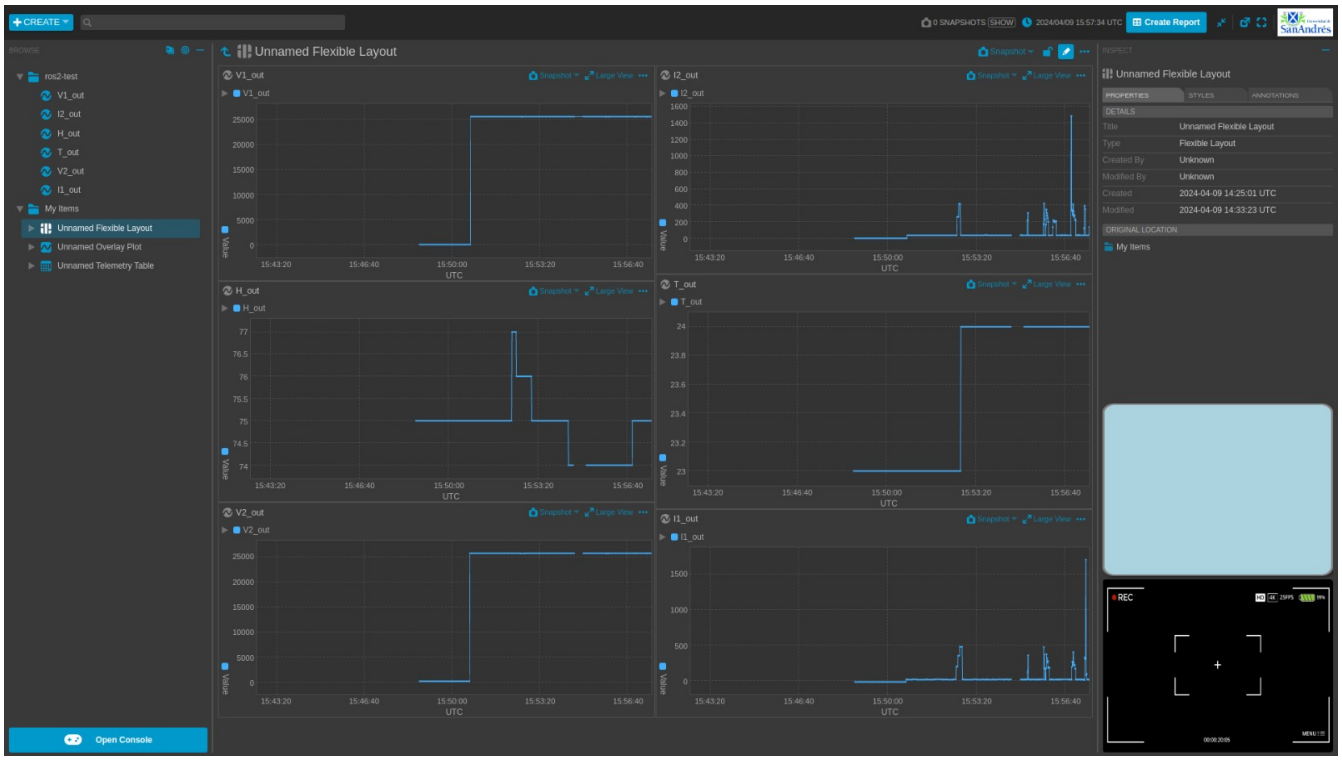

Figura 3. Algunos de los datos obtenidos en pruebas de campo.