XLIV Jornadas de Automática 2023

https://jautomatica.es

Desarrollo de un HMI en ROS-MATLAB para la teleoperación de robots industriales

Prieto-Fernández, Natalia[a], Bayón-Gutiérrez, Martín[a,*], Fernández-Blanco, Sergio[b], Fernández-Blanco, Álvaro[c], Carro-De-Lorenzo, Francisco[d], Benítez-Andrades, José Alberto[e]

[a]*Grupo de Investigación SECOMUCI, Departamento de Ingeniería Eléctrica y de Sistemas y Automática, Universidad de León, Campus de Vegazana s/n, 24071, León, España*
[b]*Departamento de Ingeniería Eléctrica y de Sistemas y Automática, Universidad de León, Campus de Vegazana s/n, 24071, León, España*
[c]*Universidad Complutense de Madrid, Pl. de Ciencias, 28040, Madrid, España*
[d]*TRESCA Ingeniería S.A. León, España*
[e]*Grupo de Investigación SALBIS, Departamento de Ingeniería Eléctrica y de Sistemas y Automática, Universidad de León, Campus de Vegazana s/n, 24071, León, España*

To cite this article: Prieto-Fernández, N., Bayón-Gutiérrez, M., Fernández-Blanco, S., Fernndez-Blanco, A., Carro-De-Lorenzo, F., Benítez-Andrades, J.A. 2023. ROS-MATLAB HMI for industrial robot teleoperation. XLIV Jornadas de Automática, 720-725. https://doi.org/10.17979/spudc.9788497498609.720

Resumen

ROS (Robot Operating System) es un framework para el desarrollo de sistemas robóticos de código abierto ampliamente utilizado en la industria y la investigación. Los sistemas robóticos asistidos requieren una comunicación fluida y fiable entre ROS y el controlador externo, a fin de lograr un control y supervisión eficaz del mismo. En este artículo, se presenta el procedimiento necesario para establecer la comunicación a través de una red local entre un robot móvil que implementa ROS y un dispositivo remoto desde el que enviar y recibir información del robot. Además, se propone un ejemplo de HMI (Interfaz Hombre-Máquina) desarrollado en MATLAB, y que puede ser instalada en equipos Windows, Linux o MacOSX, para la teleoperación de un robot en un entorno industrial. La comunicación bidireccional en tiempo real, las capacidades de procesamiento de datos y su versatilidad la convierten en una herramienta completa para la gestión de datos robóticos en entornos industriales.

Palabras clave: Interacción Hombre-Máquina, Mapeado, Modelado y control de sistemas robóticos en red, Navegación guiada, Teleoperación

ROS-MATLAB HMI for industrial robot teleoperation

Abstract

ROS (Robot Operating System) is an open-source robotics development framework widely used in industry and research. Human assisted robotic systems require of robust and constant communication between ROS and the remote controller in order to achieve effective control and monitoring of the system. In this paper, we propose a methodology to connect a mobile ROS with a remote device, using a wireless local network. This device serves as link between the robot and the rest of the system. Additionally, an HMI (Human-Machine Interface) developed using MATLAB is presented. The HMI can be implemented in Windows, Linux or MacOSX computers and provides teleoperation capabilities for a robot in an industrial environment. Real-time bidirectional communication, data processing capabilities and intrinsic versatility make this HMI a robust tool for robotic data management in industrial applications.

Keywords: Guidance navigation and control, Human and vehicle interaction, Map building, Networked robotic system modeling and control, Telerobotics

*Autor para correspondencia: martin.bayon@unileon.es

1. Introducción

Los sistemas robóticos, sean autónomos o asistidos, requieren importantes recursos en capacidad de computo y energía, siempre escasos a bordo de vehículos pequeños alimentados por batería. Transferir la inteligencia a una unidad remota reduce consumo y potencia de cálculo a bordo, aumentando la autonomía, pero requiere un sistema de comunicaciones confiable. Transferir inteligencia del móvil a tierra permite utilizar recursos fijos menos sujetos a restricciones aunque genera dependencia de la red de comunicaciones. Son posibles diferentes soluciones intermedias de reparto de carga en función de las capacidades disponibles tanto a bordo como en tierra. En este sentido, la implementación de una interfaz hombre-máquina que gestione y procese los datos procedentes del robot puede liberar de carga al controlador a bordo de la unidad robótica. Muchas de estas interfaces resultan complejas para el usuario final e incluso suponen la dedicación exclusiva de un equipo a dicha interfaz debido a la carga computacional y el entorno de programación utilizado. Estos hechos restringen el acceso al usuario final.

Gran parte de los robots se pueden programar en entorno ROS (Mokaram et al., 2017; Vivas and Sabater, 2021; Baklouti et al., 2021), un entorno complejo que requiere de conocimientos de robótica. Como alternativa surge la integración ROS con entornos más intuitivos que permitan el desarrollo de interfaces, uno de ellos es MATLAB. La integración ROS-MATLAB ya se ha llevado a cabo en ámbitos de investigación (Avanzato, 2020) o educativos (Hold-Geoffroy et al., 2013; Rosillo et al., 2020) pero con una mínima implementación en entornos industriales. ROS-MATLAB puede jugar un papel importante a la hora de controlar brazos robóticos de manera precisa, planificar rutas óptimas en almacenes automatizados o bien monitorizar en tiempo real robots autónomos en entornos hostiles. El desarrollo de interfaces intuitivas que faciliten al usuario introducir los parámetros del proceso industrial correspondiente, sin requerir conocimientos previos de robótica, agiliza el proceso. Todas estas tareas requieren del conocimiento del mapa más preciso posible del entorno. La incertidumbre del mismo condicionará la calidad de la actividad desempeñada por dicha unidad robótica.

Existen diferentes sistemas SLAM (Simultaneous Localization And Mapping) que se pueden implementar en entorno ROS. *HectorSLAM, Cartographer, KartoSLAM o Gmapping* son algunos de los métodos SLAM más utilizados (Filipenko and Afanasyev, 2018; Santos et al., 2013). En este proyecto se ha optado por el sistema *Cartographer* debido a la robustez del mismo en la reconstrucción del mapa bidimensional a partir de datos LiDAR 2D (Yagfarov et al., 2018). Es un sistema que hace frente al problema del cierre de bucle típico en SLAM. Es capaz de identificar lugares que el robot ya ha visitado previamente y de minimizar el error acumulativo en la estimación de su posición.

El objetivo principal de este artículo se enfoca en el desarrollo de una interfaz HMI (Interfaz Hombre-Máquina) intuitiva y accesible, con el propósito de facilitar tanto la adquisición de datos del robot como el envío de instrucciones en tiempo real. Se busca liberar de carga computacional al controlador del robot, ofreciendo al usuario una interacción amigable y eficiente con el mismo, además de brindar información actualizada sobre el proceso SLAM. Para lograr este propósito, se ha optado por desarrollar el HMI en MATLAB. La sencillez de la interfaz HMI presentada garantiza que cualquier usuario pueda utilizar de forma correcta el robot, a pesar de carecer de conocimiento avanzado sobre robótica o ROS.

2. Descripción del Sistema Robótico

La plataforma móvil empleada en los ensayos se corresponde con el modelo comercial *Rover Robotics 4WD Rover Zero 3*. Esta unidad en sí misma no posee la capacidad de percepción y procesamiento del entorno circundante. Para implementar su carácter robótico se han incorporado a la misma un conjunto de sensores y un mini PC. Esta integración posibilita la realización de las tareas de mapeo y posicionamiento en tiempo real, propias de SLAM.

Los sensores empleados para la percepción más real posible del entorno son una cámara de profundidad y un sensor LiDAR 2D. La cámara seleccionada es la *Intel® RealSense™ D435i* y el sensor LiDAR es el modelo *Slamtec RPLiDAR S2*. Estos sensores se encuentran conectados al mini PC *Intel NUC10i*, encargado de procesar dicha información utilizando el framework ROS.

La distribución de los diferentes elementos que componen el sistema robótico puede verse en la Figura 1. La plataforma móvil alberga los sensores y el mini PC. Los sensores se sitúan en la parte superior donde pueden percibir el entorno, sin que elementos constructivos del propio robot impidan su completa visibilidad, mientras que el mini PC se encuentra oculto en el interior de la plataforma.

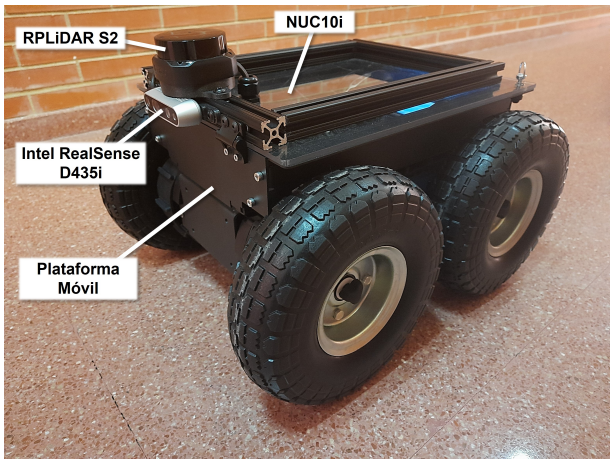


Figura 1: Configuración del sistema robótico.

En este trabajo, la plataforma robótica utilizada no incorpora funciones de navegación autónoma sino que es necesaria la intervención de un operador, encargado de la supervisión y control del robot. Este robot incorpora un controlador Bluetooth de tipo Joystick, que actúa como interfaz de control entre el operador y el robot industrial. Sin embargo debido al limitado alcance de la conexión Bluetooth, así como las interferencias que pudieran encontrarse en entornos industriales, resulta necesario la implementación de un sistema de comunicación inalámbrico alternativo, que permita superar estas limitaciones. Con este

propósito, este artículo propone la creación de un HMI que implemente un joystick virtual para el control del robot industrial mediante la conexión a una red local inalámbrica.

La finalidad de este robot es la de generar un mapa bidimensional del entorno industrial por el que navegue, al mismo tiempo que calcula su posición en el mismo en todo momento. Para ello, en este trabajo hemos implementado el sistema *Cartographer-SLAM* (Hess et al., 2016). El sistema SLAM, así como los sistemas de percepción y cálculo de trayectorias se ejecutan de forma local en el mini PC incluido en el robot industrial, lo que evita una sobrecarga en la red de comunicaciones.

3. Integración ROS-MATLAB

Para la gestión de todos los procesos concurrentes del robot, el mini PC incorpora una distribución Ubuntu 18.04, sobre la que se ha instalado el framework ROS Melodic. Uno de los principios de funcionamiento de ROS es el uso de un sistema de comunicación *publisher/subscriber*, según el cuál, cada proceso del sistema robótico (denominado *nodo* en ROS) puede enviar o recibir datos utilizando distintos canales de comunicación (denominados *topics* en ROS). En ambos casos, cada nodo tan solo es responsable de adecuar la información que desea enviar/recibir al tipo de mensaje adecuado para cada topic, con independencia del uso que otros nodos vayan a hacer de dicha información. Es decir, ningún nodo conoce qué otros nodos van a recibir la información que ha enviado a través de un topic. Este tipo de comunicación simplifica la integración en ROS de nuevos componentes, funcionalidad que aprovechamos en este trabajo para presentar una interfaz HMI que se comunica con el software incluido en el robot de forma sencilla y transparente para cualquier otro proceso que se pueda ejecutar en este.

La comunicación entre el sistema robótico y el equipo en el que se ejecuta la interfaz HMI se realiza por medio de una conexión Wifi exclusiva para los componentes del sistema presentado, de forma que se minimice la carga en la red y se garantice la correcta conexión de ambos equipos en todo momento.

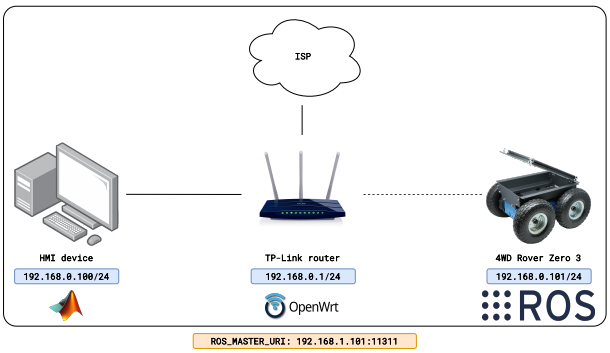


Figura 2: Esquema de conexión entre el equipo que ejecuta el HMI y el robot industrial.

Este tipo de conexión ofrece un mayor alcance y robustez respecto a la conexión Bluetooth del joystick de teleoperación que incorpora el robot industrial originalmente.

Nuestra red local está formada por un router *TP-Link TL-WR1043ND*, encargado de generar una red Wifi a la que se conecta el robot industrial. El router está además conectado mediante una conexión Ethernet con el equipo que implementa la interfaz HMI y con el proveedor de servicios de internet. Las direcciones IP de todos los equipos han sido asignadas de forma estática de acuerdo al diagrama de la Figura 2.

La integración del sistema robótico con la interfaz HMI desarrollada se realiza por medio de la herramienta *ROS Toolbox* (MathWorks, 2019) de MATLAB, que permite una correcta conexión con los *topics* disponibles en el equipo que ejecuta ROS, para el envío y recepción de mensajes. El fragmento de código 1 presenta un ejemplo del uso de esta herramienta para la recepción y envío de mensajes.

```
%Connection to the device running ROS master
rosinit("hhtp://192.168.0.101:11311/");

%We register as a publisher for the topic
movecmdTopic = rospublisher("/movement_command",
    "struct");
%We get the message type from the topic
moveMessage = rosmessage(movecmdTopic);
%We create a movement message
moveMessage.Linear.X = 1.0;
moveMessage.Angular.z = 0.0;
%The message is sent through the topic
send(movementCommandTopic, movementMessage);

%We register as a subscriber for the map topic
mapTopic = rossubscriber("/slam_pointcloud_map");
%We get a message sent through the map topic
newMapMessage = receive(mapTopic);

%Lastly, we finish the ROS communication
rosshutdown
```

Listing 1: Código de ejemplo para la integración de ROS y MATLAB

De este modo, la interfaz HMI desarrollada en MATLAB puede comunicarse con el robot industrial a través de la red inalámbrica.

En este trabajo, los *topics* de interés para la teleoperación del robot, así como la conexión de estos con los distintos nodos de nuestro sistema se exponen en la Figura 3, en la que se representa con color azul aquellos nodos correspondientes a componentes físicos del robot industrial (como sensores, actuadores y elementos de control), con color rojo a los *topics* de comunicación entre los distintos nodos, con color morado a los nodos que se ejecutan en el robot industrial, y en color verde a la interfaz HMI presentada en este trabajo, y que se ejecuta en el equipo remoto. El sentido de las flechas del diagrama indica la relación *publisher/subscriber* entre los distintos elementos.

4. HMI

El diseño y desarrollo del HMI se ha realizado íntegramente utilizando la herramienta *app designer* de MATLAB, adecuada por su versatilidad para distribuir visualmente los diversos elementos que conforman la interfaz y su capacidad para programar su funcionalidad (MathWorks, 2016). En el diseño se ha puesto especial enfoque en la integración ROS-MATLAB mencionada previamente, considerando siempre la necesidad de trabajar en tiempo real tanto en la recepción de información de los sensores como en el envío de órdenes de movimiento al robot.

La disposición de las diferentes secciones que componen el diseño del HMI se presenta de manera visual en la Figura 4. La *barra de herramientas* constituye una de las partes esenciales de la interfaz, ya que permite acceder a la página principal,

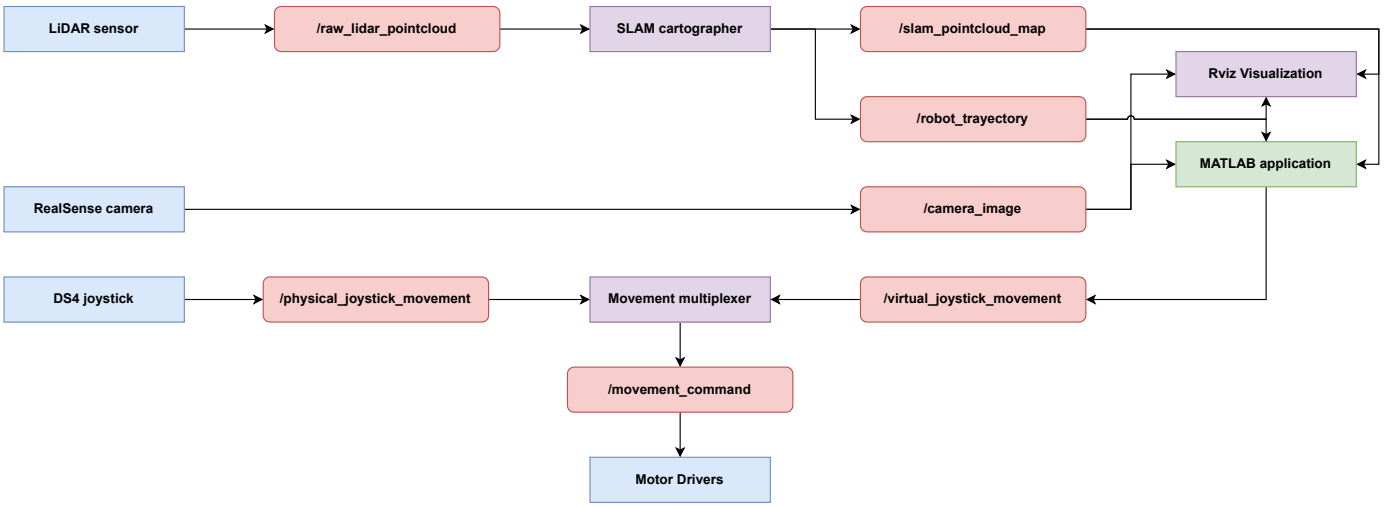


Figura 3: Esquema de comunicación entre los distintos componentes del sistema robótico.

al análisis de los datos en diferido, así como a la consulta de ayuda.

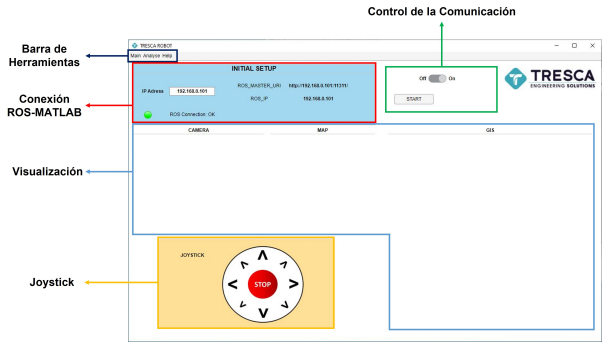


Figura 4: Secciones funcionales del HMI.

La comunicación entre ROS y MATLAB se logra gracias a la sección de *conexión ROS-MATLAB* del HMI. En esta sección, el usuario debe indicar la dirección IP del robot al que quiere acceder remotamente, tal y como se refleja en la Figura 2. Una vez establecida correctamente dicha conexión, se habilita la sección de *control de la comunicación*, que permite activar o desactivar la recepción de información procedente del robot. Esta información se mostrará en tres imágenes distribuidas en la sección de *visualización*. La primera imagen, de izquierda a derecha, mostrará la imagen de la cámara, la segunda presentará tanto el mapa SLAM como la trayectoria seguida por el robot en coordenadas cartesianas, mientras que la tercera superpondrá al GIS (Sistema de Información Geográfica) la información anterior en coordenadas geográficas. Estas imágenes se actualizarán en tiempo real a medida que el robot se desplace por el entorno. Para la representación del mapa GIS se ha optado por utilizar la base de mapas proporcionada por la empresa Esri. La teleoperación del robot es posible gracias al joystick virtual incorporado en el HMI. Al hacer clic en las flechas, el robot se moverá en la dirección indicada una distancia predeterminada. En cualquier momento se puede detener su movimiento presionando el botón *STOP* del joystick.

Una vez finalizada la adquisición de datos deseada, el HMI desarrollado permite interrumpir la conexión con el robot industrial utilizando el botón *Pause*, tal y como se puede observar en la Figura 5, tras lo cuál se ofrece al usuario la posibilidad de almacenar una copia de toda la información recibida durante la sesión de teleoperación. Por defecto, esta información se guarda utilizando el formato *aaaa_mm_dd-HH_MM_SS.m* correspondiente a la fecha y hora actual. La interfaz HMI permite la reanudación de la comunicación en cualquier momento.

La interfaz desarrollada permite además el análisis de información offline, es decir, en un momento posterior al de su adquisición. Para ello, a través de la barra de herramientas de la aplicación se puede utilizar el botón *Analyse* para importar un archivo previamente almacenado y que permita la visualización del mapa SLAM y GIS correspondiente a la sesión seleccionada.

El HMI diseñado permite intercambiar información con el robot de forma bidireccional. Por un lado, captura los datos de la cámara, el mapa y la trayectoria proporcionados por el entorno ROS. Por otro lado, es capaz de enviar al robot los comandos de movimiento generados por el joystick. Esta comunicación bidireccional entre el HMI y el entorno ROS permite una interacción fluida y en tiempo real, brindando al usuario la capacidad de controlar el robot mediante el joystick al mismo tiempo que recibe información actualizada del entorno y del proceso SLAM.

Con el fin de garantizar la compatibilidad y la portabilidad del HMI en diversos entornos se ha exportado como ejecutable. Para ello, se ha utilizado la herramienta *MATLAB Compiler* (MathWorks, 2015), la cual facilita la exportación de la interfaz como aplicación independiente prescindiendo de la necesidad de contar con una instalación completa de MATLAB en el dispositivo del usuario. El único requisito para la ejecución del HMI es la instalación del MATLAB runtime, compatible con sistemas operativos Windows, Linux y MacOSX, lo que simplifica su implementación y amplía su accesibilidad a los usuarios. Esta característica resulta especialmente valiosa en entornos industriales, donde se busca una solución práctica y eficiente para la interacción entre el usuario y el robot.

5. Resultados Experimentales

La validación y evaluación del HMI se ha llevado a cabo a través de una serie de ensayos que involucraron el mapeo del Edificio Tecnológico de la Escuela de Ingenierías Industrial, Informática y Aeroespacial (segunda fase) de la Universidad de León. Durante todo el proceso, se recopilaron datos de cámara, mapa y trayectoria que fueron analizados tanto en tiempo real como posteriormente. Este enfoque ha permitido verificar el correcto funcionamiento del HMI en condiciones reales y evaluar su desempeño en la adquisición, procesamiento y presentación de los datos de manera eficiente y precisa.

5.1. *Procesamiento en tiempo real*

La Figura 5 ilustra el funcionamiento de la interfaz en tiempo real, donde se destaca la relevancia de la imagen de la cámara para guiar este robot. La utilización de esta cámara posibilita la visualización del entorno por el cual navega el robot, mientras que el usuario brinda asistencia desde un lugar remoto a través del joystick virtual. Cabe destacar que la configuración inicial del robot no contemplaba esta asistencia remota sino que se limitaba a una teleoperación a escasos metros del robot, empleando un joystick físico con conexión Bluetooth.

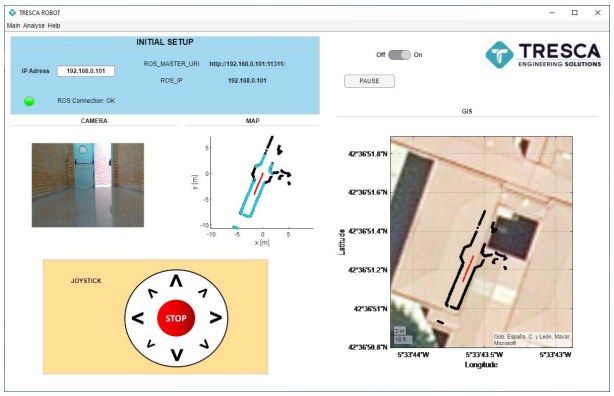


Figura 5: HMI en tiempo real. El mapa se representa en color negro, la trayectoria en rojo y el escaneo LiDAR actual en color azul.

A medida que el robot se desplaza por pasillos y espacios interiores, se va completando el mapa y la trayectoria del mismo.

5.2. *Procesamiento offline*

El HMI desarrollado permite realizar un análisis en diferido de los datos del proceso SLAM. Para facilitar esta funcionalidad, se ha implementado la opción de guardar la información recibida durante la operación. Esta opción permite almacenar tanto el mapa generado como la trayectoria seguida por el robot, en coordenadas cartesianas y geográficas. Una vez que los datos han sido guardados, es posible procesarlos en cualquier momento mediante la función *Analyse* disponible en la barra de herramientas del HMI. Esto brinda a los usuarios la flexibilidad de revisar y analizar los resultados de SLAM en cualquier instante posterior a la adquisición de los mismos.

El mapa resultante en coordenadas cartesianas se presenta en la Figura 6(a). En dicha representación gráfica, se han destacado tres puntos específicos del entorno. Uno de ellos corresponde al origen de coordenadas del mapa, que coincide con el punto de partida de la trayectoria del robot. Los otros dos puntos representan ubicaciones extremas en el mapa, ofreciendo así una estimación de las dimensiones del entorno mapeado. Esta información cartográfica resulta fundamental para comprender y analizar la distribución y la escala del entorno explorado por el robot en el contexto del mapeo.

Por otro lado, la Figura 6(b) muestra el mapa y la trayectoria en coordenadas geográficas sobre imagen de satélite. Esta representación permite una mayor contextualización y comprensión de la posición y movimiento del robot en relación con su entorno geográfico real. Esto facilita la planificación y la toma de decisiones, así como la evaluación de la eficiencia y la efectividad de las tareas realizadas por el robot en dichos entornos. Esta información es de gran importancia en entornos industriales, así como en perfiles con características simétricas, como ocurre en este caso.

Al analizar el mapa final, es importante destacar la eficacia del algoritmo SLAM utilizado. *Cartographer* aborda de manera efectiva el desafío del cierre de bucle y logra corregir el mapa resultante en función de los datos adquiridos. El resultado mostrado en la Figura 6(b) pone de manifiesto la calidad del mapa obtenido al superponerse de manera precisa con la planta del edificio. Esta capacidad es esencial en entornos industriales complejos, donde la precisión y la fiabilidad del mapa son cruciales para la navegación autónoma, optimización de rutas, la planificación de tareas y la interacción segura con el entorno.

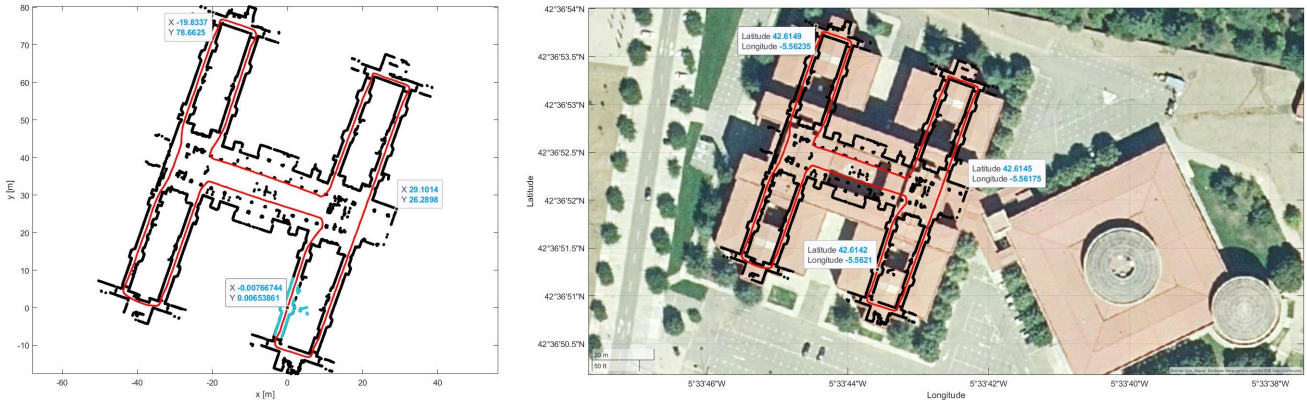
6. Conclusiones

La interacción robot-hombre a nivel industrial presenta una tendencia creciente en la automatización y optimización de procesos industriales. En este contexto, el presente artículo ha abordado el desarrollo de una interfaz Hombre-Máquina altamente funcional y accesible que facilita la interacción entre usuario y robot en entornos industriales. El HMI resultante ha demostrado ser amigable e intuitivo, lo que permite a los usuarios trabajar de manera efectiva en una aplicación desarrollada en MATLAB, evitando la complejidad asociada con el entorno ROS.

La comunicación bidireccional en tiempo real es una característica clave del HMI presentado, asegurando una interacción fluida y eficiente entre el robot y el usuario. Un aspecto fundamental a destacar es la capacidad para procesar y analizar los datos que posee la interfaz tanto en tiempo real como posteriormente. Esto se refleja en la representación visual de los datos analizados, incluyendo el mapa y la trayectoria, sobre GIS. Esta visualización facilita la interpretación del mapa resultante en coordenadas cartesianas y geográficas, proporcionando una percepción más completa del entorno mapeado.

Adicionalmente, el HMI presentado es exportable a diferentes plataformas, como Windows, Linux y MacOSX, lo que le confiere flexibilidad y compatibilidad con distintos equipos. Esta característica amplía su alcance y permite a los usuarios trabajar con la interfaz en el entorno que les resulte más conveniente.

El HMI desarrollado en este artículo ofrece una solución integral para la adquisición, análisis y control de los datos procedentes del robot en entornos industriales. Su interfaz amigable, la comunicación bidireccional en tiempo real, las capacidades



(a) Mapa y trayectoria en coordenadas cartesianas

(b) Mapa y trayectoria en coordenadas geográficas

Figura 6: Mapa resultante de la planta baja del Edificio Tecnológico de la Escuela de Ingenierías Industrial, Informática y Aeroespacial (segunda fase) de la Universidad de León.

de procesamiento de datos y la exportabilidad lo convierten en una herramienta versátil y accesible para el usuario final.

Está previsto, como línea de investigación futura, la evaluación de su desempeño en entornos industriales más agresivos con interferencias por ruido electromagnético e interferencias LiDAR por destellos y polvo en ambiente.

Agradecimientos

Este trabajo ha sido realizado gracias al apoyo de TRESCA Ingeniería S.A. En el marco del Proyecto de Investigación de *Sistemas para la Inspección de Entornos Estructurados Hostiles mediante Sistemas de Localizacion y Mapeo Simultáneo (SLAM) con Tecnología LiDAR*. El proyecto ha sido cofinanciado por el FEDER, Objetivo Temático 1, que busca promover el desarrollo tecnológico, la innovación y la investigación de calidad.

Este trabajo se ha realizado en colaboración con la Universidad de León en el marco del "Programa Propio de Investigación 2021".

Referencias

Avanzato, R. L., 2020. Development of a matlab/ros interface to a low-cost robot arm. In: 2020 ASEE Virtual Annual Conference Content Access. pp. 1–14.

Baklouti, S., Gallot, G., Viaud, J., Subrin, K., 2021. On the improvement of ros-based control for teleoperated yaskawa robots. Applied Sciences 11 (16), 7190.

Filipenko, M., Afanasyev, I., 2018. Comparison of various slam systems for mobile robot in an indoor environment. In: 2018 International Conference on Intelligent Systems (IS). IEEE, pp. 400–407.

Hess, W., Kohler, D., Rapp, H., Andor, D., 2016. Real-time loop closure in 2d lidar slam. In: 2016 IEEE international conference on robotics and automation (ICRA). IEEE, pp. 1271–1278.

Hold-Geoffroy, Y., Gardner, M.-A., Gagné, C., Latulippe, M., Giguere, P., 2013. ros4mat: A matlab programming interface for remote operations of ros-based robotic devices in an educational context. In: 2013 International Conference on Computer and Robot Vision. IEEE, pp. 242–248.

MathWorks, 2015. MATLAB Compiler.
URL: `https://es.mathworks.com/products/compiler.html`

MathWorks, 2016. MATLAB App Designer.
URL: `https://es.mathworks.com/products/matlab/app-designer.html`

MathWorks, 2019. ROS Toolbox.
URL: `https://es.mathworks.com/products/ros.html`

Mokaram, S., Aitken, J. M., Martinez-Hernandez, U., Eimontaite, I., Cameron, D., Rolph, J., Gwilt, I., McAree, O., Law, J., 2017. A ros-integrated api for the kuka lbr iiwa collaborative robot. IFAC-PapersOnLine 50 (1), 15859–15864.

Rosillo, N., Montés, N., Alves, J. P., Ferreira, N. M. F., 2020. A generalized matlab/ros/robotic platform framework for teaching robotics. In: Robotics in Education: Current Research and Innovations 10. Springer, pp. 159–169.

Santos, J. M., Portugal, D., Rocha, R. P., 2013. An evaluation of 2d slam techniques available in robot operating system. In: 2013 IEEE international symposium on safety, security, and rescue robotics (SSRR). IEEE, pp. 1–6.

Vivas, A., Sabater, J. M., 2021. Ur5 robot manipulation using matlab/simulink and ros. In: 2021 IEEE International Conference on Mechatronics and Automation (ICMA). IEEE, pp. 338–343.

Yagfarov, R., Ivanou, M., Afanasyev, I., 2018. Map comparison of lidar-based 2d slam algorithms using precise ground truth. In: 2018 15th International Conference on Control, Automation, Robotics and Vision (ICARCV). IEEE, pp. 1979–1983.