ISBN: 978-628-95207-8-1. ISSN: 2414-6390. Digital Object Identifier: 10.18687/LACCEI2024.1.1.1908

ROS and FPSoC Integration on a Turtlebot 3 Robotic Platform

Jairo Cuero, MEng Automatización Industrial[1], Pedro-F Cárdenas, PhD Automática y Robótica[2], Rony Trespalacios, Ingeniero Electrónico[3]
[1,3]Universidad de los Llanos, Colombia, jairo_cuero@unillanos.edu.co, rony.trespalacios@unillanos.edu.co
[2]Universidad Nacional de Colombia, Colombia, pfcardenash@unal.edu.co

Abstract– ***In this research, we introduce an initial strategy for incorporating Field-Programmable System on a Chip (FPSoC) technology into a Robot Operating System (ROS)-compatible mobile robot, with a focus on the TurtleBot3 platform. The primary objective was to swap the original Raspberry Pi board for an FPSoC board, equipping the robot with a Field-Programmable Gate Array (FPGA) section for enhanced function acceleration through hardware-software co-design. Through a meticulous integration effort, we were able to convert the TurtleBot3 into a working prototype that maintained its essential functionalities while leveraging the advanced capabilities of function acceleration via FPGA. Although this transformation marks an early phase in development, it sets the groundwork for future applications requiring timely processing and efficient sensor data management.***

The robot's performance remains on par with the initial Raspberry Pi configuration, laying a solid basis for further investigation. The forthcoming research phase will concentrate on the development, application, and evaluation of specific hardware functions utilizing co-design to fully exploit the advantages of FPSoC integration. Our preliminary results and continued research efforts are contributing to the expanding field of study focused on enhancing robotic functionalities for a range of uses, such as autonomous navigation, object detection, and sensor data analysis.

Keywords-- ***FPSoC Integration, ROS-enabled Robotics, TurtleBot3 Enhancement, Mobile Robot, ROS-FPSoC Implementation.***

ISBN: 978-628-95207-8-1. ISSN: 2414-6390. Digital Object Identifier: 10.18687/LACCEI2024.1.1.1908

Integración de ROS y FPSoC en una Plataforma Robótica Turtlebot 3
ROS and FPSoC Integration on a Turtlebot 3 Robotic Platform

Jairo Cuero, MEng Automatización Industrial[1], Pedro-F Cárdenas, PhD Automática y Robótica[2], Rony Trespalacios, Ingeniero Electrónico[3]
[1,3]Universidad de los Llanos, Colombia, jairo_cuero@unillanos.edu.co, rony.trespalacios@unillanos.edu.co
[2]Universidad Nacional de Colombia, Colombia, pfcardenash@unal.edu.co

Resumen - En esta investigación, presentamos una estrategia inicial para incorporar la tecnología de Sistema en Chip Programable por Campo (FPSoC) en un TurtleBot3, un robot móvil compatible con el Sistema Operativo de Robots (ROS). El objetivo principal fue reemplazar la placa Raspberry Pi original por una placa FPSoC, dotando al robot de un bloque de compuertas Programables por Campo (FPGA) para acelerar las funciones mediante el diseño concurrente hardware - software. A través de un esfuerzo meticuloso de integración, logramos convertir el TurtleBot3 en un prototipo funcional que mantiene sus funcionalidades esenciales mientras aprovecha las capacidades avanzadas de aceleración de funciones a través de FPGA. Aunque esta transformación marca una fase temprana en el desarrollo, establece las bases para futuras aplicaciones que requieren un procesamiento avanzando y una gestión eficiente de los datos de sensores. El rendimiento del robot sigue siendo comparable a la configuración inicial con la Raspberry Pi, estableciendo una base sólida para investigaciones futuras. La próxima fase de investigación se centrará en el desarrollo, aplicación y evaluación de funciones de hardware específicas utilizando el diseño concurrente para aprovechar al máximo las ventajas de la integración del FPSoC. Nuestros resultados preliminares y los continuos esfuerzos de investigación están contribuyendo al creciente campo de estudio enfocado en mejorar las funcionalidades robóticas para una variedad de usos, como la navegación autónoma, detección de objetos y análisis de datos de sensores.

Palabras Claves - Integración FPSoC, Robótica con soporte de ROS, Mejora del TurtleBot3, Robot Móvil, Implementación de ROS en FPSoC.

Abstract– In this research, we introduce an initial strategy for incorporating Field-Programmable System on a Chip (FPSoC) technology into a Robot Operating System (ROS)-compatible mobile robot, with a focus on the TurtleBot3 platform. The primary objective was to swap the original Raspberry Pi board for an FPSoC board, equipping the robot with a Field-Programmable Gate Array (FPGA) section for enhanced function acceleration through hardware-software co-design. Through a meticulous integration effort, we were able to convert the TurtleBot3 into a working prototype that maintained its essential functionalities while leveraging the advanced capabilities of function acceleration via FPGA. Although this transformation marks an early phase in development, it sets the groundwork for future applications requiring timely processing and efficient sensor data management. The robot's performance remains on par with the initial Raspberry Pi configuration, laying a solid basis for further investigation. The forthcoming research phase will concentrate on the development, application, and evaluation of specific hardware functions utilizing co-design to fully exploit the advantages of FPSoC integration. Our preliminary results and continued research efforts are contributing to the expanding field of study focused on enhancing robotic functionalities for a range of uses, such as autonomous navigation, object detection, and sensor data analysis.

Keywords-- FPSoC Integration, ROS-enabled Robotics, TurtleBot3 Enhancement, Mobile Robot, ROS-FPSoC Implementation.

I. Introducción

Los robots móviles constituyen un paradigma ejemplar de sistemas que demandan una intensiva ejecución multitarea con procesos de alto costo computacional. Su aplicación en la exploración espacial, particularmente en el estudio de Marte, ha catalizado avances sustanciales en la industria aeroespacial desde principios del siglo XXI. El punto de partida significativo se encuentra en el Sojourner, el vehículo pionero de ruedas que efectuó un aterrizaje exitoso en Marte en 1997, marcando un logro de magnitud en la investigación del planeta rojo. Subsecuentemente, en 2003, se lanzaron dos robots Rovers, el Spirit y el Opportunity; posteriormente, en 2011, el Curiosity hizo su entrada en escena, seguido por el Perseverance en 2020. Estos robots han continuado la exploración con incrementadas capacidades y una dotación de equipos robustos [1].

La presencia de robots móviles en Marte ha presentado desafíos considerables debido a las condiciones hostiles del entorno. Esto ha impulsado el desarrollo de numerosas estrategias para abordar estas adversidades, incluyendo el uso de componentes como las FPGAs y estrategias de codiseño HW/SW [2]. Estas estrategias se han aplicado de manera especialmente efectiva en la aceleración de algoritmos para tareas como la visión por computadora [3] y la navegación autónoma [4] en el contexto de la exploración marciana.

La ejecución de tareas complejas, como la navegación autónoma, la localización y el mapeo simultáneos (SLAM) y

la interacción humano-robot, exige una mayor potencia de cálculo y mejores capacidades de procesamiento en tiempo real [5, 6]. Gracias a su hardware reconfigurable, las FPGA ofrecen una importante ventaja de rendimiento sobre los microcontroladores convencionales [7] e incluso los computadores de placa única (SBC, por sus siglas en inglés) como la Raspberry Pi [8]. Las FPGA destacan en tareas de procesamiento en paralelo, como la fusión de datos de sensores, el procesamiento de imágenes y los algoritmos de control, lo que las hace idóneas para aplicaciones en tiempo real en robótica móvil [6]; y aunque tienen potencial para satisfacer estos requisitos, su integración en robots móviles se ha visto limitada por las complejidades inherentes asociadas a la difícil curva de aprendizaje de la programación de FPGA a través de la descripción de hardware, y el marco de software específico y el más utilizado habitualmente en este campo, el sistema operativo para robots (ROS) [9].

En uno de los primeros intentos de lograr la integración ROS-FPGA en robótica, el trabajo de [10] muestra el desarrollo de la plataforma hardware- software (HW/SW) "Hero" diseñada para acelerar las tareas de los sistemas autónomos. La plataforma Hero cuenta con un procesador de última generación Intel Core i5 intrincadamente emparejado con una FPGA Intel Arria 10, perfectamente integrada con el framework OpenCL. Posteriormente, [11] introdujo una pila robótica ROS de código abierto basada en el robot Parallax Arlo, integrada con la tarjeta de desarrollo FPSoC Zybo z7-10. Del mismo modo, [12] ideó un vehículo autónomo basado en ROS, con el Zybo z7-20 como placa principal.

En investigaciones no relacionadas, dos estudios realizados por Nitta en 2018 y 2019 [13, 14] demostraron las capacidades de ZytleBot, un robot que utilizaba el FPSoC Ultra96 rev1 y ejecutaba ROS en Ubuntu 18.04. ZytleBot era capaz de trabajar conjuntamente con un chasis TurtleBot3 y empleaba tecnología FPGA para la interfaz de la cámara y el motor. Además, el trabajo de Wu de 2020 introdujo el robot HydraMini [15], que estaba equipado con la placa Xilinx PYNQ-Z2 y se encargaba de gestionar el control mecánico, la inferencia de IA y el análisis de visión por ordenador, todo ello con la ventaja de la aceleración por hardware.

A partir de los conceptos previamente discutidos, en este trabajo se sugiere la fusión de ROS con un sistema en un chip (SoC FPGA) dentro de una estructura robótica para potenciar las habilidades del autómata. Optamos por el TurtleBot3 (TB3), un robot móvil de tracción diferencial equipado con dos motores Dynamixel, un tercer punto de apoyo en forma de rueda esférica, una Raspberry Pi 3B+, un sensor LiDAR y una placa OPENCR 1.0 destinados a la navegación y la odometría. El TB3 está capacitado para ejecutar diversas funciones propias de los robots móviles, incluyendo la navegación independiente, el mapeo del entorno y la identificación de obstáculos.

No obstante, la utilización de ROS señala que las limitaciones se encuentran más en el ámbito del hardware que en el software. A medida que el robot asume más responsabilidades, se observa un incremento en el consumo energético del procesador y en los requisitos de procesamiento. Por ello, se sugiere reemplazar la placa Raspberry Pi con un SoC que combine tanto el procesador como los componentes lógicos programables, permitiendo así una distribución y asignación eficiente de tareas entre el procesador y la FPGA.

II. Arquitectura HW/SW

Un robot se estructura en torno a tres componentes esenciales: un sistema electrónico, un sistema mecánico (hardware) y un sistema de gestión de datos (software). Para elaborar un robot móvil capaz de desplazarse e interactuar de manera independiente con su entorno, es crucial la integración armónica de estos tres componentes. Dicha integración es referida como la arquitectura de hardware y software (HW/SW) del robot.

La arquitectura del software se organiza en múltiples niveles, incluyendo aplicaciones, interfaces de control y redes de comunicación, entre otros. Esta estructura opera a través de uno o más procesadores disponibles en el sistema. En contraparte, el diseño del hardware se adapta y se condiciona a los recursos físicos del dispositivo.

Para optimizar las operaciones de navegación en robots móviles, se ha sugerido una arquitectura HW/SW específica. Tal arquitectura se basa en una configuración robótica dirigida por un SoC que incorpora un procesador ARM y una unidad de lógica programable (FPGA), facilitando el uso de estrategias de diseño conjunto mediante la implementación de funciones directamente en el hardware. Esto posibilita la división y asignación de tareas para su procesamiento en el segmento de software o en el componente de hardware, dependiendo de la complejidad y los requerimientos del diseño [20]. La figura 1 muestra de forma esquemática esta arquitectura HW/SW.

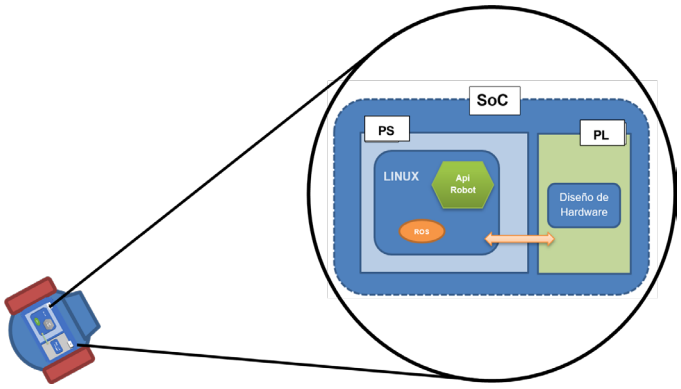

Fig 1. Arquitectura hardware-software HW/SW propuesta.

A. Ajustes de Hardware

El TurtleBot3 se presenta como una plataforma robótica innovadora y accesible, diseñada para universalizar la tecnología robótica y servir como un catalizador en el ámbito de la investigación, la educación y el desarrollo de aplicaciones robóticas prácticas. Concebido como un sistema modular y de código abierto, el TurtleBot3 destaca por su versatilidad, permitiendo a usuarios de diversos niveles explorar el vasto mundo de la robótica con una inversión relativamente baja. Equipado con una variedad de sensores y actuadores, incluyendo un sensor LiDAR para la navegación y mapeo del entorno, así como motores Dynamixel para un movimiento preciso y fiable, el TurtleBot3 ofrece una plataforma ideal para adentrarse en conceptos avanzados como la navegación autónoma, la interacción humano-robot y el aprendizaje de máquinas [19, 20].

Con su núcleo basado en la Raspberry Pi, el TurtleBot3 facilita la programación y el control mediante el popular sistema operativo ROS (Robot Operating System), promoviendo una comunidad activa de desarrolladores y entusiastas que contribuyen constantemente con tutoriales, proyectos y mejoras. Esta colaboración abierta extiende las capacidades del TurtleBot3, transformándolo en una herramienta educativa invaluable y una plataforma de investigación flexible, capaz de adaptarse a las necesidades cambiantes de proyectos robóticos complejos.

El TurtleBot3 Burger viene totalmente desmontado de fábrica, por lo que en el proceso de ensamblado y luego de unas pruebas preliminares se estableció que la batería LiPo de 11,1V y 1800mAh de la placa inferior ofrecía poca autonomía al robot. Por lo tanto, se decidió cambiar por una batería LiPo de 12.6V (carga completa) a 3000mAh correspondientes a una configuración 3S1P con 3 celdas; esto ofrece mayor capacidad y las dimensiones permiten que siga ocupando el mismo espacio que la original. De igual manera, atendiendo la idea de mejorar el rendimiento del robot se retiró la Raspberry Pi de la tercera placa del robot y se le adaptó un SoC-FPGA, una tarjeta Ultra96V2.

La Ultra96v2 es una avanzada plataforma de desarrollo que incorpora el sofisticado SoC Zynq UltraScale+ MPSoC de Xilinx, ofreciendo una solución integral para el diseño y prototipado de aplicaciones embebidas complejas. Esta tarjeta destaca por combinar un procesador ARM Cortex-A53 de cuatro núcleos con un procesador Cortex-R5 de doble núcleo, y una poderosa unidad de procesamiento gráfico (GPU) Mali-400MP2, todo integrado junto a un campo de puertas programables (FPGA) de alto rendimiento. Esta arquitectura híbrida proporciona una versatilidad excepcional, permitiendo a los desarrolladores optimizar el rendimiento al delegar tareas específicas ya sea al procesamiento de la CPU o al procesamiento paralelo del FPGA, lo que resulta ideal para aplicaciones que requieren una alta capacidad de procesamiento en tiempo real, como la visión por computadora, la inteligencia artificial y el procesamiento de señales. Además, la Ultra96v2 soporta una amplia gama de interfaces y conectividades, incluyendo USB 3.0, Wi-Fi, Bluetooth, y GPIO, lo que la hace una plataforma excepcionalmente flexible y capaz para una diversidad de proyectos de desarrollo tecnológico.

En las pruebas iniciales, la temperatura del Ultra96V2 alcanzó más de 53°C en pocos minutos de funcionamiento. Por lo que fue necesario adaptarle un ventilador a la carcasa del disipador térmico del SoC para reducir el sobrecalentamiento de la placa de desarrollo. Esta solución mantuvo la temperatura por debajo de 45 °C, como se muestra en la figura 2.

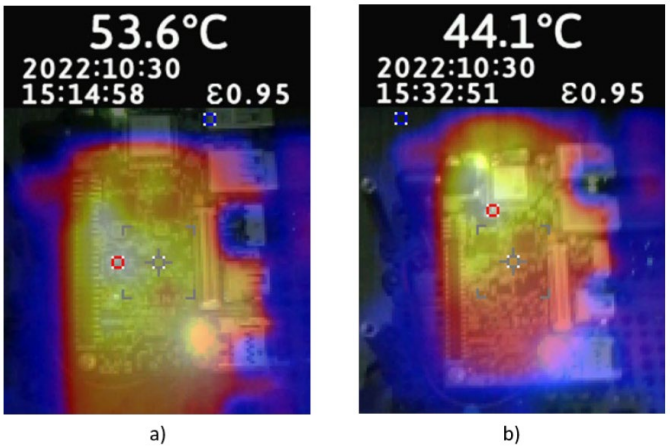

Fig 2. Imagen termográfica del SoC. a) sin ventilador. b) con ventilador.

En el proceso de conexión del ventilador, se consideraron dos opciones: su enlace al Ultra96v2 mediante los pines específicos de la tarjeta o su alimentación a través del OpenCR. La primera alternativa permite un manejo más eficaz de la temperatura y la potencia del ventilador, debido a que el FPSoC cuenta con un pin PWM exclusivo para este propósito. No obstante, esta opción exigía una manipulación eléctrica más compleja de la tarjeta, ya que carece de pines preestablecidos como conector, lo que obligaría a soldarlos directamente. Por otro lado, optar por la alimentación mediante el OpenCR aprovecha el conector originalmente destinado para proporcionar la energía a la Raspberry, eliminando la necesidad de realizar ajustes físicos en la tarjeta.

La Figura 3 muestra el esquema de interconexiones eléctricas y de comunicación de los componentes del robot. Se observa que la conexión USB entre el OpenCR y el Ultra96V2 se mantiene sin cambios en comparación con la configuración inicial. Asimismo, el LiDAR se conecta a la tarjeta Ultra96v2 mediante el controlador USB2LDS, asegurando la transmisión de datos entre estos dispositivos.

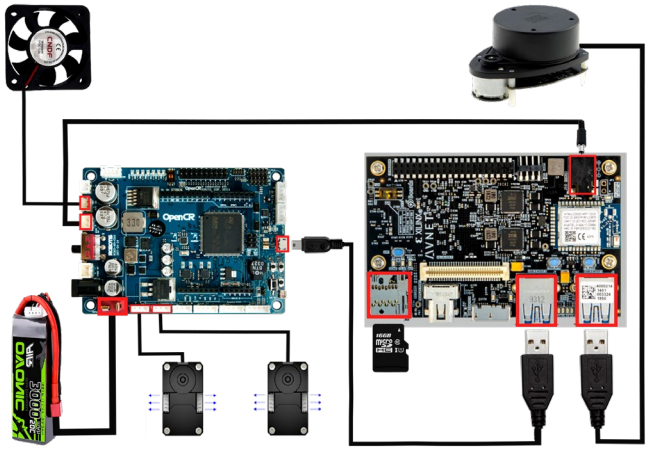

Fig 3. Esquema de las conexiones entre los componentes que forman el robot.

B. Sistema Operativo

Para el sistema operativo del robot se decidió PYNQ (Python Productivity for Zynq), una innovadora plataforma de desarrollo de software que aprovecha la flexibilidad y la potencia de los sistemas en chip (SoC) Zynq para ofrecer una experiencia de programación de alto nivel centrada en Python. Esta plataforma está diseñada para simplificar y acelerar el desarrollo de aplicaciones aprovechando las capacidades de hardware programable, como las Field-Programmable Gate Arrays (FPGAs), sin la necesidad de tener un conocimiento profundo en programación de hardware. Al combinar la facilidad de uso de Python con el rendimiento y la eficiencia de las FPGAs, PYNQ permite a ingenieros, investigadores y desarrolladores explorar soluciones complejas en áreas como el procesamiento de señales, la visión por computadora y la inteligencia artificial de manera más intuitiva y productiva.

La arquitectura de PYNQ se basa en el concepto de overlays de hardware, que son capas de abstracción que permiten a los usuarios acceder y controlar bloques de hardware de alto rendimiento a través de scripts de Python sencillos. Esta aproximación reduce significativamente la curva de aprendizaje asociada con el desarrollo de hardware tradicional y abre el camino para una amplia gama de aplicaciones, desde prototipos rápidos hasta proyectos de investigación avanzada. Además, PYNQ fomenta una comunidad activa de colaboración, donde los usuarios pueden compartir sus overlays y experiencias, enriqueciendo así el ecosistema y facilitando el acceso a tecnologías de vanguardia.

La versión de PYNQ seleccionada fue la V2.6, basada en la distribución UBUNTU Bionic Beaver. Ésta utiliza Jupyter, una plataforma informática interactiva basada en web para escribir programas en Python, administrar el almacenamiento y la organización de carpetas y archivos. Además, el sistema cuenta con una interfaz de terminal completamente operativa que permite instalar paquetes utilizando apt-get, lo que permitió instalar ROS Melodic.

Para esto último fue necesario anular la detección automática del sistema operativo mediante la variable de entorno ROS_OS_OVERRIDE que permite forzar la instalación desde la lista de paquetes de Ubuntu Bionic. Este procedimiento es esencial dado que el nombre clave de la distribución Ubuntu en PYNQ no es Bionic, sino WHF, tal como se indica en la Figura 4. Debido a esta discrepancia, el instalador no logra encontrar los paquetes correspondientes en el repositorio oficial de ROS sin esta modificación.

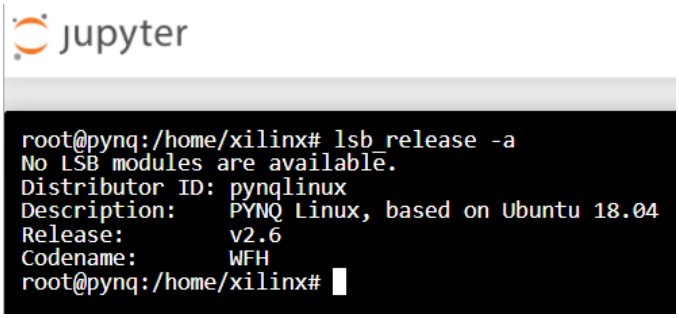

Fig 4. Terminal Jupyter que muestra la versión del sistema operativo.

C. Configuración de la red

La correcta configuración de la red es esencial para establecer una comunicación eficaz entre el robot y el ROS Master, abarcando la asignación precisa de la dirección IP, la máscara de subred, y la puerta de enlace por defecto. Adicionalmente, es imperativo ajustar las reglas del firewall para facilitar el tránsito de los paquetes de red esenciales para la comunicación dentro del entorno ROS.

Durante el proceso de configuración, se estableció un entorno de trabajo optimizado para la interacción entre el robot y el PC remoto. Aunque tradicionalmente el ROS Master se aloja en el PC remoto, esta disposición restringe la operación del robot desde un ordenador remoto desprovisto de ROS, dado que los nodos requieren registrarse con el Master para su funcionamiento. Por lo tanto, se optó por alojar el ROS Master directamente en el robot, una decisión facilitada por el uso del framework PYNQ que permite eludir la conexión SSH convencional. Este enfoque innovador permite acceder al robot a través de Jupyter Notebook desde cualquier navegador web, incluidos dispositivos móviles, ampliando significativamente las posibilidades de interacción con el sistema.

Para abordar los retos de compatibilidad se eligió evaluar dos sistemas operativos mediante la configuración de Dual Boot en el PC remoto, con Ubuntu 18.04 y Windows 11. Además, se instaló Ubuntu Bionic en el Subsistema de Windows para Linux (WSL2). La Fig. 5 muestra las dos opciones de configuración de red evaluadas. La primera

configuración es la habitual, pero con el ROS Master ubicado en el robot. En la segunda se utiliza WSL2 con Ubuntu Bionic porque ROS1 no funciona en Windows.

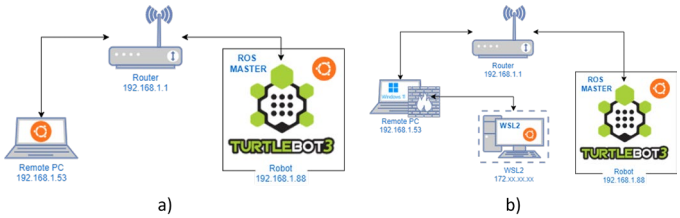

Fig 5. Diagrama de red. a) En Linux. b) En Windows 11 utilizando WSL2.

Aquí se solucionaron algunos problemas estableciendo redireccionamiento de puertos y creando algunas reglas en el firewall. Por otro lado, para solucionar la ausencia de interfaz gráfica en WSL2, se utiliza un servidor x para mostrar en windows 11 algunas aplicaciones GUI como RViz y Rqt Graph.

III. Resultados

Las modificaciones físicas del robot y la integración de los distintos elementos de hardware y software en el turtlebot3 se presentan en la figura 6. Una vez realizadas estas adaptaciones se inicia con la evaluación de la plataforma robótica.

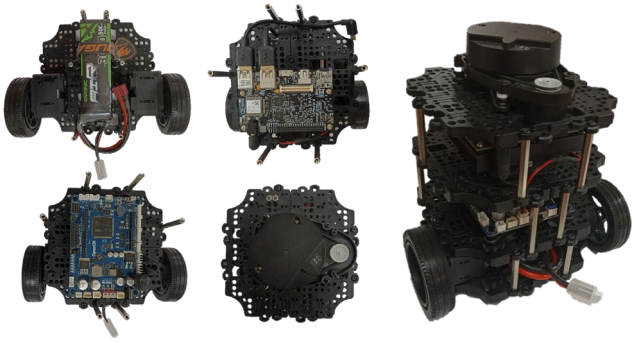

Fig 6. Modificación del robot, integración de la tarjeta Ultra96V2, instalación del LiDAR y estructura final.

Para ello, se ejecuta el tutorial original con el objetivo de verificar que el robot funciona sin problemas desde la tarjeta Ultra96v2 tal como lo haría con la raspberry Pi. Inicialmente se crea un nodo para capturar los datos LiDAR y luego se ejecutan los nodos para la teleoperación del robot, el seguimiento básico de trayectorias en entornos estructurados y la construcción del mapa. Lo anterior se realiza con Rviz y Jupyter notebook que proporciona el ecosistema PYNQ.

La Figura 7 muestra el resultado de la generación del mapa, y la Figura 8 muestra el gráfico de los nodos y temas en ejecución. La sustitución de la CPU por una tarjeta FPSoC fue transparente, lo que significa que el robot funcionó igual de bien con cualquiera de las dos tarjetas de desarrollo.

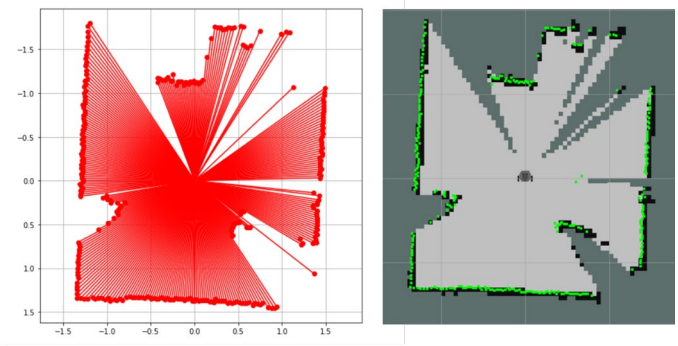

Fig 7. Mapa a partir de los datos LiDAR. a) En Jupyter notebook. b) Visualizado en Rviz.

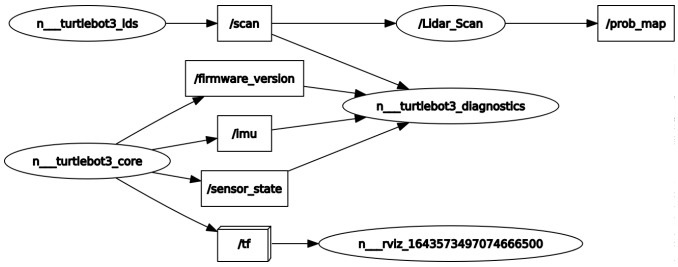

Fig 8. Ejecución de nodos y temas ROS en FPSoC.

Se logró una mejora significativa en la autonomía y capacidad de procesamiento del robot mediante la ejecución del ROS MASTER directamente desde la tarjeta integrada, eliminando la dependencia de dispositivos externos con ROS instalado. Este avance permite conexiones a través de interfaces como Jupyter Notebook, expandiendo las posibilidades de interacción y control remoto como lo podría ser desde un teléfono móvil. Además, el robot ha sido equipado con avanzados recursos de hardware gracias a la lógica programable del dispositivo FPSoC, lo que podría resultar en mejoras notables en la velocidad de procesamiento, especialmente en la adquisición de datos del sensor LiDAR, la generación de mapas y la aceleración de algoritmos de alto coste computacional que antes no eran factibles con solo la Raspberry Pi. Esta innovación supera las limitaciones de hardware anteriores y potencia significativamente el rendimiento del sistema robótico.

CONCLUSIONES

Con la implementación de este proyecto, hemos logrado desarrollar una plataforma TurtleBot3 (TB3) optimizada con ROS, con mayor autonomía y la utilización de dispositivos de

bajo consumo como las FPGA. Este enfoque no solo asegura una operación prolongada sin necesidad de recargas frecuentes, sino que también permite una gestión de tareas más eficiente durante el desempeño de algoritmos computacionalmente exigentes.

Además, la flexibilidad de la arquitectura HW/SW propuesta ha demostrado ser un factor clave para la adaptabilidad del robot en diversos entornos y situaciones. La capacidad de modificar y optimizar fácilmente el comportamiento del robot mediante el ajuste de parámetros de software y hardware abre nuevas posibilidades para la personalización y mejora continua del sistema, sin la necesidad de realizar cambios físicos en la plataforma. El trabajo realizado reduce la curva de aprendizaje de la descripción de hardware de los FPGAs, ya que la lógica programable puede ser descrita mediante C/C++ en Vitis HLS.

Se ha optimizado un robot integrando el ROS MASTER directamente en su tarjeta, lo cual aumenta su autonomía al eliminar la dependencia de dispositivos externos. Además, se incorporaron mejoras en el hardware mediante dispositivos FPSoC, mejorando la eficiencia en la adquisición de datos del sensor LiDAR y la ejecución de algoritmos complejos, lo que resulta en un rendimiento notablemente superior.

Por otro lado, el uso de herramientas de visualización como Rviz, junto con Jupyter notebook del ecosistema PYNQ, ha facilitado de manera significativa la interpretación de los datos recopilados por el robot, así como la planificación y ejecución de estrategias de navegación y mapeo. Esto no solo ha mejorado la eficiencia en la creación de mapas del entorno, sino que también ha permitido una interacción más intuitiva con el robot, haciéndolo accesible a usuarios con distintos niveles de experiencia técnica.

Se prevé que la inclusión de la parte lógica programable en la arquitectura hardware/software (HW/SW) diseñada en este proyecto debe ser capaz de gestionar tareas de alta complejidad como la navegación independiente, la evasión de obstáculos, el reconocimiento de patrones, la categorización de objetos y la interacción entre humanos y robots. Estas actividades demandan una gran capacidad de procesamiento, necesitando por ende una tarjeta de desarrollo avanzada para mejorar la capacidad de respuesta del robot.

RECONOCIMIENTOS

Este trabajo fue apoyado y financiado por la Dirección General de Investigaciones de la Universidad de Los Llanos, Colombia, y bajo el proyecto de investigación N° C09-F02-018 2022.

REFERENCIAS

[1] NASA. (2023). Mars 2020 Perseverance Rover - NASA Mars. https://mars.nasa.gov/mars2020/

[2] Lentaris, G., Stamoulias, I., Soudris, D., & Lourakis, M. (2016). HW/SW codesign and FPGA acceleration of visual odometry algorithms for rover navigation on mars. IEEE Transactions on Circuits and Systems for Video Technology, 26(8), 1563-1577. https://doi.org/10.1109/TCSVT.2015.2452781

[3] Matthies, L., Maimone, M., Johnson, A., Cheng, Y., Willson, R., Villalpando, C., Goldberg, S., Huertas, A., Stein, A., & Angelova, A. (2007). Computer vision on Mars. International Journal of Computer Vision, 75(1), 67-92. https://doi.org/10.1007/s11263-007-0046-z

[4] Kostavelis, I., Nalpantidis, L., Boukas, E., Rodrigalvarez, M. A., Stamoulias, I., Lentaris, G., Diamantopoulos, D., Siozios, K., Soudris, D., & Gasteratos, A. (2014). SPARTAN: Developing a vision system for future autonomous space exploration robots. Journal of Field Robotics, 31(1), 107-140. https://doi.org/10.1002/rob.21484

[5] Salah Bouhoun, Rabah Sadoun y Mourad Adnane. Opencl implementation of a slam system on an soc-fpga. Journal of Systems Architecture, 111, 2020.

[6] Runze Liu, Jianlei Yang, Yiran Chen y Weisheng Zhao. eslam: An energy-efficient accel- erator for real-time orb-slam on fpga platform. páginas 1-6, 2019.

[7] Veronika Rappl. Comparación entre microcontrolador y fpga: Ventajas y campos de aplicación adecuados. FMS-BERICHTE, 2022.

[8] TA Mounir, Selma Boumerdassi, A Benhamada, AM Alla y M Kherarba. Performance evaluation of basic image processing algorithms in cpu gpu raspberry pi and fpga. International Journal of Computer Science Engineering (IJCSE), 9(4):312-325, 2020.

[9] Ariel Podlubne y Diana Gohringer. Fpga-ros: Metodología para aumentar el sistema operativo de robots con diseños fpga. 2019.

[10] Xuesong Shi, Lu Cao, Dawei Wang, Ling Liu, Ganmei You, Shuang Liu y Chunjie Wang. Hero: Aceleración de tareas robóticas autónomas con fpga. Conferencia internacional del IEEE sobre robots y sistemas inteligentes, páginas 7766-7772, 2018.

[11] Taylor Joseph y Linville Whitaker. Hacia una plataforma prototipo para integraciones ros en un robot terrestre. 2019.

[12] Kento Hasegawa, Kazunari Takasaki, Makoto Nishizawa, Ryota Ishikawa, Kazushi Kawamura y Nozomu Togawa. Implementación de un vehículo autónomo basado en ros en una placa fpga. volumen 2019-Decem, páginas 457-460, 2019.

[13] Yasuhiro Nitta, Sou Tamura y Hideki Takase. A study on introducing fpga to ros based autonomous driving system. páginas 424-427. IEEE, 2018.

[14] Yasuhiro Nitta, So Tamura y Hideki Takase. Zytlebot : plataforma de desarrollo integrada Fpga para robot móvil autónomo basado en ros. páginas 422-423. IEEE, 2019.

[15] Tianze Wu, Yifan Wang, Weisong Shi y Joshua Lu. Hydramini: An fpga-based affordable research and education platform for autonomous driving. páginas 45-52, 2020.

[16] Iva'n A. Calle Flores. Navegacio'n auto'noma de un robot mo'vil usando te'cnicas probabil'ısticas de localizacio'n y mapeo basadas en me'todos monte carlo secuenciales. 2014.

[17] Takeshi Ohkawa, Kazushi Yamashina, Hitomi Kimura, Kanemitsu Ootsu y Takashi Yokota. Componentes fpga para integrar fpgas en sistemas robóticos. IEICE Transactions on Information and Systems, E101D:363-375, 2018.

[18] LIENEN, C. (2023). Enabling reconfigurable hardware acceleration for ROS-based robotics applications (Doctoral dissertation, paderborn university).

[19] Al-Ameri, Y., & Westerlund, T. (2023). FPGA-Based Hardware Accelerators for Deep Learning in Mobile Robotics.

[20] Lienen, C., Middeke, S. H., & Platzner, M. (2023, October). fpgaDDS: An Intra-FPGA Data Distribution Service for ROS 2 Robotics Applications. In 2023 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS) (pp. 6261-6266). IEEE.