REVISTA FCEFyN

Hermes III: Robot Omnidireccional con capacidad de SLAM

Orlando Micolini[1], Luis O. Ventre[1], Agustín Coutinho[1], Hernán Malatini[1] y Rubén E. Ayme[1]

[1]*Laboratorio de Arquitectura de Computadoras, Facultad de Ciencias Exactas Físicas y Naturales Universidad Nacional de Córdoba, Córdoba, Argentina.*

Fecha de recepción del manuscrito: 13/09/2021
Fecha de aceptación del manuscrito: 29/04/2022
Fecha de publicación: 30/04/2022

Resumen—En el presente trabajo se detalla el proceso de rediseño del robot móvil omnidireccional Hermes II, que fue desarrollado en el Laboratorio de Arquitectura de Computadoras de la FCEFyN de la UNC. Esta nueva versión, denominada Hermes III, posee el mismo sistema de locomoción omnidireccional que su antecesor e incorpora mejoras sustanciales en cuanto a los sistemas de control y sensado que comanda al robot móvil. Además el nuevo robot incorpora la capacidad de Mapeo y Localización Simultánea; para dotar al sistema robótico de esta capacidad se incorporó un dispositivo Kinect V1. En el nuevo sistema robótico se aumenta la escalabilidad a través del llamado Sistema Operativo Robótico, conocido por sus siglas ROS, que es un marco que facilita el desarrollo de software para robots. El proyecto se implementa en una placa de desarrollo NVIDIA Jetson TK1, que dadas sus capacidades computacionales soporta los requerimientos del sistema. El resultado de este proyecto muestra la implementación de algoritmos sofisticados para la localización y movimiento de un robot omnidireccional. Así mismo se han validado los criterios de diseño e identificado cotas de desempeño, analizadas cuantitativamente, frente a una integración de componentes específica, obteniéndose un sistema robótico móvil omnidireccional robusto y flexible con capacidad autónoma de exploración, localización y cartografía.

Palabras clave—robot; omnidireccional; Robot Operating System ROS; Kinect; SLAM.

Abstract— This paper details the redesign process of the Hermes II omnidirectional mobile robot, which was developed at the Computer Architecture Laboratory of the FCEFyN-UNC. This new version, called Hermes III, has the same omnidirectional locomotion system as its predecessor and incorporates substantial improvements in terms of control and sensing systems that command the mobile robot. Furthermore, the new robot incorporates the capability of Simultaneous Mapping and Localization; in order to provide the robotic system with this capability, a Kinect V1 device was incorporated. In the new robotic system scalability is increased through the Robotic Operating System, known by its acronym ROS, which is a framework that enhances the development of software for robots. The project is implemented on an NVIDIA Jetson TK1 development board, which, given its computational capabilities, supports the requirements of the system. The result of this project shows the implementation of sophisticated algorithms for the localization and movement of an omnidirectional robot. Likewise, the design criteria have been validated and quantitatively analyzed performance levels have been identified against a specific integration of components, obtaining a robust and flexible omnidirectional mobile robotic system with autonomous exploration, localization and mapping capabilities.

Keywords— robot; omnidirectional; Robot Operating System ROS; Kinect; SLAM.

INTRODUCCIÓN

La tecnología robótica se encuentra en plena fase de crecimiento y, por tanto, es previsible una presencia cada vez mayor de robots, tanto en el sector industrial y de servicios, como en el sector doméstico. Los robots móviles sobre ruedas omnidireccionales son capaces de brindar mayores prestaciones como herramientas por su capacidad de maniobra en los diferentes entornos con obstáculos, donde los espacios son reducidos. Además la incorporación de capacidades autónomas para exploración, localización y cartografía los convierte en una opción robusta y útil. Como ha ocurrido en la informática, el desarrollo de robots se hace cada vez más accesible y por consiguiente sus capacidades tienden a ser cada vez mayores gracias al aporte de una comunidad en auge.

A medida que fueron avanzando las investigaciones en robótica, se estableció como regla general que un robot móvil capaz de auto navegar de forma exitosa debe responder a tres preguntas: ¿Dónde estoy?, ¿Hacia dónde voy? y ¿Cómo puedo llegar allí? (Pulido Fentanes, 2012).

Actualmente las investigaciones se enfocan en diversos campos que confluyen en la búsqueda de un objetivo común: el desarrollo de un sistema autónomo. En la Fig. 1 se puede observar de manera gráfica la estrecha interrelación que existe entre los principales campos y hasta qué punto el desarrollo de todos y cada uno de ellos es necesario para dotar de autonomía a un robot(Borenstein et al., 1996).

- Navegación: Capacidad de movimiento.
- Localización: Capacidad de sensar sus movimientos y medirlos.

Dirección de contacto: Ing. Ventre, Luis O., Av. Velez Sarfield 1611, Ciudad Universitaria, X5016 CGA. Tel: 5353800 interno 29000.
Correo Electrónico: luis.ventre@unc.edu.ar

- Mapeado: Capacidad de detectar profundidad de campo.
- Localización Activa: El robot puede moverse y conocer su posición en relación a un punto de partida.
- Mapeo y Localización Simultanea (SLAM - Simultaneous Localization and Mapping): El robot, teniendo conocimiento de la magnitud de sus movimientos de traslación y contando con la capacidad de sensar profundidad de campo, genera un mapa de su entorno y lo actualiza en línea al tiempo que utiliza el mapa para ubicarse en un entorno incierto.
- Exploración: El robot puede desplazarse generando un mapa de su entorno.

En este trabajo se detalla el proceso de rediseño del robot móvil omnidireccional Hermes II (Ayme et al., 2020), que fue desarrollado en el Laboratorio de Arquitectura de Computadoras de la FCEFyN de la UNC. Este proceso

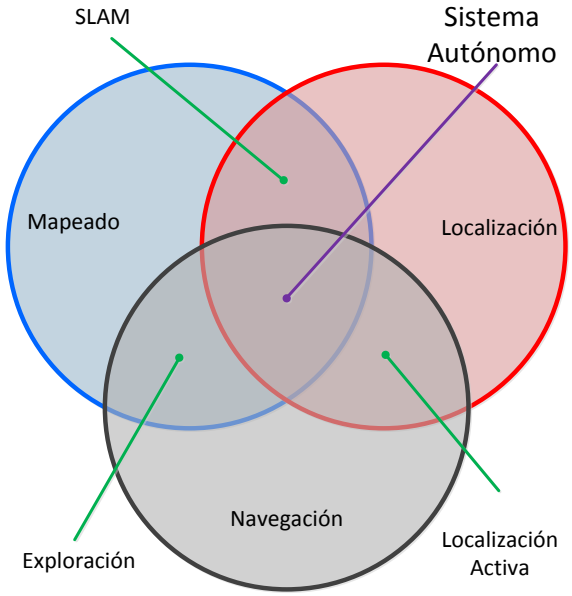

Fig. 1. Interrelación de atributos de robots móviles

incluye la actualización de los sistemas de sensores y de control, así como la inclusión de la capacidad SLAM. Esta nueva versión, bautizada como Hermes III, posee el mismo sistema de locomoción omnidireccional que su antecesor e incorpora mejoras sustanciales en cuanto a los sistemas de control y sensado que comanda al robot móvil. Para aumentar la escalabilidad del sistema se utilizó el Sistema Operativo Robótico, conocido por sus siglas ROS (Robot Operating System) (Foundation, 2020, Newman, 2017), que es un marco específico para el desarrollo de software para robots. Dado su carácter de código abierto, es una herramienta disponible, mantenida y utilizada en la actualidad por gran parte de la comunidad de desarrolladores de robots móviles. Este sistema operativo se implementa en una placa de desarrollo NVIDIA Jetson TK1, que dadas sus capacidades computacionales soporta los requerimientos del sistema. A su vez se incorpora un dispositivo Kinect V1 para dotar al robot de capacidad Mapeo y Localización Simultanea. El resultado de este proyecto es un sistema robótico móvil omnidireccional robusto y flexible con capacidad autónoma de exploración, localización y cartografía.

El cuerpo de este articulo está organizado de acuerdo al siguiente orden: a continuación se describen los materiales y métodos. Luego se presentan los casos de prueba, a continuación se listan los resultados obtenidos y una explicación de los mismos. Finalmente, se presentan las conclusiones y los trabajos futuros.

ANTECEDENTES

Esta propuesta toma la experiencia adquirida del desarrollo de los robots móviles (Caverzasi et al., 2014), Hermes (Lichtensztein et al., 2014) y Hermes II (Ayme et al., 2020), los cuales fueron implementados como parte del proyecto de investigación “Diseño e Implementación de Software y Hardware Optimizados para Sistemas de Computación Paralelos en Ingeniería”.

Para este desarrollo se han mantenido los siguientes criterios: altas relación prestaciones/costo, escalabilidad, mantenibilidad, tamaño de escritorio, posibilidades desde el punto de vista educativo e ingenieril, simplicidad de uso e información abierta. Se priorizó el criterio de funcionalidad, costo y simplicidad para el diseño y la selección de las partes.

OBJETIVOS

El objetivo de este trabajo es el rediseño del robot móvil omnidireccional Hermes II con capacidad de SLAM, y la comprensión de la gestión de los recursos mediante la utilización de un sistema operativo especifico de robots. De esta forma se materializa el objeto de estudio con la capacidad de agregar interacción social al contexto de aprendizaje.

Esta iniciativa tiene como objetivo ir más allá del aprendizaje puramente basado en software y/o la simulación (Mubin et al., 2013). Además se ha mantenido como objetivo común resaltar los procesos de construcción y programación de un robot móvil omnidireccional para que los estudiantes reflexionen por medio de la experimentación, con el fin de elaborar comprensiones más profundas sobre lo estudiado.

MATERIALES Y MÉTODOS

La arquitectura general del sistema está compuesta por un sistema de control principal, sensores, actuadores y el sistema de locomoción. A continuación se describen el análisis y selección de estos componentes.

Sistema de Locomoción

Existen gran variedad de sistemas de locomoción, a continuación se enumeran los principales sistemas que utilizan ruedas (Schmidt et al., 2011):

- Diferencial
- Síncrona
- Triciclo
- Ackerman
- Omnidireccional

Para el caso particular del proyecto Hermes III, el sistema de locomoción utilizado es el omnidireccional, en donde se tienen más de dos ruedas, las cuales poseen la particularidad de desplazarse en cualquier dirección. La

representación matemática vectorial de este sistema de fuerzas, donde la resultante es la dirección y el sentido de movimiento del robot móvil se observa en la Fig. 2. Este diseño permite mayor libertad de movimientos que los sistemas de ruedas clásicos, convirtiéndose en una ventaja en lo referente a capacidad de maniobra principalmente en entornos cargados de obstáculos. A este tipo de vehículos se los conoce como holonómicos. Estos sistemas son capaces de modificar su dirección instantáneamente (considerando masa nula), sin necesidad de rotar previamente. Una forma de conseguir este tipo de tracción/dirección es mediante el uso de ruedas omnidireccionales. En la Fig. 2(a) se muestra el sistema de referencia de la configuración utilizada con cuatro ruedas omnidireccionales y las combinaciones de fuerzas necesarias para rotación y traslación. El modelo de rueda utilizado se aprecia en la Fig. 2(b).

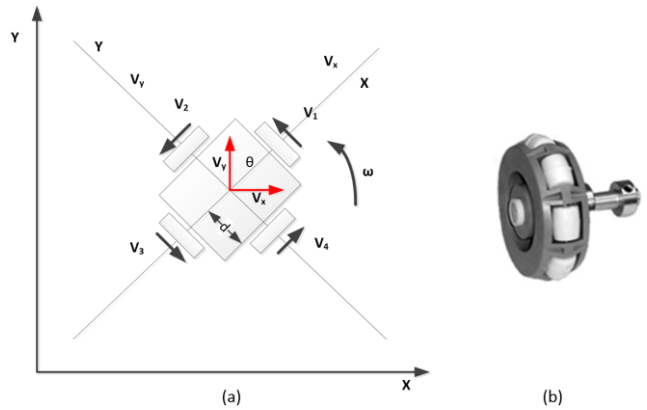

Fig. 2. (a) Sistema omnidireccional modelo de ejes , (b) Rueda Omnidireccional

Sistema de Control Principal

Para el desarrollo del control principal del robot se ha seleccionado la placa de desarrollo NVIDIA Jetson Tegra K1, la cual soporta la instalación del Framework ROS (Robot Operating System). Los factores característicos relevantes de la arquitectura del sistema embebido NVIDIA Jetson (Nvidia, 2017) considerados para el presente diseño son:

Mayor Autonomía: El procesador cuenta con un modo de ultra-ahorro de energía en el cual trabaja exclusivamente un núcleo de extra bajo consumo, cuando el sistema se encuentre bajo poca demanda computacional. Esto significa para el robot Hermes III mayor autonomía de su batería.

- **Alivio de Trabajo de CPU**: Posee doble ISP (Image Signal Processor), estos pre-procesadores de imágenes complementan el trabajo de las CPU al filtrar la información procedente de cámaras de video, interpretando los datos de los sensores y transformándolos en imágenes RGB que son accedidas por la CPU o GPU indistintamente desde una "Memoria Unificada". Adicionalmente estos módulos tienen la capacidad de manejar el auto-foco, el balance de blancos, el ajuste de exposición, etc. Éstas características para el proyecto Hermes III son fundamentales, pues hacen a un procesamiento de imágenes potencialmente más veloz, brindando incluso la posibilidad de procesar las imágenes captadas por las cámaras totalmente en la GPU, aprovechando sus 192 núcleos CUDA.
- **GPU de Alto Rendimiento:** la GPU NVIDIA de 192 núcleos CUDA con micro-arquitectura Kepler y una litografía de 28 nanómetros está desarrollada especialmente para el procesamiento de imágenes y renderizado 3D sin desatender el consumo de energía, ya que el SoC está destinado a dispositivos móviles.

Sensores y Actuadores

Los sensores son una parte fundamental de cualquier robot móvil, son los que permiten adquirir datos acerca del estado del robot y del estado de su entorno en cada instante de tiempo, lo cual es esencial para concebir movimientos controlados en el espacio.

Se utiliza el sensor Kinect (Zhang, 2012), el cual es un periférico de entrada desarrollado por PrimeSense, una compañía Israelí experta en innovación. El dispositivo consta de un controlador que es capaz de percibir y reconocer objetos y movimientos, así como reconocimiento de voz. Para esto hace uso de dos cámaras frontales, una convencional de RGB y un sensor de distancia, y de una serie de micrófonos. El sistema de percepción de profundidad consta de tres partes básicas: el proyector láser de infrarrojos, el sensor CMOS y el microchip que procesa la información. Su funcionamiento se basa en la proyección de un patrón de puntos pseudo-aleatorios y su lectura y triangulación mediante el sensor CMOS.

Este sensor es utilizado con el objetivo de aplicar técnicas de SLAM basadas en capturas de imágenes RGB y de profundidad, lo cual implica el desarrollo de las capacidades de Localización y Mapeado simultáneas tal como se muestra en la Fig. 1. El montaje de este sensor se puede observar en la Fig. 3.

Fig. 3. Sensor Kinect sobre Hermes III

Por otra parte, los actuadores, son los responsables de dotar de movilidad al robot. Implementados con motores de corriente continua que funcionan acorde a los datos producidos por los sensores, luego de ser procesados por el sistema de control.

El sistema de control, que comanda las ruedas del robot, se basa tanto en los datos del entorno como en el movimiento de rotación de las ruedas. Por lo cual, es

necesario sensar de la manera más directa posible, las velocidades de cada una de las ruedas que son las encargadas de producir el movimiento del robot. Esto se implementó mediante "encoders" FC-03 como se muestra en la Fig. 4. Las ondas generadas por los sensores FC-03 son procesadas por un microcontrolador AVR ATmega 128A (Atmel, 2015), haciendo uso de sus timers de 16 bits en modo contador asíncrono, como se muestra en la Fig. 5.

La información de los encoder es obtenida y acondicionada por el AVR ATmega 128A, luego se envía a través de un interface I^2C a una plataforma Arduino. Esta plataforma transforma lo recibido en dato de velocidad de traslación lineal acorde a las dimensiones de la rueda y la envía en formato de tópicos ROS a través de la interfaz Serial que interconecta el Arduino con el Sistema de Control Principal (NVIDIA Jetson TK1). Por cada solicitud de información, el procesador ATmega 128A envía los datos de sus sensores, cada dato es de 16 bits, ver Fig. 6.

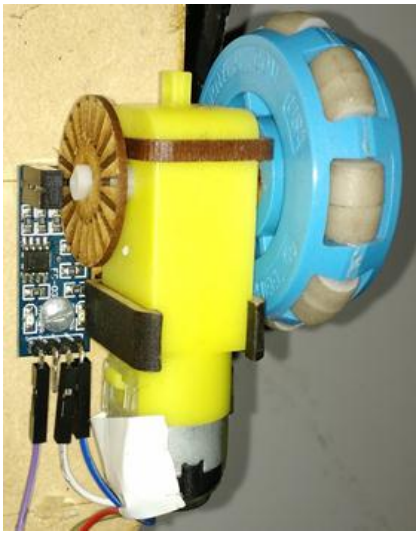

Fig. 4. Encoder implementado en el Hermes III

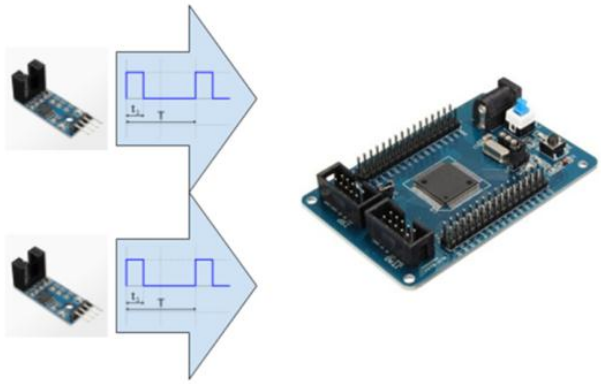

Fig. 5. ATmega128 M128 AVR Minimum Development Board

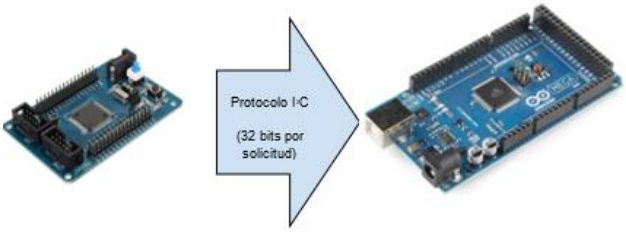

Fig. 6. Comunicación con Arduino Mega 2560

SLAM (Simultaneous Localization and Mapping)

Existen numerosos algoritmos en el ecosistema ROS que se nutren de los datos de sensores de profundidad para generar los resultados deseados de SLAM.

Esta propuesta toma la experiencia del trabajo "Robot Móvil Autónomo para crear mapas 3D en un ambiente acotado" desarrollado en (Caverzasi et al., 2014); como así también, el trabajo de investigación realizado en (da Silva et al., 2017) que realiza una evaluación experimental de los algoritmos compatibles con ROS para sensores RGB-D concluyendo que: "Gmapping es una solución precisa para el mapeo 2D en robots computacionalmente limitados, mientras que RTAB-Map (Real-Time Appearance-Based Mapping) da como resultado soluciones coherentes para un 3D SLAM, aunque con mayores requisitos de procesamiento".

Para la implementación de este proyecto se utilizó el algoritmo denominado RTAB-Map (IntroLab, 2018, Labbe and Michaud, 2013), mostrado en la

Fig. 7. RTAB-Map es el acrónimo de mapeo basado en apariencia en tiempo real. La "detección de lazo cerrado" es el proceso en el cual se intenta encontrar una coincidencia entre la ubicación actual y las ubicaciones visitadas anteriormente en SLAM.

RTAB-Map hace uso de un detector de lazo cerrado bayesiano global junto a un modelo de aproximación de "bolsa de palabras" para determinar la probabilidad de que una nueva imagen provenga de una ubicación anterior o una nueva ubicación.

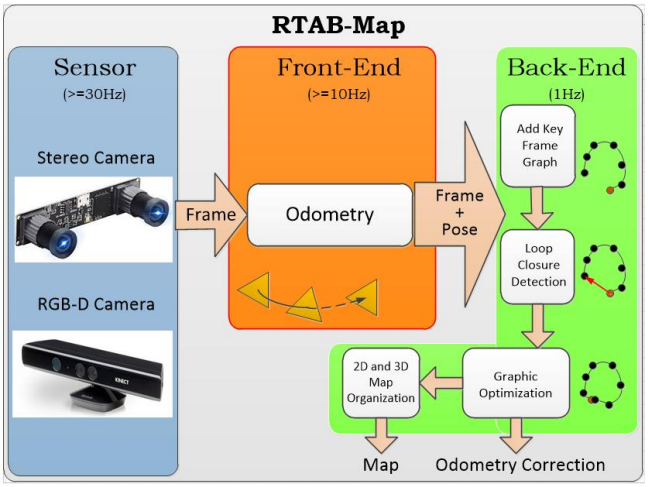

Fig. 7. Algoritmo RTAB-Map

Sistema de Control

El sistema de control es utilizado para que el robot pueda desplazarse con el menor error de trayectoria posible, respondiendo correctamente a los comandos de control tanto en velocidad como en dirección, ajustando la intensidad de movimiento de cada rueda.

En el SoC (Nvidia Jetson TK1), se ha configurado un sistema operativo Ubuntu 14.04 y el Framework ROS en su versión "Indigo Igloo", el cual ya posee integrado nodos de control PID listos para ser configurados y utilizados. Esto permite controlar las velocidades de las ruedas totalmente por software.

Se ha implementado un sistema de control PID para cada rueda del robot. Todas ellas son independientes entre sí, lo

cual simplifico la tarea de desarrollo manteniendo el rendimiento.

A su vez, se cuenta con un coordinador de los 4 controladores PID independientes, denominado PID_General, como se observa en la Fig. 8. El mismo se suscribe al tópico de ROS que proviene del mando de control que se utilice (Joystick, Teclado o Smartphone). Al publicarse un mensaje en dicho tópico, el nodo coordinador obtiene las velocidades Euclidianas deseadas, y con el cálculo de la ecuación (1) se determinan las velocidades individuales de cada rueda, en (Rojas and Förster, 2006) se encuentra un detalle exhaustivo de este cálculo.

$$\begin{matrix} v_1 \\ \vdots \\ v_n \end{matrix} = \begin{pmatrix} -sin\theta_1 & cos\theta_1 & 1 \\ -sin\theta_2 & cos\theta_2 & 1 \\ \vdots & \vdots & \vdots \\ -sin\theta_n & cos\theta_n & 1 \end{pmatrix} * \begin{pmatrix} v_x \\ v_y \\ R\omega \end{pmatrix} \quad (1)$$

Cálculo de Velocidad

Los encoders y el sistema ATmega128, registran la cantidad de transiciones (flanco positivo) que representan el paso por cada una de los cortes del disco ranurado como se observa en la Fig. 4. La velocidad de traslación resultante es calculada a partir de la cantidad de transiciones contadas en un periodo de tiempo, la cantidad de ranuras del disco y las dimensiones de la rueda.

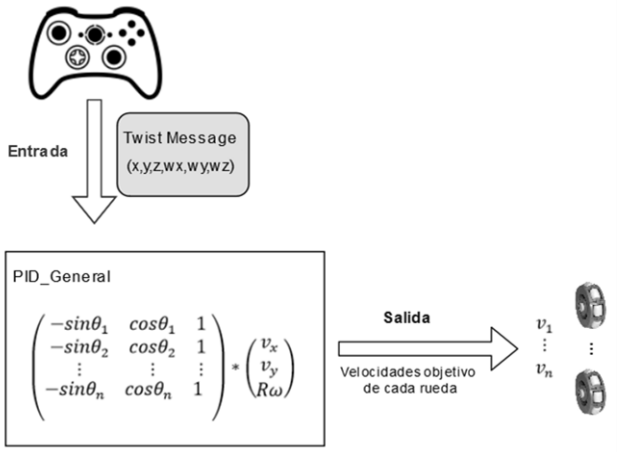

Fig. 8. Nodo ROS - PID_General

Considerando que el disco encoder cuenta con 20 ranuras, según se muestra en la Fig. 9., la cantidad de ranuras medidas por intervalo de tiempo puede transformarse en velocidad con la ecuación (2), la cual es utilizada para obtener la ecuación (3).

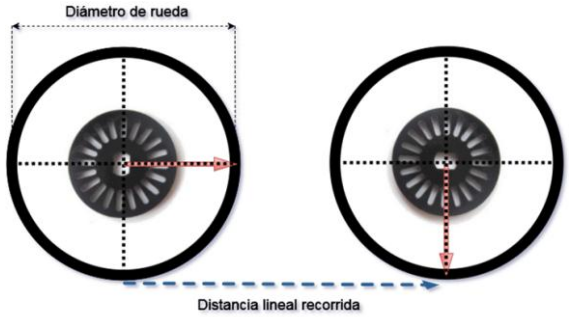

Fig. 9. Distancia en base al radio de la rueda

$$\frac{\frac{ranuras\ contadas}{cantidad\ total\ de\ ranuras} * (\pi * diámetro)}{intervalo\ de\ tiempo} \quad (2)$$

$$\frac{distancia\ lineal\ recorrida}{intervalo\ de\ tiempo} = velocidad\ en\ \frac{m}{s} \quad (3)$$

Importancia de los FPS en la captura de imágenes en SLAM

Para el correcto funcionamiento del algoritmo de detección de obstáculos y posicionamiento por imágenes es fundamental la velocidad de captura/procesamiento de las mismas. Con el fin de determinar esta hipótesis se realizó los cálculos necesarios para graficar la relación como se observa en la Fig. 10. Para contrastar los resultados a continuación se analizan dos casos de la curva, una imagen capturada cada 90 ms (10 fps) y otra cada 33 ms (30 fps):

Considerando una velocidad de desplazamiento del robot de 1 m/s, se tiene:

- Caso 1 (10fps)
  El robot se moverá a ciegas aproximadamente 10 cm.
- Caso 2 (30 fps)
  El robot se moverá a ciegas aproximadamente 3,3 cm.

En la Fig. 10 se observa la curva que relaciona las distancias recorridas a ciegas versus velocidad captura/procesamiento.

Este cálculo, la capacidad computacional y capacidad de captura de frames del SoC, NVIDIA Jetson Tegra K1, con respecto a sistemas alternativos de valores similares han motivado su elección.

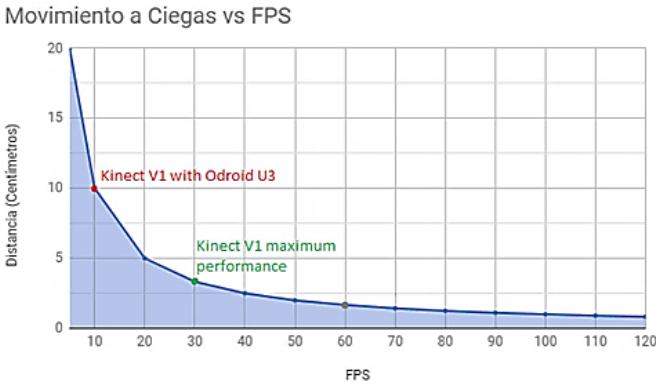

Fig. 10. Distancia a ciegas vs FPS

Precisión del algoritmo SLAM utilizado

Se han realizado pruebas que demuestran la precisión, en cuanto al desplazamiento del robot, que es capaz de detectar el algoritmo RTAB-Map y cómo la precisión varía acorde a los datos que recibe del sensor de profundidad Kinect.

Antes de realizar las pruebas, se midió en laboratorio la distancia mínima que es capaz de detectar el dispositivo Microsoft Kinect V1, colocando frente al mismo una pizarra donde se ha anotado la distancia a la que se encontraba del sensor , como se detalla en la Fig. 11. Se determinó que si un objeto se ubica frente al sensor Kinect a una distancia menor a 46.5 centímetros, no será detectado, esta lo muestra la Fig. 11.

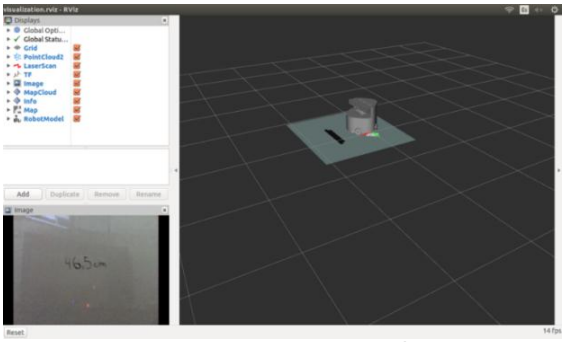

Fig. 11. Experimento distancia mínima

Para el primer caso de prueba se ubica el sensor Kinect frente a una pared de color homogéneo y se aleja el robot 0,5 m de manera perpendicular a la misma.

Cabe aclarar que cada cuadrícula de las siguientes figuras representa un área de 0.5 m por 0.5 m.

En la Fig. 12, Fig. 13 y Fig. 14 puede apreciarse cómo, a pesar de mover al robot perfectamente en línea recta perpendicular a la pared, la medición posee error acumulativo, por lo que al regresar al punto de partida la diferencia es de 30 cm aproximadamente.

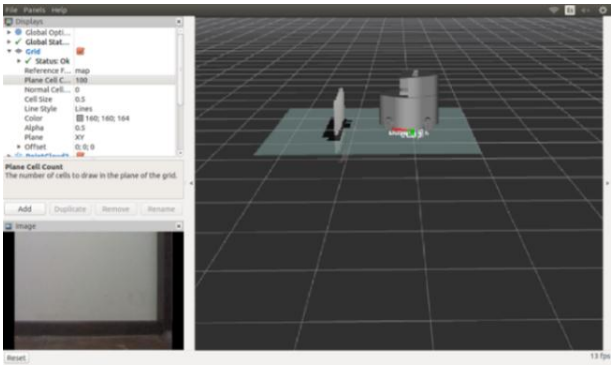

Fig. 12. Posición inicial sin referencia (vista superior)

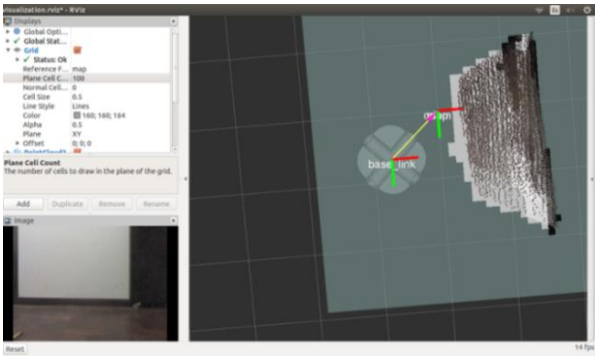

Fig. 13. Desplazamiento de medio metro sin referencia (vista inferior)

Para confirmar la dependencia de la exactitud del algoritmo SLAM con el nivel de detalle que se obtiene de las imágenes tanto RGB como de profundidad del sensor Kinect, se ha realizado una segunda prueba, ver Fig. 15, Fig. 16 y Fig. 17, colocando un objeto de una estructura tridimensional sobre la pared y se procedió a repetir exactamente el mismo movimiento que en la prueba anterior. Se comprueba una gran mejora de la exactitud en las mediciones del algoritmo cuando lo hace en base a datos de profundidad heterogéneos, el error para el mismo caso fue aproximadamente de 6 cm.

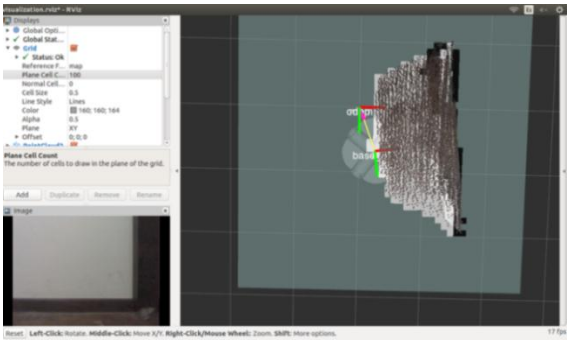

Fig. 14. Regreso al punto de partida sin referencia (vista inferior)

Esto muestra que los resultados son más precisos cuando se posee un objeto heterogéneo delante del sensor de profundidad el cual puede usarse como punto de referencia inequívoco y realizar en base al mismo cálculo de distancia que cuando posee únicamente una pared homogénea delante.

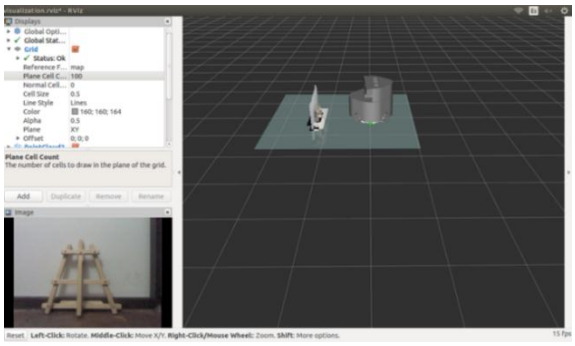

Fig. 15. Posición inicial con referencia (vista superior)

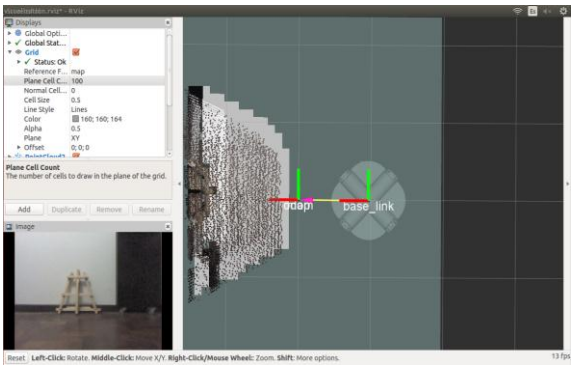

Fig. 16.Desplazamiento de medio metro con referencia (vista inferior)

CASOS DE PRUEBAS

Se realizaron los siguientes casos de pruebas:

Prueba 1

Mover el robot una distancia de 1 metro en línea recta en cualquier dirección.

Para probar la calidad del control de trayectoria con el sistema de control implementado, se grabó en video a cámara fija el desplazamiento en línea recta del robot, esto se aprecia en la Fig. 18. Puede observarse un desvío mayormente pronunciado al inicio del movimiento del robot, debido a que en los primeros 0,35 metros, el sistema de control de cada rueda se encuentra en su estado transitorio (Ogata, 2010). Una vez que se alcanza el estado estable, el robot continúa en línea recta, como puede verse a partir del tercer punto de la Fig. 18.

Si se consideran ambos estados, transitorio y estable, el error en línea recta fue de 10 $^{cm}/_{m}$ con una incertidumbre de 1 cm.

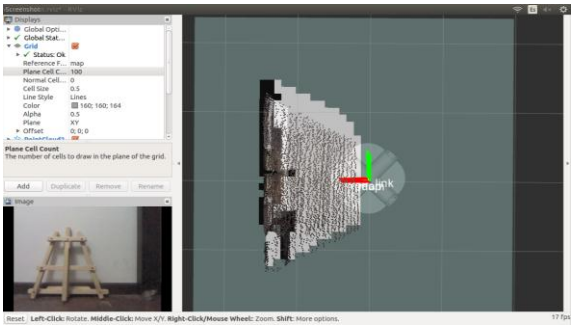

Fig. 17. Regreso al punto de partida con referencia (vista inferior)

Si se descarta el estado transitorio (midiendo a partir del tercer punto), y extrapolando la trayectoria en línea recta en base a los puntos 3 y 4 de la Fig. 18, se traza una línea roja ideal que dista unos 2 cm con una incertidumbre de 1 cm, desde la posición final del robot.

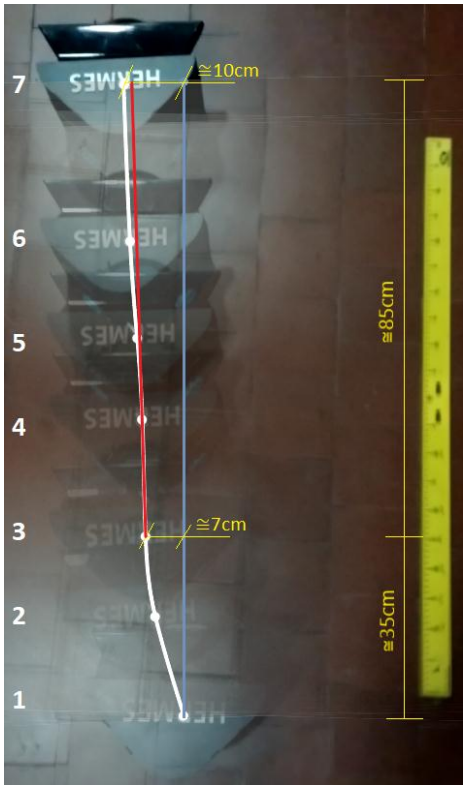

Fig. 18. Prueba de trayectoria en línea recta del robot Hermes III

Prueba 2

Poner en funcionamiento el robot en un ambiente cerrado y que recorra la mayor superficie posible del lugar.

Se ha logrado realizar un mapa muy preciso del entorno físico que rodea al robot siempre y cuando el algoritmo no pierda la relación entre la imagen actual capturada y una anterior. Puede verse el mapa generado del Laboratorio de Arquitectura de Computadoras en la Fig. 19.

RESULTADOS

El robot fue capaz de reconocer su posición en tiempo real dentro del entorno donde se encontraba. Esto responde exitosamente a una de las preguntas fundamentales para que un robot fuese totalmente autónomo, la cual es "¿dónde estoy?". En cuanto al mapeo, se han obtenido resultados satisfactorios, siempre que se cumpla con los limites de velocidad de captura/procesamiento establecidos para el sistema y manteniéndose a una distancia mayor a la mínima posible para no perder referencias y detener el proceso de mapeo. Se ha logrado mejorar la estabilidad con la modificación de los parámetros, esto hace a RTAB-Map un algoritmo de SLAM flexible y totalmente modular e integrado con el hardware del robot, mediante el sistema de comunicación por tópicos que ofrece ROS.

Lo anterior puede considerarse una prueba de integración del funcionamiento de todos los componentes del robot Hermes III.

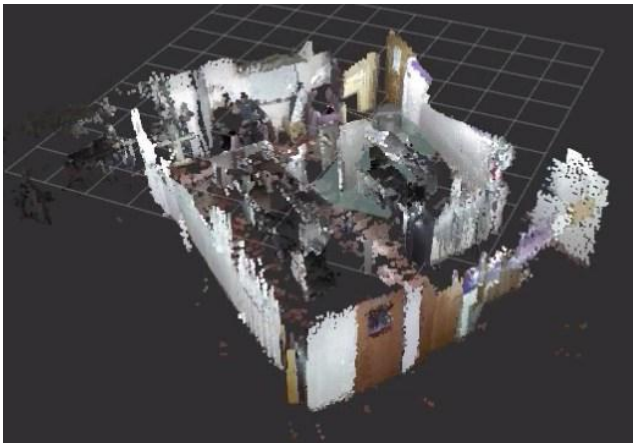

Fig. 19. Mapa 3D del Laboratorio de Arquitectura de Computadoras

Se ha logrado comprobar el correcto funcionamiento de los sistemas embebidos involucrados, Arduino Mega 2560, ATmega128 AVR Minimum Development Board, Microsoft Kinect V1 y NVIDIA Jetson Tegra K1, demostrando una comunicación eficaz entre ellos.

En cuanto a las pruebas sobre los límites de velocidad posibles, son los siguientes:

- Frecuencia mínima: 15.3 Hz (por desbordamiento del contador de 16 bits en placas ATmega). Esto es equivalente a 12 $^{cm}/_{s}$ según la ecuación (2).
- Frecuencia máxima: 31.25 KHz (con lo cual se puede medir con holgura la velocidad máxima a la que es capaz de funcionar el robot Hermes III).

Refiriéndonos a la frecuencia como la cantidad de ranuras por segundo que el sistema implementado es capaz de contar.

En el algoritmo SLAM, es importante destacar el incremento de precisión en las mediciones obtenidas cuando se realizan los cálculos en base a datos de profundidad heterogéneos.

CONCLUSIONES

Se logró construir un robot omnidireccional capaz de realizar un mapa en 3D del entorno que lo rodea y al mismo tiempo ubicarse dentro del mismo. Estas capacidades sientan las bases para el desarrollo de un robot totalmente independiente del control humano, es decir, que sea capaz de auto navegar una habitación, evitando obstáculos y realizando trayectorias de barrido eficientes para lograr un mapa tridimensional en tiempo real.

Ésta versión del robot que se ha desarrollado íntegramente en el Laboratorio de Arquitectura de Computadoras, de la Facultad de Ciencias Exactas, Físicas y Naturales de la Universidad Nacional de Córdoba, ha significado un salto tecnológico con respecto a su predecesor, el Hermes II.

Desde el punto de vista académico, el robot desarrollado representa una adecuada base para el estudio del Framework utilizado. El denominado Robot Operating System (ROS), tiene un gran futuro por su naturaleza Open Source y la gran comunidad que posee, conformada tanto por investigadores de universidades de gran prestigio, como por estudiantes y aficionados.

Por su desarrollo modular, el robot diseñado, es fácilmente escalable por lo que puede ser dotado de nuevas capacidades.

Respecto a la utilización de un sistema omnidireccional heredado de su antecesor Hermes II, si bien su control trae aparejado una mayor complejidad matemática que otros sistemas de locomoción, facilita su implementación mecánica debido a que los ejes de las ruedas permanecen fijos y permite modificar la trayectoria del robot sin cambiar la orientación del mismo.

Finalmente, la placa de desarrollo Nvidia Jetson Tegra K1, ha probado tener la capacidad computacional suficiente para realizar cálculos complejos, tales como los vinculados al procesamiento de los datos provenientes del sensor Microsoft Kinect y al sistema de control que comanda los movimientos del robot.

TRABAJOS FUTUROS

Existe una gran oportunidad de mejora con respecto a la performance de SLAM del robot, para lo cual se está realizando una investigación con sensores de profundidad que capturan imágenes a una mayor velocidad y a una mejor resolución. También se planea integrar en el robot un scanner laser tipo LIDAR (Light Detection and Ranging o Laser Imaging Detection and Ranging) con el fin de complementar los datos obtenidos por el Kinect y así mejorar la precisión del SLAM.

Por otra parte, otro equipo de trabajo está abocado a transformar los datos de velocidades de las ruedas, en datos odométricos, y utilizarlos en el algoritmo RTAB-Map para hacerlo más robusto.

REFERENCIAS

[1] J. Pulido Fentanes, "Exploración y reconstrucción tridimensional de entornos mediante robots móviles - Valladolid," 2012.

[2] J. Borenstein, H. Everett, and L. Feng, Navigating mobile robots: Systems and techniques: AK Peters, Ltd., 1996.

[3] R. E. Ayme, O. Micolini, L. O. Ventre, A. B. G. Cabral, and S. S. Sagripanti, "Hermes II: Robot Educativo Holonómico para la Enseñanza en Ingeniería," Revista de la Facultad de Ciencias Exactas, Físicas y Naturales, vol. 7, pp. 63-71, 2020.

[4] O. S. R. Foundation. (2020). ROS Ubuntu ARM install of ROS Indigo-Available: http://wiki.ros.org/indigo/Installation/UbuntuARM.

[5] W. Newman, A systematic approach to learning robot programming with ROS: CRC Press, 2017.

[6] A. Caverzasi, F. Saravia, O. Micolini, L. Mathé, and L. F. Lichtensztein, "Robot móvil autónomo para crear mapas 3D en un ambiente acotado," in 2014 IEEE Biennial Congress of Argentina (ARGENCON), 2014, pp. 786-791.

[7] L. F. Lichtensztein, O. Micolini, and M. Cebollada, "“Hermes”: Sistema robótico embebido para la educación," in Biennial Congress of Argentina (ARGENCON), 2014 IEEE, 2014, pp. 310-315.

[8] O. Mubin, C. J. Stevens, S. Shahid, A. Al Mahmud, and J.-J. Dong, "A review of the applicability of robots in education," Journal of Technology in Education and Learning, vol. 1, p. 13, 2013.

[9] D. Schmidt, C. Hillenbrand, and K. Berns, "Omnidirectional locomotion and traction control of the wheel-driven, wall-climbing robot, Cromsci," Robotica, vol. 29, pp. 991-1003, 2011.

[10] Nvidia. (2017). Nvidia Kepler Architecture. Available: https://www.nvidia.com/en-us/data-center/tesla-product-literature/

[11] Z. Zhang, "Microsoft kinect sensor and its effect," IEEE multimedia, vol. 19, pp. 4-10, 2012.

[12] Atmel. (2015). Atmel ATmega 128A Risc Microcontroller. Available: http://ww1.microchip.com/downloads/en/devicedoc/atmel-8151-8-bit-avr-atmega128a_datasheet.pdf

[13] B. M. da Silva, R. S. Xavier, T. P. do Nascimento, and L. M. Gonsalves, "Experimental evaluation of ROS compatible SLAM algorithms for RGB-D sensors," in 2017 Latin American Robotics Symposium (LARS) and 2017 Brazilian Symposium on Robotics (SBR), 2017, pp. 1-6.

[14] U. d. S.-. IntroLab. (2018). IntroLab RTAB-Map. Available: http://introlab.github.io/rtabmap/

[15] M. Labbe and F. Michaud, "Appearance-based loop closure detection for online large-scale and long-term operation," IEEE Transactions on Robotics, vol. 29, pp. 734-745, 2013.

[16] R. Rojas and A. G. Förster, "Holonomic control of a robot with an omnidirectional drive," KI-Künstliche Intelligenz, vol. 20, pp. 12-17, 2006.

[17] K. Ogata, Ingeniería de control moderna: Pearson Educación, 2010.