DESDE 2013
https://repository.uaeh.edu.mx/revistas/index.php/icbi/issue/archive
Pädi Boletín Científico de Ciencias Básicas e Ingenierías del ICBI
ISSN: 2007-6363

Publicación Semestral Pädi Vol. 12 No. Especial 2 (2024) 93–99

Plataforma experimental para el modelado y control de robots omnidireccionales
Experimental platform for the modeling and control of omnidirectional robots

L. Y. Leal-Ramos [a], L. A. Jonguitud-Indalecio [a], M. F. Ortiz-Michimani [a], J. Díaz-Téllez [a,b], J. P. Sánchez-Santana [a,c], J. F. Guerrero-Castellanos [a,*]

[a] *Facultad de Ciencias de la Electrónica, Benemérita Universidad Autónoma de Puebla, 72000, Puebla, Puebla, México.*
[b] *Instituto Tecnológico de Puebla, Del Tecnológico 420, Corredor Industrial la Ciénega, 72220, Puebla, Puebla, México.*
[c] *Universidad Politécnica de Puebla, Tercer Carril del Ejido, Serrano s/n, Cuanalá, 72640 Puebla, Puebla, México.*

Resumen

En este artículo es presentada una plataforma experimental para el desarrollo de algoritmos de robótica, control y aprendizaje. La arquitectura propuesta basada en ROS es abierta, permitiendo la integración de diferentes sensores, unidades de procesamiento y robots. Un Controlador por Rechazo Activo de Perturbaciones (CRAP) es diseñado sobre un robot omnidireccional para validar la plataforma desarrollada. Incertidumbres paramétricas, la fricción de las ruedas sobre la superficie y las perturbaciones externas son agrupadas en una perturbación total, la cual es estimada por un Observador de Estado Extendido y y compensada mediante un término feedforward en la ley de control. El robot omnidireccional se puede comunicar a través de ROS mediante un sistema de captura de movimiento (servidor). Se presentan los resultados de las simulaciones y experimentos en tiempo real. El algoritmo de control es ligero y fácil de implementar y ajustar en sistemas embebidos con pocos recursos computacionales o de bajo costo.

Palabras Clave: Robot Omnidireccional, CRAP, ESO, ROS.

Abstract

In this paper, an experimental platform for the development of robotics, control, and learning algorithms is presented. The proposed ROS architecture is open, allowing the integration of different sensors, processing units, and robots. An Active Disturbance Rejection Control (ADRC) is designed for an omnidirectional mobile robot to validate the proposed platform. Parametric uncertainties, wheel friction on the surface, and external disturbances are lumped as a total disturbance, which is estimated by an Extended State Observer (ESO) and compensated via a feedforward term in the control law. The omnidirectional robot can communicate through ROS with a motion capture system (server). Simulation and experimental results in real-time are presented. The control algorithm is lightweight and easy to implement and adjust in embedded systems with low computational resources or low-cost processors.

Keywords: Omnidirectional robot, ADRC, ESO, ROS.

1. Introducción

En esta sección se proporciona un panorama general de los robots móviles, su importancia y transcendencia en la actualidad, así como la clasificación de los mismos. Un énfasis especial es puesto al robot omnidireccional. Posteriormente, se describe el ahora tan conocido sistema ROS y sus ventajas de uso. Finalmente se describen las contribuciones del presente trabajo y la organización del artículo.

1.1. Robots móviles

En los últimos años, los robots móviles autónomos han sido motivo de investigación en la academia y la industria. Los robots móviles pueden tener diferentes configuraciones que dependen principalmente del tipo de ruedas que utilicen (motoras, pasivas y omnidireccionales), así como su principio de funcionamiento y dinámica. Debido a su maniobrabilidad, precisión, autonomía, capacidad de carga y flexibilidad, estos robots móviles se han utilizado para diferentes tareas, principalmente

*Autor para correspondencia: fermi.guerrero@correo.buap.mx
Correo electrónico: luis.lealra@alumno.buap.mx (Luis Yahir Leal-Ramos), luis.jonguitud@alumno.buap.mx (Luis Antonio Jonguitud-Indalecio), maria.ortizmi@alumno.buap.mx (María Fernanda Ortiz-Michimani), quatro0166@gmail.com (Juan Díaz-Téllez), jose.sanchez@uppuebla.edu.mx (José Pedro Sánchez-Santana), fermi.guerrero@correo.buap.mx (José Fermi Guerrero-Castellanos).

Historial del manuscrito: recibido el 15/12/2023, última versión-revisada recibida el 04/03/2024, aceptado el 08/03/2024, publicado el 22/04/2024. **DOI:** https://doi.org/10.29057/icbi.v12iEspecial2.12282

en el sector industrial. Por ejemplo, en agricultura (Pak *et al.*, 2022; Gao *et al.*, 2018), movilidad personal (Curiel-Olivares *et al.*, 2021), entrega de paquetes (Lee *et al.*, 2021), servicios de entrega (Hu *et al.*, 2022; Wang *et al.*, 2010), vigilancia (Parada-Salado *et al.*, 2018), búsqueda y rescate (Niroui *et al.*, 2019). La clasificación de robots móviles con ruedas, convencionales u omnidireccionales, con base en sus propiedades estructurales y grados de libertad, que determinan su desplazamiento en un espacio de operación, se dividen en cinco categorías (Campion *et al.*, 1996), de acuerdo a los grados de movilidad (δ_m) y grados de direccionabilidad (δ_s), y la suma de ambos será el total de grado de libertad del robot. Considerando la notación (δ_m, δ_s) los robots móviles propulsados por ruedas se dividen en tipo: (2,0), (2,1), (1,1), (1,2) y (3,0) o comúnmente nombrado como robot móvil omnidireccional, del tipo holónomo. Este último, tiene la habilidad de moverse sobre una superficie plana con gran destreza sin restricción de su orientación, por consecuente, realiza tareas en un entorno industrial reducido.

1.2. Sistema Operativo Robótico ROS

El Sistema Operativo Robótico (Robot Operating System, ROS) es una plataforma de desarrollo de software para aplicaciones de robótica. Aunque no es un sistema operativo realmente, ROS ofrece una estructura de comunicación en capas hospedado en un sistema operativo (Quigey *et al.*, 2009).

ROS actúa como un meta-sistema operativo para robots, proporciona abstracción de hardware, control de dispositivos de bajo nivel, paso de mensajes entre procesos y gestión de paquetes (Bihlmaier y Wörn, 2016). ROS permite administrar diferentes capas, paralelizar procesos, y comunicarse con diferentes sensores, unidades de procesamiento y robots. Además, ROS realiza cómputo distribuido en múltiples nodos (considerados también como ejecutables) dentro de un grafo, alojando en cada uno procesos independientes. Generalmente cada nodo puede contener el procesamiento de la información de cada sensor o el cómputo de los algoritmos de control y movimiento que describen el comportamiento del robot. A pesar de que cada nodo funciona de manera independiente, es posible que puedan comunicarse entre sí mediante mensajes. Los nodos junto con los archivos de configuración de un proyecto se organizan en paquetes, unidades de almacenamiento replicables y fácilmente distribuibles. Entre las ventajas de ROS, destaca que es una plataforma Open-Source, cualidad fomentada desde años atrás (Cousins *et al.*, 2010), alentando a la comunidad a publicar abiertamente los paquetes que han desarrollado, originándose así diversos paquetes que proporcionan funcionalidades típicas en un robot, e.g. visión por computadora, navegación, detección de obstáculos, cinemática y dinámica.

Debido a la flexibilidad de integración con software de terceros, es posible vincular ROS con MATLAB-Simulink para diferentes aplicaciones, incluyendo robots móviles (Galli *et al.*, 2017), añadiendo una plataforma para desarrollar, probar y ejecutar prototipos para robots omnidireccionales.

Algunos trabajos basados en ROS se pueden encontrar en (Gentilini *et al.*, 2021; Martin *et al.*, 2021; Mišeikis *et al.*, 2020; Haghighat y Sadeghnejad, 2022). Sin embargo, la mayoría de los robots que integran el ROS-middleware pertenecen a empresas ya establecidas. Son robots complejos, caros y que requieren una instrumentación alta.

1.3. Contribuciones

La plataforma experimental presentada en este trabajo, ha sido desarrollada de manera colaborativa entre estudiantes de tres niveles educativos, a saber, licenciatura, maestría y doctorado. Dicha plataforma, a su vez, ha permitido el rápido prototipado de robot heterogéneos, introduciendo conceptos como control de sistemas no lineales, control cooperativo, estabilidad, diseño de sistemas embebidos, navegación y desarrollo de algoritmos de aprendizaje maquina, conceptos que también son abordados en los programas de los niveles educativos antes mencionados.

Las contribuciones del presente trabajo se enumeran a continuación:

- Diseño de una plataforma de arquitectura abierta con capacidades de comunicación en ROS, que integra diferentes capas paraleliza procesos, tareas y comunicación entre sensores, actuadores, coprocesadores y otros robots.
- Desarrollo de un sistema embebido basado en ROS para un robot móvil omnidireccional (3,0).
- Diseño e implementación un observador de estado extendido al algoritmo de control para estimar las incertidumbres paramétricas y perturbaciones externas.
- Realización de pruebas en tiempo real para validar la ley de control.

El resto del documento está estructurado de la siguiente manera. La sección II presenta el modelo matemático del robot móvil omnidireccional tipo (3,0). La sección III se presenta el diseño de la estrategia de control ADRC. En la sección IV se ilustran las simulaciones numéricas. En la sección V se presenta el diseño de la arquitectura propuesta y los resultados experimentales. Finalmente, se dan las conclusiones y trabajo futuro en la sección VI.

2. Ecuaciones de movimiento del robot omnidireccional

La representación matemática del robot omnidireccional se obtendrá a partir de considerar que se mueve sobre una superficie plana, que el deslizamiento es despreciable y que no existe elementos flexibles en su estructura. En la Figura 1 se muestra la vista isométrica de robot móvil omnidireccional, $\{w\}$ representa el sistema de coordenadas inercial (plano de movimiento) y $\{m\}$el sistema de referencia móvil. Sea $\dot{\eta}_w = [\dot{x}_w\ \dot{y}_w\ \dot{\phi}_w]^T$ y $\dot{\eta}_m = [\dot{x}_m\ \dot{y}_m\ \dot{\phi}_m]^T$ la velocidad lineal y angular del robot móvil con respecto al sistema de coordenadas inercial $\{w\}$ y al sistema de coordenadas del móvil $\{m\}$, respectivamente. Considerando los ángulos descritos por el eje Y_m y el eje axial de la rueda como $\delta_1 = \frac{\pi}{6}$ y $\delta_3 = \frac{\pi}{3}$. Basándose en el trabajo Guerrero-Castellanos *et al.* (2014), el modelo cinemático se puede representar en (1). Cabe mencionar que los valores de los ángulos δ_1 y δ_2 forman parte del diseño del robot móvil ya que maximizan su destreza. Además, el vector $\xi = [\xi_{x_m}\xi_{y_m}]^T$ representa la perturbación a la velocidad de desplazamiento, debido a las incertidumbres paramétricas en los actuadores y llantas, fricciones e irregularidades del terreno, perturbaciones externas, entre otras.

$$\begin{aligned}\dot{x}_w &= (\dot{x}_m + \xi_{x_m})\cos\phi_w - (\dot{y}_m + \xi_{y_m})\sin\phi_w \\ \dot{y}_w &= (\dot{x}_m + \xi_{x_m})\sin\phi_w + (\dot{y}_m + \xi_{y_m})\cos\phi_w \\ \dot{\phi}_w &= \dot{\phi}_m\end{aligned} \tag{1}$$

El mapeo entre la velocidad lineal de las ruedas y la velocidad angular y lineal del sistema de coordenadas del robot móvil (3,0) se representa en (2), donde $\dot{\theta} = [\dot{\theta}_1\ \dot{\theta}_2\ \dot{\theta}_3]^T$ es la velocidad angular de las ruedas, r_r es el radio de las ruedas y la distancia entre el centro geométrico del robot móvil y la rueda se representa como l.

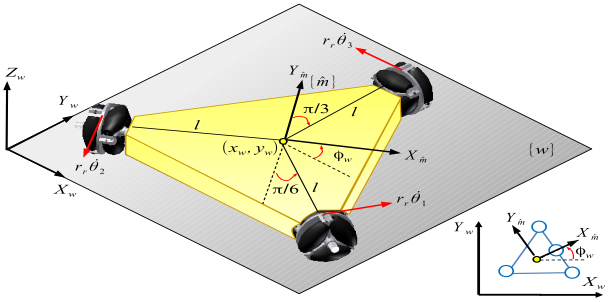

Figura 1: Esquemático del robot móvil (3,0).

$$\begin{aligned}\dot{\theta}_1 r_r &= \tfrac{1}{2}\sqrt{3}\dot{x}_m + \tfrac{1}{2}\dot{y}_m + l\dot{\phi}_m, \\ \dot{\theta}_2 r_r &= -\dot{y}_m + l\dot{\phi}_m, \\ \dot{\theta}_3 r_r &= -\tfrac{1}{2}\sqrt{3}\dot{x}_m + \tfrac{1}{2}\dot{y}_m + l\dot{\phi}_m.\end{aligned} \tag{2}$$

3. Diseño de la estrategia de control

El buen desempeño de las estrategias de control depende directamente de su diseño, cálculo y forma de implementación. Puesto que en el presente trabajo, se usará el modelo cinemático para el diseño de las leyes de control, las velocidades en el sistema de coordenadas móvil fungirán como señales de control. Para ello, y con fines de claridad, se definen el siguiente vector de posicón en el sistema de coordenadas inerciales $x := [x_1\ x_2\ x_3]^T := [x_w\ y_w\ \phi_w]^T$ y el vector de control $u = [u_1\ u_2\ u_3] := [\dot{x}_m\ \dot{y}_m\ \dot{\phi}_m]^T$. Entonces, el modelo (1) se reescribe como

$$\Sigma_{Om} := \begin{cases}\dot{x}_1 = (u_1 + \xi_{x_m})\cos x_3 - (u_2 + \xi_{y_m})\sin x_3 \\ \dot{x}_2 = (u_1 + \xi_{x_m})\sin x_3 + (u_2 + \xi_{y_m})\cos x_3 \\ \dot{x}_3 = u_3\end{cases}. \tag{3}$$

Para estimar los componentes del vector de perturbaciones se propone el siguiente observador de orden extendido

$$\Sigma_{ESO} := \begin{cases}\dot{\hat{x}}_1 = (u_1 + \hat{\xi}_{x_m})\cos x_3 - (u_2 + \hat{\xi}_{y_m})\sin x_3 \\ \qquad + l_1(x_1 - \hat{x}_1) \\ \dot{\hat{x}}_2 = (u_1 + \hat{\xi}_{x_m})\sin x_3 + (u_2 + \hat{\xi}_{y_m})\cos x_3 \\ \qquad + l_1(x_2 - \hat{x}_2) \\ \dot{\hat{\xi}}_{x_m} = l_0(x_1 - \hat{x}_1) \\ \dot{\hat{\xi}}_{y_m} = l_0(x_2 - \hat{x}_2)\end{cases}. \tag{4}$$

donde las ganancias l_1 y l_2 se eligen tales que el polinomio característico de la dinámica del error coincida con el polinomio característico deseado dado por $P(\lambda) = \lambda^2 + 2\omega_n\zeta\lambda + \omega_n^2$. De esta forma, $l_1 = 2\omega_n\zeta$, $l_0 = \omega_n^2$.

El control por su parte toma la siguiente forma:

$$\Sigma_C := \begin{cases}u_1 = -\cos(x_3)r_1 + \sin(x_3)r_2 \\ u_2 = -\sin(x_3)r_1 - \cos(x_3)r_2 \\ u_3 = r_3\end{cases}. \tag{5}$$

con

$$\Sigma_r := \begin{cases}r_1 = \dot{x}_1^d - \hat{\xi}_{x_m} + K(x_1^d - x_1) \\ r_2 = \dot{x}_2^d - \hat{\xi}_{y_m} + K(x_2^d - x_2) \\ r_3 = \dot{x}_3^d + K(x_3^d - x_3)\end{cases}. \tag{6}$$

donde x_i^d representa la trayectoria deseada con $i \in \{1, 2, 3\}$.

El sistema de tracción de cada vehículo está compuesto por un motor de corriente continua, la llanta y un encoder permitiendo obtener información de la velocidad angular de la flecha de cada motor. Con esta información, la velocidad de cada motor del robot móvil puede ser controlada de forma local por ejemplo con un control PI. No obstante, en este trabajo la dinámica en lazo abierto de cada motor fue identificada y el modelo inverso es utilizado para generar el voltaje necesario tal que se siga la trayectoria deseada mediante el control (5). Las perturbaciones en la velocidad causadas por la rugosidad de la superficie donde navega el robot, así como las incertidumbres paramétricas en los motores, son concentradas en los términos de perturbación $\xi = [\xi_{x_m}\xi_{y_m}]^T$ el cual es estimado mediante (4) y compensado con (5)-(6). Este esquema de control es lo que se conoce como control por rechazo activo de perturbaciones ADRC (del inglés, Active Disturbance Rejection Control). La prueba de estabilidad del sistema en lazo cerrado (robot móvil-control-observador) se puede revisar dirigiendose a (Guerrero-Castellanos *et al.*, 2018).

4. Resultados en Simulación

Esta sección presenta las simulaciones realizadas para validar el control ADRC. Las simulaciones se realizaron en el entorno de MATLAB/Simulink. Se realiza el control de seguimiento de trayectoria de una Lemniscata. Matemáticamente la referencia deseada esta descrita por la siguiente expresión:

$$\Sigma_{X_d} := \begin{cases}x_1^d = 3(0{,}216\sin(0{,}314t) - 0{,}589\sin(0{,}157t)) \\ x_2^d = 3(0{,}336\sin(0{,}314t) + 0{,}378\sin(0{,}157t)) \\ x_2^d = 0{,}3;\end{cases}. \tag{7}$$

Se ha agregado al modelo del robot omnidireccional tipo (3,0) una perturbación externa variante en el tiempo (8) posiblemente causada por deslizamiento, fricciones, irregularidades del terreno o desgaste del motor.

$$\Sigma_\xi := \begin{cases}\xi_{x_m} = 0{,}1(1 + e^{(-\sin(3*t)*\sin(t))})\cos(t) \\ \xi_{y_m} = 0{,}1(1 + e^{(-\sin(3*(t-2))*\sin(t))})\cos(t-2)\end{cases}. \tag{8}$$

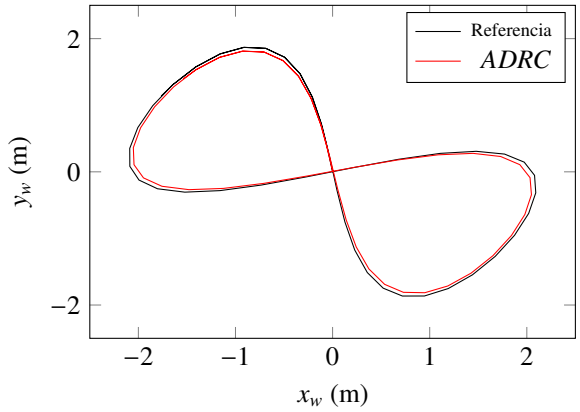

Figura 2: Posición del robot omnidireccional para la trayectoria de lemniscata.

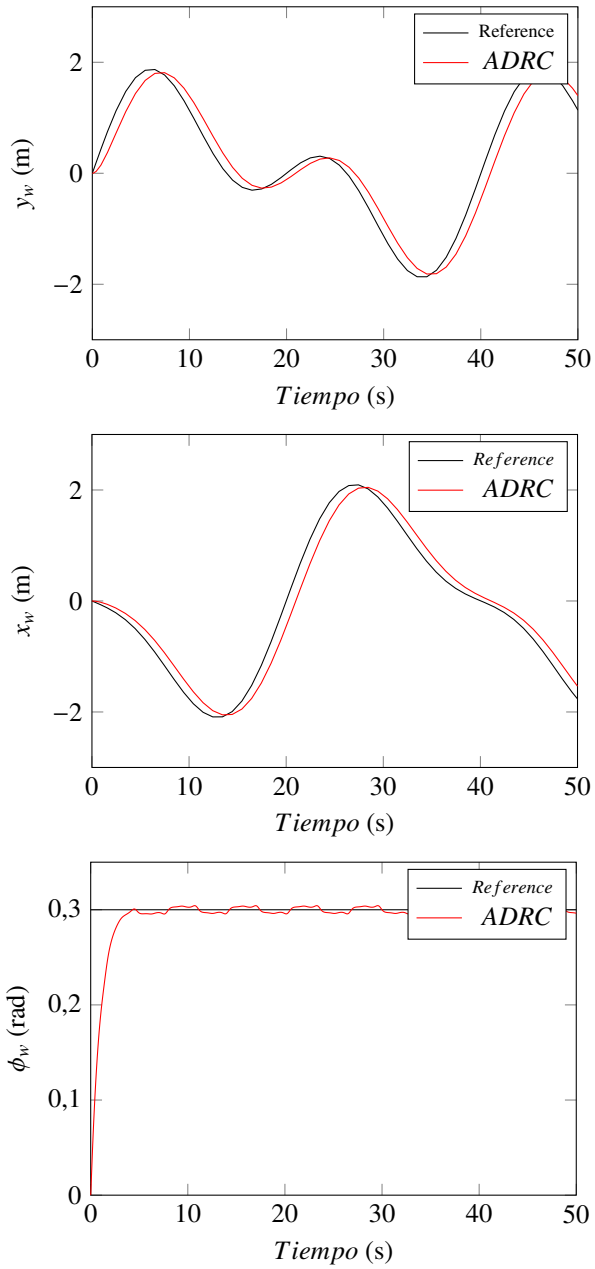

Figura 3: Evolución de la posición del robot omnidireccional.

La Figura 2 muestra la evolución de la trayectoria Lemniscata que sigue el robot móvil omnidireccional en el sistema de coordenadas inercial $\{w\}$. En la Figura 3 muestra la evolución de las variables x_w, y_w y ϕ_w del robot omnidireccional. Como puede verse en 2, el control ADRC es capaz de eliminar las perturbaciones externas, obtener una respuesta suave y rápida. La referencia la sigue con exactitud a pesar de no contar con la información del disturbio.

5. Resultados experimentales

5.0.1. Descripción del robot experimental

El desarrollo experimental se llevo a cabo con un robot omnidirecional con tres ruedas, del tipo (3,0) que se caracteriza por tener tres grados de libertad: adelante-atrás, izquierda-derecha y rotación, gracias a que es holónomo este permite cambio de direcciones sin necesidad de realizar rotaciones, lo que da un manejo más sencillo, debido a que las configuraciones de los motores para generar movimientos son únicas para cada trayectoria. El robot omnidireccional de la Figura 4 cuenta con los siguientes elementos:

- Un chasis con dos placas de soporte, con forma de triángulo equilátero, hecho de aleación de aluminio y con dimensiones de 230.8mm X 202.6mm X 98.5mm.
- Tres ruedas del tipo omnidireccional, las ruedas tienen una dimensión de 48 mm.
- Tres motorreductores DC con encoder modelo Nexus.
- Dos puentes H modelo H-L298.
- Una tarjeta Node-MCU ESP8266.
- Una batería LiPo de 3 celdas a 11.1V.
- Un interruptor.

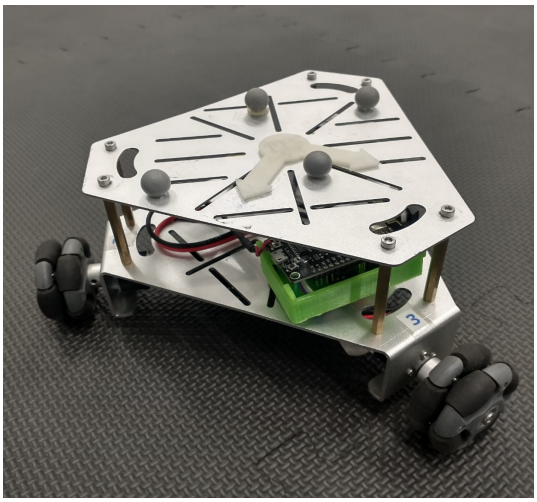

Figura 4: Robot omnidireccional implementado.

El peso neto del robot experimental es de 2kg, (revisar más detalles en Apéndice A).

Se muestran los 3 motores modelo Nexus, 2 puentes H-L298, una batería tipo LiPo de 11.1V y una tarjeta de desarrollo Node-MCU ESP8266 (con un código único cargado desde Arduino IDE).

5.0.2. *Descripción del sistema de captura de movimiento e interfase*

Se ha desarrollado la arquitectura de una plataforma experimental para la puesta en marcha de algoritmos de control en robots omnidireccionales, conformada principalmente por una zona de despliegue para la detección de cuerpos rígidos por medio de un sistema de captura de movimiento, en este caso se trata del sistema OptiTrack, el cuál opera a partir del software Motive para tener acceso a las coordenadas y orientación del robot móvil de estudio con una alta precisión (ver Figura 5).

Figura 5: Zona de despliegue del laboratorio LACSCIF con 6 cámaras.

A través de Motive, se definen cuerpos rígidos utilizando marcadores reflejantes sobre los robots omnidireccionales, colocados de forma asimétrica para establecer una orientación y posición única. El software transmite las lecturas de posición y orientación por medio de un puerto de comunicación Ethernet mediante VRPN, una interfaz basada en red independiente del dispositivo para acceder a periféricos de realidad virtual en aplicaciones de esta índole. ROS permite generar un nodo cliente que se conecta a este servidor VRPN y expone la información a través de tópicos propios de ROS (ver Figura 6).

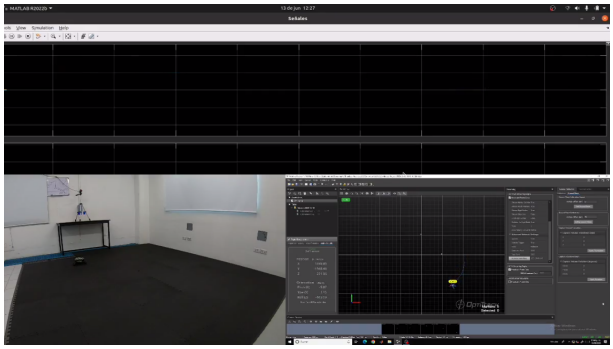

Figura 6: Detección de robot omnidireccional a través de Motive y transmisión de datos mediante VRPN, es posible visualizar el trayecto del robot.

Por otra parte, se ha desarrollado una estación de control sobre la plataforma Simulink/MATLAB en un equipo con sistema operativo Ubuntu que hace uso del middleware ROS, en este sistema es posible desarrollar y configurar algoritmos de control dedicados al movimiento de robots omnidireccionales, así como establecer posiciones y trayectorias deseadas. La estación posee un nodo suscriptor para la recepción de datos de Optitrack a través de VRPN, así como un nodo publicador que realiza el envío de las velocidades lineales y angulares por medio de un puerto TCP al robot omnidireccional, haciendo uso del paquete *rosserial* propio del sistema operativo ROS. El robot opera mediante el microcontrolador ESP8266, necesario para la recepción de las velocidades previamente mencionadas por medio del protocolo TCP. El microcontrolador realiza la traducción de las velocidades obtenidas a velocidades angulares para cada motor del robot. La comunicación de la plataforma es llevada a cabo por la red local “dlink”, siendo el intermediario que permite el intercambio de datos entre cada uno de los bloques de la plataforma. Cuando el nodo de Motive y el nodo del robot omnidireccional se encuentran activos, comienzan con el envío y recepción de datos a través de la red ”dlink”, respectivamente.

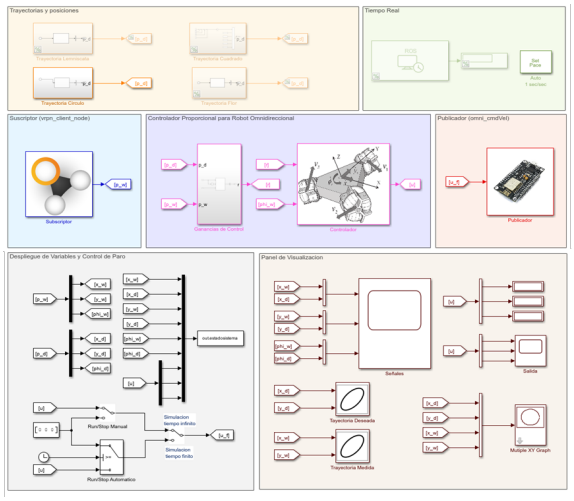

Figura 7: Diagrama a bloques de algoritmo de control desarrollado.

La estación intermedia generada en Simulink (ver Figura 7) posee un nodo suscriptor cliente al servicio de VRPN, recibiendo un mensaje de tipo *Pose*, perteneciente al paquete *geometry_msgs* de ROS, dicho mensaje contiene las coordenadas de posición x, y, z del robot y la orientación en formato de cuaternión, esta última se traduce a ángulos de Euler para poder ser interpretada en la ley de control. La estación también posee un nodo publicador que se comunica a través de *rosserial*, tiene la función de publicar las velocidades calculadas por la ley de control mediante un mensaje de tipo *Twist*, perteneciente al paquete *geometry_msgs*, este mensaje se envía a través de un tópico de ROS definido específicamente para el robot, denominado */omni1_cmdVel* (ver Figura 8).

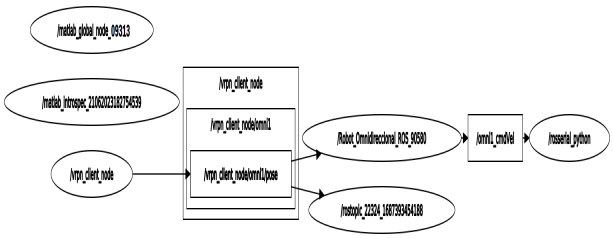

Figura 8: Ejecución de diagrama Simulink realizando la conexión a través de ROS.

5.0.3. *Resultados experimentales*

Es esta subsección analizamos el control ADRC en el prototipo diseñado. Se realizan pruebas en tiempo real para validar la plataforma experimental, el controlador ADRC y el sistema embebidos diseñado (ver Figura 9). El objetivo del controlador se basa principalmente en seguir la trayectoria deseada y rechazar las perturbaciones endógenas y exógenas que se generan en el entorno. Al igual que en la simulación, se ha elegido la trayectoria lemniscata, solo hemos variado el ángulo $\phi_w(t)$ como se muestra en (9).

$$\Sigma_{X_d} := \begin{cases} x_1^d = 3(0{,}216\sin(0{,}314t) - 0{,}589\sin(0{,}157t)) \\ x_2^d = 3(0{,}336\sin(0{,}314t) + 0{,}378\sin(0{,}157t)) \\ x_3^d = 3\sin(0{,}157t); \end{cases} . \quad (9)$$

Las figuras 10 y 11 muestra la evolución de la trayectoria del robot omnidireccional. Como se muestra en 10, el ángulo de rotación presenta algunas interferencias debido a los giros propios de la trayectoria. Estos disturbios son generados por la variación en el angulo $\phi_d(t)$, Sin embargo, se tiene un mejor rendimiento en los demás estados del robot. Otro efecto que se puede observar, son las perturbaciones debido a la fricción de las llantas con el suelo, al momento de que se realizan las curvas de lemniscata, provocando que se reduzca la velocidad y muestre una pequeña señal de ruido.

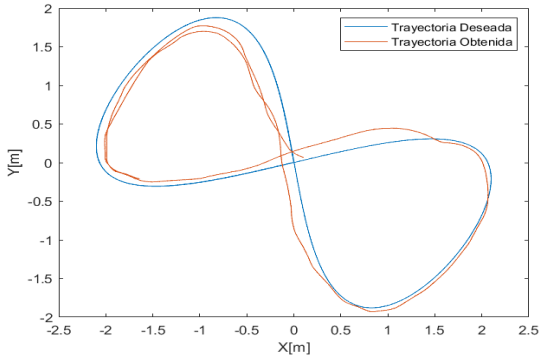

Figura 10: Resultados experimentales: Evolución de la posición del robot omnidireccional.

Se observa que la trayectoria real es parecida a la trayectoria deseada, presenta diferencia en las dimensiones de las curvas, pero el robot busca estabilizar su posición, el control actual es óptimo para alcanzar la posición deseada.

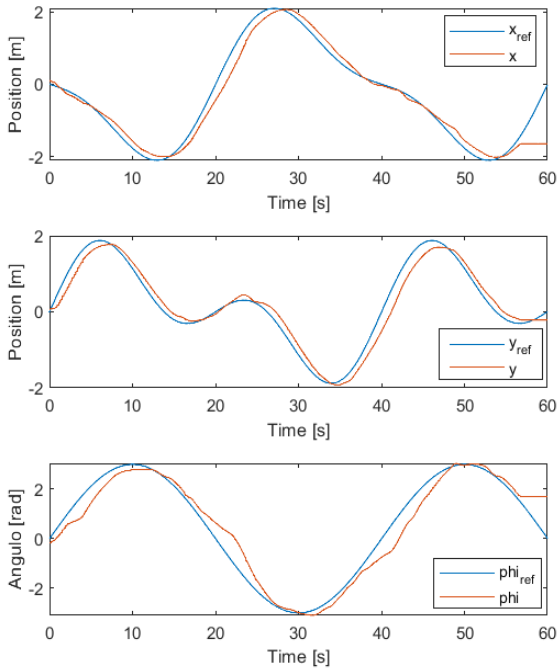

Figura 11: Resultados experimentales: Posición del robot omnidireccional en el sistema inercial w.

El algoritmo de control ADRC es liviano y robusto. Es capaz de estabilizar al robot omnidirecional tipo (3,0) a pesar de los disturbios externos generados por el entorno.

6. Conclusiones

En este artículo se presenta una plataforma experimental de arquitectura abierta basada en ROS para el desarrollo de algoritmos de control para robots heterogéneos. La arquitectura ROS es abierta y permite la integración de diferentes módulos. Esta arquitectura puede ser un punto de partida para diseñar nuevas aplicaciones de robótica. Por otra parte, el control ADRC muestra robustez frente a perturbaciones externas acotadas, su implementación es simple y ligera.

Referencias

Bihlmaier, A. y Wörn, H. (2016). *Hands-on Learning of ROS Using Common Hardware*, pp. 29–50. Springer International Publishing, Cham.

Campion, G., d'Andrea Novel, B., y Bastin, G. (1996). Structural properties and clasification on kinematic and dynamic models of wheeled mobile robots. *IEEE Transactions on Robotics and Automation*.

Cousins, S., Gerkey, B., y Conley, K. (2010). Sharing software with ros. *Robotics and Automation Magazine, IEEE*.

Curiel-Olivares, G., Linares-Flores, J., Guerrero-Castellanos, J., y Hernández-Méndez, A. (2021). Self-balancing based on active disturbance rejection controller for the two-in-wheeled electric vehicle, experimental results. *Mechatronics*, 76:102552.

Galli, M., Barber, R., Garrido, S., y Moreno, L. (2017). Path planning using matlab-ros integration applied to mobile robots.

Gao, X., Li, J., Fan, L., Zhou, Q., Yin, K., Wang, J., Song, C., Huang, L., y Wang, Z. (2018). Review of wheeled mobile robots' navigation problems and application prospects in agriculture. *IEEE Access*, 6:49248–49268.

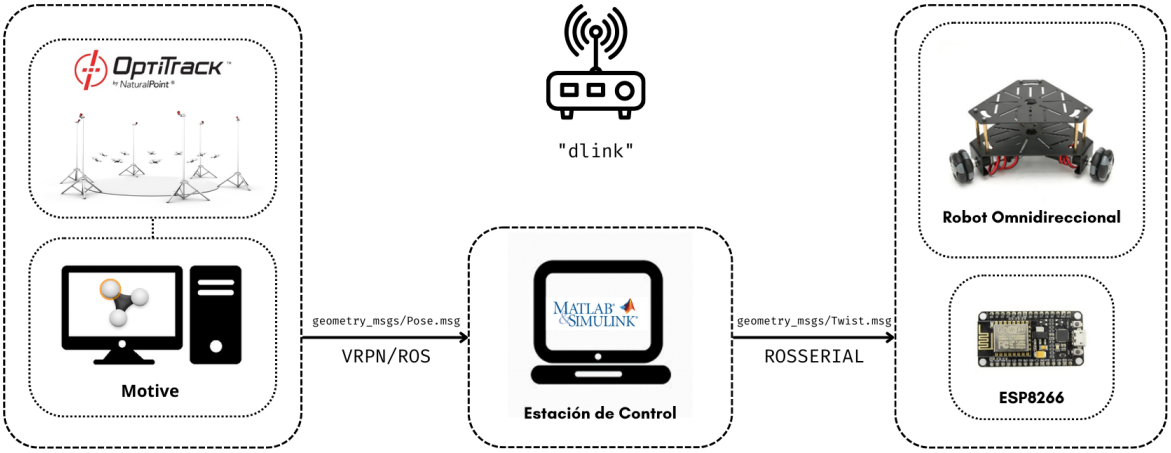

Figura 9: Diagrama general de la plataforma experimental desarrollada.

Gentilini, L., Rossi, S., Mengoli, D., Eusebi, A., y Marconi, L. (2021). Trajectory planning ros service for an autonomous agricultural robot. En *2021 IEEE International Workshop on Metrology for Agriculture and Forestry (MetroAgriFor)*, pp. 384–389.

Guerrero-Castellanos, J., Rifaï, H., Arnez-Paniagua, V., Linares-Flores, J., Saynes-Torres, L., y Mohammed, S. (2018). Robust active disturbance rejection control via control lyapunov functions: Application to actuated-ankle–foot-orthosis. *Control Engineering Practice*, 80:49 – 60.

Guerrero-Castellanos, J., Villarreal Cervantes, M., Sánchez Santana, J., y Ramírez Martínez, S. (2014). Seguimiento de trayectorias de un robot móvil (3,0) mediante control acotado. *Revista Iberoamericana de Automática e Informática industrial*, 11(4):426–434.

Haghighat, M. R. y Sadeghnejad, S. (2022). Simulation of a two-wheeled self-balancing wheelchair using ros-gazebo. En *2022 10th RSI International Conference on Robotics and Mechatronics (ICRoM)*, pp. 95–101.

Hu, K., Gu, C., y Chen, J. (2022). Ltrack: A lora-based indoor tracking system for mobile robots. *IEEE Transactions on Vehicular Technology*, 71(4):4264–4276.

Lee, D., Kang, G., Kim, B., y Shim, D. H. (2021). Assistive delivery robot application for real-world postal services. *IEEE Access*, 9:141981–141998.

Martin, J., Ansuategi, A., Maurtua, I., Gutierrez, A., Obregón, D., Casquero, O., y Marcos, M. (2021). A generic ros-based control architecture for pest inspection and treatment in greenhouses using a mobile manipulator. *IEEE Access*, 9:94981–94995.

Mišeikis, J., Caroni, P., Duchamp, P., Gasser, A., Marko, R., Mišeikienė, N., Zwilling, F., de Castelbajac, C., Eicher, L., Früh, M., y Früh, H. (2020). Lio-a personal robot assistant for human-robot interaction and care applications. *IEEE Robotics and Automation Letters*, 5(4):5339–5346.

Niroui, F., Zhang, K., Kashino, Z., y Nejat, G. (2019). Deep reinforcement learning robot for search and rescue applications: Exploration in unknown cluttered environments. *IEEE Robotics and Automation Letters*, 4(2):610–617.

Pak, J., Kim, J., Park, Y., y Son, H. I. (2022). Field evaluation of path-planning algorithms for autonomous mobile robot in smart farms. *IEEE Access*, 10:60253–60266.

Parada-Salado, J. G., Ortega-Garcia, L. E., Ayala-Ramirez, L. F., Perez-Pinal, F. J., Herrera-Ramirez, C. A., y Padilla-Medina, J. A. (2018). A low-cost land wheeled autonomous mini-robot for in-door surveillance. *IEEE Latin America Transactions*, 16(5):1298–1305.

Quigey, M., Gerkey, B., Conley, K., Faust, J., Foote, T., Leibs, J., Berger, E., Wheeler, R., y Ng, A. (2009). Ros: an open-source robot operating system. En *ICRA Workshop on Open Source Software*.

Wang, D., Leung, H., Kurian, A. P., Kim, H.-J., y Yoon, H. (2010). A deconvolutive neural network for speech classification with applications to home service robot. *IEEE Transactions on Instrumentation and Measurement*, 59(12):3237–3243.

Apéndice A. Dimensiones y estructura del Robot Omnidireccional

El chasis se conforma de dos capas para soporte, la etapa de potencia es colocada en la parte inferior del robot para la conexión de los motores a los puentes H. La etapa de control se ubica en la parte superior de la placa, se realiza la interconexión a la tarjeta NodeMCU/ESP8266 y la bateria de alimentación LiPo, además del interruptor de encendido.

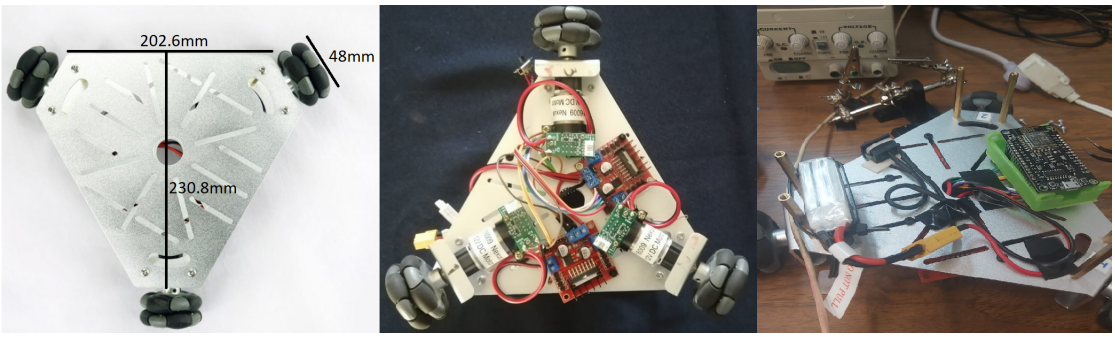

Figura A.12: Dimensión de chasis del robot y conexión de los elementos.