Sistema control modelo

1st Martinez, Brayan Steven
Ingeniería Mecatrónica
Universidad EIA
Medellín, Colombia
brayan.martinez@eia.edu.co

2nd Gongora, Juan Pablo
Ingeniería Mecatrónica
Universidad EIA
Medellín, Colombia
juan.gongora@eia.edu.co

3rd Isaza, Luis Miguel
Ingeniería Mecatrónica
Universidad EIA
Medellín, Colombia
luis.isaza@eia.edu.co

4th Jiménez, Sebastian David
Ingeniería Mecatrónica
Universidad EIA
Medellín, Colombia
sebastian.jimenez10@eia.edu.co

5th aaa, aaaa aaaa
Ingeniería Mecatrónica
Universidad EIA
Medellín, Colombia
email

6th aaa, aaaa aaaa
Ingeniería Mecatrónica
Universidad EIA
Medellín, Colombia
email

***Abstract*—Ut id augue ut ex convallis blandit quis ac libero. Donec ante nunc, dictum non erat id, pharetra tempus nunc. Cras eget dignissim sapien. Vestibulum convallis dolor non odio vestibulum sagittis. In hac habitasse platea dictumst. Praesent varius arcu quis quam hendrerit, a euismod elit gravida. Ut dapibus vulputate elementum. Aliquam posuere et nibh ut molestie.**

***Index Terms*—LaTeX, robotica, otra**

I. Introducción

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Aenean et tortor ornare, mattis augue vitae, vestibulum nulla. Ut lacus orci, viverra nec lectus consequat, laoreet luctus neque. Donec scelerisque ante ac fringilla malesuada. Cras porttitor lectus erat, at convallis ex cursus a. Ut vehicula eget magna sit amet mattis. Cras at fermentum mauris, semper luctus nibh. Nunc consectetur scelerisque tellus at volutpat. Nullam ut vulputate urna. Donec sem elit, blandit quis cursus ut, ullamcorper nec sapien. Sed mauris elit, mattis sed luctus ut, tristique ac libero. Donec a sollicitudin urna, at congue nulla. Nulla a blandit dolor, ut vehicula sapien. Donec at augue diam. Quisque turpis eros, luctus id aliquam efficitur, pulvinar sit amet magna. Fusce at dui tortor.

II. Materiales y método

A. Materiales

Matlab para realizar matemática computacional, programación de algoritmos y gráficos a partir de los resultados.

CoppeliaSim para llevar a cabo simulaciones en un entorno virtual teniendo a disposición robots predeterminador, sensores y un API para conectarse con otros lenguajes de programación.

Odroid-XU4 como tarjeta de desarrollo para ser implementada de forma tentativa en el robot. Características de buen rendimiento y tamaño la convierten en una opción para integrar todos los sistemas.

ROS (Robot Operating System) como framework para complementar varios sistemas en uno solo y permitir una fluida comunicación entre los mismo. Se instala sobre una distribución de Linux.

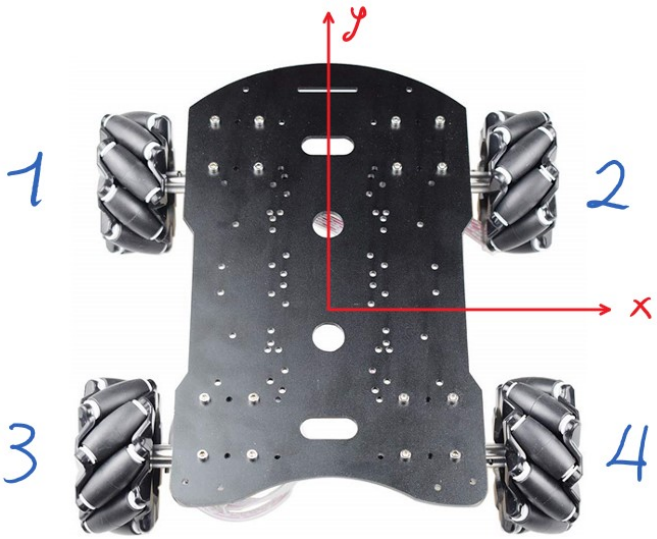

Fig. 1. Plataforma mecánica con motores y ruedas

Ubuntu como sistema operativo para implementar en el robot aprovechando las ventajas de ROS y la compatibiliadd con sistemas de desarrollo como la ODROID-XU4 y sensores.

Modelo cinemático [1] para predecir el comportamiento de la plataforma que posee cuatro ruedas suecas (mecanum)

Plataforma móvil con cuatro motores y ruedas suecas (mecanum), la configuración para este trabajo es como sigue en la Figura 1.

B. Método

1) Simulación Matlab: Se analiza el modelo cinemático ya conocido:

$$\begin{bmatrix}\omega_1\\ \omega_2\\ \omega_3\\ \omega_4\end{bmatrix} = \frac{1}{r}\begin{bmatrix}1 & 1 & -(L+l)\\ -1 & 1 & (L+l)\\ -1 & 1 & -(L+l)\\ 1 & 1 & (L+l)\end{bmatrix}\begin{bmatrix}V_s\\ V_y\\ \omega\end{bmatrix} \tag{1}$$

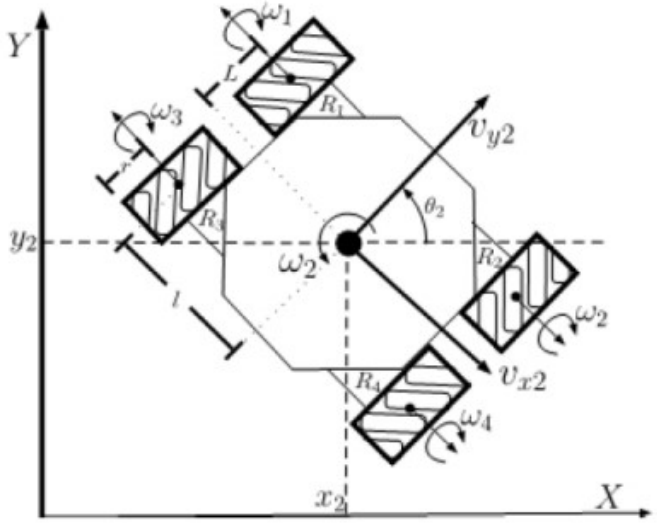

Fig. 2. Robot omnidireccional a modelar

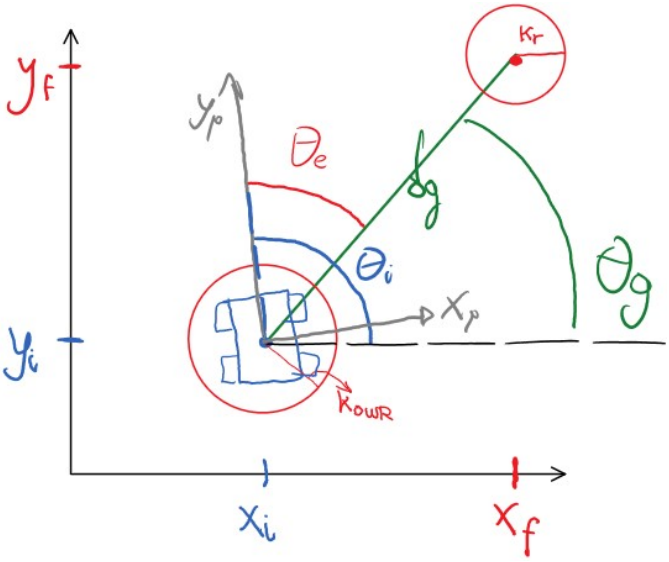

Fig. 3. Robot en el plano con campos potenciales

En base a la Figura 2, Los parametros L, l y r, pertenecen al robot, para esta simulación, al KUKA YouBot [2], donde: r = 100 mm, L = 235.5 mm, l = 150 mm.

A este modelo matemático se le ingresan las velocidades en X, Y y rotación angular del robot, para obtener la velodiadad angular de cada rueda. Se programa la función que representa este modelo cinámatico en Matlab y además, otra función para convertir las posiciones deseadas en velocidad lineal usando las ecuaciones 3, 2 y 4 teniendo en cuenta una metodología de control por campos potenciales [3] visibles en la Figura 3

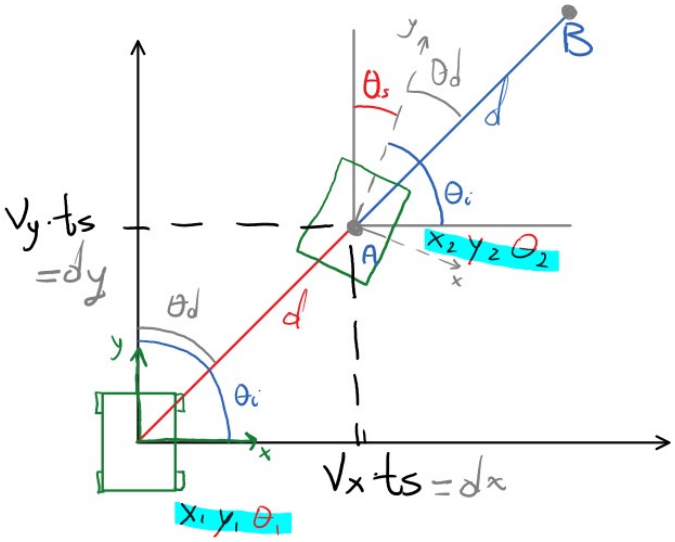

Fig. 4. Desplazamiento del robot en el plano cartesiano

$$V_x = \begin{cases} Vx_{max} \sin\theta_e & si \quad d_g > K_r + K_{owr} \\ \frac{d_g}{K_r + K_{owr}} \sin\theta_e & si \quad d_g \leq K_r + K_{owr} \\ 0 & si \quad d_g = 0 \end{cases} \tag{2}$$

$$V_y = \begin{cases} Vx_{max} \cos\theta_e & si \quad d_g > K_r + K_{owr} \\ \frac{d_g}{K_r + K_{owr}} \cos\theta_e & si \quad d_g \leq K_r + K_{owr} \\ 0 & si \quad d_g = 0 \end{cases} \tag{3}$$

$$\omega = \omega_{max} + \sin\theta_e \tag{4}$$

Con las ecuaciones 3, 2 y 4 se pueden obtener las velocidades angulares de cada rueda gracias al modelo cinemático.

Observando la Figura 3 se nota que es necesario en todo momento conocer la posición actual del robot y así calcular los vectores y ángulos; es fácil de obtener cuando hay otro sistema externo que proporciona esas coordenadas, pero, para una simulación en Matlab, es necesario encontrar los nuevos valores cada que el objeto se desplazada con las ecuaciones 9, 10 y 11, cuyas variables se pueden observar en la Figura 4.

$$d = \sqrt{(d_x)^2 + (d_y)^2} \tag{5}$$

$$d_x = V_x T_s \tag{6}$$

$$d_y = V_y T_s \tag{7}$$

$$\theta_d = \tan^{-1} \frac{d_x}{d_y} \tag{8}$$

$$x_2 = x_1 + \left(d \sin\left(\theta_d + \frac{\pi}{2} - \theta_i\right)\right) \tag{9}$$

$$y_2 = y_1 + \left(d \cos\left(\theta_d + \frac{\pi}{2} - \theta_i\right)\right) \tag{10}$$

$$\theta_{i_2} = \theta_{i_1} + \omega T_s \tag{11}$$

Se recomienda ir al código para analizar la secuancia lógica.

2) Simulación con CoppeliaSim: Es necesario usar la API de CoppeliaSim en Matlab para llevar a cabo la simulación aprovechano que ya se tiene el modelo cinemático programado y funcionando en Matlab.

Buscar los archivos del API en la ruta de CoppeliaSim : "CoppeliaRobotics CoppeliaSimEdu programming remoteApiBindings matlab matlab", para copiar únicamente los siguientes archivos en la carpeta con el main de Matlab para controlar el robot:

- remApi
- remoteApiProto
- simpleText

En la misma carpeta, copiar otro archivo llamado remoteApi.dll que se encuentra en: "CoppeliaRobotics CoppeliaSimEdu programming remoteApiBindings lib lib Windows"

El archivo 'simpleText' sirve como guía para usar el API con Matlab.

III. Resultados

A. Simulación Matlab

Se obtiene una primera simulación numérica usando Matlab. El robot comienza en (0,0) a 90° de la horizontal, se desea que llegue a (1.5, 3.2). El resultado se puede ver en la Figura 5. La orientación del robot se va alineando con el vector definido por el punto inicial y final.

Aprovechando el modelo cinemático, se puede graficar la velocidad angular de los motores, como se observa en la Figura 6, la velocidad se reduce hasta 0, llegando así a la posición final.

B. Simulación CoppeliaSim

Se obtiene también, una simulación virtual en base al modelo cinemático y un robot con el cual se prueba el modelo. En CoppeliaSim se dispone del KUKA YouBot que viene por defecto, una plataforma con ruedas omnidireccionales.

Se programó un desplazamiento como en la Figura 9 donde al final se incluye una rotación y se puede visualizar en la Figura 10.

Se conocen alternativas más cercanas al entorno real de implementación como Gazebo, que no se logró probar para esta iteración del semillero; se sabe que se puede complementar con ROS (Robot Operating System) y eso significa que se puede aprovechar para elementos o dispositivos electrónicos físicos.

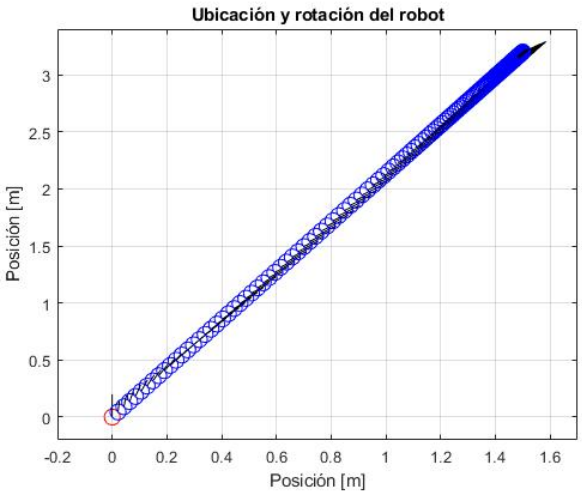

Fig. 5. Registro de movimiento del robot

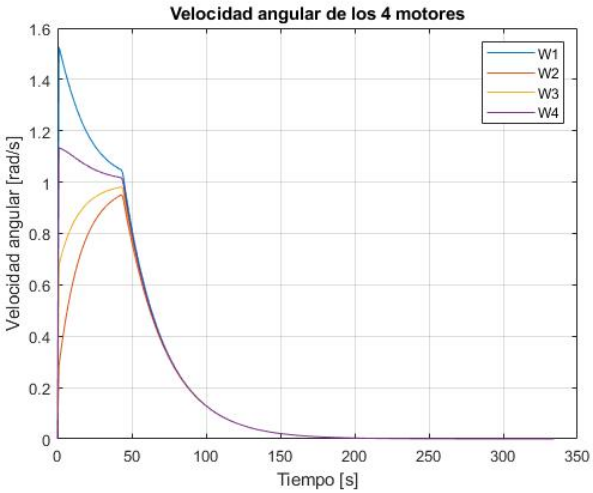

Fig. 6. Velocidad angular de los motores en el tiempo

Fig. 7. KUKA YouBot en la realidad

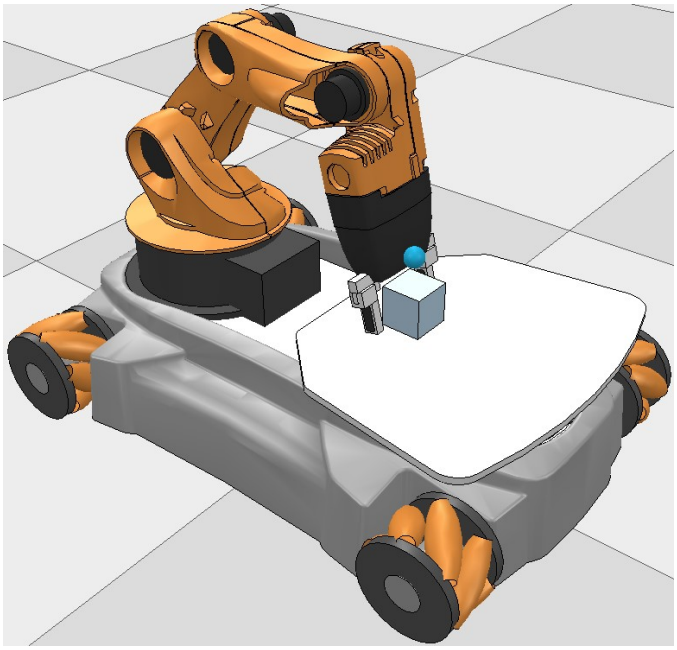

Fig. 8. KUKA YouBot en CoppeliaSm

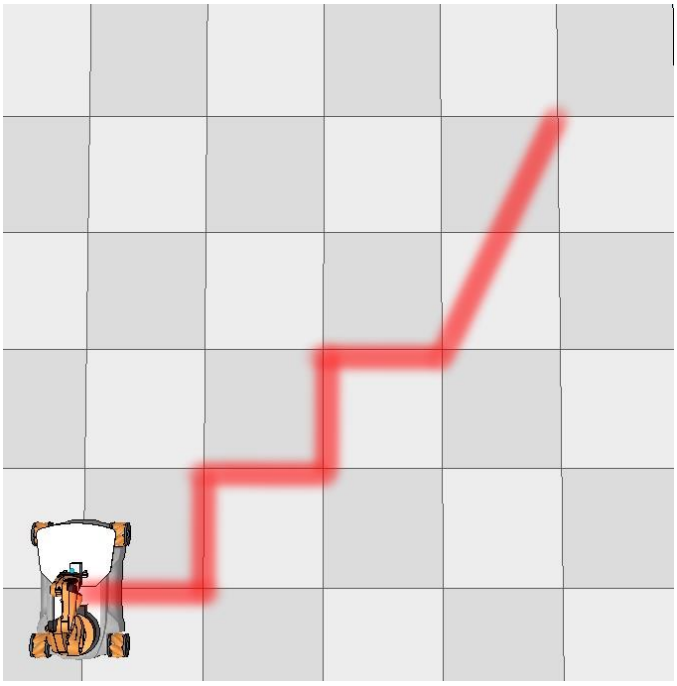

Fig. 9. Ruta programada para el KUKA YouBot y posición inicial

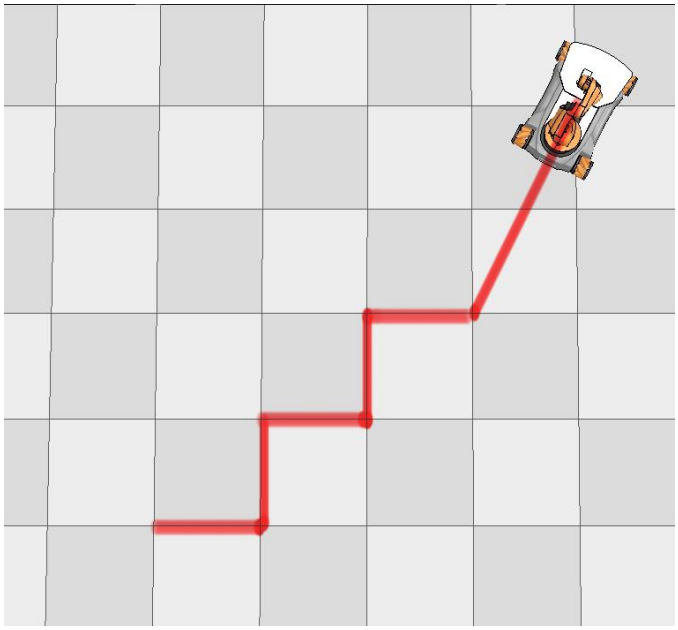

Fig. 10. Posición final KUKA YouBot

C. Revisión Odroid

Se instaló satisfactoriamente la distribución Ubuntu de Linux en sus versiones 16 y 18 sobre la tarjeta de desarrollo Odroid-XU4 presentando algunos problemas iniciales de apagado indeseado cuando se le cargaba al sistema operativo con más procesos, solucionable con una fuente de mejor calidad o también con un arreglo de diodo, capacitor y resistencia para evitar los cortes súbitos pero momentáneos de energía.

D. Aprendizajes en ROS

En relación con el punto anterior, se logró instalar ROS (Robot Operating System) sobre Ubuntu 16 y correr algunos de sus ejemplos predeterminados como la TurtleBot, además, entender brevemente aunque sin profundizar, en la estructura de este sistema operativo.

E. Cámara Orbbec Astra

Se realizaron pruebas con la cámara de profunidad Orbbec Astra concluyendo que es un excelente complemento para determinar el entorno tridimensional que rodea al robot de servicio, este dispositivo permite conocer la profuncidad de cada punto de la imagen, el programa se corre en Linux y la cámara es USB.

F. LiDAR

Se sabe de paquetes de ROS que son una gran alternativa para realizar mapeo con ROS como son:

- Google Cartographer
- Hector Slam

IV. Discusión o análisis de resultados

La simulacuón en CoppeliaSim no se pudo realizar más compleja porque los ángulos de rotación entregados por el API a Matlab no son cartesianos.

La velocidad angular de cada motor corresponde efectivamente con el movimiento que debería hacer el robot en la realidad.

El movimiento simulado en CoppeliaSim no tuvo ningun tipo de problema, además el fin de cada trayectoria se veía suavizado gracias al enfoque de campos potenciales [3].

El punto de inicio y el punto de final de una trayectoria definen un vector al cual el robot se intenta alinear si se incluye una medida de error entre la horientación del robot y el mencionado vector.

En la simulación no es posible determinar el ángulo final de robor, no se implementó.

V. Conclusión

- La simulación con Matlab y CoppeliaSim en conjunto es muy útil, dado que el segundo tiene muchos robots, sensores y objetos para crear entornos de simulación, además se aprovecha la potencia de Matlab para la computación sin programar el childScript en CoppeliaSim que resulta ser poco práctico.
- Se valido de manera aceptable por medio de simulaciones que el modelo cinamático es funcional.
- Se pueden combinar la cámara de profunidad con el LiDAR para obtener una representación más fiel del entorno que rodea el robot.
- Matlab es un buen comienzo para planear el algoritmo de control y luego migrarlo a alternativas más útiles con la actual aplicación Robótica, usar ROS desde etapas temprana si se tiene todo el hardware puede resultar provechoso.
- Las simulaciones son muy aceptables pero en la realidad también hay consideraciones mecánicas y eléctricas, los problemas realmente están en el hardware.
- Es muy importante elegir un algoritmo o ley de control adecuada para los motores de la forma correcta, el fallo en la actuación de estos elementos compromete todo el robot.

References

[1] "Gazebo." [Online]. Available: http://www.gazebosim.org/

[2] "KUKA youBot." [Online]. Available: https://www.kuka.com/

[3] V. Gonzalez-Villela, A. Sánchez-Balpuesta, and S.-A. Adan, "Cinemática de una plataforma móvil omnidireccional con llantas Mecanum en configuración AB," Ph.D. dissertation, Universidad Nacional Autónoma de México, Sep. 2015.