

# RETO SEMANAL 2. Manchester Robotics

Abraham Ortiz Castro

Dpto. de Ingenierías Tec de Monterrey  
Tecnológico de Monterrey  
Puebla, México  
A01736196@tec.mx

Alan Iván Flores Juárez

Dpto. de Ingenierías Tec de Monterrey  
Tecnológico de Monterrey  
Puebla, México  
a01736001@tec.mx

Jesús Alejandro Gómez Bautista

Dpto. de Ingenierías Tec de Monterrey  
Tecnológico de Monterrey  
Puebla, México  
A01736171@tec.mx

Ulises Hernández Hernández

Dpto. de Ingenierías Tec de Monterrey  
Tecnológico de Monterrey  
Puebla, México  
A01735823@tec.mx

## I. RESUMEN

En este texto concentra toda la información referente a un tema fundamental para este nuevo reto: la implementación de trayectorias para el PuzzleBot mediante ROS 2, para ello, además de implementar la ya tradicional estructuras de publicadores y receptores mediante nodos en ROS, debemos definir dos trayectorias, una cerrada (un cuadrado) y una abierta (zig zag) que nuestro robot deberá seguir mediante el uso de odometría en esta primera actividad. Para lograr lo anterior es necesario comprender el movimiento de un robot diferencial, el tipo de mensaje Twist (fundamental para la publicación en de tópicos encargados de la velocidad lineal y angular del robot) así como el control de lazo abierto y la odometría para el diseño del control. El conjunto de todos estos conocimientos nuevos y previos nos permitirán construir un sistema robusto para el diseño del movimiento del robot.

## II. OBJETIVOS.

- Desarrollar las capacidades en el manejo del framework ROS.
- Incrementar el grado de interacción a partir de las funciones que ROS nos permite desarrollar.
- Introducir conceptos básicos de control de lazo abierto para la robótica móvil.
- Introducir la navegación multipunto en el PuzzleBot.

## III. INTRODUCCIÓN.

### A. Mensajes Twist

El tipo de mensaje Twist en ROS son uno de los elementos más importantes para la comunicación entre nodos de un sistema robótico, primordialmente empleados para enviar comandos de velocidad a un robot.

El mensaje tipo Twist consta de dos componentes de velocidad en espacio libre, las cuales se dividen en un vector de tres dimensiones definidos de la siguiente manera:

- **Twist.linear:** Representando la velocidad lineal con sus componentes en x, y y z.
- **Twist.angular:** Representando la velocidad angular alrededor de x, y y z.

En un robot terrestre, es altamente probable que la velocidad lineal pueda cambiar en los componentes de x y de y, sin embargo, la velocidad angular se concentrará principalmente en el eje z.

1) *¿Qué significan X, Y y Z en los mensajes Twist?:* En el espacio existen 3 ejes (x, y, z) los cuales son mutuamente perpendiculares entre sí y su punto de intersección es el origen  $x=0$ ,  $y=0$  y  $z=0$ . Esto funge como un marco de referencia que nos permite definir varios puntos. Tanto la velocidad angular y la linear emplean el mismo marco de referencia.

[1] 2) *Creación de mensajes tipo Twist:* Para crear un mensaje tipo twist en Python, se sigue la siguiente estructura:

```
import rospy
from geometry_msgs.msg import Twist

# Crear una instancia de rospy.Publisher()
pub = rospy.Publisher('turtle1/cmd_vel', Twist, queue_size=1)

# Crear una instancia de un mensaje Twist
twist = Twist()

# Asignar valores a las componentes lineales y angulares
twist.linear.x = 1.0
twist.angular.z = 1.0

# Publicar el mensaje Twist
pub.publish(twist)
```

[3]

### B. Cinemática de un Robot Diferencial

Se entiende a la cinemática de un Robot Diferencial como el estudio del movimiento de este tipo de robots sin contemplar las fuerzas que originan este movimiento. Un robot diferencial es un tipo de vehículo terrestre que emplea dos ruedas independientes o lo que es igual a dos ruedas con su propio motor, provocando que su locomoción o movimiento se base en la diferencia de velocidades de sus dos ruedas sobre un único eje.

El propósito de la cinemática diferencial no es otro que el de encontrar las relaciones entre las velocidades de nuestro punto de control  $h(x, y)$  y las velocidades que intervienen en el movimiento del robot, omitiendo las fuerzas que ejercen sobre el robot (como la inercia y el rozamiento). [4]

*1) Principio de funcionamiento de un robot diferencial::*

- Si dos ruedas giran en la misma dirección y velocidad, el robot se moverá en línea recta hacia adelante.
- Si cambia el sentido de giro y mantiene la velocidad, el vehículo se desplazará hacia atrás.
- Si dos ruedas giran en la misma dirección, pero con diferentes velocidades, el robot se alejará del motor más rápido, es decir, si la rueda derecha gira más que la otra, el vehículo se moverá a la izquierda.
- Si ambas ruedas giran a la misma velocidad, pero en direcciones opuestas, el vehículo girará en su propio eje en sentido horario o antihorario.

*2) Modelo cinemático::* Para determinar la localización del móvil en el plano cartesiano, se emplea el modelo cinemático diferencial directo, el cual relaciona las velocidades del punto de control con las velocidades de los actuadores. Para obtener el modelo, se guiará de la geometría del vehículo, cuya posición se define en torno al punto h y con respecto a la orientación del ángulo. [5]

*3) Uso de modelos cinemáticos::* En general, el uso de estos modelos cinemáticos nos permiten realizar análisis y diseños de algoritmos de control, particularmente para tareas de robótica donde existe una baja velocidad y una poca carga en la estructura. [5]

### C. Lazo abierto de control en un Robot Móvil

*1) Definición::* Un sistema de control de lazo abierto es aquel cuyas acciones de control son previamente programadas y sin realizar ajustes en función de una retroalimentación, como en lazo cerrado. Este tipo de control asume que tanto las entradas como las salidas del sistema se conocen y pueden ser predichas, es decir, el controlador envía señales de control sin conocer a ciencia cierta que el estado real del sistema sea este. [6]

*2) Usos:* Se utiliza en situaciones donde la precisión no es imprescindible o cuando la variable de salida es fácilmente predecible, podría decirse que este tipo de control es recomendable para tareas repetitivas. El controlador enviará una secuencia predefinida de movimientos sin considerar si los mismos están siendo ejecutados de manera correcta.

*3) Ventajas y Desventajas:* Una de las ventajas más importantes de un control de lazo abierto es la simplicidad del mismo, pues al necesitar un sistema de retroalimentación, es mucho más fácil de implementar. No obstante, es esta simplicidad también un arma de doble filo, pues no hay forma de corregir errores durante la ejecución del sistema de control. Por lo anterior, es necesario que el control de lazo abierto sea empleado en sistemas donde la retroalimentación no es necesaria y, por el contrario, genera un costo adicional frente a los beneficios que provee. [7]

### D. Cálculo de la distancia y ángulo recorrido por un Robot Móvil

Para el cálculo de la distancia y el ángulo se utilizan las ecuaciones 1 y 2, las cuales utilizan la velocidad angular de

cada uno de los motores ( $\omega_L$  y  $\omega_R$ ), la distancia entre las ruedas (r) y la distancia entre los ejes (l).

$$d = r(\frac{\omega_R + \omega_L}{2}) * dt \quad (1)$$

$$\theta = r(\frac{\omega_R - \omega_L}{l}) * dt \quad (2)$$

Estos datos serán utilizados posteriormente para definir si se ha alcanzado las coordenadas preestablecidas en el nodo `path_generator`.

### IV. SOLUCIÓN DEL PROBLEMA

Para dar solución a este reto, se realizó la creación de dos nodos, `path_generator` y `controller`, el primero de los cuales es el encargado de recibir la ruta del usuario, para posteriormente enviar dichas coordenadas por un tópico llamado `pose`, éstas coordenadas serán usadas posteriormente para calcular la velocidad lineal y angular requeridas para alcanzar las coordenadas deseadas, de igual manera el nodo envía los valores de velocidad lineal y angular deseada por el usuario. El nodo `controller` es el encargado de recibir las coordenadas del tópico `pose` y asignarlas a las variables 'x' y 'y' de cada posición, de igual manera recibe los valores de la posición de cada una de las coordenadas, así como el valor de theta, que son enviados a través del tópico `odometria`. Es dentro de la función `timer_callback` que se realizan las operaciones necesarias para calcular la velocidad lineal y angular requerida para cada momento o segmento de la trayectoria. Un ejemplo del código se muestra a continuación:

```
#Sección de código en las que se realizan las
comparaciones del valor deseado y el actual
if self.Posx >= self.x2 and self.bandera == 1:
    self.velL = 0.0
    self.velA = 0.1
    self.angulo = math.atan2((self.y2-self.y1),
                           (self.x2-self.x1))
    if self.Postheta >= self.angulo:
        self.velA = 0.0
        self.velL = 0.1
        self.bandera = 2
```

El nodo `controller` es de igual forma responsible de enviar las velocidades calculadas a través del tópico `cmd_vel` como tipo de mensaje `twist` modificando únicamente el elemento 'linear.x' y 'angular.z' ya que únicamente buscamos que el robot avance y rote sobre su eje. Esto se realiza de esta manera para que el robot sea capaz de interpretar los valores enviados y mueva los motores de manera correspondiente.

Dentro de nuestra solución a su vez se utiliza el nodo creado para el reto de la semana 1, nodo `subscriber` que recibe las velocidades angulares de ambos de los motores y realiza las operaciones necesarias para obtener la posición en el eje 'x', eje 'y' y el valor de theta. Datos que posteriormente envía por el tópico `odometria` y que es leído por el nodo `controller`.

Aunado a esto se realizaron todas las configuraciones al launcher para la ejecución de los tres nodos, además de mostrar `rqt_graph` y `rqt_plot` que nos ayudan a observar las variables enviadas por los tópicos. De igual forma se realizó la configuración de los mensajes, los cuales incluyen los

mensajes personalizados Path y Vector, que representan las coordenadas deseadas así como la posición en el eje 'x', 'y' y 'theta' respectivamente.

## V. RESULTADOS

### A. Primera Trayectoria (cuadrado)

EL resultado del reto fue la obtención de las trayectorias establecidas, la primera de las cuales es un cuadrado de 1x1 como se puede observar en el video 1.

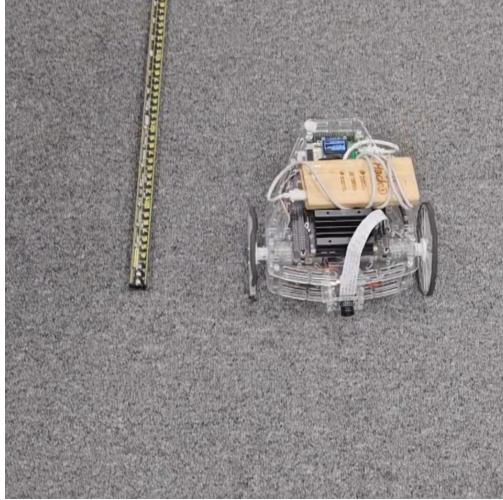


Fig. 1. Trayectoria cuadrado

En el video se puede observar como el robot avanza en las coordenadas preestablecidas, presentando un ligero error respecto a los valores internos calculados con las velocidades angulares de los motores en comparación de los valores reales, teniendo una variación de al rededor del 10% el cual se puede deber a una multitud de factores, como lo es el hecho de que se esta realizando una integración numérica para la obtención de valores, así como la resistencia de la superficie sobre la cual se desplaza, aunado a los desperfectos presentes en las ruedas. Este error se encuentra de un margen relativamente aceptable, tomando en cuenta que se esta realizando la trayectoria en manera de lazo abierto, sin tomar en cuenta ninguna retroalimentación.

### B. Segunda trayectoria

Para esta segunda trayectoria se establecieron las siguientes coordenadas [0.4,0.2; 0.2,0.6; 0.8,0.8; 1.0, 0.0] dentro de nuestro nodo Controller, trayectoria que se puede observar en el video 2. En este se puede observar como alcanza las posiciones correctas para las primeras dos coordenadas, realizando a su vez de manera exitosa la tercera rotación, es en este momento cuando observamos una ligera corrección que hace el robot, desplazándose ligeramente en el eje vertical 'y' para detenerse en la tercera coordenada, posteriormente realizando correctamente su rotación para orientarse hacia su última coordenada designada, es aquí cuando encontramos un ligero problema ya que el robot aunque en la orientación correcta, este no realiza el movimiento deseado para alcanzar



Fig. 2. Segudna trayectoria

e último punto de la trayectoria. Cumpliendo así parcialmente la trayectoria propuesta.

Dentro de los puntos de mejora para esta trayectoria encontramos que el error medido a través de nuestra variable theta, aunque pequeño con su valor real correspondiente, crece con cada trazo, por lo que debíamos establecer márgenes de error para los puntos deseados ligeramente elevados, especialmente para las últimas coordenadas de la trayectoria, lo que provocaba que el robot no realizara las rotaciones en el punto exacto deseado, de igual manera encontramos áreas de mejora en el manejo de ciertas circunstancias, por ejemplo, en el manejo de ángulos negativos o cercanos a cero, ya que en estos puntos requeríamos cambiar las comparaciones así como cambiar la dirección de rotación, aspectos que se podrían mejorar en un futuro para que el programa se encargue de estos edge cases de manera automática y no requiera de tantas correcciones por nuestra parte.

## VI. CONCLUSIONES

En retrospectiva, los retos semanales previos, aunque presentaban un grado de dificultad considerable, no se comparan con el desafío que representó el reto final. Este último implicó un esfuerzo significativo, especialmente en el diseño de la sección del controller. Durante mucho tiempo, trabajamos desde diferentes enfoques que nos acercaban prometedoramente a una solución. Sin embargo, cada uno de estos enfoques tenía limitaciones que impedían su empleabilidad como solución definitiva para esta actividad.

El control de lazo abierto que empleamos, aunque es más simple de implementar al no requerir un proceso de retroalimentación que verifique el desempeño adecuado de las trayectorias descritas por nuestro PuzzleBot, no es exacto. En particular, la descripción de las longitudes y magnitudes de desplazamiento y rotación de nuestro robot no siempre coincidía con los diagramas de trayectorias esperados. Además, dado que nuestro PuzzleBot cuenta con elementos como el

encoder, creemos que sería más conveniente emplearlos para hacer trayectorias precisas y reducir el error.

Está claro que a medida que las trayectorias se vuelven más complejas, se necesitan mayores habilidades de programación y control. No basta con conocer la estimación de una posición, sino que también es necesario estar conscientes de la posición de nuestro robot en todo momento para realizar maniobras exactas con el fin de cumplir con un movimiento determinado. Para reforzar esto, en el futuro contaremos con el uso de la cámara, que puede facilitar enormemente esta labor, en lugar de solo calcular e inferir posiciones de forma matemática mediante cálculos que no toman en cuenta el error como posible entrada de nuestro sistema.

A pesar de las dificultades presentadas durante la etapa de desarrollo de las trayectorias, logramos aproximar un modelo bastante funcional. Sin embargo, tenemos una gran oportunidad para mejorar nuestros sistemas de trayectorias predefinidas. Esperamos eliminar estas limitaciones en futuras implementaciones, especialmente aquellas que requieran una precisión milimétrica. En última instancia, el uso de un sistema computacional más robusto que permita al movimiento del robot entender su entorno mediante elementos más allá de la odometría será esencial para nuestro éxito continuo.

#### REFERENCES

- [1] What do X,Y, and Z mean in geometry msgs Twist message in ROS. (s.f.). Stack Overflow. [link](#).
- [2] Twist Message Example and /cmd vel - ROS Answers: Open Source QA Forum. (s.f.). [link](#).
- [3] Working with Twist Messages in Python - COM2009 ROS Labs. (s.f.). [link](#).
- [4] Edisonsasig. (2021, Abril 13). La cinemática en la robótica. Roboticoss. [link](#).
- [5] Edisonsasig. (2023, Julio 14). Modelo cinemático y simulación de un robot móvil diferencial. Roboticoss. [link](#).
- [6] Control Automático Educación (2024, Febrero 13). Control Automático Educación. [link](#).
- [7] Tipos de sistemas de control - Brazo robótico y sus clasificaciones. (s.f.). [link](#).
- [8] ManchesterRoboticsLtd. (s. f.). GitHub - ManchesterRoboticsLtd/TE3002B-2024: Intelligent Robotics Implementation. GitHub. [link](#).