Implementación de robótica inteligente (Gpo 501)

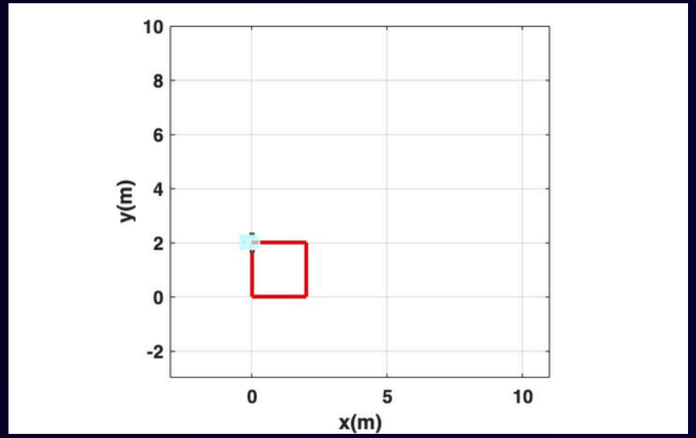
TRAYECTORIA EN LAZO ABIERTO

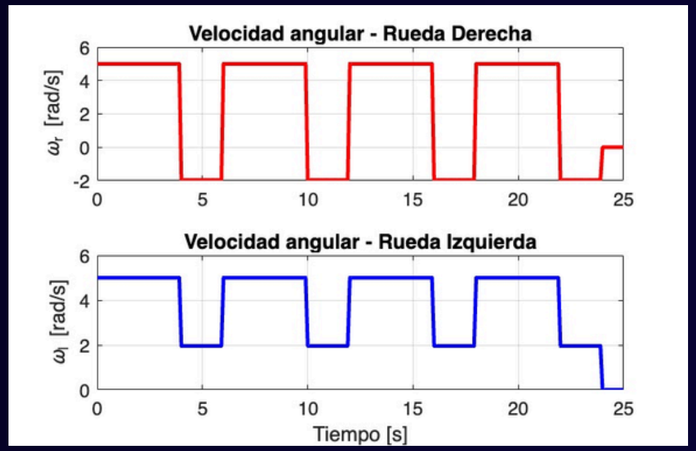
MARIAM LANDA A01736672
YESTLI SANCHEZ A01736992
EMILIANO OLGUIN A01737561
ELIAS GUERRA A01737354

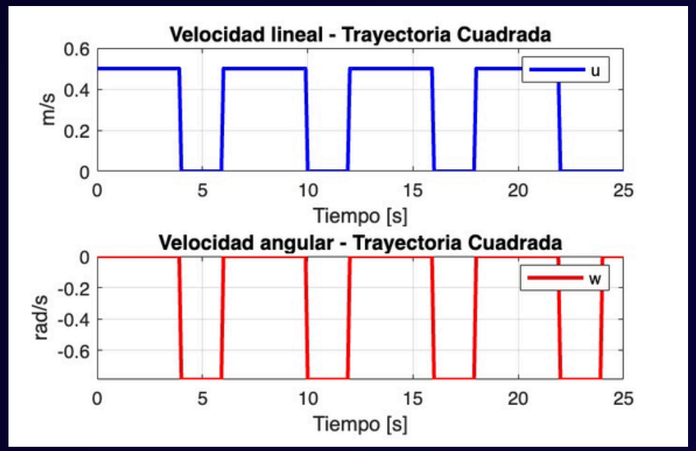
Simulación de trayectorias (Act3.1)

Trayectoria cuadrada:

- Se configura una secuencia de movimientos lineales y giros de 90° con velocidades constantes para recorrer un cuadrado. Se visualiza tanto la animación como las gráficas de velocidad lineal y angular.



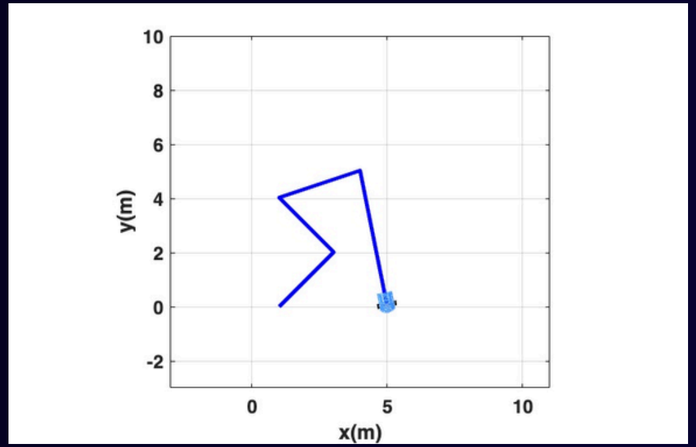


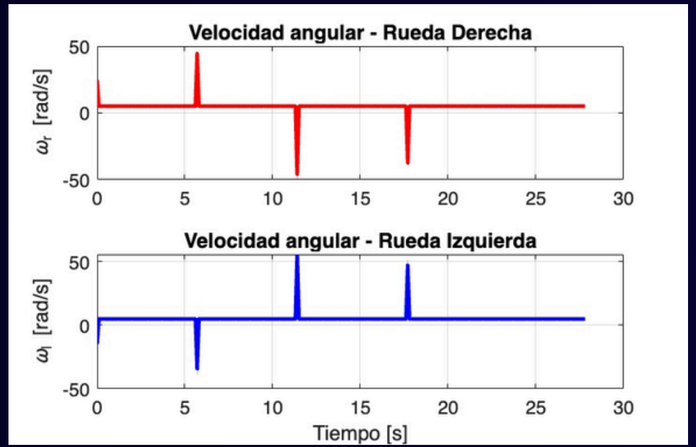


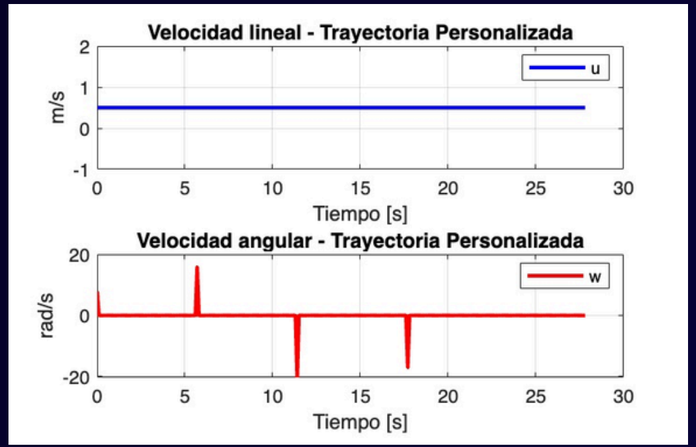
Simulación de trayectorias (Act3.1)

Trayectoria personalizada:

- Se definen puntos de paso que el robot debe seguir. El robot se orienta hacia el siguiente punto y se desplaza en línea recta, actualizando su posición en cada paso. Se calcula la orientación con atan2 y se deduce la velocidad angular.



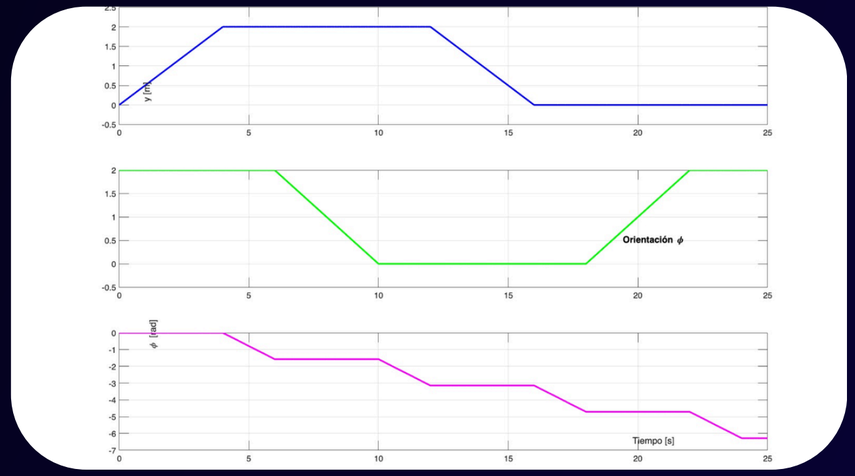


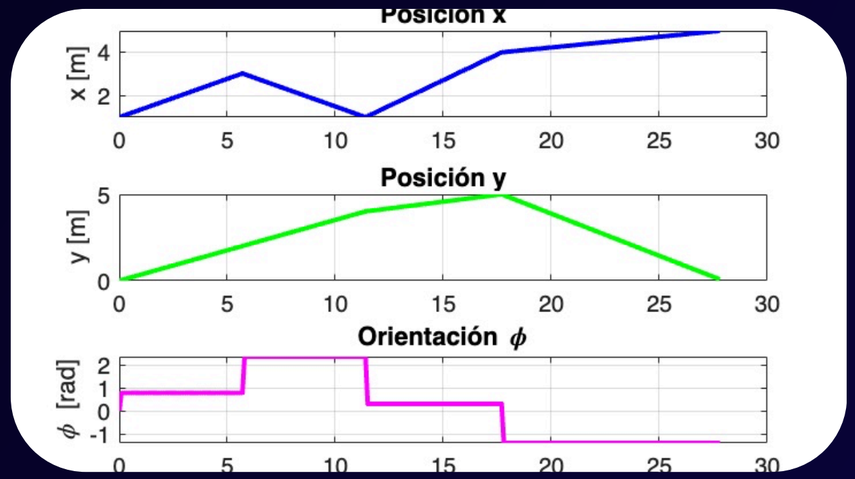


Implementación de robótica inteligente (Gpo 501)

Análisis de pose (Act3.2)

- Se trabaja con los datos de posición y orientación del robot generados en las trayectorias.
- Se visualiza y analiza cómo evoluciona la pose del robot:
  - x(t), y(t): posiciones a lo largo del tiempo.
  - ϕ(t): orientación angular en radianes.
- Este análisis es importante para evaluar la precisión del modelo cinemático y verificar que el robot mantiene la orientación correcta al cambiar de dirección.





Aplicación del tópico /cmd_vel

- En ROS, el nodo Puzzlebot Sim escucha el tópico /cmd_vel, que se usa para controlar el movimiento del robot en términos de velocidad lineal y angular.
- Las simulaciones de MATLAB emulan lo que pasaría si se publicaran esos comandos directamente en ROS.

Cálculo de Valores

En la trayectoria personalizada, se definieron manualmente puntos de referencia (p1 a p5) por los que debía pasar el robot (líneas 153–158). Para cada par de puntos consecutivos, se calcularon las diferencias dx, dy, la distancia entre ellos y el ángulo de orientación phi_actual = atan2(dy, dx) (líneas 165–169). Con la distancia y velocidad v, se calculó el tiempo de recorrido y el número de pasos necesarios para avanzar (líneas 170–171). En esos pasos, se actualizó la posición del robot avanzando con orientación fija (líneas 174–180). Luego, la velocidad angular w_custom se obtuvo como la derivada numérica del ángulo de orientación entre pasos consecutivos (líneas 191–193). Este proceso generó una trayectoria suave que conecta todos los puntos deseados.

Para la trayectoria cuadrada, se definieron valores constantes para la velocidad lineal v y la velocidad angular w_val(líneas 32–34). Usando la longitud del lado del cuadrado L, se calculó el tiempo de avance t_lado = L / v, y con la velocidad angular se obtuvo el tiempo de giro t_giro = (pi/2) / abs(w_val) (líneas 35–36). Estos tiempos se tradujeron en pasos de simulación usando el paso de muestreo ts (líneas 38–39). Luego, se rellenaron los vectores u_square y w_square para avanzar recto y girar durante los pasos necesarios (líneas 41–50). Durante la simulación, en cada paso se actualizó la orientación phi_square y la posición (x1_square, y1_square) del robot usando las ecuaciones de cinemática directa (líneas 53–65), generando así el recorrido cuadrado completo.

Thank You