Análisis de Trayectorias en un Robot Diferencial a partir de /cmd_vel y Puzzlebot Sim

Bruno Manuel Zamora Garcia A01798275

Juan Paulo Salgado Arvizu A01737223

Carlos Adrian Delgado Vazquez A01735818

Alfredo Diaz Lopez A01737250

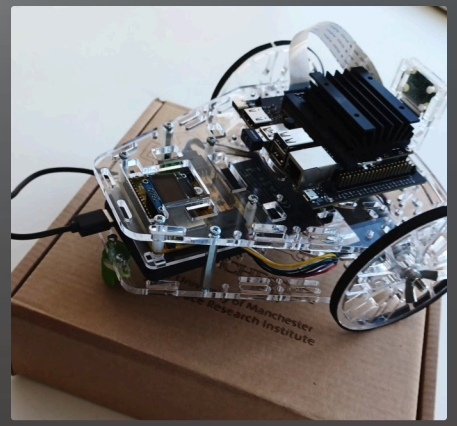
Contenido

- **Uso de /cmd_vel:**
  En sistemas ROS, el tópico **/cmd_vel** es el canal mediante el cual se transmiten los comandos de velocidad lineal y angular al robot. Este tópico es fundamental, ya que permite que el robot reciba instrucciones de movimiento en tiempo real.
- **Simulación con Puzzlebot Sim:**
  El nodo Puzzlebot Sim es una herramienta de simulación que interpreta los comandos publicados en **/cmd_vel** y reproduce el movimiento del robot de manera visual.
- **Objetivo de la Presentación:**
  Explicar cómo, mediante la generación de trayectorias de velocidad (definidos en u y w), se obtiene la pose del robot en el plano y se calculan las velocidades angulares de las ruedas.

Objetivos Específicos

1. **Cinemática Diferencial del Robot:**
   Ecuaciones fundamentales de la cinemática diferencial, que relacionan la velocidad lineal u y la velocidad angular w con la evolución de la posición y la orientación del robot.

- **Generación de Perfiles de Velocidad:**
  Construccion de perfiles escalonados para u y w a partir de tramos de giro y tramos de movimiento lineal. Enfocado a que los perfiles son apegados a los mensajes que se envían a través del tópico **/cmd_vel**.
- **Simulación de la Pose:**
  Mostrar cómo se actualizan las variables x, y y ϕ en cada iteración, simulando la evolución del robot en el entorno de **Puzzlebot Sim**.
- **Cálculo de Velocidades Angulares de las Ruedas:**
  Explicar la fórmula que permite derivar las velocidades angulares ωL y ωR a partir de u y w, demostrando cómo se usa el comando recibido de **/cmd_vel** entre las ruedas del robot.



Descripción del Robot Diferencial

> **Modelo Cinemático:**
>
> - Ecuaciones clave:
> - $x = u\cos(\phi)$
> - $y = u\sin(\phi)$
> - $\phi = w$

Parámetros del Robot:

- **u:** Velocidad lineal (se encarga directamente de la traslación del robot).
- **w:** Velocidad angular (controla la tasa de giro del robot).
- **r:** Radio de las ruedas, que afecta el cálculo de las velocidades angulares de cada rueda.
- **L:** Distancia entre las ruedas, determinante para diferenciar el giro de cada rueda.

Relación con /cmd_vel y Puzzlebot Sim

- **El Tópico /cmd_vel en ROS:**

En ROS, el tópico **/cmd_vel** se utiliza para enviar mensajes de tipo Twist que contienen la velocidad lineal y la velocidad angular al robot.

- **Puzzlebot Sim:**

Puzzlebot Sim es el nodo de simulación que interpreta los mensajes de **/cmd_vel** y visualiza el movimiento del robot en un entorno 3D o 2D. Permite observar cómo los cambios en u y w se traducen en trayectorias y giros.

La integración de /cmd_vel y Puzzlebot Sim permiten probar la lógica de control del robot antes de desplegarla en hardware real, lo cual es fundamental en el desarrollo de sistemas robóticos.

Estructura General del Código

Organización del Código:

- **Inicialización:**
  Se define el tiempo total de simulación, el paso de muestreo y las condiciones iniciales (posición y orientación).
- **Generación de Perfiles de Velocidad:**
  Aquí se construyen los vectores u y w de manera escalonada, simulando los comandos que se enviarían a través del tópico **/cmd_vel**.
  - **Tramos de Giro:** Se definen ángulos y duración, estableciendo $w \neq 0$ y $u = 0$
  - **Tramos Lineales:** Se calculan velocidades basadas en distancias y tiempos, estableciendo $u \neq 0$ y $w = 0$.
- **Bucle de Simulación:**
  Se recorre el tiempo de simulación actualizando la pose del robot (x,y,ϕ) en cada paso, replicando la respuesta del robot a los comandos de **/cmd_vel**.
- **Cálculo de Velocidades Angulares de las Ruedas:**
  A partir de u y w, se derivan las velocidades de las ruedas ωL y ωR .
- **Visualización y Animación:**
  Se emplean funciones como MobileRobot_5 y MobilePlot_4 para mostrar en tiempo real la evolución de la trayectoria y la pose del robot.

Toda esta estructura general nos permite ver cómo se pasa desde el comando de velocidad recibido en /cmd_vel hasta la simulación visual en Puzzlebot Sim.

Parámetros de Tiempo y Condiciones Iniciales

- **Definición del Tiempo de Simulación:**

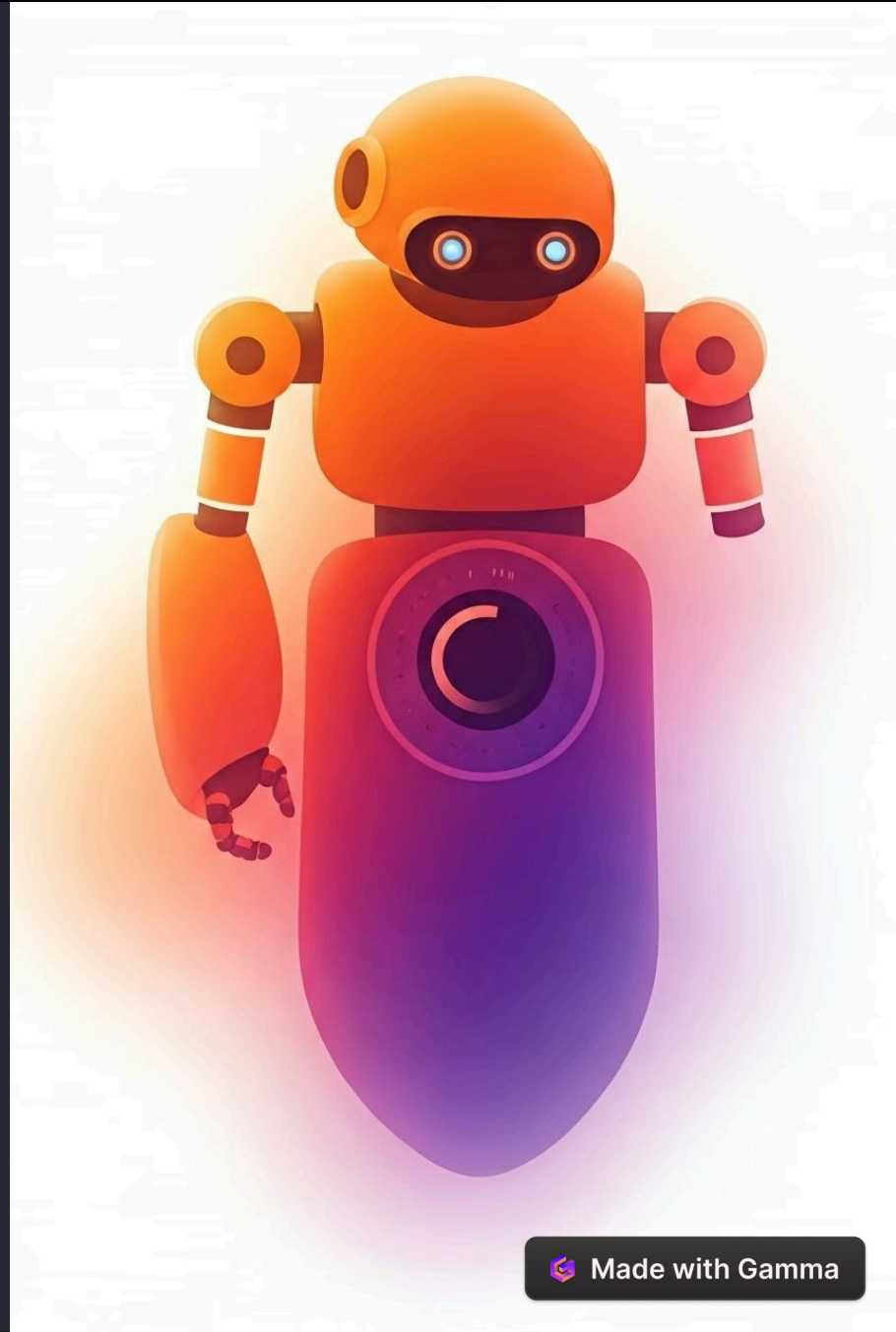
```
tf = 20;    % Tiempo total de simulación (s)
ts = 0.1;   % Intervalo de muestreo (s)
t = 0:ts:tf;
N = length(t);  % Número de muestras
```

Explicación:
Se establece el tiempo total y el intervalo en el que se muestreará la evolución del robot. Estos parámetros son fundamentales para reproducir de manera precisa el efecto de los comandos que se simularían en **/cmd_vel**.

- **Condiciones Iniciales:**

```
x1(1) = 0;    % Posición inicial en x
y1(1) = 0;    % Posición inicial en y
phi(1) = 0;   % Orientación inicial
```

Dichos parametros iniciales son equivalentes a poner al robot en una posición conocida antes de que se le envíen los comandos por /cmd_vel.



Generación de Velocidades de Referencia (u y w)

- **Definición de los Tramos de Movimiento:**

```
angles = [0, pi/2, pi/2, pi/2];        % Ángulos de giro en radianes
distances = [4, 4, 4, 4];              % Distancias lineales (m)
duration_turn = [1/ts, 1/ts, 1/ts, 1/ts]; % Duración de cada giro (en muestras)
duration_lineal = [4, 4, 4, 4];        % Duración de cada tramo lineal (s)
```

- **Construcción del Perfil de Velocidades:**

Durante los intervalos de giro:

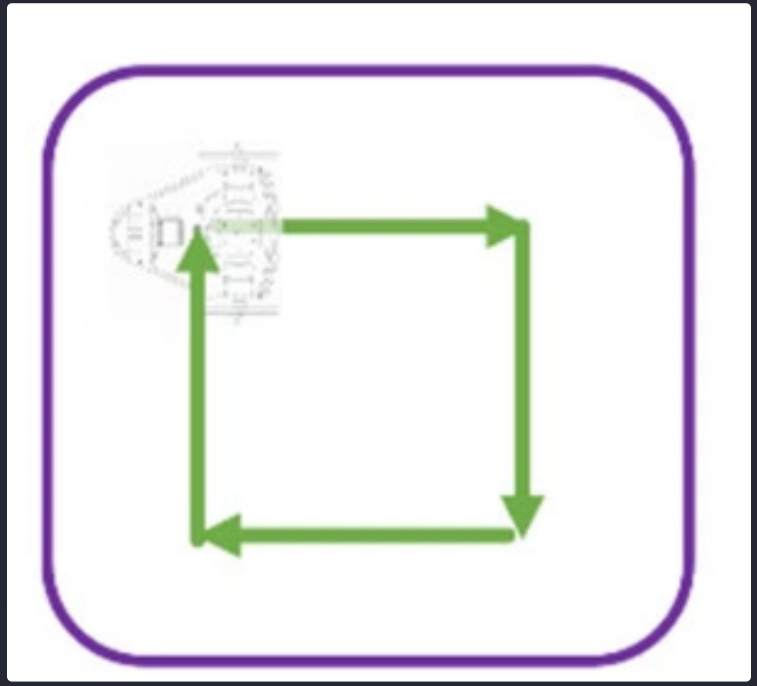
```
w(idx_start:idx_end_turn) = angles(i) / (duration_turn(i)*ts);
u(idx_start:idx_end_turn) = 0;
```

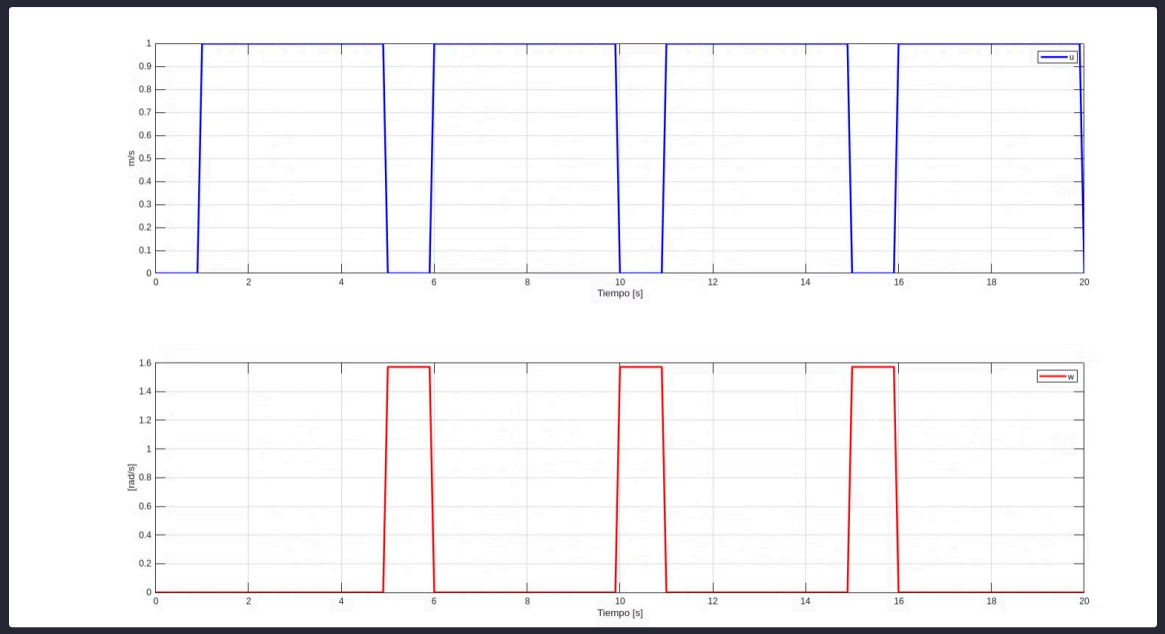
Durante los intervalos de movimiente lineal:

```
velocity_linear = distances(i) / duration_lineal(i);
u(idx_start_move:idx_end_move) = velocity_linear;
w(idx_start_move:idx_end_move) = 0;
```

Estos perfiles escalonados son equivalentes a lo que se enviaría por **/cmd_vel** en el sistema ROS , donde se especifica en cada instante la velocidad lineal y angular que el robot debe ejecutar.

Cuadrado





Gráficas de Velocidades de Referencia (u y w)

Codigo de la graficas:

```
subplot(211)
plot(t, u, 'b', 'LineWidth', 2);
grid on;
xlabel('Tiempo [s]');
ylabel('Velocidad Lineal [m/s]');

subplot(212)
plot(t, w, 'r', 'LineWidth', 2);
grid on;
xlabel('Tiempo [s]');
ylabel('Velocidad Angular [rad/s]');
```

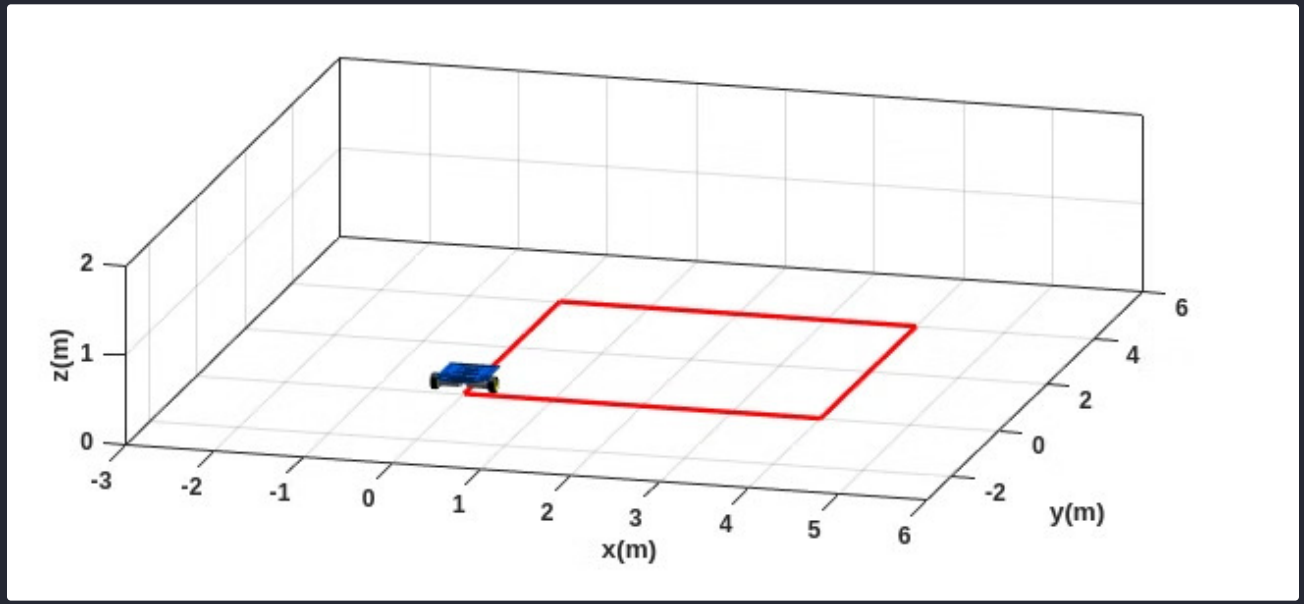
- **Gráfica Superior (u):**
  Muestra cómo se distribuyen los tramos de movimiento lineal. Se observa que durante los periodos de giro la velocidad lineal es cero, lo que es coherente con el comportamiento esperado cuando se envían comandos por **/cmd_vel**.
- **Gráfica Inferior (w):**
  Son los intervalos en los que se producen giros. La velocidad angular es diferente de cero en esos momentos y vuelve a cero durante los tramos de avance lineal.

Estas gráficas permiten comprobar que los perfiles de velocidad se han generado correctamente y simulan con precisión los comandos de velocidad que se recibirían a través del tópico **/cmd_vel**.

Trayectoria de Cuadrado

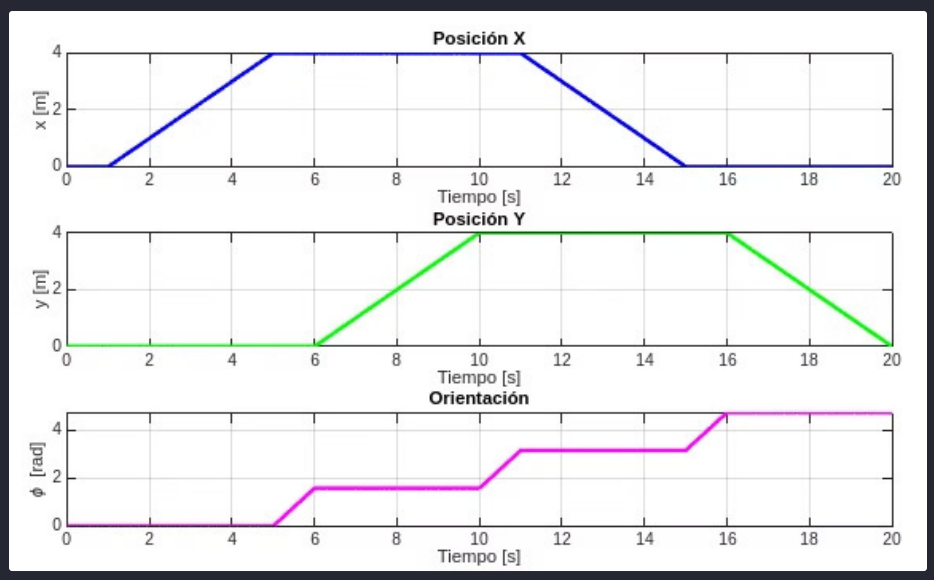
Generacion de las trayectoria:

```
% Parámetros de los giros y movimientos
angles = [0, pi/2, pi/2, pi/2]; % Ángulos de giro (puedes modificarlos)
distances = [4, 4, 4, 4]; % Distancias de los movimientos lineales en metros (puedes modificarlas)
duration_turn = [1/ts, 1/ts, 1/ts, 1/ts];     % Duración de cada giro (puedes ajustarlo)
duration_lineal = [4, 4, 4, 4]; % Duración de cada movimiento lineal en segundos (puedes ajustarlo)
```



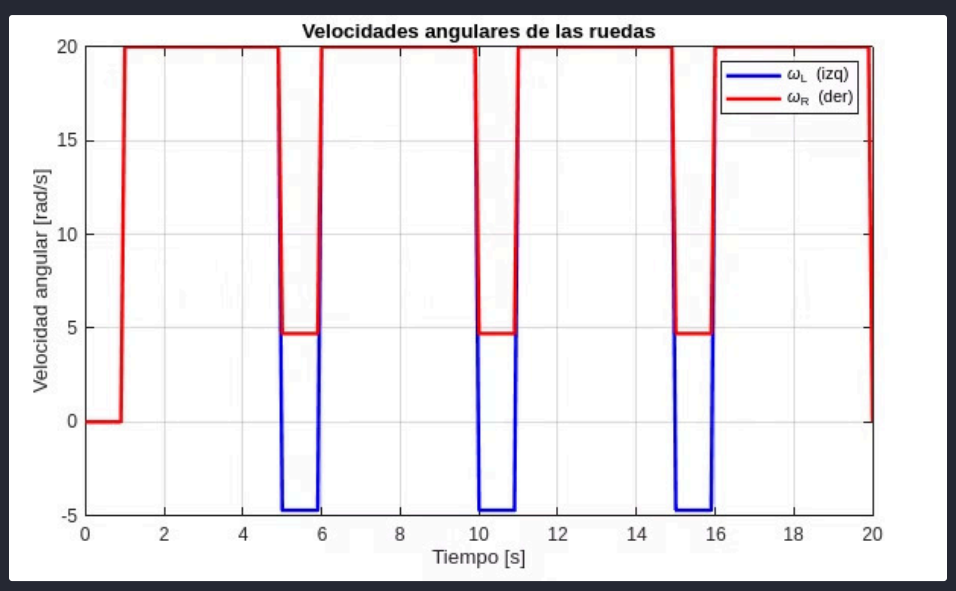
Gráficas de la Pose y Velocidades Angulares

- **Pose:**



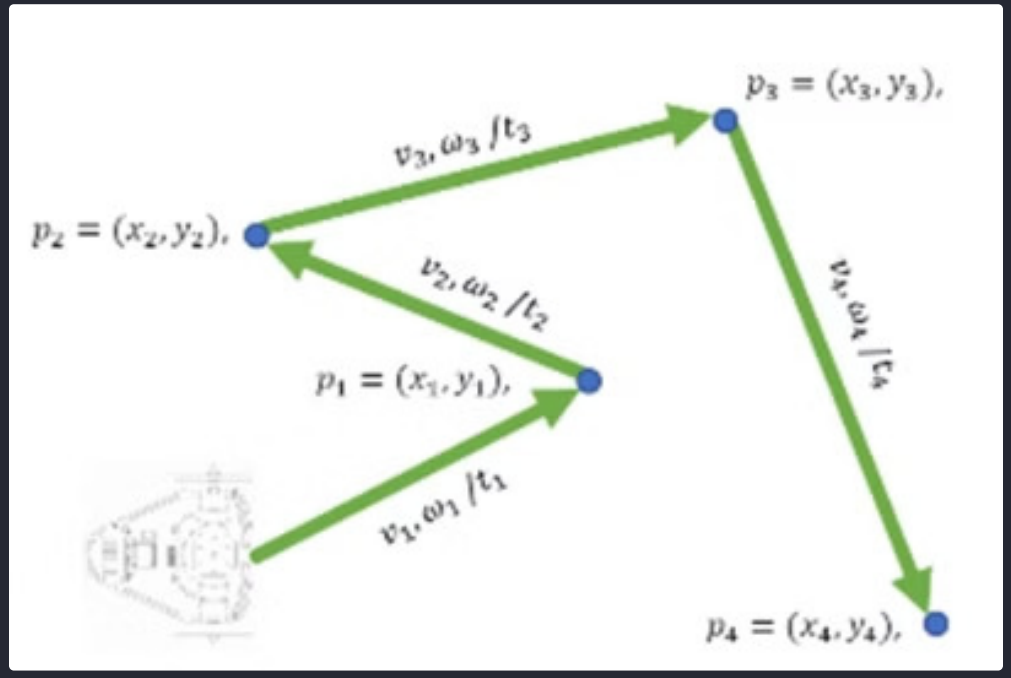

```
pose_graph = figure;
set(pose_graph, 'position', sizeScreen);

subplot(311)
plot(t, x1(1:N), 'b', 'LineWidth', 2), grid on, xlabel('Tiempo [s]'), ylabel('x [m]'), title('Posición X');

subplot(312)
plot(t, y1(1:N), 'g', 'LineWidth', 2), grid on, xlabel('Tiempo [s]'), ylabel('y [m]'), title('Posición Y');

subplot(313)
plot(t, phi(1:N), 'm', 'LineWidth', 2), grid on, xlabel('Tiempo [s]'), ylabel('\phi [rad]'), title('Orientación');
```

- **Velocidades angulares:**

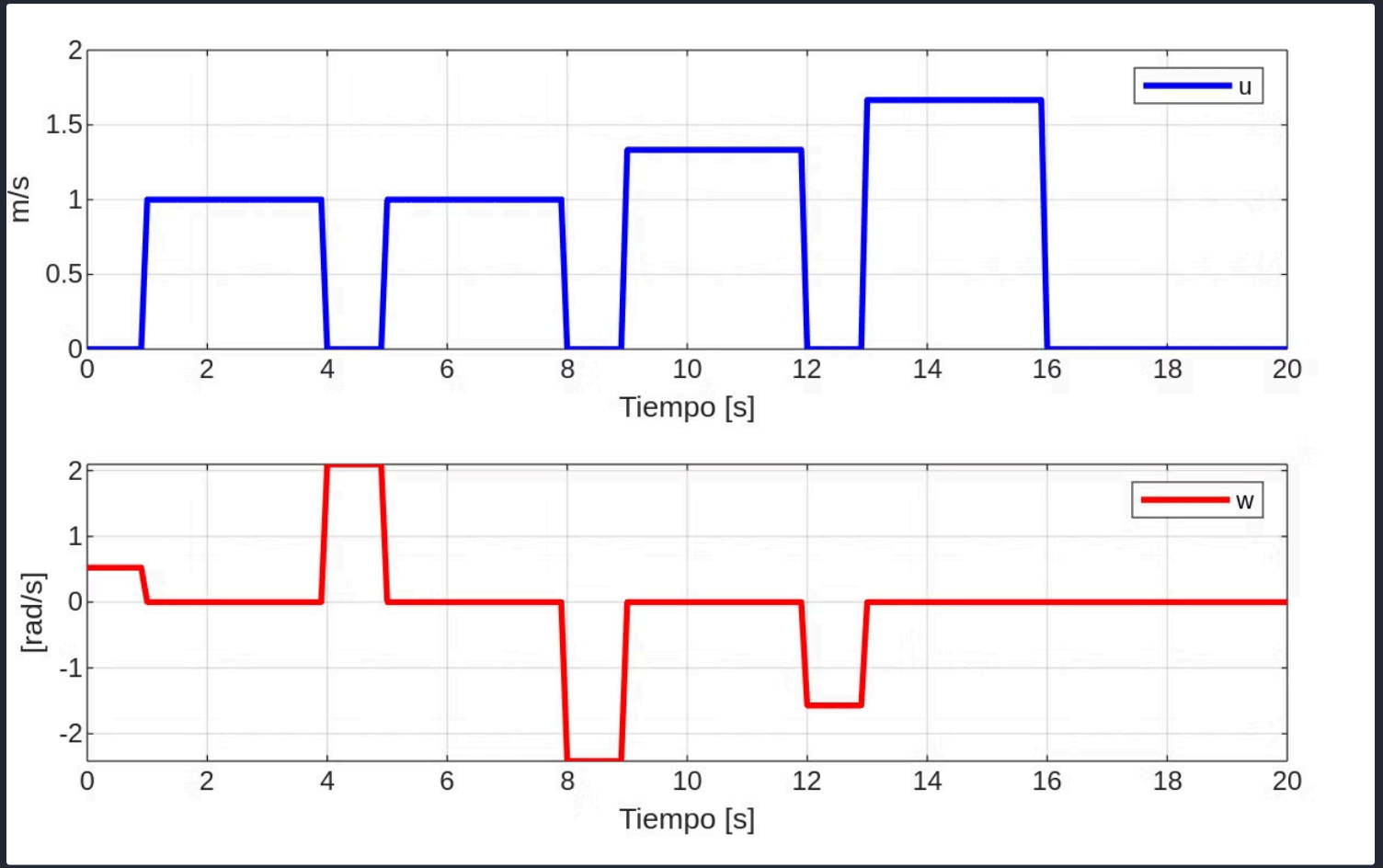



```
pose_graph = figure;
set(pose_graph, 'position', sizeScreen);

subplot(311)
plot(t, x1(1:N), 'b', 'LineWidth', 2), grid on, xlabel('Tiempo [s]'), ylabel('x [m]'), title('Posición X');

subplot(312)
plot(t, y1(1:N), 'g', 'LineWidth', 2), grid on, xlabel('Tiempo [s]'), ylabel('y [m]'), title('Posición Y');

subplot(313)
plot(t, phi(1:N), 'm', 'LineWidth', 2), grid on, xlabel('Tiempo [s]'), ylabel('\phi [rad]'), title('Orientación');
```

ZigZag

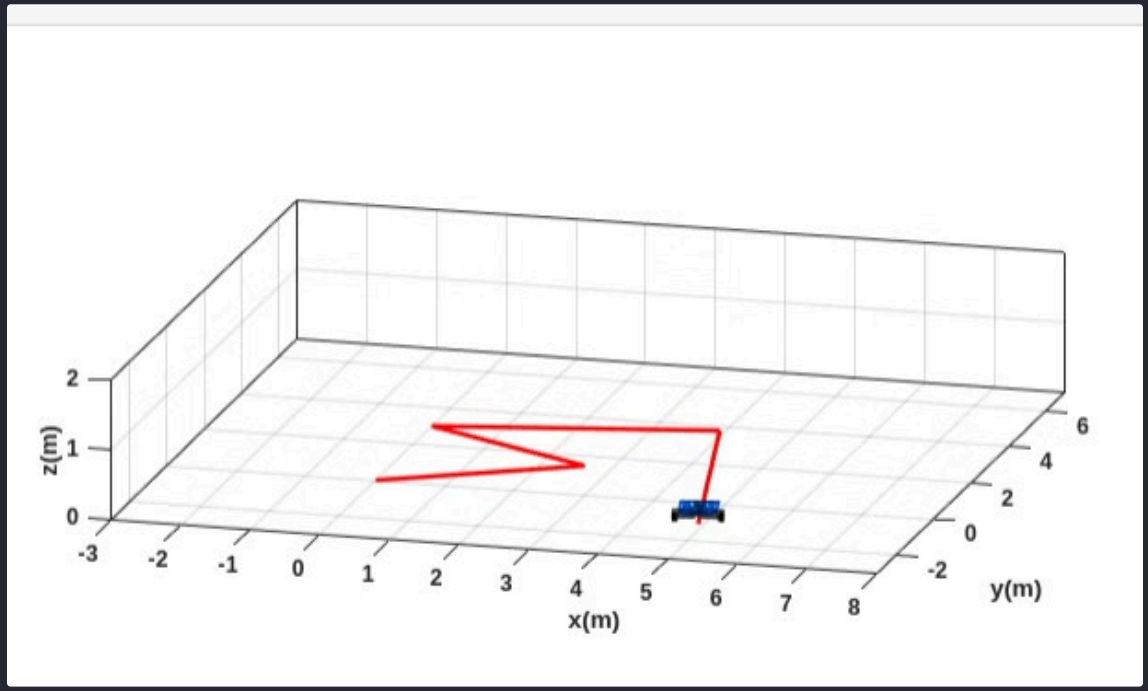


Gráficas de Velocidades de Referencia (u y w)

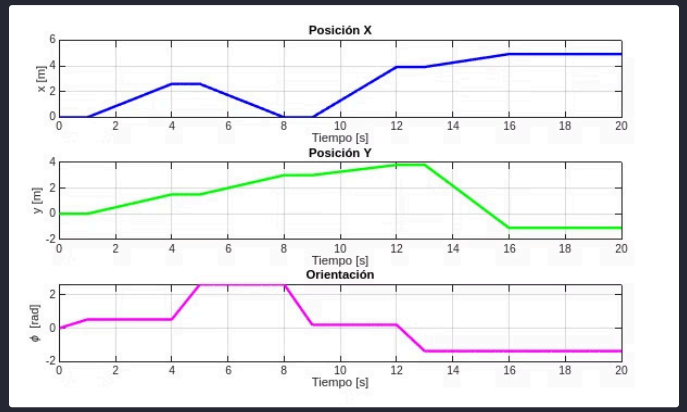


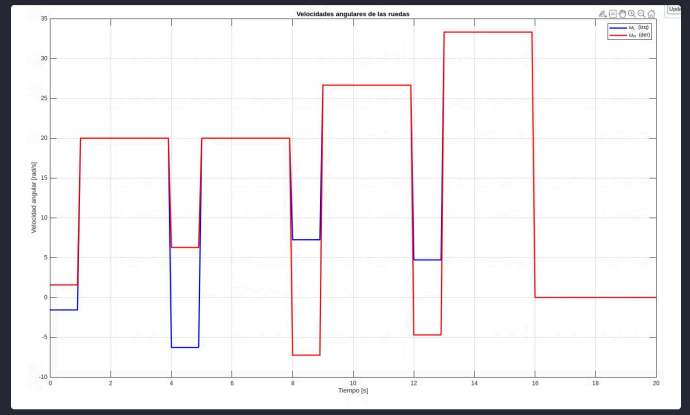
Trayectoria del Zigzag

```
% Parámetros de los giros y movimientos
angles = [pi/6, pi/1.5, -pi/1.3, -pi/2]; % Ángulos de giro (puedes modificarlos)
distances = [3, 3, 4, 5]; % Distancias de los movimientos lineales en metros (puedes modificarlas)
duration_turn = [1/ts, 1/ts, 1/ts, 1/ts];     % Duración de cada giro (puedes ajustarlo)
duration_lineal = [3, 3, 3, 3]; % Duración de cada movimiento lineal en segundos (puedes ajustarlo)
```



Gráficas de la Pose y Velocidades Angulares






```
%%%%%%%%%%%%%%%%%%%% GRAFICAR POSE Y VELOCIDADES ANGULARES %%%%%%%%%%%%%%%%%%%%
pose_graph = figure;
set(pose_graph, 'position', sizeScreen);

subplot(311)
plot(t, x1(1:N), 'b', 'LineWidth', 2), grid on, xlabel('Tiempo [s]'), ylabel('x [m]'),
title('Posición X');

subplot(312)
plot(t, y1(1:N), 'g', 'LineWidth', 2), grid on, xlabel('Tiempo [s]'), ylabel('y [m]'),
title('Posición Y');

subplot(313)
plot(t, phi(1:N), 'm', 'LineWidth', 2), grid on, xlabel('Tiempo [s]'), ylabel('\phi [rad]'),
title('Orientación');
```

```
wheel_graph = figure;
set(wheel_graph, 'position', sizeScreen);
plot(t, wl, 'b', 'LineWidth', 2), hold on
plot(t, wr, 'r', 'LineWidth', 2), grid on
xlabel('Tiempo [s]')
ylabel('Velocidad angular [rad/s]')
legend('\omega_L (izq)', '\omega_R (der)')
title('Velocidades angulares de las ruedas')
```