Memoria Teoría y Actividades
Robótica – Curso 21/22

David Campero Maña

Universidad de Huelva
Escuela Técnica Superior de Ingeniería

ÍNDICE

1. Introducción

Esta memoria corresponde con el trabajo general de la asignatura. Se va a dividir en 2 fases:

La primera fase corresponde con un resumen de la parte teórica de la asignatura, mediante las transparencias y los apuntes tomados en clase a lo largo del curso. Tendrá un orden al visto en teoría.

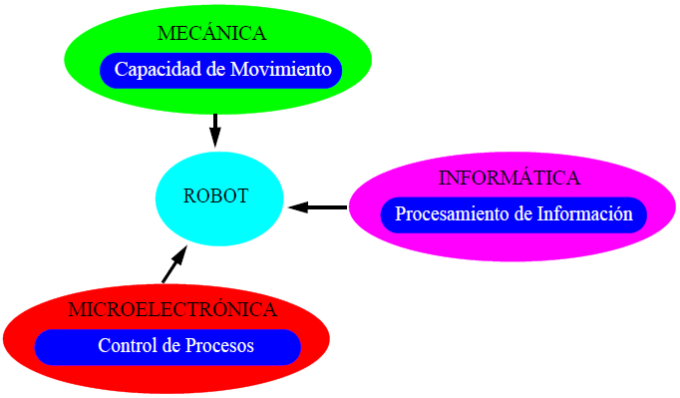
La segunda fase corresponde a las actividades de simulación previas al desarrollo de las diferentes prácticas planteadas, que han sido realizadas en pareja.

2. Resumen teórico de la asignatura

En esta parte se realizará un resumen general de la teórica impartida en la asignatura, centrados más concretamente en los conceptos empleados en simulación y prácticas.

2.1. Tema 1. Introducción.

Un robot es un dispositivo electrónico y generalmente mecánico, que desempeña una tarea automáticamente, ya sea de acuerdo con supervisión humana, un programa o un conjunto de reglas. A destacar de la historia de la robótica, que en periodo muy corto de tiempo se han realizado avances significativos, de pequeños robots con una simple función a robots capaces de realizar acciones muy complejas.



Los robots se pueden dividir en tres partes, la mecánica que es la que le da la capacidad de movimiento sobre el espacio con el uso de motores y sensores, la informática que le permite capturar y procesar la información obtenida por estos sensores y la microelectrónica que integra los componentes del robot y le permite el control de procesos.

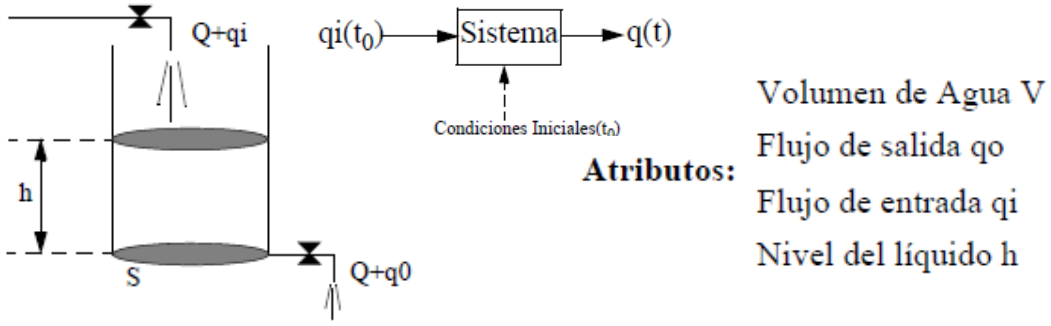
2.1.1. Modelado de Robots

Diferenciamos 2 tipos de modelado:

- Sistemas estáticos: El valor presente de los atributos depende solamente del valor presente de las interacciones externas. Ejemplo: Decodificador.

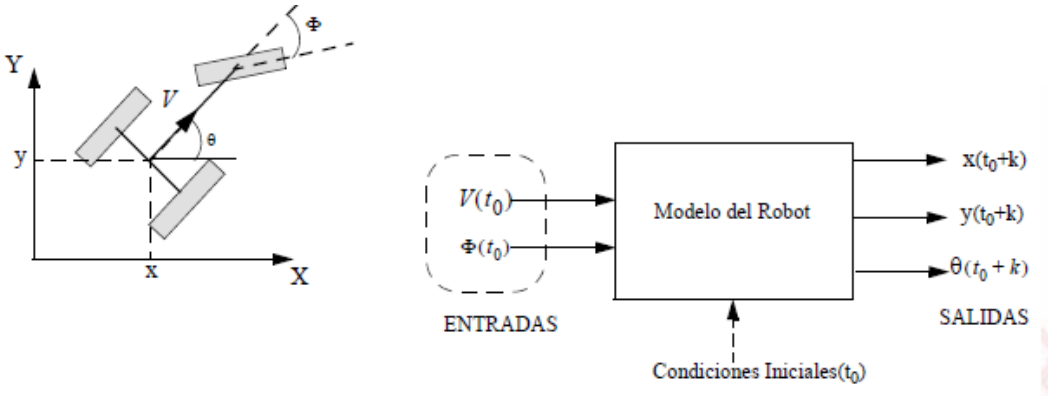
- Sistemas dinámicos: El valor presente de los atributos depende tanto del valor presente de las interacciones como de los valores pasados de los atributos. Ejemplo: Sistemas secuenciales.

En los sistemas dinámicos tenemos los estados, a partir de ellas podemos obtener el valor de los atributos que caracterizan a nuestro sistema, y junto con las variables de entrada nos permite predecir la evolución de estos atributos



Sistema dinámico de un depósito con sus atributos, entradas y salidas

Podemos aplicar este concepto al área de la robótica de esta manera:

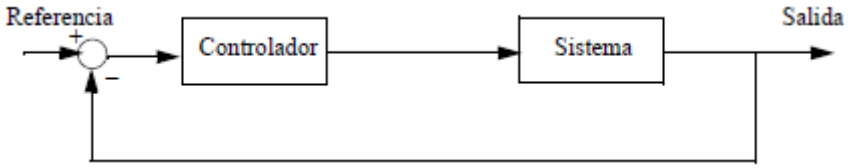


Robot triciclo con el conjunto de entradas y salidas

La magnitud de la entrada puede estar expuesta de manera continua si los datos varían de forma continua en el tiempo o de manera discreta si estos datos vienen dados en forma de muestras. Ejemplo de entrada continua es el sistema de llenado de una caldera y otro de manera discreta una cinta transportadora.

2.1.2. Sistemas de Control

El control de un sistema dinámico es un mecanismo que nos permite actuar durante un proceso continuo con el fin de que las magnitudes alcancen un determinado valor



Sistema con controlador

El control de sistemas de eventos discretos se denomina control secuencial, ya que los sistemas de mando adquieren una estructura secuencial:

- El proceso se divide en una serie de estados.

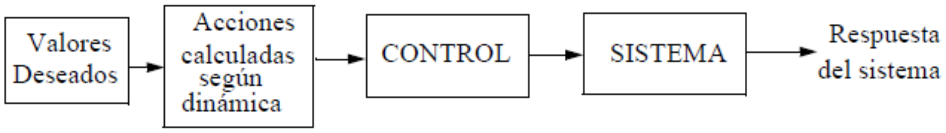
- Cada estado se activa y desactiva de forma secuencial.

- Cada estado activo tiene asociada una serie de acciones.

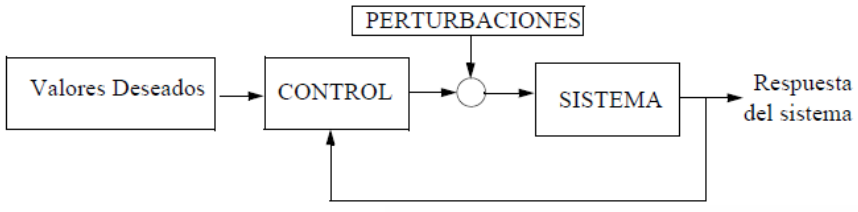
Si en un proceso se involucran magnitudes continuas y discretas a la vez hablamos de un control hibrido y si le añadimos técnicas de inteligencia artificiales como la lógica borrosa, redes neuronales, algoritmos genéticos… hablamos de control inteligente.

Hay dos tipos de estructura de control:

- Control en bucle abierto: El bloque de control actúa sobre el sistema de acuerdo con unos objetivos previamente establecidos, es una técnica muy sensible a perturbaciones ya que pueden alejar al sistema del comportamiento deseado.

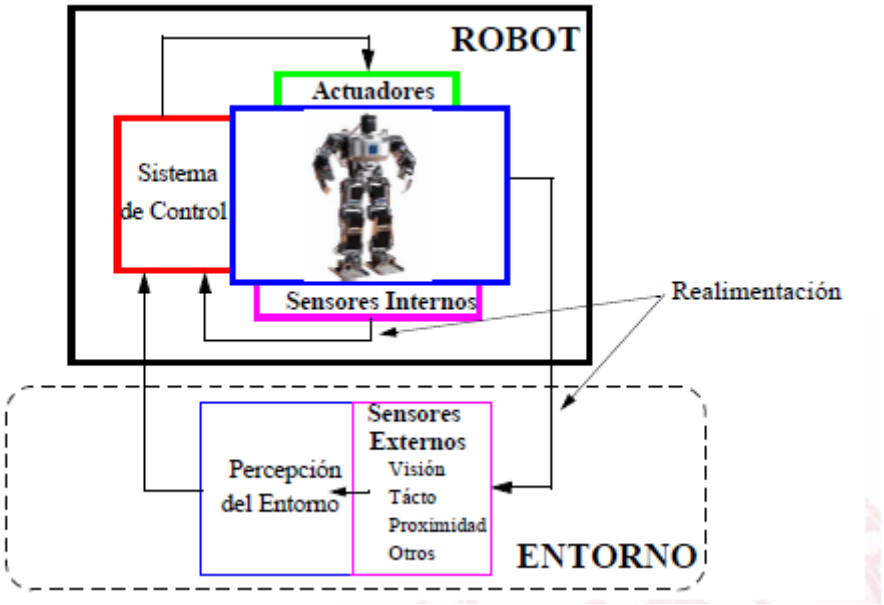


- Control en bucle cerrado: El controlador considera la salida, modificando en función de esta, la acción a realizar sobre el sistema. Este tipo presenta una estructura de realimentación.



2.1.3. Sistemas Robóticos

Esquema General de un sistema robótico



El funcionamiento del robot se puede dividir una arquitectura de capas, en la etapa de inteligencia artificial (¿Cómo lo voy a hacer?), etapa de control (¿Cómo consigo hacerlo?) y la etapa de mecatrónica (¿Qué medios necesito?), que es la que interactúa con el entorno.

Podemos diferenciar diferentes tipos de robot:

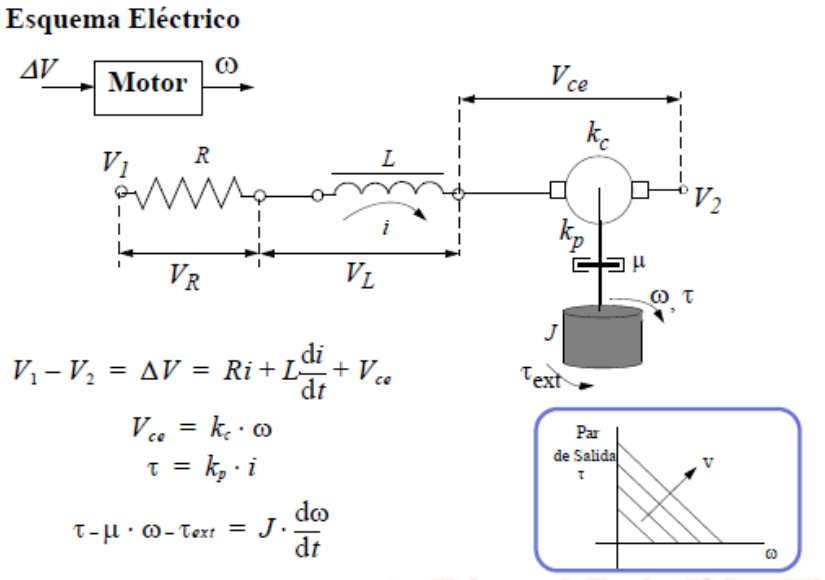
- Humanoides: Robots con apariencia humana que intentan imitar su comportamiento.
- Robot Móvil Terrestre: Robot sobre una plataforma móvil que se desplaza de forma autónoma.
- Robot Marinos.
- Robot Aéreos.
- Robot Industrial: Se usan en cadenas de montaje y están diseñados para cargar piezas, herramientas o dispositivos especializados mediante movimientos variables programados para el desarrollo de diferentes tareas.
- Robot de Servicios: Robots que operan total o parcialmente para realizar servicios útiles, excluyendo aquel que realiza operaciones de fabricación.

2.2. Tema 2. Actuadores y Sensores

2.2.1. Actuadores

Los actuadores son los elementos que generan las fuerzas o pares necesarios para animar la estructura mecánica del robot. Estudiaremos los eléctricos, neumáticos e hidráulicos.

- Accionadores eléctricos: En este apartado podemos incluir los solenoides o bobinas que producen el desplazamiento de un núcleo de hierro, pero en realidad los actuadores eléctricos más usados son los motores.



Existen varios tipos de control para los motores:

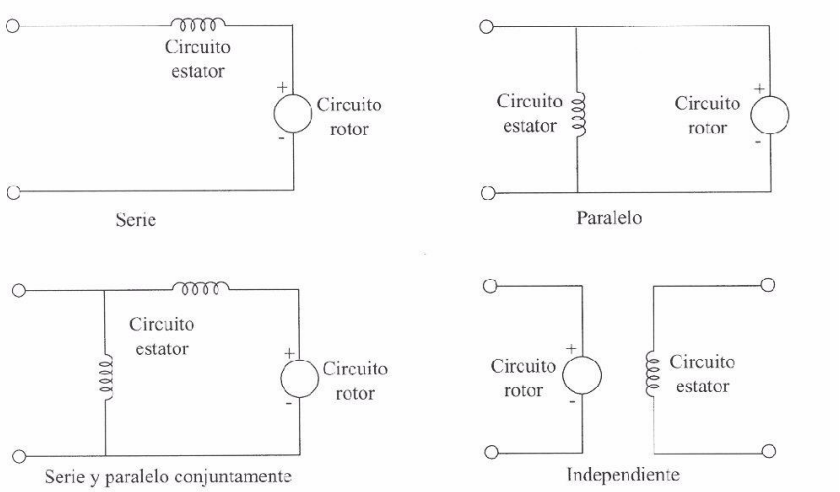
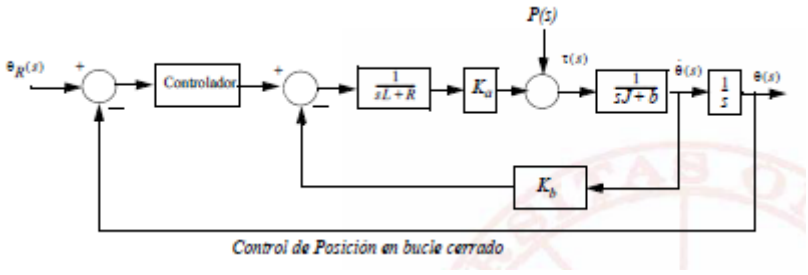
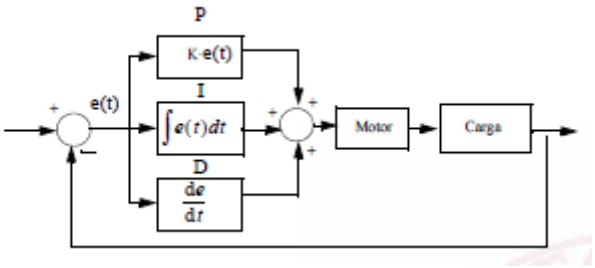


Figura 2.27: Configuraciones del estator y el rotor.

Motores controlados por la corriente de excitación



Motores controlados por el inducido (posición)



Motores controlados por el inducido. Estrategia de control

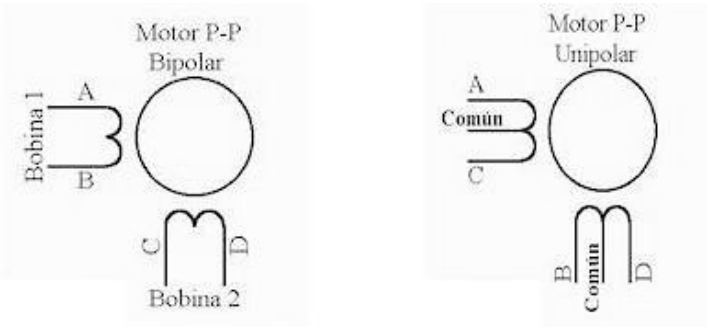
Diferentes tipos de motores que podemos encontrar:

- Motores sin escobillas: En estos motores el circuito por donde pasa la corriente está situado en el estator y el imán permanente está en el rotor. Para conseguir el giro en el estator se realizan varios bobinados que se excitan mediante circuitos conmutadores electrónicos. Tienen un bajo mantenimiento.
- Motores paso a paso: Estos motores pueden moverse un paso por cada pulso que se aplique. Este paso puede ser grande (90º) o pequeño (1.8º) y poseen la propiedad de poder quedar enclavados en una posición o estar completamente libres. Están formados por un rotor sobre el que van aplicados distintos imanes permanentes y por un cierto número de bobinas en su estator. Estas bobinas están manejadas por un controlador externamente. Este tipo de motores tienen dos variaciones:
    - Reluctancia variable: No tienen imanes permanentes y el rotor sólo tiene material ferromagnético. Es el equivalente magnético a la resistencia de un circuito eléctrico.
    - Imán permanente: Ya mencionados anteriormente. Está formado por un rotor sobre el que se van aplicando distintos imanes permanentes y un numero de bobinas excitadoras bobinas en el estator.

Tipos de motor P-P:

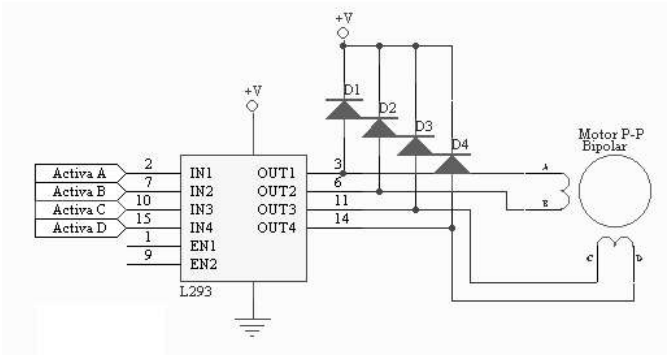
- <u>Bipolar:</u> tienen cuatro terminales generalmente para controlar la excitación de las bobinas. Requieren cambio en la dirección de flujo para realizar el movimiento.

- <u>Unipolar:</u> Suelen tener 6 o 5 terminales para controlar la excitación del estator. Son más simples de controlar.

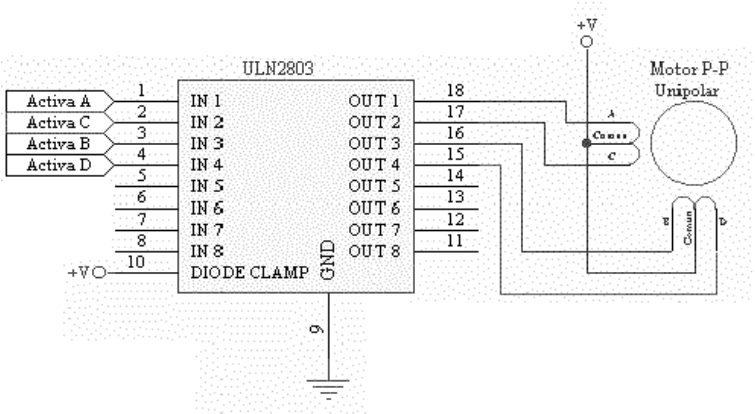


Tipos de motor P-P

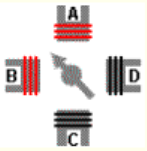
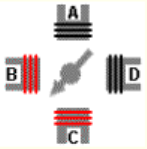
PASO	TERMINALES			
	A	B	C	D
1	+V	-V	+V	-V
2	+V	-V	-V	+V
3	-V	+V	-V	+V
4	-V	+V	+V	-V



Control motor bipolar



Control motor unipolar

PASO	Bobina A	Bobina B	Bobina C	Bobina D	
1	ON	ON	OFF	OFF	
2	OFF	ON	ON	OFF	
3	OFF	OFF	ON	ON	
4	ON	OFF	OFF	ON	

Secuencia normal para avanzar un paso por vez

Existen otros dipos de secuencia como el tipo “wave drive”, donde se activa una sola bobina a la vez para brindar un funcionamiento más suave, pero el torque y la retención del motor es menor, y los de tipo medio paso, en los que se activan las bobinas de maneara que el movimiento es la mitad del paso real.

- Accionadores Neumáticos e Hidráulicos: Suelen utilizarse cuando es necesaria una potencia mayor que la que puede suministrar un motor. El principio de accionamiento en ambos es igual y existen dos tipos de cilindro. Los hidráulicos permiten generar mayores fuerzas mientras que los neumáticos permiten velocidades más altas:

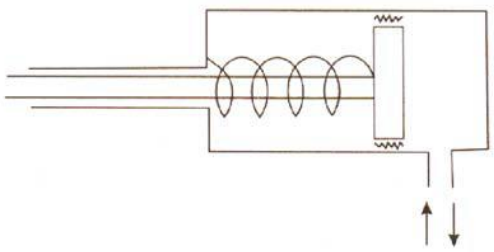


Figura 2.34: Cilindro de simple efecto.

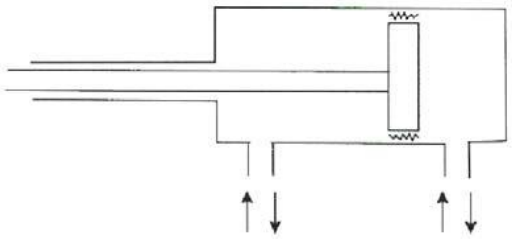


Figura 2.35: Cilindro de doble efecto.

El problema de los neumáticos es la compresibilidad del aire que hace complejo el control continuo de posición. Se usa mucho en robots de tipo “coger” y “poner”.

En los hidráulicos es más fácil el control continuo de posición, pero necesitan mayor mantenimiento frente a posibles fugas del líquido.

Accionadores Eléctricos *Ventajas*	*Desventajas*
•Rápidos y precisos •Posibilidad de aplicar varias técnicas de control •Más económicos •Timepos de respuesta rápidos	•Altas velocidades implican bajo par • Necesidad de engranajes •No adecuados en atmósferas inflamables •Sobrecalientamiento en condiciones de parada •Coste alto en motores grandes
Accionadores Neumáticos *Ventajas*	*Desventajas*
•Más económicos •Alta velocidad de accionamiento •No contaminan	• Compresibilidad del aire: limita el control y la precisión • Mala precisión con cargas •Necesidad de instalación adicional
Accionadores Hidráulicos *Ventajas*	*Desventajas*
•Relación potencia-peso muy buena •Muy buen servo control •Trabajo en paro sin problemas •Adecuado en atmósferas inflamables	• Instalación hidraúlica costosa • Necesidad de mantenimiento, fugas de aceite • Problemas de miniaturización •Necesidad de instalación adicional

Resumen de los actuadores.

2.2.2. Sensores

Podemos clasificar los sensores según el qué hacen y el cómo lo hacen:

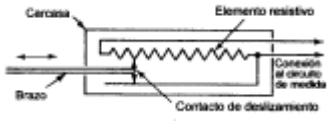
- Sensores propioceptivos: Miden los valores internos del robot (velocidad del motor, estado de la batería…)
- Sensores exteroceptivos: Obtienen información del entorno donde se encuentra el robot (distancia a los objetos, intensidad de la luz…)
- Sensores pasivos: La energía viene del entorno.
- Sensores activos: Emiten su propia energía y miden la reacción. Tienen mejores resultados, pero tienen cierta influencia sobre el entorno.

Características de los sensores:

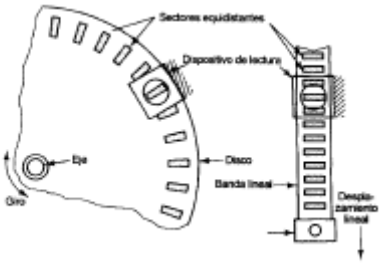
- Rango: Valores mínimos y máximos que son posibles medir
- Linealidad: Relación lineal entre entrada y salida.

- Exactitud: Mayor error esperado entre la señal real y la medida.
- Histéresis: La salida del sensor para una determinada entrada varía dependiendo de que la entrada este aumentando o disminuyendo.
- Repetitividad: Variación de salida frente a una misma entrada.
- Resolución: Unidad más pequeña que puede medir.
- Saturación: Una vez pasado un determinado valor de entrada, la salida deja de aumentar para pasar a un valor fijo.
- Zona Muerta: Rango de entrada en la que el sensor no es capaz de apreciar medida.
- Sensibilidad: Relación entre el cambio de valor en la salida producido por un cambio de valor en la entrada.
- Ruido: El nivel de señal espuria que no corresponde a un cambio en la entrada.

Para medir desplazamientos lineales y giros se usa los potenciómetros que funcionan con un sistema de dientes que controlan la posición de un brazo o en caso de que sean ópticos, una plantilla que va girando se encuentran los sensores que controla un dispositivo de lectura.



Potenciómetro



Codificador óptico

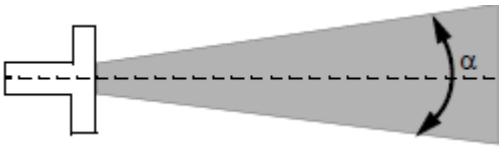
Sensores inerciales:

- Acelerómetros: Miden las aceleraciones, se basan en la medida de la fuerza de inercia generada cuando una masa es afectada por un cambio de velocidad.
- Giróscopos: Miden la velocidad de rotación devolviendo una señal proporcional a la velocidad de rotación
- IMU: Es un dispositivo electrónico que recoge información acerca de la velocidad, orientación y fuerzas gravitacionales de un aparato usando una combinación de acelerómetros y giróscopos.

Los sensores de tacto, proximidad y presencia miden la distancia o el contacto del robot con el entorno. El sensor de proximidad envía una onda que viaja y rebota contra la superficie y vuelve al sensor. Dependiendo del tiempo que tarda en llegar, se calcula la distancia al obstáculo.

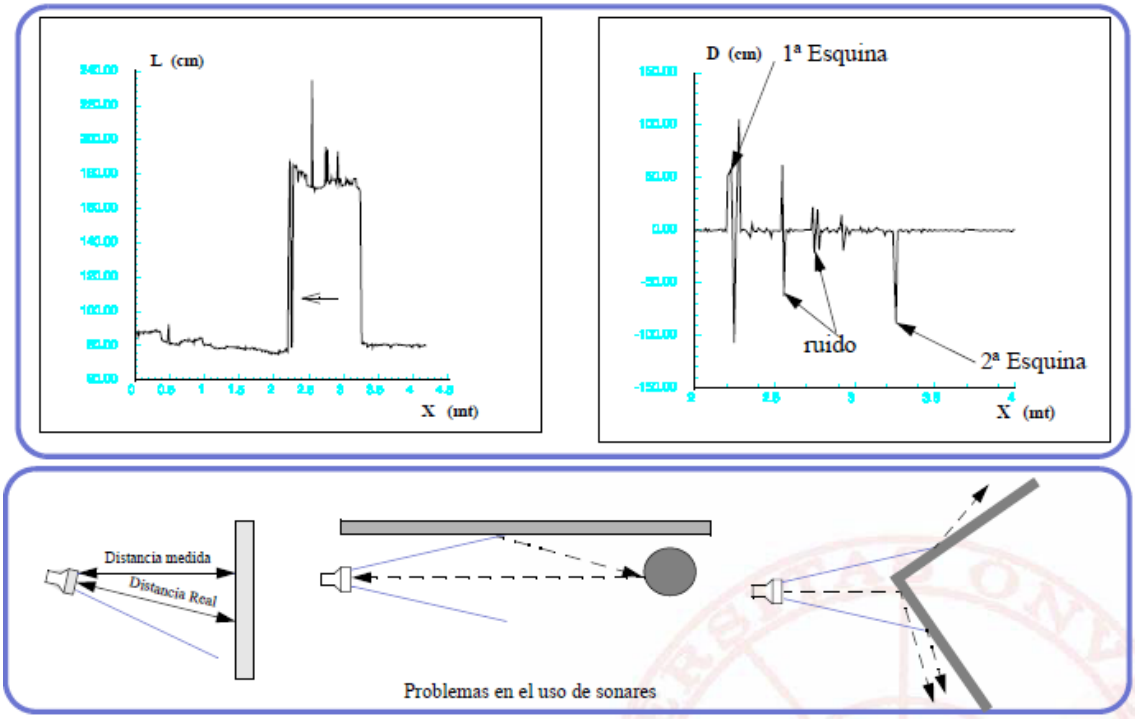
$$d = c \text{ x } t$$

d = distancia recorrida
c = velocidad de propagación de la onda
t = tiempo de vuelo

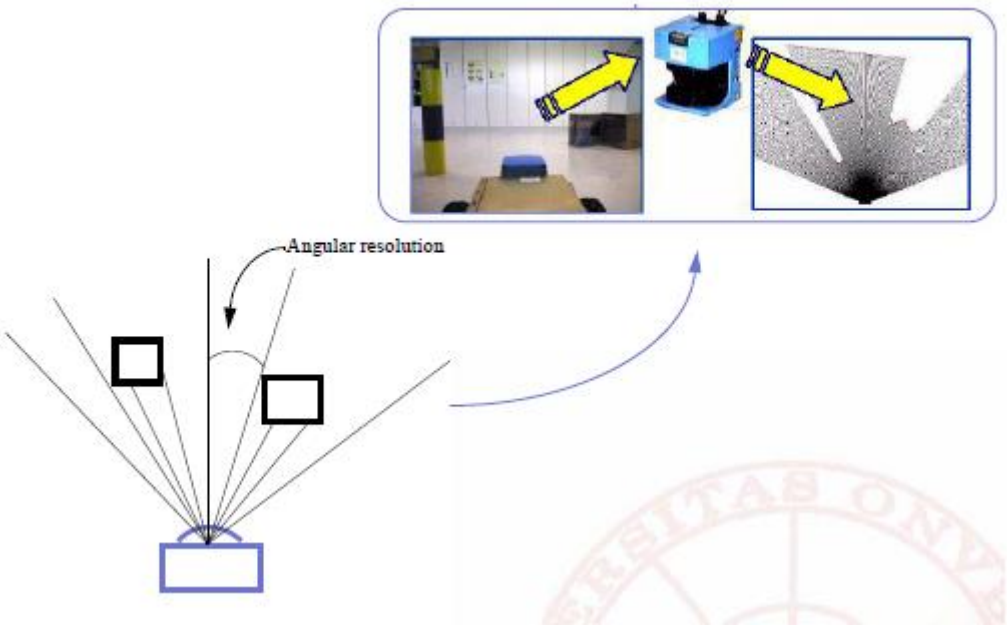


Amplitud del sensor

El sensor sonar no es un sensor muy preciso como hemos podido observar en las prácticas, dependiendo del ángulo de incidencia de la onda, hay casos en los que la onda rebota en otra dirección y no vuelve al sensor



Los sensores infrarrojos y los laser son sensores más precisos que los sonar.

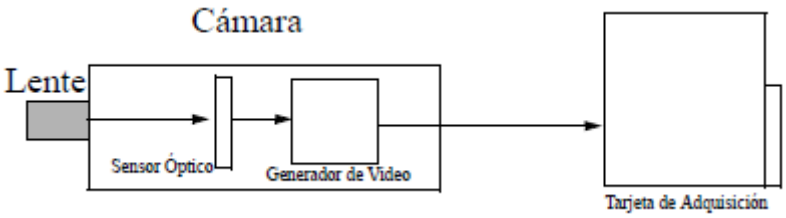


Sensor Laser

Otro tipo de sensores son los de posicionamiento global:

- Navegación Interior: Sistema laser para detectar balizas, basados en triangulación desde un punto del entorno
- Navegación Externa: GPS, funciona mediante una red de 24 satélites, 21 operativos y 3 de reposo. El receptor recibe de los satélites unas señales indicando la posición de estos y el reloj de cada uno de ellos. En base a estados datos, calcula la distancia al satélite. Con esta información se obtiene la posición 3D mediante la intersección de tres esferas. Dado que los relojes no están sincronizados, y la posición no es muy precisa. Con la información de un cuarto satélite es posible eliminar esta falta de sincronización entre los relojes de los receptores GPS y relojes de satélites, pudiendo así determinar una posición exacta 3-D (latitud, longitud y altitud).
- DGPS (Differential GPS), es un sistema que proporciona a los receptores de GPS correcciones a los datos recibidos de los satélites GPS.
- NMEA: El protocolo NMEA 0183 es un medio a través del cual los instrumentos marítimos y también la mayoría de los receptores GPS puede comunicarse los unos con los otros. Ha sido definido, y está controlado por la organización estadounidense National Marine Electronics Association
- Otros Sistema: GLONASS (antigua URSS), Galileo (Unión Europea), Beidou (China)
- Sensores de Fuerza-Par: Permiten medir las 3 fuerzas y 3 momentos de torsión que están actuando en cada momento sobre el punto a estudiar

- Sensores de velocidad:
  - Efecto Doppler: Se basan en emisión de una señal, lumínica, de hiperfrecuencia o sonara y medir el cambio de frecuencia de la señal reflejada en el elemento móvil con la señal emitida
  - Dinamo tacométrica: Son sensores de velocidad angular los cuales entregan una tensión continua analógica proporcional a la velocidad angular del objeto que gira, o sea que entreguen una tensión proporcional a la velocidad
  - Tacoalternador: Su principio de funcionamiento es similar que el de la dinamo con la diferencia que trabaja con alterna
- Cámaras
  - Tipos de sensores: Los sensores CCD son circuitos integrados que contiene un número determinado de condensadores enlazados o acoplados. Bajo el control de un circuito interno, cada condensador puede transferir su carga eléctrica a uno o a varios de los condensadores que estén a su lado en el circuito impreso. La alternativa digital a los CCD son los CMOS utilizados en algunas cámaras digitales y webcams
  - Tipos de generación de video: Entrelazado, progresivo
  - Tipos de cámaras: Cámaras matriciales y cámaras lineales
  - Calibración de la cámara: Proceso que permite establecer la relación entre las coordenadas tridimensionales de los objetos del entorno con sus correspondientes proyecciones bidimensionales en la imagen plana capturada por la cámara

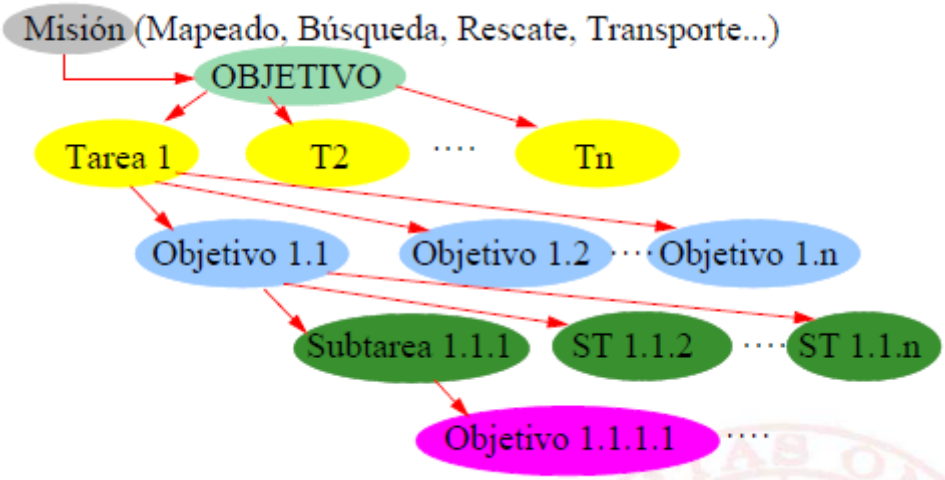


Sensores de una cámara

2.3. Tema 3. Robots Móviles

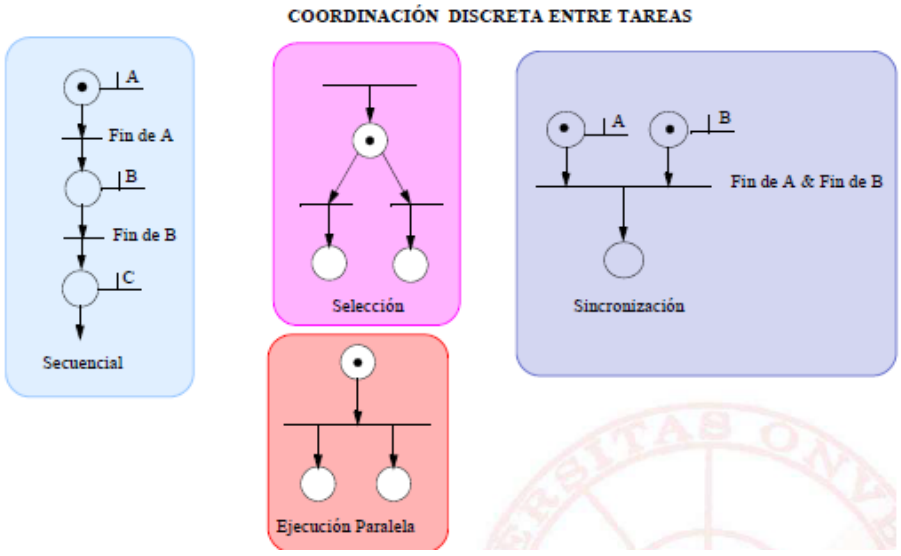
2.3.1. Introducción

La planificación de los robots se hace sobre una misión, que puede ser de mapeado, búsqueda, rescate, transporte…

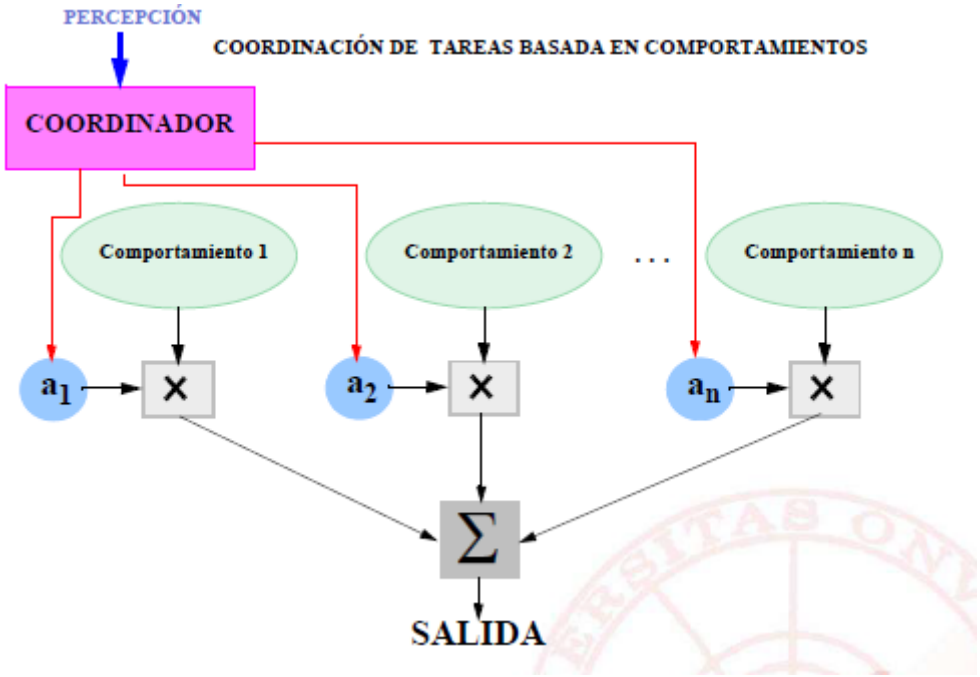


Subdivisión de tareas

Para que la ejecución de la misión se realice exitosamente, hay que realizarlas en coordinar las tareas. Estos son los modelos de coordinación:

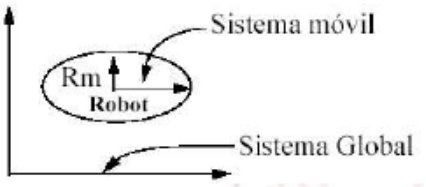


Esquemas de coordinación de las tareas



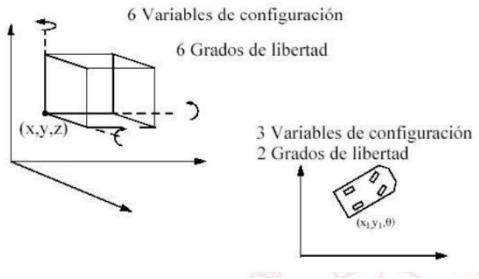
Coordinación basada en comportamiento

Al realizar la planificación, tenemos que tener en cuenta la configuración de nuestro robot, la especificación de la posición de cada uno de los puntos del mismo. Se considera Rm, conjunto de espacio euclídeo determinado por las variables de configuración, las coordenadas locales del robot.



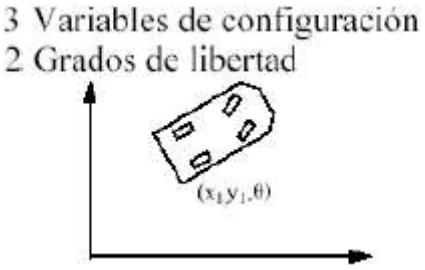
Sistema local frente al global

- Parámetros de configuración: Conjunto de valores que describen las relaciones de translación y orientación entre el sistema global y el local.
- Variables de configuración o generalizadas: Conjunto mínimo de magnitudes que permiten determinar la configuración del sistema.



- Espacio de Configuración: Espacio formado por todas las configuraciones posibles del robot.
- Espacio de trabajo: Subconjunto del espacio de configuración que puede ser ocupado por el robot en unas circunstancias determinadas.

- Variables de estado: Representan el menor conjunto posible de magnitudes que permiten describir la situación actual y la evaluación futura del sistema.
- Grados de Libertad: Número de variables de configuración que pueden cambiarse de forma independiente

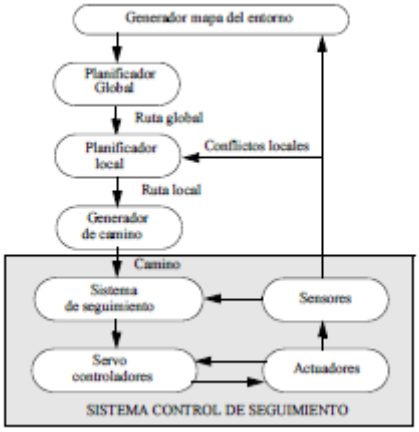


Ejemplo en vehículo móvil

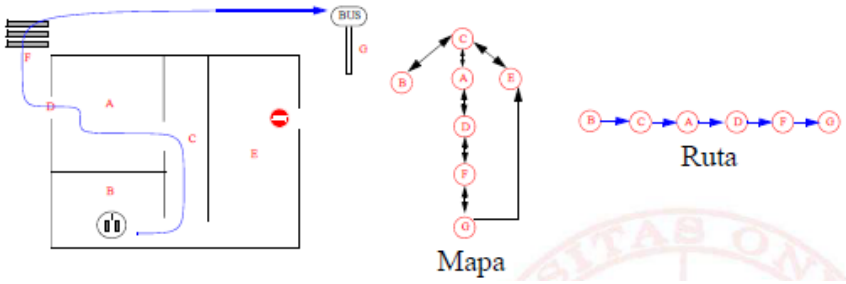
- Ruta: Secuencia ordenada de puntos del espacio de trabajo que representan metas intermedias entre la configuración de inicio y la configuración final.
- Camino: función continua que interpola la secuencia de objetivos definidos por una ruta.
- Trayectoria: Camino que tiene asociado un perfil cinemático.
- Inversor: punto de la trayectoria donde la velocidad lineal cambia de signo.
- Maniobra: Concatenación de trayectorias separadas por inversores.
- Maniobra restringida: Trayectoria, al final de la cual solo cambian los valores de ciertas variables de la configuración. En el resto de las variables los valores finales son iguales a los iniciales.
- Restricciones geométricas: El entorno está definido por objetos rígidos o móviles que limitan el espacio de trabajo.
- Cinemáticas: Restricciones intrínsecas del sistema, impiden que la velocidad del sistema pueda tomar cualquier valor.
- Camino admisible: Conjunto de Configuraciones que pueden ser alcanzadas sin colisión y sin violar las restricciones cinemáticas.

Tipos de Planificación:

- La navegación planificada puede tener a su vez una planificación global o local.
    - Planificación Global: Construir o planificar la ruta o camino que lleva al robot a cada una de las configuraciones determinadas por una tarea. Se genera un camino admisible aproximado al que finalmente se seguirá
    - Planificación Local: Permite resolver los conflictos particulares que presenta una planificación local, modificando el camino en función de la realimentación sensorial.
    - Control Planificado: Algoritmos de control que permiten seguir un camino o trayectoria previamente planificada.

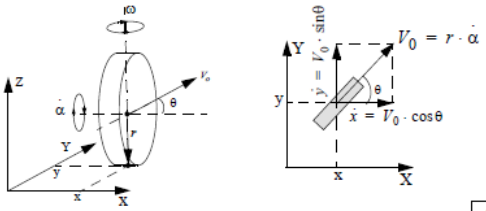


- Navegación Reactiva: Controla el movimiento del robot en virtud de la información sensorial que adquiere del entorno. Necesita sensores de proximidad y mecanismos para encontrar mapas locales.
- Navegación Topológica: No tiene en cuenta el espacio métrico para ubicar el robot. El mapa se a un elevado nivel de abstracción representándolo como un grafo de conectividad entre distintos tipos de nodos (pasillo, salas, parada de autobuses …) en los que el robot puede ubicarlos. Requiere de técnicas sensoriales de extracción de características, navegación reactiva y planificación métrica local



2.3.2. Características de los robots móviles

- Modelo de la rueda:



Variables de configuración: $P = [x_1\ Y_1\ \theta\ \alpha]$

Restricciones Cinemáticas: $\begin{cases} \dot{x}\ \sin\theta - \dot{y}\ \cos\theta = 0 \\ \dot{x}\ \cos\theta + \dot{y}\ \sin\theta - \dot{\alpha}\cdot r = 0 \end{cases}$

N° D.O.F: 4-2 = 2 → Modelo Cinemático: $\dot{P} = f(P,u)$

2 Grados de Libertad:
$\dot{\alpha}$ velocidad de rodado
ω velocidad de giro

$$\begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{\alpha} \\ \dot{\theta} \end{bmatrix} = \begin{bmatrix} r\cdot\cos\theta & 0 \\ r\cdot\sin\theta & 0 \\ 1 & 0 \\ 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} \dot{\alpha} \\ \omega \end{bmatrix}$$

Modelo Completo

$$\begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{\theta} \end{bmatrix} = \begin{bmatrix} \cos\theta & 0 \\ \sin\theta & 0 \\ 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} V_0 \\ \omega \end{bmatrix}$$

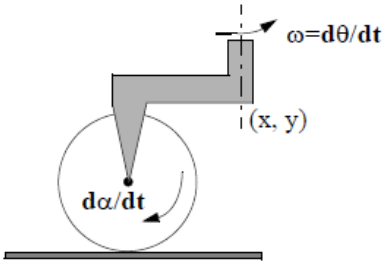
Modelo Simplificado

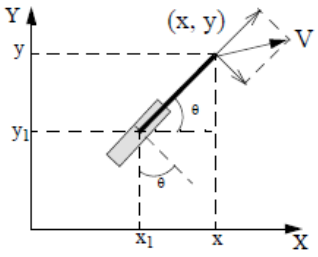
El modelo simplificado lo hemos empleado en prácticas

$$Vo = \frac{(\dot{\alpha_1} + \dot{\alpha_2}) * r}{2}$$

$$\omega = \frac{(\dot{\alpha_1} + \dot{\alpha_2}) * r}{2 * l}$$

- Rueda Castora:





Parámetros de configuración: $[x_1\ Y_1\ \theta_1\ \alpha\ \ x\ Y\ \theta]$
rueda enlace

Restricciones Holónomas:

$$x_1 = x + l \cdot \cos(\theta) \quad ; \quad y_1 = y + l \cdot \sin(\theta)$$
$$\theta_1 = \theta$$

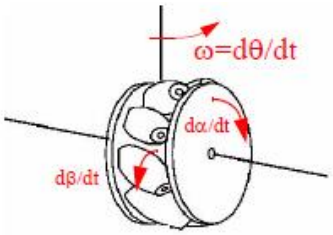
$$V = V_1 + \omega \wedge l$$

Variables de configuración (7 - 3) = 4: P = [x Y θ α]

Restricciones No Holónomas:

$$\begin{cases} \dot{x}\ \sin\theta - \dot{y}\ \cos\theta + l \cdot \dot{\theta} = 0 \\ \dot{x}\ \cos\theta + \dot{y}\ \sin\theta - \dot{\alpha} \cdot r = 0 \end{cases}$$

- Rueda Sueca:

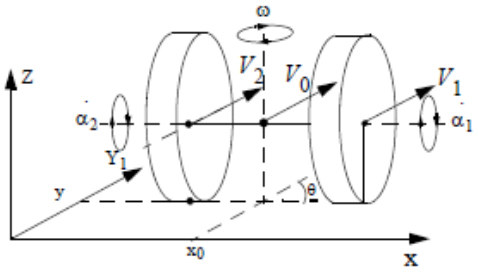


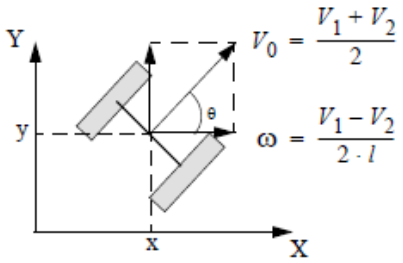
Variables de configuración = 5: P = [x Y θ α β]

Restricciones No Holónomas:

$$\begin{cases} \dot{x}\ \sin\theta - \dot{y}\ \cos\theta + l \cdot \dot{\beta} = 0 \\ \dot{x}\ \cos\theta + \dot{y}\ \sin\theta - \dot{\alpha} \cdot r = 0 \end{cases}$$

- Configuración diferencial:





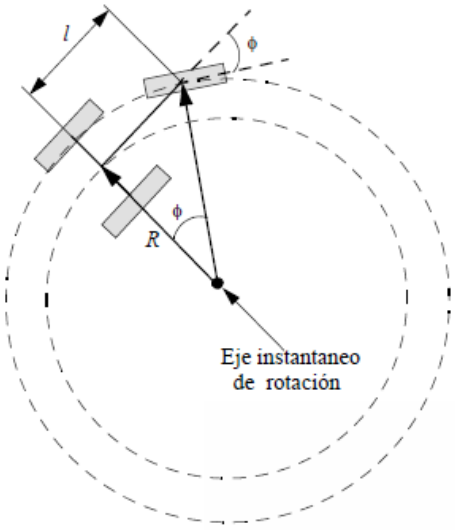
Restricciones no holónomas
(una por cada rueda y otra para el enlace, se resumen en una sola)

$$\dot{x}_0 \sin\theta - \dot{y}_0 \cos\theta = 0$$

N° D.O.F: 3-1 = 2 → 2 Grados de Libertad:
$\dot{\alpha}_1$ velocidad de rodado 1ª rueda
$\dot{\alpha}_2$ velocidad de rodado 2ª rueda

$$\begin{bmatrix}\dot{x}\\ \dot{y}\\ \dot{\theta}\end{bmatrix} = \begin{bmatrix}\cos\theta & 0\\ \sin\theta & 0\\ 0 & 1\end{bmatrix} \cdot \begin{bmatrix}V_0\\ \omega\end{bmatrix} \longrightarrow \begin{bmatrix}\dot{x}\\ \dot{y}\\ \dot{\theta}\end{bmatrix} = \begin{bmatrix}\frac{r\cos\theta}{2} & \frac{r\cos\theta}{2}\\ \frac{r\sin\theta}{2} & \frac{r\sin\theta}{2}\\ \frac{r}{2\cdot l} & -\frac{r}{2\cdot l}\end{bmatrix} \cdot \begin{bmatrix}\dot{\alpha}_1\\ \dot{\alpha}_2\end{bmatrix}$$

- Configuración de triciclos:



Modelo Simplificado A

$$\begin{bmatrix}\dot{x}\\ \dot{y}\\ \dot{\theta}\end{bmatrix} = \begin{bmatrix}\cos\theta & 0\\ \sin\theta & 0\\ 0 & 1\end{bmatrix} \cdot \begin{bmatrix}V\\ V \cdot \frac{\tan\phi}{l}\end{bmatrix}$$

Modelo Simplificado B

$$\begin{bmatrix}\dot{x}\\ \dot{y}\\ \dot{\theta}\end{bmatrix} = \begin{bmatrix}\cos\theta & 0\\ \sin\theta & 0\\ 0 & 1\end{bmatrix} \cdot \begin{bmatrix}V_d \cdot \cos\phi\\ V_d \cdot \frac{\sin\phi}{l}\end{bmatrix}$$

Modelo Completo A

$$\begin{bmatrix}\dot{x}\\ \dot{y}\\ \dot{\theta}\\ \dot{\phi}\end{bmatrix} = \begin{bmatrix}\cos\theta & 0\\ \sin\theta & 0\\ \frac{\tan\phi}{l} & 0\\ 0 & 1\end{bmatrix} \cdot \begin{bmatrix}V\\ \dot{\phi}\end{bmatrix}$$

Modelo Completo B

$$\begin{bmatrix}\dot{x}\\ \dot{y}\\ \dot{\theta}\\ \dot{\phi}\end{bmatrix} = \begin{bmatrix}\cos\theta\cos\phi & 0\\ \sin\theta\cos\phi & 0\\ \frac{\sin\phi}{l} & 0\\ 0 & 1\end{bmatrix} \cdot \begin{bmatrix}V_d\\ \dot{\phi}\end{bmatrix}$$

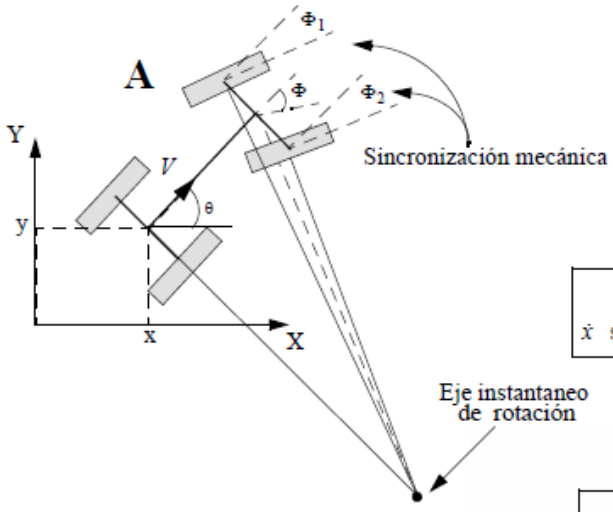
Limitación física
$|\phi| < \phi_{max}$

$$R = \frac{1}{\rho} = \frac{l}{\tan\phi}$$

$$\dot{\theta} = V \cdot \rho$$

El modelo simplificado ha sido utilizado en simulaciones de las practicas

- Configuración Ackerman: Consiste en un vehículo de cuatro que consiste en un vehículo de cuatro ruedas donde las dos ruedas delanteras definen el sentido del movimiento:



Restricciones no holónomas:
(una por cada rueda y la estructura del vehículo se resumen en dos)

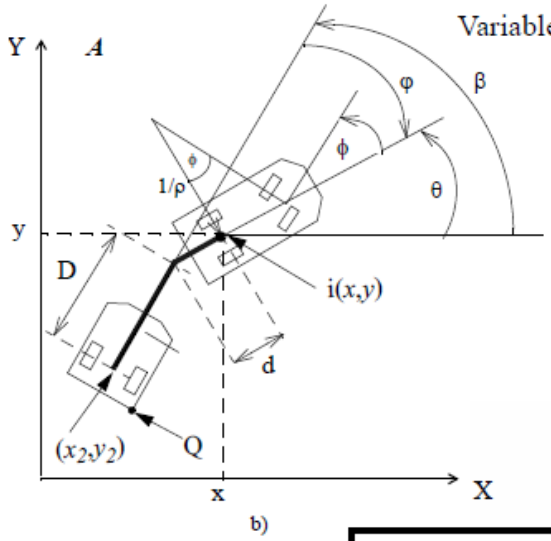
$$\dot{x}\ \sin\theta - \dot{y}\ \cos\theta = 0$$

$$\dot{x}\ \sin(\phi+\theta) - \dot{y}\cos(\phi+\theta) - \dot{\theta} l\cos(\phi) = 0$$

2 Grados de Libertad:
V velocidad de desplazamiento
ϕ ángulo de conducción

MODELO CINEMÁTICO SIMILAR AL DEL TRICICLO

- Tractor-Tráiler:



Variables de configuración: P = [x Y θ φ]

$$\begin{bmatrix} -\sin\theta & \cos\theta & 0 & 0 \\ \cos\theta\sin\varphi & \sin\theta\sin\varphi & -D-d\cos\varphi & D \end{bmatrix} \begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{\theta} \\ \dot{\varphi} \end{bmatrix} = 0$$

2 Grados de Libertad:
V velocidad de desplazamiento
ϕ ángulo de conducción

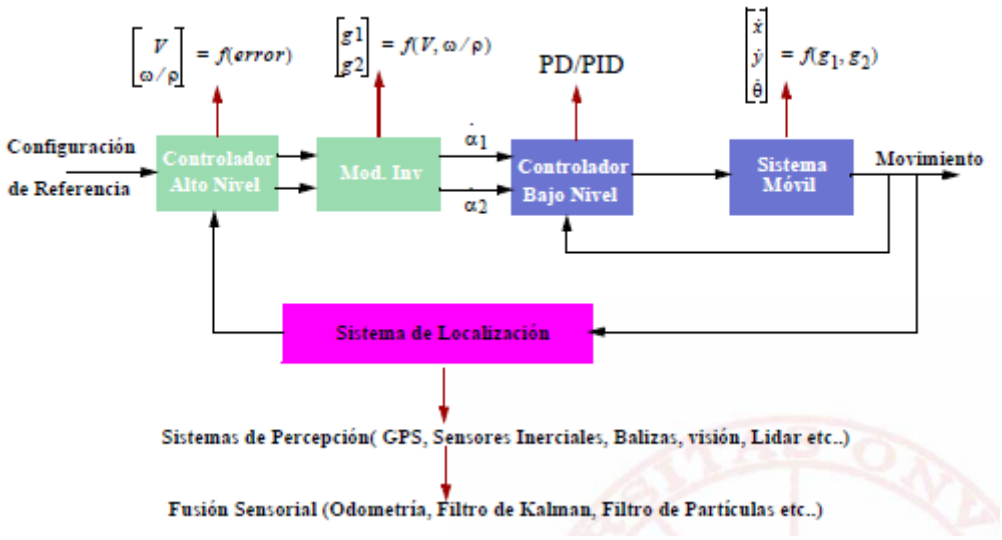
$$\begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{\theta} \\ \dot{\varphi} \end{bmatrix} = \begin{bmatrix} \cos\theta & 0 \\ \sin\theta & 0 \\ 0 & 1 \\ -\frac{\sin(\varphi)}{D} & \frac{\cos(\varphi)d}{D}+1 \end{bmatrix} \begin{bmatrix} v(t) \\ v(t)\rho(t) \end{bmatrix}$$

2.3.3. Estrategias de control

En este apartado estudiaremos las técnicas de control usadas en nuestro robot. Es un apartado que se explica mejor en el apartado práctico que en el teórico.

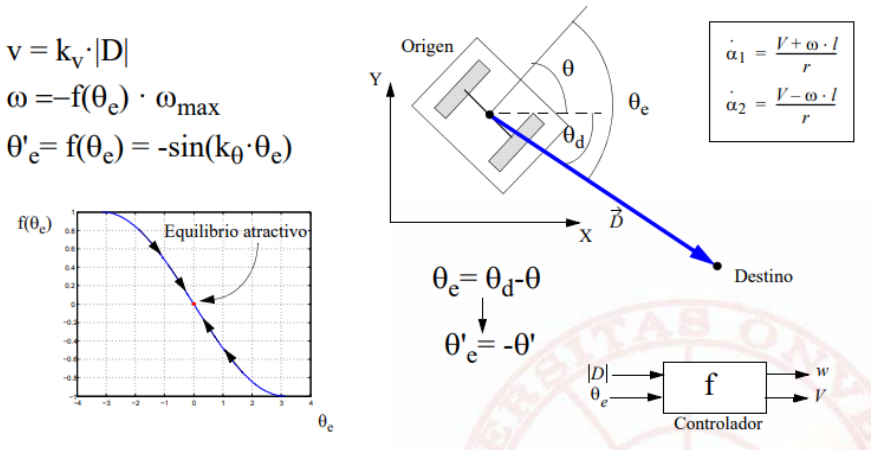
- Control Atractivo: Estrategia que hace que el robot converja hacia una configuración determinada.
- Control Repulsivo: Estrategia que hace que el robot sea repelido de algunas configuraciones.

Modelo de transformación los datos de referencia en el movimiento del robot es el siguiente:

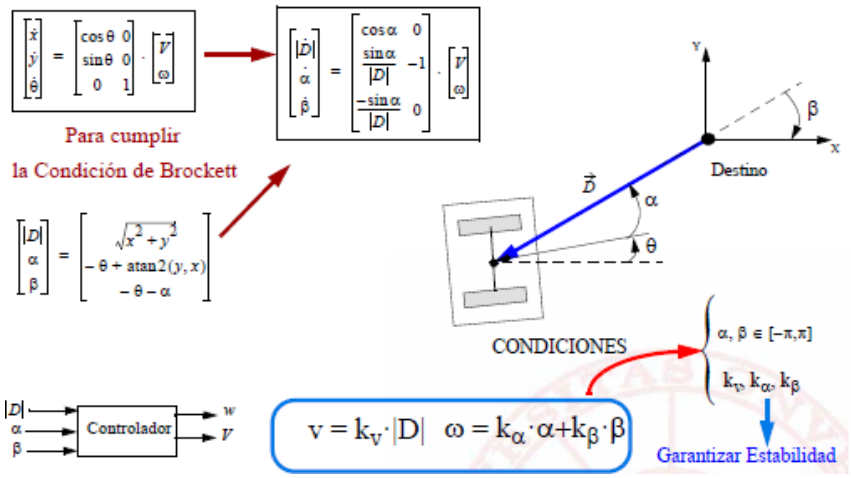


En las prácticas hemos empleado la conducción diferencial dadas las propiedades de nuestro robot, y en simulación ambas:

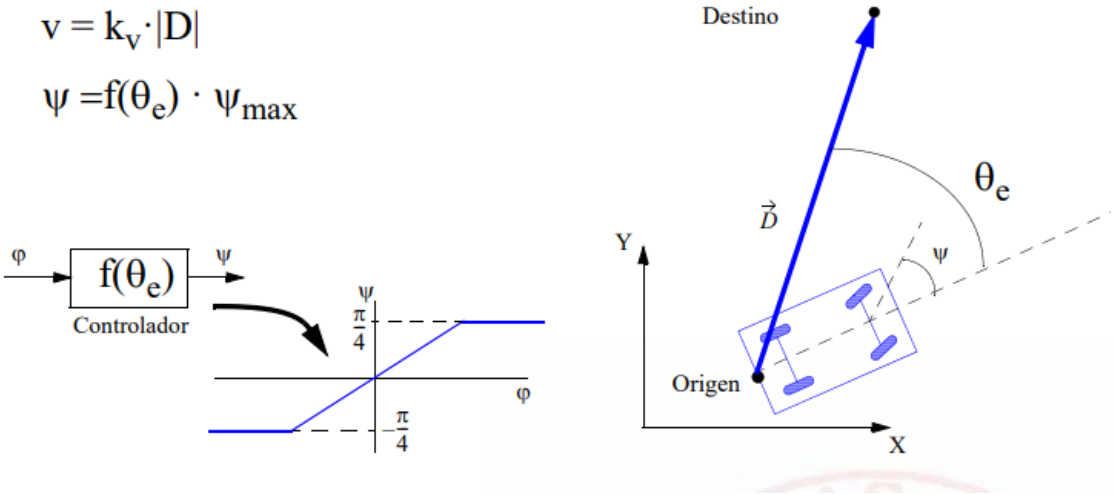
- Estrategia Diferencial-Drive



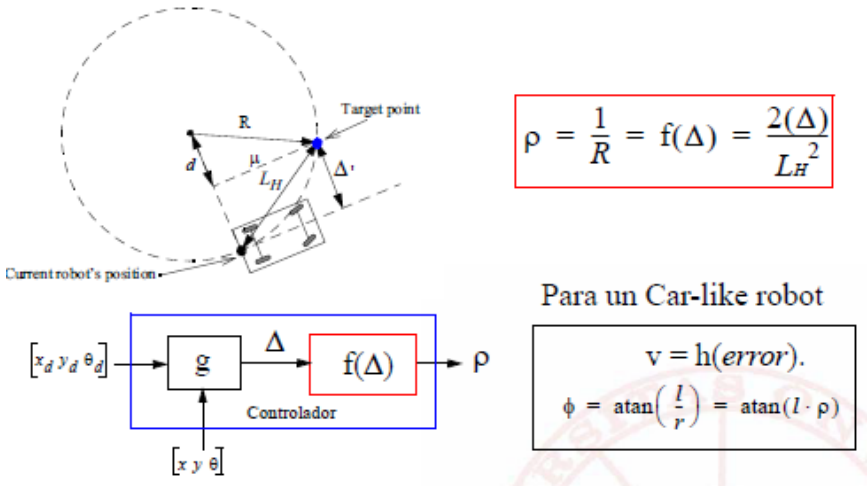
Para conseguir estabilidad al efectuar el movimiento, aplicamos el concepto de convergencia asintótica:



- Estrategia Triciclo y Ackerman



- Método Geométrico: Utilizado en las prácticas para conseguir el desplazamiento de un punto a otro obteniendo el centro de la circunferencia que contiene al robot y al punto de destino



2.3.4. Seguimiento de trayectorias

Dado un camino o trayectoria, se pretende que el robot lo siga de la forma más aproximada posible:

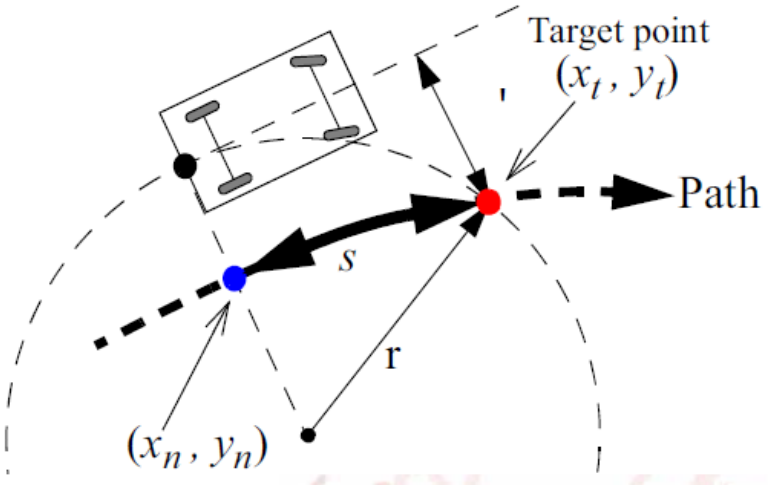
- Seguimiento de Trayectorias: Se especifica una evolución temporal tanto de la posición como de la velocidad calculando el error en un instante t teniendo en cuenta la velocidad y posición planificadas en ese instante.
- Seguimiento de Caminos: Se especifica un camino(x,y) donde para cada instante t se calcula el error en posición teniendo en cuenta la posición del robot y un punto seleccionado del camino.

- Seguimiento de Caminos

Dentro de los métodos geométricos, el más conocido y sencillo es el conocido como Persecución Pura (Pure Pursuit). Permite seguir cualquier tipo de camino donde para trayectorias suficientemente suaves permite asegurar bajo error en posición y orientación.

Pure Pursuit:

- En cada instante se obtiene el punto del camino más cercano a la posición actual del vehículo(x_n,y_n).
- El punto objetivo(x_t,y_t) se calcula escogiendo el punto del camino que dista un valor s de (x_n,y_n).
- Se aplica el algoritmo de control para converger al punto objetivo.
- Se realizan los cálculos para cada vez que se ejecute el bucle de control.



Esquema Pure Pursuit

```
Pure_pursuit () {

    definicion_camino
    valores_iniciales
    v=v0   // se mantiene constante la velocidad

    while(!fin)
       {
        posicion_actual=estimacion_posicion() // en nuestro caso odometria
        punto=punto_mas_cercano()
        punto_objetivo=punto+distancia
        rho=calcula_curvatura(posicion_actual, punto_objetivo)
        potencia=calculo_potencia( v, rho)
       actuacion(potencia) // se manda al robot los valores de control calculados
        }

}
```

Pseudocódigo Pure Pursuit

2.3.5. Algoritmos de planificación

Los algoritmos de planificación nos permiten generan un camino. En nuestro, el empleado en las prácticas ha sido el algoritmo A*. Consiste en encontrar el camino mínimo dado un punto inicial y un punto final.

Divide el mapa en diferentes nodos. Los nodos que presentan obstáculos se marcan como no disponibles. El camino se calcula recorriendo la lista de nodos buscando el camino mínimo entre el punto inicial y el punto final.

De los algoritmos propuestos en clase para cada grupo, en nuestro caso hemos elegido Campos Potenciales:

Los métodos basados en campos potenciales están basados en técnicas reactivas de navegación. El ámbito de uso de esta técnica se centra en la planificación local en entornos desconocidos, como puede ser el sorteo en tiempo real o de los obstáculos de los que no se tiene constancia.

Se considera al robot como una partícula bajo la influencia de un campo potencial artificial, cuyas variaciones modelan el espacio libre. La función potencial U en un punto “p” del espacio euclideo, se define sobre el espacio libre y consiste en la composición de un potencial atractivo Ua(p), que atrae al robot hacia la posición destino, y otra de repulsión Ur(p) que lo hace alejarse de los obstáculos. Se calcula:

$$U(p) = \ Ua(p) + \ Ur(p)$$

$$\mathrm{F(p)} = \ -\nabla U(p) = Fa(p) + Fr(p)$$

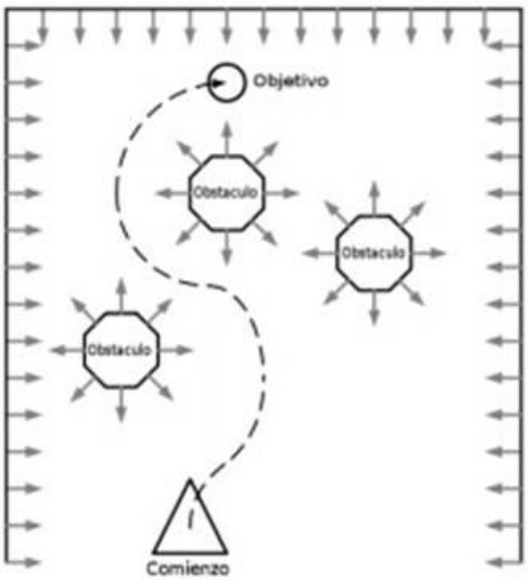
Así, la navegación basa en campos potenciales se basa en llevar a cabo la siguiente secuencia de acciones:

- Calcular el potencial U(p) que actúa sobre el vehículo en la posición actual p según la información recabada de los sensores.
- Determinar el vector fuerza artificial F(p).
- En virtud del vector calculado construir las consignas adecuadas para los actuadores del vehículo que hagan que éste se mueva según el sentido, dirección y aceleración especificadas por F(p).

El campo se construye de forma que marca el recorrido del robot. El origen se coloca en una posición elevada desde la que parte el robot y comienza el descenso por el mapa. Los obstáculos son objetos elevados en el terreno. Conforme el robot se va acercando a la meta, el potencial de atracción debe disminuir de forma que en la posición destino sea 0.

El problema en este tipo de métodos deviene en la aparición de mínimos locales,

Lugares que no son la posición destino en los cuales el potencial resulta nulo. Una posición de este tipo puede hacer que el robot quede atrapado en una posición que no sea la de destino o girar alrededor de ella. Una de las posibles soluciones es dividir el entorno en rejillas, donde cada una tiene un valor que indica su potencial. Un algoritmo de búsqueda utilizable es A*, usándose como función de coste la función potencial.



Campos potenciales generados por los diferentes objetos de una superficie

2.3.6. Introducción a la localización

La localización del robot es uno de los principales problemas que hemos podido experimentar en las prácticas ya que la posición real no coincidía con la simulada debido a los errores de posicionamiento y que generan los diferentes mecanismos del robot.

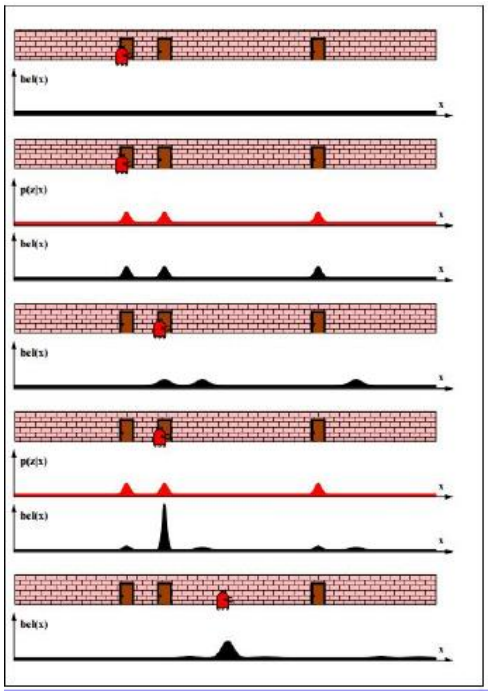
Una de las posibles soluciones es con el uso de la localización bayesiana que se apoya en las bases probabilísticas del teorema de Bayes para predecir la posible posición del robot.

Localización Bayesiana

$$P(x \mid z) = \frac{P(z \mid x) \cdot P(x)}{P(z)}$$

$$P(x \mid z) = \alpha \cdot P(z \mid x) \cdot P(x)$$

Fórmula



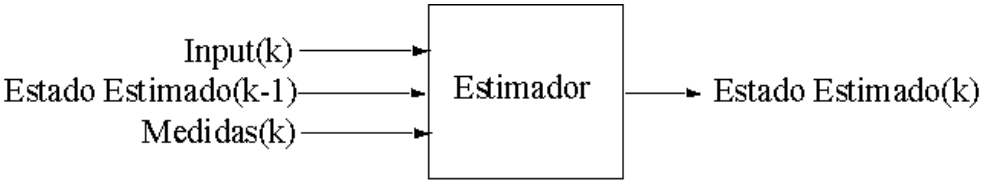
Esquema de funcionamiento

Como observamos en clase, podemos tener un control del robot a lo largo del tiempo usando los datos obtenidos del entorno de la posición. Cada vez que el robot pasa por una puerta recibimos una señal que almacenamos. Conforme va avanzando a lo largo del tiempo, la señan se va ensanchando, y volviéndose menos precisa. Cuando vuelve a pasar por una puerta, recibimos otra señal que con la almacena anteriormente generamos una mayor precisión en la posición de nuestro robot. El proceso se divide en dos partes:

- Filtrado: Consiste en obtener información más relevante sobre el estado del sistema y la incertidumbre asociada a la misma.
- Estimación: Consiste en determinar el valor más representativo de dichas variables.
  - Si el valor de las magnitudes de estos valores no varía en el tiempo se trata de un problema de estimación estática.
  - En el caso más general dichos valores evolucionarán en el tiempo y se aplicarán algoritmos de estimación dinámica.

El problema de localización en robótica se resuelve utilizando técnicas de Filtrado y Estimación. Estos algoritmos tienen que ser robustos ante la incertidumbre. Para mantener con eficiencia la incertidumbre suele adoptarse en enfoque probabilística. Siguiendo este enfoque, los tres principales algoritmos son:

- El Filtro del Histograma
- El Filtro de Kalman
- El Filtro de Partículas



Bases de Estimación Dinámica

A partir de las medidas recibidas por los sensores que tiene nuestro robot y del valor estimado en la simulación se pretende obtener la representación actual del sistema mediante la estimación del estado.

- Modelo de la Dinámica del Sistema. Permite predecir el futuro.

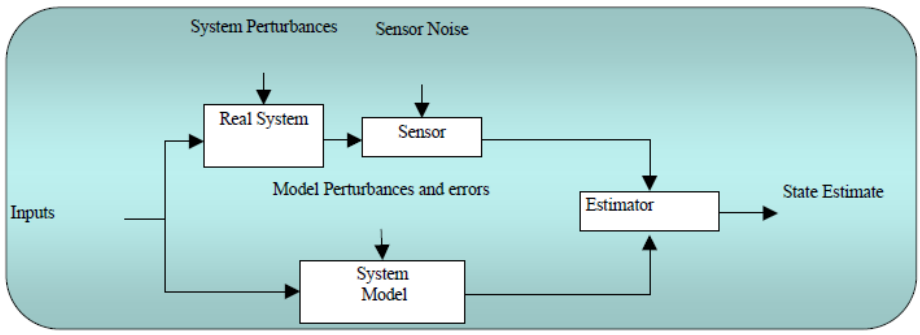
$$X_{k+1} = f(X_k, U_k, w_k)$$

- Modelo de las medidas. Permite medir el presente.

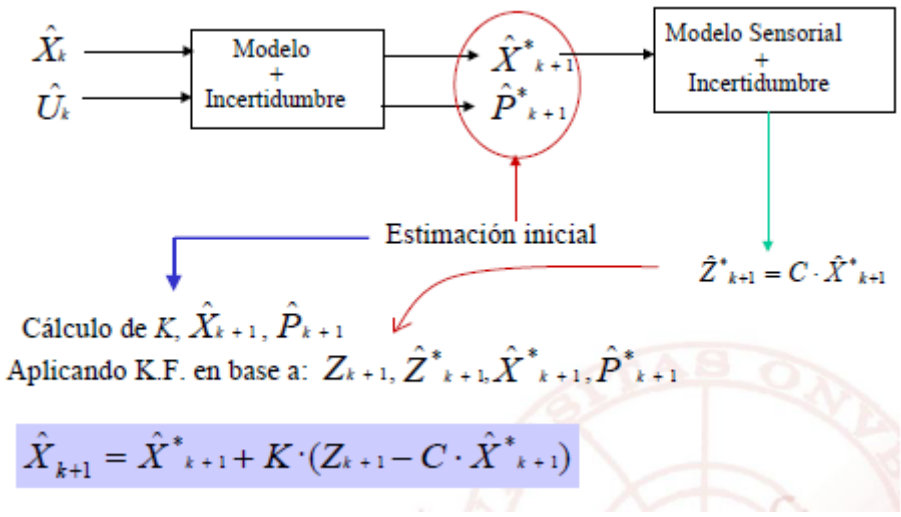
$$Z_k = h(X_k, v_k)$$

$$Z_k = [Z_1, Z_2, \dots Z_m]^T \qquad h_k = [h_1(X_k), h_2(X_k), \dots h_m(X_k)]^T$$

Fundamento de las técnicas probabilísticas



El filtro de Kalman es un método que permite estimar variables de estado no observables a partir de variables observables que pueden contener algún error de medición.



Predicción:

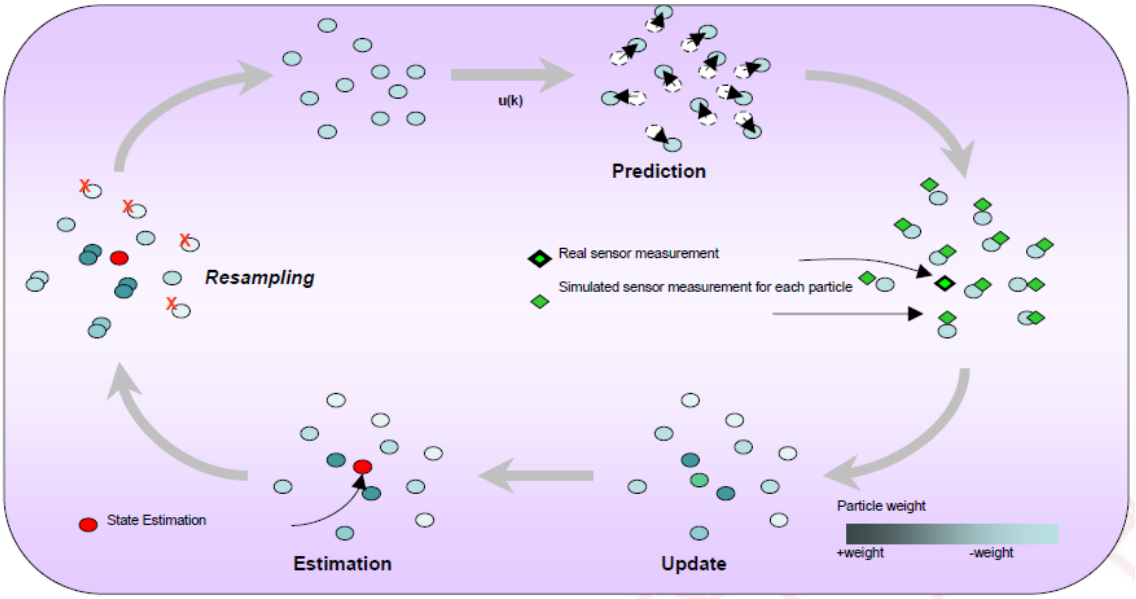
$$\hat{X}^*_{k+1} = A\hat{X}_k + BU_k$$
$$\hat{P}^*_{k+1} = A\hat{P}_k A^T + GQG^T$$

Actualización:

$$K_{k+1} = \hat{P}^*_{k+1} C^T \left(C\hat{P}^*_{k+1} C^T + R\right)^{-1} \quad \text{Ganancia}$$
$$\hat{X}_{k+1} = \hat{X}^*_{k+1} + K_{k+1}\left(Z_{k+1} - C\hat{X}^*_{k+1}\right) \quad \text{Estimado}$$
$$\hat{P}_{k+1} = \left(I - K_{k+1} * C\right)\hat{P}^*_{k+1} \quad \text{Varianza Estimada}$$

Predicción y Actualización Filtro Kalman

El filtro de partículas es un filtro que consiste en evaluar las posibles posiciones en las que puede estar el robot dado un número de partículas. En cada instante se predice la posición siguiente del robot generando el punto medio de las nubes de puntos generadas. Los puntos alejados, se descartar y generan puntos aleatorios cercanos al centroide. Estos pasos se generan de forma recursiva de manera que la posición estimada se corresponda con la real



Funcionamiento del filtro de partículas

3. Resolución Actividades

Las actividades se corresponden con las que encontramos en la página de Moodle de la asignatura, y alguna de ellas servían como introducción a las diferentes prácticas, implementando las fórmulas necesarias para la resolución de estas en un entorno de simulación controlado usando Matlab. Mi versión de Matlab es la R2021b.

3.1. Actividad 3. Simulación de modelos cinemáticos.

La primera simulación que vamos a explicar se corresponde con el ejercicio 3 expuesto en la Moodle y consiste en implementar las funciones necesarias para que el robot describa una circunferencia de un determinado radio. Esta actividad se dividido en dos partes, la primera aplicando las fórmulas a un triciclo y la otra aplicando las fórmulas a un vehículo diferencial, que son las que posteriormente usaremos con el robot real.

Para la simulación se nos da previamente un entorno controlado donde podemos simular el comportamiento del robot pasándole la velocidad que deben tener las ruedas. Nosotros hemos implementado las fórmulas expuestas en el enunciado para conseguir la velocidad necesaria para que el robot describa una circunferencia del radio que nosotros queramos.

Simulación triciclo:

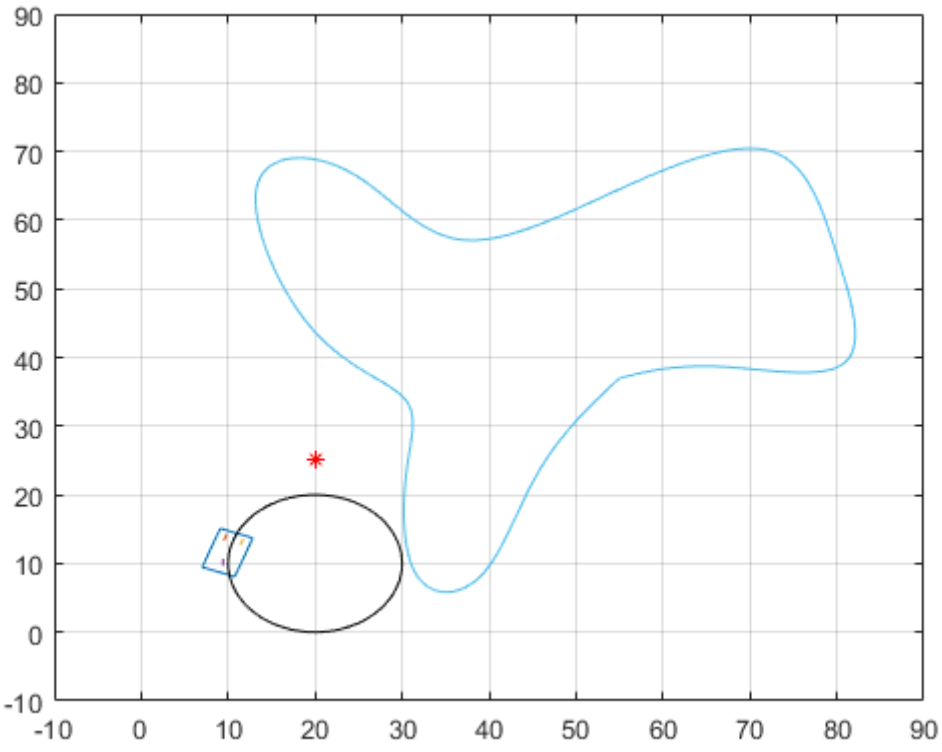
Para conseguir que el triciclo describa la circunferencia únicamente debíamos calcularle el ángulo de giro de la rueda delantera que es la que marca el movimiento.

Despejando la fórmula que nos dan y sabiendo el radio, ya estaría hecho.

$$\rho = \frac{\tan \Phi}{l}$$

Expresión dada en el enunciado

```
% MODIFICACIÓN DE PARÁMETROS PARA ESTE EJERCICIO:

% APARTADO B)
% R = 10; Dabemos obtener el ángulo para que el radio sea 10
% Sabemos que R = l / tan(theta), debemos otener theta
R = 10;
aGiro = l/R;
volante = atan(aGiro);

velocidad = 5;
conduccion=[velocidad volante];
```



Simulación del código anterior. Radio 10.

Simulación diferencial:

Para el diferencial es un poco más complicado ya que únicamente tenemos dos ruedas y para calcular el giro debemos aplicarle velocidades diferente a cada una de ellas. Esto lo conseguimos despejando las fórmulas dadas en el enunciado.

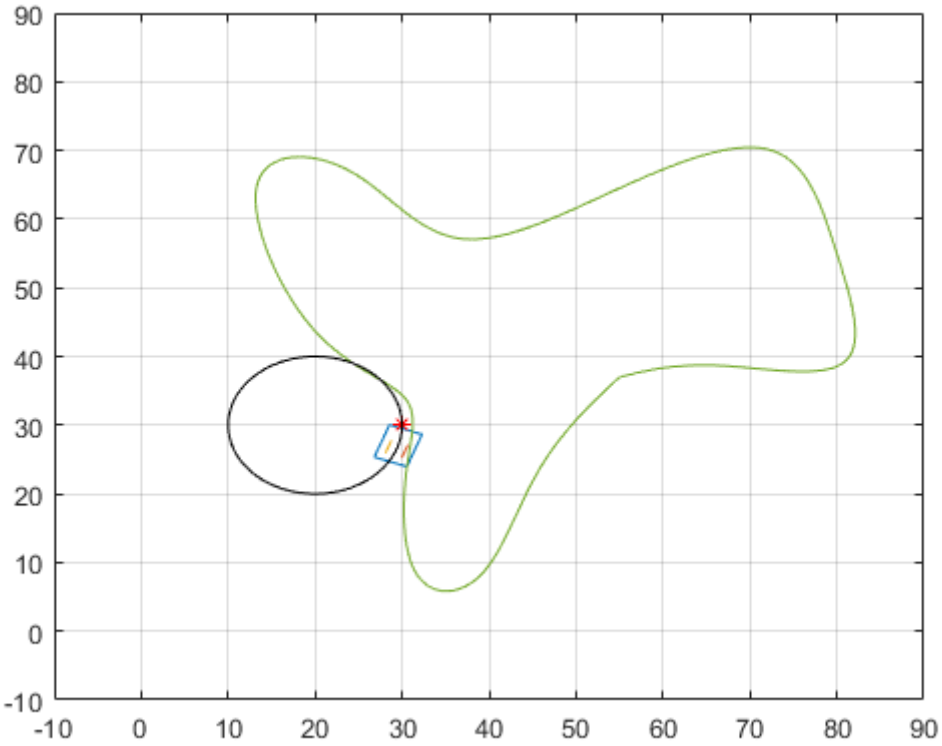
$$V_0 = \frac{(\dot{\alpha}_1 + \dot{\alpha}_2)}{2} \cdot r \qquad \omega = V_0 \cdot \rho \qquad \omega = \frac{(\dot{\alpha}_1 - \dot{\alpha}_2)}{2 \cdot l} \cdot r$$

Fórmulas dadas en el enunciado de la práctica

Debemos despejar las alphas que corresponden con las velocidades de las ruedas, en el código viene perfectamente explicado esta conversión y la forma de conseguir estas velocidades dependiendo del radio que queramos describir.

```
% Para el ejercicio 2 debemos obtener que velocidades deben tener las
    % ruedas para que el radio que describa sea de 10, para ello debemos
    % partir de las expresiones que tenemos en los apuntes.

% DATOS:
    Vo = 50; % Velocidad lineal
    R=10; % Radio
    p=1/R; % Curvatura
    w = Vo*p;

    velocidad_derecha = ((w*l + Vo)/R);
    velocidad_izquierda = ((Vo-w*l)/R);

    conduccion=[velocidad_derecha velocidad_izquierda];
```

Para el cálculo de estas fórmulas debemos configurar una velocidad lineal y el radio para posteriormente calcular la curvatura y aplicar la fórmula despejada para cada una de las ruedas. El resultado en la simulación es el siguiente:



Simulación del vehículo diferencial para radio 10.

3.2. Actividad 4. Simulación control geométrico.

Esta actividad consiste en aplicar una serie de fórmulas para calcular el recorrido del vehículo hacia un punto que nosotros decidamos en el espacio. Programaremos un algoritmo de control geométrico sobre nuestro vehículo diferencial que le permitirá colocarse en el punto que hemos mencionado anteriormente.

Aplicaremos los conceptos vistos en clase calculando la curvatura necesaria para que el robot llegue al punto destino.

Las fórmulas dadas en el enunciado y en las transparencias sirven para el cálculo de la delta, la distancia que separa el robot del punto y la curvatura:

$$\Delta = (x_0 - x_f)\cdot \sin\theta_0 - (y_0 - y_f)\cdot \cos\theta_0 ;$$

$$L_H = \sqrt{(x_0 - x_f)^2 + (y_0 - y_f)^2} ;$$

$$\rho = \frac{2\cdot\Delta}{L_H^2} .$$

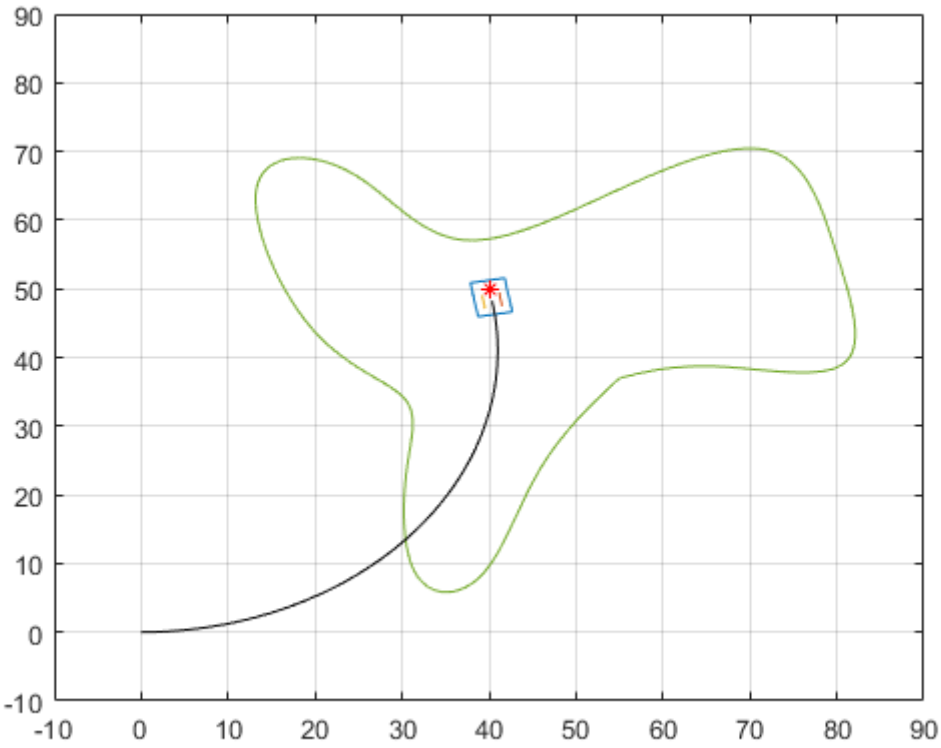
Fórmulas dadas en el enunciado

Los cambios realizados se representan en el código implementado en la simulación:

```
% APARTADO B. RECORRIDO HASTA UN PUNTO EN CONCRETO.
theta = pose(3,k);
delta = ((pose(1,k) - punto(1)) * sin(theta)) - ((pose(2,k) - punto(2)) *
cos(theta));
L = sqrt((punto(1) - pose(1,k))^2 + (punto(2) - pose(2,k))^2);
rho = (2*delta) / L^2;
% La velocidad depende de L para que se pare en el punto. La saturamos
% un poco
Vo = L *0.25;

velocidad_derecha = (Vo / radio_rueda)*(1 + rho * l);
velocidad_izquierda = (Vo / radio_rueda)*(1 - rho * l);

conduccion=[velocidad_derecha velocidad_izquierda];
```

Como podemos ver, calculamos la velocidad dependiente de la distancia que quede hasta llegar al punto en un momento en determinado multiplicado por una constante. Esta modificación hace que el robot vaya decrementando la velocidad cuanto más cerca este del destino.

La posición del robot va actualizándose con los valores que devuelve kuta_diferencial, que es una función compilada y facilitada por el profesor Fernando Gómez Brvo.



Simulación de la práctica 4. (0,0) -> (40,50)

3.3. Actividad 5. Introducción al Path Following.

En esta actividad partimos de las actividades implementadas anteriormente y ahora debemos conseguir que el robot siga el camino expuesto en mapa.

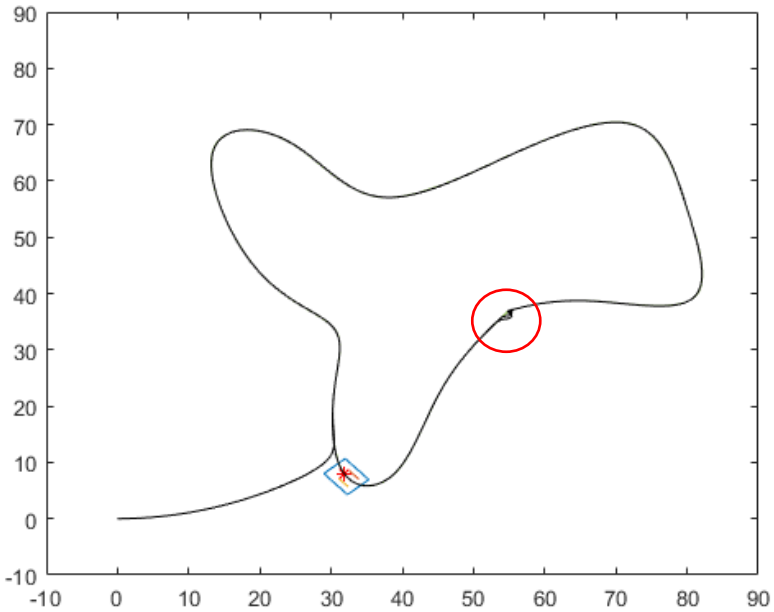
Para implementar esta actividad debíamos conseguir que al final de la práctica devolviera el punto más cercano perteneciente al camino del robot. Este punto, lo vamos a obtener con la fórmula `punto=[camino(orden_minimo+indice_siguiente, 1) camino(orden_minimo+indice_siguiente, 2)]` y la función `mínima_distancia`, función se debe aplicar dentro del bucle devolviendo como punto final un punto a una distancia D pasada por parámetro del robot.

El script que implementamos es *simulación_diferencial_actividad5, que a*vanza n pasos dentro de la matriz del camino devolviendo el punto resultante como destino de nuestro robot. Teniendo el punto inicial (posición actual del robot) y el punto destino calculamos los valores de velocidad.

```
while (t0+h*k) < tf,
    %actualización
    k=k+1;

    %Nos da el punto del camino más cercano
    orden_minimo = minima_distancia(camino, pose(:,k));

    indice_siguiente = 15;

    punto=[camino(orden_minimo+indice_siguiente, 1)
camino(orden_minimo+indice_siguiente, 2)];

    %Para evitar que el robot tambalee, es el valor de delta el que lo
    %produce. Hay que darle un valor mayor
    delta = ((pose(1,k)-punto(1))*sin(pose(3,k))-(pose(2,k)-
punto(2))*cos(pose(3,k)));
    LH = sqrt((pose(1,k)-punto(1))^2+(pose(2,k)-punto(2))^2);
    rho = 2*delta/LH^2; %p en la diapositiva de la actividad

    %Para la actividad 5, le damos una velocidad CONSTANTE
    %En este caso, le damos valor 5
    velocidad = 5;
    velocidad_derecha = velocidad*(1+l*rho)/radio_rueda;
    velocidad_izquierda = velocidad*(1-l*rho)/radio_rueda;

    conduccion=[velocidad_derecha velocidad_izquierda];

    %metodo de integración ruge-kuta

    pose(:,k+1)=kuta_diferencial(t(k),pose(:,k),h,conduccion);

end
```

Para evitar inestabilidad, aumentamos el valor de indice_siguiente. Así, podemos saltar un poco más al siguiente punto de la curva cerrada y evitamos tener problemas.

La simulación tal y como está el código funciona bien describiendo el camino de forma correcta sin cortes ni errores.



3.4. Actividad 6. Definición de Caminos con Splines.

En esta actividad, trabajamos la forma que deben tener los caminos para que el robot lo recorra de la forma más correcta posible.

Como hemos estudiado en la teoría el camino debe seguir una trayectoria suave, esto supone que sea continua en el tiempo, continua en la primera derivada y continua en la segunda derivada. Para asegurarnos estas restricciones en nuestro camino usaremos *función_spline_cubica_varios_puntos,* a la que le pasamos un vector con los puntos que forman nuestro camino y este genera el camino en un formato tratable para el robot.

En la actividad, también vamos a implementar los puntos de despegue y los puntos de aterrizaje para que conseguir que el robot parta con la posición que le pasemos nosotros por parámetro. Para calcularlo le pasamos una distancia arbitraria a la que le calculamos el seno o el coseno del ángulo que queramos como inicial o como final, multiplicado por la distancia de separación con respecto al punto inicial y final.

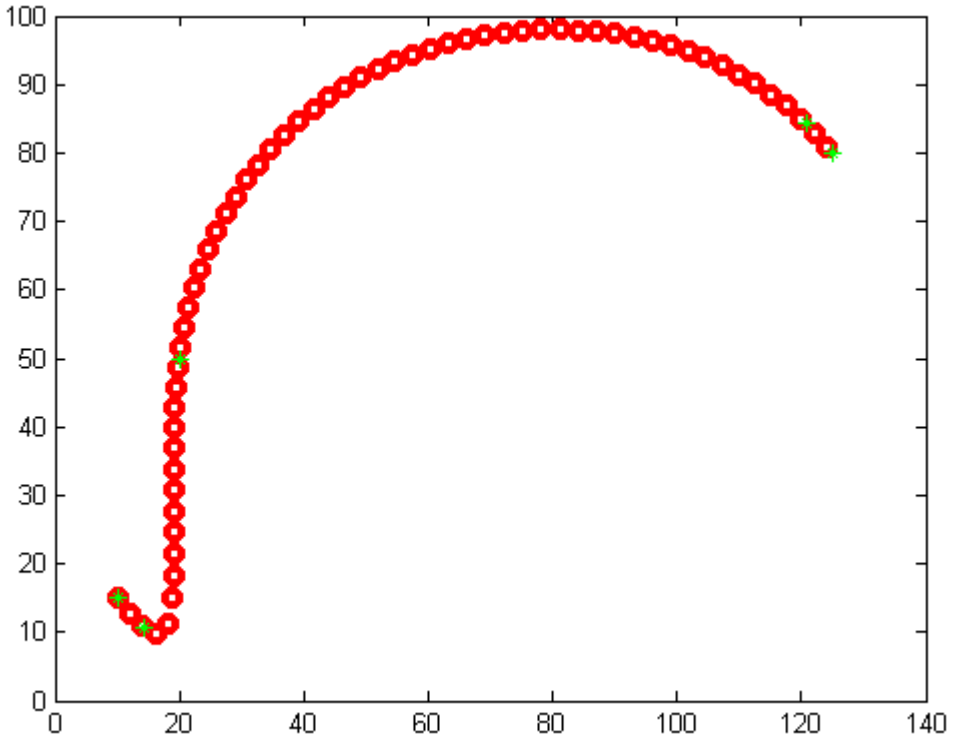
A continuación, se muestra el código con ambas actividades pedidas en el enunciado.

```
% Ejercicio 2. Actividad 6 --> para el seguimiento del camino
% Datos del problema:
% Inicial = (X0,Y0,theta 0)= (10,15,-pi/4)
% Final = (Xf,Yf,theta f)= (80,80,-pi/4).

%Prueba de otras configuraciones añadiendo algún punto más
X = [10 40 80];
Y = [15 50 80];
```

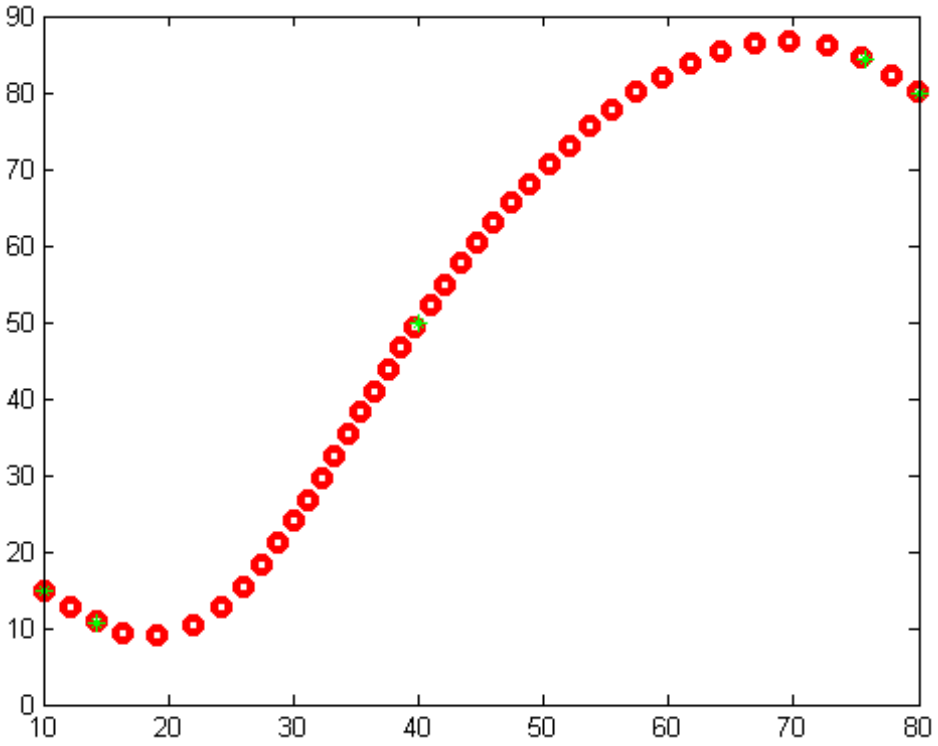
```
% Necesitamos dos puntos ficticios para conseguir en la llegada y en la
% salida los theta que nos interesen a nosotros.
thetaini = -pi/4;
thetafin = -pi/4;

% Punto de despegue
dd = 6; % Normalmente el tamaño del robot
Pdx = X(1) + dd*cos(thetaini) ;
Pdy = Y(1) + dd*sin(thetaini);

% Punto de aterrizaje
da = 6;
[f c] = size(X);
Pax = X(end) - da*cos(thetafin) ;
Pay = Y(end) - da*sin(thetafin);

Xf = [X(1),Pdx,X(2:c-1),Pax ,X(c)];
Yf = [Y(1),Pdy,Y(2:c-1),Pay ,Y(c)];
```

Al aplicar estos datos, generamos las siguientes curvas:

- Curva con los datos facilitados en el ejercicio 2 de la actividad 6

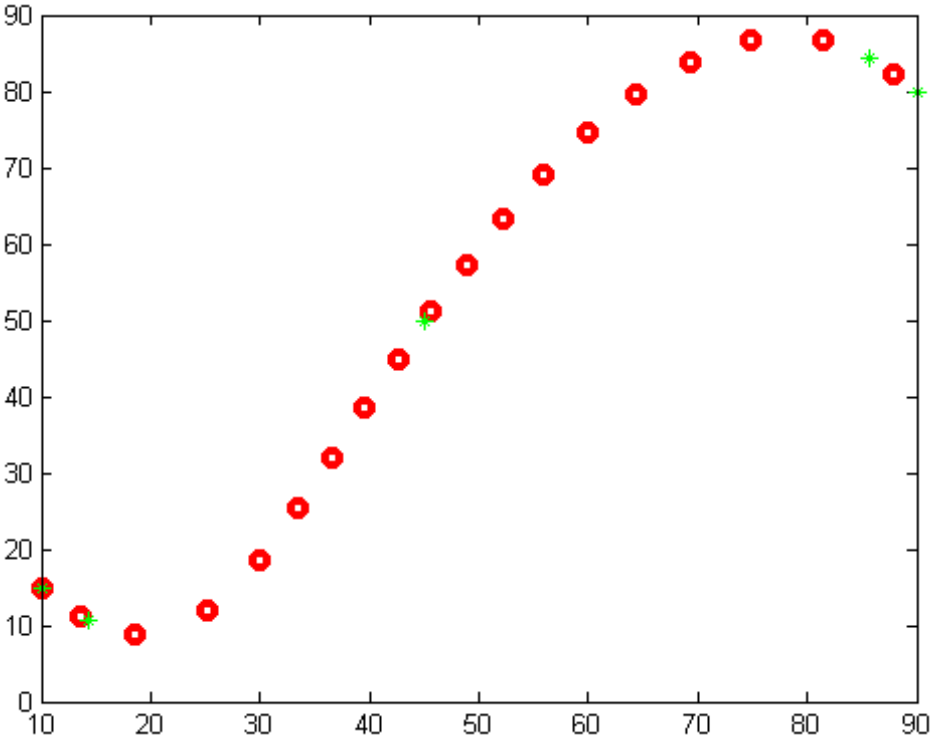


- Curva añadiendo un punto adicional al camino con d = 3



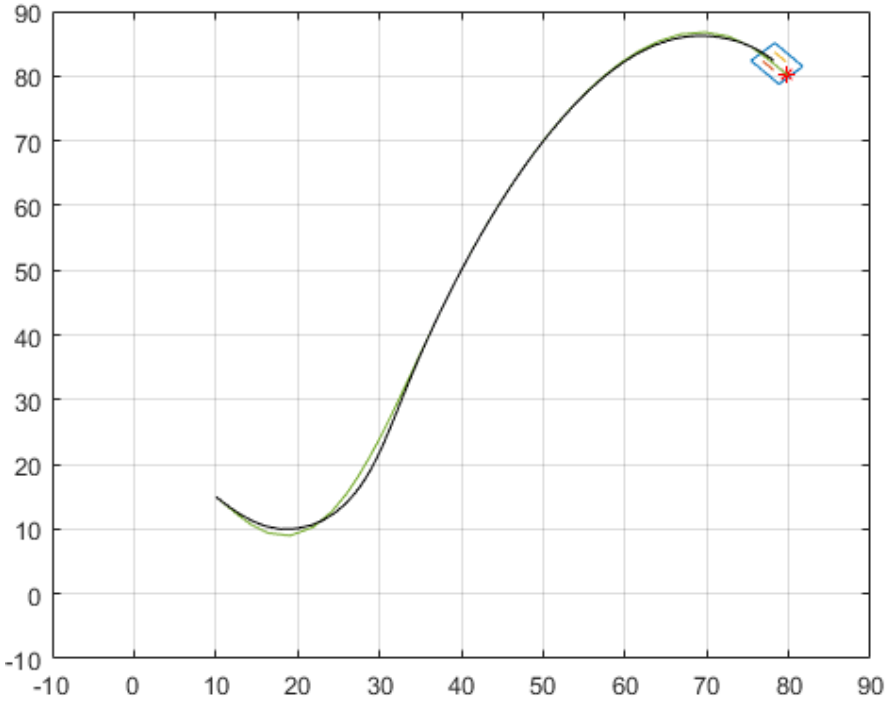
Camino X= [10,40,80]; Y = [15,50,80]; d = 3, siendo d la distancia en cm entre punto y punto

- Curva con d = 7



Camino X= [10,45,80]; Y = [15,50,80]; d = 7

Este camino se puede pasar a la ejecución de la simulación de la actividad 5, en el archivo *simulación_diferencial*, haciendo que el robot recorra el camino pasado por parámetro. Tan solo debemos de copiar el código anteriormente mencionado, e implementarlo en el archivo mencionado.



Robot recorriendo un camino creado por nosotros.

3.5. Actividad 7. Planificando Rutas con A*.

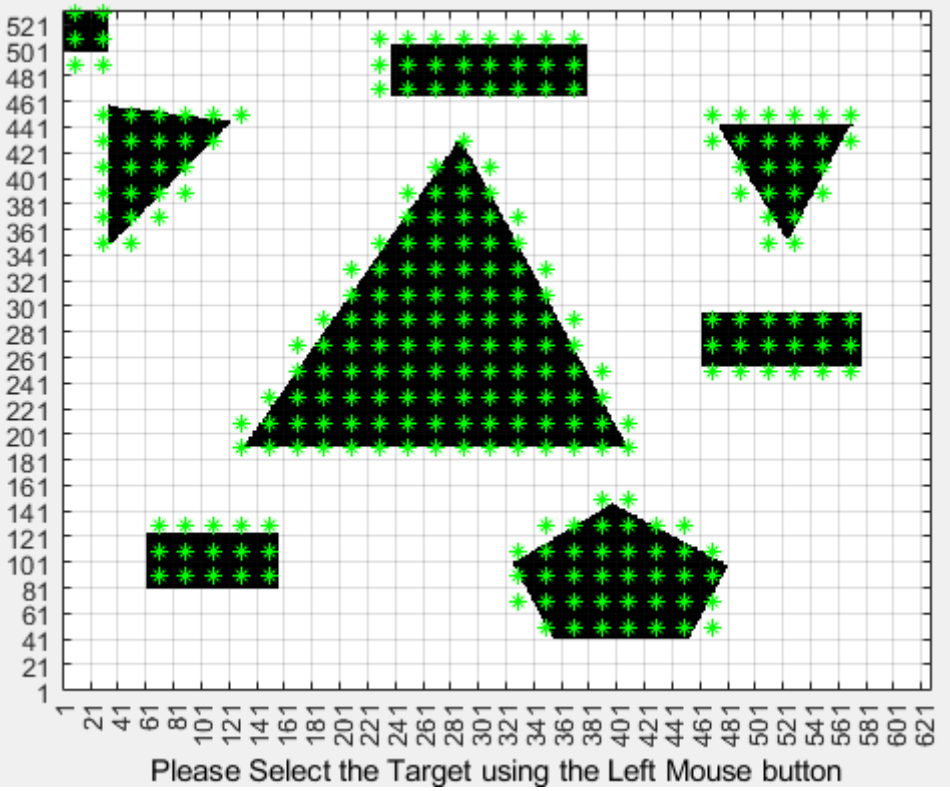
Una vez que sabemos cómo funciona perfectamente la generación de caminos, vamos a usar el algoritmo A* para calcular el camino mínimo dentro de un mapa. Este mapa debemos tratarlo para que este en las coordenadas del robot, ya que las imágenes en Matlab vienen con los ejes expuestos desde la esquina superior izquierda y nuestro robot los tiene en la esquina inferior izquierda.

Al algoritmo A* inicialmente le pasamos el mapa sobre el que vamos a aplicar el algoritmo y este lo divide en diferentes nodos según número que le pasamos por parámetro a la función. Marcamos un punto inicial y un punto final dentro del mapa y se genera automáticamente un camino al que le aplicaremos la función spline para que el camino tenga una trayectoria suave. Veamos su código:

```
l=3.5; %semidistancia entre rudas delanteras y traseras, tambien definido en
modelo
radio_rueda=1;

%Carga el fichero  BMP
MAPA = imread('cuadro4.bmp');

%Transformación para colocar correctamente el origen del Sistema de
```

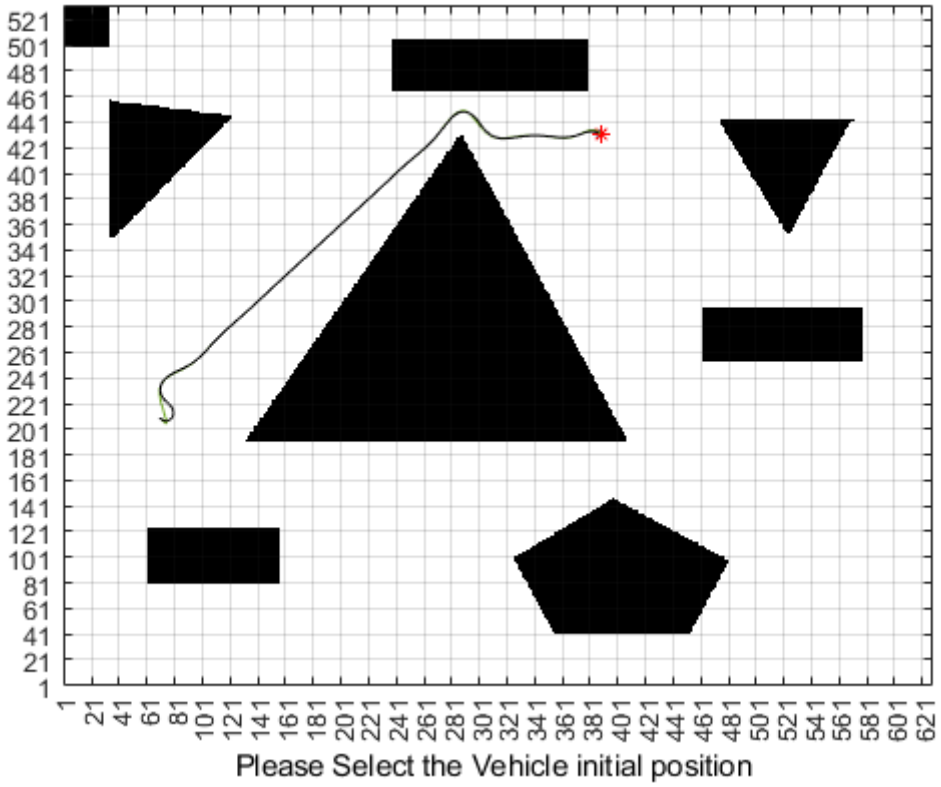
```
%Referencia
MAPA(1:end,:,:)=MAPA(end:-1:1,:,:);

%Tamaño de las celdas del grid
delta=20;
%genera la ruta óptima
Optimal_path=A_estrella(MAPA, delta);

%Condiciones iniciales
pose0=[Optimal_path(1,1); Optimal_path(1,2); pi/2];
posef=[Optimal_path(end,1); Optimal_path(end,2); 3*pi/2];

%definir camino
dd=5;
da=dd;

posicion_despegue=[pose0(1)+(dd*cos(pose0(3))) pose0(2)+(dd*sin(pose0(3)))];
posicion_aterriza=[posef(1)-(da*cos(posef(3))) posef(2)-(da*sin(posef(3)))];

xc=[pose0(1) posicion_despegue(1) Optimal_path(2:end-1,1)'
posicion_aterriza(1) posef(1)];
yc=[pose0(2) posicion_despegue(2) Optimal_path(2:end-1,2)'
posicion_aterriza(2) posef(2)];

ds=1; %distancia entre puntos en cm.
camino=funcion_spline_cubica_varios_puntos(xc,yc,ds)';

%-------------------------------------------------------------------------

t0=0;

%final de la simulación
tf=100;

%paso de integracion
h=0.1;
%vector tiempo
t=0:h:tf;
%indice de la matriz
k=0;

%inicialización valores iniciales
pose(:,k+1)=pose0;

t(k+1)=t0;

while (t0+h*k) < tf,

    %actualización
    k=k+1;

       %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

       %punto más cercano
       orden_minimo= minima_distancia(camino, pose(1:2,k));


       Look_ahead=20;
       seguir=orden_minimo+Look_ahead;
```

```
        if(orden_minimo+Look_ahead>length(camino))
              seguir=length(camino);
        end
        punto=[camino(seguir,1), camino(seguir, 2)];

    delta = (pose(1,k)-punto(1))*sin(pose(3,k))-(pose(2,k)-
punto(2))*cos(pose(3,k));
    LH=sqrt((pose(1,k)-punto(1))^2 + (pose(2,k)-punto(2))^2);
    rho=2*delta/LH^2;

    Distancia_al_final=sqrt((pose(1,k)-camino(end,1))^2 + (pose(2,k)-
camino(end,2))^2);

    V0=1*LH;
    if(V0>50)
        V0=50;
    end

    W=V0*rho;
    velocidad_derecha=(1/radio_rueda)*(V0+W*l);
    velocidad_izquierda=(1/radio_rueda)*(V0-W*l);

    conduccion=[velocidad_derecha velocidad_izquierda];

      %metodo de integración ruge-kuta

      pose(:,k+1)=kuta_diferencial_mapa(t(k),pose(:,k),h,conduccion,MAPA);

end
```



Algoritmo A Ejecutado sobre el mapa de pruebas.*

Si el camino generado y tratado lo fusionamos con los resultados de la actividad 5, hacemos que el robot siga el camino de forma precisa.



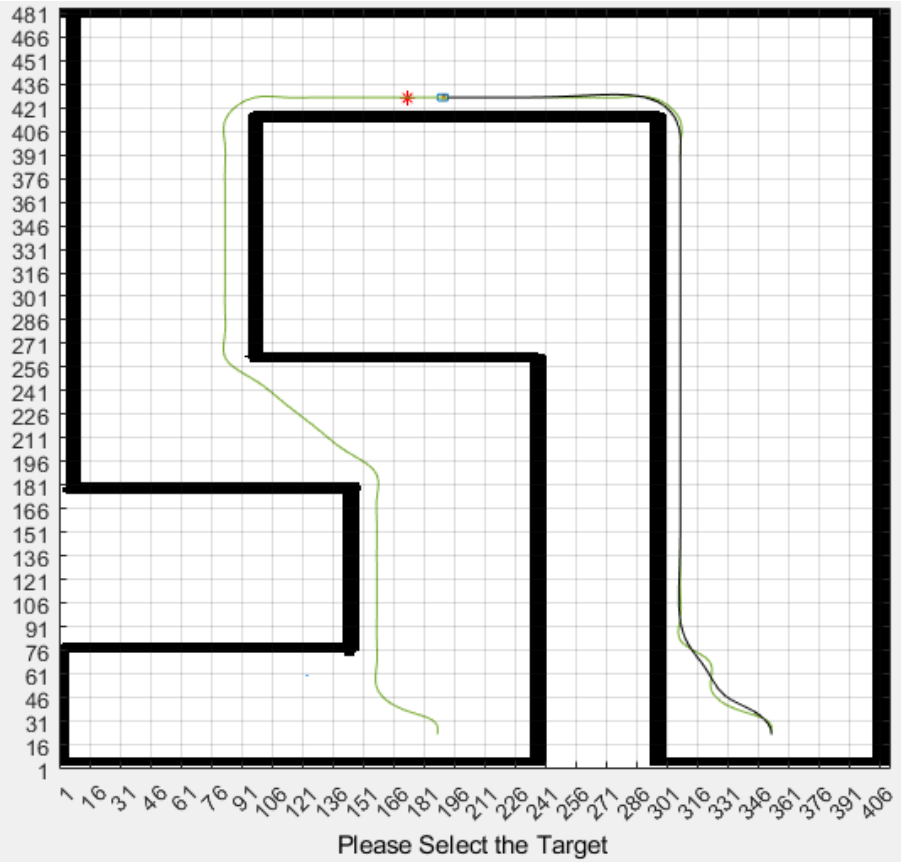
*Seguimiento del camino generado por el algoritmo A**

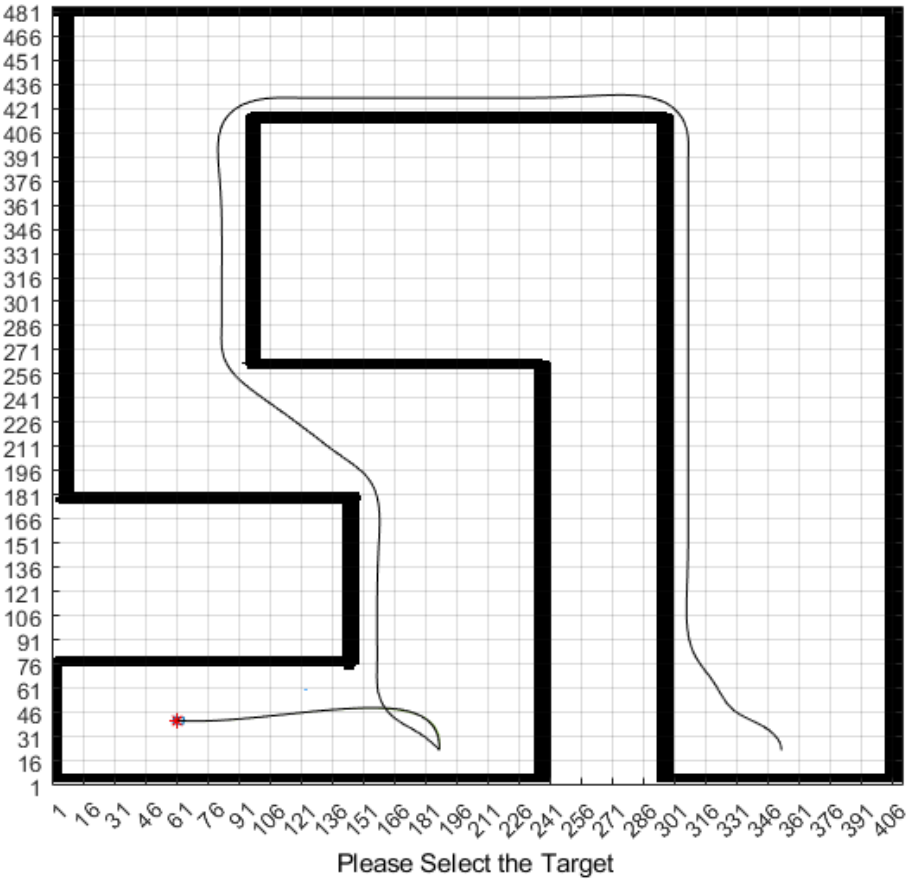
A continuación, se expone un nuevo mapa de ejemplo para que el robot llegue a una zona y luego aparque. He puesto delta a 15 en este apartado para una mejor visualización.

```
%PROGRAMACIÓN APARCAMIENTO EL APARCAMIENTO
%Genero el camino del aparcamiento
Optimal_path2=[posef(1) posef(2);posicion_aterriza(1) posicion_aterriza(2);61
41];
camino=funcion_spline_cubica_varios_puntos(Optimal_path2(:,1)',Optimal_path2(
:,2)',ds)';

%ahora, la posición inicial de este bucle es la final del anterior
pose0=[posef(1); posef(2); posef(3)];


%final de la simulación
tf=tf+30;

%vector tiempo
t=0:h:tf;
```

```
%inicialización valores iniciales
pose(:,k+1)=pose0;

t(k+1)=t0;

while (t0+h*k) < tf,

    %actualización
    k=k+1;

      %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

      %punto más cercano
      orden_minimo= minima_distancia(camino, pose(1:2,k));

      %10 para que sea más preciso a la hora de dar marcha atrás y aparcar
      Look_ahead=10;
      seguir=orden_minimo+Look_ahead;
      if(orden_minimo+Look_ahead>length(camino))
            seguir=length(camino);
      end
      punto=[camino(seguir,1), camino(seguir, 2)];

    delta = (pose(1,k)-punto(1))*sin(pose(3,k))-(pose(2,k)-
punto(2))*cos(pose(3,k));
    LH=sqrt((pose(1,k)-punto(1))^2 + (pose(2,k)-punto(2))^2);
    rho=2*delta/LH^2;

    Distancia_al_final=sqrt((pose(1,k)-camino(end,1))^2 + (pose(2,k)-
camino(end,2))^2);

    V0=1*LH;
    if(V0>20)
        V0=20;
    end

    W=V0*rho;
    velocidad_derecha=(1/radio_rueda)*(V0+W*l);
    velocidad_izquierda=(1/radio_rueda)*(V0-W*l);

      %velocidad con signo negativo para que vaya hacia atrás
    conduccion=[-velocidad_derecha -velocidad_izquierda];

      %metodo de integración ruge-kuta

      pose(:,k+1)=kuta_diferencial_mapa(t(k),pose(:,k),h,conduccion,MAPA);
end
```

Este es su resultado:





3.6. Actividad 8. Introducción al Control reactivo

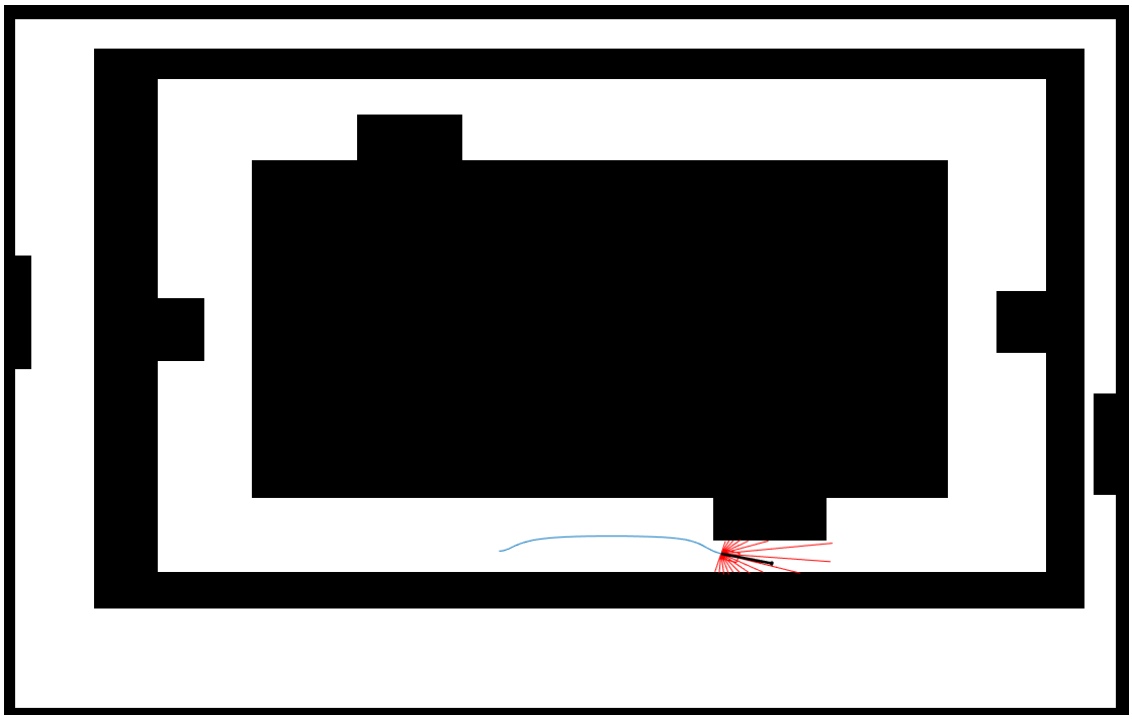
Esta última actividad trata sobre un robot que tiene un sensor LIDAR. Aquí solo me centraré en la modificación que hay que realizar, la cual trata de reducir su velocidad si encuentra en su radio de alcance algún obstáculo, y esquivarlo. Veamos la implementación en código dentro del bucle de simulación:

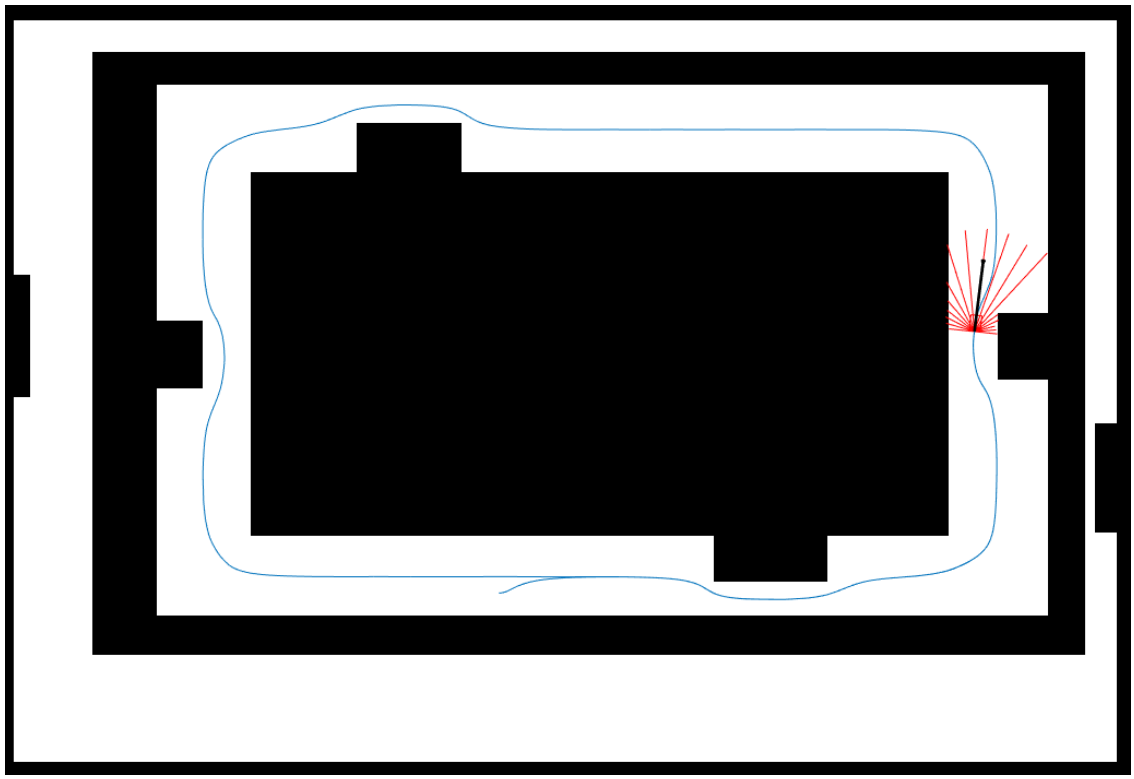
```
%Esto sirve para calcular los angulos de ruta del LIDAR
xt = laser(:, 1).*cos(laser(:, 2))
yt = laser(:, 1).*sin(laser(:, 2))

%Esto sirve para el vector representación de nuestro robot
componentes_x=sum(xt);
componentes_y=sum(yt);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Aquí hay que definir el controlador
%-------------------------------------
velocidad=2*norm([componentes_x componentes_y]);
volante=5*atan(componentes_y/componentes_x);
%-------------------------------------


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%Limitación del ángulo de conducción

if volante>pi/4 volante=pi/4;
end

if volante<-pi/4 volante=-pi/4;
end
```

Lo que hemos hecho es que la componente x del vector sea igual a la sumatoria de las componentes x de cada de uno de los vectores. Lo mismo ocurre con la componente y.

La forma más sencilla de implementar esto en Matlab es creando una matriz que tenga el seno y otra que tenga el coseno de los ángulos y multiplicándolo por el LIDAR.

Ya una vez que tenemos definidas las componentes, solo falta añadirlas al controlador con a velocidad y el volante.





3.7. Aplicación Actividad Final. Tarea Robótica de Navegación

La actividad final **se basa en el código de la actividad 7**, pero con modificaciones en el algoritmo **A*,** para que parta de un punto inicial siempre establecido luego que vaya al destino y vuelva, y todo esto para poder conseguir el objetivo de la aplicación en el mundo real.

Mi aplicación consiste en un robot autónomo que se encuentra en la cocina de un restaurante y que se dedica a llevar los platos que los empleados de cocina ponen en el mismo, para llevarlos a la mesa deseada. A su vez, el robot está preparado para esperar un tiempo determinado en la mesa para recoger todos los platos y además, por si algún comensal quiere dejar uno o varios platos en el robot, y luego que este lo lleve a la cocina para su limpieza. Dejando platos o no, el robot volverá al punto Start en la posición [41 540], que es el punto donde se encuentra en la cocina.

El robot, tiene un movimiento inicial Splite, para evitar que otro robot, en caso de que lo hubiese, choquen. Cuando vuelve a la posición inicial, viene desde un punto inferior para que así no choque con otro robot que salga.

Las modificaciones que hemos hecho para que el robot salga de una posición en concreta han sido en la función ***A_estrella***.

Hemos comentado la línea y el bucle que pide la posición inicial y que espera a por el clic en la pantalla, y hemos añadido manualmente los puntos xvlal = 41; y yval = 540; que será nuestro punto de partida. A continuación, dejo el código.

```
%The initial position

% ************* MODIFICACIÓN PARA LA ACTIVIDAD FINAL *************
% xlabel('Please Select the Vehicle initial position using the Left Mouse
button','Color','black');
% but=0;
% while (but ~= 1) %Repeat until the Left button is not clicked
%     [xval,yval,but]=ginput(1);
% end

%Ponemos como punto inicial las siguientes coordenadas para que siempre
%salga de este punto, es decir a donde se encuentra el robot en la cocina.
xval = 41;
yval = 540;
```

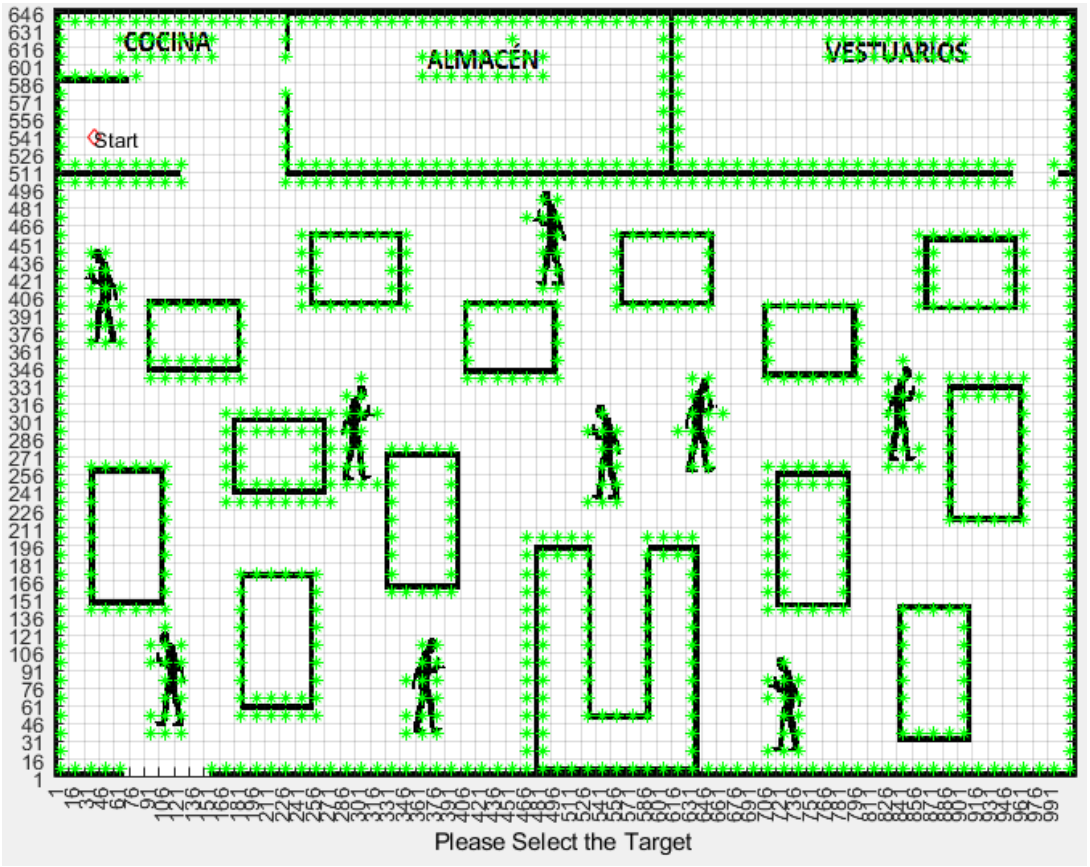
En el script *simulación_diferencial,* encontramos la implementación de mi robot. Como he dicho anteriormente, es una copia de la actividad 7, pero la diferencia es que mientras que en la actividad 7, aparcaba, aquí vuelve al punto inicial. Para conseguir esto, he hecho un cambio en el camino, con la funcion ***flip*** de Matlab. La implementación de todo mi programa quedaría de la siguiente manera:

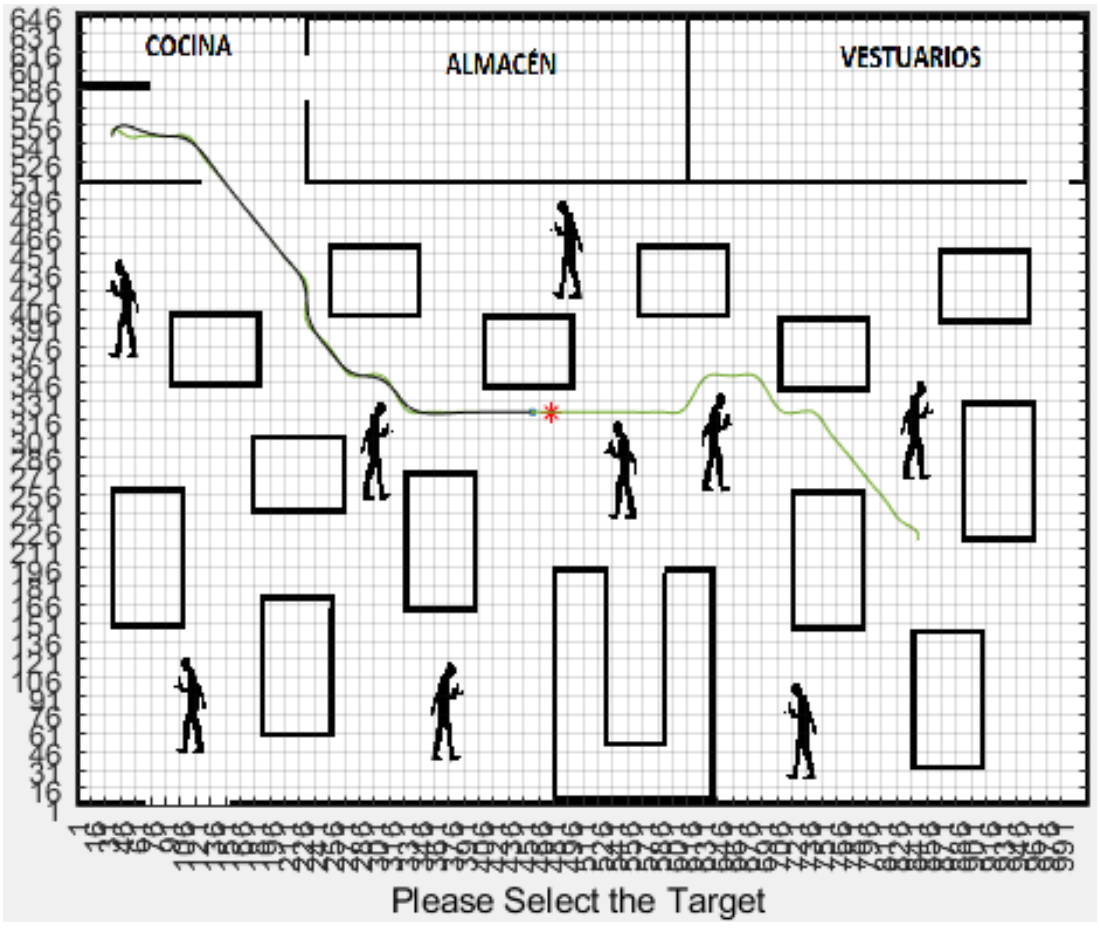
```
clear all
clc

%joy = vrjoystick(1);

global l
global radio_rueda
global camino
global pose
global pose2
global punto


l=3.5; %semidistancia entre rudas delanteras y traseras, tambien definido en
modelo
radio_rueda=1;

%Carga el fichero  BMP
MAPA = imread('restaurante.bmp');

%Transformación para colocar correctamente el origen del Sistema de
%Referencia
MAPA(1:end,:,:)=MAPA(end:-1:1,:,:);

%Tamaño de las celdas del grid
delta=15;
%genera la ruta óptima
Optimal_path=A_estrella(MAPA, delta);

%Condiciones iniciales
pose0=[Optimal_path(1,1); Optimal_path(1,2); pi/2];
posef=[Optimal_path(end,1); Optimal_path(end,2); 3*pi/2];

%definir camino
dd=5;
da=dd;

posicion_despegue=[41 540];
posicion_aterriza=[posef(1)-(da*cos(posef(3))) posef(2)-(da*sin(posef(3)))];

xc=[pose0(1) posicion_despegue(1) Optimal_path(2:end-1,1)'
posicion_aterriza(1) posef(1)];
yc=[pose0(2) posicion_despegue(2) Optimal_path(2:end-1,2)'
posicion_aterriza(2) posef(2)];

ds=1; %distancia entre puntos en cm.
camino=funcion_spline_cubica_varios_puntos(xc,yc,ds)';

%-------------------------------------------------------------------------
----------------------
t0=0;

%final de la simulación
tf=100;

%paso de integracion
h=0.1;
%vector tiempo
t=0:h:tf;
```

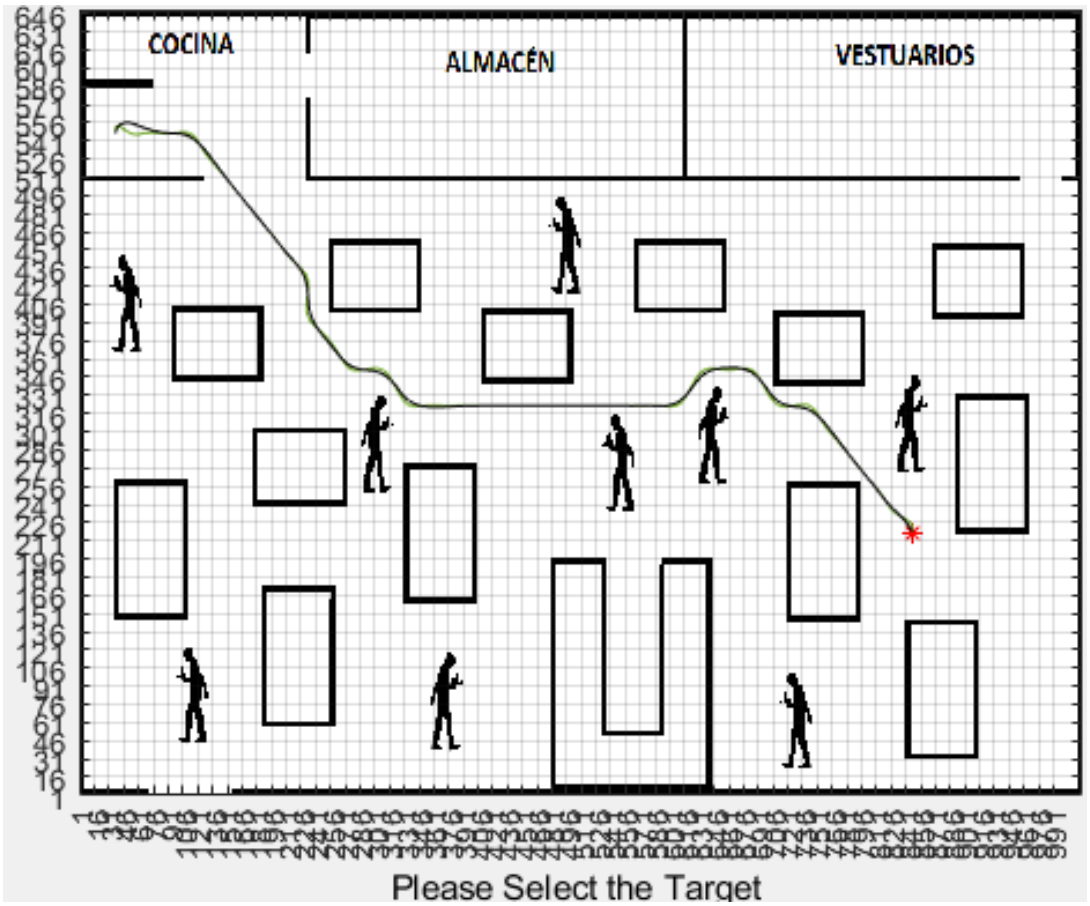
```
%indice de la matriz
k=0;

%inicialización valores iniciales
pose(:,k+1)=pose0;

t(k+1)=t0;

while (t0+h*k) < tf,

    %actualización
    k=k+1;

      %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

      %punto más cercano
      orden_minimo= minima_distancia(camino, pose(1:2,k));

      Look_ahead=20;
      seguir=orden_minimo+Look_ahead;
      if(orden_minimo+Look_ahead>length(camino))
            seguir=length(camino);
      end
      punto=[camino(seguir,1), camino(seguir, 2)];

    delta = (pose(1,k)-punto(1))*sin(pose(3,k))-(pose(2,k)-
punto(2))*cos(pose(3,k));
    LH=sqrt((pose(1,k)-punto(1))^2 + (pose(2,k)-punto(2))^2);
    rho=2*delta/LH^2;

    Distancia_al_final=sqrt((pose(1,k)-camino(end,1))^2 + (pose(2,k)-
camino(end,2))^2);

    V0=1*LH;
    if(V0>50)
        V0=50;
    end

    W=V0*rho;
    velocidad_derecha=(1/radio_rueda)*(V0+W*l);
    velocidad_izquierda=(1/radio_rueda)*(V0-W*l);

    conduccion=[velocidad_derecha velocidad_izquierda];

      %metodo de integración ruge-kuta
      pose(:,k+1)=kuta_diferencial_mapa(t(k),pose(:,k),h,conduccion,MAPA);

end

%PROGRAMACIÓN EL VOLVER HACIA ATRÁS
%Con la función flip, podemos hacer que invierta la matriz y por tanto
%recorra el camino hacia atrás
camino = flip(camino);

%Ahora pose0, tendrá como posición inicial, los valores de posef del
%anterior bucle
pose0=[posef(1); posef(2); posef(3)];
```

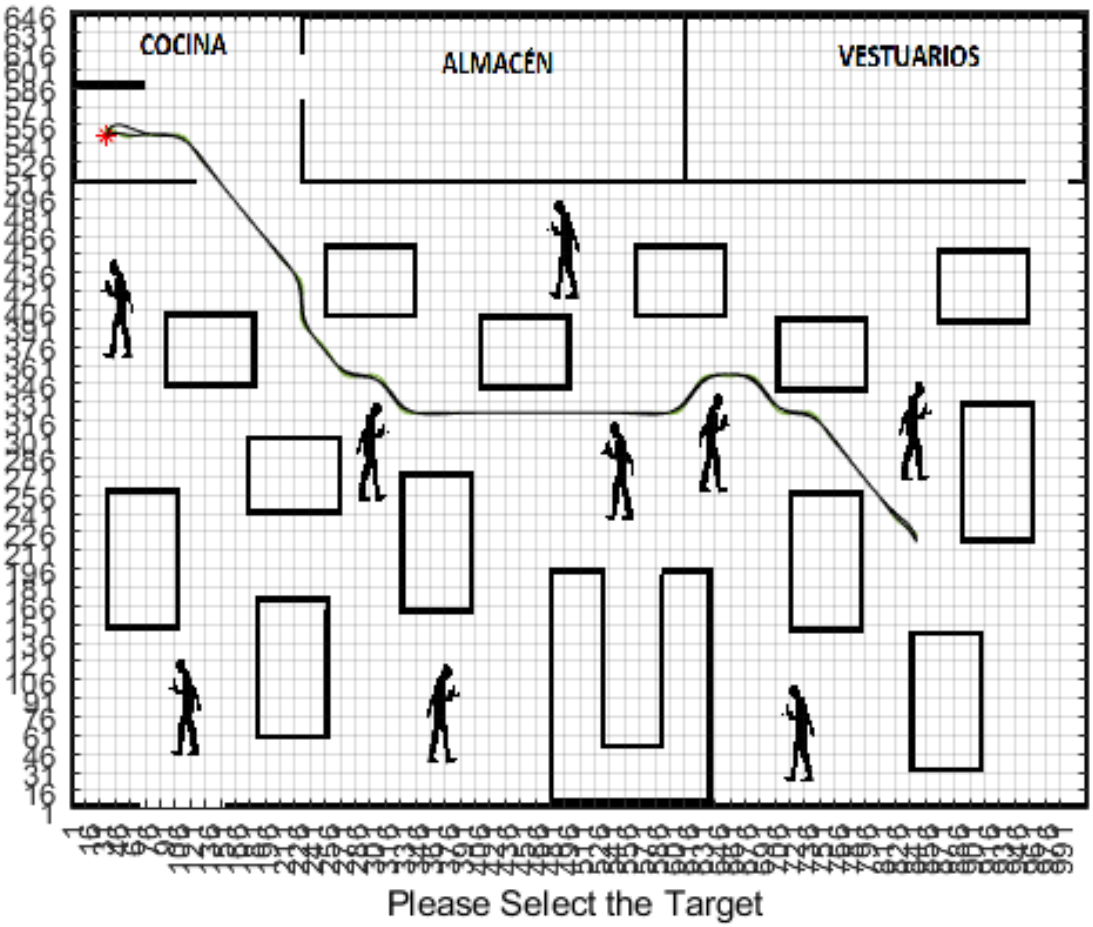
```
%final de la simulación -- amplio tanto tiempo en caso de que se elija el
camino más largo
tf=tf+150;

%vector tiempo
t=0:h:tf;

%inicialización valores iniciales
pose(:,k+1)=pose0;

t(k+1)=t0;

while (t0+h*k) < tf,

    %actualización
    k=k+1;

      %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

      %punto más cercano
      orden_minimo= minima_distancia(camino, pose(1:2,k));

      Look_ahead=20;
      seguir=orden_minimo+Look_ahead;
      if(orden_minimo+Look_ahead>length(camino))
            seguir=length(camino);
      end
      punto=[camino(seguir,1), camino(seguir, 2)];

    delta = (pose(1,k)-punto(1))*sin(pose(3,k))-(pose(2,k)-
punto(2))*cos(pose(3,k));
    LH=sqrt((pose(1,k)-punto(1))^2 + (pose(2,k)-punto(2))^2);
    rho=2*delta/LH^2;

    Distancia_al_final=sqrt((pose(1,k)-camino(end,1))^2 + (pose(2,k)-
camino(end,2))^2);

    V0=1*LH;
    if(V0>50)
        V0=50;
    end

    W=V0*rho;
    velocidad_derecha=(1/radio_rueda)*(V0+W*l);
    velocidad_izquierda=(1/radio_rueda)*(V0-W*l);

      %velocidad con signo negativo para que vaya hacia atrás
    conduccion=[-velocidad_derecha -velocidad_izquierda];

      %metodo de integración ruge-kuta
      pose(:,k+1)=kuta_diferencial_mapa(t(k),pose(:,k),h,conduccion,MAPA);

end
```

Con esto, obtenemos el siguiente resultado:


COCINA
ALMACÉN
VESTUARIOS
Start
Please Select the Target


COCINA
ALMACÉN
VESTUARIOS
Please Select the Target


COCINA
ALMACÉN
VESTUARIOS
Please Select the Target


COCINA
ALMACÉN
VESTUARIOS
Please Select the Target