

# Exploración de caminos y búsqueda



Escuela de  
Ingeniería  
Informática  
Universidad de Oviedo



Universidad de Oviedo  
*Universidá d'Uviéu*  
*University of Oviedo*

Cristian González García  
gonzalezcristian@uniovi.es

Material original de Jordán Pascual  
Espada

v 1.2.2 Noviembre 2022

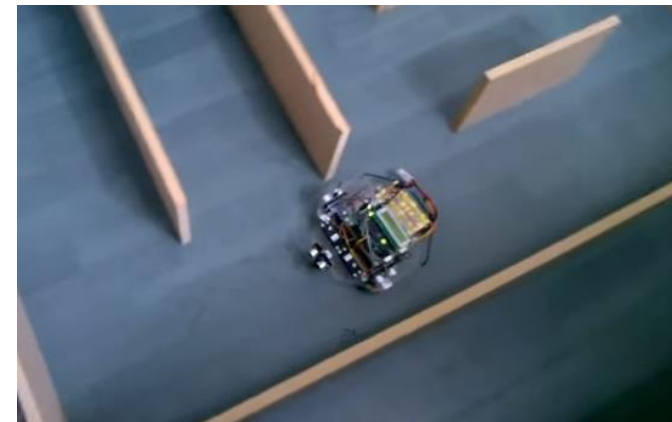
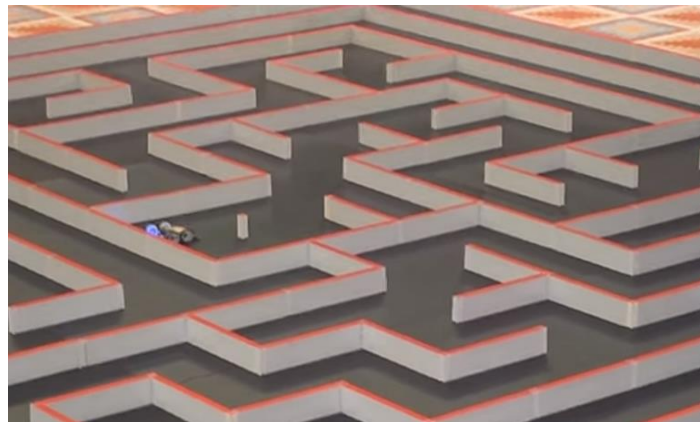
# Introducción

# Introducción

- Un **objetivo** recurrente en muchos **robots móviles** es
  - **Encontrar un objetivo en un entorno no conocido**
    - Entorno no conocido: un almacén, oficina, casa, etc.
    - Objetivo: una puerta, un lugar, un objeto, etc.
  - **Necesita explorar** el entorno
    - **Situarse en el mapa**/entorno en base a percepciones
      - Sensores, GPS, creación de un mapa, etc.
    - **Moverse** por el entorno **de forma lógica**
      - Sin chocar, sin dar vueltas en círculo ni aleatoriamente
    - **Tomar las decisiones** de movimiento **apropiadas**
      - Intersecciones: ¿hacia que lado voy?
      - No quedarse encerrado en bucles
      - Esquivar obstáculos
    - **Tener memoria**
      - **Recordar** los caminos tomados, la situación de los objetos, etc.
    - **Buscar objetivos**/Recorrer todo el entorno

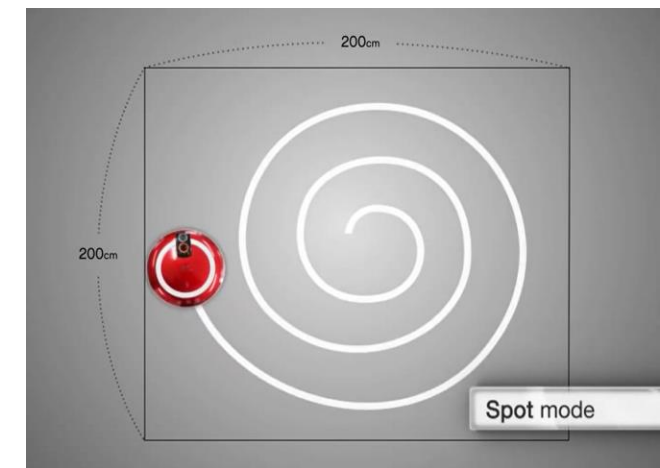
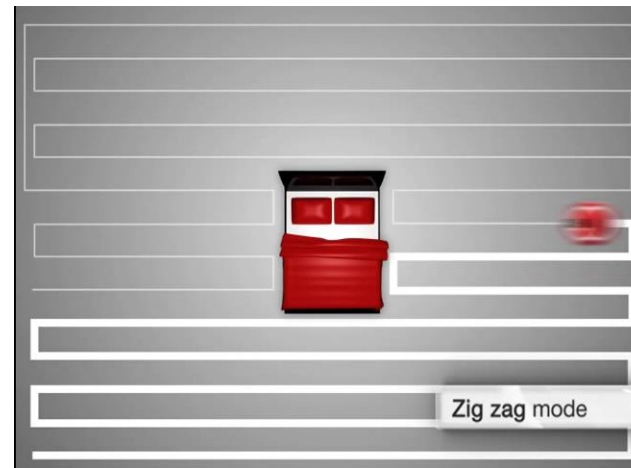
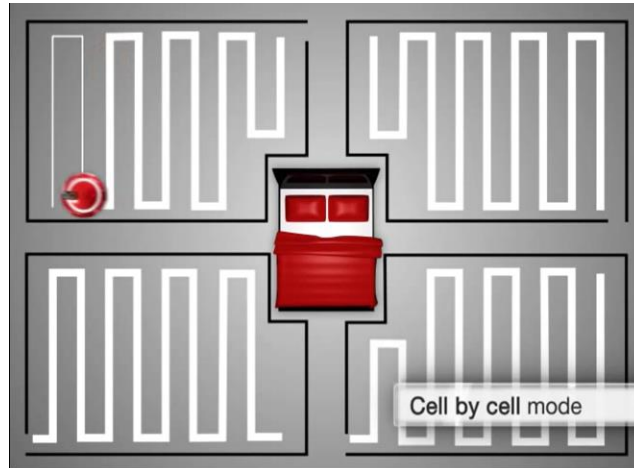
## Ejemplos

- Robot que aprende a llegar al centro del laberinto de forma cada vez más rápidas
  - <https://youtu.be/76bllun09Q>
  - <https://youtu.be/IngelKjme cg>
- Competición de laberintos
  - <https://youtu.be/M8YNhlpdzQ0>
- Robot autónomo
  - <https://youtu.be/OX0GXgEgfA0>



# Problema

- Si se tratase de habitaciones grandes, dentro de ellas se debería **realizar un barrido** (diferentes algoritmos)



# Laberinto de líneas

# Laberinto de líneas I

- o **Laberinto de líneas**

- o Es una **simplificación** muy utilizada **de este problema**
- o Cuenta con todos los elementos necesarios

- o Para **el robot es relativamente rápido detectar el camino**

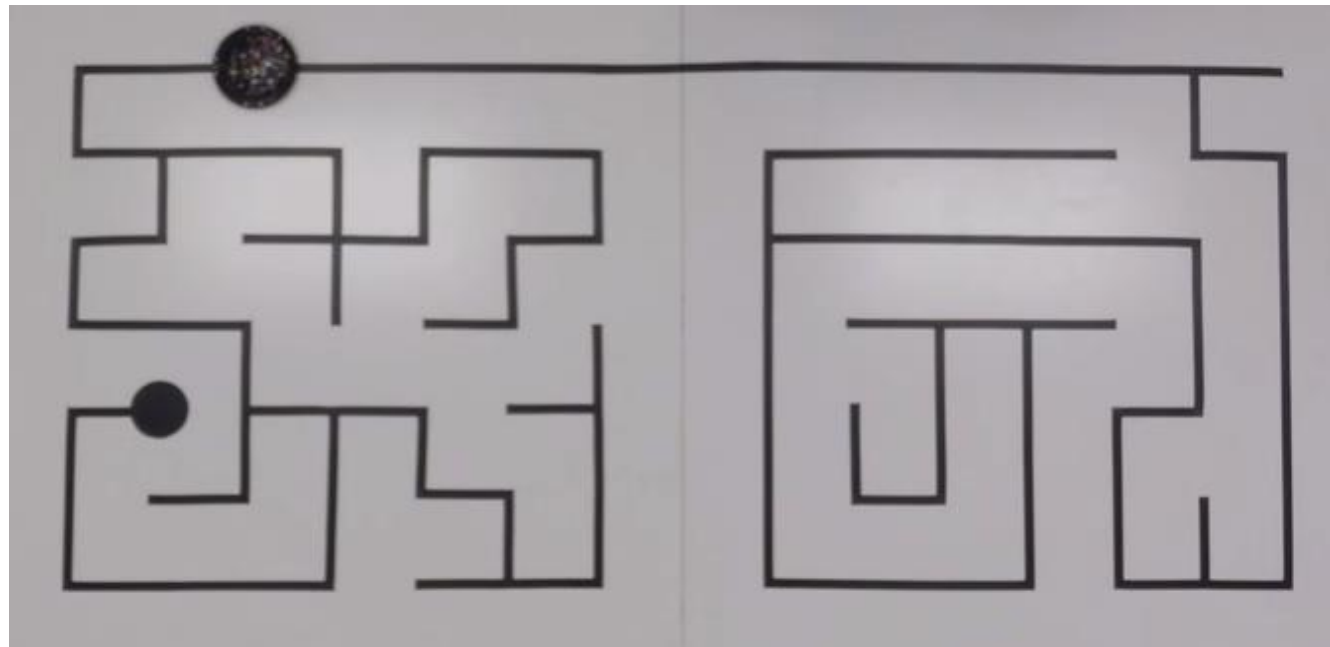
- o **No requiere de gran número de sensores o visión por computador**

- o **Rápidos y baratos**

- o Se pueden modificar fácilmente

## Laberinto de líneas II

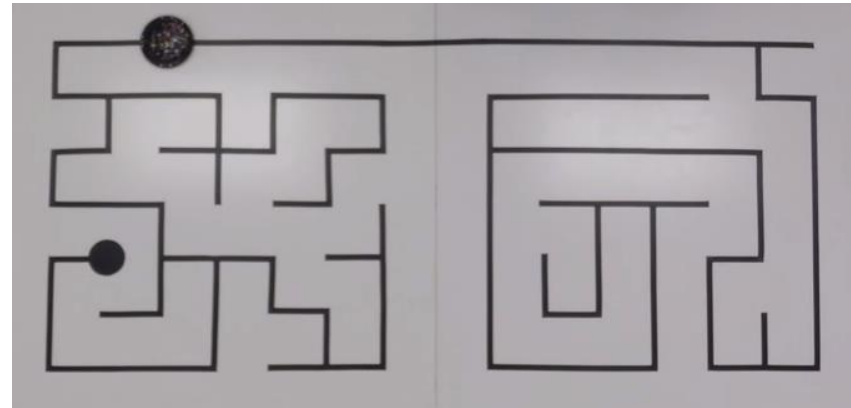
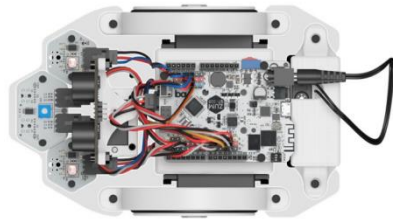
- Competición
  - <https://youtu.be/xyplzYmZmP8>
- Aprendizaje y recorrido bueno
  - <https://youtu.be/mJV-KDqHgDQ>





## Descripción del problema

- Una serie de **caminos con posibles intersecciones** que llegan a **una meta** (objetivo)
- El robot comienza en un punto** y debe encontrar la meta
  - No tiene por qué recorrer todo el laberinto, pero...

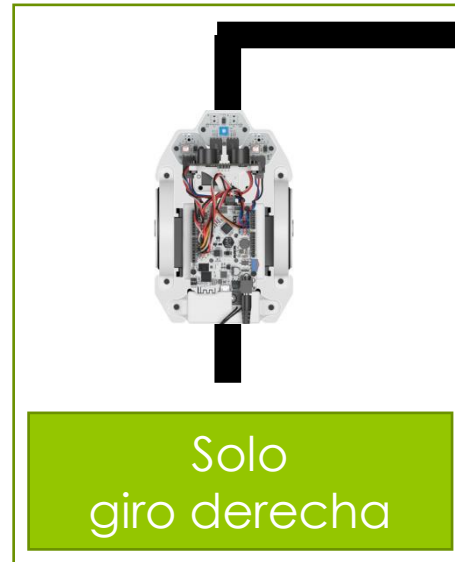
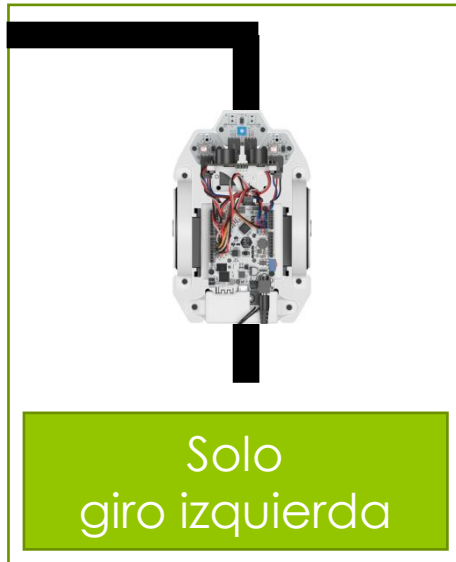


- Diferentes tipos de caminos**
  - Con/sin bucles (poder dar vueltas en círculo o no)
  - Ángulos rectos para caminos, máximo intersección de 4 caminos

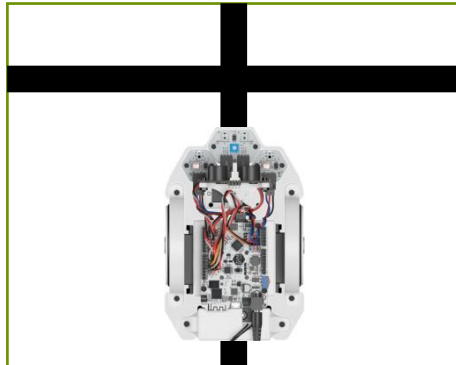
# Consideraciones

- **El robot debe detectar**
  - Si está en el camino
  - Y como es el camino en cada momento
- Para ello **utiliza sensores y percepciones**
- **Debe «pensar»** como moverse (Algoritmo)
  - Asegurarse de **recorrer** (casi) **todos los posibles caminos**
    - **Aprenderse todas las posibilidades**
    - De esta forma **se garantiza llegar a la meta**
  - **No puede perderse**
    - **Tener memoria** de por donde pasó y que le queda por recorrer
  - Una vez llega a la meta, **debe recordar que caminos correctos tomó**
    - Debe **calcular el camino más corto**

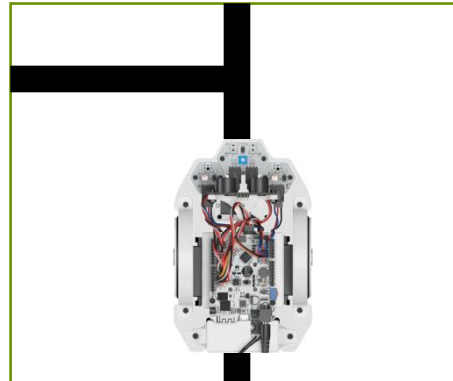
## Posibilidades I



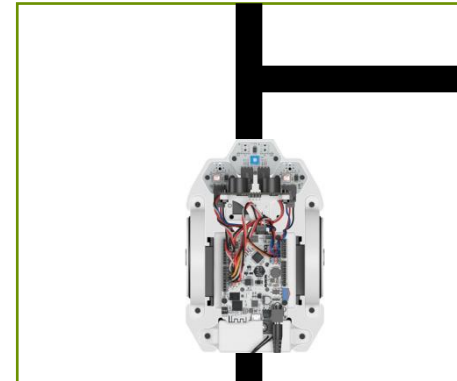
## Posibilidades II



Adelante, giro  
derecha o giro  
izquierda



Adelante o giro  
izquierda



Adelante o giro  
derecha

## Posibilidades III



Camino sin  
salida



Objetivo

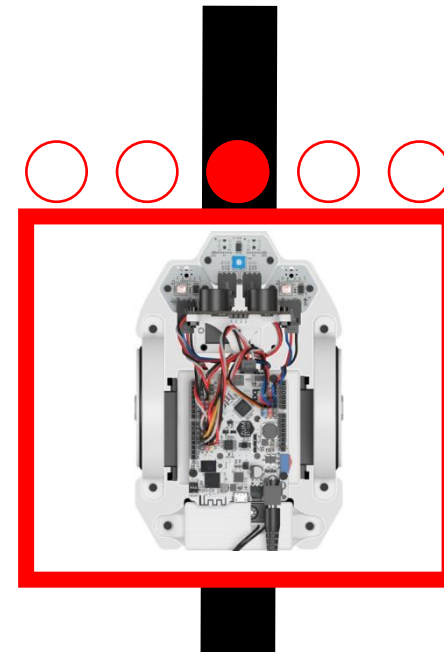
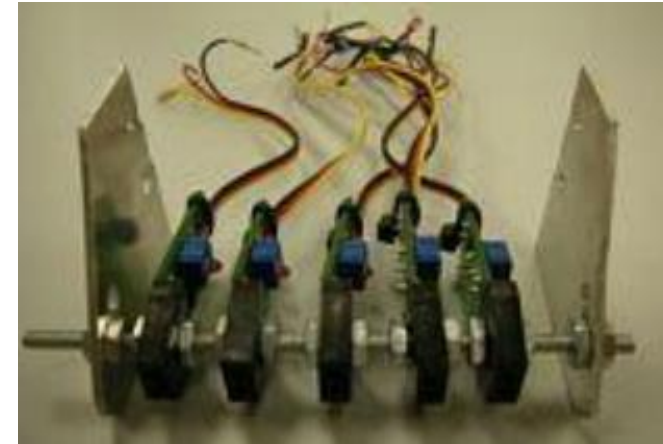
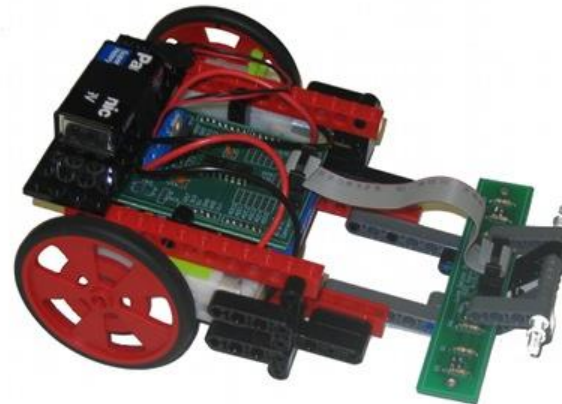
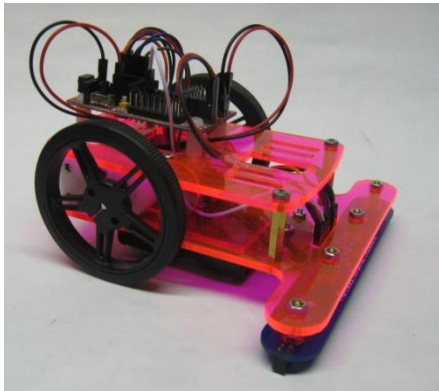
## Detección de tipo de camino

- Se debe **detectar ante que tipo de camino se encuentra**
- **Sensores infrarrojos** para la detección de la línea
  - Al menos 2 sensores
  - Con 3 sensores
    - El del medio y laterales para seguir la línea
    - Los laterales para detectar cruce
  - Con 4
    - 2 para seguir la línea (independientes)
    - Los laterales para detectar cruce
- **Cuantos más sensores...**
  - **Más sencilla** resulta la **detección**
  - **Menos movimiento** requiere
  - **Más precisión** para «recolocar» el robot en las líneas

# Colocación y funcionamiento de los sensores

# Sistema I – 5 sensores infrarrojos I

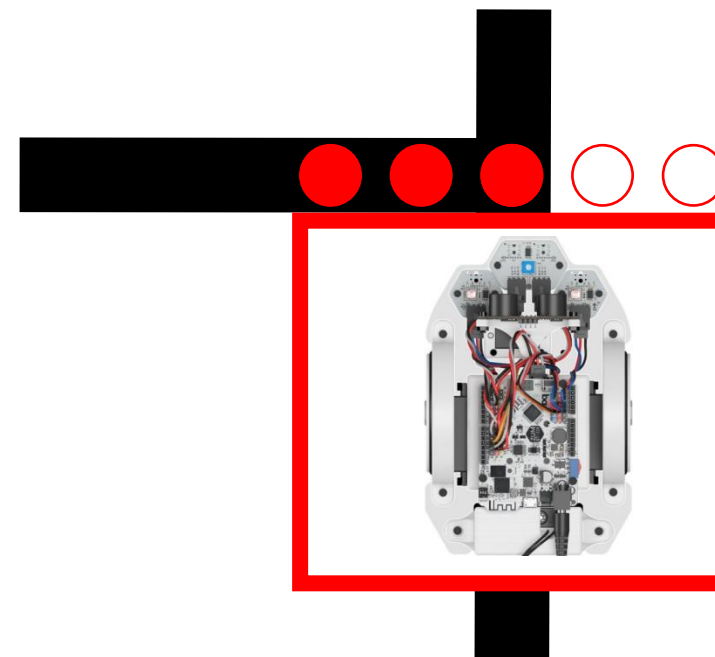
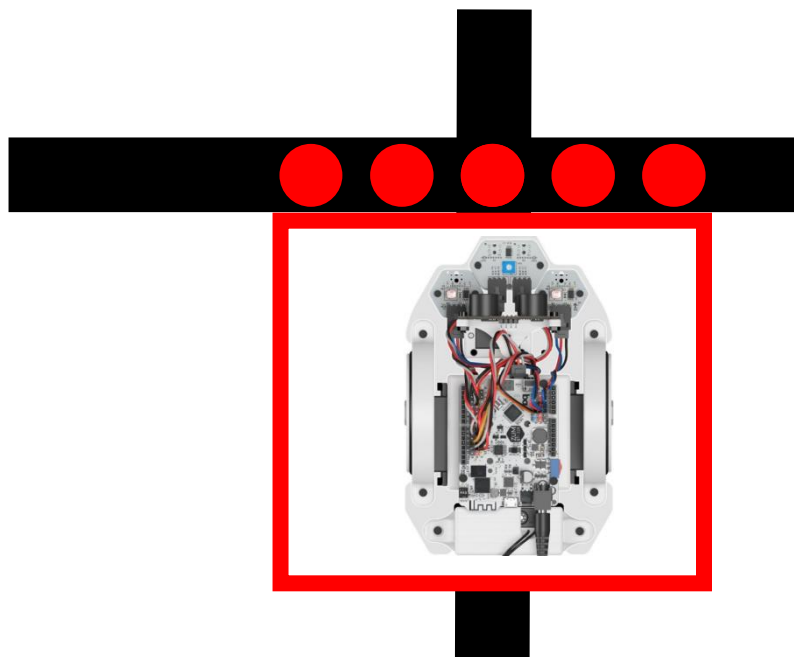
- Muchos robots utilizan **5 sensores**
  - Posición respecto 1 camino
    - 1 0 0 0 0 = a la izquierda
    - 0 1 0 0 0 = centro un poco a la izquierda
    - 0 0 1 0 0 = centro
    - 0 0 0 1 0 = centro un poco a la derecha
    - 0 0 0 0 1 = a la derecha





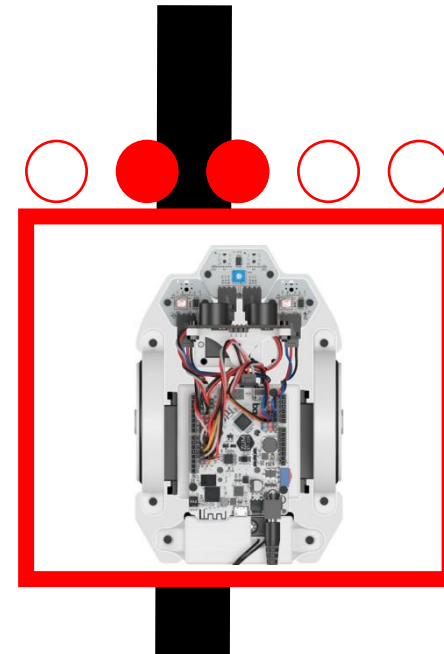
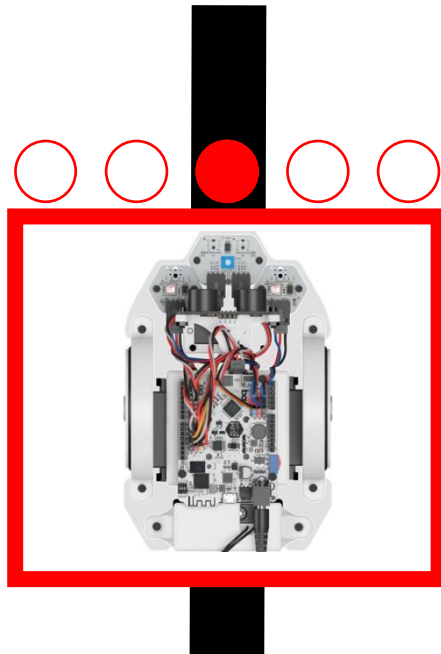
## Sistema I – 5 sensores infrarrojos II

- Se pueden detectar rápidamente el tipo de camino / intersección



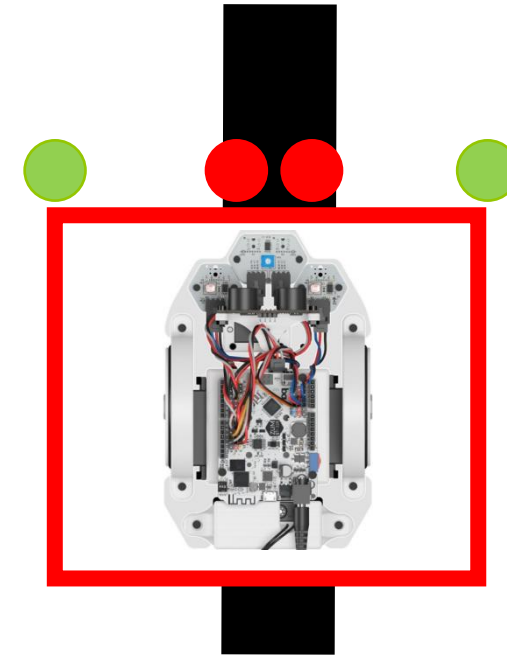
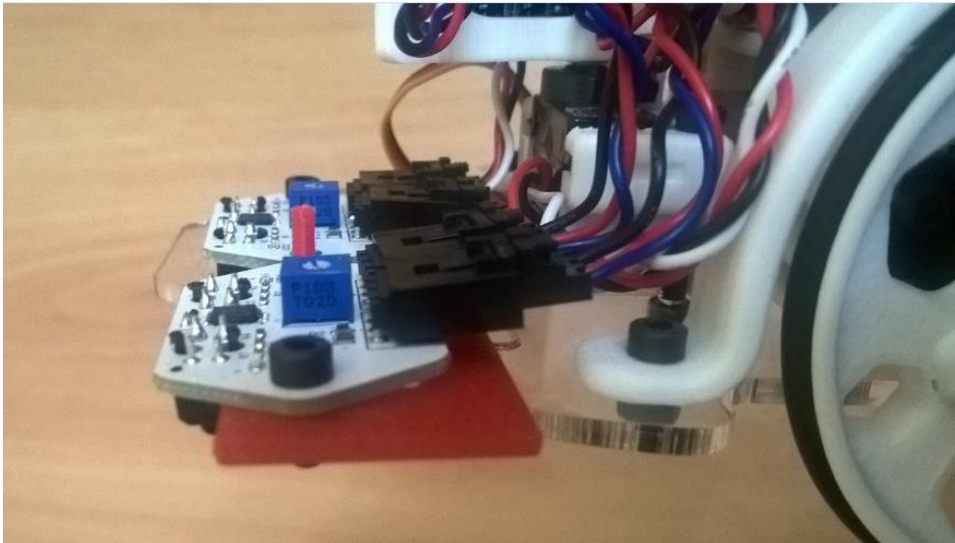
## Sistema I – Detección de tipo de camino

- Tamaño de la línea y distancia entre sensores
  - Dependiendo de esta relación se activarán 1 o 2 sensores para un camino simple



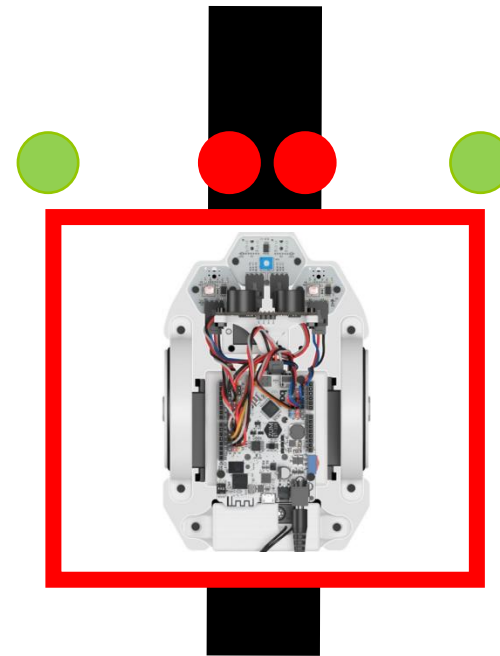
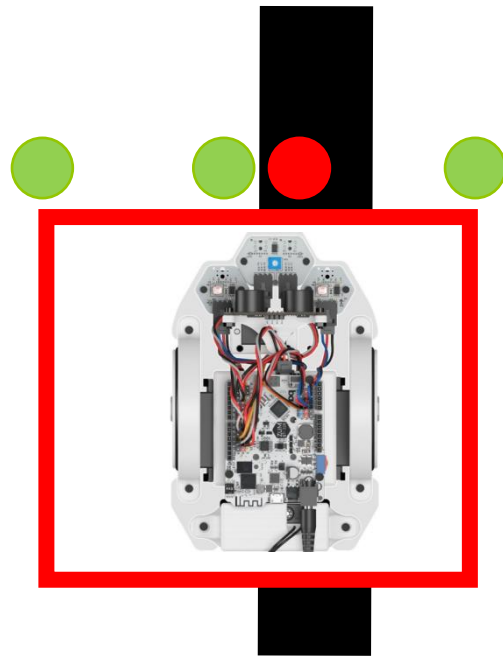
## Sistema II – 4 sensores infrarrojos

- Utilizaremos 4 sensores colocados en la parte frontal del Robot



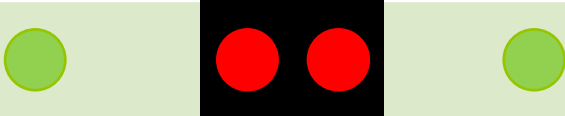
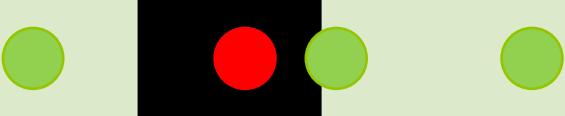

## Sistema II – Detección de tipo de camino

- Camino simple
  - Activará 1 o 2 sensores



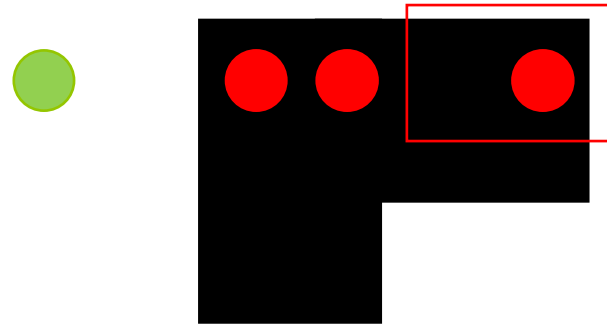
# Sistema II – Controlador I – Seguir camino simple

## o Seguir camino simple

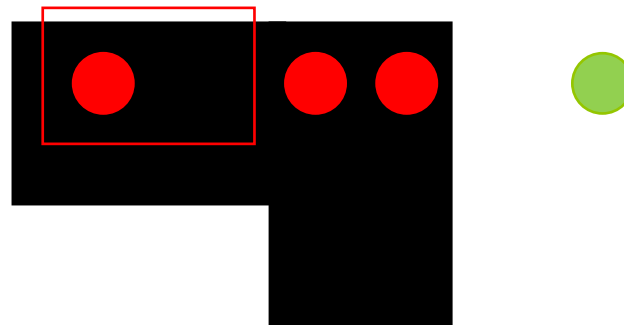
Sensores	Acción
	Avanza hacia delante
	Avanza recolocándose ligeramente hacia la derecha
	Avanza recolocándose ligeramente hacia la izquierda
	Casi fuera del camino, debe avanzar y girar significativamente hacia la izquierda No debería de producirse
Otros ...	...

## Sistema II – Controlador II – Curvas I

- o Giro a la derecha



- o Giro a la izquierda



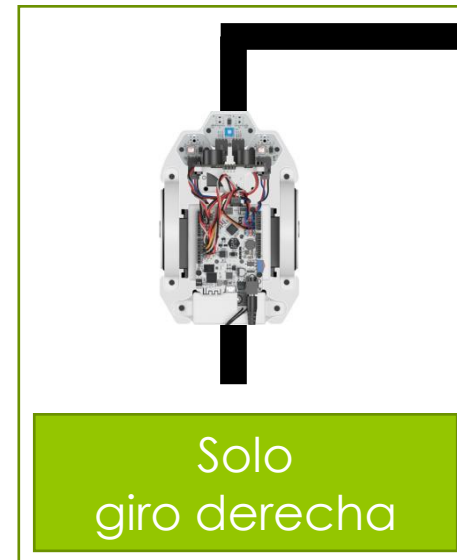
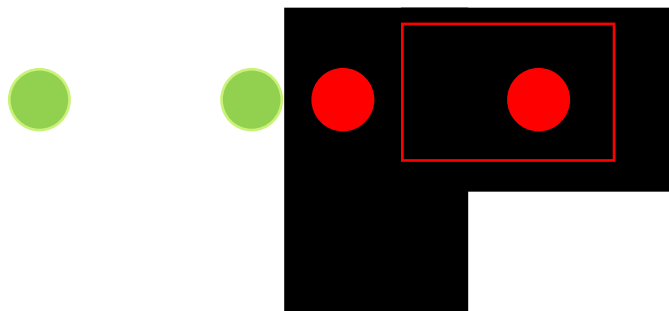
Solo  
giro derecha



Solo  
giro izquierda

## Sistema II – Controlador II – Curvas II

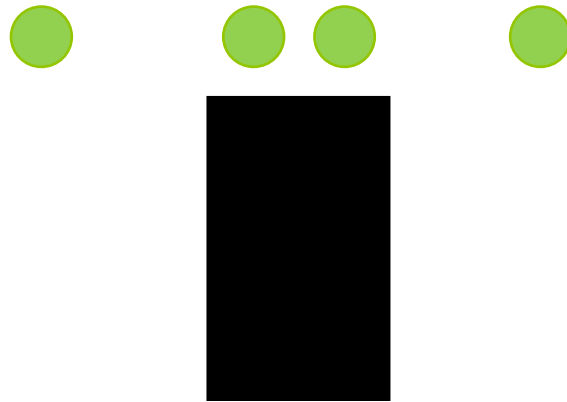
- Se podrían incluir **otras percepciones en las curvas**
  - Aunque podrían dar lugar a confusiones y problemas... o solucionarlos
  - Muy extraño que se de el caso



# Sistema II – Controlador III – Camino sin salida

- o **Camino sin salida**

- o Dar la vuelta -> Rotar 180 grados



Camino sin  
salida



# Sistema II – Controlador IV – Intersecciones I

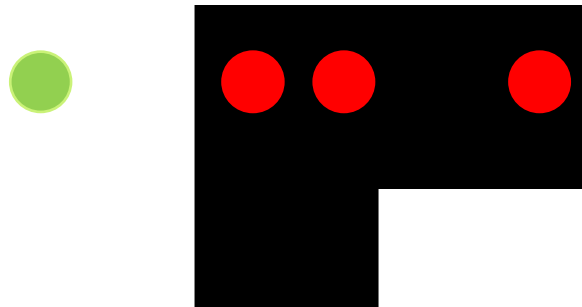
- **Toma de decisiones en Intersecciones**

- Debe **tomar la decisión correcta** en función del tipo de intersección



## Sistema II – Controlador IV – Intersecciones II

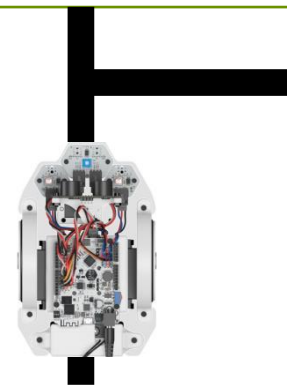
- Derecha vs. Derecha + adelante
  - Las dos tienen la misma percepción



- ¿Como diferenciarlas?



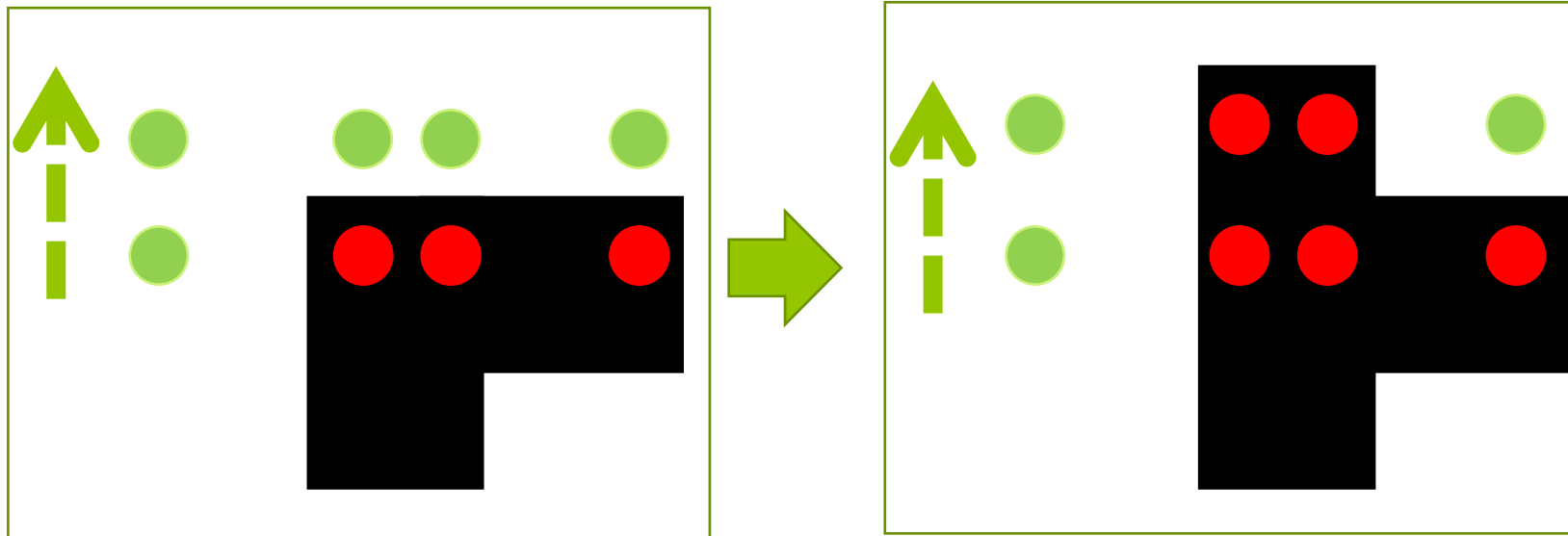
Solo  
Giro Derecha



Adelante o giro  
derecha

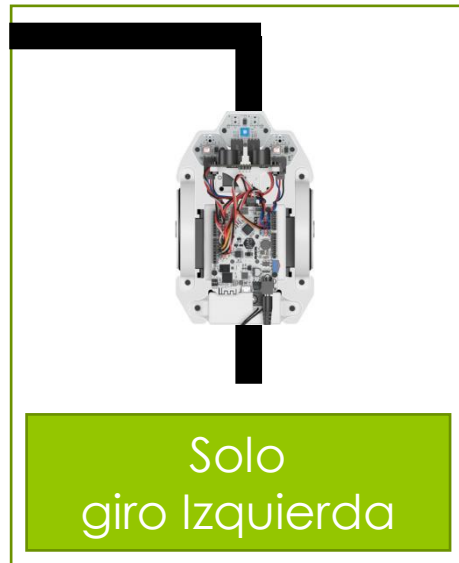
## Sistema II – Controlador IV – Intersecciones III

- ¿Como diferenciarlas?
- Avanzar ligeramente hasta conseguir una percepción diferente



## Sistema II – Controlador IV – Intersecciones IV

- Mismo problema
  - Izquierda vs. Izquierda + adelante



## Sistema II – Controlador IV – Intersecciones V

- Mismo problema
  - Debemos avanzar para determinar el tipo



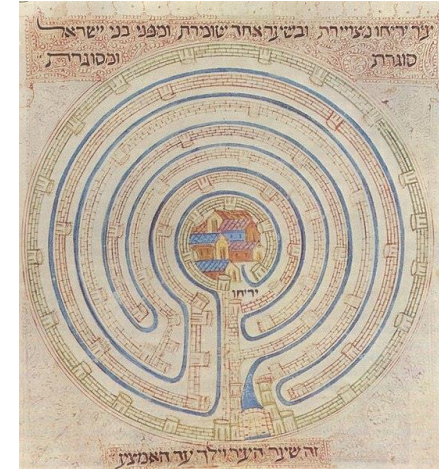
## Sistema II – Controlador V

- **Encontrar la meta** seguramente implicará
  - **Explorar varios caminos erróneos**
  - **Volver a caminos ya recorridos para seleccionar otras opciones no recorridas**
    - Esto depende también del algoritmo utilizado
- (Extra) **Queremos que vaya almacenando los caminos correctos**
  - Si tiene que volver a encontrar el objetivo
    - Lo hará sin equivocarse
    - Sin repetir caminos ya visitados
    - Sin entrar en bucle
- Usar diferentes algoritmos

# Laberintos

# Tipos de laberintos

- «Labyrinth» o Laberinto unicursal
  - No te puedes perder
  - Simple y sin necesidad de decidir
  - Suelen tener solo 1 camino posible
    - Si hay varios, todos van a la «meta»
  - Solo 1 entrada y 1 salida
- «Maze» o Laberinto multicursal
  - Te puedes perder
  - Es complejo y tienes que tomar decisiones
  - Puede tener varias entradas y/o salidas



[https://commons.wikimedia.org/wiki/File:Map\\_of\\_Jericho\\_in\\_14c\\_Farhi Bible by Elisha ben Avraham Crescas.jpg](https://commons.wikimedia.org/wiki/File:Map_of_Jericho_in_14c_Farhi_Bible_by_Elisha_ben_Avraham_Crescas.jpg)



[http://www.langer.ws/leeds\\_castle\\_maze\\_solution.jpg](http://www.langer.ws/leeds_castle_maze_solution.jpg)



# Algoritmos

## Algunos posibles algoritmos

- ◉ RHR Right Hand Rule (Seguimiento de muros o caminos)
  - ◉ LHR Left Hand Rule
- ◉ Tremaux
- ◉ Recorridos sobre arboles
- ◉ Recorridos sobre grafos
- ◉ Otros...

## Elegir un camino

- En este caso
  - **Todos los caminos son iguales**
  - **No sabemos si unos tienen más posibilidades de éxito que otros**
- En otros escenarios, **si disponemos de información adicional podríamos utilizar heurísticos**
  - **Heurístico:** algoritmo para «acortar» las posibilidades valorando las existentes y dándoles un peso en base a una información conocida
    - **Explorar primero los caminos con más posibilidad de éxito**
      - pero que **no tienen** porqué ser los más rápidos
    - O, **explorar el posible camino más corto**
      - pero no con las mayores posibilidades de éxito

# Right/Left Hand Rule – Introducción

- **Regla de la mano derecha - Right Hand Rule (RHR)**
  - Variante: **Regla de la mano izquierda – Left Hand Rule (LHR)**
- **Válido si los caminos**
  - No tienen bucles
    - Laberintos con círculos concéntricos con salida en el medio
    - Nos quedaríamos dando vueltas en uno de esos círculos
  - Todas las paredes estén conectadas a la que tocamos con la mano
- Recorreremos casi todo el laberinto o muy poco, dependerá de la situación, pero no repetiremos partes y no nos perderemos
  - Este algoritmo reduce **la velocidad**, que **dependerá de la situación inicial** (suerte)
- No hace falta conocer dónde está la salida al inicio

# Right Hand Rule I – Funcionamiento

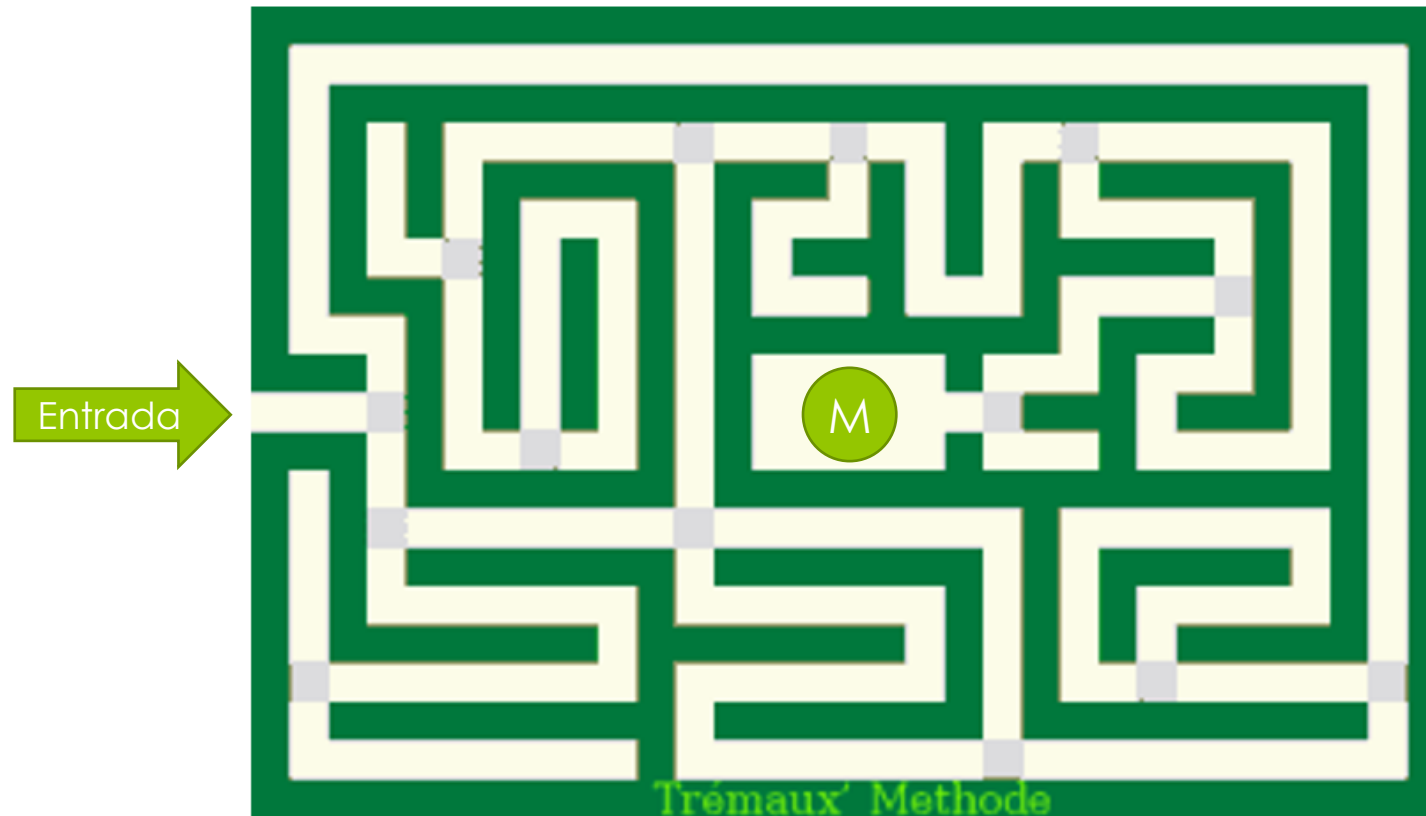
- o **Funcionamiento**

- o Colocar la mano derecha sobre la pared
  - o Caminar sin despegar la mano **nunca** de la pared
- o Si tiene que tomar una **decisión en una intersección**, aplica este **orden de prioridad**
  1. Gira a la derecha
  2. Sigue de frente
  3. Gira a la izquierda
- o **Toma siempre la decisión disponible de mayor prioridad**
- o La variante de este es **Left Hand Rule** (LHR), usando la mano izquierda
- o **Optimizaciones**
  - o Visión por computador: saltar caminos sin salida visibles
- o Ejemplo
  - o FFXV





## Right Hand Rule III – Ejemplo fallido



[https://commons.wikimedia.org/wiki/File:Tremaux%27 Methode - animiertes Beispiel.gif](https://commons.wikimedia.org/wiki/File:Tremaux%27_Methode_-_animiertes_Beiispiel.gif)



# Tremaux I – Introducción

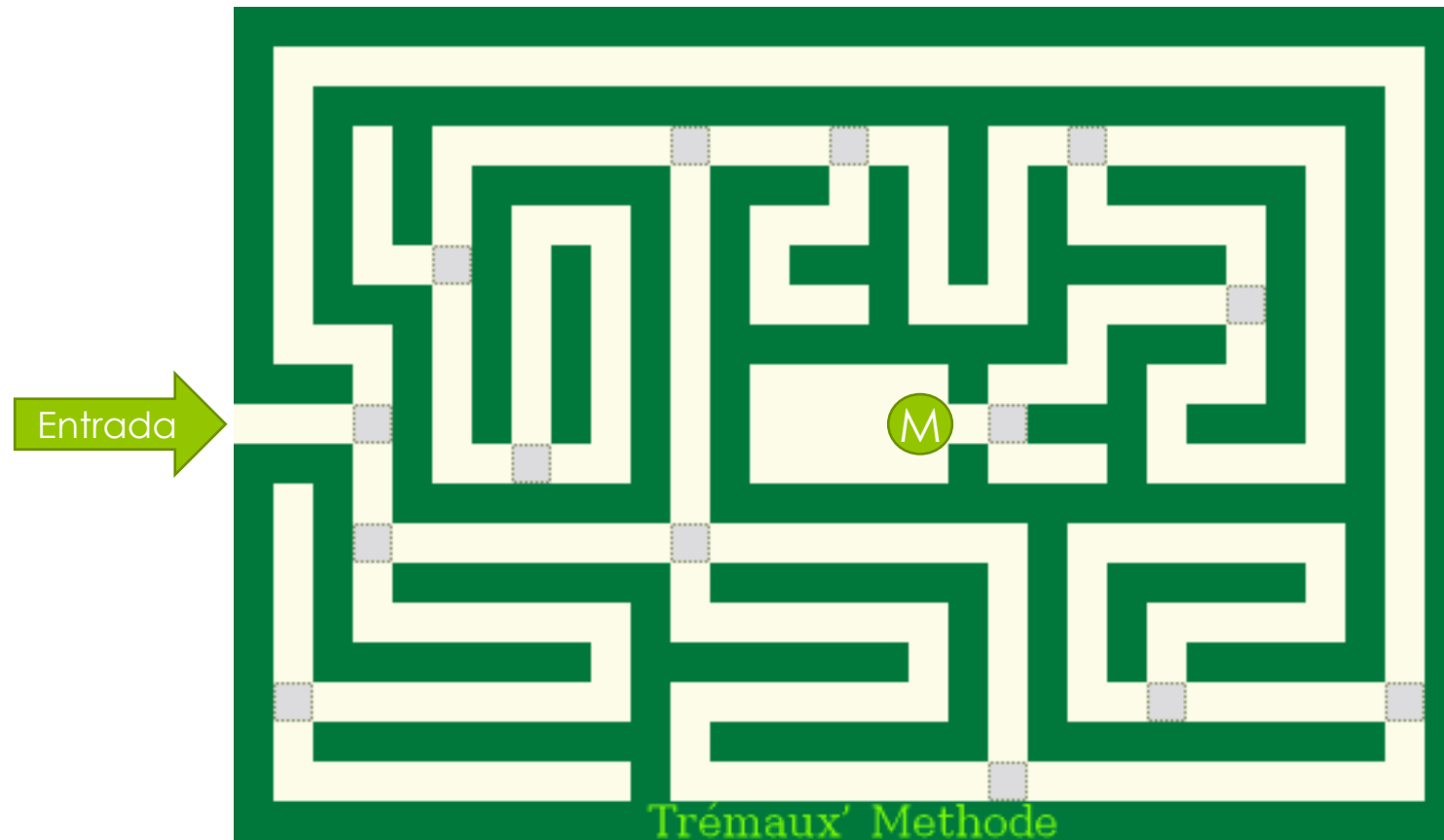
- Diseñado por Charles Trémaux
  - Ingeniero francés del siglo XIX
- Actualmente, es el **algoritmo más eficiente**
- **Asegura la escapatoria de casi cualquier tipo de laberinto**
- **Notas**
  - Hay que ir **dejando marcas** en los cruces
  - Funciona como **una pila**, donde se apilan los cruces y sus intersecciones
  - Se basa en **probar ordenadamente todos los posibles caminos** sin repetirlos
    - Los **laberintos** son **finitos**
  - Si tenemos **suerte**, será corto ya que saldrá a la primera, con mala suerte recorreremos todo el laberinto
    - Se puede salir en minutos como horas, pero saldremos
  - En caso de que el laberinto no tenga salida, volveremos a la entrada
- No hace falta conocer dónde está la salida al inicio

# Tremaux II – Funcionamiento

## ○ **Funcionamiento**

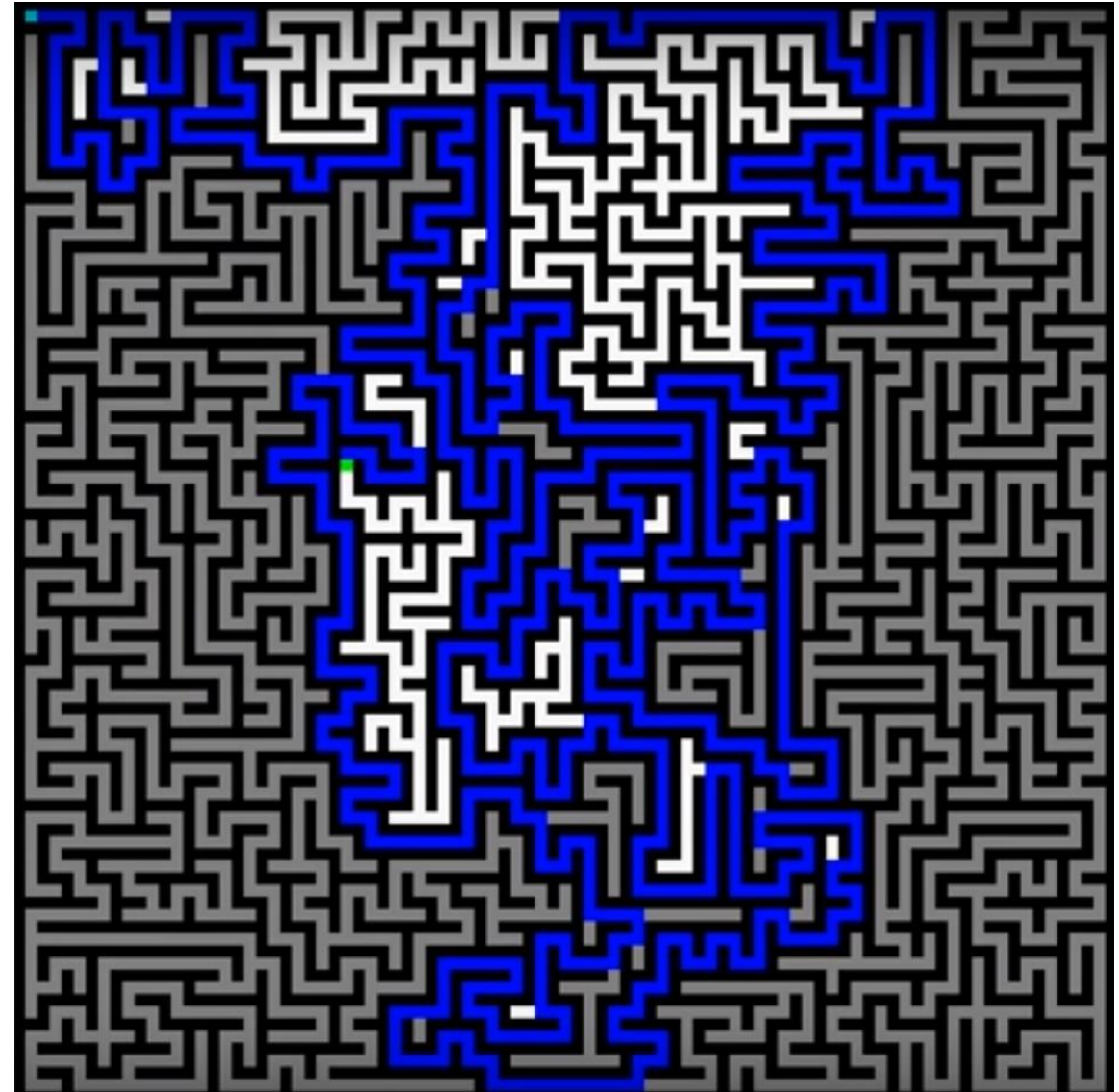
- No seguir un camino ya seguido
- Al llegar a una intersección nueva -> Seleccionar un camino aleatoriamente
  - Algoritmo, siempre mismo orden (derecha, frente, izquierda), probabilidad, valoración visual, ...
- Si un camino nuevo lleva a una intersección ya visitada o a un camino sin salida -> Retroceder hasta la entrada del camino
- Si un camino visitado lleva a un intersección ya visitada -> Tomar un camino nuevo, y si no la hay, tomar cualquier camino que no haya sido visitado ya dos veces

## Tremaux III – Ejemplo I



## Tremaux III – Ejemplo II

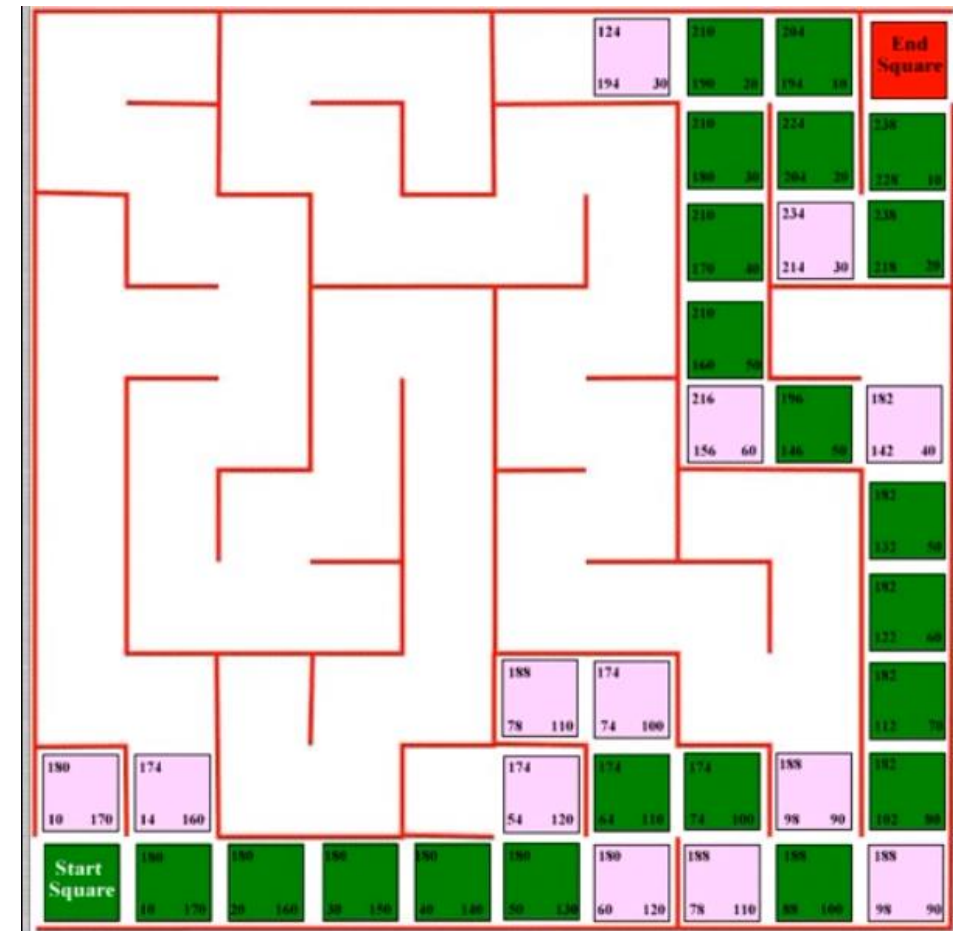
- Negro -> desconocido
- Cuadrado blanco -> Intersección
- Cuadrado rojo -> Meta
- Blanco -> 0 veces, sin recorrer
- Azul -> 1 vez
- Gris -> 2 veces, recorrido en profundidad



<https://youtu.be/7mzNbGOhw08>

# Recorridos sobre arboles

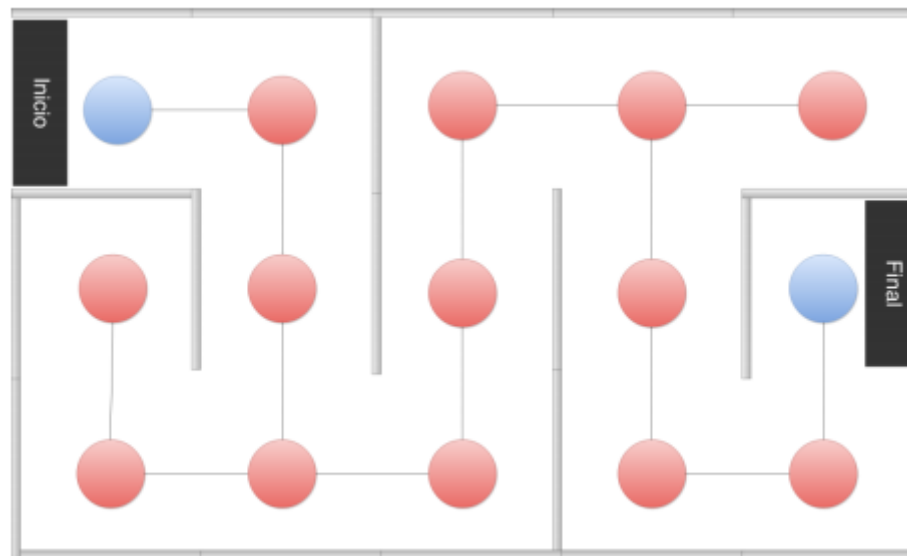
- Almacenar los caminos como un árbol
- Recorrerlas utilizando un algoritmo
  - Ej: A\*
    - Algoritmo completo
      - Siempre encuentra la solución si existe
- Necesitáis conocer dónde está la salida



<https://youtu.be/xUXcTnNG6ag>

# Recorridos sobre grafos

- Almacenar los caminos como un grafo
  - Posiblemente requiera tener un conocimiento inicial en muchos casos
- Recorrerlas utilizando un algoritmo de grafos
  - Floyd-Warshall, Dijkstra, Bellman-Ford, ...
- Necesitáis conocer el mapa



Preguntas tema

○ <https://forms.office.com/r/YSYz89Wsym>

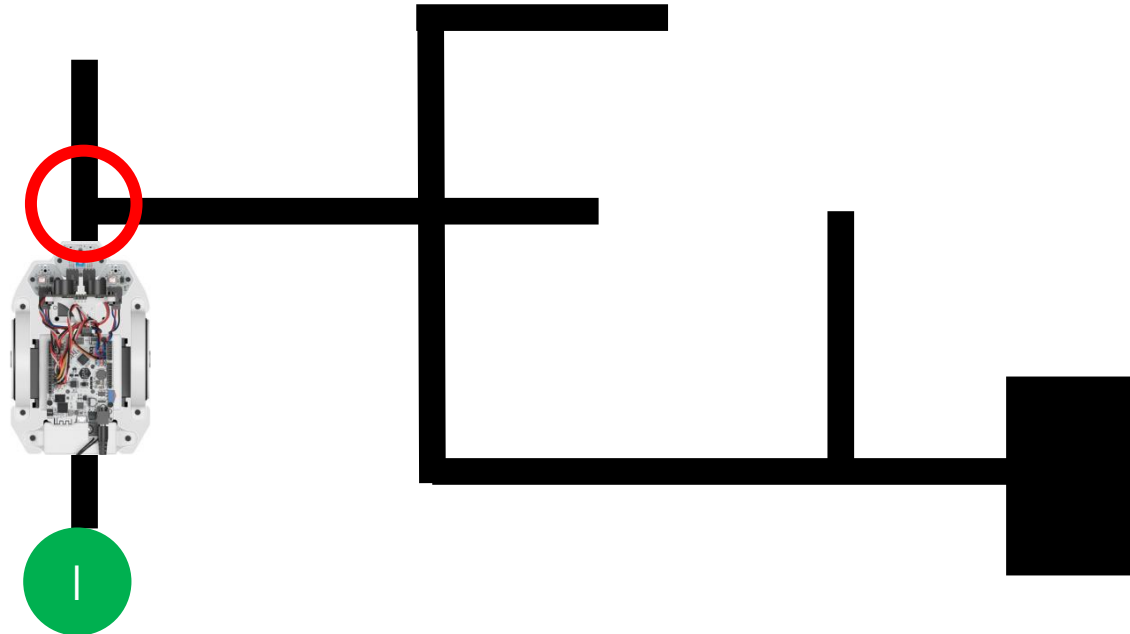


Ejemplo con almacenamiento de camino



## Ejemplo I

- Controlador **Left Hand Rule** con almacenamiento de camino correcto
- Primera intersección -> **Adelante**
  - Prioridad: izquierda – adelante – derecha



## Ejemplo II

- Guardamos en la memoria: A
  - A -> Adelante



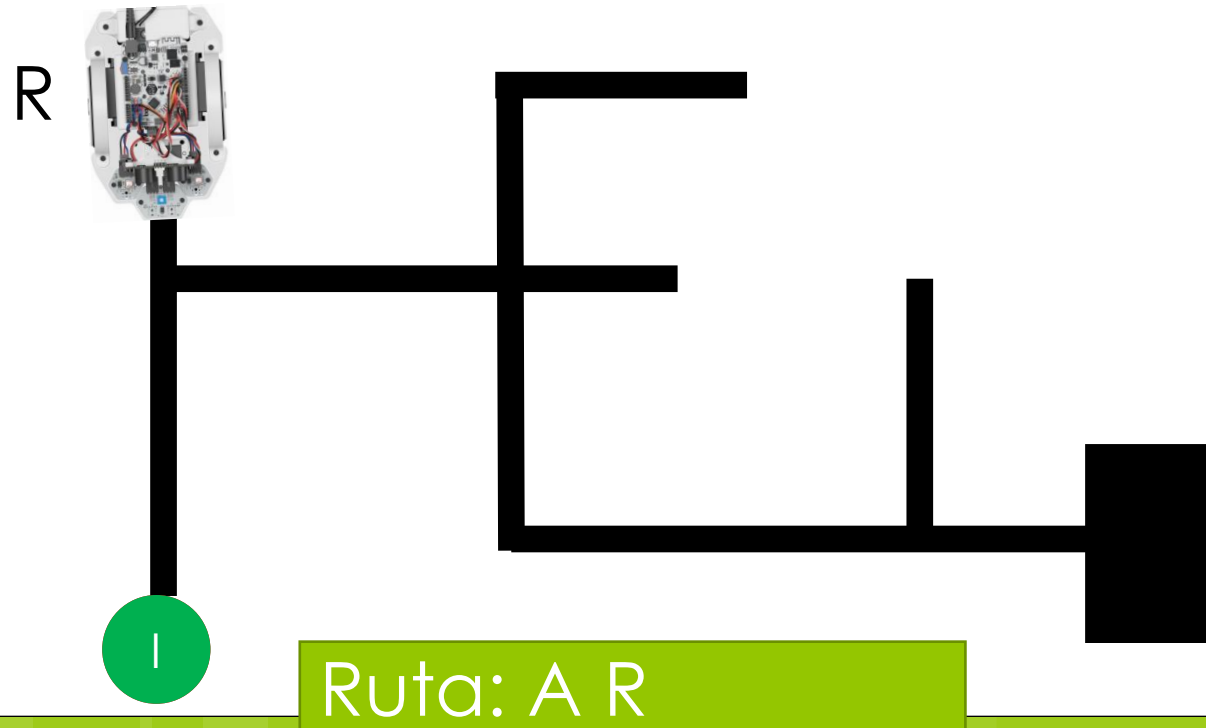
A



Ruta: A

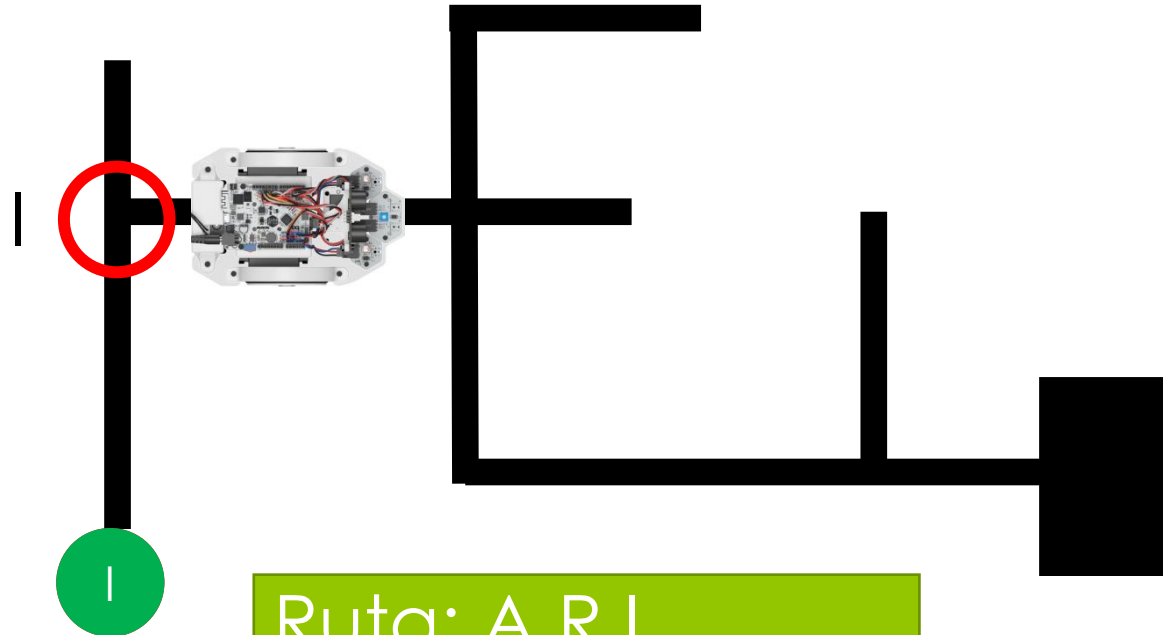
## Ejemplo III

- Al encontrarse con un camino -> **Retroceder** (180°)
- Guardamos en la memoria: R
  - R – Retroceder



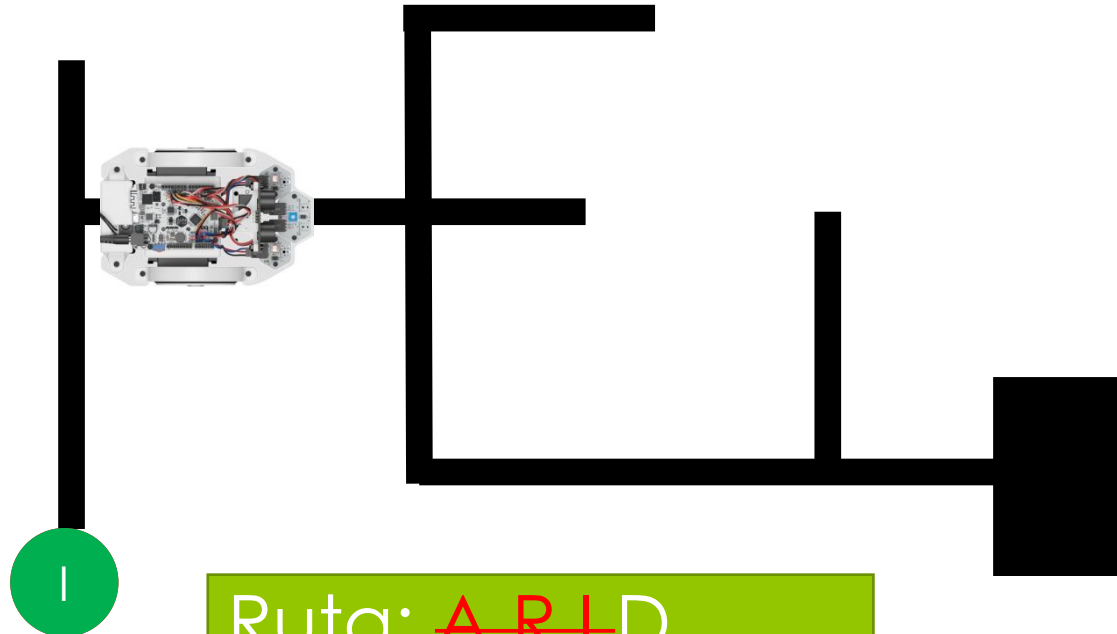
## Ejemplo IV

- Al llegar a la **intersección**, va a la **izquierda**
- Guardamos en la memoria: I
  - I -> Izquierda



## Ejemplo V

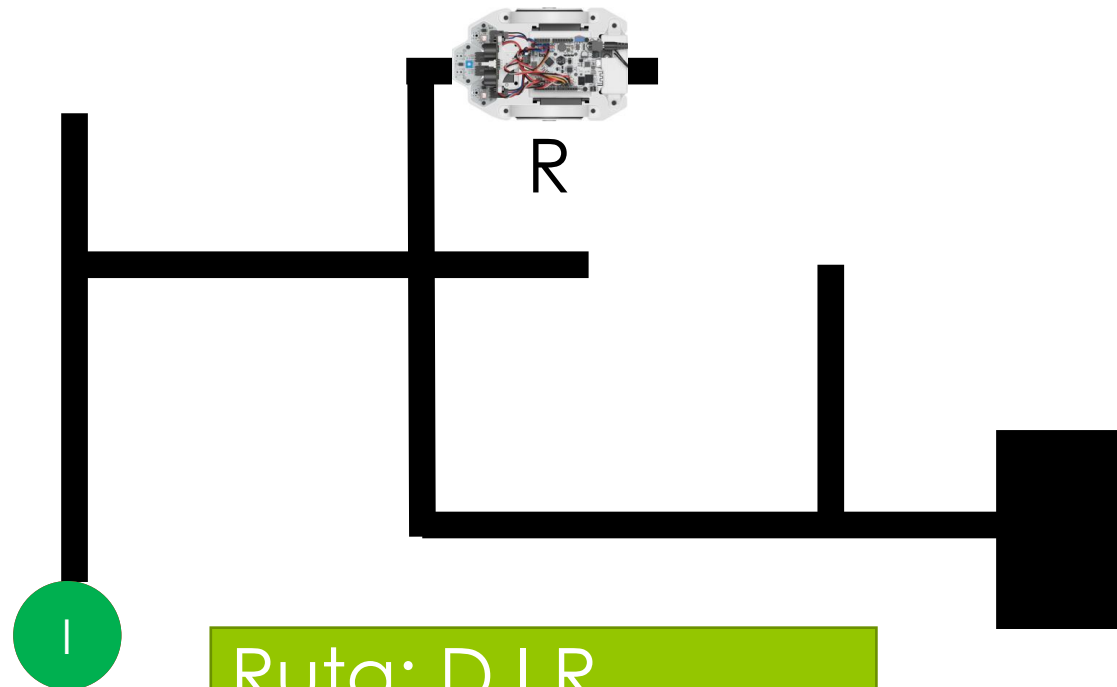
- El «A R l» que hay en la ruta significa que el robot **ha dado una vuelta inútil**
  - Adelante, retroceder, Izquierda, es lo mismo que ir a la Derecha**
  - Sustituimos «A R l» por D





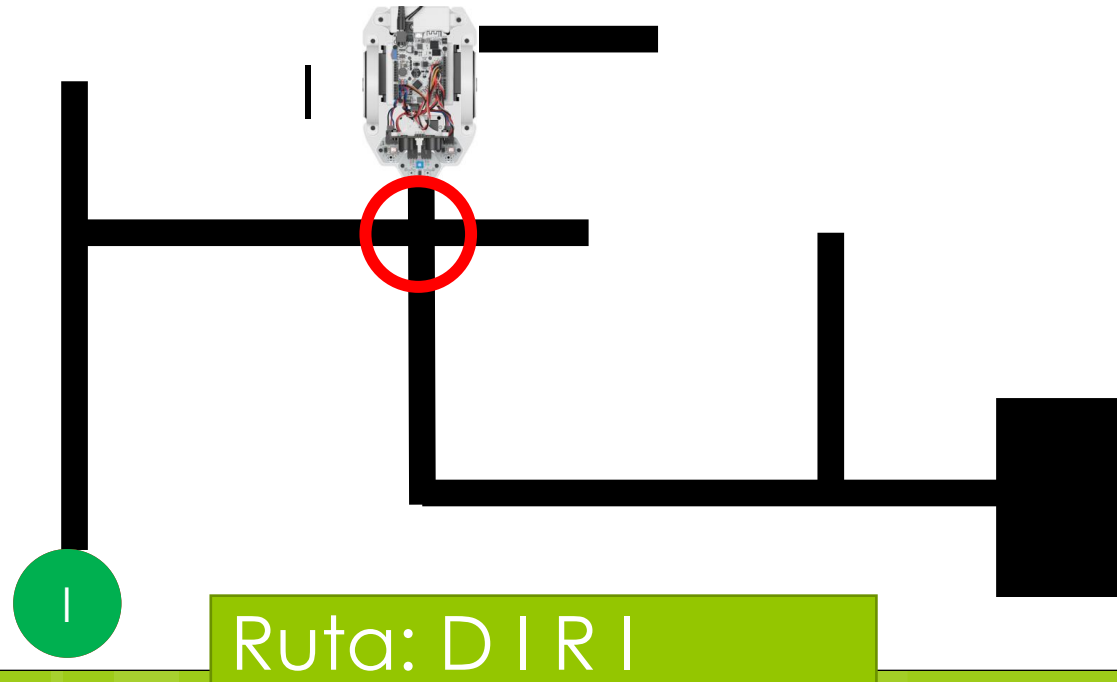
## Ejemplo VII

- Llega a un **camino sin salida**
- Guardamos en la memoria: R
  - R – Rotar / Retroceder



## Ejemplo VIII

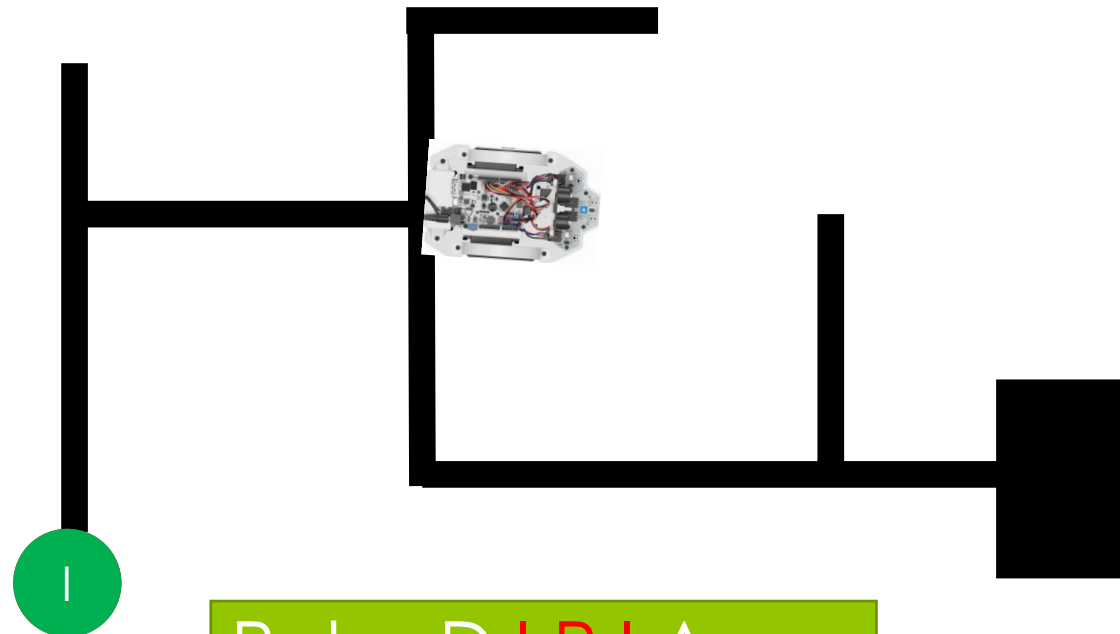
- Vuelve a la intersección
- Toma el camino de la izquierda
  - Guardamos en la memoria: I





## Ejemplo IX

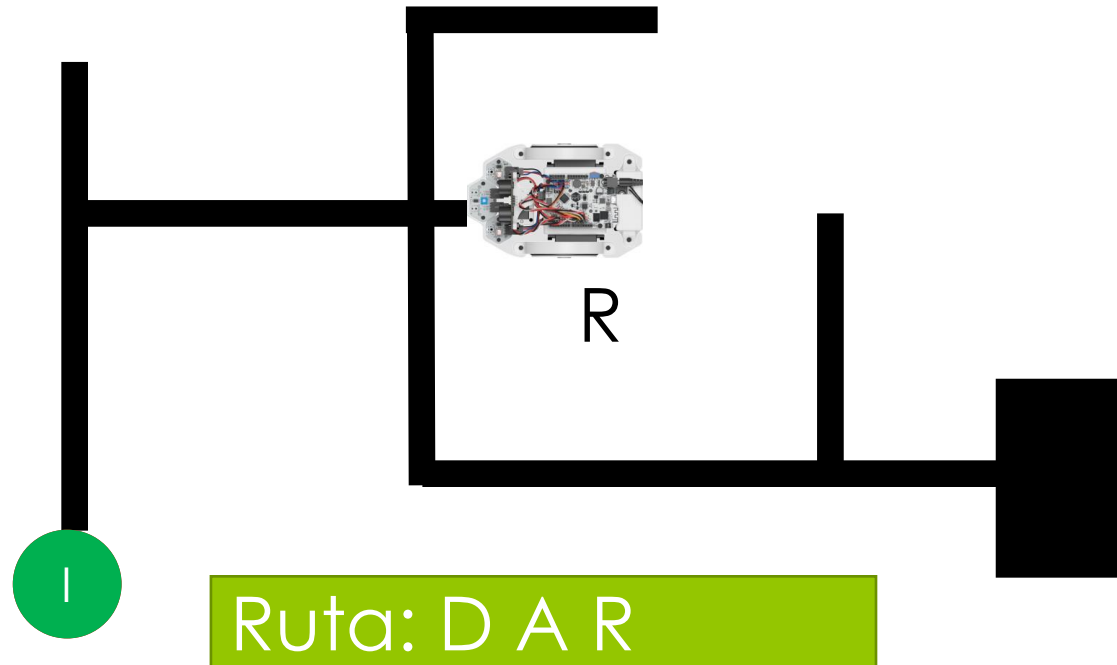
- Toma el camino de la Izquierda
  - «l R l» en memoria también indica que se ha confundido
  - Cambiamos «l R l» por A, debería haber ido hacia adelante



Ruta: D ~~IR~~A

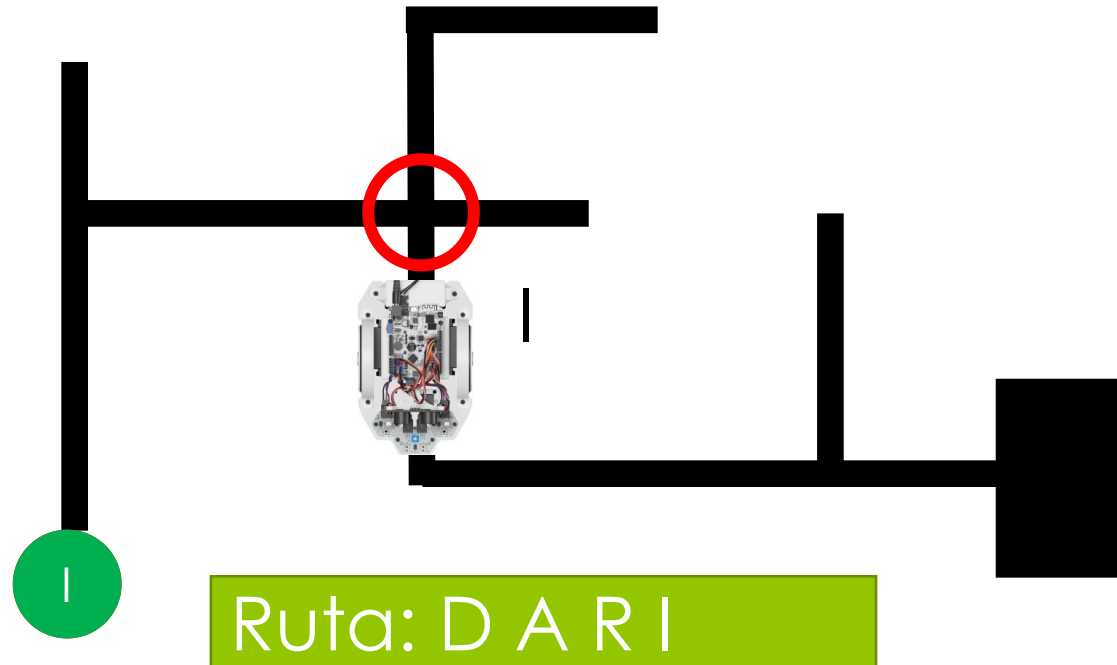
## Ejemplo X

- Llega a otro **camino sin salida**
  - Agregamos R a la memoria



## Ejemplo

- Vuelve a la **intersección** y toma el camino **a la izquierda**
- Agregamos I a la memoria



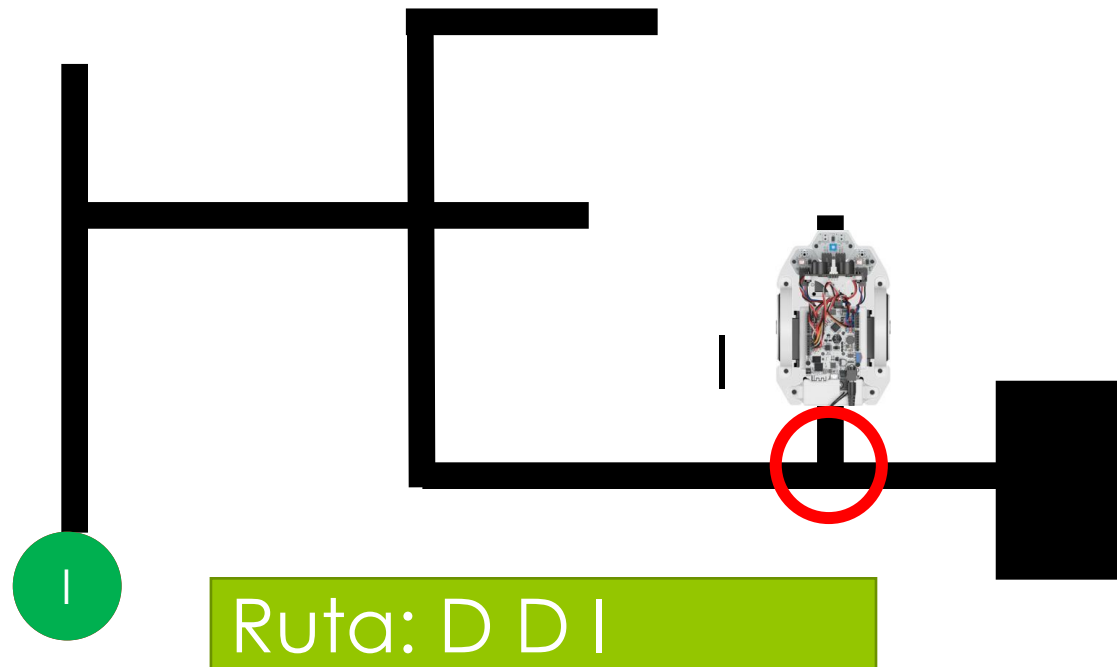
## Ejemplo XI

- ◉ «A R l» indica que se era un camino sin salida
- ◉ «A R l» se sustituye por D



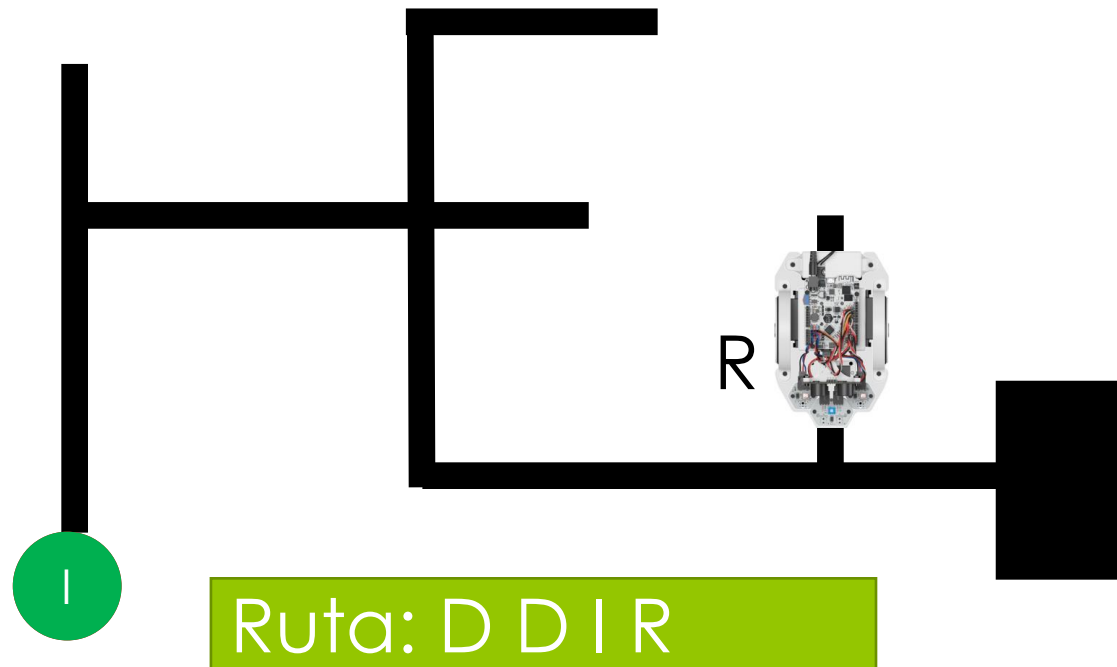
## Ejemplo XII

- En la nueva intersección toma el camino a la izquierda
- Agregamos I a la memoria



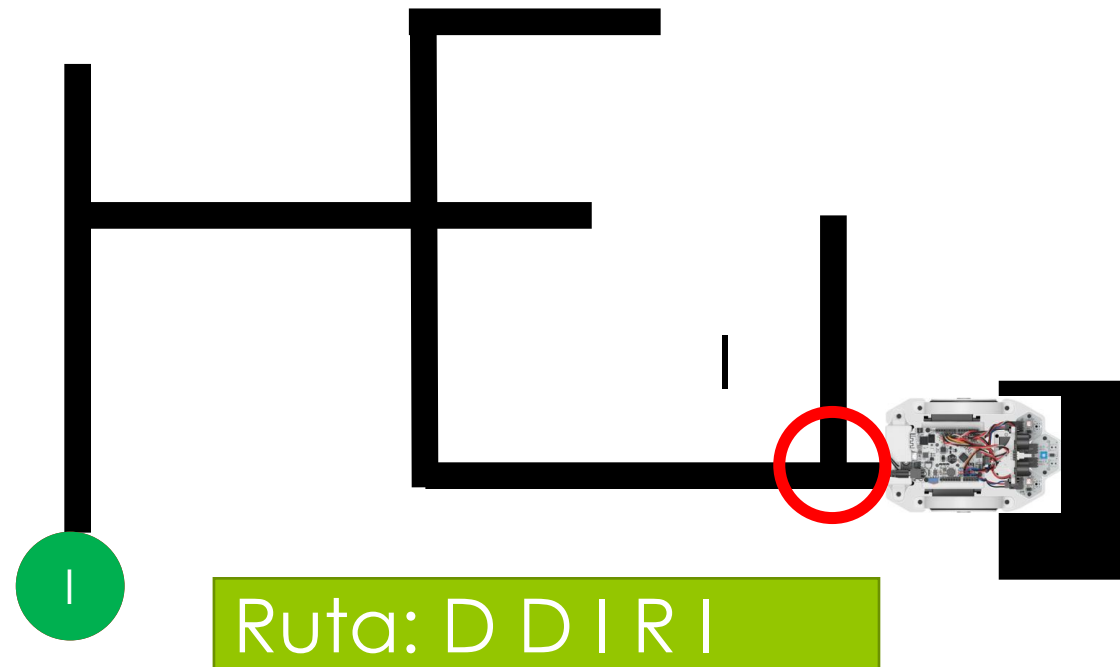
## Ejemplo XIII

- Es un camino sin salida, luego **retrocede**
  - Agregamos R a la memoria



## Ejemplo XIV

- Toma el camino a la izquierda
- Agregamos I a la memoria





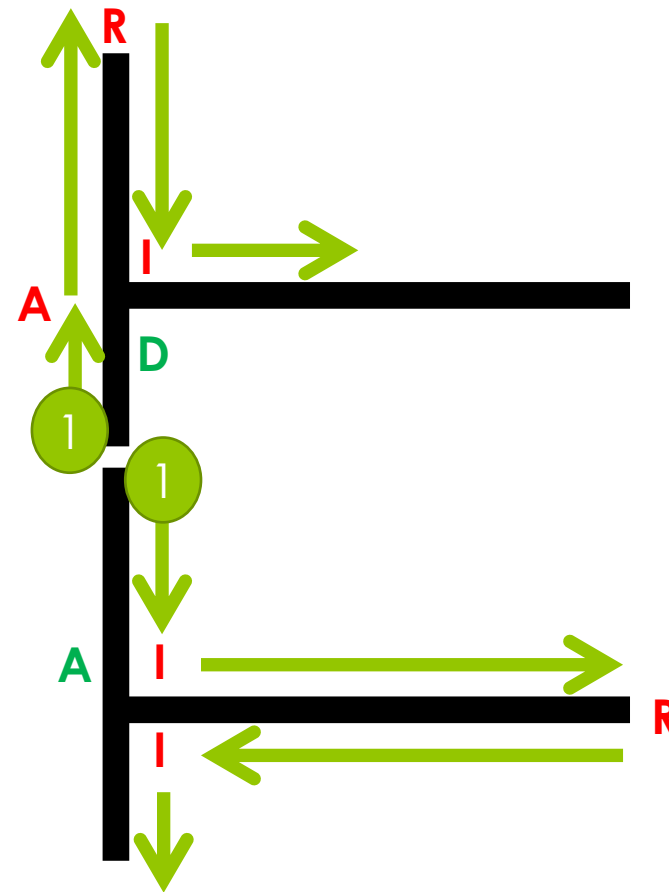




## Reglas de reemplazo I

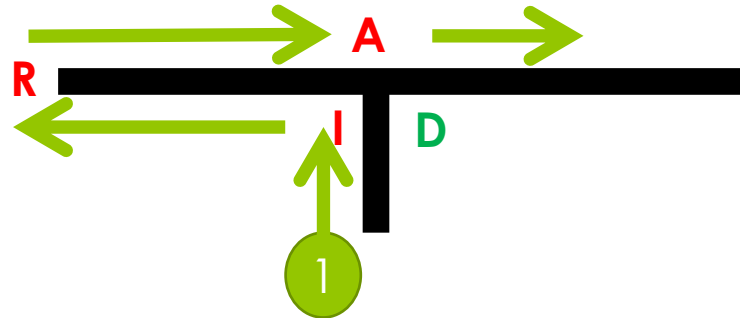
- A R I  $\rightarrow$  por D

- I R I  $\rightarrow$  por A



## Reglas de reemplazo II

- I R A -> por D



- Puede haber muchas más reglas
  - Todo **dependerá del laberinto y su tipo**
    - Ángulos rectos, circular, etc.
  - Y **de los sensores y software del robot**
    - Infrarrojos, visión por computador, etc.

# Exploración de caminos y búsqueda



Escuela de  
Ingeniería  
Informática  
Universidad de Oviedo



Universidad de Oviedo  
*Universidá d'Uviéu*  
*University of Oviedo*

Cristian González García  
gonzalezcristian@uniovi.es

Material original de Jordán Pascual  
Espada

v 1.2.2 Noviembre 2022