

# Robótica

# Práctica 1

Robótica Basada en Comportamiento:  
Controlador con Arquitectura Subsumida

**Autores:**

*Sergio Marcos Vázquez*

*Ángel Álvarez Rey*

# Índice

<b>1 - Contexto del problema</b>	<b>3</b>
<b>2 - Decisiones de diseño</b>	<b>4</b>
1. Escapar	4
2. Dirigirse hacia luz	4
3. Seguir paredes	5
4. Acercarse a pared	5
5. General	5
<b>3 - Problemáticas encontradas</b>	<b>7</b>

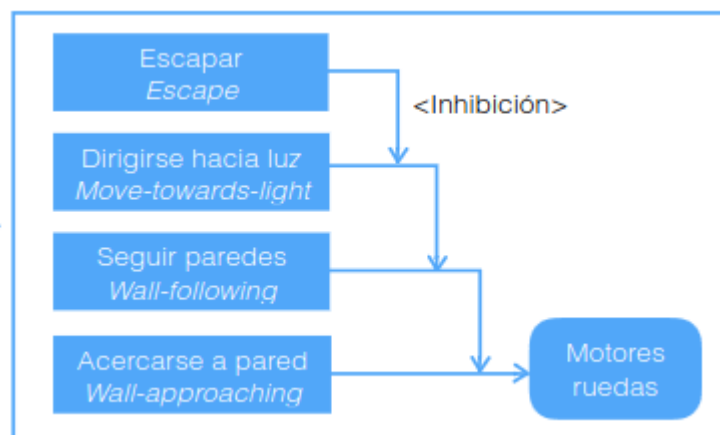
# 1 - Contexto del problema

El problema a tratar consiste en la implementación, en RobotC, de un sistema para el Robot Lego Mindstorms EV3 en donde procuramos un comportamiento basado en buscar y dirigirse hacia una luz partiendo de un entorno donde el robot no la ve inicialmente, utilizando para ello una aproximación de arquitectura subsumida.

Para obtener este comportamiento mediante dicha arquitectura se implementan 4 tareas/comportamientos individuales que finalmente se ejecutarán de manera conjunta. Las 4 tareas son:

1. Acercarse a pared
2. Dirigirse hacia la luz
3. Escapar (salir de situaciones de colisión)
4. Seguir paredes

Al realizar esto cada tarea se encargará de inhibir a aquellas que estén en niveles inferiores de la siguiente manera:



## 2 - Decisiones de diseño

Aquí explicaremos las decisiones de diseño e implementación tomadas tanto para la resolución general del problema como para la implementación individual de cada tarea.

### 1. Escapar

El robot escapará de situaciones de colisión, comprobando la proximidad con la pared mediante el sensor de ultrasonidos (umbral 5cm) y haciendo uso también del sensor de contacto para reconocer si el robot se ha chocado. Cuando se cumplen alguno de estos casos el robot deberá dar marcha atrás durante un segundo y girar 90° hacia la derecha. Es una solución simple que se podría haber mejorado haciendo otras comprobaciones (como por ejemplo que el robot no consiga girar o avanzar durante un tiempo), pero al final no se hizo por un motivo que ya aparece expuesto en el apartado de problemáticas encontradas.

### 2. Dirigirse hacia luz

Para dirigirnos hacia la luz lo primero que hacemos es obtener el valor de la luz ambiente para usarlo como umbral, y una vez se tomen más mediciones de la luz y se supere dicho umbral el robot tratará de acercarse hacia la fuente de luz. Para acercarse hacia dicha fuente el robot girará primero hacia la derecha midiendo continuamente el valor de luz ambiente y comprobando que dicho valor va subiendo. En el momento en que el valor obtenido baje, el robot cambiará el sentido del giro realizando más mediciones y continuará girando hasta que el valor baje de nuevo, donde el proceso se volverá a repetir. De esta manera, el robot estará orientándose y acercándose cada vez más al punto de mayor luz.

### 3. Seguir paredes

En esta tarea el robot emplea el sensor de ultrasonidos para detectar la pared, si el robot se encuentra a menos de 25cm de la pared realizará un giro en el sitio de  $135^\circ$ . Posteriormente avanzará haciendo un semicírculo de forma que el robot se vuelva a encontrar con una pared mas o menos al frente, lo que permite volver a ejecutar la función de nuevo.

### 4. Acercarse a pared

El robot se acercará a una pared yendo recto y a una velocidad constante. Una vez el sensor de ultrasonidos mide un valor inferior a 50cm, la velocidad del robot será igual a la medida por este sensor en cada instante, siendo así que cuánto más cerca esté más disminuirá la velocidad a la que se acerca.

### 5. General

Las tareas han de inhibirse unas a otras en el orden que se comentó en el contexto del problema. Para hacer esto se utilizan 3 semáforos de la siguiente manera:

1. Si la tarea de escapar activa su condición para ejecutarse se bloqueará el primer semáforo.
2. La tarea de seguir luz intenta bloquear el primer semáforo. Si no lo consigue porque el semáforo estaba bloqueado por otra tarea, entonces bloquea el segundo semáforo. Si puede bloquear el primer semáforo comprueba si se cumple la condición para ejecutar su comportamiento, y si se cumple la condición, bloquea el segundo semáforo y ejecuta dicho comportamiento.
3. De aquí en adelante tan solo es repetir este proceso, comprobar si el semáforo anterior estaba bloqueado, si es el caso bloquear el siguiente semáforo, si no es el caso comprobar la condición para ejecutar el comportamiento y bloquear el siguiente semáforo.

De esta manera se estará ejecutando todo el comportamiento de manera conjunta.


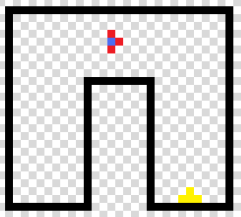
Además de esto contaremos con una condición de parada que se ejecutará cuando el robot alcance la luz. De primeras dicha condición se obtenía de la siguiente manera:

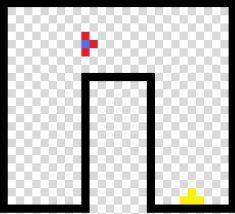
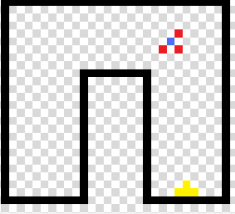
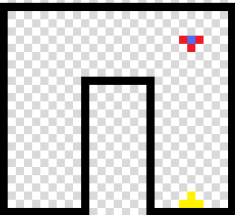
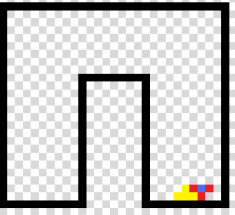
1. Obtener una primera medición de la luz ambiente que se usará como umbral.
2. Multiplicar dicho umbral por un valor constante.
3. Comprobar continuamente si la luz supera el umbral establecido.

Esta solución al final la cambiamos ya que era muy dependiente del valor obtenido inicialmente, además de el hecho de que multiplicar el valor daba umbrales muy dispares. Al final decidimos utilizar un umbral de luz fijo que mediante mediciones establecimos en 30, por lo que el robot parará sus motores cuando supere dicho umbral.

### 3 - Problemáticas encontradas

- En el caso de la función de “Escapar”, hemos comprobado que durante la ejecución del programa, cuando se activa la condición de entrada a dicha función, el robot no la realiza correctamente. Lo que hace el robot cuando entra en modo “Escapar” es ir hacia delante sin control o quedarse girando sin control durante unos instantes. No sabemos exactamente a que se puede deber este comportamiento, ya que, las ordenes que damos al robot son claras (ir hacia atrás durante 1 segundo y girar 90° hacia la derecha) pero la ejecución de las mismas por parte del robot dista bastante de lo esperado. Creemos que esto puede ser a causa de algún conflicto con los semáforos, pero no lo tenemos muy claro.
- En el caso de la tarea “Dirigirse hacia luz”, nuestra primera orientación para escoger el umbral que define el criterio de parada del robot era tomar una medida de luz al inicio del programa, la cual multiplicábamos por un valor (por ejemplo \*5) para obtener el umbral de luz que determinaría el criterio de parada de nuestro robot. Esto nos dió muchos problemas y quebraderos de cabeza a la hora de intentar buscar un valor por el cual multiplicar la luz medida inicialmente y así obtener buenos resultados. En la siguiente tabla aparece explicado detalladamente como se comportaba nuestro sistema para distintos valores de luz umbral.

Posición inicial del robot	Valor medido por el sensor de luz	Valor resultante de nuestra variable umbral (*5)	Posición final del robot cuando se supera el umbral
	1	5	

	4	20	
	15	75	 COLISIONA CON LA LUZ

Como podemos ver, el criterio de parada del robot varía muchísimo en función del punto de partida del mismo, por lo que finalmente hemos decidido definir un umbral manualmente para que el criterio de parada del robot no dependa de la medida inicial de luz del mismo.