UNIVERSIDADE DA CORUÑA

facultade de informática da coruña

Robótica Basada en Comportamiento: *Controlador con Arquitectura Subsumida*

Grao en Enxeñaría Informática - 4º curso
ROBÓTICA

Enunciado

- Esta práctica, consistirá en la implementación en RobotC de un comportamiento utilizando la aproximación de arquitecturas subsumidas.

- El comportamiento a implementar similar al que se verá en teoría como ejemplo de **arquitecturas subsumidas**: buscar y dirigirse hacia una luz en un entorno donde el robot no la ve inicialmente.

- No se pueden utilizar mapas.

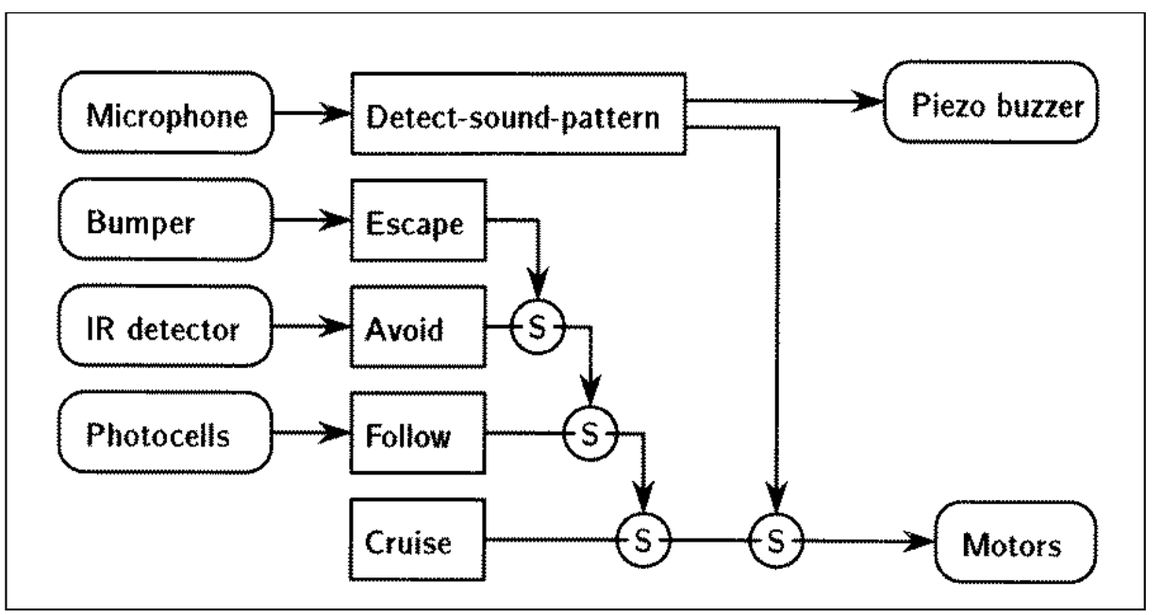
- Se aconseja probar primero todos los módulos que sean posibles en el simulador y después probarlos en el robot real EV3, ajustando lo que sea necesario.

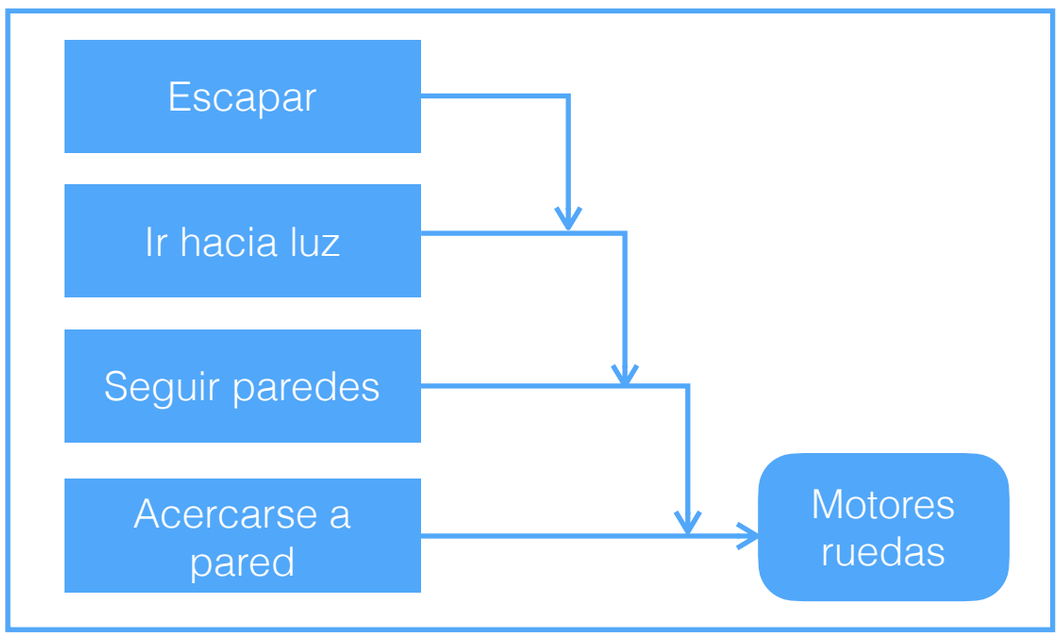
- Se aconseja realizar una calibración del sensor de luz para las condiciones de la habitación en el instante de ejecución (con robot real).

    - En el simulador no existen un elemento para representar una luz en el entorno. Puede realizarse una detección de color que luego se reemplace por la detección de luz en el robot real (aunque el comportamiento no será el mismo).

Enunciado

- Se partirá del siguiente controlador (que se verá en la clase de teoría) adaptándolo para:
  - Utilizar ultrasonidos en vez de infrarrojos.
  - Omitir el uso del micrófono.
  - Reemplazar “avoid” con un comportamiento de “wall following".
  - Intercambiar en la jerarquía “follow” con “avoid” (ahora “wall following”).





Enunciado

- Se sugiere implementar y probar aisladamente, y en este orden, los siguientes comportamientos antes de utilizarlos conjuntamente para producir el comportamiento global:

  - Acercarse a pared. Ir recto. Puede utilizarse otra estrategia más sofisticada.

  - Dirigirse hacia luz. Se debe de utilizar el sensor de color en modo luz ambiente.

  - Salir de situaciones de colisión. Se pueden utilizar el giroscopio, el sensor de rotación del motor, el sensor de ultrasonidos y el sensor de contacto.

  - Seguir paredes. Utilizando el sensor de ultrasonidos, al acercarse a un objeto seguir su contorno. Esto permite recorrer un entorno fácilmente siguiendo sus paredes, siempre y cuando no haya columnas (y no las habrá).

Enunciado

- Esta práctica se desarrollará utilizando RobotC.

- Cada comportamiento deberá de estar implementado como una tarea de RobotC.

- La sincronización entre tareas se realizará mediante variables globales y semáforos.

- Por coherencia con los principios de las arquitecturas subsumidas, usar una variable global diferente para coordinar la ejecución de cada dos niveles adyacentes en la arquitectura subsumida. De esta forma, una variable global solo podrá ser escrita por una tarea (la que implemente el comportamiento del nivel n) y leída por otra (la que implemente el comportamiento del nivel n+1).

    - Una tarea tiene la capacidad de inhibir la ejecución de la de nivel inferior y tiene que comprobar si su ejecución está inhibida por la superior.

Enunciado

- **Fecha y modo de entrega:**
  - En el Campus Virtual con fecha límite *16 de marzo a las 23:55*.
- **Entregables:**
  - Código fuente.
  - Una pequeña memoria, de 4 páginas sin contar portada, explicando:
    - Las decisiones de diseño que se hayan tomado que se consideren relevantes.
    - Los problemas que tiene la solución implementada (si es que los tiene), cuándo no funciona bien, por qué y cuál sería la solución.
- También habrá una ***defensa en horario de clase de prácticas***, consistente en ejecutar el comportamiento global en el robot real y en un entorno físico construido por el profesor.

Enunciado

- Si un grupo desea realizar algún cambio en esta práctica al respecto de:
  - Tarea a realizar.
  - Paradigma de arquitectura de control.
  - Lenguaje de programación.
  - Método de sincronización entre tareas.
- Debe de notificárselo al profesor el cual podrá aceptar o no la propuesta. En caso de aceptarla, ésta será publicada en el Campus Virtual.
- ***No se admitirá ninguna desviación de este enunciado si no ha sido aceptada explícitamente en el Campus Virtual.***

Uso de semáforos en RobotC

```
//Declaración de variables
TSemaphore semaphore;

//Inicialización
semaphoreInitialize(semaphore);

//Bloqueo y desbloqueo
semaphoreLock(semaphore);
if (bDoesTaskOwnSemaphore(semaphore))
{
   //doSomething
   semaphoreUnlock(semaphore);
}
```