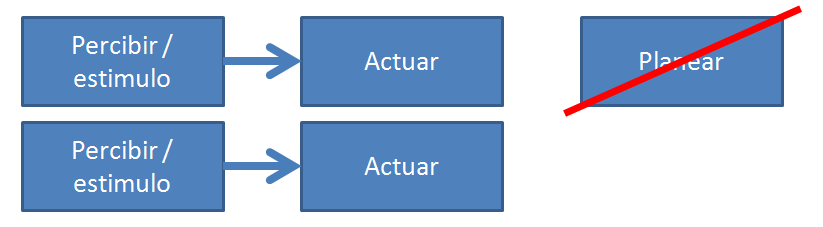
|  |
| --- |
| Software para Robots Jordán Pascual : pascualjordan@uniovi.es |

|  |  |
| --- | --- |
| 9 | Actividades |

# Actividad 1 (3 Puntos) - Sigue líneas

## 1.1 Seguir la línea simple

Implementar un controlador que permita seguir la línea. El controlador debe estar basado en un **paradigma reactivo simple**, para cada percepción realiza directamente una acción



Los paradigmas reactivos están inspirados en el sistema estimulo-respuesta de los animales. Ej, comportamiento de un insecto:

* No percibe comida -> vuela "aleatoriamente"
* Percibe comida -> vuela hacia la comida
* Percibe peligro -> escapa en sentido contrario al peligro

Existe una correspondencia entre lo que se **percibe** y como se **actúa.**  
Para cada tipo de **percepción** se realiza una **acción**.

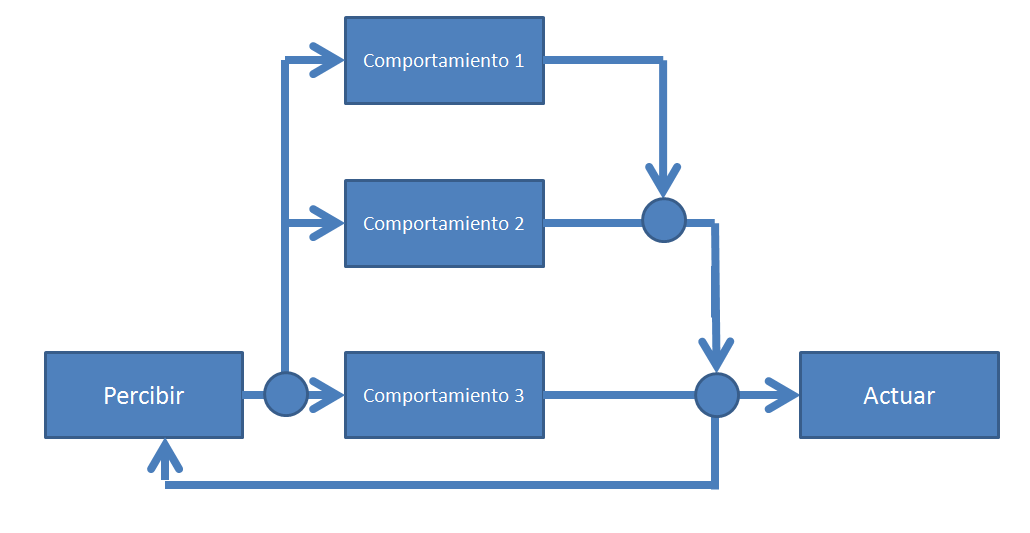
|  |  |
| --- | --- |
| Percepción | Acción |
| Sensor.L. derecho : LINEA  Sensor.L izquierdo : LINEA | Avanzar |
| Sensor.L. derecho : NO\_LINEA  Sensor.L izquierdo : LINEA | Rotar hacia la izquierda |
| Sensor.L. derecho : LINEA  Sensor.L izquierdo : NO\_LINEA | Rotar hacia la derecha |
| Sensor.L. derecho : NO\_LINEA  Sensor.L izquierdo : NO\_LINEA | Detenerse |

## 1.2 Encontrar el circuito

Implementar un controlador que permita seguir la línea. El controlador debe estar basado en un **paradigma reactivo basado en comportamientos**.

Cada comportamiento debe ser implementado en una función independiente. Cada comportamiento es capaz de realizar un conjunto de acciones más o menos complejas, puede tratarse de secuencias de acciones fijas o dinámicas, internamente el comportamiento puede utilizar memoria, estados o técnicas de planificación.

Los comportamientos deben ser totalmente independientes entre sí, la percepción obtenida del entorno (por los sensores) se envía a todos los comportamientos, a partir de esa percepción los comportamientos se activarán o no.



En algunos controladores cada percepción solo activa un comportamiento, en otros la misma percepción puede activar dos o más comportamientos. Dependiendo del objetivo del programa se utiliza una u otra técnica para resolver la colisión entre comportamientos:

* **Arbitraje:** en las técnicas por arbitraje los comportamientos compiten para ver cuál será la salida ejecutada. Algunos sistemas comunes dentro del arbitraje son la prioridad, selección o el voto.
* **Cooperativo**: mezcla las múltiples salidas de los comportamientos activos de forma coherente, para ello se pueden utilizar diversas técnicas: campos potenciales, operaciones vectoriales, algoritmos heurísticos.

|  |  |
| --- | --- |
| Comportamiento | Se activa con percepción |
| **Entrar en pista**  Buscar la línea realizando trayectoria en espiral (estoy fuera del circuito)    http://images.clipartlogo.com/files/images/26/266194/square-spiral_f | Sensor.L. derecho : NO\_LINEA  Sensor.L izquierdo : NO\_LINEA |
| **Corregir trayectoria**  Recolocarse en la línea ya que parte del robot está fuera (Por el lado derecho o izquierdo) | Sensor.L. derecho : NO\_LINEA  Sensor.L izquierdo : LINEA  ó  Sensor.L. derecho : LINEA  Sensor.L izquierdo : NO\_LINEA |
| **Avanzar**  Avanzar por la línea, en línea recta ya que el robot está | Sensor.L. derecho : LINEA  Sensor.L izquierdo : LINEA |

**Plantilla de código, que se debe usar obligatoriamente para realizar la actividad 1.2:**

|  |
| --- |
| struct Salida **{**  boolean activo **=** false**;**  int servoIzq **=** **-**1**;**  int servoDer **=** **-**1**;**  **};**  Salida sComp1 **;**  Salida sComp2 **;**  Salida sComp3 **;**  void setup**()** **{**  Serial**.**begin**(**9600**);**  **}**  void loop**()** **{**  int valorSensorIzq **=** random**(**0**,**2**);**  int valorSensorDer **=** random**(**0**,**2**);**  Serial**.**println**(**"Izq: "**+**String**(**valorSensorIzq**)+**" Der: "**+**String**(**valorSensorDer**)+**" -------------------------------------------------------------"**);**  delay**(**3000**);**      comp1\_EntratEnPista**(**valorSensorIzq**,**valorSensorDer**);**  **if** **(** sComp1**.**activo **){**  Serial**.**println**(**"Comp1 ACTIVO "**);**  Serial**.**println**(**"ServoIzq:"**+**String**(**sComp1**.**servoIzq**));**  Serial**.**println**(**"ServoDer:"**+**String**(**sComp1**.**servoDer**));**  // Sí este está activo hay que entrar en pista  **return;**  **}**    comp2\_CorregirTrayectoria**(**valorSensorIzq**,**valorSensorDer**);**  **if** **(** sComp2**.**activo **){**  Serial**.**println**(**"Comp2 ACTIVO "**);**  Serial**.**println**(**"ServoIzq:"**+**String**(**sComp2**.**servoIzq**));**  Serial**.**println**(**"ServoDer:"**+**String**(**sComp2**.**servoDer**));**  // Sí este está activo hay que entrar en pista  **return;**  **}**    comp3\_Avanzar**(**valorSensorIzq**,**valorSensorDer**);**  **if** **(** sComp3**.**activo **){**  Serial**.**println**(**"Comp3 ACTIVO "**);**  Serial**.**println**(**"ServoIzq:"**+**String**(**sComp3**.**servoIzq**));**  Serial**.**println**(**"ServoDer:"**+**String**(**sComp3**.**servoDer**));**  // Sí este está activo hay que entrar en pista  **return;**  **}**    **}**  void comp1\_EntratEnPista**(**int valorSensorIzq**,**int valorSensorDer**){**  sComp1**.**activo **=** false**;**  **if(** valorSensorIzq **==** 1 **&&** valorSensorDer **==** 1 **){**  sComp1**.**activo **=** true**;**  sComp1**.**servoIzq **=** 392**;** // Valores inventados  sComp1**.**servoDer **=** 234**;** // Valores inventados  **}**  **}**  void comp2\_CorregirTrayectoria**(**int valorSensorIzq**,**int valorSensorDer**){**  sComp2**.**activo **=** false**;**  **if(** valorSensorIzq **==** 0 **&&** valorSensorDer **==** 1 **){**  sComp2**.**activo **=** true**;**  sComp2**.**servoIzq **=** 111**;** // Valores inventados  sComp2**.**servoDer **=** 333**;** // Valores inventados  **}**  **if(** valorSensorIzq **==** 1 **&&** valorSensorDer **==** 0 **){**  sComp2**.**activo **=** true**;**  sComp2**.**servoIzq **=** 333**;** // Valores inventados  sComp2**.**servoDer **=** 111**;** // Valores inventados  **}**  **}**  void comp3\_Avanzar**(**int valorSensorIzq**,**int valorSensorDer**){**  sComp3**.**activo **=** false**;**  **if(** valorSensorIzq **==** 0 **&&** valorSensorDer **==** 0**){**  sComp3**.**activo **=** true**;**  sComp3**.**servoIzq **=** 4**;** // Valores inventados  sComp3**.**servoDer **=** 4**;** // Valores inventados  **}**  **}** |

|  |  |
| --- | --- |
| Tiempo entre ejecuciones del bucle Loop  Cuando incluimos mucho código en un programa las ejecuciones del bucle Loop tienden a hacerse más lentas. Esto puede provocar que en lugar de tardar 1ms entre ejecuciones del bucle loop pase a tardar mucho más (En el caso anterior hasta 44ms). Este retardo unido a que los motores no se paran nunca provoca que el motor continúa moviéndose durante esos 44ms sin ningún control (ya que los sensores no están leyendo, se está ejecutando otro código).  En el caso anterior lo que está aumentando considerablemente el tiempo entre ejecuciones son los **Serial.println** , podemos optar por eliminarlos del programa.   |  | | --- | | comp1\_EntratEnPista**(**valorSensorIzq**,**valorSensorDer**);**  **if** **(** sComp1**.**activo **){**  ~~Serial~~**~~.~~**~~println~~**~~(~~**~~"Comp1 ACTIVO "~~**~~);~~**  ~~Serial~~**~~.~~**~~println~~**~~(~~**~~"ServoIzq:"~~**~~+~~**~~String~~**~~(~~**~~sComp1~~**~~.~~**~~servoIzq~~**~~));~~**  ~~Serial~~**~~.~~**~~println~~**~~(~~**~~"ServoDer:"~~**~~+~~**~~String~~**~~(~~**~~sComp1~~**~~.~~**~~servoDer~~**~~));~~**  // Sí este está activo hay que entrar en pista  **return;**  **}** |   En algunos casos puede ser que no baste con eliminar los Serial.println (o puede que no nos interese hacerlo) . **¿Cómo podríamos hacer para que el programa no perdiese precisión a pesar del retardo entre ejecuciones?**   * **Mover el robot más despacio:** la opción más sencilla es haciendo que las ruedas se muevan muchísimo más despacio, de esta forma tratamos de “nivelar” la lenta frecuencia de actualización de los sensores con una velocidad lenta. Por ejemplo: **100 en lugar de 180 y 80 en lugar de 0.** * **Realizar pausas**: opción es hacer que el robot este parado mientras se ejecuta el código (para que los motores no avancen sin ningún control), cuando colocamos velocidades en los motores hacemos que mantenga esta velocidad durante un corto periodo por ejemplo 15ms y después paramos el motor (para que no continúe moviéndose mientras se procesa el código) esta opción hará que el robot arranque y se detenga en cada iteración, ira a ”saltos”. |

Código para ver el tiempo entre ejecuciones.

|  |
| --- |
| unsigned long tiempo **=** 0**;**  void setup**()** **{**  Serial**.**begin**(**9600**);**  **}**  void loop**()** **{**  // Tiempo entre ejecuciones  Serial**.**println**(**millis**()-**tiempo**);**  tiempo **=** millis**();**  **}** |

# Actividad 2 (0 Puntos) - Esquivar obstáculos

Ampliar el **controlador reactivo basado en comportamientos** realizado en el ejercicio anterior incluyendo un nuevo comportamiento "esquivar" que se active cuando detecta un obstáculo justo delante (Colocaremos un vaso de plástico en medio de la línea del circuito). En ese caso el robot debe abandonar la línea rodear el obstáculo y volver a entrar en el circuito.