ROBÓTICA BASADA EN COMPORTAMIENTO: CONTROLADOR CON ARQUITECTURA SUBSUMIDA

Nuria Iglesias Traviesa
Xoel González Pereira

Grupo de prácticas: 1.2

Robótica
Universidade da Coruña
Curso 2022/23

Índice

1. Introducción

Esta práctica consiste en la implementación en RobotC de cuatro comportamientos, en los que el robot realizará las siguientes acciones: escapar (salir de situaciones de colisión), dirigirse hacia la luz, seguir paredes (el robot se mantendrá a una distancia objetiva de la pared y seguirá su contorno) y acercarse a la pared (el robot caminará recto).

Para la implementación de estos comportamientos, se ha utilizado una aproximación de arquitecturas subsumidas. Se parte de un controlador, que ha sido adaptado para utilizar ultrasonidos en vez de infrarrojos, para usarlo como sensor de distancia, se omite el uso del micrófono, el comportamiento de avoid se substituye por el de seguir paredes y se intercambia en la jerarquía follow con avoid (seguir paredes actualmente).

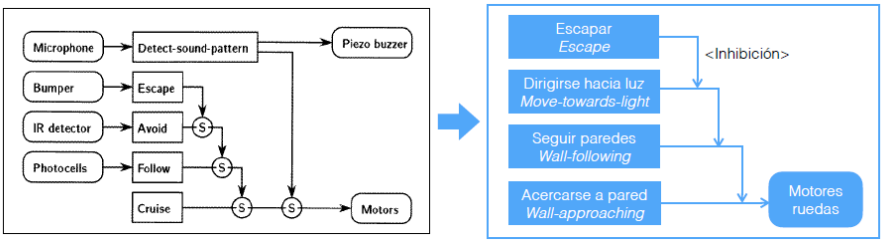

Figura 1: Arquitectura subsumida

2. Descripción del problema

Se plantea un comportamiento global en el robot LEGO EV3, en un entorno físico, el cual a través del uso de los sensores y de la arquitectura subsumida usada, se llegará hasta una luz que no se veía previamente.

El robot tiene acceso a varios sensores, los cuales fueron usandos en la práctica para implementar los distintos comportamientos. Entre ellos se encuentran:

1. Sensor de ultrasonidos: para medir la distancia hasta la pared o hasta ciertos obstáculos.
2. Color Sensor: detecta la luz del entorno en un espacio físico, mientras que el color en un ambiente virtual
3. Gyro Sensor: permite saber los grados que gira el robot físico de manera más concreta, de esta manera se puede controlar mejor .

4. Touch Sensor: cuando es pulsado sucede debido a una colisión, de esta forma controlamos que no se quede chocando en un bucle infinito.

3. Decisiones de diseño

Para el diseño de los comportamientos, se han implementado de la siguiente forma:

1. Escapar (salir de situaciones de colisión): se han usado el sensor de contacto (touchSensor) y el sensor de ultrasonidos (sonarSensor) para la detección de la colisión. Cuando el robot colisione contra una pared o esté a una distancia muy cercana, este retrocederá y hará un pequeño giro para escapar de la colisión. El motivo del uso de dos sensores es debido a que el sensor de contacto no siempre detecta correctamente las colisiones, ya que hay casos en los que el robot no choca de frente contra el obstáculo, por lo que se usa el sensor de ultrasonidos para lidiar con el problema.

```
// Si se detectó una colisión, escapar
if (getTouchValue(touchSensor) == 1 ||  getUSDistance(sonarSensor) < 10 ) {

    semaphoreLock(semaphoreIrrecto);
    inhibitedirrecto = false;
    semaphoreUnlock(semaphoreIrrecto);
    setLEDColor(ledOrange);

    clearTimer(T1);
    while (time10[T1] < 100) {
        setMotorSpeed(leftMotor, -20);
        setMotorSpeed(rightMotor, -20);
    }
    resetGyro(gyroSensor);
    setMotorSpeed(leftMotor, 20);
    setMotorSpeed(rightMotor, -20);
    while (getGyroDegrees(gyroSensor) < 70) {}
```

Figura 2: Lógica de detección de colisiones

2. Dirigirse hacia la luz: hemos seguido la implementación a través de la búsqueda del valor mínimo de la luz. El robot realiza un pequeño giro en busca de la luz, una vez este valor empieza a decrecer, implica que la luz está en esa dirección, por lo que avanza hacia delante.
   También hemos definido un valor máximo para saber cuándo consideramos que ya ha llegado hasta la luz.

```
if (getColorAmbient(colorSensor) > 10) {

    semaphoreLock(semaphoreSeguirparedes);
    inhibitedSeguirParedes = false;
    semaphoreUnlock(semaphoreSeguirparedes);

     setLEDColor(ledOrangeFlash);
    while (getColorAmbient(colorSensor) <= luz) {
        luz = getColorAmbient(colorSensor);
        setMotorSpeed(leftMotor, 20);
        setMotorSpeed(rightMotor, -20);

        if(getColorAmbient(colorSensor) >= luz){
          setMotorSpeed(leftMotor, -80);
        setMotorSpeed(rightMotor, 80);
      }
    }

    setMotorSpeed(leftMotor, 20);
    setMotorSpeed(rightMotor, 20);

    if (getColorAmbient(colorSensor) > umbral) {
        setMotorSpeed(leftMotor, 0);
        setMotorSpeed(rightMotor, 0);
    }
```

Figura 3: Lógica de detección de luz

3. Seguir paredes: Utilizando el sensor de ultrasonidos, si se detecta que el robot está a menos de una determinada distancia del obstáculo, este primero hará un giro buscando la distancia mínima a la que está de la pared, guardando esta cada vez que decrezca. Una vez se haya encontrado la distancia más pequeña, el robot girará hacia ese lado, poniéndose así en paralelo a la pared.
   Una vez en paralelo, con la ayuda de un Timer se hace que el robot camine recto durante dos segundos, y depués que compruebe si se ha alejado de la pared que está siguiendo en paralelo. Si se ha alejado, el robot recuperará la posición en la que estaba. Además, mientras está caminando en paralelo a la pared, también se valora el caso de que tenga una nueva pared de frente, y en el caso de ser así, el robot girará para seguir ahora en paralelo el nuevo obstáculo.

```
resetGyro(gyroSensor);
setMotorSpeed(leftMotor, 20);
setMotorSpeed(rightMotor, -20);
while (getGyroDegrees(gyroSensor) < 70) {}
clearTimer(T1);
while (time10[T1] < 200) {
    setMotorSpeed(leftMotor, 20);
    setMotorSpeed(rightMotor, 20);
    if(getUSDistance(sonarSensor)<= 15  ){
resetGyro(gyroSensor);
setMotorSpeed(leftMotor, 20);
setMotorSpeed(rightMotor, -20);
while (getGyroDegrees(gyroSensor) < 70) {}
  }
}
resetGyro(gyroSensor);
setMotorSpeed(leftMotor, -30);
setMotorSpeed(rightMotor, 30);
while (getGyroDegrees(gyroSensor) > -60) {}
if (getUSDistance(sonarSensor) > 25) {
    setMotorSpeed(leftMotor, 50);
    setMotorSpeed(rightMotor, 50);
}
if (getUSDistance(sonarSensor) < 20) {
    setMotorSpeed(leftMotor, -50);
    setMotorSpeed(rightMotor, -50);
}
```

Figura 4: Lógica de seguir paredes

4. Buscar pared: implementamos un código como base para que el robot camine de manera recta, hasta que el sensor de ultrasonidos detecte un obstáculo (la pared). Al estar en la última posición de nuestra arquitectura subsumida, siempre entrará otro comportamiento.

```
if (inhibitedirrecto) {

        setLEDColor(ledGreen);
    if (getUSDistance(sonarSensor) > 25) {
        setMotorSpeed(leftMotor, 30);
        setMotorSpeed(rightMotor, 30);
    }
}
```

Figura 5: Lógica de buscar pared

4. Arquitectura

Para lograr que los comportamientos funcionen de forma compartida, se han creado cuatro variables globales, una por comportamiento, en los que se comprueba si la tarea está activa o no. También se han creado cuatro semáforos, uno por comportamiento. Así, en cada comportamiento se sigue la siguiente lógica:

1. Bloqueo del semáforo del comportamiento

2. Comprobación de que la tarea está activa.

   a) Desbloqueo del semáforo del comportamiento

   b) Leer sensorización necesaria para detectar si la tarea tiene que realizarse según el estado del entorno

   c) En caso de tener que actuar, se inhibirá la tarea de nivel inferior y se actuará. Si no tiene que actuar, se desinhibirán las tareas de niveles inferiores.

3. Si la tarea no está activa, se desbloquea el semáforo del comportamiento actual y se inhibe la tarea de nivel inferior.

Es importante destacar que la inhibición es en cascada, es decir, una tarea inhibe a su tarea de nivel inferior, y así sucesivamente. Así, la tarea de mayor prioridad inhibe a todas las demás, y la tarea de menor prioridad no inhibe a ninguna. Finalmente, se inicializan los cuatro semáforos en la tarea principal (task main), y se llama a los cuatro comportamientos en la prioridad indicada.

5. Peculiaridades de la práctica

Debido a varias causas, entre ellas que el robot físico en el que se ha ejecutado la implementación funciona muchas veces de manera distinta al robot virtual y los valores de los sensores son bastante sensibles a cambios, nuestro robot no siempre ejecuta los comportamientos de manera correcta.
Por ejemplo, a la hora de ejecutar el comportamiento de seguir paredes, el sensor de ultrasonidos no siempre detecta bien la distancia a la pared, por lo que a veces cuando está en ese comportamiento, se acerca lo suficiente a la pared (sin medir adecuadamente el valor del sensor de ultrasonidos), acaba entrando en evitar colisiones, para más adelante volver a seguir pared.
Además, no siempre que es pulsado el touch sensor se detecta correctamente, por lo que no siempre ejecuta el comportamiento de evitar colisiones en el instante que debería, sino que tarda algo más de lo correcto.