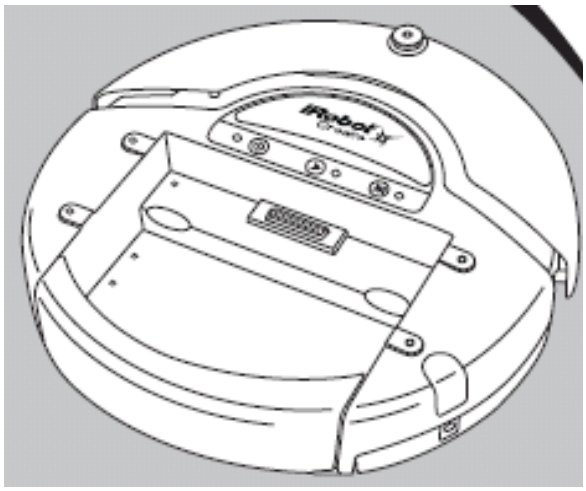
PRÁCTICA 4

Técnicas de Driving: Guiado mediante marcas y evitación de obstáculos

OBJETIVOS

Programar un algoritmo de navegación por marcas (mediante el seguimiento de una línea negra marcada en el suelo) y de evitación de obstáculos, para iRobot Create



MATERIAL NECESARIO

Hardware
- iRobot Create
- Raspberry Pi2

Software
- NetBeans
- Archivo Workspace_iCreate.zip con el código C/C++
- Archivo src_ejemplo.zip con el código de ejemplo del sigue_línea.cpp

Manuales (disponibles en eGela)
- Guía del usuario: iRobot Create Owner's Guide
- Manual del lenguaje de comandos: iRobot Create Open Interface
- Lista de funciones iRobotConnection

1. Puesta en marcha

a. Crear un nuevo proyecto a partir del archivo "Framework_Practica4.zip"
b. Seleccionar los estándares C11 y C++11, enlazar la librería wiringPi
c. Seleccionar como host vuestra Raspberry Pi 2.

Además de los fuentes y headers de la práctica, el contenido de proyecto debería ser el programa principal de ejemplo: Sigue_linea.cpp

2. Programa de Ejemplo

El ejemplo en **Sigue_linea.cpp** contiene un algoritmo básico para iCreate que hace uso del sensor de acantilado frontal izquierdo para seguir una línea por el suelo del

laboratorio. El programa principal se encuentra en el fichero **Sigue_linea.cpp** y usa varias funciones encapsuladas en la clase **ControlRobot.h/cpp**.

Compilar y ejecutar el programa ejemplo **Sigue_linea.cpp**.

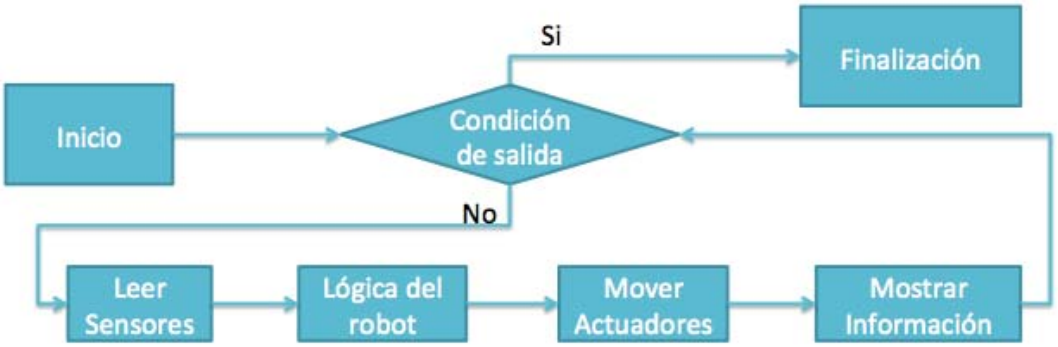


Figura 1. Algoritmo principal

a. La máquina de estados que usa iRobot Create en el ejemplo es la siguiente:

L	FL	FR	R	W	ESTADO ANTERIOR	NUEVO ESTADO
-	0	-	-	-	INICIAL	BUSCA: {Girar derecha}
-	1	-	-	-	INICIAL	SIGUE: {Avanzar}
-	0	-	-	-	SIGUE	BUSCA
-	1	-	-	-	SIGUE	SIGUE
-	0	-	-	-	BUSCA	BUSCA
-	1	-	-	-	BUSCA	SIGUE

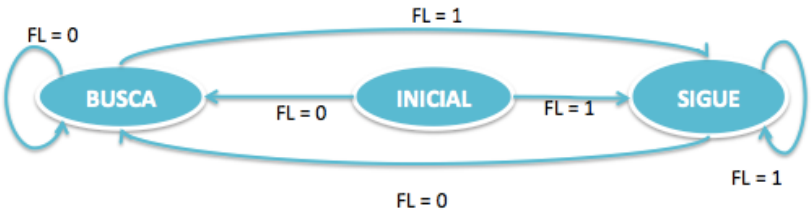


Figura 2. Máquina de estados

3. Programación en C

Para cambiar el comportamiento del robot hay que modificar los ficheros **ControlRobot.h** y **ControlRobot.cpp.** No es necesario cambiar **sigue_linea.cpp**:

a. Cambios en los sensores:
   - Modificar la estructura ***Sensores_iCreate*** en ControlRobot.h
   - Modificar el método ***void ControlRobot::leerSensores()*** en ControlRobot.cpp.
   - Algunos sensores de iRobot, por ejemplo, el sensor de distancia y el de ángulo, requieren ser inicializados para funcionar correctamente. Para ello hay que leerlos en la inicialización (ignorando su respuesta).

b. Cambios en la máquina de estados:
   - Incluir un ***#define NUEVO_ESTADO valor*** por cada nuevo estado.
   - Modificar el método ***void ControlRobot::logicaEstados()*** en ControlRobot.cpp

c. Cambios en los actuadores:
   - Modificar la estructura ***Actuadores_iCreate*** en ControlRobot.h
   - Modificar el método ***void ControlRobot::logicaEstados()*** en ControlRobot.cpp
   - Usar macros **#define** para los estados de los actuadores

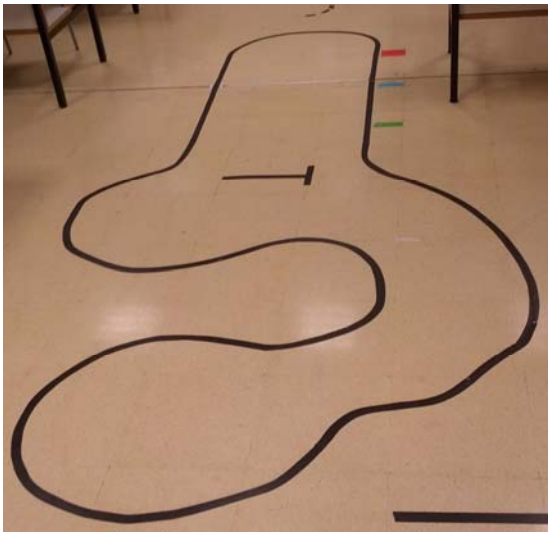
d. Cambios en la función de imprimir por pantalla:
   - Modificar el método ***void ControlRobot::imprimirInfo*** en ControlRobot.h

e. Otras modificaciones en el comportamiento del robot pueden requerir:
   - Modificar el método ***void ControlRobot::inicialización*** en ControlRobot.h
   - Modificar el método ***void ControlRobot::condiciónSalida*** en ControlRobot.h
   - Modificar el método ***void ControlRobot::finalización*** en ControlRobot.h

4. Realización de la práctica

a. Primera parte: Guiado por seguimiento de marcas

- Teniendo en cuenta la diferente sensibilidad de cada sensor de acantilado del iRobot (calibrada en la práctica anterior) hacer que el iRobot siga una raya marcada en el suelo con cinta aislante.
- Podeis usar 2, 3 o 4 sensores de barranco, según os convenga.



Calificación

Para calificar esta parte se realizará una prueba del seguimiento de un circuito de cinta aislante (que no tiene por qué ser el del laboratorio) en ambos sentidos. La puntuación de esta parte será inversamente proporcional al tiempo empleado en recorrer el circuito. Si un robot se sale del circuito el tiempo se considerará ∞.

Sugerencias

- Tened en cuenta que las comunicaciones con el PC alargan el ciclo y pueden hacer que la frecuencia de muestreo de los sensores baje y, por lo tanto, que se pierda el seguimiento de la línea). Para evitarlo, sacad por pantalla solamente los datos que os ayuden a verificar cómo funciona el algoritmo y cuando ya no sean necesarios comentad esas instrucciones para que no se ejecuten. Si se usan los LEDs del robot para recibir feedback de la actividad de los sensores, se ahorran comunicaciones con el PC.
- Para evitar el tráfico innecesario de mensajes no enviar comandos drive/drive direct si no hay cambios en la velocidad/dirección.

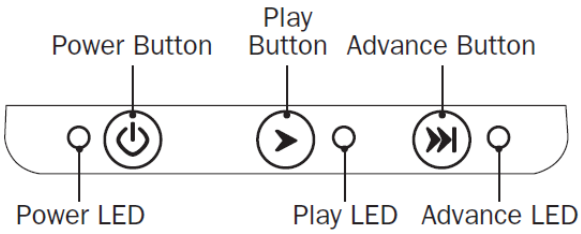
b. Segunda parte: Evitación de obstáculos

- Cuando el robot, mediante los *bumpers*, detecte un obstáculo (un ladrillo) bloqueando la línea, deberá rodearlo.
  - Haciendo un rodeo mediante *dead reckoning* y/o
  - Rodeando el obstáculo mediante el sensor de pared (*wall*). Esta técnica puntúa el doble que la anterior.
- Después de superar el obstáculo el robot debe seguir recorriendo el circuito.

Calificación

Para la evaluación de esta parte se realizará una prueba de evitación de un ladrillo.

c. Inicio y final del programa:



- Ya que el robot va a estar lejos del PC, utilizad el sensor del botón PLAY para que el robot se ponga en marcha (sin necesidad de hacerlo desde el PC).
- Aunque los programas en sistemas empotrados no finalizan nunca, en la fase de pruebas necesitamos detenerlo cuando queramos. Por ello, utilizad el sensor del botón de ADVANCE para detener el programa del robot, actuando sobre la condición de salida de la máquina de estados. Para ello, será necesario modificar el algoritmo para que a la finalización se desactiven los actuadores.

5. Informe a presentar

1. Código fuente de los programas en un fichero .zip obtenido mediante el comando exportar proyecto. [Puntúa la legilibilidad de los programas]
2. Descripción detallada de la solución diseñada para el seguimiento de la línea, incluyendo una explicación de la máquina de estados utilizada, el número de sensores de barranco utilizado y por qué habéis seleccionado ese número.
3. Descripción detallada de la solución diseñada para la evitación de obstáculos, incluyendo la técnica que habéis utilizado y por qué la habéis seleccionado.