eman ta zabal zazu

Informatika Fakultatea, EHU
Konputagailuen Arkitektura eta Teknologia Saila

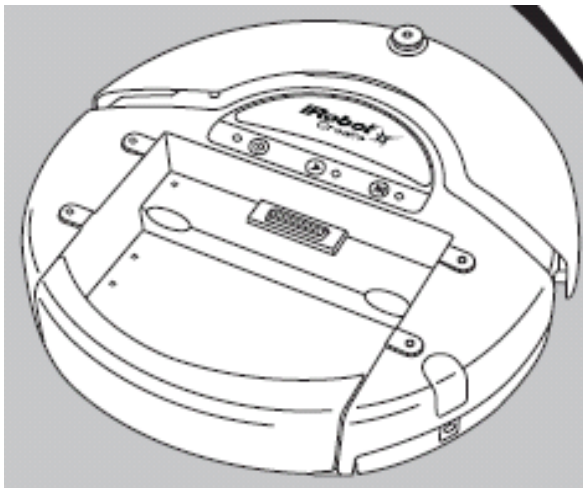
LABORATORIO DE ROBÓTICA

ROB 2017-18

PRÁCTICA 3
Introducción al iRobot Framework
Lectura y calibración de los sensores del iRobot

OBJETIVOS

- Aprender a usar el ***iRobot Framework***[1]
- Programar la lectura de los sensores de iRobot Create y su calibración en lenguaje C/C++.



MATERIAL NECESARIO

Hardware
- iRobot Create
- Raspberry Pi2

Software
- NetBeans
- iRobot Framework

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

Para instalar y verificar el iRobot_Framework seguir el procedimiento detallado en Instalacion iRobot_Framework.pdf disponible en eGela.

2. Programación de sensores del iRobot Create

La práctica tiene por objetivo probar y calibrar los distintos sensores que existen en el robot iCreate. Para ello la lista de funciones de iRobotConnection dispone de la instrucción "**int IRobotConnection::updateSensor (char sensorId)**". Los códigos **sensorId** se encuentran en el fichero **iRobotInstructionSet.h** dentro del namespace **iRobotSensors**. Si por ejemplo,

[1] La versión inicial del iRobot Framework fue creada por Gorka Montero como proyecto fin de carrera en 2012. En 2015 fue modificada por Borja Gamecho para funcionar sobre Raspberry Pi2.

queremos consultar el modo de funcionamiento en el que se encuentra iCreate, usaremos la siguientes instrucciónes:

```
int  modo = robot.updateSensor(iRobotSensors::OIMODE);
cout << modo << endl;
```

Este código nos devolverá los siguientes valores:

- 1 si estamos en el modo Passive
- 2 si estamos en el modo Safe
- 3 si estamos en el modo Full

Tenemos que tener en cuenta que el método **updateSensor** siempre devuelve un **int** (entero con signo de 16 bits) donde se guarda el valor que recibe del robot iCreate (a través de Open Interface), dónde puede tener otro formato. Por ejemplo, si se leen los sensores *Bumps & WheelDrops* hay que interpretar bit a bit, mediante una máscara de bits, el entero de 16 bits que devuelve el framework.

Sensor	**Codificación OI iRobot**
Bumps & WheelDrops	Binaria (usar máscara de bits)
Wall, Cliffs x 4, Virtual Wall	1 bit value (el menos significativo)
Infrared	1 Byte [0,255]
Distance, Angle, Requested Radious	Signed 16 bit value [-32768 - 32768]
Wall Signal, Cliffs Signal x 4	Unsigned 16 bit [0 – 4095]
Requested Velocity x3	Signed 16 bit value [-500, 500]

Hay más información sobre los sensores disponibles en las páginas 24 y 25 del manual Create_Open_Interface_v2 (disponible en eGela).

3. Práctica

Nota: En todos los programas se valorará positivamente la cantidad y calidad de la información que devuelva el programa sobre los valores de los sensores que va leyendo.

3.1 Calibración de sensores de choque y caída de ruedas (Bumps & WheelDrops):

- Diseñar un programa que reaccione a los cambios en los Bumpers emitiendo una señal audible diferente
- Diseñar otro programa que reaccione a los Wheeldrops, encendiendo alguno de los leds del robot

3.2 Calibración de sensores de barranco (Cliffs Signal):

- Diseñar un programa capaz de recoger 5 muestras de cada sensor de barranco sobre la cinta aislante negra y sobre el suelo del laboratorio.
- Para cada sensor calcular los valores medios de lectura de cinta y suelo.
- Definir un umbral de suelo y un umbral de cinta que permitan diferenciar el suelo y la cinta aislante.

3.3 Calibración del sensor de distancia a la pared (Wall):

- Recoger 10 muestras del sensor de pared variando la distancia. Encontrar el rango de distancias en las que el sensor devuelve 1 (wall seen).

3.4 Calibración de sensores de distancia para Dead-Reckoning:

- Diseñar un programa que haga avanzar el robot recorriendo un cuadrado usando los sensores de distancia y ángulo. El robot terminará en la posición inicial.
  Antes de empezar, el programa debe pedir al usuario el lado del cuadrado en centímetros y la dirección de los giros (derecha/ horario/ dextrógiro o izquierda/ antihorario/ levógiro).
  El programa debe compensar los errores para que el robot acabe exactamente en el mismo punto y con la misma orientación, para cualquier tamaño del cuadrado.
  Al terminar hacer una demostración ante el profesor.

4. Informe a presentar

1. Código de los programas y ejecutables.
2. Tablas con los valores de los sensores medidos.
3. Comentarios e incidencias.