eman ta zabal zazu

Informatika Fakultatea, EHU
Konputagailuen Arkitektura eta Teknologia Saila

LABORATORIO DE ROBÓTICA

ROB 2017-18

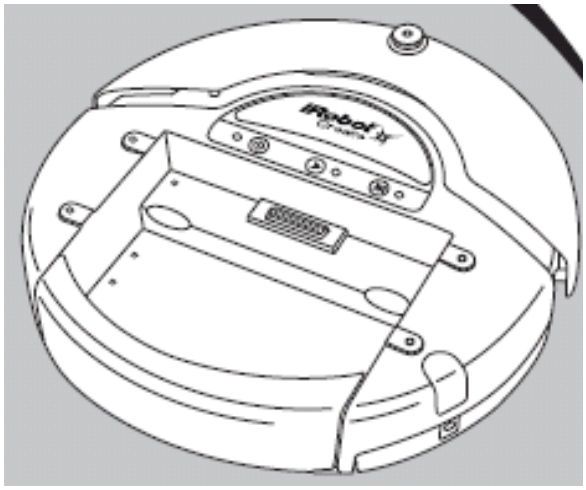
PRÁCTICA 2

Conexión Raspberry Pi 2 - iRobot Create y calibración de sus sensores mediante scripts

---

OBJETIVOS

- Probar y entender la conexión de la Raspberry Pi2 con el iRobot (alimentación y línea serie).
- Programar el iRobot mediante scripts desde la Raspberry Pi2.
- Calibrar el iRobot para la navegación por dead reckoning



MATERIAL NECESARIO

Hardware

- iRobot Create
- Raspberry Pi2
- Cables de alimentación y conexión serie Raspberry Pi2 <---> IRobot

Software

- NetBeans

Manuales (en eGela)

- Guía del usuario: iRobot Create Owner's Guide
- Manual del lenguaje de comandos: iRobot Create Open Interface.

1. Puesta en marcha

a. Conectar los cables de alimentación y línea serie entre Raspberry Pi2 e iRobot.
b. Encender el iRobot y comprobar que se la Raspberry Pi2 se enciende.

2. Programación con scripts

El programa **enviarls.cc** lee los enteros contenidos en el fichero **myscript.txt** y se los envía como caracteres por la línea serie al iRobot. Se detine cuando lee -1.
De esta manera, para enviar comandos o scripts al iRobot basta con meterlos en el fichero myscripts.txt terminando en -1 y ejecutar enviarls.cc.

Precauciones:

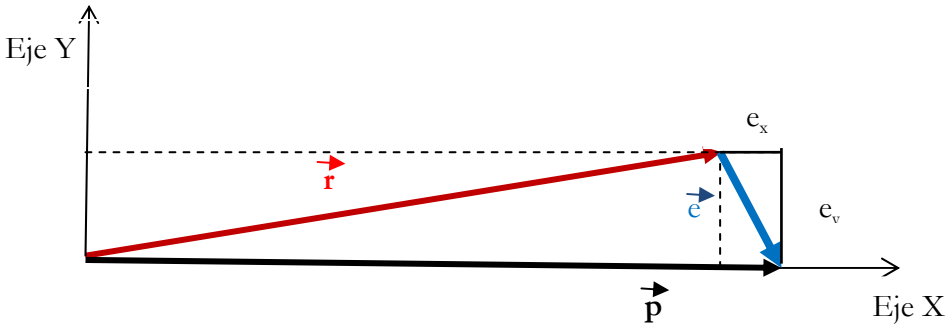
- Hay que pasar a enviarls.cc el *path* completo de myscript.txt (dentro de "Run" en las propiedades del proyecto).
- Recordad que todo script debe empezar por 152 y que para que se ejecute hay que enviar un 153.
    - Por ejemplo: 152 14 145 0 80 0 80 155 20 128 132 145 0 00 0 00 153 -1

Probar que funciona incluyendo en el fichero myscript.txt los siguientes scripts que aparecen en la página 15 del manual iRobot Create Open Interface:

- Drive 40 cm and stop
- Toggle led and bump
- Drive in a square

3. Odometría en línea recta

- Programar, utilizando scripts, un recorrido en línea recta de 150 cm.
    a) Utilizando un comando de espera por tiempo
    b) Utilizando un comando de espera por distancia
- Hacer tres pruebas para cada caso.
- Marcar el punto de salida y el punto de llegada con un rotulador no permanente y medir en cada caso, con un metro, el error X, y el Error Y, tal como se ve en la siguiente figura:



$\vec{e} = \vec{p} - \vec{r}$
p: distancia objetivo
r: distancia recorrida
e: vector de error

- Completar la siguiente tabla de errores absolutos, donde **d** = $|\vec{e}|$ (por tanto, se debe cumplir que $e_x^2 + e_x^2 = d^2$)

Tipo de Evento	**Velocidad**	**Prueba 1**			**Prueba 2**			**Prueba 3**			**Valores medios**		
		e_x	**e_y**	**d**	**e_x**	**e_y**	**d**	**e_x**	**e_y**	**d**	**e_x**	**e_y**	**d**
Espera por Tiempo	200 mm/s												
	500 mm/s												
Espera por Distancia	200 mm/s												
	500 mm/s												

- El error relativo es el cociente de los módulos del error absoluto y del valor real. Sin embargo, para corregir errores nos interesa calcular el error de estimación, **con su signo**:

$$\mathbf{Ee_x = e_x / p}$$
$$\mathbf{Ee_y = e_y / p}$$

- Rellenar esta tabla con los errores de estimación ee_x y ee_y, y el error relativo de r:

$$\mathbf{Er_r} = |\vec{e}| / |\vec{p}|$$

Tipo de Evento	Velocidad	Prueba 1			Prueba 2			Prueba 3			Valores medios		
		Ee_x	Ee_y	Er_r	Ee_x	Ee_y	Er_r	Ee_x	Ee_y	Er_r	Ee_x	Ee_y	Er_r
Espera por Tiempo	200 mm/s												
	500 mm/s												
Espera por Distancia	200 mm/s												
	500 mm/s												

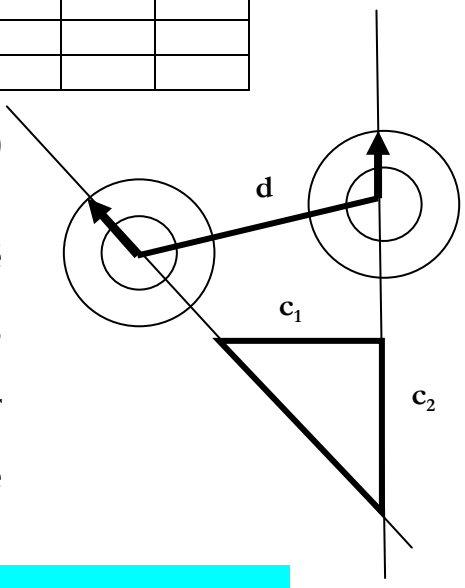
PREGUNTA 1: ¿Qué nos dice el error de estimación? ¿Cómo podemos usarlo para corregir el error acumulado en la navegación estimativa?

4. Odometría en un recorrido cuadrado

- Programar un script que recorra un cuadrado de **120 cm** de lado, girando 4 veces a la derecha.
- Hacer tres pruebas de cada recorrido y velocidad. En cada prueba, medir la distancia entre ambos puntos y el ángulo que hace el robot con respecto de su propia posición inicial.
- Rellenad la siguiente tabla, donde d es la distancia entre los puntos de origen y de final (tomados por ejemplo en la parte delantera del iRobot) y α= arc tan (c_1/c_2).:

Tipo de Evento	Velocidad	Prueba 1		Prueba 2		Prueba 3		Medias	
		d	α	d	α	d	α	d	α
Espera por Tiempo	200mm/s								
	500mm/s								
Espera por Distancia	200mm/s								
	500mm/s								

- Repetid las pruebas con un script que recorra un cuadrado de **120 cm** de lado, girando 4 veces a la izquierda.



PREGUNTA 2: ¿Tendría sentido calcular el error relativo?¿qué información nos daría?
PREGUNTA 3: ¿Cómo podemos usar los resultados de estas pruebas para corregir el error acumulado en la navegación por dead reckoning?
PREGUNTA 4: ¿Qué diferencias encontráis entre la espera por tiempo y la espera por distancia?
PREGUNTA 5: ¿Qué diferencias encontráis entre las velocidades de 200mm/s y 500 mm/s?

5. Dead Reckoning corregido

Programar un script de *dead reckoning* que recorra un cuadrado de **200 cm** de lado minimizando los errores d y α.
PREGUNTA 5: ¿Qué hacéis para corregir los errores de odometría?

6. Informe a presentar

- Listado y descripción de los scripts programados
- Tablas con las medidas
- Contestar a las preguntas 1, 2, 3, 4 y 5

Anexo Programa Enviarls.cc

```
/* Envía los scripts contenidos en myScript.txt al iRobot Create*/

#include <iostream>
#include <stdio.h>
#include <stdlib.h>
#include <wiringPi.h>
#include <wiringSerial.h>

#define SERIAL "/dev/ttyUSB0"

void send_script(char *myScript);
int main(int argc, char**argv) {
    std::cout << "Script: " << argv[1] << std::endl;
    if(argc==2){
        send_script(argv[1]);
    }else{
        std::cout << "Error no argument found" << std::endl;
    }
    return EXIT_SUCCESS;
}

void send_script(char *myScript){
    int value, fd_serial, num_bytes;
    FILE * fd_script;
    fd_script = fopen(myScript, "r");
    if(fd_script < 0 ){
        std::cout << "Error opening the script file" << std::endl;
    }
    fd_serial =0;
    fd_serial = serialOpen(SERIAL, 57600);
    if(fd_serial < 0 ){
        std::cout << "Error opening serial line" << std::endl;
    }
    serialPutchar(fd_serial,  128);
    delay(10);
    std::cout << "escribiendo 128" << std::endl;
    serialPutchar(fd_serial,  132);
    delay(10);
    std::cout << "escribiendo 132" << std::endl;
    value = 0;
    num_bytes = 0;
    while(value!= -1){
        fscanf( fd_script, "%d", &value );

        if(value != -1){
            num_bytes++;
            std::cout << std::dec << value  << " " << std::endl;

            serialPutchar(fd_serial,  value);
            delay(10);
        }
    }
    std::cout <<  "Script ends: " << num_bytes << " bytes read" <<
std::endl;

    serialClose(fd_serial);
return;
}
```