PROGRAMACIÓN DE ROBOTS

Programación de robots móviles
PRÁCTICA 5.- Localización odométrica de robots móviles Lego Mindstorms NXT

El objetivo de la práctica consiste en realizar la localización del robot Lego usando su odometría.

NOTAS GENERALES:
a) Puedes usar en los cálculos la constante PI, predefinida en NXC. Además, cualquier constante que escribas en el código (ya sea con #define o no) que sea flotante, escríbela para que el compilador sepa que lo es y no la pueda confundir con un entero (por ejemplo, en vez de 1000, escribe 1000.0).
b) Las rutinas Cos() y Sin() de NXC necesitan valores en grados como parámetros y devuelven un entero que hay que dividir por 100.0 para obtener el valor deseado. Nunca insertes una llamada a estas funciones en una expresión: úsalas siempre por separado y guarda su resultado en una variable.
*c) NXC no es un lenguaje muy robusto y tiene algunos problemas no documentados. Cuando tengas que escribir una expresión matemática en el código, divídela mejor en pasos pequeños. Por ejemplo, la sentencia "x = f * Cos(a) / 100.0;" puede dar problemas, a no ser que se escriba como "x=Cos(a); x*=f/100.0;".*
d) Por el mismo motivo, declara todas las variables locales de cada rutina al principio de ésta, antes de que empiece el código, y no las inicialices en la declaración, sino después, usando sentencias de asignación.
e) El lenguaje NXC no admite typedef ni castings.
f) Si al ejecutar el programa da "File Error", puede ser porque haya demasiadas variables declaradas (que no quepan en la RAM de 64KB). Procura tener sólo el programa .rxe que estás usando (y el de demo que trae por defecto) en la memoria del robot. Si aún así aparecen problemas, disminuye el número de muestras.

1.- Planteamiento del problema

En primer lugar haz un programa que mueva al robot recorriendo un cuadrado de manera continua, al menos una vez completa, y tomando muestras de los sensores propioceptivos y del tiempo. Para recorrer ese cuadrado hay que realizar dos tipos de movimiento:

- Para recorrer los tramos rectos, arranca los motores A y C a potencia igual a 50 durante un cierto tiempo;
- Para hacer los giros apaga el motor A durante otro cierto tiempo (dejando el C a máxima potencia).

El esquema general de tu programa debe ser parecido al que sigue, y debe basarse en el esqueleto de programa dejado en la página web de la asignatura para esta práctica:

1. Declarar como constantes globales (#define) aquéllas que consideres interesantes para tu programa; en particular, una que guarde el número de muestras a tomar de la ejecución del movimiento del robot, que en principio debería ser 1300. Cuidado: no superes la RAM (64Kb).

2. Declarar como variables globales (arrays) los vectores necesarios para almacenar en RAM todas las muestras de la ejecución del movimiento, que incluyen tuplas del tipo: [tiempo actual (ms), codificador_rueda_A (grados), codificador_rueda_B (grados), x, y, theta]

3. Ya en la rutina *main*, comprobar que hay memoria Flash suficiente para grabar en fichero los resultados, y termina el programa si no (este trozo está ya escrito en el esqueleto).

4. Inicializar la primera pose, el primer tiempo y los primeros valores de los encoders a 0. Asimismo, establecer el estado actual del robot a "parado".

5. Entrar en un bucle que itere por cada una de las muestras a tomar. Dentro de ese bucle:
   - Poner las potencias en los motores correspondientes al estado actual ("recta" o "girando").

- Tomar una muestra de los sensores y del tiempo y rellenar los vectores globales correspondientes (la pose, o sea, x, y y theta, ponlas a 0 en este primer apartado de la práctica).
- Medir cuánto tiempo ha pasado desde que nos pusimos en el estado actual hasta el momento presente. Si ese tiempo supera al que tenemos que estar en el estado actual, cambiar al estado contrario y anotar el tiempo en el que esto ha sucedido.
- La última instrucción del bucle debe ser un wait de 30 milisegundos, que asegure que se realizan todas las lecturas y escrituras correctamente.

6. Cuando se termine el bucle de muestreo, parar los motores y grabar en fichero todos los datos de los vectores globales, en formato de texto (el esqueleto de programa que hay en la web contiene este trozo bastante completo).

Ajusta los valores de tiempo de tramo recto y de giro a mano para tu robot hasta que consigas que siga el cuadrado lo mejor posible.

2.- Localización off-line

Una vez ejecutado el programa del apartado 1 satisfactoriamente, vamos a estimar offline, usando Matlab la posición cartesiana que ha tenido el robot en cada momento de muestreo, usando los datos de tiempo y encoders que éste ha grabado en fichero, y el siguiente modelo cinemático:

$$\Delta x=(\Delta rotr*radio+\Delta rotl*radio)/2)\cos(theta0)$$
$$\Delta y=(\Delta rotr*rad+\Delta rotl*radio)/2)\text{sen}(theta0)$$
$$\Delta theta=(\Delta rotr*radio-\Delta rotl*radio)/d)$$

Para ello escribe una función en Matlab que calcule la odometría del robot en un paso:

```
function [x1,y1,theta1]=odometry(x0,y0,theta0,t0,rotl0,rotr0,t1,rotl1,rotr1)
```

donde (*x0*,*y0*,*theta0*) es la última pose calculada para el robot, en metros y radianes, *t0* el tiempo (en milisegundos) cuando se calculó esa pose, y *rotl0* y *rotr0* lo que medían los sensores de rotación de cada rueda en aquel momento, en grados. En el primer paso todos esos valores serán 0. La rutina debe rellenar (*x1*,*y1*,*theta1*) con la nueva pose del robot, en metros y radianes, calculada en el momento actual mediante la medición del tiempo en milisegundos (que está en *t1*) y la de los encoders en grados (que están en *rotl1* y *rotr1*).

Si el incremento de tiempo entre una llamada y la anterior (o sea, *t1-t0*) es igual a cero, la rutina deberá devolver la misma pose que se le ha pasado, como si no hubiera habido movimiento. Dibuja en una gráfica la trayectoria seguida por el robot según esta función (todas sus poses, incluidas las orientaciones).

3.- Localización on-line

Basándote en la rutina que has tenido que implementar en Matlab para calcular la odometría en cada iteración, escribe ahora una rutina análoga para el robot que calculará en tiempo real su pose mediante el mismo modelo cinemático. La rutina en NXC debería tener el siguiente prototipo:

```
void odometry(float x0, float y0, float theta0, long t0, int rotl0, int rotr0,
 float & x1, float & y1, float & theta1, long & t1, int & rotl1, int & rotr1)
```

donde los parámetros son análogos a los de la rutina de Matlab, sólo que cambiados de sitio para que tengan más sentido en un programa NXC. En este caso, todos los parámetros que empiezan por "&" son de salida, es decir, deben ser rellenados por la rutina internamente. Ten cuidado de asignarles un valor dentro de la rutina y de no usarlos para nada más (ni siquiera como parte de otras expresiones ni llamadas a otras rutinas).

Recoge los datos devueltos de una ejecución de esta rutina (si puedes, graba un vídeo del robot recorriendo el cuadrado, a vista de pájaro, en el que se aprecie bien la trayectoria seguida sobre el suelo) y superpón en una misma gráfica los resultados de la odometría calculada por Matlab para ese experimento y de la odometría calculada por el robot, y saca conclusiones con su comparación, así como con la comparación con la trayectoria real del robot en el vídeo.