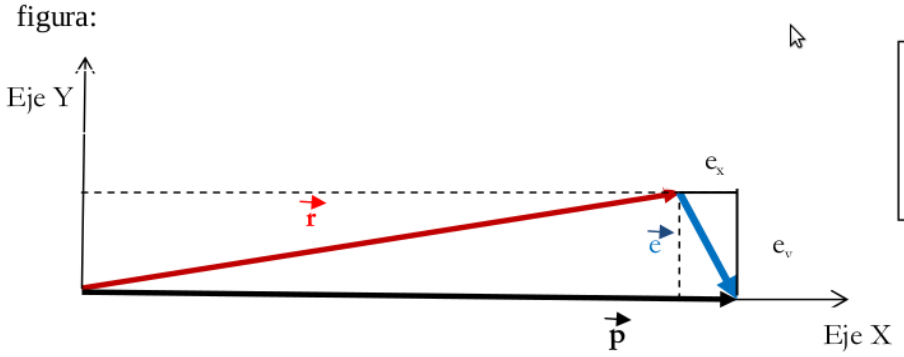
Práctica 2.2
-
Informe

Julen D. y Julen F.

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 $e_x^2 + e_x^2 = d^2$)

Scripts:

- Espera por tiempo

  - 152 : Comando para determinar que se va a ejecutar una secuencia de órdenes
  - 12 : Cantidad de bytes que tiene la secuencia de órdenes a ejecutar
  - 137 : Comando que determina que se quiere conducir el robot
  - 1 94 : Par de bytes que determina la velocidad a tomar (1 94 = 15E = 350)
  - 128 0 : Par de bytes que determina el radio de giro durante la marcha
  - 155 : Comando que determina una espera por tiempo
  - 44 : Cantidad de decimas de segundo que se esperará (1500mm / (350mm/s) = 4,4 s)
  - 137 0 0 0 0 : Comando para detener la marcha del robot

- Espera por distancia

  - 152 : Comando para determinar que se va a ejecutar una secuencia de órdenes
  - 13 : Cantidad de bytes que tiene la secuencia de órdenes a ejecutar
  - 137 : Comando que determina que se quiere conducir el robot
  - 1 94 : Par de bytes que determina la velocidad a tomar (1 94 = 15E = 350)
  - 128 0 : Par de bytes que determina el radio de giro durante la marcha
  - 156 : Comando que determina una espera por distancia
  - 5 220 : Cantidad de milimetros que se esperará a recorrer (1500mm = 5 DC = 5 220)
  - 137 0 0 0 0 : Comando para detener la marcha del robot

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	350 mm/s	14	32	34,9	11	30	31,9	11	32	33,8	12	31,3	33,5
Espera por distancia	350 mm/s	10	11	14,8	10	23	25	8	11	13,6	10,3	15	17,8

- 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	**Veloc. (mm/s)**	**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
Esp. (tiempo)	350	0,093	0,213	0,2326	0,073	0,2	0,2126	0,073	0,213	0,2253	0,08	0,208	0,223
Esp. (distancia)	350	0,066	0,073	0,986	0,066	0,153	0,166	0,053	0,073	0,0906	0,0686	0,1	0,1186

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

El error de estimación nos indica la desviación que tenemos en los ejes x e y (con sus signos + o -) respecto al punto al que queremos llegar, desde el que partimos para la hacer la medición.

Con estos datos o este error, podemos calcular una media de desviación y a la hora de ejecutar la acción corregir en positivo o negativo la distancia a recorrer y el ángulo a girar.

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).:

Script

- Espera por tiempo

  - 152 : Comando para determinar que se va a ejecutar una secuencia de órdenes
  - 65 : Cantidad de bytes que tiene la secuencia de órdenes a ejecutar
  - Avance y giro nº 1:
    - 137 : Comando que determina que se quiere conducir el robot
    - 1 94 : Par de bytes que determina la velocidad a tomar (1 94 = 15E = 350)
    - 128 0 : Par de bytes que determina el radio de giro durante la marcha
    - 155 : Comando que determina una espera por tiempo
    - 34 : Cantidad de decimas de segundo que se esperará (1200mm / (350mm/s) = 3,4 s)
    - 137 : Comando que determina que se quiere conducir el robot
    - 1 94 : Par de bytes que determina la velocidad a tomar (1 94 = 15E = 350)
    - 255 255 : Par de bytes que determina el radio de giro durante la marcha
    - 157 : Comando que determina una espera por ángulo
    - 255 166 : ángulo de giro (256 (0º) – 90º (giro a la derecha) = 166 (90º a la derecha))
  - 137 1 94 128 0 155 34 137 1 94 255 255 157 255 166 : avance y giro nº 2
  - 137 1 94 128 0 155 34 137 1 94 255 255 157 255 166 : avance y giro nº 3
  - 137 1 94 128 0 155 34 137 1 94 255 255 157 255 166 : avance y giro nº 4
  - 137 0 0 0 0 : Comando para detener la marcha del robot
  - 153 -1 : Comando para ejecutar el script

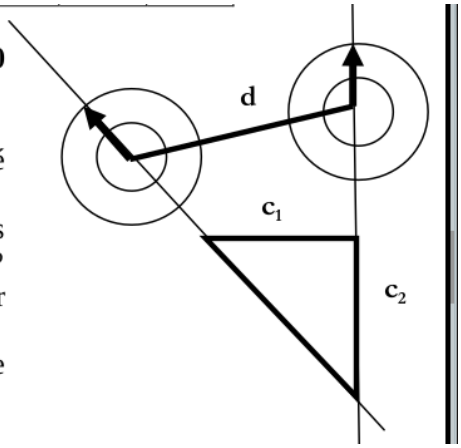
- Espera por distancia

  - 152 : Comando para determinar que se va a ejecutar una secuencia de órdenes
  - 69 : Cantidad de bytes que tiene la secuencia de órdenes a ejecutar
  - Avance y giro nº 1:
    - 137 : Comando que determina que se quiere conducir el robot
    - 1 94 : Par de bytes que determina la velocidad a tomar (1 94 = 15E = 350)
    - 128 0 : Par de bytes que determina el radio de giro durante la marcha
    - 156 : Comando que determina una espera por distancia
    - 4 176 : Cantidad de milimetros que se esperará a recorrer (1200mm = 4 B0 = 4 176)
    - 137 : Comando que determina que se quiere conducir el robot
    - 1 94 : Par de bytes que determina la velocidad a tomar (1 94 = 15E = 350)
    - 255 255 : Radio de giro durante la marcha (en este caso, a la izquerda)
    - 157 : Comando que determina una espera por ángulo
    - 255 166 : ángulo de giro (256 (0º) – 90º (giro a la derecha) = 166 (90º a la derecha))
  - 137 1 94 128 0 155 34 137 1 94 255 255 157 255 166 : avance y giro nº 2
  - 137 1 94 128 0 155 34 137 1 94 255 255 157 255 166 : avance y giro nº 3
  - 137 1 94 128 0 155 34 137 1 94 255 255 157 255 166 : avance y giro nº 4
  - 137 0 0 0 0 : Comando para detener la marcha del robot
  - 153 -1 : Comando para ejecutar el script

Tipo de evento (giro a la dcha)	Velocidad	Prueba 1		Prueba 2		Prueba 3		Valores medios	
		d	α	d	α	d	α	d	α
Espera por tiempo	350 mm/s	43,5	0,69	50,8	0,65	53	0,64	49,1	0,66
Espera por distancia	350 mm/s	58,5	0,84	55,2	π/4	56,8	π/4	56,833	0,804

- 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?



Script

152 65
137 1 94 128 0 155 34 137 1 94 0 1 157 0 90
137 1 94 128 0 155 34 137 1 94 0 1 157 0 90
137 1 94 128 0 155 34 137 1 94 0 1 157 0 90
137 1 94 128 0 155 34 137 1 94 0 1 157 0 90
137 0 0 0 0 153 -1

- <u>Espera por tiempo</u>

  - 152 : Comando para determinar que se va a ejecutar una secuencia de órdenes
  - 65 : Cantidad de bytes que tiene la secuencia de órdenes a ejecutar
  - Avance y giro nº 1:
    - 137 : Comando que determina que se quiere conducir el robot
    - 1 94 : Par de bytes que determina la velocidad a tomar (1 94 = 15E = 350)
    - 128 0 : Par de bytes que determina el radio de giro durante la marcha
    - 155 : Comando que determina una espera por tiempo
    - 34 : Cantidad de decimas de segundo que se esperará (1200mm / (350mm/s) = 3,4 s)
    - 137 : Comando que determina que se quiere conducir el robot
    - 1 94 : Par de bytes que determina la velocidad a tomar (1 94 = 15E = 350)
    - 0 1 : Radio de giro durante la marcha (en este caso, a la derecha)
    - 157 : Comando que determina una espera por ángulo
    - 0 90 : ángulo de giro (90º a la izquierda)
  - 137 1 94 128 0 155 34 137 1 94 0 1 157 0 90 : avance y giro nº 2
  - 137 1 94 128 0 155 34 137 1 94 0 1 157 0 90 : avance y giro nº 3
  - 137 1 94 128 0 155 34 137 1 94 0 1 157 0 90 : avance y giro nº 4
  - 137 0 0 0 0 : Comando para detener la marcha del robot
  - 153 -1 : Comando para ejecutar el script

- Espera por distancia

    - 152 : Comando para determinar que se va a ejecutar una secuencia de órdenes
    - 69 : Cantidad de bytes que tiene la secuencia de órdenes a ejecutar
    - Avance y giro nº 1:
        - 137 : Comando que determina que se quiere conducir el robot
        - 1 94 : Par de bytes que determina la velocidad a tomar (1 94 = 15E = 350)
        - 128 0 : Par de bytes que determina el radio de giro durante la marcha
        - 156 : Comando que determina una espera por distancia
        - 4 176 : Cantidad de milimetros que se esperará a recorrer (1200mm = 4 B0 = 4 176)
        - 137 : Comando que determina que se quiere conducir el robot
        - 1 94 : Par de bytes que determina la velocidad a tomar (1 94 = 15E = 350)
        - 0 1 : Radio de giro durante la marcha (en este caso, a la derecha)
        - 157 : Comando que determina una espera por ángulo
        - 0 90 : ángulo de giro (90º a la izquierda)
    - 137 1 94 128 0 156 4 176 137 1 94 0 1 157 0 90 : avance y giro nº 2
    - 137 1 94 128 0 156 4 176 137 1 94 0 1 157 0 90 : avance y giro nº 3
    - 137 1 94 128 0 156 4 176 137 1 94 0 1 157 0 90 : avance y giro nº 4
    - 137 0 0 0 0 : Comando para detener la marcha del robot
    - 153 -1 : Comando para ejecutar el script

Tipo de evento (giro a la izqda)	**Velocidad**	**Prueba 1**		**Prueba 2**		**Prueba 3**		**Valores medios**	
		d	α	d	α	d	α	d	α
Espera por tiempo	350 mm/s	37,5	-0,66	35,6	-0,53	37,8	-0,58	36,966	-0,59
Espera por distancia	350 mm/s	41,4	-0,53	40,5	-0,53	44,3	-0,87	42,066	-0,64

PREGUNTA 2: ¿Tendría sentido calcular el error relativo?¿qué información nos daría?

El error relativo nos informa de cuánto sería el error en relación a la posición final que se quiere conseguir. En este caso, el comportamiento del robot varía considerablemente tras cada giro, dando a cada ejecución del programa una posición distinta. Por lo tanto, no tiene sentido calcular ese error relativo, porque sería un margen de error demasiado grande como para poder usarlo para corregir el rumbo del iRobot.

PREGUNTA 3: ¿Cómo podemos usar los resultados de estas pruebas para corregir el error acumulado en la navegación por dead reckoning?

En nuestro caso, se trata de dibujar un cuadrado, con lo que la acción de ir recto y girar 90º se repetiría 4 veces, por lo que el error final dividido entre 4 nos daría el error a corregir en cada una de esas repeticiones y así disminuir el error final.

PREGUNTA 4: ¿Qué diferencias encontráis entre la espera por tiempo y la espera por distancia?

La diferencia está en que la espera por distancia es más exacta que la espera por tiempo. En las pruebas que hemos realizado, tanto en línea recta como al dibujar el cuadrado, la media de errores es menor en las esperas por distancia.

PREGUNTA 5: ¿Qué diferencias encontráis entre las velocidades de 200mm/s y 500 mm/s?

Estas pruebas no se realizaron en clase, pero a mayor velocidad el error debería ser mayor, ya que en el tiempo en mandar la señal de frenado al robot recorrería más distancia al ir a mayor velocidad.

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?

Script

- Dibujar un cuadrado (200cm * lado) (350mm/s) (por distancia:2000mm) (giro a la izqda)

152 69
137 1 94 128 0 156 7 200 137 1 94 0 1 157 0 82
137 1 94 128 0 156 7 200 137 1 94 0 1 157 0 82
137 1 94 128 0 156 7 200 137 1 94 0 1 157 0 82
137 1 94 128 0 156 7 200 137 1 94 0 1 157 0 82
137 0 0 0 0 153 -1

- 152 : Comando para determinar que se va a ejecutar una secuencia de órdenes
- 69 : Cantidad de bytes que tiene la secuencia de órdenes a ejecutar
- Avance y giro nº 1:
  - 137 : Comando que determina que se quiere conducir el robot
  - 1 94 : Par de bytes que determina la velocidad a tomar (1 94 = 15E = 350)
  - 128 0 : Par de bytes que determina el radio de giro durante la marcha
  - 156 : Comando que determina una espera por distancia
  - 7 200 : Cantidad de milimetros que se esperará a recorrer (2000mm = 7 D0 = 7 200)
  - 137 : Comando que determina que se quiere conducir el robot
  - 1 94 : Par de bytes que determina la velocidad a tomar (1 94 = 15E = 350)
  - 0 1 : Radio de giro durante la marcha (en este caso, a la derecha)
  - 157 : Comando que determina una espera por ángulo
  - 0 81 : ángulo de giro (81º a la izquierda) (explicación, a continuación)
- 137 1 94 128 0 156 7 200 137 1 94 0 1 157 0 81 : avance y giro nº 2
- 137 1 94 128 0 156 7 200 137 1 94 0 1 157 0 81 : avance y giro nº 3
- 137 1 94 128 0 156 7 200 137 1 94 0 1 157 0 81 : avance y giro nº 4
- 137 0 0 0 0 : Comando para detener la marcha del robot
- 153 -1 : Comando para ejecutar el script

PREGUNTA 6: ¿Qué hacéis para corregir los errores de odometría?

Para reducir d:

Para reducir el error en la distancia, debemos calcular cuanto y hacía que lado (signo) ha sido la desviación, para luego restar o sumar esa distancia en los recorridos que sean. Es decir, si el robot se ha pasado 2cm y se ha desviado a la derecha 5cm respecto al punto de llegada, debemos restar 2cm cuando el sentido del robot vaya hacia el punto desde uno de sus lados y restar 5 cm cuando el sentido del robot vaya hacia el punto desde el frente o por detrás del punto.

Para reducir el error en α:

Como hemos dicho antes, tenemos 4 giros en total y sabemos la media de los ángulos de error al finalizar el cuadrado. Con estos datos, dividimos el ángulo de error entre 4 y en nuestro caso (debido al signo calculado) restamos a los 90º que deberían ser del giro los 81º que serían la parte proporcional a cada giro.

[CALCULOS]

<u>Para corregir el ángulo:</u>

Siendo α el desvío de ángulo acumulado en radianes del cuadrado dibujado con giros a la izquierda y con espera por distancia. Si 90º = π/2, calculamos el equivalente de α en grados usando la regla de 3:

$\pi/2$ (≈ 1’5708) → 90º ; -0’64 → -36,67º

Teniendo en cuenta que el desvío es la acumulación del desvío de cada giro, y que se dibujan cuatro giros, se divide el resultado entre los giros que se ejecutan.

-36,67º / 4 ≈ -9,17º

Para terminar, sólo falta sumar el desvío resultante en el ángulo inicial para lograr un resultado más preciso.

90º + (-9’17) = 80,83º

Como el comando de espera por ángulo opera con grados enteros (sin decimales), el resultado se redondea a 81º. A su vez, como la suma de los restos ((1-0,83) * 4 = 0,17*4 = 0,71) no alcanza 1º, no se añade esta cantidad al ángulo final.