PL1-SENSORES Y ACTUADORES

David Bachiller Vela

Pablo Bermejo

1. Resumen

Esta práctica es una pequeña introducción a la robótica dividida en dos partes, por un lado trabajo con un simulador y por otra parte trabajo con el robot real, en donde tenemos un mapa por el que se va moviendo el robot y el objetivo principal de este practica es utilizar los diferentes sensores que nos proporciona el robot para la toma y análisis de datos que a su vez se harán de forma paralela con el robot real, así podremos comprar y poder sacar conclusiones precisas, mediante graficas o mediante análisis de datos y ver valores como el margen de error, la exactitud, la precisión.

El principal problema que tuvimos a la hora de enfrentar este trabajo fue sin duda la falta de conocimientos ya que nunca habíamos tratado una materia parecida a esta. Por suerte, gracias a los archivos colgados en blackboard y a diferentes paginas nombradas en la bibliografía, pudimos adquirir los conocimientos necesarios. Es por esto que decidimos enfocar la tarea en fases, que más adelante nos dividiríamos entre los dos, de esta manera cada uno podría centrarse en un tema en concreto para así poder llegar a mejores resultados. A pesar de esto hemos estado continuamente pendiente el uno del otro para no quedarnos atrás y poder aprender sobre las actividades que se nos proponen en la práctica.

2. Introducción

El objetivo de esta práctica es un control completo de los conceptos aprendidos en la primera parte de la Asignatura “Sistemas de Percepción”, en la que vamos a emplear la plataforma de desarrollo robótico ROS, el simulador STDR y el robot real Amigobot.

Durante esta práctica se va a trabajar con diferentes suscripciones a diferentes tópicos como pueden ser sensores de distancia sonar y laser, sensores de odometría (odom), en los que leeremos la información que nos proporcionan estos sensores.

Además, vamos a trabajar con una publicación en el topic de los motores (cmd_vel) que permite enviar comandos a los actuadores del robot y poder así poder desplazar el robot por el entorno ya sea de forma lineal o de forma angular como se vera en los diferentes puntos de la práctica.

Vamos a emplear como entorno de la practica un mapa con forma de laberinto de prueba con una resolución de 1000 x 1000 pixeles que en dimensiones reales equivale a 10 x 10.

Nota importante: Todos los códigos están explicados línea por línea por lo que se harán referencias a el código en determinadas ocasiones.

3. Sección Principal

3.1 SENSORES EN ROS

3.1.1 Sensores de odometria

A) Indique y describa la información que nos ofrece el mensaje disponible en el topic odom. Muestre algún ejemplo de captura.

El mensaje disponible en el tópico "odom" en MATLAB proporciona información sobre la odometría del robot, que es la estimación de su posición y orientación en función de su movimiento en el espacio. La información que proporciona este mensaje incluye:

Pose: la posición y orientación estimadas del robot en el espacio tridimensional, representadas por una estructura que incluye la posición (x, y, z) y la orientación (representada por un cuaternión).

Twist: la velocidad lineal y angular del robot en el espacio tridimensional, representada por una estructura que incluye la velocidad lineal (en x, y, z) y la velocidad angular (en x, y, z).

En resumen, el mensaje "odom" en MATLAB proporciona información importante sobre la posición, orientación y velocidad del robot, lo que puede ser el control del robot en tiempo real

```
Roll (radianes): 0
Pitch (radianes): 0
Yaw (radianes): 0.090319
```

```
dist =

    0.1497
```

B) Mida la resolución máxima (q) de odometría lineal y angular máxima con las diferentes combinaciones de velocidades propuestas en el simulador STDR. Construya una tabla como la que se muestra a continuación con las 8 combinaciones propuestas.

V (ms-1)	Ω (rads-1)	q_lineal (m)	q_angular (r)
0.1	0.0	0.0201	0.0
0.3	0.0	0.0599	0.0
0.5	0.0	0.0524	0.0
0.7	0.0	0.0738	0.0
0.9	0.0	0.0936	0.0

Para calcular q_lineal(m) primero vamos cambiando según se pida el comando de velocidad en este caso lo hemos hecho con X, luego guardamos en un array distancias la distancia que se ha desplazado hasta coger alrededor de 1000 datos para poder obtener una medida lo suficientemente precisa, después con un bucle for se recorre el array anterior y almaceno en un nuevo array llamado distancias la resta de la posición actual menos la posición anterior.

```
differences = []
Para i desde 2 hasta el final de la lista distances:
    diff = distances[i] - distances[i-1]
```

```
    agregar diff a la lista differences
    mostrar la lista differences
fin del bucle
```

V (ms-1)	Ω (rads-1)	q_lineal (m)	q_angular (r)
0.0	0.3	0.0	0.0336
0.0	0.7	0.0	0.0787
0.0	0.9	0.0	0.1000

Para calcular q_angular(r) primero vamos cambiando según se pida el comando de velocidad en este caso lo hemos hecho con Z, primero obtenemos la posición actual y luego convertimos esa posición en ángulos de Euler en radianes, e imprimimos el valor de los ángulos en este caso yaw, y lo almacenamos todo en un array distances y repito el mismo proceso que el anterior con el array diferences

```
//Obtenemos la posición actual
pos = odom.LatestMessage.Pose.Pose.Orientation
qpos = [pos.W, pos.X, pos.Y, pos.Z]
//Convertir el resultado anterior en ángulos de Euler en radianes
[yaw, pitch, roll] = quat2angle(qpos, 'ZYX')
//Imprimir los valores de los ángulos en radianes
escribir('Roll (radianes): ' + convertir_a_cadena(roll))
escribir('Pitch (radianes): ' + convertir_a_cadena(pitch))
escribir('Yaw (radianes): ' + convertir_a_cadena(yaw))
//Agregar yaw a la lista de distancias
agregar_a_lista(distancias, yaw)
```

```
differences = []
Para i desde 2 hasta el final de la lista distances:
    diff = distances[i] - distances[i-1]
    agregar diff a la lista differences
    mostrar la lista differences
fin del bucle
```

C) Mida la resolución máxima (q) de odometría lineal y angular con las diferentes combinaciones de velocidades propuestas en el robot real (en este caso, debe tener en cuenta las aceleraciones y deceleraciones del robot). Construya una tabla como la del apartado anterior con las 8 combinaciones propuestas.

V (ms-1)	Ω (rads-1)	q_lineal (m)	q_angular (r)
0.1	0.0	0.0203	0.0
0.3	0.0	0.0342	0.0
0.5	0.0	0.0514	0.0
0.7	0.0	0.0514	0.0
0.9	0.0	0.0506	0.0

Para calcular q_lineal(m) en este caso se trata de un robot real por lo que primeramente en los subscribers en odom hay que subscribirse a /pose y después para poder genar movimiento hay que habilitar el motor con msg_enable y pub_enable_motor.Data = 1, y el resto del código es igual que en el simulador obtenemos la distancia la guardamos en el array y luego recorremos ese array para calcular la diferencia entre la posición inicial menos la posición anterior.

V (ms-1)	Ω (rads-1)	q_lineal (m)	q_angular (r)
0.0	0.3	0.0	0.0698
0.0	0.7	0.0	0.0698
0.0	0.9	0.0	0.07679

Para calcular q_angular(r) en este caso se trata de un robot real por lo que primeramente en los subscribers en odom hay que subscribirse a /pose y después para poder genar movimiento hay que habilitar el motor con msg_enable y pub_enable_motor.Data = 1. Luego vamos cambiando según se pida el comando de velocidad en este caso lo hemos hecho con Z, después obtenemos la posición actual y luego convertimos esa posición en ángulos de Euler en radianes, e imprimimos el valor de los ángulos en este caso yaw, y lo almacenamos todo en un array distances y repito el mismo proceso que el anterior con el array diferences

3.1.2 Sensores de distancia ultrasónicos

A) Indique y describa la información que nos ofrece el mensaje disponible en el topic sonar. Muestre algún ejemplo de captura.

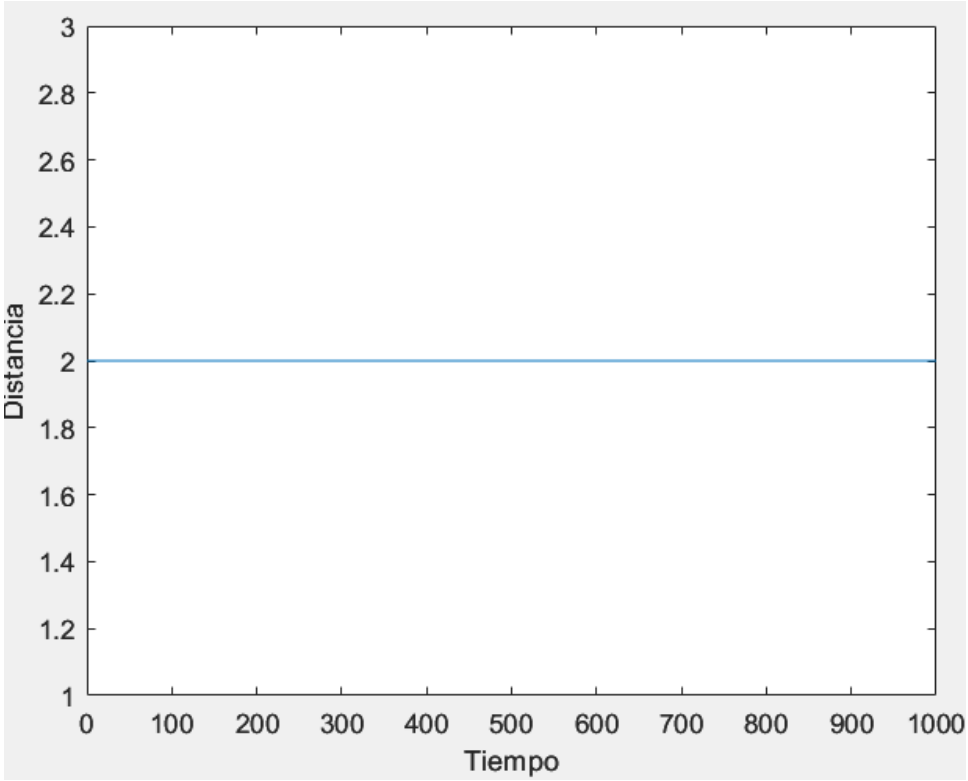
```
   MessageType: 'sensor_msgs/Range'
    ULTRASOUND: 0
      INFRARED: 1
        Header: [1×1 Header]
 RadiationType: 0
   FieldOfView: 0.2618
      MinRange: 0.1000
      MaxRange: 6
        Range_: Inf
```

Como se puede ver en la imagen, el mensaje del sonar nos muestra entre otros datos el rango máximo, el mínimo y el dato que está detectando que en este caso es inf ya que no hay ningún obstáculo detectado.

B) En el simulador STDR, posicione el robot de tal forma que exista una distancia de 2m de uno de los sensores sónar concretos del robot. ¿Qué posición y sónar se ha elegido?

Se ha elegido el sonar 0 y se ha puesto de frente a una pared, dándonos así el valor a la que esta se encuentra, es decir 2m.

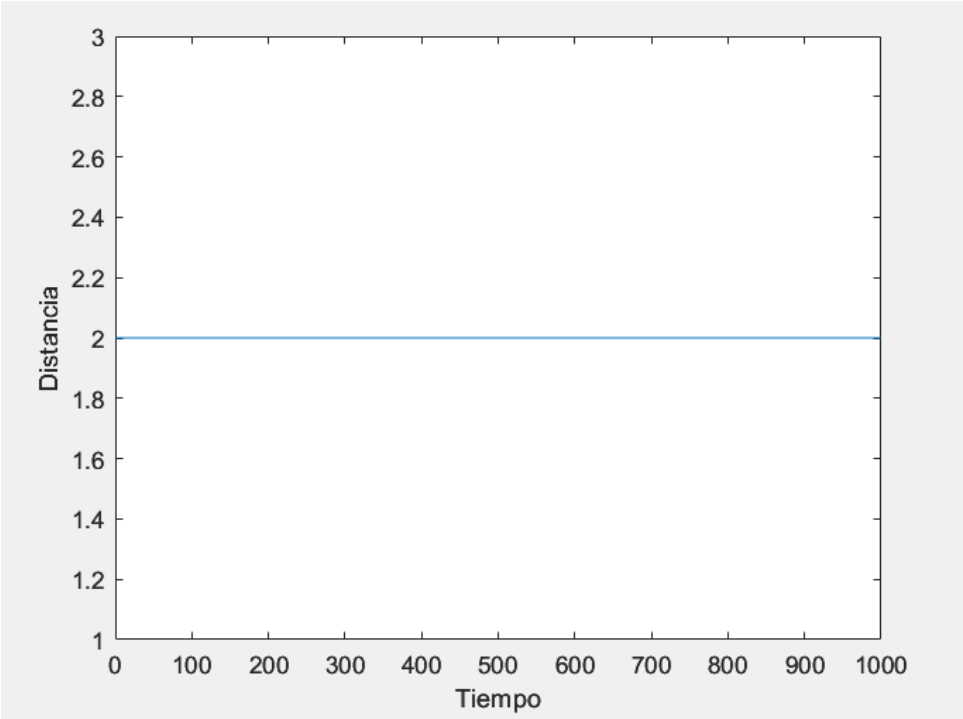
C) Obtenga 1.000 medidas de distancia del sensor sónar elegido y dibuje en una gráfica (comando plot en Matlab) la distancia medida. ¿Son estables las medidas? ¿Hay ruido en la medida? En caso afirmativo, calcule el valor máximo, medio y la varianza del ruido.



Como se puede ver en la gráfica, los valores son constantes. Esto se debe a que estamos trabajando en un simulador por lo que no habrá ruido que afecte a las estadísticas siendo así la media de 2, el máximo de 2 y varianza de 0 ya que no existe dispersión.

Para poder obtener esta grafica lo primero que hemos hecho es crear un bucle que vaya de 1 a 1000 de tal modo que en cada vuelta recopile el valor que corresponde a ese momento y lo guarde en un array, además también guardamos el número de vuelta que es. Con esto usamos el comando plot(time,distances) para mostrar una gráfica que se mueva en el tiempo y muestre la distancia correspondiente. Todo esto se encuentra en el fichero llamado MilMedidas

D) Implemente un filtro media móvil con los últimos 5 valores de distancia y dibuje en una gráfica. ¿Son más estables las medidas que en el caso anterior? ¿Sería útil este método si el robot está en movimiento en lugar de permanecer estático?



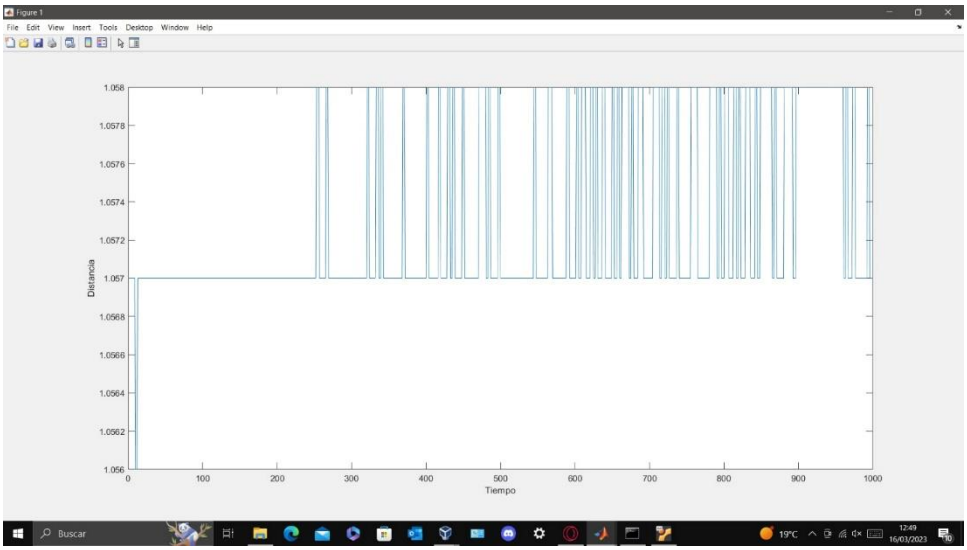
Al igual que en el caso anterior, la gráfica sigue siendo uniforme ya que se trata de un simulador. Este método no nos serviría para un robot en movimiento ya que, al hacer medias sobre medias, los datos se verían demasiado modificados sobre los reales, Por ejemplo, si un dato en 3 metros y hasta la siguiente vuelta avanza hasta 6 metros nos daría un valor que no se corresponde con lo que queremos mostrar.

Esta grafica la conseguimos creando de nuevo un bucle de 1000 turnos que en cada una de ellas lo que hace es almacenar el valor en un array y en caso de que ya haya 5 datos hace la media y lo almacena como un nuevo valor en el array definitivo que luego usaremos en el plot de la misma manera que hicimos en el ejercicio anterior. Este código corresponde al archivo llamado MediaMovil.

E)

B con el real: Hemos elegido el sonar 0 que se corresponde con uno de los laterales y nos da 1,87 metros.

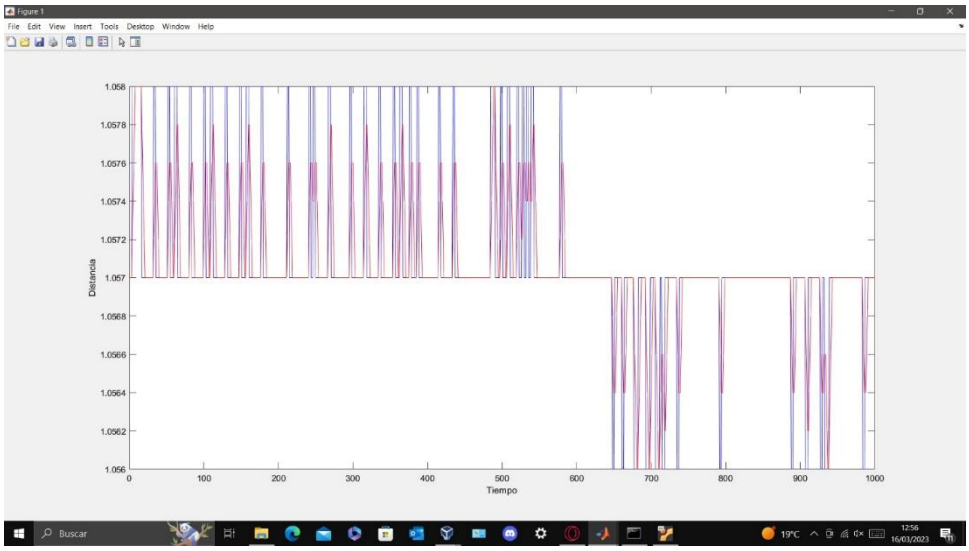
C con el real:



En este caso el codigo utilizado es el mismo que para el ejercicio en el simulado con la diferencia de que tenemos que cambiar algunos parámetros ya que ahora no estamos llamando a robot0 y tenemos que activar otros sensores.

A diferencia del robot simulado, aquí sí que podemos ver un poco de ruido, aunque solo es de 0,001 metro. Esto se puede deber no solo al ruido del propio robot, sino que el entorno influye en la toma de medidas. Aun así, al ser una diferencia vemos que la media el máximo y el mínimo apenas se mueven por lo que la varianza es mínima.

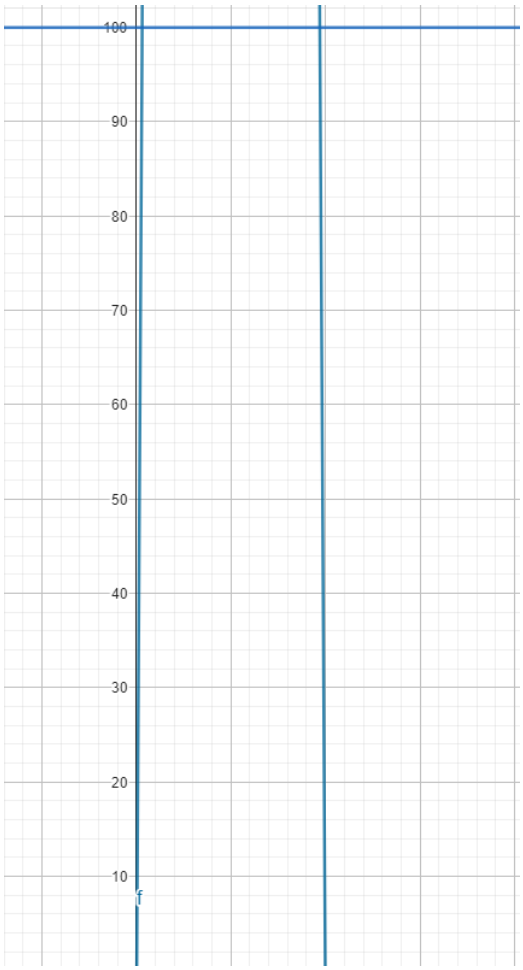
D con el real:



Aquí el código es el mismo que en el simulado con la diferencia de los parámetros que hacen referencia al robot ya que ahora no es robot0. Esto se puede ver en Laser4

Esta grafica nos muestra 2 graficas una con la media móvil y otra con simplemente los datos. Esto lo hemos hecho para comprobar que con la media los valores sufren mucha menos desviación, aunque a diferencia del robot simulado, aquí sí que tenemos ruido generado tanto por el propio robot como por el entorno.

F) En el simulador, posicione al robot en la casilla X. Estando el robot perfectamente paralelo a las paredes de la celda, seleccione las medidas de los sensores sonar del robot que podrían resultar útiles para obtener las cuatro rectas que definen las paredes que lo rodean. Compruebe que las orientaciones de las rectas son paralelas dos a dos y perpendiculares entre ellas. Defina una función de calidad para obtener el grado de confianza de dichas paredes.

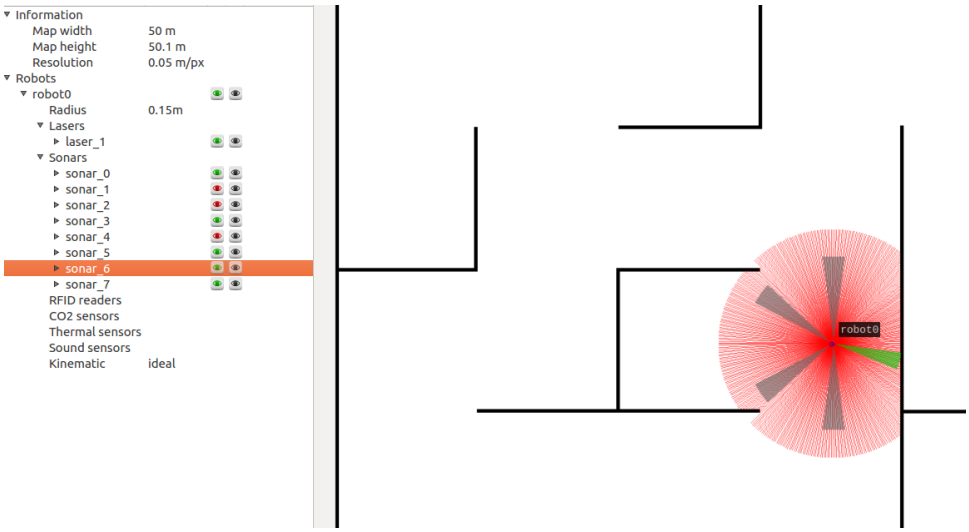


En nuestro caso hemos basado la función de calidad en lo cerca o lejos que se encuentra de una pared. Como se puede ver en caso de que este muy muy cerca o demasiado lejos (Aunque dentro del rango del sonar) nuestro grado de confianza es prácticamente 0, pero en cuanto nos separamos un poco de esos límites, vemos como la confianza es bastante alta llegando casi 100 (es 99,95).

Recalcamos que el área a tener en cuenta es la delimitada por las dos líneas azules ya que el resto de los datos se saldrían completamente de las capacidades del sonar por lo que siempre será 0 en esos casos.

G) Diseñe una función que indique, mediante un código, el número de paredes que se encuentra el robot en sus laterales. La codificación que se puede emplear para proporcionar la salida es la que se muestra en la Figura 4. Además, indique el grado de confianza basado en la función de calidad definida en el apartado anterior.

Para poder completar este ejercicio y detectar las paredes con el robot lo primero que hemos hecho es suscribirnos a los diferentes sonars que vamos a utilizar, en nuestro caso serán los siguientes (Los que hemos puesto visibles en la imagen).



Una vez que estamos suscritos a los sonars lo que hacemos es guardar los mensajes que estos dan en distintas variables, en concreto guardamos el apartado de Range_ ya que es este el que nos indica la distancia a la que está un objeto.

Tras esto ya simplemente nos queda identificar cada mensaje con la pared en concreto a la que jhace referencia y en caso de que esta exista mandamos un mensaje indicando la orientacion del objeto detectado. Para esto usamos el siguiente código:

```
Si (izqDist < distm y izqDist2 < distm) entonces
Mostrar "Hay una pared a la izquierda"
Fin Si
Si (derDist < distm) entonces
Mostrar "Hay una pared a la derecha"
Fin Si
Si (frontDist < distm) entonces
Mostrar "Hay una pared enfrente"
Fin Si
Si (backDist < distm) entonces
Mostrar "Hay una pared detras"
Fin Si
Fin
```

Todo esto mencionado con anterioridad se puede encontrar en el archivo Parte2.m

Para el caso del robot real el código es el mismo cambiando las líneas que hacen referencia a este ya que ahora no estaríamos usando el robot0.

H) Comprueba los resultados de la función diseñada en el apartado anterior tanto con el simulado STDR como con el robot real y con las 16 posibles combinaciones. Complete la tabla siguiente indicando los resultados.

Para calcular el grado de confianza en el simulado hemos visto que al no contar con factores externos el cálculo es exacto es por eso que hemos puesto un 100% en todos los casos.

En cambio, al pasar al robot real vemos que por cada pared detectada los valores pueden verse modificados en un 0,05%, esto lo comprobamos gracias a las distintas pruebas y las gráficas de ejercicios anteriores. Tras ver esto hemos calculado que el grado de confianza será X * 0,05 siendo X el número de paredes detectadas

Robot real/simulado	Combinación Real	Combinación detectada	Grado confianza
Simulado	0	0	100%
Simulado	1	1	100%
Simulado	2	2	100%
Simulado	3	3	100%
Simulado	4	4	100%
Simulado	5	5	100%
Simulado	6	6	100%
Simulado	7	7	100%
Simulado	8	8	100%
Simulado	9	9	100%
Simulado	10	10	100%
Simulado	11	11	100%
Simulado	12	12	100%
Simulado	13	13	100%
Simulado	14	14	100%
Simulado	15	15	100%
Simulado	16	16	100%
Real	0	0	100%
Real	1	1	99,95%
Real	2	2	99,95%
Real	3	3	99,95%
Real	4	4	99,95%
Real	5	5	99,9%
Real	6	6	99,9%
Real	7	7	99,9%
Real	8	8	99,9%
Real	9	9	99,9%
Real	10	10	99,9%
Real	11	11	99,85%
Real	12	12	99,85%
Real	13	13	99,85%
Real	14	14	99,85%
Real	15	15	99,85%
Real	16	16	99,8%

3.1.3 Sensores de distancia láser

A) Indique y describa la información que nos ofrece el mensaje disponible en el topic laser. Muestre algún ejemplo de captura.

En el caso del láser también se muestra el rango máximo y rango mínimo, pero a diferencia del sonar, no te da una sola distancia a la que está el objeto, sino que devuelve un array con 400 valores correspondientes a los 400 láser que contiene el robot.

```
    MessageType: 'sensor_msgs/LaserScan
         Header: [1×1 Header]
       AngleMin: -3.1416
       AngleMax: 3.1416
  AngleIncrement: 0.0157
   TimeIncrement: 0
       ScanTime: 0
       RangeMin: 0.1500
       RangeMax: 8
         Ranges: [400×1 single]
    Intensities: [0×1 single]
```

Esta imagen es el mensaje general que te devuelve a nivel general, pero si quieres ver los valores de distancia devuelve algo como lo mostrado en la próxima foto.

```
   Inf
   Inf
   Inf
   Inf
   Inf
   Inf
   Inf
   Inf
7.3500
7.2500
7.2500
7.4000
7.5000
7.6500
7.8000
7.9500
   Inf
   Inf
   Inf
   Inf
   Inf
   Inf
   Inf
   Inf
   Inf
   Inf
   Inf
   Inf
   Inf
   Inf
```

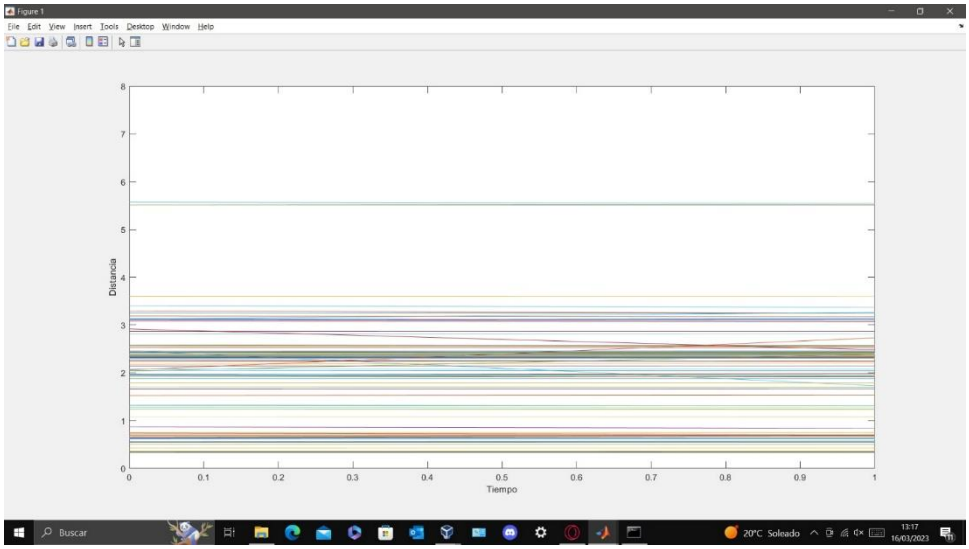
B) Repita los pasos 2-4 y 6-8 del apartado anterior con el sensor láser.

B.2)

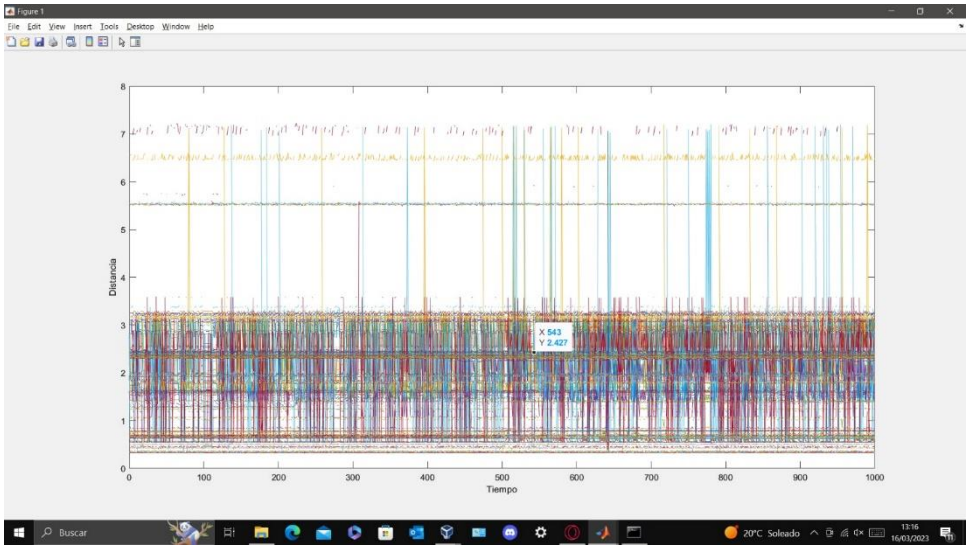
Se ha posicionado a cinco metros y se muestra el valor de todos los láseres, aunque algunos devuelven inf ya que ellos en concreto no están detectando nada.

B.3 y B.5.3) Obtenga 1.000 medidas de distancia del sensor sónar elegido y dibuje en una gráfica (comando plot en Matlab) la distancia medida. ¿Son estables las medidas? ¿Hay ruido en la medida? En caso afirmativo, calcule el valor máximo, medio y la varianza del ruido.

Para hacer esto el código es igual al ya explicado en el apartado 3 del sonar, con la única modificación que ahora nos suscribimos al laser y la gráfica muestra los datos de los 400 por lo que queda un poco ilegible, aunque en este caso al ser el robot simulado no importa ya que es una gráfica lineal ya que no existe ruido.



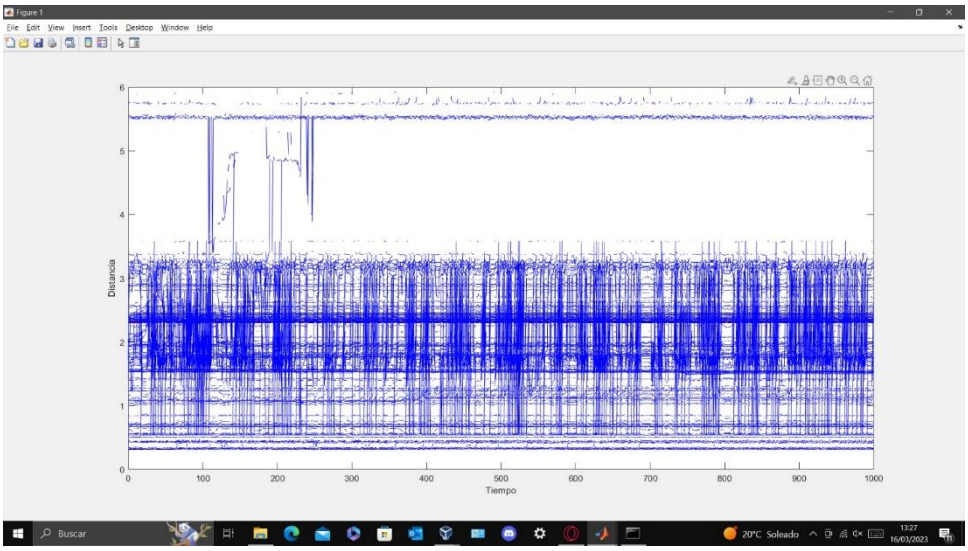
Por otro lado, en el robot real sí que existe ruido debido a factores tanto de la misma maquina como del propio entorno, es por eso que la gráfica queda con variaciones.



B.4 y B.5.4) Implemente un filtro media móvil con los últimos 5 valores de distancia y dibuje en una gráfica. ¿Son más estables las medidas que en el caso anterior? ¿Sería útil este método si el robot está en movimiento en lugar de permanecer estático?

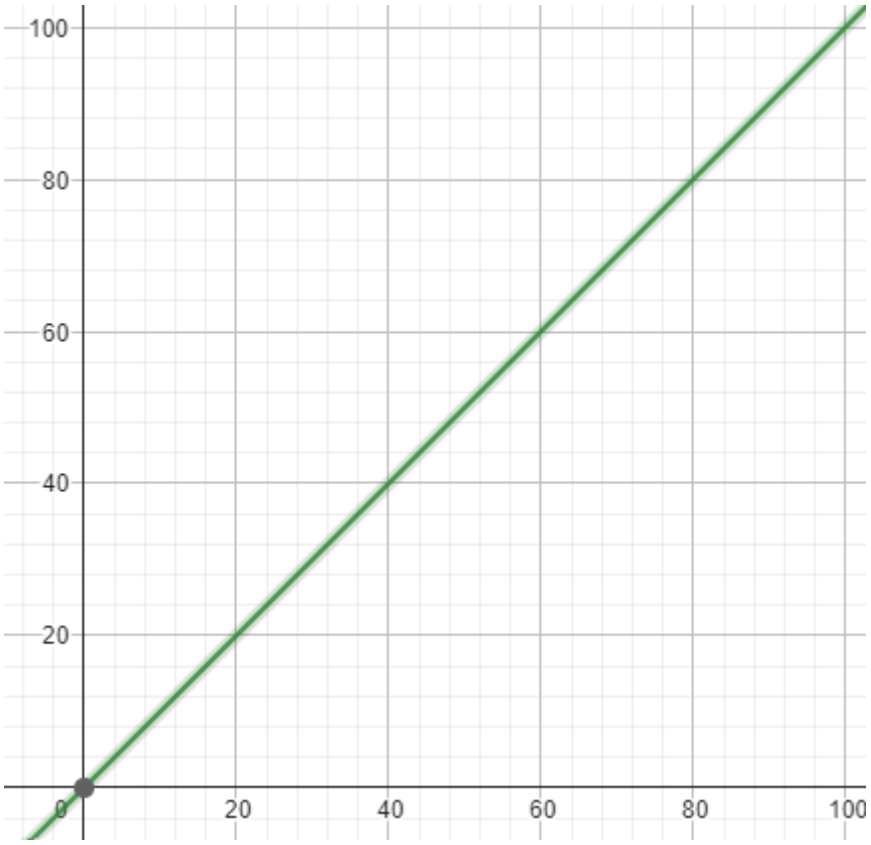
Para este ejercicio nuevamente el código es el ya explicado en el apartado del sonar, pero modificando las variables que hacen referencia a este.

La media móvil del sonar, al no tener interferencias vuelve a quedarnos una gráfica lineal igual a la mostrada en el apartado anterior, en cambio en el robot real sí que existe ruido por lo que nos quedaría algo como esto:



B.6) En el simulador, posicione al robot en la casilla X. Estando el robot perfectamente paralelo a las paredes de la celda, seleccione las medidas de los sensores sonar del robot que podrían resultar útiles para obtener las cuatro rectas que definen las paredes que lo rodean. Compruebe que las orientaciones de las rectas son paralelas dos a dos y perpendiculares entre ellas. Defina una función de calidad para obtener el grado de confianza de dichas paredes empleando, por ejemplo, la relación entre las diferentes pendientes.

Para este caso hemos pensado que el grado de confianza depende de la cantidad de láser que detecten una pared teniendo en cuenta que 100 son el máximo que podrían detectarla y 0 el mínimo. Es por esto que se nos quedaría una función exponencial como la siguiente:



En nuestra función que veremos más adelante, consideramos que 60 láseres como mínimo deben detectar una pared para poder decir que esta existe por lo que siempre el grado de confianza será mayor a 60%

B.7) Diseñe una función que indique, mediante un código, el número de paredes que se encuentra el robot en sus laterales. La codificación que se puede emplear para proporcionar la salida es la que se muestra en la Figura 4. Además, indique el grado de confianza basado en la función de calidad definida en el apartado anterior.

En esta función lo primero que hemos hecho es suscribirnos al laser.

Tras esto lo que hacemos es dividir los 400 lasers entre los 4 cuadrantes (delante, detrás, izquierda y derecha). Para conseguir esto usamos el siguiente código:

```
cont3 = 0;
Para i desde 1 hasta 50 hacer
Si (ranges(i) < dist) entonces
cont3 = cont3 + 1;
Fin Si
Fin Para
%cuadrante 2
cont2 = 0;
Para i desde 350 hasta 400 ha
Si (ranges(i) < dist) entonces
cont2 = cont2 + 1;
Fin Si
Fin Para
izqda = cont3 + cont2;
%cuadrante 1
detras = 0;
Para i desde 50 hasta 150 hac
Si (ranges(i) < dist) entonces
detras = detras + 1;
Fin Si
Fin Para
%cuadrante 4
derecha = 0;
Para i desde 150 hasta 250 ha
Si (ranges(i) < dist) entonces
derecha = derecha + 1;
Fin Si
Fin Para
delante = 0;
Para i desde 250 hasta 350 ha
Si (ranges(i) < dist) entonces
delante = delante + 1;
Fin Si
Fin Para
Fin
```

Una vez tenemos esto lo que hacemos es si la cantidad de láser detectando la pared que les corresponde es mayor a 60 y si esto se cumple mandamos un mensaje indicando la pared detectada.

Todo este código corresponde al fichero EldelLaser.m.

B.8) Comprueba los resultados de la función diseñada en el apartado anterior tanto con el simulado STDR como con el robot real y con las 16 posibles combinaciones. Complete la tabla siguiente indicando los resultados.

Robot real/simulado	Combinación Real	Combinación detectada	Grado confianza
Simulado	0	0	100%
Simulado	1	1	92%
Simulado	2	2	92%
Simulado	3	3	92%
Simulado	4	4	84%
Simulado	5	5	84%
Simulado	6	6	84%
Simulado	7	7	84%
Simulado	8	8	84%
Simulado	9	9	84%

Simulado			10	10	84%
Simulado			11	11	76%
Simulado			12	12	76%
Simulado			13	13	76%
Simulado			14	14	76%
Simulado			15	15	76%
Simulado			16	16	68%
Simulado	0	0	100%		
Simulado	1	1	92%		
Simulado	2	2	92%		
Simulado	3	3	92%		
Simulado	4	4	84%		
Simulado	5	5	84%		
Simulado	6	6	84%		
Simulado	7	7	84%		
Simulado	8	8	84%		
Simulado	9	9	84%		
Simulado	10	10	84%		
Simulado	11	11	76%		
Simulado	12	12	76%		
Simulado	13	13	76%		
Simulado	14	14	76%		
Simulado	15	15	76%		
Simulado	16	16	68%		

Para el cálculo del grado confianza como ya hemos dicho, hemos utilizado la cantidad de láser que detectan una pared, al hacer varias pruebas hemos visto que sueles ser 68 sobre 100 los que lo detectan, es decir un 68% de grado de confianza por pared y un 100% cuando esta no exista. Teniéndolo en cuenta, lo que hemos hecho es (68X+100(4-x))/4 siendo x el número de paredes detectadas.

C). ¿Tiene más o menos ruido que el sensor sónar?

Consideramos que el ruido es mayor al existir más cantidad de sensores que se pueden ver perjudicados (400), teniendo en cuenta que individualizamos cada rayo ya que cada uno puede tener sus propias perturbaciones

D) ¿Cómo es el grado de confianza empleando este sensor?

Esta cuestión está explicada en el apartado B.8

3.2 ACTUADORES EN ROS

A) Diseñe una función avanzar que reciba como parámetro de entrada la distancia a avanzar (2m, 4m…).

El código esta en la carpeta ActuadoresRos y esta explicada línea por línea aquí se hace una mención de lo más destacado.

En esta función avanzar le paso como parámetro la distancia recorrida.

1) En primer lugar nos conectamos a las correspondientes IPs y iniciamos ros

2) En segundo lugar declaramos los subscribers y los Publisher en este caso usando cmd_vel del tipo geometry_msgs/Twist, luego generamos los diferentes mensajes y le pasamos a X una velocidad de 0.3 para poder tomar los datos

3) Luego obtenemos la distancia que se ha desplazado el robot en un bucle y en el momento que esa distancia sea superior a la distancia recorrida que es la que hemos introducido paramos el robot y salimos del bucle de control

B) Diseñe una función girar que se reciba como parámetro de entrada el ángulo a girar (45º, 90º,…).

El código esta en la carpeta ActuadoresRos y esta explicada línea por línea aquí se hace una mención de lo más destacado.

En esta función girar le pasamos como parámetro un angulo

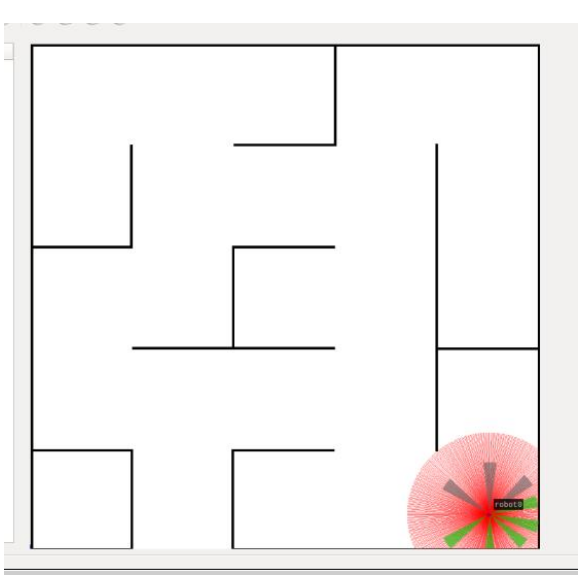
1) En primer lugar nos conectamos a las correspondientes IPs y iniciamos ros

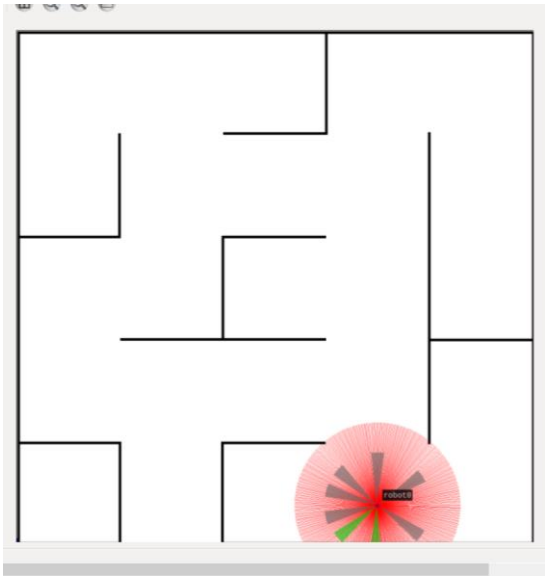
2) En segundo lugar declaramos los subscribers y los Publisher en este caso usando cmd_vel del tipo geometry_msgs/Twist, luego generamos los diferentes mensajes y le pasamos a dependiendo del angulo si es mayor que 0 una velocidad positiva y si es menor que 0 una velocidad negativa.

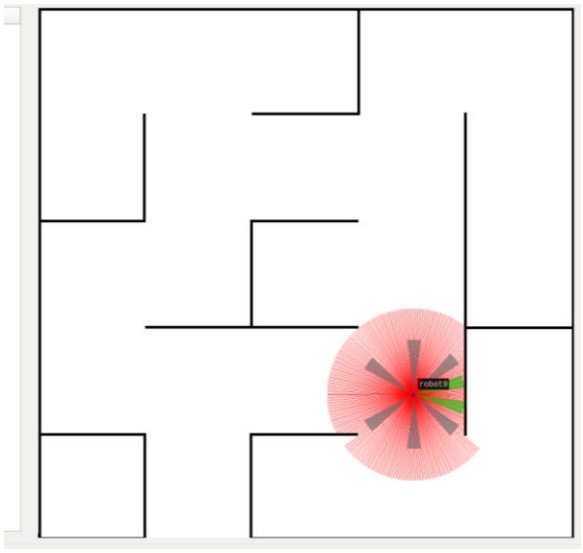
3) Un paso importante será pasar a radianes para poder trabajar con el robot por lo que multiplicamos el ángulo por pi y lo dividimos entre 180 grados , realizando así la conversión, así sabemos que en el momento que yaw supere al ángulo introducido en radianes lanzamos el mensaje para parar el robot.

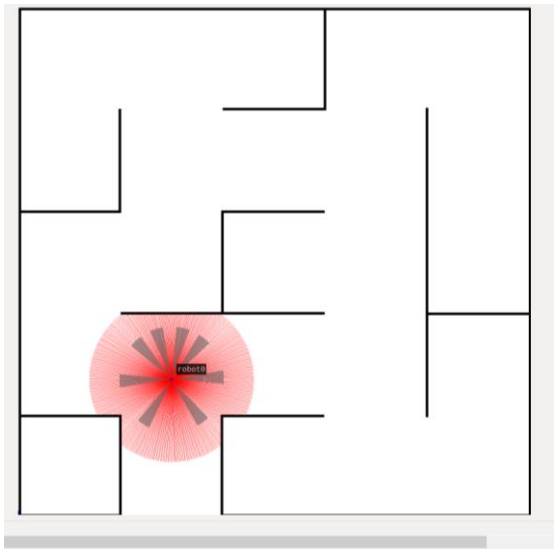
C)Empleando el simulador STDR, y utilizando las funciones avanzar y girar, navegue con el robot desde la esquina inferior derecha (4,-4,0) hasta la salida utilizando los puntos centrales de las casillas como coordenadas de destino

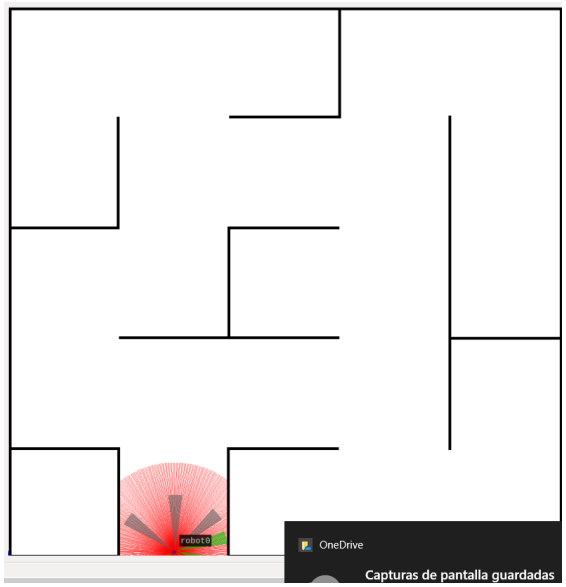
Usando las dos funciones anteriores funciones











El objetivo es llevar el robot a la posición de salida , solo se desplaza en el eje x y yax por lo que hay que hacer que gire 180 grados para colocar el eje x y poder avanzar , luego avanzamos 4 metros y giramos -90 grados , luego avanzamos de nuevo otros 5 metros y volvemos a girar 90 grados luego avanzamos 18 metros, y giramos el robot 90 grados de nuevo y ya avanzamos hasta la salida 7 metros, luego la secuencia seria:

Girar(180)→Avanzar(8)→girar(90)→avanzar(8)→

girar(90)→avanzar(18)→girar(90)→avanzar(8)

C) 4. Realice un recorrido con el robot real concatenando los siguientes tramos: o 1 tramo recto de 2m o 1 giro de 90º o 1 tramo recto de 1m o 1 giro de -90º o 1 tramo recto de 1m Compruebe el error final que se ha obtenido con el robot real. ¿Cuál es el error de odometría global que se ha cometido?

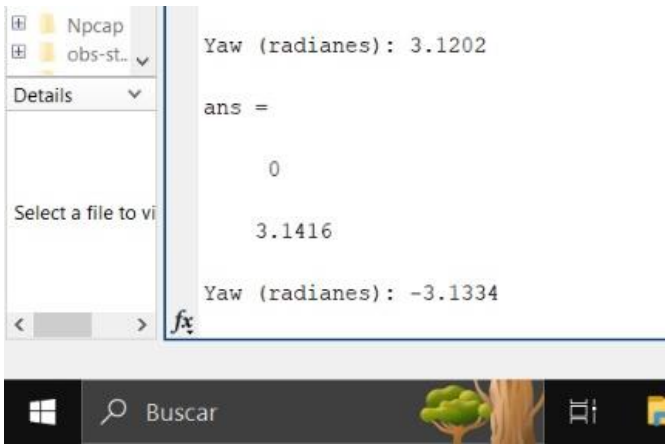
El código se puede ver en la carpeta ActuadoresRos y se llama ActuadoresRosReal.m

Respecto al error de odometría con el robot real es ligeramente mayor pero aun así sigue siendo muy breve tal y como se puede ver en la siguiente captura:

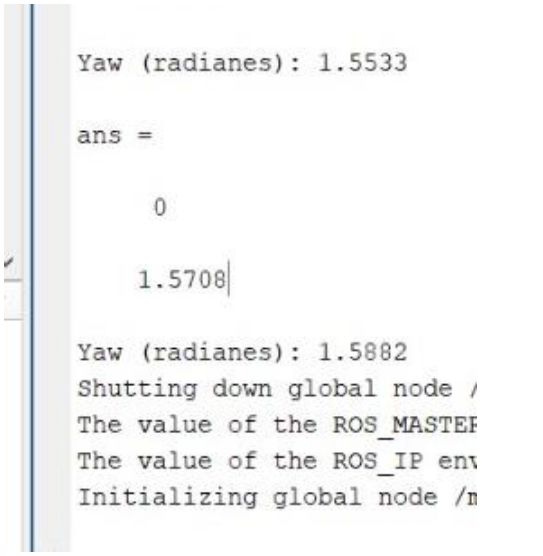
```
dist =

    1.0290

    1.0290
```

Primero le pasamos que avance un metro y avanza exactamente hasta 1.0290 luego hay un error de 0,0290

Otro análisis que se saca es que con la odometria yaw va desde 0 hasta pi ósea desde 0 a 3,1416 rad , en el momento que llega a pi vuelve hasta 0 pero en este caso los valores son negativos y iría desde -3,1416 hasta 0. Un error de la odometria es el valor que le pasa a pi



Yo he hecho una conversión exacta de 180 grados a radianes `que seria 3.1416 pero como se puede ver en la imagen la odometria solo lee hasta 3.1202 y ya empieza con los valores negativos hasta 0 por lo que hay otro pequeño margen de error de odometria , y si le pasas al robot exactamente el valor de pi , no lo va a leer bien hay que pasarle un valor un poco mas pequeño yo en este caso le estoy pasando 3.1



Cuando le paso que gire 90 grados al hacer la conversión a radianes pasa a valer 1.5708 y en el momento que yaw es mayor , como se puede ver en la imagen deja de girar.

Luego como conclusión final basado en las capturas es que con la odometria podemos obtener valores con una precisión de casi del 98% y márgenes de error de 0,01-0,02.

4. Conclusiones

En resumen, los sensores de odometría, sensores láser y sensores ultrasónicos son herramientas clave en el campo de la robótica móvil. Con la ayuda de MATLAB y ROS, estos sensores pueden integrarse y utilizarse para permitir que los robots móviles comprendan su entorno y tomen decisiones en consecuencia.

Los sensores de odometría se utilizan para estimar la posición y la orientación de un robot en función de la distancia recorrida por sus ruedas. Los sensores láser son útiles para realizar un mapeo detallado del entorno y detectar obstáculos. Los sensores ultrasónicos, por otro lado, son adecuados para detectar obstáculos cercanos y medir la distancia a ellos.

Utilizando MATLAB y ROS, se pueden programar y controlar estos sensores para que trabajen juntos y proporcionen información precisa sobre el entorno circundante. Esto permite a los robots móviles navegar con seguridad y eficacia en entornos desconocidos.

En general, los sensores de odometría, sensores láser y sensores ultrasónicos son herramientas críticas en la robótica móvil y su integración con MATLAB y ROS permite la creación de sistemas robóticos más avanzados y efectivos.

5. Bibliografía

Documentos puestos a nuestra disposición en el blackboard.

https://es.mathworks.com/videos/understanding-control-systems-part-3-components-of-a-feedback-control-system-123645.html

https://es.mathworks.com/help/matlab/ref/plot.html

https://es.mathworks.com/help/matlab/ref/for.html

https://www.geogebra.org/?lang=es-ES

https://chat.openai.com/chat