U. CARLOS III DE MADRID

Proyecto final de robótica industrial

Trabajo realizado por:

Miguel Ángel de Miguel Paraíso 100292950

Ainoa Zugasti Hernández 100292754

Índice de la memoria

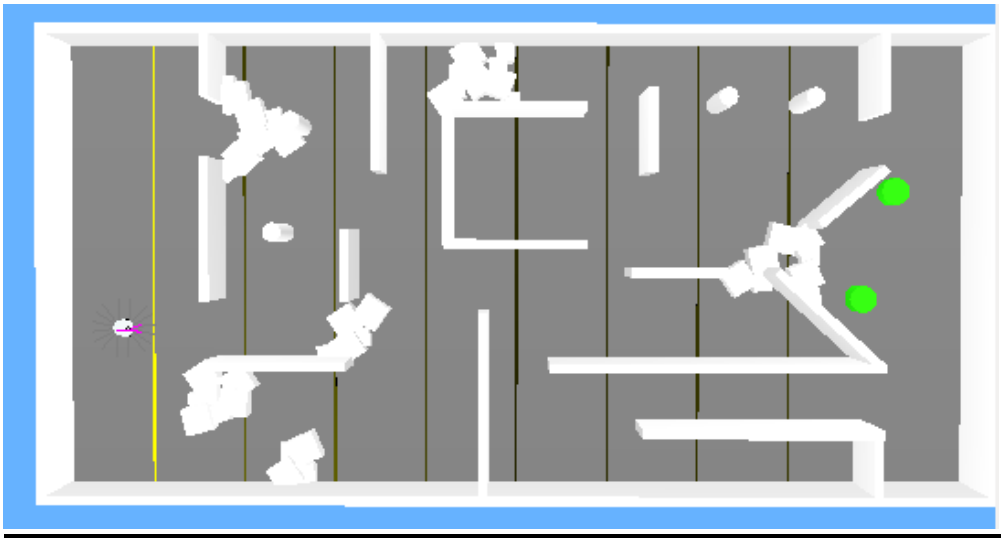
Descripción del problema

El objetivo de este proyecto es conseguir que un robot similar al de la imagen consiga traspasar el laberinto entero, sin chocarse, ni quedarse atascado en ninguno de los puntos y obstáculos que pueda encontrarse en su interior.

Al salir de dicho laberinto, se encontrará a dos personas (representadas mediante dos cilindros verdes), a las cuales deberá rescatar. Cuando encuentre a una, se situará 2 segundos en frente de ella y posteriormente dará una vuelta sobre sí mismo. Después se dirigirá a la siguiente.

Cuando el robot termine el "salvamento" de las dos personas, volverá a su punto de partida recorriendo de nuevo el laberinto, esta vez en sentido inverso.

Un ejemplo donde podremos visualizar el problema planteado es este:



La programación de esta simulación se realizará en lenguaje de programación C++, mediante la herramienta Qt Creator, cuyos comentarios serán hechos mediante la técnica DoxyGen.

Para poder visualizar la programación del controlador se utilizará el programa Webots 7.2.4, donde mediante su interfaz gráfica se podrán obtener los distintos mundos de prueba, en los que realizarán mas adelante las pruebas necesarias para obtener los resultados de fiabilidad de nuestro controlador.

Diagrama de estados

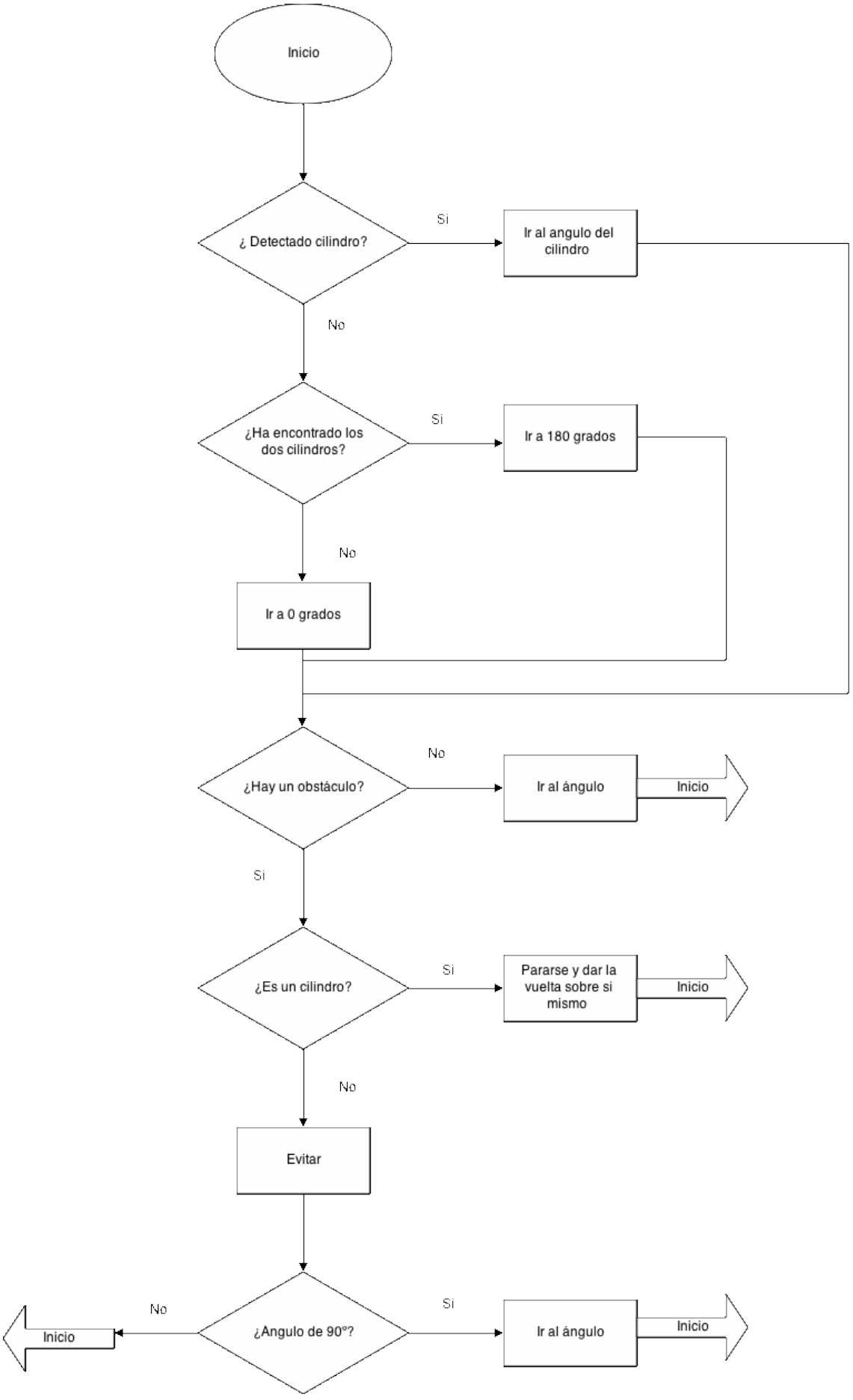
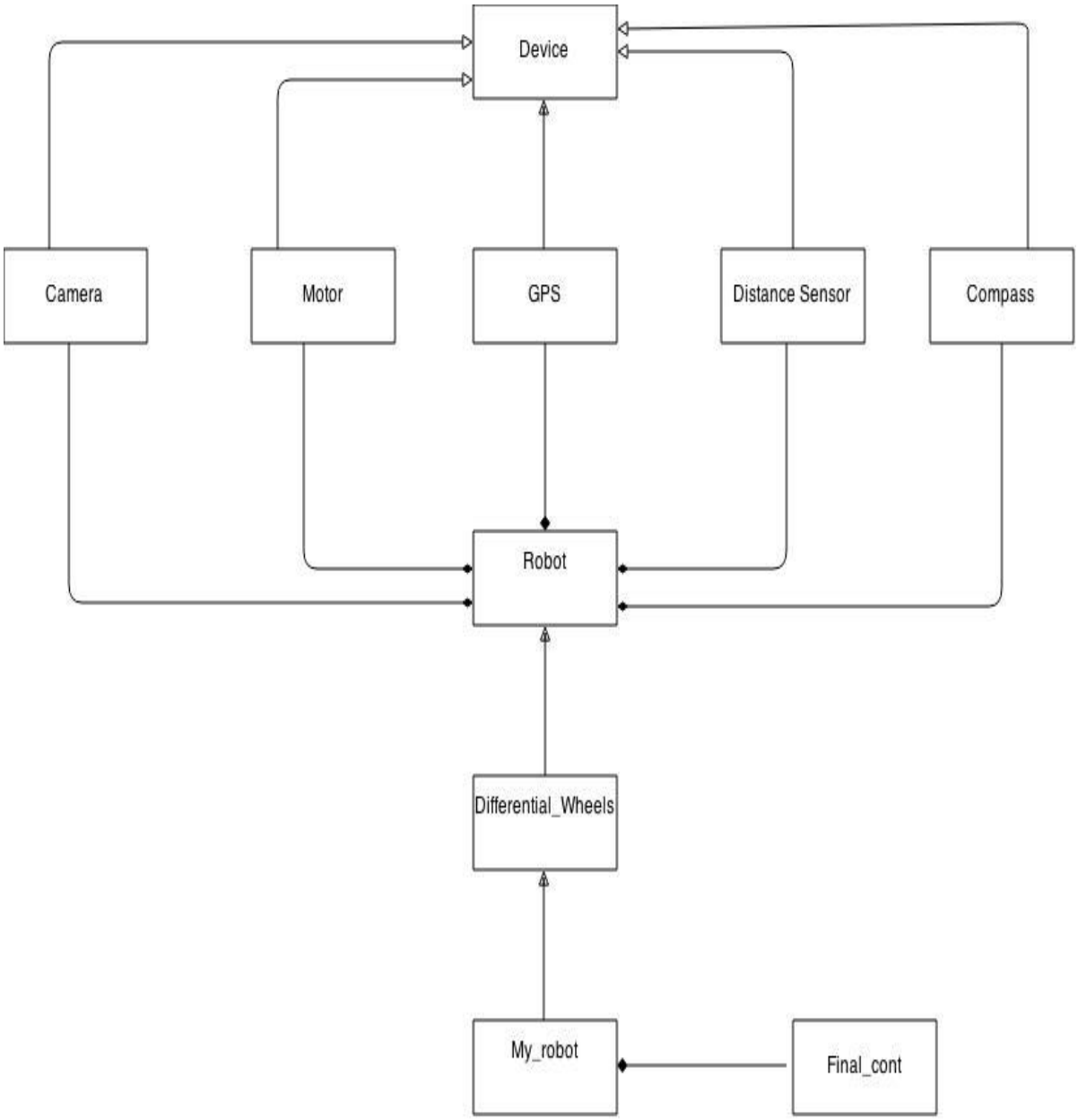


Diagrama UML.

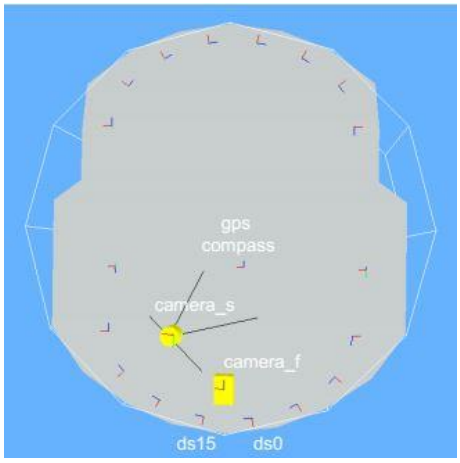


Proceso de diseño y soluciones empleadas

Para justificar el diseño que hemos utilizado deberemos ir por partes:

- Llevar el robot hacia la persona a rescatar:
  Para conseguir que el robot se dirija hacia las personas teniendo que atravesar el mundo, en un primer momento habíamos decidido hacerlo mediante el GPS. Esta opción fue descartada porque el GPS interno que incluye este modelo de robot tiene un error de 10 metros, medida que se corresponde con el largo de nuestros entornos, por lo que resultaría difícil y costoso situar el robot, ya que habría que realizar una media de todas las medidas del GPS y esto ralentizaría mucho el robot.
  Por lo tanto, la opción elegida es utilizar la brújula para llevar el robot hacia el otro lado del laberinto. Esta ha sido calibrada, es decir, el ángulo nuevo será igual a 45 grados menos el ángulo por defecto de la brújula, de tal forma que el resultado sea un ángulo igual a 0 grados (que corresponde al otro lado del laberinto donde se encuentran los cilindros), y será el que comenzará a seguir el robot desde el comienzo de la simulación.
  Además, por otro lado, para evitar problemas a la hora de realizar operaciones, el rango de la brújula se ha limitado de -180º a 180º.
  Para poder posicionar el robot y situar el primer cilindro en el mapa se utiliza odometría que nos proporcionara un resultado sin errores acumuladores puesto que no existen errores en los encoders.

- Evitar obstáculos en el interior del laberinto:
  Para poder evitar dichos obstáculos, hemos utilizado los sensores propios del robot (desde el sensor 0 al 15 incluyéndolos todos) y las distancias que estos nos ofrecen.
  La mecánica para poder evitar un obstáculo es sencilla. Cuando el robot detecta en alguno de sus sensores una distancia mayor que la distancia que el usuario impone como crítica (es decir, que se encuentra muy cerca de algún objeto), el robot gira hacia el lado contrario para no chocarse posteriormente.



  Esto puede hacerse efectivo en nuestro programa primeramente, llamando a la función "`choose_situation`" donde, como hemos explicado anteriormente, dependiendo de los sensores que detecten una distancia crítica

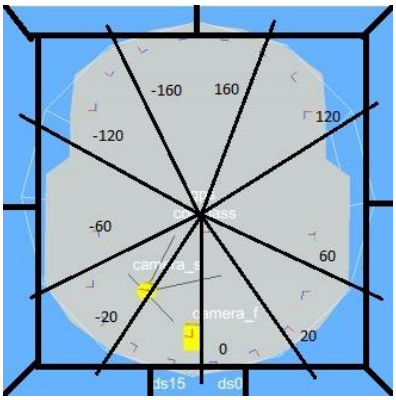
("FOLLOWING_DISTANCE"), elegirá seguir la pared derecha o la pared izquierda.
Por otro lado, cuando la pared que el robot esta siguiendo y el ángulo de destino forman 90 grados, el robot se separará de la pared y seguirá el ángulo deseado.
Estas funciones estarán regidas por la función que hemos denominado "`send_mode_to_motors`", que controlará tanto la velocidad derecha como la velocidad izquierda que deberá adoptar el robot.
En un primer momento en ciertos puntos del laberinto (cuando el robot se encontraba entre paredes laterales y frontales) este no podía continuar porque se encontraba atrapado, sin embargo, hallamos una solución ante este problema: Cuando se encuentre en dicha situación hemos programado el robot para que se pare y gire sobre si mismo 180 grados.

- <u>Encontrar una persona a rescatar:</u>

Cuando el robot atraviesa todo el laberinto, su principal objetivo es encontrar a la persona que necesita ser rescatada, para ello el robot calcula a que ángulo se encuentra esta de sí mismo, y comienza a seguir dicho ángulo.
Para ello hemos utilizado las cámaras que nos ofrece el propio robot. (Forward y Spherical).

1. <u>Cámara "Spherical":</u>

   Esta herramienta nos proporcionará una doble utilidad.
   Por un lado, podremos conseguir que el robot sepa en que zona se encuentra un cilindro verde, mediante el conteo de pixeles de dicho color que se puede encontrar en la imagen que ve la cámara (tomando siempre en cuenta que puede existir un mínimo error en dicho conteo).
   Se divide la cámara en nueve zonas, y se calcula los porcentajes de verde que existen en cada zona. A partir de la zona donde se encuentra el porcentaje más alto, se hace una estimación del ángulo donde se encuentra el cilindro.



   Por otro lado, nos permitirá "*anular*" ciertas zonas de la cámara para que el robot no vaya al mismo cilindro varias veces y pueda continuar la simulación encontrando a la otra persona. Como se conoce la posición del cilindro y del robot, se puede calcular el ángulo que hay entre ellos e igualar a cero los pixeles verdes de las zonas de la cámara que apuntan al primer cilindro.

La reacción del robot ante esto, es tratar al cilindro como un obstáculo cualquiera como pudiera ser la pared.

2. Cámara "Forward" :
   Esta cámara nos permitirá detectar un cilindro que se encuentre frente al robot, para poder ir hacia él.
   De forma análoga al caso anterior, cuando el color verde en la imagen supere un tanto por cierto definido por el programador, emitirá una señal de "`cilynder_found`" (o cilindro encontrado en español), para realizar posteriormente la vuelta sobre si mismo y la detección durante dos segundos frente a él.

- Acciones al detectar a una persona:
  Cuando la cámara Forward detecta una persona, realiza las acciones de pararse frente a ella durante dos segundos y girar.
  Además, si es el primer cilindro que ha encontrado, almacena su posición para no volver a ir a él y si el segundo se activa una variable para volver a la posición de inicio.
  El problema encontrado en esta ocasión, fue que el robot al girar dejaba de visualizar por su cámara el cilindro, por lo que no terminaba de girar. Para ello, la solución encontrada es activar una variable ("`stopped`") que hará que se siga metiendo en esta función hasta que termine de dar la vuelta.

- Llevar al robot al punto de partida:
  Para devolver el robot a su punto de partida, también utilizaremos la brújula, la única diferencia que podemos encontrar con respecto al caso contrario, es decir, al de ir hacia los cilindros es que ahora, el ángulo que seguirá el robot será el de 180 grados.
  Para evitar un problema en la forma de programar la función "`go_to_angle`", se invierte el ángulo dado por la brújula para que ahora los 0 grados, sean los antiguos 180, ya que en caso contrario, el robot empezaría a dar vueltas y no conseguiría realizar la acción propuesta.
- Evitar la ralentización del controlador
  Para que el tiempo de simulación sea el adecuado (x1.0) hay que evitar realizar demasiados cálculos en cada ciclo del programa.
  Por lo tanto, hemos dividido el programa en dos ciclos más pequeños que deberán ser completados para que realice el programa total.
  En los ciclos pares se ejecuta la odometría (que tiene cálculos de senos y cosenos) y la actualización de la lectura de la brújula mientras que en los impares se realizan las operaciones propias de la cámara.

Descripción de resultados.

Para probar la fiabilidad de nuestro controlador hemos realizado dos pruebas en cada uno de los mundos proporcionados por el profesor.
Los resultados obtenidos se recogen en las siguientes tablas:

Mundo 1	Prueba 1	Prueba 2
Distancia recorrida	16 metros	16 metros
Tiempo empleado en llegar al final	2 minutos 35 segundos	1 minuto 10 segundos
Tiempo en el que encuentra el primer cilindro	2 minutos 40 segundos	1 minuto 26 segundos
Tiempo en el que encuentra el segundo cilindro	3 minutos 40 segundos	-
Tiempo en volver al punto de partida	-	-
Incidencias	Se queda atrancado al volver	Se queda en el final buscando el segundo

Mundo 2	Prueba 1	Prueba 2
Distancia recorrida	16 metros	16 metros
Tiempo empleado en llegar al final	2 minutos 37 segundos	-
Tiempo en el que encuentra el primer cilindro	3 minutos 0 segundos	-
Tiempo en el que encuentra el segundo cilindro	3 minutos 40 segundos	-
Tiempo en volver al punto de partida	-	-
Incidencias	No le da tiempo a volver	No consigue llegar al final

Mundo 3	Prueba 1	Prueba 2
Distancia recorrida	14 metros	16 metros
Tiempo empleado en llegar al final	-	-
Tiempo en el que encuentra el primer cilindro	-	-
Tiempo en el que encuentra el segundo cilindro	-	-
Tiempo en volver al punto de partida	-	-
Incidencias	No consiguel llegar al final	No ve el cilindro

Mundo 4	Prueba 1	Prueba 2
Distancia recorrida	16 metros	16 metros
Tiempo empleado en llegar al final	3 minutos 15 segundos	1 minuto 10 segundos
Tiempo en el que encuentra el primer cilindro	2 minutos 15 segundos	1 minuto 10 segundos
Tiempo en el que encuentra el segundo cilindro	-	1 minuto y 50 segundos
Tiempo en volver al punto de partida	-	-
Incidencias	Se queda atrancado	No le da tiempo a volver

Mundo 5	Prueba 1	Prueba 2
Distancia recorrida	16 metros	16 metros
Tiempo empleado en llegar al final	2 minutos 0 segundos	2 minutos 0 segundos
Tiempo en el que encuentra el primer cilindro	3 minutos 0 segundos	2 minutos 20 segundos
Tiempo en el que encuentra el segundo cilindro	-	-
Tiempo en volver al punto de partida	-	-
Incidencias	Se queda en el final buscando el segundo	Se queda en el final buscando el segundo

Mundo 6	Prueba 1	Prueba 2
Distancia recorrida	16 metros	16 metros
Tiempo empleado en llegar al final	1 minutos 50 segundos	2 minutos 10 segundos
Tiempo en el que encuentra el primer cilindro	-	3 minutos 25 segundos
Tiempo en el que encuentra el segundo cilindro	-	-
Tiempo en volver al punto de partida	-	-
Incidencias	Se queda en el final buscando el primero	Se queda en el final buscando el segundo

Mundo 7	Prueba 1	Prueba 2
Distancia recorrida	16 metros	16 metros
Tiempo empleado en llegar al final	1 minutos 30 segundos	1 minutos 55 segundos
Tiempo en el que encuentra el primer cilindro	-	2 minutos 35 segundos
Tiempo en el que encuentra el segundo cilindro	-	-
Tiempo en volver al punto de partida	-	-
Incidencias	Se queda atrancado	Se queda en el final buscando el segundo

Mundo 8	Prueba 1	Prueba 2
Distancia recorrida	32 metros	16 metros
Tiempo empleado en llegar al final	50 segundos	1 minuto 05 segundos
Tiempo en el que encuentra el primer cilindro	50 segundos	1 minuto 05 segundos
Tiempo en el que encuentra el segundo cilindro	1 minutos 20 segundos	1 minutos 23 segundos
Tiempo en volver al punto de partida	2 minutos 16 segundos	3 minutos 56 segundos
Incidencias		

Mundo 9	Prueba 1	Prueba 2
Distancia recorrida	16 metros	14 metros
Tiempo empleado en llegar al final	1 minutos 50 segundos	1 minutos 08 segundos
Tiempo en el que encuentra el primer cilindro	4 minutos 10 segundos	1 minutos 23 segundos
Tiempo en el que encuentra el segundo cilindro	-	-
Tiempo en volver al punto de partida	-	-
Incidencias	Se queda en el final buscando el segundo	Se queda atrancado

Mundo 10	Prueba 1	Prueba 2
Distancia recorrida	16 metros	16 metros
Tiempo empleado en llegar al final	3 minutos 0 segundos	2 minuto 0 segundos
Tiempo en el que encuentra el primer cilindro	3 minutos 10 segundos	2 minuto 30 segundos
Tiempo en el que encuentra el segundo cilindro	-	-
Tiempo en volver al punto de partida	-	-
Incidencias	Se queda en el final buscando el segundo	Se queda en el final buscando el segundo

Conclusiones y futuras mejoras.

Una vez realizado el código y experimentado en todos los mundos los posibles resultados que podemos obtener de nuestro controlador, podemos deducir que este podría tener una serie de mejores:

En primer lugar, podría ajustarse mejor los parámetros para que el robot siga las paredes de una manera mas eficiente a la actual, y poder conseguir así, un tiempo menor de simulación o incluso conseguir que el robot, en todos los mundos, llegase a su objetivo (los cilindros verdes), pues como se puede observar, esto no ocurre en el 100% de los casos expuestos.

El hecho de mejorar esta característica también puede desembocar en que en determinadas zonas, nuestro robot no se quede atascado en situaciones complicadas como esquinas con obstaculos.

En segundo lugar, quizás podríamos encontrar una solución para que las cámaras integradas en el robot tuvieran una visión mas nítida del verde de los cilindros con niebla. Una solución que se nos ocurre es cambiar el tono de verde y ajustarlo mas adecuadamente al de los cilindros que representan a las personas de salvamento.

Por último, hemos comprobado que nuestro controlador, en situaciones de oscuridad, no funciona como debería, una futura mejora podría ser dedicarle mas tiempo a mejorar esta característica.

Como conclusión podríamos decir que hemos obtenido unos resultados óptimos de acuerdo a nuestros conocimientos acerca del robot, pero como es de esperar, nuestro controlador posee defectos que podrían mejorarse.