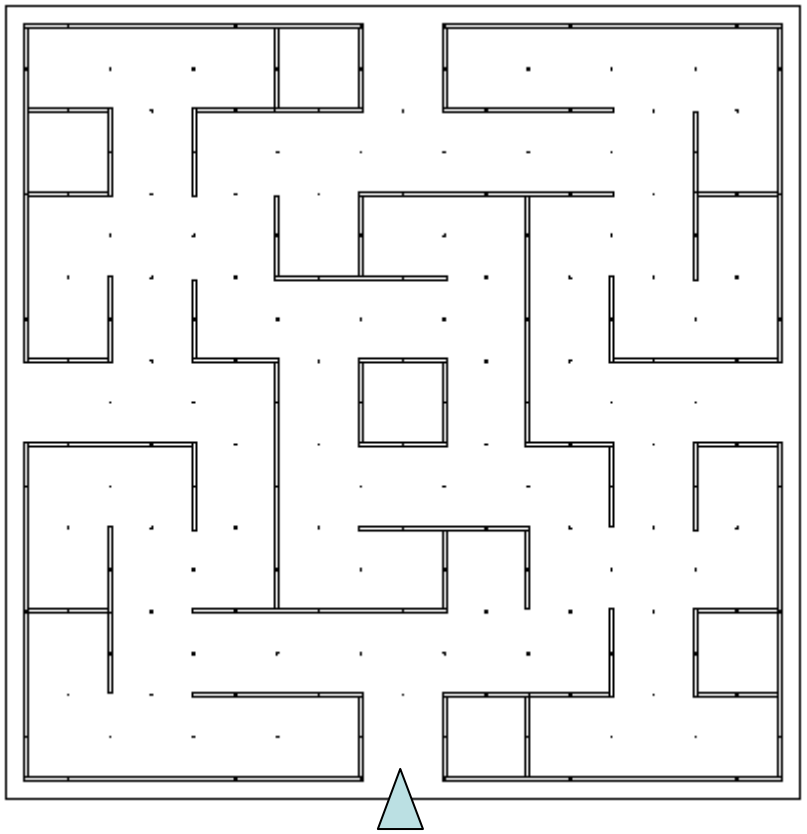


Estrategias de solución para la prueba del Laberinto, Madrid-Bot 2009

Salustiano Nieva
Juan Antonio Breña Moral

CONSEJERÍA DE EDUCACIÓN
Comunidad de Madrid

EDUCAMADRID

MADRIDBOT

Índice

“Divide et vinces”
Julio Cesar

Introducción

La prueba del Laberinto, es la clásica prueba de robótica que consiste en desarrollar un robot capaz de resolver un laberinto en el menor tiempo posible.

Para resolver esta prueba, se desarrollara la construcción de un móvil autónomo capaz de detectar y esquivar las paredes de un laberinto, para salir de dicho laberinto, todo ello por sus propios medios, es decir mediante el algoritmo y sensores que tenga instalados y sin ningún tipo de ayuda externa.

Arquitectura del robot

El robot a desarrollar que permitirá superar la prueba del Laberinto constara de la siguiente arquitectura:

Sistema locomotor

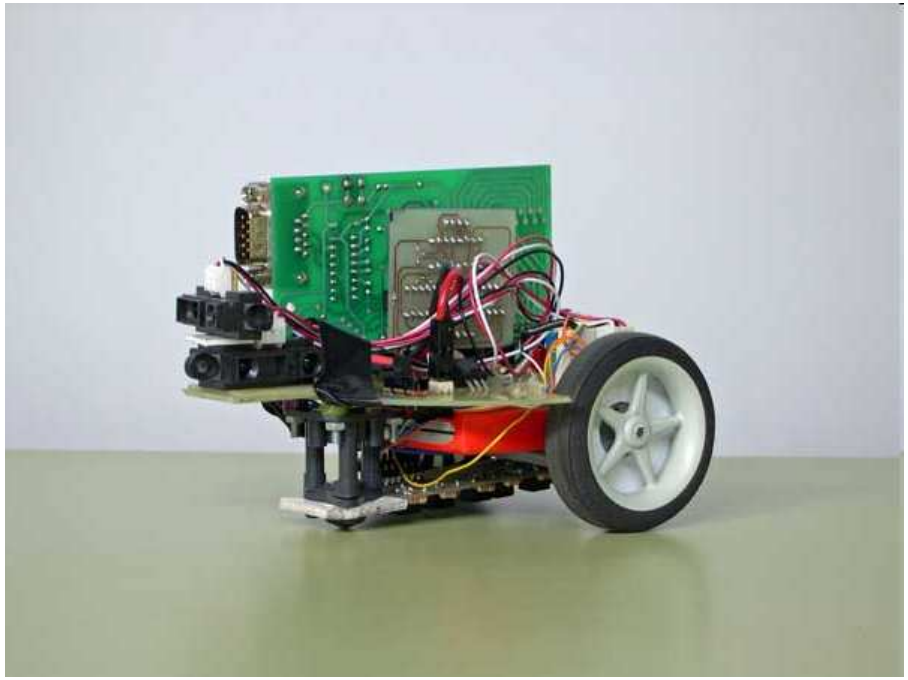
El sistema locomotor del robot constara de 2 motores, los cuales emplean protocolo **PWM** para su control.

Sistema sensitivo

El sistema sensitivo del robot esta compuesto por 3 sensores que miden la distancia

Plataforma de control

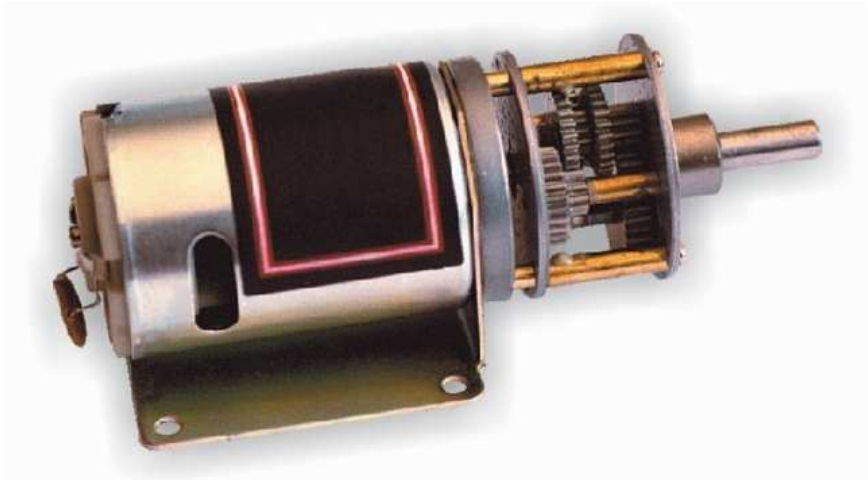
La plataforma de control es desarrollada por IES Antonio Machado.

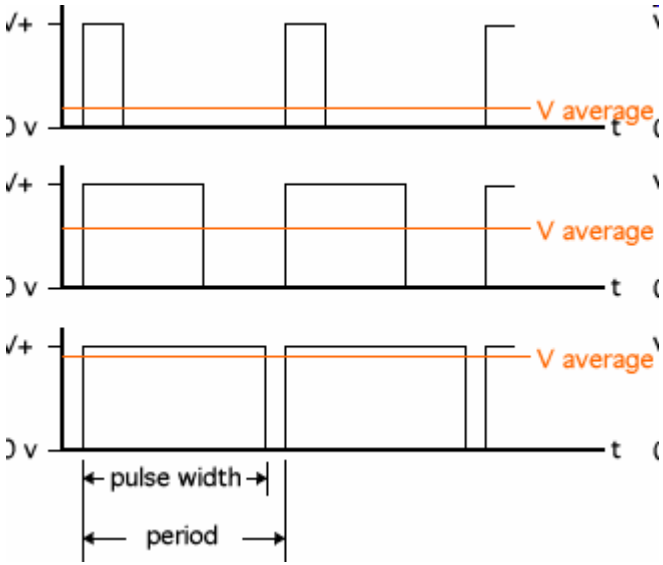


Arquitectura del robot

Sistema locomotor

El sistema locomotor del robot constara de 2 motores, los cuales emplean protocolo **PWM** para su control.





Arquitectura del robot

Sistema locomotor: PWM

La modulación por ancho de pulsos (o PWM, de pulse-width modulation en inglés) de una señal o fuente de energía es una técnica en la que se modifica el ciclo de trabajo de una señal periódica (por ejemplo sinusoidal o cuadrada) ya sea para transmitir información a través de un canal de comunicaciones o control de la cantidad de energía que se envía a una carga.

La aplicación de las técnicas PWM sobre los motores, permitirá desarrollar el sistema de navegación del robot.

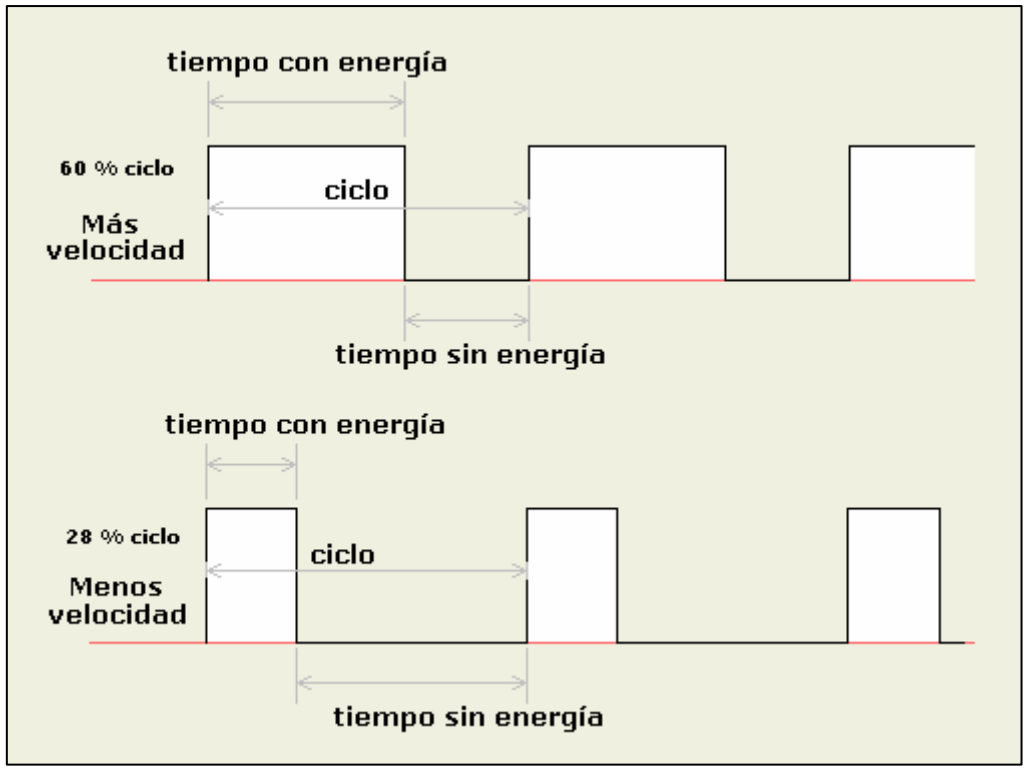
Arquitectura del robot

Sistema locomotor: PWM

La Regulación por Ancho de Pulso de un motor de CC está basada en el hecho de que si se recorta la CC de alimentación en forma de una onda cuadrada, la energía que recibe el motor disminuirá de manera proporcional a la relación entre la parte alta (habilita corriente) y baja (cero corriente) del ciclo de la onda cuadrada. Controlando esta relación se logra variar la velocidad del motor de una manera bastante aceptable.

Arquitectura del robot

Sistema locomotor: PWM



Arquitectura del robot

Sistema locomotor: Objetivos

Los objetivos del sistema locomotor son los siguientes:

1. Desarrollo de API para realizar las siguientes operaciones:
    1. Marcha hacia delante
    2. Marcha hacia atrás
    3. Giro a la izquierda de 90º
    4. Giro a la izquierda de X Grados
    5. Giro a la derecha de 90º
    6. Giro a la derecha de X Grados
    7. Stop
    8. Set/Get de velocidad

Arquitectura del robot

Sistema sensitivo

El sistema sensitivo del robot esta compuesto por 3 sensores que miden la distancia.

Se puede emplear ultrasonidos o infrarrojos para desempeñar dicha tarea.

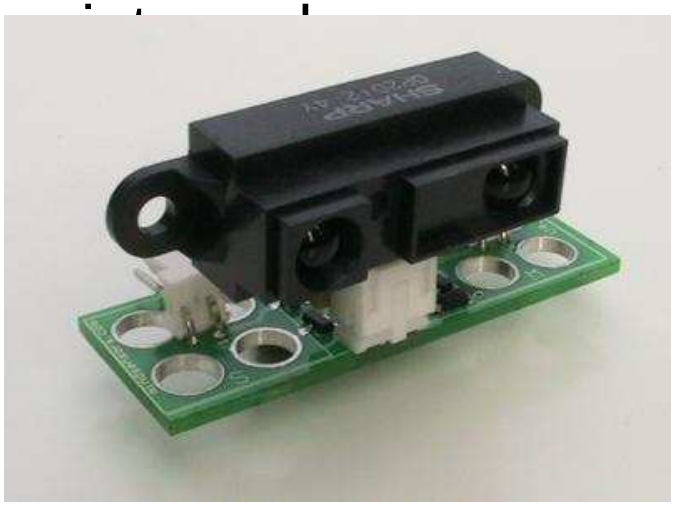
Sensor infrarrojo

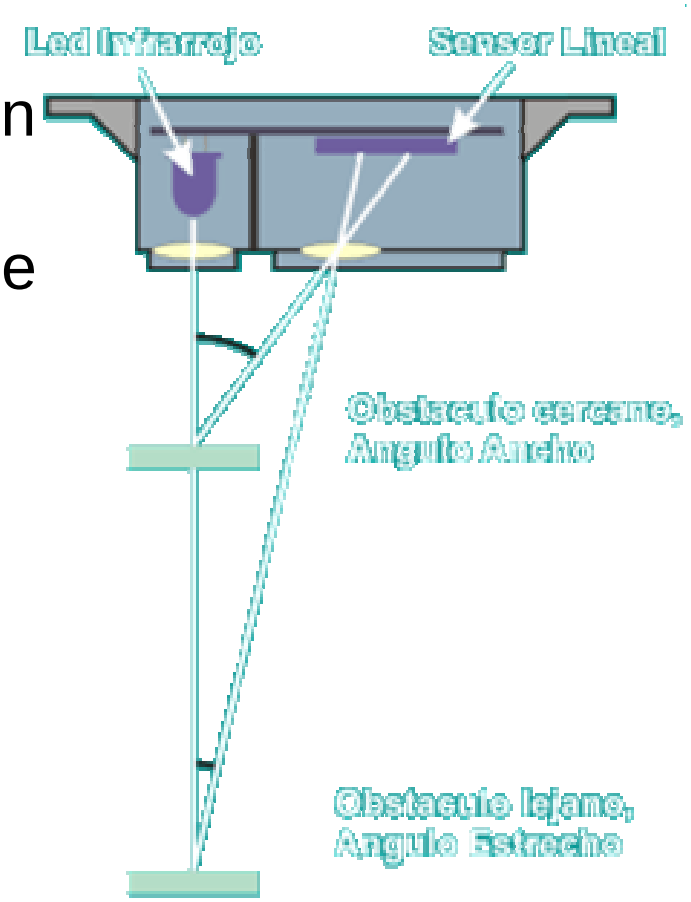
Sensor ultrasonidos

Arquitectura del robot

Sistema sensitivo : Sensores de Infrarrojos

La familia de sensores GP2DXX de Sharp son dispositivos de reflexión por infrarrojos con medidor de distancia proporcional al ángulo de recepción del haz de luz que incide en un sensor lineal





Arquitectura del robot

Sistema sensitivo : Objetivos

Los objetivos del sistema sensitivo son los siguientes:

1. Desarrollo de API para realizar las siguientes operaciones:
    1. Establecer rangos de confianza y valores comparativos
    2. Medir distancias en los 3 sensores
    3. Detectar si el robot esta en un limite y tiene que corregir rumbo

Arquitectura del robot

Plataforma de control

La plataforma de control del robot se basara en 2 elementos: Placa controladora con microcontrolador (PIC16F876) y una plataforma de desarrollo

Placa controladora

La placa electrónica será desarrollada por el IES Antonio Machado

Plataforma de desarrollo

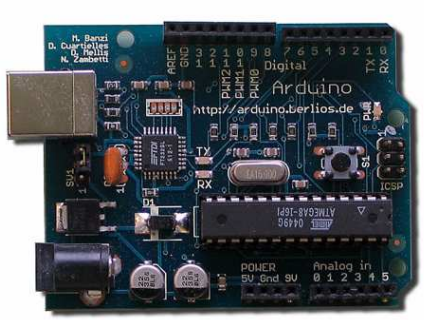
Para la modelización del robot y su posterior codificación en código para el microcontrolador PIC16F876, se empleara **Proteus**.

Arquitectura del robot

Alternativas

La construcción del desarrollo también podría ser posible a través de las siguientes opciones:

1. Lego Mindstorms NXT
2. Arduino
3. Sun spot
4. VEX



Madrid-Bot

Madrid-Bot es un concurso de micro robótica organizado por los Centro que imparten las enseñanzas del Ciclo Formativo de Grado Superior de Desarrollo de Productos Electrónicos en la Comunidad Autónoma de Madrid.

Madrid-Bot intenta ser un concurso de micro-robótica y más aun un lugar de encuentro donde los alumnos matriculados en Centros de Educación Secundaria y especialmente los alumnos y alumnas de Bachillerato, Ciclos Formativos de la familia profesional de electrónica, informática, etc.. puedan no sólo competir con sus prototipos sino también compartir sus conocimientos.

Madridbot

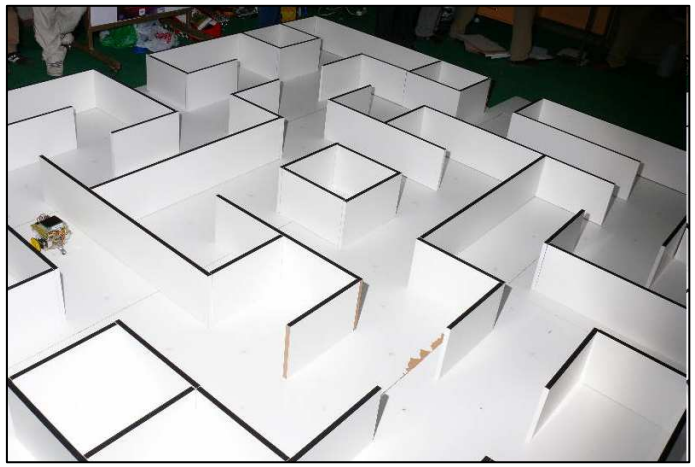
Pruebas del concurso

Las pruebas que se realizan en Madridbot, son las siguientes:

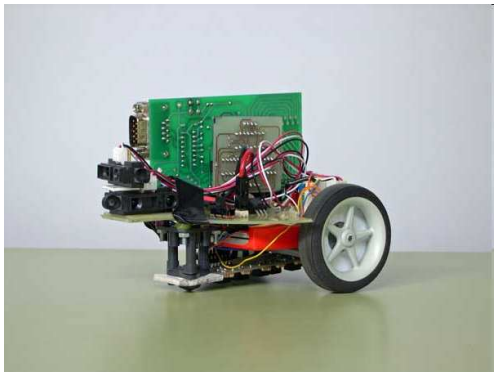
1. Rastreadores
2. Velocistas
3. Laberinto
4. Mini sumo
5. Prueba Libre

La prueba del laberinto

La prueba consiste en la navegación autónoma a través de un laberinto, siendo el objetivo de la prueba salir del laberinto en el menor tiempo posible. El plano del laberinto será conocido a priori y podrá ser utilizado en los algoritmos de guiado del robot. El robot inicia su recorrido desde fuera del laberinto y podrá salir por la salidas existente en la cara opuesta



Resolver un laberinto conocido a priori



Maquinas de estados finitos

Las Máquinas de Estados Finitos (FSM), también conocidas como Autómatas de Estados Finitos (FSA), explicado de forma simple, son modelos de comportamiento de un sistema o un objeto complejo, con un número limitado de modos o condiciones predefinidos, donde existen transiciones de modo. Las FSMs están compuestas por 4 elementos principales:

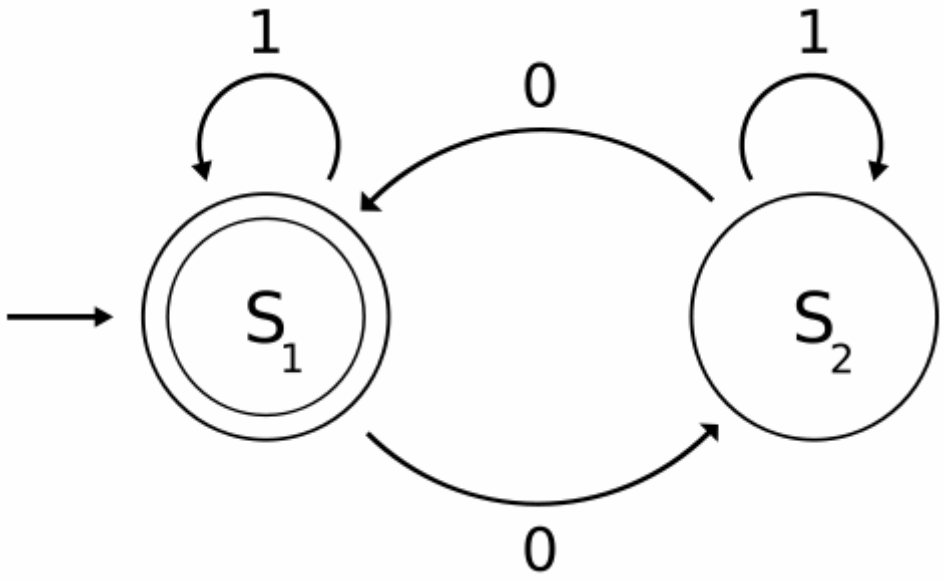
1. Estados que definen el comportamiento
2. Transiciones de estado
3. Reglas o condiciones que deben cumplirse para permitir un cambio de estado
4. Eventos de entrada que son externos o generados internamente

Maquinas de estados finitos

Una máquina de estados finitos debe tener un estado inicial que actúa de punto de comienzo, y un estado actual que recuerda el producto de la anterior transición de estado. Los eventos recibidos como entrada actúan como disparadores, que causan una evaluación de las reglas que gobiernan las transiciones del estado actual a otro estado. La mejor manera de visualizar una FSM es pensar en ella como un diagrama de flujo o un grafo dirigido de estado, aunque como se verá existen técnicas de abstracción más precisas que pueden ser usadas.

Maquinas de estados finitos

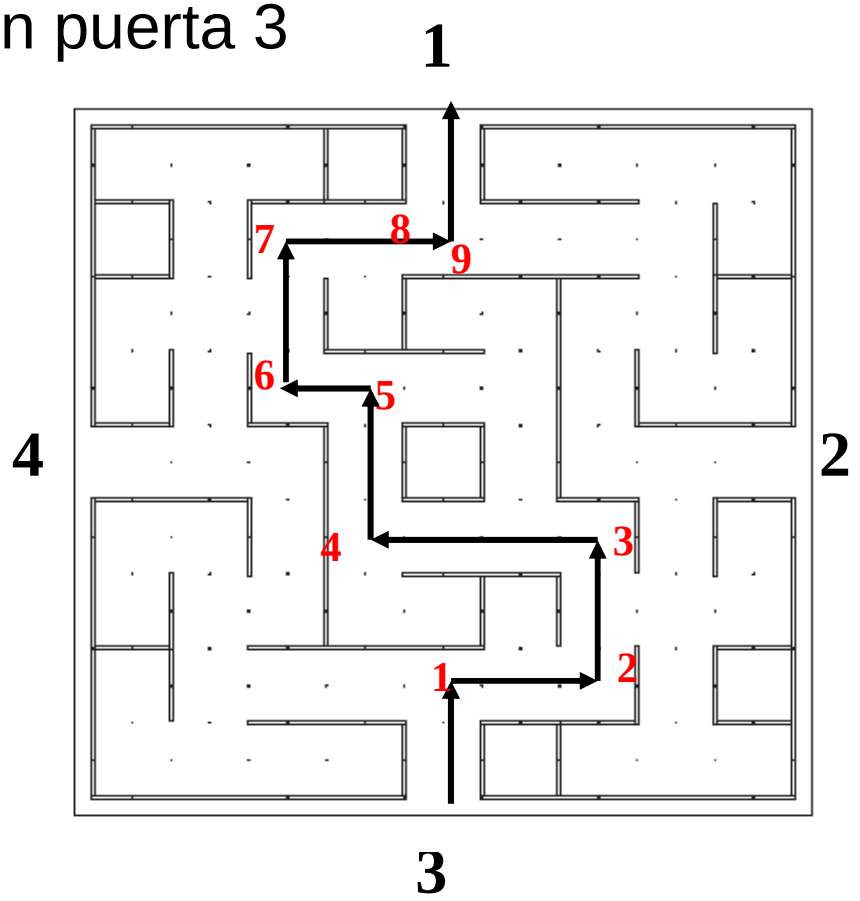
La resolución de ciertos problemas, puede ser modelizado a través de teoría de maquinas de estado finitos. En el caso de la prueba del laberinto, este enfoque puede ser aceptado.



Una estrategia de solución

En este punto se abordara la solución del problema del laberinto, en el caso de explicara una posible solución al caso de inicio de prueba en puerta 3

El robot para superar la prueba, necesita gestionar 8 giros.



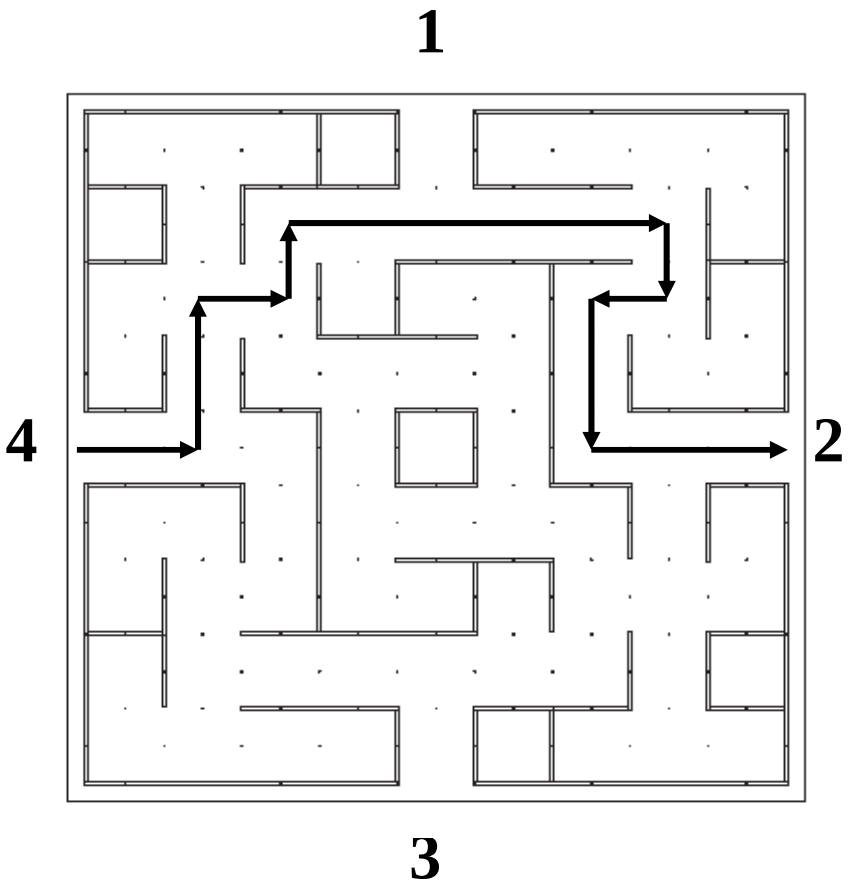
La prueba del laberinto

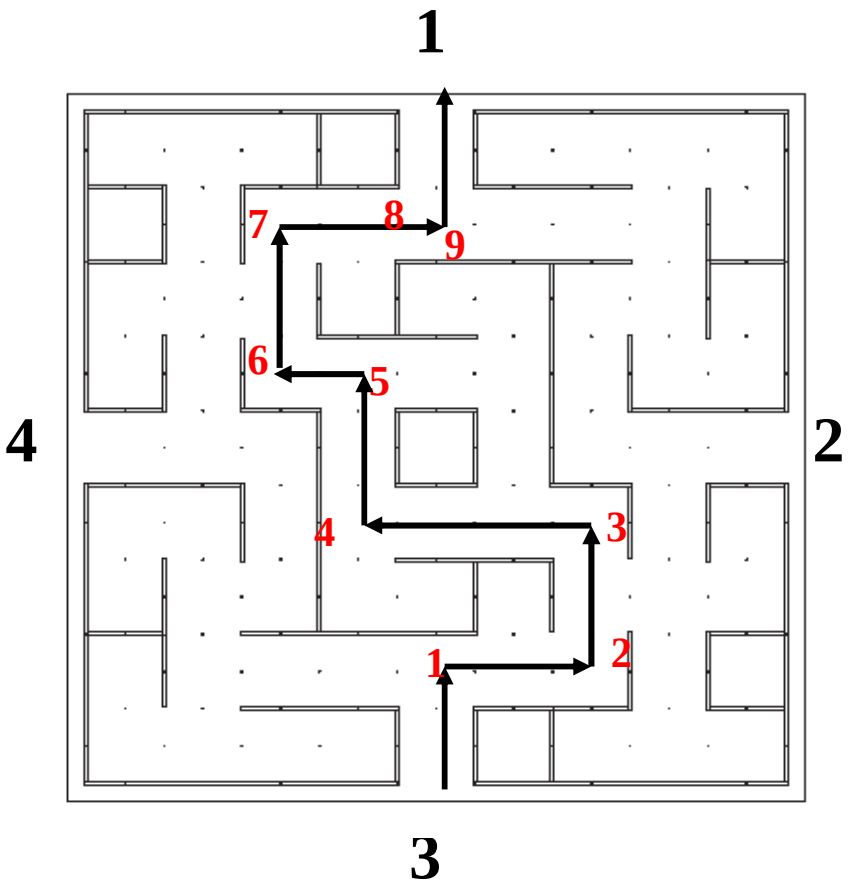
Las estrategias para salir del laberinto pueden ser múltiples, tal vez la mas sencilla es seguir una de las paredes laterales del laberinto, ya sea por la izquierda o por la derecha, ahora bien esta estrategia tiene un inconveniente y es que el comino recorrido es mas largo y hay que tener en cuenta que no solo se trata de salir del laberinto, si no que además se prima el hacerlo lo más rápido posible.

Parece que lo conveniente es memorizar el camino más corto y efectuarlo.

La prueba del laberinto

La solución para todos los casos, muestran un camino simétrico entre la solución 4-2 y 3-1



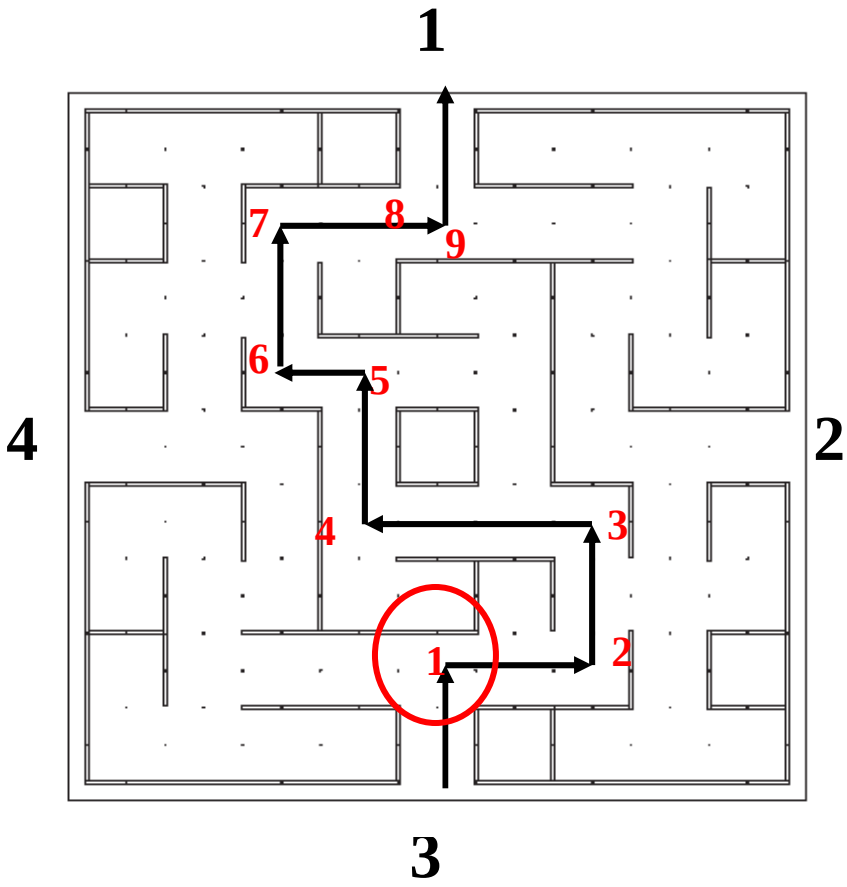


La prueba del laberinto

Estado 1:

Según salimos de la puerta 3, comenzamos a avanzar recto, y hacemos lecturas del sensor frontal, hasta que este detecte una pared al frente, hemos alcanzado el punto 1.

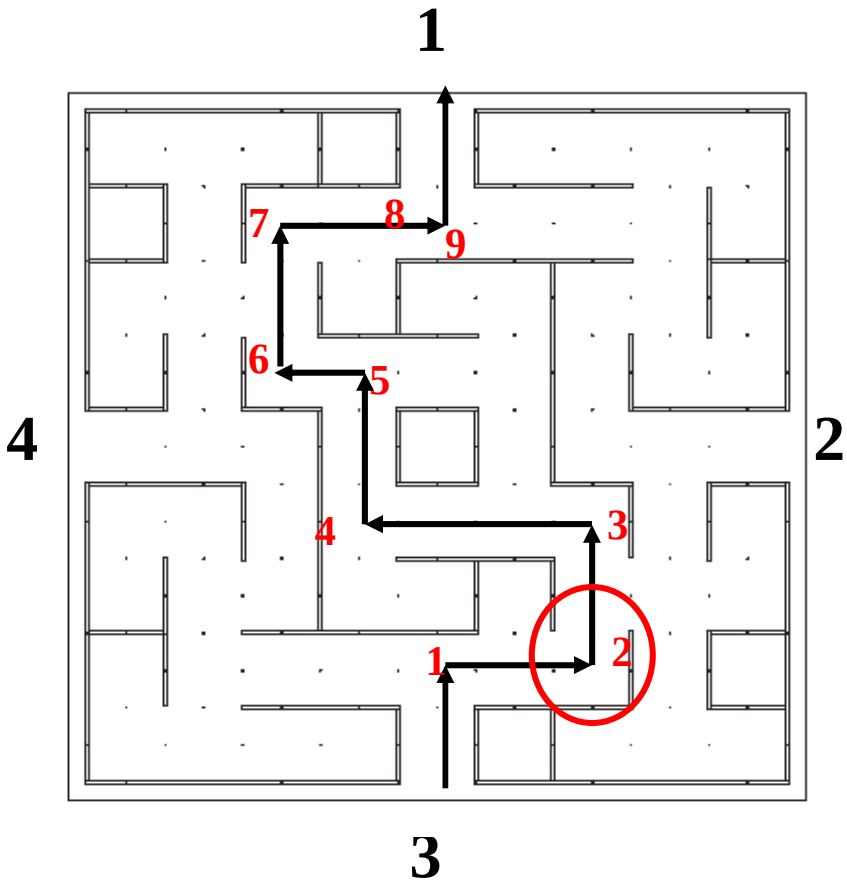
Una vez situados en el punto 1 giramos a la derecha 90º y seguimos avanzando recto y haciendo lecturas con el sensor frontal hasta detectar pared, momento en el que alcanzaremos el punto 2.



La prueba del laberinto

Estado 2:

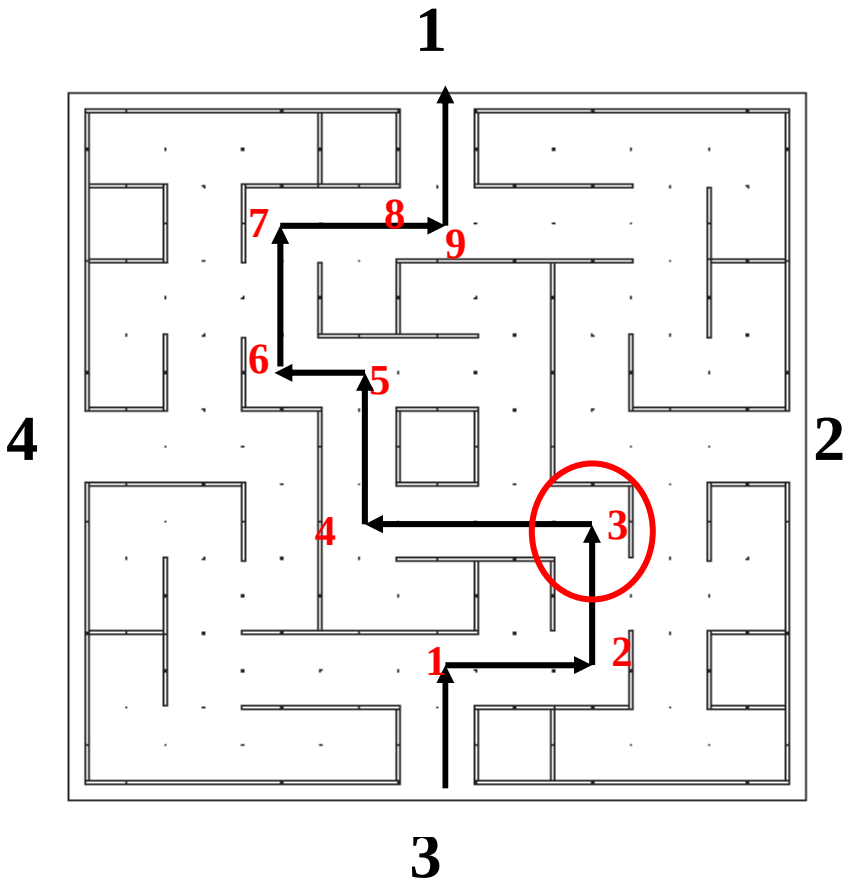
Una vez situados en el punto 2 giramos a la izquierda 90º y seguimos avanzando recto y haciendo lecturas con el sensor frontal hasta detectar pared, momento en el que alcanzaremos el punto 3 y giramos 90º a la Izquierda



La prueba del laberinto

Estado 3:

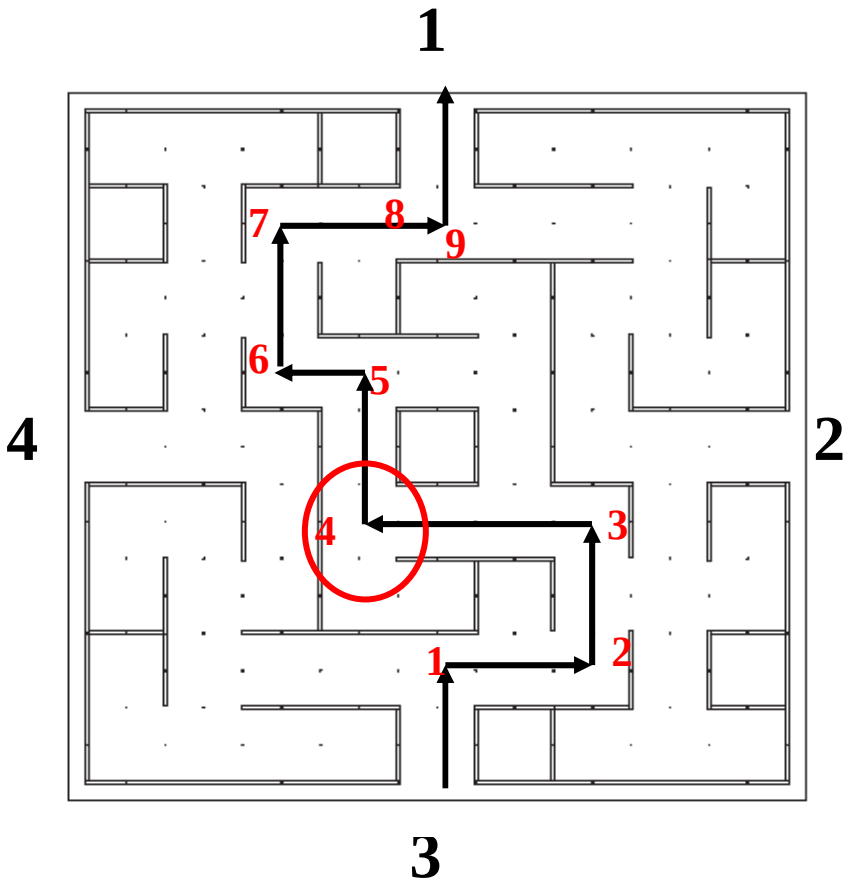
Una vez situados en el punto 3 giramos a la izquierda 90º y seguimos avanzando recto y haciendo lecturas con el sensor frontal hasta que detectemos la pared, momento en el que alcanzaremos el punto 4.



La prueba del laberinto

Estado 4:

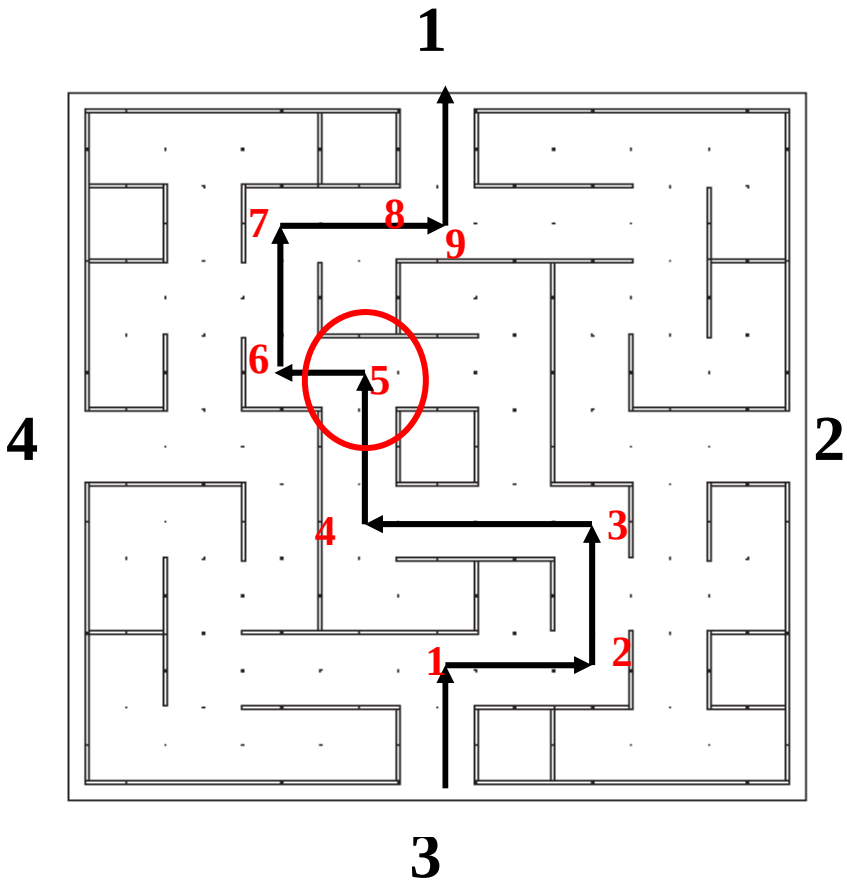
Una vez situados en el punto 3 giramos a la izquierda 90º y seguimos avanzando recto y haciendo lecturas con el sensor frontal hasta que detectemos la pared, momento en el que alcanzaremos el punto 4.



La prueba del laberinto

Estado 5:

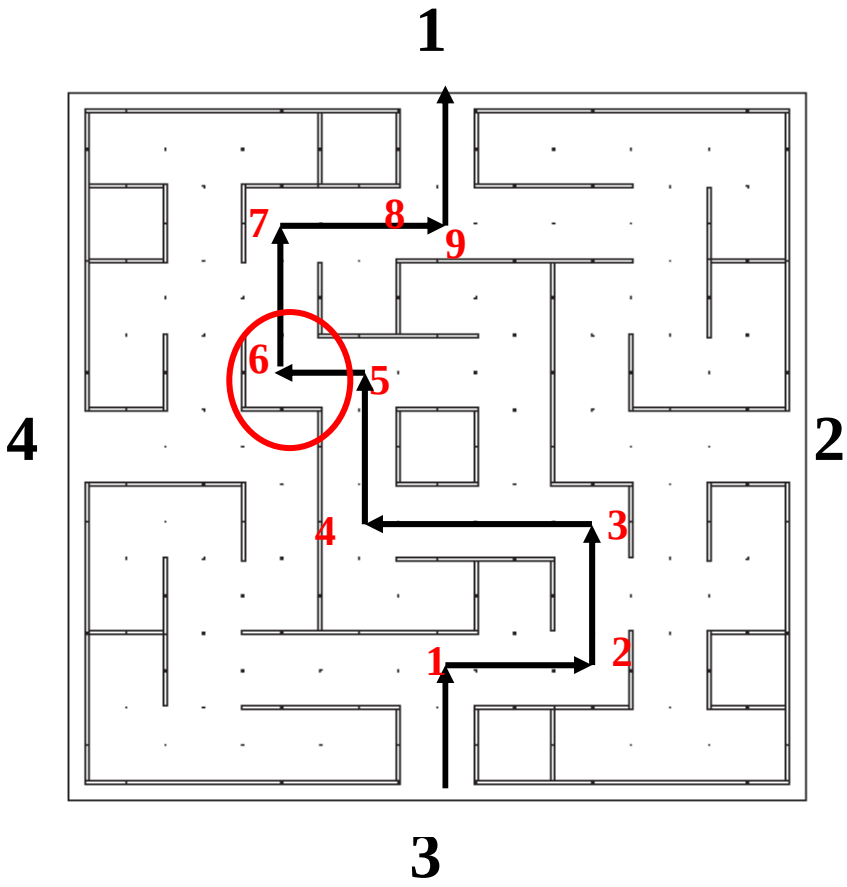
Una vez situados en el punto 4 giramos a la derecha 90º y seguimos avanzando recto y haciendo lecturas con el sensor frontal hasta que detectemos la pared, momento en el que alcanzaremos el punto 5.



La prueba del laberinto

Estado 6:

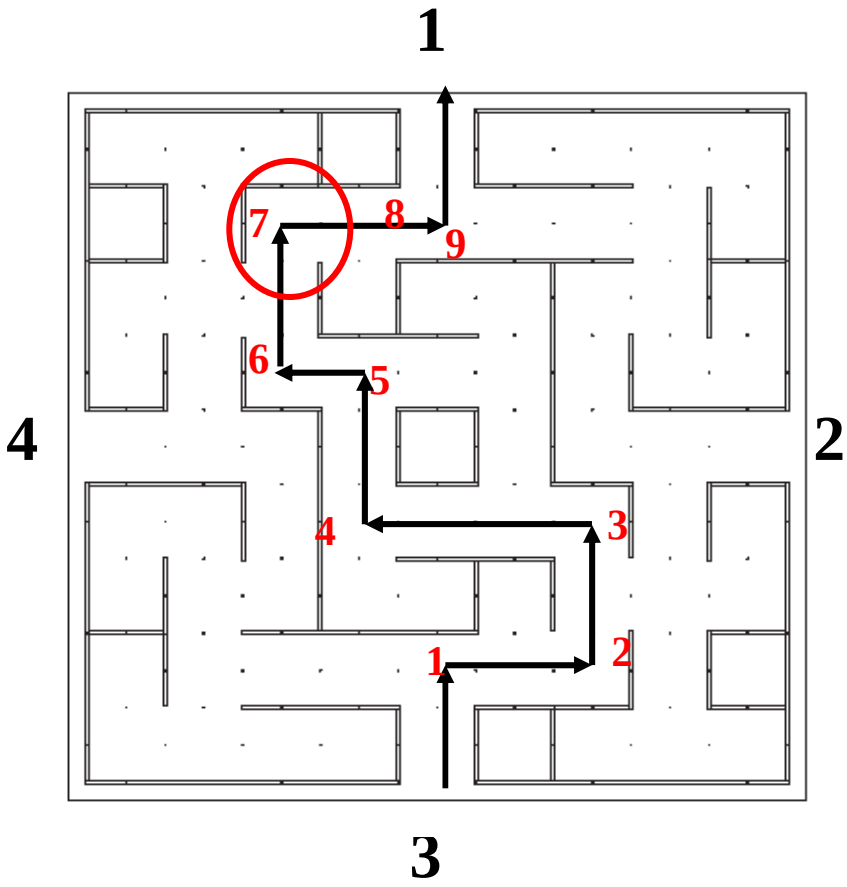
Una vez situados en el punto 5 giramos a la izquierda 90º, seguimos avanzando recto y haciendo lecturas con el sensor frontal hasta que detectemos la pared, momento en el que alcanzaremos el punto 6.



La prueba del laberinto

Estado 7:

Una vez situados en el punto 6 giramos a la derecha 90º, seguimos avanzando recto y haciendo lecturas con el sensor frontal hasta que detectemos la pared, momento en el que alcanzaremos el punto 7.

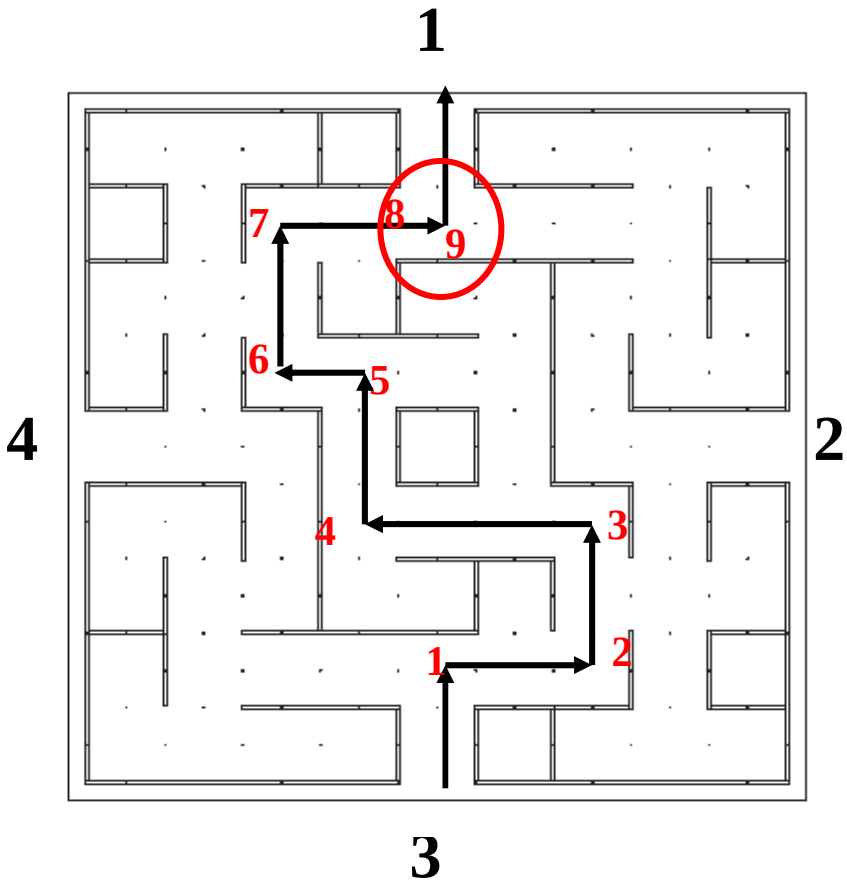


La prueba del laberinto

Estado 8:

Una vez situados en el punto 7 giramos a la derecha 90,º avanzando recto, haciendo lecturas con el sensor lateral hasta que dejemos de detectar pared, momento en el que alcanzaremos el punto 8.

Una vez situados en el punto 8 avanzamos un tiempo mínimo para situarnos en el centro de la calle (punto 9); posteriormente giramos a la izquierda 90º, saliendo del laberinto.



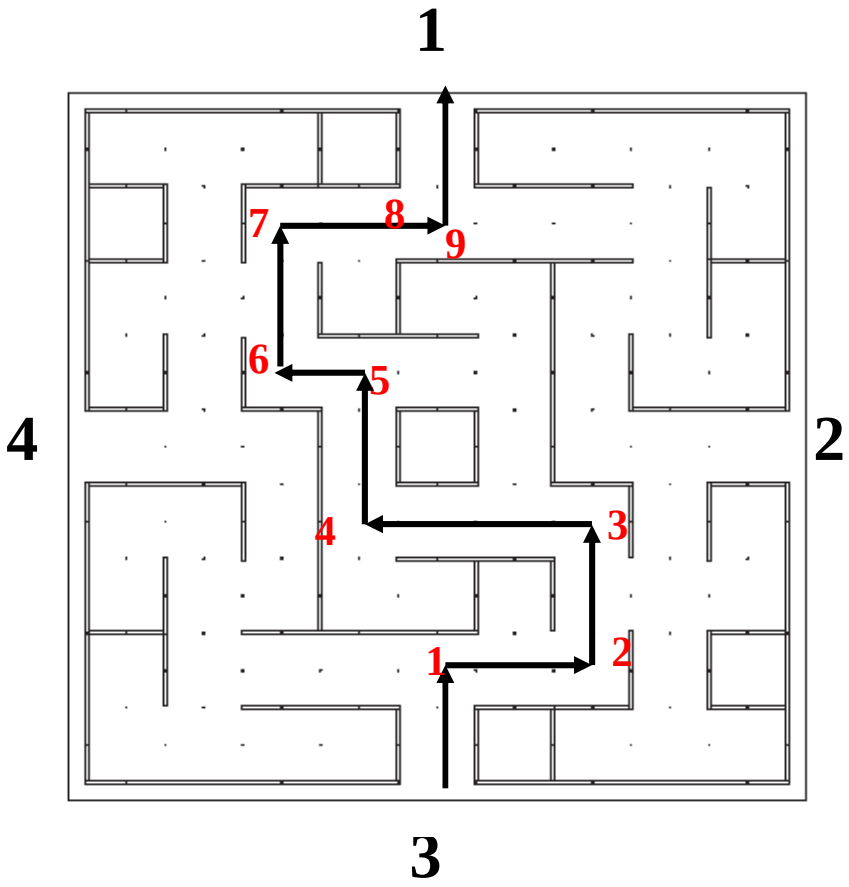
La prueba del laberinto

Conclusiones

En cada giro, es necesario:

1. Ir hacia delante hasta alcanzar una distancia X
2. Mantener distancias laterales con un rango Y-Z
3. Capacidad de giro de 90º
4. Detectar pasillos laterales
5. Ir hacia delante durante un tiempo X

<u>Internamente debe existir la capacidad de corregir distancias laterales</u>



Pseudo código

Una vez hemos realizado el análisis es necesario diseñar un seudo código que permita conseguir la meta, resolver el laberinto en el menor tiempo posible.

Para ello es necesario definir una serie variables que modelizaran el sistema de control.

Por otro lado, es necesario definir una serie de funciones que definirá la actuación del robot con respecto al entorno:

Pseudo codigo

Parámetros:

Parámetros del robot:

ancho: 10
largo: 10
velocidad: 2m/seg

Parámetros del entorno:

distanciaSeguridad: 10
anchoPasillo: 40
distanciaMinimaLateral: 10

distanciaLateralIzquierdaN-1
distanciaLateralIzquierda
distanciaFrontal
distanciaLateralDerechaN-1
distanciaLateralDerecha
tiempoHastaCentro

Pseudo codigo

Funciones:

seleccionarCaso(caso)
solucionarCaso1()
solucionarCaso2()
solucionarCaso3()
solucionarCaso4()
giro1()
giro2()
giro3()
giro4()
giro5()
giro6()
giro7()
giro8()

Funciones:

irHastaDistancia()
irDuranteTiempo(ms)
giro90Izquierda()
giro90Derecha()
parar()
Ir()
IrConCorreccionIzquierda()
IrConCorreccionDerecha()
leerDistanciaFrontal()
leerDistanciaLateralIzquierda()
leerDistanciaLateralDerecha()

Pseudo codigo

Seudocodigo Programa Principal:

```
laberinto(){
      caso = seleccionCaso()
      si(caso  =1){
            solucionarCaso1()
      }sino si(caso = 2){
            solucionarCaso2()
      }sino si(caso = 3){
            solucionarCaso3()
      }sino{
            solucionarCaso4()
      }
}
```

Pseudo codigo

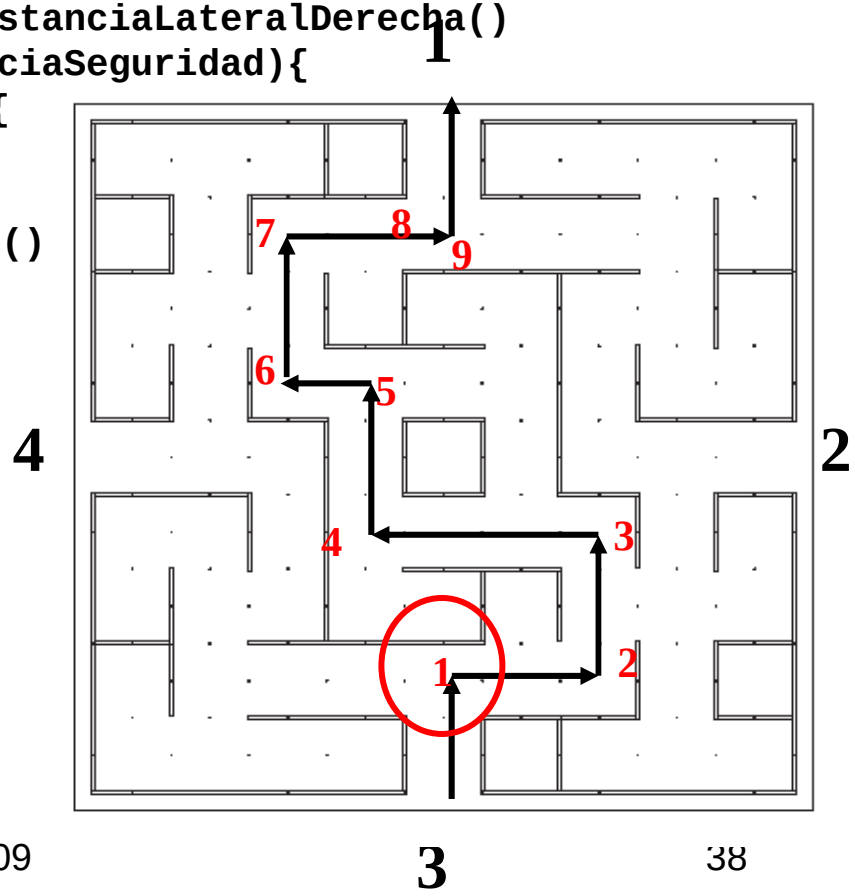
Seudocodigo Caso 3:

```
solucionarCaso1(){
      giro1()
      giro2()
      giro3()
      giro4()
      giro5()
      giro6()
      giro7()
      giro8()
}
```

Pseudo codigo

Seudocodigo Caso 3:

```
giro1(){
        distanciaFrontal = leerDistanciaFrontal()
        distanciaLateralIzquierda = leerDistanciaLateralIzquierda()
        distanciaLateralIzquierda = leerDistanciaLateralDerecha()
        mientras(distanciaFrontal <=distanciaSeguridad){
                si(distanciasLateralOk()){
                        ir()
                }sino{
                        corregirDireccion()
                }
        }
        parar()
        giro90Derecha()
}
```



Recursos

Documentación adicional

Eventos en España:
http://www.madridbot.org/index.htm
http://complubot.educa.madrid.org/inicio.php?seccion=principal
http://www.eis.uva.es/amuva/robolid/

Odometría:
http://www.itoosoft.com/motorolos/odometria/odometria.html
http://www-personal.umich.edu/~johannb/shared/pos96rep.pdf
http://www.seattlerobotics.org/encoder/200108/using_a_pid.html
http://rossum.sourceforge.net/papers/DiffSteer/DiffSteer.html

Steering Behaviors:
http://www.red3d.com/cwr/steer/gdc99/

Foros:
http://lrobotikas.net/