Contenido

[INTRODUCCION 1](#_Toc28612205)

[PARTE 1: Diseño manual de un control borroso de tipo MAMDANI. 1](#_Toc28612206)

[PARTE 1.1: Circuito Sin Obstáculos 1](#_Toc28612207)

[PARTE 1.2: Circuito con Obstáculos 1 1](#_Toc28612208)

[PARTE 1.3: Circuito con Obstáculos 2 1](#_Toc28612209)

[PARTE 2: Diseño automático de un controlador neuroborroso de tipo SUGENO. 1](#_Toc28612210)

[PARTE 2.1: Circuito Sin Obstáculos 1](#_Toc28612211)

[PARTE 2.2: Circuito con Obstáculos 1 1](#_Toc28612212)

[PARTE 2.3: Circuito con Obstáculos 2 1](#_Toc28612213)

# INTRODUCCION

El objetivo de la práctica es el diseño del control de velocidad (lineal y angular) de un robot móvil para que éste recorra un circuito cerrado (delimitado por paredes), en el que pueden aparecer obstáculos estáticos, en el menor tiempo posible.

El robot cuenta con una serie de ***sensores de ultrasonidos*** distribuidos alrededor de este que son capaces de detectar objetos a una distancia en un rango [0.1- 5]m.

Imagen que contiene dispositivo

Descripción generada automáticamente

**Figura 1:** Distribución de los sensores

El comportamiento del robot y su interacción con el entorno se simula en la ***plataforma ROS-STDR***. Robot Operating System o ***ROS***, es una plataforma de desarrollo robótico que permite crear aplicaciones con múltiples sensores y actuadores de forma flexible. Es una colección de herramientas, bibliotecas y convenciones que tiene como objetivo simplificar la tarea de crear un comportamiento de robot complejo y robusto en una amplia variedad de plataformas robóticas.

A través de una Máquina Virtual con el sistema ROS Kinetic ya preinstalado podremos controlar nuestro robot desde Matlab.

Lo primero que hemos hecho ha sido configurar el entorno de trabajo necesario para manejar todo lo relativo al robot en el directorio ***robótica\_movil\_ws****.*

Además debemos añadir el espacio de trabajo al path por defecto añadiendo estas 2 lineas al final del archivo ***.bashrc***:

source ~/robotica\_movil\_ws/devel/setup.bash

export ROS\_PACKAGE\_PATH=$ROS\_PACKAGE\_PATH:~/robotica\_movil\_ws/

Y también añadiremos las IPs necesarias, extraídas con el comando ***ifconfig***, al final del mismo archivo, en nuestro caso serán las siguientes:

export ROS\_MASTER\_URI=http://192.168.1.48:11311

export ROS\_IP=:192.168.1.48

Una vez realizado todo este proceso de configuración ya podremos iniciar el simulador con el siguiente comando:

roslaunch stdr\_launchers PracticaFinal.launch

A lo largo de la práctica necesitaremos cambiar el mapa que recorrerá nuestro robot entre 3 distintos que se incluyen entre los archivos del Miniproyecto: ***EntornoSinObstaculos.png, EntornoConObstaculos.png y EntornoConObstaculos2.png***. Para ello simplemente basta con modificar el archivo de descripción del mapa ***EntornoPracticaFinal.yaml*** y especificar que archivo .png deseamos utilizar al iniciar el simulador.

Imagen que contiene captura de pantalla

Descripción generada automáticamente

**Figura 2:** Archivo de descripción del mapa EntornoPracticaFinal.yaml

# PARTE 1: Diseño manual de un control borroso de tipo MAMDANI.

En esta primera parte de la práctica se nos pide que nuestro ***amigobot*** recorra 3 circuitos utilizando un Controlador Borroso de ***tipo Mamdani***.

Utilizaremos la función ***fuzzy*** para generar un ***controlador.fis*** que añadiremos a nuestro bloque de Controlador en Simulink y conectaremos con el ***bloque ROS Robot*** a través de bloques multiplexores y demultiplexores. En la Figura 3 podemos ver como quedaría el archivo Simulink sin ningún sensor conectado de los 8 disponibles, a lo largo de la práctica conectaremos aquellos que consideremos necesarios para una trayectoria exitosa:

INSERTAR IMAGEN DEL SIMULINK SIN CONECTAR LOS SENSORES

**Figura 3:** Esquema de Simulink

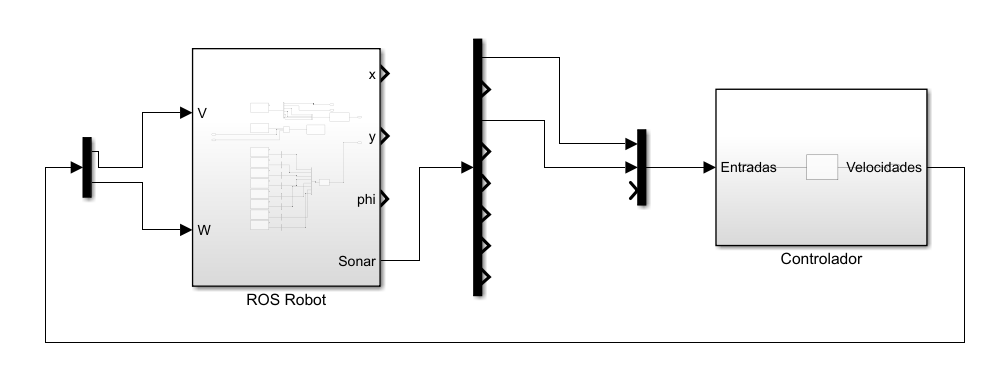
Por último, para terminar de configurar todo antes de comenzar las simulaciones debemos conectar los bloques Suscriptores del bloque ROS Robot a nuestra máquina virtual configurando la dirección red:

INSERTAR CAPTURA MATLAB CONFIGURE NETWORK ADRESSES

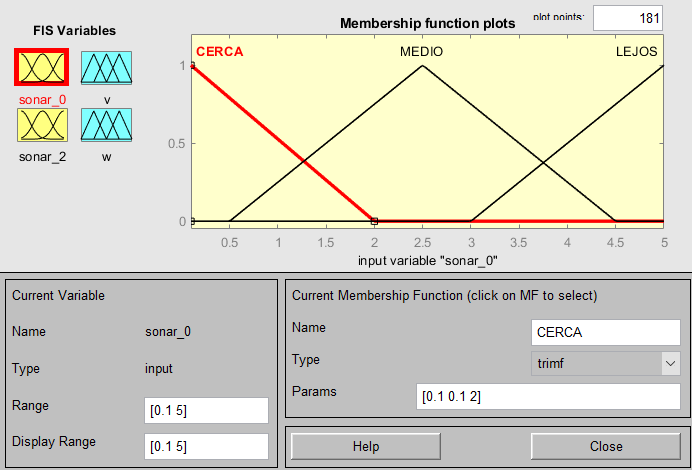
**Figura 4:** Configuración de red de bloques suscriptores

## PARTE 1.1: Circuito Sin Obstáculos

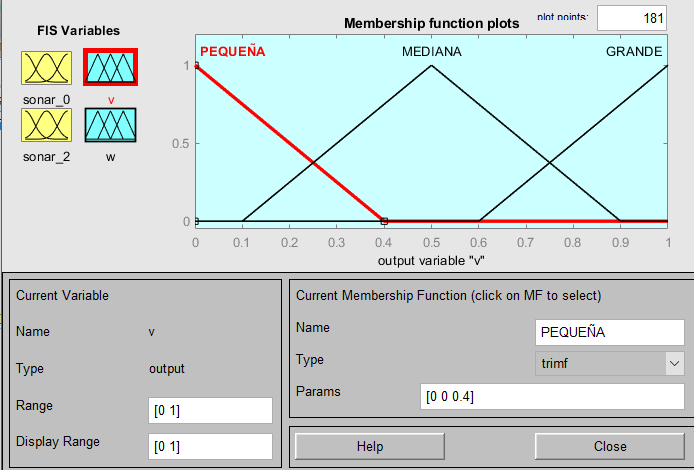
Lo primero que tendremos que hacer es realizar un controlador borroso de tipo Mamdani que ayude a nuestro robot a superar el circuito con el mapa sin obstáculos.

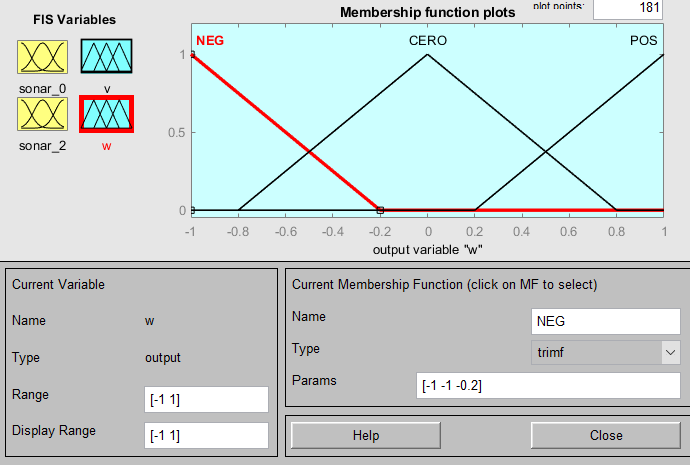
Para ello, lo primero que haremos será determinar cuántos sensores vamos a utilizar para este caso. Nuestra decisión fue utilizar dos sensores (sonar\_0 y sonar\_2), uno para controlar la pared izquierda y otro para controlar la pared derecha. La arquitectura sería la siguiente:

**Figura 5:** Arquitectura del sistema circuito sin obstáculos

Una vez tuvimos clara la arquitectura, nos pusimos con el controlador borroso. Teniendo en cuenta las medidas del circuito establecimos los mismos límites para el sonar\_0 y el sonar\_2:

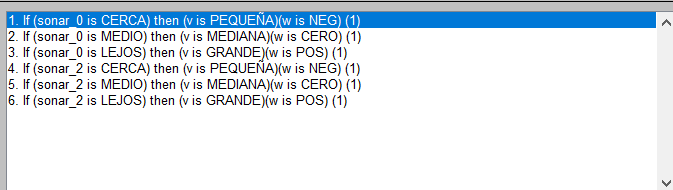
**Figura 6:** Límites y funciones de pertenencia controlador sin obstáculos

Por otro lado, para las velocidades lineal y angular:



**Figura 7:** Límites y funciones de pertenencia w y v

Y, por último, las reglas que modelarán el comportamiento del robot serán las siguientes:

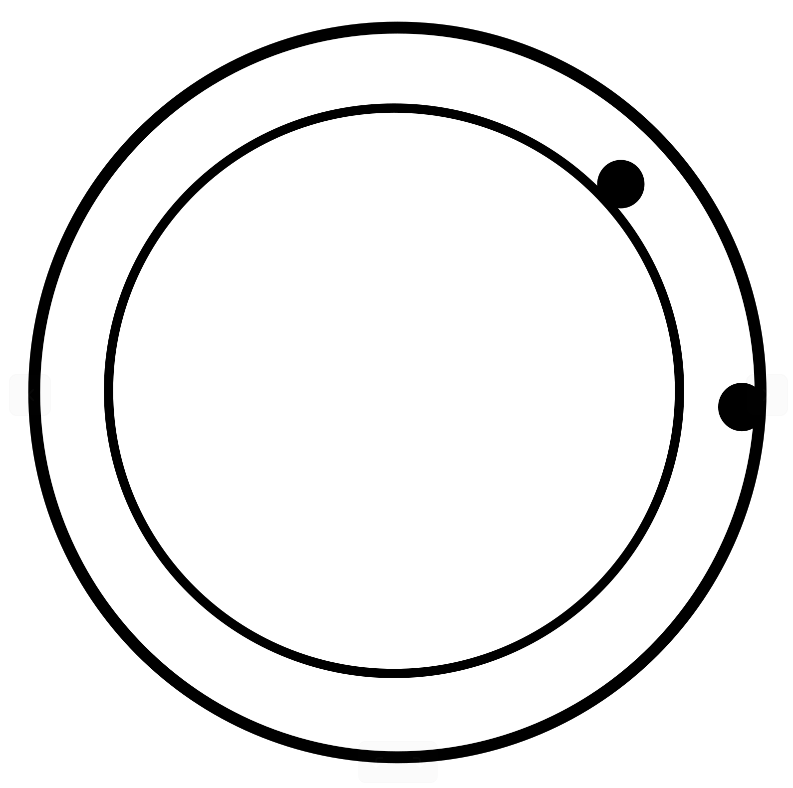


**Figura 8:** Reglas controlador sin obstáculos

Una vez tuvimos todo conectado y configurado correctamente, procedimos a probar nuestro controlador con ROS. Los resultados obtenidos fueron satisfactorios, ya que nuestro robot daba una vuelta completa al circuito sin chocarse con nada. Estos resultados pueden observarse en el archivo de vídeo adjuntado en el entregable del proyecto.

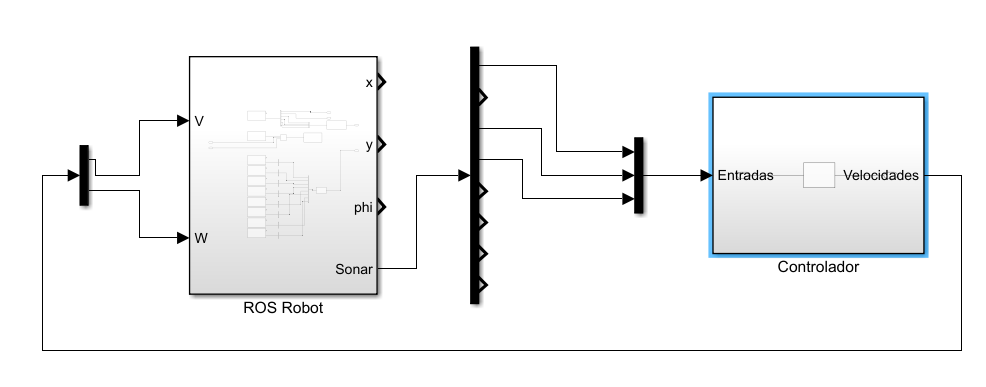
## PARTE 1.2: Circuito con Obstáculos 1

El siguiente circuito que tendrá que superar nuestro controlador es el “CircuitoConObstaculos1”, que podemos encontrar en la máquina virtual y tiene el siguiente aspecto:

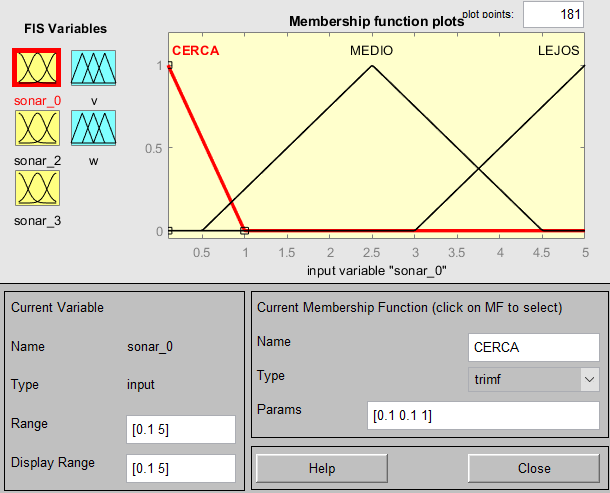
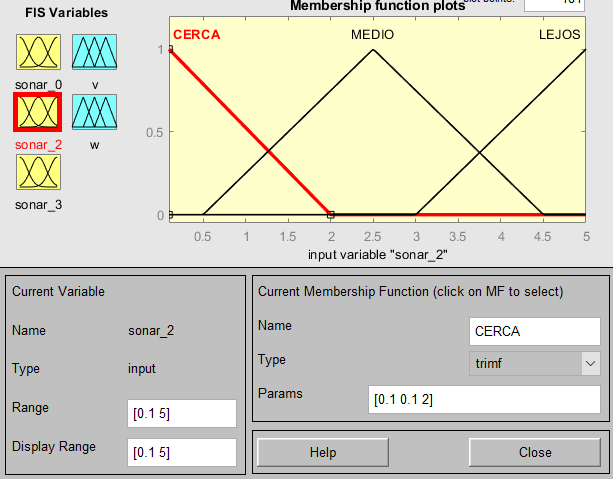


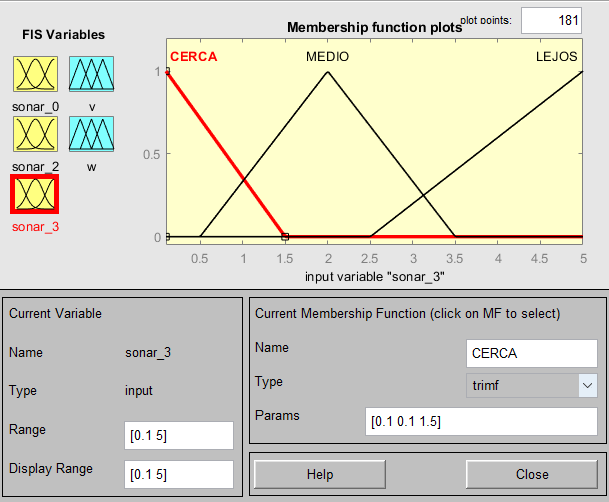
**Figura 9:** Circuito con obstáculos 1

Como se puede observar, ahora nuestro robot tendrá que sortear dos obstáculos para dar una vuelta completa al circuito, por lo que deberemos gestionarlo en el controlador. La arquitectura cambiará, ahora utilizaremos 3 sensores para controlar que el robot no se choca con ningún obstáculo (sonar\_0, sonar\_2 y sonar\_3). Así, la nueva arquitectura queda de la siguiente manera:



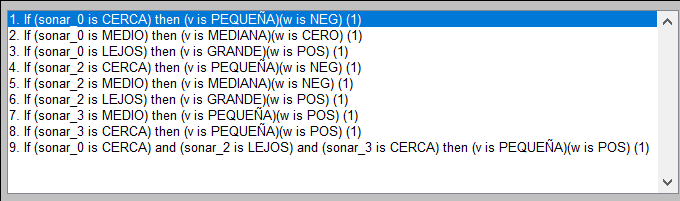
**Figura 10:** Arquitectura del sistema para el circuito con obstáculos 1

El siguiente paso será modelar las funciones de pertenencia y límites de cada sonar, así como de la velocidad lineal y angular. Al introducir un nuevo sonar en el modelo, los límites de los sónares cambian, pero los de w y v quedan igual que en el modelo anterior. Así, los nuevos límites de los sónares serán:



**Figura 11:** Funciones de pertenencia de los sónares para el circuito con obstáculos 1

Y, por último, las reglas que modelarán el comportamiento del robot serán las siguientes:



**Figura 12:** Reglas para el circuito con obstáculos 1

Una vez tuvimos todo conectado y configurado correctamente, procedimos a probar nuestro controlador con ROS. Los resultados obtenidos fueron satisfactorios, ya que nuestro robot daba una vuelta completa al circuito sin chocarse con nada y evitando los obstáculos. Estos resultados pueden observarse en el archivo de vídeo adjuntado en el entregable del proyecto.

## PARTE 1.3: Circuito con Obstáculos 2

# PARTE 2: Diseño automático de un controlador neuroborroso de tipo SUGENO.

## PARTE 2.1: Circuito Sin Obstáculos

## PARTE 2.2: Circuito con Obstáculos 1

## PARTE 2.3: Circuito con Obstáculos 2