



PRÁCTICA FINAL

SISTEMAS DE CONTROL INTELIGENTE

Álvaro Zamorano Ortega, Gabriel López Cuenca

Contenido

Introducción 2

Mamdani 3

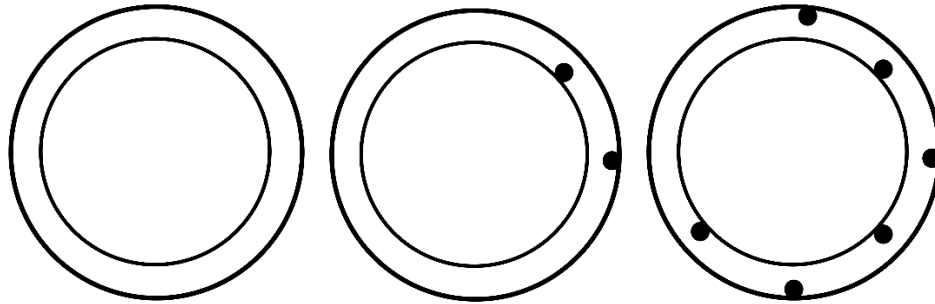
Sugeno..... 6

Conclusiones 10


Introducción

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.

En este caso dispondremos de 3 circuitos diferentes en función de cómo tienen situados los obstáculos en ellos:



El robot móvil sobre el que se probarán los controladores tiene las siguientes características:

Longitud	33 cm	
Anchura	28 cm	
Altura	15 cm	
Velocidad lineal máx.	1 m/s	
Velocidad angular máx.	1 rad/s	

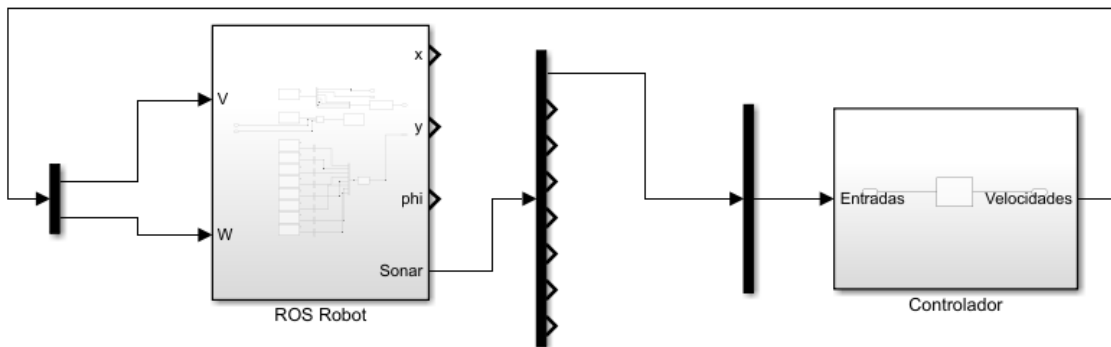
Este dispone de sensores de posicionamiento absoluto con respecto al centro geométrico del círculo que forma el circuito y un anillo de 8 sensores de ultrasonidos.

El comportamiento del robot y su interacción con el entorno se simula en la plataforma ROS STDR.

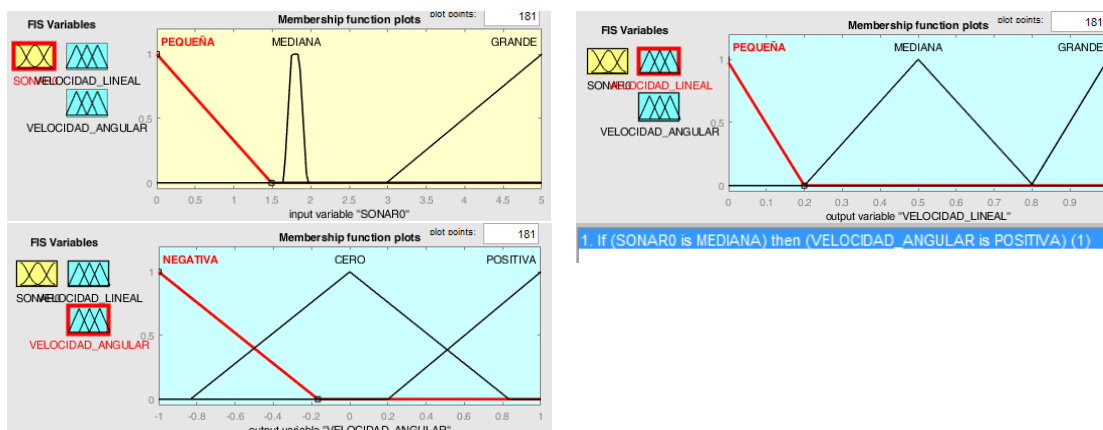
Mamdani

Hemos realizado diferentes controladores hasta conseguir el objetivo deseado, es decir, recorrer en el menor tiempo posible el circuito mediante el mínimo número de sensores posibles.

En primer lugar (**controlador1**), llegamos a la conclusión de que únicamente con los datos del sensor **SONAR_0** nuestro robot sería capaz de dar la vuelta al circuito en el caso de que este no contuviera obstáculos. Para usar estos datos, el esquema de Simulink empleado era el siguiente:



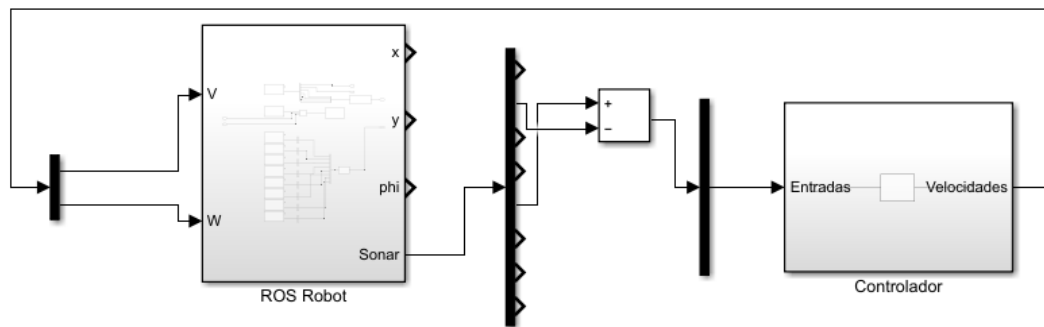
El controlador de tipo Mamdani empleado únicamente tenía **una entrada y dos salidas** (ambos tipos de velocidades), aunque solo controlaba la velocidad angular ya que la velocidad lineal no se cambiaba, es decir, se dejaba fija a 0.5 (predeterminada). Este controlador tiene el siguiente contenido:



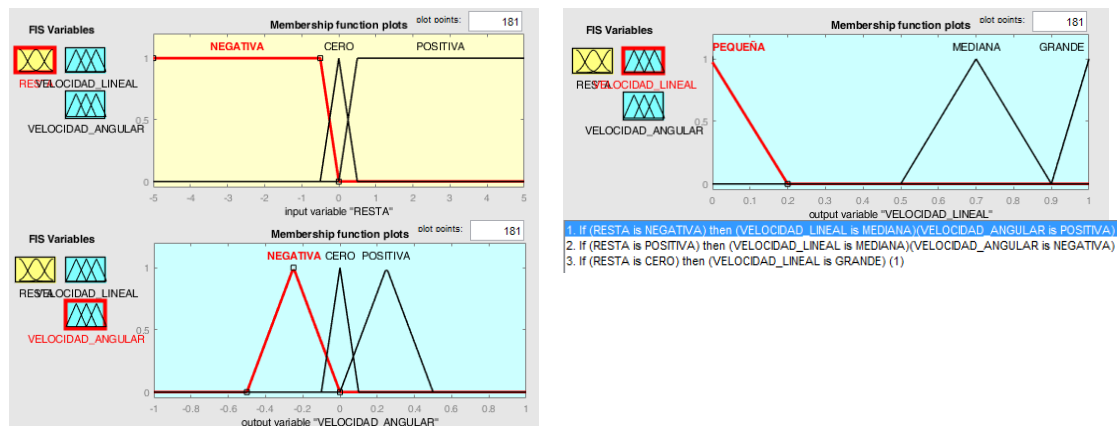
De esta forma, solo con **una regla** y los datos de **un sensor**, el robot es capaz de dar la vuelta **completa** al circuito. En caso de que este se separe más de 1.5 m de la pared interior, se le aplica una velocidad angular positiva para que continúe cerca de la misma. Así únicamente conseguíamos uno de los objetivos, el de recorrer el circuito, pero no el de hacerlo en el menor tiempo posible.

El comportamiento del robot se puede visualizar en el siguiente [vídeo](#).

A continuación (**controlador2**), utilizamos los datos de los sensores **SONAR_1** y **SONAR_4** por su posición en el robot ya que nos serviría para controlar las **paredes** y detectar los **obstáculos** que nos encontrásemos en el recorrido. Estos datos son usados en forma de **resta** y se le pasan al controlador de la siguiente forma:



El controlador tiene las siguientes características:



En este caso fue necesario indicar que, mientras el robot estuviera **girando**, es decir, cuando la entrada del controlador fuera negativa o positiva, la velocidad lineal disminuyera ya que sino este necesitaba mayor tiempo para realizar el mismo giro. Por otro lado, en el resto de los casos en los que el robot no necesitaba girar, se le indicaba una velocidad lineal entre 0.9 y 1 m/s.

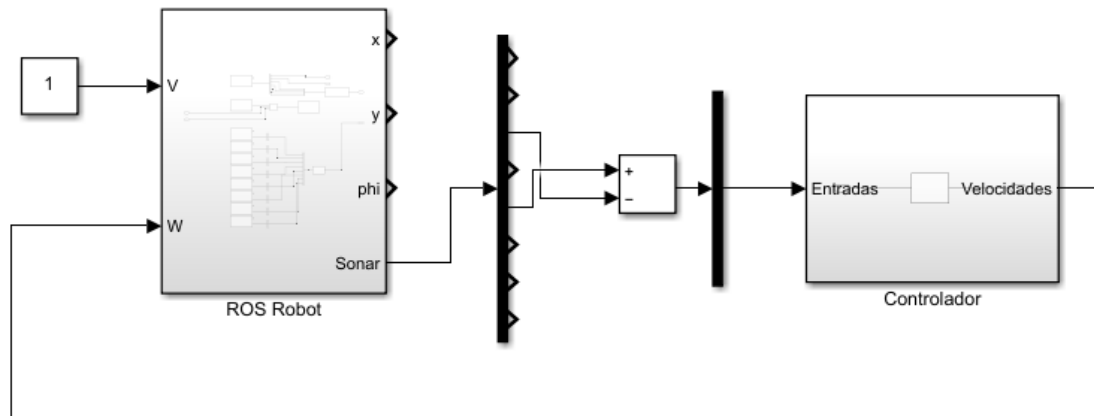
Como todas las reglas se tienen en cuenta, en la mayoría de los casos la velocidad lineal se encontraba entre 0.8 y 0.9 m/s por lo que no conseguíamos recorrer el circuito de la forma más **rápida** posible.

Es importante indicar que dicho controlador sirve tanto para recorrer el circuito con obstáculos como sin ellos, pruebas de ello son:

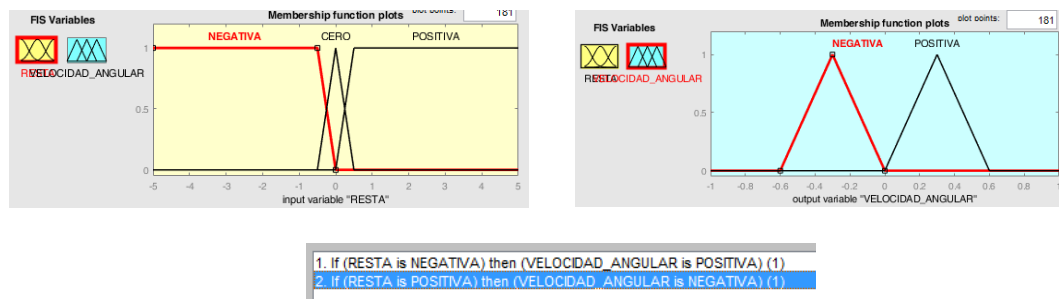
- [Recorrido sin obstáculos.](#)
- [Recorrido con obstáculos 2.](#)

Por último (**controlador3**), a partir del controlador anterior, observamos que, para disminuir el tiempo, la velocidad **lineal** se podía dejar constante a **1 m/s** (máximo del robot) y únicamente controlar la velocidad **angular**. Además, por la posición en la que se inicia el robot en el circuito y como este continua por él, hemos optado por usar el **SONAR_2** ya que este detecta con anterioridad los obstáculos al SONAR_1. Por tanto, la entrada del controlador será ahora la **resta** del SONAR_4 y el SONAR_2.

El esquema de Simulink es:



Y las características del controlador nombrado son:



Es decir, ahora este únicamente tiene **una salida**, la VELOCIDAD_ANGULAR con dos funciones de pertenencia (NEGATIVA Y POSITIVA). Las funciones a la entrada seguirán siendo las mismas.

Como se ha dicho anteriormente, de esta forma se consigue recorrer el circuito a la **máxima velocidad** que nos ofrece el robot y, con el intercambio del SONAR_1 por el SONAR_2, conseguimos ir más pegados a la pared interior (lo máximo que se puede para posteriormente conseguir pasar los obstáculos que en ella hay), es decir, recorrer **menos espacio**.

En los siguientes vídeos podemos observar cómo el robot realiza su objetivo de una forma más **rápida y eficiente** que anteriormente:

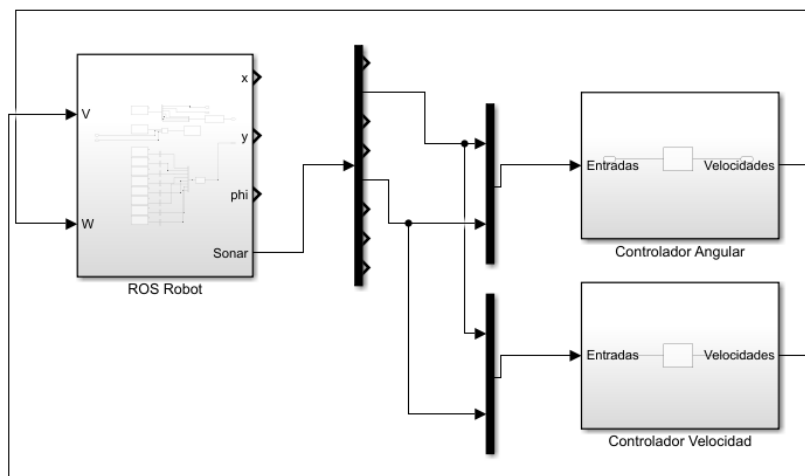
- [Recorrido sin obstáculos.](#)
- [Recorrido con obstáculos 2.](#)

Sugeno

Al igual que en el tipo Mamdani, hemos realizado diferentes controladores para llegar al objetivo de recorrer el circuito en el menor tiempo posible.

En primer lugar (**sugenoSinObstaculos**), controlamos **manualmente** el robot en el circuito sin obstáculos mediante el script de Matlab *ControlManualRobot.m*. El control se realizó para que este siempre fuera por el medio del circuito, llevando una velocidad lineal de **1 m/s** y angular entre **0 y 0.1 rad/s**.

Debido a que los controladores de tipo Sugeno solo tienen una salida, creamos **dos controladores**, uno para la velocidad lineal y otro para la angular. Los bloques de Simulink indicados para esta tarea se crearon de la siguiente forma:

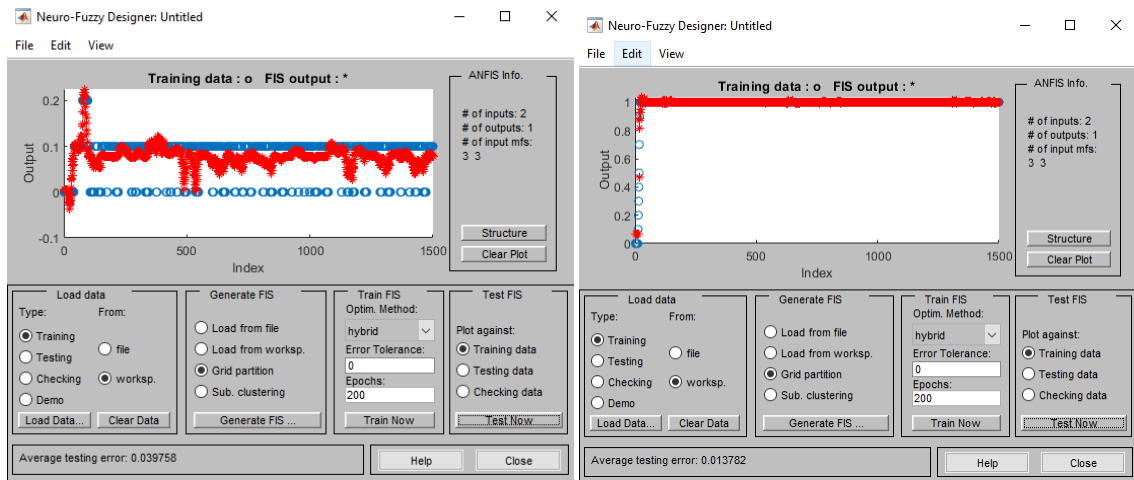


Conociendo por pruebas anteriores (al realizar Mamdani) que con los datos de los sensores **SONAR_1** y **SONAR_4** el robot se comportaba según lo requerido, fueron estos los datos de **entrenamiento** escogidos para realizar el controlador de tipo Sugeno. Dichos datos fueron introducidos en dos variables distintas (una por cada controlador) de la siguiente manera:

```
train_angular = training(:, [2,5,12])
indices = round(linspace(1, size(train_angular, 1), 1500))
train_angular = train_angular(indices, :)
train_angular(isinf(train_angular)) = 5.0
train_angular = double(train_angular)

train_velocidad = training(:, [2,5,13])
indices = round(linspace(1, size(train_velocidad, 1), 1500))
train_velocidad = train_velocidad(indices, :)
train_velocidad(isinf(train_velocidad)) = 5.0
train_velocidad = double(train_velocidad)
```

Mediante la herramienta **Anfisedit** creamos los controladores cargando las variables **train_angular** y **train_velocidad**. Ambos fueron creados con funciones **triangulares** para las entradas y **lineales** para la salida, y con un método de entrenamiento **híbrido** con **200 épocas**, quedando los gráficos de test de la siguiente manera:

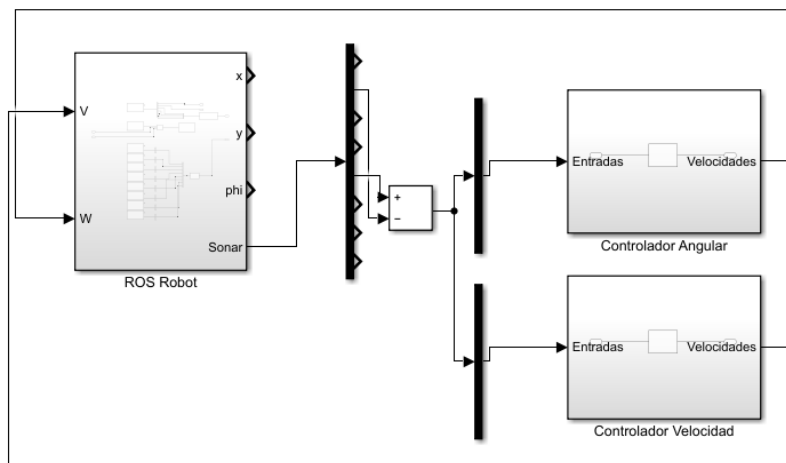


El error del controlador de velocidad angular fue de 0.0397 y el de velocidad lineal a 0.0137.

Como se ha dicho anteriormente, este controlador fue creado para el [recorrido sin obstáculos](#).

Ya que este controlador solo cumplía una parte de las expectativas de la práctica, procedimos a realizar un **nuevo entrenamiento** del mismo, es decir, recoger nuevos datos durante el control manual del robot en el mapa con obstáculos 2 (*sugenoManual*).

Ahora a este controlador, al igual que en Mamdani, se le pasan los datos restados:



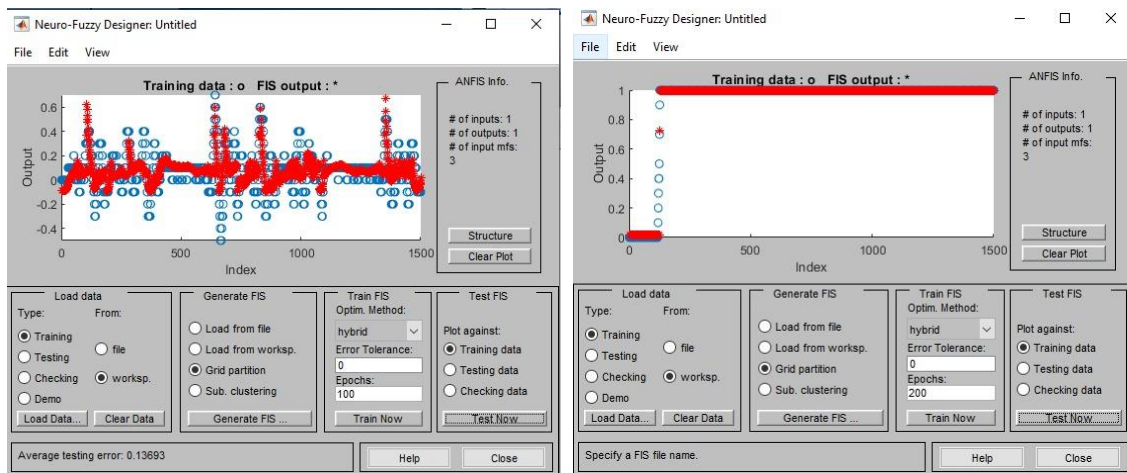
De igual forma, estos datos siguen siendo los provenientes de los sensores SONAR_1 y SONAR_4. Para realizar la **resta** y posteriormente usarla en la herramienta Anfisedit se realizó el siguiente script de Matlab.

```
columna_2 = training(:,[2]);
columna_4 = training(:,[5]);
resta = columna_4-columna_2;

train_angular = [resta,training(:,12)]
indices = round(linspace(1,size(train_angular,1),1500))
train_angular = train_angular(indices,:)
train_angular(isinf(train_angular)) = 5.0
train_angular = double(train_angular)

train_velocidad = [resta,training(:,13)]
indices = round(linspace(1,size(train_velocidad,1),1500))
train_velocidad = train_velocidad(indices,:)
train_velocidad(isinf(train_velocidad)) = 5.0
train_velocidad = double(train_velocidad)
```


Y los gráficos de test obtenidos en la nombrada herramienta fueron:



Como se puede apreciar, el **error** de test para la velocidad angular (0.13) es mucho mayor que anteriormente ya que esta tiene muchas más variaciones a la hora de esquivar los obstáculos y, por lo tanto, resulta más difícil obtener una buena **generalización**. Por otro lado, la velocidad lineal se sube a **1 m/s** en el comienzo del control.

Con este controlador prácticamente se consigue recorrer el entorno con obstáculos 2 a **excepción** de uno de los últimos obstáculos. Pruebas de ello son:

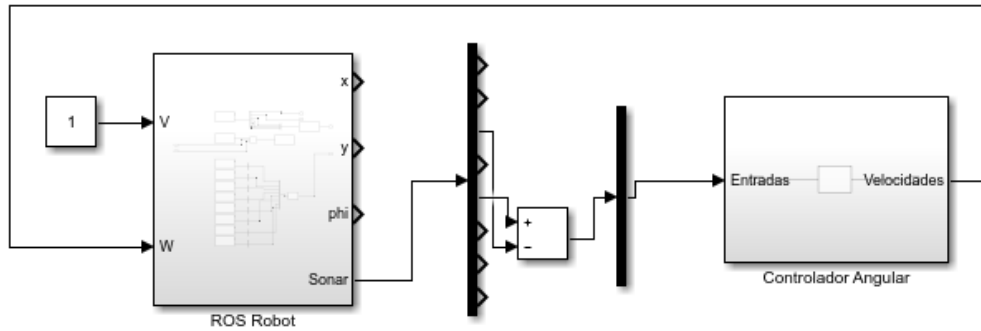
- [Recorrido sin obstáculos.](#)
- [Recorrido con obstáculos 2.](#)

Tras varias pruebas de control manual y entrenamiento **no** se ha conseguido obtener un controlador que lleve a cabo la tarea deseada, a pesar de disminuir la velocidad lineal prácticamente a 0.2 m/s. Siempre había algún momento en el que el robot no conseguía pasar alguno de los obstáculos colocados en la pared interior del recorrido.

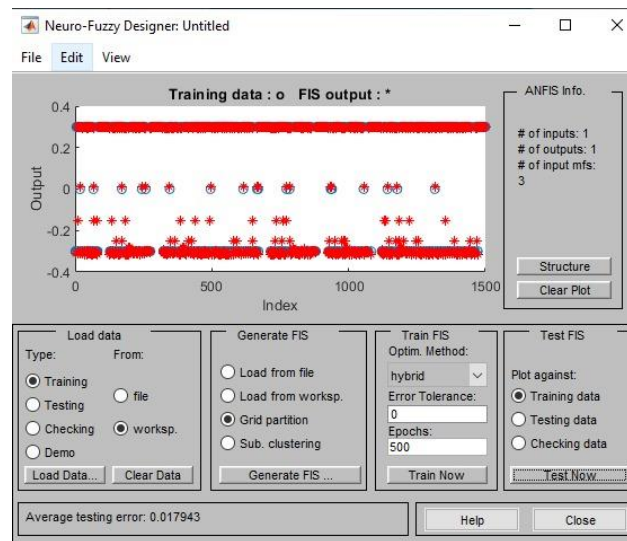
Por último (**sugenoDatosMamdani**), fueron usados como datos de **entrenamiento** los generados por el control realizado con **Mamdani**, es decir, pusimos a funcionar el robot con el controlador generado en la parte anterior y recogimos los datos que este nos ofrecía. Para realizar dicha tarea se usó el siguiente script de Matlab:

```
resta = out.resta;
train_angular = [resta,out.angular]
indices = round(linspace(1,1501,1500))
train_angular = train_angular(indices,:)
train_angular(isinf(train_angular)) = 5.0
train_angular = double(train_angular)
```

Los sensores también se cambiaron, ahora se usan los **SONAR_2** y **SONAR_4** en forma de **resta**. El esquema de Simulink es el siguiente:



La gráfica de test para la velocidad angular fue:



En este caso se puede apreciar una menor variación de la velocidad por lo que se consigue una mayor **generalización**. El error de test en este caso es de 0.017.

Respecto a la velocidad lineal se hace constante a **1 m/s** directamente desde Simulink, es decir, no se realiza control de ella.

De esta forma se **consigue** dar la vuelta completa al circuito con obstáculos, así como al circuito sin obstáculos, de una forma muy **similar** a la que se consigue con el último de los controladores de tipo Mamdani. Encontramos pruebas de ello en:

- [Recorrido sin obstáculos.](#)
- [Recorrido con obstáculos 2.](#)

Conclusiones

A pesar de que al usar un controlador de tipo Mamdani es necesario desarrollar las funciones de pertenencia de sus entradas y salidas junto a los rangos de cada una de ellas, los resultados obtenidos resultan más fáciles de comprender y, por lo tanto, de poder corregir errores.

Por otro lado, al emplear Sugeno, como el controlador se hace de forma automática, aunque los errores de test obtenidos sean muy bajos y en la gráfica se observe una aparente generalización, el robot puede no realizar de forma correcta la tarea deseada. Han sido numerosas las ocasiones en las que no entendíamos el comportamiento del mismo, a pesar de haber realizado un control manual exhaustivo.

Por tanto, cabe decir que realizar un controlador de tipo Mamdani ofrece una mayor interpretabilidad, así como una ventaja en los sistemas MIMO como es este caso.