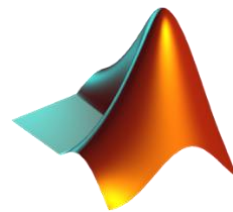


# Sistemas de control Inteligente

Práctica final

Juan José Córdoba Zamora  
Juan Casado Ballesteros



Universidad  
de Alcalá

## Índice

|   |           |
|---|-----------|
| <b>INTRODUCCIÓN.....</b>  | <b>3</b>  |
| <b>PARTE 1. DISEÑO MANUAL DE UN CONTROL BORROSO DE TIPO MAMDANI .....</b>             | <b>5</b>  |
| SIN OBSTÁCULOS .....  | 5         |
| <i>Primeros intentos .....</i>  | <i>5</i>  |
| <i>Solución final .....</i>   | <i>5</i>  |
| <i>Diseño del controlador .....</i>   | <i>5</i>  |
| <i>Resultados .....</i>   | <i>7</i>  |
| CON OBSTÁCULOS .....  | 7         |
| <i>Resultados .....</i>   | <i>10</i> |
| <b>PARTE 2. DISEÑO AUTOMÁTICO DE UN CONTROLADOR NEUROBORROSO DE TIPO SUGENO .....</b> | <b>13</b> |
| SIN OBSTÁCULOS .....  | 13        |
| <i>Resultados .....</i>   | <i>15</i> |
| CON OBSTÁCULOS .....  | 16        |
| <i>Resultados .....</i>   | <i>18</i> |
| <b>PARTE 3. CONTROLADOR PARA LA VELOCIDAD LINEAL Y LA ANGULAR. ....</b>               | <b>20</b> |
| CONTROLADOR DE TIPO MAMDANI CIRCUITO SIN OBSTÁCULOS .....                             | 20        |
| <i>Resultados .....</i>   | <i>21</i> |
| CONTROLADOR DE TIPO MAMDANI CIRCUITO CON OBSTÁCULOS .....                             | 22        |
| <i>Resultados .....</i>   | <i>23</i> |
| CONTROLADOR DE TIPO SUGENO CIRCUITO SIN OBSTÁCULOS .....                              | 24        |
| <i>Resultados .....</i>   | <i>25</i> |
| CONTROLADOR DE TIPO SUGENO CIRCUITO CON OBSTÁCULOS.....                               | 25        |
| <i>Resultados .....</i>   | <i>26</i> |
| CONCLUSIONES.....   | 27        |

## Introducción

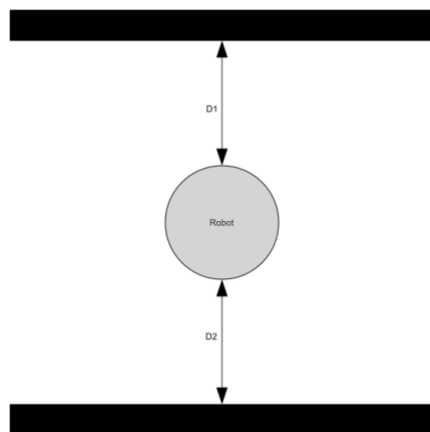
El objetivo de esta práctica es diseñar un conjunto de controladores borrosos capaces de dirigir a un robot móvil diferencial por el entorno sin chocarse. Debido a que las condiciones del entorno son muy concretas los controladores no serán de propósito general si no que se diseñarán de forma específica para funcionar lo mejor posible en el entorno.

Un objetivo adicional al de no chocarse con los obstáculos es que los controladores generados sean lo más rápidos posible. Esto se debe a que posteriormente deberemos competir con el mejor controlador que hayamos creado siendo el ganador el que más rápido complete el circuito.

Los robots móviles diferenciales que controlaremos serán simulados mediante ROS y STRD. Utilizaremos ROS como intermediario comunicando el código del controlador realizado en Matlab con el simulador STRD. Para hacer esto utilizaremos el sistema de publicación suscripción de mensajes sobre topics que ofrece ros.

Los apartados que se nos han propuesto para esta práctica se han realizado secuencialmente, hemos comenzado por crear un controlador Mamdani capaz de no chocarse en el entorno sin obstáculos, posteriormente hemos mejorado ese controlador para que fuera capaz de esquivar los obstáculos también añadiendo nuevas entradas al controlador. Finalmente hemos utilizado ambos controladores para capturar datos con los que poder entrenar los controladores de Sugeno.

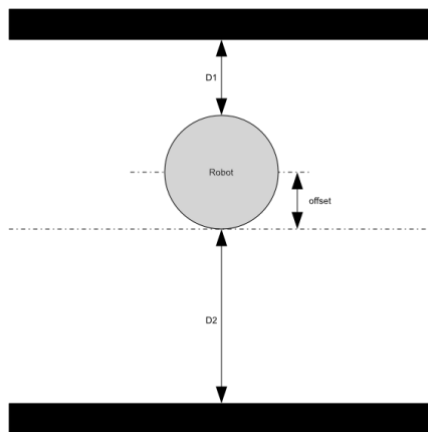
Como entradas de los controladores hemos decidido utilizar distintas diferencias entre las medias de distancias captadas por los ultrasonidos. Hemos hecho esto, pues nos ha parecido más intuitivo hacer que el resultado de restar dos ultrasonidos deba de intentar hacerse cero en vez de intentar que las distancias medidas por ambos ultrasonidos fueran la misma. Adicionalmente esto nos ha permitido establecer valores de offset que hagan que el robot se centre en un punto u otro de la pista.



Podemos centrar el robot en la pista haciendo que la diferencia entre las distancias sea cero. Cuando dicha diferencia tenga un valor positivo el controlador deberá producir una velocidad angular con un signo que será del signo contrario cuando la diferencia también lo sea.

En este caso se puede considerar que el offset es 0.

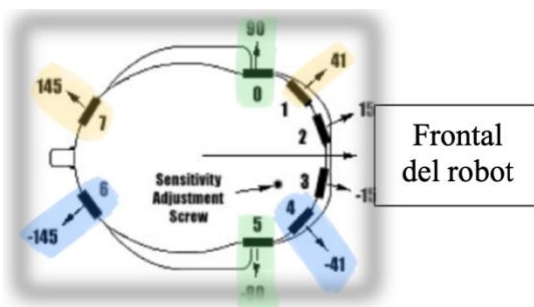
*Ilustración 1 Robot centrado sin offset*



Si el offset tiene un valor distinto de 0 nos centraremos desplazados del centro tanto como sea el valor del offset.  
 Hacer esto tiene la gran ventaja de que el controlador tendrá la misma lógica siendo lo único que cambia el valor de la entrada.  

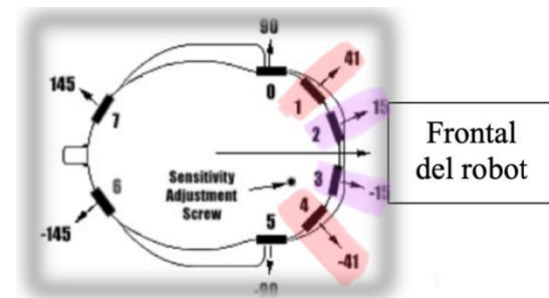
$$\text{Entrada} = D1 - D2 + \text{offset}.$$

Ilustración 2 Robot centrado con offset



Las entradas del controlador utilizadas para centrarnos son restas de las distancias proporcionadas por las siguientes parejas de ultrasonidos, cada una tendrá su propio offset.

Ilustración 3 Sonares utilizados para centrar al robot



Del mismo modo utilizamos las siguientes restas de distancias de estas parejas de ultrasonidos para esquivar los obstáculos.

Ilustración 4 Sonares utilizados para esquivar obstáculos

La velocidad lineal será una constante, de modo que los controladores creados actúan solo sobre la velocidad angular. Realizamos esto ya que con controlar solo esta magnitud hemos sido capaces de realizar toda la práctica y además porque deseamos ir lo más rápido posible.

En el último apartado (**Parte 3**) hemos realizado controladores adicionales para controlar tanto la velocidad angular como la lineal. De este modo demostramos que para ir más rápido no es necesario controlar la velocidad lineal si no que es mejor hacerlo con esta constante al máximo posible. No obstante al controlar también la velocidad lineal los controladores deberían ser más seguros y no se acercarse tanto a los obstáculos.

## Parte 1. Diseño manual de un control borroso de tipo MAMDANI

Realizaremos dos controladores de tipo Mamdani para que el robot pueda completar el circuito sin obstáculos y con obstáculos. El trabajo realizado en esta parte ha consistido por tanto en elegir las entradas más adecuadas para los controladores, así como elegir las funciones de pertenencia más adecuadas; en número, posición y forma para los conjuntos borrosos de entrada y salida.

### SIN obstáculos

#### Primeros intentos

Al principio decidimos utilizar solo la diferencia de los sensores de los lados (el 0 y el 5), para mantenernos en el centro o en la parte del círculo que se decidiera mediante el offset explicado anteriormente. Al realizar este controlador nos dimos cuenta de que el robot cabeceaba mucho y era complicado seguir la trayectoria porque había veces en las que el robot cabeceaba demasiado y se giraba o se chocaba con las paredes.

Para solucionar esto decidimos aumentar el número de sensores a utilizar, primero pensamos en utilizar la diferencia del 1 y el 4 aparte del anterior, para así tener más puntos de referencia para guiarnos.

#### Solución final

Por último, se nos ocurrió utilizar la diferencia de sensores del mismo lado para así detectar cuando el robot se desplaza más de la cuenta hacia alguno de los dos lados. Así que los sensores utilizados para mantener el robot en el centro (utilizando el offset) fueron la diferencia de los sensores, 0 con el 5, la diferencia del 1 con el 7 y la diferencia del 4 con el 6 como se muestra en la **ilustración 3**.

Como se ha comentado anteriormente este conjunto de sensores utilizados nos permitía realizar el circuito sin cabecear y además siguiendo una circunferencia casi perfecta.

Otra mejora que introducimos fue utilizar el offset para centrar el robot más a la izquierda de la pista como se muestra en la **ilustración 2**, en vez de centrar el robot justo en el medio del círculo como se muestra en la **ilustración 1**. Para así al realizar la circunferencia recorrer menos distancia y por lo tanto intentar realizar más rápido la vuelta. Esto se ha conseguido utilizando el offset explicado en la introducción.

En este apartado solo controlamos la velocidad angular, ya que la velocidad lineal hemos decidido de ponerla constante, a la máxima posible que el robot admite, con la intención de realizar el circuito en el menor tiempo posible.

#### Diseño del controlador

Tenemos 3 conjuntos borrosos para la entrada y un conjunto borroso para la salida que se explicarán más adelante para controlar únicamente la velocidad angular ya que como se ha explicado anteriormente la velocidad lineal decidimos ponerla constante al máximo para dar la vuelta al circuito más rápido.

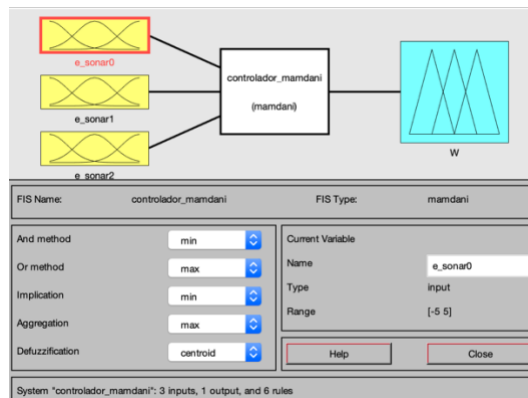


Ilustración 5 Controlador tipo Mamdani sin obstáculos

Tenemos dos reglas para cada conjunto borroso que dependiendo de donde este situado el robot, lo cual conocemos mediante la diferencia de las parejas de sonares, nos permiten decidir el signo y la magnitud que el controlador debería de generar.

1. If (e\_sonar0 is negativo) then (W is negativo) (1)
2. If (e\_sonar0 is positivo) then (W is positivo) (1)
3. If (e\_sonar1 is negativo) then (W is negativo) (1)
4. If (e\_sonar1 is positivo) then (W is positivo) (1)
5. If (e\_sonar2 is negativo) then (W is positivo) (1)
6. If (e\_sonar2 is positivo) then (W is negativo) (1)

Ilustración 6 Reglas del controlador tipo Mamdani

La **ilustración 7** corresponde con la función de pertenencia de la diferencia del sonar 5 con el sonar 0, aunque las funciones de pertenencia de los conjuntos borrosos e\_sonar1 y e\_sonar2 son las mismas. La **ilustración 8** corresponde a la salida del controlador, la velocidad angular.

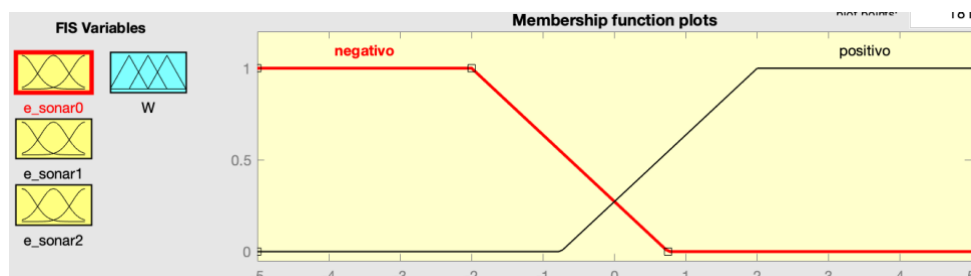


Ilustración 7 Conjunto borroso de la diferencia de los sonares

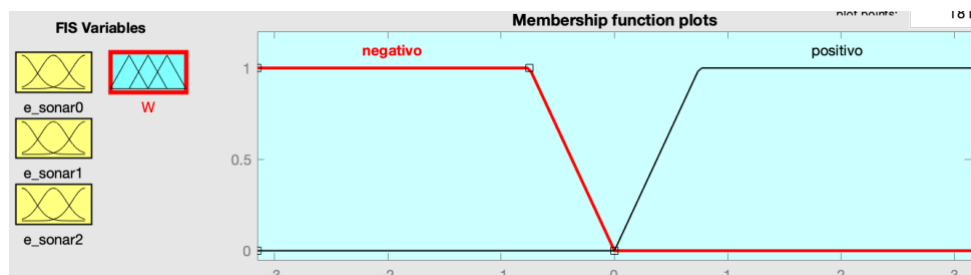


Ilustración 8 Conjunto borroso de la velocidad angular (salida)

Para unir el controlador con la planta (simulador en ROS) hemos utilizado el siguiente slx. Se ve como se obtiene la diferencia de los sonares y cómo se aplican los offset, también que saturamos la salida para que no de valores erróneos y pueda crear error al introducir la velocidad angular al robot.

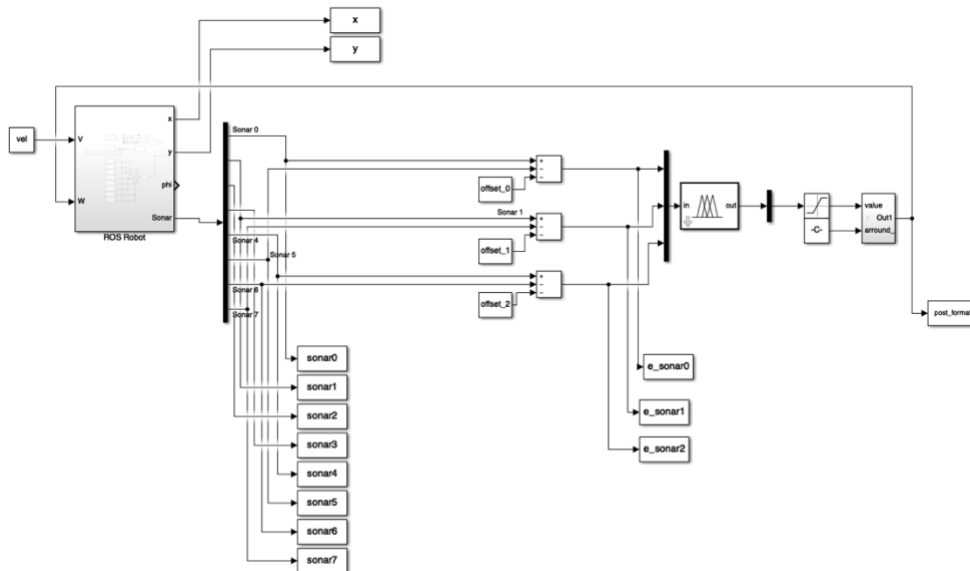
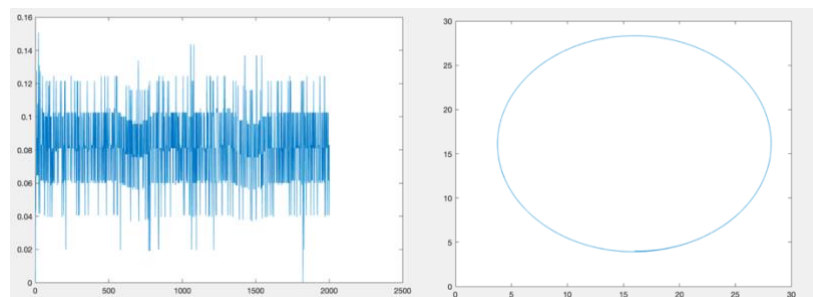


Ilustración 9 Diseño del archivo SLX sin obstáculos

## Resultados

El controlador sigue adecuadamente la trayectoria de forma muy suave. En ningún momento llega si quiera a tener que corregir aplicando velocidades negativas lo que podría haber ocurrido si el robot estuviera cabeceando. Las velocidades que aplica oscilan entorno a 0.08rad/s por lo que esa debe ser la velocidad angular constante que debería de aplicarse para realizar la circunferencia con una velocidad lineal constate de 1m/s.



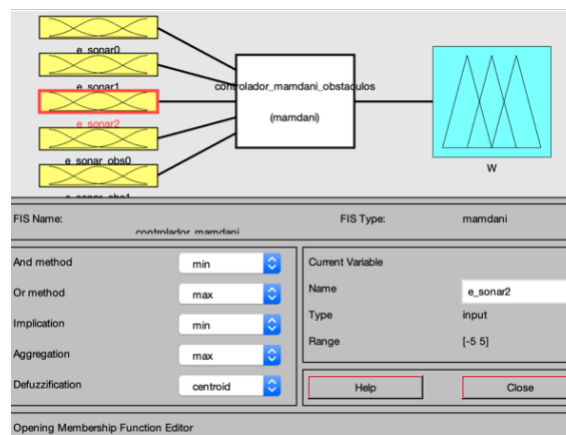
Gráficas con resultados

## CON obstáculos

Para realizar este circuito, utilizamos la misma configuración de sensores que en el de sin obstáculos, aunque añadimos dos conjuntos borrosos más para poder detectar cuando nos estamos aproximando a un obstáculo y así poder evitarlo. Estos nuevos conjuntos borrosos son la diferencia de los sensores 1 y 4 y de los sensores 2 y 3, para así poder observar los obstáculos de delante y ver en que lado se encuentran, como se muestra en la **ilustración 4**.

Al igual que en el controlador anterior, únicamente controlamos la velocidad angular y mantenemos constante la velocidad lineal al máximo, para así poder llegar a alcanzar al dar la vuelta el tiempo de 1 min con 10 segundos aproximadamente.

La estructura del controlador de Mamdani para el circuito con obstáculos es muy parecida al circuito sin obstáculos. Se han añadido dos conjuntos borrosos nuevos y en la salida del controlador se han añadido 2 funciones de pertenencia más que se explicarán más adelante.



*Ilustración 10 Controlador tipo Mamdani con obstáculos*

Las reglas de este controlador se pueden apreciar cómo son iguales excepto por las 4 últimas, dos para cada entrada nueva, dónde si la diferencia de los sonares utilizados para observar los obstáculos encuentra un obstáculo varía la velocidad angular hacia los extremos, es decir, si el robot encuentra un obstáculo en un lado, el controlador cambia la velocidad angular hacia valores muy negativos o muy positivos. Como se ha comentado varias veces anteriormente sólo se controla la velocidad angular y la velocidad lineal se mantiene constante al máximo.

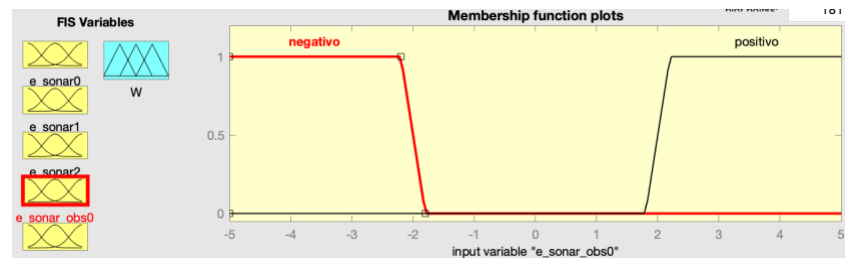
1. If (e\_sonar0 is negativo) then (W is negativo) (1)
2. If (e\_sonar0 is positivo) then (W is positivo) (1)
3. If (e\_sonar1 is negativo) then (W is negativo) (1)
4. If (e\_sonar1 is positivo) then (W is positivo) (1)
5. If (e\_sonar2 is negativo) then (W is positivo) (1)
6. If (e\_sonar2 is positivo) then (W is negativo) (1)
7. If (e\_sonar\_obs0 is positivo) then (W is muy\_positivo) (1)
8. If (e\_sonar\_obs0 is negativo) then (W is muy\_negativo) (1)
9. If (e\_sonar\_obs1 is positivo) then (W is muy\_positivo) (1)
10. If (e\_sonar\_obs1 is negativo) then (W is muy\_negativo) (1)

*Ilustración 11 Reglas del controlador tipo Mamdani*

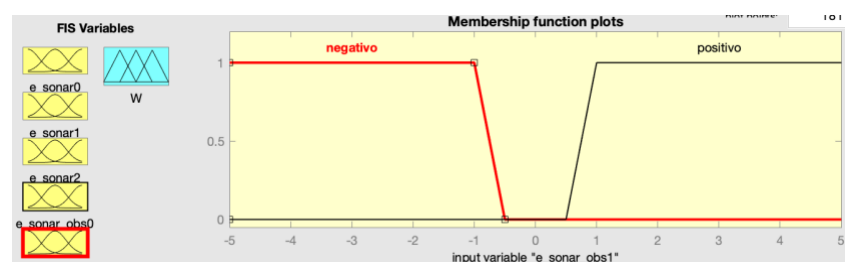
Los conjuntos borrosos nuevos en las entradas son distintos al resto **ilustración 10**. Entre ellos son muy parecidos, la única diferencia es cuando se produce que el conjunto active la regla, en la **ilustración 13** se ha reducido el espacio entre ambos porque en cuánto se reconozca que haya un obstáculo por los sensores 1 y 4 interesa que se mueva antes hacia un lado o mantenerse centrado en el caso de que no haya nada, en cambio, para la **ilustración 12**, interesa que las reglas que tengan esas funciones solo se activen cuando los sensores de más al frente el 2 y el 3 visualicen un obstáculo y así poder girar rápidamente hacia el lado contrario al que se ha encontrado el obstáculo.



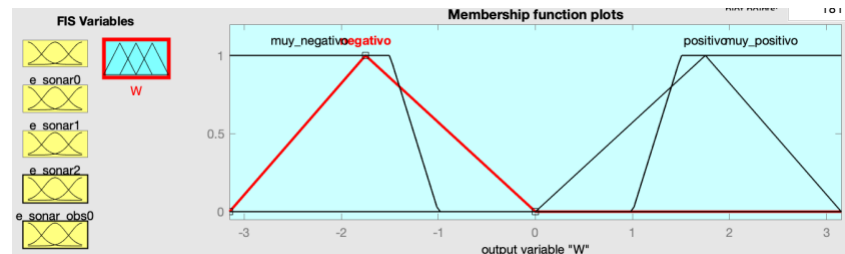
En la **ilustración 14** se puede apreciar que se han añadido 2 nuevas funciones de pertenencia complementarias a las anteriores para hacer que la velocidad angular sea muy negativa o muy positiva y así poder tener más posibilidades a la hora de observar y esquivar los obstáculos.



*Ilustración 12 Conjunto borroso de la diferencia de los sensores para detectar obstáculos*



*Ilustración 13 Conjunto borroso de la diferencia de los sensores para detectar obstáculos*



*Ilustración 14 Conjunto borroso de la velocidad angular (salida)*

Este diseño del slx es igual que el anterior, sólo que se han añadido 4 sensores más utilizando la diferencia entre ellos y también entre un offset (igual que los demás). Fue necesario saturar las entradas del controlador, porque había algunas veces en los que el valor que leía se pasaba del rango del controlador.

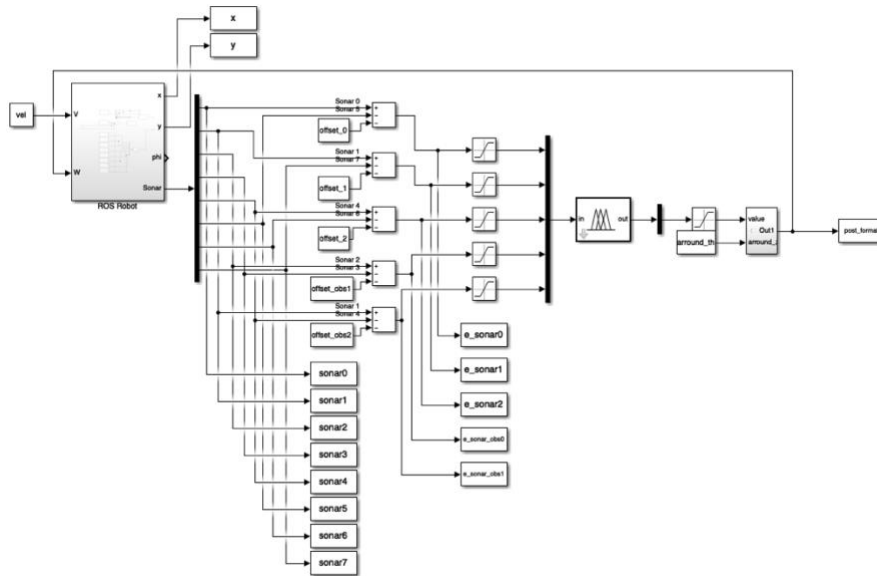


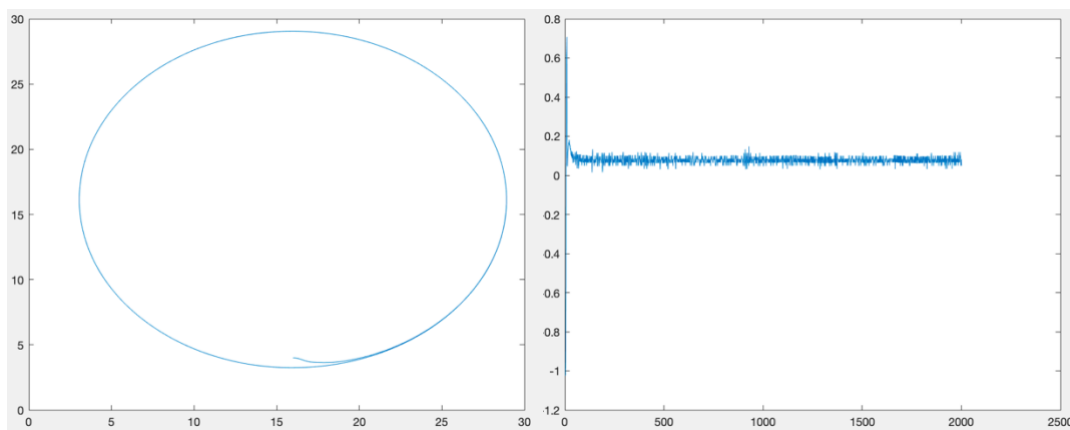
Ilustración 15 Diseño del archivo SLX con obstáculos

## Resultados

El controlador logra completar los circuitos con obstáculos y sin ellos incluyendo un circuito extra que nosotros mismos creamos.

### Mapa sin obstáculos

Podemos apreciar una gran diferencia entre este controlador y el anterior. El controlador anterior utiliza el offset para centrarse en la pista a una distancia de la pared igual a la que ya tiene cuando se inicia para poder ir más rápido. Por el contrario, para este otro controlador debido a que debe de esquivar los obstáculos no podemos centrarlo en esa posición en la pista si no en otra más próxima a su centro. Es debido a esto que cuando se inicia debe desplazarse a esa posición tras lo cual su comportamiento ya es el mismo que el del controlador anterior.

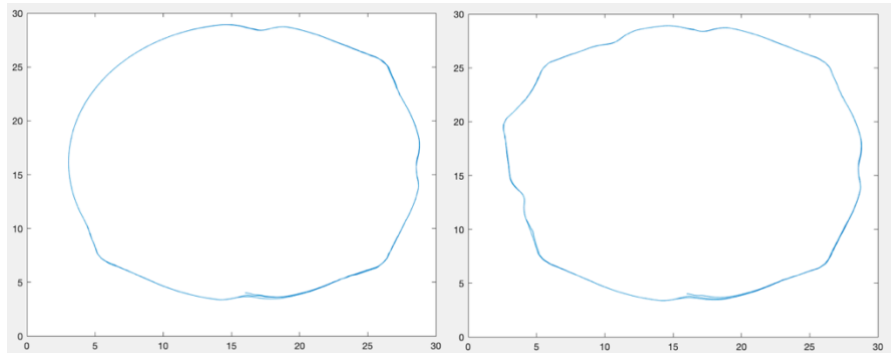


Gráficas con resultados

### Mapas con obstáculos

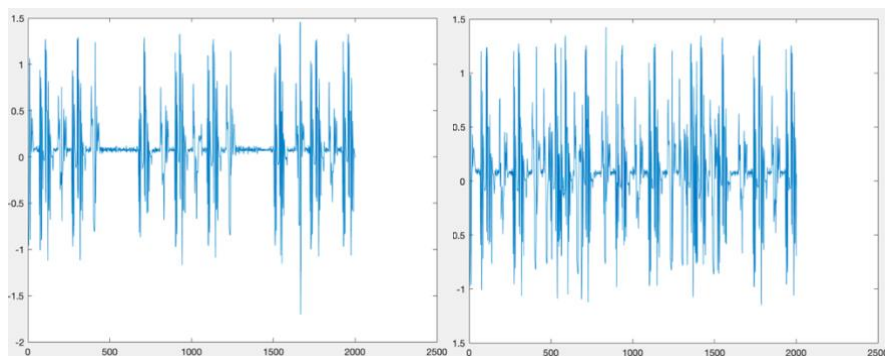
La imagen de la izquierda es la que contiene más obstáculos de las 2 proporcionadas para realizar la práctica y como se puede ver ha sido capaz de seguir la trayectoria esquivando todos los obstáculos sin chocarse y además yendo siempre por el mismo lado.

La imagen de la derecha corresponde a un mapa creado por nosotros para probar el controlador de Sugeno con obstáculos (se explicará más adelante el mapa en la parte de Sugeno con obstáculos). Y también como se observa ha sido capaz de superar todos los obstáculos sin chocarse siguiendo siempre la misma trayectoria.



*Gráficas con resultados*

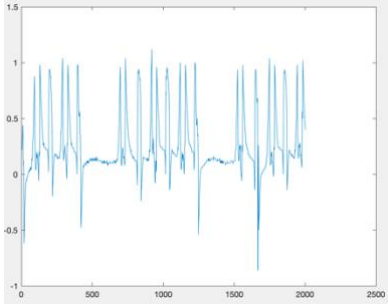
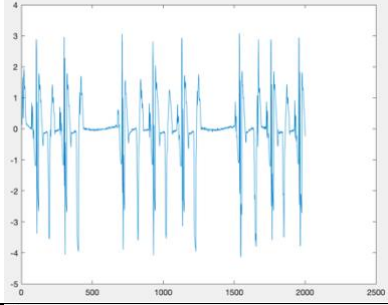
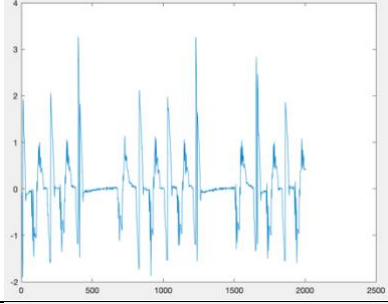
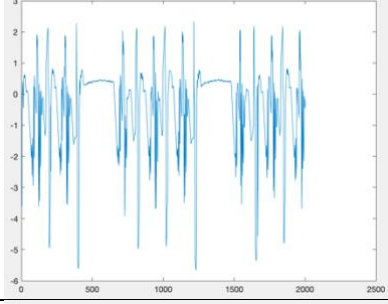
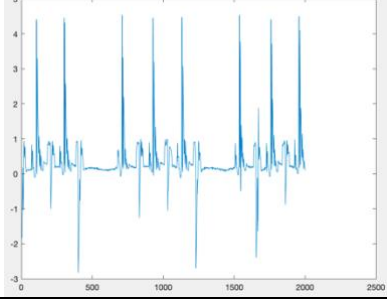
Mostramos ahora la velocidad angular en el tiempo, para ambas trayectorias. Se puede destacar que cuando no hay ningún obstáculo la velocidad angular está aproximadamente en el  $0.8\text{rad/s}$  y se mantiene, en cambio cuando el robot ve y esquiva algún obstáculo se aprecia como la velocidad angular varía dependiendo del lado donde este el obstáculo para esquivarlo. No obstante, también se puede ver que del cuando el controlador esquiva los obstáculos cabecea, tiene picos alternando entre velocidades positivas y negativas, mientras que cuando no hay obstáculos no lo hace.



*Gráficas con resultados*

Si nos fijamos en los datos que ve el sonar 0 dedicado a esquivar obstáculos cuando el robot se aproxima al obstáculo hay bastantes oscilaciones y es imposible que el robot sea capaz de esquivar el obstáculo únicamente con ese. En cambio, si añadimos el otro sonar que utilizamos para esquivar se ve como si los dos se combinan el controlador y el robot puede perfectamente esquivar el obstáculo, ya que las oscilaciones se compensan y al final el robot se desplazada hacia el lado correcto evitando chocarse.

En la siguiente tabla las entradas del controlado en el mapa de con obstáculos 2 dado en el material de la práctica.

|  |  |
|--|--|
| Diferencia del sonar 0 y el sonar 5  |    |
| Diferencia del sonar 1 y el sonar 7  |    |
| Diferencia del sonar 4 y el sonar 6  |   |
| Diferencia del sonar 2 y el 3 utilizados para ver si hay algún obstáculo cerca |  |
| Diferencia del sonar 1 y el 4 utilizados para ver si hay algún obstáculo cerca |  |

Podemos ver las dos últimas entradas tienen grandes picos cuando están cerca de los obstáculos, la primera con más ruido y la segunda con menos. Por el contrario, las otras no son tan fiables para detectar obstáculos pues los ven demasiado tarde.

## Parte 2. Diseño automático de un controlador neuroborroso de tipo SUGENO

En esta segunda parte el objetivo era el de crear un controlador neuroborroso de tipo Sugeno que aprendiera por el solo a imitar unos pares de entradas y salidas proporcionados.

Para obtener estos pares de entradas y salidas se nos propuso controlar nosotros el robot y grabarlos. No obstante, en vez de hacer esto hemos utilizado los controladores Mamdani creados anteriormente para obtenerlos.

Para entrenar los controladores comenzamos ha hacerlo utilizando tanto datos de test como de validación a parte de los de training. Posteriormente decidimos dejar de utilizar los datos de test ya que no los estábamos utilizando realmente. Veíamos la comparación de la salida obtenida por el controlador con las esperadas según los datos de test pero no sin ser capaces de discernir a partir de ella si el entrenamiento había sido adecuado o no. Para hacerlo era mucho mejor verlo sobre el simulador.

Adicionalmente decidimos probar a entrenar sin datos de validación, utilizando solo datos de entrenamiento lo cual nos proporcionó mucho mejores resultados al ejecutar el controlador en el simulador incluso utilizando mapas para los que el controlador no había sido entrenado.

### SIN obstáculos

Debido a la simplicidad de la tarea a realizar llevarla a cabo fue sencillo también. Grabamos las entradas y las salidas del controlador Mamdani creado para recorrer una vuelta en el circuito sin obstáculos. Utilizando esos datos entrenamos en el controlador Sugeno.

```
e_sonar0_ = e_sonar0.signals.values;
e_sonar1_ = e_sonar1.signals.values;
e_sonar2_ = e_sonar2.signals.values;
out_ = post_format.signals.values;

e_sonar0_(isinf(e_sonar0_)) = 5.0;
e_sonar1_(isinf(e_sonar1_)) = 5.0;
e_sonar2_(isinf(e_sonar2_)) = 5.0;

train = [e_sonar0_ e_sonar1_ e_sonar2_ out_];
train = double(train);

save train.dat train -ascii
```

*Ilustración 16 Guardado datos de entrenamiento*

Una vez se han guardado los datos de entrenamiento, utilizamos la herramienta de anfisedit con la siguiente configuración para entrenar el controlador en el mapa sin obstáculos.

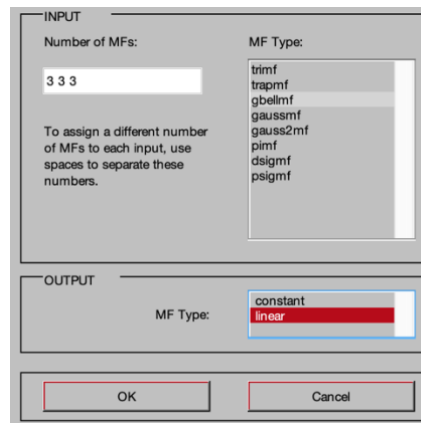


Ilustración 17 Configuración del fis

En la configuración del fis utilizamos funciones de tipo gbellmf y 3 funciones en cada conjunto borroso.

Hemos decidido utilizar funciones de pertenencia de tipo gbellmf debido a que son derivables en todo su dominio. Otras funciones como las trapezoidales, que son las que hemos utilizado en Mamdani no son derivables en los vértices del trapecio.

En total hemos utilizado tres funciones de pertenencia para definir cada conjunto borroso. El ajuste de las funciones se ha realizado con la opción lineal en vez de constante.

Para entrenar el controlador como se muestra en la **ilustración 18** creamos un fis mediante grid partition, y lo entrenamos con 10 épocas únicamente con los datos de entrenamiento guardados con el código de la **ilustración 16**.

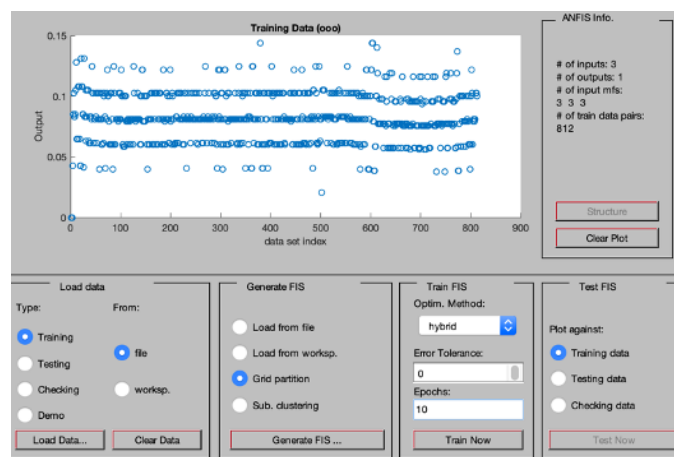


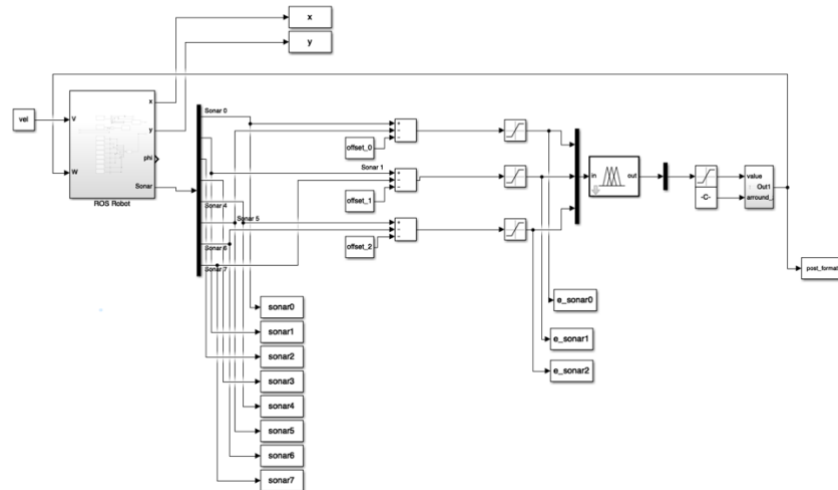
Ilustración 18 Configuración del controlador Sugeno

Una vez entrenado el controlador se exporta y se utiliza como si fuera la parte anterior.

En los controladores Sugeno pueden llegar valores fuera del rango en el que los esperan. Esto se debe a que el rango de los conjuntos será el de los datos de entrenamiento y en ellos puede que haya situaciones que no estén cubiertas pero que al ejecutar el controlador las descubra. Para solucionar esto, se ha decidido saturar la entrada del controlador para que este entre los rangos que el controlador conoce.

Como resultado hemos obtenido un controlador que imita la que ya teníamos. Las decisiones tomadas en esta parte respecto de la forma de los conjuntos de entrada o el número de funciones de pertenencia para cada entrada no tenían repercusión visible en el resultado obtenido. Con cualquier cantidad o tipo de funciones el controlador obtenido era capaz de imitar los datos que le habíamos proporcionado.

El diseño del slx se le añade como se ha comentado los 3 saturadores a la entrada del controlador.

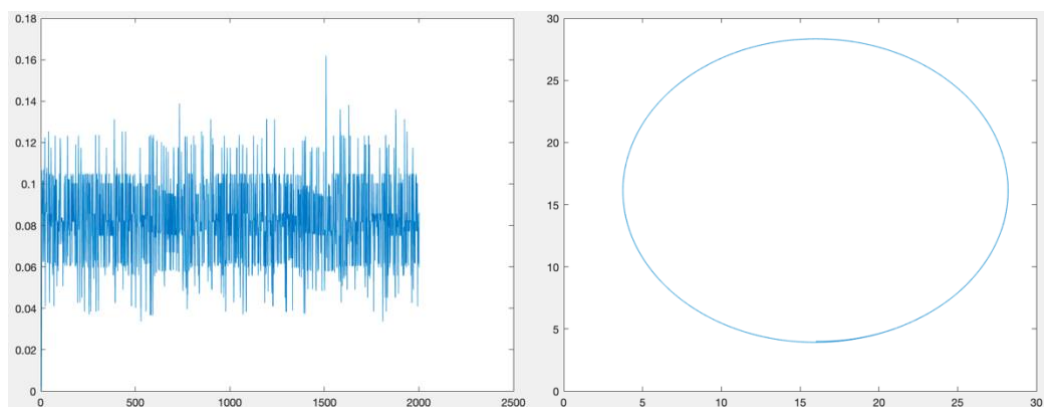


*Ilustración 19 Configuración del slx*

Como se puede apreciar en las imágenes, se aprecia como el controlador es capaz de seguir la trayectoria

## Resultados

Al igual que ocurría con el controlador Mamdani, el controlador Sugeno que lo imita produce una velocidad angular muy estable entorno a 0.08 rad/s sin en ningún momento llegar a producir velocidades negativas.



*Gráficas con resultados*

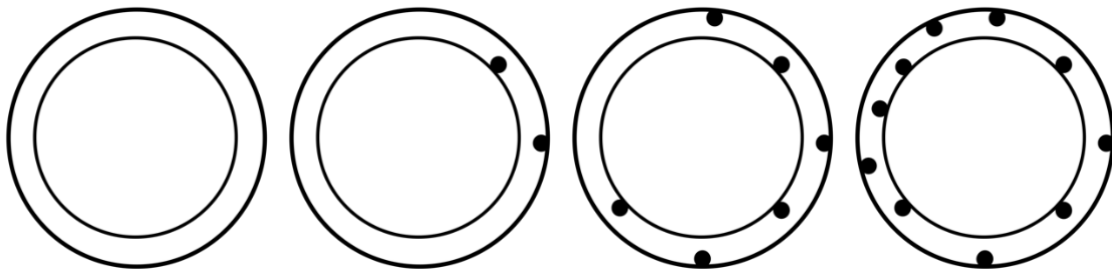
## Con obstáculos

En este caso utilizamos en controlador Mamdani capaz de recorrer el circuito con obstáculos para recoger los datos que luego utilizaríamos para entrenar el controlador Sugeno.

```
e_sonar0_ = e_sonar0.signals.values;  
e_sonar1_ = e_sonar1.signals.values;  
e_sonar2_ = e_sonar2.signals.values;  
e_sonar_obs0_ = e_sonar_obs0.signals.values;  
e_sonar_obs1_ = e_sonar_obs1.signals.values;  
out_ = post_format.signals.values;  
  
e_sonar0_(isinf(e_sonar0_)) = 5.0;  
e_sonar1_(isinf(e_sonar1_)) = 5.0;  
e_sonar2_(isinf(e_sonar2_)) = 5.0;  
e_sonar_obs0_(isinf(e_sonar_obs0_)) = 5.0;  
e_sonar_obs1_(isinf(e_sonar_obs1_)) = 5.0;  
  
all_data = [e_sonar0_ e_sonar1_ e_sonar2_ e_sonar_obs0_ e_sonar_obs1_ out_];  
all_data = double(all_data);  
  
save all_dataTrain.dat all_data -ascii
```

*Ilustración 20 Guardado datos de entrenamiento*

Debido a que hemos decidido no utilizar datos de validación ni de test ya que nos estaban proporcionando peores resultados que al utilizar solo datos de entrenamiento decidimos crear algún mecanismo para detectar si nuestro controlador estaba aprendiéndose los datos siendo capaz de generalizar o no. Para hacerlo decidimos crear un nuevo mapa con más obstáculos que los anteriores. Para entrenar recogeremos datos solo de los dos mapas con obstáculos proporcionados. Para validar el controlador utilizaremos el mapa sin obstáculos y el mapa creado por nosotros.



*Mapas para realizar pruebas*

Las funciones de pertenencia elegidas en este caso han sido del tipo gauss2mf pues son derivables en todo su dominio y se parecen lo máximo posible a los trapecios del controlador Mamdani que estamos tratando de imitar. Hemos utilizado dos funciones de pertenencia para definir cada conjunto borroso de entrada pues es la misma cantidad que teníamos en Mamdani. Hemos probado con otros tipos de funciones y otras cantidades de ellas sin obtener tan buenos resultados como con esta combinación.



**INPUT**

Number of MFs: 2 2 2 2 2

To assign a different number of MFs to each input, use spaces to separate these numbers.

MF Type:

- trimf
- trapmf
- gbellmf
- gaussmf
- gauss2mf
- pimf
- dsgimf
- psigmf

**OUTPUT**

MF Type: constant, linear

OK Cancel

Ilustración 21 Configuración del fis

Para entrenar este controlador creamos un fis mediante grid partition, y lo entrenamos con 10 épocas únicamente con los datos guardados.

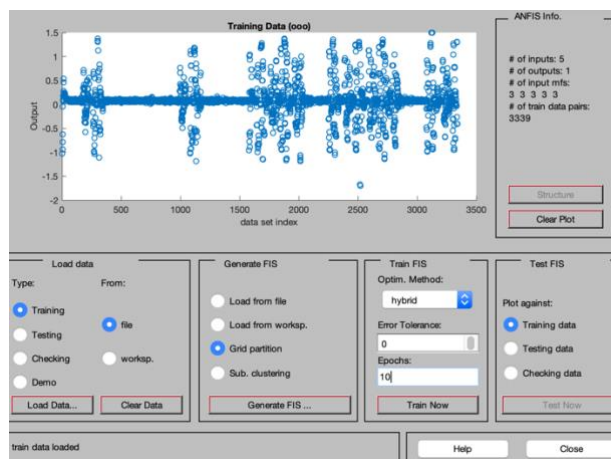


Ilustración 22 Configuración del controlador Sugeno

Con este controlador tuvimos otro problema adicional. Al recoger datos con el controlador Mamdani algunos puntos del dominio posible de los datos de entrada no los llegábamos a explorar. Por ejemplo, nunca estábamos a punto de chocarnos ni demasiado cerca de las paredes. Por ello el dominio de los conjuntos borrosos del controlador Sugeno es más reducido que el que teníamos en Mamdani.

Al igual que en el apartado 1 que se satura la entrada del controlador para evitar que a este lleguen datos fuera del rango en el que los espera.

En el diseño de este slx, es prácticamente igual que el anterior, pero añadiendo los sonares utilizados en el apartado 1.2, es decir, el de Mandami con obstáculos (los sonares utilizados para esquivar los obstáculos).

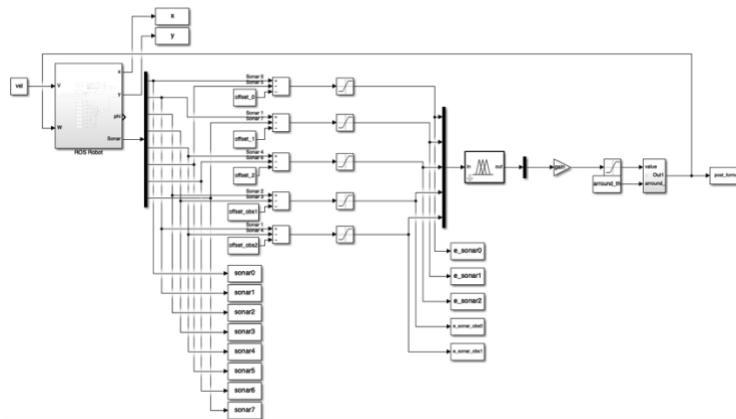


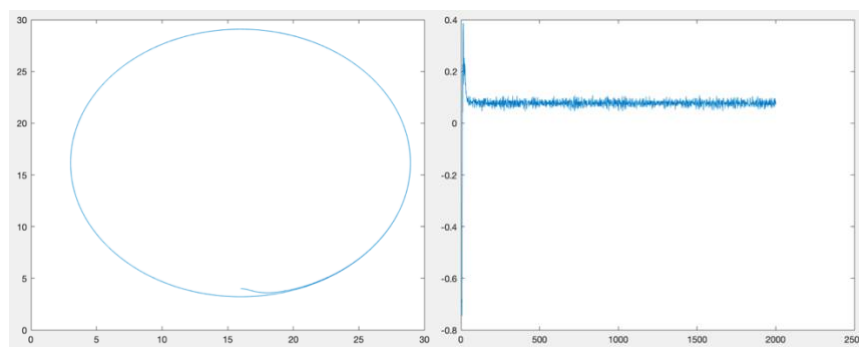
Ilustración 23 Configuración del slx

## Resultados

El controlador es capaz de realizar tanto el circuito sin obstáculos como los dos que se nos proporcionaron con obstáculos y sobre los que hemos tomado datos para entrenar el controlador.

### Mapa sin obstáculos

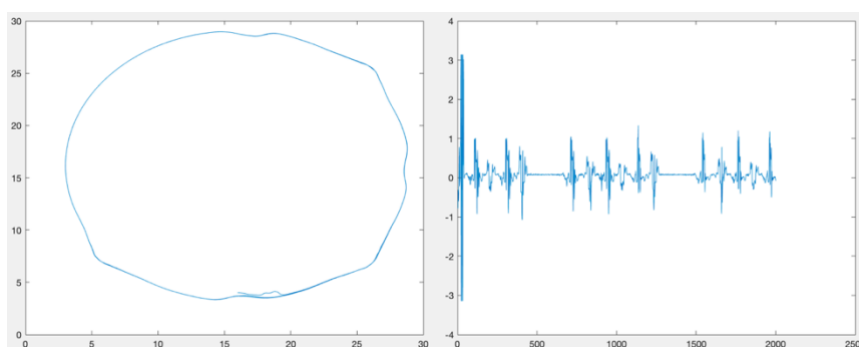
La velocidad angular no tiene cambios bruscos excepto al principio cuando se centra en la pista.



Gráficas con resultados

### Mapa con obstáculos 2

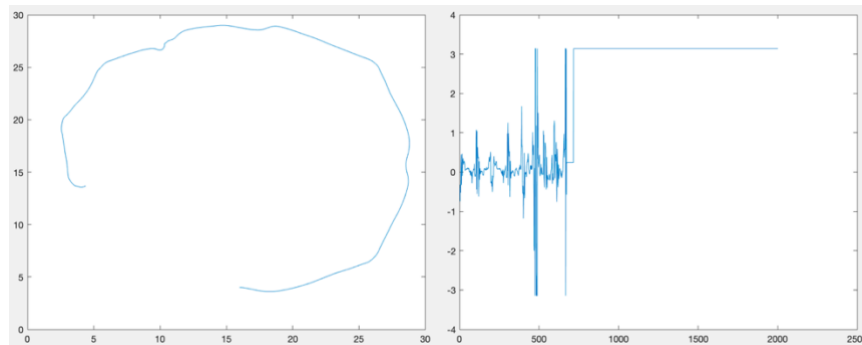
El mayor cambio se produce cuando se centra en la pista pues estos datos son los más distintos del resto ya que posteriormente cuando esquiva los obstáculos pasa más cerca de ellos que cuando se inicia que está en el punto medio entre un obstáculo y la pared.



Gráficas con resultados

No obstante, cuando probamos el controlador sobre el mapa creado por nosotros el controlador no logra completarlo como sí lograba hacer Mamdani. Esto se debe a que el último obstáculo de los que hemos añadido sobresale en exceso de la pared exterior en la cual los obstáculos son mucho más difíciles de esquivar.

El resto de los obstáculos que hemos añadido nuevos sí fueron esquivados correctamente.



*Gráficas con resultados*

Podemos ver que el robot acaba chocándose debido a que al esquivar el obstáculo gira demasiado ya que este sobresale demasiado, de modo que acaba mirando de frente a la pared lo cual es una situación nueva para la que no está entrenado. La acción que realiza es seguir recto y por ello se acaba chocando.

### Parte 3. Controlador para la velocidad lineal y la angular.

Al inicio de la memoria hemos dicho que no íbamos a controlar la velocidad lineal si no que la íbamos a dejar fija a la máxima posible con la intención de completar los circuitos en el menor tiempo posible. No obstante, aunque esto parecía razonable hemos decidido comprobarlo. Para hacerlo hemos modificado los controladores Mamdani añadiendo control sobre la velocidad lineal. En el caso de los controladores Sugeno hemos creado otro controlador que se encargue de producir esta salida; de modo que habrá dos controladores Sugeno, uno para la velocidad angular y otro para la lineal.

#### Controlador de tipo Mamdani circuito sin obstáculos

Para este tipo de controlador, se ha utilizado el explicado en la parte 1 y se ha añadido un nuevo conjunto borroso a la salida, llamado **V**, que permite controlar la velocidad lineal del robot.

También se han añadido tres nuevas funciones de pertenencia en el primer conjunto borroso de la entrada llamado **e\_sonar0** (la diferencia de las medidas de los sonares 0 y 5). Que permite saber si el robot se encuentra centrado (para aumentar la velocidad) o si se encuentra desplazado hacia algún lado y por lo tanto mal posicionado (para disminuir la velocidad).

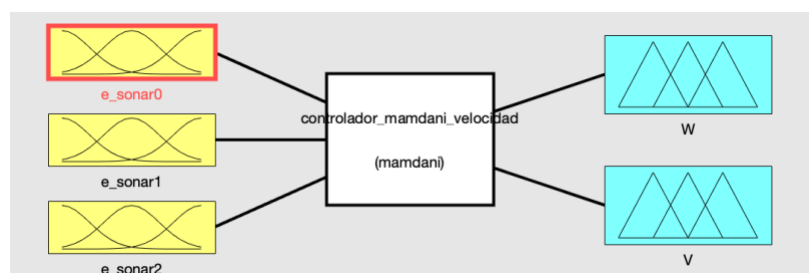


Ilustración 24 Controlador tipo Mamdani sin obstáculos con velocidad lineal



Ilustración 25 Nuevas funciones de pertenencia en el primer conjunto borroso

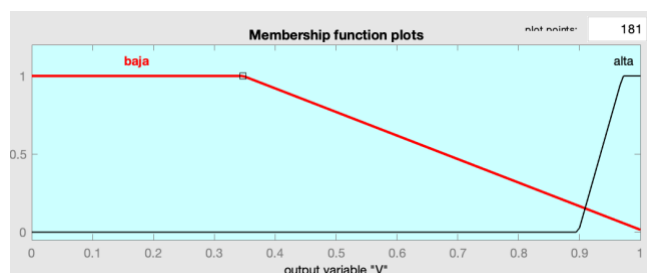
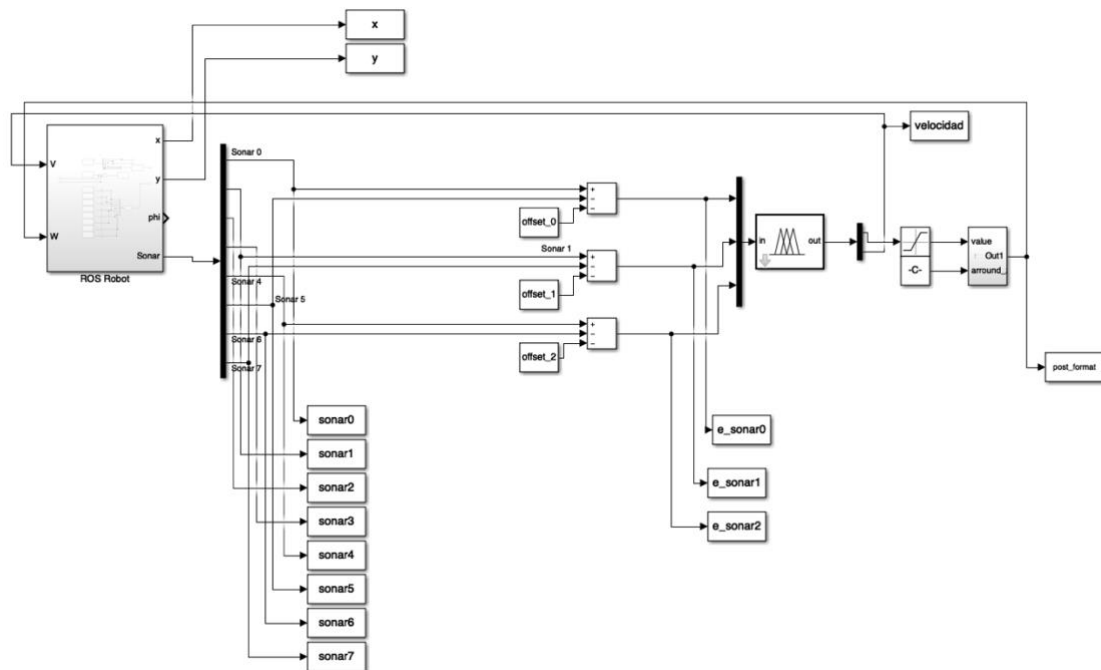


Ilustración 26 Conjunto borroso velocidad lineal

En las reglas, se han añadido también 3 nuevas reglas para poder realizar lo comentado anteriormente.

1. If (e\_sonar0 is negativo) then (W is negativo) (1)
2. If (e\_sonar0 is positivo) then (W is positivo) (1)
3. If (e\_sonar1 is negativo) then (W is negativo) (1)
4. If (e\_sonar1 is positivo) then (W is positivo) (1)
5. If (e\_sonar2 is negativo) then (W is positivo) (1)
6. If (e\_sonar2 is positivo) then (W is negativo) (1)
7. If (e\_sonar0 is cero) then (V is alta) (1)
8. If (e\_sonar0 is muy\_negativo) then (V is baja) (1)
9. If (e\_sonar0 is muy\_positivo) then (V is baja) (1)

*Ilustración 27 Reglas controlador Mamdani para controlar la velocidad lineal*



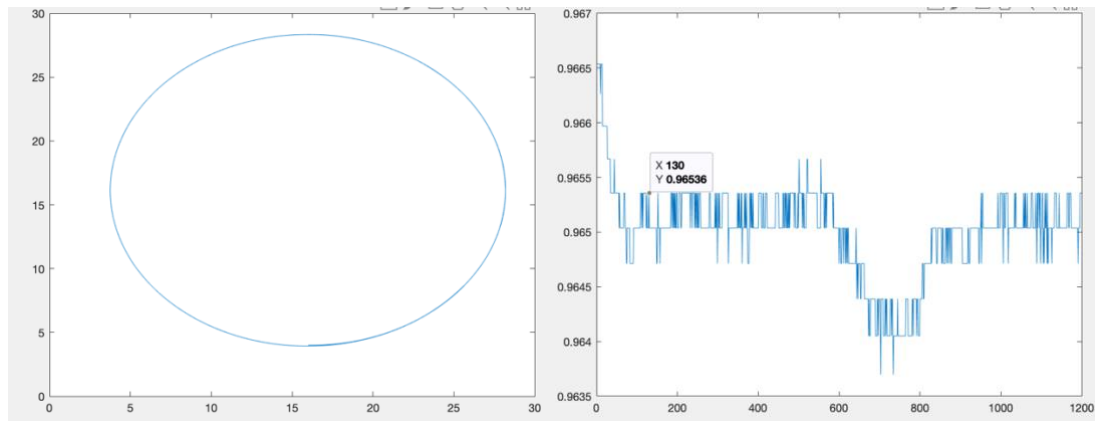
*Ilustración 28 Modificación del slx*

Una vez realizado estos cambios y añadiendo en el slx la velocidad de la salida del controlador, se procede a comentar los resultados obtenidos.

## Resultados

Se puede apreciar como es capaz de completar el circuito sin chocarse. Además, si observamos la velocidad lineal que lleva el robot en todo momento es muy próxima a 1, esto es porque el robot siempre va centrado y por lo tanto al no estar desviado siempre va a la máxima velocidad que el controlador le dice.

La captura de la izquierda es la trayectoria del robot y la de la derecha es la velocidad lineal del robot en todo el recorrido.



Gráficas con resultados

### Controlador de tipo Mamdani circuito con obstáculos

Para este apartado, al igual que en el anterior, se ha utilizado el controlador realizado en el apartado 1 con obstáculos y se ha modificado para realizar que el controlador maneje también la velocidad lineal al igual que la angular.

Para ello se ha creado un nuevo conjunto borroso a salida llamado **V**. También se han añadido 1 función de pertenencia a los sensores utilizados para detectar si hay obstáculos, para indicar si se encuentra centrado, es decir no hay ningún obstáculo y puede aumentar la velocidad, o si por el contrario hay algún obstáculo en algún lado y por el contrario tiene que disminuir la velocidad.

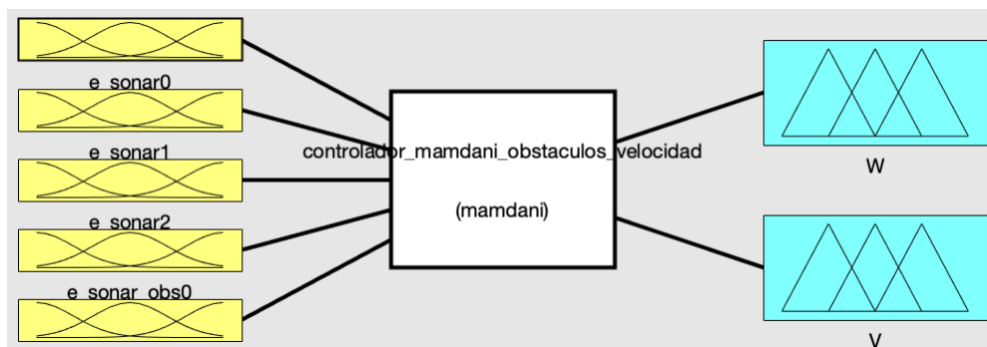


Ilustración 29 Controlador tipo Mamdani con obstáculos con velocidad lineal

El conjunto borroso de salida para la velocidad **V**, es igual que en el controlador anterior.

La función de pertenencia añadido en el cuarto conjunto de entrada (la diferencia de los sensores 2 y 3), se utiliza para detectar si el robot esta esquivando algún obstáculo y tiene que reducir la velocidad o si por el contrario no esta esquivando ningún obstáculo y puede aumentar la velocidad.

Se han añadido 3 nuevas reglas que permiten controlar la velocidad del robot de la forma que se ha comentado anteriormente.

2. If (e\_sonar0 is positivo) then (W is positivo) (1)
3. If (e\_sonar1 is negativo) then (W is negativo) (1)
4. If (e\_sonar1 is positivo) then (W is positivo) (1)
5. If (e\_sonar2 is negativo) then (W is positivo) (1)
6. If (e\_sonar2 is positivo) then (W is negativo) (1)
7. If (e\_sonar\_obs0 is positivo) then (W is muy\_positivo) (1)
8. If (e\_sonar\_obs0 is negativo) then (W is muy\_negativo) (1)
9. If (e\_sonar\_obs1 is positivo) then (W is muy\_positivo) (1)
10. If (e\_sonar\_obs1 is negativo) then (W is muy\_negativo) (1)
11. If (e\_sonar\_obs0 is cero) then (V is alta) (1)
12. If (e\_sonar\_obs0 is positivo) then (V is baja) (1)
13. If (e\_sonar\_obs0 is negativo) then (V is baja) (1)

Ilustración 30 Reglas controlador Mamdani para controlar la velocidad lineal

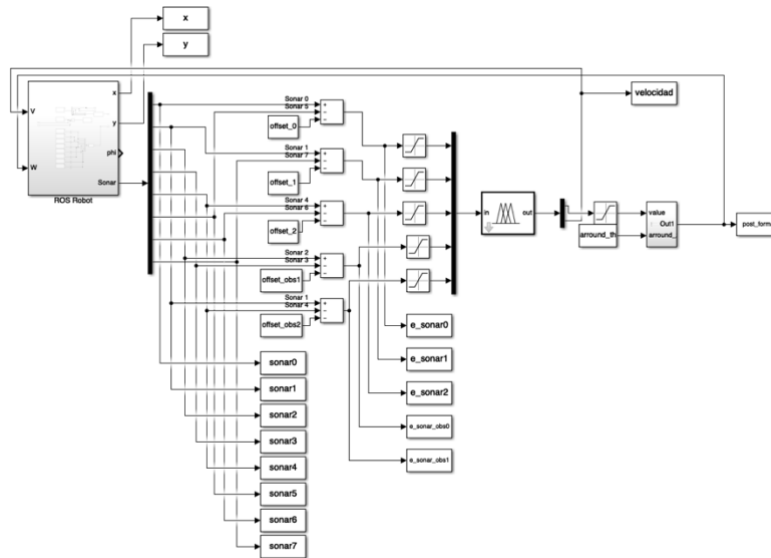
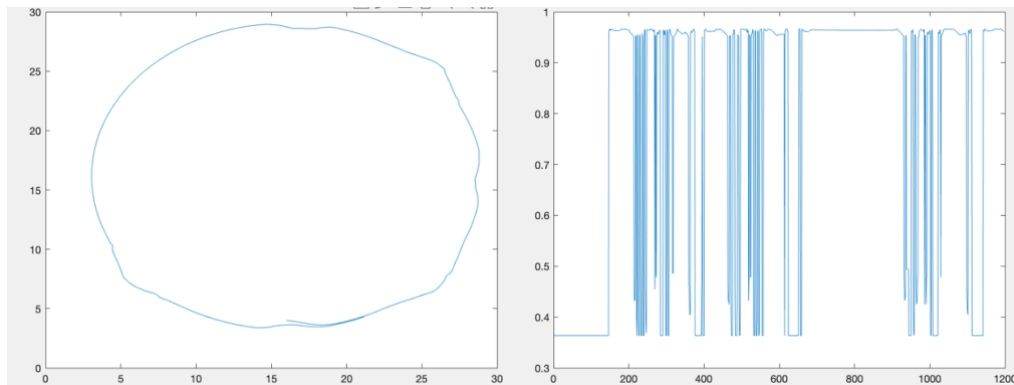


Ilustración 31 Diseño final del slx

## Resultados

Al utilizar este controlador se puede apreciar como la velocidad lineal se hace más pequeña cuando el robot se aproxima a los obstáculos y más grande cuando estos no están presentes.

También se puede ver como el robot es capaz de realizar el circuito sin chocarse con ningún obstáculo, la única diferencia de utilizar este frente a no controlar la velocidad lineal es que el robot tarda más en dar la vuelta completa utilizando este controlador que con el explicado en el apartado 1.



Gráficas con resultados

En la gráfica de la velocidad se puede apreciar como el robot es capaz de reducirla cuando se acerca a algún obstáculo y como cuando se aleja es capaz de aumentarla.

### Controlador de tipo Sugeno circuito sin obstáculos

Para este problema, se han tenido que realizar dos controladores, uno para controlar la velocidad angular (que es el mismo que en el apartado 2) y otro nuevo para controlar la velocidad lineal, ya que utilizando Sugeno no es posible entrenar un único controlador para controlar las dos variables.

Para la realización y entrenamiento de este controlador se han seguido los mismos pasos explicados en el apartado 2.

Primero se han obtenido los datos de entrenamiento del controlador de Mamdani controlando la velocidad con el script mostrado en la **ilustración 32**.

```
e_sonar0_ = e_sonar0.signals.values;
e_sonar1_ = e_sonar1.signals.values;
e_sonar2_ = e_sonar2.signals.values;
out_ = post_format.signals.values;
velocidad_ = velocidad.signals.values;

e_sonar0_(isinf(e_sonar0_)) = 5.0;
e_sonar1_(isinf(e_sonar1_)) = 5.0;
e_sonar2_(isinf(e_sonar2_)) = 5.0;

train = [e_sonar0_ e_sonar1_ e_sonar2_ velocidad_];
train = double(train);

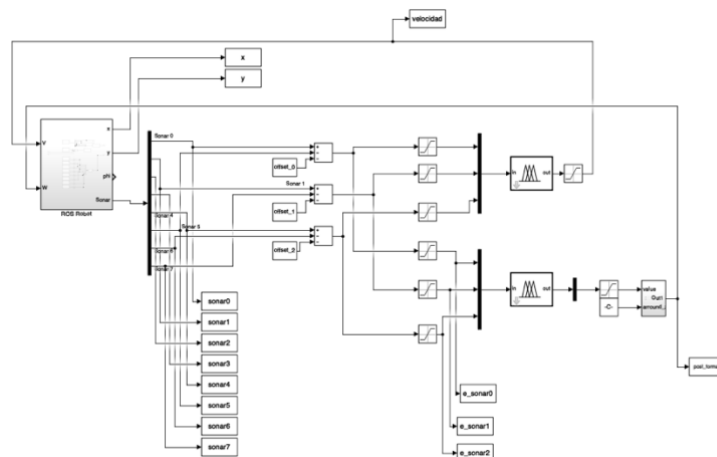
save trainVelocidad.dat train -ascii
```

*Ilustración 32 Código para obtener datos de entrenamiento*

Una vez obtenidos los datos de entrenamiento se entrena de la misma forma que la **ilustración 17 y 18**. Los pasos seguidos están explicados en este documento.

Una vez obtenido el controlador, es necesario saturar las entradas a los rangos que cada entrada tiene (explicado también anteriormente), para evitar que al controlador lleguen datos fuera de los rangos en los que los espera.

Como se puede apreciar en el slx se han creado dos controladores independientes para controlar cada variable del robot por separado.



*Ilustración 33 Diseño del slx*

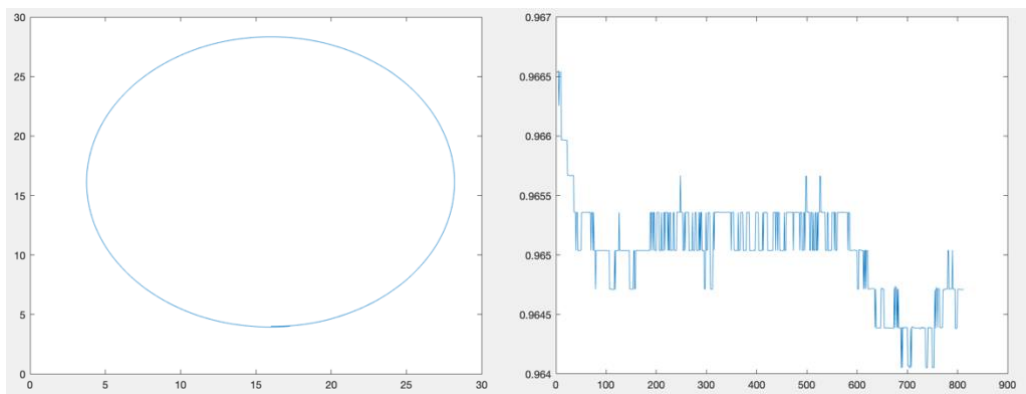


## Resultados

Este controlador ofrece muy buenos resultados (igual de buenos que el controlador Mamdani para recorrer el circuito con obstáculos de la **parte 3**), ya que los datos de entrenamiento son cogidos de este controlador.

Se observa como los valores de la velocidad se mantienen muy próximos a 1, del mismo modo que el controlador de Mamdani, esto es así porque el robot en este circuito siempre va centrado y no se desvía del centro del mapa.

La imagen derecha representa la trayectoria seguida por el robot con este controlador y la imagen de la derecha representa la velocidad que el robot llevaba en cada momento de la vuelta al circuito.



Gráficas con resultados

## Controlador de tipo Sugeno circuito con obstáculos

Para este controlador, al igual que el anterior, ha sido necesario crear 2 controladores para controlar las dos velocidades del robot.

Para el controlador que maneja la velocidad angular se ha utilizado el creado en la **parte 2**, por lo que solo ha sido necesario crear un controlador para la velocidad lineal.

Para obtener los datos de entrenamiento de la velocidad lineal se ha utilizado el controlador de Mamdani en el circuito con obstáculos de la **parte 3** mediante el script de la **ilustración 34**.

```
e_sonar0_ = e_sonar0.signals.values;
e_sonar1_ = e_sonar1.signals.values;
e_sonar2_ = e_sonar2.signals.values;
e_sonar_obs0_ = e_sonar_obs0.signals.values;
e_sonar_obs1_ = e_sonar_obs1.signals.values;
out_ = post_format.signals.values;
velocidad_ = velocidad.signals.values;

e_sonar0_(isinf(e_sonar0_)) = 5.0;
e_sonar1_(isinf(e_sonar1_)) = 5.0;
e_sonar2_(isinf(e_sonar2_)) = 5.0;
e_sonar_obs0_(isinf(e_sonar_obs0_)) = 5.0;
e_sonar_obs1_(isinf(e_sonar_obs1_)) = 5.0;

all_data = [e_sonar0_ e_sonar1_ e_sonar2_ e_sonar_obs0_ e_sonar_obs1_ velocidad_];
all_data = double(all_data);

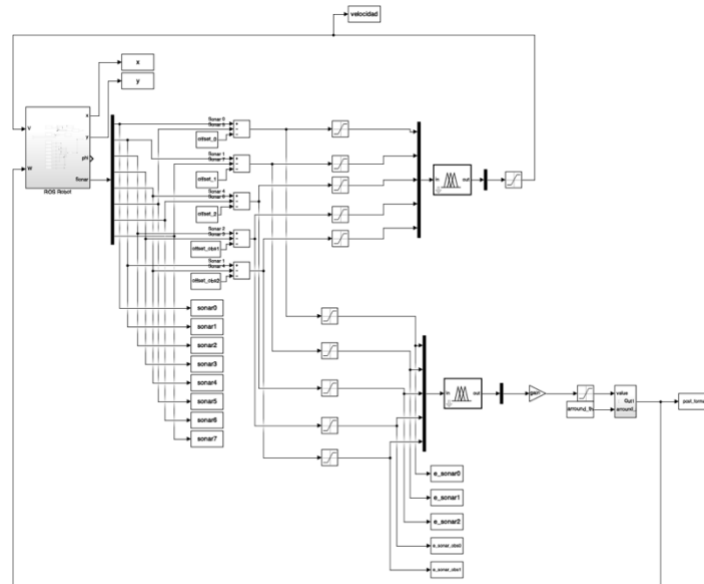
save all_dataTrain2.dat all_data -ascii
```

Ilustración 34 Código para obtener los datos de entrenamiento

La toma de datos se ha realizado igual que se ha entrenado el controlador de Sugeno con obstáculos de la **parte 2** (leyendo los valores de los dos mapas proporcionados para la realización de la práctica y juntando los datos en un solo archivo).

Los pasos seguidos para entrenar este controlador han sido explicados en la **parte 2, Sugeno con obstáculos** de este documento, concretamente en la **ilustración 21 y 22**. Una vez obtenido el controlador para la velocidad lineal, al igual que en el anterior, se saturan las entradas a los valores del rango para que no pueda llegar valores fuera de ese rango.

Por último, se muestra el controlador en el slx para poder controlar la velocidad.

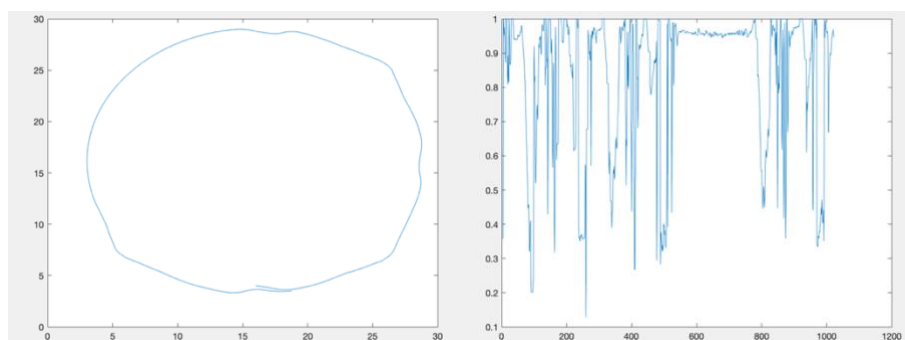


*Ilustración 35 Diseño final del slx*

## Resultados

Este controlador ofrece resultados muy buenos al igual que el Mamdani con obstáculos controlando la velocidad lineal, es decir, es capaz de esquivar todos los obstáculos y realizar la vuelta completa. El único problema que se ha comentado anteriormente es que al reducir la velocidad lineal al esquivar un obstáculo aumenta el tiempo en el que se completa el circuito, aunque se gane otros aspectos, el robot tiene más tiempo para aplicar una velocidad angular tal que se esquiven los obstáculos.

La imagen de la izquierda representa la trayectoria seguida por el robot al esquivar los obstáculos, la de la derecha representa la velocidad del robot en la vuelta al circuito. Como se puede ver la velocidad lineal cambia en función si esta esquivando un obstáculo o si no lo esta esquivando, llegando incluso a ponerse a 1.



*Gráficas con resultados*

## Conclusiones

Una vez realizado todos los controladores controlando únicamente la velocidad angular manteniendo al máximo la velocidad lineal y también controlando la velocidad lineal, se ha llegado a la conclusión de que si mantenemos la velocidad lineal siempre al máximo el tiempo que tarda en realizar una vuelta al circuito se reduce y además es capaz de esquivar todos los obstáculos.

En cambio, si controlamos la velocidad lineal el tiempo que tarda en realizar el circuito aumenta (sobre todo en los mapas con obstáculos), aunque es capaz de asegurar que no se va a chocar ya que al esquivar un obstáculo reduce la velocidad tanto como sea necesario.

Para comprobar esto se han ejecutado todos los controladores en todos los mapas y se ha realizado una tabla de tiempos, para decir con seguridad cuál es mejor y cuánto de mejor es.

### Tabla de tiempos:

Los tiempos han sido capturados manualmente por lo que su precisión es relativa.

|                            | <b>Mamdani</b> sin obstáculos sin controlar velocidad lineal | <b>Mamdani</b> sin obstáculos controlando velocidad lineal | <b>Sugeno</b> sin obstáculos sin controlar velocidad lineal | <b>Sugeno</b> sin obstáculos controlando velocidad lineal |
|----------------------------|--|--|---|---|
| <b>Mapa sin obstáculos</b> | 1 minuto 16 segundos   | 1 minuto 19 segundos                                       | 1 minuto 16 segundos  | 1 minuto 20 segundos                                      |

Se esperaba que la diferencia entre controlar y no controlar la velocidad lineal fuera mayor, no obstante, se descubre que esta es mínima. También se observa que los controladores Mamdani y Sugeno tardan aproximadamente lo mismo. Esto se debe a que los controladores Sugeno tratan de imitar a los controladores Mamdani.

|   | <b>Mamdani</b> con obstáculos sin controlar velocidad lineal | <b>Mamdani</b> con obstáculos controlando velocidad lineal  | <b>Sugeno</b> con obstáculos sin controlar velocidad lineal   | <b>Sugeno</b> con obstáculos controlando velocidad lineal   |
|---|--|---|---|---|
| <b>Mapa con obstáculos simples</b>          | 1 minuto 21 segundos   | 1 minuto 29 segundos  | 1 minuto 22 segundos  | 1 minuto 32 segundos  |
| <b>Mapa con obstáculos complejo</b>         | 1 minuto 22 segundos   | 1 minuto 37 segundos  | 1 minuto 23 segundos  | 1 minuto 40 segundos  |
| <b>Mapa con obstáculos complejo nuestro</b> | 1 minuto 22 segundos   | No consiguió terminar el recorrido, se chocó en un obstáculo del final. Aunque llevaba 10 segundos más que Mamdani con velocidad lineal constante | No consiguió terminar el recorrido, se chocó en un obstáculo del final. Aunque llevaba el mismo tiempo aproximadamente que Mamdani con velocidad lineal constante | No consiguió terminar el recorrido, se chocó en un obstáculo del final. Aunque llevaba 12 segundos más que Mamdani con velocidad lineal constante |

Como se aprecia en la tabla, el controlador de tipo Mamdani que deja la velocidad lineal constante da muy buenos resultados y además muy parecidos en todos los mapas independientemente del número de objetos que haya y de como se encuentren situados. Luego el controlador de tipo Sugeno sin controlar la velocidad lineal ofrece también resultados muy parecidos que el controlador de tipo Mamdani sin controlar la velocidad lineal y es capaz de recorrer muy bien los mapas propuestos en la práctica, sin embargo en el mapa creado por nosotros no fue capaz de superarlo quedando parado en un obstáculo del final.

En cambio, los dos controladores que también controlan la velocidad lineal y la velocidad angular han dado los peores resultados en todos los mapas, esto es así porque al pasar cerca de un obstáculo reducen la velocidad y al reducirla tardan en recorrer el mapa bastante más. Cabe destacar que nos parece curioso como el controlador de tipo Mamdani controlando la velocidad lineal no haya sido capaz de superar el mapa creado por nosotros y se haya chocado, ya que en todas las pruebas reduce la velocidad lo suficiente como para esquivar los obstáculos sin ningún problema, al igual que el mapa de tipo Sugeno controlando la velocidad lineal también.

En conclusión si tuviéramos que elegir un controlador para recorrer cualquier tipo de mapa elegiríamos el controlador de tipo Mamdani sin controlar la velocidad lineal, no solo porque de los mejores tiempos a la hora de recorrer el mapa, sino también porque ha sido el único controlador capaz de superar el último mapa sin chocarse en el mismo punto donde los otros 3 controladores se han chocado.