VEHICULO AUTONOMO SEGUIDOR DE LINEA CONTROLADO CON LOGICA BORROSA

Edwin Ambiorix Marte Zorrilla
Asociación de Robótica Dominicana
Santiago, Republica Dominicana
emarte@aiolosrd.com

Abstract. *El propósito de este documento es el de desarrollar un vehículo autónomo utilizando lógica borrosa para controlarlo. El vehículo que utilizamos es un seguidor de línea del tipo Robot 3pi, que es un kit comercial. La tarea principal del robot es moverse de tal forma que pueda mantener su trayectoria sobre una línea negra trazada en un panel de PVC de color blanco. El vehículo está formado por 2 motores de tracción diferencial con una rueda de soporte en la parte trasera. El sistema de control es gobernado por la salida y estado de cinco sensores infrarojos que se encuentran en el frente del robot y por medio de los cuales podemos determinar la posición del robot con respecto a la línea negra. Se implementa Logica de Control Borroso en un micorcontrolador ATmega328 programandolo en el IDE de arduino y utilizando las librerías EFLL.*

Palabras Clave: autonomous vehicle, line-follower, fuzzy control, microcontroller.

1 Introducción

En la actualizad podemos observar una tendencia en la construccion de vehiculos. Se les estan incluyendo componentes electronicos y equipos cada vez mas complicados en el sistema electrico. Dispositivos como lo son radios y componentes de CD o DVD son ahora equipos standard. Nuevos accesorios son introducidos con frecuencia como lo son los GPS, cámaras y otros tipos de sensores que facilitan la navegación de carreteras. Algunos vehiculos inclusive son capaces de actualizar sus mapas basados en datos de sensores de entradas. Aún asi son introducidos nuevos equipos para facilitar la conducción cada vaez con mas frecuencia, incluyendo en los años recientes como lo ha sido con el uso de Tablets inteligentes. Algunos utilizan sensores especiales y sistemas para parqueo automatico. El próximo paso deberia de ser el conductor automático o autónomo similar al de los aviones. Ademas de esto podemos destacar que en el 2010 Google anuncio un proyecto de un automovil que se conduce de forma autónoma haciendo que el conducir sea más seguro, más eficiente y mas satisfactorio. Para la presentación ellos muestran un video y hacen énfasis en haber completado 200,00 millas de prueba conducidas por una computadora [1] y [3].

Son muchas las grandes compañías y organizaciones que se han unido a la investigación y han desarrollado prototipos funcionales, incluyendo a Google, Continental Automotive Systems, Bosch, Nissan, Toyota, Audi y Oxford University. En junio del 2011, el estado de Nevada fue la primera jurisdicción de los estados unidos en aprobar una ley acerca de vehículos de operación autónoma [11]. La primera licencia de su tipo fue expedida en Mayo del 2012.

2 Vehículo Utilizado

2.1 Descripción General

La mayoría de los sistemas de vehículos autónomos que son desarrollados utilizan sistemas integrados de video cámaras o a veces sistemas de radar como fuente de información. Nosotros proponemos la utilización de un kit de robótica comercialmente disponible como lo es el 3pi de Pololu [9]. Este robot se caracteriza por ser un vehículo diferencial que se guía por un sistema de sensores de reflexión Infra rojo que le permiten seguir trayectorias de líneas de color blanco o negro en un piso de un color contrastante. La idea de la implementación de este vehículo es simular un vehículo autónomo para transporte de mercancía en un almacén o en un supermercado por ejemplo. Para la aplicación práctica hemos utilizado un panel blanco de PVC con una trayectoria de Cinta adhesiva eléctrica negra. Este tipo de vehículo o robot es conocido como un seguidor de línea.

Figura 1: Kit Robot 3pi de Pololu.

El kit de Robot 3pi posee 2 motores DC para tracción diferencial y 5 sensores ubicados en la parte frontal para detección de línea. Este kit viene

equipado con un LCD de 8x2 para visualización de mensajes y depuración. Posee un par de leds de diferente colores para indicación general además en caso de ser necesario se puede habilitar un canal de conversión Analógico Digital extra para colocar dispositivos externos, un sensor de distancia por ejemplo. El procesador que gobierna el sistema es un ATmega328p del fabricante ATMEL lo que lo hace compatible con la plataforma Arduino. Para complementar también se tiene acceso a 3 pulsadores que pueden ser utilizados para control de programa o depuración.

El procesador ATmega328 es un procesador capaz de ejecutar instrucciones a una velocidad de 20MHZ y viene incorporado con 32KB de memoria de programa, 2 KB RAM, y 1 KB memoria EEPROM.

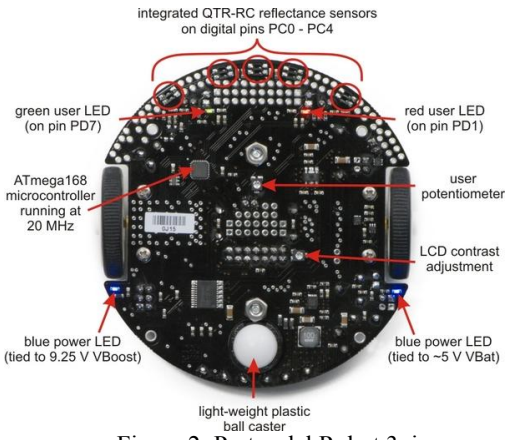


Figura 2: Partes del Robot 3pi.

Para programarlo es necesario utilizar un programador externo, en nuestro caso utilizamos el USB AVR programmer también del fabricante Pololu por su simplicidad y costo.


Figura 3: Programador AVR USB de Pololu.

2.2 Estructura de Hardware

El robot 3pi es un kit compacto de 9.5 cm de Diámetro capaz de alcanzar velocidades de 100cm/s. Es operado por 4 baterías del tipo AAA que suplen el voltaje suficiente para alimentar a los motores y el procesador.

2.2.1 Fuente de Alimentación y Baterías

El sistema de alimentación es provisto por 4 baterías del tipo AAA. El voltaje de estas pueden variar en cualquier momento entre 3.5 y 5.5 V (inclusive hasta 6V si son utilizadas baterías alcalinas). Esto hace imposible no solo para el 3pi sino para cualquier robot o sistema el simplemente regular el voltaje digamos a 5V reduciendo o aumentando por la variación que pueden mostrar las baterías. La implementación del 3pi utiliza 2 fuentes, una conmutada elevadora que lleva el voltaje de la batería a 9.25V (Vboost) y luego una lineal que regula el Vboost a 5V. El Vboost es aplicado a los motores y los 5V al microcontrolador. Este tipo de implementación asegura que el 3pi se comporte de igual manera y a la misma velocidad para todos los rangos de estado de la batería.

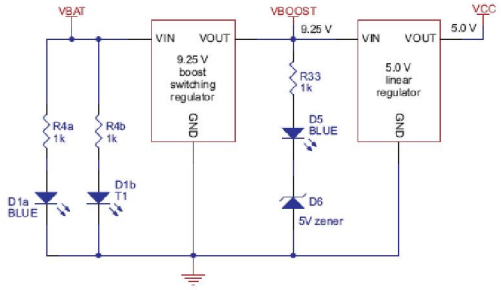


Figura 4: Fuente de Alimentación del Robot 3pi.

2.2.2 Motores y Transmisiones

Como habiamos mencionado el 3pi es un robot de traccion difrencial. Posee 2 motores con transmision de relacion 30:1. Estos motores son capaces de velocidades de 700 rpm y un consumo de 60mA corriendo libre. El robot completo pesa 7 OZ y por sus caracteristicas es capaz de recorrer inclinaciones de hasta 30-40 grados.

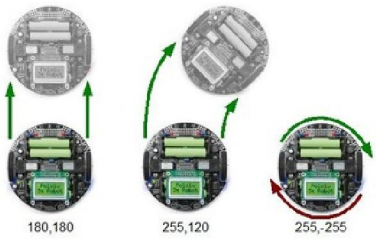


Figura 5: Giros de forma diferencial vs potencia en cada motor del robot 3pi.

Los motores utilizan el Driver de Circuito integrado TB6612FNG. Cada motor utiliza 4 pines del microcontrolador para activarlo en un sentido o el otro o pararlo en el caso de ser necesario. Utilizando las librerías de Pololu se pueden activar los motores con la función set_motors() suministrando valores de 0-255 para cada motor para obtener velocidades de 0 a 100%. La figura 5 muestra como se puede conducir

y hacer girar el 3pi utilizando potencia de velocidades diferentes en cada motor.

2.2.3 Sensores detectores de Línea

El vehículo de nuestra aplicación viene equipado con 5 sensores del Tipo QTR ubicados en la parte frontal del robot y sirven para detectar líneas o superficies contrastante de color. Los 3 sensores centrales están separados a 13 grados equitativamente uno del otro y los laterales están a 70 grados de separación, 35 grados del sensor central. En la figura 6 que se muestra la separación de los sensores.

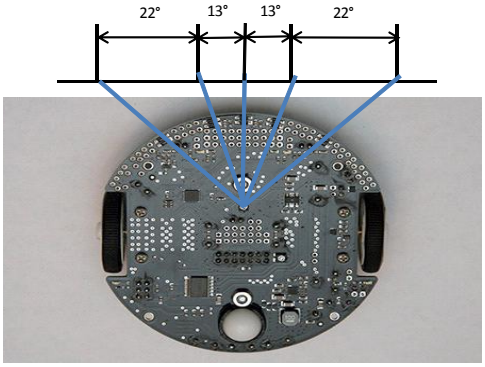


Figura 6: Espaciamiento angulares entre los sensores de piso del robot 3pi.

Los sensores internamente están conformados por un led infra rojo que se proyectan con el piso y la reflexión es leída por un foto transistor el cual proporciona una salida de voltaje proporcional a la reflexión detectada. El esquema de conexión se muestra en la figura 7.

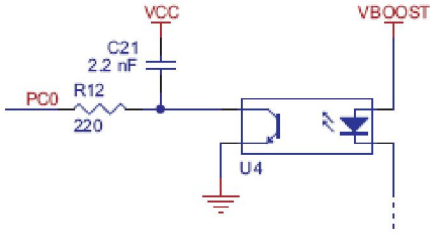


Figura 7: Esquemático de conexión de los sensores de Piso del robot 3pi.

Para la lectura de los sensores podemos utilizar la librería provista por el fabricante para este propósito donde el valor que leemos de los sensores es un valor promediado ponderado de la posición de los sensores con respecto a la línea. Siendo s1 el valor leído por el sensor más a la izquierda y s5 el valor del sensor de más a la derecha. En (1) se muestra como es el procedimiento de cálculo. Un valor de 2000 representa que el robot está bien centrado sobre la línea y 0 ó 4000 que está bien a la derecha o a la izquierda.

$$Cal = \frac{0*s1 + 1000*s2 + 2000*s3 + 3000*s4 + 4000*s5}{s1 + s2 + s3 + s4 + s5} \quad (1)$$

3 El Algoritmo

El seguidor de línea que se plantea esta desarrollado de una forma un tanto no convencional. El control del vehículo estará basado en control borroso. Para poder implementar este proyecto hemos creado un grupo de variables lingüísticas con sus apropiados valores, sets borrosos y funciones de pertenencia, reglas de fuzzyficacion y un método apropiado de
defuzzificacion.

3.1 Variables Lingüísticas

Las variables lingüísticas fueron desarrolladas para el apropiado control del vehículo. Los valores fueron determinados midiendo los valores de los sensores utilizando la librería de Pololu, utilizando la función de calibración y midiendo los valores de posición de acorde a (1).

Los sensores pueden quedar ubicados en diferentes posiciones con respecto a la cinta negra pero en (1) obtendremos un valor oscilante entre 0 y 4000 de acuerdo si el robot está muy a la derecha o muy la izquierda siendo 2000 cuando el robot esta exactamente en el centro.

Tabla 1: Nombres y Rangos de Variables de Conjuntos de Entrada.

Nombre	Valor	Posicion del Robot
PmIZQ	0-1000	Muy a la Izquierda
PIZQ	0-2000	A la Izquierda
Pcentrado	1000-3000	Centrado
PDER	2000-4000	A la Derecha
PmDER	3000-4000	Muy a la Derecha

Las funciones de pertenencias se muestran en la figura 8 para cada conjunto borroso de los valores de los sensores. Los nombres de los conjuntos y sus rangos se listan en la tabla 1.

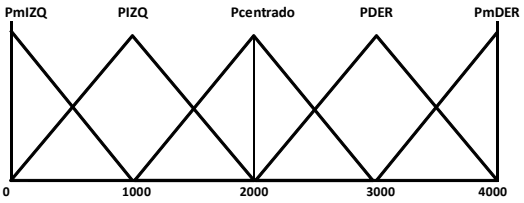


Figura 8: Conjunto Borroso para Posición del Robot con respecto a la línea en el de piso.

La segunda variable lingüística diseñada fue la variable ángulo de salida. Esta variable define el

ángulo que el robot tiene que girar para mantenerse centrado en la línea de acuerdo a la posición actual dada por los sensores.

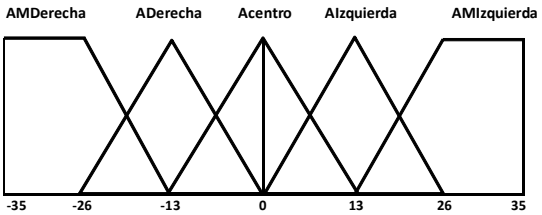


Figura 9: Conjunto Borroso para el Angulo de Salida del Robot con respecto a la línea en el de piso.

Tabla 2: Nombre y Rangos de Variables de Conjuntos de Salida.

Nombre	Valor	Angulo a Girar
AMDerecha	-35 a -13	Gira Mucho a la Derecha
ADerecha	-26 a 0	Gira un poco a la Derecha
Acentro	-13 a 13	Mantente en el Centro
AIzquierda	0 a 26	Gira un Poco a la Izquierda
AMIzquierda	13 a 35	Gira Mucho a la Izquierda

Esta variable ángulo es la que participa en el proceso de desfuzificacion y toma de decisión para la salida, se muestra en la figura 9. Los nombres de Variables y sus rangos son listados en la tabla 2.

3.2 Fuzzificacion o Borrosificación

Acorde a la definición, el proceso de fuzzicación envuelve la asignación de un peso o grado de pertenencia al correspondiente conjunto borroso de acorde al valor de entrada. Esta función es diferente para cada rango y para cada conjunto. Para facilidad de implementación hemos utilizado en su mayoría funciones triangulares que se rigen por las ecuaciones (2), (3) y (4) respectivamente siendo x el valor de entrada.

$$ux=(c-x)/(c-b) \quad (2)$$

$$ux=(x-b)/(c-b) \quad (3)$$

$$ux=1 \quad (4)$$

3.3 Reglas y Razonamiento

Después de seleccionar las variables lingüísticas relevantes y el conjunto borroso, para poder conducir el vehículo, las reglas apropiadas deben de ser creadas y un razonamiento aproximado debe ser ejecutado. Las reglas utilizadas para conducir nuestro vehículo fueron las siguientes:

a-) Si la **Posición** del robot es Centrado **Entonces** la acción del **Angulo** es Mantener en El Centro.

b-) Si la **Posición** del robot es Izquierda **Entonces** la acción del **Angulo** es Gira un Poco a la Derecha

c-) Si la **Posición** del robot es Muy a la Izquierda **Entonces** la acción del **Angulo** es Gira Mucho Poco a la Derecha

d-) Si la **Posición** del robot es Derecha **Entonces** la acción del **Angulo** es Gira un Poco a la Izquierda.

e-) Si la **Posición** del robot es muy a la Derecha **Entonces** la acción del **Angulo** es Gira mucho a la Izquierda.

El orden de las reglas no es importante. En cualquier momento un máximo de dos reglas podrían estar activadas o lo que significa que 2 conjuntos podrían estar aportando peso para actuar en la salida. Hemos aplicado el método de mínimos de mandami [6], como se muestra en (6).

$$\mu A \rightarrow B\ (x,y) = \min\ \{\mu A(x), \mu B(y)\} \quad (6)$$

3.4 Defuzificacion o Desborrosificación

Para obtener un valor de salida que podamos utilizar para controlar el vehículo es necesario realizar la operación de defuzzificación para obtener valores representativos de los ángulos que podamos enviar al sistema de control. Utilizamos el método Centro de Área que se representa en (7).

$$\text{COA} = \frac{\int_{Xmin}^{Xmax} F(x).x\,dx}{\int_{Xmin}^{Xmax} F(x)dx} \quad (7)$$

4. Movimiento del Vehículo

El proceso de defuzzifciacion es usado para obtener el ángulo que debe girar el vehículo. La velocidad de las ruedas del robot definirá el giro angular del mismo. La velocidad de las ruedas dependerá del voltaje que se le apliquen a los motores el cual controlamos por programa de la forma de PWM.

El valor obtenido luego de aplicar la lógica borrosa tiene que ser convertido en velocidades individuales de cada rueda para que el robot pueda girar apropiadamente y mantenerse en la trayectoria esperada. En el transcurso del proyecto realizamos un experimento que nos permitiera obtener la relación apropiada para realizar esta conversión.

Durante el trayecto de un tramo de recorrido cada rueda recorre una distancia. La diferencia entre las dos distancias nos permite encontrar (8).

Donde ΔD es la diferencia de distancia recorrida por ambas ruedas, □ es el ángulo a girar, r

es la distancia entre ruedas el cual encontramos en 8.135 cm para el Robot 3pi.

$$\frac{\Delta D}{2\,\pi r} = \frac{\square}{360} \qquad (8)$$

Tabla 1: Datos medidos del Experimento De Potencia vs Distancia.

Power	Distancia cm
10	0
20	6.7
30	11.4
50	20.7
80	35.4
100	44.4
120	53.7
150	67.3
180	81.4
200	89.6
225	102
255	113.8

Se midieron algunos valores de velocidad que se muestran en la tabla 1. La figura 9 entonces muestra la distancia recorrida por el robot en espacios de 1 segundo en función de la potencia aplicada a los motores (0 a 255). Se obtuvo una función que es aproximado a una línea recta y se estableció una formula matemática.

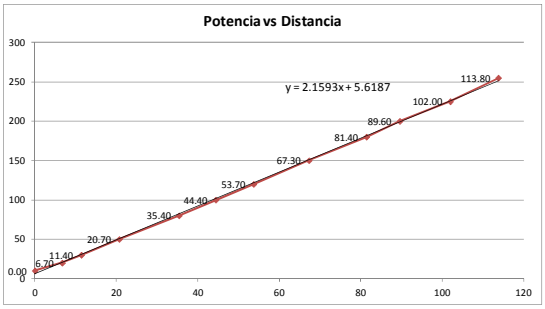


Figura 9: Grafica de Experimento Potencia vs Distancia para 1 segundo.

Tomando en consideración la formula obtenida y que el ángulo de rotación depende la diferencia de velocidad entre las ruedas (ΔV) encontramos la siguiente relación:

$$\Delta V = \frac{\emptyset\,\pi}{180} * 8.135 * 2.1593{+}5.6187 \qquad (9)$$

Y despreciando luego las velocidades mínimas donde el Robot no se mueve nos queda entonces que la diferencia de Velocidad queda expresada por:

$$\Delta V = \emptyset\, 2.5841 \qquad (10)$$

Esta es la potencia de velocidad que enviamos en el robot a los motores dividiéndolo entre los dos de manera que el vehículo pueda rotar al ángulo requerido. En nuestra implementacion de código quedo representado en (11).

power_difference=((output*2.5841)/2); (11)

Figura 10: Pista de Prueba del Sistema.

Para las pruebas del sistema se diseñaron varias pistas de pruebas de diferentes formas en paneles de PVC. En la figura 10 se muestra uno de esos paneles con una pista circular.

5. Conclusiones

El proyecto desarrollado demostró la factibilidad de utilizar el control borroso en un vehículo autónomo seguidor de linear. Se realizaron experimentos para elaborar una base de conocimiento de manera que el controlador borroso se comportara de manera satisfactoria.

El kit de robot 3pi mostró ser una herramienta útil en este tipo de implementación respondiendo a las necesidades aunque para una mayor cantidad de reglas o un controlador borroso más elaborado se tendrá posiblemente que evaluar una arquitectura alternativa por su limitante de memoria a menos que el desarrollador decida desarrollar en lenguaje ensamblador.

Para el desarrollo del controlador borroso en el software, a pesar que desarrollamos nuestro propio código como prueba, también utilizamos las librerías EFLL (embedded fuzzy logic library) que mostraron ser muy confiables aunque hay que tener mucho cuidado al inicializar las variables y los conjuntos borrosos. La forma en que se inicializan estas librerías hace muy fácil el dejar una regla sin activarse si nos equivocamos semánticamente.

Apéndice I: Codigo Principal del Programa implementado

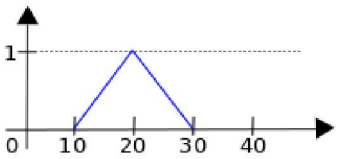
```
void loop()
{
  // Get the position of the line.
  unsigned int position = robot.readLine(sensors, IR_EMITTERS_ON);

 //Calcula diferencia de Velocidad entre ruedaas
   fuzzy->setInput(1, position);
   fuzzy->fuzzify();
   float output = fuzzy->defuzzify(1);
   power_difference=((output*2.5841)/2);

 //Limita Maxima Velocidad
  const int maximum = 60;      if (power_difference > maximum)
    power_difference = maximum;
  if (power_difference < -maximum)
    power_difference = -maximum;

    //Acciona los motores
  if (position < 2000)
    OrangutanMotors::setSpeeds(maximum + power_difference, maximum);
  else
    OrangutanMotors::setSpeeds(maximum, maximum - power_difference);

}
```

Apéndice II: Código para la creación de Conjuntos Borrosos con la Librería EFLL

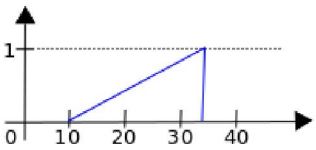
```
// Creacion de Conjunto Difuso para Posicion de Linea
        FuzzyInput* Posicion = new FuzzyInput(1);
        FuzzySet* PmIZQ = new FuzzySet(0, 0, 0, 1000);
         Posicion->addFuzzySet(PmIZQ);
        FuzzySet* PIZQ = new FuzzySet(0, 1000, 1000, 2000);
        Posicion->addFuzzySet(PIZQ);
        FuzzySet* PCentrado = new FuzzySet(1000, 2000, 2000, 3000);
         Posicion->addFuzzySet(PCentrado);
         FuzzySet* PDER = new FuzzySet(2000, 3000, 3000, 4000);
        Posicion->addFuzzySet(PDER);
         FuzzySet* PmDER = new FuzzySet(3000, 4000, 4000, 4000);
         Posicion->addFuzzySet(PmDER);
        fuzzy->addFuzzyInput(Posicion);
// Adicionando La entrada Posicion al Conjunto Borroso

// Criando el conjunto borroso de Salida de Angulo
        FuzzyOutput* Angulo = new FuzzyOutput(1);
        FuzzySet* AMDerecha = new FuzzySet(-35,-35,-26,-13);
        Angulo->addFuzzySet(AMDerecha);
        FuzzySet* ADerecha = new FuzzySet(-26, -13, -13, 0);
         Angulo->addFuzzySet(ADerecha);
        FuzzySet* ACentro = new FuzzySet(-13, 0, 0, 13);
        Angulo->addFuzzySet(ACentro);
        FuzzySet* AIzquierda = new FuzzySet(0, 13, 13, 26);
        Angulo->addFuzzySet(AIzquierda);
        FuzzySet* AMIzquierda = new FuzzySet(13, 26, 35, 35);
        Angulo->addFuzzySet(AMIzquierda);
        fuzzy->addFuzzyOutput(Angulo);
//Adicionando la Salida Angulo al Conjunto Borroso
```

Apéndice III: Configuración de funciones de pertenencias con librería EFLL.

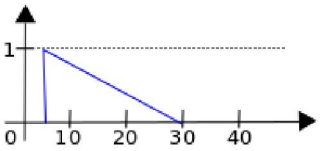
Funciones de Pertenencia Triangulares:



FuzzySet* fs = FuzzySet(10, 20, 20, 30);

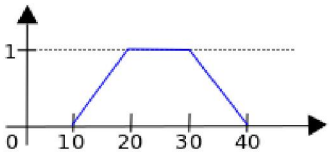


FuzzySet* fs = FuzzySet(10, 33, 33, 33);

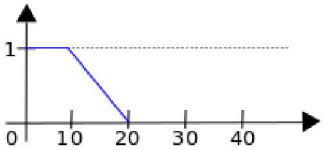


FuzzySet* fs = FuzzySet(5, 5, 5, 30);

Funciones de Pertenencia Trapezoidales:

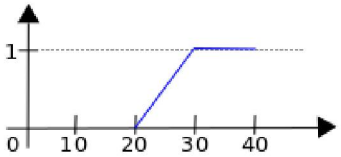


FuzzySet* fs = FuzzySet(10, 20, 30, 40);



FuzzySet* fs = FuzzySet(0, 0, 10, 20);

Cualquier valor por debajo de 10 tendra una pertenencia=1



FuzzySet* fs = FuzzySet(20, 30, 40, 40);

Cualquier valor sobre 30 tendrá una pertenencia=1

Referencias

[1] Blog Oficial de Google, http://goo.gl/SurRq6

[2] Cao, M., "Fuzzy Logic Control for an Automated Guided Vehicle", http://goo.gl/2LXEVI , University of Cincinnati, USA

[3] CNET, http://goo.gl/JpldTX

[4] Diankrov, D., (2001) "Fuzzy logic Techniques for Autonomous Vehicle Navigation", *Orebro University, Suiza.*

[5] Ibrahim, A., (2004) Fuzzy Logic for Embedded Systems Applications, Elsevier Sicence , USA

[6] MIT Technolgy Review, http://goo.gl/Yu6lwB

[7] Mousa T, (2011) "An Autonomous Fuzzy-Controlled Indoor Mobile Robot for Path Following and Obstacle Avoidance ", International Journal of Computers, Issue 3

[8] Olsen, D., "Fuzzy Logic Control in Autonomous Robotics", Investigating the Motorola MC68HC12 on a Line Following Robot, http://goo.gl/nJXvFY, University of Minnesota Duluth, USA

[9] Pagina de Producto del 3pi, http://goo.gl/NJOWF7

[10] TARMIZI M., (2011) "Line and Wall Follower Hexapod Robot" , Master Thesis, University Tun Hussein Onn Malaysia

[11] State of Nevada Department of Motor Vehicle, http://goo.gl/fzu6dW

"Copyright ©2014. "Edwin Marte": El autor delega a la Organización del Tercer Congreso Virtual de Microcontroladores la licencia para reproducir este documento para los fines del Congreso ya sea que este artículo se publique en el sitio web del congreso, en un CD o en un documento impreso de las ponencias del Segundo Congreso Virtual de Microcontroladores.