IMPLEMENTACIÓN DE UN CONTROLADOR PID MEDIANTE RNA PARA EL CONTROL DE MOTORES D.C DE ROBOTS MOVILES DIFERENCIALES

Implementation of PID controller using ANN for D.C motors control of differential mobile robots

Germán A. Hernández Millán[1], Luis Hernando Ríos González[2], Hernando Parra Lara[3]
Ingeniería Eléctrica, Universidad Tecnológica de Pereira, Pereira, Colombia
lhgonza@utp.edu.co

***Resumen*— Dado que no siempre es posible conocer con exactitud la función de transferencia de todo sistema se hace necesario utilizar herramientas diferentes a las tradicionalmente conocidas para desarrollar el control. Una de estas herramientas son las redes neuronales. Estas no necesitan conocer de manera directa el comportamiento de cada uno de los componentes del sistema para emularlo. Lo único que necesitan son datos para aprender el comportamiento deseado del sistema).**

En este artículo se demuestra que las redes neuronales pueden emular el comportamiento de un controlador PID para controlar los perfiles de velocidad de los motores de tracción un robot móvil tipo diferencial.

***Palabras clave*— Robót Móvil, Redes Neuronales, Controlador PID.**

***Abstract*— Since it is not always possible to know the exact transfer function of the whole system is necessary to use of tools to the traditionally knowns for developing the control. One of these tools are neural networks. They do not need to know directly the behavior of each of the components of the system to emulate the system. All they need are data to learn the desired behavior of the system.**

This paper shows that neural networks can emulate the behavior of a PID controller to control the speed profiles of traction motors of a mobile robot.

***Key Word* — Mobile Robot, Neural Network, PID Controller.**

I. INTRODUCCIÓN

Los robots móviles son dispositivos electromecánicos capaces de desplazarse en un espacio de trabajo con cierto grado de autonomía.
De acuerdo a su forma de locomoción se clasifican en:
Robots móviles de locomoción mediante orugas, mediante patas y mediante ruedas. Los robots móviles propulsados por ruedas a su vez, se clasifican de acuerdo al número y al tipo de grados de libertad[1].

Cuando se trata de generar trayectorias o caminos en sistemas no holonomos, hay características relevantes directamente relacionadas con el tipo de trayectorias que pueden seguir estos sistemas. En efecto, una configuración inicial y una final no pueden unirse mediante cualquier trayectoria. Las restricciones cinemáticas del sistema imponen unas condiciones que sólo algunos caminos cumplirán. Por camino se entiende la discretización de una función continua que interpola las configuraciones definidas en una ruta. Finalmente, cuando se habla de trayectoria, se hace referencia a un camino que tiene asociado un perfil cinemático; es decir, a cada configuración perteneciente al camino se le asocia una velocidad [2].

La simulación de un sistema de control nos permite comprobar con anterioridad si la implementación práctica de determinados métodos que se pretende implementar será viable o no. Mientras más completo sea el modelo a simular, por ejemplo las funciones de transferencia de cada uno de los componentes eléctricos, electrónicos y mecánicos, los resultados arrojados por la simulación serán lo más cercanos a la realidad.

[1] Ingeniero Electricista.

[2] Ingeniero Electrónico, M.sC.

[3] Ingeniero Mecánico, MsC

Fecha de Recepción: 25 de Enero de 2011
Fecha de Aceptación: 28 de Marzo de 2012

No siempre es posible conocer con exactitud la función de transferencia de todo el sistema por lo que se hace necesario utilizar herramientas diferentes a las tradicionalmente conocidas para desarrollar el control.

Una de estas herramientas son las redes neuronales. Estas no necesitan conocer de manera directa el comportamiento de cada uno de los componentes del sistema, para emularlo. Lo único que necesitan son datos para aprender el comportamiento deseado del sistema.

La configuración de la red neuronal es un punto fundamental para que una aplicación produzca buenos resultados. Si los datos de entrenamiento son correctos, pero la arquitectura de la red no es la adecuada para la aplicación, la red neuronal no se comportará como se espera.

Para el control de velocidad se implementó un controlador PID, el cual fué sintonizado y reajustado mediante la herramienta *sisotool* de MatLab [3]. Después de tener sintonizado el controlador de velocidad PID, se implementaron varias arquitecturas de RNA´s con el tipo de redes de Propagación hacia atrás(Back-propagation), las cuales fueron entrenadas con los datos obtenidos del controlador PID.

La función del controlador PID de velocidad se reemplazó por un controlador con redes neuronales, obteniéndose un esquema simplificado que permitió transformar datos de entrada en datos de salida utilizando multiplicaciones, sumas y cálculos exponenciales, de bajo costo computacional, haciendo para esta aplicación una eficiente implementación de las redes neuronales.

Con las primeras simulaciones se pudo comprobar que este método funciona muy bien tanto para seguimiento de trayectorias rectilíneas como curvilíneas.

En este artículo se demuestra que las redes neuronales pueden emular el comportamiento de un controlador PID para controlar los perfiles de velocidad de los motores de tracción de un robot móvil tipo diferencial.

II. MODELADO DEL MOTOR DC Y CONTROLADOR PID.

Para controlar la velocidad de los motores del robot móvil diferencial, se utilizó un controlador PID. Una vez obtenido el modelo matemático del sistema, se utilizó SimuLink para diseñar el controlador PID seleccionado y su respectiva sintonización. Como sistema a modelar en este caso se utilizaron los motores de corriente continua de imanes permanentes, que realizan la tracción de la plataforma móvil diferencial.

A. Parámetros del motor.

Inicialmente se obtuvo un modelo en forma de función de transferencia de todas las partes eléctricas y mecánicas del sistema físico que son necesarias para la implementación del controlador. En este modelo no se tuvo en cuenta las no linealidades y se considero el motor libre de carga. Para simular el modelo del motor es necesario tener los valores numéricos de cada una de las variables mostradas en el diagrama de bloques.

Las variables son:

Símbolo	Significado Físico	Unidades
Ra	Resistencia de armadura	(Ω)
La	Inductancia de armadura	(H)
Kb	Constante de par	(N·m /A)
f	Coeficiente de fricción	(N·m·s/rad)
J	Inercia del rotor	(kg·m2)
Kω	Constante de velocidad	(rad/(s · V))

Tabla 1. Parámetros del Motor

Los valores de los parámetros fueron obtenidos del motor que impulsará el robot móvil tipo diferencial. En este caso se utilizará el datasheet del Servo Motor DC 9236.

B. Circuito equivalente del motor DC [10].

Partiendo de las ecuaciones en función del tiempo que describen a nuestro sistema, se transforma todo al dominio de los números imaginarios con la transformada de Laplace y se efectúan algunas operaciones algebraicas para llegar a las siguientes ecuaciones:

$$\frac{U(s)-K\omega*\omega(s)}{(La*s+Ra)} = I(s) \qquad (1)$$

$$I(s) * Kb = T(s) \qquad (2)$$

$$\frac{T(s)}{J*s+f} = \omega(s) \qquad (3)$$

Estas ecuaciones se modelan en forma de bloques de transferencia en Simulink como se aprecia en la figura 1.

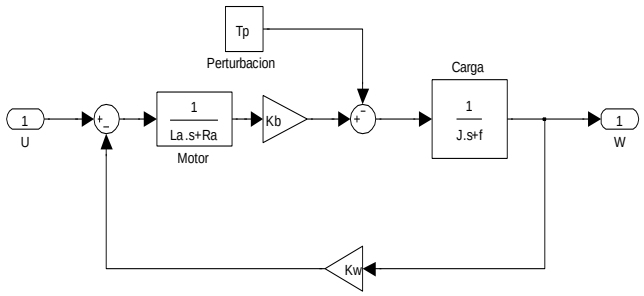


Figura 1. Diagrama de Bloques del motor.

Del diagrama de bloques anterior se puede deducir la función de transferencia en lazo abierto del sistema relacionando el par con el voltaje de entrada:

$$\frac{T(s)}{U(s)} = \frac{Kb*J*s*Kb*f}{J*La*s^2+(J*Ra+La*f)*s+Ra*f+K\omega*Kb} \qquad (4) \quad (4.15)$$

C. Modelado matemático del control PID.

Un controlador es un dispositivo que se usa para modificar la conducta dinámica de un proceso. Existen 2 decisiones clave en el diseño de sistemas de control:

1. Selección del tipo de controlador (P, PI, PID, PD).
2. Selección de los parámetros del controlador (*K, T i, T d*).

El tipo de controlador, así como los parámetros de dicho controlador, se deben elegir de tal forma que se satisfagan ciertos objetivos de operación en lazo cerrado.

Algunos de los objetivos de operación más comunes son:

a) Mantener Estabilidad.
b) Minimizar el error en estado estacionario.
c) Dar cierta forma a la respuesta dinámica.

El controlador PID, es el algoritmo de control más común. Numerosos lazos de control utilizan este algoritmo, que puede ser implementado de diferentes formas y su estudio puede ser abordado desde múltiples puntos de vista. Puede ser tratado como un dispositivo que puede ser operado utilizando unas cuantas reglas prácticas, pero también puede ser estudiado analíticamente. El controlador PID (Proporcional, Integral y Derivativo) es un controlador realimentado cuyo propósito es hacer que el error en estado estacionario, entre la señal de referencia y la señal de salida de la planta, sea cero de manera asintótica en el tiempo, lo que se logra mediante el uso de la acción integral. Además el controlador tiene la capacidad de anticipar el futuro a través de la acción derivativa que tiene un efecto predictivo sobre la salida del proceso.

El comportamiento del algoritmo PID se puede describir como:

$$u(t) = K\left(e(t) + \frac{1}{T_i}\int_0^t e(\tau)d\tau + T_d\frac{de(t)}{dt}\right) \qquad (5)$$

Donde u es la variable de control y e es el error de control, el cual se define como $e = (y_{sp} - y)$, donde y_{sp} es el valor de la referencia y y es la salida del proceso.

De esta manera, la variable de control es una suma de tres términos: el término P, que es proporcional al error; el término I, que es proporcional a la integral del error; y el término D, que es proporcional a la derivada del error. Los parámetros del controlador son: la ganancia proporcional *K*, el tiempo integral T_i y el tiempo derivativo T_d.

D. Controlador PID.

Después de conseguir los datos de los parámetros necesarios y partiendo de las ecuaciones que describen nuestro sistema se procedió a montar en Simulink un bloque que modelara el motor, el cual se denominó ***moto1.mdl*** como se muestra en la figura 2.

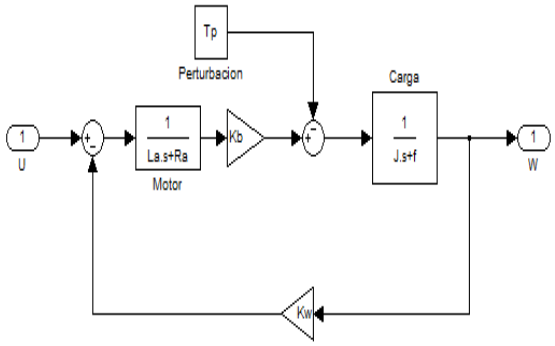


Figura 2. Diagrama de Bloques del motor.

Este bloque permite calcular la función de trasferencia en MatLab de la siguiente forma:

```
[A,B,C,D]=linmod('moto1');
[num,den]=ss2tf(A,B,C,D);
planta=tf(num,den);
```

De esta manera se calcula la siguiente función de transferencia para los motores del modelo de robot propuesto.

$$\frac{4.908e006}{s^2 + 1076\ s + 1.134e005} \qquad (6)$$

Con la ayuda de la GUI Sisotool de MatLab se procedió a calcular un controlador PID para la función de transferencia, observándose que el sistema se estabiliza en $ts = 0.0121s$ y que la señal tiene un sobrepaso máximo del 42%. Teniendo en cuenta este tiempo de asentamiento (ts), seleccionamos un tiempo de establecimiento de $ts = 0.005seg$ y un $sobrepaso = 5\%$ para el diseño del controlador.

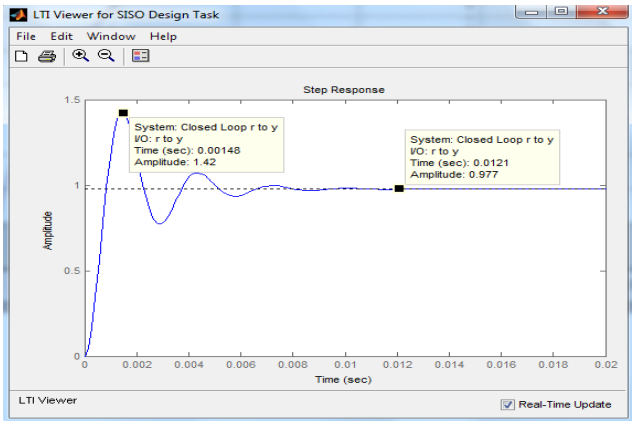


Figura 3. Ventana LTI Viewer

A continuación se procedió a calcular el controlador PID.

El objetivo al que se quiere llegar para obtener el controlador deseado, es el de posicionar los ceros y el polo de tal forma que la línea de acción pase justamente en el lugar donde se cruzan el sobrepaso y el tiempo de establecimiento; para lograr esto se llevan los polos dominantes a ese punto, en el cual el controlador cumple con las condiciones de diseño establecidas. Ver figura 4.

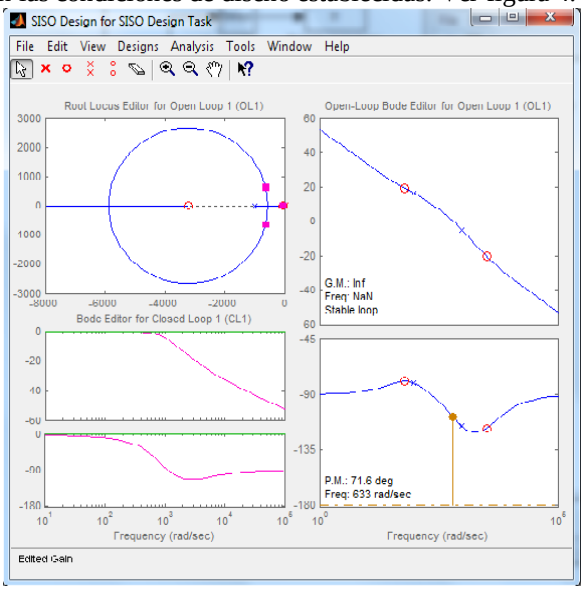


Figura 4. Ventana Edited Gain.

El controlador que se obtuvo fue el siguiente:

$$17.551 * \frac{(1 + 0.0003S) * (1 + 0.014S)}{S}$$

Y la respuesta al escalón del sistema en lazo cerrado con el controlador obtenido, se observa en la siguiente figura.

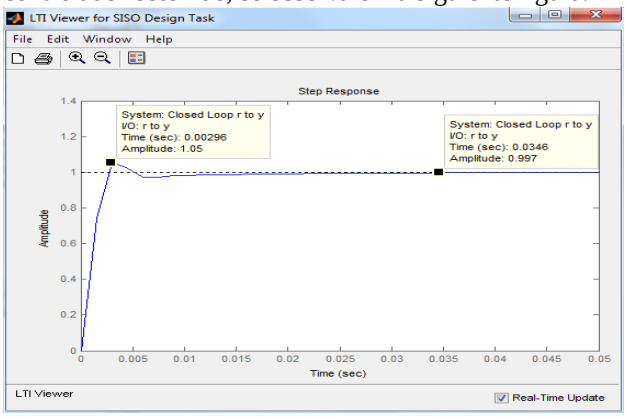


Figura 5. Ventana Step Response

En la figura 5 se puede observar que el sobrepaso es del 5% y el tiempo de establecimiento ts de 3.5. E-2 seg, que son los parámetros deseados de tiempo de establecimiento propuestos para el controlador. Por lo tanto, se elige implementar este controlador. La figuras 6 muestra las respuestas del motor ante un tren de escalones.

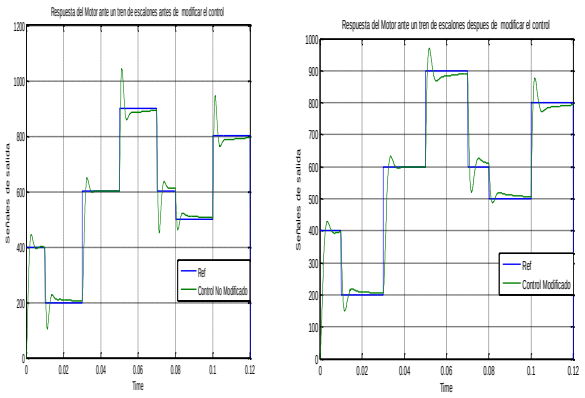


Figura 6. Respuesta del Motor ante un tren de escalones antes y después de modificar el control

Posteriormente se implementa en SimuLink, un nuevo diagrama de bloques el cual está conformado por una señal de referencia de entrada, el controlador diseñado y el modelo del motor. Este modelo permite exportar los datos de entrada y salida del controlador a Workspace de MatLab por medio de las variables P1 y T1. Estas variables permiten hacer un entrenamiento supervisado a una red neuronal artificial, la cual se diseña con ayuda de la GUI NNtool. El bloque de SimuLink se llama PID_Var.mdl el cual se muestra en la figura 7.

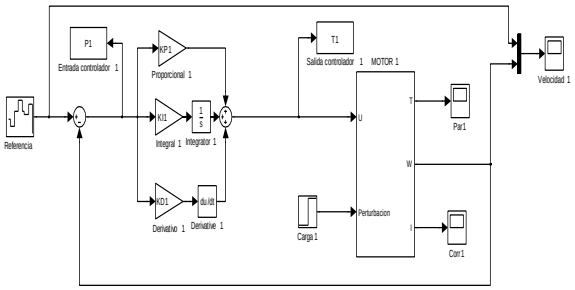


Figura 7. Diagrama de bloques de SimuLink de donde se obtienen los datos P1, T1.

E. Controlador RNA.

Para el diseño de la red neuronal artificial se emplea la GUI ***nntool*** la cual permite crear RNA´s tipo BP de múltiple capa con prealimentación (Feed-Forward Backpropagation).

Los pasos necesarios para crear la RNA son los siguientes:

• Obtener un controlador que genere la señal de control a emular.

• Seleccionar el conjunto de datos de entrenamiento para la RNA BP.

• Implementar en la GUI ***nntool*** de MatLab la RNA BP.

• Entrenar la RNA BP en base al conjunto de datos obtenidos.

Con ayuda del bloque de SimuLink llamado PID_Var.mdl se obtuvieron las variables P1 y T1 con las cuales se obtuvieron los datos de entrenamiento para la RNA BP. Del arreglo vectorial se obtuvieron 1200 datos discriminados así: Los datos de entrada en un vector de 900 datos que se emplearon para entrenar la

RNA y otro vector de 300 datos para probar la red, el código también hace lo mismo con los datos de salida deseados T1 pero con la diferencia que con el comando mapminmax los vectores de salida quedan normalizados, para posteriormente exportarlos a la GUI del *NNtool* de MatLab. Luego en la GUI del *NNtool* se comienza importando los vectores mediante la opción *Import*, para posteriormente diseñar una red capaz de emular el control proporcional integral derivativo (PID).

En esta etapa se intento entrenar varias redes con diferentes arquitecturas, las cuales no arrojaron buenos resultados, estas arquitecturas se implementaron con una estrategia de prueba y error, después de intentar con varias redes se llego a dos redes que si arrojaron buenos resultados, tales redes fueron una RNA de tres capas, que consistía en una primera capa (capa de entrada) de diez neuronas con una función de transferencia TANSIG, dos neuronas en la capa interna (capa oculta) con una función de transferencia PURELIN, y una neurona en la capa externa (capa de salida) con una función de transferencia PURELIN. La segunda red que arrojo buenos resultados fue una red con diez neuronas en la capa de entrada con una función de transferencia TANSIG, y una capa de salida de una neurona con una función de transferencia PURELIN.

Para generar los datos en SimuLink fue necesario realizar algunos ajustes en los parámetros de simulación para conseguir 1200 datos por simulación, para conseguir esta cantidad de datos se optó por modificar los parámetros del SimuLink como el cambiar en el ***solver*** la opción de paso fijo a 0.001 segundos ya, que en este caso, la red neuronal va a funcionar con muestreos de paso fijos de 0.001 segundos.Considerando el control PID a emular por la RNA BP se procedío a implementarla mediante la herramienta nntool de MatLab. Para crear la RNA se definió la regla de aprendizaje de gradiente descendiente y la cantidad de épocas (cantidad de iteraciones a realizar antes de parar) en que se entrenó la RNA, hasta llegar a un error permitido. En este caso se optó por entrenar ambas redes utilizando 500 épocas y una meta de error (goal) de 0.01.

III. SIMULACIÓN DEL CONTROLADOR PID MEDIANTE RNA EN SIMULINK.

Después de tener las redes RNA entrenadas con la herramienta nntool se procedió a exportarlas al espacio de trabajo de MatLab, para posteriormente crear la RNA como un bloque de SimuLink con ayuda del comando *gensim(red,st)* (donde "red" es el nombre de la RNA y "st" es el tiempo de muestreo) y poder sustituir el controlador PID por este bloque. Las Redes Neuronales RNA's obtenidas se almacenaron con los nombres *RNA_2c.mdl* y *RNA_3c.mdl.* Luego con un Archivo de SimuLink llamado *com_PID_Vs_RNA.mdl(Ver* figura 8), se procedió a simular las RNA´s y el PID. Las graficas obtenidas por este modelo se presentan en la figura 9.

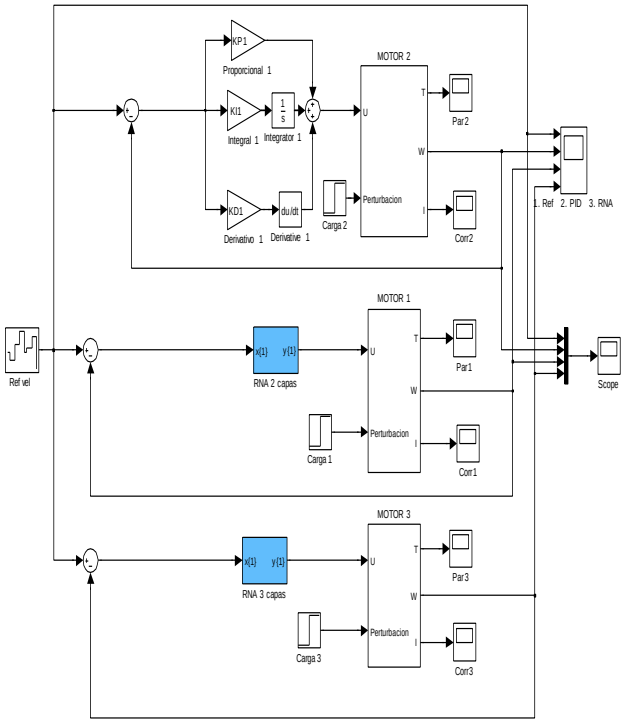


Figura 8. Archivo de SimuLink *com_PID_Vs_RNA.mdl*

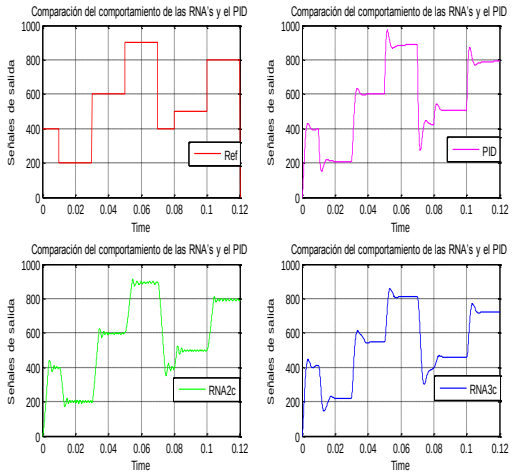


Figura 9. Comparación del comportamiento de las RNA's y el PID frente a un tren de pulsos.

IV. RESULTADOS OBTENIDOS.

En la figura 10 puede apreciarse la superposición de la señal de salida de las RNA's y el control PID, en Azul se muestra la señal de referencia, en color verde la señal del control PID, en color rojo la señal de la RNA de dos capas y en color azul la señal de la RNA de 3 capas.

Este par de redes neuronales se escogieron porque fueron las que mejor comportamiento presentaron ante diferentes entradas de

entrenamiento. Estas graficas se obtuvieron con un modelo de SimuLink llamado *Prueba_RNA_2c.mdl* para la RNA de dos capas y Prueba_RNA_3c.mdl, estos esquemas son iguales a los empleados para tomar los datos de entrenamiento solo que aquí se reemplazo el control PID por la RNA entrenada. Con dichos bloques se procedió a probar las redes con entradas diferentes a las presentadas para su entrenamiento para así comprobar la eficiencia de las redes diseñadas. En la figura 11, se muestran las salidas obtenidas por las 2 arquitecturas de redes seleccionadas frente a un tren de pulsos. La figura 12 muestra las salidas obtenidas por las 2 arquitecturas de redes seleccionadas frente a una entrada sinusoidal. En la figura 13 se muestra el esquema del modelo Prueba_RNA_2c.mdl.

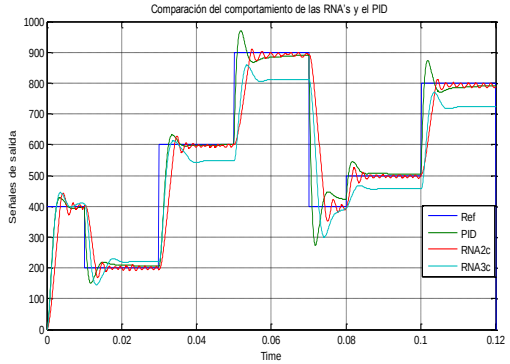
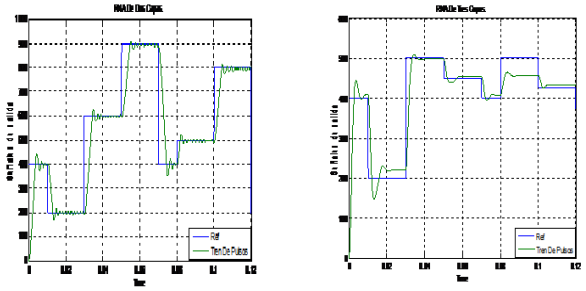
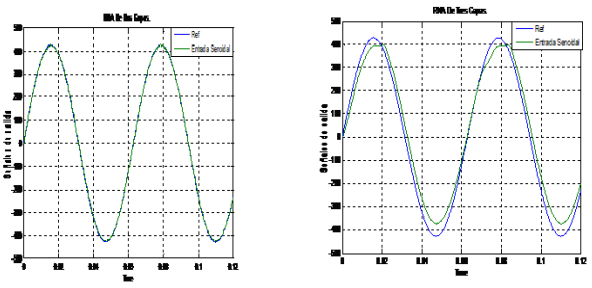


Figura 10. Superposición del comportamiento de las RNA's y el PID



a) b)

Figura 11. Salida frente a un tren de pulsos:a) Arquitectura de Red de 2 capas, b) Arquitectura de Red de 3 capas.



a) b)

Figura 12. Salida frente a un entrada sinusoidal:a) Arquitectura de Red de 2 capas, b) Arquitectura de Red de 3 capas.

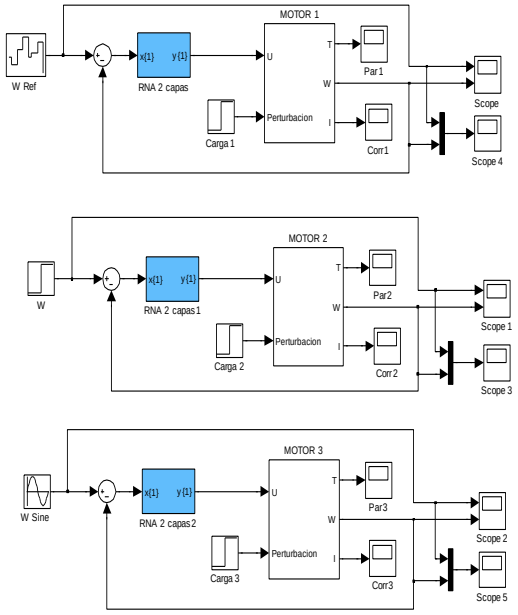


Figura 13. Modelo Prueba_RNA_2c.mdl

V. CONCLUSIONES.

Como puede observarse en las figuras anteriores las RNA´s no solo imitan bien la entrada de los datos de entrenamiento sino que también presenta un buen comportamiento frente a diferentes entradas.

Como puede observarse en las figuras la RNA que presenta mejor comportamiento fue la RNA de dos capas, motivo por el cual, será esta la red seleccionada para el control de los motores de la plataforma.

REFERENCIAS

[1]. G. Campion, G. Bastin, and B. D .Andrea Novel. Structural properties and clasification of kinematic and dynamics models of wheeled mobile robots.IEEE Transaction on Robotics and Automation. Vol 12, No. 1, 1996, pp. 47-61.

[2]. Ollero, A. (2001). "Robótica: manipuladores y Robots Móviles". Marcombo Boixareu.
[3]. Implementación de dos estrategias de control para la velocidad de un Motor DC. Pagina Web: http:// *www.revistas.unal.edu.co/index.php/ingeinv/article/*
[4]. Modelado de un motor DC. Análisis Dinámico de sistemas (Teleco).Pagina Web: http://isa.uniovi.es/~idiaz/ADSTel/Practicas/ModeladoMotorCC.html
[5]. Control de procesos. Acción de los controladores. Página Web: http://controldprocesos.blogspot.com/2010/05/accion-de-los- controladores.html