

INSTITUTO POLITÉCNICO NACIONAL

ESCUELA SUPERIOR DE INGENIERÍA MECÁNICA Y ELÉCTRICA

**SECCIÓN DE ESTUDIOS DE POSGRADO E INVESTIGACIÓN
DEPARTAMENTO DE INGENIERÍA ELÉCTRICA**

**IMPLEMENTACIÓN HIL DE UN CONTROL DE
VELOCIDAD PARA UN MOTOR DE CD UTILIZANDO
UNA RED NEURONAL**

TESIS

**QUE PARA OBTENER EL GRADO DE:
MAESTRO EN CIENCIAS
EN INGENIERÍA ELÉCTRICA**

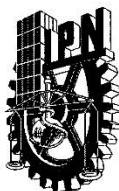
PRESENTA:

ING. EDGAR ADEMIR MORALES PÉREZ



MÉXICO, D. F.

2015



INSTITUTO POLITÉCNICO NACIONAL
SECRETARÍA DE INVESTIGACIÓN Y POSGRADO
ACTA DE REVISIÓN DE TESIS

En la Ciudad de México, D. F. siendo las 12:00 horas del día 15 del mes de Diciembre del 2015 se reunieron los miembros de la Comisión Revisora de la Tesis, designada por el Colegio de Profesores de Estudios de Posgrado e Investigación de ESIME-Zacatenco para examinar la tesis titulada:

**IMPLEMENTACIÓN HIL DE UN CONTROL DE VELOCIDAD PARA UN MOTOR
DE CD UTILIZANDO UNA RED NEURONAL**

Presentada por el alumno:

MORALES

Apellido paterno

PÉREZ

Apellido materno

EDGAR ADEMIR

Nombre(s)

Con registro:

B	1	3	0	7	3	5
---	---	---	---	---	---	---

aspirante de:

MAESTRO EN CIENCIAS EN INGENIERÍA ELÉCTRICA

Después de intercambiar opiniones, los miembros de la Comisión manifestaron **APROBAR LA TESIS**, en virtud de que satisface los requisitos señalados por las disposiciones reglamentarias vigentes.

LA COMISIÓN REVISORA

Director(a) de tesis

DR. DOMITILLO LIBREROS

PRESIDENTE

DR. DAVID ROMERO ROMERO

SEGUNDO VOCAL

DR. RAÚL ÁNGEL CORTÉS MATEOS

TERCER VOCAL

DR. GERMAN ROSAS ORTIZ

SECRETARIO

DR. JAIME JOSÉ RODRÍGUEZ RIVAS

PRESIDENTE DEL COLEGIO DE PROFESORES

DR. MAURO ALBERTO ENCISO AGUILAR

IPN
SECCIÓN DE ESTUDIOS DE
POSGRADO E INVESTIGACIÓN

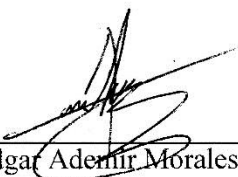


INSTITUTO POLITÉCNICO NACIONAL
SECRETARÍA DE POSGRADO E INVESTIGACIÓN

CARTA CESIÓN DE DERECHOS

En la Ciudad de México, D.F. el día 26 del mes de Noviembre del año 2015, el que suscribe EDGAR ADEMIR MORALES PÉREZ alumno del Programa de Maestría en Ciencias en Ingeniería Eléctrica, con número de registro B130735, adscrito a la Sección de Estudios de Posgrado e Investigación de la ESIME-Zacatenco del IPN, manifiesta que es el autor intelectual del presente trabajo de Tesis bajo la dirección del Doctor Domitilo Libreros y cede los derechos del trabajo titulado “IMPLEMENTACIÓN HIL DE UN CONTROL DE VELOCIDAD PARA UN MOTOR DE CD UTILIZANDO UNA RED NEURONAL”, al Instituto Politécnico Nacional para su difusión, con fines académicos y de investigación.

Los usuarios de la información no deben reproducir el contenido textual, gráficas o datos del trabajo sin el permiso expreso del autor y/o director(es) del trabajo. Este puede ser obtenido escribiendo a las siguientes direcciones ademirmorales501@gmail.com y dlibreros@ipn.mx. Si el permiso se otorga, el usuario deberá dar el agradecimiento correspondiente y citar la fuente del mismo.



Edgar Ademir Morales Pérez

RESUMEN

En el presente trabajo se desarrolla un sistema de simulación computacional de un control de velocidad de un motor de Corriente Directa en lazo cerrado el cual incluye al dispositivo físico con el algoritmo de control integrado. La simulación está diseñada para funcionar en tiempo real, y se auxilia de un software de instrumentación virtual con una interfaz gráfica de usuario, LabVIEW.

La ejecución en tiempo real de la simulación denominada HIL por sus siglas en inglés (*Hardware in the Loop*), o Hardware dentro del lazo de simulación, se logró gracias a un modelo del caso de estudio basado en una red neuronal.

Para el caso de estudio, motor de Corriente Directa, se diseña un regulador de velocidad con un algoritmo de control inteligente basado en redes neuronales y un algoritmo convencional con fines comparativos, los cuales son implementados en el sistema embebido FRDM-K64F.

El diseño de los sistemas de control implementados en el sistema embebido fue llevado a la simulación HIL con los resultados esperados, obteniendo una visualización en tiempo real de la velocidad del motor y de la regulación de la velocidad.

ABSTRACT

In the present work a computer simulation of a speed control closed-loop system of a Direct Current Motor that includes hardware was developed. The simulation is designed to be a real-time execution and was based on the virtual instrumentation software, LabVIEW, developing a graphical user interface.

Real-time simulation HIL (Hardware in the Loop), was based on a Neural Network Model of the study case.

A speed regulator is designed using an intelligent algorithm based on a Neural Network and a classical algorithm for comparative purposes. Both controllers, intelligent and classical, were implemented on the embedded system FRDM-K64F.

The design of the control systems was taken to the HIL simulation with the expected results, obtaining a real time dc motor speed regulation and visualization.

DEDICATORIA

A mis padres:

Miguel Morales Martinez

Dora Maria Leticia Pérez Medellín

A mis hermanos:

Omar Morales Pérez

Miguel Andrei Morales Pérez

Rafael Alberto Olvera Morales

Laura Manzana Olvera Morales

A mi prima:

Martha Laura Morales Martinez

A mi pareja y amiga:

Nancy Nathaly Pérez Hernández

Son el motivo, lo mejor y lo más importante de mi vida.

AGRADECIMIENTOS

A Dios, por permitirme ser parte de este mundo, darme salud y oportunidades para desarrollarme como individuo.

A mi padre, Miguel, por su apoyo incondicional y la sabiduría que ha compartido conmigo toda la vida.

A mi madre, Leticia, por su cariño y comprensión, su apoyo y consuelo en mis momentos difíciles, y por el amor que me ha regalado toda la vida.

A mis hermanos, Manzana, Micky, Beto y Omar, por compartir y disfrutar de la vida junto conmigo.

A mi prima, Martha, por estar siempre al pendiente, por ayudarme cuando lo necesito, por su apoyo y alegría.

A mi novia, Nathaly, por su manera de hablarme, cuidarme, amarme y regañarme, sin ti, la vida no sería tan maravillosa como es.

A mis amigos, Julio, Joana, Jesús, por los divertidos días que hemos compartido, sus consejos y alegrías.

A mis compañeros de la maestría, por compartir este reto y por el apoyo que me brindaron numerosas veces, en especial a Francisco Limón, Pedro De Mata y Viviana Reyes.

A los profesores de la SEPI, en especial al Dr. Domitilo Libreros, por la asesoría y consejos para esta tesis, al Dr. Raúl Cortés Mateos, por sus enseñanzas y conocimientos, al Dr. David Romero Romero, por compartir su experiencia y sabiduría, al Dr. Jaime Rodríguez por sus excelentes clases, al Dr. German Rosas Ortiz y al Dr. David Sebastian por los consejos que mejoraron mi redacción y el trabajo de tesis y al resto de Profesores y personal.

Al Dr. Daniel Ruiz, a la Sra. Lilia Cruz y a la Sra. Ana Cruz, por su excelente trabajo y apoyo en los trámites y vida escolar.

Al Conacyt por el apoyo económico durante el periodo de la elaboración de la tesis y al IPN por la formación académica brindada todos estos años.

¡MUCHAS GRACIAS!

CONTENIDO

	Página
RESUMEN.....	VII
ABSTRACT.....	IX
DEDICATORIA	XI
AGRADECIMIENTOS	XIII
CONTENIDO.....	XV
LISTA DE FIGURAS	XVII
LISTA DE TABLAS	XIX
ABREVIATURAS.....	XXI
CAPÍTULO 1: INTRODUCCIÓN.....	1
1.1 PRELIMINARES.....	1
1.2 INTRODUCCIÓN.....	1
1.3 PLANTEAMIENTO DEL PROBLEMA.....	2
1.4 OBJETIVO.....	3
1.5 OBJETIVOS ESPECÍFICOS.....	3
1.5 JUSTIFICACIÓN.....	4
1.6 LIMITACIONES Y ALCANCES	4
1.7 ESTADO DEL ARTE.....	5
1.7.1 Trabajos Desarrollados a Nivel Internacional	5
1.7.2 Trabajos Desarrollados en la S.E.P.I.-E.S.I.M.E. ZAC.	6
1.8 APORTACIONES.....	6
1.9 ESTRUCTURA DE LA TESIS	7
CAPÍTULO 2: DESARROLLO DEL CONTROL DE VELOCIDAD DEL MOTOR DE CD.	9
2.1 INTRODUCCIÓN.....	9
2.2 MÁQUINA DE CORRIENTE DIRECTA.....	9
2.2.1 Modelado Matemático del Motor de CD.	9
2.2.2 Análisis del Motor de CD en lazo abierto.....	13
2.3 DISEÑO DEL SISTEMA DE CONTROL PI.....	17
2.3.1 Primer Método de Ziegler Nichols.....	18
2.3.2 Herramienta de Auto Ajuste de MATLAB.....	22
2.3.3 Comparativa de los Controladores PI.....	24
2.4 DISEÑO DEL SISTEMA DE CONTROL NEURONAL	25
2.5 ESTUDIO COMPARATIVO ENTRE EL CONTROL PI Y EL NNDIC	32
CAPÍTULO 3: IMPLEMENTACIÓN HIL	35
3.1 INTRODUCCIÓN.....	35
3.2 IDENTIFICACIÓN DEL MOTOR DE CD.	37
3.2.1 Recolección de Datos.....	37
3.2.2 Selección de la Estructura del Modelo.	38
3.2.3 Ajuste del Modelo a los Datos.	39
3.2.4 Validación del Modelo.....	40
3.3 PROGRAMACIÓN EN LABVIEW	41
3.3.1 Programación de la RN.....	42
3.3.2 Comunicación con el Sistema Embebido.	45
3.3.3 Programación de la Interfaz Gráfica de Usuario.....	45
3.4 IMPLEMENTACIÓN DE LOS ALGORITMOS DE CONTROL EN EL SISTEMA EMBEBIDO.....	48

3.4.1 Control PI.....	48
3.4.2 Control NNDIC	49
3.5 SIMULACIÓN HIL EN LAZO CERRADO	51
CAPÍTULO 4: PRUEBAS EXPERIMENTALES Y RESULTADOS.....	53
4.1 INTRODUCCIÓN	53
4.2 PRUEBAS DE DESEMPEÑO DEL SISTEMA DE CONTROL	53
4.3 RESULTADOS DEL SISTEMA DE CONTROL MÁQUINA DE CORRIENTE DIRECTA	53
4.3.1 Prueba Entrada Escalón	54
4.3.2 Prueba Cambio de Punto de Ajuste.....	55
4.3.3 Prueba de disturbio, aumento de carga mecánica de 1 N.m.	57
CAPÍTULO 5: CONCLUSIONES	59
5.1 CONCLUSIONES	59
5.2 APORTACIONES	60
5.3 TRABAJOS FUTUROS Y RECOMENDACIONES.....	61
REFERENCIAS.....	63
APÉNDICE A: DATOS DEL MOTOR DE CD.....	65
APÉNDICE B: FUNDAMENTOS TEÓRICOS DE LAS REDES NEURONALES ARTIFICIALES	67
B.1 INTRODUCCIÓN	67
B.2 CAPACIDAD DE APRENDIZAJE.....	69
B.3 REGLA DELTA	69
B.4 ALGORITMO DE RETROPROPAGACIÓN	72
B.5 CONTROL NEURONAL.....	74
APÉNDICE C: ÍNDICES DE DESEMPEÑO	77
C.1 INTRODUCCIÓN.....	77
C.2 ITAE	77
C.3 IAE.....	77
C.3 ISE	78
APÉNDICE D: CÓDIGOS DEL SISTEMA EMBEBIDO.....	79
D.1 INTRODUCCIÓN.....	79
D.2 SISTEMA EMBEBIDO	79
D.2.1 Características del FRDM-K64F	79
D.3 CÓDIGOS DEL CASO DE ESTUDIO MOTOR DE CD	80
D.3.1 Control PI.....	80
D.3.2 Control NNDIC.....	81
APÉNDICE E: PROGRAMACIÓN DEL SIMULADOR EN LABVIEW	85
E.1 INTRODUCCIÓN	85
E.2 EJECUCIÓN DEL PROGRAMA	86
E.2.1 Configuración de la Comunicación RS232	86
E.2.2 Conversión de Valores	86
E.2.3 Lectura y Escritura de las Señales.....	87
E.2.4 Modelo Neuronal	87
E.2.5 Visualización de las variables, simulación de disturbios y cambios en el Punto de Ajuste	87
E.2.6 Criterios de Simulación.....	88
E.2.7 Panel Frontal: Interfaz de Usuario.....	88

LISTA DE FIGURAS

FIGURA 1.1 ESQUEMA GENERAL DE LA SIMULACIÓN HIL.....	3
FIGURA 2.1 ESQUEMA GENERAL DE LAS PARTES ELÉCTRICA Y MECÁNICA DEL MOTOR [16].	10
FIGURA 2.2 DIAGRAMA DE SIMULACIÓN MOTOR DE CD.	12
FIGURA 2.3 RESPUESTA A UNA ENTRADA DE 1 VCD SIN CARGA.	13
FIGURA 2.4 CORRIENTE DE ARMADURA ENTRADA DE 1 VCD.....	14
FIGURA 2.5 RESPUESTA A UNA ENTRADA DE 220 VCD SIN CARGA.	15
FIGURA 2.6 CORRIENTE DE ARMADURA ENTRADA DE 220VCD SIN CARGA.....	15
FIGURA 2.7 CAMBIO DE CARGA DE 1 N.M A LOS 1.5 SEGUNDOS.	16
FIGURA 2.8 CURVA AUXILIAR DEL PRIMER MÉTODO DE ZIEGLER NICHOLS.....	18
FIGURA 2.9 DERIVADA DE LA RESPUESTA ESCALÓN DEL MOTOR DE CD.	19
FIGURA 2.10 RESPUESTA A ESCALÓN UNITARIO CON LÍNEA TANGENTE AL PUNTO DE INFLEXIÓN.	20
FIGURA 2.11 SISTEMA DE CONTROL PI SINTONIZADO CON EL PRIMER MÉTODO DE ZIEGLER NICHOLS	21
FIGURA 2.12 SISTEMA DE CONTROL PI CON SINTONIZACIÓN FINA.	22
FIGURA 2.13 MODELO SIMULINK CON HERRAMIENTA DE AUTO AJUSTE DE CONTROL PI.....	22
FIGURA 2.14 HERRAMIENTA DE AUTO AJUSTE DE GANANCIAS PI DE MATLAB.	23
FIGURA 2.15 RESULTADOS DE LA HERRAMIENTA DE SINTONIZACIÓN AUTOMÁTICA.	23
FIGURA 2.16 RESPUESTA EN EL TIEMPO DE SISTEMA DE CONTROL PI CON LOS DOS MÉTODOS DE SINTONIZACIÓN.	24
FIGURA 2.17 ESQUEMA DE RN PARA IDENTIFICAR SISTEMAS DINÁMICOS.....	27
FIGURA 2.18 ESQUEMA DE LA RN PROPUESTA COMO NNDIC.	28
FIGURA 2.19 PROGRAMA DE SIMULINK PARA LA GENERACIÓN DE PATRONES DE ENTRENAMIENTO.....	29
FIGURA 2.20 EJEMPLO DE PATRONES DE ENTRENAMIENTO.	29
FIGURA 2.21 ENTRENAMIENTO DE LA DINÁMICA INVERSA.....	30
FIGURA 2.22 RESPUESTA ESCALÓN DEL SISTEMA DE CONTROL NNDIC.....	31
FIGURA 2.23 COMPARATIVA DE SISTEMAS DE CONTROL PI (LÍNEA PUNTEADA) Y NNDIC (LÍNEA SÓLIDA).	32
FIGURA 3.1 DIAGRAMA A BLOQUES DEL SIMULADOR HIL.	36
FIGURA 3.2 DIAGRAMA DE FLUJO PARA LA IDENTIFICACIÓN DE SISTEMAS.....	37
FIGURA 3.3 MODELO DE SIMULINK PARA GENERAR PATRONES DE ENTRENAMIENTO.....	38
FIGURA 3.4 PRUEBA ESCALÓN UNITARIO PARA VERIFICACIÓN DE LA RN COMO MODELO.....	40
FIGURA 3.5 PRUEBA ESCALÓN DE 0.5, PARA VERIFICACIÓN DE LA RN COMO MODELO.....	41
FIGURA 3.6 SUBVI RN COMO MOTOR DE CD.	42
FIGURA 3.7 VECTOR DE ENTRADAS	43
FIGURA 3.8 VECTOR DE PESOS DE LA NEURONA 1 DE LA CAPA OCULTA.	43
FIGURA 3.9 OPERACIONES DE LA ECUACIÓN 3.14 POR NEURONA.....	43
FIGURA 3.10 OPERACIONES DE LA CAPA DE SALIDA DE LA ECUACIÓN 3.4.....	44
FIGURA 3.11 RESPUESTA DE LA RN EN EL SIMULADOR DE LABVIEW.....	44
FIGURA 3.12 CONFIGURACIÓN SERIAL LABVIEW.....	45
FIGURA 3.13 PROGRAMACIÓN DE LA VISUALIZACIÓN SIMPLE.	46
FIGURA 3.14 PERILLA VIRTUAL PARA LA SIMULACIÓN DE CAMBIOS DE CARGA.	46
FIGURA 3.15 SELECTOR HORIZONTAL PARA SIMULAR EL PUNTO DE AJUSTE.	47
FIGURA 3.16 BOTONES VIRTUALES PARA EL MANEJO DE LA SIMULACIÓN.....	47
FIGURA 3.17 DIAGRAMA DE FLUJO DEL CONTROL PI EMBEBIDO EN FRDM-K64F.....	49
FIGURA 3.18 DIAGRAMA DE FLUJO DEL CONTROL NNDIC EMBEBIDO EN FRDM-K64F.....	50
FIGURA 3.19 INTERFAZ GRÁFICA DE USUARIO DEL SIMULADOR HIL EN TIEMPO REAL.....	51
FIGURA 4.1 PRUEBA A ENTRADA ESCALÓN CONTROL PI	54
FIGURA 4.2 PRUEBA A ENTRADA ESCALÓN CONTROL NNDIC.	55
FIGURA 4.3 CAMBIO EN EL PUNTO DE AJUSTE A 0.5 CONTROL PI.....	56
FIGURA 4.4 CAMBIO EN EL PUNTO DE AJUSTE A 0.5 CONTROL NNDIC.....	56
FIGURA 4.5 AUMENTO DE CARGA DE 1 N.M COMPENSADO POR CONTROL PI.....	57
FIGURA 4.6 AUMENTO DE CARGA DE 1.N.M COMPENSADO POR CONTROL NNDIC.	58
FIGURA B.1 MODELO DE McCULLOCH PITTS DE LA NEURONA ARTIFICIAL.....	68

FIGURA B.2 SISTEMA DE CONTROL.....	74
FIGURA B.3 EFECTO DE LA DINÁMICA INVERSA EN SERIE	75
FIGURA D.1 CÓDIGO CONTROL PI MOTOR DE CD.....	80
FIGURA D.2 CÓDIGO CONTROL NNDIC MOTOR DE CD, FUNCIÓN PRINCIPAL.....	81
FIGURA E.1 DIAGRAMA DE FLUJO PROGRAMA DE LABVIEW	85
FIGURA E.2 CONFIGURACIÓN SERIAL	86
FIGURA E.3 CONVERSIÓN DE VALORES	86
FIGURA E.4 LECTURA (R) Y ESCRITURA (W).....	87
FIGURA E.5 MODELO NEURONAL EN LABVIEW	87
FIGURA E.6 INSTRUMENTOS VIRTUALES	87
FIGURA E.7 CRITERIOS DE SIMULACIÓN	88
FIGURA E.8 PANEL FRONTAL.....	89

LISTA DE TABLAS

TABLA 2.1 PARÁMETROS DEL MOTOR DE CD.	10
TABLA 2.2 DESCRIPCIÓN DE LAS VARIABLES DE LA FIGURA 2.1.	11
TABLA 2.3 PARÁMETROS DE LA RESPUESTA DE LA FIGURA 2.3.	14
TABLA 2.4 INTERVALOS NORMALIZADOS.	16
TABLA 2.5 REQUISITOS DEL SISTEMA DE CONTROL [21].	17
TABLA 2.6 DESCRIPCIÓN DE LOS PARÁMETROS DE LA FIGURA 2.8.	18
TABLA 2.7 SINTONIZACIÓN DE CONTROLADORES PI CON EL PRIMER MÉTODO DE ZIEGLER NICHOLS [20].	20
TABLA 2.8 RESULTADOS DEL SISTEMA DE CONTROL PI SINTONIZADO.	21
TABLA 2.9 RESULTADOS DE LA SINTONIZACIÓN FINA Y AUTOMÁTICA.	24
TABLA 2.10 ÍNDICES DE DESEMPEÑO DE LOS SISTEMAS DE CONTROL PI.	25
TABLA 2.11 PARÁMETROS DE LA RN PROPUESTA.	28
TABLA 2.12 RESULTADOS DEL ENTRENAMIENTO DE LA RN COMO NNDIC.	30
TABLA 2.13 PESOS DE LA CAPA OCULTA DESPUÉS DEL ENTRENAMIENTO.	30
TABLA 2.14 PESOS DE LA CAPA DE SALIDA DESPUÉS DEL ENTRENAMIENTO.	31
TABLA 3.1 ESTRUCTURA DE LA RN PARA IDENTIFICAR AL MOTOR DE CD.	38
TABLA 3.2 RESULTADOS DEL ENTRENAMIENTO DE LA RN COMO MODELO DEL MOTOR DE CD.	39
TABLA 3.3 PESOS DE LA CAPA OCULTA.	39
TABLA 3.4 PESOS DE LA CAPA DE SALIDA	39
TABLA 3.5 DESCRIPCIÓN DE LA ECUACIÓN 3.3.	42
TABLA 3.6 DESCRIPCIÓN DE LA ECUACIÓN 3.4.	43
TABLA 3.7 CONFIGURACIÓN DE LA COMUNICACIÓN SERIAL.	45
TABLA 3.8 DESCRIPCIÓN DE LOS BLOQUES DE LA FIGURA 3.13.	46
TABLA 3.9 DESCRIPCIÓN DE LOS BOTONES VIRTUALES DE LA FIGURA 3.16.	47
TABLA 3.10 DESCRIPCIÓN DE LOS INSTRUMENTOS VIRTUALES DE LA FIGURA 3.19.	52
TABLA 4.1 COMPARATIVA PRUEBA ESCALÓN.	55
TABLA 4.2 COMPARATIVA CAMBIO DE PUNTO DE AJUSTE.	57
TABLA 4.3 COMPARATIVA CAMBIO DE CARGA DE 1 N.M.	58

ABREVIATURAS

<i>CD</i>	Corriente Directa
<i>RN</i>	Red Neuronal
<i>PI</i>	Proporcional e Integral.
<i>DIC</i>	Control Inverso Directo (Direct Inverse Control*).
<i>NNDIC</i>	Controlador Inverso Directo basado en Redes Neuronales (Neural Network Direct Inverse Controller*).
<i>HIL</i>	Hardware dentro del lazo de simulación (Hardware in the Loop*)
<i>LVK</i>	Ley de Tensión de Kirchoff
<i>VCD</i>	Tensión de Corriente Directa

*Por sus siglas en inglés.

CAPÍTULO 1:

INTRODUCCIÓN

1.1 PRELIMINARES

En este capítulo se presenta la introducción general del trabajo, se menciona la importancia de las simulaciones en tiempo real, y se explica brevemente el tipo de controladores diseñados para el caso de estudio. Se muestra un panorama internacional de trabajos similares así como los realizados en la Sección de Estudios de Posgrado e Investigación de Ingeniería Eléctrica Unidad Zacatenco del Instituto Politécnico Nacional.

1.2 INTRODUCCIÓN

Un algoritmo de control clásico depende en gran medida del conocimiento preciso del comportamiento del sistema que se desea controlar. De la misma manera, su aplicación se ve limitada a sistemas lineales e invariantes en el tiempo [1, 2, 3].

Una propuesta alternativa es el uso de algoritmos de control inteligente, basados en la inteligencia artificial informática, donde se puede tomar ventaja de resultados experimentales de los sistemas, así como la experiencia de los operadores con el fin de diseñar un sistema inteligente capaz de controlar variables de sistemas complejos [4,5].

La principal desventaja en el diseño de controladores inteligentes consiste en la validación de estos, es decir, demostrar y garantizar su funcionamiento en las condiciones de operación requeridas [6, 7].

La tecnología actual permite desarrollar algoritmos inteligentes de manera satisfactoria en amplios campos de estudio. Las simulaciones son cada vez más precisas y exactas, y los sistemas digitales son capaces de llevar a cabo los cálculos complejos de los algoritmos inteligentes [8, 9, 10].

En este trabajo se implementa un sistema de control basado en el uso de las redes neuronales artificiales (RN) la cual consiste en la emulación de redes neuronales biológicas con el fin de reconocer patrones y desarrollar un nivel primitivo de inteligencia artificial [4, 6].

Dentro de las variedad de aplicaciones de las RN, se encuentra el ajuste de funciones, que consiste en presentar a la RN mediante un entrenamiento supervisado, un conjunto

de patrones entrada-salida para que la red se comporte de igual manera que la función que se desea aproximar o ajustar.

Esta capacidad de las RN es aprovechada en la identificación de sistemas y el modelado matemático, donde a partir de datos experimentales se obtienen modelos matemáticos precisos.

En el control automático se utiliza la capacidad de las RN de ajuste de funciones con el fin de aproximar el comportamiento de la función requerida para llevar un sistema al estado deseado de operación [5].

Por lo general dicha función es desconocida, por lo que se propone [11] mapear la dinámica inversa de la planta con el fin de anular la dinámica de esta y obtener así una respuesta del sistema igual al del valor deseado.

La principal ventaja de aproximar la función que representa la dinámica inversa de un sistema radica en la obtención de un sistema de control teóricamente ideal, es decir, un controlador que, puesto en serie con el sistema a controlar, anula la dinámica con la inversa de la dinámica, logrando una respuesta exactamente igual a la deseada, sin oscilaciones ni etapa transitoria. En la práctica, si existe la dinámica inversa es posible únicamente acercarse a su comportamiento. Sin embargo, una buena aproximación de la dinámica inversa produce excelentes controladores que superan a los algoritmos de control clásicos [11].

La implementación digital de las RN se vio detenida muchos años debido al nivel tecnológico con el que se contaba en sus inicios. El desempeño de las RN está directamente relacionado con el procesamiento de cálculos en paralelo, característica que les permite ser algoritmos poderosos pero con un alto coste computacional [4, 10, 11].

Los sistemas digitales de la actualidad, tales como los sistemas embebidos, cuentan con la capacidad suficiente para ejecutar una RN de manera adecuada.

1.3 PLANTEAMIENTO DEL PROBLEMA

El diseño de sistemas de control es un área extensa que va desde el diseño de algoritmos que regulen variables de un sistema, hasta la implementación y ejecución del mismo. Es debido a esto que para garantizar un diseño exitoso, el ingeniero de control debe conocer el sistema que se propone controlar, además de tener los conocimientos para realizar simulaciones e implementaciones digitales [3].

En el presente trabajo se lleva a cabo un análisis del caso de estudio, Motor de Corriente Directa, se diseñan sistemas de control convencional e inteligente para regular la velocidad del rotor, y se implementan en un sistema embebido digital.

El problema principal en la implementación digital de los sistemas de control es la comprobación del desempeño buscado. Las simulaciones digitales que incluyen la ejecución del hardware deben realizarse en tiempo real con el fin de validar el diseño en tiempo discreto. Estas simulaciones se conocen como HIL (Hardware in the Loop),

y permiten verificar en tiempo real el desempeño del sistema de control. La figura 1.1 muestra el esquema general de la simulación HIL propuesta.

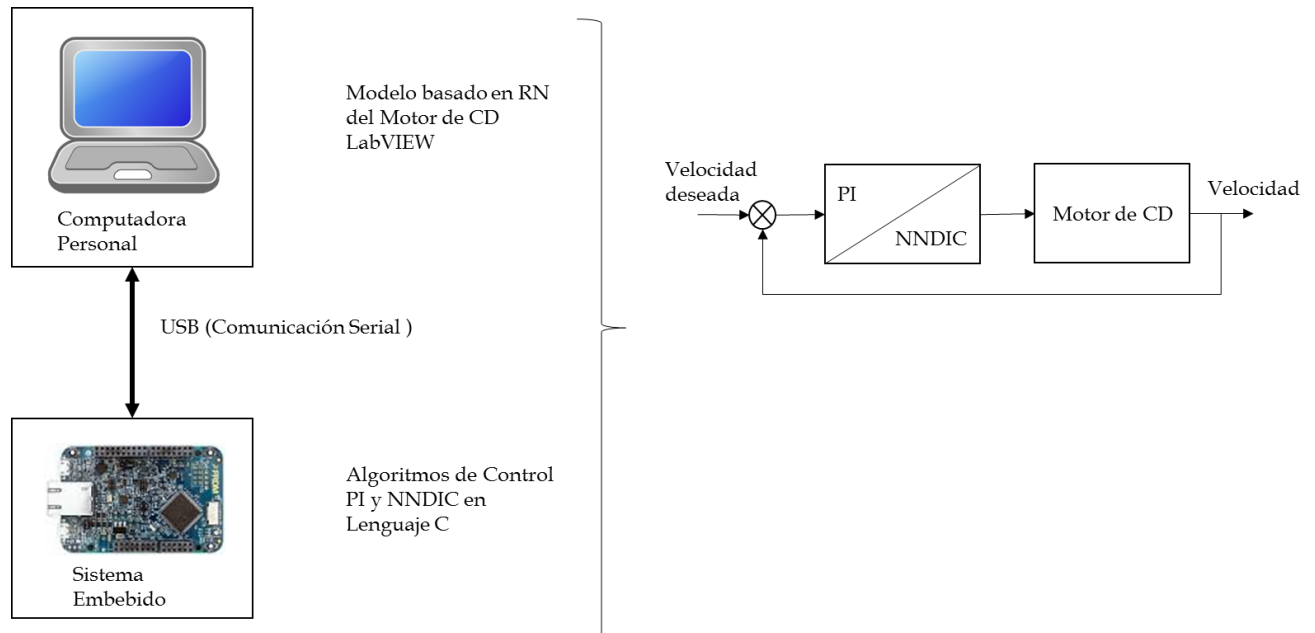


Figura 1.1 Esquema General de la simulación HIL

1.4 OBJETIVO

Diseñar un sistema de control de velocidad de un motor de CD, utilizando un algoritmo de control inteligente y uno convencional e implementarlos en una simulación HIL en tiempo real.

1.5 OBJETIVOS ESPECÍFICOS

- Realizar el modelo del caso de estudio Motor de Corriente Directa.
- Diseñar un control PI para controlar la velocidad del rotor.
- Diseñar un control NNDIC para controlar la velocidad del rotor.
- Entrenar una RN para identificar la dinámica del caso de estudio.
- Programar la RN que emula la dinámica del Motor de Corriente Directa en una simulación en tiempo real utilizando LabVIEW.
- Programar los controladores diseñados PI y NNDIC en el sistema embebido FRDM-K64F.
- Desarrollar la simulación HIL en tiempo real y en lazo cerrado, utilizando la RN que emula al motor de CD en LabVIEW y los controladores implementados en el sistema embebido.
- Realizar pruebas de desempeño de los sistemas de control.

1.5 JUSTIFICACIÓN

Los algoritmos de Control Inteligente, en específico los basados en RN, son poderosas herramientas de identificación y control. La capacidad de las RN de reconocer patrones permite al Sistema de Control Inteligente identificar la dinámica del sistema a controlar y proponer una acción de control acorde a su entrenamiento.

Como el objetivo de la RN es reproducir la dinámica inversa del sistema a controlar, el resultado del control es la aproximación al controlador ideal de cero oscilaciones, es decir, la dinámica inversa se anula con la dinámica del sistema y la respuesta es igual al valor deseado de operación.

El uso de las simulaciones HIL, comprueba los resultados simulados del diseño de algoritmos embebidos en hardware. La herramienta de LabVIEW permite realizar simulaciones en tiempo real y comunicación directa con hardware, dando como resultado una simulación HIL en tiempo real.

El Sistema Embebido utilizado es el FRDM-K64F de la compañía Freescale, un microcontrolador de 32 Bits y 120 MHz muy poderoso y económico, suficiente para manejar los algoritmos de control convencionales y la RN.

1.6 LIMITACIONES Y ALCANCES

Las limitaciones del trabajo se resumen en los siguientes puntos.

MOTOR DE CD

- Se considera un modelo lineal del motor de CD.

SIMULACIÓN HIL

- Se utiliza el protocolo de comunicación RS232.
- Se utiliza el modelo del motor de CD basado en RN.

SISTEMA DE CONTROL NEURONAL

- Se considera la dinámica inversa del sistema como patrón de control.
- No se realizó ningún análisis de estabilidad.

Los puntos no considerados anteriormente representan las principales limitaciones del trabajo, que pueden ser explorados en trabajos futuros.

Los alcances resumidos logrados a lo largo de la elaboración del trabajo y las pruebas experimentales son:

- Un modelo lineal de un motor de CD de laboratorio.

- Un controlador PI y un controlador NNDIC implementados en lenguaje C dentro del sistema embebido FRDM-K64F.
- Un simulador HIL programado en LabVIEW capaz de operar en tiempo real y ejecutar modelos matemáticos basados en RN.

1.7 ESTADO DEL ARTE

1.7.1 Trabajos Desarrollados a Nivel Internacional

Las RN han sido desarrolladas a nivel internacional como algoritmos de identificación de patrones desde la década de 1960 [4], sin embargo, la aplicación al control automático de sistemas dinámicos es relativamente reciente.

En el año de 1988, se presentaron diversos artículos en la revista de Sistemas de Control de la IEEE, entre ellos, la introducción de las redes neuronales como algoritmos de control inteligente [37], en la Universidad de California.

En ese mismo año, en la Universidad McMaster en Ontario Canadá, se efectuó un controlador para un sistema de molino, basado en redes neuronales [38].

A partir de la década de 1990, las redes neuronales se orientaron al control de sistemas más complejos y de reciente aparición: los robots manipuladores industriales [39].

En el año 2000, se desarrolló en la Universidad de Glasgow, Escocia, un control neuronal basado en la dinámica inversa de un molino de laminado, el cual fue comparado con un controlador PI, obteniendo mejores resultados [40].

De la misma manera, en el año 2000, un controlador neuronal basado en la dinámica inversa de un motor de CD fue desarrollado en la Universidad de Galati en Rumania [41] y en el año 2002, un estabilizador de un sistema eléctrico de potencia basado en un neurocontrolador de dinámica inversa fue implementado en la Universidad del Estado de Pensilvania, EU [42].

A partir del año 2005 la investigación mundial dio un giro debido a la creciente tecnología de los sistemas embebidos, por lo que más tipos de sistemas de control inteligente se comenzaron a implementar. En 2006, la Universidad de Hong Kong China, desarrollo un sistema de control inteligente adaptable para el sistema de regulación del tráfico vehicular de la ciudad de Hong Kong [43].

Para el año 2010, los sistemas de simulación en tiempo real utilizando Hardware se convirtieron en tendencia, en la Universidad de Lyngby, Dinamarca, se desarrolló un sistema HIL de pruebas en lazo cerrado para sistemas de electrónica de potencia [44], y en el año 2011, en la Universidad de Xiang, China, se desarrolló un sistema HIL para pruebas en control de motores síncronos [44].

En 2012, la Universidad de Teesside, en el Reino Unido, desarrollo un sistema HIL para la verificación y validación de sistemas de control en sistemas embebidos [44].

En 2014, la Universidad de Maryland, EU, desarrollo una metodología para la validación y verificación de operaciones en lazo cerrado y las simulaciones en tiempo real que involucren hardware [45].

1.7.2 Trabajos Desarrollados en la S.E.P.I.-E.S.I.M.E. ZAC.

A continuación se mencionan algunos trabajos desarrollados en la Sección de Estudios de Posgrado e Investigación de la Escuela Superior de Ingeniería Mecánica y Eléctrica (S.E.P.I.-E.S.I.M.E).

En 2005, se desarrolló un regulador de tensión de un generador utilizando una Red Neocognitron [46], que identifica la dinámica inversa del modelo del generador y propone acciones de control acorde a las simulaciones.

En 2006, se desarrolló un controlador neuronal para regular el voltaje en un aerogenerador [47].

En 2008, se utiliza una tarjeta de prototipos rápidos para implementar un algoritmo de control vectorial de un motor de inducción [48].

En 2013, se diseña un sistema embebido que contiene algoritmos de control para la regulación de velocidad de motores de inducción [49], en ese mismo año, se implementó un regulador de tensión de un generador de laboratorio, utilizando un sistema ANFIS, que identifica la dinámica inversa del generador [50].

1.8 APORTACIONES

Las aportaciones derivadas del trabajo son:

- Una metodología para el diseño de controladores neuronales NNDIC.
- Una metodología para la identificación de Sistemas utilizando RN y su implementación en simulaciones digitales en tiempo real.
- Una simulación HIL y la metodología para simular sistemas en tiempo real utilizando LabVIEW.
- Una metodología para implementar RN en microcontroladores utilizando lenguaje C.
- Una comparativa entre Controladores Clásicos e Inteligentes aplicada a un caso de estudio.

1.9 ESTRUCTURA DE LA TESIS

Capítulo 1. Contiene una introducción y resumen del trabajo. Se mencionan los alcances y limitaciones, además de los trabajos relevantes a nivel internacional y a nivel de la SEPI Zacatenco.

Capítulo 2. En este capítulo se desarrolla el modelo del motor de CD, se diseñan los controladores PI (*Proporcional Integral*) y NNDIC (*Controlador por Dinámica Inversa Neuronal*) para regular la velocidad del modelo. Se hace un análisis de los diseños y una comparativa.

Capítulo 3. Se desarrolla el simulador en tiempo real HIL (*Hardware in the Loop*). Se describe la identificación del motor de CD a un modelo basado en RN (*Red Neuronal*), la programación del simulador en LabVIEW y la codificación de los algoritmos de control en el sistema embebido.

Capítulo 4. Este capítulo muestra una serie de pruebas a ambos sistemas de control utilizando el simulador desarrollado HIL (*Hardware in the Loop*), así como indicadores de desempeño de control como auxiliar en la comparativa del desempeño de los controladores PI y NNDIC.

Capítulo 5. Se presenta la conclusión del trabajo así como el resultado final de los índices de desempeño de los controladores. Se mencionan las aportaciones y las recomendaciones generales para el diseño de controladores con RN. Se recomiendan tópicos y se detallan problemáticas para trabajos futuros.

Apéndice A. Contiene los datos de placa del motor de CD utilizado como caso de estudio, así como los parámetros utilizados en el desarrollo del modelo.

Apéndice B. Contiene fundamentos de la teoría de las Redes Neuronales Artificiales, fundamentos del algoritmo de Control, así como la descripción de la metodología del control neuronal inverso.

Apéndice C. Se detallan los algoritmos utilizados en el cálculo de los índices de desempeño en tiempo continuo y discreto.

Apéndice D. Incluye los códigos implementados en el sistema embebido.

Apéndice E. Desarrollo del simulador en tiempo real de LabVIEW.

CAPÍTULO 2:

DESARROLLO DEL CONTROL DE VELOCIDAD DEL MOTOR DE CD.

2.1 INTRODUCCIÓN

En este capítulo se describe el modelo del motor de Corriente Directa, analizando la dinámica eléctrica y mecánica que describe su funcionamiento, así como el diseño de los sistemas de control propuestos (PI y NNDIC).

2.2 MÁQUINA DE CORRIENTE DIRECTA

La máquina de corriente directa, en su funcionamiento como motor, transforma energía eléctrica en energía mecánica. Un motor de CD básicamente es un maquina eléctrica que transforma corriente directa en par electromecánico [12, 13, 14,15].

La configuración utilizada en la tesis es la de un motor de CD con excitación separada, es decir, la alimentación de corriente directa de la armadura y del campo provienen de diferentes fuentes de alimentación [13].

La regulación de la velocidad del rotor, la variable de estudio de la tesis, se realizó utilizando el método del control de velocidad de armadura, el cual consiste en variar la velocidad variando la alimentación de tensión eléctrica al circuito de armadura.

El motor que se analiza a lo largo del trabajo es un micro máquina de laboratorio de 400 rad/s de velocidad nominal y 220VCD de alimentación de armadura. Los datos relevantes de la máquina se encuentran en el apéndice A.

2.2.1 Modelado Matemático del Motor de CD.

La Tabla 2.1 contiene los parámetros principales para el modelado de la dinámica del motor de CD utilizado como caso de estudio.

Tabla 2.1 Parámetros del Motor de CD.

Parámetro	Nombre	Valor	Unidades
Resistencia de Armadura	R_a	2.50	$[\Omega]$
Inductancia de Armadura	L_a	0.100	$[H]$
Constante de Fricción Viscosa	b	0.01	$[N.m.s]$
Constante de Fuerza Electromotriz	k_e	0.5	$[V/rad/s]$
Momento de Inercia	J	0.0022	$[Kg.m^2]$
Constante de Par	k_t	0.5	$[N.m/A]$

El modelo completo del motor de CD consta de dos ecuaciones diferenciales que representan la dinámica eléctrica y mecánica. Para determinar estas ecuaciones, se utilizó el esquema de la Figura 2.1, donde se muestra el circuito equivalente de armadura, así como un diagrama de cuerpo libre del rotor acoplado a la carga mecánica.

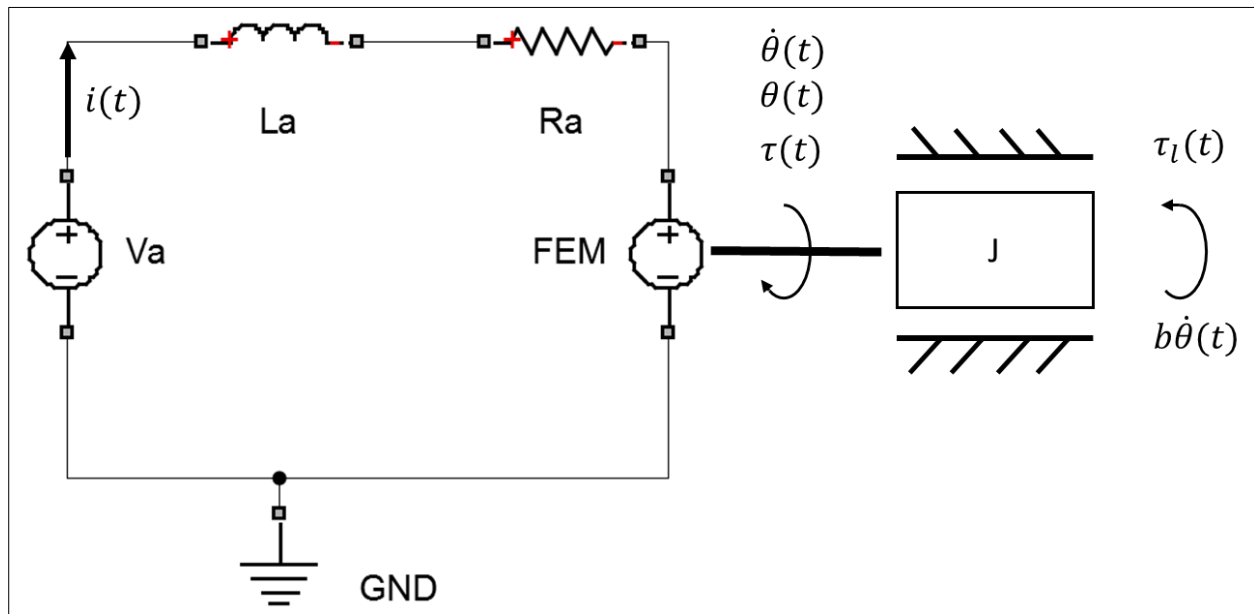


Figura 2.1 Esquema General de las partes eléctrica y mecánica del Motor [16].

La Tabla 2.2 contiene las variables mostradas en la Figura 2.1, así como su descripción.

Tabla 2.2 Descripción de las variables de la Figura 2.1.

Variable	Descripción
$i(t)$	Corriente de Armadura
V_a	Tensión de Armadura
L_a	Inductancia de Armadura
R_a	Resistencia de Armadura
FEM	Fuerza Contra Electromotriz
GND	Referencia Eléctrica
$\theta(t)$	Posición de la flecha, Angulo
$\dot{\theta}(t)$	Velocidad Angular
$\tau(t)$	Par Mecánico
J	Momento de Inercia del Rotor
$\tau_l(t)$	Par de Carga
b	Constante de Fricción Viscosa

Para la ecuación diferencial eléctrica, utilizando el circuito de armadura mostrado en la figura 2.1, y aplicando la LVK, se obtuvo la ecuación 2.1.

$$L_a \frac{di(t)}{dt} + R_a i(t) = V_a - FEM \quad (2.1)$$

El par del motor de CD es proporcional a la corriente de armadura y la fuerza del campo magnético. Ya que se supone un campo constante [12, 13], el par es proporcional a la corriente de armadura multiplicado por la constante de par, mostrado en la ecuación 2.2.

$$\tau(t) = k_t * i(t) \quad (2.2)$$

La fuerza contra electromotriz es proporcional a la velocidad angular multiplicada por la constante de fuerza contra electromotriz [13], como se aprecia en la ecuación 2.3.

$$FEM = \dot{\theta} * k_e \quad (2.3)$$

Las constantes del motor de CD, son iguales en el sistema internacional de unidades [12, 13, 14, 15], y se representan como una sola constante, denominada K , de la ecuación 2.4.

$$K = K_T = K_e \quad (2.4)$$

Sustituyendo las ecuaciones 2.3, y 2.4 en 2.1, se obtiene la ecuación diferencial eléctrica del modelo siguiente:

$$L_a \frac{di(t)}{dt} + R_a i(t) = V_a - \theta K \quad (2.5)$$

La ecuación 2.5, representa la ecuación diferencial eléctrica.

Para la ecuación diferencial mecánica, utilizando el diagrama de cuerpo libre de la Figura 2.1, realizando un balance de fuerzas y basándose en la segunda ley de Newton para fuerzas angulares [16], se obtiene la expresión 2.6.

$$\tau(t) - \tau_l(t) = J\ddot{\theta} + b\dot{\theta} \quad (2.6)$$

Donde $\ddot{\theta}$ de la ecuación 2.6 representa la aceleración angular.

Finalmente, utilizando la transformada de Laplace en las ecuaciones 2.5 y 2.6, se tiene:

$$\text{Circuito de Armadura} \quad I(s) = \frac{1}{L_a s + R_a} (V_a - \dot{\theta} K) \quad (2.7)$$

$$\text{Carga Mecánica} \quad \dot{\theta}(s) = \frac{1}{J s + b} (\tau(s) - \tau_l(s)) \quad (2.8)$$

Utilizando las ecuaciones 2.7 y 2.8, se creó el siguiente diagrama de simulación en Simulink [17, 18], mostrado en la figura 2.2.

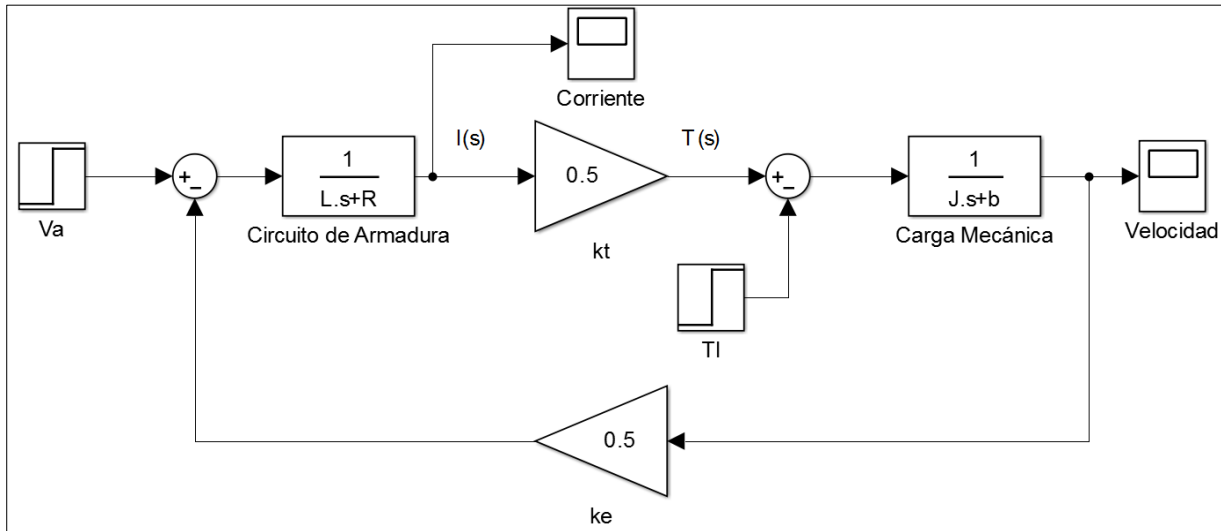


Figura 2.2 Diagrama de Simulación Motor de CD.

El diagrama de simulación de la Figura 2.2, permite visualizar la velocidad del motor ante cambios en la tensión de armadura, además, permite medir la corriente de

armadura y realizar cambios de carga en la entrada de par de carga para observar los efectos en la velocidad y en la corriente.

La función de transferencia del motor se obtuvo sustituyendo la ecuación 2.7 en 2.8, tomando como entrada la tensión de armadura y la salida la velocidad angular, con el fin de tener un modelo útil para el diseño del sistema de control PI. La función de transferencia se muestra en la ecuación 2.9.

$$Motor = \frac{\dot{\theta}(s)}{V_a} = \frac{K}{(L_a s + R_a)(Js + b) + K^2} \quad (2.9)$$

Sustituyendo los valores de la tabla 2.1 en la función de transferencia de la ecuación 2.9, se tiene:

$$\frac{\dot{\theta}(s)}{V_a} = \frac{0.5}{0.00022s^2 + 0.0065s + 0.275} \quad (2.10)$$

La función de transferencia de la ecuación 2.10 es el modelo lineal que se utiliza en el diseño del control PI.

2.2.2 Análisis del Motor de CD en lazo abierto

Las pruebas realizadas al diagrama de simulación de la figura 2.2, consisten en cambios en la tensión de armadura, y en el par de carga.

La figura 2.3, muestra la velocidad angular del motor con una entrada de 1 volt en el circuito de armadura, produciendo una velocidad en estado estacionario de 1.82 rad/s.

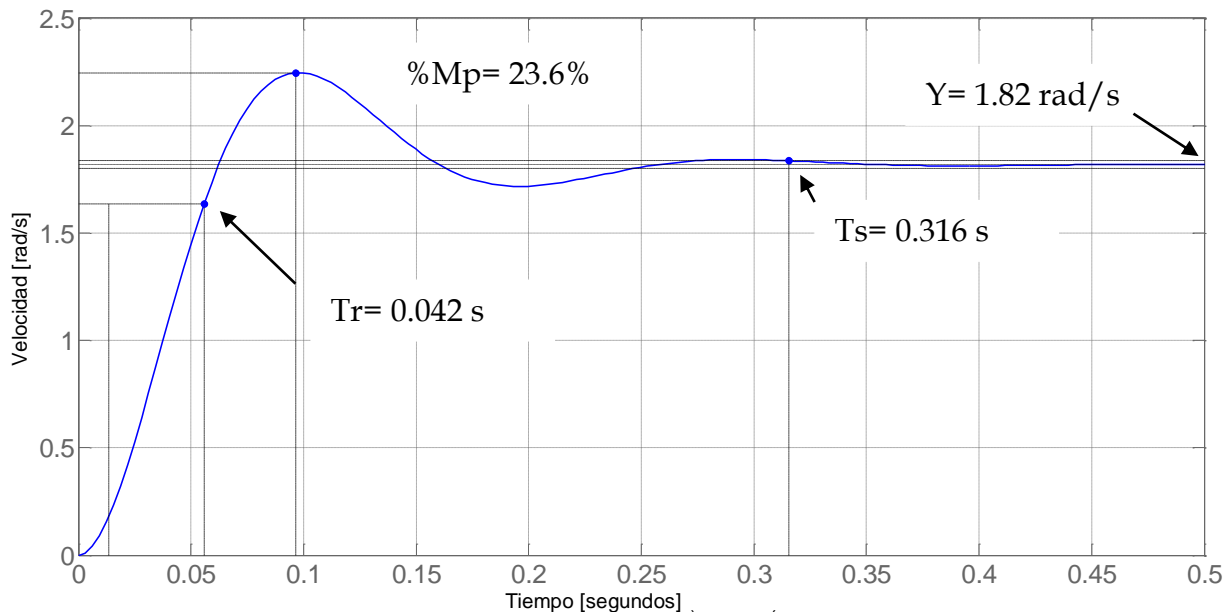


Figura 2.3 Respuesta a una entrada de 1 VCD sin carga.

Las características de la respuesta mostrada en la figura 2.3, se resumen en la tabla 2.3, donde se indica el criterio del tiempo de subida (0-90%) y el del tiempo de asentamiento (1%) [1, 2, 3].

Tabla 2.3 Características de la respuesta de la figura 2.3.

Característica	Nombre en la Figura	Valor
Tiempo de Subida	Tr	0.042 s (0-90% de la salida)
Sobre Impulso	%Mp	23.6%
Tiempo de Asentamiento	Ts	0.316 s (1%)
Valor Estado Estacionario	Y	1.82 rad/s

La siguiente prueba, consiste en el arranque del motor utilizando 1 volt (considerando una fuente de alimentación ideal), y la figura 2.4, muestra la respuesta de la corriente de armadura.

Puede observarse de la figura 2.4, que la corriente alcanza un valor de 0.1087 A y después de la parte transitoria de la dinámica del motor, alcanza un valor estacionario de 0.0363 A.

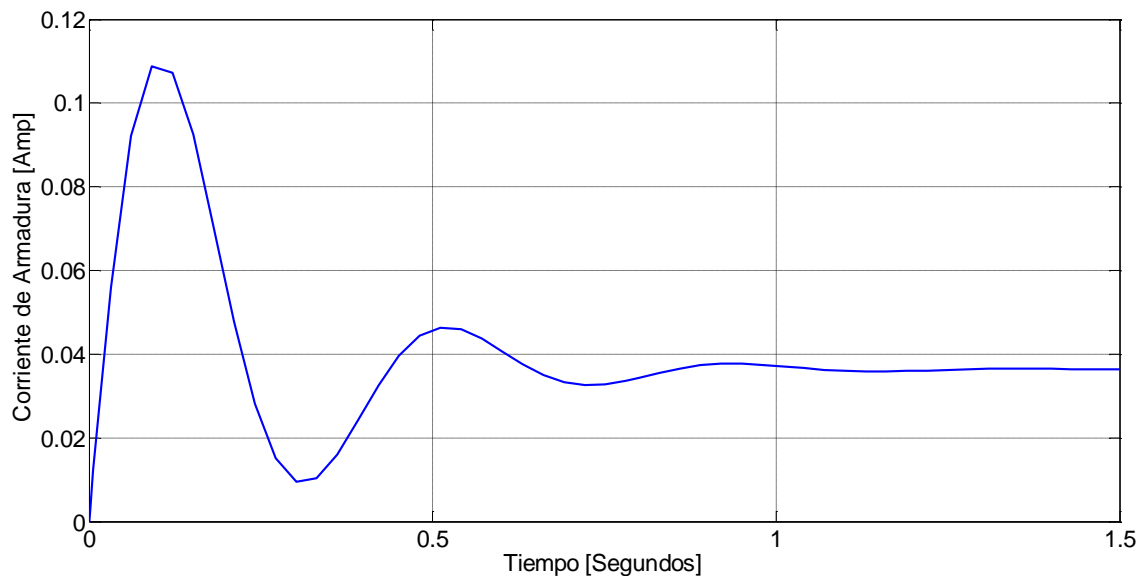


Figura 2.4 Corriente de Armadura entrada de 1 VCD.

El motor de CD funciona con una alimentación de armadura de 220 VCD nominal. Este valor de tensión representa aproximadamente una velocidad de 400 rad/s o 3820 rpm. La corriente nominal a esta tensión y sin carga mecánica en la flecha es de aproximadamente 8 A. La Figura 2.5, muestra la respuesta de la velocidad a una entrada de tensión de 220 VCD.

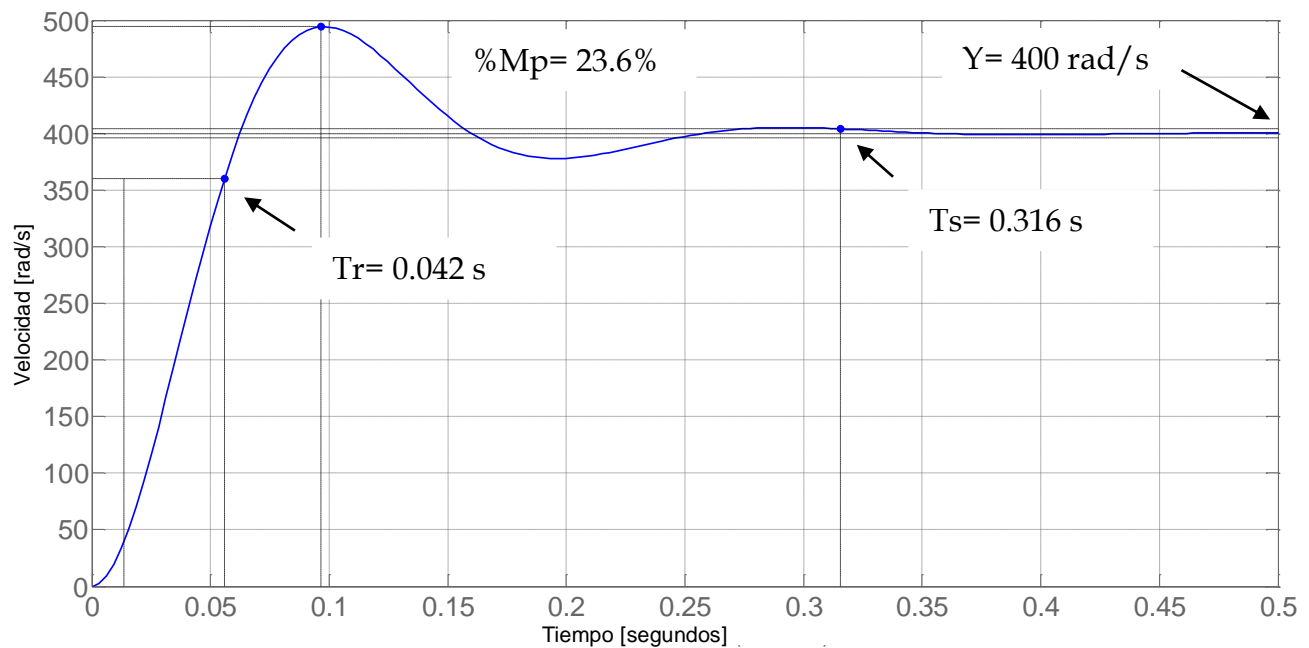


Figura 2.5 Respuesta a una entrada de 220 VCD sin carga.

Como puede observarse en las figuras 2.3 y 2.5, la forma de la respuesta a las entradas de tensión suministradas es idéntica debido a que se trata de un sistema lineal. Lo mismo pasa con la respuesta de la corriente, sin embargo, los valores de la corriente al arranque con la entrada de 220 VCD son muy grandes, como se muestra en la figura 2.6, la corriente alcanza un valor de 24.2 A, suficiente para causar daños al motor o a los componentes eléctricos y electrónicos que componen su funcionamiento. Para alcanzar la velocidad nominal, se requiere de un arrancador o de algún tipo de arranque como puede ser una alimentación en rampa [9].

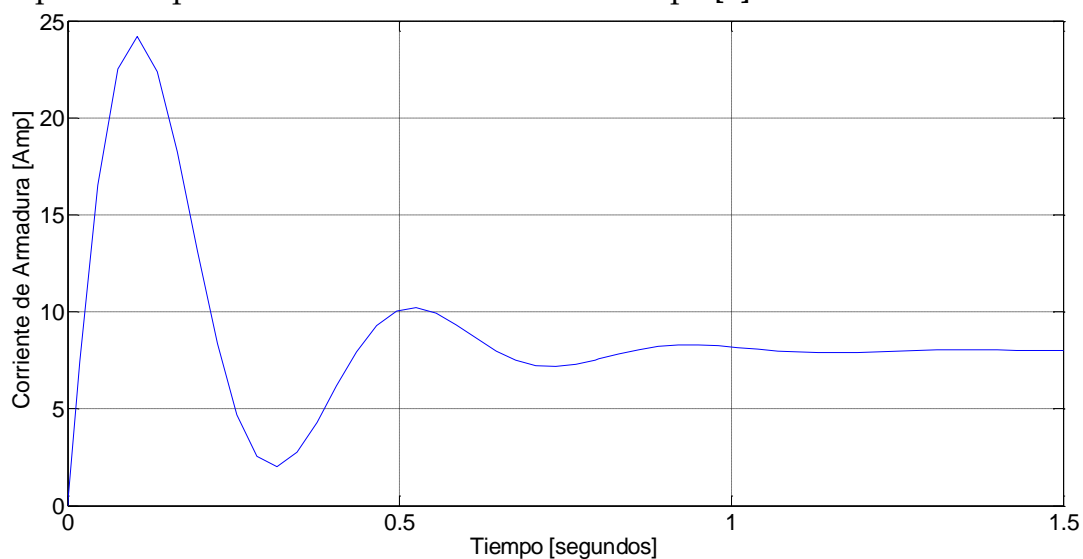


Figura 2.6 Corriente de Armadura entrada de 220VCD sin carga.

Para los fines del diseño de los sistemas de control, la entrada y salida del modelo serán normalizadas en un intervalo que va desde cero hasta uno, como se puede observar en la tabla 2.4.

Tabla 2.4 Intervalos Normalizados.

Intervalo Real	Intervalo Normalizado
Entrada 0 VCD hasta 220 VCD	0 a 1
Salida 0 $\frac{\text{rad}}{\text{s}}$ hasta 377 $\frac{\text{rad}}{\text{s}}$	0 a 1

Las ventajas de tener un sistema con sus valores normalizados son el mejor desempeño de la RNA actuando como controlador (NNDIC) [11] y como modelo para la simulación que se tratara en el capítulo 3.

Con el modelo desarrollado, las pruebas en lazo abierto muestran el comportamiento del motor de manera general y lineal. Las siguientes pruebas representan la dinámica del motor ante cambios de carga, es decir, aumento en el par de carga del modelo.

Los cambios de carga en el motor producen efectos tanto en la velocidad como en la corriente de armadura, en donde al aumentar dicha carga, la velocidad disminuye y la corriente aumenta. La figura 2.7 muestra la respuesta de velocidad y corriente ante un aumento en la carga de 1 N.m.

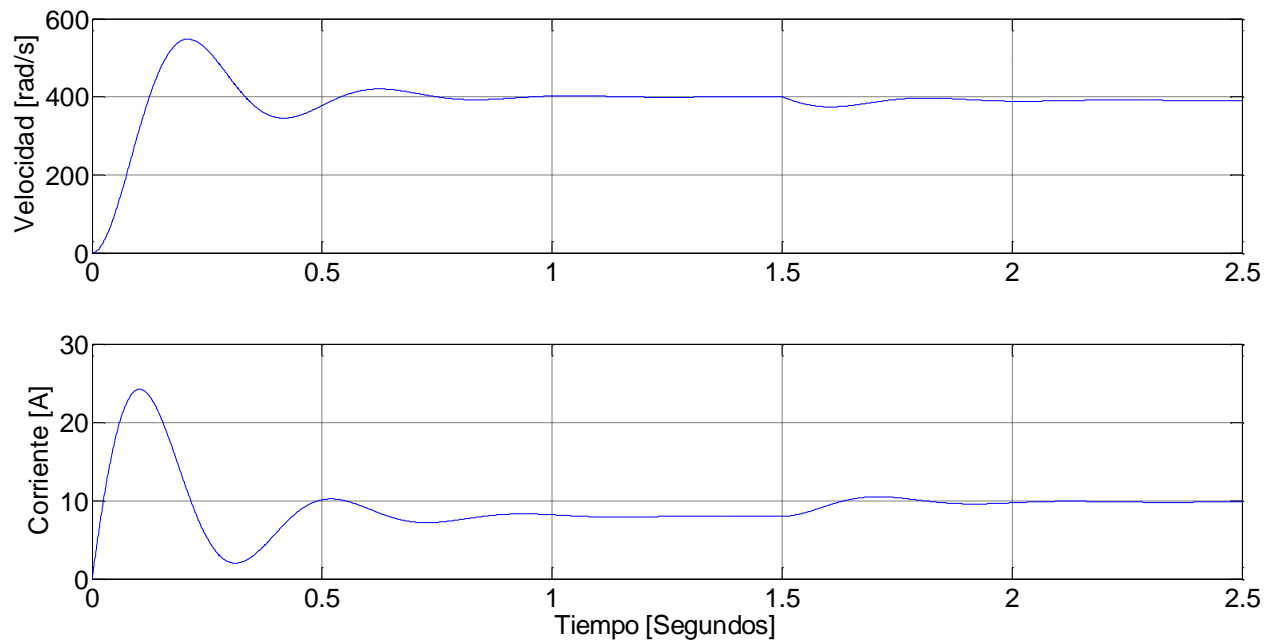


Figura 2.7 Cambio de carga de 1 N.m a los 1.5 segundos.

Como se observa en la Figura 2.7, la velocidad disminuye al momento de aplicarse la carga, a los 1.5 segundos de la simulación, en un valor de 10 rad/s. Así mismo la corriente aumenta a un valor de 9.8 A.

Como se puede observar en las pruebas en lazo abierto de la figura 2.7, la necesidad de un controlador es evidente para aplicaciones en donde se requiera un valor constante de la velocidad a pesar de los disturbios presentes originados por los cambios de carga [13, 16].

El modelo en función de transferencia de la ecuación 2.10, fue analizado para el diseño del controlador PI, descrito a continuación.

2.3 DISEÑO DEL SISTEMA DE CONTROL PI

Los parámetros de Control se resumen en la Tabla 2.5. El objetivo principal del diseño de un controlador para este tipo de dispositivos es la regulación de la velocidad a pesar de la presencia de disturbios, sin embargo, ciertas características de la respuesta transitoria deben cumplirse para obtener una dinámica aceptable.

Tabla 2.5 Requisitos del Sistema de Control [21].

Características	Valor en Lazo Abierto	Valor Deseado
Sobretiro	23.6%	<15%
Tiempo de Asentamiento	0.316 s	0.4 a 0.8 s
Tiempo de Subida	0.042 s	0.01 a 0.2 s

Además del cumplimiento de estos parámetros, la estabilidad del sistema de control debe estar garantizada, así como una salida saturada del controlador, es decir, que opere en el intervalo de tensión eléctrica del circuito de armadura del motor de CD (0 a 150 % del valor nominal de tensión).

El algoritmo de control PI consiste en la generación de una señal de control a partir de una señal de error del sistema. La señal de error es la diferencia entre el valor real de la salida a controlar y el valor deseado. La ecuación 2.11 representa la señal de control producida por el controlador PI en el tiempo.

$$u(t) = Kp[e(t)] + Ki \left[\int e(t)dt \right] \quad (2.11)$$

Donde:

$u(t) :=$ Señal de Control

$e(t) :=$ Señal de Error

$Kp :=$ Ganancia Proporcional

$Ki :=$ Ganancia Integral

El objetivo de sintonizar las ganancias del controlador PI es disponer de una señal de control que lleve al sistema al estado deseado de operación.

Existen diversos métodos para lograr encontrar la combinación de valores de las ganancias para cumplir los requerimientos del diseño.

Para el diseño del Sistema de Control PI, es decir, la sintonización de sus constantes, se utilizaron los siguientes métodos:

- Primer Método de Ziegler Nichols
- Herramienta de Auto Ajuste de MATLAB

2.3.1 Primer Método de Ziegler Nichols

Este método consiste en el análisis de la respuesta escalón unitario del sistema a controlar en lazo abierto. La selección de este método radica en la disposición de dicha gráfica además de ser un método que beneficia a sistemas cuyo modelo fue obtenido o no, de manera experimental [2, 19].

El análisis de la respuesta escalón consiste en trazar una línea tangente al punto de inflexión para conocer los valores de las constantes L y T mostradas en la figura 2.8, y descritas en la tabla 2.6.

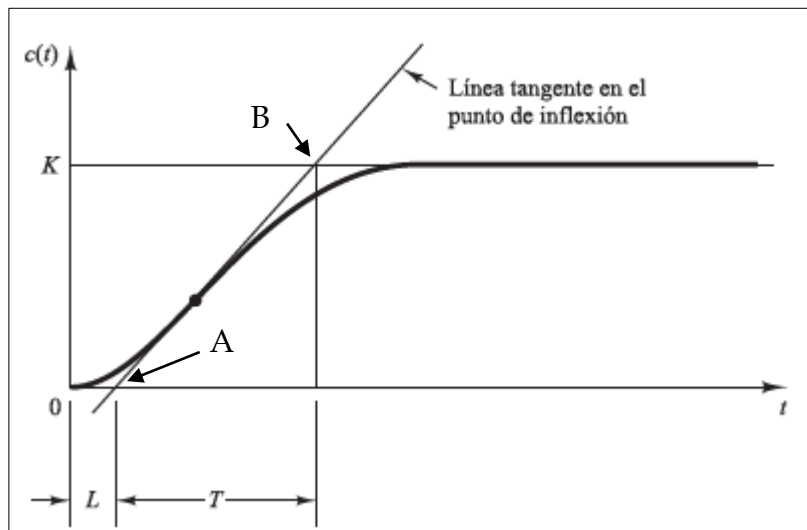


Figura 2.8 Curva Auxiliar del Primer Método de Ziegler Nichols.

Tabla 2.6 Descripción de los Parámetros de la figura 2.8.

Parámetro	Descripción
$C(t)$	Magnitud de la Respuesta.
t	Tiempo
K	Valor de la Respuesta en estado estacionario.
L	Intervalo entre cero segundos y el punto donde corta la línea tangente el eje del tiempo. Punto A en la figura 2.8.
T	Intervalo desde el punto A, hasta el punto de corte de la línea tangente en el valor K , Punto B en la figura 2.8.

Para conocer el punto de inflexión de la respuesta escalón del sistema, se utiliza la gráfica de la derivada. Se localiza el punto de inflexión, donde la pendiente de dicha gráfica cambia de signo, es decir, en el punto máximo. Su componente en el tiempo, representa el valor de tiempo en donde se encuentra el punto de inflexión, y la componente de magnitud en el punto máximo, representa el valor de la pendiente de la línea tangente.

La figura 2.9 contiene la gráfica de la derivada, donde el punto máximo marcado indica la información del punto de inflexión. En resumen, el componente x del punto máximo es el tiempo donde aparece el punto de inflexión y el componente y del punto máximo es la pendiente de la recta tangente.

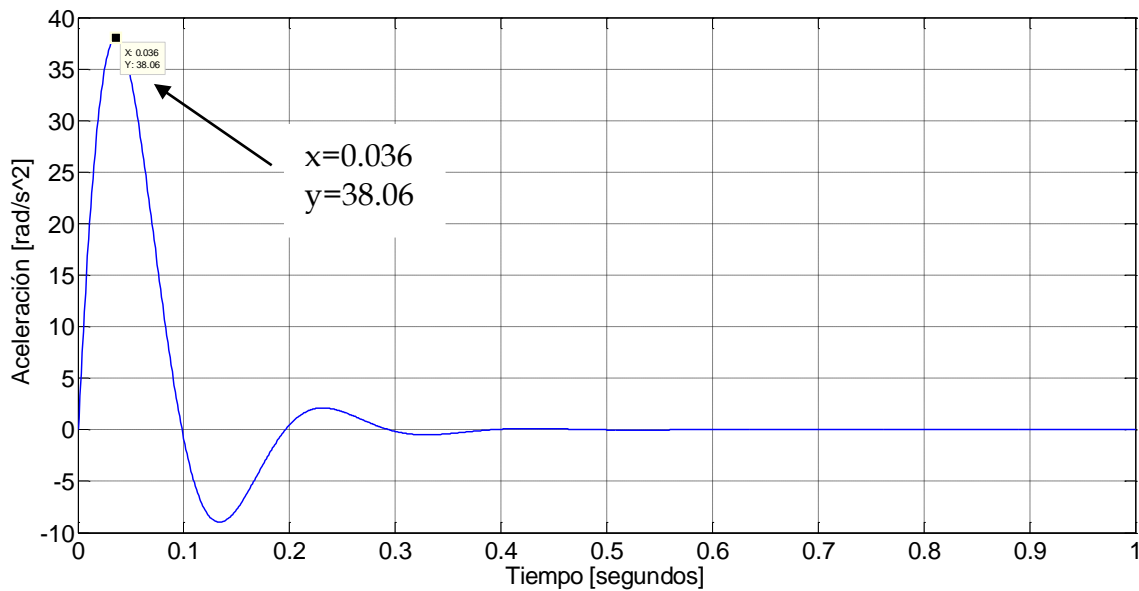


Figura 2.9 Derivada de la Respuesta Escalón del Motor de CD.

Ubicando el punto de inflexión en la respuesta de velocidad, se obtienen las coordenadas en la gráfica de la respuesta de la figura 2.10, la recta tangente a este punto se traza utilizando la información obtenida de la figura 2.9 y la ecuación de la recta, donde la pendiente es el valor de la componente y del punto marcado en la derivada.

La ecuación final de la recta tangente, utilizando el método de punto pendiente de la geometría básica se muestra en la ecuación 2.12.

$$y = 19.26x - 0.5163 \quad (2.12)$$

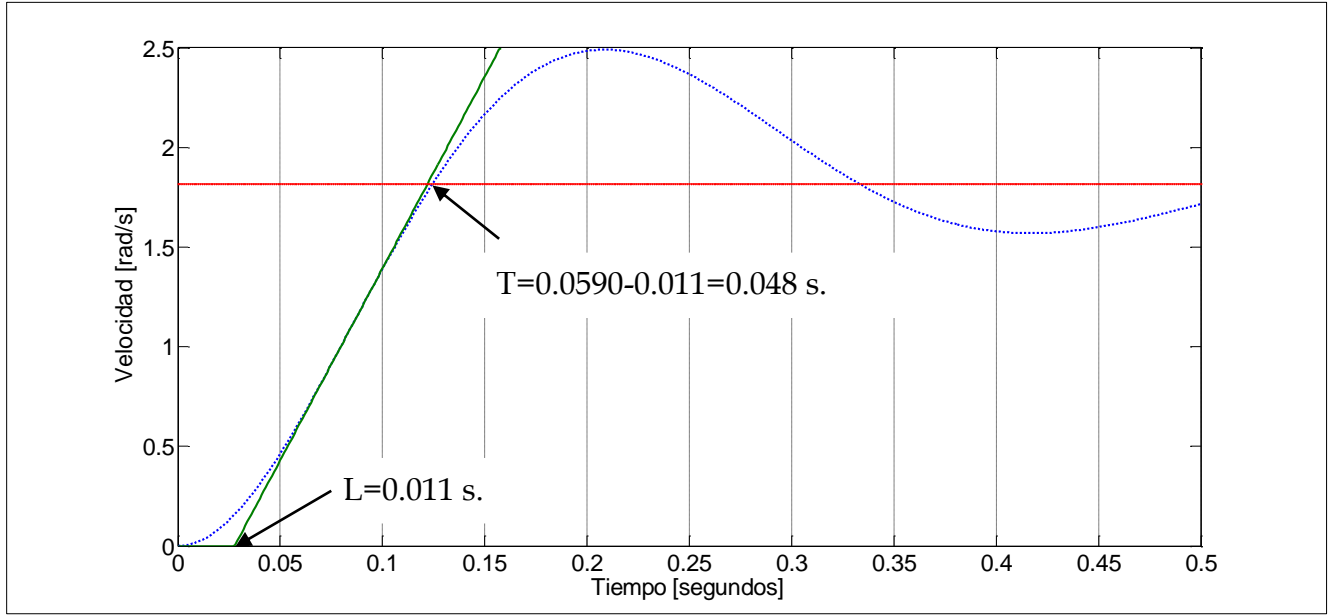


Figura 2.10 Respuesta a escalón unitario con línea tangente al punto de inflexión.

Al trazar la recta tangente al punto de inflexión sobre la respuesta escalón, se obtiene la gráfica de la figura 2.10 y la información requerida por el método. Ambos valores representan la información necesaria para utilizar la tabla 2.7, que contiene los valores de las ganancias K_p y K_i propuestas por el método de Ziegler Nichols.

Tabla 2.7 Sintonización de Controladores PI con el primer método de Ziegler Nichols [20].

Controlador	K_p	K_i
P	$\frac{T}{L}$	
PI	$0.9 \frac{T}{L}$	$\frac{L}{0.3}$

Con los valores obtenidos de la figura 2.10, se calculan las ganancias del controlador PI, ecuaciones 2.13 y 2.14, utilizando la tabla 2.7, se obtiene la siguiente función de transferencia de la ecuación 2.15, que representa al controlador PI sintonizado por este método.

$$K_p = 0.9 \left(\frac{0.048}{0.011} \right) = 3.94 \quad (2.13)$$

$$K_i = \frac{0.011}{0.3} = 0.036 \quad (2.14)$$

$$u(t) = 3.94[e(t)] + 0.036 \left[\int e(t) dt \right] \quad (2.15)$$

La respuesta del sistema de control PI sintonizado con el primer método de Ziegler Nichols se muestra en la figura 2.11.

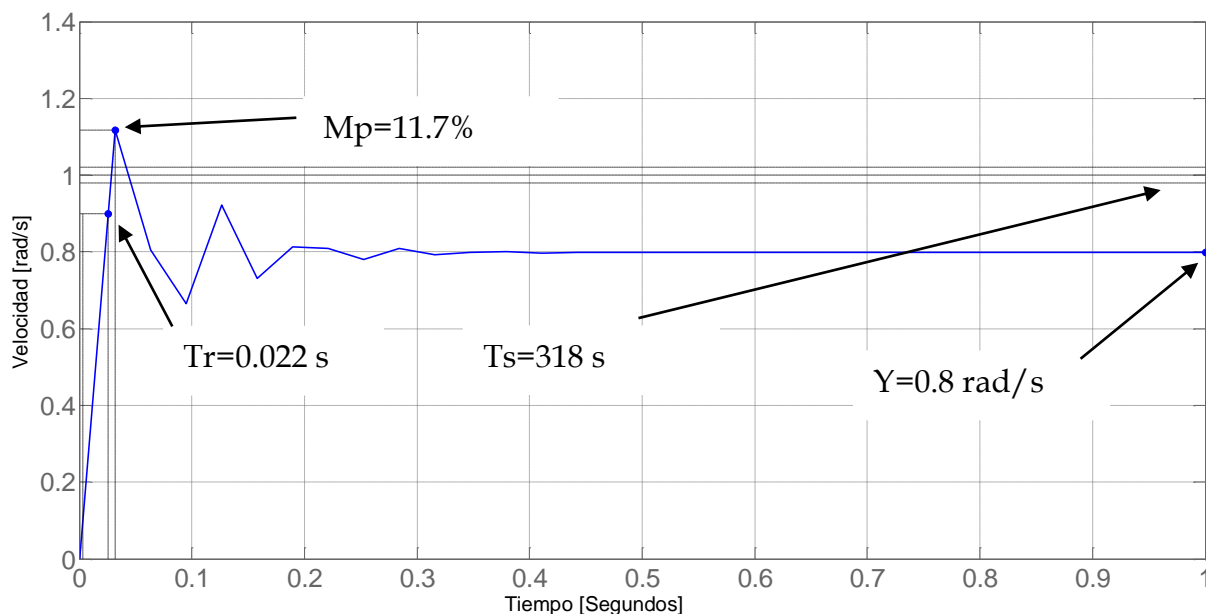


Figura 2.11 Sistema de Control PI sintonizado con el Primer Método de Ziegler Nichols

Como se menciona en la literatura [1, 2, 3, 20], las ganancias obtenidas con los métodos de sintonización de Ziegler Nichols representan un punto de partida únicamente, por lo que el proceso de sintonización pasa a una etapa experimental con el fin de obtener una respuesta deseada.

La sintonización fina se llevó a cabo en el entorno de Simulink [17], variando el valor de las ganancias con el fin de obtener una respuesta que cumpla todos los requisitos de la norma [21]. Debido a la presencia de un error en estado estacionario inaceptable, la ganancia integral fue incrementada a un valor de 23.014, la ganancia proporcional se redujo a un valor de 0.6213 para disminuir el sobretiro. Los resultados de la sintonización fina y los del Primer Método de Ziegler Nichols, se detallan en la tabla 2.8.

Tabla 2.8 Resultados del Sistema de Control PI Sintonizado.

Características	Primer Método de Ziegler Nichols	Sintonización Fina	Requisito
Sobretiro ($M_p\%$)	11.7%	3.11%	<15%
Tiempo de Asentamiento (T_s)	318 s	0.475 s	0.4 a 0.8 s
Tiempo de Subida (T_r)	0.022 s	0.078 s	0.01 a 0.2 s
Error en Estado Estacionario ($1-Y\%$)	20%	0%	$\pm 1\%$

La tabla 2.8 muestra que la sintonización fina cumple los requisitos establecidos por la norma [21], concluyendo así el diseño del controlador PI utilizando este método, muy efectivo y ampliamente utilizado en la industria [19, 22], incluso con la etapa de la sintonización experimental.

Finalmente, la figura 2.12 muestra la respuesta a un escalón unitario del sistema de control de velocidad PI para el motor de CD.

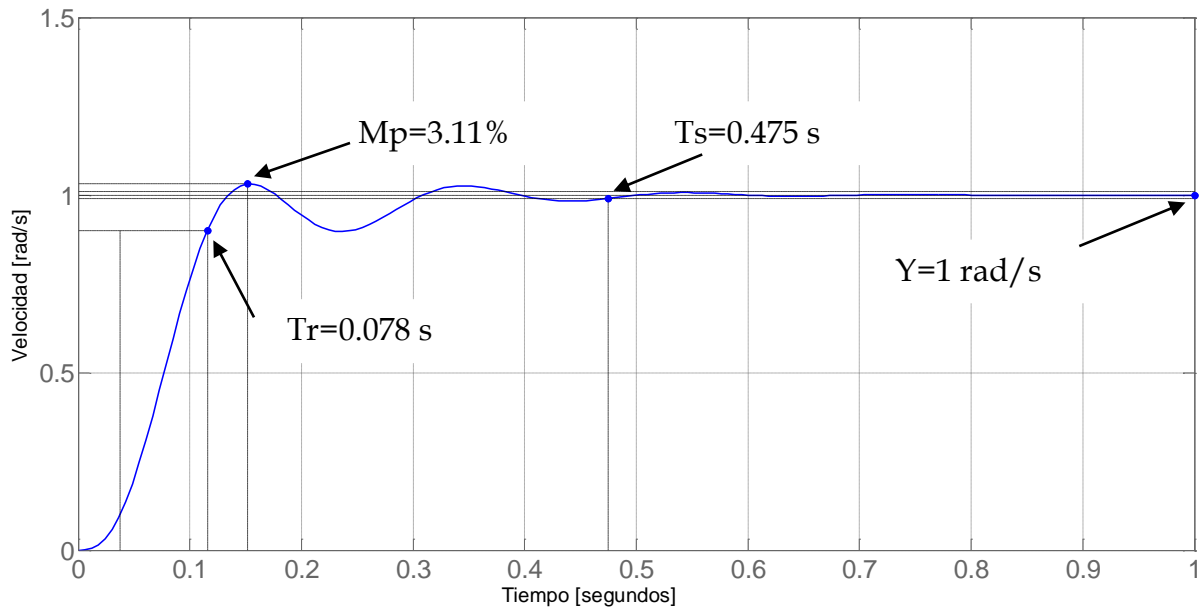


Figura 2.12 Sistema de Control PI con sintonización fina.

2.3.2 Herramienta de Auto Ajuste de MATLAB

Una manera de sintonizar los controladores convencionales es el uso de técnicas de optimización computacionales. La paquetería de MATLAB incluye una herramienta que consiste en una interface gráfica para el ajuste de controladores clásicos y un algoritmo de optimización que calcula las constantes para obtener la respuesta deseada [16, 17].

En la figura 2.13 se muestra el programa desarrollado en Simulink, que incluye el bloque de controlador PI (Recuadro Punteado), que contiene la herramienta de autoajuste de los parámetros del controlador.

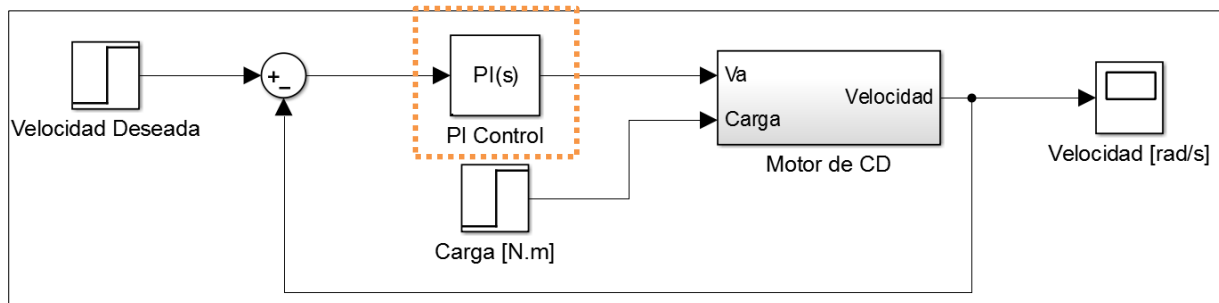


Figura 2.13 Modelo Simulink con Herramienta de Auto ajuste de Control PI.

Los objetivos del algoritmo de optimización de la respuesta del sistema de control, en términos de la propia herramienta de MATLAB, para la respuesta en el tiempo del sistema son:

- Tiempo de Respuesta (Más Rápido , Más Lento)
- Comportamiento Transitorio (Agresivo, Robusto)

La herramienta permite seleccionar una barra de desplazamiento horizontal, de los objetivos mencionados; Al variar estas opciones, la respuesta en la interfaz gráfica cambia, y las ganancias del controlador son calculadas de forma automática. La figura 2.14 muestra una captura de pantalla de la herramienta de selección de objetivos (Recuadro Punteado).

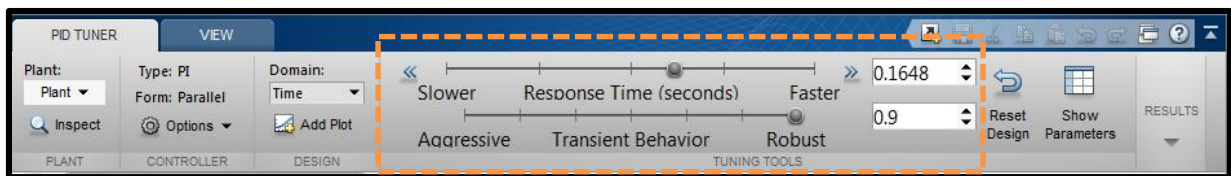


Figura 2.14 Herramienta de Auto Ajuste de Ganancias PI de MATLAB.

La figura 2.15 muestra la respuesta en el tiempo del sistema de control, utilizando la herramienta de Auto ajuste, donde se aprecia la respuesta del sistema antes y después del ajuste automático, la línea punteada representa el sistema antes del ajuste automático, y la línea gruesa el resultado del cálculo de la herramienta. La figura 2.14 y la figura 2.15 son tomas de pantalla de la herramienta de Matlab.

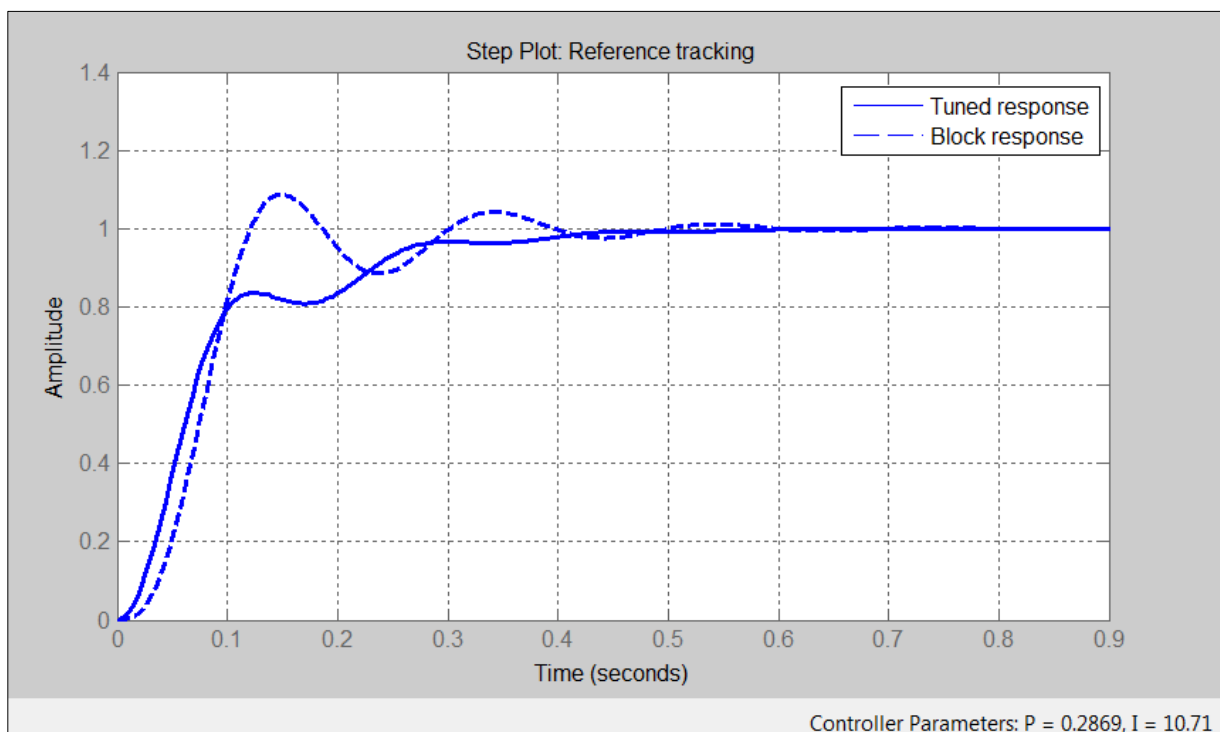


Figura 2.15 Resultados de la herramienta de sintonización automática.

Después de la ejecución de la herramienta de Auto ajuste, seleccionando un comportamiento robusto y no tan rápido, como se ve en el recuadro punteado de la figura 2.14, las ganancias del controlador PI, calculadas de manera automática por la herramienta fueron $k_p = 0.2869$ y $k_i = 10.71$.

Los resultados del sistema de control PI utilizando las ganancias de la herramienta se muestran en la tabla 2.9 junto con la sintonización fina con fines comparativos.

Tabla 2.9 Resultados de la sintonización fina y automática.

Características	Sintonización Fina	Sintonización Automática	Requisito
Sobretiro (Mp%)	3.11%	0%	<15%
Tiempo de Asentamiento (Ts)	0.475 s	0.404 s	0.4 a 0.8 s
Tiempo de Subida (Tr)	0.078 s	0.20 s	0.01 a 0.2 s
Error en Estado Estacionario (1-Y%)	0%	0%	$\pm 1\%$

Se puede observar de la tabla 2.9, que los requisitos se cumplen de forma adecuada para ambos métodos de sintonización.

2.3.3 Comparativa de los Controladores PI

Los dos métodos de sintonización mostraron ser efectivos en el cumplimiento de los requisitos del control, sin embargo, sus desempeños son diferentes y para elegir el mejor de los dos, se utilizan los índices de control ITAE, IAE e ISE, mostrados en el apéndice C [23]. La figura 2.16, contiene la respuesta en el tiempo a una entrada escalón con los controladores PI sintonizados con los dos métodos propuestos.

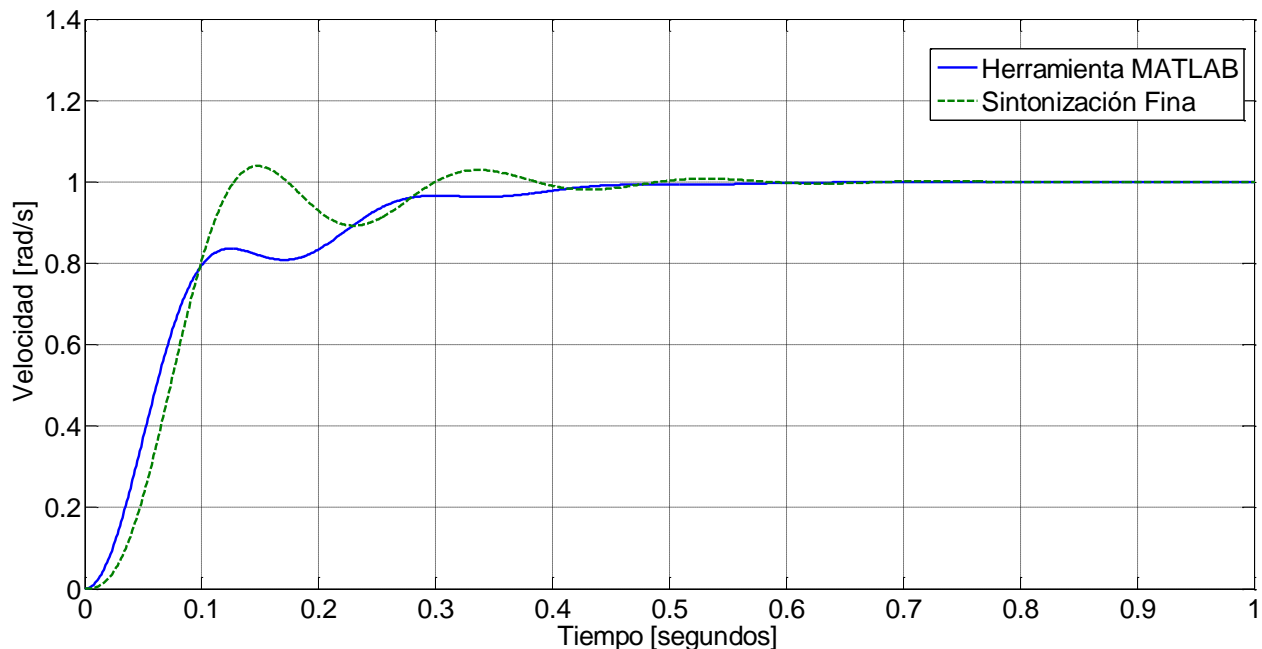


Figura 2.16 Respuesta en el tiempo de Sistema de Control PI con los dos métodos de sintonización.

La tabla 2.10 muestra los índices de desempeño de control mencionados, donde se puede comparar los resultados obtenidos de ambos métodos y determinar el mejor.

Las pruebas de los índices se realizaron con una simulación de un segundo con el tiempo de integración de un milisegundo, siendo un total de 1001 muestras.

Tabla 2.10 Índices de desempeño de los sistemas de control PI.

Índice de Desempeño	Sintonización Fina	Herramienta MATLAB
ITAE	0.0068	0.0090
IAE	0.0860	0.0933
ISE	0.0576	0.0502

De la tabla 2.10 se puede concluir que el desempeño general del controlador PI sintonizado con el primer método de Ziegler Nichols y luego ajustado manualmente, responde ligeramente de una mejor manera que el de la herramienta de MATLAB, sin embargo, el tiempo de diseño y la complejidad es sumamente mayor que al utilizar la herramienta.

2.4 DISEÑO DEL SISTEMA DE CONTROL NEURONAL

El control neuronal es un tipo de algoritmo de control de sistemas dinámicos que está basado en el uso de las redes neuronales [4, 5].

El objetivo principal de la teoría de control es encontrar una función en el tiempo que lleve una o unas variables de un sistema a un estado o estados deseados de operación, como se puede observar en la ecuación 2.16, donde se muestra la dinámica de un sistema que puede ser o no lineal.

$$\dot{x}(t) = f[x(t), u(t)] \quad (2.16)$$

Donde:

$\dot{x}(t) :=$ Dinámica del Sistema

$f :=$ Función lineal o no lineal

$x(t) :=$ Variables de Estado

$u(t) :=$ Entrada al sistema

$t :=$ tiempo

Entonces la función que puede llevar al sistema a un valor deseado de operación es:

$$u(t) = g[x(t)] \quad (2.17)$$

Con frecuencia, la función g de la ecuación 2.17, constituye el problema principal del control, y muchas veces es resuelto utilizando la función de control PID; Sin embargo, la solución ideal es encontrar la función g como la inversa de la función f . Esto permite

una cancelación de la dinámica del sistema, permitiendo que la salida sea igual a la entrada, sin error y sin oscilaciones [4, 6, 11].

La ecuación 2.18 resume el efecto de la dinámica inversa colocada en serie con la dinámica del sistema.

$$\frac{y(t)}{u(t)} = F(t) * G(t) = 1 \quad (2.18)$$

Donde:
 $G(t) = F^{-1}(t)$

La función que incluye la dinámica inversa de un sistema lineal puede no existir [4, 6, 11]. Incluso cuando existe, determinarla utilizando métodos analíticos puede resultar en una problemática adicional y muy compleja. La alternativa propuesta es aproximar esa función aprovechando las RN, debido a que son excelentes aproximadores de funciones no lineales o muy complejas [7].

La metodología principal del control neuronal consiste en aproximar la dinámica inversa del sistema que se desea controlar utilizando una RN, y colocarla en serie con el sistema.

Las etapas del diseño del controlador neuronal, denominado por sus siglas en inglés NNDIC (*Neural Network Direct Inverse Controller*), o Controlador Inverso Directo basado en una Red Neuronal, se enlistan a continuación (La denominación Directo, se debe a que la RN funciona directamente como controlador) [11].

1. Si el sistema es lineal, comprobar que existe la dinámica inversa.
2. Configurar una RN para identificar sistemas dinámicos.
3. Generar patrones de entrenamiento.
4. Entrenar la RN con los patrones y colocarla en serie como controlador.

Esta metodología propuesta en [4, 11], permite obtener un excelente controlador utilizando datos experimentales.

Para el diseño del NNDIC se emplea la metodología mencionada a continuación.

El modelo del Motor de CD que se desarrolló en el presente capítulo es de naturaleza lineal, esto permite verificar la existencia de la dinámica inversa verificando la controlabilidad, es decir, que el sistema sea totalmente controlable, garantiza la existencia de la dinámica inversa en sistemas lineales únicamente[6, 11].

Para verificar la controlabilidad, el modelo de la ecuación 2.10, se transformó a un modelo en espacio de estados, para poder conocer las matrices necesarias para el cálculo. El modelo en espacio de estados se muestra en la ecuación 2.19.

$$\begin{bmatrix} \dot{x}_1 \\ \dot{x}_2 \end{bmatrix} = \begin{bmatrix} -29.55 & -1250 \\ 1 & 0 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} + \begin{bmatrix} 1 \\ 0 \end{bmatrix} u$$

$$y = \begin{bmatrix} 0 & 2273 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}$$
(2.19)

Utilizando la fórmula para determinar la controlabilidad de un sistema lineal 2.20

$$C = [B \ AB \ A^2B \ \dots \ A^{n-1}B]$$
(2.20)

Y sustituyendo las matrices A y B de la ecuación 2.19, tomando en cuenta que n=2, se tiene la ecuación 2.21

$$C = [B \ AB] = \begin{bmatrix} \begin{bmatrix} 1 \\ 0 \end{bmatrix} & \begin{bmatrix} -29.55 & -1250 \\ 1 & 0 \end{bmatrix} \begin{bmatrix} 1 \\ 0 \end{bmatrix} \end{bmatrix}$$
(2.21)

Desarrollando la ecuación 2.21 se tiene:

$$C = \begin{bmatrix} 1 & -29.5455 \\ 0 & 1 \end{bmatrix}$$
(2.22)

Calculando el rango de la matriz C de la ecuación 2.22, se tiene que es igual a la longitud de la matriz A, es decir, 2. Por lo tanto, el sistema es totalmente controlable, por lo que la dinámica inversa existe y se puede aproximar con la RN.

El siguiente paso de la metodología mencionada consiste en la configuración de una RN para la identificación de sistemas dinámicos. Una RN puede aproximar funciones identificando el patrón de esta, utilizando un conjunto de pares de entrenamiento. Para que una RN identifique la función de un sistema dinámico, se requieren entradas adicionales a la red, como se puede observar en la figura 2.17.

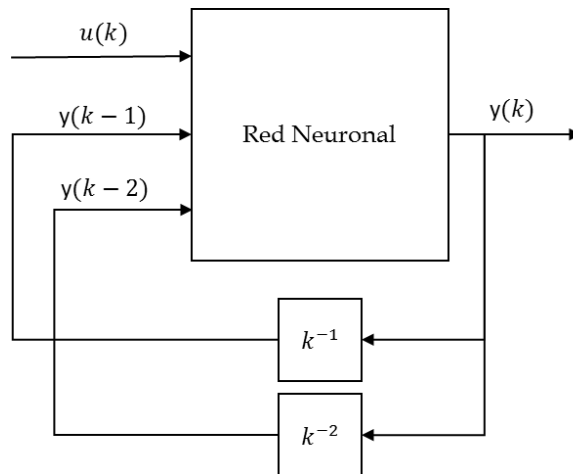


Figura 2.17 Esquema de RN para identificar Sistemas Dinámicos.

Las entradas de la RN de la figura 2.17 son necesarias para determinar los cambios en la salida de la propia red y ajustarse para aproximar los sistemas que dependen del tiempo [11, 24].

La RN propuesta, incluye además de las entradas de la figura 2.17, dos entradas más de la salida del Motor de CD, esto constituye un total de 5 entradas, como se muestra en la figura 2.18.

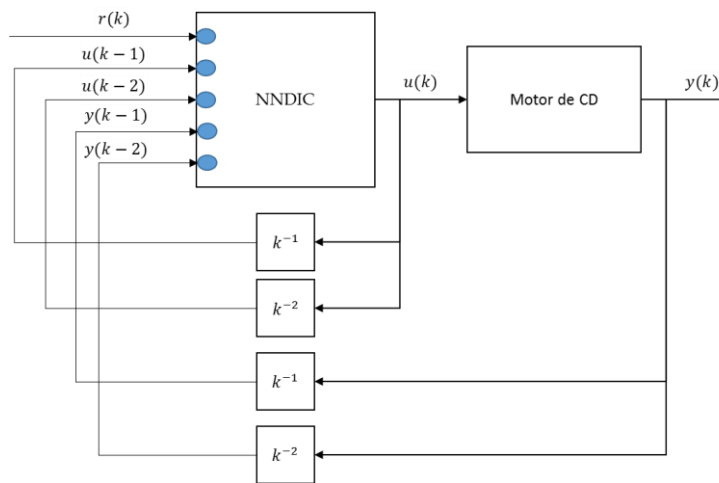


Figura 2.18 Esquema de la RN propuesta como NNDIC.

La tabla 2.11 muestra los parámetros resumidos de la RN seleccionada para la aproximación de la dinámica inversa para el NNDIC.

Tabla 2.11 Parámetros de la RN propuesta.

Parámetro	Descripción
Entradas	5
Salidas	1
Capas	3 (Entrada, Oculta, Salida)
Neuronas Capa Oculta	5
Entrenamiento	Retropropagación (Levenberg Marquard)
Datos de Entrenamiento	80000 muestras (Cada Entrada/Salida)

A continuación, siguiendo la metodología de diseño de NNDIC, se generaron los patrones de entrenamiento de entrada y salida [24]. El entrenamiento de la RN para identificar la dinámica inversa del sistema se llevó a cabo utilizando los datos del modelo en Simulink. La Figura 2.19 muestra el programa de Simulink que contiene los bloques que generan los arreglos de 80000 muestras.

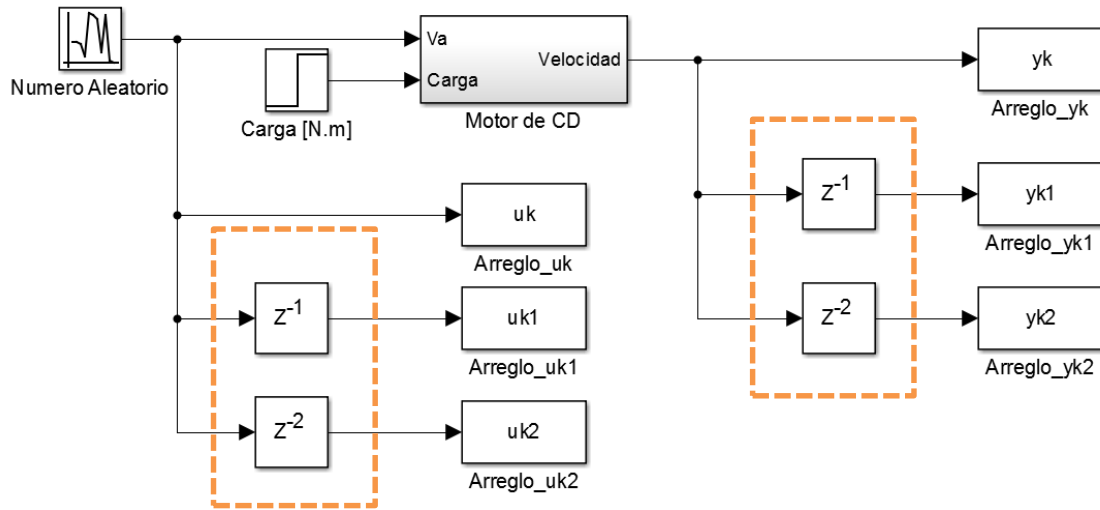


Figura 2.19 Programa de Simulink para la generación de patrones de entrenamiento.

En la figura 2.19 se puede observar la entrada *Numero Aleatorio*, que excita al modelo del Motor de CD, generando datos de salida que son almacenados en un arreglo de datos de MATLAB de 80000 muestras. Los bloques de retardo de tiempo dentro de los recuadros punteados de la figura 2.19, realizan la función de retrasar la información uno o dos pasos de integración según es el caso.

Un ejemplo de los patrones de entrada y salida, *uk* y *yk*, se muestra en la figura 2.20, donde se aprecia el conjunto de entrenamiento con 100 muestras para poderse visualizar.

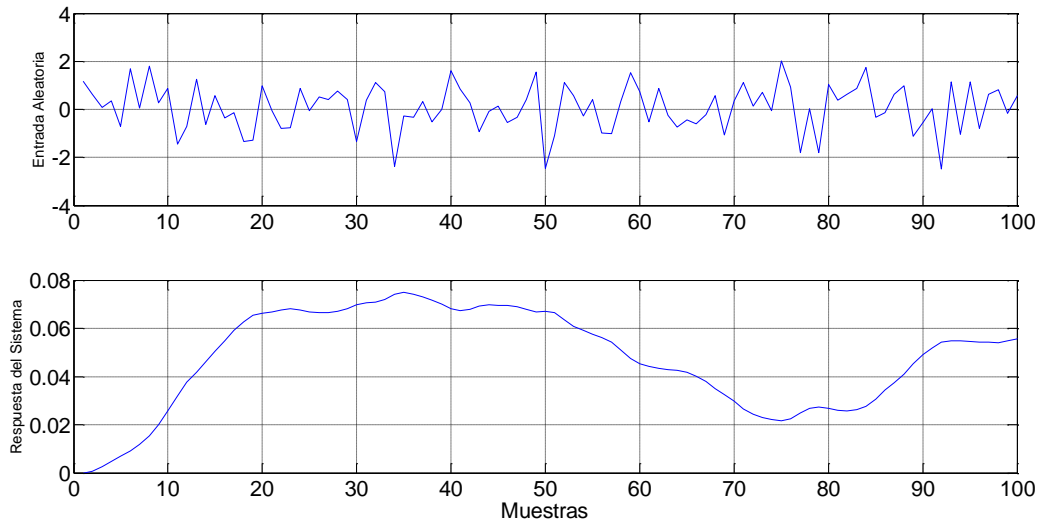


Figura 2.20 Ejemplo de Patrones de Entrenamiento.

Finalmente, la RN fue entrenada con los arreglos de patrones generados utilizando el programa de Simulink de la figura 2.19, el objetivo del entrenamiento es identificar y aproximar la dinámica inversa del sistema.

La figura 2.21 muestra la configuración del entrenamiento, donde se aprecia que la entrada de la RN es la salida del motor (velocidad) y la salida/objetivo de la RN es la entrada del motor (valor aleatorio de tensión). La señal de error mostrada y la flecha que cruza a la RN simbolizan al algoritmo de entrenamiento que modifica los pesos.

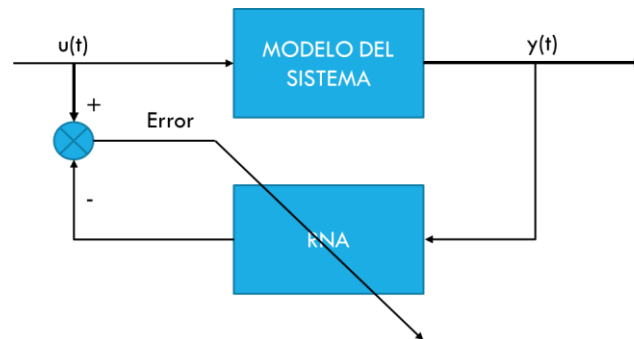


Figura 2.21 Entrenamiento de la Dinámica Inversa.

El entrenamiento se lleva a cabo utilizando un programa desarrollado en MATLAB, utilizando los algoritmos de entrenamiento incluidos en el paquete. Los resultados del entrenamiento se resumen en la tabla 2.12.

Tabla 2.12 Resultados del entrenamiento de la RN como NNDIC.

Resultado	Valor
Iteraciones	2768
Tiempo de Entrenamiento	01:47:16
Desempeño	2.17e-05
Gradiente	9.99e-08

El tiempo de entrenamiento es mucho menor si se seleccionan menos neuronas en la capa oculta, sin embargo, un número mayor de estas puede otorgar mejores resultados [4, 11, 24]. El criterio utilizado fue la elección del número de neuronas igual al número de entradas [25].

Los resultados de la modificación de los pesos de la RN debido al entrenamiento se muestran en la tabla 2.13 (Capa Oculta) y en la tabla 2.14 (Capa de Salida).

Tabla 2.13 Pesos de la Capa Oculta después del Entrenamiento.

Entradas	Pesos Neurona 1	Pesos Neurona 2	Pesos Neurona 3	Pesos Neurona 4	Pesos Neurona 5
$u(k)$	3.30544087	12.149488881	-39.933177	7.7221484	10.156072146
$y(k-1)$	1.19723935	4.4012235265	78.2539999	-18.84461	4.3901510864
$y(k-2)$	1.725560613	2.1294288105	-38.321583	11.1685175	-1.791166499
$u(k-1)$	-3.60468111	-14.16547140	-0.0057886	-0.0148845	12.87983435
$u(k-2)$	4.390151086	0.0045970382	0.89232393	-0.004589	2.8274008125
Sesgo	-0.00569774	0.0345395432	0.44347205	1.23948274	-0.12983412

Tabla 2.14 Pesos de la Capa de Salida después del Entrenamiento.

Neurona de la Capa Oculta	Pesos Neurona de Salida
N1	0.00697038232131660
N2	-0.0056977405221828
N3	-97.299367792667596
N4	0.00444974053490492
Sesgo	-0.21076256705461804

La principal desventaja en este tipo de entrenamiento, donde se busca identificar una función muy compleja como lo es la dinámica inversa de un sistema, es la cantidad masiva de información que la RN requiere para identificar correctamente. Esto produce tiempos de cómputo muy grandes que deben considerarse en la elección de la computadora para el entrenamiento.

La RN finalmente se colocó en serie con el modelo del Motor de CD, y se procedió a efectuar la simulación del sistema de control completo, obteniendo la respuesta a la entrada escalón de la siguiente forma mostrada en la figura 2.22.

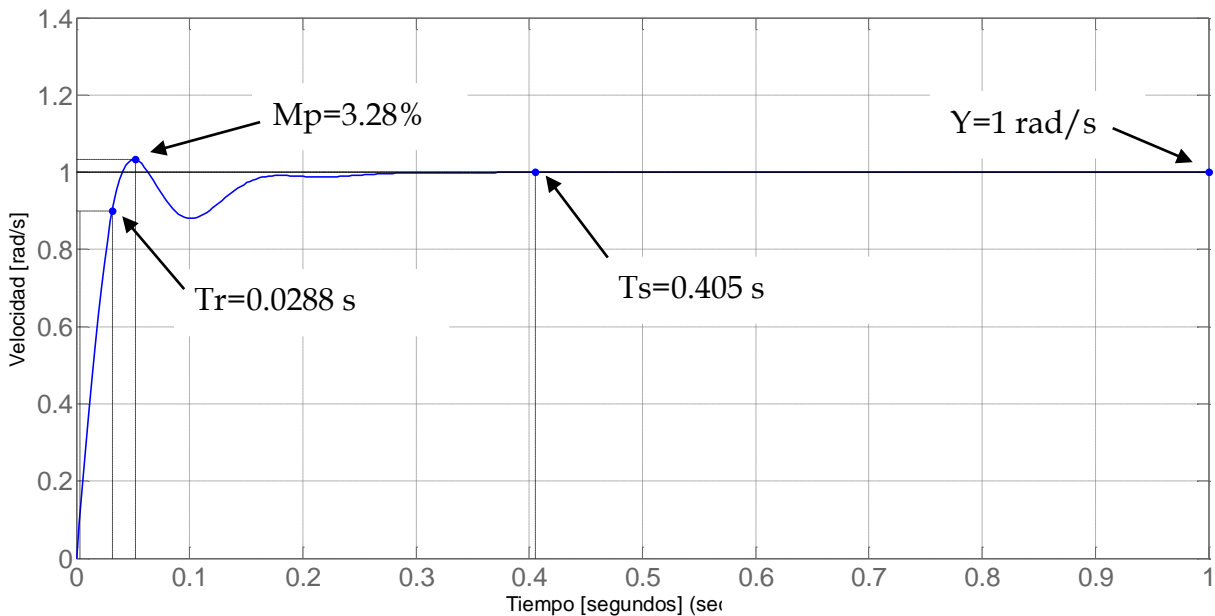


Figura 2.22 Respuesta escalón del sistema de control NNDIC.

Los resultados del sistema de control NNDIC se muestran en la tabla 2.15, se puede observar el mejor desempeño contra el sistema de control PI, además de cumplir con los requerimientos del control establecidos.

Tabla 2.15 Resultados del sistema de control NNDIC de la figura 2.22.

Característica	Sintonización Fina
Sobretiro ($M_p\%$)	3.28%
Tiempo de Asentamiento (T_s)	0.405 s.
Tiempo de Subida (T_r)	0.0288 s.
Error en Estado Estacionario ($1-Y\%$)	0%

2.5 ESTUDIO COMPARATIVO ENTRE EL CONTROL PI Y EL NNDIC

Los índices de desempeño ITAE, IAE e ISE se utilizaron para determinar el mejor sistema de control entre los dos desarrollados en este capítulo.

La prueba fue una entrada escalón unitario, y los índices se calcularon a partir de la señal de error (apéndice C), y se obtuvieron los siguientes resultados mostrados en la figura 2.23 y detallados en la tabla 2.16.

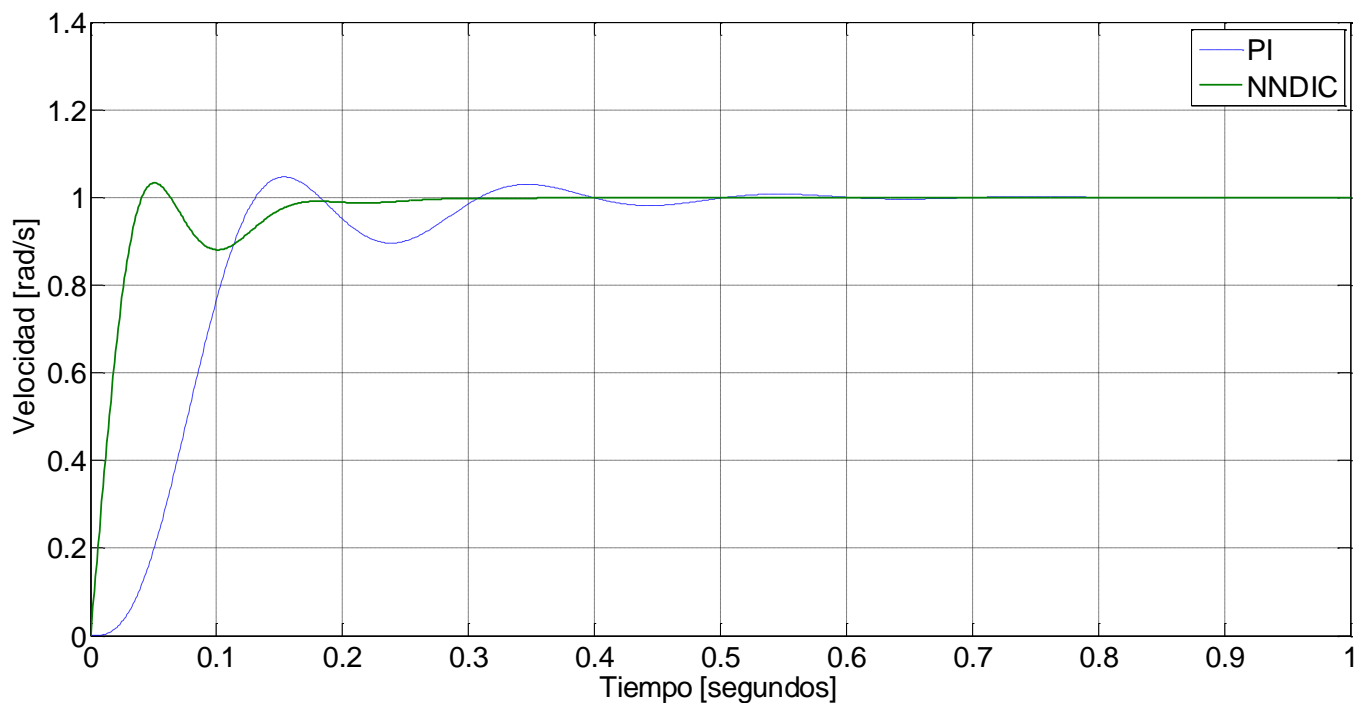


Figura 2.23 Comparativa de sistemas de control PI (línea punteada) y NNDIC (línea sólida).

Tabla 2.16 Comparativa de sistemas de control PI y NNDIC.

Características	PI	NNDIC
Sobretiro	4.65%	3.28%
Tiempo de Asentamiento	0.572	0.405
Tiempo de Subida	0.0764	0.0288
Error en Estado estacionario	0%	0%
ITAE	0.0072	0.0012
IAE	0.0899	0.0249
ISE	0.0611	0.0109

Por lo tanto se concluye de la figura 2.23 y de los datos de la tabla 2.16, que el desempeño del controlador NNDIC es superior, concordando con la teoría.

En el capítulo 4, se abordaran las pruebas de los sistemas de control en lazo cerrado utilizando el simulador HIL, donde se detallaran más resultados y características de ambos sistemas.

CAPÍTULO 3:

IMPLEMENTACIÓN HIL

3.1 INTRODUCCIÓN

La simulación en tiempo real es una herramienta moderna que permite analizar, visualizar y experimentar con la dinámica de sistemas, en una relación de tiempo uno a uno, es decir, donde un segundo de la simulación equivale aproximadamente a un segundo del tiempo de la vida real, sin perder cantidades importantes de información y sin afectar de alguna forma la dinámica de los sistemas con los que interactúa [26].

Una de las principales problemáticas de este tipo de simulaciones, son los tiempos de cálculo computacionales que pueden crear retardos en el desempeño del simulador. Recientemente, los sistemas de simulación en tiempo real han comenzado a incluir dispositivos físicos con los que interactuar y experimentar, realizando pruebas detalladas antes de la implementación con el sistema físico real, reduciendo los riesgos y costos de las pruebas físicas [27].

La interacción con dispositivos físicos (hardware), requiere de medios de comunicación de datos que cumplan el intercambio de información de una manera confiable (pocas pérdidas de información), rápida (poca latencia) y precisa (alta precisión en los datos).

Este tipo de simuladores (HIL) constituyen un alto costo, sin embargo existen alternativas de desarrollo tales como la paquetería LabVIEW, software de instrumentación virtual con una alta capacidad de interacción con hardware [28].

En el presente trabajo, se procede a desarrollar un emulador del motor de CD dentro de LabVIEW, utilizando una RN entrenada para identificar la dinámica del motor y servir como modelo en el simulador, posteriormente, los algoritmos de control propuestos en el capítulo 2 (PI y NNDIC) se implementan en el Sistema embebido FRDM-K64F de la marca Freescale (apéndice D), y se coloca en lazo cerrado con LabVIEW, emulando un sistema de control en tiempo real, listo para visualizar la velocidad del motor de CD y simular disturbios y cambios en el punto de ajuste.

La figura 3.1 muestra un diagrama a bloques del desarrollo del simulador HIL, donde se observa el emulador del motor de CD en LabVIEW, la línea de comunicación con el sistema embebido, la acción de control generada por el FRDM-K64F y los algoritmos

de control embebidos en su memoria, y la visualización de la velocidad regulada, todo esto en una interfaz gráfica de LabVIEW y en tiempo real.

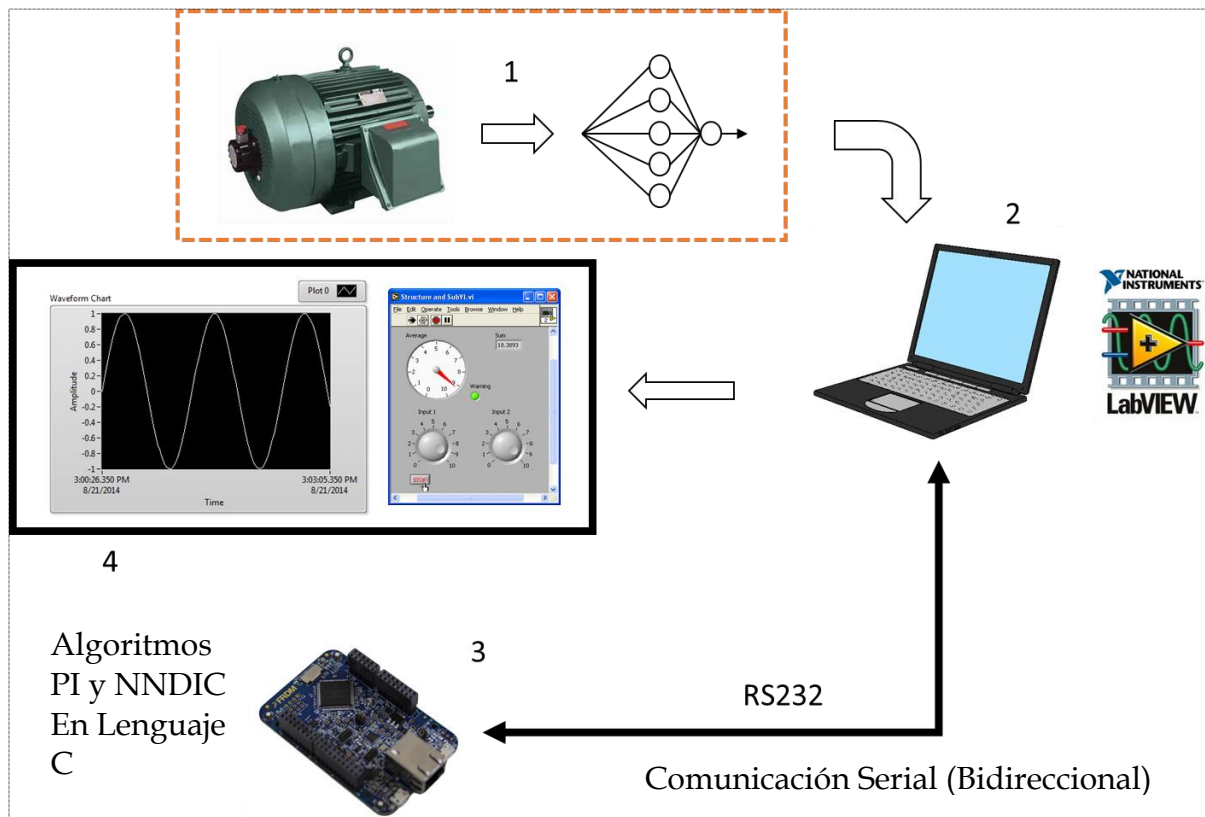


Figura 3.1 Diagrama a bloques del Simulador HIL.

En la figura 3.1 se pueden observar las 4 etapas generales del Simulador HIL, descritas a continuación:

1. Identificación de la dinámica del motor de CD utilizando una RN, los datos para el entrenamiento se obtienen del modelo desarrollado en el capítulo 2.
2. Programación de la RN en LabVIEW, además de la creación de la interface gráfica.
3. Implementación de los algoritmos de control diseñados en el capítulo 2 (PI y NNDIC) en el sistema embebido FRDM-K64F.
4. Visualización en tiempo real del control de velocidad del motor de CD.

En este capítulo se desarrollan las cuatro etapas generales mostradas en la figura 3.1, se presenta la metodología de identificación de sistemas con RN, y la implementación en software y hardware de las RN. Finalmente se unen las etapas para concluir el desarrollo del simulador HIL.

3.2 IDENTIFICACIÓN DEL MOTOR DE CD.

La identificación de sistemas es un área de estudio que utiliza métodos estadísticos para construir modelos matemáticos que describan de manera exacta la dinámica de sistemas a partir de datos medidos. Esta área es auxiliar en la teoría de control, pues supone una excelente herramienta para modelar sistemas cuya dinámica no se conoce, pero se cuenta con información experimental.

La metodología general para la identificación de sistemas se muestra en el diagrama de flujo de la figura 3.2 [24].

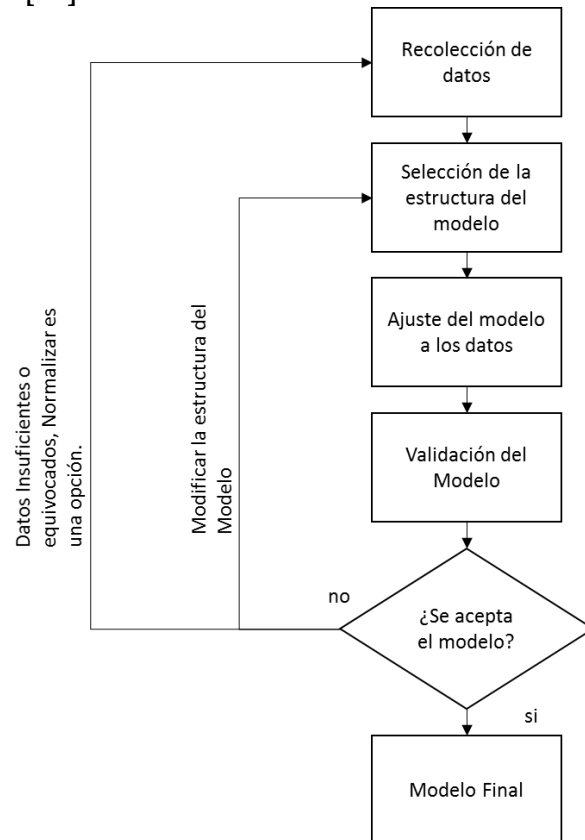


Figura 3.2 Diagrama de Flujo para la identificación de Sistemas.

Siguiendo la metodología de la figura 3.2, se desarrolló un modelo basado en una RN para simular la dinámica del motor de CD.

3.2.1 Recolección de Datos.

Los datos utilizados para el entrenamiento se obtuvieron del modelo del motor de CD de la figura 2.2. Sin embargo, datos experimentales pueden ser utilizados y se recomienda esta metodología para sistemas que no cuenten con un modelo preciso [24].

La principal ventaja de utilizar información experimental es que el modelo que se obtendrá será mucho más preciso que cualquier modelo analítico que se desarrolle,

esto debido a que una RN puede identificar patrones no lineales, por lo que la dinámica casi total será emulada por el modelo basado en RN [4, 5, 6].

Debido a que el modelo del motor de CD para el caso de estudio tiene un modelo lineal e invariante en el tiempo, se utilizó la metodología de la figura 3.3, y se generaron arreglos de muestras de entrada y salida, así como los datos de los retardos de tiempo explicados en el capítulo 2. La entrada al modelo es un barrido de una señal sinusoidal en el intervalo de 0.1 Hz a 1 Hz durante 10 segundos, con el fin de abarcar el espectro completo del modelo del motor de CD.

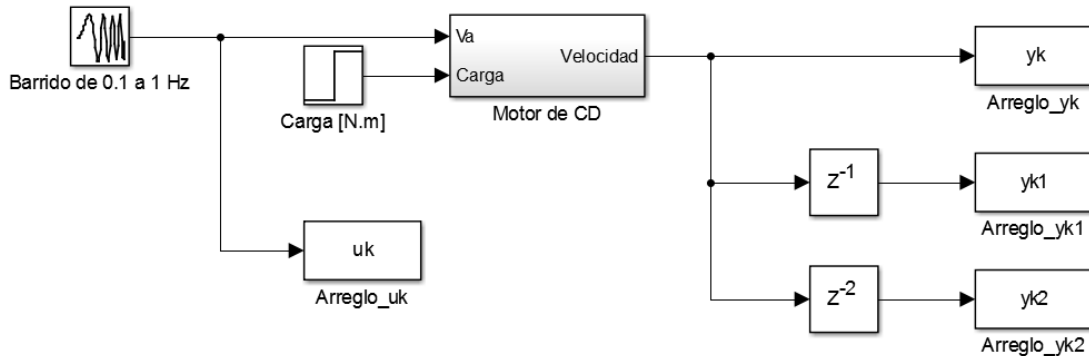


Figura 3.3 Modelo de Simulink para generar patrones de entrenamiento.

Por lo que finalmente se contó con 3 arreglos de entrada de 5001 muestras aleatorias y un arreglo de salida con 5001 muestras de la respuesta del motor a esas entradas.

3.2.2 Selección de la Estructura del Modelo.

Como se ha mencionado, la base del modelo es una RN. Otros modelos pueden ser utilizados en esta etapa de la identificación de sistemas, con su respectivo algoritmo de ajuste, que va más allá del alcance de este trabajo.

Seleccionando el modelo como una RN, se propone una con la estructura resumida en la tabla 3.1.

Tabla 3.1 Estructura de la RN para identificar al motor de CD.

Parámetro	Descripción
Entradas	3
Salidas	1
Capas	3 (Entrada, Oculta, Salida)
Neuronas Capa Oculta	3
Entrenamiento	Retropropagación (Levenberg Marquard)
Datos de Entrenamiento	5001 muestras (Cada Entrada/Salida)

Las entradas de la RN para identificar el motor de CD son:

- $u(k)$. Tensión de Armadura.
- $y(k - 1)$. Valor anterior de la Velocidad.
- $y(k - 2)$. Valor ante anterior de la Velocidad.

La salida de la RN u objetivo de la RN es:

- $y(k)$. Velocidad del motor.

3.2.3 Ajuste del Modelo a los Datos.

Para ajustar la RN a los datos recolectados, se procede a entrenar la red utilizando el algoritmo de Levenberg Marquard auxiliándose de la paquetería de MATLAB. La tabla 3.2 muestra los resultados del entrenamiento.

Tabla 3.2 Resultados del entrenamiento de la RN como modelo del Motor de CD.

Resultado	Valor
Iteraciones	1732
Tiempo de Entrenamiento	0:12:36
Desempeño	9.56e-10
Gradiente	9.98x10-9

Los pesos de la RN fueron ajustados de manera exitosa, mostrándose los resultados en la tabla 3.3 para la capa oculta y en la tabla 3.4 para la capa de salida.

Tabla 3.3 Pesos de la Capa Oculta.

Entradas	Pesos Neurona 1	Pesos Neurona 2	Pesos Neurona 3
$u(k)$	2.157521e-05	0.000178472518	-0.0003992796
$y(k - 1)$	-0.6962444648	0.403290491704	-0.4654196120
$y(k - 2)$	0.34844518325	-0.19950964382	0.227889879158
<i>Sesgo</i>	-1.6165959875	-0.79701574392	-0.31127378387

Tabla 3.4 Pesos de la Capa de Salida

Neurona de la Capa Oculta	Pesos Neurona de Salida
N1	-1.4869319130024057
N2	3.8257187857378998
N3	-2.249465948299473
<i>Sesgo</i>	0.48147883772129324

3.2.4 Validación del Modelo.

Después del ajuste del modelo seleccionado a los datos, en este caso la RN, se procede a realizar una verificación para validar su uso. La técnica de MSE permite tener un cuantificador de que tan preciso es el modelo desarrollado en comparación de la dinámica del sistema a modelar. Esta comparativa puede realizarse con unas diversas pruebas a entradas escalón, y calcular el valor del MSE [24].

La figura 3.4 muestra la respuesta del motor de CD y la de la RN a una entrada escalón unitario.

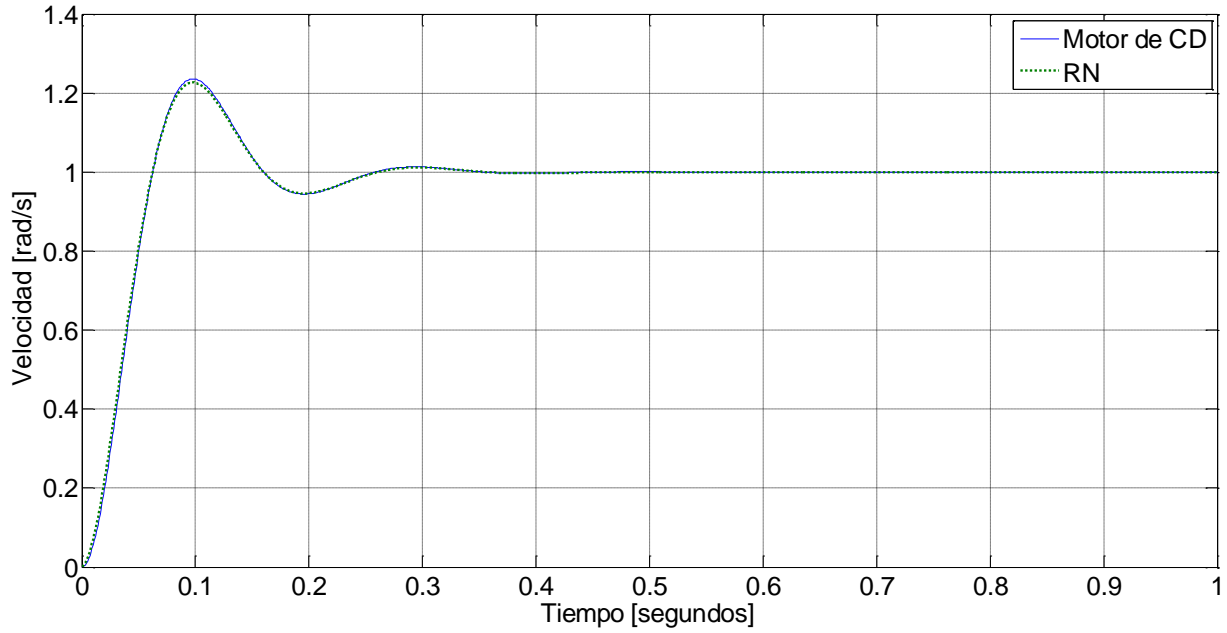


Figura 3.4 Prueba escalón unitario para verificación de la RN como modelo.

La respuesta de la RN a una entrada escalón es satisfactoria ya que es muy parecida a la del motor de CD, sin embargo, para tener el indicador cuantitativo confiable, el cálculo del MSE se efectuó de la siguiente manera:

$$Error = Respuesta\ del\ Motor - Respuesta\ de\ la\ RNA \quad (3.1)$$

$$MSE = \frac{1}{n} \sum_{i=0}^n Error[i]^2 \quad (3.2)$$

Donde n es el número de muestras = 1001

Entonces para esta prueba el valor del MSE fue $MSE = 1.24341 \times 10^{-5}$, que representa el 0.0012% de error.

La siguiente prueba fue una entrada escalón de 0.5 en magnitud. La figura 3.5 muestra el resultado de la RN comparado con el del motor de CD.

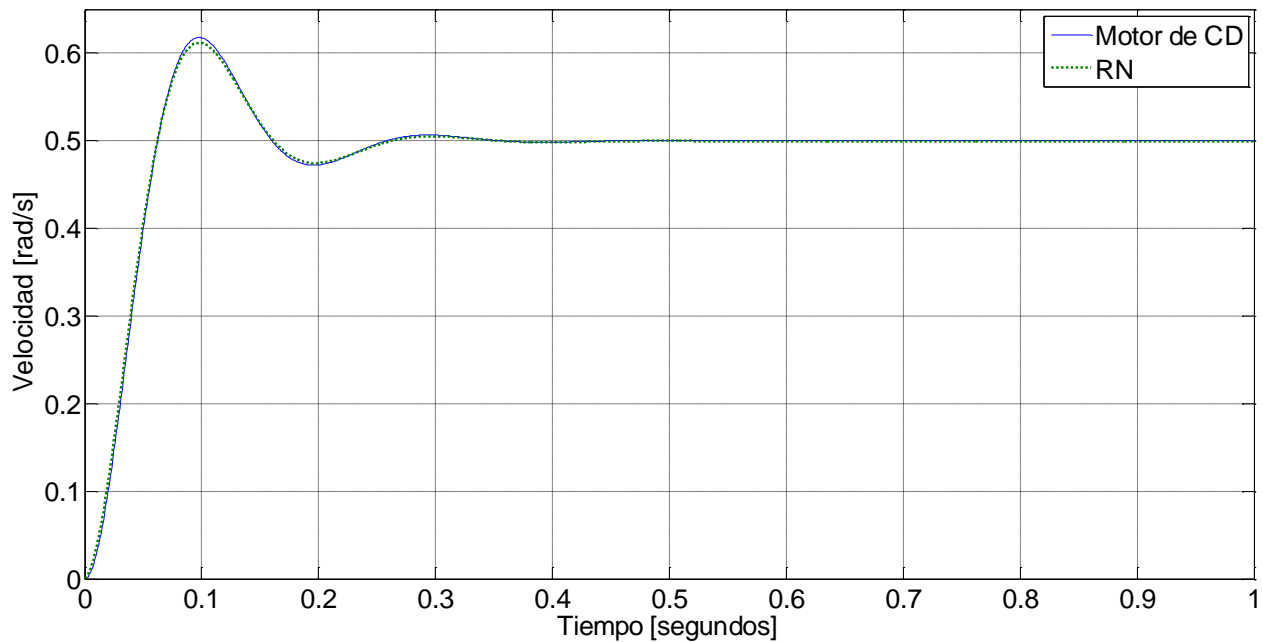


Figura 3.5 Prueba Escalón de 0.5, para verificación de la RN como modelo.

De igual manera, se calculó el MSE del resultado, obteniéndose el valor $MSE = 1.242931 \times 10^{-5}$, que representa el 0.0012% de error.

Por lo que se concluye así el proceso de la identificación del modelo, obteniéndose una RN que identificó exitosamente la dinámica del motor de CD con un promedio de 0.0012% de error obtenido de dos pruebas a entrada escalón.

3.3 PROGRAMACIÓN EN LABVIEW

La programación del simulador HIL en el software LabVIEW consiste de 3 partes generales:

1. Programación de la RN
2. Comunicación con el Sistema Embebido
3. Interface de Usuario Gráfica

La paquetería de LabVIEW contiene las herramientas necesarias para desarrollar el simulador de manera exitosa, además de que se aprovechó la capacidad de procesamiento en paralelo del software para la ejecución en tiempo real de la RN como modelo del motor de CD.

3.3.1 Programación de la RN

La RN entrenada para simular la dinámica del motor de CD, fue programada en LabVIEW dentro de una subrutina conocida como *subvi* ("SubVirtualInstrument") [29], que consiste de una entrada (Tensión de Armadura, $u(k)$) y una salida Velocidad Angular, $y(k)$. El bloque de la subrutina se muestra en la figura 3.6.



Figura 3.6 Subvi RN como Motor de CD.

El procedimiento para programar la RN entrenada es el cálculo hacia adelante común del perceptrón multicapa, es decir, como se muestra en la ecuación 3.3 [4, 6, 30].

$$H_i = f \left\{ \sum_{j=1}^n u_j w_i \right\} \quad (3.3)$$

Detallando la ecuación 3.3 en la tabla 3.5.

Tabla 3.5 Descripción de la ecuación 3.3

Variable	Descripción
j	Número de Entrada
i	Número de Neurona
n	Número de Entradas
$u(k)$	Vector de Entradas
w_i	Vector de Pesos de la neurona i
f	Función de activación Sigmoidal
$H_i(k)$	Salida de la Neurona i

Se construye un vector de las salidas de las neuronas de la capa oculta H_i , que es utilizado en la capa oculta de la forma de la ecuación 3.4 [4, 6, 30].

$$o = g \left\{ \sum_{i=0}^m H_i w_o \right\} \quad (3.4)$$

Detallando la ecuación 3.4 en la tabla 3.6.

Tabla 3.6 Descripción de la Ecuación 3.4.

Variable	Descripción
H_i	Vector de Salidas de las Neuronas Ocultas
w_o	Vector de Pesos de la Capa de Salida
m	Numero de Neuronas de la Capa Oculta
g	Función de Activación Lineal
o	Salida de la RN

Utilizando las ecuaciones 3.3 y 3.4 se procede a programar la RN como modelo de Motor de CD, como se muestra en la figura 3.7, el vector de entradas $u(k)$.

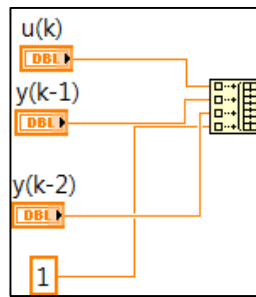


Figura 3.7 Vector de Entradas

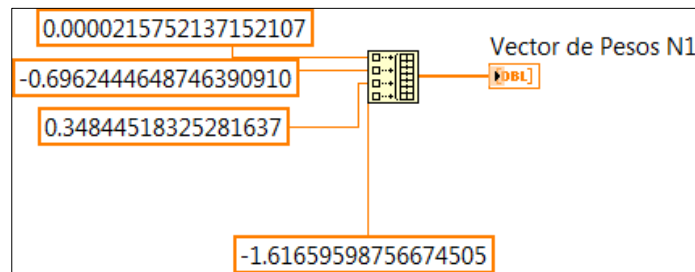


Figura 3.8 Vector de Pesos de la Neurona 1 de la Capa Oculta.

La operación equivalente a la mostrada en la ecuación 3.14 es el producto escalar [4], por lo que ambos vectores (entrada, pesos N1), de las figuras 3.7 y 3.8, se procesan por el bloque de producto escalar y por la función de activación sigmoideal, figura 3.9.

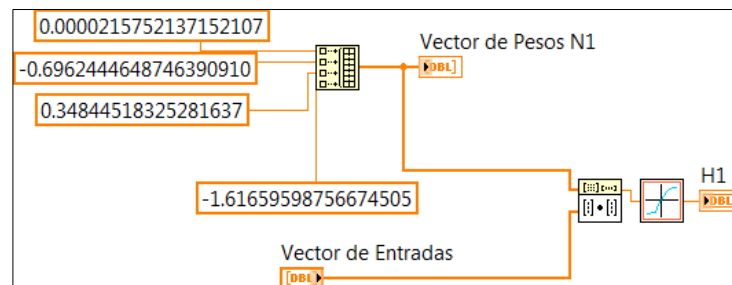


Figura 3.9 Operaciones de la Ecuación 3.14 por neurona.

Posteriormente, al tener las salidas de las tres neuronas, se crea el vector H , y se procesa en la capa de salida de la red, como se muestra en la ecuación 3.4, la figura 3.10 contiene las operaciones mencionadas.

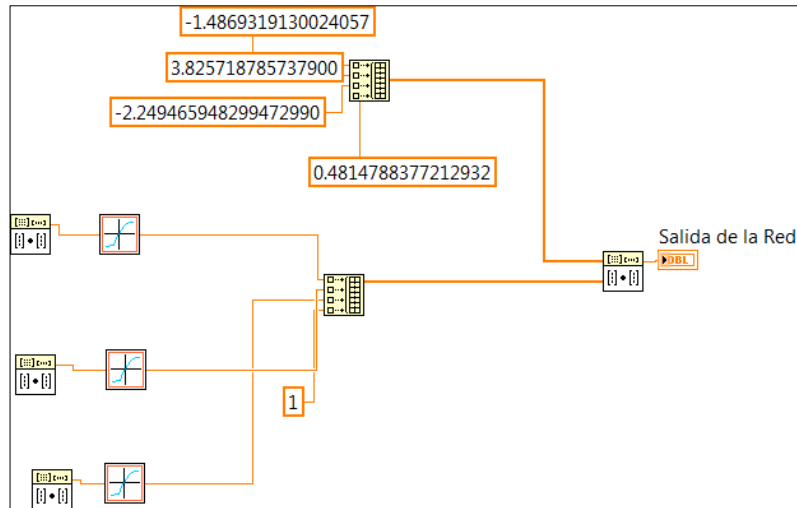


Figura 3.10 Operaciones de la capa de salida de la ecuación 3.4.

Cabe mencionar que el sesgo se ha tomado como entrada. En las figuras 3.7 y 3.10 aparece conformando el vector $u(k)$ (Vector de Entradas) y $H(k)$ (Vector de Salidas de las Neuronas Ocultas), con su respectivo peso en las neuronas ocultas y de salida.

Así se concluye la programación de la RN mostrada en este apartado en detalle con la neurona 1 como ejemplo, el código completo se anexa en el apéndice E.

La salida de la RN en el entorno de LabVIEW se puede visualizar en la herramienta denominada registrador (*chart*), en tiempo real, como se muestra en la figura 3.11, la respuesta de la RN a una entrada escalón unitario.

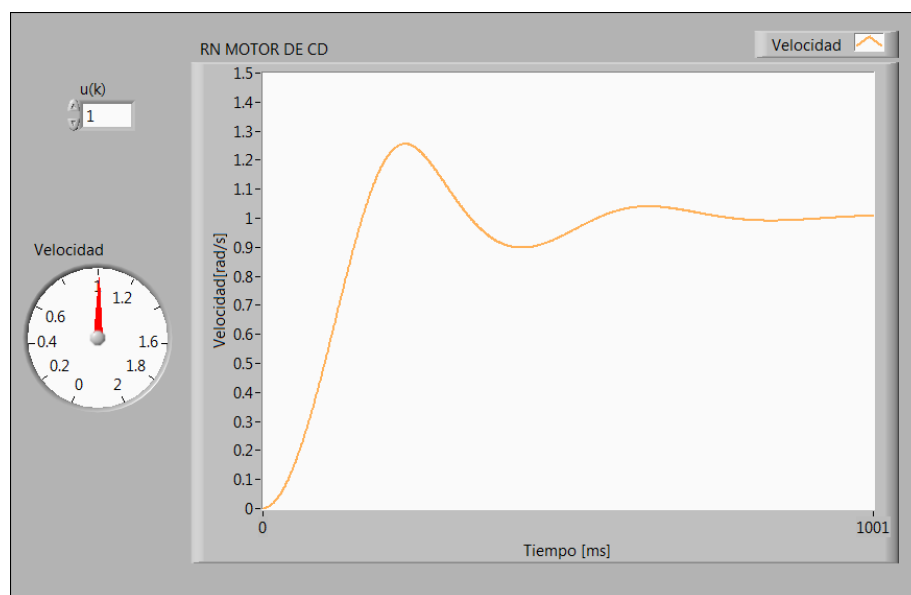


Figura 3.11 Respuesta de la RN en el simulador de LabVIEW.

3.3.2 Comunicación con el Sistema Embebido.

El protocolo de comunicación utilizado para enlazar el sistema embebido FRDM-K64F con el simulador programado en LabVIEW fue el RS232 [31, 32].

El RS232 es un protocolo de comunicación serial punto a punto que representa un estándar muy antiguo pero que es aún vigente, consiste en el intercambio de información con dos canales unidireccionales, uno para el envío de datos y otro para la recepción. El enlace es asíncrono, por lo que los dispositivos que estén comunicados utilizando este protocolo, deben ser configurados de la misma manera [33]. El FRDM-K64F contiene un módulo de comunicación RS232 virtual, es decir, utilizando el puerto USB que contiene puede comunicarse utilizando este protocolo con otro dispositivo con un módulo similar, prácticamente cualquier computadora personal moderno.

Los parámetros principales de la configuración del módulo de la comunicación serial en ambas plataformas (FRD-K64F y LabVIEW), se muestra en la tabla 3.7.

Tabla 3.7 Configuración de la Comunicación Serial.

Parámetro	Valor
Tasa de Transmisión	230400 baud
Longitud de los datos	8 bits
Carácter Terminador	0xA='\n'=10
Puerto Virtual	COM3

La configuración de estos parámetros debe de ser igual en ambos dispositivos para garantizar el intercambio exitoso de la comunicación.

La biblioteca de comunicación con Hardware de LabVIEW, contiene los bloques para la configuración de la comunicación serial estándar, como se muestra en la figura 3.12, el protocolo fue configurado de acuerdo a la tabla 3.7.

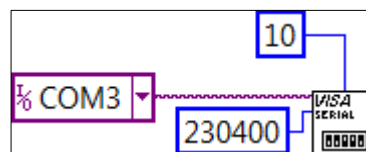


Figura 3.12 Configuración Serial LabVIEW.

La configuración mostrada en este apartado, se utiliza de la misma forma con su respectivo lenguaje de programación en el Sistema Embebido FRDM-K64F descrito después de la programación del HIL en este capítulo.

3.3.3 Programación de la Interfaz Gráfica de Usuario

El desarrollo de la simulación del motor de CD llevada a cabo en LabVIEW fue programada con el fin de tener una interacción directa y en tiempo real con el usuario.

Utilizando la instrumentación virtual que el paquete contiene [29], la variable controlada del simulador, la velocidad angular del motor de CD, puede visualizarse utilizando un registrador. La figura 3.13, muestra el *subvi* (subrutina) que contiene la RN con la dinámica del motor y los elementos que permiten su visualización simple.

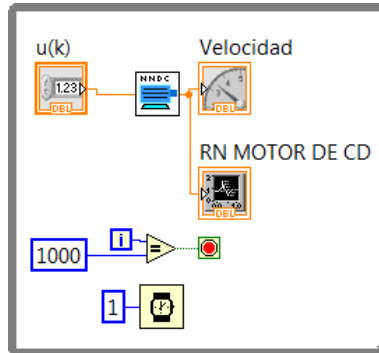


Figura 3.13 Programación de la Visualización Simple.

La tabla 3.8 contiene la explicación de los bloques de la figura 3.13.

Tabla 3.8 Descripción de los bloques de la Figura 3.13.

Bloque	Descripción
	Valor de entrada a la RN
	RN entrenada como Motor de CD
	Medidor de carátula virtual
	Registrador de velocidad virtual
	Tiempo de Muestreo (ms)

El ejemplo de la visualización simple es la figura 3.11 de la página 44.

Como el objetivo es simular además disturbios en el motor, representados por cambios de carga, una perilla virtual fue implementada para servir de entrada al sistema y simular el aumento o disminución del par de carga en la flecha. La figura 3.14 muestra la perilla virtual de LabVIEW, y se muestra el intervalo de operación de la carga, que va desde 0 N.m hasta 5 N.m.

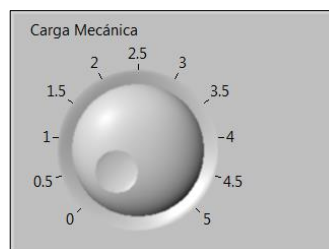


Figura 3.14 Perilla virtual para la simulación de cambios de carga.

Para simular el cambio de punto de ajuste, se utilizó el selector horizontal mostrado en la figura 3.15.

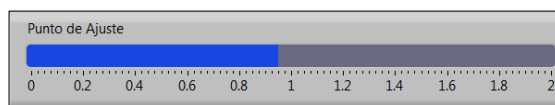


Figura 3.15 Selector Horizontal para simular el Punto de Ajuste.

Y para completar la interfaz gráfica, el conjunto de botones virtuales de la figura 3.16, permiten el manejo de la simulación en general. La tabla 3.9 contiene la descripción de cada elemento.

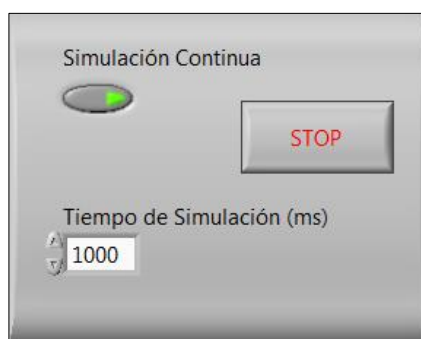





Figura 3.16 Botones virtuales para el manejo de la Simulación.

Tabla 3.9 Descripción de los Botones virtuales de la figura 3.16.

Botón	Descripción
	Selector de simulación continua o limitada por un tiempo definido
	Entrada numérica en milisegundos para definir el tiempo de la simulación
	Botón para detener inmediatamente la Simulación

El código de programación y la interfaz gráfica completa se pueden visualizar en el apéndice E.

Con estas herramientas la simulación de la velocidad del motor de CD en lazo abierto esta completada, como se ha mencionado, los algoritmos de control fueron implementados en el sistema embebido FRDM-K64F y tanto la tarjeta como el simulador de LabVIEW, se colocaron en lazo cerrado y se verifico su funcionamiento con una serie de pruebas típicas de la teoría de control (Capítulo 4).

3.4 IMPLEMENTACIÓN DE LOS ALGORITMOS DE CONTROL EN EL SISTEMA EMBEBIDO

3.4.1 Control PI

La implementación del controlador PI en hardware se elaboró introduciendo el algoritmo en el microcontrolador FRDM-K64F. El objetivo principal del sistema embebido es tomar la señal de error de la simulación en tiempo real utilizando el protocolo de comunicación RS232, hacer un cálculo de la señal de control, y enviarlo de regreso al sistema emulado con un tiempo de muestreo de 1 ms. Para el cálculo de la señal de control, la siguiente ecuación del controlador PI en tiempo discreto fue utilizada:

$$u(k) = Kp * [error(k) - error(k - 1)] + Ki * error + u(k - 1) \quad (3.5)$$

Donde:

$u(k) :=$ Señal de Control

$Kp :=$ Ganancia Proporcional

$Ki :=$ Ganancia Integral

$error(k) :=$ Señal de Error

$error(k - 1) :=$ Señal de Error anterior

$u(k - 1) :=$ Señal de Control anterior

Esta ecuación en diferencias fue implementada en lenguaje C, utilizando el compilador gratuito MBED, especializado en microcontroladores ARM [33]. El tipo de programación de MBED es intuitivo y fácil de manejar, por lo que el tiempo de programación del sistema embebido fue relativamente corto.

La figura 3.17 muestra el diagrama de flujo del programa en C del microcontrolador, y en el apéndice D, el código completo.

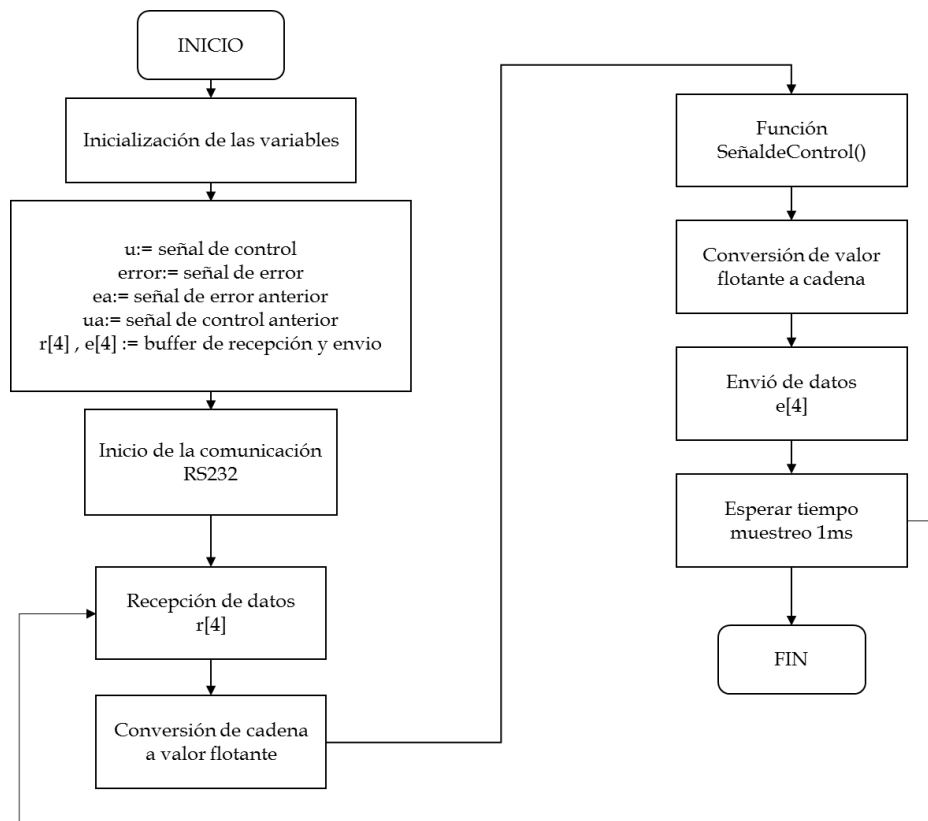


Figura 3.17 Diagrama de flujo del Control PI Embebido en FRDM-K64F.

La función “*SeñalControl()*” que aparece en la figura 3.17, contiene la ecuación 3.5, con las ganancias del controlador mencionadas anteriormente.

3.4.2 Control NNDIC

La RNA fue programada de la misma manera en lenguaje C en el microcontrolador FRDM-K64F utilizando MBED.

La implementación de las RNA en hardware se vio limitada mucho tiempo hasta la llegada de sistemas embebidos capaces de manejar información en cantidades grandes y a frecuencias muy altas. El FRDM-K64F satisface las necesidades de una RNA implementada en hardware con las especificaciones detalladas en el anexo.

El barrido de información debe hacerse en paralelo según la teoría de las redes neuronales, sin embargo, el microcontrolador hace una instrucción a la vez, excepto que a una frecuencia muy alta (120 MHz), por lo que las operaciones son lo suficientemente rapidez para considerarse en paralelo para esta aplicación.

El diagrama de flujo mostrado en la figura 3.18 fue implementado en un programa cuyo código se observa en el apéndice D.

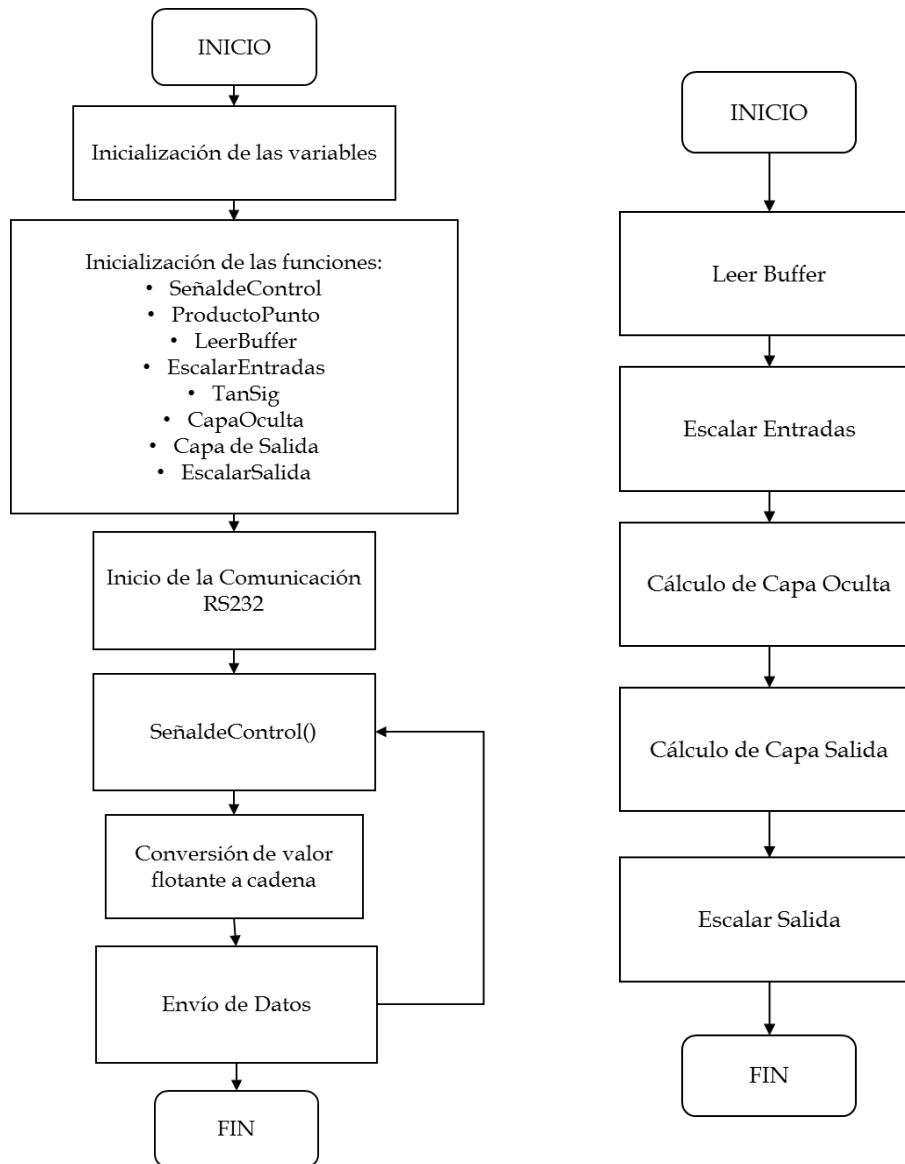


Figura 3.18 Diagrama de Flujo del Control NNDIC embebido en FRDM-K64F

La función de Señal de Control mostrada en el diagrama de flujo de la figura 3.18 contiene el grueso de la RN implementada y es un conjunto de funciones que trabajan para desarrollar las capas de entrada, oculta y salida, así como las funciones de activación y escalamiento de las señales. Su funcionamiento general se muestra en el diagrama de flujo de la figura.

Las entradas de la RNA para el neurocontrolador son el valor deseado, la salida anterior del sistema y la salida ante anterior del sistema.

3.5 SIMULACIÓN HIL EN LAZO CERRADO

Una vez que los sistemas de la simulación fueron desarrollados, la velocidad simulada en el emulador del motor de CD programado en LabVIEW fue controlada utilizando los algoritmos desarrollados en el capítulo 2 implementados en Hardware. Las ventajas de este tipo de simulaciones son variadas y las desventajas radican en los costes como se ha mencionado en la tesis.

La alternativa propuesta en la tesis de utilizar LabVIEW como simulador principal y el sistema embebido de desarrollo FRDM-K64F, mostro ser eficiente y económica, pues las simulaciones se realizan en tiempo real sin demoras por tiempos de ejecución gracias al modelo basado en RN implementado.

Los resultados de los controladores implementados en FRDM-K64F se probaran en el capítulo 4, pero en términos de la simulación HIL, el intercambio de información se realizó de manera exitosa y en tiempo real.

La figura 3.19 muestra una captura de pantalla del sistema HIL funcionando en tiempo real. La computadora personal contiene la paquetería de LabVIEW con la interfaz gráfica mostrando la velocidad del motor de CD siendo regulada por el NNDIC y el sistema embebido en el recuadro punteado con la interfaz USB protocolo RS232.

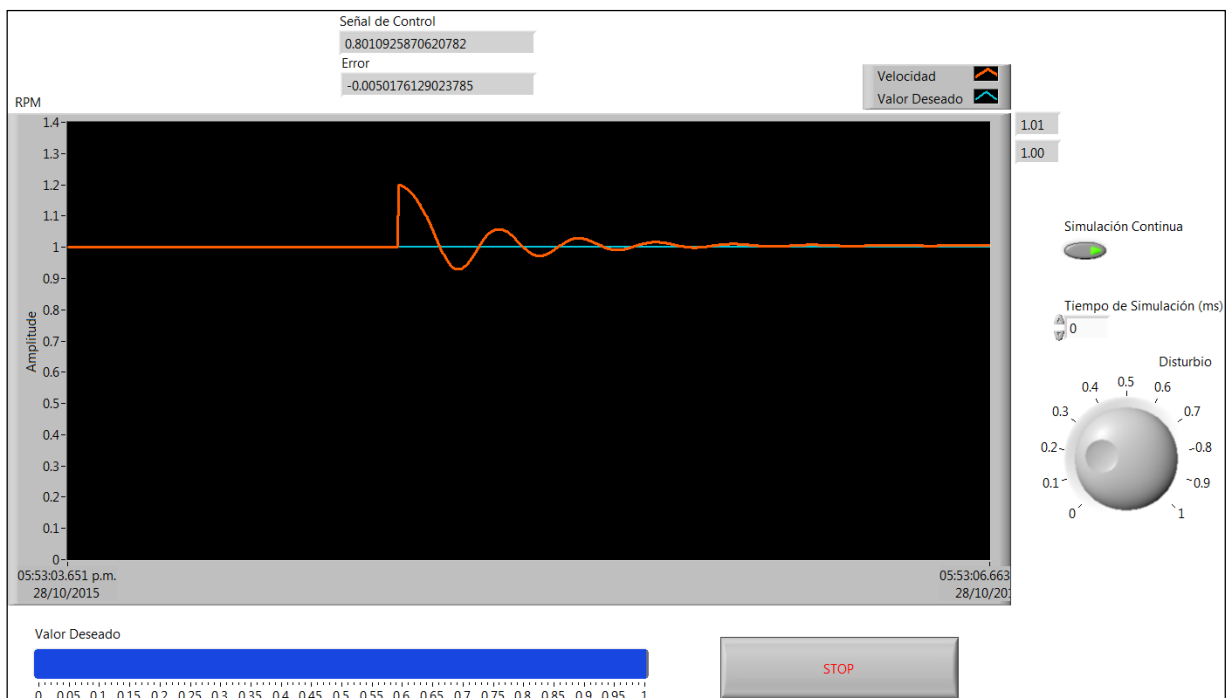


Figura 3.19 Interfaz Gráfica de Usuario del simulador HIL en tiempo real.

La figura 3.19 muestra en la interfaz de usuario, las opciones con las que se cuentan se detallan en la tabla 3.10.

Tabla 3.10 Descripción de los instrumentos virtuales de la Figura 3.19.

Función	Descripción	Intervalo de Operación
Valor Deseado	Punto de Ajuste del sistema a controlar	0 a 1
Disturbio	Señal simulada que afecta al desempeño del sistema y modifica su valor	0 a 1
Simulación Continua	Opción de Simulación continua o hasta que se detenga manualmente con el botón “STOP”	Criterio del Usuario
Tiempo de Simulación	Simulación en un valor dado en milisegundos	Criterio del Usuario

CAPÍTULO 4:

PRUEBAS EXPERIMENTALES Y RESULTADOS

4.1 INTRODUCCIÓN

Este capítulo aborda directamente los resultados de las pruebas experimentales realizadas al control de velocidad del motor de CD, desarrollados en la simulación HIL.

Las pruebas consisten en el análisis a disturbios en los sistemas y el desempeño de los controladores para el rechazo, también se incluyen pruebas de cambio de punto de ajuste así como el desempeño del microcontrolador mientras actúa en tiempo real.

Todas las pruebas fueron realizadas en LabVIEW, utilizando el simulador propuesto, que contiene una RNA simulando cada caso de estudio. Los resultados fueron graficados y mostrados tanto en Excel como en LabVIEW con fines de precisión en las figuras.

4.2 PRUEBAS DE DESEMPEÑO DEL SISTEMA DE CONTROL

Las pruebas a cada sistema de control consisten generalmente en cambios en el punto de ajuste, así como la simulación de disturbios que modifiquen la salida del sistema y obliguen al controlador a regular las variables de cada caso.

Para el motor de CD, las pruebas consisten en:

- Cambios en el punto de ajuste (intervalo de cero a uno)
- Disturbios en la velocidad del rotor (Aumento de carga, simulada con la disminución de velocidad, intervalo de 0 a 0.5)
- Comparación de los controladores con ITAE, IAE e ISE en cada prueba

4.3 RESULTADOS DEL SISTEMA DE CONTROL MÁQUINA DE CORRIENTE DIRECTA

El sistema de control sintonizado PI responde adecuadamente como se analizó en el capítulo 2, la comparación general de este controlador con el neurocontrolador propuesto NNDIC se presentó de igual manera en dicho capítulo.

A continuación una serie de pruebas determinan el desempeño ante disturbios de ambos controladores, sus respectivos índices de control para los fines comparativos, utilizando el simulador HIL en lazo cerrado.

4.3.1 Prueba Entrada Escalón

Esta prueba consiste en la respuesta a una entrada escalón unitario, seleccionando el valor (1.00) en el selector horizontal marcado en el recuadro punteado de la figura 4.1 y figura 4.2. La figura 4.1 representa el simulador HIL utilizando el controlador PI en el sistema embebido, y la figura 4.2, representa el simulador HIL utilizando el controlador NNDIC en el sistema embebido.

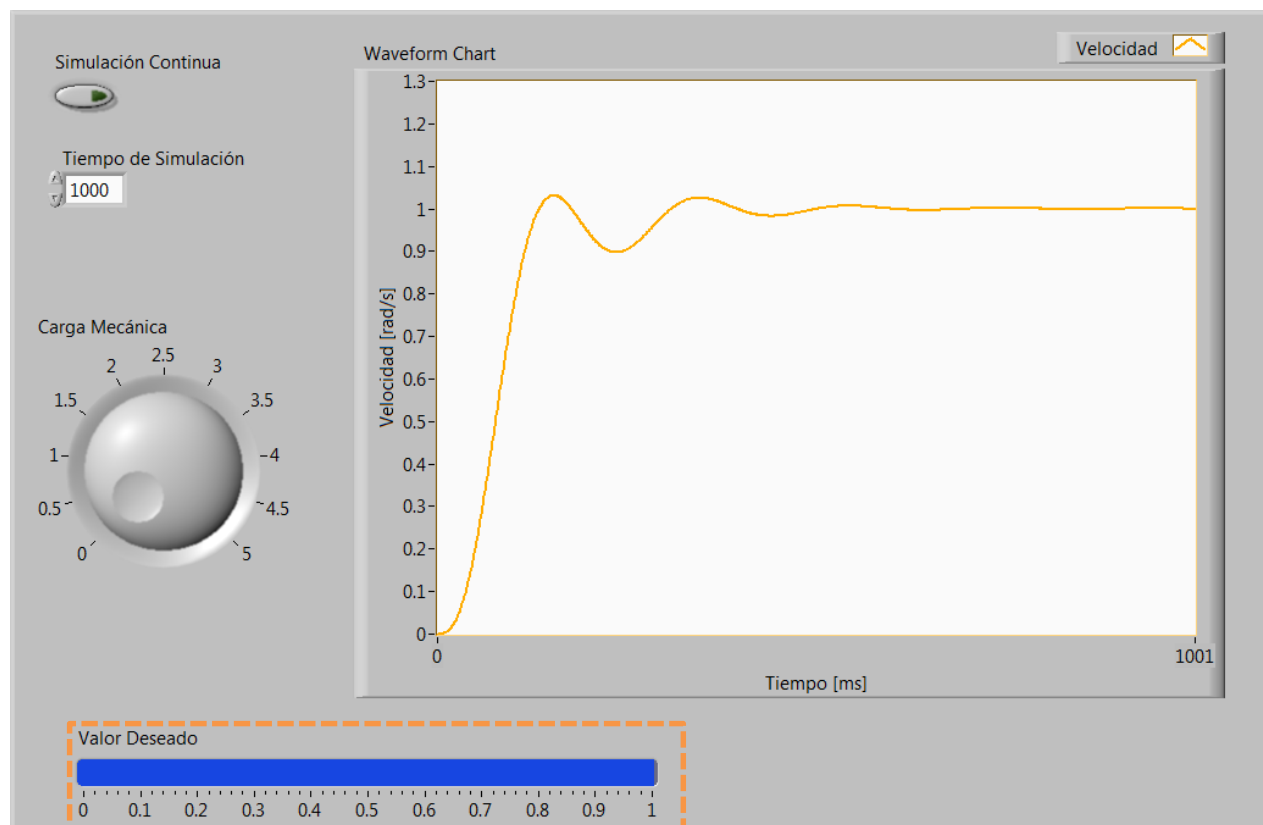


Figura 4.1 Prueba a entrada Escalón Control PI

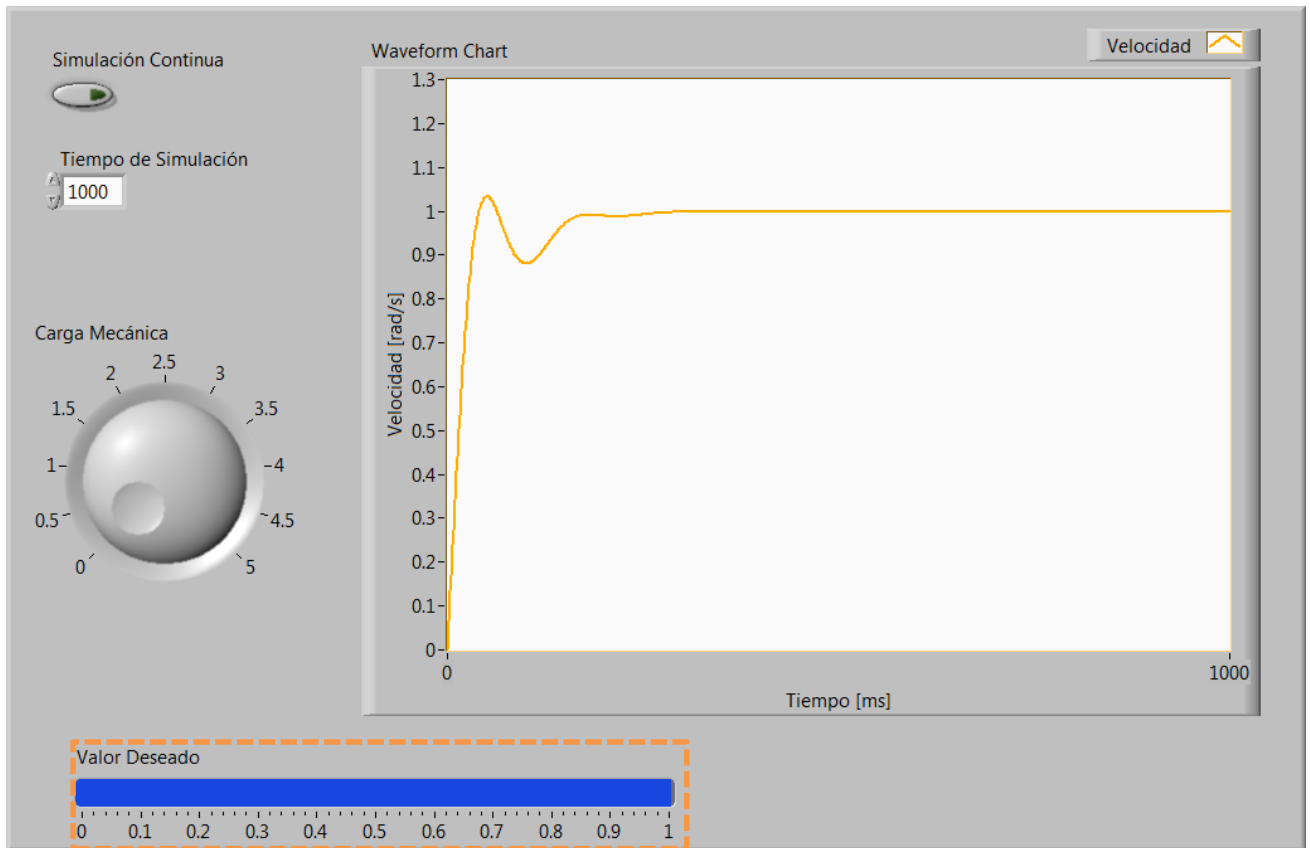


Figura 4.2 Prueba a entrada Escalón Control NNDIC.

Tabla 4.1 Comparativa Prueba Escalón

Índice	PI	NNDIC
ITAE	0.0056	0.0012
IAE	0.0124	0.0037
ISE	0.0045	0.0034

4.3.2 Prueba Cambio de Punto de Ajuste.

Esta prueba consiste en hacer un cambio en el punto de ajuste para verificar las características de servosistema de los controladores.

Cabe mencionar, que el NNDIC, al ser entrenado como regulador, es efectivo como servosistema únicamente en un intervalo definido de puntos de ajuste.

La figura 4.3 muestra un cambio en el punto de ajuste después del estado estacionario de la velocidad de motor, representado en una disminución a un valor de 0.5 en el sistema de control PI. La figura 4.4 muestra la misma prueba con el sistema de control NNDIC. La tabla 4.2 detalla los resultados para la comparación.

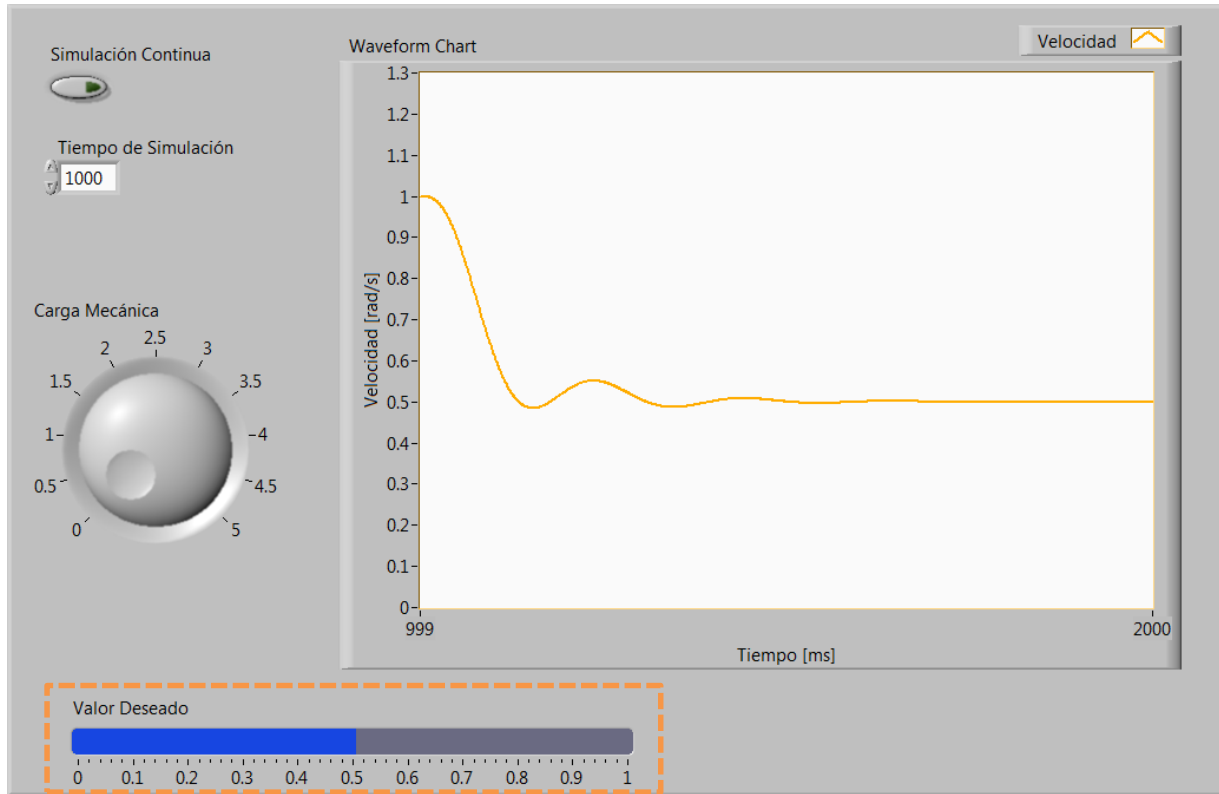


Figura 4.3 Cambio en el Punto de Ajuste a 0.5 Control PI.

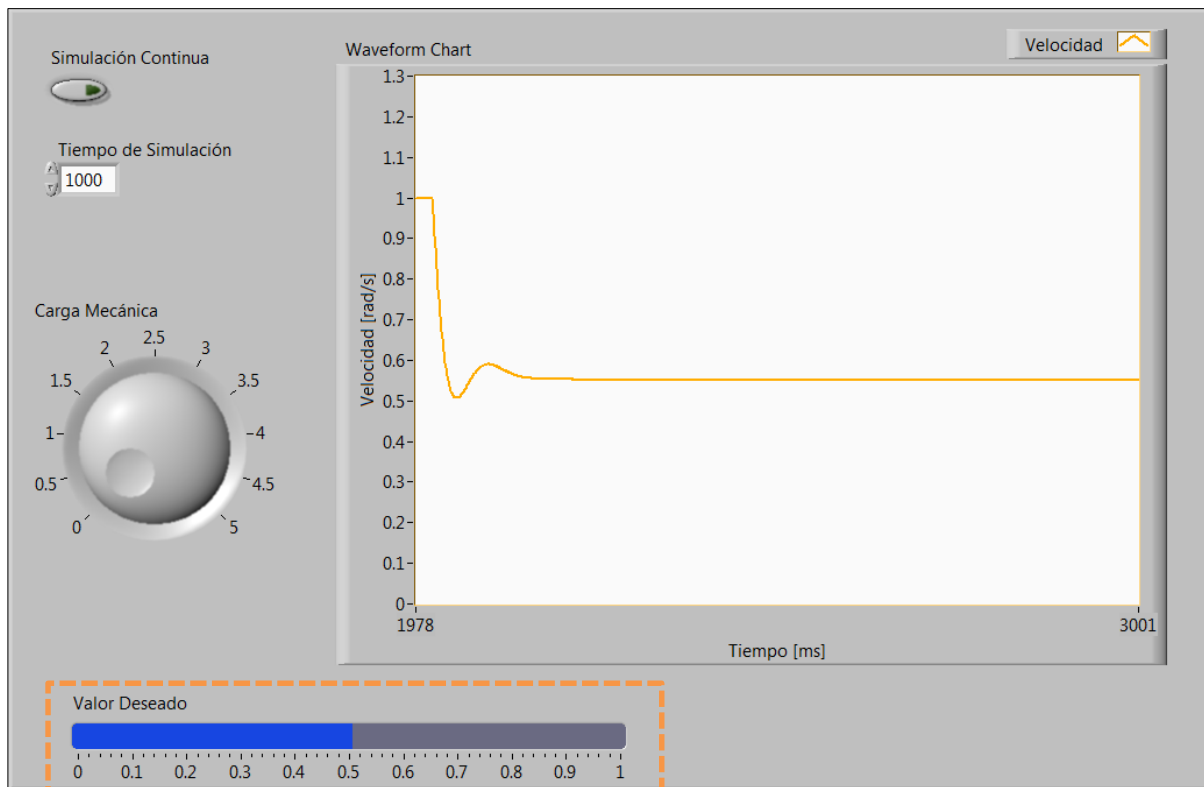


Figura 4.4 Cambio en el Punto de Ajuste a 0.5 Control NNDIC.

Tabla 4.2 Comparativa Cambio de Punto de Ajuste.

Índice	PI	NNDIC
ITAE	0.00987	0.0124
IAE	0.0032	0.3670
ISE	0.0185	0.7301
Error en Estado estacionario	0%	5%
Tiempo de Estabilización	0.35 s	0.10 s

4.3.3 Prueba de disturbio, aumento de carga mecánica de 1 N.m.

La prueba consiste en aumentar la carga mecánica del modelo del motor de CD, lo que produce una disminución en la velocidad que los sistemas de control deben compensar. Ambos sistemas operan en el punto de ajuste de 1, por lo que significa que la velocidad se encuentra en su punto nominal.

La figura 4.5 representa el disturbio y su compensación por el sistema de control PI.

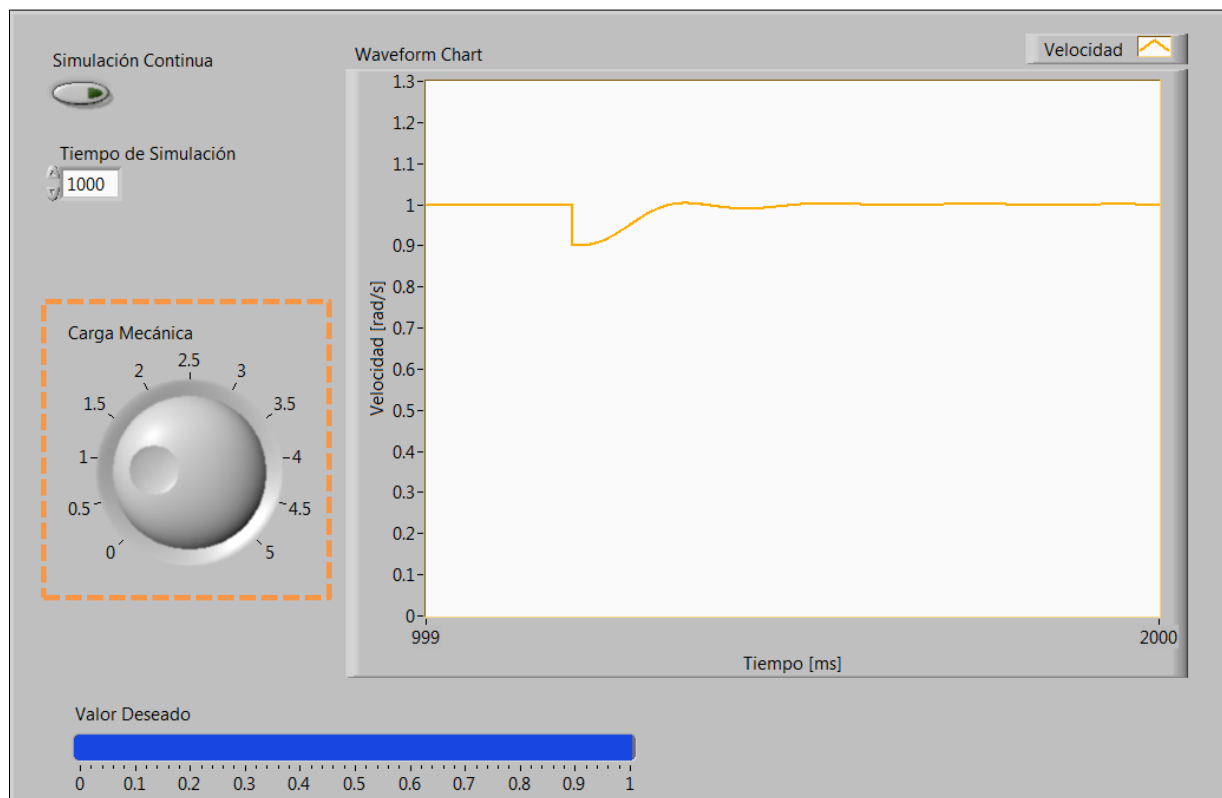


Figura 4.5 Aumento de carga de 1 N.m compensado por Control PI.

La figura 4.6 representa el disturbio y su compensación por el sistema de control NNDIC.

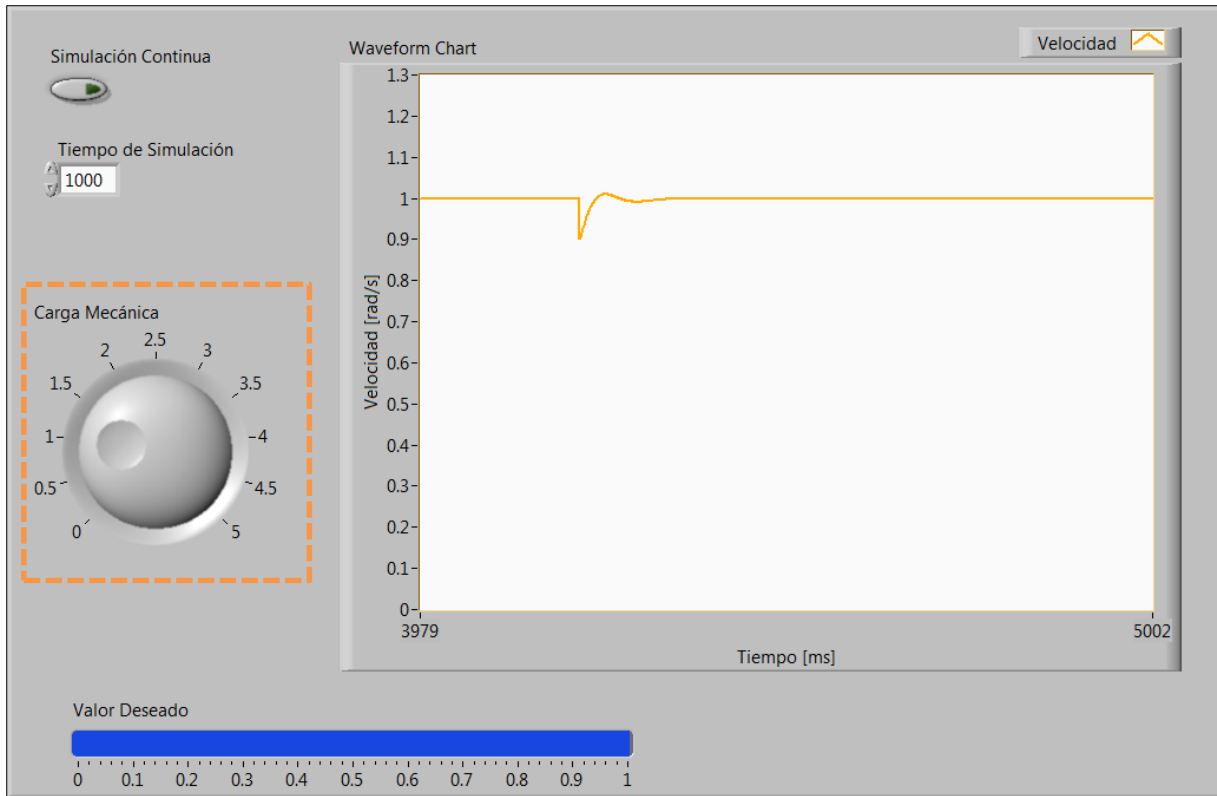


Figura 4.6 Aumento de Carga de 1.N.m compensado por Control NNDIC.

La tabla 4.3 detalla los resultados para la comparativa del desempeño de los controladores ante el disturbio provocado por el aumento de carga.

Tabla 4.3 Comparativa cambio de carga de 1 N.m.

Índice	PI	NNDIC
ITAE	0.0378	0.0016
IAE	0.0043	0.0020
ISE	0.0028	0.00045
Error en Estado estacionario	0%	0%
Tiempo de Estabilización	0.21 s	0.115 s

CAPÍTULO 5:

CONCLUSIONES

5.1 CONCLUSIONES

La motivación principal de la tesis fue el desarrollo de una simulación digital en tiempo real de la dinámica del caso de estudio, con el fin de verificar el desempeño de controladores convencionales e inteligentes embebidos en hardware para la regulación de la velocidad pese a disturbios.

El caso de estudio, Motor de Corriente Directa, es un sistema básico en la Ingeniería Eléctrica y un buen punto de partida en el desarrollo de sistemas de regulación de velocidad experimentales, en este caso, de simulaciones en tiempo real incluyendo hardware con los algoritmos embebidos. El modelo lineal del caso de estudio fue desarrollado a partir de las ecuaciones básicas, por lo que no representa trabajo adicional al ser un tópico altamente explorado.

La regulación de la velocidad utilizando el controlador PI se desarrolló de manera exitosa basándose en la metodología de sintonización de Ziegler Nichols. Esta metodología presenta la desventaja de requerir de una sintonización fina en función de los requisitos de control, lo cual implica un trabajo de sintonización experimental adicional.

La alternativa propuesta de utilizar la dinámica inversa como controlador produce resultados ideales teóricos. Para obtener un controlador que se base en esta teoría, se requiere del entrenamiento de una RN que identifique esta dinámica lo mejor posible. El entrenamiento de la RN como controlador por dinámica inversa requiere de un gran número de datos de entrada y salida así como de grandes tiempos de cómputo, los cuales deben ser considerados antes del diseño. Sin embargo, a pesar de estas dificultades, el resultado de utilizar estos controladores es altamente satisfactorio y recomendado para sistemas que requieran de un desempeño que los controladores clásicos no pueden ofrecer.

Los resultados de las simulaciones estáticas de los controladores diseñados en la tesis, PI y NNDIC, fueron exitosos en ambos casos. El desempeño del controlador NNDIC fue superior al control PI en la regulación de la velocidad del motor de CD, concordando con la teoría de control inteligente.

La RN sirvió además de modelo no lineal del motor de CD, permitiendo una identificación precisa de la dinámica del motor. Esta RN fue implementada de manera correcta en el software de instrumentación virtual LabVIEW, logrando así una simulación en tiempo real con tiempos muertos despreciables, esto debido a los cálculos en paralelo y la simplicidad de operaciones de las RN.

El sistema embebido FRDM-K64F cumple con los requerimientos de hardware necesarios tanto para la comunicación con software, como para la implementación de algoritmos de control en su memoria. El controlador PI fue implementado utilizando la ecuación de diferencias en tiempo discreto y su desempeño igualo al simulado en tiempo continuo. La implementación del controlador NNDIC, basado en una RN entrenada previamente con datos experimentales, represento un desafío mayor debido a la precisión de las variables utilizadas, es decir, la longitud de memoria requerida por los valores de los pesos de la RN, sin embargo, la programación es simple siguiendo el barrido hacia adelante convencional del perceptrón multicapa, dando como resultado una implementación exitosa de RN en el sistema embebido utilizado.

La comunicación RS232 demostró ser suficiente para enlazar el simulador con el sistema embebido, no se encontraron problemas de latencia relevantes. La interacción de ambos elementos permitió la visualización en tiempo real del control de la velocidad del caso de estudio. Las simulaciones de disturbios generados en LabVIEW en la interfaz gráfica desarrollada demostraron el desempeño correcto de ambos controladores funcionando en tiempo real.

La simulación HIL desarrollada en la tesis utilizando los elementos descritos a lo largo de esta, representa una herramienta conveniente al momento de verificar el desempeño de hardware involucrado en sistemas controlados. El caso de estudio de la tesis, el motor de CD, fue simulado en tiempo real de manera exitosa basándose en una RN, sin embargo, la metodología propuesta para este trabajo no se limita a este caso de estudio, es decir, teóricamente cualquier sistema que pueda ser identificado por una RN puede aplicarse al simulador HIL propuesto.

5.2 APORTACIONES

Las aportaciones de la tesis se pueden resumir en los siguientes puntos:

- Un sistema de simulación en tiempo real de modelos no lineales basados en RNA de un motor de CD.
- Una metodología de diseño de controles basados en RNA.
- Una metodología de la implementación de algoritmos de control en un sistema embebido.
- Un simulador HIL de dos casos de estudio de la ingeniería eléctrica.

5.3 TRABAJOS FUTUROS Y RECOMENDACIONES

Como trabajos futuros se recomienda lo siguiente:

- Diseño de controladores inteligentes adaptables
- Implementación de controladores inteligentes adaptables en un sistema embebido.
- Desarrollar una metodología más conveniente del análisis de estabilidad de sistemas de control inteligente.
- Desarrollar simulaciones HIL con otros sistemas embebidos y otras arquitecturas de control inteligente.
- Desarrollar simulaciones HIL para otros casos de estudio.

Las recomendaciones generales son:

- Seleccionar adecuadamente el sistema embebido a utilizar, tomando en cuenta la frecuencia de trabajo, la capacidad de la memoria y la longitud de las variables
- Desarrollar controladores clásicos con metodologías inteligentes, además de la sintonización clásica

Las recomendaciones al sintonizar controladores NNDIC:

- Utilizar entradas aleatorias de magnitud y frecuencias variables para la generación de patrones de entrenamiento
- Normalizar los datos de entrada y salida
- Entrenar diversas RN, con números de neuronas diferentes, con el fin de encontrar el mejor desempeño, un valor inicial recomendable es el doble de neuronas en la capa oculta que entradas, sin embargo no hay garantía.
- Utilizar el menor número de neuronas posibles acelera el entrenamiento y da mejores resultados al implementarse.
- Auxiliarse de software de entrenamiento de RN, como MATLAB, pues contienen optimización de código y tiempos de ejecución mucho menores que programación de algoritmos convencional.
- Utilizar mínimo dos pasos de integración anteriores en las entradas mejora el desempeño de la RN para identificar sistemas dinámicos.
- Al implementarse en un sistema digital, asegurarse de utilizar una alta precisión en los pesos obtenidos del algoritmo de entrenamiento.

REFERENCIAS

- [1] Ogata K., *Modern Control Engineering*, Prentice Hall, 1997.
- [2] Kuo B., *Sistemas de Control Automático*, Prentice Hall, 1997.
- [3] Nise N., *Control Systems Engineering*, Wiley, 2011.
- [4] Zurada J., *Introducción to Neural Network Systems*, West Publishing Company, 1992.
- [5] Hiyama T., Bevrani H., *Intelligent Automatic Generation Control*, CRC Press, 2011.
- [6] Nguyen H., Prasad N., Walker C., *A first Course in Fuzzy and Neural Control*, Chapman and Hall, 2003.
- [7] Psaltis D., Sideris A., Yamamura A., *A Multilayered Neural Network Controller*, IEEE Control Systems Conference, April 1988.
- [8] Rodríguez V., Garzón J., López J., *Control Neuronal por Modelo Inverso de un Servosistema usando algoritmos de entrenamiento Levenberg Marquard y Bayesiano*, VIII Congreso Internacional de la Universidad Tecnológica de Bolívar, 2009.
- [9] Bhongade S., *Artificial Neural Network based Automatic Generation Control*, Indian Institute of Technology, IPEC 2010.
- [10] Rai N., Rai B., *Neural Network based Closed Loop Speed Control Of DC Motor using Arduino*, International Journey of Engineering Trends, Vol. 4, Issue 2, 2013.
- [11] Norgaard M., Ravn O., Poulsen N., *Neural Networks for Control and Modelling of Dynamic Systems*, Springer, 2000.
- [12] Chapman S. *Máquinas Eléctricas*, Mc. Graw-Hill, 2005.
- [13] Wildi T., *Electrical Machines, Drives and Power Systems*, Prentice Hall, 2002.
- [14] Gallegos R., *Máquinas Eléctricas de Corriente Continua*, IPN, 2002.
- [15] Zaccarian L., *DC Motors, Dynamic Models and Control Techniques*, Pergamon Press, 2005.
- [16] Michigan University USA, *Control Tutorials for MATLAB and Simulink*, <http://ctms.engin.umich.edu/CTMS/index.php?aux=Home>, 2015.
- [17] Tewari A., *Modern Control Design with MATLAB and Simulink*, Wiley, 2002.
- [18] Klee H., Allen R., *Simulation of Dynamic Systems with MATLAB and Simulink*, CRC Press, 2011.
- [19] Åström K., *PID Controllers, Theory, Design and Tuning*, ISA, 1995.
- [20] Mansour T. *PID Control, Implementation and Tuning*, Intech, 2011.
- [21] ANSI NEMA MG 1-2011 *DC Motors and Generators*, Rev. 2012.
- [22] Dwyer A. *Handbook of PI and PID Controllers*, Imperial College Press, 2006.
- [23] Kugemoto H., *Development of Control Performance Analysis*, Sumitomo Chemical Corp, 2010.
- [24] Almurib H., Mat Isa A., *Direct Neural Network Control via Inverse Modelling*, The University of Nottingham, INTECH 2006.
- [25] Karsoliya S., *Aproximating Number of Hidden layer Neurons in Multiple Hidden Layer BPNN Architecture*, International Journal of Engineering Trends and Technology, Vol 3 Issue 6, 2012.
- [26] Crosbie R.E., *High-Speed Real-Time Simulation*, IEEE Modeling and Simulation Conference 2007.
- [27] Kohler C., *Enhancing Embedded Systems Simulation*, Springer 2011.
- [28] Panchal P., *PI Control of Level Control System using PLC and LabVIEW*, IEEE
- [29] Larsen R., *LabVIEW for Engineers*, Mc Graw-Hill, 2013.
- [30] Weerasoyja S., El-Sharkawi M., *Identification and Control of a DC Motor using Backpropagation Neural Networks*. IEEE Transactions of Energy Conversion 1991.
- [31] Rai N., Rai B., *Neural Network based Closed Loop Speed Control Of DC Motor using Arduino*, International Journey of Engineering Trends, Vol. 4, Issue 2, 2013.
- [32] Yu Q., Shu H., *Development of PID Neural Network Control System with Virtual Instrument*, IEEE Computer Society, 2008.
- [33] ARM mbed, *Handbook Serial* [online], <https://developer.mbed.org/handbook/Serial>, 2015.
- [34] Macias Macias M., *The "MBED" platform for teaching electronics applied to Product Design*, IEEE, TAEE 2014.
- [35] Aaron K., *Closed Loop Position Control System using LabVIEW*, Proceedings IEEE, SouthEastCon, 2012.
- [36] Gong S., *LabVIEW based automatic speed control of stepper motor*, International Conference of Electric Machines and Systems, IEEE, ICEMS 2009.

Referencias del Estado del Arte

- [37] Bavarian B., *Introduccion to Neural Networks for Intelligent Control*, IEEE Control Systems Magazine 1988.
- [38] Mohamed J., *Intelligent Control for Integrated Manufacturing*, IEEE Conference on Decision and Control, 1988.
- [39] Hunmo K., *Artificial Neural Network for Identification and Tracking control of a flexible joint Robot*, IEEE Symposium on System Theory, 1993.
- [40] Sbarbaro-Hofer D., *Neural Control of a Steel Rolling Mill*, IEEE International Symposium on Intelligent Control, 1992.
- [42] Palade V., *An approach on DC Driving System Neural Control by Inverse forward Model*, IEEE Electronics Conference, 2000.
- [43] Lee K., *Power System Stabilization using a free-model based inverse dynamic Neurocontroller*, IEEE International Conference on Neural Networks, 2002.

APÉNDICE A:

DATOS DEL MOTOR DE CD

Los datos de placa del motor de CD utilizado como caso de estudio se resumen en la tabla A.1.

Tabla A.1 Datos de Placa del Motor de CD.

Datos	Valor	Unidades
Potencia de Salida	1.7	kW
Velocidad Nominal	400	rad/s
Par Nominal	4	N.m
Voltaje de Armadura Nominal	220	VCD
Corriente de Armadura Nominal	8	A
Voltaje de Campo Nominal	160	VCD
Corriente de Campo Nominal	0.6	A

Los parámetros eléctricos y mecánicos del motor de CD utilizado se observan en la tabla A.2.

Tabla A.2 Parámetros del Motor de CD.

Parámetros	Valor	Unidades
Inductancia de Armadura	0.1	H
Inductancia de Campo	17.25	H
Resistencia de Armadura	2.5	Ω
Resistencia de Campo	228.2	Ω
Inductancia Mutua	0.8513	H
Momento de Inercia del Rotor	0.0022	Kg.m ²
Coeficiente de Fricción Viscosa	0.01	N.m.s

APÉNDICE B:

FUNDAMENTOS TEÓRICOS DE LAS REDES NEURONALES ARTIFICIALES

B.1 INTRODUCCIÓN

Los organismos biológicos son capaces de aprender todo el tiempo y pueden ejecutar tareas complejas. Debido a la generalización y estímulos externos las neuronas biológicas adquieren conocimiento y aprenden de él.

Las redes neuronales artificiales, son modelos matemáticos que están basados en nuestro sistema nervioso. Las redes neuronales artificiales son útiles al establecer relaciones entre entradas y salidas de cualquier sistema.

Una red neuronal es un conjunto de neuronas artificiales, las neuronas artificiales son modelos de las neuronas biológicas en su forma simple. Las neuronas biológicas son las unidades fundamentales que procesan información en el sistema nervioso.

El modelo matemático de una red neuronal está basado en:

- Las neuronas son fundamentales en el sistema nervioso cuando ocurre el procesamiento de información.
- La información entra en forma de señales que pasan entre neuronas a través de enlaces de conexión.
- Cada enlace de conexión tiene su propio peso que procesa las señales entrantes.
- Cada neurona tiene una acción interna, la cual depende de un sesgo o umbral de disparo que es reflejado en una función de activación.

En la siguiente ecuación:

$$y = f \left(\sum_{i=1}^n w_i x_i - b \right)$$

Donde:

x_i : señales de entrada

w_i : pesos

b : umbral de disparo

f : función de activación

La entrada a la neurona es procesada por su respectivo peso, el sesgo y la función de activación definida creando así una respuesta y que seguirá su camino a lo largo del resto de las capas de la red neuronal.

El modelo propuesto por McCulloch y Pitts es representado como sigue en la figura B.1.

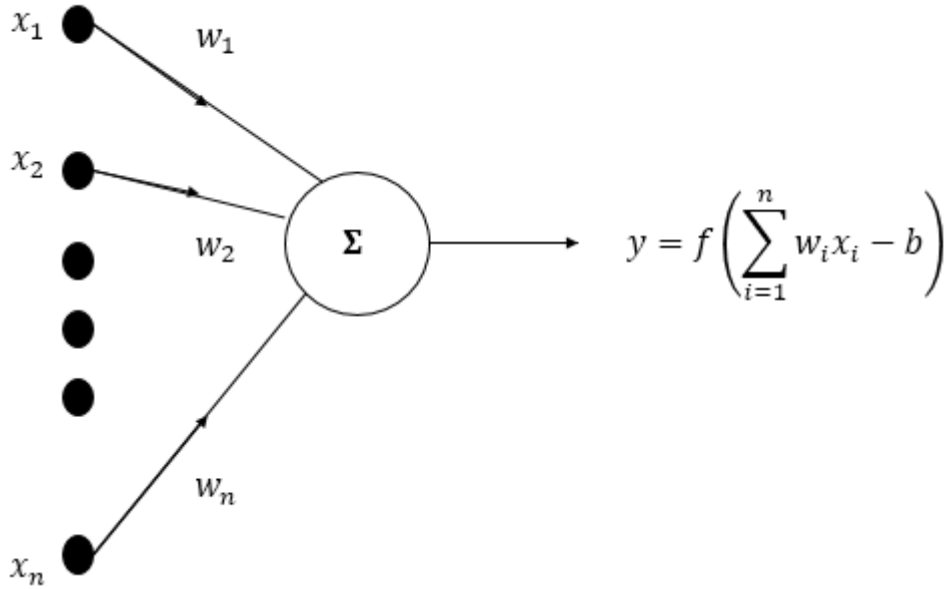


Figura B.1 Modelo de McCulloch Pitts de la neurona artificial

En donde las salidas son representadas de forma binaria y f es la función escalón que esta definido por:

$$f(x) = \begin{cases} 1 & \text{si } x \geq 0 \\ 0 & \text{si } x < 0 \end{cases}$$

Siendo la activación de la neurona:

$$f\left(\sum_{i=1}^n w_i x_i - b\right) = \begin{cases} 1 & \text{si } \sum_{i=1}^n w_i x_i \geq b \\ 0 & \text{si } \sum_{i=1}^n w_i x_i < b \end{cases}$$

Los parámetros que caracterizan a una neurona artificial son:

$$\theta = (w_1, w_2, \dots, w_n, b, f)$$

En el siguiente esquema la red neuronal comprende una capa de entrada y una capa de salida que consisten de neuronas. Lo cual es visto como una simple capa de redes neuronales debido a que la capa de entrada no es una capa de neuronas y por lo tanto el cálculo no ocurre en los nodos de entrada. A esta simple capa de red neuronal se le llama perceptrón.

Cuando una red neuronal tiene más de una capa de neuronas es llamada red neuronal multicapa. En una red multicapa las neuronas de una capa tienen conexiones pesadas con neuronas de la siguiente capa pero no tienen conexiones entre neuronas de la misma capa. Para el caso de la red multicapa la función de activación puede ser diferente para otras neuronas.

En el perceptrón la capa de entrada tiene $n + 1$ nodos, la capa media es llamada capa oculta, esto es debido a que no se puede observar directamente la salida, esta capa tiene $p + 1$ nodos, mientras que la capa de salida tiene m nodos y es llamada red neuronal " $n - p - m$ ".

B.2 CAPACIDAD DE APRENDIZAJE

Las redes neuronales son entrenadas mediante aprendizaje supervisado. Estas son diseñadas para ejecutar una tarea específica, por lo que la arquitectura a utilizar depende del problema planteado. Para el entrenamiento de la red neuronal es necesario que se tenga la tarea bien definida. Lo más importante para la práctica es el entrenamiento de la red neuronal antes de que sea puesta en uso.

El problema del aprendizaje, reside en elegir una función de un grupo de funciones dadas dependiendo del criterio presentado. Lo primero que se debe tener en cuenta es la función que puede ser representada o implementada por una red neuronal, conociendo lo anterior podemos determinar el diseño que se ajusta más para la red neuronal.

B.3 REGLA DELTA

Es un algoritmo de aprendizaje para redes neuronales de una capa, es el precursor del algoritmo de retropropagación siendo este empleado para redes neuronales de múltiples capas.

La regla delta consiste en obtener un indicador de la ejecución promedio de la red. Los algoritmos de aprendizaje pretenden cambiar los pesos para que la salida cada vez sea más parecida a los objetivos.

La medida de ejecución promedio es:

$$E = \sum_{q=1}^N E^q$$

Siendo:

$$E^q = \frac{1}{2} \sum_{i=1}^m (y_i^q - o_i^q)^2$$

Donde:

$$\begin{aligned} o^q &: \text{salida} \\ y^q &: \text{objetivos para toda } q = 1, 2, \dots, m \\ x^q &= \text{entrada a la red} \end{aligned}$$

El objetivo es minimizar E con los pesos w_{ij} de la red. Siendo w_{ij} los pesos del nodo j a la neurona de salida i . Para utilizar el método de gradiente descendente es necesario que los pesos w_{ij} y los elementos de la red sean diferenciables.

Resultando:

$$o_i^q = f_i \left(\sum_{j=0}^n w_{ij} x_j^q \right)$$

La función de activación f_i de la neurona debe ser una función diferenciable. Por ejemplo, la función sigmoide, tangencial, tangente hiperbólica, lineal, etc.

El error E es una función de las variables w_{ij} , para $i=1, 2, 3, \dots, m$, y $j=1, 2, \dots, n$.

Por lo que el gradiente ∇E de E en el punto w con componentes w_{ij} es el vector de derivadas parciales $\frac{\partial E}{\partial w_{ij}}$.

Para minimizar el error E se mueve al valor negativo de ∇E , permitiendo actualizar cada peso w_{ik} como:

$$w_{jk} \rightarrow w_{jk} + \Delta w_{jk}$$

Donde:

$$\Delta w_{jk} = -\eta \frac{\partial E}{\partial w_{ij}}$$

Siendo $\eta > 0$ un número llamado tasa de aprendizaje.

Teniendo:

$$\frac{\partial E}{\partial w_{ij}} = \sum_{q=1}^N \frac{\partial E^q}{\partial w_{ij}}$$

Y

$$\frac{\partial E^q}{\partial w_{ij}} = \frac{\partial}{\partial w_{ij}} \left(\frac{1}{2} \sum_{i=1}^m (y_i^q - o_i^q)^2 \right)$$

$$\frac{\partial E^q}{\partial w_{ij}} = (o_j^q - y_j^q) \frac{\partial}{\partial w_{ij}} f_i \left(\sum_{i=0}^n w_{ji} x_i^q \right)$$

Ya que:

$$\frac{\partial}{\partial w_{ij}} (o_i^q - y_i^q) = 0 \quad \text{para } i \neq j$$

Siendo:

$$o_j^q = f_j \left(\sum_{i=0}^n w_{ji} x_i^q \right)$$

Entonces:

$$\frac{\partial E^q}{\partial w_{jk}} = x_k^q (o_j^q - y_j^q) f_j' \left(\sum_{i=0}^n w_{ji} x_i^q \right)$$

$$\frac{\partial E^q}{\partial w_{jk}} = \delta_j^q \cdot x_k^q$$

Donde:

$$\delta_j^q = (o_j^q - y_j^q) f_j' \left(\sum_{i=0}^n w_{ji} x_i^q \right)$$

Para la j neurona de salida.

Por lo que:

$$\Delta w_{jk} = \sum_{q=1}^N \Delta^q w_{jk}$$

$$= -\eta \sum_{q=1}^N \frac{\partial E^q}{\partial w_{jk}}$$

$$= \sum_{q=1}^N -\eta \delta_j^q x_k^q$$

Con la regla delta se busca el vector de pesos w^* tal que el gradiente de E a w^* sea cero. Para algunas neuronas de una capa w^* es el único mínimo absoluto de $E(w)$.

B.4 ALGORITMO DE RETROPROPAGACIÓN

El algoritmo de retropropagación es una generalización de la regla delta.

Cuando el patrón de entrada de aprendizaje x^q se presenta, con la regla delta se observa la actualización de los pesos w_{ik} para una red neuronal de una capa, por lo que se necesita:

$$\Delta w_{ik}^q = -\eta \delta_i^q x_k^q$$

Se necesitará calcular la siguiente función:

$$\delta_i^q = (o_i^q - y_i^q) f' \left(\sum_{j=0}^n w_{ij} x_j^q \right)$$

Como se tiene el valor y_i^q , el cual es conocido como la salida objetivo para el nodo de salida i .

Considerando una red neurona de dos capas $(n - m - p)$ por simplicidad, pareciera que no se puede actualizar un peso w_{ik} en el enlace de conexión del nodo de entrada k , en la capa de entrada de la neurona oculta i , de la capa oculta debido a que no se cuenta con el patrón objetivo de la neurona i de la entrada x^q . Los patrones y^q son patrones objetivos de neuronas en la capa de salida y no en la capa oculta.

Entonces cuando se busca en los errores $(o_i^q - y_i^q)^2$ en la capa de salida no se puede detectar que neuronas son las responsables.

Para actualizar estos pesos, es suficiente con calcular $\frac{\partial E}{\partial o_i}$, que es la derivada parcial del error global E , con respecto a la salida o_i , $i = 1, \dots, p$. La estrategia del gradiente descendente se puede aplicar a funciones de activación diferenciables para encontrar la configuración de los pesos de la red neuronal w^* que minimiza el error E .

A continuación se muestra la fórmula de actualización de pesos para una red neuronal de dos capas, siendo v_{ji} la conexión de enlace de la neurona oculta i a la neurona de salida j , y w_{ik} los pesos de la conexión de enlace del nodo de entrada k a la neurona oculta i .

La regla de actualización para los v_{ji} 's es el mismo que en la regla delta, viendo ahora la capa oculta como una capa de entrada. Para neuronas de salida j y neurona oculta i , el peso v_{ji} es actualizado usando:

$$\Delta v_{ji} = \sum_{q=1}^N \Delta^q v_{ji}$$

$$= -\eta \sum_{j=1}^N \frac{\partial E^q}{\partial v_{ji}}$$

$$= \sum_{j=1}^N (-\eta) \delta_j^q z_i^q$$

Donde z_i^q es la entrada a la red de la neurona oculta i
Siendo

$$z_i^q = f_i \left(\sum_{k=0}^n w_{ik} x_k^q \right)$$

Y

$$\delta_j^q = (o_j^q - y_j^q) f'_i \left(\sum_{k=1}^m v_{jk} z_k^q \right)$$

Para una neurona i y nodo de entrada k , los pesos w_{ik} son actualizados como:

$$\Delta^q w_{ik} = -\eta \frac{\partial E^q}{\partial w_{ik}}$$

Con

$$\frac{\partial E^q}{\partial w_{ik}} = \frac{\partial E^q}{\partial o_i^q} \frac{\partial o_i^q}{\partial w_{ik}}$$

Donde o_i^q es la salida de la neurona oculta i .

$$o_i^q = f_i \left(\sum_{l=0}^n w_{il} x_l^q \right) = z_i^q$$

Tenemos:

$$\frac{\partial o_i^q}{\partial w_{ik}} = f'_i \left(\sum_{l=0}^n w_{il} x_l^q \right) x_k^q$$

Dejando la capa oculta $\delta_i^q = \frac{\partial E^q}{\partial o_i^q}$

Siendo que:

$$\delta_i^q = \frac{\partial E^q}{\partial o_i^q} = \sum_{j=1}^p \frac{\partial E^q}{\partial o_j^q} \frac{\partial o_j^q}{\partial o_i^q}$$

Donde j es la capa de salida. Los $\frac{\partial E^q}{\partial o_j^q}$ son conocidos a partir de cálculos anteriores usando la regla delta.

Entonces:

$$o_i^q = f_j' \left(\sum_{l=1}^m v_{il} z_l^q \right) v_{ji}$$

Por lo tanto el δ_i^q , para las neuronas ocultas i , son calculados a partir de los valores ya conocidos δ_j^q , para toda j en la capa de salida.

Debido a esto la regla delta generalizada es llamada algoritmo de retropropagación.

B.5 CONTROL NEURONAL

Una red neuronal artificial (RNA), es una red sintética que intenta emular el comportamiento de las redes neuronales biológicas, con el fin de servir de algoritmo matemático para resolver problemas en diferentes áreas como la medicina, la economía y en este caso, la ingeniería.

El problema del control automático consiste básicamente en proveer, a un sistema dinámico que se desea controlar, con una señal que ajuste su respuesta de manera deseada.

El método más común para resolver el problema de control, es emplear un algoritmo que utiliza la señal de salida (señal medida) y la compara con la señal deseada, para producir una señal de error, la cual es procesada por el algoritmo y se determina la señal de control, que actúa directamente con el sistema modificando su comportamiento, y reduciendo en cada iteración el error de la respuesta (Señal de Salida), como se ilustra en la figura A.2.

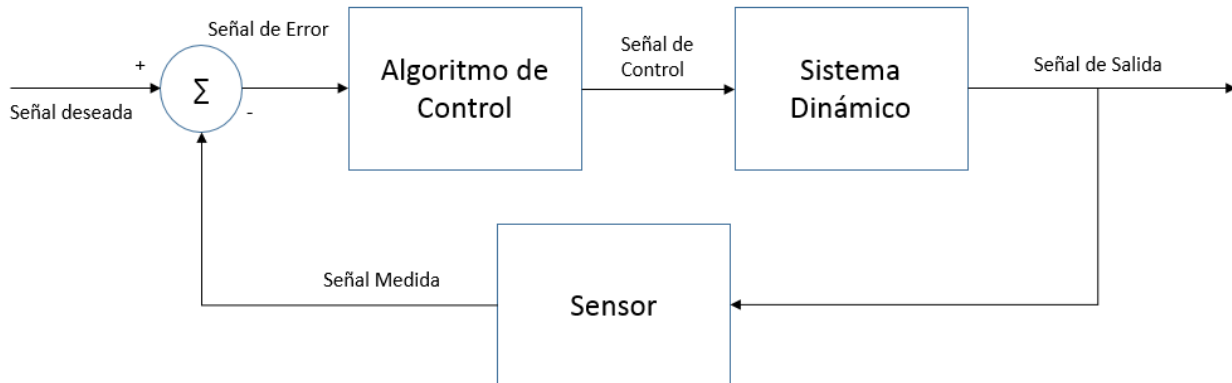


Figura B.2 Sistema de Control

Sin embargo, este método no es totalmente eficaz. Depende completamente del conocimiento del sistema dinámico y de un modelo preciso.

Al diseñar el algoritmo de control se depende del modelo del sistema, que al aumentar su complejidad, aumenta la precisión del control, aunque también la dificultad del diseño. Esto suponiendo que se cuenta con un modelo lineal e invariante en el tiempo. Para sistemas no lineales e invariantes en el tiempo, la situación se complica, y aunque existen métodos para resolver la cuestión, por ejemplo, utilizando retroalimentación y diseño de control en espacio de estados, no resuelven de manera total el problema y representan grandes retos al momento de diseñar.

Una técnica alternativa es el denominado control por dinámica inversa, la cual consiste en desarrollar un algoritmo que se comporte como el inverso de la dinámica del sistema a controlar, colocarlo en serie con el sistema, y obtener un resultado ideal, donde no existe error.

Para implementar esta idea se sugiere un algoritmo que contenga la dinámica deseada antes de la dinámica inversa, para que sea esta la que actúe en la respuesta final del sistema como se muestra en la figura A.3.

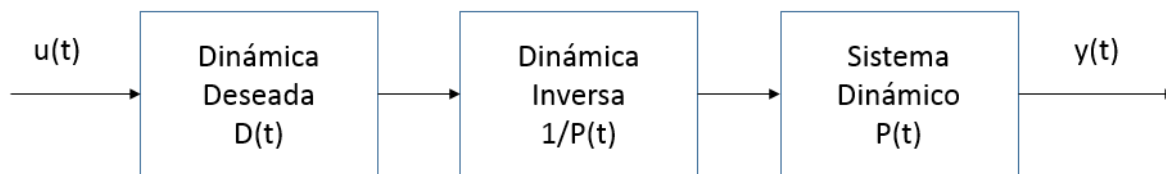


Figura B.3 Efecto de la Dinámica Inversa en Serie

Donde:

$u(t)$ = Señal de Entrada

$y(t)$ = Señal de Salida

Realizando un análisis de la respuesta se tiene, al ser señales en serie:

$$y(t) = P(t) \frac{1}{P(t)} D(t) u(t)$$

$$y(t) = D(t) u(t)$$

Obteniéndose así una respuesta ideal del sistema.

El problema de esta solución consiste en la dinámica inversa del sistema. Para que funcione depende de dos factores importantes [4,6].

1. La dinámica no varía durante el transcurso de operación normal del sistema.
2. La dinámica es completamente conocida antes del diseño.

Para satisfacer estos requerimientos es necesario realizar un modelo matemático de la Inversa de la dinámica del sistema, suponiendo que existe.

Estos inconvenientes son resueltos utilizando las RNA. Una red neuronal entrenada con los datos de la dinámica del sistema puede operar como un excelente y muy preciso modelo no lineal del sistema, además de poder reproducir con exactitud la dinámica inversa del modelo y actuar como se conoce a este tipo de algoritmos, neurocontrolador o neurocontrol [4,6].

La metodología del diseño consta de 4 pasos generales

1. Identificación del sistema
2. Identificación de la dinámica inversa del sistema
3. Entrenamiento de las RNA
4. Validación y ajustes

APÉNDICE C:

ÍNDICES DE DESEMPEÑO

C.1 INTRODUCCIÓN

Los índices de desempeño utilizados en la tesis corresponden a los más utilizados dentro de la teoría de control, que constituyen una herramienta para conocer el desempeño en una cantidad medible con fines comparativos.

Otro uso muy común es el utilizado en el diseño de controladores óptimos, donde se busca minimizar estos índices con algoritmos propiamente diseñados, tópico fuera del alcance de la tesis.

C.2 ITAE

El índice de desempeño ITAE consiste en la integral del valor absoluto del error, multiplicado por el tiempo. La ecuación C.1, describe su ejecución en el tiempo.

$$ITAE = \int_0^t t|e(t)|dt \quad (C.1)$$

El intervalo mostrado en la integral representa en las respuestas dinámicas analizadas, el tiempo de simulación, la ecuación C.2 representa el ITAE en tiempo discreto.

$$ITAE_z = \sum_0^t z|e(z)|dz \quad (C.2)$$

Donde dz equivale al paso de integración utilizado en las simulaciones digitales.

C.3 IAE

El índice de desempeño IAE consiste en la integral del valor absoluto del error. La ecuación C.3, describe su ejecución en el tiempo.

$$IAE = \int_0^t |e(t)|dt \quad (C.3)$$

El intervalo mostrado en la integral representa en las respuestas dinámicas analizadas, el tiempo de simulación, la ecuación C.4 representa el ITAE en tiempo discreto.

$$IAE_z = \sum_0^t |e(z)| dz \quad (C.4)$$

Donde dz equivale al paso de integración utilizado en las simulaciones digitales.

C.3 ISE

El índice de desempeño ISE consiste en la integral del error al cuadrado. La ecuación C.5, describe su ejecución en el tiempo.

$$ISE = \int_0^t e(t)^2 dt \quad (C.5)$$

El intervalo mostrado en la integral representa en las respuestas dinámicas analizadas, el tiempo de simulación, la ecuación C.6 representa el ITAE en tiempo discreto.

$$ITAE_z = \sum_0^t (e(z))^2 dz \quad (C.6)$$

Donde dz equivale al paso de integración utilizado en las simulaciones digitales.

APÉNDICE D:

CÓDIGOS DEL SISTEMA EMBEBIDO

D.1 INTRODUCCIÓN

Los programas utilizados en el microcontrolador se programaron en lenguaje C, utilizando el entorno de programación MBED, especializado en microcontroladores ARM. Este apéndice trata un resumen del sistema embebido utilizado, así como los códigos realizados para la simulación HIL de la tesis.

D.2 SISTEMA EMBEBIDO

La plataforma *Freedom* de *Freescaler*[®], permite el desarrollo de aplicaciones de prototipo rápido basadas en Microcontroladores. El FRDM-K64F es un diseño sofisticado de la serie *kinetis K*, construido en el ARM[®] Cortex[®]-M4 core.

D.2.1 Características del FRDM-K64F

El FRDM-K64F presenta las siguientes características con respecto a hardware:

Tabla D.1 Características de hardware del FRDM-K64F.

Característica	Valor
Frecuencia de operación	120 MHz
Memoria Flash	1MB
Memoria RAM	256KB
Interface USB	Conector doble micro-B USB
LED	RGB
Accesorios	Acelerómetro y magnetómetro
botones	2 botones
Opciones de alimentación flexible	USB OpenSDAv2, USB K64 y fuente externa
Comunicación	Ethernet, USB

D.3 CÓDIGOS DEL CASO DE ESTUDIO MOTOR DE CD

D.3.1 Control PI

Las ganancias calculadas del controlador como aparecen en el capítulo 2, son

$$K_p = 0.6213 \text{ y } K_i = 23.014$$

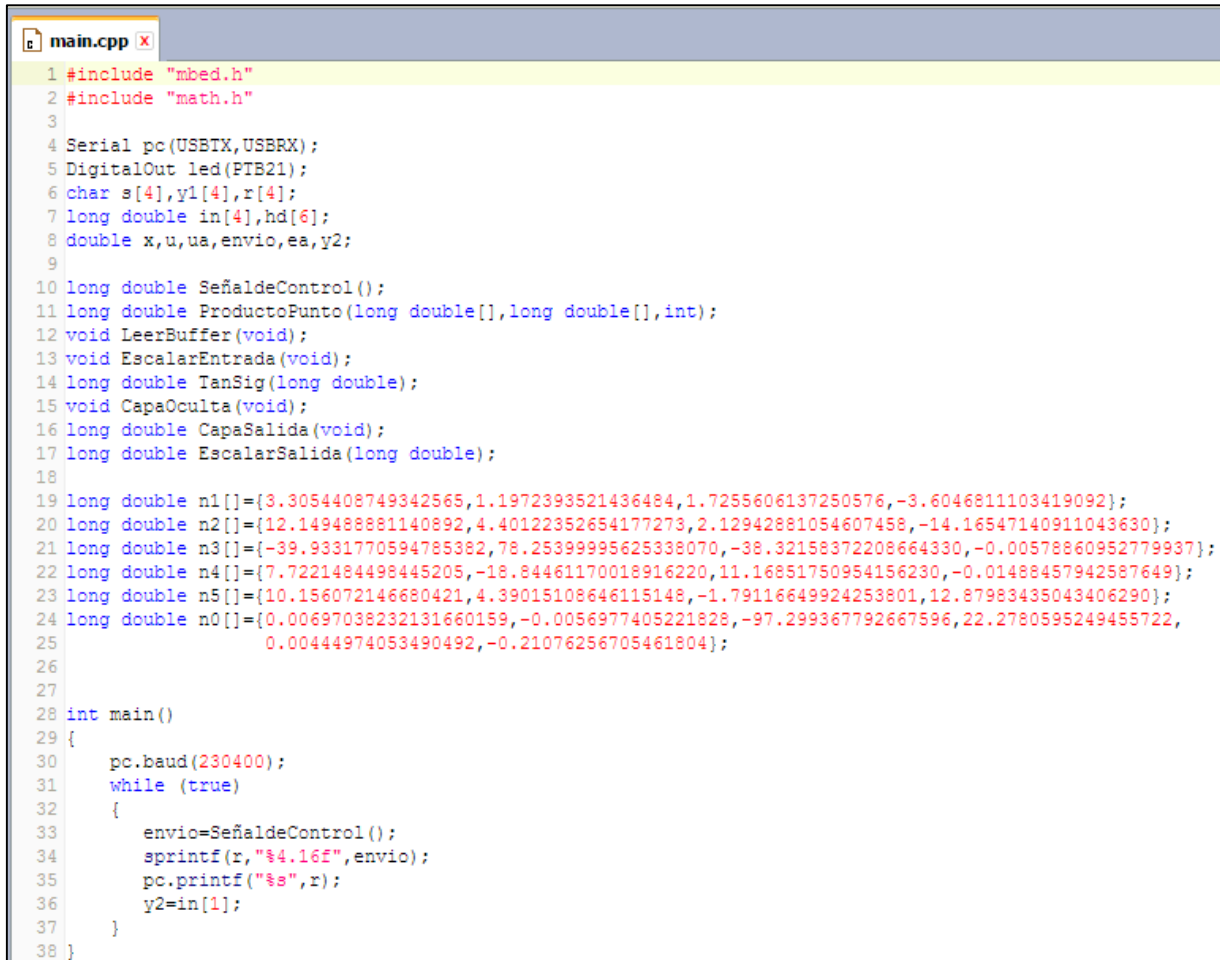
El código mostrado en la figura D.1.

```
1 #include "mbed.h"
2 #define KP 0.6213
3 #define KI 23.014
4 //Variables
5 //x->Valor numerico recibido
6 //u->Señal de Control
7 //ua->Señal de Control anterior
8 //envio->Variable auxiliar para envio
9 //ea->Error anterior
10 Serial pc(USBTX,USBRX);
11 DigitalOut led(PTB21);
12 char r[4];
13 double x,u,ua,envio,ea;
14 double SeñaldeControl(double);
15 ///////////////////////////////////////////////////
16 int main()
17 {
18     led=1;
19     pc.baud(230400);
20     while (true)
21     {
22         pc scanf("%s",r);
23         led=0;
24         x=atof(r);
25         envio=SeñaldeControl(x);
26         sprintf(r,"%4.16f",envio);
27         pc.printf("%s",r);
28         wait(0.001);
29         led=1;
30     }
31 }
32 ///////////////////////////////////////////////////
33 double SeñaldeControl(double error)
34 {
35     u=KP*(error-ea)+KI*error+ua;
36     ua=u;
37     ea=error;
38     if(u>=2)
39         return 2;
40     else if(u<=0)
41         return 0;
42     else
43         return u;
```

Figura D.1 Código Control PI Motor de CD.

D.3.2 Control NNDIC

El control basado en una RNA, se codifico con los algoritmos de la teoría de las Redes neuronales perceptrón multicapa, y entrenada con el algoritmo de retropropagación, el resultado de la red se programó utilizando lenguaje C en el sistema embebido. La figura D.2 muestra el código implementado de la función principal.



```
1 #include "mbed.h"
2 #include "math.h"
3
4 Serial pc(USBTX,USBRX);
5 DigitalOut led(PTB21);
6 char s[4],y1[4],r[4];
7 long double in[4],hd[6];
8 double x,u,ua,envio,ea,y2;
9
10 long double SeñaldeControl();
11 long double ProductoPunto(long double[],long double[],int);
12 void LeerBuffer(void);
13 void EscalarEntrada(void);
14 long double TanSig(long double);
15 void CapaOculta(void);
16 long double CapaSalida(void);
17 long double EscalarSalida(long double);
18
19 long double n1[]={3.3054408749342565,1.1972393521436484,1.7255606137250576,-3.6046811103419092};
20 long double n2[]={12.149488881140892,4.40122352654177273,2.12942881054607458,-14.16547140911043630};
21 long double n3[]={-39.9331770594785382,78.25399995625338070,-38.32158372208664330,-0.00578860952779937};
22 long double n4[]={7.7221484498445205,-18.84461170018916220,11.16851750954156230,-0.01488457942587649};
23 long double n5[]={10.156072146680421,4.39015108646115148,-1.79116649924253801,12.87983435043406290};
24 long double n0[]={0.00697038232131660159,-0.0056977405221828,-97.299367792667596,22.2780595249455722,
25 0.00444974053490492,-0.21076256705461804};
26
27
28 int main()
29 {
30     pc.baud(230400);
31     while (true)
32     {
33         envio=SeñaldeControl();
34         sprintf(r,"%4.16f",envio);
35         pc.printf("%s",r);
36         y2=in[1];
37     }
38 }
```

Figura D.2 Función Principal del Código de NNDIC

Las funciones auxiliares del código son las siguientes:

- SeñaldeControl

Que incluye dentro de sí misma las siguientes:

- LeerBuffer
- EscalarEntrada
- CapaOculta

- CapaSalida
- EscalarSalida

Las figuras D.3, D.4, D.5, D.6, D.7, D.8 muestran la sección del código de las funciones mencionadas respectivamente.

```
39 //////////////////////////////////////////////////
40 long double SeñaldeControl()
41 {
42     long double dummy;
43     LeerBuffer();
44     EscalarEntrada();
45     CapaOculta();
46     dummy=CapaSalida();
47     return (EscalarSalida(dummy));
48 }
49 //////////////////////////////////////////////////
```

Figura D.3 Función SeñaldeControl

```
51 //////////////////////////////////////////////////
52 void LeerBuffer()
53 {
54     pc.scanf("%s",s);
55     pc.scanf("%s",y1);
56     led=0;
57     in[0]=atof(s)+0.42;
58     in[1]=atof(y1);
59     in[2]=y2;
60     in[3]=(long double)1;
61 }
62 //////////////////////////////////////////////////
```

Figura D.4 Función LeerBuffer

```
64 //////////////////////////////////////////////////
65 void EscalarEntrada()
66 {
67     in[0]=(in[0]+0.79282884128457176)*(0.72737626933610373023446187778991)-1.0000000000000000;
68     in[1]=(in[1]+0.79282884128457176)*(0.72737626933610373023446187778991)-1.0000000000000000;
69     in[2]=(in[2]+0.79282884128457176)*(0.72737626933610373023446187778991)-1.0000000000000000;
70 }
71 //////////////////////////////////////////////////
```

Figura D.5 Función EscalarEntrada

```
73 //////////////////////////////////////////////////
74 void CapaOculta()
75 {
76     hd[0]=ProductoPunto(in,n1,5);
77     hd[0]=TanSig(hd[0]);
78     hd[1]=ProductoPunto(in,n2,5);
79     hd[1]=TanSig(hd[1]);
80     hd[2]=ProductoPunto(in,n3,5);
81     hd[2]=TanSig(hd[2]);
82     hd[3]=ProductoPunto(in,n4,5);
83     hd[3]=TanSig(hd[3]);
84     hd[4]=ProductoPunto(in,n5,5);
85     hd[4]=TanSig(hd[3]);
86     hd[5]=(long double)1;
87 }
88 //////////////////////////////////////////////////
```

Figura D.6 Función CapaOculta


```

90 //////////////////////////////////////////////////
91 long double CapaSalida()
92 {
93     return (ProductoPunto(hd,n0,6));
94 }
95 //////////////////////////////////////////////////

```

Figura D.7 Función CapaSalida

```

97 //////////////////////////////////////////////////
98 long double EscalarSalida(long double x)
99 {
100     long double dummy;
101     dummy=(x-1.0000000000000000)*(1.24679215942924996)-(0.69651253516368172);
102     if(dummy>=2)
103         return 2;
104     else if(dummy<=0)
105         return 0;
106     else
107         return dummy;
108 }
109 //////////////////////////////////////////////////

```

Figura D.8 Función EscalarSalida

Finalmente, las funciones utilizadas como herramienta, ProductoPunto y TanSig, requeridas por cada neurona como el operador principal y la función de activación respectivamente, se muestran en las figuras D.9 e D.10.

```

113 //////////////////////////////////////////////////
114 long double ProductoPunto(long double x[],long double y[],int num)
115 {
116     long double dot=0;
117     int i;
118     for(i=0;i<num;i++)
119     {
120         dot=dot+x[i]*y[i];
121     }
122     return dot;
123 }
124 //////////////////////////////////////////////////

```

Figura D.9 Función ProductoPunto

```

127 //////////////////////////////////////////////////
128 long double TanSig(long double x)
129 {
130     long double y;
131     y=2.0000000000000000/(1.0000000000000000+exp(-2.0000000000000000*x))-1.0000000000000000;
132     return y;
133 }
134 //////////////////////////////////////////////////

```

Figura D.10 Función TanSig

NOTA: Las variables denominadas *dummy* que aparecen en las funciones tienen un uso meramente auxiliar, por lo tanto no son mencionadas propiamente en el capítulo 3.

APÉNDICE E:

PROGRAMACIÓN DEL SIMULADOR EN LABVIEW

E.1 INTRODUCCIÓN

Tomando en cuenta que el desarrollo de este programa fue en paralelo, utilizando la programación gráfica de LabVIEW, el diagrama de flujo de la figura se toma de base para explicar la ejecución del programa etapa por etapa.

Ambos simuladores de los casos de estudio, motor de CD y Turbina Generador, con ambos controladores PI y NNDIC, fueron implementados en LabVIEW utilizando esta metodología.

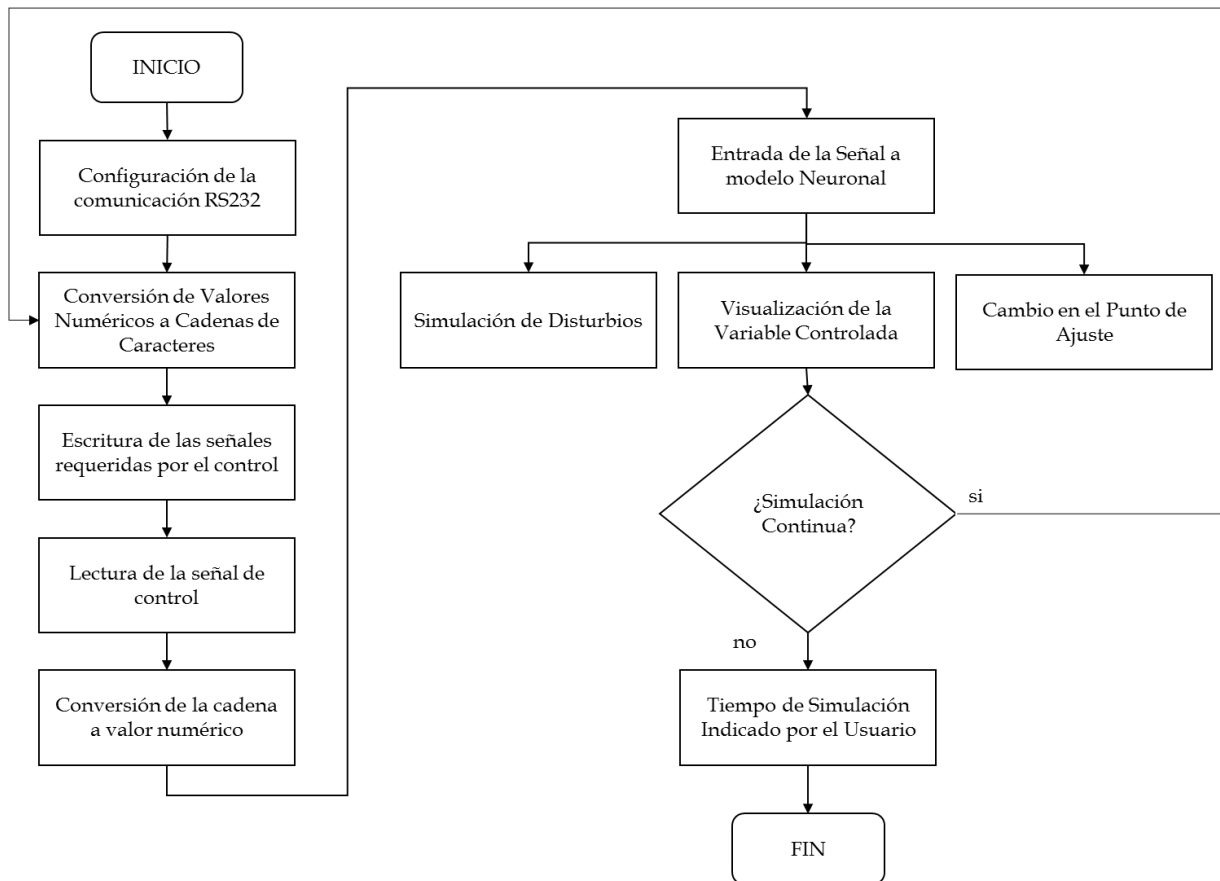


Figura E.1 Diagrama de Flujo Programa de LabVIEW

E.2 EJECUCIÓN DEL PROGRAMA

E.2.1 Configuración de la Comunicación RS232

La comunicación serial con el protocolo RS232, requiere de una serie de parámetros que deben especificarse con el fin de tener sincronía, el más importante es la tasa de transferencia de bytes, denominada bauds, que en el caso de estudio, es de 230400.

Otro parámetro importante es el puerto de comunicación, que en el caso de la tesis es COM3, este puerto depende de la computadora que se utilicé.

La figura muestra la sección de código en LabVIEW que configura el puerto serial.

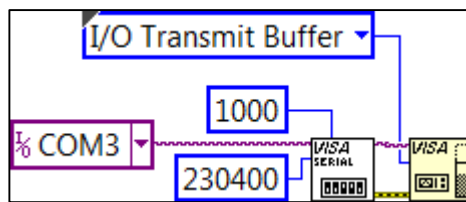


Figura E.2 Configuración Serial

E.2.2 Conversión de Valores

La conversión de los valores es una etapa vital, ya que el protocolo de comunicación maneja datos en ASCII, por lo que si se requiere el manejo de las variables, una conversión es requerida. Después de los cálculos y despliegue de información en LabVIEW, los resultados son transformados ahora a cadenas nuevamente, para ser enviados al sistema embebido en lazo cerrado. En la figura F.3 se muestran los bloques de conversión donde se indica la longitud de la precisión requerida y se muestran con líneas rosa las cadenas de caracteres y con azul y naranja, valores numéricos.

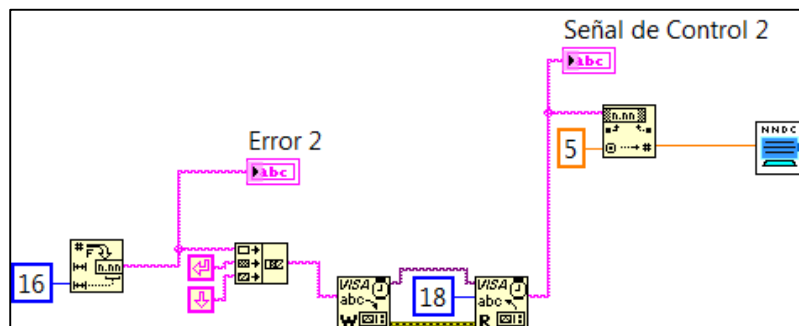


Figura E.3 Conversión de Valores

E.2.3 Lectura y Escritura de las Señales

Utilizando el puerto serial de comunicación, las señales de salida del modelo Neuronal llegan al sistema embebido, y así, las señales de control producidas por el microcontrolador son devueltas por el mismo puerto al simulador, y así en lazo cerrado, la figura muestra los bloques necesarios para el intercambio de datos, donde se especifica el número de bytes que se leerán, en este caso 18, y se especifica el número de bytes a enviar, en este caso 16.

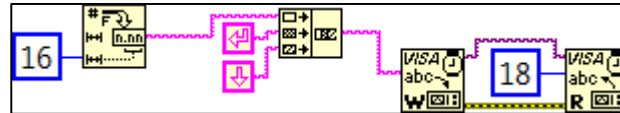


Figura E.4 Lectura (R) y Escritura (W)

E.2.4 Modelo Neuronal

Una RN entrenada con la dinámica de los casos de estudio fue implementada en un subbloque denominado subvi. La programación de este modelo se muestra en el apéndice G. La figura E.5 muestra un bloque con un diseño elaborado por el autor.



Figura E.5 Modelo Neuronal en LabVIEW

E.2.5 Visualización de las variables, simulación de disturbios y cambios en el Punto de Ajuste

Las herramientas de LabVIEW permiten la elaboración de detalladas interfaces de usuario, capaces de simular mediante controles virtuales, paneles de control industrial o de laboratorio. Se utilizaron los bloques de registrador, para visualizar la velocidad y la frecuencia, una perilla para simular disturbios, y un *slider* para cambiar el punto de ajuste. La figura E.6 muestra los bloques de programación utilizados.

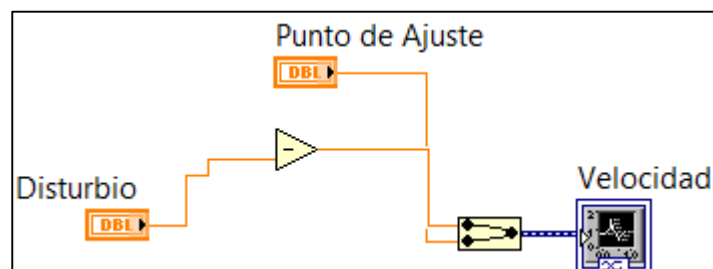


Figura E.6 Instrumentos Virtuales

E.2.6 Criterios de Simulación

Para realizar los estudios que se visualizan en la tesis, los criterios de simulación se programaron para facilitar la tarea. Se puede elegir entre simulación continua o simulación en un tiempo específico en milisegundos, con un mínimo de un milisegundo. Esto se logró deteniendo el programa con una variable lógica manipulada por un botón en el panel del usuario, o una señal lógica si se cumple el tiempo especificado por el usuario. La figura E.7 muestra la programación.

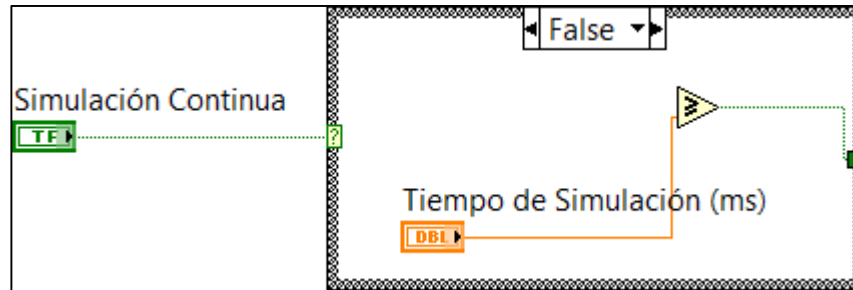


Figura E.7 Criterios de Simulación

E.2.7 Panel Frontal: Interfaz de Usuario

La interfaz de usuario permite iniciar o detener la simulación en cualquier momento, o definir un tiempo para que se detenga automáticamente, permite utilizar los controles para simular disturbios y cambiar el valor deseado. Además permite exportar los datos obtenidos a otros programas como son MATLAB y EXCEL, para realizar graficas o analizar los resultados. La figura E.8 muestra el panel frontal.

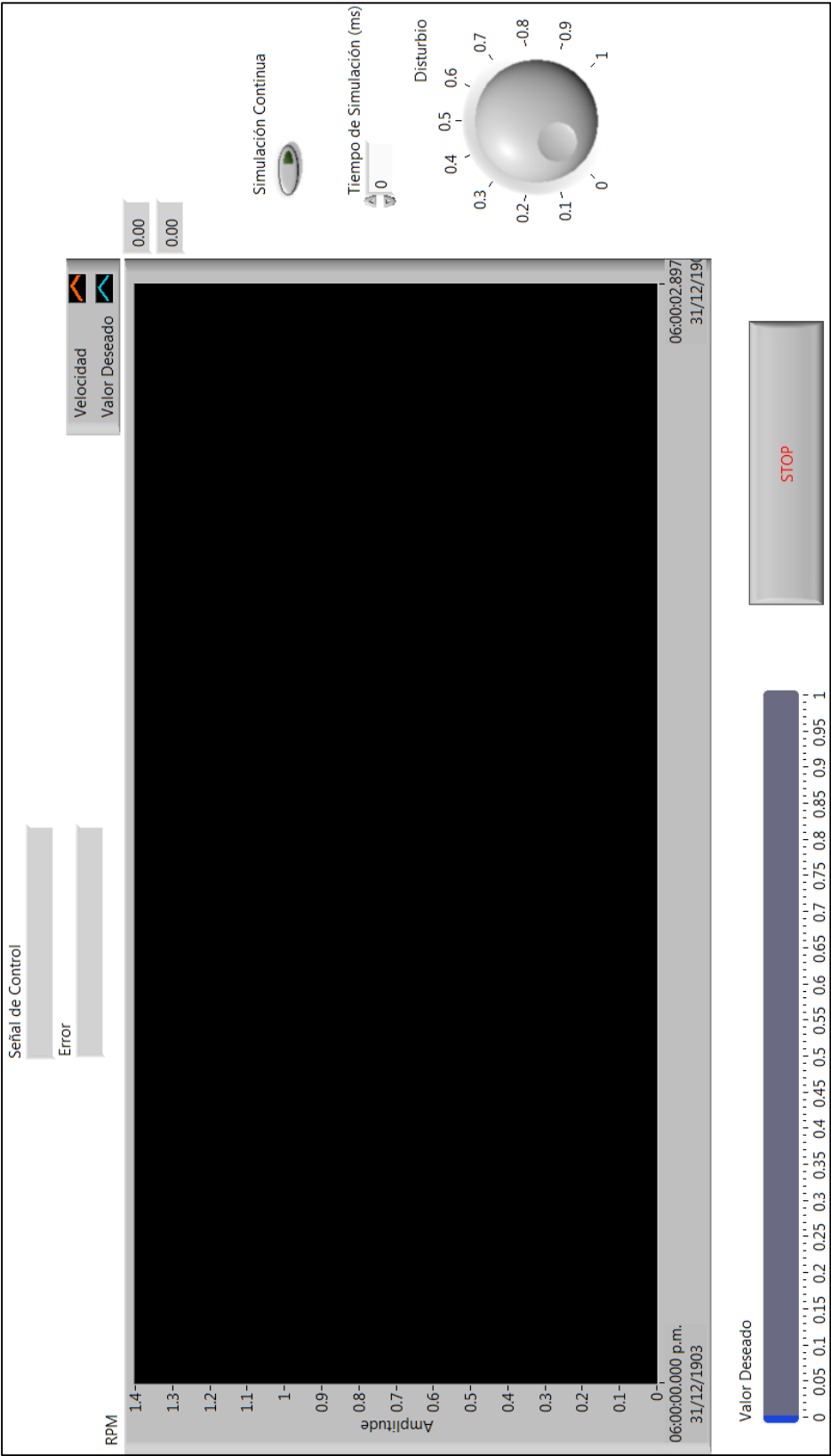


Figura E.8 Panel Frontal

