



Universidad Nacional Mayor de San Marcos

Universidad del Perú. Decana de América

Dirección General de Estudios de Posgrado
Facultad de Ingeniería Electrónica y Eléctrica
Unidad de Posgrado

**Diseño e implementación de un controlador de lógica
difusa bajo el enfoque SOC orientado a FPGA y
procesador ARM**

TESIS

Para optar el Grado Académico de Magíster en Microelectrónica

AUTOR

José Alfredo ESTRADA MONTES

ASESOR

Dr. Rubén Virgilio ALARCÓN MATUTTI

Lima, Perú

2024



Reconocimiento - No Comercial - Compartir Igual - Sin restricciones adicionales

<https://creativecommons.org/licenses/by-nc-sa/4.0/>

Usted puede distribuir, remezclar, retocar, y crear a partir del documento original de modo no comercial, siempre y cuando se dé crédito al autor del documento y se licencien las nuevas creaciones bajo las mismas condiciones. No se permite aplicar términos legales o medidas tecnológicas que restrinjan legalmente a otros a hacer cualquier cosa que permita esta licencia.

Referencia bibliográfica

Estrada, J. (2024). *Diseño e implementación de un controlador de lógica difusa bajo el enfoque SOC orientado a FPGA y procesador ARM*. [Tesis de maestría, Universidad Nacional Mayor de San Marcos, Facultad de Ingeniería Electrónica y Eléctrica, Unidad de Posgrado]. Repositorio institucional Cybertesis UNMSM.

Metadatos complementarios

Datos de autor	
Nombres y apellidos	José Alfredo Estrada Montes
Tipo de documento de identidad	DNI
Número de documento de identidad	07858584
URL de ORCID	https://orcid.org/0000-0001-6031-0105
Datos de asesor	
Nombres y apellidos	Rubén Virgilio Alarcón Matutti
Tipo de documento de identidad	DNI
Número de documento de identidad	08696547
URL de ORCID	https://orcid.org/0000-0003-4743-5135
Datos del jurado	
Presidente del jurado	
Nombres y apellidos	Darío Utrilla Salazar
Tipo de documento	DNI
Número de documento de identidad	10625916
Miembro del jurado 1	
Nombres y apellidos	Nicanor Raúl Benites Saravia
Tipo de documento	DNI
Número de documento de identidad	10189914
Miembro del jurado 2	
Nombres y apellidos	Juan Francisco Tisza Contreras
Tipo de documento	DNI
Número de documento de identidad	08596442
Datos de investigación	

Línea de investigación	C.0.5.8. Inteligencia Artificial
Grupo de investigación	Tecnologías, Inteligencia Artificial, Lógica Difusa - EEE
Agencia de financiamiento	Sin financiamiento.
Ubicación geográfica de la investigación	País: Perú Departamento: Lima Provincia: Lima Distrito: San Miguel Calle: Paititi 134 Latitud: -12.079584924577649 Longitud: -77.10230034388499
Año o rango de años en que se realizó la investigación	Enero 2019 – diciembre 2023
URL de disciplinas OCDE	Robótica, Control automático https://purl.org/pe-repo/ocde/ford#2.02.02 Hardware, Arquitectura de Computadoras https://purl.org/pe-repo/ocde/ford#2.02.06 Ingeniería eléctrica, Ingeniería electrónica https://purl.org/pe-repo/ocde/ford#2.02.01 Sistemas de automatización, Sistemas de control https://purl.org/pe-repo/ocde/ford#2.02.03



Acta de Sustentación de Tesis para Optar el Grado Académico de Magíster en Microelectrónica

Siendo las **16:13** horas del día 26 de setiembre de dos mil veinticuatro, en el salón de grados de la Facultad de Ingeniería Electrónica y Eléctrica, el Jurado de Tesis conformado por los siguientes docentes:

Presidente : Dr. Darío Utrilla Salazar
Miembro : Dr. Nicanor Raúl Benites Saravia
Miembro : Mg. Juan Francisco Tisza Contreras
Asesor : Dr. Rubén Virgilio Alarcón Matutti

Se reunieron para la sustentación de la tesis titulada **“Diseño e implementación de un controlador de lógica difusa bajo el enfoque SOC orientado a FPGA y procesador ARM”**, presentada por el bachiller, José Alfredo Estrada Montes con código N° 0949433 egresado del programa de posgrado Maestría en Microelectrónica.

Concluida la sustentación, los miembros del Jurado de Tesis procedieron a formular sus preguntas las que fueron absueltas por el graduando; acto seguido se procedió con la evaluación correspondiente. En forma colegiada, el jurado asignó el calificativo:

Aprobado por logro (16) dieciséis

Aprobada la sustentación de la Tesis, el jurado evaluador recomienda al Consejo de Facultad que se apruebe el otorgamiento del Grado Académico de Magíster en Microelectrónica a don José Alfredo Estrada Montes.

Siendo las **17:30** horas, se levantó la sesión.

Se deja constancia del acto mediante las firmas del jurado de sustentación y asesor de la tesis en la presente acta:



Dr. Darío Utrilla Salazar
Presidente



Dr. Nicanor Raúl Benites Saravia
Miembro



Mg. Juan Francisco Tisza Contreras
Miembro



Dr. Rubén Virgilio Alarcón Matutti
Asesor

CERTIFICADO DE SIMILITUD

Yo, Rubén Virgilio Alarcón Matutti en mi condición de asesor acreditado con la Resolución Decanal N° 001031-2023-D-FIEE/UNMSM de la tesis cuyo título es “Diseño e implementación de un controlador de lógica difusa bajo el enfoque SOC orientado a FPGA y procesador ARM”, presentado por el bachiller José Alfredo Estrada Montes para optar el grado de Magister en Microelectrónica, CERTIFICO que se ha cumplido con lo establecido en la Directiva de Originalidad y de Similitud de Trabajos Académicos, de Investigación y Producción Intelectual. Según la revisión, análisis y evaluación mediante la plataforma PIDS, el documento evaluado cuenta con el 11% de similitud, nivel **PERMITIDO** para continuar con los trámites correspondientes y para su **publicación en el repositorio institucional**.

Se emite el presente certificado en cumplimiento de lo establecido en las normas vigentes, como uno de los requisitos para la obtención del grado correspondiente.

Lima, 22 de agosto de 2024

Firma del Asesor:



Nombres y Apellidos del asesor:

Dr. Rubén Virgilio Alarcón Matutti

DNI: 08696547

DEDICATORIA

Dedico este trabajo a mi mama Ana y a mi papa José, que me criaron dándome todo su amor y enseñándome a levantarme ante la adversidad.

A mis hermanos Nelida, Javier, Nilton, David, Rolando, Diana, Edward y Jenny.

A Elsa y mi hijo José

A todos ellos por soportarme y darme su apoyo sin condiciones.

AGRADECIMIENTOS

A mi alma mater la Universidad Nacional Mayor de San Marcos
Al Dr. Rubén Alarcón Matutti por su ayuda en el presente trabajo de tesis.

GLOSARIO

A

ARM:

Procesador con diseño en RISC con reducción de transistores, temperatura y energía,

D

DISPOSITIVO CON ARQUITECTURA AP SoC (All Programmable System on Chip):

Es un dispositivo que Integra una FPGA XILINX de la serie 7 con un procesador ARM Cortex-A9 de doble núcleo.

F

FPGA:

Dispositivo programable que contiene bloques de lógica cuya interconexión y funcionalidad puede ser configurada in situ mediante un lenguaje de descripción especializado.

FUZZY LOGIC (Lógica Difusa):

Lógica basada en la incertidumbre, que modela como el ser humano toma sus decisiones.

I

IP (INTELLECTUAL PROPERTY):

Es un bloque funcional de lógica o datos que se utiliza para crear una FPGA de aplicación específica. Siendo una unidad reutilizable de lógica o diseño de circuito integrado (CI) Se configura y se añade a su diseño sin tener que mirar nunca el código de HDL lo que también acelera su tiempo de desarrollo.

IoT (Internet of Things);

Es una red de dispositivos que se conectan a la nube y que se comunican entre ellos; permitiendo la comunicación en cualquier parte del mundo.

L**Lenguaje VHDL (Very High Speed Integrated Circuits Hardware Description Language):**

Es un Lenguaje de descripción de hardware (circuitos electrónicos digitales)

LUTs (Look-Up Tables):

Es una tabla de verdad, donde diferentes combinaciones de entradas, generan determinadas salidas dependiendo de la lógica configurada (funciones) permitiendo rapidez de los procesos y ahorrando realizar cálculos.

R**RISC:**

Conjunto de instrucciones reducido, ejecución de la instrucción en un ciclo de máquina, tamaño único en bits de la instrucción

V**VIVADO DESIGN SUITE HLx EDITIONS:**

Plataforma de Software usado en aplicaciones de dispositivos FPGA, SoC, MPSoC, RFSoc o ACAP permitiendo que el desarrollador de hardware aproveche el diseño basado en C y la reutilización optimizada así como también proporciona la reutilización de subsistemas IP

VITIS UNIFIED SOFTWARE PLATFORM

Plataforma de Software usado en aplicaciones de dispositivos FPGA, SoC, MPSoC, RFSoc o ACAP proporcionando la capacidad de abstracción al implementar los problemas de aplicación usando lenguajes de alto nivel como el C

TABLA DE CONTENIDO

DEDICATORIA.....	iii
AGRADECIMIENTOS	iv
GLOSARIO	v
TABLA DE CONTENIDO	vii
RESUMEN	xii
ABSTRACT	xiii
Capítulo 1 INTRODUCCION	1
1.1 Situación Problemática	1
1.2 Formulación del Problema	3
1.3 Justificación de la Investigación.....	4
1.4 Objetivos de la Investigación	5
1.4.1 <i>Objetivo General</i>	5
1.4.2 <i>Objetivos específicos</i>	5
Capítulo 2 MARCO TEORICO	6
2.1 Antecedentes de investigación	6
2.2 Base Teórica.....	9
2.2.1 <i>La Lógica Difusa</i>	9
2.2.2 <i>Sistema de Control Difuso</i>	12
2.2.3 <i>Motor BLDC (Brushless Direct Current)</i>	13
2.2.4 <i>Plataforma Zybo Z7-10</i>	19
2.2.5 <i>Entorno desarrollo de software VIVADO 2020.1 y VITIS 2020.1</i>	21
2.2.6 <i>El módulo Pmod</i>	21
Capítulo 3 HIPOTESIS Y VARIABLES	25
3.1 Hipótesis General	25
3.2 Hipótesis específica	25
3.3 Identificación de las Variables	26
Capítulo 4 METODOLOGIA	27
4.1 Sistema de Control de Velocidad de un motor BLDC	28

4.2 Alternativas de Diseño de Controlador Difuso	33
4.3 Diseño del Controlador Difuso	34
4.3.1 <i>Implementación del Sistema de Control Difuso</i>	36
4.3.2 <i>Consideraciones de la programación del Algoritmo difuso</i>	46
4.4 Implementación del Controlador Difuso	46
Capítulo 5 RESULTADOS Y DISCUSION	55
5.1 Análisis, Interpretación y Discusión de Resultados.....	55
5.2 Pruebas de Hipótesis.....	60
5.3 Presentación de Resultados	62
Capítulo 6 IMPACTOS	66
6.1 Propuesta de la Solución del Problema	66
6.2 Costos de la Implementación de la Propuesta.....	67
6.3 Beneficios que aporta la Propuesta.	68
CONCLUSIONES	69
RECOMENDACIONES	70
REFERENCIAS.....	72
ANEXO A. DIAGRAMA DE ICHICAWA	75
ANEXO B. PROGRAMA DEL CONTROLADOR DIFUSO (VITIS).....	77
ANEXO C. MANUAL DEL MOTOR BLDC HST23	88

LISTA DE TABLAS

Tabla 1.1 Metodología	4
Tabla 2.1 Sensor en el sentido de las agujas del reloj y bits de accionamiento por orden de fase	17
<i>Tabla 2.2</i> Sensor en el Sentido de las agujas del reloj y bits de accionamiento por orden de fase	18
Tabla 2.3 Sensor Sentido contra las agujas del reloj y Bits de accionamiento .	19
Tabla 2.4 Descripción de las señales del conector del PMod AD2	22
Tabla 2.5 Descripción de las señales del conector del PMod DA2	23
Tabla 2.6 Descripción de las señales del conector del PMod OLED 128x32....	24
Tabla 4.1 Comparación de valores medidos del Controlador Difuso y el control Convencional PI a 3,000 RPM	49
Tabla 4.2 Comparación de valores medidos del Controlador Difuso y el Control Convencional PI a 2,500 RPM	50
Tabla 4.3 Comparación de valores medidos del Controlador Difuso y el Control Convencional PI a 1,000 RPM	51
Tabla 4.4 Comparación de error estacionario del Controlador Difuso y el Control Convencional PI a 3,000 RPM	52
Tabla 4.5 Comparación de error estacionario del Controlador Difuso y el Control Convencional PI a 2,500 RPM	53
Tabla 4.6 Comparación de error estacionario del Controlador Difuso y el Control Convencional PI a 1,000 RPM	53
Tabla 5.1 Comportamiento del Sistema de Control de velocidad de un motor BLDC con respecto a un Controlador Difuso y a un Controlador PI.	55
Tabla 5.2 Comparación de valores medidos del Controlador Difuso y la Simulación en Simulink – MatLab	59
Tabla 5.3 Resultado de prueba de hipótesis	62
<i>Tabla 6.1</i> Costo de la implementación	67
Tabla 6.2 Beneficio de la Propuesta	68

LISTA DE FIGURAS

<i>Figura 2.1</i> Metodología de Mamdani.....	11
<i>Figura 2.2</i> Sistema de control difuso.....	12
<i>Figura 2.3</i> Controlador Difuso	13
<i>Figura 2.4</i> Diagrama de un motor BLDC.....	15
<i>Figura 2.5</i> Diagrama de tiempos de las salidas del sensor Vs Los voltajes de accionamiento requerido por el motor BLDC	16
<i>Figura 2.6</i> Circuito puente trifásico con transistores MOSFET de Potencia.....	17
<i>Figura 2.7</i> Plataforma ZYBO Z7 - 10.....	19
<i>Figura 2.8</i> Arquitectura Zynq – Sistema software y Sistema hardware	20
<i>Figura 2.9</i> Puerto Pmod de la plataforma ZYBO Z7.....	21
<i>Figura 2.10</i> Trama del dato del PMod DA2.....	23
<i>Figura 2.11</i> Diagrama del dato del PMod DA2.....	23
<i>Figura 4.1</i> Diagrama de Bloques de un Sistema de Control de Velocidad de un motor BLDC	27
<i>Figura 4.2</i> Sistema de Control Difuso de Velocidad de un motor BLDC	29
<i>Figura 4.3</i> El Conjunto Difuso en Fuzzy Logic Toolbox.....	30
<i>Figura 4.4</i> El Conjunto Difuso de Entrada en Fuzzy Logic Toolbox	31
31	
<i>Figura 4.5</i> El Conjunto Difuso de Salida en Fuzzy Logic Toolbox.....	31
<i>Figura 4.6</i> Las Reglas de Control en Fuzzy Logic Toolbox.....	32
<i>Figura 4.7</i> Los Diagramas de las Reglas de Control en Fuzzy Logic Toolbox ..	32
<i>Figura 4.8</i> La Respuesta Transitoria del Controlador Difuso y el Control PI en	33
<i>Figura 4.9</i> Herramientas de diseño de Xilinx VIVADO y VITIS.	35
<i>Figura 4.10</i> Diagrama de bloques del Controlador Difuso	36
<i>Figura 4.11</i> Zybo Z7-10.....	36
<i>Figura 4.12</i> Pmod AD2.....	37
<i>Figura 4.13</i> Pmod DA2.....	37

<i>Figura 4.14</i> Pmod OLED	38
<i>Figura 4.15</i> IP Integrator del Controlador Fuzzy	39
<i>Figura 4.16</i> Configuración del IP para el SPI del Pmod AD2	40
<i>Figura 4.17</i> Repositorio IP	40
<i>Figura 4.18</i> Plataforma Zybo Z7.....	41
<i>Figura 4.19</i> Pin Out de los puertos del Zybo Z7	41
<i>Figura 4.20</i> Asignación del puerto JC para el pmod AD2	42
<i>Figura 4.21</i> Restricciones (constraints)para el Pmod AD2.....	42
<i>Figura 4.22</i> Asignación del puerto JD para Pmod DA2.....	43
<i>Figura 4.23</i> Restricciones (constraints)para el Pmod DA2.....	43
<i>Figura 4.24</i> Asignación para el pmod OLED	44
<i>Figura 4.25</i> Restricciones (constraints)para el Pmod OLED	44
<i>Figura 4.26</i> Dirección de Registros IP.....	45
<i>Figura 4.27</i> Programación en el Software VITIS	45
<i>Figura 4.28</i> Esquema de la Simulación de un Sistema de Control de un motor BLDC con Controlador Fuzzy y Controlador PI	48
<i>Figura 4.29</i> Medición de la salida del Sistema de Control de motor BLDC con Controlador PI a 3,000 RPM	49
<i>Figura 4.30</i> Medición de la salida del Sistema de Control de motor BLDC con Controlador PI a 2,500 RPM	50
<i>Figura 4.31</i> Medición de la salida del Sistema de Control de motor BLDC con Controlador PI a 1,000 RPM	51
<i>Figura 4.32</i> Medición de la salida de error estacionario del Sistema de Control de motor BLDC con Controlador PI a 3,000 RPM	52
<i>Figura 4.33</i> Medición de la salida de error estacionario del Sistema de Control de motor BLDC con Controlador PI a 2,500 RPM	53
<i>Figura 4.34</i> Medición de la salida de error estacionario del Sistema de Control de motor BLDC con Controlador PI a 1,000 RPM	54
<i>Figura 5.1</i> Curva de Interpolación.	57
<i>Figura 5.2</i> Laboratorio de pruebas del Controlador Difuso en la plataforma ZYBO Z7-10	65
<i>Figura 6.1</i> Sistema de Control Difuso	66
<i>Figura A.1</i> Diagrama de Ichicawa del Controlador Difuso.....	76

RESUMEN

El presente trabajo de Tesis muestra el enfoque SoC (FPGA + ARM PROCESSOR) en la implementación de un controlador difuso para en un Sistema de control de velocidad de un motor de corriente continua BLDC.

Se usa un dispositivo Zynq AP – SoC dividido en dos (02) bloques como son la lógica programable(PL) en el FPGA y PS en el ARM dual core Cortex-A9. En el bloque PL se hace el Hardware en VHDL y en bloques IP, usados para para la implementación de las interfaces periféricas. En el otro bloque PS se ha realizado la codificación del algoritmo de la lógica difusa en lenguaje C, ambos bloques son unidos a nivel de software, y compilado en un solo archivo binario a ser grabarlo en el dispositivo SoC, que se encuentra dentro de la plataforma Zybo Z7 implementándose el controlador difuso.

Se ha desarrollado el modelo de la lógica difusa (metodología de Mamdani), con el Sistema de control de velocidad de un motor BLDC, usando el software MATLAB con su herramienta de lógica difusa (Tool Box Fuzzy Logic) y la simulación se ha realizado en SIMULINK. Además en dicho software compara la performance del Controlador difuso con un Controlador convencional más usado como el PI.

Palabras Clave: Dispositivo SoC, Lógica Difusa, Plataforma Zybo Z7, Motor BLDC, MATLAB-TOOL BOX FUZZY LOGIC-SIMULINK

ABSTRACT

This Thesis work shows the SoC (FPGA + ARM PROCESSOR) approach in the implementation of a Fuzzy Controller for a Speed Control System of a BLDC direct current motor. A Zynq AP – SoC device is used divided into two (02) blocks such as the programmable logic (PL) in the FPGA and PS in the ARM dual core Cortex-A9. In the PL block, the Hardware is made in VHDL and in IP blocks, used for the implementation of the peripheral interfaces. In the other PS block, the coding of the Fuzzy Logic algorithm has been code in C language, both blocks are united at the software level, and compiled into a binary file to be programed in the SoC device, in the Zybo Z7 platform implementing the Fuzzy Controller.

A SoC device is used divided into two (02) blocks such as the PL in the FPGA and PS in the ARM dual core Cortex-A9. In the PL block, the Hardware is made in VHDL and in IP blocks, used for the implementation of the peripheral interfaces. In the other PS block, the Fuzzy Logic model has been developed. The Fuzzy Logic model has been developed (Mamdani methodology), with the Speed Control System of a BLDC motor, using the MATLAB software with its tool. of Fuzzy Logic (Tool Box Fuzzy Logic) and the simulation used SIMULINK. This software compares the performance of the Fuzzy Controller with a more commonly used Conventional Controller such as the PI.

Keywords: SoC Device, Fuzzy Logic, Zybo Z7 Platform, BLDC motor,
MATLAB-TOOL BOX FUZZY LOGIC-SIMULINK

Capítulo 1 INTRODUCCION

1.1 Situación Problemática

Hoy en día, ante la creciente modernidad aplicada en las diferentes industrias aumentan de tamaño y complejidad en sus procesos de sus diferentes etapas siendo necesario un sistema de control optimo y robusto que permita lograr un sistema estable en el manejo de sus diferentes variables que intervienen en el resultado esperado.

En la industria dedicadas a producción de equipos o a extracción de minerales, existen problemas de control en algunas etapas en su proceso que son manejados en forma tradicional usando controladores convencional donde uno de los más usados se basa en el algoritmo del controlador Proporcional, y Integral (PI), aunque el sintonizado o afinación de sus dos constantes siempre demora dependiendo de la experiencia del especialista que realiza dicha sintonía. No obstante, que dicho controlador convencional tiene problemas de bajo desempeño debido al ajuste de sus constantes para un estado de funcionamiento en condiciones no previstas que se presentan provocando un funcionamiento inadecuado u oscilaciones alrededor del Señal de entrada de Referencia o posición deseada (Set Point).

Por ello, el avance de la tecnología tendencia a la Inteligencia Artificial ha llevado a desarrollar diversas técnicas para el desarrollo de los controladores en los sistemas de control donde la complejidad del proceso y el pobre desempeño del controlador es muy alta. Una de las técnicas actuales, confiables y estables son los controladores desarrollados en el ámbito de la Inteligencia Artificial basados en la teoría de la Incertidumbre o llamada también lógica difusa (Zadeh, 1973).

La tecnología basada en lógica difusa tiene actualmente una gran variedad de aplicaciones debido a sus grandes características que son las siguientes:

- Control robusto del proceso de diseño en sistemas multivariantes con modelos sin precisión y con incertidumbre en la medición.
- Control eficiente aplicado en sistemas de control no lineal donde existen demasiadas condiciones restrictivas.
- La alta tolerancia a fallos funcionando en ambientes con perturbaciones de diferentes naturalezas
- Fácil manejo, rapidez de mantenimiento y actualizaciones debido a que el controlador se basa en variables lingüísticas cualitativas.

Las aplicaciones de un control difuso van desde el control de complejos procesos industriales hasta el diseño de dispositivos artificiales de decisión, así como en la aplicación en desarrollo de artefactos electrónicos de uso doméstico y del entretenimiento y en los sistemas de diagnóstico.

La presente Tesis presenta el desarrollo e implementación de un Controlador difuso aplicado en los Sistemas de control de velocidad motores de alta eficiencia de alto torque, gran velocidad y pequeñas dimensiones como lo es el motor de corriente eléctrica directa Brushless (DC), llamado motor BLDC (Brushless Direct Current)

1.2 Formulación del Problema

Las consideraciones expuestas nos llevan a plantear la siguiente interrogante:

¿Cómo el Diseño e Implementación de un controlador de Lógica Difusa con tecnología FPGA XILINX 7 (Zynq-7000 - AP SoC) y ARM CORTEX A9, brindara un mejor comportamiento así como flexibilidad que un controlador convencional?

Asimismo, cabe indicar, que en nuestra investigación se desarrolla todo lo que concierne a el uso de la Microelectrónica en los problemas específicos a tratar como son:

- *¿Cómo diseño de un Controlador de Lógica Difusa con Tecnología FPGA XILINX 7 Y ARM CORTEX A9 a usarse en un sistema de control tiene un mejor desempeño que un Controlador tradicional?*
- *¿De qué manera el desarrollo de un software orientado a la Microelectrónica, contribuirá en un diseño robusto y versátil a un diseño de un Controlador por Lógica difusa?*
- *¿De qué manera la el diseño y la implementación de un Controlador por lógica difusa con tecnología FPGA XILINX 7 Y ARM CORTEX A9 aplicado a un Sistema de control de velocidad de un motor de corriente continua Brushless (BLDC), tiene una mejor flexibilidad que un Controlador convencional?*

1.3 Justificación de la Investigación

- El uso de la lógica difusa en el diseño de un controlador con el fin de usarlo en el Control de motor de corriente continua Brushless lograra soluciones existentes simplificando el procedimiento de diseño.
- El diseño de interacción hardware/software de un Sistema electrónico por Lógica difusa en dispositivos SoC servirá a la comunidad de electrónica (profesionales y estudiantes) tengan una alternativa de diseño con flexibilidad (firmware actualizable), velocidad de procesamiento (ejecución de la instrucción) y potencia de cálculo, también aplicables en visión artificial.
- **Metodología**
En este trabajo se ha de aplicar dos técnicas para obtener los datos tal como se indica en la Tabla 1.1

Tabla 1.1 Metodología

TÉCNICAS PARA LA OBTENCIÓN DE DATOS	INSTRUMENTOS DE OBTENCIÓN DE DATOS
Medición de los valores del Controlador difuso	– Software especializado – Tarjeta electrónica Zybo Z7-10
Medición de valores de los Sistemas de Control difuso y Proporcional e Integrativo	– Software especializado

Fuente. Elaboración propia

1.4 Objetivos de la Investigación

1.4.1 Objetivo General

Desarrollar e implementar un controlador electrónico por LÓGICA DIFUSA y con TECNOLOGÍA FPGA XILINX 7 y ARM CORTEX A9 para un Sistema de control de velocidad de un motor de corriente continua BLDC de uso minero.

1.4.2 Objetivos específicos

- Desarrollar un Controlador por lógica difusa con tecnología FPGA XILINX 7 y ARM CORTEX A9 para un Sistema de control de un motor de corriente continua Brushless.
- Desarrollar el software en los lenguajes de programación VHDL (IP) y C a nivel de abstracción para nuestro controlador.
- Analizar y comparar bajo un modelamiento matemático el desempeño del Controlador por Lógica Difusa con tecnología FPGA XILINX 7 y ARM CORTEX A9 versus un Controlador convencional.

Capítulo 2 MARCO TEORICO

2.1 Antecedentes de investigación

Con respecto a los Sistemas de control por lógica difusa se tiene el trabajo de investigación de un “FPGA BASED CO-DESIGN OF A SPEED FUZZY LOGIC CONTROLLER APPLIED TO AN AUTONOMOUS CAR”, (Aridhi, Popescu, & Mami, 2021) en la revista International Journal of Reconfigurable and Embedded Systems (IJRES) 10(3), que indica el uso de la tecnología FPGA para controlar la velocidad de un automóvil utilizando lógica difusa. Fue desarrollado utilizando el lenguaje de hardware VHDL y manejado por el procesador Zynq a través de un diseño de software SDK escrito en C. el controlador difuso actúa de acuerdo a la temperatura ambiente y a la detección de un obstáculo y su distancia al auto. La partición de las tareas de codiseño las divide en partes de hardware y software. Los resultados de la simulación del sistema difuso y los de la implementación completa del codiseño en una placa Xilinx Zynq que muestra la eficacia del controlador difuso para cumplir con las restricciones objetivo y generar señales PWM adecuadas. Asimismo, se puede actualizar el firmware fácilmente.

Este trabajo de investigación mencionado. permite el uso de la tecnología SoC en su diseño, que hace sea una buena alternativa en esta aplicación. El utilización de la lógica difusa es adecuada para este diseño gracias al uso de bloques de procesamiento de señales digitales que aseguran el procesamiento paralelo.

Tesis de maestría “DISEÑO E IMPLEMENTACIÓN EN FPGA DE UN SISTEMA DE CONTROL PARA BIPEDESTADOR MÓVIL” (Gómez, 2015), que muestra el diseño y construcción de un exoesqueleto cuya función es permitir la postura de pie a usuarios con debilidad muscular general.

La Tesis de maestría mencionada permite el uso de la tecnología FPGA (Arreglo de Compuertas Lógicas Programable en (Lomas , 2016) Campo). para el sistema de control de un dispositivo tipo Péndulo Invertido sobre ruedas (WIP - Wheeled Inverted Pendulum) en la plataforma de locomoción para el exoesqueleto., presentando su implementación

La tesis de doctorado “APRENDIZAJE Y RECONOCIMIENTO INVARIANTE DE OBJETOS EN ENSAMBLE CON ROBOTS EMPLEANDO REDES NEURONALES ARTIFICIALES IMPLEMENTADAS CON FPGA” (Lomas , 2016), presenta un sistema embebido (SE) de imagen utilizando un algoritmo para el reconocimiento invariable de objetos en una línea de producción.

Esta Tesis de doctorado hace uso de tecnologías de hardware reconfigurable que es una técnica propia en Sistemas basados en FPGA para la obtención de la imagen, así como la abstracción del procesamiento que culminan con el reconocimiento de un objeto.

El trabajo de investigación de un “CONTROL BASADO EN LÓGICA BORROSA EN DSPIC33 PARA UN MOTOR BRUSHLESS” (Rodríguez, 2015), que indica bajo la lógica difusa se puede regular la velocidad de un motor Brushless (BLDC), usando un DCS (Digital Signal Controller) dsPIC33FJ12MC202 unido a una interface de potencia de seis transistores Mosfet (IRF840) con tres sensores de efecto Hall situados en el motor

BLDC los cuales envían la posición del rotor al DCS para determinar el par de devanados a excitar (Rodríguez, 2015). Se usa el software MATLAB (toolbox SIMULINK - FUZZY LOGIC) y para el DSC el simulador PROTEUS – ISIS

Este trabajo de investigación permite ver una aplicación de control de un motor Brushless por lógica difusa usando como eje principal un DSC, usando software como el PROTEUS-ISIS y el MATLAB (Tool Box - Fuzzy Logic)

También se tiene como referencia el libro titulado Fuzzy Control (Passino & Yurkovich, 1997).

En esta Libro presenta aplicaciones de la lógica difusa como solución en controles No lineales en una planta industrial, enfatizando la obtención del modelo difuso de dicho control y realizando la comparación con los Sistemas de control convencionales.

2.2 Base Teórica

2.2.1 La Lógica Difusa

El presente trabajo de tesis se basa en la Lógica Difusa que es parte de la Inteligencia Artificial (IA) que es una de las tendencias de tecnología de hoy y del futuro de la humanidad.

Definimos la IA como la expresión en formas de lenguaje o reglas simbólicas en busca de imitar en el comportamiento inteligente. La Lógica Difusa ha sido desarrollada buscando modelar la forma como el cerebro maneja la información imprecisa. La lógica difusa es una herramienta válida para representar dichos mecanismos de razonamiento, se cita el siguiente párrafo del creador Lofti Zadeh de la lógica difusa "The approach described in this paper represents a substantive departure from the conventional quantitative techniques of system analysis." (Zadeh, 1973, pág. 3), que significa "El enfoque descrito en este artículo representa una desviación sustancial de las técnicas cuantitativas convencionales de análisis de sistemas". Indicando la esencia de La Lógica Difusa.

En los últimos años se ha producido un incremento considerable del número de aplicaciones de control que emplean técnicas de inferencias (estrategias de control) basadas en Lógica Difusa permitiendo:

- Desarrollar Sistemas de control complejos a partir de descripciones lingüísticas del conocimiento de un operador experto, siendo una excelente alternativa sin la necesidad de emplear modelos matemáticos normalmente complejos.

- No requiere identificar el sistema, ni necesita aproximar el modelo y tampoco necesita linealizarlo
- Cuando no se conozca el modelo que se usa:

Por lo tanto, un Sistema de control basado en lógica difusa está formada por un conjunto de reglas que se aplica a las entradas de datos previamente fuzzificados (difusión), convertidos a conjuntos difusos, luego se evalúan las Reglas de Control del sistema aplicadas a las entradas fuzzificadas (difusión) cuyo resultado se envía a defuzzificar (concreción) obteniéndose la(s) salida(s), de control respectiva, se cita el párrafo "After all the inputs and outputs are defined for the fuzzy controller, we can specify the fuzzy control system."The linguistic variables and values provide a language for the expert to express her or his ideas about the control decision-making process in the context of the framework established by our choice of fuzzy controller inputs and outputs". (Passino & Yurkovich, 1997, pág. 29). Donde se indica que existen dos (02) metodologías normalmente usadas en un Sistema Difuso (Passino & Yurkovich, 1997), una es la de Mamdani y la otra es la de TSK (Takagi, Sugeno y Kang), en el presente trabajo de Tesis usaremos la de Mamdani, en el cual el proceso de obtener un valor de salida para valores de entrada puede verse en la figura 2.1.

Donde la Función de Membresía o Función de Pertenencia esta definido como:

$$\mu_{(x=A)} \rightarrow [0,1]$$

Siendo:

x : Es un elemento que pertenece totalmente o parcialmente al conjunto difuso A

$\mu_{(x=A)}$: Es una función que varía de 0 a 1

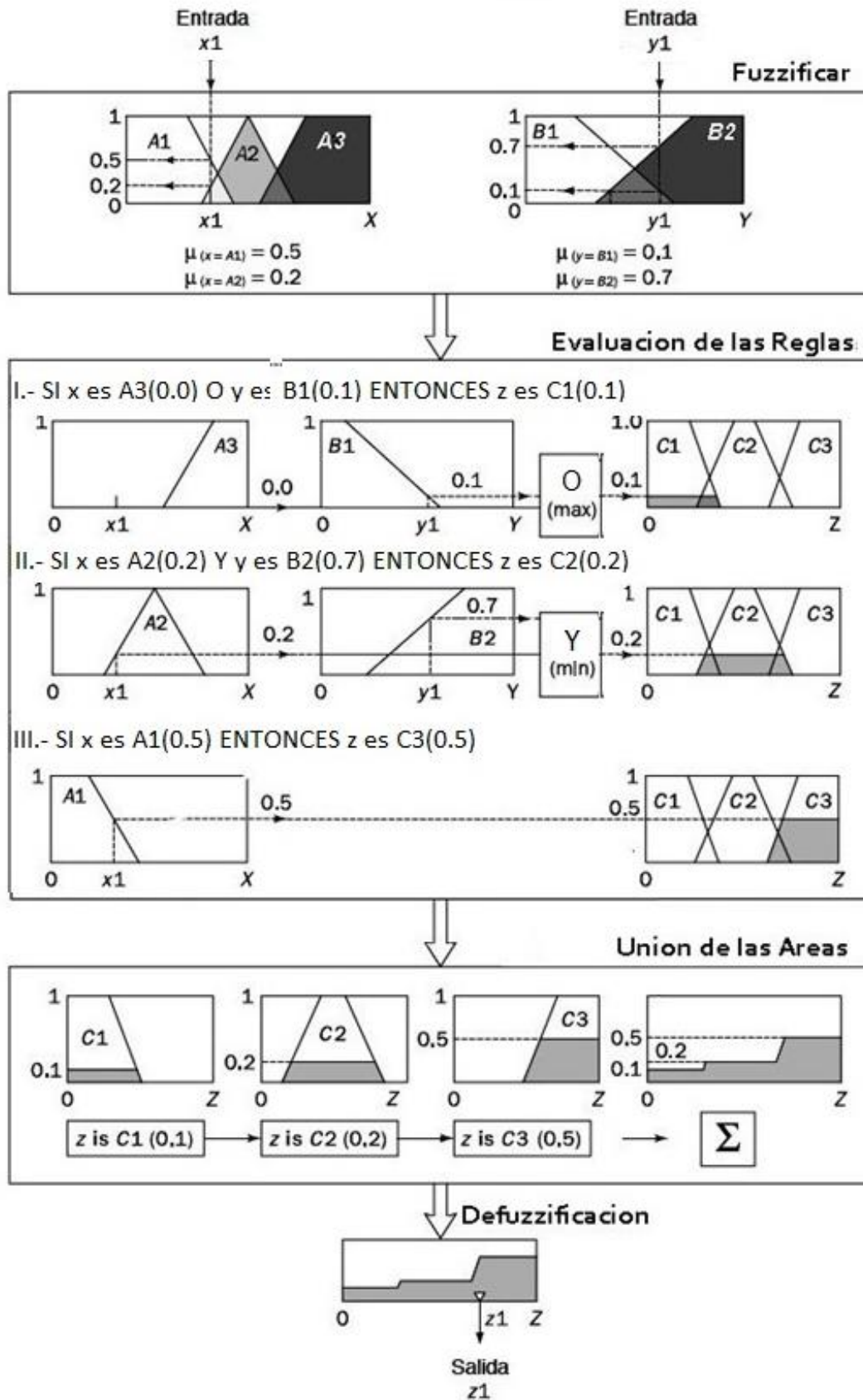


Figura 2.1 Metodología de Mamdani

Fuente. Elaboración propia

2.2.2 Sistema de Control Difuso

Un Sistema de control en el ámbito industrial es el conjunto de elementos, que hace posible la regulación de forma automática de las distintas variables que intervienen en el desarrollo de un producto o también podemos decir que puede ser parte de otro sistema, proceso o planta ajustándose a un proceso establecido. Existen variedad de Sistemas de control, en el presente trabajo de Tesis usamos la Inteligencia Artificial a través de la Lógica difusa mostrando el Diagrama de Sistema de control dpDifuso usado como puede verse en la *Figura 2.2*

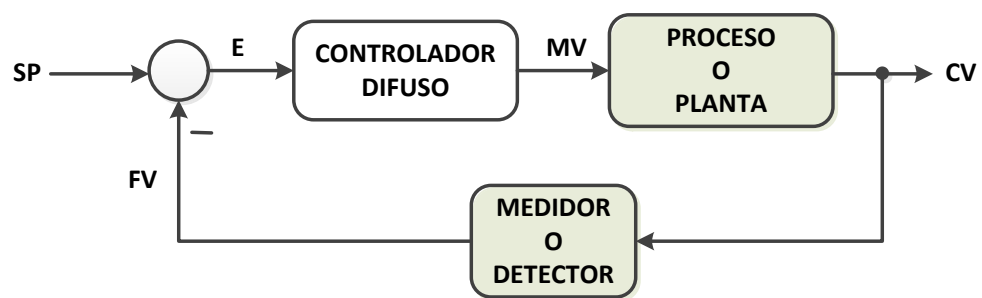


Figura 2.2 Sistema de control difuso

Fuente. Elaboración propia

Siendo:

SP: Set Point (Referencia o Valor deseado).

E: Error.

FV: Feedback Variable (Valor medido de CV).

MV: Manipulated Variable (Variable manipulada).

CV: Controlled Variable o Process Variable (Variable a controlar o Variable del proceso o Salida).

Un Controlador difuso es un dispositivo de control de realimentación que calcula la desviación o error entre un valor medido y el valor que ha de ser obtenido para implementar acciones correctivas y ajustar el proceso. El Controlador difuso como puede verse en la Figura 2.3, se ajusta usando la Fuzzificación, Las Reglas (estrategias) de control y la Defuzzificación, configurando las variables en el algoritmo de Control Difuso, con el fin de que su diseño proporcione un control requerido para realizar el proceso.



Figura 2.3 Controlador Difuso

Fuente. Elaboración propia

2.2.3 Motor BLDC (Brushless Direct Current)

Este motor no emplea escobillas presentando características muy singulares como son

- La conmutación se realiza solamente en forma electrónica en lugar de mecánica.
- Los imanes permanentes van alojados en el rotor en lugar de en el estator y las bobinas van alojadas en el estator, constituyendo un devanado polifásico.
- Los devanados están distribuidos a lo largo del estator en múltiples fases.
- Su alimentación tiene tres fases con un desfase de 120° entre ellas.

- Su funcionamiento se basa en la alimentación secuencial de cada una de las fases del estator del motor de forma sincronizada con el movimiento del rotor. De esta forma, los imanes permanentes alcanzan el movimiento del campo magnético donde el desplazamiento depende a su vez del giro del rotor.

Ante esto, en el presente trabajo de Tesis se muestra un controlador basado en lógica difusa dentro de la teoría de la Inteligencia Artificial para el control de motores Brushless (BLDC) que mantenga las expectativas de eficiencia y estabilidad en el ambiente industrial como el minero.

La Figura 2.4 muestra al motor BLDC. Un motor sin escobillas está construido con un rotor de imán permanente y polos de estator devanados con alambre.

La energía eléctrica se convierte en energía mecánica por las fuerzas de atracción magnética entre el rotor de imán permanente y un campo magnético giratorio inducido en los polos del estator bobinado.

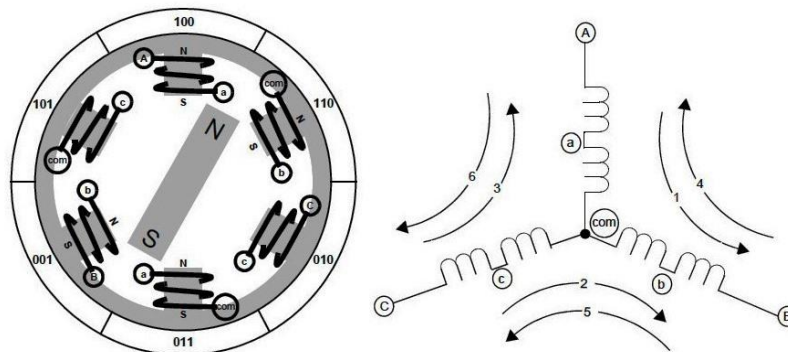


Figura 2.4 Diagrama de un motor BLDC

Fuente. AN857 Brushless DC Motor Control Made Easy, Microchip Technology Inc. (Brown, 2002)

En este sistema hay tres circuitos electromagnéticos conectados en un punto común. Cada circuito electromagnético se divide en el centro, lo que permite que el rotor de imanes permanentes se mueva en medio del campo magnético inducido. Los motores BLDC tienen una topología de devanado trifásico con conexión en estrella, esto permite que sea accionado energizando dos fases a la vez. La alineación estática que se muestra en la Figura 2.4 es la que se lograría creando un flujo de corriente eléctrica desde la terminal A a B, indicada como ruta 1 en la Figura 2.4. El rotor puede girar en el sentido de las agujas del reloj 60 grados desde la A hasta B. Alineación B cambiando la ruta actual para que fluya de la terminal C a la B, indicada como ruta 2 en el esquema.

Para la conmutación en un BLDC es detectar la posición del rotor y luego energizar las fases que producirán la mayor cantidad de torque. El rotor viaja 60 grados por paso de conmutación. La ruta de corriente del estator adecuada se activa cuando el rotor está a 120 grados de la alineación con el campo magnético del estator correspondiente. Luego se desactiva cuando el rotor está a 60 grados de la alineación, en cuyo momento se activa el siguiente circuito y se repite el proceso. La conmutación para la posición del rotor, que se muestra en la Figura 2.4, sería al final de la ruta de corriente 2 y al comienzo

de la ruta de corriente 3 para la rotación en el sentido de las agujas del reloj.. Conmutar las conexiones eléctricas a través de las seis combinaciones posibles, numeradas del 1 al 6, en los momentos precisos hará que el rotor realice una revolución eléctrica. En el motor de la Figura 2.4, una revolución eléctrica es lo mismo que una revolución mecánica. La forma de saber el momento correcto para conmutar las corrientes de los devanados es mediante un sensor de posición. Los fabricantes de motores BLDC suministran motores con un sensor de posición de efecto Hall de tres elementos. Cada elemento sensor entrega un nivel alto digital para 180 grados eléctricos de rotación eléctrica y un nivel bajo para los otros 180 grados eléctricos. Los tres sensores están desplazados entre sí por 60 grados eléctricos para que cada salida del sensor esté alineada con uno de los circuitos electromagnéticos. En la Figura 2.5 se muestra un diagrama de tiempo que muestra la relación entre las salidas del sensor y los voltajes de accionamiento (drivers) requeridos por el motor.

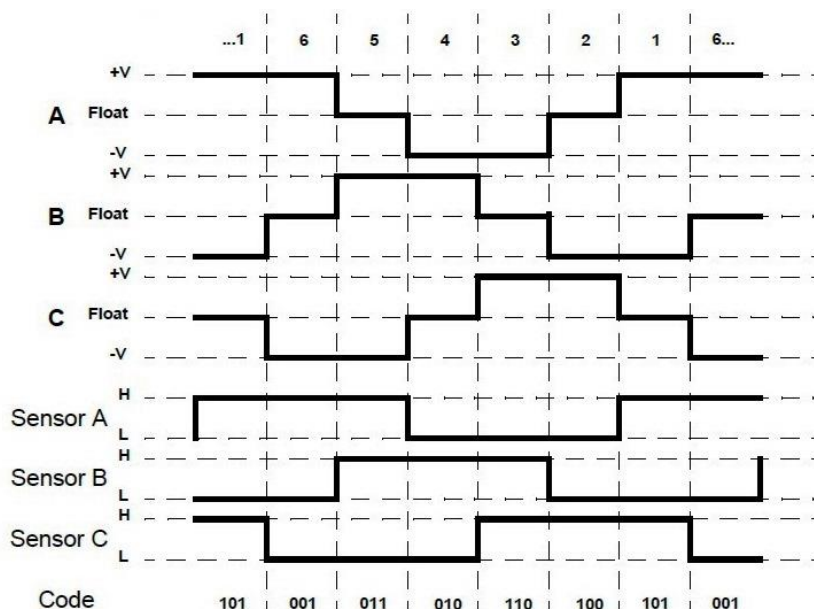


Figura 2.5 Diagrama de tiempos de las salidas del sensor Vs Los voltajes de accionamiento requerido por el motor BLDC

Fuente. AN857 Brushless DC Motor Control Made Easy, Microchip Technology Inc. (Brown, 2002) .

En la Figura 2.6 se muestra la etapa de potencia que va entre el procesador y el motor BLDC, logrando las señales de Salida Trifásica de Voltaje que requiere dicho motor de la Figura 2.5.

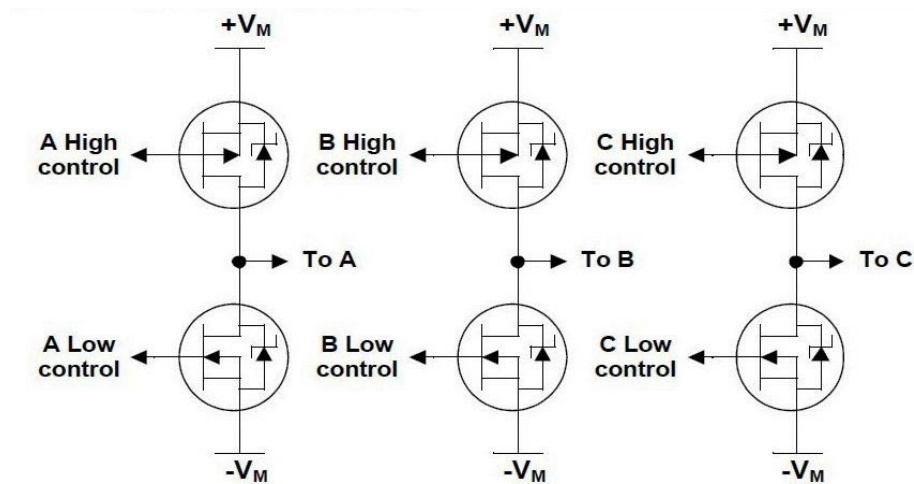


Figura 2.6 Circuito puente trifásico con transistores MOSFET de Potencia
Fuente. AN857 Brushless DC Motor Control Made Easy, Microchip Technology Inc. (Brown, 2002).

En la Tabla 2.1 se encuentran las fases o estados del diagrama de tiempos que se muestra en la Figura 2.5, incluyendo las tres (03) señales de salida de los sensores y las seis señales de entrada de los Mosfet (canales P y N) de potencia.

Tabla 2.1 Sensor en el sentido de las agujas del reloj y bits de accionamiento por orden de fase

Phase	Sensor C	Sensor B	Sensor A	C High Drive	C Low Drive	B High Drive	B Low Drive	A High Drive	A Low Drive
1	1	0	1	0	0	0	1	1	0
2	1	0	0	1	0	0	1	0	0
3	1	1	0	1	0	0	0	0	1
4	0	1	0	0	0	1	0	0	1
5	0	1	1	0	1	1	0	0	0
6	0	0	1	0	1	0	0	1	0

Fuente. AN857 Brushless DC Motor Control Made Easy, Microchip Technology Inc. (Brown, 2002)

Ordenadas las fases de la Tabla 2.1 por el peso en código binario de los sensores da como resultado la Tabla 2.2. La activación de los Transistores Mosfet del motor, de acuerdo con una tabla de estado construido a partir de la Tabla 2.2 hará que el motor de la Figura 2.6 gire en el sentido de las agujas del reloj.

Tabla 2.2 Sensor en el Sentido de las agujas del reloj y bits de accionamiento por orden de fase

Phase	Sensor C	Sensor B	Sensor A	C High Drive	C Low Drive	B High Drive	B Low Drive	A High Drive	A Low Drive
6	0	0	1	0	1	0	0	1	0
4	0	1	0	0	0	1	0	0	1
5	0	1	1	0	1	1	0	0	0
2	1	0	0	1	0	0	1	0	0
1	1	0	1	0	0	0	1	1	0
3	1	1	0	1	0	0	0	0	1

Fuente. AN857 Brushless DC Motor Control Made Easy, Microchip Technology Inc. (Brown, 2002).

La rotación en el sentido contrario se implementa en la Tabla 2.3 complementando las entradas de los 06 transistores Mosfet de potencia de alto a bajo de la Tabla 2.2. La activación de las bobinas del motor, de acuerdo con la Tabla 2.3, haciendo que el motor gire en sentido contrario a las agujas del reloj. Los números de fase de

la Tabla 2.3 están precedidos por una barra oblicua indicando que es opuesta a las fases en la Tabla 2.2.

Tabla 2.3 Sensor Sentido contra las agujas del reloj y Bits de accionamiento

Phase	Sensor C	Sensor B	Sensor A	C High Drive	C Low Drive	B High Drive	B Low Drive	A High Drive	A Low Drive
/6	0	0	1	1	0	0	0	0	1
/4	0	1	0	0	0	0	1	1	0
/5	0	1	1	1	0	0	1	0	0
/2	1	0	0	0	1	1	0	0	0
/1	1	0	1	0	0	1	0	0	1
/3	1	1	0	0	1	0	0	1	0

Fuente. AN857 Brushless DC Motor Control Made Easy, Microchip Technology Inc. (Brown, 2002)

2.2.4 Plataforma Zybo Z7-10

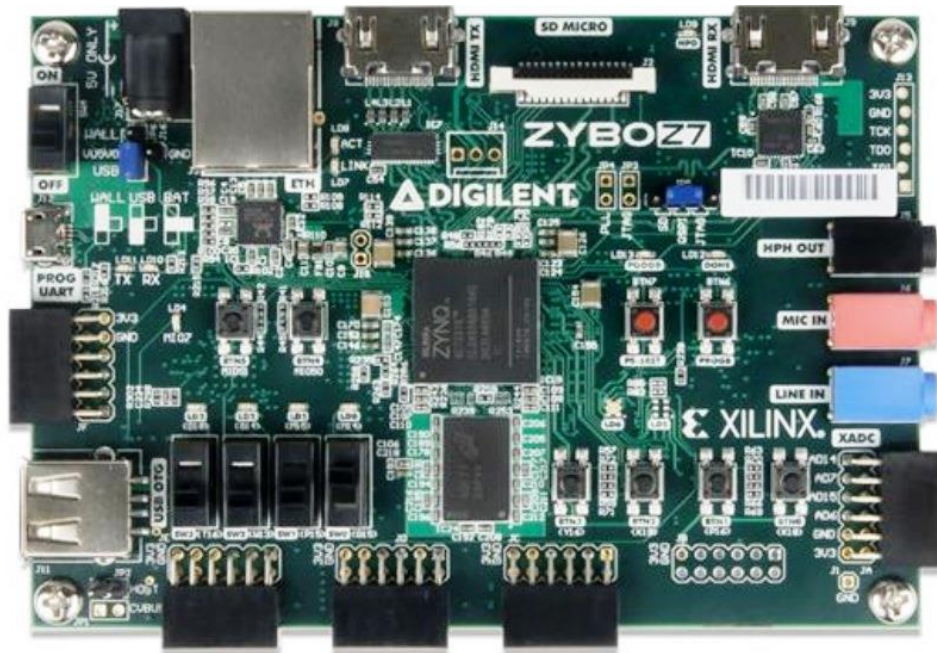


Figura 2.7 Plataforma ZYBO Z7 - 10

Fuente. (DIGILENT, Manual de ZYBO – Z7, 2018)

La placa de desarrollo ZYBO - Z7 (ZYNq BOard) como puede verse en la Figura 2.7, es una plataforma de desarrollo de software embebido e implementación de circuitería digital, construida en función de dispositivo la familia Xilinx Zynq-7000

XC7Z010CLG400ABX2041 con sus dos (02) variantes uno con el dispositivo Zynq-7010 y el otro con el dispositivo Zynq-7020. El Zynq-7010 o llamado Z7-10 (usado en el proyecto) se basa en la arquitectura Xilinx All Programmable System-on-Chip (AP SoC), que integra un procesador ARM Cortex-A9 de doble núcleo, (como puede verse en la Figura 2.8 en color celeste siendo el Sistema de Procesamiento) con el arreglo de bloques lógicos idénticos con interconexiones Programables (FPGA-Field-Programmable Gate Array) Xilinx, (como puede verse en la Figura 2.8 en color amarillo siendo la Lógica Programable). Además se combina con un conjunto de periféricos multimedia con la conectividad disponible en el ZYBO-Z7n (Zynq Z-7010), pudiéndose alojar todo un diseño de Sistema Difuso como en el presente trabajo de Tesis.

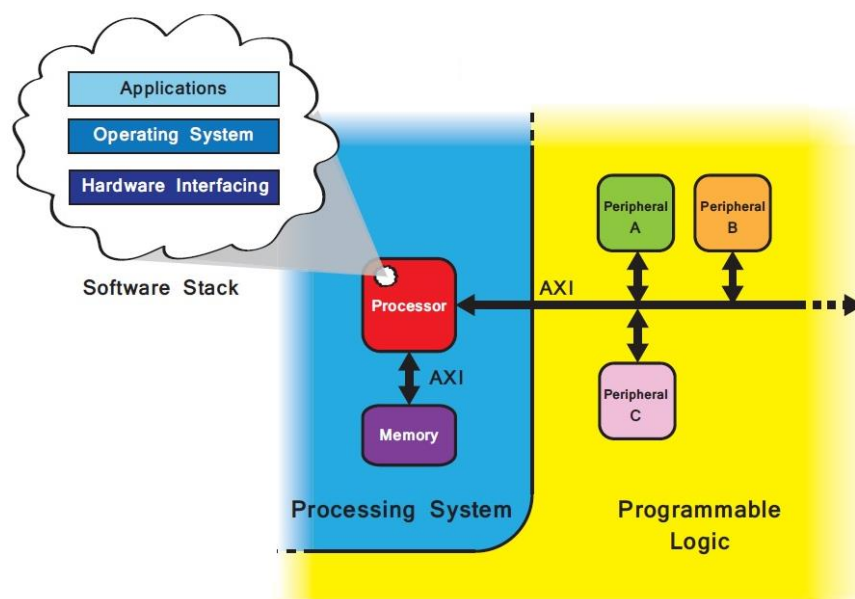


Figura 2.8 Arquitectura Zynq – Sistema software y Sistema hardware
Fuente. The Zynq Book (Crockett, Elliot , Enderwitz, & Stewart, 2014)

Con las memorias integradas, USB, Ethernet y la ranura SD (Secure Disk) se obtiene un sistema listo a trabajar sin necesidad de hardware adicional. Asimismo tiene puertos tipo Pmod (entrada y Salida) que están disponibles para manejar entradas

Digitales y entradas Análogas en un ADC y la salida análoga mediante un DAC.

2.2.5 Entorno desarrollo de software VIVADO 2020.1 y VITIS 2020.1

Es un entorno de desarrollo de software especializado en diseño de sistemas digitales y analógicos aplicado a una plataforma basado en un dispositivo SoC con interfaces al exterior con elevado grado de flexibilidad facilitando las diferentes etapas de desarrollo de un Controlador basado en lógica difusa. Para ello incluye el citado software interactuando con la citada plataforma con sus herramientas de descripción, simulación, síntesis e implementación que permiten definir, verificar, y optimizar el diseño del Controlador difuso que materia de esta Tesis.

2.2.6 El módulo Pmod

La interfaz Digilent Pmod (Peripheral Modules) se usa para conectar módulos periféricos a través puertos externos con pines de Entrada/Salida al controlador host ZYBO Z7.

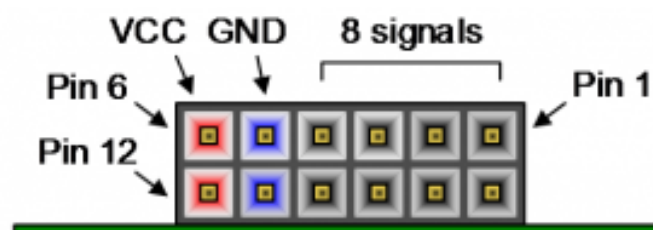


Figura 2.9 Puerto Pmod de la plataforma ZYBO Z7

Fuente. (DIGILENT, Manual de ZYBO – Z7, 2018)

Pmod AD2 I2C 4x12 bits

El Digilent Pmod AD2 (Rev. A) es un conversor análogo/digital de cuatro canales de 12 bits de resolución basado en el dispositivo AD7991 del fabricante Analog Devices.

El PmodAD2 se conecta a la placa host ZYBO Z7 través del protocolo I²C. El ZYBO Z7 pueden llamar al Pmod renviando la dirección del dispositivo de 0b0101000 seguido del bit de lectura o escritura apropiado. La elección de un bit de escritura, configura al dispositivo incorporado para seleccionar los canales adecuados o si se envía el bit de lectura se puede comenzar a leer inmediatamente los 12 bits de datos del registro de datos de 16 bits.

Tabla 2.4 Descripción de las señales del conector del PMod AD2

Pin	Signal	Description
1 & 5	SCL	Serial Clock
2 & 6	SDA	Serial Data
3 & 7	GND	Power Supply Ground
4 & 8	VCC	Power Supply (3.3V/5V)

Fuente. (DIGILENT, Hoja de Datos del PMOD AD2, 2016)

Pmod DA2 SPI 2x12 bits

El Digilent Pmod DA2 es un módulo de conversión digital/análogo de 12 bits y 2 canales con la capacidad de generar datos de salida de hasta 16,5 MSA/seg. El PmodDA2 está basado en un conversor digital/análogo de 12 bits del fabricante Texas Instruments DAC121S101. Puede convertir simultáneamente dos canales separados de información digital proporcionada a través del protocolo SPI, El PmodDA2 se comunica con el ZYBO Z7 por

el protocolo SPI, colocando el pin Chip Select (CS) a un estado de bajo, Pudiéndose enviar una serie de 16 pulsos de reloj en la línea Serial Clock (SCLK). Los datos se envían con el bit más significativo (MSB) primero en los últimos 12 pulsos de reloj.

Tabla 2.5 Descripción de las señales del conector del PMod DA2

Pin	Signal	Description
1	~SYNC	Chip Select
2	DINA	Data In for Channel A
3	DINB	Data In for Channel B
4	SCLK	Serial Clock
5	GND	Power Supply Ground
6	VCC	Power Supply (3.3V/5V)

Fuente. (DIGILENT, Hoja de Datos del PMOD DA2, 2016)

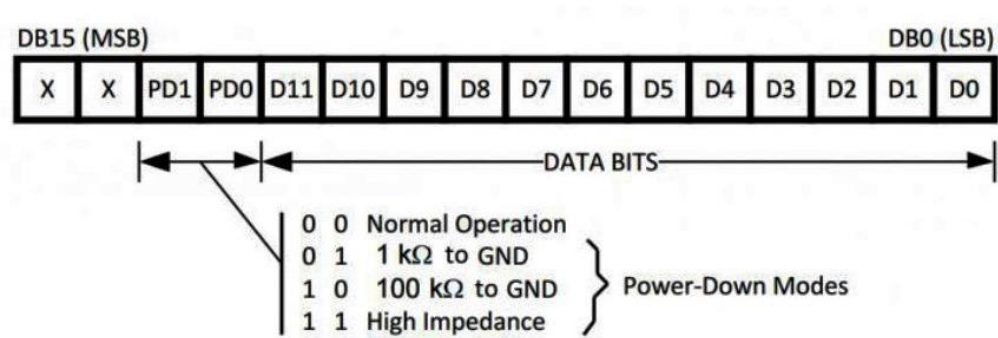


Figura 2.10 Trama del dato del PMod DA2

Fuente. (DIGILENT, Hoja de Datos del PMOD DA2, 2016)

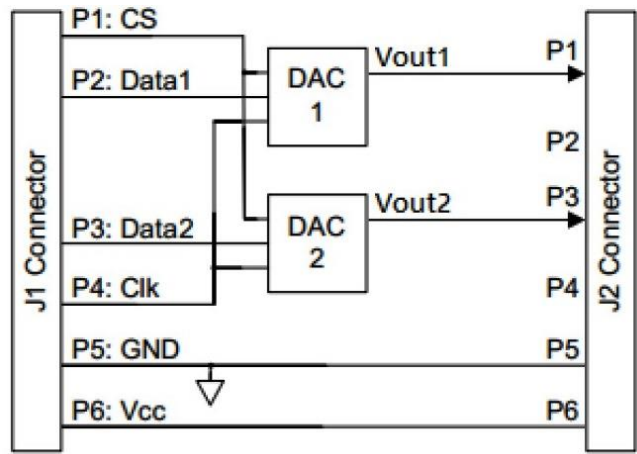


Figura 2.11 Diagrama del dato del PMod DA2

Fuente. (DIGILENT, Hoja de Datos del PMOD DA2, 2016)

PMod OLED 128×32

El Digilent Pmod OLED (Rev. A) es un módulo LED orgánico con una pantalla grafica de 128×32 píxeles.

El Pmod OLED se comunica con la placa host ZYBO Z7 a través del protocolo SPI. Para manejarlo se mantiene el pin Chip Select (CS) en un nivel lógico bajo, se puede enviar comandos y/o datos al controlador de pantalla OLED según el estado del pin de datos/comandos (D/C).

Tabla 2.6 Descripción de las señales del conector del PMod OLED 128x32

Connector J1		
Pin	Signal	Description
1	CS	SPI Chip Select (Slave Select)
2	SDIN	SPI Data In (MOSI)
3	None	Unused Pin
4	SCLK	SPI Clock
7	D/C	Data/Command Control
8	RES	Power Reset
9	VBATC	V _{BAT} Battery Voltage Control
10	VDDC	V _{DD} Logic Voltage Control
5, 11	GND	Power Supply Ground
6, 12	VCC	Power Supply

Fuente: (DIGILENT, Hoja de Datos del PMOD OLED, 2016)

Capítulo 3 HIPOTESIS Y VARIABLES

3.1 Hipótesis General

El Diseño e Implementación de un Controlador de lógica difusa con tecnología FPGA XILINX 7 Y ARM CORTEX A9 que será aplicado a un Sistema de control de velocidad de un motor de corriente continua BLDC, brindando un mejor desempeño que un Controlador convencional.

3.2 Hipótesis específica

- El Controlador por Lógica difusa con tecnología FPGA XILINX 7 y ARM CORTEX A9, es un diseño de interacción hardware/software de alta velocidad de ejecución que además tiene mejores características y versatilidad que el controlador convencional.
- El Desarrollo del software del Controlador difuso logrará un diseño flexible y eficiente que un controlador convencional.
- El Controlador por Lógica Difusa con tecnología FPGA XILINX 7 y ARM CORTEX A9 usado para un Sistema de Control de velocidad de un motor de corriente continua BLDC, es de actualización muy rápida en comparación con los Controladores convencionales.

3.3 Identificación de las Variables

Variables Independientes:

1. Uso de la tecnología bajo el *enfoque FPGA XILINX 7 y ARM CORTEX A9*
2. Diseño e implementación del *Controlador por lógica difusa*
3. Desarrollo del algoritmo difuso con el software en VHDL y C

Variables Dependientes:

1. La obtención de un Sistema de control difuso de velocidad del motor BLDC (medidos en función de transitorios de la respuesta del sistema).
2. Agregar nuevas funciones (upgrade) para aumentar la vida útil del controlador diseñado.
3. Respuesta rápida y eficiente a cambios futuros de la aplicación

Capítulo 4 METODOLOGIA

Para el diseño del Controlador Difuso con enfoque SoC , primero como el controlador difuso forma parte del Sistema de Control (Fig. 4.1) para luego obtener las consideraciones de diseño que luego se desarrolla e implementa en el Zybo Z7-10 y también dicha consideraciones se desarrolla la simulación en el MatLab -Simulink- Fuzzy Logic Tool Box para construir un Sistema de Control de Velocidad de un motor BLDC..

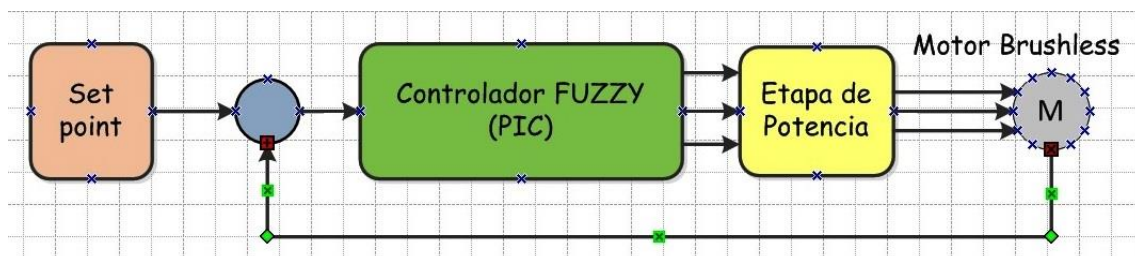


Figura 4.1 Diagrama de Bloques de un Sistema de Control de Velocidad de un motor BLDC

Fuente. Elaboración propia

Este Controlador Difuso tiene los siguientes consideraciones de diseño:

- a. La variable de entrada ERROR (conjunto difuso) es:
 GN (1/-2000, 1/-1735, 0/550), SN (0/-1560, 1/1050, 0/-300),
 ZE (0/-550, 1/0, 1/550), SP (0/100, 1/900, 0/1600) y
 GP (0/1000, 1/1400, 1/2000)
- c. La variable de SALIDA (conjunto difuso) es:
 GN (1/-25, 1/-9, 0/-6), SN (0/-7, 1/-4, 0/-1),
 ZE (0/-3, 1/0, 1/2), SP (0/1, 1/4, 0/7)
 GP (0/6, 1/7, 1/25).

Las Reglas de Control (estrategia de control) son:

- 1) IF ERROR is GN THEN SALIDA is GN
- 2) IF ERROR is SN THEN SALIDA is SN
- 3) IF ERROR is ZE THEN SALIDA is ZE
- 4) IF ERROR is SP THEN SALIDA is SP
- 5) IF ERROR is GP THEN SALIDA is GP

4.1 Sistema de Control de Velocidad de un motor BLDC

Se presenta un diagrama de bloques de un Sistema de Control por Lógica Difusa (Fuzzy Logic) de un motor (Brushless – BLDC) para regular su velocidad (ver Figura 4.1), el Controlador tendrá el enfoque SoC (FPGA XILINX 7 y ARM CORTEX A9), con las siguientes características del Sistema de Control:

- Velocidad Programable entre 1,000 RPM y 3,000 RPM
- Señal de entrada de los sensores de velocidad del motor BLDC.
- Señales de salida que van al Puente de trifásico construida en MOSFET de potencia, desfasadas en 120°

Comparación de un Controlador Difuso con un Controlador Convencional PI.

Se utilizó la herramienta Simulink de MatLab para la simulación por comparación de los dos (02) Sistemas de Control de

Velocidad del motor brushless uno de ellos es el PI y el otro el Difuso. Cabe indicar que la contribución derivativa en el PID despreciable en este caso por lo que se usara el adecuado que es el PI y con los parámetros siguientes el Proporcional = 0.013 y el Integrativo = 16.61, como puede verse en el Simulink de la Figura 4.2. Con el fin de demostrar dicha comparación se incluye un motor BLDC de 1 KW, 500 VDC, 3000 RPM de la biblioteca Specialized Power Systems de MatLab (MathWorks I. , 2024).

Se usado un motor equivalente al BLDC HST231320A11 cuyo manual se encuentra en el -Anexo C.

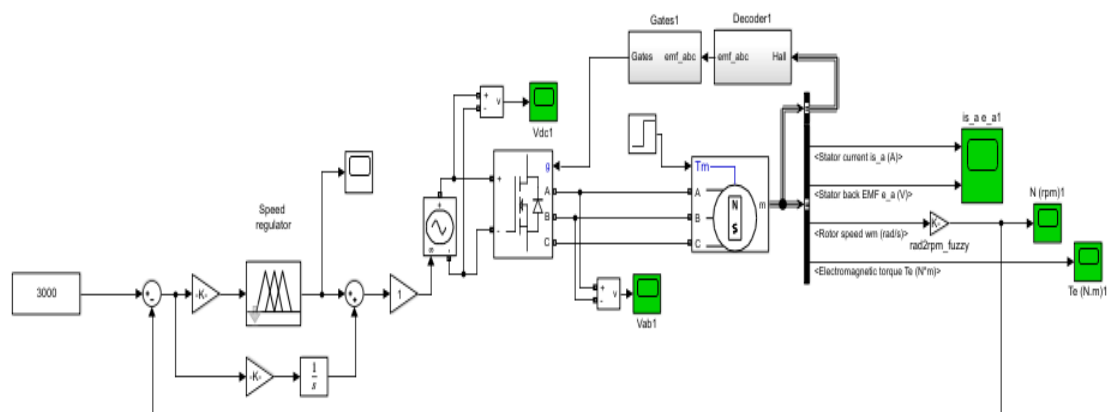


Figura 4.2 Sistema de Control Difuso de Velocidad de un motor BLDC

Fuente. MatLab 2020a– Simulink (MathWorks, 1999)

Se presenta los conjuntos difusos en las Figura 4.3, 4.4, 4.5, 4.6 y 4.7 usado el Toolbox de Lógica difusa del MatLab. También en las Figuras 4.6 y 4.7 se encuentran las Reglas de control.

Después de defuzzyficado por la técnica de Mamdani en el Fuzzy Logic Toolbox de MatLab genera un archivo (ver la Figura 4.3). que es usado en el Controlador difuso del Simulink Luego se corre el Simulink (ver la Figura 4.2) para obtener el resultado de la comparación de ambos controles.

Entonces para obtener el archivo Defuzzyficado, se realiza inicialmente ingresando al Fuzzy Logic Toolbox de MatLab como puede verse en las Figuras 4.4, 4.5 y 4.6 para luego seguir con el procedimiento como puede verse en las figuras siguientes.

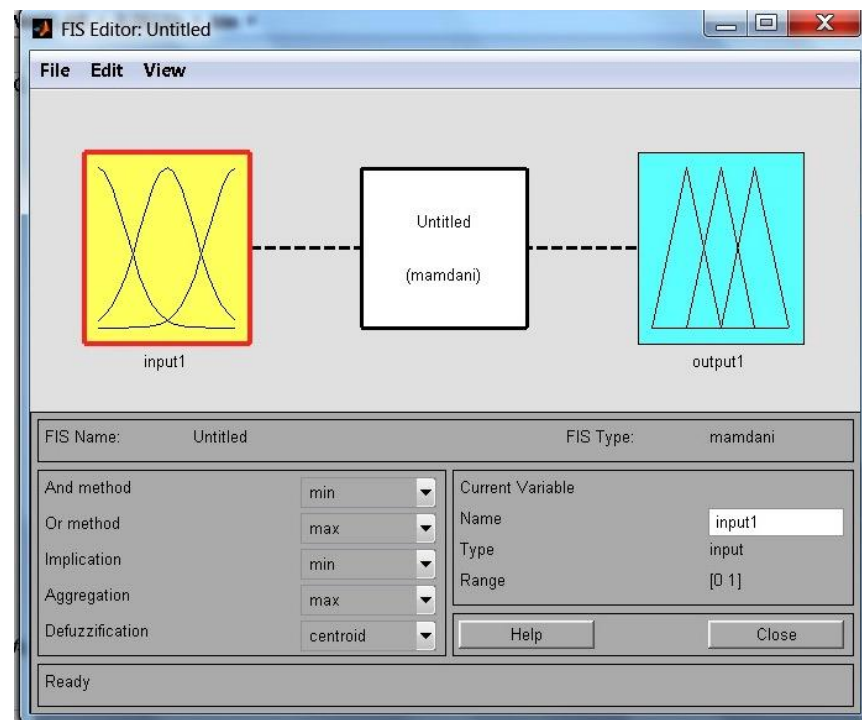


Figura 4.3 El Conjunto Difuso en Fuzzy Logic Toolbox

Fuente. MatLab 2020a– Simulink (MathWorks, 1999)

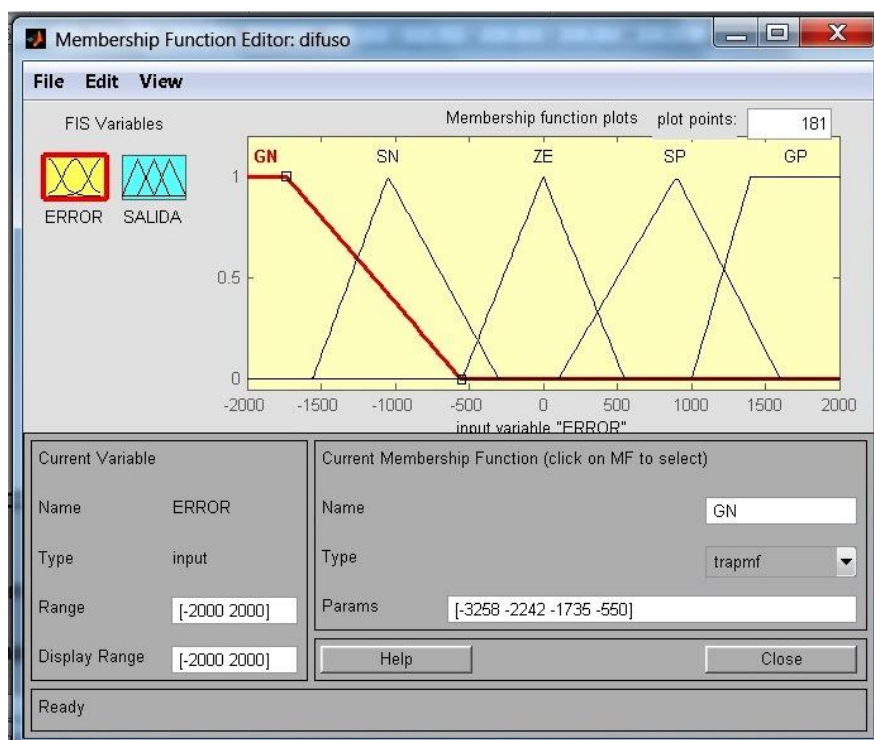


Figura 4.4 El Conjunto Difuso de Entrada en Fuzzy Logic Toolbox

Fuente. MatLab 2020a– Simulink (MathWorks, 1999)



Figura 4.5 El Conjunto Difuso de Salida en Fuzzy Logic Toolbox

Fuente. MatLab 2020a– Simulink (MathWorks, 1999)

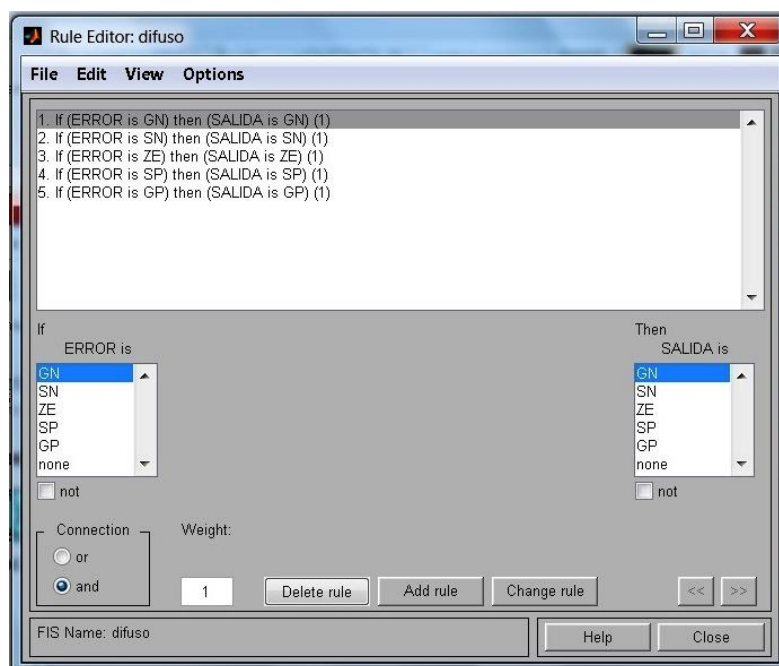


Figura 4.6 Las Reglas de Control en Fuzzy Logic Toolbox

Fuente. MatLab 2020a– Simulink (MathWorks, 1999)

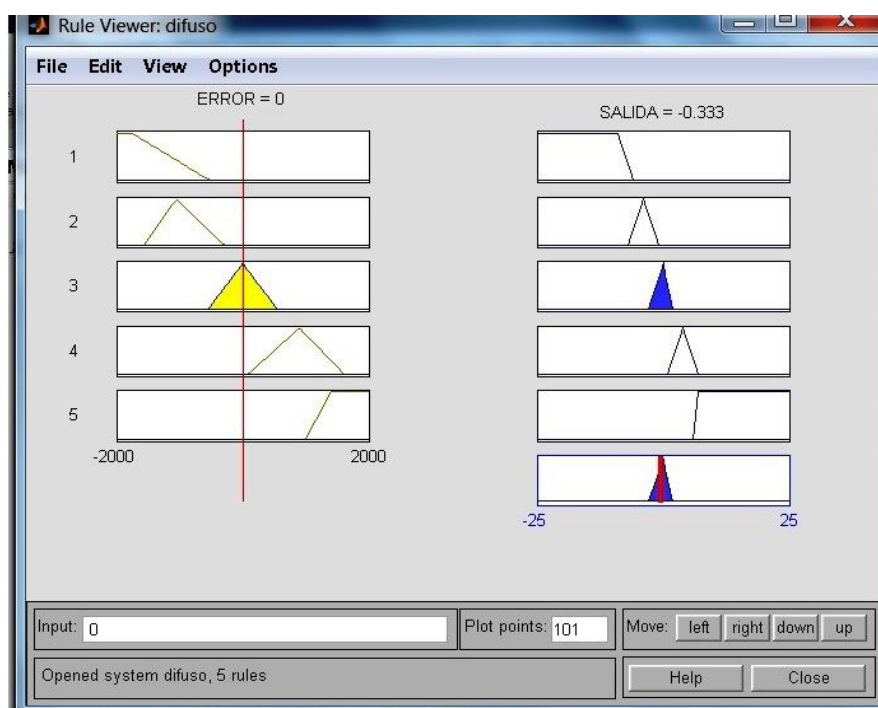


Figura 4.7 Los Diagramas de las Reglas de Control en Fuzzy Logic Toolbox

Fuente. MatLab 2020a– Simulink (MathWorks, 1999)

Usando el Scope del Simulink de la Figura 4.2, podemos ver en la Figura 4.8 las tres señales en el tiempo de Entrada y Salidas

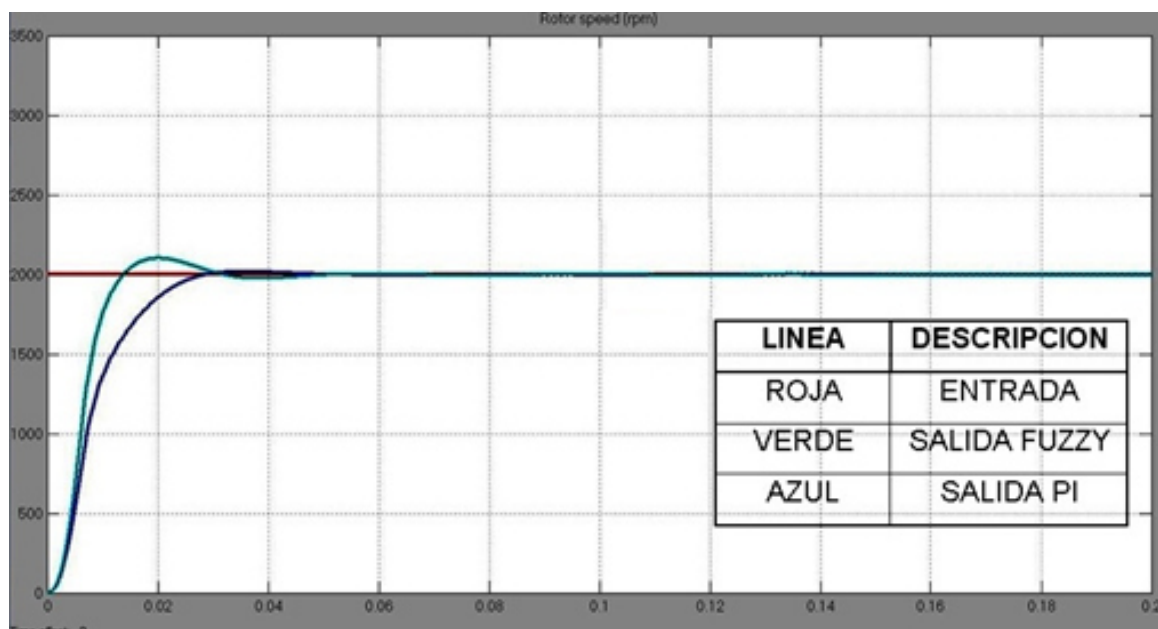


Figura 4.8 La Respuesta Transitoria del Controlador Difuso y el Control PI en

Fuente. MatLab 2020a– Simulink (MathWorks, 1999)

Se nota evidentemente que para un Set Point de 2,000 RPM (representado por una función escalón) el Controlador Difuso tiene una mejor respuesta en el tiempo otorgando un control más rápido, esto se observa con el tiempo subida es mucho mayor que el del controlador PI. Asimismo, puede indicarse que ambos presentan un tiempo de establecimiento similar.

4.2 Alternativas de Diseño de Controlador Difuso

A lo largo de los últimos años se han propuesto diversas alternativas para implementar los Controladores Difusos agrupándose en desarrollo en software, desarrollo en hardware y desarrollos híbridos.

- Los desarrollos en software han sido las soluciones más ampliamente adoptadas y se caracterizan por su elevado grado de flexibilidad, aunque su tiempo de respuesta viene limitado por la inherente ejecución secuencial de los programas.
- Los desarrollos en hardware del controlador difuso representan una alternativa totalmente opuesta a la anterior y son requeridas para aquellas aplicaciones en las que la velocidad del proceso sea un factor determinante.
- Los desarrollos híbridos basados en técnicas de codiseño hardware/software (HW/SW) permiten obtener un compromiso adecuado entre las ventajas e inconvenientes de las alternativas anteriores.

El presente trabajo de Tesis usa el desarrollo híbrido permitiendo que la distribución entre la implementación hardware y la ejecución software de las tareas que debe realizar el Controlador difuso permitiendo obtener una solución con mejores características de flexibilidad y velocidad. Dicha selección de la alternativa del desarrollo Híbrido hace posible la estrategia de dividir las tareas que consiste en la implementación sobre hardware/software de una plataforma flexible (sistema electrónico digital) con la unión de la otra parte que es el desarrollo del software del procesamiento difuso.

4.3 Diseño del Controlador Difuso

El diseño, la síntesis y la simulación de un Control Difuso usando las herramientas de software como el VIVADO DESIGN SUITE 2020.2, y el VITIS 2020.2 para sintetizar, simular, depurar e implementar en un dispositivo digital con Arquitectura AP SoC (All Programmable System-on-Chip) ZYNQ XC7Z010-1CLG400C de XILINX

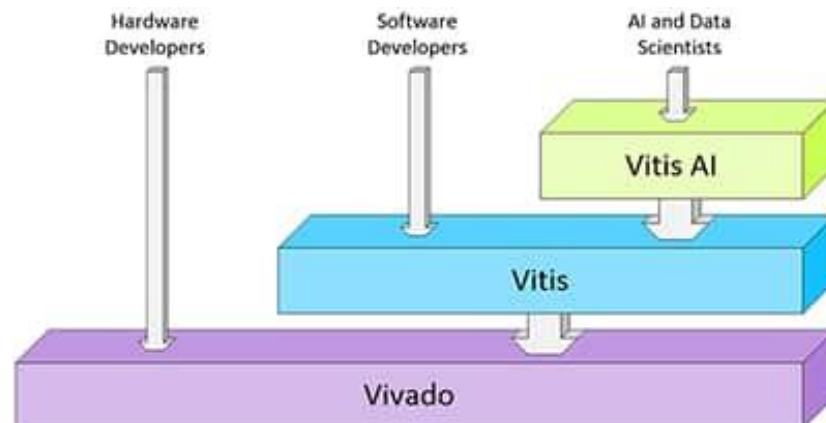


Figura 4.9 Herramientas de diseño de Xilinx VIVADO y VITIS.

Fuente. Fundamentals of FPGAs – Part 4: Getting Started with Xilinx FPGAs (Maxfield, 2020)

Como puede verse en la Figura 4.9 los diseñadores pueden trabajar con las herramientas en los niveles de abstracción más adecuados. Los diseñadores de hardware trabajan con VIVADO, y los desarrolladores de software trabajan con VITIS, y además los diseñadores pueden trabajar en un nivel de abstracción adicional donde se encuentra la Inteligencia Artificial (IA) trabajan con VITIS AI (Artificial Intelligence).

En el diseño del Controlador Difuso se usa la Plataforma de Desarrollo ZYBO con el VIVADO DESIGN SUITE de XILINX y el VITIS que combinan el diseño lógico FPGA (programación en VHDL) con el desarrollo del software en lenguaje C para el procesador ARM CORTEX A9), se usa esta tecnología para realizar el diseño propuesto

con el fin de lograr el manejo de la complejidad de este controlador materia de esta Tesis, puede verse dicho Controlador Difuso en bloques pueda verse en la Figura 4.10.

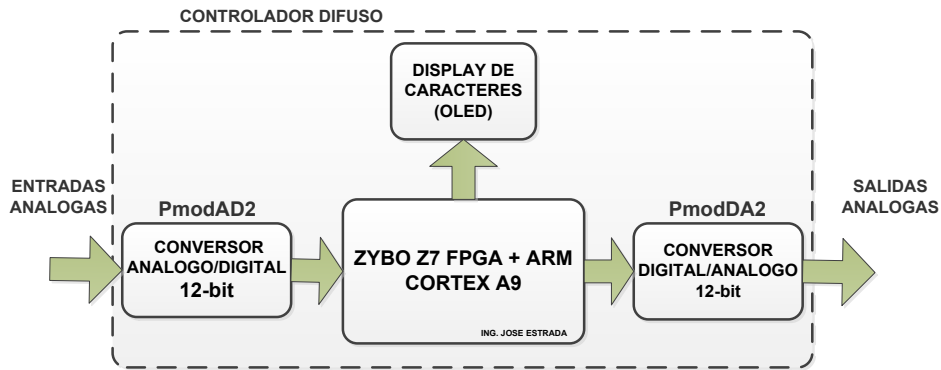


Figura 4.10 Diagrama de bloques del Controlador Difuso

Fuente. Elaboración propia

El Controlador difuso plantea mejorar el sistema previamente sintonizado por lo cual se realizó un aumento de la acción integral, y el control difuso de la acción proporcional como alternativa para el mejoramiento del controlador, para ello se determinaron los siguientes conjuntos difusos de entrada y salida del controlador.

4.3.1 Implementación del Sistema de Control Difuso.

Para la implementación de un Control difuso se realizó usando los módulos Pmod y la placa Zybo Z7-10, tal como puede verse en este capítulo

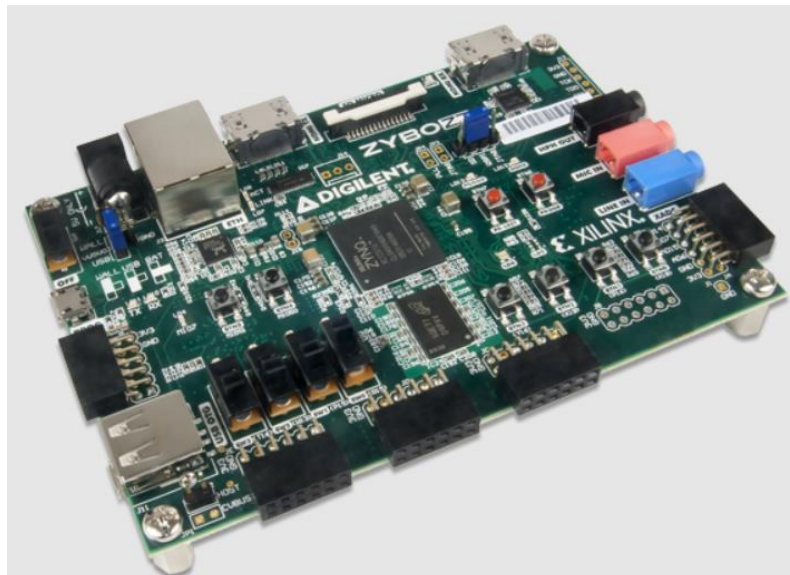


Figura 4.11 Zybo Z7-10

Fuente. (DIGILENT, Manual de ZYBO – Z7, 2018)



Figura 4.12 Pmod AD2

Fuente. (DIGILENT, Hoja de Datos del PMOD AD2, 2016)

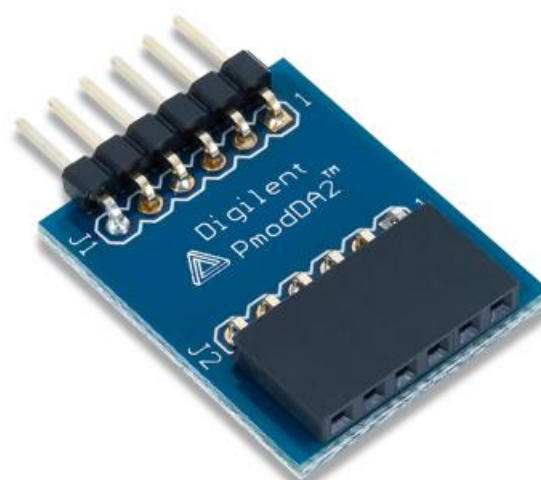


Figura 4.13 Pmod DA2

Fuente. (DIGILENT, Hoja de Datos del PMOD DA2, 2016)



Figura 4.14 Pmod OLED

Fuente: (DIGILENT, Hoja de Datos del PMOD OLED, 2016)

Para la configuración se usó los bloques IP (Intellectual Property), para su rápida configuración, además de ello algunos módulos Pmod traen consigo, algunas librerías para agilizar la implementación de estas, (DIGILENT, Hoja de Datos del PMOD OLED, 2016) (DIGILENT, Digilent/vivado-library, 2022) sin embargo el módulo Pmod DA2, requirió el uso y configuración propia de un bloque SPI integrado en el Software VIVADO 2020-1 dado que no se contaba con dicha librería en el repositorio, a continuación se plantea el diagrama de conexiones que se realizó para ello.

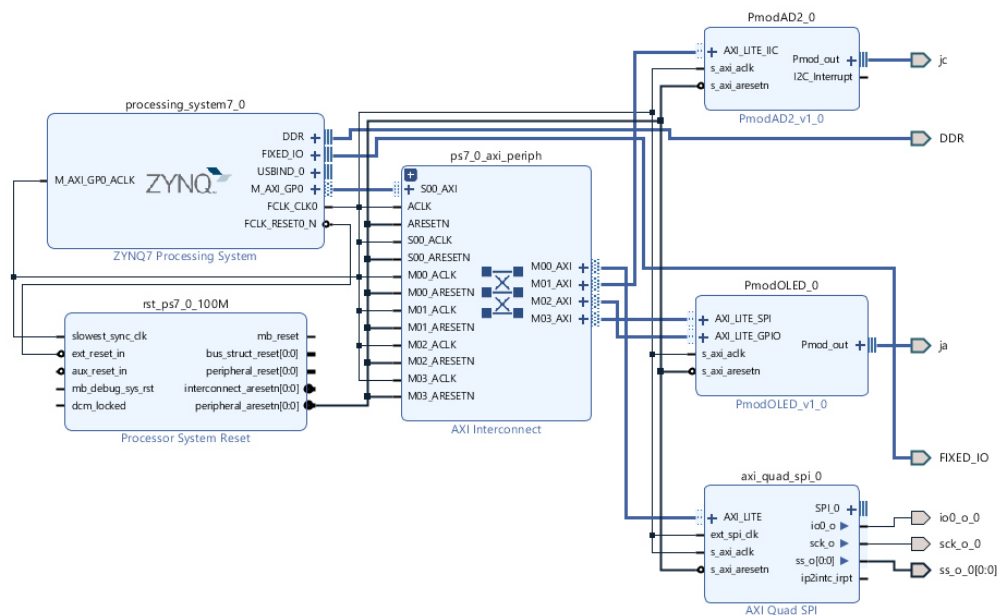


Figura 4.15 IP Integrator del Controlador Fuzzy

Fuente. VIVADO 2020.1

Se agregó 4 bloques IP los cuales son `axi_quad_spi`, `PmodAD2`, `PmodOled` y el IP `ZYNQ7 Processing System`, como se puede ver en la Figura 4.14 se han agregado 2 bloques adicionales, los cuales configuran los periféricos y el bus de datos AXI.

Dado que el módulo `Pmod AD2` no cuenta con una librería que facilite su uso, se hace uso del bloque IP `axi_quad_spi_0` que nos proporciona una interfaz para poder realizar la comunicación por protocolo SPI, como se puede observar en la Figura 4.15 se configura el modo de trabajo y se ajusta la frecuencia de trabajo con un frequency Ratio igual a 4 dado que se configuró la frecuencia de trabajo del Zybo a 100Mhz, por lo que el clock para la comunicación tendrá una frecuencia de 25 MHz, la cual es menor de la frecuencia de trabajo del convertidor digital análogo indicada en la hoja de datos de dicho DA2.

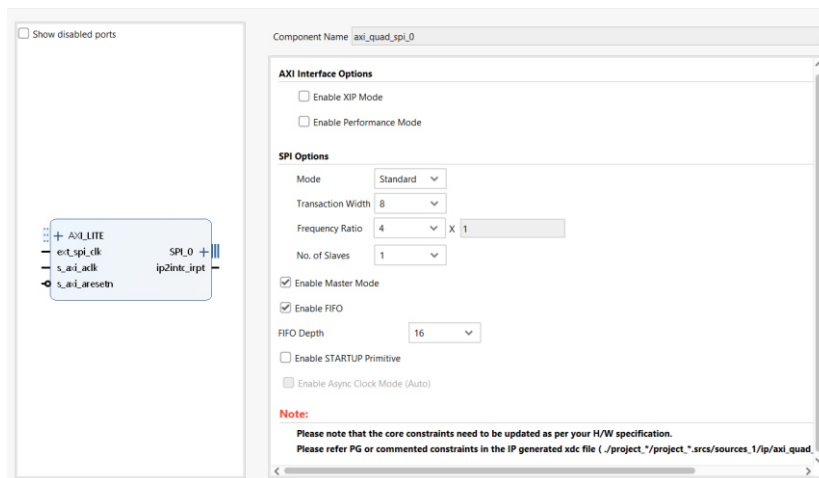


Figura 4.16 Configuración del IP para el SPI del Pmod AD2
Fuente. VIVADO 2020.1

Luego se agrega el repositorio de IP para los Pmods lo cual agilizará la implantación.

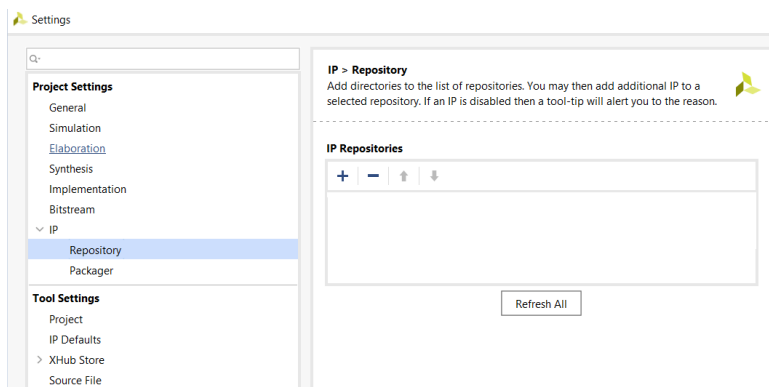


Figura 4.17 Repositorio IP
Fuente. VIVADO 2020.1

Una vez realizado ello se procede a hacer la conexión de pines y asignación mediante el archivo de constraints (DIGILENT, Digilent/vivado-library, 2022) , para la implementación se usaran los puertos Pmod JA, JC y JD, dado que la tarjeta Zybo Z7-10, no trae esos pines conectados.

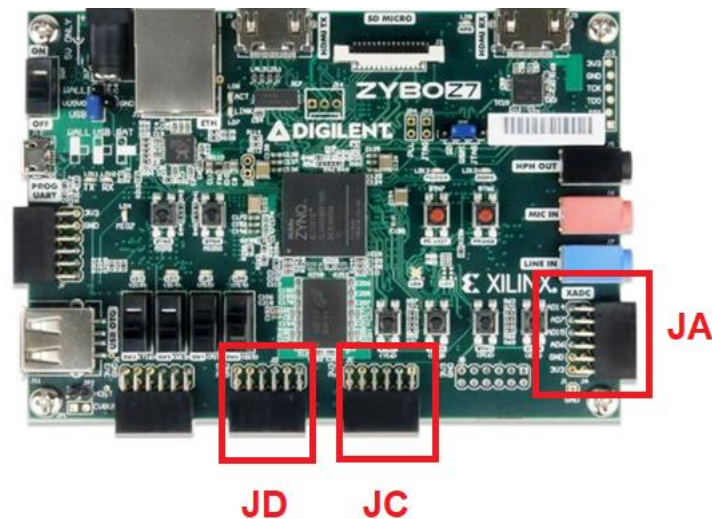


Figura 4.18 Plataforma Zybo Z7

Fuente. (DIGILENT, Manual de ZYBO – Z7, 2018)

	Pmod JA	Pmod JB*	Pmod JC	Pmod JD	Pmod JE	Pmod JF
Pmod Type	XADC	High-Speed	High-Speed	High-Speed	Standard	MIO
Pin 1	N15	V8	V15	T14	V12	MIO-13
Pin 2	L14	W8	W15	T15	W16	MIO-10
Pin 3	K16	U7	T11	P14	J15	MIO-11
Pin 4	K14	V7	T10	R14	H15	MIO-12
Pin 7	N16	Y7	W14	U14	V13	MIO-0
Pin 8	L15	Y6	Y14	U15	U17	MIO-9
Pin 9	J16	V6	T12	V17	T17	MIO-14
Pin 10	J14	W6	U12	V18	Y17	MIO-15

Figura 4.19 Pin Out de los puertos del Zybo Z7

Fuente. (DIGILENT, Manual de ZYBO – Z7, 2018)

Pmod AD2

Se le asigna la salida al puerto JC para ello debemos considerar que la comunicación de este módulo es por I²C.

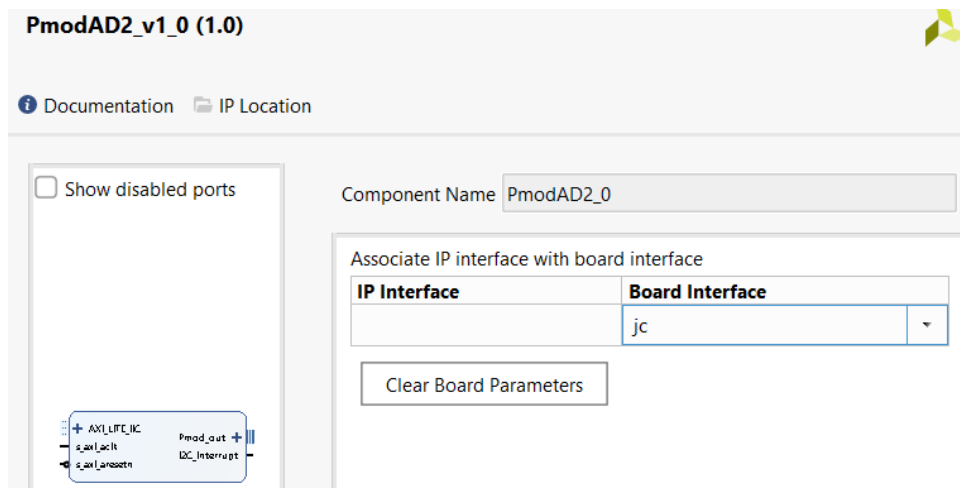


Figura 4.20 Asignación del puerto JC para el pmod AD2

Fuente. VIVADO 2020.1

Luego se asigna los pines a los pines respectivos en el archivo de constraints (.XDC)

```
#Pmod Header JC
set_property -dict { PACKAGE_PIN V15 IOSTANDARD LVCMOS33 } [get_ports { jc[0] }]; #IO_L10P_T1_34 Sch=jc_p[1]
set_property -dict { PACKAGE_PIN W15 IOSTANDARD LVCMOS33 } [get_ports { jc[1] }]; #IO_L10N_T1_34 Sch=jc_n[1]
set_property -dict { PACKAGE_PIN T11 IOSTANDARD LVCMOS33 } [get_ports { jc[2] }]; #IO_L1P_T0_34 Sch=jc_p[2]
set_property -dict { PACKAGE_PIN T10 IOSTANDARD LVCMOS33 } [get_ports { jc[3] }]; #IO_L1N_T0_34 Sch=jc_n[2]
set_property -dict { PACKAGE_PIN W14 IOSTANDARD LVCMOS33 } [get_ports { jc[4] }]; #IO_L8P_T1_34 Sch=jc_p[3]
set_property -dict { PACKAGE_PIN Y14 IOSTANDARD LVCMOS33 } [get_ports { jc[5] }]; #IO_L8N_T1_34 Sch=jc_n[3]
set_property -dict { PACKAGE_PIN T12 IOSTANDARD LVCMOS33 } [get_ports { jc[6] }]; #IO_L2P_T0_34 Sch=jc_p[4]
set_property -dict { PACKAGE_PIN U12 IOSTANDARD LVCMOS33 } [get_ports { jc[7] }]; #IO_L2N_T0_34 Sch=jc_n[4]
```

Figura 4.21 Restricciones (constraints) para el Pmod AD2

Fuente. VIVADO 2020.1

Pmod DA2

Se le asigna la salida al puerto JD, para ello debemos considerar que la comunicación de este módulo es por SPI y para el caso requiere una línea de selección (CS), una línea de datos (MOSI) y una línea de reloj (SCK), que para el caso son ss_o_0, i0_o_0 y sck_o_0 respectivamente (Figura 4.22)

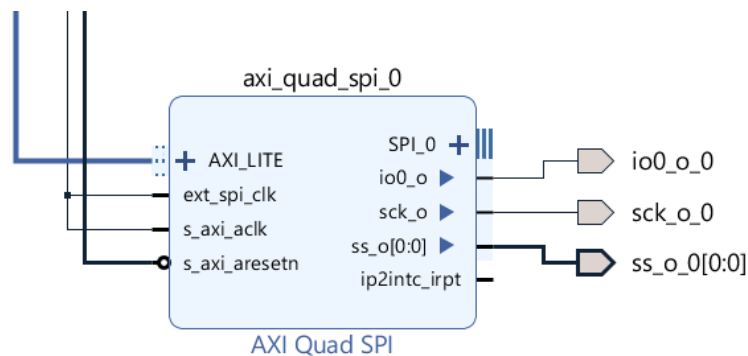


Figura 4.22 Asignación del puerto JD para Pmod DA2

Fuente. VIVADO 2020.1

Luego se asigna los pines a los pines respectivos en el archivo de constraints (.XDC)

```
##Pmod Header JD
set_property -dict { PACKAGE_PIN T14 IOSTANDARD LVCMOS33 } [get_ports { ss_o_0 }]; #IO_L5P_T0_34 Sch=jd_p[1]
set_property -dict { PACKAGE_PIN T15 IOSTANDARD LVCMOS33 } [get_ports { io0_o_0 }]; #IO_L5N_T0_34 Sch=jd_n[1]
#set_property -dict { PACKAGE_PIN P14 IOSTANDARD LVCMOS33 } [get_ports { jd[2] }]; #IO_L6P_T0_34 Sch=jd_p[2]
set_property -dict { PACKAGE_PIN R14 IOSTANDARD LVCMOS33 } [get_ports { sck_o_0 }]; #IO_L6N_T0_VREF_34 Sch=jd_n[2]
#set_property -dict { PACKAGE_PIN U14 IOSTANDARD LVCMOS33 } [get_ports { jd[4] }]; #IO_L11P_T1_SRCC_34 Sch=jd_p[3]
#set_property -dict { PACKAGE_PIN U15 IOSTANDARD LVCMOS33 } [get_ports { jd[5] }]; #IO_L11N_T1_SRCC_34 Sch=jd_n[3]
#set_property -dict { PACKAGE_PIN V17 IOSTANDARD LVCMOS33 } [get_ports { jd[6] }]; #IO_L21P_T3_DQS_34 Sch=jd_p[4]
#set_property -dict { PACKAGE_PIN V18 IOSTANDARD LVCMOS33 } [get_ports { jd[7] }]; #IO_L21N_T3_DQS_34 Sch=jd_n[4]
```

Figura 4.23 Restricciones (constraints) para el Pmod DA2

Fuente. VIVADO 2020.1

Pmod OLED

Se le asigna la salida al puerto JA

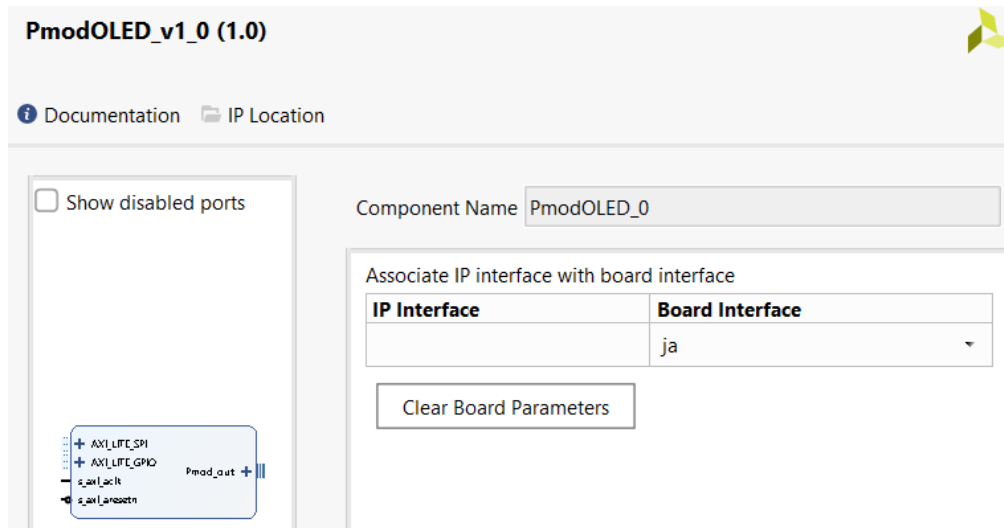


Figura 4.24 Asignación para el pmod OLED

Fuente. VIVADO 2020.1

Luego se asigna los pines a los pines respectivos en el archivo de constraints(.XDC)

```
#Pmod Header JA (XDC)
set_property -dict { PACKAGE_PIN N15 IOSTANDARD LVCMOS33 } [get_ports { ja[0] }]; #IO_L21P_T3_DQS_AD14P_35 Sch=JA1_R_P
set_property -dict { PACKAGE_PIN L14 IOSTANDARD LVCMOS33 } [get_ports { ja[1] }]; #IO_L22P_T3_AD7P_35 Sch=JA2_R_P
set_property -dict { PACKAGE_PIN K16 IOSTANDARD LVCMOS33 } [get_ports { ja[2] }]; #IO_L24P_T3_AD15P_35 Sch=JA3_R_P
set_property -dict { PACKAGE_PIN K14 IOSTANDARD LVCMOS33 } [get_ports { ja[3] }]; #IO_L20P_T3_AD6P_35 Sch=JA4_R_P
set_property -dict { PACKAGE_PIN N16 IOSTANDARD LVCMOS33 } [get_ports { ja[4] }]; #IO_L21N_T3_DQS_AD14N_35 Sch=JA1_R_N
set_property -dict { PACKAGE_PIN L15 IOSTANDARD LVCMOS33 } [get_ports { ja[5] }]; #IO_L22N_T3_AD7N_35 Sch=JA2_R_N
set_property -dict { PACKAGE_PIN J16 IOSTANDARD LVCMOS33 } [get_ports { ja[6] }]; #IO_L24N_T3_AD15N_35 Sch=JA3_R_N
set_property -dict { PACKAGE_PIN J14 IOSTANDARD LVCMOS33 } [get_ports { ja[7] }]; #IO_L20N_T3_AD6N_35 Sch=JA4_R_N
```

Figura 4.25 Restricciones (constraints)para el Pmod OLED

Fuente. VIVADO 2020.1

Una vez realizado ello se procede a generar el archivo bitstream para posteriormente complementarlo con el software VITIS 2020.1, en la programación usando los registros, para ello se debe de considerar la dirección de registro de los bloques IP con las que se trabajada dado que con ello se realizara la posterior programación.

/PmodAD2_0	AXI_LITE_IIC	Reg0	0x4000_0000	4K	0x4000_0FFF
/axi_quad_spi_0	AXI_LITE	Reg	0x41E0_0000	64K	0x41E0_FFFF
/PmodOLED_0	AXI_LITE_GPIO	Reg0	0x4000_1000	4K	0x4000_1FFF
/PmodOLED_0	AXI_LITE_SPI	Reg0	0x4001_0000	64K	0x4001_FFFF

Figura 4.26 Dirección de Registros IP

Fuente. VIVADO 2020.1

Una vez hecho ello se procede a usar el software VITIS para la programación, además de ello los registros previamente configurados en el software VIVADO se encuentran una la librería “parameters.h”, lo que facilita su uso y configuración.

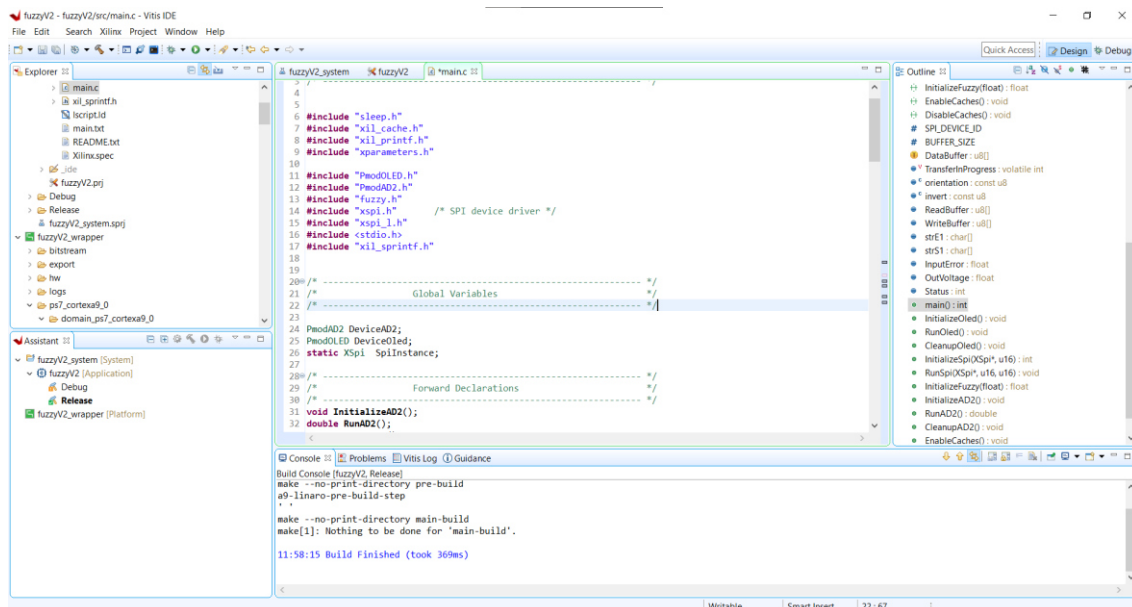


Figura 4.27 Programación en el Software VITIS

Fuente. VITIS 2020.1

4.3.2 Consideraciones de la programación del Algoritmo difuso

Cabe indicar que para el correcto funcionamiento del algoritmo Fuzzy se debe usar un conjunto difuso en el cual el valor de la membresía no supere el valor de uno. Durante la etapa previa de prueba se hizo en la plataforma ZYBO, luego se migró a la plataforma ZYBO Z7-10, usándose el software VIVADO en su versión 2020.1, la cual tenía problemas al momento de usar librerías de lenguaje C estándar como “mat.h”, algo que no sucedía en el software VIVADO en su versión 2020.1, cuando se realizó las pruebas con la plataforma ZYBO Z7-10.

Durante la implementación se detectó algunos problemas en el uso de la función printf y sprintf, dado que el programa no trabajaba cuando se hacía varios usos de estas, la cual se debía a que el uso de la función sprintf demanda demasiada memoria, es por ello que para evitar estos problemas, se usa la función xil_printf la cual es una alternativa ligera a la función sprintf, in embargo está limitada dado que no es compatible a todos los formatos de cadena como las de punto flotante;

también se plantea otra solución la cual consta de usar un entorno de subprocesos múltiples, como forma de prevenir funcionamientos incorrectos.

Lográndose implementar correctamente el algoritmo con una alta precisión, lo cual se pudo ver reflejado en los valores obtenidos en la simulación haciendo uso del Software MATLAB 2020a y que para programar en el software VITIS se debe considerar, y tener precaución con el uso de algunas funciones dado que se puede provocar overflow y provocar que el programa tenga errores.

4.4 Implementación del Controlador Difuso.

Se realizó la implementación de un controlador difuso para un motor Brushless DC, la cual se sintonizó con un control proporcional integrativo

(PI), en la cual se analizó la respuesta al escalón como se observa en las figuras 4.28, 4.29, y 4.30.

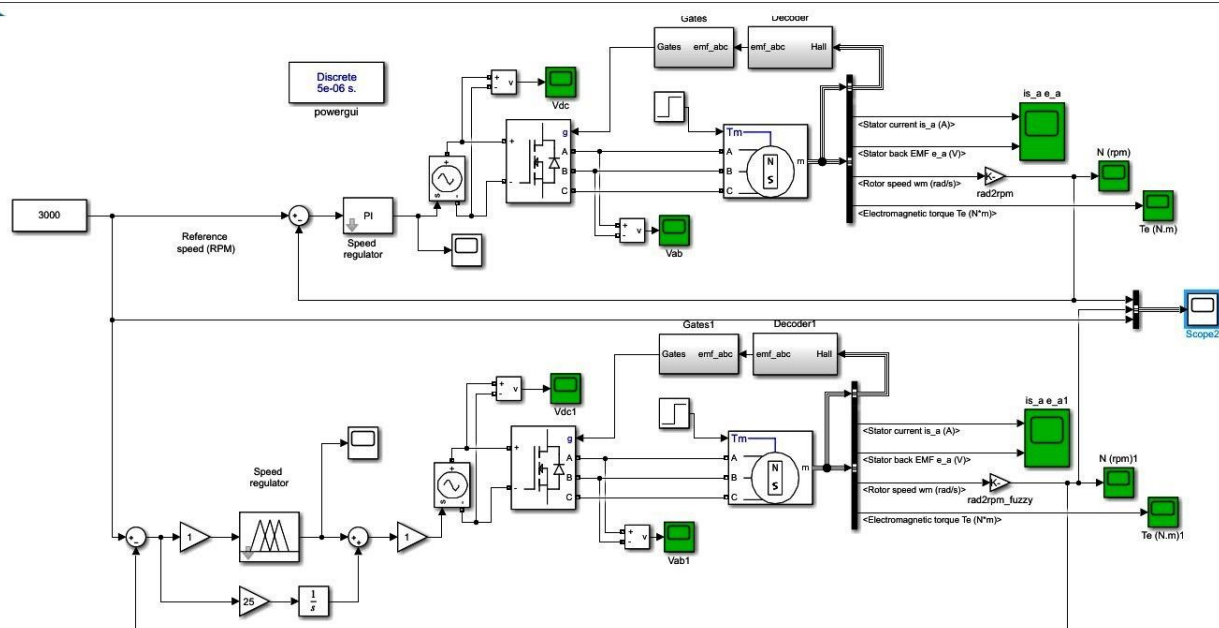


Figura 4.28 Esquema de la Simulación de un Sistema de Control de un motor BLDC con Controlador Fuzzy y Controlador PI

Fuente. MatLab 2020a– Simulink (MathWorks, 1999)

Para la comparación se ha realizado de acuerdo a los recursos de MatLab (Olivier Tremblay, 2024):

- Un motor BLDC de 1 kW, 500 Vdc, 3000 rpm alimentado por un inversor de voltaje de seis pasos.
- El inversor es un puente MOSFET de la biblioteca Specialized Power Systems.
- Se utiliza un regulador de velocidad para controlar el voltaje del bus de CC.
- El par de carga aplicado al eje de la máquina se establece primero en 0 y se incrementa a su valor nominal (3 N.m) en $t = 0,1$ s.
- Se utilizan dos lazos de control.
- El bucle interno sincroniza las señales de las puertas del inversor con las fuerzas electromotrices. El lazo exterior controla la velocidad del motor variando el voltaje del bus de CC.
- El Tiempo máximo de las curvas de Respuesta en el Tiempo en el Simulink es 0.1 Segundos.

Análisis de entradas y salidas:

Tabla 4.1 Comparación de valores medidos del Controlador Difuso y el control Convencional PI a 3,000 RPM

ENTRADA	Comparación de valores					
	Rise time (mS)		Overshoot (%)		Settling Time (Seg.)	
RPM	FUZZY	PI	FUZZY	PI	FUZZY	PI
3,000	14	27	8.70	2.70	0.051	0.050

Fuente. Elaboración propia

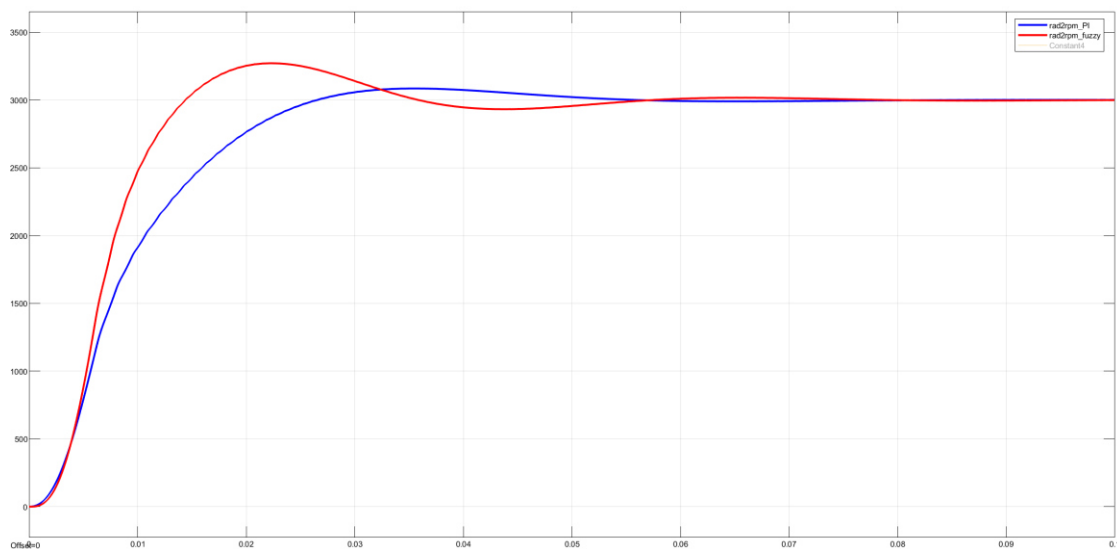


Figura 4.29 Medición de la salida del Sistema de Control de motor BLDC con Controlador PI a 3,000 RPM

Fuente. MatLab 2020a– Simulink (MathWorks, 1999)

Tabla 4.2 Comparación de valores medidos del Controlador Difuso y el Control Convencional PI a 2,500 RPM

ENTRADA	Comparación de valores					
	Rise time (mS)		Overshoot (%)		Settling Time (Seg.)	
RPM	FUZZY	PI	FUZZY	PI	FUZZY	PI
2,500	14	27	7.80	2.14	0.045	0.043

Fuente. Elaboración propia

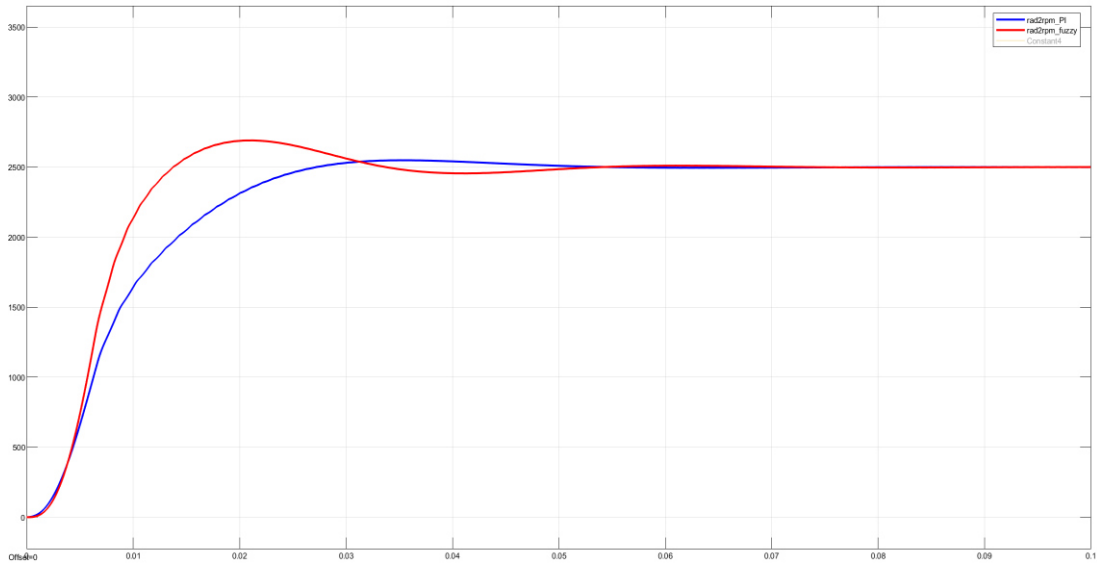


Figura 4.30 Medición de la salida del Sistema de Control de motor BLDC con Controlador PI a 2,500 RPM

Fuente. MatLab 2020a– Simulink (MathWorks, 1999)

Tabla 4.3 Comparación de valores medidos del Controlador Difuso y el Control Convencional PI a 1,000 RPM

ENTRADA	Comparación de valores					
	Rise time (mS)		Overshoot (%)		Settling Time (Seg.)	
RPM	FUZZY	PI	FUZZY	PI	FUZZY	PI
1,000	11	40	6.50	None	0.029	0.030

Fuente. Elaboración propia

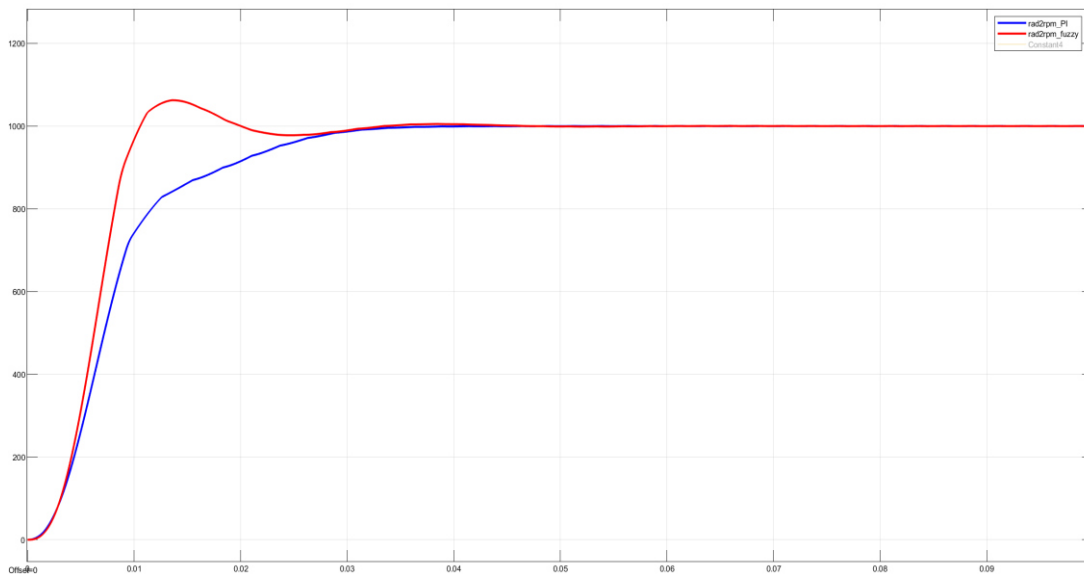


Figura 4.31 Medición de la salida del Sistema de Control de motor BLDC con Controlador PI a 1,000 RPM

Fuente. MatLab 2020a– Simulink (MathWorks, 1999)

Podemos ver el error de estado estacionario del Sistema de Control con el efecto del Controlador Difuso y con el Control PI como puede verse en las Tablas 4.4, 4.5 y 4.6.

Tabla 4.4 Comparación de error estacionario del Controlador Difuso y el Control Convencional PI a 3,000 RPM

Entrada(RPM)	Fuzzy(RPM)	Error(%)	PI(RPM)	Error(%)
3,000	2,811	6.30	2,715	9.50

Fuente. Elaboración propia

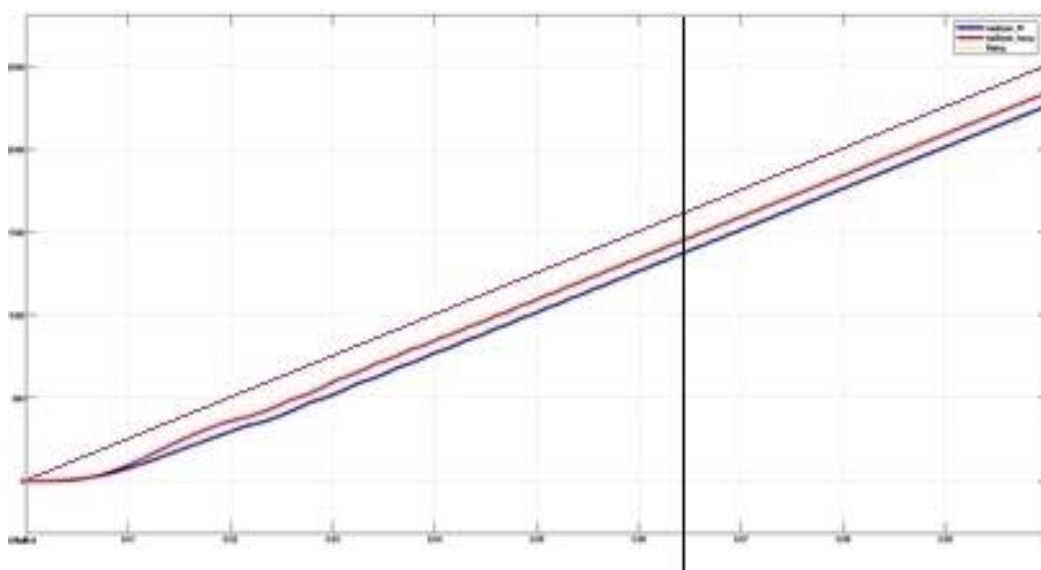


Figura 4.32 Medición de la salida de error estacionario del Sistema de Control de motor BLDC con Controlador PI a 3,000 RPM

Fuente. MatLab 2020a– Simulink (MathWorks, 1999)

Tabla 4.5 Comparación de error estacionario del Controlador Difuso y el Control Convencional PI a 2,500 RPM

Entrada(RPM)	Fuzzy(RPM)	Error(%)	PI(RPM)	Error(%)
2,500	2,345	6.20	2,265	9.40

Fuente. Elaboración propia

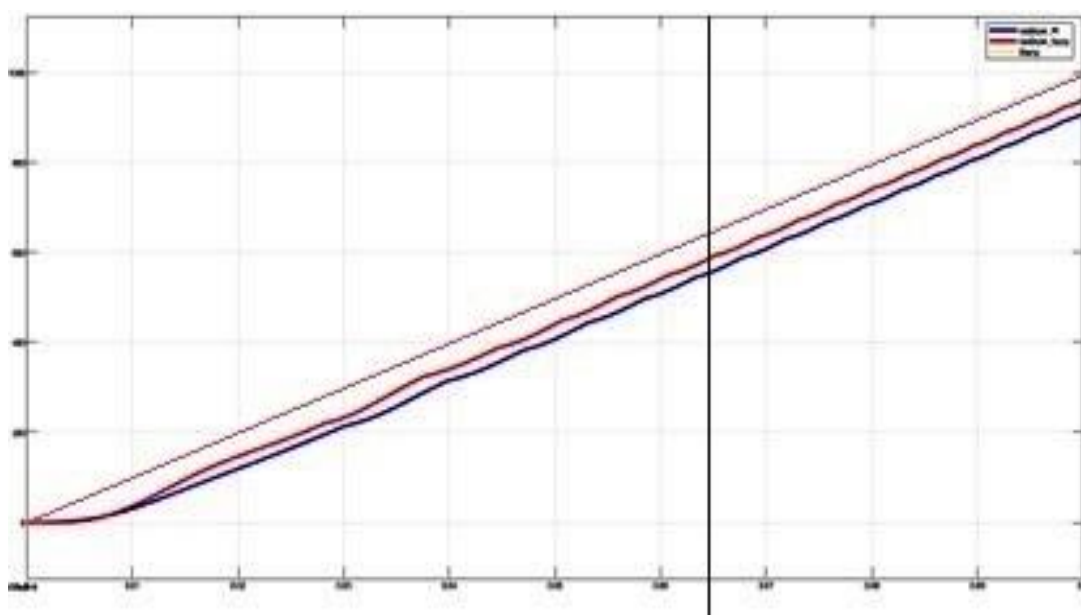


Figura 4.33 Medición de la salida de error estacionario del Sistema de Control de motor BLDC con Controlador PI a 2,500 RPM

Fuente. MatLab 2020a– Simulink (MathWorks, 1999)

Tabla 4.6 Comparación de error estacionario del Controlador Difuso y el Control Convencional PI a 1,000 RPM

Entrada(RPM)	Fuzzy(RPM)	Error(%)	PI (RPM)	Error(%)
1,000	939.3	6.07	908.8	9.12

Fuente. Elaboración propia

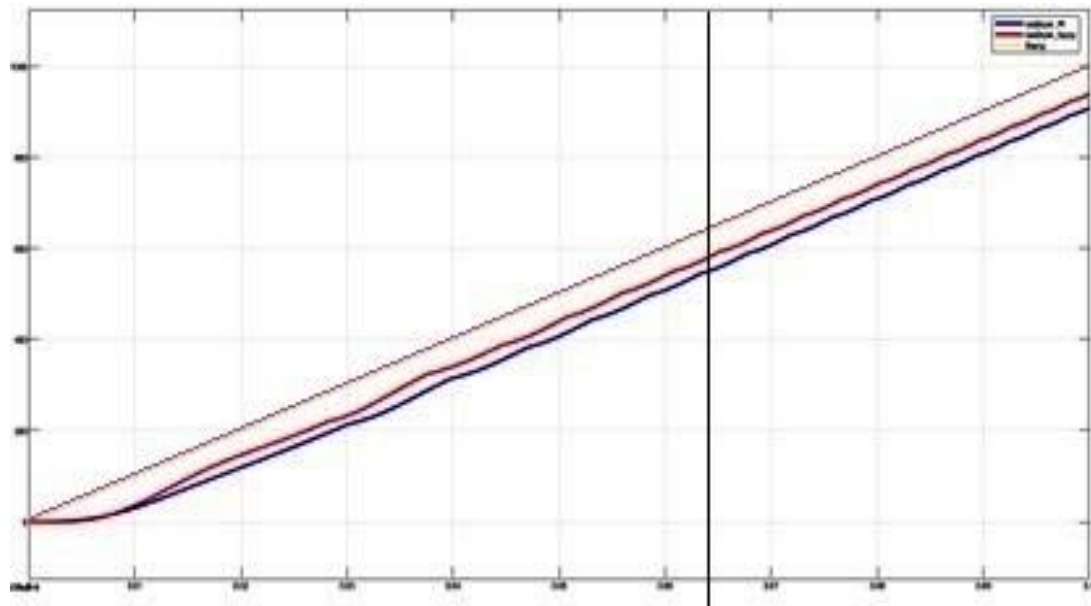


Figura 4.34 Medición de la salida de error estacionario del Sistema de Control de motor BLDC con Controlador PI a 1,000 RPM
Fuente. MatLab 2020a– Simulink (MathWorks, 1999)

De acuerdo a las Tablas 4.4, 4.5 y 4.6 los resultados indican el error estacionario del Control Difuso es menor que el Control Convencional PI

Capítulo 5 RESULTADOS Y DISCUSION

5.1 Análisis, Interpretación y Discusión de Resultados

En primer lugar se realiza el análisis de acuerdo a la respuesta en el tiempo de una señal de RPM en la entrada de un Sistema de Control de Velocidad de un motor BLDC con un Controlador difuso y otro con un Controlador PI (Proporcional - Integrativo) como se observa en los esquemas del Simulink - MatLab 2020.1 (MathWorks, 1999), cuya salida de dicho Sistema de control se muestran en las Figuras 4.29, 4.30, y 4.31 del capítulo anterior y cuyos resultados de medición pueden verse en la tabla 5.1 observándose que el efecto del Controlador Difuso con respecto al Controlador PI.

Tabla 5.1 Comportamiento del Sistema de Control de velocidad de un motor BLDC con respecto a un Controlador Difuso y a un Controlador PI

ENTRADA	Comparación de valores			
	Rise Time (mS)		Overshoot(%)	
	Difuso	PI	Difuso	PI
3,000	14	27	8.70	2.70
2,500	14	27	7.80	2.14
1,000	11	40	6.50	0

Fuente. Elaboración propia

Interpretándose que el efecto de un Controlador difuso en citado Sistema de control en el dominio del tiempo tiene una mejor respuesta en la comparación con el Controlador PI de acuerdo a las lecturas de Rise Time.:

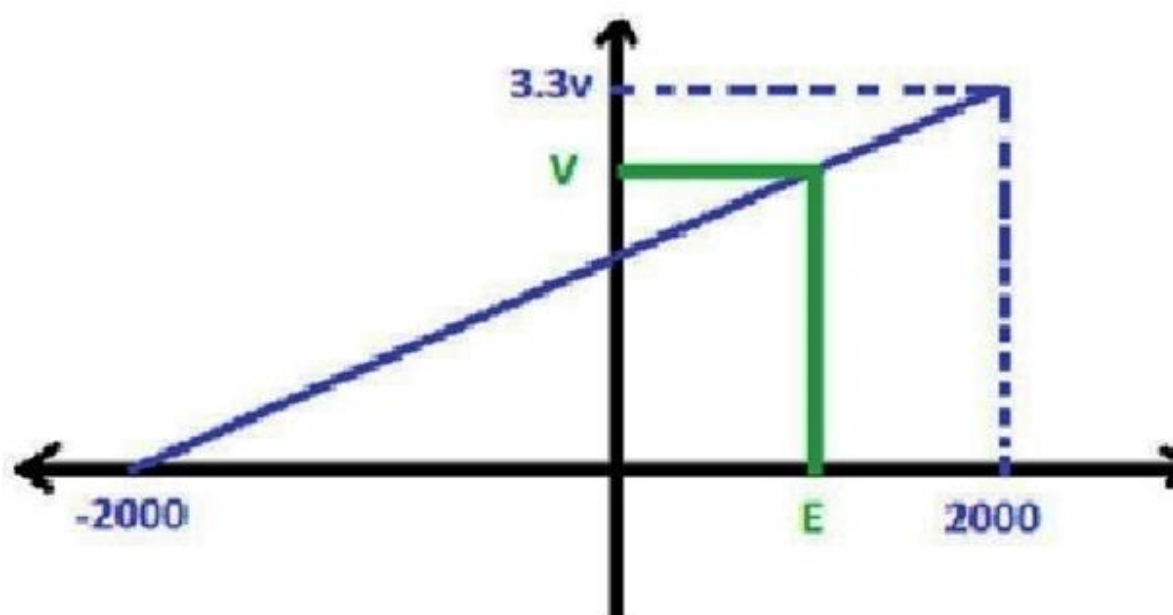
Además tomando el mayor valor de los valores de Sobreimpulso máximo (Overshoot) expresado en porcentaje (%).obtenido de las Tablas 4.1, 4.2 y 4.3 a a la velocidad de trabajo de 2,500 RPM se encuentra en el rango de trabajo tolerable para Sistemas de Control de Velocidad de motores, de acuerdo al artículo científico (Varshney A., 2017).

En segundo lugar el análisis de los resultados de Error estacionario de los diagramas hechos en el Simulink - MatLab 2020.1 (MathWorks, 1999) de las Figuras 4.32, 4.33, 4.34, cuyos resultados están son las Tablas 4.4, 4.5 y 4.6. del capítulo anterior

Se interpreta que el Sistema de control de velocidad de un motor BLDC con el Controlador Difuso tiene un menor error estacionario que con el Controlador PI

En tercer lugar para demostrar que el diseño del Controlador Difuso funciona correctamente en el presente trabajo para un Sistema de Control de Velocidad de un motor BLDC,

- a. Se ha implementado el Controlador Difuso con todas las consideraciones de diseño de la Lógica Difusa en hardware y software en la Plataforma ZYBO Z7, donde esta el dispositivo SoC,
- b. Se ha desarrollado un Sistema de Control Difuso de Velocidad de un motor BLDC en Simulink y con el Tool Box del MatLab 2020.1
- c. Se ha medido con el Scope (Simulink), obteniéndose la 3ra. Columna de la Tabla 5.2, en el que se la usado el método de la interpolación, en donde los RPM del motor se traduce en voltaje de 0 a 3.3V, tal como se muestra.



$$E = 2000 \frac{2}{3.3} \left(v - \frac{3.3}{2} \right)$$

Figura 5.1 Curva de Interpolación.

Fuente: Elaboración propia.

Para construir la Tabla 5.2 se ha realizado los siguientes pasos

1. Para construir la primera columna de la Tabla 5.2 la entrada del Controlador Fuzzy (Zybo Z7) se usó una señal de voltaje de continua (DC) de 0 V a 3.3 V, luego se realizó una interpolación donde, 0 V = - 2000 RPM y 3.3 V = 1998.05 RPM, estos valores están relacionados con los conjuntos difusos del Fuzzy Logic Tool Box del MatLab, (MathWorks, 1999).
2. Las señales de voltaje de entrada aplicadas al Controlador Difuso (Zybo Z7) provocan una salida que es mostrada en la pantalla OLED en el registro Outvoltage como puede verse en el fragmento del programa main.c (ver Anexo C) hecho en lenguaje C en VITIS 2020.1, el valor de este registro Outvoltage se obtiene el valor de voltaje en la salida del PMOD DA2 (DIGILENT, Hoja de Datos del PMOD DA2, 2016) construyéndose la segunda columna de la Tabla 5.2
3. Para la construcción de la tercera columna de la Tabla 5.2 se uso los conjunto difuso de salida del Controlador difuso hecho en el Tool Box del MatLab 2020a proveniente del Conjunto difuso de salida en Fuzzy Logic Toolbox mostrado en la Figura 4.5.
4. En consecuencia puede verse los resultados de las mediciones antes detallado para el Controlador Difuso y para el Controlador convencional PI, en la siguiente Tabla 5.2

Tabla 5.2 Comparación de valores medidos del Controlador Difuso y la Simulación en Simulink – MatLab

RESULTADOS	Entrada		Salida	
	Voltaje (v) (v)	E (RPM)	Zybo Z7 Pmod DA2	Tool Box Fuzzy Logic (salida)
1	0.00	-2000	-16.35	-16.4
2	0.28	-1664.06	-16.29	-16.3
3	0.59	-1280.27	-13	-13
4	0.95	-848.63	-4	-4
5	1.27	-455	-2.64	-2.63
6	2.09	531.4	3.74	3.74
7	2.40	904	4	4
8	2.71	1289.06	13.97	14
9	2.81	1402.34	14.86	14.9
10	3.30	1998.05	15.87	15.9
11	3.30	1999.02	15.87	15.9

Fuente. Elaboración propia

En concordancia a la discusión de los resultados mostrados de la comparación de los efectos del Controlador Difuso y el Controlador PI en un Sistema de Control de Velocidad de un motor BLDC, se muestran los resultados de las mediciones de respuesta en el tiempo, Sobre amortiguamiento y Error estacionario

Agregándose que se ha comprobado (Tabla 5.2) que el diseño implementado del Controlador Difuso del presente trabajo esta desarrollado basado en la Lógica Difusa (Zadeh, 1973).

De acuerdo a la discusión basado en los Resultados mostrados indican que la mejor solución es el Controlador Difuso.

5.2 Pruebas de Hipótesis.

Prueba de la Hipótesis General (HG)

Los resultados del análisis en el tiempo de las pruebas mostradas en la Tabla 5.1 sobre la comparación del Controlador Difuso y un Controlador Convencional bajo las mismas condiciones, dentro de un Sistema de control de Velocidad de un motor BLDC, para lo cual se aplica un señal escalón (RPM) en las entradas de dichos Sistemas de control donde el efecto de sus Controladores en sus salidas demuestran el mejor desempeño del Controlador Difuso en el dominio del tiempo:

Mostrando una buena solución difusa de interacción Hardware/Software, Por lo tanto, se acepta la Hipótesis General

Prueba de la Hipótesis Especifica 1 (HE1)

El uso de una plataforma ZYBO Z7 usando un dispositivo de la familia Xilinx Zynq-7010 con una señal de reloj de 33.3333 MHz permitiendo que el procesador funcione a una frecuencia máxima de 667MHz y que el controlador de memoria DDR3 funcione a una frecuencia de reloj máxima de 533MHz, logrando altas velocidades del Procesador en las ejecuciones de las instrucciones por encima de los Microcontroladores usados para este fin, en nuestro diseño lo hemos configurado a 100 MHz ajustándolo a la velocidad en la transmisión de datos de nuestras interfases seriales, mostrando su versatilidad en la selección de sr velocidad de procesamiento de la información dentro de su amplio espectro de Clock.

Por lo tanto, se acepta la Hipótesis Especifica 1

Prueba de la Hipótesis Especifica 2 (HE2)

El uso de una plataforma ZYBO Z7 usando un dispositivo de la familia Xilinx Zynq-7010 tiene gran flexibilidad, esto quiere decir, Respuesta rápida y eficiente a cambios futuros en las modificaciones de hardware/software y protocolos que sucedan luego que el sistema se ha implementado y esta funcionando en el campo.

Que el diseño del hardware como son las interfaces de entrada se han realizado en el software VIVADO, y el diseño del software se han realizado en VITIS,

Por lo tanto, se acepta la Hipótesis Especifica 2

Prueba de la Hipótesis Especifica 3 (HE3)

La plataforma ZYBO Z7 que usa un dispositivo de la familia Xilinx Zynq-7010, es un desarrollo basado en técnicas de codiseño hardware/software (HW/SW), que tiene como una de sus características principales la rápida actualización de la versión del Controlador Difuso, no solo en software si no también en el hardware. Por lo tanto, se acepta la Hipótesis Especifica 3

5.3 Presentación de Resultados

Tabla 5.3 Resultado de prueba de hipótesis

Nº	HIPOTESIS	INDICADOR	PRUEBA HIPOTESIS	RESULTADO
1	El Diseño e Implementación de un Controlador de lógica difusa con tecnología FPGA XILINX 7 Y ARM CORTEX A9 aplicado a un Sistema de control de velocidad de un motor de corriente continua BLDC, brindará un mejor desempeño que un Controlador convencional	X1; El resultado de las pruebas de comparación entre el Controlador Difuso y el Controlador Convencional PI a través de sus modelos matemáticos elaborados en MatLab 2020 ^a X2; Mediciones realizadas en el Controlador Difuso implementado en la plataforma ZYBO Z7 comparado con su modelo hecho en Simulink-MatLab	Si	Se prueba que el comportamiento del Controlador Difuso es mejor Que el Controlador convencional
	Hipótesis Especifica 1 (HE1) El Controlador por Lógica Difusa con Tecnología	Y1 El Controlador de Lógica Difusa con		Una mayor velocidad del Procesador

	FPGA XILINX 7 Y ARM CORTEX A9, es un diseño de interacción hardware/software que tiene mejores características y versatilidad que el controlador convencional	Tecnología FPGA XILINX 7 Y ARM CORTEX A9, se ha trabajado con una frecuencia de sistema de 100 MHz, logrando velocidades de ejecución superiores a los MCU tradicionales	Si	logra tiempos de ejecuciones de instrucción muy alta, mejorando su característica de velocidad en su funcionamiento
	Hipótesis Especifica 2 (HE2) El Desarrollo del software del Controlador difuso logrará un diseño flexible y eficiente que un controlador convencional..	Y2: Debido a su origen la plataforma ZYBO Z7 es un Sistema de Desarrollo de interacción Hardware/Software permite un diseño flexible	Si	La flexibilidad permite una depuración en Software y de ser necesario cambiar también el Hardware a nivel de FPGA.
	Hipótesis Especifica 3 (HE3) El Controlador por Lógica Difusa con tecnología FPGA XILINX 7 y ARM	Siendo el Controlador Difuso un codiseño Hardware/Software		permite una rápida actualización de la versión

	CORTEX A9 para un Sistema de Control de velocidad de un motor de corriente continua BLDC, es de actualización muy rápida en comparación con los Controladores convencionales..	Viene incluidas las características propias de fácil actualización	Si	del Controlador Difuso
--	--	--	----	------------------------

Fuente. Elaboración propia



Figura 5.2 Laboratorio de pruebas del Controlador Difuso en la plataforma
ZYBO Z7-10

Fuente. Elaboración propia

Capítulo 6 IMPACTOS

6.1 Propuesta de la Solución del Problema SISTEMA DE CONTROL DIFUSO

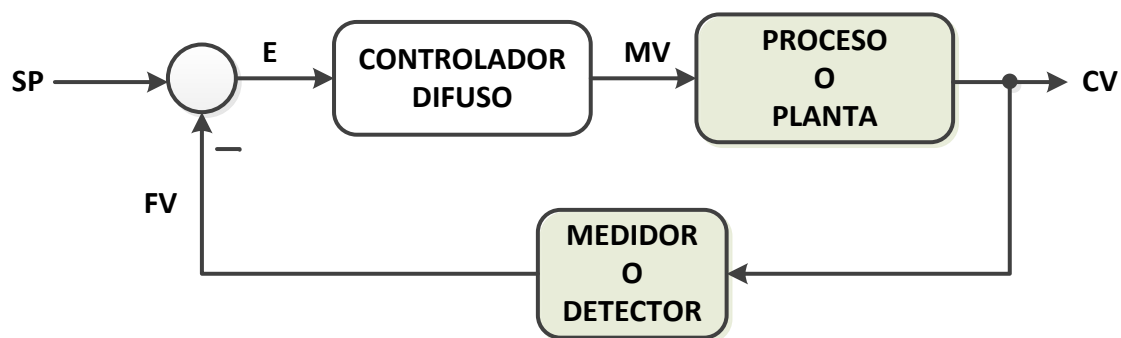


Figura 6.1 Sistema de Control Difuso

Fuente. Elaboración propia

Siendo:

SP = Valor deseado (Set Point)

E = Error

FV = medición de la Variable CV (Feedback Variable)

MV = Variable manipulada (Manipulated Variable)

CV = Variable controlada (Controlled Variable)

La razón del Controlador Difuso es la confiabilidad del funcionamiento del Sistema de Control como en la Figura 6.1 que representa a cualquier Sistema de Control, ya sea de Temperatura, de Presión, de Caudal, de Capacidad, de Flujo, de Peso, etc. lo que significa que es aplicable a un campo de acción muy grande

Además, el Controlador difuso permite múltiples entradas y salidas con alta inmunidad al ruido.

6.2 Costos de la Implementación de la Propuesta.

Se presenta en la siguiente tabla los gastos del trabajo de Tesis.

Tabla 6.1 Costo de la implementación

UNIDADES	COMPONENTES	PRECIO UNITARIO	PRECIO TOTAL
1	Plataforma de Desarrollo ZYBO Z7	\$ 300.00	\$ 300.00
1	Pmod AD2	\$ 20.00	\$ 20.00
1	Pmod DA2	\$ 20.00	\$ 20.00
1	Pimod Display OLED	\$ 10.00	\$ 10.00
5	Potenciometros lineales de 10K	\$ 2.00	\$ 10.00
10	Bornas	\$ 1.00	\$ 10.00
1	Otros (Resistencias, Leds, cables Interruptores, pulsadores etc.)	\$ 30.00	\$ 30.00
1	Mano de Obra	\$ 40.00	\$ 40.00
TOTAL (EN DOLARES AMERICANOS)			\$ 440.00

Fuente. Elaboración propia

6.3 Beneficios que aporta la Propuesta.

Tabla 6.2 Beneficio de la Propuesta

N°	ANALISIS DEL DISEÑO	BENEFICIO
1	El Control Difuso es muy versátil y flexible	Maneja múltiples entradas y salidas. Esto ocurre en el Control Convencional.
2	El Control Difuso es un diseño basado en la incertidumbre.	No requiere identificar el sistema, ni necesita aproximarse el modelo en funcionamiento y es altamente inmune al ruido
3	El Control Difuso no requiere etapas de linealización	No necesita linealizarlo y está hecho para reemplazar Controles basados en modelos matemáticos muy complejos

Fuente. Elaboración propia

CONCLUSIONES

Se ha logrado el objetivo del Desarrollo de un Controlador Difuso usando la tecnología SoC para un Sistema de Control de Velocidad de un motor BLDC.

Se comprueba que de acuerdo a los resultados obtenidos, que un Controlador Difuso es una buena solución, .

Se concluye que el uso del Controlador Difuso bajo en enfoque SoC tiene un gran velocidad en la ejecución agregándole a ello su versatilidad y flexibilidad.

Se concluye que el diseño y la implementación de un Controlador Difuso pueda probarse su funcionamiento un modelo matemático hecho en MatLab.

Esta trabajo de Tesis presenta una metodología de diseño en inteligencia artificial (Artificial Intelligence) como lo es la Lógica Difusa.

Cabe indicar que este Control Difuso con pequeños ajustes puede ser una solución basada en Inteligencia Artificial a usarse en cualquier otra aplicación

La totalidad de esta Tesis tiene fines de manejo de esta nueva tecnología usándose herramientas de fácil acceso para fomentar su uso en la investigación en la Inteligencia Artificial en sus diferentes aplicaciones.

Visto el resultado positivo de la implementación en una plataforma basada en el dispositivo AP SoC, El presente trabajo de tesis está listo para su fabricación.

RECOMENDACIONES

La importancia del uso de los dispositivos SoC en las plataformas como el ZYBO Z7 con sus herramientas de software (VIVADO y VITIS) logrando en el diseño flexibilidad, velocidad de procesamiento y potencia, requisitos para usarlo en una amplia gama de aplicaciones, desde interfaces inteligentes hasta visión artificial e inteligencia artificial, que es la tendencia de hoy y el futuro en la tecnología.

Se sugiere diseñar con el dispositivo SoC, debido a que en su entorno se puede usar software libre como son los sistemas operativos desde Linux hasta FreeRTOS.

Se sugiere diseñar en el área de la Inteligencia Artificial como lo es la Lógica Difusa que es la tendencia en el futuro de la tecnología, debido a su grandes características, en la implementación de las aplicaciones como se presenta en este trabajo de Tesis en un diseño e implementación de un Controlador Difuso para un Sistema de Control, al que podría fácilmente aplicarse múltiples entradas y salidas, agregándole la alta inmunidad al ruido y la facilidad de ser actualizable como sistema, algo que no ocurre normalmente con los Sistemas de Control tradicionales.

En este presente Trabajo de Tesis se ha logrado resultados satisfactorios, usándose el enfoque SoC con el dispositivo de la familia Zynq 7000S como es el Zynq xc7z010-1clg400c que cuenta con un procesador ARM Cortex-A9 unido con el FPGA – XILINX con tecnología de fabricación basada en Artix™ 7 de 28nm, con transceptores de 6,25 Gb/s, se recomienda usarse en aplicaciones como la IoT (Internet Of Things) o en el manejo de la visión y/o aplicaciones en el sistema de la telefonía celular, etc.

REFERENCIAS

- DIGILENT. (22 de Octubre de 2022). *Digilent/vivado-library*. (D. T. Inc, Ed.) Recuperado el 27 de Noviembre de 2022, de [www.digilentinc.com: https://github.com/Digilent/vivado-library/tree/master/ip/Pmods](https://github.com/Digilent/vivado-library/tree/master/ip/Pmods)
- A. Varshney, D. G. (2017). *Journal of Electrical Systems and Information Technology*. Obtenido de Speed response of brushless DC motor using fuzzy PID controller under varying load condition: <https://www.sciencedirect.com/science/article/pii/S2314717217300077>
- Aridhi, E., Popescu, D., & Mami, A. (2021). FPGA based co-design of a speed Fuzzy Logic Controlles applied to an Autonomous Car. *revista International Journal of Reconfigurable and Embedded Systems*, 18. doi:10.11591/ijres.v10.i3.pp195-211
- Brown, W. (2002). *AN857 Brushless DC Motor Control Made Easy*. (M. T. Incorporated, Ed.) Recuperado el 27 de Mayo de 2020, de Microchip Technology Inc.: <https://ww1.microchip.com/downloads/en/AppNotes/00857B.pdf>
- Crockett, L. H., Elliot , R. A., Enderwitz, M. A., & Stewart, R. W. (2014). *The Zynq Book Embedded Processing with the ARM® Cortex®-A9 on the Xilinx® Zynq®-7000*. Glasgow, Scotland, United Kingdom: Strathclyde Academic Media. Recuperado el 9 de Septiembre de 2022, de https://is.muni.cz/el/1433/jaro2015/PV191/um/The_Zynq_Book_ebook.pdf
- DIGILENT. (24 de Mayo de 2016). *Hoja de Datos del PMOD AD2*. Recuperado el 26 de Agosto de 2022, de [reference.digilentinc: https://reference.digilentinc.com/reference/pmod/pmodad2/start](https://reference.digilentinc.com/reference/pmod/pmodad2/start)
- DIGILENT. (24 de Mayo de 2016). *Hoja de Datos del PMOD DA2*. Recuperado el 27 de Agosto de 2022, de [reference.digilentinc: https://reference.digilentinc.com/reference/pmod/pmodda2/start](https://reference.digilentinc.com/reference/pmod/pmodda2/start)

- DIGILENT. (24 de Mayo de 2016). *Hoja de Datos del PMOD OLED*. Recuperado el 27 de Agosto de 2022, de reference.digilentinc.com/reference/pmod/pmodoled/start
- DIGILENT. (21 de Febrero de 2018). *Manual de ZYBO – Z7*. Recuperado el 27 de Agosto de 2022, de [reference.digilentinc.com/reference/_media/reference/programmable-logic/zybo-z7/zybo-z7_rm.pdf](https://digilent.com/reference/_media/reference/programmable-logic/zybo-z7/zybo-z7_rm.pdf)
- DIGILENT. (22 de Octubre de 2022). *Digilent/vivado-library*. (D. T. Inc, Ed.) Recuperado el 27 de Noviembre de 2022, de [www.digilentinc.com: https://github.com/Digilent/vivado-library/tree/master/ip/Pmods](https://github.com/Digilent/vivado-library/tree/master/ip/Pmods)
- Gómez, J. L. (Agosto de 2015). Diseño e implementación en FPGA de un Sistema de Control para Bipestador Movil. Mexico, La Paz, Baja California: Instituto Tecnológico de la Paz. Obtenido de <https://docplayer.es/9196299-Diseno-e-implementacion-en-fpga-de-un-sistema-de-control-para-bipedestador-movil.html>
- Lomas , V. M. (Octubre de 2016). Aprendizaje y reconocimiento invariante de objetos de ensamble con Robots empleando Redes Neuronales Artificiales pplementadas con FPGA. Copilco, Mexico, Mexico: Universidad Autónoma de México. Recuperado el 2 de Diciembre de 2021, de <http://132.248.52.100:8080/xmlui/handle/132.248.52.100/11015>
- MathWorks. (1 de Enero de 1999). MATLAB R2020A (Simulink + Fuzzy Logic Toolbox), Mathematical Software. Natick, Massachusetts, Estados Unidos. Recuperado el 9 de Septiembre de 2021, de <https://www.mathworks.com/products/matlab.html>
- MathWorks, I. (2024). *MathWorks*. Obtenido de Brushless DC Motor Fed by Six-Step Inverter: [https://es.mathworks.com/help/sps/ug/brushless-dc-motor-fed-by-six-step-inverter.html#d124e55047%20\(MathWorks,%201999\)](https://es.mathworks.com/help/sps/ug/brushless-dc-motor-fed-by-six-step-inverter.html#d124e55047%20(MathWorks,%201999))
- Maxfield, C. (2020). Fundamentals of FPGAs. En C. Maxfield, *Part 4: Getting Started with Xilinx FPGAs*. Minnesota: Digi-Key's North American Editors. Recuperado el 9 de Septiembre de 2022, de <https://www.digikey.com/en/articles/fundamentals-of-fpgas-part-4-getting-started-with-xilinx-fpgas>
- Olivier Tremblay, L.-A. D. (2024). *Brushless DC Motor Fed by Six-Step Inverter*. Obtenido de <https://www.mathworks.com/help/sps/ug/brushless-dc-motor-fed-by-six-step-inverter.html#d124e55047>

- Passino, K. M., & Yurkovich, S. (1997). *Fuzzy Control*. Columbus, Ohio, Estados Unidos: Addison - wesley. Recuperado el 2 de Diciembre de 2020, de <http://www4.hcmut.edu.vn/~hthoang/dktm/Fuzzy%20Control.pdf>
- Rodríguez, J. D. (2015). Control basado en Logica Borrosa en DSPIC33 para un motor BRUSHLESS. Carabobo, Venezuela: Instituto Universitario de Tecnología de Valencia. Recuperado el 2 de Diciembre de 2021, de http://www.edutecne.utn.edu.ar/microcontrol_congr/industria/Control-logica_borrosa-dsPIC33.pdf
- Varshney A., G. D. (25 de Enero de 2017). *www.sciencedirect.com*. Recuperado el 01 de Agosto de 2023, de [www.sciencedirect.com](https://www.sciencedirect.com/science/article/pii/S2314717217300077): <https://www.sciencedirect.com/science/article/pii/S2314717217300077>
- Zadeh, L. A. (1973). Outline of a new approach to the analysis of complex system. *IEEE Transaction on System Man and Cybernetics*, 28– 44. doi:10.1109/TSMC.1973.5408575

ANEXO A. DIAGRAMA DE ICHICAWA

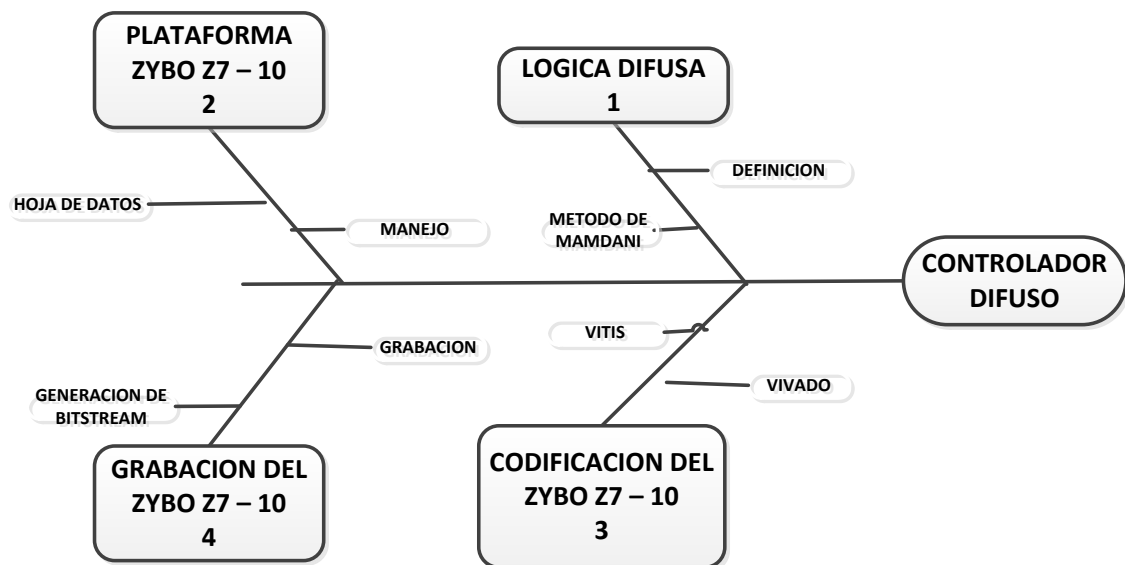


Figura A.1 Diagrama de Ichicawa del Controlador Difuso

Fuente. Elaboración propia

**ANEXO B. PROGRAMA DEL CONTROLADOR DIFUSO
(VITIS)**

```

//fuzzy.h
#include "xil_types.h"
#include "xstatus.h"
#include <math.h>
#define INFIMO_OUTPUT -25 //rango inferior del dominio de salida
#define MAXIMO_OUTPUT 25 //rango superior del dominio de salida
#define LOOP_STEP 0.5 //paso del diferencial en la integracion par ael
calculu del centroide (dejar en 0.5 para no relentizar)

//////////DEESCRIPCION DE
FUNCIONES//////////
float Triangulo(float ang, float a, float b, float c);
float Trapecio(float ang, float a, float b, float c, float d);
float Gaussiana(float x, float o, float u);
float And(float a, float b);
float Or(float a, float b);
void alturaMayorMembresia(float memDeReglas[], char t[], char p[]);
float CalculoCentroide1(float GN, float SN, float Z, float SP, float GP);
//DECLARACION DE CONJUNTOS DIFUSOS
float memE1(float e, char s, char r);
float memS1(float ss, char s, char r);

//DECLARACION DE VARIABLES
float valorMembRegla[11], GN = 0, SN = 0, Z = 0, SP = 0, GP = 0;
char tipo[5] = {}, tamano[5] = {};

//*****
*****
//*****DEFINIR LOS CONJUNTOS
DIFUSOS*****
float memE1(float e, char s, char r) {
    float menBre = 0;
    if ((s == 'G') & (r == 'N')) {
        menBre = Trapecio(e, -2000, -2000, -1735, -1000);
    } else {
        if ((s == 'S') & (r == 'N')) {
            menBre = Triangulo(e, -1560, -1050, -300);
        } else {
            if ((s == 'Z') & (r == 'E')) {
                menBre = Triangulo(e, -550, 0, 550);
            } else {
                if ((s == 'S') & (r == 'P')) {
                    menBre = Triangulo(e, 100, 900, 1600);
                } else {
                    if ((s == 'G') & (r == 'P')) {
                        menBre = Trapecio(e, 1000, 1500, 2000, 2000);
                    }
                }
            }
        }
    }
    return menBre;
}

```

```

float memS1(float ss, char s, char r) {
    float menBre = 0;
    if ((s == 'G') & (r == 'N')) {
        menBre = Trapecio(ss, -25, -25, -9, -6);
    } else {
        if ((s == 'S') & (r == 'N')) {
            menBre = Triangulo(ss, -7, -4, -1);
        } else {
            if ((s == 'Z') & (r == 'E')) {
                menBre = Triangulo(ss, -3, 0, 2);
            } else {
                if ((s == 'S') & (r == 'P')) {
                    menBre = Triangulo(ss, 1, 4, 7);
                } else {
                    if ((s == 'G') & (r == 'P')) {
                        menBre = Trapecio(ss, 6, 7, 25, 25);
                    }
                }
            }
        }
    }
    return menBre;
}

//*****
//*****DESARROLLO DE LAS FUNCIONES DEL ALGORITMO*****

//FUNCION DE CALCULO DE LA MAYOR MEMBRESIA
//calcula la mayor altura en cada diferencial que se calcula
void alturaMayorMembresia(float memDeReglas[], char t[], char p[]) {
    //memDeReglas[]->memb tipo ,
    int tt; //numero de reglas
    for (tt = 0; tt < 5; tt++) {

        if ((t[tt] == 'G') & (p[tt] == 'N')) {
            GN = Or(GN, memDeReglas[tt]);
        }

        else if ((t[tt] == 'S') & (p[tt] == 'N')) {
            SN = Or(SN, memDeReglas[tt]);
        }

        else if ((t[tt] == 'Z') & (p[tt] == 'E')) {
            Z = Or(Z, memDeReglas[tt]);
        }

        else if ((t[tt] == 'S') & (p[tt] == 'P')) {
            SP = Or(SP, memDeReglas[tt]);
        }

        else if ((t[tt] == 'G') & (p[tt] == 'P')) {
            GP = Or(GP, memDeReglas[tt]);
        }

    }
}

//FUNCION DE CALCULO DEL CENTROIDE

```

```

float CalculoCentroide1(float GN, float SN, float Z, float SP, float GP) {
    float As = 0, Xs = 0, X = 0, A = 0;
    float h1, h2, h3, h4, h5;
    //REDEFINO DOMINIO DE SALIDA
    for(float i=INFIMO_OUTPUT; i<=MAXIMO_OUTPUT; i = i + LOOP_STEP) {
        h1 = And(GN, memS1(i, 'G', 'N'));
        h2 = And(SN, memS1(i, 'S', 'N'));
        h3 = And(Z, memS1(i, 'Z', 'E'));
        h4 = And(SP, memS1(i, 'S', 'P'));
        h5 = And(GP, memS1(i, 'G', 'P'));
        A = (float) 0.1 * Or(h1, Or(h2, Or(h3, Or(h4, h5))));
        As = (float) As + A;
        Xs = (float) Xs + A*i;
    }
    if (As == 0) {
        X = 0;
    } else {
        X = (float) Xs / As;
    }
    return X;
}

//////////////////////////////////////////COMPARADORES//////////////////////////////////////////
float And(float a, float b) {
    float and;
    if (a <= b) {
        and = a;
    } else {
        and = b;
    }
    return and;
}

float Or(float a, float b) {
    float or;
    if (a <= b) {
        or = b;
    } else {
        or = a;
    }
    return or;
}

//////////////////////////////////////////FUNCIONES MATEMATICAS//////////////////////////////////////////
float Triangulo(float ang, float a, float b, float c) {
    float menBre = 0;
    if ((ang >=a)&(ang <= b)) {
        menBre = (float) (ang - a) / (b - a);
    } else {
        if ((ang > b)&(ang < c)) {
            menBre = (float) (c - ang) / (c - b);
        } else {
            menBre = 0;
        }
    }
    return menBre;
}

float Trapecio(float ang, float a, float b, float c, float d) {
    float menBre = 0;

```

```

if ((ang >=a)&(ang < b)) {
    menBre = (float) (ang - a) / (b - a);
} else {
    if ((ang >= b)&(ang <= c)) {
        menBre = 1;
    } else {
        if ((ang > c)&(ang < d)) {
            menBre = (float) (d - ang) / (d - c);
        } else {
            menBre = 0;
        }
    }
}
return menBre;
}

```

```

float Gaussiana(float x, float o, float u) { //0: varianza      u: media
    float menBre = 0;
    //por ver despues

    //menBre = (float) expf(-0.5 * (float) powf((x - u)/o, 2));

    return menBre;
}

```

```

//main.c

/* ----- */
/*           Include File Definitions           */
/* ----- */

#include "sleep.h"
#include "xil_cache.h"
#include "xil_printf.h"
#include "xparameters.h"

#include "PmodOLED.h"
#include "PmodAD2.h"
#include "fuzzy.h"
#include "xspi.h"           /* SPI device driver */
#include "xspi_1.h"
#include <stdio.h>
// #include "xil_sprintf.h"

/* ----- */
/*           Global Variables                   */
/* ----- */

PmodAD2 DeviceAD2;
PmodOLED DeviceOled;
static XSpi SpiInstance;

/* ----- */
/*           Forward Declarations               */
/* ----- */
void InitializeAD2();
double RunAD2();
void CleanupAD2();

void InitializeOled();
void RunOled();
void CleanupOled();

int InitializeSpi(XSpi *SpiInstancePtr, u16 SpiDeviceId);
void RunSpi(XSpi *SpiInstancePtr, u16 SpiDeviceId, u16 data);

float InitializeFuzzy(float input1);

void EnableCaches();
void DisableCaches();

// #define SPI_DEVICE_ID          XPAR_SPI_0_DEVICE_ID
#define SPI_DEVICE_ID 0x41E00000
// This is the size of the buffer to be transmitted/received in this
// example.
#define BUFFER_SIZE 2

/* The following data type is used to send and receive data on the SPI
 * interface.
 */
typedef u8 DataBuffer[BUFFER_SIZE];

```

```

volatile int TransferInProgress;

// To change between PmodOLED and OnBoardOLED is to change Orientation
const u8 orientation = 0x0; // Set up for Normal PmodOLED(false) vs normal
                             // Onboard OLED(true)
const u8 invert = 0x0; // true = whitebackground/black letters
                       // false = black background /white letters

/* ----- */
/*                               Function Definitions                               */
/* ----- */

/*
 * The following variables are used to read and write to the Spi device,
 they
 * are global to avoid having large buffers on the stack.
 */
u8 ReadBuffer[BUFFER_SIZE];
u8 WriteBuffer[BUFFER_SIZE];
char strE1[20],strS1[15];
float InputError,OutVoltage;
int Status;

int main() {
    InitializeAD2();
    InitializeOled();
    Status=InitializeSpi(&SpiInstance, SPI_DEVICE_ID);
    xil_printf("-----inicio todo----- \n\r");

    while(1){

        InputError=RunAD2();
        OutVoltage=InitializeFuzzy(InputError);

        sprintf(strE1,"E: %.2f",InputError);
        sprintf(strS1,"S: %.2f",OutVoltage);
        RunOled();
        RunSpi(&SpiInstance, SPI_DEVICE_ID,(OutVoltage+25)*81.9);
        xil_printf("\n\rdata fuzzyr\n\r");
        //xil_sprintf(strE1);
        xil_printf( "strE1: %s \n\r", strE1 );
        xil_printf( "strS1: %s \n\r", strS1 );
        xil_printf("-----\n\r");

    }
    return 0;
}

void InitializeOled() {
    EnableCaches();
    OLED_Begin(&DeviceOled, XPAR_PMODOLED_0_AXI_LITE_GPIO_BASEADDR,
               XPAR_PMODOLED_0_AXI_LITE_SPI_BASEADDR, orientation, invert);
    u8 *pat;

```

```

// Choosing Fill pattern 0
pat = OLED_GetStdPattern(0);
OLED_SetFillPattern(&DeviceOled, pat);
// Turn automatic updating off
OLED_SetCharUpdate(&DeviceOled, 0);
}

void RunOled(){

    for (int irow = 0; irow < OledRowMax; irow++) {
        OLED_ClearBuffer(&DeviceOled);
        OLED_SetCursor(&DeviceOled, 0, 0);
        //OLED_PutString(&DeviceOled, "canbiar");
        OLED_PutString(&DeviceOled, strE1);

        OLED_SetCursor(&DeviceOled, 0, 2);
        //OLED_PutString(&DeviceOled, "codigo");
        OLED_PutString(&DeviceOled, strS1);

        OLED_Update(&DeviceOled);
        usleep(2000);
    }
}

void CleanupOled() {
    OLED_End(&DeviceOled);
    DisableCaches();
}

int InitializeSpi(XSpi *SpiInstancePtr, u16 SpiDeviceId){
    int Status;
    XSpi_Config *ConfigPtr; /* Pointer to Configuration data */
    /*
     * Initialize the SPI driver so that it is ready to use.
     */
    ConfigPtr = XSpi_LookupConfig(SpiDeviceId);
    if (ConfigPtr == NULL) {
        return XST_DEVICE_NOT_FOUND;
    }
    Status = XSpi_CfgInitialize(SpiInstancePtr, ConfigPtr,
                               ConfigPtr->BaseAddress);
    if (Status != XST_SUCCESS) {
        return XST_FAILURE;
    }
    /*
     * Set the SPI device as a master and in manual slave select mode
such
     * that the slave select signal does not toggle for every byte of a
     * transfer, this must be done before the slave select is set
     */
    Status = XSpi_SetOptions(SpiInstancePtr, XSP_MASTER_OPTION|
                               XSP_MANUAL_SSELECT_OPTION|
                               XSP_CLK_ACTIVE_LOW_OPTION);

    if (Status != XST_SUCCESS) {
        return XST_FAILURE;
    }
    /*

```

```

    * Select the slave on the SPI bus, the EEPROM device so that it can
be
    * read and written using the SPI bus
    */
    Status = XSpi_SetSlaveSelect(SpiInstancePtr, 0x01);
    if (Status != XST_SUCCESS) {
        return XST_FAILURE;
    }
    /*
    * Start the SPI driver so that interrupts and the device are
enabled
    */
    XSpi_Start(SpiInstancePtr);
    XSpi_IntrGlobalDisable(SpiInstancePtr);

    return XST_SUCCESS;
}

```

```

void RunSpi(XSpi *SpiInstancePtr, u16 SpiDeviceId, u16 data){
    u8 temp=0, temp1=0;
    //para el da2 4configuracion +12data
    temp=(data>>8)&0x0F;//de 12 dejo los 4MSB y aseguro que el resto
sea 0
    temp1=0x00;//0xconfiguracion 0 0x00
    temp|=temp1;
    temp1=data;//dejo los 8 LSB

    xil_printf("\n dAENVIO : %d \n",data);

    /*print("-----");
    printf("  entrada: %.02f V\n",entrada);
    printf("\n primera parte %d \n",temp);
    printf("\n segunda parte %d \n",temp1);
    printf("\n buf1 %d \n",WriteBuffer[0]);
    printf("\n buf2 %d \n",WriteBuffer[1]);
    print("-----");
    */

    u8 WriteBuffer[BUFFER_SIZE]={temp,temp1};

    //u8 WriteBuffer[BUFFER_SIZE]={0x07,0xD0};
    u8 WriteBuffer2[BUFFER_SIZE]={0x00,0x00};

    XSpi_Start(SpiInstancePtr);
    //TransferInProgress = TRUE;
    print("-envio-");
    XSpi_Transfer(SpiInstancePtr, WriteBuffer, NULL, 2);
    print("-espero-");
    while (TransferInProgress);
    print("-finaliza-");
    //usleep(1000000);
}

```

```

float InitializeFuzzy(float input1){
    float salida=0;

```

```

//inicializa los conjuntos difusos de entrada
GN = 0, SN = 0, Z = 0, SP = 0, GP = 0;

//aplicación de las Estrategias de control

    valorMembRegla[0] = And(memE1(input1, 'G', 'N'),1);tipo[0] =
'G';tamanio[0] = 'N';
    valorMembRegla[1] = And(memE1(input1, 'S', 'N'),1);tipo[1] =
'S';tamanio[1] = 'N';
    valorMembRegla[2] = And(memE1(input1, 'Z', 'E'),1);tipo[2] =
'Z';tamanio[2] = 'E';
    valorMembRegla[3] = And(memE1(input1, 'S', 'P'),1);tipo[3] =
'S';tamanio[3] = 'P';
    valorMembRegla[4] = And(memE1(input1, 'G', 'P'),1);tipo[4] =
'G';tamanio[4] = 'P';
    //obtención de la salida aplicando el calculo del Centroide (metolodologia
    //de Mamdani)
    alturaMayorMembresia(valorMembRegla, tipo, tamanio);
    salida = CalculoCentroide1(GN, SN, Z, SP, GP);
    return salida;
}

void InitializeAD2() {
    EnableCaches();
    AD2_begin(&DeviceAD2, XPAR_PMODAD2_0_AXI_LITE_IIC_BASEADDR,
AD2_IIC_ADDR);
    // Turn on all channels
    AD2_WriteConfig(&DeviceAD2, AD2_CONFIG_CH0 );
    //AD2_WriteConfig(&DeviceAD2, AD2_DEFAULT_CONFIG);

}

double RunAD2() {
    u16 conv;
    double voltage,entrada;

    AD2_ReadConv(&DeviceAD2, &conv);
    // Scale captured data such that 0x000:0xFFF => 0.0:3.3
    voltage = (double) (conv & AD2_DATA_MASK) * 3.3 / (AD2_DATA_MASK + 1);
    usleep(10000);

    //escalamiento
    entrada=(float)((voltage-3.3/2)*2000*2/3.3);

    //escalamiento de 0 a 5v ----->rango de -2000 a 2000
    //entrada1= (float)((analogRead(AN0)-RESOLUCION/2)*2000*2/RESOLUCION);

    //printf("\n voltage: %.02f V\n",voltage);
    //printf("    entrada: %.02f V\n",entrada);

    return entrada;
}

```

```
void CleanupAD2() {  
    DisableCaches();  
}
```

```
void EnableCaches() {  
#ifdef __MICROBLAZE__  
#ifdef XPAR_MICROBLAZE_USE_ICACHE  
    Xil_ICacheEnable();  
#endif  
#ifdef XPAR_MICROBLAZE_USE_DCACHE  
    Xil_DCacheEnable();  
#endif  
#endif  
}
```

```
void DisableCaches() {  
#ifdef __MICROBLAZE__  
#ifdef XPAR_MICROBLAZE_USE_DCACHE  
    Xil_DCacheDisable();  
#endif  
#ifdef XPAR_MICROBLAZE_USE_ICACHE  
    Xil_ICacheDisable();  
#endif  
#endif  
}
```

ANEXO C. MANUAL DEL MOTOR BLDC HST23

ANEXO C. MANUAL DEL MOTOR BLDC HST23



**Better Performance.
Higher Efficiency.**

NTTM **HST**

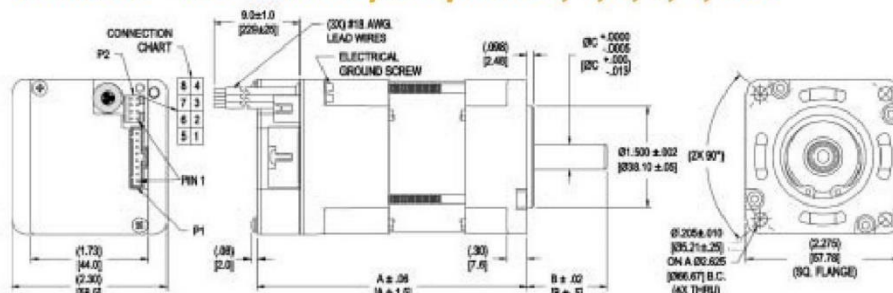


Typical Applications:

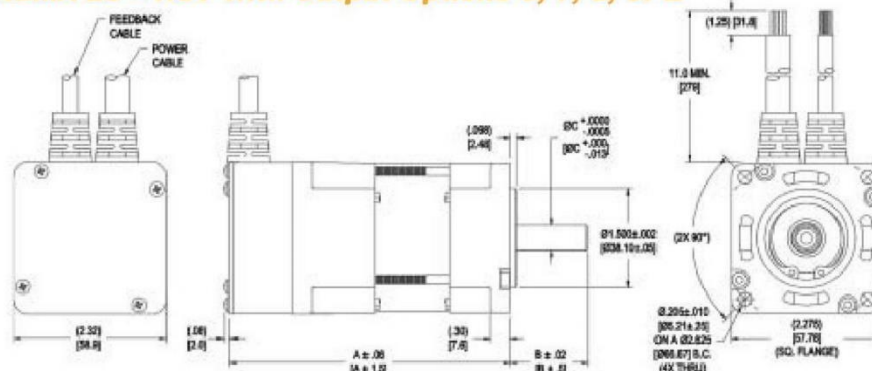
- Robotics
- High Speed Presses
- Office Automation
- Packaging
- Mail Handling Equipment
- X-Y Tables
- Conveyors
- Pumps
- Electric Mobility



NEMA 23 – HST with Output Options 0, 1, 2, 3, 4, or 5



NEMA 23 – HST with Output Options 6, 7, 8, or Z



Refer to 'Model Selection Guide'			'A' inches [mm]	'B' inches [mm]	'C' inches [mm]
Model	Output Option	Mechanical	Overall Length	Shaft Extension	Shaft Diameter
1	0, 1, 2, 3, 4, or 5	1	4.00 [101.6]	.81 [20.6]	.2500 [6.350]
		2	4.00 [101.6]	1.25 [31.8]	.3750 [9.525]
	6, 7, 8, or Z	1	4.26 [108.2]	.81 [20.6]	.2500 [6.350]
		2	4.26 [108.2]	1.25 [31.8]	.3750 [9.525]
2	0, 1, 2, 3, 4, or 5	1	5.00 [127.0]	.81 [20.6]	.2500 [6.350]
		2	5.00 [127.0]	1.25 [31.8]	.3750 [9.525]
	6, 7, 8, or Z	1	5.26 [133.6]	.81 [20.6]	.2500 [6.350]
		2	5.26 [133.6]	1.25 [31.8]	.3750 [9.525]

The information contained herein is subject to change without notice.

Contact Hurst Manufacturing at 812-386-7504 or www.hurst-motors.com

NTTMHST – Electrical Connections

Connections for Output Option 0, 1, 2, 3, 4, or 5

Connector / Lead	Pin # or Color	Function
Motor Power Leads	Blue	Phase A
	Red	Phase B
	Black	Phase C
	Customer Supplied	Ground (Frame)
*Halls (P1)	1	V _S
	2	V _S (return)
	3	Hall S2
	4	Hall S1
	5	Hall S3
	6	No Connection
	7	No Connection
	8	No Connection
**Encoder (P2)	1	+5 V _S
	2	A
	3	B
	4	Index
	5	+5 V _S (return)
	6	A not
	7	B not
	8	Index not

Connections for Output Option 7 or 8

Connector / Lead	Pin # or Color	Function
Motor Power Leads	Red	Phase A
	Black	Phase B
	White	Phase C
	Customer Supplied	Ground (Frame)
	Shield***	Shield
**Encoder Power	Red	+5Vs
	Black	+5Vs (return)
	Black/White	Case Ground
	Shield***	Shield
Encoder Outputs	Brown	A
	White	A not
	Blue	B
	Green	B not
	Orange	Z
	Yellow	Z not
	Violet	U
	Gray	U not
Commutation Outputs	Brown/White	V
	Red/White	V not
	Orange/White	W
	Yellow/White	W not

Connections for Output Option 6

Connector / Lead	Pin # or Color	Function
Motor Power Leads	Red	Phase A
	Black	Phase B
	White	Phase C
	Customer Supplied	Ground (Frame)
	Shield***	Shield
Resolver	Red	Ref Hi
	Black	Ref Lo
	White	Sine+
	Light Brown	Sine-
	Dark Brown	Cos-
	Green	Cos+
	Shield***	Shield

* Input voltage: 4.2 to 24 Vdc
 ** Input voltage: 4.75 to 5.25 Vdc
 *** Shield grounded at motor

Connectors & Terminals (mating):
 Halls (Molex brand): 50-57-9408, 16-02-0103
 Encoder (FCI brand): 65846-016, 48236-000



Hurst continually applies technology to provide our customers with innovative motion control solutions. These products deliver servo performance in a competitive package.

Combined with a precision bearing system and an insulation system using advanced magnet wire technology, the HST 2.3 is truly one of the most progressive motors in the industry.

Motor technology facilitating innovative motion control solutions.

The HST 2.3 is well suited for most servo applications demanding low rotor inertia, fast dynamic response, and high torque in a small package with low cogging torque. The standard flying leads, and the encoder or resolver

options, allow for easy installation regardless of the brand of amplifier utilized. Other lead configurations and feedback devices can be accommodated to meet user specifications.

Performance specifications are based on trapezoidal amplifier input. Increased performance can be achieved with the use of a sinusoidal amplifier.

Contact Hurst Manufacturing at 812-386-7504 or www.hurst-motors.com



NTTM HST – Selection Guide

Model Selection Guide

To construct a motor part number, substitute the appropriate digits for the x's as shown below.

H S T 2 3 x x x x x x x

Output Options
0 = Hall Sensor Outputs only
1 = 100 Line encoder w/ Hall Outputs
2 = 250 Line encoder w/ Hall Outputs
3 = 400 Line encoder w/ Hall Outputs
4 = 1000 Line encoder w/ Hall Outputs
5 = 256 Line encoder w/ index & Hall Outputs
6 = Resolver
7 = 1024 Line encoder w/ Hall Outputs
8 = 2048 Line encoder w/ Hall Outputs
Z = No Feedback Options (Power Leads only)

Model
Refer to Speed Torque Curves
1
2

Voltage
VDC Buss
012 = 12V
024 = 24V
036 = 36V
048 = 48V
160 = 160V
320 = 320V

Curve
Refer to Speed Torque Curves
A
B
C
D
E
*Consult Factory for Speed Torque Curves

Cables
0 = None
1 = Hall Output
2 = Hall Output & Encoder
*Cables required with Output Options 0 through 5 Additional Cables not required with Options 6, 7, 8 and Z

Mechanical
1 = NEMA 23 (23DO25)
2 = NEMA 23 (23DO38)





Features

- Brushless DC
- High Power Density for Demanding Applications
- Compact Package Size
- High Dynamic Performance
- Low Rotor Inertia
- Encoder and Resolver Options
- Gearing Available
- NEMA 23 Mounting

Electrical

- High Efficiency
- Precision Motion Quality
- UL Approved Insulation System
- 100% Final Tested
- Custom Windings Available
- Up to 140 oz-in Torque (without gearing)
- 12 Vdc to 320 Vdc Buss Voltage
- Pulse Resistant Magnet Wire
- Trapezoidal or Sine Drive Input Power

Mechanical

- Long Life Ball Bearing System
- Neodymium Ring Magnets (not arcs)
- Stainless Steel Shaft (custom sizes available)
- NEMA 23 Mounting
- Fast Dynamic Response
- Up to 4000 RPM Operation Direct Drive
- Cable and Connector Options Available

Encoders and Resolvers

- Encoder Line Counts Available:
 - 100 - 1000
 - 250 - 1024
 - 256 w/ Index Pulse - 2048
 - 400
- Resolver Options Available
- Encoders and Resolvers are Integrated into a Housing on the Rear of the Motor
- Hall Output Signals are Available for an Inexpensive Solution for Speed Control

Gearing Options

- Planetary
 - Wide Selection of Gear Ratios and Features
 - Inline, Right Angle, and Metric
 - High Torque and Low Backlash
- Integrated Design
- High Shaft Loading Capacity
- NEMA 23 Mounting

Contact Hurst Manufacturing at 812-386-7504 or www.hurst-motors.com



Hurst Manufacturing

www.hurst-motors.com

1551 E. Broadway • Princeton, IN 47670

(812) 385-2564 • Fax (812) 386-7504

www.hurst-motors.com

www.hurst-motors.com