SEP
SECRETARÍA DE EDUCACIÓN PÚBLICA

INSTITUTO TECNOLOGICO de la laguna

DIVISIÓN DE ESTUDIOS DE POSGRADO E INVESTIGACIÓN

"Implementación en hardware de redes neuronales para el control de presión invertida en horno."

POR

Ing. Jennifer Castillo García.

TESIS

PRESENTADA COMO REQUISITO PARCIAL PARA OBTENER EL GRADO DE MAESTRA EN CIENCIAS EN INGENIERÍA ELÉCTRICA

DIRECTOR DE TESIS
M.C. Martín Gerardo Vázquez Rueda.

ISSN: 0188-9060

RIITEC: (14)-MCIE-2013

Torreón, Coahuila. México,
Noviembre 2013

"2013, Año de la Lealtad Institucional y Centenario del Ejército Mexicano"

Dependencia: DEPI
Oficio: **DEPI/370/2013**
Asunto: **Autorización de impresión de tesis.**

Torreón, Coah., 20/Noviembre/2013

C. JENNIFER CASTILLO GARCIA
CANDIDATO AL GRADO DE MAESTRO EN CIENCIAS EN INGENIERÍA ELÉCTRICA.
PRESENTE

Después de haber sometido a revisión su trabajo de tesis titulado:

"Implementación en hardware de redes neuronales para el control de presión invertida en horno"

Habiendo cumplido con todas las indicaciones que el jurado revisor de tesis hizo, se le comunica que se le concede la autorización con número de registro RITEC: **(14)-MCIE-2013**, para que proceda a la impresión del mismo.

ATENTAMENTE

DR. JOSE LUIS MEZA MEDINA
Jefe de la División de Estudios de Posgrado e Investigación

SECRETARÍA DE EDUCACIÓN PÚBLICA
INSTITUTO TECNOLÓGICO de la Laguna
División de Estudios de Posgrado e Investigación

"2013, Año de la Lealtad Institucional y Centenario del Ejercito Mexicano"

Torreón, Coah., 12/Noviembre/2013

DR. JOSE LUIS MEZA MEDINA
JEFE DE LA DIVISIÓN DE ESTUDIOS DE POSGRADO E INVESTIGACIÓN
PRESENTE

Por medio de la presente, hacemos de su conocimiento que después de haber sometido a revisión el trabajo de tesis titulado:

"IMPLEMENTACIÓN EN HARDWARE DE REDES NEURONALES PARA EL CONTROL DE PRESION INVERTIDA EN HORNO"

Desarrollado por el C. **JENNIFER CASTILLO GARCIA**, con número de control **M05130818** y habiendo cumplido con todas las correcciones que se le indicaron, estamos de acuerdo que se le conceda la autorización de la fecha de examen de grado para que proceda a la impresión de la misma.

ATENTAMENTE

DR. MARTIN VAZQUEZ RUEDA
Asesor/Director

DR. FRANCISCO VALDES PEREZGASGA
Comité Tutorial

DR. ENRIQUE CUAN DURON
Comité Tutorial

DR. FRANCISCO G. FLORES GARCIA
Comité Tutorial

Contenido

Índice de figuras

Índice de tablas.

Índice de gráficas.

Agradecimientos y dedicatoria.

Gracias al **Consejo Nacional de Ciencia y Tecnología** por el apoyo brindado durante el transcurso de la maestria

Un agradecimiento muy especial al **M.C. Martín Gerardo Vázquez Rueda**, por sus consejos y guia durante todo el proyecto de investigación, gracias a su apoyo y confianza en mi trabajo y su capacidad para guiar mis ideas ha sido un aporte invaluable, no solamente en el desarrollo de esta tesis, sino también en mi formación como profesionista.

Gracias al **M.C. Juan Sifuentes Mijares** por introducirme al mundo de las FPGA y por la orientación que me brindo durante el estudio de posgrado.

Le dedico este trabajo a **DIOS** por brindarme los recursos necesarios en todos los aspectos para la culminación exitosa de este proyecto.

Papá, gracias por tu apoyo, la orientación que me has dado, por iluminar mi camino y darme la pauta para poder realizarme en mis estudios y mi vida. Agradezco los consejos sabios que en el momento exacto has sabido darme para no dejarme caer y enfrentar los momentos dificiles, por ayudarme a tomar las decisiones que me ayuden a balancear mi vida y sobre todo gracias por el amor tan grande que me das.

Mami, tu eres la persona que siempre me ha levantado los ánimos tanto en los momentos dificiles de mi vida estudiantil como personal. Gracias por tu paciencia y esas palabras sabias que siempre tienes para mis enojos, mis tristezas y mis momentos felices, por ser mi amiga y ayudarme a cumplir mis sueños, te quiero mucho. Hermanito...te quiero con todas las fuerzas de mi alma gracias por tus consejos y palabras de apoyo cuando más los necesite. Por las llamadas de atención que me brindabas cuando me desviaba de mi propósito, eres grande hermano. Gracias **Melissa y a Karla** por todo...por ser mis mejores amigas y por siempre estar ahí cuando las cosas se ponian dificiles. Eduardo muchas gracias por tu apoyo incondicional y por ser mi confidente. Gracias a **Sarita, Ariel** mis compañeros en esta aventura. Finalmente gracias a todas esas personas especiales que de una u otra manera contribuyeron para la culminación de este proyecto.

Implementación en hardware de redes neuronales para el control de presión invertida en horno.

Resumen

El trabajo de investigación que aquí se expone muestra un análisis sobre un fenómeno llamado presión invertida, el cual se presenta en el área de descarga de *clinker* del horno de calcinación de cemento al enfriador de parrillas para la producción del cemento Portland.

La presión invertida es causada por los movimientos de masa y la dirección de los flujos de aire dentro del sistema para la recuperación del calor. La aparición de la presión invertida provoca turbulencia que afecta el correcto funcionamiento del horno, disminuyendo su eficiencia y la calidad del proceso productivo.

Se implementaran Redes Neuronales Artificiales y el algoritmo de *Backpropagation* en elementos de *hardware*, con el objetivo de controlar los diferentes factores que contribuyen en la aparición y variación de la presión invertida.

Para validar el algoritmo de *Backpropagation* se realizara un prototipo, en el cual se simulará el control de la presión invertida, dicho control representará la unión del lenguaje gráfico con el lenguaje de descripción de *hardware* y el uso de FPGA (Dispositivo Lógico Programable/*Field Programmable Gate Array*) para un control eficiente.

Este reporte propone una solución para el control de la presión invertida, la cual consiste en el uso de un control inteligente (Redes Neuronales Artificiales) implementado en FPGA trabajando en paralelo con un control PID, con el objetivo de aprovechar los beneficios que se obtienen de realizar aplicaciones en elementos de *hardware*, así como utilizar a su mayor capacidad las características que la propia naturaleza del algoritmo de control nos ofrece. Los resultados obtenidos fueron satisfactorios con un margen de error de ±0.975 cm con respecto a la posición de set-point en un tiempo de estabilización promedio de 29.875 segundos.

Palabras Claves: Redes Neuronales Artificiales, *Backpropagation*, FPGA, PID.

Hardware implementation of neural networks for inverted pressure control in oven.

Summary

The research presented here shows an analysis of a phenomenon called reverse pressure, which occurs in the discharge area of *clinker* cement kiln to cooler grills used in Portland cement production.

Reverse pressure is caused by the mass movements and by the direction of the air flow within the system for heat recovery. The occurrence of reverse pressure causes turbulence that affects the proper functioning of the furnace, reducing its efficiency and affecting the quality of the product.

Artificial Neural Networks and a *Backpropagation* algorithm were built in hardware to control the various factors that contribute to the onset and the variation of inverted pressure. In order to validate the *Backpropagation*, an algorithm will be applied to a prototype where the inverted pressure control will be simulated. The control is built using a graphical language, a hardware description and using FPGAs (programmable logic device / Field Programmable Gate Array). This report proposes a solution to the inverted pressure control, which involves the use of intelligent control (Artificial Neural Networks) implemented in FPGA all working in parallel. Satisfactory results were obtained with a margin error of ± 0,975 cm with respect to the set-point position with a stabilization time average of 29,875 seconds.

Keywords: Artificial Neural Networks, *Backpropagation*, FPGA, PID

Capítulo I. Introducción

El cómputo inteligente se integra principalmente por tres grandes áreas: las redes neuronales artificiales, la lógica difusa y el cómputo evolutivo. En los últimos años se han propuesto nuevos conceptos como resultado de la observación y análisis del comportamiento de grupos de organismos y la forma en que resuelven diferentes tareas, en donde surge el término *Swarm Intelligence* [1].

En donde la primera de ellas (llamada habitualmente RNA o por sus iniciales en inglés ANN *Artificial Neural Network*), es un procesamiento inspirado en el funcionamiento del sistema nervioso de los seres vivos, en el cual un conjunto de neuronas interconectadas entre si dentro una red colaboran para producir una salida deseada.

La lógica difusa (*fuzzy logic*, en inglés) pretende agregar un grado de ambigüedad en las cosas que evalúa. En el mundo en el que vivimos existe mucho conocimiento ambiguo e impreciso por naturaleza, y el ser humano interactúa con este tipo de información con frecuencia, por lo que la lógica difusa fue diseñada para imitar este tipo de comportamiento.

Finalmente, el cómputo evolutivo interpreta a la naturaleza como una inmensa máquina de resolver problemas, es decir, es una técnica cuyo objetivo es encontrar una solución óptima a situaciones que son consideradas como no idóneas para el correcto funcionamiento de un sistema. Busca la solución a problemas encontrando su inspiración en la evolución de los seres vivos. Dentro del cómputo evolutivo se encuentra el desarrollo de los *algoritmos genéticos* [2].

El sistema que es motivo de análisis a lo que este trabajo de investigación se refiere, es la corrección de un fenómeno que se produce en la zona de descarga del horno de calcinación para la obtención del *clinker* necesario para la elaboración del cemento tipo Portland, y recibe el nombre de presión invertida.

En este sentido, la tesis propone la creación de una red neuronal artificial, donde el comportamiento de la red será regulado por el algoritmo de *Backpropagation* con el propósito de lograr el correcto desempeño del sistema. En donde las entradas a la RNA serían las principales variables que influyen a la aparición de la presión invertida, y la salida de la red sería el control de los elementos encargados (acción física de corrección) de realizar los ajustes necesarios hasta obtener la respuesta deseada.

Las redes neuronales son un concepto cuyo estudio y evolución data desde 1888 cuando Santiago Ramón y Cajal (1852- 1934) demostró que le sistema nervioso estaba formado por una red de células individuales llamadas neuronas [3].

El uso de RNA es la manera más eficiente de resolver el problema que se presenta en el horno de calcinación, ya que cuando la información de este sistema se encuentra en forma de pares de datos de entrada-salida, nos permite transferir una regla de control al sistema por medio de ejemplos de comportamiento, sin tener un modelo matemático exacto, además que el número de variables que se pueden monitorear y a su vez controlar aumenta en contraste a lo que sucede con un sistema de control tradicional, por ejemplo un controlador tipo PID como el que se tiene en este momento en el horno de calcinación para la producción del cemento Portland. Este controlador PID no ha presentado los resultados deseados, y de ahí la razón de este trabajo el cual pretende introducir el uso del control moderno para la solución y/o eliminación de la presión invertida dentro del sistema.

1.1 Motivación

Por más complejo que sea el sistema computacional implementado para la resolución de algún obstáculo, jamás tendrá la capacidad de procesamiento que el sistema nervioso del cerebro posee para la solución de un problema. Por eso, las redes neuronales artificiales imitan en un estructura *hardware* al sistema nervioso, con la intención de construir sistemas de procesamiento de información paralelos, distribuidos y adaptativos (que son las principales características que las RNA pretenden reproducir del funcionamiento del cerebro humano) para que dicho sistema se comporte de forma "inteligente".

En el caso de estudio del sistema analizado en concreto, que tiene como elemento final el control de diversos parámetros que definen el desempeño de un motor de corriente directa, en donde la determinación de los valores de estos parámetros serán definidos por diversas variables, las cuales al mismo tiempo contribuyen a que el motor muestre un comportamiento no lineal, provocando alteraciones en el sistema completo.

Una opción para disminuir la complejidad computacional requerida para lograr el ajuste necesario para que el motor se comporte de la manera deseada, es la implementación en *hardware* de redes neuronales artificiales, aprovechando los beneficios que el desarrollo sobre *hardware* implica, dentro de los cuales se encuentra que se puede emular del comportamiento del cerebro humano el paralelismo en la ejecución de diversas tareas.

1.2 Justificación

Actualmente existen métodos definidos para el control de motores de corriente directa, el uso de técnicas de control como lo son las RNA y más específicamente el algoritmo de *Backpropagation* ofrece una técnica de gran potencial, ya que nos permite transferir una regla de control al sistema, empleando ejemplos de comportamiento. Una regla de control nos permite obtener una respuesta especifica a raiz de una entrada en concreto, es por eso que se justifica el uso de redes neuronales artificiales, dado a la naturaleza no lineal del sistema.

Gracias a la capacidad de generalización que tienen las RNA se pueden aplicar a sistemas de naturaleza y complejidad diferente, ya que a partir de un conjunto de patrones de comportamiento, el sistema es capaz de producir perfiles de movimiento.

El uso de FPGA (*en inglés, Field Programmable Gate Array*), nos permite reproducir las principales caracteristicas del sistema nervioso, además de ser reprogramables lo que nos otorga la oportunidad de crear varios tipos de topologia de redes neuronales, la red neuronal implementada será tan grande y compleja como se necesite, solo limitada por la capacidad de recursos lógicos de la tarjeta.

Por los motivos mencionados anteriormente, para resolver el problema de la aparición de la presión invertida dentro del horno de calcinación, llamada también presión de la caperuza, en descarga del material proveniente del horno con dirección al enfriador de parrillas, se realizara la implementación de una RNA en una tarjeta FPGA.

1.3 Planteamiento del problema

En los hornos de calcinación para la producción del cemento Portland se presenta un fenómeno llamado presión invertida en la descarga del horno al enfriador, causada por los movimientos de masa y la dirección de los flujos de aire dentro del sistema para la recuperación del calor. La aparición de la presión invertida provoca turbulencia que afecta el correcto funcionamiento del horno, disminuyendo su eficiencia y la calidad del proceso productivo.

1.4 Hipótesis

Considerando las características no lineales del motor a controlar, y la complejidad que conlleva la obtención de un modelo matemático del sistema de funcionamiento del horno del calcinación para la obtención del *clinker*, al implementarse una red neuronal artificial en *hardware*, dicha red percibirá la información proveniente del exterior, la analizará y creará modelos de comportamiento, con el objetivo de ajustar diversos parámetros en la proporción necesaria, para corregir el fenómeno de la presión invertida en el horno de calcinación.

1.5 Objetivos

1.5.1 Objetivo general del proyecto

El objetivo principal del proyecto es controlar la presión invertida dentro del horno de calcinación del cemento implementando redes neuronales artificiales como técnica de control.

1.5.2 Objetivo especifico

1. Realizar una red neuronal genérica y/o reconfigurable, en la cual el usuario indicará la cantidad de neuronas que tendrá la red tanto en la capa de entrada, en la capa oculta y en la capa de salida.
2. Simular con elementos análogos las principales caracteristicas que contribuyen a la aparición de la presión invertida dentro del horno.
3. Realizar un prototipo.
4. Implementar en *hardware* el algoritmo de *Backpropagation*.

1.6 Límites y alcances

1.6.1 Limites

El tamaño, y por lo tanto la complejidad de la red neuronal artificial está determinada por la cantidad de recursos lógicos de la tarjeta FPGA

1.6.2 Alcances

Se espera que la red neuronal propuesta, sirva como un esquema básico que pueda ser empleado para construir y moldear sistemas de mayor complejidad disminuyendo la dificultad que implica la programación en estos sistemas.

1.7 Contribuciones

Implementación en la planta Cemex de Torreón del control realizado.

1.8 Descripción del proyecto

El proyecto fue desarrollado en cuatro etapas:

1) Instrumentación de los sensores.
2) Implementación de etapa de potencia.
3) Adquisición de las señales de entrada utilizando FPGA Spartan-3E.
4) Monitoreo y análisis de los valores entregados por los diversos sensores.

Esta etapa se desarrolló utilizando los lenguajes de programación grafica LabVIEW y lenguaje de programación de *hardware* VHDL.

5) Control de los actuadores (motores).

1.9 Metodología

- Se realizó la documentación necesaria con el objetivo de tener un mayor dominio de los temas a abordar.
- Se analizó el comportamiento global del sistema.
- Se definió el problema a corregir.
- Se establecieron las señales de entrada y salida del sistema a controlar y monitorear, respectivamente.
- Se caracterizó el comportamiento del sistema, y se definieron a través de un conjunto de pares de entrada-salida.
- Se definió que el control del sistema se realizaría por medio de la implementación de una red neuronal artificial en *hardware*.
- Se eligieron las herramientas de *hardware* y *software* para el desarrollo del proyecto.
- Se hizo el diseño del prototipo a construir.
- Se hizo el diseño de la electrónica asociada.
- Se realizó la programación de una red neuronal artificial.
- Se montaron los sensores en el prototipo.
- Se realizaron pruebas y correcciones al algoritmo propuesto.
- Se hizo el análisis de resultados

1.10 Organización de la tesis

La tesis se encuentra dividida en 12 capítulos. Se proporciona una descripción del contenido de cada capítulo.

El capítulo I "Introducción", contiene la motivación, justificación, planteamiento del problema, hipótesis, objetivos, límites y alcances, contribuciones, la metodología que se siguió finalmente se incluye una breve descripción del proyecto.

El capítulo II "Estado del arte de RNA", es una introducción a la teoría detrás del control utilizando redes neuronales artificiales, así como la definición de control inteligente, la estructura y la arquitectura de las RNA, tipos de aprendizaje, concluyendo con la Estructura de la red neuronal multicapa con el algoritmo de entrenamiento *Backpropagation.*

El capítulo III "Estado del arte FPGA", menciona la definición y evolución de los FPGA, la estructura general de los dispositivos FPGA, sus ventajas y desventajas. De igual forma se explican los grandes beneficios de la implementación por *hardware* y sus aplicaciones.

El capítulo IV "Proceso para la elaboración del cemento", tiene como objetivo abordar de manera breve y concisa los elementos involucrados en la fabricación el cemento Portland mediante vía seca, en donde se obtiene un producto intermedio llamado *clinker*.

El capítulo V "Funcionamiento del horno rotativo", proporciona el estado del arte del horno rotativo, el cual explica el funcionamiento del horno de calcinación de *clinker* para la fabricación del cemento Portland, también son mencionadas las principales partes del horno.

El capítulo VI "Descripción del problema", especifica el problema a resolver y su origen, las variables contribuyen a la aparición del problema y finalmente la solución propuesta.

El capítulo VII "Descripción del prototipo", documenta las etapas realizadas para la elaboración del prototipo, las cuales incluyen el diseño de la arquitectura de la estructura, construcción de la estructura, diseño de los diversos circuitos electrónicos utilizados en el prototipo, así como la instrumentación y montaje de algunos sensores.

En el capítulo VIII "Herramientas exploradas antes de LabVIEW" se abordan las diversas plataformas de programación con las que se pueden enlazar y programar las tarjetas FPGA.

El capítulo IX "Descripción de programación en VHDL", se centrará en la explicación de los programas realizados en VHDL.

El capítulo X "Descripción del algoritmo en LabVIEW", describe el algoritmo de control como una unión entre el lenguaje gráfico y el lenguaje de descripción de *hardware*.

El capítulo XI "Pruebas y resultados", proporciona las pruebas realizadas al sistema así como un análisis de los resultados obtenidos.

El capítulo XII "Conclusiones", hace referencia a las conclusiones y trabajos futuros.

ANEXOS

Bibliografía

Programación de FPGA

Son explicados diversos entornos de desarrollo especializados para el diseño de sistemas que se implementaran en FPGA haciendo uso de un lenguaje especializado llamado, HDL (*Hardware Description Language*) como por ejemplo: VHDL, Verilog Abel.

Manual de usuario

Es un manual que brinda al usuario las indicaciones necesarias para el control del prototipo y el correcto manejo de la interfaz de usuario.

Capítulo II. Estado del arte RNA

2.1 Control inteligente

El concepto de control inteligente se refiere al estudio y comprensión de como los seres humanos y algunos animales reaccionan a determinadas situaciones a las que se enfrentan, lo cual proporciona un medio para inferir procedimientos de cómo resolver problemas de control de un mayor grado de dificultad. Una metodología de control es inteligente si usa técnicas inspiradas en el comportamiento de seres humanos, animales o sistemas biológicos.

Existen diversas técnicas de control inteligente, pero en este trabajo se trabajará exclusivamente con el control utilizando redes neuronales artificiales.

2.2 Redes neuronales artificiales

Este capítulo proporciona el marco teórico sobre los fundamentos de las redes neuronales artificiales (RNA) o ANN (*Artificial Neural Network*), ya que el uso de esta metodología proporciona una eficaz alternativa para el diseño de sistemas de control, que debido a su alto grado de complejidad es difícil representar por medio de un modelo matemático convencional.

2.3 Breve introducción biológica

En 1888 Santiago Ramón y Cajal demuestra que el sistema nervioso estaba compuesto por una red de células individuales interconectadas entre sí, llamadas neuronas, desarrollando la idea de que la neurona es el componente más pequeño en la estructura del sistema nervioso,

finalmente afirmó que la información fluye en el interior de estas células individuales desde las dendritas hacia el axón atravesando el cuerpo o soma, figura 2.1.

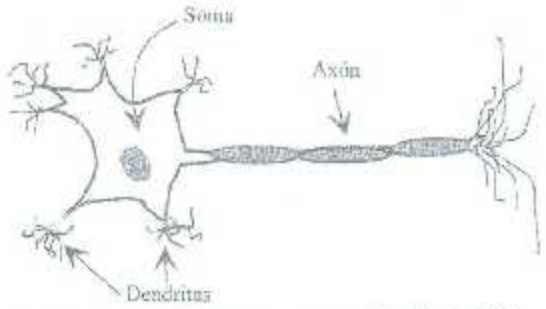


Figura 2.1 Estructura de neurona biológica [3].

2.4 Definición de red neuronal artificial

Diversos autores definen a las redes neuronales artificiales de forma distinta, entre las citas más sobresalientes, se encuentran:

Una red neuronal artificial (o, simplemente, una red neuronal) es un modelo computacional inspirado biológicamente que consiste en elementos de proceso (llamadas neuronas) con conexiones entre ellos, donde cada una de estas conexiones tiene coeficientes (pesos), y algoritmos de entrenamiento. Las redes neuronales son llamados modelos conexionistas debido al papel principal, que juegan las conexiones entre las neuronas. Los pesos de las conexiones son la "memoria" del sistema [4].

"Las redes neuronales artificiales son sistemas de *hardware* o *software*, de procesamiento, que copian esquemáticamente la estructura neuronal del cerebro para tratar de reproducir sus capacidades" (Nikola K. Kasabov, A Bradford Book, 1998, p251). [3]

No existe una definición establecida, ya que varía dependiendo del texto, artículo o publicación consultada, pero en la mayoría de las explicaciones que brindan los autores y específicamente en los conceptos mencionados en los párrafos anteriores, está presente el componente de simulación del comportamiento biológico. Es decir los conceptos, que se pretenden emular del funcionamiento de las redes neuronales biológicas son:

1. Paralelismo de cálculo: diversas neuronas trabajando en lo mismo proyecto de forma paralela en distintas áreas.
2. Memoria distribuida: La información es almacenada en cada sinapsis, por lo que si una sinapsis se daña solo se pierde una pequeña parte de información. Los sistemas neuronales biológicos son redundantes, por lo tanto son tolerante a fallos.
3. Adaptabilidad. Las ANN su adaptan a su entorno modificando su sinapsis, y aprenden de la experiencia, a esto se le llama generalización a partir de ejemplos.

2.5 Estructura de un sistema neuronal artificial

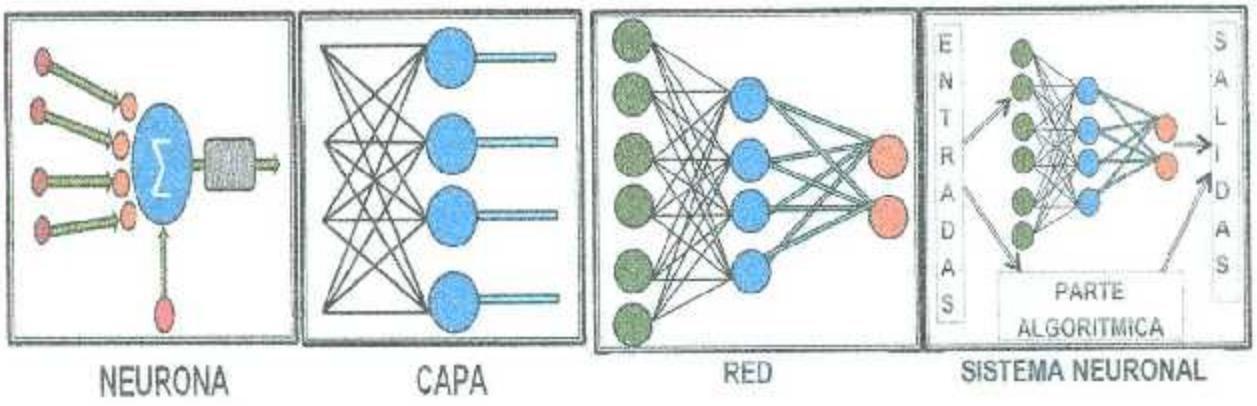


Figura 2.2 Estructura jerárquica de una RNA.

El elemento basico de una RNA comienza con una neurona, la cual es un modelo matematico simplificado de una neurona biologica, posteriormente, si a esa neurona se le conectan señales de entradas proveneitnes directamerte del exterior y se espera una salida resultante directamente al actuador a controlar (exterior) se forma una capa. En el caso en el cual las entradas a la neurona provienen de las salidas de neuronas adyacentes y a su vez la salida de estas neruonas va como entrada a otra neurona, y asi sucesivamente se forma una red neuronal en donde agregando una función de activación tenemos un sistema neuronal completo.

A toda metodología de control que utilice redes neuronales artificiales para el diseño de controladores se le llama "Neurocontrol".

2.6 Neurocontrol

El Neurocontrol, se refiere al uso de redes artificiales para controlar ciertas acciones encaminadas a producir un resultado especifico.

La capacidad de generalización que tienen las redes neuronales artificiales hace posible que puedan utilizarse como dispositivos de aproximación de funciones para realizar acciones de control, principalmente cuando la información disponible del sistema a controlar se encuentra en forma de pares de entrada y salida de información, es decir, cuando los modelos matemáticos de la dinámica de la planta que describen el funcionamiento de la misma no están disponibles.

2.7 Modelo estándar de neurona artificial

El modelo considera a las neuronas como unidades elementales de procesamiento de información, la cual ingresa a la red, es transmitida a otras neuronas por medio de las conexiones entre ellas, donde cada conexión tiene un "peso" o valor que multiplicara al valor de la señal de entrada. Cada neurona realiza un procesamiento propio tomando los valores de entrada de las neuronas de las capas anteriores Xi y los pesos sinápticos Wi de las conexiones hacia otras neuronas. Para "disparar" o inhibir la neurona, esto depende del valor del "bias" el cual, con fines de entrenamiento de la RNA puede ser tratada como una entrada adicional a la neurona.

Una neurona estándar consiste en:

- Un conjunto de entradas Xi y pesos sinápticos Wi.
- Una regla de propagación: Multiplicación de valor de entrada por el valor del peso sináptico de la conexión.
- Una función de activación

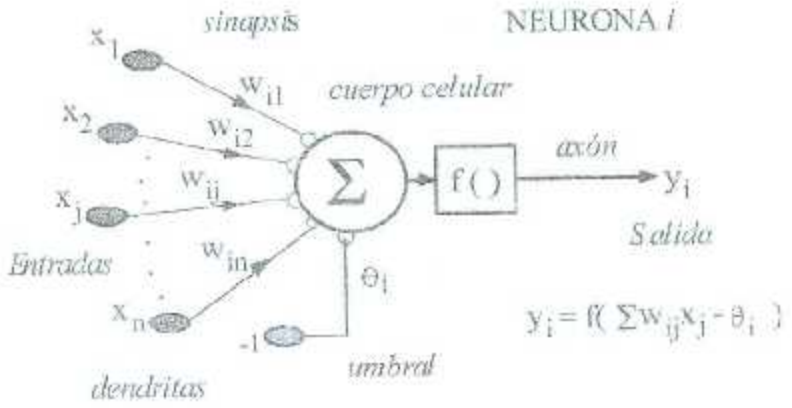


Figura 2.3 Modelo de una neurona estándar [3].

Existen diversas funciones de activación que dependiendo del rango de valores requeridos por la variable a controlar, son los valores que arrojaran como resultado a la salida de la red neuronal.

	Función	Rango	Gráfica
Identidad	$y = x$	$[-\infty, +\infty]$	
Escalón	$y = \text{signo}(x)$ $y = H(x)$	$\{-1, +1\}$ $\{0, +1\}$	
Lineal a tramos	$y = \begin{cases} -1, & si\ x < -1 \\ x, & si\ +1 \le x \le -1 \\ +1, & si\ x > +1 \end{cases}$	$[-1, +1]$	
Sigmoidea	$y = \frac{1}{1+e^{-x}}$ $y = \text{tgh}(x)$	$[0, +1]$ $[-1, +1]$	
Gaussiana	$y = Ae^{-Bx^2}$	$[0, +1]$	
Sinusoidal	$y = A.\text{sen}(\omega x + \varphi)$	$[-1, +1]$	

Figura 2.4 Funciones de activación habituales [3].

El modelo de la neurona estándar quedaría de la siguiente

Si,

$$Yi(t) = Fi\left(\sum Wij * Xj - \theta i\right);$$

Por lo tanto

$$Yi = 1, \text{ Si } \sum Wij \geq \theta i;$$

$$Yi = 0, \text{ Si } \sum Wij \leq \theta i;$$

Si la suma de sus entradas multiplicadas por sus pesos sinápticos es mayor o igual al umbral de disparo, la neurona se activa, y de caso contrario la neurona se inhibe.

2.8 Arquitectura de redes neuronales

La arquitectura de una red neuronal se refiere a la forma en la que las neuronas están interconectadas dentro de la estructura de la red. Las neuronas se agrupan en capas, y más de una capa forma una red neuronal.

En las redes neuronales monocapa, las neuronas solamente se pueden conectar con otras neuronas de la misma capa, es decir, conexiones laterales con las neuronas adyacentes, donde las entradas a las neuronas provienen del exterior y de la misma forma las salidas de la neurona son dirigidas exclusivamente al exterior.

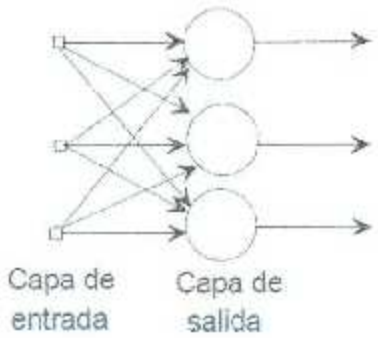


Figura 2.5 Red Neuronal Monocapa.

Las redes neuronales multicapa. son aquellas donde las neuronas pueden establecer conexiones con neuronas de capas diferentes a la propia.

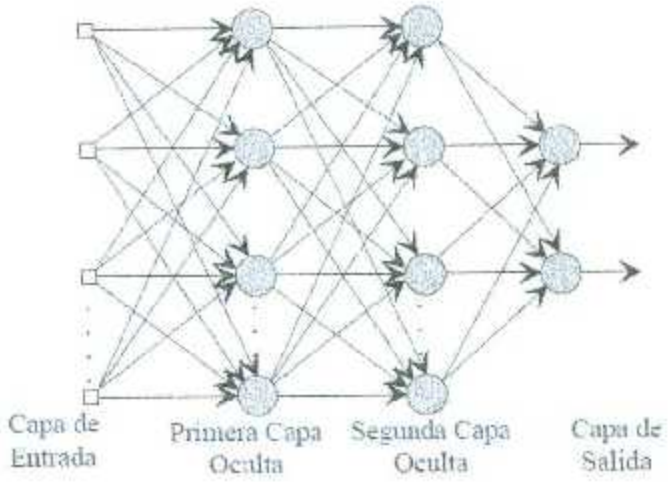


Figura 2.6 Red Neuronal Multicapa.

Dependiendo del sentido que tienen las conexiones entre las neuronas se pueden clasificar a las redes neuronales con conexiones hacia adelante y con retroalimentación, *Feedforward* y *Feedback*, respectivamente.

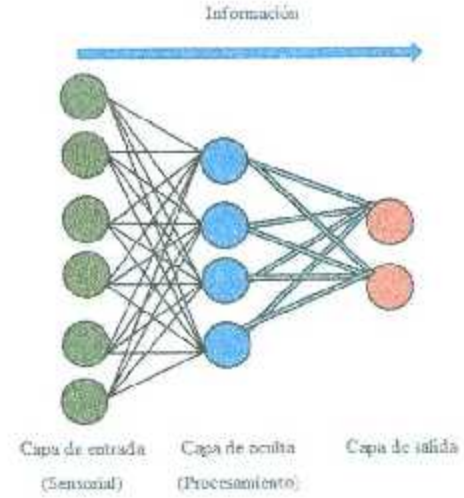


Figura 2.7 Arquitectura *feedforwad*, con una capa oculta.

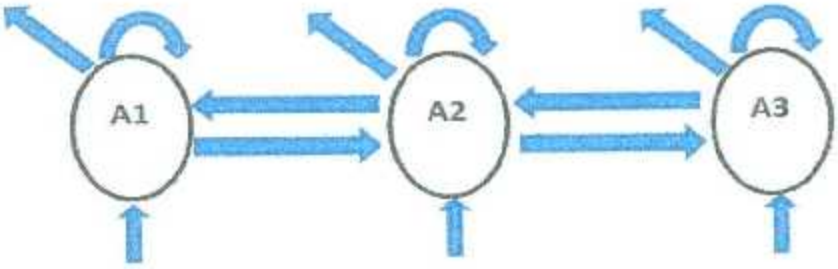


Figura 2.8 Red Neuronal Monocapa retroalimentada (*Feedback*).

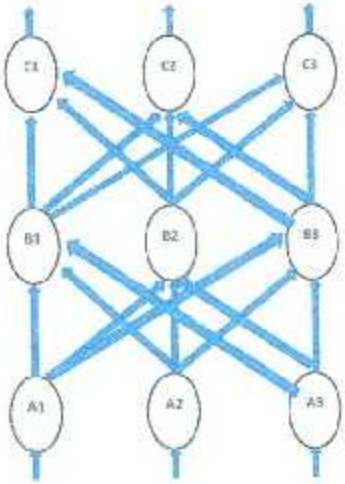


Figura 2.9 Red Neuronal Multicapa Unidireccional.

2.9 Redes *Feedforward*

Las redes tipo *Feedforward*, son redes neuronales con múltiples capas, en donde solo existen conexiones entre neuronas de niveles diferentes, es decir, capas diferentes, no existen lazos entre neuronas de la misma capa, son conexiones en secuencia hacia adelante, donde las conexiones comienzan en la capa que recibe el estímulo del exterior (capa de entrada), continuando con las capas intermedias u ocultas, hacia la capa de salida.

Previamente a utilizar cualquier tipo de arquitectura de red neuronal, es necesario entrenar la red, para el desarrollo de este trabajo de investigación se hace uso de una red multicapa con

conexión *Feedforward*. El objetivo de entrenar una red neuronal es que a un estímulo externo, la reacción de la red neuronal conduzca a una salida en específico.

Para realizar el entrenamiento de la red, se aplica una regla de entrenamiento, cuyo objetivo es adaptar los valores de los pesos sinápticos de las conexiones entre las neuronas, calculando los cambios requeridos para que los pesos y las polarizaciones o bias, respondan a un vector de entrada con la salida deseada.

Al proceso para encontrar los pesos y bias ideales hasta que el error sea mínimo, refiriéndonos a error como la diferencia entre la salida deseada y la salida obtenida, se le conoce como entrenamiento.

Existen dos tipos de aprendizaje: supervisado y no supervisado, en donde el primero se refiere a cuando se conoce la entrada y salida deseada y el segundo es desconocido el parámetro de salida deseada. En el desarrollo de esta aplicación se utiliza el aprendizaje supervisado.

2.9.1 Aprendizaje supervisado

En el aprendizaje supervisado, la red neuronal debe aprender el mapeo entre un vector de entradas X(t) y un vector de salidas deseadas Y(t).

En 1960, se introducen los primeros algoritmos de entrenamiento para redes *Feedforward*, refiriéndonos al algoritmo LMS (Widrow y Hoff) y la regla del perceptrón (Rosenblatt 1962), pero el mayor avance tiene lugar en el año 1971, Paul Werbos desarrolla el algoritmo de *Backpropagation* (algoritmo de entrenamiento) para arquitecturas neuronales con múltiples capas.

2.10 Estructura de la red del perceptrón simple

La estructura del perceptrón simple se muestra en la imagen 2.10, la red está constituida por un conjunto de entradas (capa de entrada) y una neurona en la capa de salida. Todos los valores de las neuronas de la capa de entrada provienen del exterior y la neurona de la capa de

salida está conectada a un actuador al exterior del sistema. Cada conexión tiene su respectivo peso sináptico, en donde al hacer la multiplicación del valor de la entrada por el valor del peso sináptico (repitiendo esta operación con cada una de las entradas), se realiza la suma ponderada en donde el valor total de la suma va a la función de activación, que por tratarse de un perceptrón simple la función sería una función escalón.

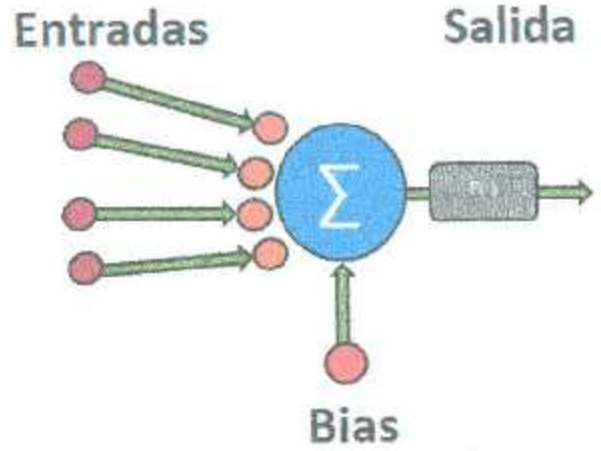


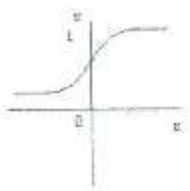
Figura 2.10 Estructura del perceptrón simple.

2.11 Red neuronal multicapa con *Backpropagation*

Si se añade una o más capas intermedias a un perceptrón simple, se obtendrá un perceptrón multicapa o MLP (Multi-Layer Perceptrón). La arquitectura multicapa se entrena por medio del algoritmo de retro propagación de errores o *Backpropagation*.

El algoritmo de *Backpropagation* es una red de aprendizaje supervisado. Cuando se aplica un patrón de entrenamiento a la capa de entrada, este se propaga desde la primera a través de las capas ocultas hasta llegar a la capa de salida de la red. Al final se compara la salida de la red con la salida deseada, indicándonos el error. La función de activación debe ser diferenciable, generalmente se utiliza la función sigmoide, limitando los valores de salida a un rango de entre 0 y 1 figura 2.11.

$$z = \frac{1}{1+e^{-n}}$$



si $n >> 0$ entonces $z \rightarrow 1$
si $n << 0$ entonces $z \rightarrow 0$

Figura 2.11 Función sigmoide.

Una vez obtenido el error, este es propagado hacia las capas de adelante, comenzando por la capa de salida, pasando por las capas ocultas hasta llegar a la capa de entrada.

Basándonos en la señal del error percibido, se actualizarán los pesos sinápticos de las conexiones entre las neuronas. Este procedimiento de calcular el error de la red y posteriormente retornarlo hacia adelante (*Feedforward* y *Backpropagation*) se repetirá hasta encontrar los pesos sinápticos óptimos que permitan la correcta respuesta de la red ante algún patrón de comportamiento en específico.

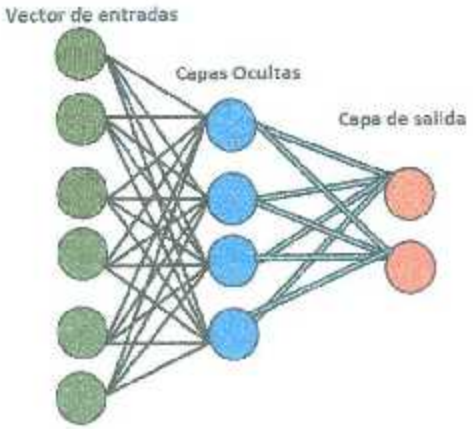


Figura 2.12 Estructura del perceptrón multicapa.

2.12 Descripción del algoritmo de *Backpropagation*

El algoritmo se basa en calcular el gradiente descendiente del error cuadrático medio a la salida, para obtener el error en los pesos sinápticos. A continuación se describe el algoritmo a implementa.

- ❖ Pasos hacia delante:

1. Selecciona un vector de entrada, es decir, un patrón de entrenamiento.

2. Con el patrón de entrenamiento seleccionado, se calcula la salida de la red (suma ponderada y función de activación)

- ❖ Pasos hacia atrás:

3. Calcular el error entre la salida obtenida y la salida deseada.
4. Ajustar los pesos sinápticos para que el error entre la salida obtenida y la salida deseada sea mínimo.
5. Se vuelven a ejecutar los pasos del 1 al 4 con todos los patrones de entrenamiento, hasta que el error mínimo cuadrático global sea aceptablemente bajo y nos brinde las salidas deseadas.

2.13 Determinación del tamaño de la red neuronal.

Es necesario definir parámetros como:

- Tamaño de la red neuronal
- Función de activación de las neuronas
- Cantidad de neuronas en la capa de entrada
- Cantidad de neuronas en la capa de salida
- Numero de patrones de entrenamiento

El **tamaño de la red,** se refiere a la cantidad de capas que tendrá la arquitectura de la red, así como la cantidad de neuronas en cada capa. A su vez el conjunto de neuronas en la capa de entrada es directamente proporcional a la cantidad de señales a monitorear, y las neuronas en la capa de salida es determinada por suma total de actuadores a controlar de una aplicación en específica. Para establecer una cifra de capas intermedias u ocultas existen criterios, reglas heurísticas o métodos especializados para acotar estos parámetros [5].

Los **criterios,** toman en cuenta aspectos muy generales sobre el diseño de la red, por ejemplo, si la red es muy pequeña, los resultados obtenidos serán poco precisos pero de la misma manera, si la red es muy grande, se complica el aprendizaje y el tiempo de convergencia de los

pesos sinápticos es mayor. También se sabe, que una red neuronal con al menos una capa oculta, donde su función de activación sea no lineal, y en la capa se salida se utilice una función de activación que sea linealmente separable, puede aproximar cualquier función con un error relativamente pequeño, su comportamiento se puede describir como un aproximador universal de funciones.

Por otro lado, las **reglas heurísticas,** toman consideraciones que no están comprobadas matemáticamente, pero han demostrado ser eficientes para diversas aplicaciones prácticas. Lippmann [6] considera que un perceptrón multicapa (MLP: *Multilayer perceptron*) con una sola capa intermedia, es suficiente para resolver problemas de relativa complejidad, solo si, la capa oculta tiene mínimo 2m neuronas, donde m se refiere a la cantidad de neuronas de la primera capa.

El teorema de Kolmogorov, donde Hech-Nielsen [7] afirma que una red neuronal con una capa oculta, con neuronas con función de activación no lineal que tiene 2m+1 neuronas, es apta para aproximar cualquier función, con m cantidad de entradas.

La regla de la pirámide geométrica, considera que una red neuronal con una única capa oculta, el número de neuronas en la capa oculta se puede determinar utilizando $\sqrt{m \times n}$, donde m y n corresponden a la cantidad de entradas y salidas, respectivamente.

2.14 Sistema de control propuesto

El sistema de control propuesto es un sistema no lineal basado en la implementación en *hardware* para el control de la presión invertida en horno cementero.

En la primera fase, en el proceso de la captura de los datos provenientes de los diversos sensores colocados en diversas áreas del prototipo, cuya posición fue estratégicamente determinada para obtener la lectura de los valores lo más preciso posible de la variable a monitorear, la red neuronal artificial *Feedforward* aprenderá del mapeo entre el valor que uno de los sensores (sensor ultrasónico) y la señal de control (señal pwm) necesaria para alcanzar

el valor de set point determinado por el usuario y/o proceso, con lo cual se transferirá la regla de control para uno de los actuadores (motor).

Tanto las señales de entrada (señales codificadas) y las señales de salida (señal de control), serán monitoreadas utilizando la tarjeta FPGA Spartan 3E de Xilinx hacia la PC por medio del puerto USB.

Después de la etapa de aprendizaje el sistema podrá controlar los actuadores de forma que estos se ajusten para mantener y/o alcanzar cualquier valor de *Set Point* establecido por el usuario y/o proceso.

2.15 Ventajas del sistema de control propuesto.

Las ventajas del sistema de control propuesto son

- No es necesario establecer un modelo matemático no linea de la planta.
- El mapeo entre la señal de set point determinada por el usuario y la señal de control que le corresponde para mantener la señal de set point en el valor adecuado, se estará estableciendo la regla de control para el sistema.
- La red será reconfigurable, por lo cual si el proceso a controlar requiere el monitoreo un mayor número de sensores y/o variables o si es necesario el control de más actuadores el total establecido inicialmente, es posible realizar estos cambios.

2.16 Resumen

En el trabajo de investigación realizado para la elaboración de este proyecto de tesis, se plantea un sistema de control de motores que tienen la capacidad de aprender la tarea de control, empleando una cantidad finita de patrones de comportamiento, donde el entrenamiento y/o modo de trabajo de la red se lleva a cabo por medio de la implementación en hardware de una red neuronal artificial.

Capitulo III. Estado del arte FPGA

3.1 Definición de FPGA

Los dispositivos FPGA (*Field Programmable Gate Array*) se basan en lo que se conoce como arreglos de compuertas, los cuales consisten en la parte de la arquitectura que contiene tres elementos configurables: bloques lógicos configurables (CLB), bloques lógicos de entrada y salida (IOB) y canales de comunicación [8]. Por dentro, un FPGA está formado por arreglos de bloques lógicos configurables que se comunican entre ellos y con las terminales de entrada/salida (E/S) por medio de alambrados llamados canales de comunicación [9]. El funcionamiento del dispositivo FPGA será definido por el programa que el usuario diseñe, donde la única limitante será la capacidad de interconexiones disponibles que la FPGA posea, en otras palabras.

Los dispositivos FPGA son chips integrados por componentes básicos que se pueden conectar de la forma que el usuario decida para que se comporte de la forma del circuito digital que queremos diseñar. Una vez que se descarga el programa en la FPGA, lo que se tiene es *hardware* diseñado a nuestra medida.

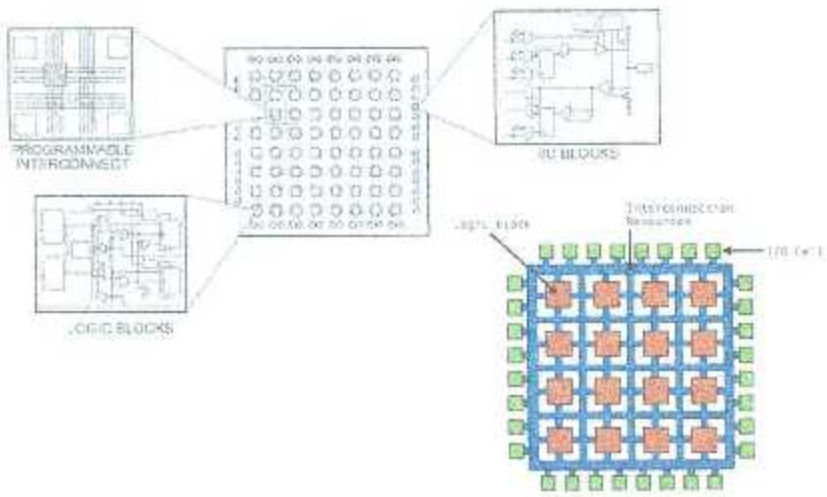


Figura 3.1 Arquitectura Básica de un FPGA.

3.2 Evolución de los FPGA

La definición de dispositivo lógico programable, se refiere a aquel circuito de propósito general que posee una estructura interna que puede ser modificada por el usuario final para implementar una amplia gama de aplicaciones [10].

La evolución en el desarrollo de los circuitos integrados se ha venido perfeccionando a través de los años. Primero se desarrollaron los circuitos de baja escala de integración (SSI o *Small Scale Integration*), después los de mediana escala de integración (MSI o *Medium Scale Integration*) y posteriormente los de larga integración (LSI o *Large Scale Integration*) para continuar con los de muy alta escala de integración (VLSI o *Very Large Scale Integration*) para llegar finalmente a los circuitos integrados de propósito específico (ASIC o *Aplication Specific Integratded Circuit*)

Las FPGA surgen como una evolución de los CPLD (*Complex Programmable Logic Device*). Los FPGA fueron inventados en 1984 por Ross Freeman y Bernand Vonderschmitt co-fundadores de Xilinx. Las FPGAs contienen hasta 100 mil bloques lógicos, mientras que los CPLD contienen 100 bloques lógicos, ambos con flip-flops. Además, las FPGAs pueden contener funciones aritméticas como sumadores, comparadores y memoria RAM. Sin embargo, los CPLDs poseen tiempos de entrada/salida más rápidos que las FPGA.

Una tendencia reciente ha sido combinar los bloques lógicos e interconexiones de los FPGA con microprocesadores y periféricos relacionados para formar un sistema programable en un chip [11].

3.3 Estructura general de los dispositivos FPGA

Un dispositivo lógico programable contiene CLB's y estos a su vez LUTs, enseguida se explicará cada uno de estos conceptos.

Una FPGA consiste en arreglos de varios bloques programables (bloques lógicos) los cuales están interconectados entre sí y con celdas de entrada/salida mediante canales de conexión verticales y horizontales, tal como muestra la figura 3.2.

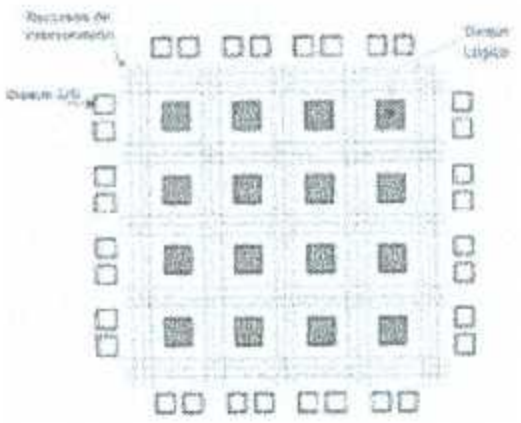


Figura 3.2 Arquitectura básica de un FPGA.

3.3.1 Bloques Lógicos Configurables (CLB)

El bloque lógico consta de una parte conbinacional que permite implementar funciones lógicas booleanas, más una parte secuencial que permite sincronizar la salida con una señal de reloj externa e implementar registros.

3.3.2 Bloque lógico basado en LUT (*Look-Up Table*)

Una LUT también llamada generadora de funciones es un componente de células de memoria SRAM que almacena una tabla de verdad, como el de la figura 3.3. Las direcciones de las células son las entradas de la función lógica que se quiere implementar, y en cada celda de memoria se guarda el resultado para cada una de las combinaciones de las entradas. En una LUT de n x 1 es posible implementar cualquier función lógica de n entradas.

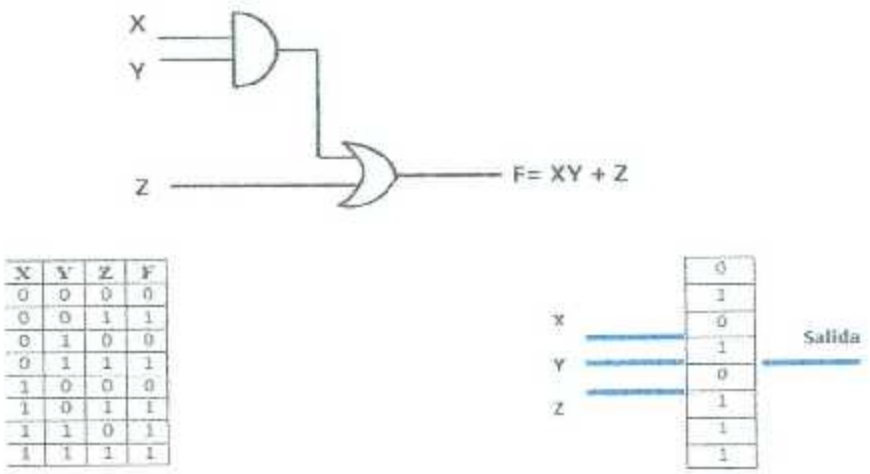


Figura 3.3 Bloque lógico basado en LUT

3.3.3 Bloques de entrada /salida (I/O)

La función de un bloque de entrada/salida es permitir el paso de una señal hacia dentro o hacia el exterior del dispositivo, las cuales deben de tener la posibilidad de configurar I/O de *pull-up* o *pull-down*.

3.4 Ventajas de las FPGA

Un FPGA contiene bloques de lógica cuya interconexión y funcionalidad se puede programar, dicho de otra manera, los FPGA son reprogramables, y gracias a ello los programadores pueden desarrollar aplicaciones con un mayor grado de diversidad, refiriéndonos tanto al giro de la aplicación y en cuanto al grado de complejidad característico del diseño, reduciendo el tiempo de implementacion y sus costos de desarrollo [12].

3.5 Desventajas de las FPGA

La principal desventaja que estos dispositivos poseen, es que la complejidad del diseño se ve limitada por la densidad de compuertas lógicas que tiene el FPGA, la cual es una característica

determinada por el fabricante, es decir, un FPGA es un conjunto de compuertas logias programables, la cantidad de estas compuertas es fijo por lo tanto el usuario no puede modificar este recurso.

3.6 Beneficios de la implementación por *hardware*

Los microprocesadores son circuitos integrados de funcionamiento configurable y las FPGA son circuitos integrados de arquitectura configurable (modificando los recursos a utilizar, ya sea la cantidad de entradas y salidas, el tamaño de los buses, la cantidad de memoria, etc. a diferencia de los microprocesadores en los cuales solo puedo modificar el programa, porque los recursos disponibles están definidos por el fabricante y no se pueden alterar.

Otro de los beneficios de la implementación en *hardware* por encima de los microcontroladores es que ellos están basados en una arquitectura CPU y ejecutan las instrucciones de una manera secuencial, a diferencia de las FPGAs que son dispositivos de lógica programable y el algoritmo se ejecuta de una manera paralela. Las FPGAs se utilizan como prototipos, los cuales se pueden depurar y permiten mejorar el diseño.

3.7 Principales fabricantes

Debido al éxito que las FPGA han tenido desde sus inicios, en la actualidad existen una serie de fabricantes que ofrecen una gran variedad de modelos de FPGA y tarjetas de desarrollo con diferentes características y a precios variables, donde el programador elegirá la tarjeta que mejor se acomode a sus necesidades.

Entre los principales fabricantes de FPGA, se encuentran:

- Xilinx, es uno de los dos grandes líderes en la fabricación de FPGA.
- Altera es el otro gran líder.

- Lattice Semiconductor, lanzó al mercado dispositivos FPGA con tecnologia de 90nm. En adición, Lattice es un provecdor lider en tecnologia no volátil, FPGA basadas en tecnologia Flash, con productos de 90nm y 130nm.
- Actel, tiene FPGAs basados en tecnologia Flash reprogrammable. También ofrece FPGAs que incluyen mezcladores de señales basados en Flash.
- QuickLogic ,tiene productos basados en antifusibles (programables una sola vez).
- Atmel, es uno de los fabricantes cuyos productos son reconfigurables (el Xilinx XC62xx fue uno de éstos, pero no están siendo fabricados actualmente). Ellos se enfocaron en proveer microcontroladores AVR con FPGAs, todo en el mismo encapsulado.
- Achronix Semiconductor, tienen en desarrollo FPGAs muy veloces.
- MathStar, Inc., ofrecen FPGA que ellos llaman FPOA (Arreglo de objetos de matriz programable).

3.8 Aplicaciones

Las FPGA al ser dispositivos lógicos programables, sus aplicaciones son tan diversas y complejas como el programador disponga, siendo de las principales aplicaciones las ramas que incluyan a los DSP, sistemas aeroespaciales y de defensa, sistemas de procesamiento digital de imágenes para medicina, sistemas de visión para computadoras, reconocimiento de voz, bioinformática, emulación de *hardware* de computadora, implementación de redes neuronales artificiales (RNA), donde la última hace mención al proyecto de tesis, ya que las técnicas de control con RNA requieren un alto grado de paralelismo en sus operaciones, que por ser programación de *hardware* no se hace en forma secuencial sino en paralelo .

Capítulo IV. Proceso para la elaboración del cemento

El presente capitulo tiene como objetivo abordar de manera breve y concisa los elementos involucrados en la fabricación el cemento Portland mediante vía seca, en donde se obtiene un producto intermedio llamado *clinker*.

4.1 Generalidades

El horno rotativo es el elemento central dentro de todo el proceso de fabricación del *clinker*, el cual es una estructura que se encuentra interconectada con otros dispositivos para de esa forma integrar un proceso productivo completo.

El cemento es "un conglomerante hidráulico, es decir, un material inorgánico finamente molido que amasado con agua, forma una pasta que fragua y endurece en virtud de reacciones y procesos de hidratación y que una vez endurecido, conserva su resistencia y estabilidad aún bajo el agua. Específicamente, el cemento Portland es un tipo de cemento que se obtiene de la cocción en horno rotativo de una mezcla de caliza y arcilla hasta la sinterización (en la que parte del material se encuentra en estado líquido), obteniéndose un producto intermedio denominado *clinker*, y que finalmente la mezcla y molienda de este producto intermedio con otras adicciones y la aportación de yeso como regulador de fraguado" [13].

4.2 Materia prima

Como materia prima se emplean sustancias minerales que contienen los componentes principales del cemento: cal, sílice, alúmina y óxido férrico, pero muy escasamente se encuentran en una sola muestra las proporciones adecuadas, por lo que es necesario hacer una nueva mezcla el cual tendrá un componente rico en cal (componente calcáreo) mezclado con otro pobre en cal pero que contiene más alúmina y óxidos de hierro (componente arcilloso).

Por lo tanto las dos materias primas principales son los materiales calizos (ricos en carbonato cálcico) y los materiales arcillosos (ricos en silicatos de aluminio hidratados).

4.3 Proceso de producción para la obtención de cemento Portland.

Actualmente, el proceso de producción del cemento Portland es un sistema totalmente automatizado, el cual es sometido a controles de calidad para así obtener el producto final óptimo que cumpla con los requerimientos del cliente.

El cemento se obtiene a partiendo del procesamiento de los minerales adecuados que extraídos de cantera, se preparan, muelen y se introducen en unos hornos donde se producen las reacciones químicas necesarias para su transformación en un producto intermedio denominado *clinker*.

Este producto intermedio, tras haber sido enfriado, se mezcla y muele finamente con otros productos como yeso y otras adiciones para obtener finalmente el tipo de cemento deseado. En conclusión los diferentes tipos de cemento se obtienen en función de la adición aplicada y en *clinker*.

A continuación se describirá de forma genérica el procedimiento a seguir dentro del proceso productivo, figura 4.1.

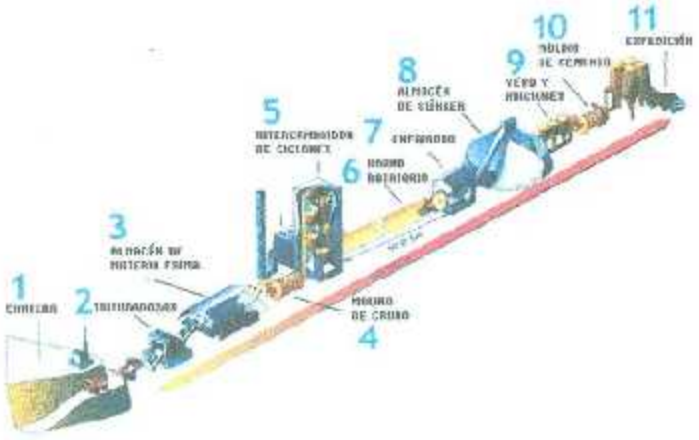


Figura 4.1 Proceso de producción del ciclo del cemento [13].

1. Extracción de materias primas: Se extrae la piedra caliza de las canteras, la cual es la materia prima del proceso.
2. Trituración: Las rocas fragmentadas que se obtuvieron en el paso anterior, pueden llegar a medir hasta un metro, por lo cual es necesario triturarlas hasta un máximo de 20 mm., que serán transportadas hasta los almacenes de prehomogenización.
3. Prehomogenización y almacenamiento de materia prima: Es necesario alcanzar una composición mineralógica uniforme y óptima, lograr esto ayudará a un mayor ahorro energético en el proceso de fabricación y aumentara la calidad del producto final.
4. Molienda de crudo: La mezcla de material prehomogenizado se transporta a los molinos de crudo, de barras o bolas de acero. La molienda tiene como objetivo el conseguir la composición química adecuada y la granulometría deseada, con el mínimo de consumo energético. A la par de la molienda se realiza el secado del material.
5. Pre-calentamiento: Antes de ingresar al horno, la harina de crudo homogenizada pasa por el intercambiador de ciclones de precalcinación. Aprovechando el calor residual del horno se con sigue un importante ahorro energético.
6. Horno rotativo: La harina de crudo pasa a los hornos rotativos de calcinación, los cuales son grandes cilindros de acero recubiertos internamente de un material refractario. Dentro del horno el material sufre una serie de transformaciones físicas y químicas conforme va aumentando la temperatura.

   -Secado, hasta los 150°C

   -Deshidratación de la arcilla, hasta los 500°C

   -Descarbonatación, entre 550°C y 1100°C

   -Clinkerización, entre 1300°C y 1500°C
7. Enfriamiento: El *clinker* pasa de 1450°C a 140°C aproximadamente, mediante un enfriador de parrillas. Los gases liberadas con el calor residual del horno se envían a los ciclones de precalentamiento en un proceso continuo que optimiza el aprovechamiento energético.
8. Almacenamiento del *clinker*: El *clinker* es almacenado en silos.
9. Yeso y adiciones: Antes de efectuar la molienda de *clinker* se dosifican cantidades variables de yeso (3-10%) para alargar el tiempo de fraguado del cemento.

10. Molienda de cemento: Una vez dosificado el yeso y las adiciones, los materiales se mueles y homogenizan dentro de molinos de bolas de acero y con eso se obtienen el producto final, cemento Portland.
11. Expedición: El cemento es empacado en sacos de papel con una capacidad de 25 Kg.

4.4 Reacciones químicas en la cocción del cemento

Para fabricar el *clinker* del cemento Portland partiendo del material crudo, es preciso calcinarlo hasta una temperatura de aproximadamente 1500°C para alcanzar la clinkerización. Durante el proceso de cocción del crudo tienen lugar importantes procesos físico-químicos, los cuales se muestran en la tabla 4.1.

Tabla 4.1 Reacciones químicas que ocurren durante la cocción del crudo [15].

PUNTO DEL PROCESO	T°C	PROCESO	TRANSFORMACIONES QUÍMICAS MAS IMPORTANTES
TORRE DE INTERCAMBIO DE CALOR	<200	Secado. Eliminación del agua libre.	
	100-400	Eliminación del agua	
	400-750	Descomposición de la arcilla con formación de metacaolinita	$Al[(OH)_8 Si_4O_{10}] \rightarrow 2(Al_2O_3 \cdot 2SiO_2) + 4H_2O$
HORNO ROTATORIO	600-900	Descomposición de la metacaolinita y otros componentes con formación de una mezcla de óxidos reactivos.	$Al_2O_3 \cdot SiO_2 \rightarrow Al_2O_3 + 2SiO_2$
	600-1000	Descomposición de la caliza con formación de	$CaCO_3 \rightarrow CaO + CO_2$

	CS y CA y formación de CO2	$3CaO + 2SiO_2 + Al_2O_3 \rightarrow 2(CaO \cdot SiO_2) + CaO \cdot Al_2O_3$
800-1300	Fijación de la cal por CS (silicato cálcico) y CA (aluminato cálcico) con formación de C4AF (aluminoferrito tetracálcico) y C3A (silicato tricálcico).	$CaO \cdot 2SiO_2 + CaO \rightarrow 2CaO \cdot SiO_2$ $2CaO + SiO_2 \rightarrow 2CaO \cdot SiO_2$ $CaO \cdot Al_2O_3 + 2CaO \rightarrow 3CaO \cdot Al_2O_3$ $CaO \cdot Al_2O_3 + 3CaO + Fe_2O_3 \rightarrow 4CaO \cdot Al_2O_3 \cdot Fe_2O_3$
1250-1450	Nueva fijación de cal por C2S (silicato bicálcico) con formación de C3S en presencia de fase líquida.	$2CaO \cdot SiO_2 + CaO \rightarrow 3CaO \cdot SiO_2$

Es de extrema importancia que se lleven a cabo las reacciones químicas completas, ya que de esto dependerá la calidad del cemento resultante.

4.5 Composición química final del *clinker*

El 95% del *clinker* está formado por óxidos de cal, sílice, aluminio y hierro. El resto lo forman óxidos que proceden de las impurezas y entre los cuales están los de magnesio, sodio, potasio, titanio, azufre, fósforo y manganeso.

Los principales óxidos que se encuentran presentes en el cemento Portland se muestran en la tabla 4.2.

Tabla 4.2. Principales óxidos presentes en el cemento Portland. [14]

Componente	**% en el cemento Portland**
Cal (CaO)	58-67

Sílice (SiO_2)	16-26
Al_2O_3	4-8
Fe_2O_3	2-5
Magnesia (MgO)	1-5
Álcalis (N_2O-K_2O)	0-1
SO_3	0.1-2.5
P_2O_5	0-1.5
Mn_2O_5	0-3
TiO_2	0-0.5
Pérdida al rojo	0.5-3

Los cuatro primeros óxidos, llamados óxidos principales, son los responsables de la formación, mediante las reacciones de clinkerización, de los constituyentes principales del *clinker* en la etapa de cocción del crudo. Los constituyentes principales se nombran en la tabla 4.3, y son los que otorgan a los cementos sus propiedades técnicas características.

Tabla 4.3. Principales constituyentes del *clinker* [14].

Nombre	Composición	Forma abreviada	Nombre del mineral
Silicato tricálcico	$3CaO \cdot SiO_2$	C_3S	Alita
Silicato bicálcico	$2CaO \cdot SiO_2$	C_2S	Belita
Aluminato tricálcico	$3CaO \cdot Al_2O_3$	C_3A	-
Ferritoaluminato tretacálcico	$4CaO \cdot Al_2O_3 \cdot Fe_2O_3$	C_4AF	Celita

El silicato tricálcico o alita, concede las altas resistencias iniciales al cemento.

El silicato bicálcico o belita, da lugar a pocas resistencias en los primeros días, pero luego las va desarrollando progresivamente hasta alcanzar al silicato tricálcico.

El aluminato tricálcico por sí solo contribuye poco a las resistencias pero, en presencia de los silicatos desarrolla unas resistencias iniciales buenas, se cree que actúa como catalizador de la reacción de los silicatos. Su fraguado es muy rapido al tomar contacto con el agua Para retardar su gran actividad se emplea el yeso que actúa como regulador de fraguado.

El ferrito aluminato tetracálcico o celita, apenas si tiene contribución en la resistencia de los cementos. El hierro que forma la celita tiene una gran importancia como fundente en el horno y es responsable del color gris verdoso de los cementos Portland, normalmente en éstos se encuentra en una proporción de un 3% y si su contenido se reduce al 0.5% o menos, se obtiene el cemento Portland blanco.

Capítulo V. Estado del arte de horno rotativo

En la información que se muestra a continuación se explica el funcionamiento del horno de calcinación para la obtención del *clinker* en la elaboración del cemento Portland, el cual es un tipo de horno rotativo, en donde el horno tiene el rol principal en el proceso productivo de este tipo de conglomerante.

5.1 Funcionamiento del horno de calcinación del cemento

Consiste en un cilindro de acero, que se encuentran ligeramente inclinado respecto a la horizontal (menos de 10 grados) y que gira a velocidades inferiores a 5 r.p.m. El calentamiento del horno se efectúa con gases calientes que se producen por combustión, en un quemador, de gas, u otros combustibles. El quemador se ubica en el extremo más bajo.

Esto significa que el extremo inferior del horno, donde se ubica el quemador, es la zona más caliente. Los gases producidos van recorriendo el horno y entregando su calor, saliendo por el extremo opuesto [14].

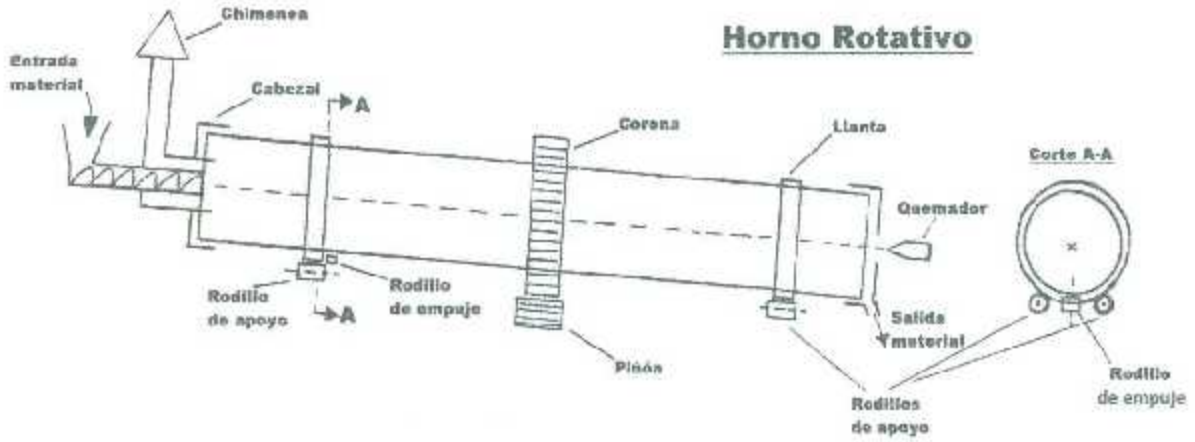


Figura 5.1 Dibujo de un horno rotativo para fabricación de *clinker* [13].

El material a procesar es alimentado por el extremo superior del horno. A consecuencia de la inclinación y rotación del horno, el material se desplaza a lo largo del mismo hasta el extremo inferior (lado del quemador). El material circula a contracorriente con respecto al calor.

5.2 Intercambiador de ciclones con precalcinador

El precalentador se usa para calentar el material que va a entrar al horno rotativo, a efectos de lograr un mayor rendimiento térmico del proceso y economizar combustible.

Existen precalentadores de distintos tipos, pero todos se basan en aprovechar los gases calientes que salen del horno e intercambiar su calor en forma directa con el material ingresante al horno en grandes torres que cuentan con conductos y ciclones.

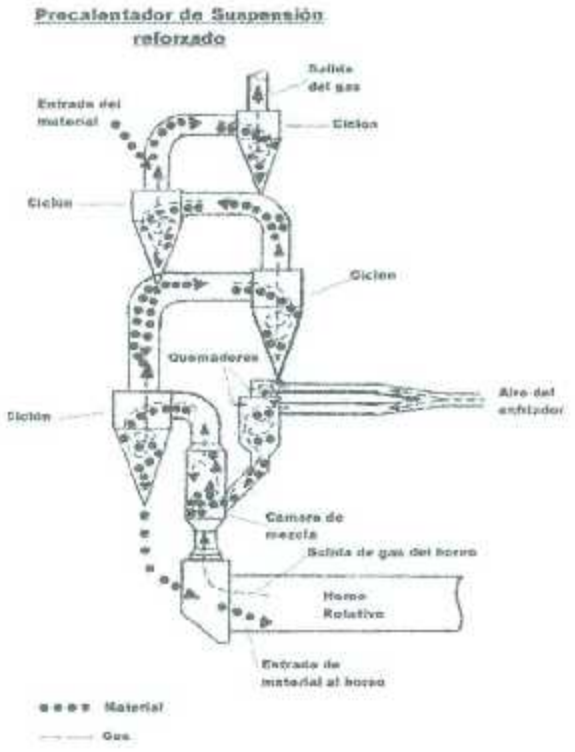


Figura 5.2 Esquema de una torre de Precalentador de un horno rotativo [13].

Cuando el crudo sale del intercambiador de ciclones ingresa al precalcinador, donde el crudo es descarbonatado y calcinado casi en su totalidad.

5.3 Enfriadores de *clinker*

El crudo caliente a la salida del horno, se trata seguidamente en los enfriadores de *clinker*. La característica común que tienen todos los enfriadores es el flujo directo de aire de enfriamiento, a contracorriente o transversalmente a través del *clinker*, recuperando el calor del aire así calentado, como aire secundario para la combustión en el horno.

5.3.1 Los enfriadores de parrillas

La parrilla consiste en una correa o cadena sin fin formada por elementos sueltos. Durante el proceso de enfriamiento, el *clinker* permanece sobre las placas de la parrilla móvil.

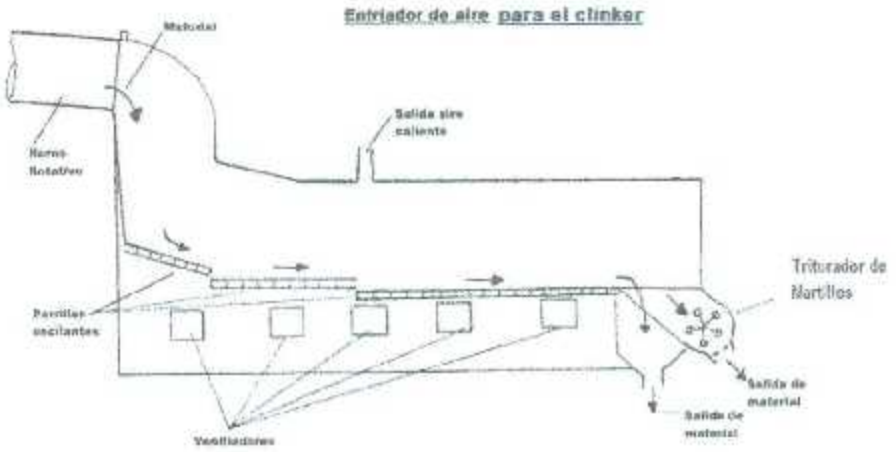


Figura 5.3 Salida de *clinker* del horno rotativo al enfriador de parrillas [13].

El enfriamiento se efectúa en dos zonas. En la zona primaria se ingresa aire por debajo de la cama de *clinker*, a través de las rendijas de las placas de la parrilla. Al mismo tiempo, esta se contribuye a la distribución uniforme del *clinker* a todo lo ancho de la parrilla. Una presión

menor de aire se aplica en la zona de enfriamiento secundario, para que el *clinker* se asiente y permanezca en reposo.

Un Triturador de martillos desmenuza los trozos de *clinker* de mayor tamaño al caer a la parrilla móvil esparciéndola para su mejor enfriamiento.

En la figura 5.4, se aprecia la conexión entre los componentes que forman parte del proceso productivo de cemento Portland y que fueron descritos en los párrafos anteriores.

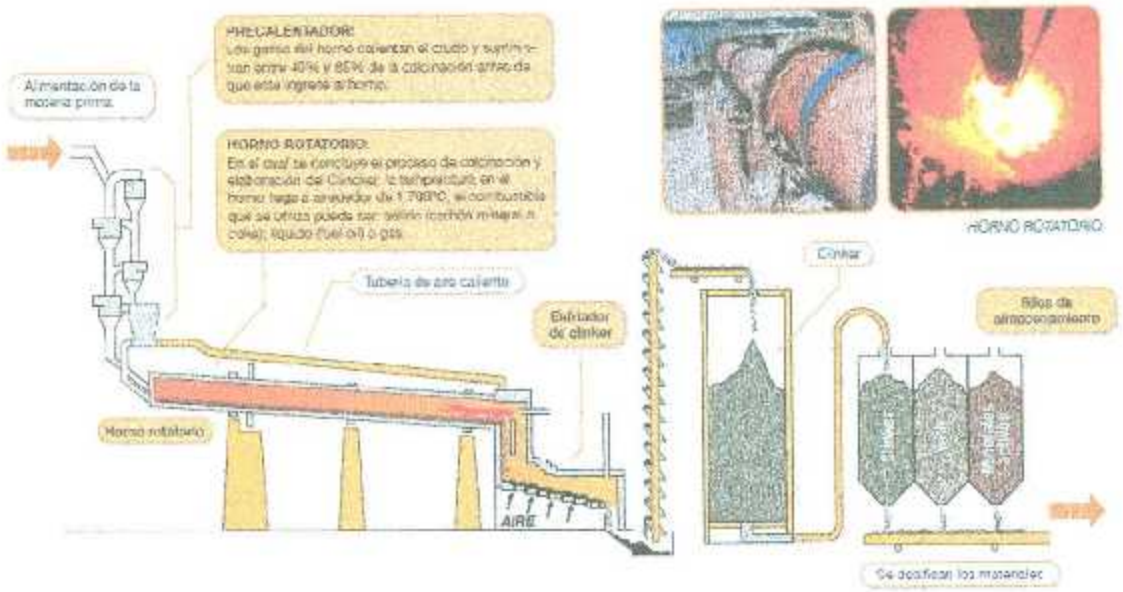


Figura 5.4 Funcionamiento del horno rotativo [13].

En conclusión, las condiciones de operación del horno son las siguientes, el crudo sale del molino a una temperatura de 70°C, inmediatamente pasa a un intercambiador de ciclones de cuatro etapas donde el crudo es calentado hasta una temperatura de 340°C, una vez que el material ha pasado por la última etapa del intercambiador de ciclones se introduce a un precalcinador, donde el crudo es descarbonatado y calcinado casi en su totalidad.

Posteriormente el material es ingresado al horno rotativo, donde se alcanzan las temperaturas lo suficientemente elevadas para que se lleven a cabo las reacciones químicas para la sinterización del *clinker*, donde fácilmente el material alcanza temperaturas de 1450°C y los gases de combustión hasta los 2000°C. Finalmente el *clinker* es ingresado al enfriador de

parrillas utilizando una corriente de aire a 22°C, hasta que la temperatura del *clinker* disminuya hasta alrededor de 140°C.

Capítulo VI. Presión invertida

Dentro de los hornos rotativos para la fabricación del *clinker* y posteriormente la obtención del cemento Portland, se presenta una presión invertida, este fenómeno es también llamado presión de la caperuza. En el desarrollo de este capítulo se abordaran temas referentes al origen de esta presión inversa, que provoca dentro del horno, es decir, como afecta la calidad del proceso productivo, para finalmente concluir con una posible solución a este problema.

6.1 Origen del problema

Uno de los principales problemas en la elaboración del cemento Portland, además de la fabricación de un producto de alta calidad que cubra las necesidades de los consumidores, es el ahorro energético durante el proceso productivo. Fue mencionado en capítulos anteriores, que cuando el *clinker* sale del horno rotativo a temperaturas de alrededor de 1500°C es necesario disminuir la temperatura para que el *clinker* continúe con su proceso productivo para así obtener como producto final el cemento Portland, y para esto se hace uso de un enfriador de parrillas el cual se encarga de disminuir la temperatura del *clinker*.

De la misma forma se cuenta con otro ventilador llamado "ventilador residual" o " ventilador VTI", el cual funciona cono extractor, es decir, tiene como objetivo el redireccionar de forma propicia el aire que en algún momento fue introducido al sistema por los ventiladores de enfriamiento, logrando así un mayor aprovechamiento energético el aire caliente es conducido al intercambiador de ciclones, para calentar el crudo y de esa manera cuando la mezcla ingrese al horno de calcinación, se utilice una menor cantidad de combustible para llevar el material a la temperatura óptima.

Si el ventilador VTI succiona una mayor cantidad de aire, al que están ingresando los ventiladores de enfriamiento, provoca una presión invertida, en la parte de descarga de horno a la parrilla de ventiladores de enfriamiento, provocando que la flama del quemador, la cual se encuentra en uno de los extremos del horno, cambie de sentido ocasionando turbulencias y afectando el correcto funcionamiento del horno, disminuyendo la calidad del cemento.

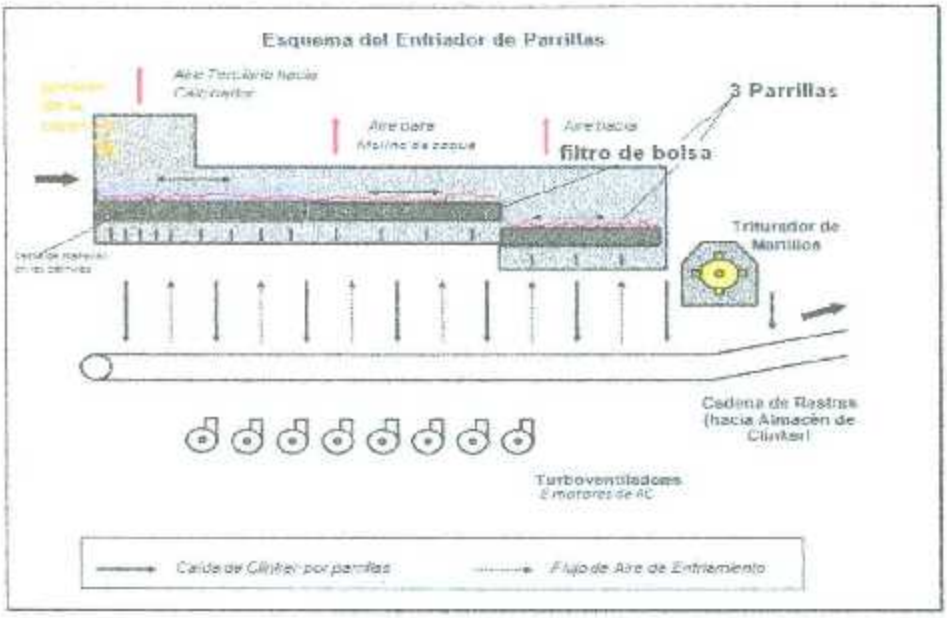


Figura 6.1. Esquema de enfriador de parrillas [13].

En la figura 6.1, el punto amarillo que se encuentra localizado a la izquierda de la imagen, es la sección del sistema en donde se produce la presión invertida, ya que en este punto es donde ocurre la descarga del *clinker* saliente del horno rotativo, además que es la sección de enfriamiento primario de los ventiladores de parrilla. El aire terciario que aparece en la imagen 6.1, es el aire que es redirigido al intercambiador de ciclones, como se explicó en los capítulos previos.

6.2 Problema a resolver

El problema a resolver es la aparición de la presión invertida dentro del horno de calcinación, en la descarga del material proveniente del horno con dirección al enfriador de parrillas

Tratando de mantener una presión de succión baja por parte del ventilador VT1 para evitar un tirón de aire hacia el horno.

6.3 Variables que contribuyen en la aparición de la presión de la caperuza.

- Flujo total de aire de los ventiladores del enfriador de parrillas
- Flujo de aire succionado por el motor VTI (motor de aire residual).

6.4 Forma de simular las diversas variables del sistema real.

1. El flujo de aire total de los ventiladores de parrilla, se simularán con un solo ventilador.
2. El flujo de aire del motor VT1 se simulará con un motor, el cual funcionará como extractor.
3. La presión invertida (presión de la caperuza) lo indicará la posición de la pelota, es decir, como simulación de la flama, se añadirá al prototipo a construir una esfera la cual indicará cuando la presión invertida se presente.

6.5 Definición de variables de entrada y de salida del sistema

ENTRADAS:

1. Velocidad del ventilador de enfriamiento del material.
2. Velocidad del ventilador de extracción (motor VTI).
3. Calidad de la flama: Sensor ultrasónico (posición de la pelota dentro de la estructura).

SALIDA:

1. Control de la velocidad de motor VTI.

6.6 Solución propuesta

Migrar del control tradicional con el que cuenta actualmente CEMEX planta Torreón (Control PI) por el uso de redes neuronales artificiales, haciendo una implementación en *hardware*, utilizando como dispositivo de control una tarjeta FPGA para el control de la velocidad del motor residual.

Capítulo VII. Descripción del prototipo

En este capítulo se documentan las etapas realizadas para la elaboración del prototipo, las cuales incluyen el diseño de la arquitectura de la estructura, construcción de la estructura, diseño de los diversos circuitos electrónicos utilizados en el prototipo, así como la instrumentación y montaje de algunos sensores.

7.1 Analogía del proceso real con la construcción del prototipo

En el proceso productivo original cuenta con dos flujos de aire de origen distinto producidos dentro de la estructura real, donde uno de ellos es originado por los ventiladores de parrillas. En el proceso real el enfriador de parrillas cuenta con ocho ventiladores, pero en el prototipo se tendrá un solo ventilador que simulará el flujo total de ellos. El segundo flujo de aire proviene en el proceso real de un extractor de aire llamado VTI o residual, cuya función es redireccionar el aire caliente que fue emitido originalmente por los ventiladores de enfriamiento con dirección a la torre de precalentamiento. Para simular la presencia del ventilador VTI se tendrá otro ventilador.

Debido a que el diseño de las aspas de los motores que se consiguieron para la construcción de está diseñado para funcionar como extractor de aire, por lo tanto, en lugar de tener un ventilador y un extractor, se contará con dos extractores de flujo de aire en la estructura. Los dos extractores de aire que estarán montados uno en cada extremo de un tubo de acrílico.

El problema a resolver en la elaboración de este proyecto de investigación, es controlar la presión invertida en la descarga del *clinker* del horno al enfriador de parrillas, el prototipo no tendrá una flama de quemador real ni un sensor de presión, así que la forma de indicar que la presión invertida está presente, es por medio del monitoreo de la posición del objeto que dentro del tubo de acrílico.

Tomando en cuenta las consideraciones ya mencionadas, se prosiguió con el diseño de la estructura.

7.2 Modelado en Solidworks

Antes de hacer la construcción física del prototipo del proyecto, se realizó un modelado del mismo utilizando Solidworks, este es un programa de diseño asistido por computador para modelado mecánico.

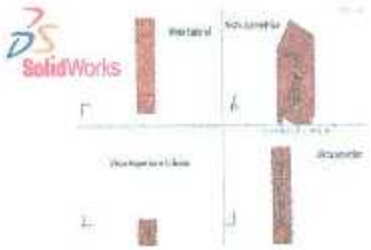


Figura 7.1 Logotipo de *software* SolidWorks.

Se pensó que era conveniente hacer el diseño de forma virtual en la computadora antes de llevarlo a cabo físicamente, para visualizar de forma analítica los diferentes aspectos de la arquitectura y hacer las modificaciones pertinentes para que dicha estructura cumpliera con los requisitos que el proyecto le exigiría, y de esta manera evitar hacer modificaciones estructurales posteriores a la construcción y armado del prototipo y así ahorrar dinero y tiempo de posibles modificaciones.

En la figura 7.2, se muestra el diseño final del prototipo. Se decidió pertinente hacer la estructura del prototipo con esta arquitectura, para que brindará una mayor estabilidad y así las lecturas de los sensores que serán explicados más adelante en este capítulo nos proporcionaran una lectura más confiable y certera, ya que se montaron dos ventiladores, los cuales alcanzan hasta los 4A de consumo de corriente, y por lo tanto giran a velocidades muy altas ocasionando elevadas vibraciones en la estructura, por lo tanto al tener una arquitectura de cuatro posters unidos por una tapa superior e inferior, se disminuía de forma considerable las vibraciones del prototipo.

Entre dichas tapas (superior e inferior) estaría sostenido un tubo de acrílico, y dentro de este tubo estará una figura de nieve seca la cual simula la posición y orientación de la flama del proceso real

Figura 7.2 Vista frontal del diseño final de arquitectura del prototipo.

En la imagen 7.3, se muestras diferentes vistas del prototipo a construir.

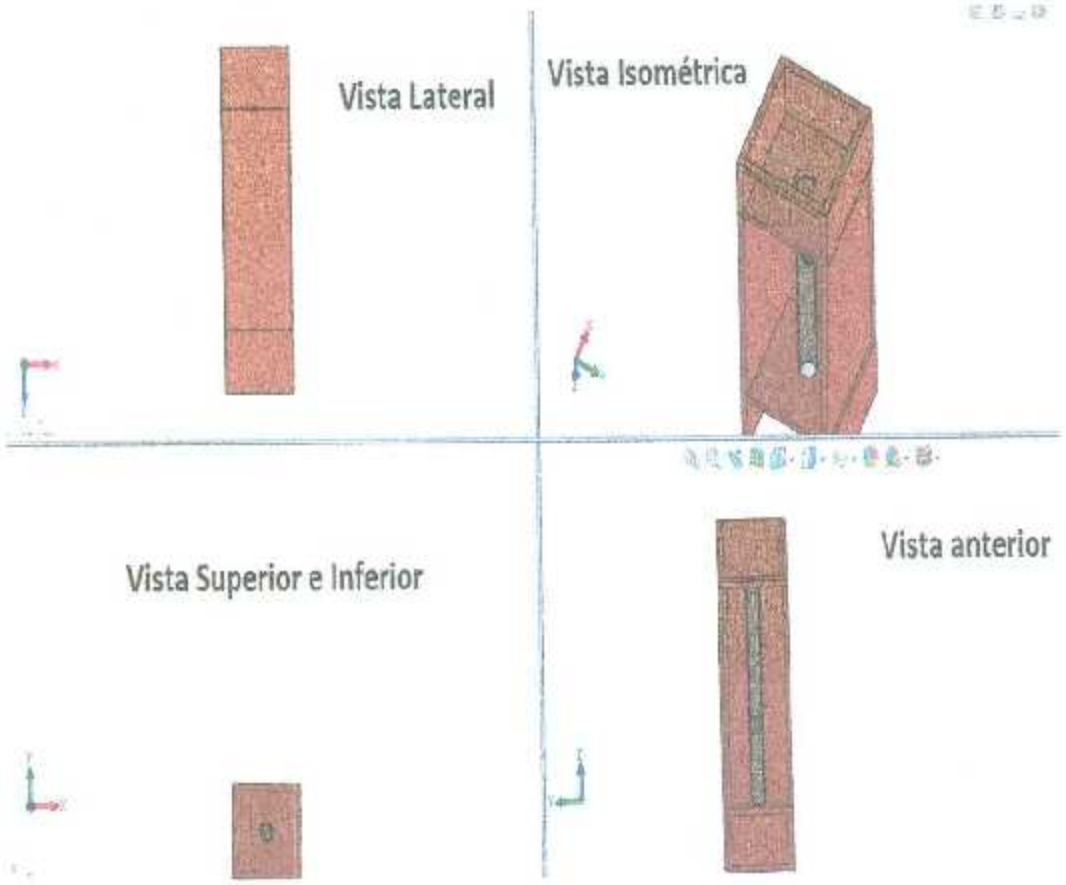


Figura 7.3 Diversas vistas del prototipo, diseño realizado en Solidworks.

Una vez tomada la decisión de hacer el prototipo igual a la figura 7.3, se prosiguió con la construcción.

7.3 Construcción del prototipo

Las paredes laterales de la estructura del prototipo están hechas de madera con las medidas de 6 mm x 0.3 m x 1.43 m. Se tienen dos tapas (superior e inferior) de 0.015 m x 0.3 m x 0.3 m.

Para unir las tapas superior e inferior, se fijaron cuatro posters de 0.02 m x 0.02 m x 1.30 m.

En cada una de las tapas se realizaron agujeros concéntricos al centro de la tabla, de 0.06 m de diámetro.

El tubo de acrílico utilizado es de 1.10 m de longitud por 0.06 m de diámetro, con un espesor de 3 mm. En la figura 7.4, se muestra una imagen real de la estructura, ejemplificando las medidas ya mencionadas.

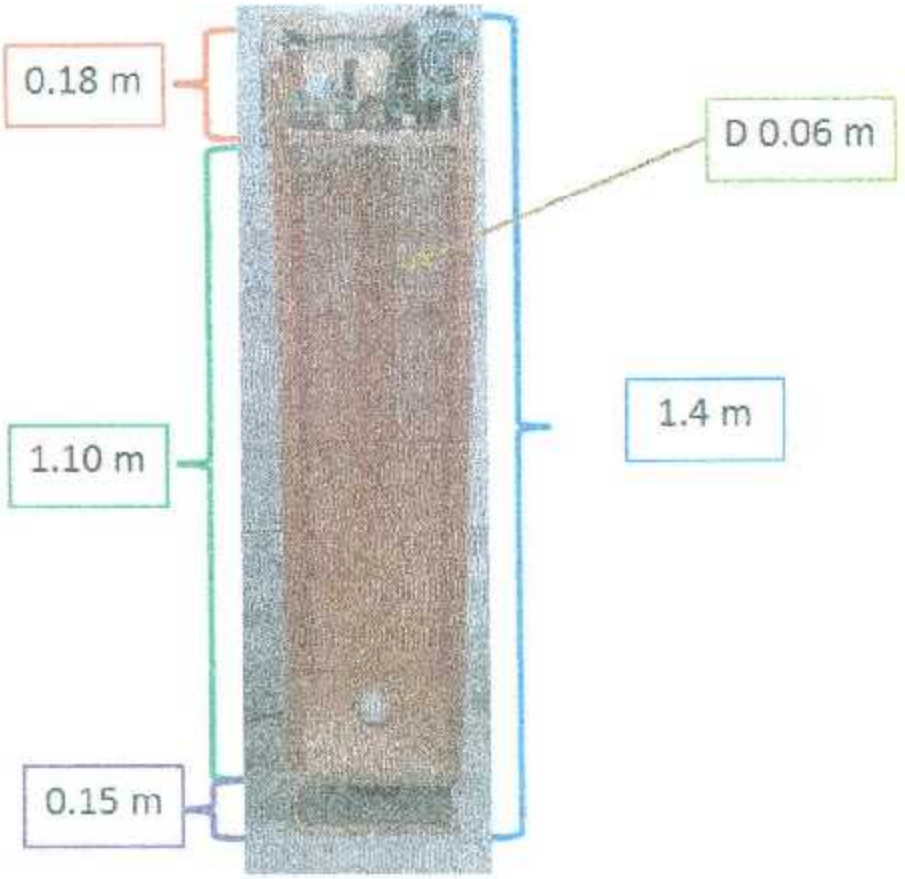


Figura 7.4 Imagen real del prototipo.

7.4 Electrónica asociada

Es necesario desarrollar los circuitos electrónicos para conocer la velocidad de giro de los ejes de los motores acoplados a la estructura del prototipo, se utilizará un sensor óptico infrarrojo cuyo funcionamiento se basa en la capacidad de reflexión del objeto. Además, se requiere

variar la velocidad de giro de los motores, para lograr esto es posible utilizar un método de modulación digital que nos permita modificar este parámetro en el comportamiento de los motores. El método que se acondicionará por su facilidad de control, es la modulación por ancho de pulso, PWM. Asimismo se diseñará una fase de acoplamiento de señales entre la etapa de potencia y la de control.

A continuación, se describe brevemente el funcionamiento e implementación del control PWM, del sensor óptico infrarrojo y la etapa de acoplamiento de señales.

7.4.1 Codificador óptico

Para determinar la velocidad del eje del motor se utiliza el sensor CNY70, el cual es un sensor de infrarrojos de corto alcance basado en un emisor de luz y un receptor, ambos apuntando en la misma dirección, y cuyo funcionamiento se basa en la capacidad de reflexión del objeto, y la detección del rayo reflectado por el receptor.

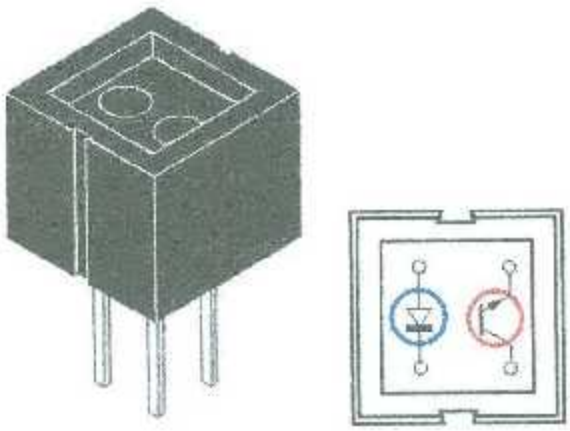


Figura 7.5 Vista externa y diagrama de conexión interno de CNY70.

El CNY70 tiene cuatro pines de conexión. Dos de ellos se corresponden al ánodo y cátodo del emisor (led), y las otras dos corresponden con el colector y el emisor del receptor (transistor).

En la imagen 7.6, se muestra la el diagrama de conexión y la obtención de la señal de salida del sensor.

Figura 7.6 Diagrama de conexión de CNY70.

En la imagen 7.7 se muestra la asignación de pines del sensor CNY70.

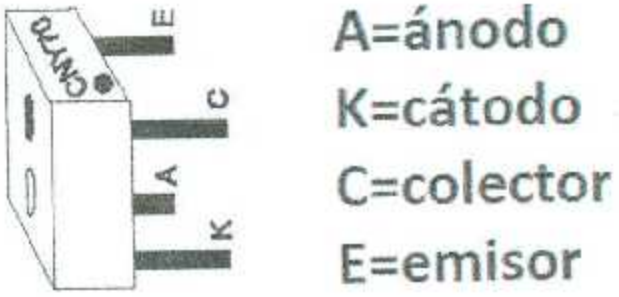


Figura 7.7 Asignación de pines de CNY70.

7.4.1.1 Principio de funcionamiento del sensor CNY70

El CNY70 devuelve por el pin de salida de señal un voltaje relacionado con la cantidad de luz reflectada por el objeto. Es decir, se leerá del emisor un '1' cuando se refleje luz y un '0' cuando no se refleje.

Para lograr esto se hizo un barreno en las aspas del motor y se colocó un led emisor de luz blanca, cuyo haz refleja directamente en el sensor CNY70 cada vez que gira el aspa del motor y pasa por enfrente del sensor CNY70, imagen 7.8.

Figura 7.8 Aspa del motor y barreno de visión.

Es necesario conectar la salida del sensor CNY70 a un inversor disparador de Schmitt 74HCT14, para obtener una señal más cuadrada que la que proviene directamente del sensor óptico.

El disparador de Schmitt usa la histéresis para prevenir el ruido que podría tapar a la señal original y que causaría falsos cambios de estado si los niveles de referencia y entrada son parecidos, en otras palabras, la función del 74HCT14 es evitar cambios involuntarios de los niveles de tensión de una señal provocado por variaciones en el ruido, donde los niveles de referencia pueden ser controlados ajustando las resistencias R_1 y R_2, imagen 7.9.

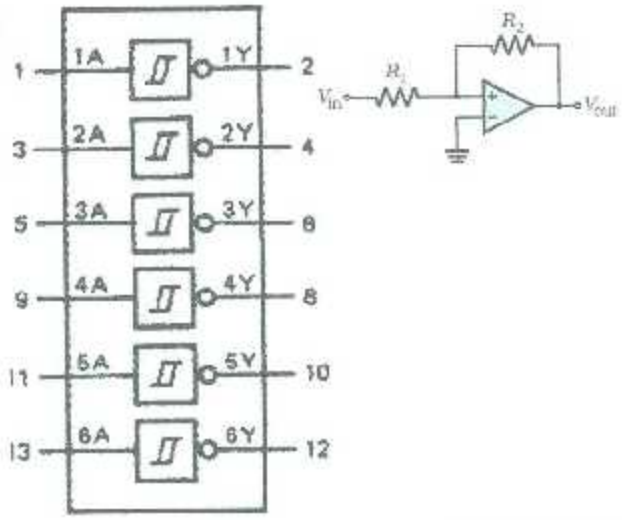


Figura 7.9 Diagrama de conexión de disparador de Schmitt 74HCT14.

La salida del disparador de Schmitt 74HCT14, se introduce a una etapa de acoplamiento de señales, de la cual se hablará más adelante, y de la etapa de potencia se introduce la señal a una entrada digital de la tarjeta FPGA Spartan-3E. Seguido a eso se contabiliza la cantidad de pulsos ya sean positivos '1' o negativos '0', que de acuerdo a la definición de Hertz la cual nos indica, que la cantidad de ciclos ocurridos en un segundo equivale a un Hertz, por lo tanto para nosotros un ciclo es equivalente a una revolución del aspa del motor, por lo tanto se obtiene la frecuencia de giro del motor registrando la cantidad de pulsos que la Spartan-3E detecta en un segundo.

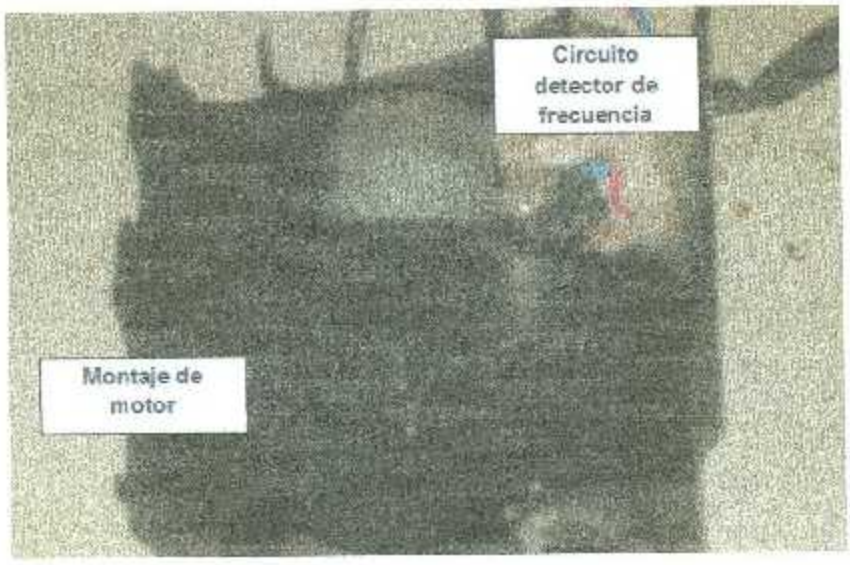


Figura 7.10 Montaje del circuito detector de frecuencia.

7.4.2 PWM: Modulación por ancho de pulso

La modulación por ancho de pulsos (PWM por sus siglas en ingles *Pulse-Width Modulation*) de una señal es una técnica en la que se modifica el ciclo de trabajo de una señal periódica, en la cual la frecuencia se mantiene fija.

Posteriormente, la señal de control PWM es llevada al *Driver* (se implementaron dos *Drivers*, uno para cada motor) que a su vez controla la velocidad de giro del eje del motor, dependiendo

del valor de PWM que reciba, más adelante se explicará la conexión y funcionamiento de los *Drivers* utilizados.

La señal de control PWM es generada y controlada con la tarjeta FPGA Spartan-3E. En el este proyecto de investigación la frecuencia de trabajo se estableció a 25 KHz.

El ancho de pulso de la señal PWM determinará la velocidad de giro del motor. La cual será proporcional al promedio de corriente que el respectivo *Driver* entregue al motor para que pueda alcanzar la velocidad deseada, figura 7.11.

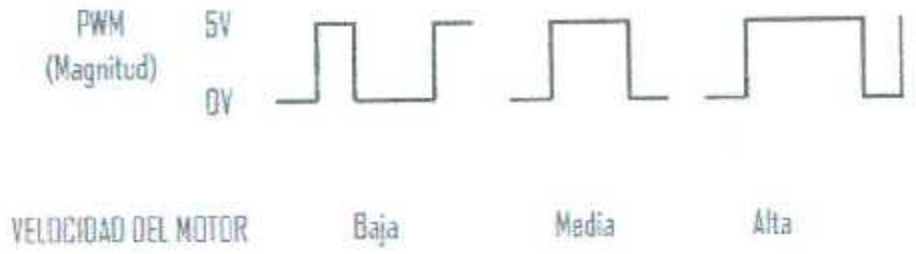


Figura 7.11 Magnitud del ciclo de trabajo de la señal PWM.

En la imagen 7.12, se muestra como al ir aumentado el ciclo de trabajo, por ejemplo hasta el 50% a una frecuencia de 25 KHz, el voltaje que estará recibiendo el motor será de 2.5 Volts, aproximadamente, debido a que los motores en el montaje del prototipo están siendo alimentando con 5V de DC.

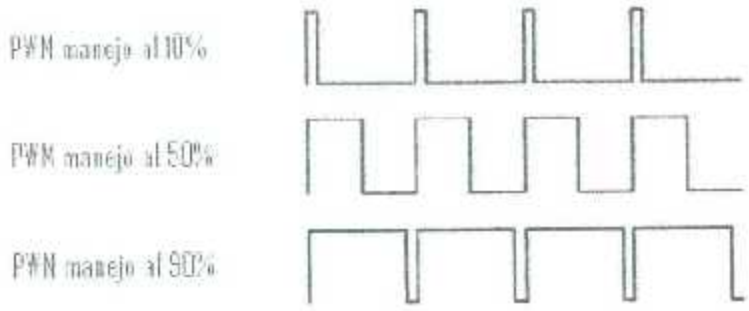


Figura 7.12 Variación del ciclo de trabajo.

7.4.3 Sensor ultrasónico SFR05

Los sensores de ultrasonidos son detectores de proximidad que trabajan libres de roces mecánicos y que detectan objetos a distancias de hasta 8m. El sensor emite impulsos ultrasónicos. Estos reflejan en un objeto, el sensor recibe el eco producido y lo convierte en señales eléctricas.

Los sensores ultrasónicos emiten un sonido, el cual es reflejado por un objeto, en donde el tiempo que la señal tarda en regresar al sensor es equivalente a la distancia recorrida por el sonido, que es la misma distancia que existe entre el sensor y el objeto.

Ultrasonido, hace referencia a las frecuencias arriba de 20KHz (límite de sonido audible). La generación y lectura de ultrasonido se hace a través de dos unidades piezoeléctricas en donde una de ellas es el emisor y la otra el receptor de ondas de presión ultrasónicas.

El módulo SRF05 tiene un rango de medida de 3 cm hasta los 4 metros de distancia.

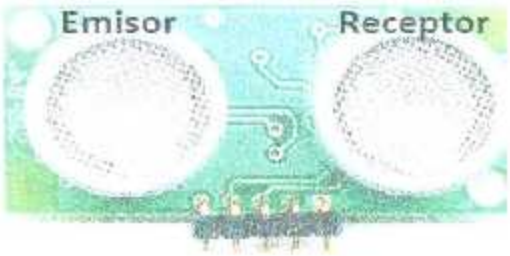


Figura 7.13 Sensor ultrasónico SFR05.

7.4.3.1 Conexionado

La manera de conectar este sensor, es relativamente sencilla, ya que solamente es necesario hacer la alimentación TTL, medir el tiempo que ocurre entre la emisión y recepción del sonido y generar un pulso de disparo para tomar una nueva medida, donde los últimos dos puntos son

generados y controlados por la tarjeta FPGA Spartan-3E, dicha programación se explicará mas adelante.

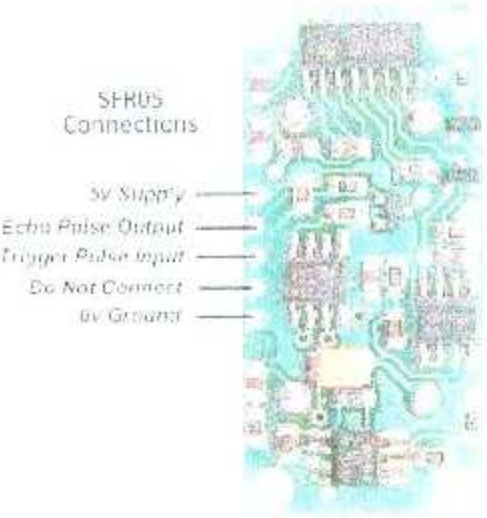


Figura 7.14 Conexiones del sensor SFR05.

En la imagen 7.14, se muestra la asignación de pines del sensor, a continuación se explicará la función de cada pin:

- 5V Supply :Tensión positiva de alimentación
- Echo Pulse Output :Salida del pulso cuya anchura determina el tiempo del recorrido de la señal ultrasónica
- Trigger Pulse Output : Entrada de inicio de una nueva medida. Se aplica un pulso con una duración mínima de 10µs.
- Modo (N.C.) : Sin conexión
- 0V Ground : Tierra de alimentación.

En la imagen 7.15, se muestra el sensor montado en la estructura del prototipo, en la parte superior del tubo de acrílico.

Figura 7.15 Montaje y conexión sensor SFR05.

7.5 Desarrollo de la etapa de potencia

La importancia de esta etapa radica en que nos sirve para aislar físicamente la etapa de control y la etapa de potencia, ya que si alguna de las cargas conectadas al circuito sufre de un corto circuito o demanda un consumo de corriente mayor a la que el circuito de control entrega, puede provocar un daño de forma permanente en el circuito de control

Además de servir para hacer un acoplamiento entre los voltajes de las señales, ya que la etapa de control que corresponde a la tarjeta FPGA Spartan-3E maneja tanto en sus periféricos de entrada como de salida un voltaje de 0 - 3.3 V y tanto los *Drivers* de control de los motores como los sensores instrumentados manejan voltajes de 0 - 5 V.

En la etapa de potencia se usó el opto acoplador de rápida respuesta, el SFH6135, ya que su rango de frecuencia de trabajo soporta hasta 2 MHz, figura 7.15.

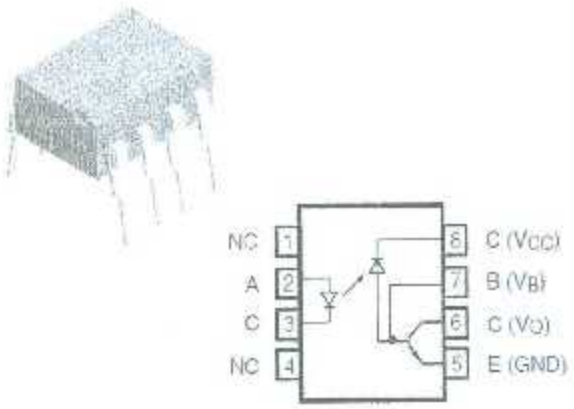


Figura 7.16 Diagrama de conexión interna de SFH6135.

Se tomó la decisión de hacer en un mismo PCB la etapa de potencia de todos los sensores y actuadores, con el objetivo de ahorrar espacio en el montaje del mismo, así como de facilitar el acceso a las conexiones. Se explicará cada sección de conexión del PCB.

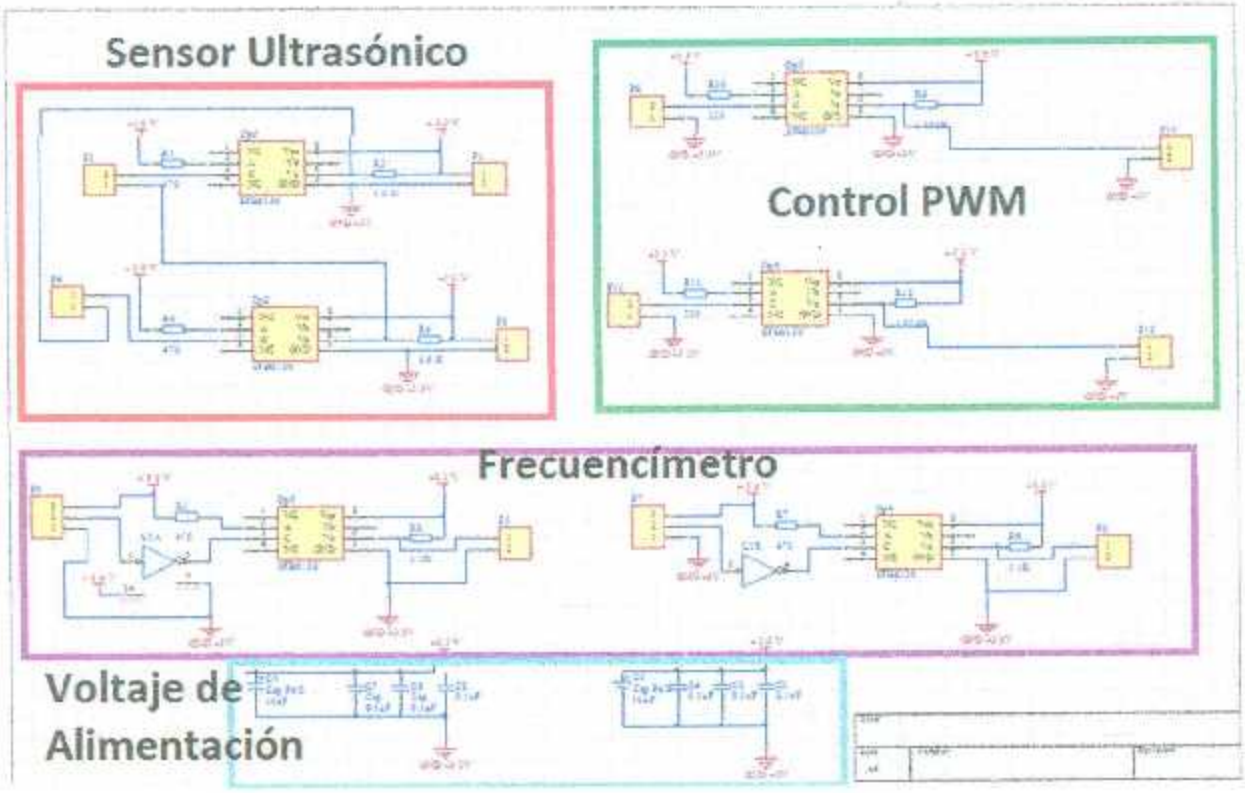


Figura 7.17 Diagrama esquemático correspondiente a la etapa de potencia

En la imagen 7.17, se encuentra seccionada en cuatro partes, donde cada una de ellas corresponde al acoplamiento de señales entre el sensor ultrasónico y la tarjeta FPGA Spartan-3E, al control PWM de cada motor, así como la etapa de potencia de los frecuencímetros montados en los motores y finalmente el voltaje de alimentación del PCB, cada sección se explicará a continuación.

7.5.1 Etapa de potencia del sensor ultrasónico

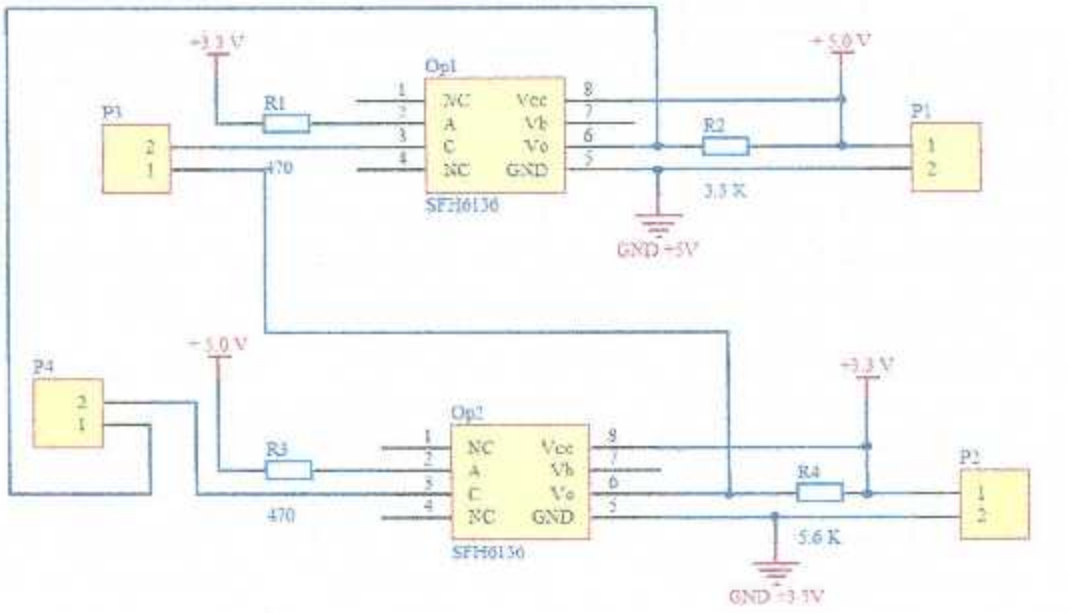


Figura 7.18 Diagrama esquemático de etapa de potencia sensor SFR05.

La nomenclatura a utilizar para hacer la descripción del diagrama esquemático de la figura 7.18, es la siguiente:

Op#_P#_#= Numero de operacional _Numero borne _Numero de terminal en el borne.

La abreviación Op se refiere al opto acoplador de alta respuesta SFH6135, y la abreviación P indica el número de borne.

La tarjeta Spartan-3E debe de generar un pulso de disparo (Trigger) hacia el sensor ultrasónico para indicar que deberá tomar una nueva lectura, por lo tanto en Op1__P3_2 se conecta la señal de disparo emitida por la tarjeta FPGA a 3.3 V. En Op1_P3_1 recibe la señal de eco proveniente del SFR05 a 3.3 V, la cual es indicador de la distancia entre el sensor y el objeto.

En Op2_P4_2 corresponde a la señal de disparo (Trigger) a 5V y Op2_P4_1 recibe la señal de ECO del sensor SFR05 a 5V.

7.5.2 Etapa de potencia del frecuencímetro

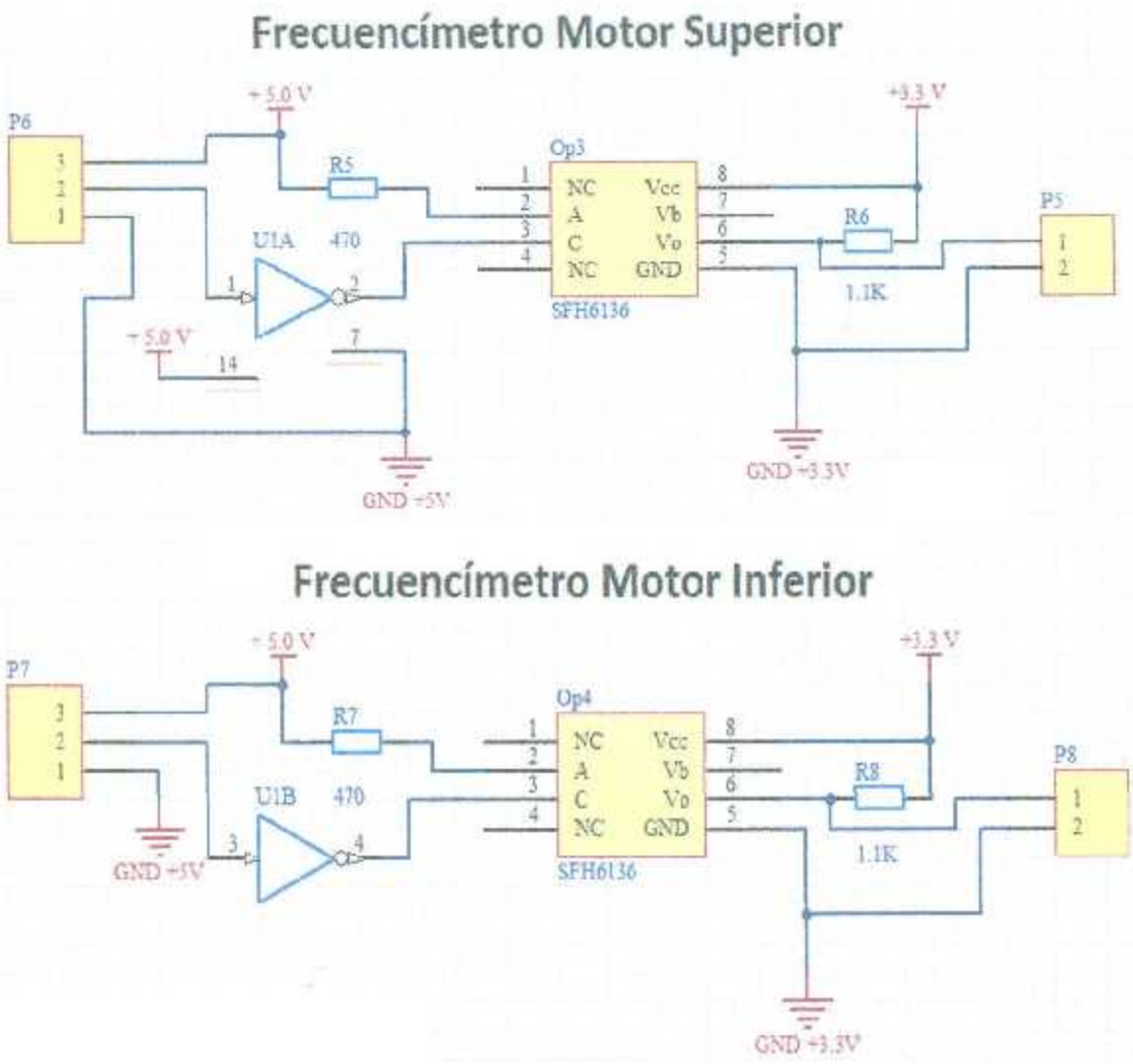


Figura 7.19 Diagrama esquemático de etapa de potencia de frecuencímetro

Se seguirá la misma nomenclatura mencionada previamente.

La señal percibida por el sensor CNY70 del sensor superior a 5V es dirigida al Op3_P6_2 y la frecuencia detectada del motor inferior es conectada al Op4_P7_2 de igual forma a 5V.

Para acoplar los voltajes a 3.3V que es el valor que recibe la tarjeta FPGA Spartan-3E se conecta la señal del frecuencímetro de motor superior a Op3_P5_1 y del motor inferior a Op4_P8_1 y de estas terminales directo a la FPGA. La conexión del disparador de Schmitt es representada por U1A y U1B.

7.5.3 Etapa de potencia para la señal PWM

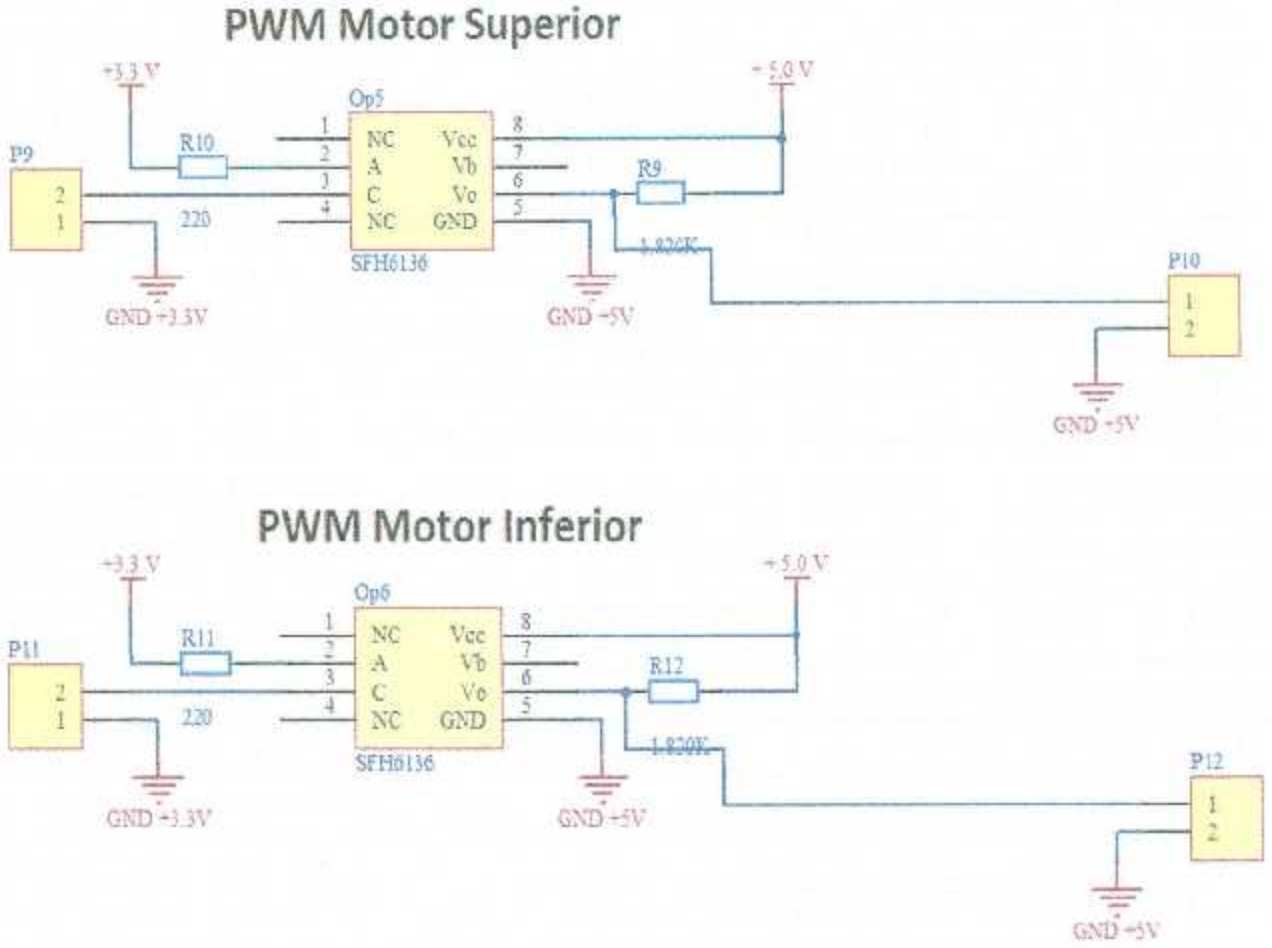


Figura 7.20 Diagrama esquemático de etapa de potencia PWM.

La señal de control PWM generada por la tarjeta FPGA Spartan-3E es conectada a Op5_P9_2 a un voltaje de 3.3V del motor superior y para controlar el motor inferior se utiliza Op6 P11_2. Pero a los *Drivers* de control de los motores es necesario elevar el voltaje a 5V, por lo tanto se utilizarán las terminales Op5_P10_1para control del motor superior y para el inferior es Op6_P12_1.

En la imagen 7.21, se muestra el PCB (*Printed Circuit Board*), el cual incluye todos los circuitos mencionados anteriormente.

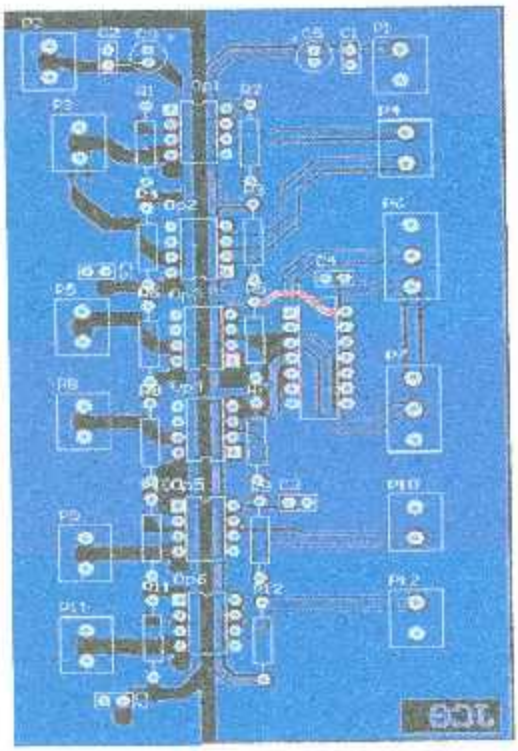


Figura 7.21 PCB de etapa de potencia.

En la imagen 7.22, se muestra el PCB ya montado y conectado en el prototipo

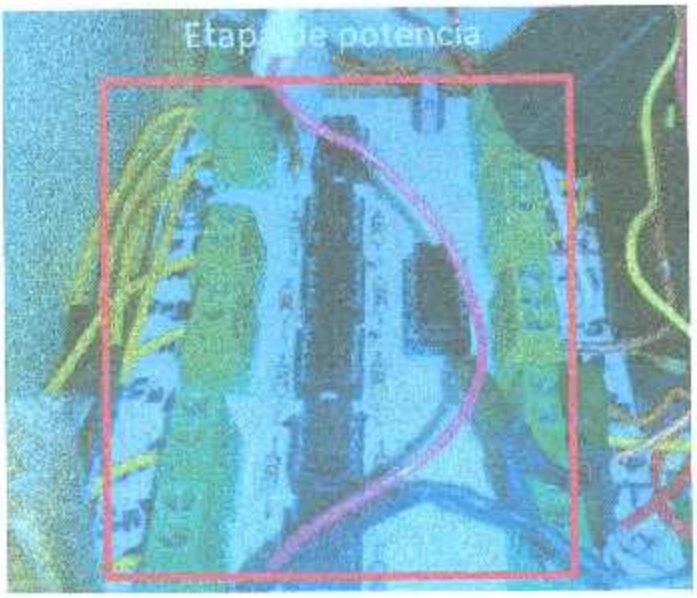


Figura 7.22 Montaje y conexión del circuito de etapa de potencia.

7.5.4 Control de la velocidad del motor inferior.

Para el control de la velocidad del motor inferior se utilizó el L298D, el cual soporta cargas de entre 5 a 46 VDC proporcionando una corriente máximo de 2A pero si se conecta en paralelo puenteando las salidas es posible alimentar un motor cuyo consume de corriente alcance los 4A.

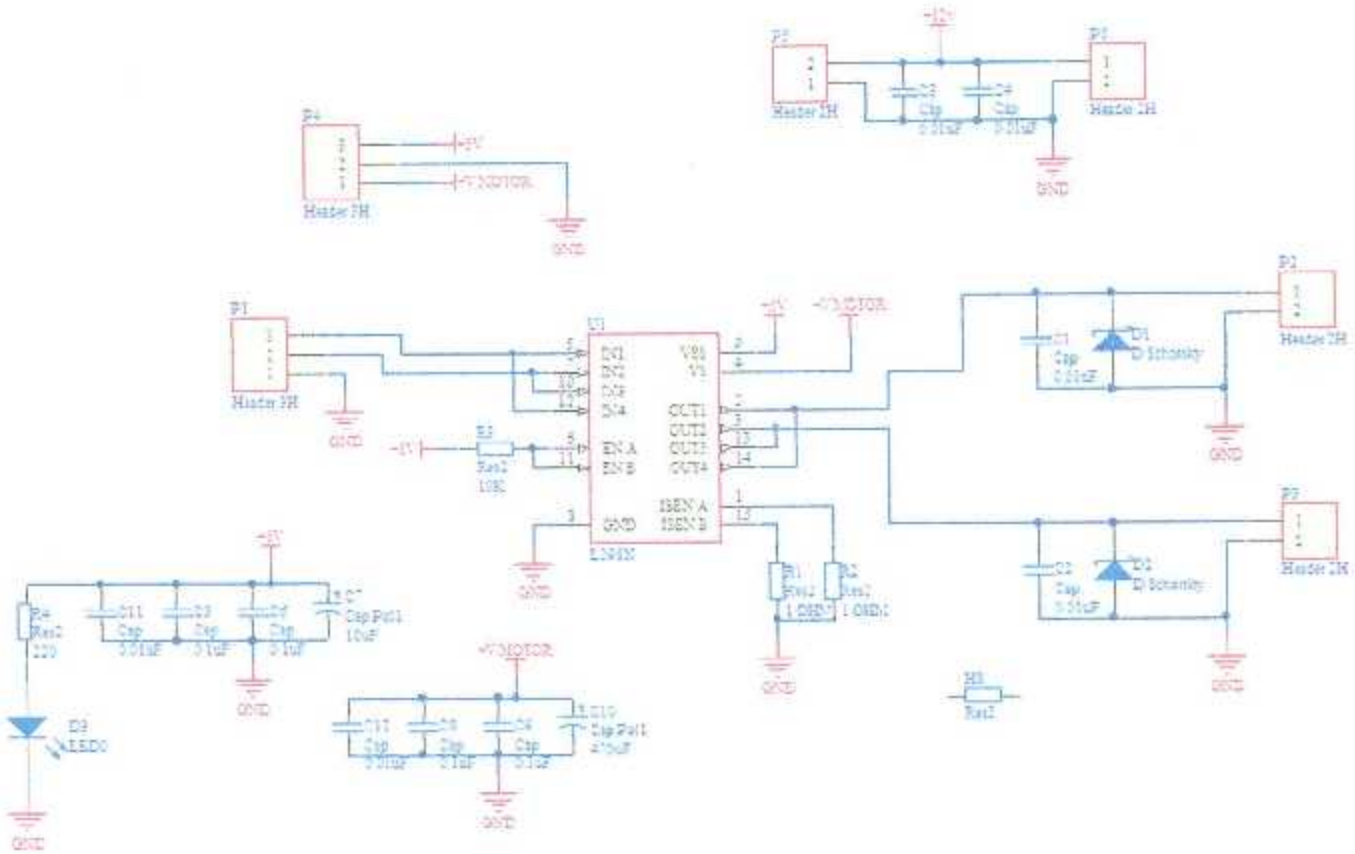


Figura 7.23 Diagrama de conexión del *Driver* L298D.

Es alimentado con 12VDC en los bornes P5_2 y P6_1, en P5_1 y P6_2 es GND. Se utiliza la nomenclatura especificada anteriormente.

En el U1_P1_3 y U1_P1_2 son las señales PWM del motor superior e inferior, respectivamente. La señal de control PWM es generada por la tarjeta FPGA a un voltaje de 3.3 VDC. El motor superior es conectado a U1_P2_1 y el motor inferior a U1_P3_1.

Se conecta un disipador, con el objetivo de evitar que el *Driver* aumente demasiado su temperatura.

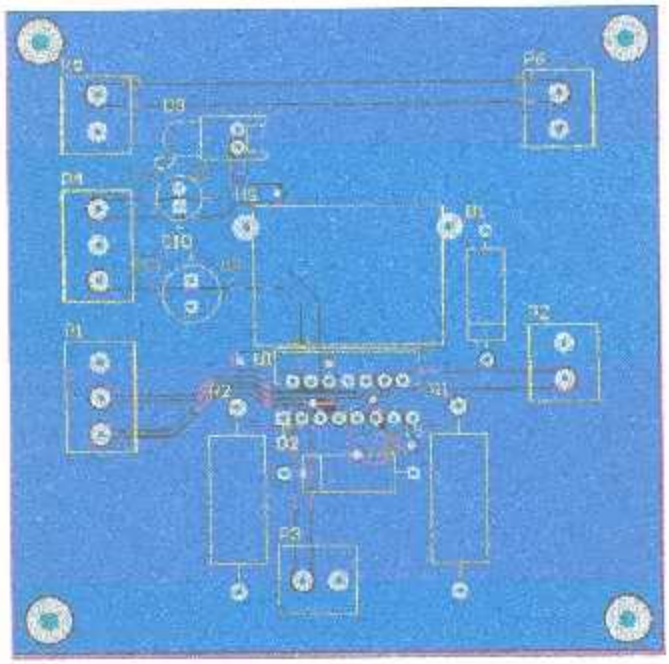


Figura 7.24 PCB del circuito para el control de la velocidad del motor inferior.

En la imagen 7.25, se muestra el montaje y conexión del PCB para el control PWM del motor inferior.

Figura 7.25 Montaje y conexión del circuito para el control de la velocidad del motor inferior.

Debido a que los transistores internos del puente H del L298D consumen un total de 1.4 VDC, repercutiendo en que el voltaje máximo que le llega al motor es de aproximadamente 3.6 VCD, por lo tanto fue necesario alimentar el voltaje de la carga, es decir al motor con 6.5 VDC para que el motor a un ciclo de trabajo del 100% consuma aproximadamente 5Volts, para esto fue necesario implementar un circuito regulador. En la imagen 7.26, se muestra el diagrama de conexión.

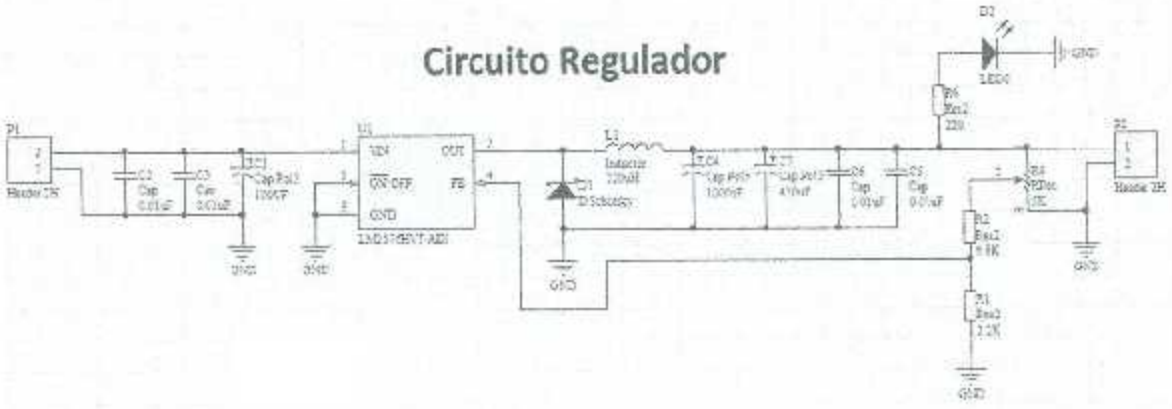


Figura 7.26 Diagrama esquemático del circuito regulador.

Se utilizó el LM2576, el cual limita el voltaje de 12 VDC a un voltaje de 5-7 VDC. En la figura 7.27 se muestra el circuito PCB.

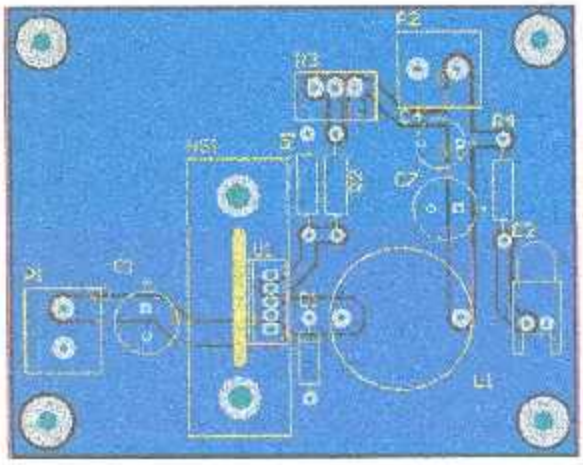


Figura 7.27 PCB de circuito regulador.

7.5.5 *Driver* MD03

Para el control del motor superior se implementó el *Driver* MD03, el cual está diseñado para el control de un motor, tiene diversos modos de control y la capacidad para entregar a la carga que se le conecte hasta 20A para motores desde 5-24 VDC.

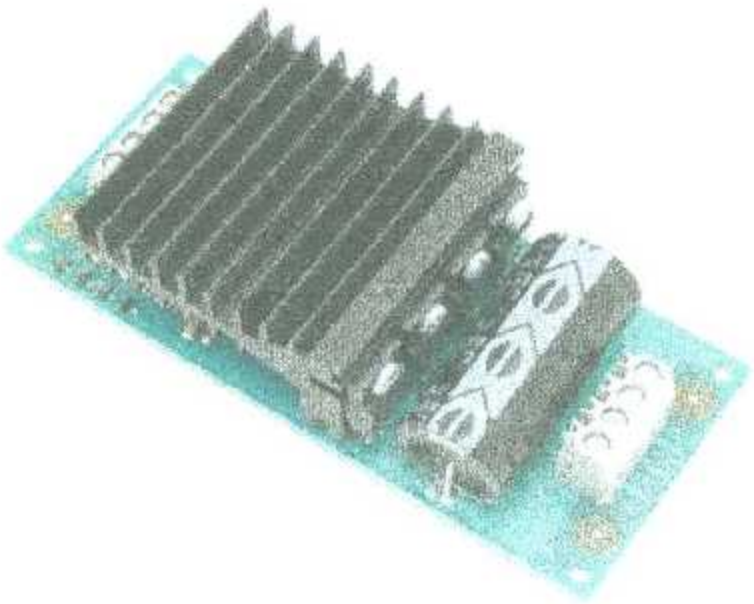


Figura 7.28 *Driver* MD03

Los modos de control son los que se explican en la tabla 7.1.

Tabla 7.1. Especificaciones de funcionamiento del MD03.

Especificaciones	
Voltaje	La alimentación de los integrados es de 5 VDC y el voltaje de alimentación de la carga es desde 5-24 VDC
Corriente	En la parte de lógica la corriente máxima que entrega es de 50 mA y para el motor hasta 20 A máximo.

Modo 1	0- 2.5-VDC para un voltaje análogo, donde 0V indica giro hacia atras, 2.5 V es para detener el giro del motor y 5V es para que el motor gire hacia adelante
Modo 2	Control PWM de 0-5 VDC, a una frecuencia superior a 20 KHz, con control de sentido de giro.
Modo 3	Radio control
Modo 4	Interfaz I2C, control de la velocidad, monitoreo de la temperatura y consumo de corriente del motor.
Limitador de corriente	El *Driver* entrega un máximo de 20 A
Limitador de temperatura	Si el motor sufre de sobrecalentamiento el *Driver* se apaga para evitar daño permanente en el MD03.

El modo de trabajo seleccionado es el modo 2, donde la señal PWM es generada por la Spartan-3E a una frecuencia de 25 KHz. En la figura 7.29, muestra el diagrama interno del *Driver*.

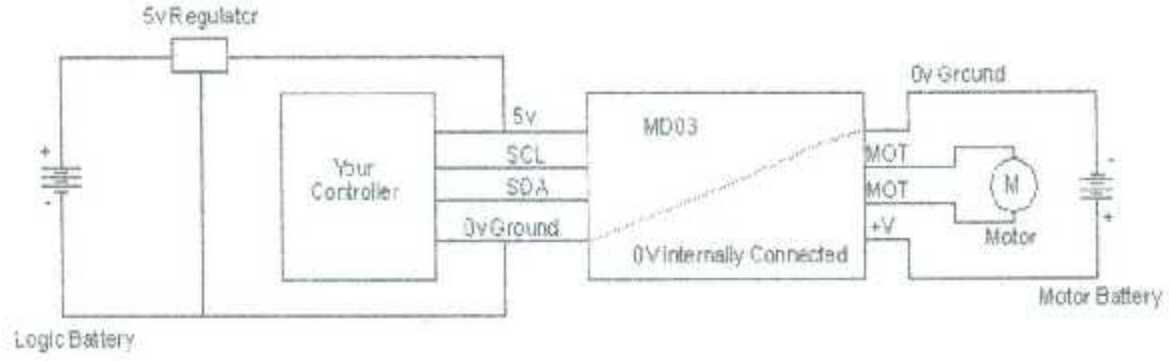


Figura 7.29 Diagrama de conexión del *Driver* MD03.

La señal SCL es la que determina el sentido de giro dependiendo si se le conecta 5 VDC o GND. La señal PWM es enviada a MD03 por la línea de control SDA. La terminal +V es el voltaje de alimentación de la carga, en este trabajo el motor superior trabaja hasta un máximo de 5 VDC.

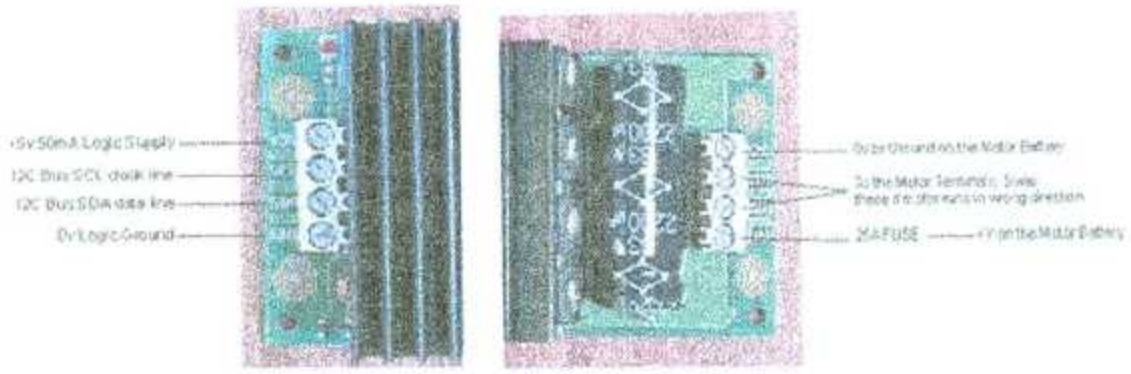


Figura 7.30 Conexión del *Driver* MD03

En la figura 7.31, se muestra el montaje y conexión del MD03 en el prototipo.

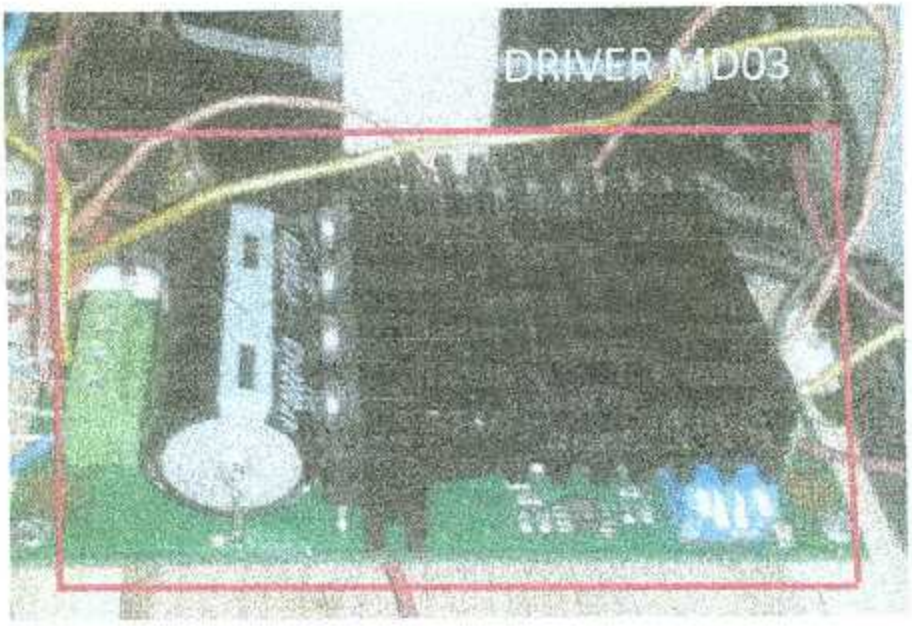


Figura 7.31 Montaje y conexión del *Driver* MD03.

Para facilitar la conexión tanto de la tarjeta FPGA Spartan-3E y de la fuente de poder que alimenta los circuitos electrónicos, se decidió colocar un enchufe en el interior del prototipo, figura 7.32.

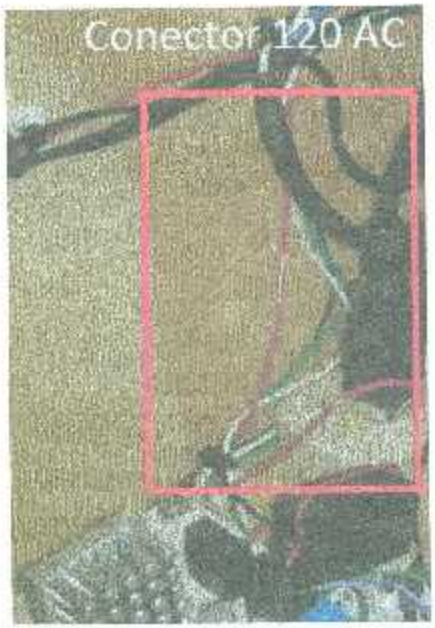


Figura 7.32 Montaje de conector 127 AC.

En la imagen 7.33, se muestra la vista posterior del prototipo, donde la flecha señala el cable del enchufe que se montó en la imagen 7.32.

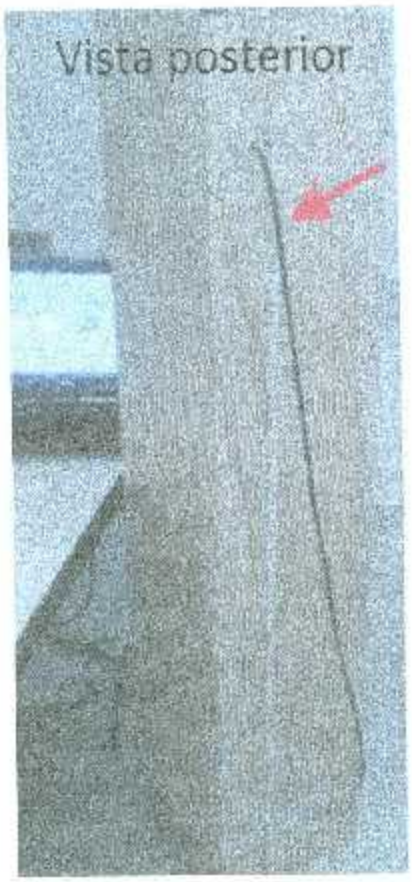


Figura 7.33. Vista posterior de estructura.

7.5.6 Montaje de tarjeta FPGA

En la figura 7.34. se muestra la tarjeta FPGA Spartan-3E montada y conectada al prototipo.

Figura 7.34 Conexión de la tarjeta Spartan 3E-*Starter Kit*.

La figura 7.35, es una imagen de la vista superior de la estructura.

Figura 7.35 Vista superior de prototipo

La figura 7.36 muestra la unión de la etapa de control y monitoreo con la estructura.

Figura 7.36 Conexión entre el sistema de control y monitoreo al prototipo.

Capítulo VIII. Herramientas de *software* preliminares

8.1 Introducción

En el desarrollo del capítulo, se abordan las diversas plataformas de programación con las que se pueden enlazar y programar las tarjetas FPGA, específicamente se trabajará sobre la tarjeta Spartan 3E- *Starter Kit* de Xilinx. Seguido de un tutorial en el cual se enlazarán dos metodologías de programación, donde cada una de ellas tiene características que las hacen únicas y eficientes en su ramo y que al unirse hacen una herramienta extremadamente poderosa y eficiente, refiriéndonos a la integración entre el lenguaje de programación grafico LabVIEW 2011 con el lenguaje de programación de *hardware ISE Design suite* 14.1 de Xilinx [15].

Se consideró que valía la pena incluir dicho tutorial, ya que es una forma de hacer una programación de gran alcance, además que a mi particular punto de vista no está lo suficientemente documentado y por lo tanto los posibles lectores de este trabajo de investigación tendrán una herramienta adicional a su formación como ingenieros, desarrolladores de *software*, de firmware, de *hardware* ,etc. teniendo una idea más clara de los alcances que la programación de las tarjetas FPGA puede tener, en diversas ramas de la industria, dependiendo de la aplicación para la cual se diseñe.

8.2 Software

La tarjeta de desarrollo SPARTAN 3E, tiene las características necesarias para enlazarse con las siguientes plataformas de programación:

Figura 8.1 Plataformas de programación que se enlazan con las tarjetas FPGA.

Enseguida se hablará brevemente de las herramientas que se exploraron durante el desarrollo de este trabajo de investigación, ya que antes de tomar la decisión en cuanto a con que lenguaje de programación se trabajaría, se decidió hacer una breve indagación sobre los beneficios e inconvenientes que cada plataforma presentaba, además de cuál de ellas se adaptaba mejor a la aplicación que este trabajo de tesis documenta.

8.2.1 Xilinx *Ise Desing Suite*

Figura 8.2 Logotipo de *Ise Design Suite*.

Los sistemas electrónicos reconfigurables FPGA son un buen ejemplo de la complejidad al programar *hardware*, dicha complejidad no sería alcanzable sin la ayuda de un entorno con herramientas que asistan en el proceso de diseño, simulación, síntesis del resultado y configuración del *hardware* [16].

La herramienta que se utiliza en este proyecto es *Ise Desing Suite* 14.1 (o por siglas en inglés, *Integrated Software Environment*) del fabricante de tarjetas FPGA de Xilinx, en donde el lenguaje de programación utilizado es VHDL, este es un lenguaje de descripción de *hardware* (*Hardware Description Language*).

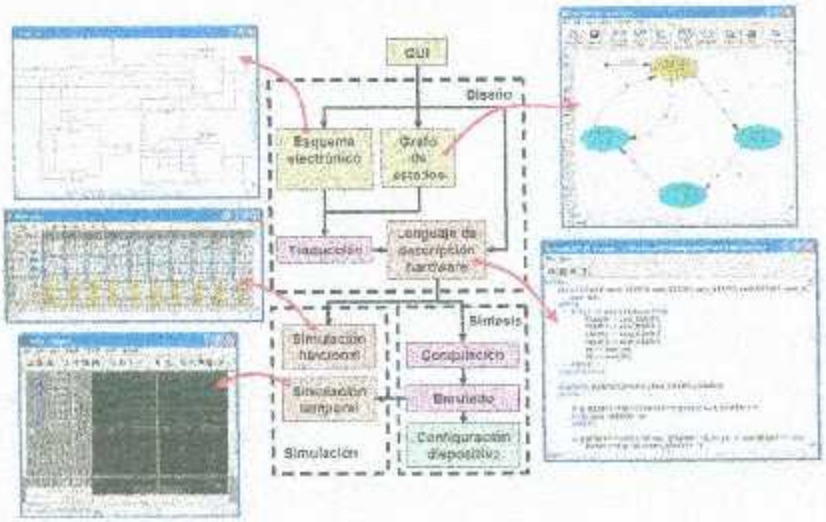


Figura 8.3 Entorno de programación de *Ise Desing Suite*.

El entorno de programación EDA (Electronic Desing Automation). La Figura 8.3 representa el esquema de los componentes más importantes del ISE y la secuencia en que se utilizan.

Las plataformas de programación SDK y XPS también nos las brinda el mismo fabricante Xilinx. Estas herramientas nos ayudan a empotrar un micro de forma virtual en la FPGA, es decir, desarrollar un micro por *software*.

Xilinx Software Development Kit

Figura 8.4 Logotipo de SDK.

Se determinan las características de *hardware* que tendrá el micro.

Xilinxs Platform Studio

Figura 8.5 Logotipo de XPS.

Nos permite hacer la programación del micro, la cual puede ser en C o C++.

Xilinx Vivado Desing Suite

Figura 8.6 Logotipo de Vivado.

Es un compilador desarrollado por Xilinx, y lanzado al mercado en Abril 2012, que nos permite general diseños digitales a partir de descripciones realizadas en C/C++, es decir, partiendo de lenguajes de alto nivel (HLL – *High Level Languages*) a lenguajes de descripción de *hardware* (HDL - *Hardware desciption languages*), ver figura 8.7.

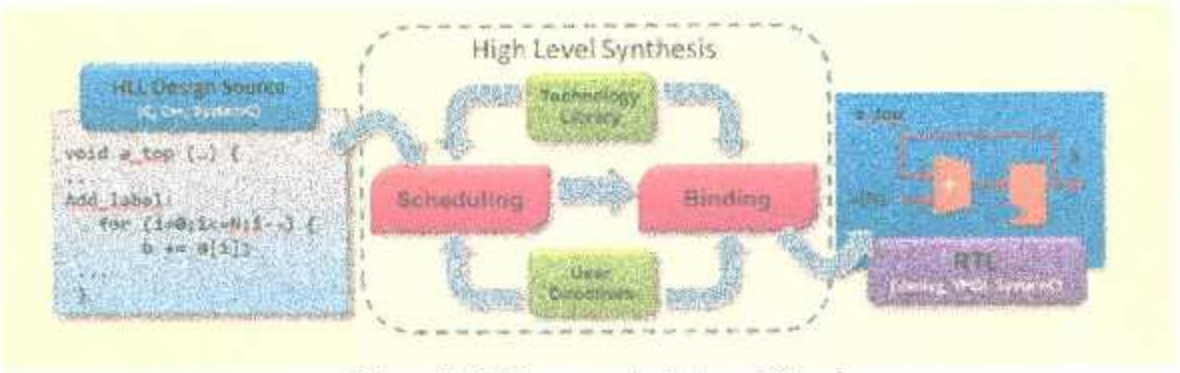


Figura 8.7 Diagrama de flujo en Vivado.

8.3 Matlab (2012)

Simulink HDL Coder

La plataforma de desarrollo Matlab, a través de Simulink, cuenta con una herramienta llamada HDL Coder, la cual nos permita programar las tarjetas FPGA utilizando los bloques que nos ofrece simulink.

Figura 8.8 Logotipo de Simulk HDL Coder, Matlab.

Xilinx nos permite enlazar los bloques existentes en Simulink a la tarjeta FPGA.

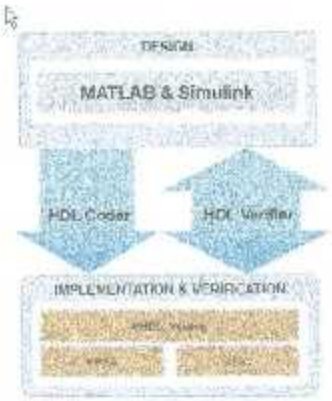


Figura 8.9 Diagrama de flujo entre HDL Coder de Simulink en Matlab conectada con lenguaje VHDL de la tarjeta FPGA.

En la figura 8.9 y 8.10, se aprecia el flujo de información y la manera en que se genera el archivo *.bit, el cual es el que se cargará en la tarjeta FPGA.

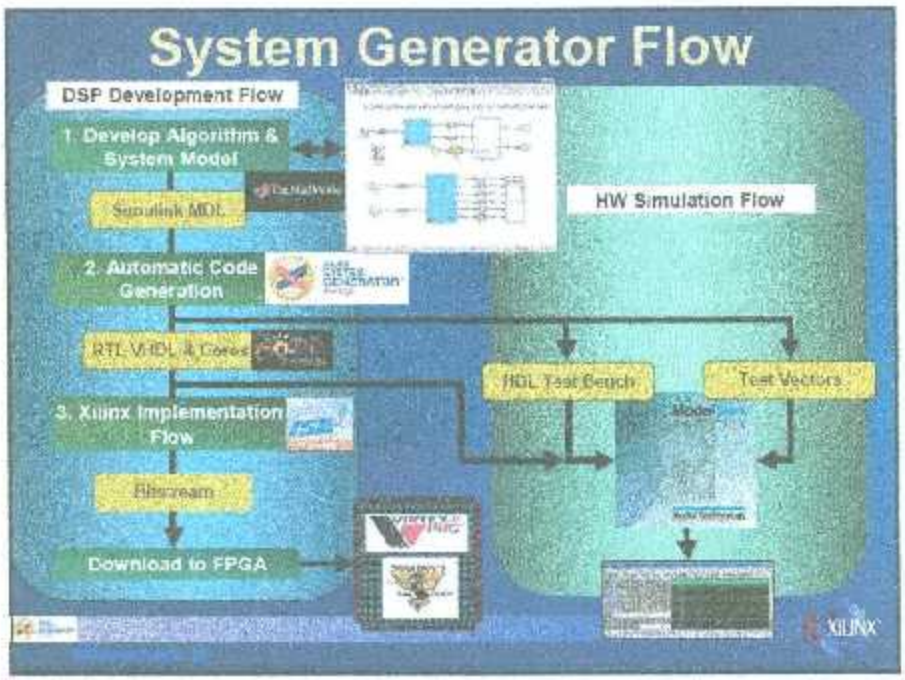


Figura 8.10 Procedimiento de generación de archivo *.bit por medio de Matlab [17].

8.3.1 Programación de la herramienta HDL Coder de Simulink.

Enseguida se hace una breve explicación de la forma en que funciona la herramienta HDL Coder.

1.- Inicializo Simulink.

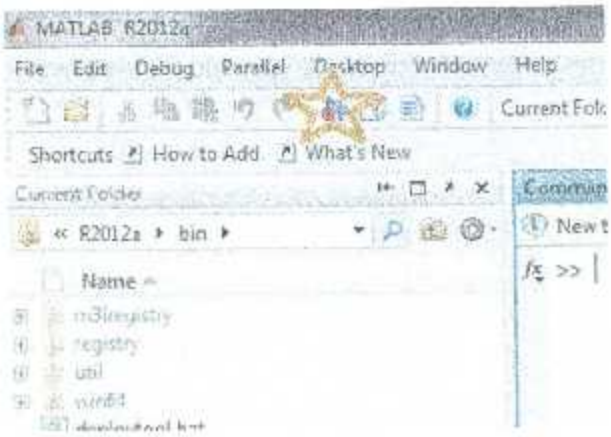


Figura 8.11 Inicio Simulink

2.- Al abrir Simulink, aparece la siguiente ventana, en la cual se cargan las diferentes tarjetas FPGA que MatLab Simulink HDL Coder soportan.

Figura 8.12 *Configuring System Generator*.

3.- Se muestra el toolbox de Xilinx

Seleccionamos el bloque de Xilinx Blockset

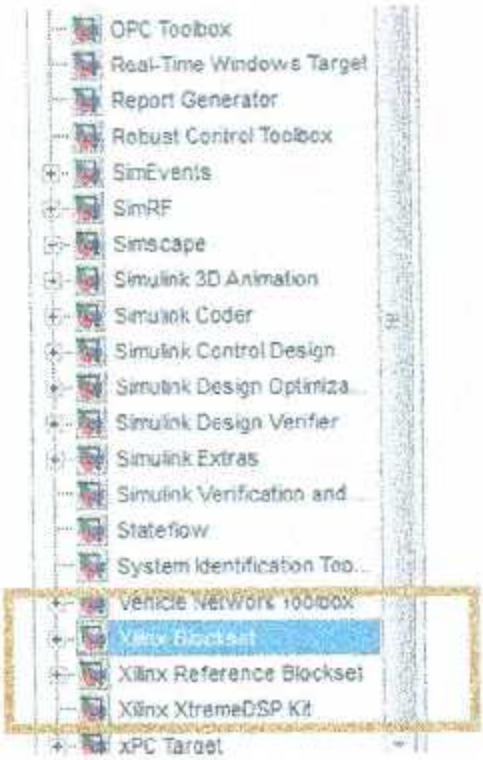


Figura 8.13. Selección de Xilinx Blockset.

Posteriormente, se abre un nuevo proyecto, y se agrega el bloque

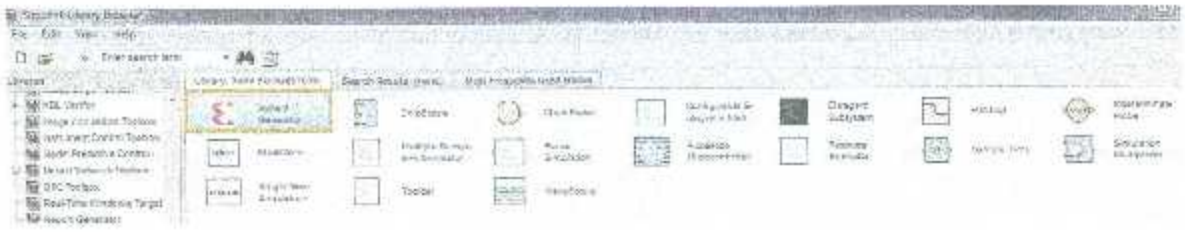


Figura 8.14 *System Generator*.

En el documento nuevo, agregamos el bloque *System generator*, para hacer la configuración de la tarjeta a utilizar.

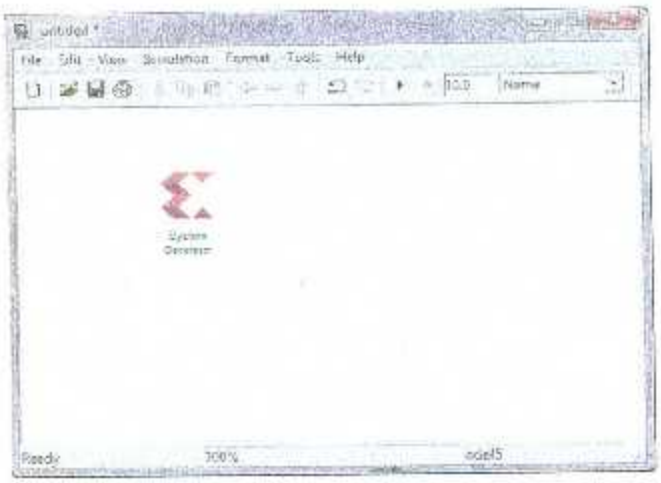


Figura 8.15 *System Generator* en documento nuevo.

>>Doble Click en *System Generator*

4.- Se selecciona la tarjeta a utilizar.

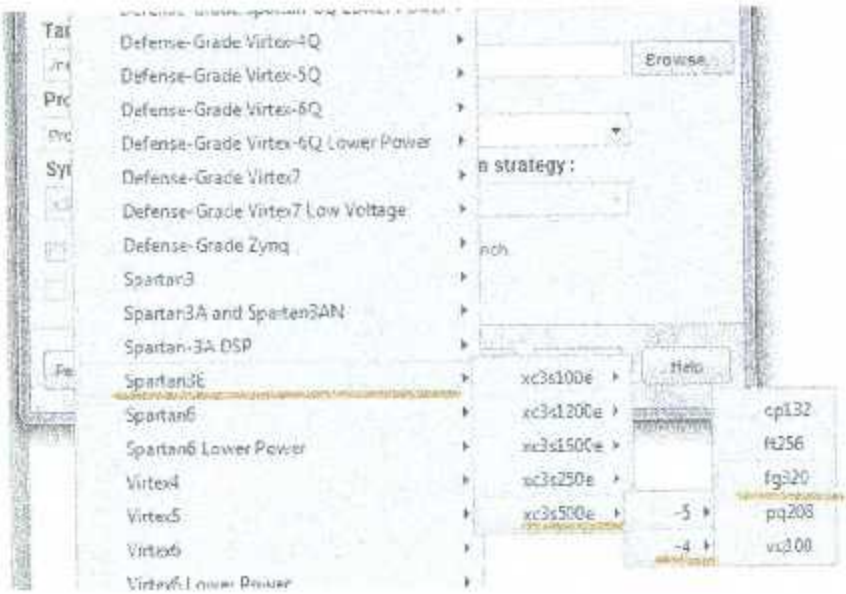


Figura 8.16 Configuración de tarjeta FPGA en bloque *System Generator*.

Después de haber hecho la configuración de la tarjeta FPGA, se continúa con la programación de la misma, en este tutorial no se aborda esa parte ya que el objetivo principal de este documento es ilustrar la programación entre LabVIEW 2011 e *Ise Design Suite* 14.1 de Xilinx.

8.4 LabVIEW 2011

LabVIEW es una herramienta gráfica para pruebas, control y diseño mediante la programación. El lenguaje que usa se llama lenguaje G, donde la G simboliza que es lenguaje Gráfico.

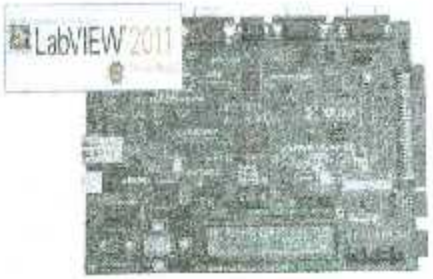


Figura 8.17 Logotipo de entorno LabVIEW 2011.

8.4.1 Instalación

1.- Cuando se instala el módulo ***NI LabVIEW FPGA MODULE***, en la ventana de inicialización del programa aparece la siguiente imagen, indicando que el módulo FPGA LV fue instalado correctamente.

Figura 8.18 Logotipo de LabVIEW2011 y Modulo FPGA

2.- Posteriormente es necesario instalar el *Driver* de la tarjeta FPGA en específico a utilizar directamente de la página de **National Instruments**.

Figura 8.19 Download LabVIEW FPGA for SPARTAN 3E XUP drive.

8.4.2 Programación

1. Iniciar un proyecto nuevo.

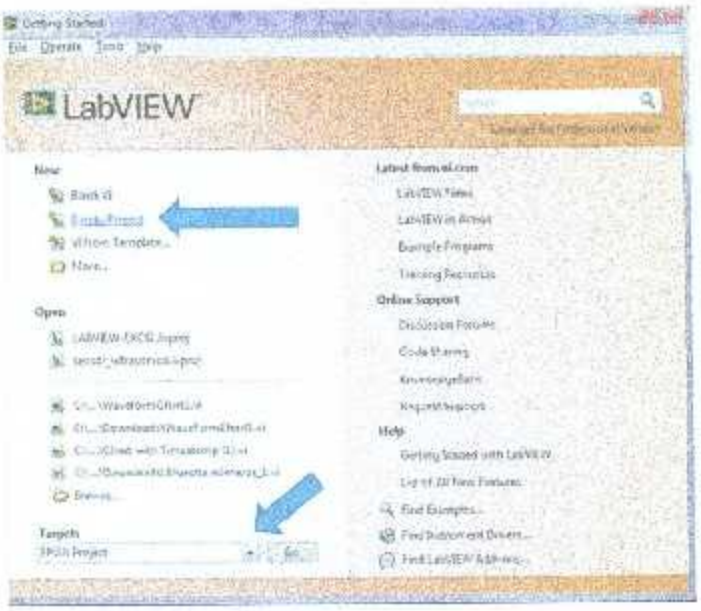


Figura 8.20 Pantalla de inicio.

2. Seleccionar la tarjeta FPGA a utilizar.

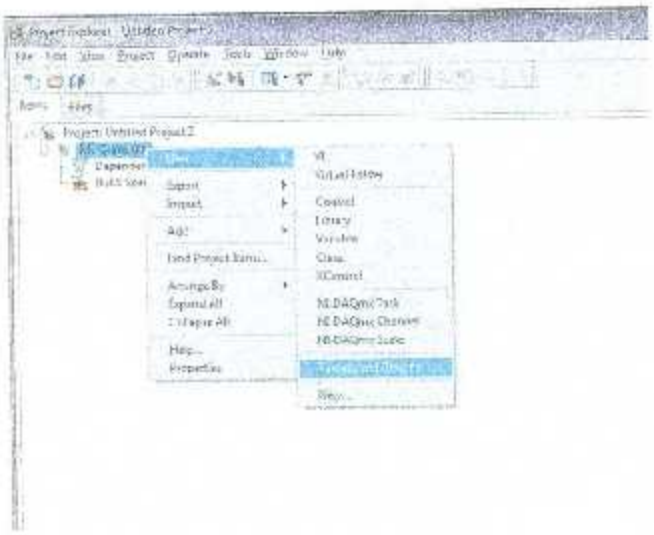


Figura 8.21 Selección de tarjeta FPGA.

3. Seleccionamos la tarjeta SPARTAN 3E *Started Board*

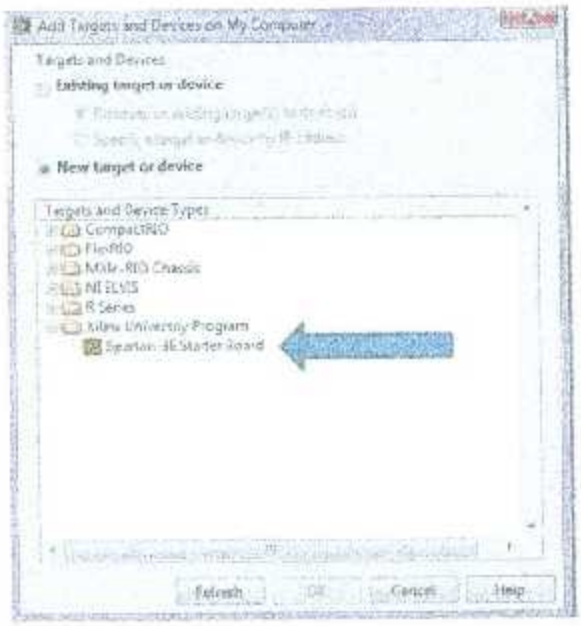


Figura 8.22 Selección de tarjeta FPGA, Spartan- 3E Starter *Board*.

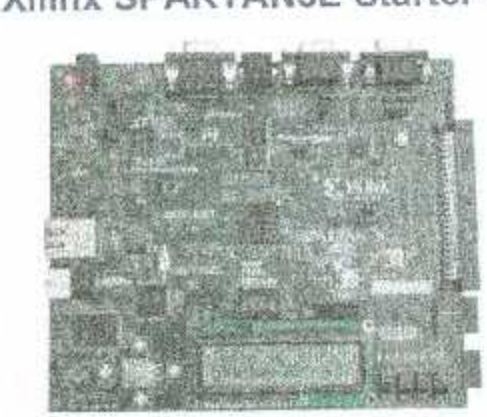


Figura 8.23 Spartan- 3E Starter *Board*.

En el *Project Explorer*, aseguramos que la tarjeta seleccionada fue agregada al proyecto.

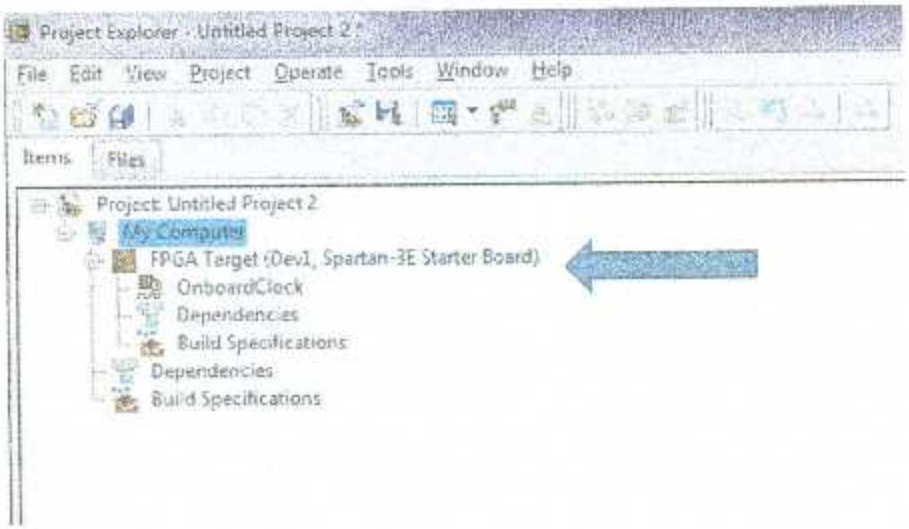


Figura 8.24 Vista de *Project Explorer*.

4. Agregar los periféricos necesarios para el desarrollo del proyecto.

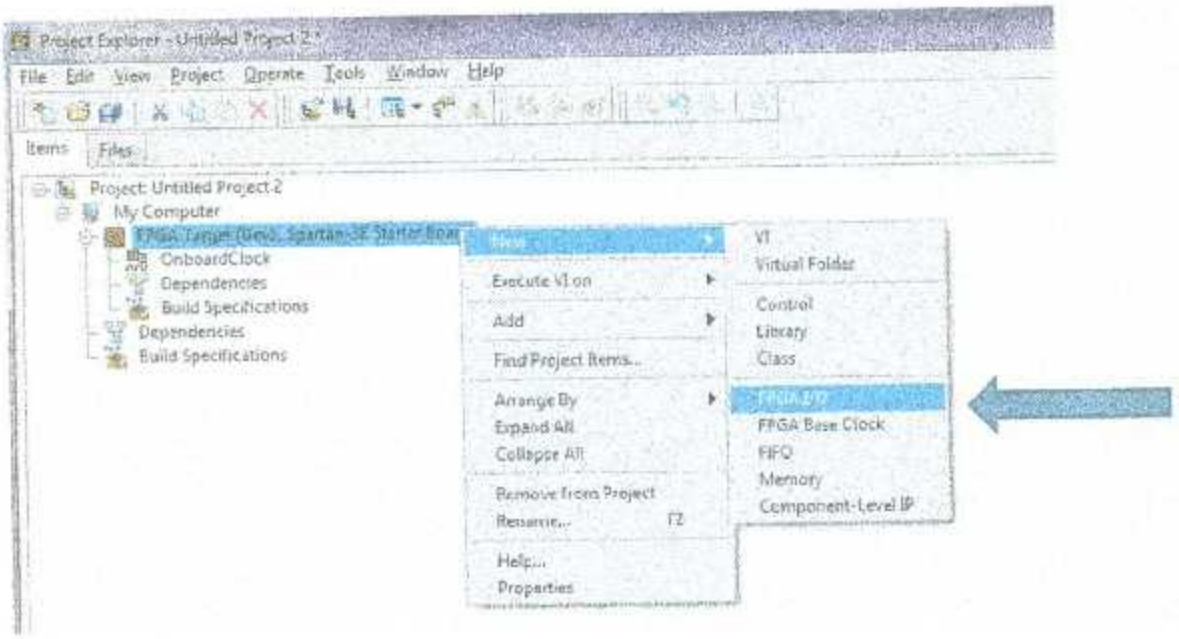


Figura 8.25 Selección de periféricos de entrada y salida

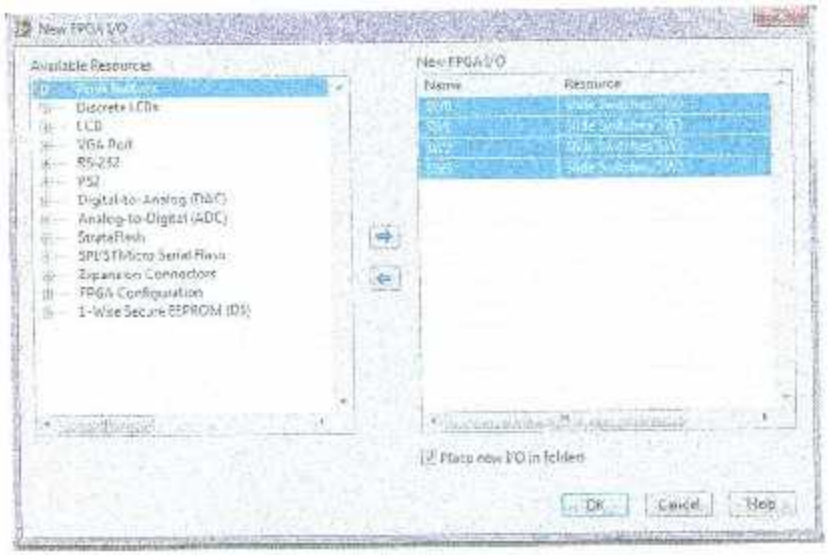


Figura 8.26 Periféricos disponibles en la tarjeta FPGA Spartan 3E

Se pueden agregar al proyecto bloques de memoria y FIFO´s.

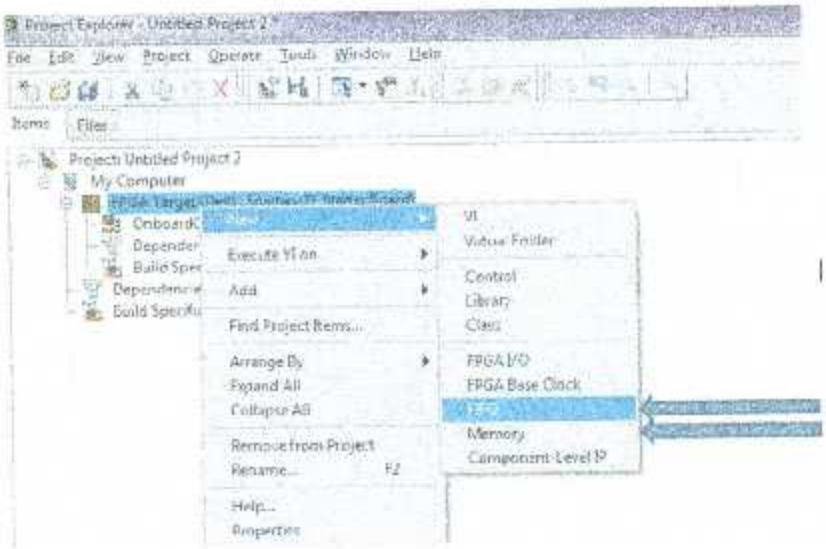


Figura 8.27 Bloques de Memoria y FIFO's

En el explorador del proyecto, se muestran los periféricos seleccionados.

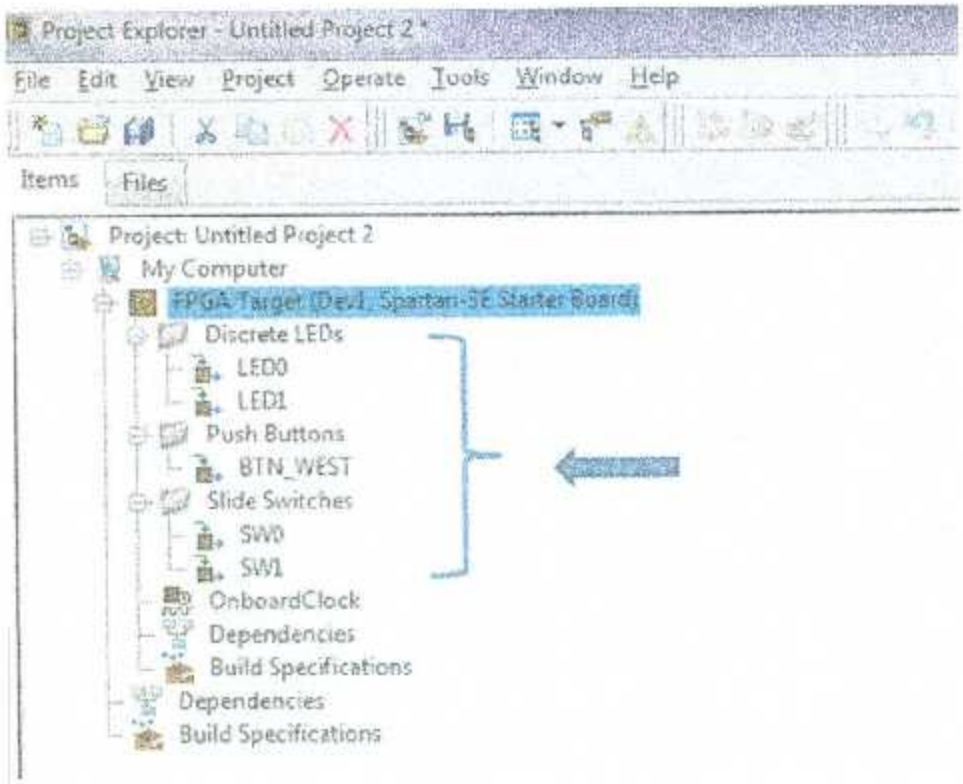


Figura 8.28 Periféricos seleccionados

5. Realizado lo anterior, agregamos un nuevo VI a la tarjeta FPGA Spartan 3E

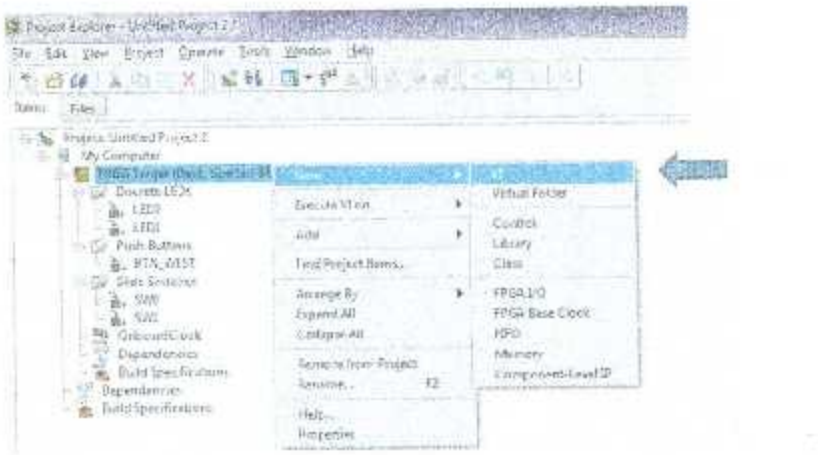


Figura 8.29 Nuevo Instrumento Virtual VI (*Virtual Instrument*).

8.4.3 Funciones disponible en el diagrama a bloques para VI LabVIEW FPGA.

A continuación se explicarán las funciones que el bloque para FPGA de LabVIEW 2011 tiene disponibles, porque es importante resaltar que LabVIEW cuenta con una gran variedad de funciones para realizar aplicaciones casi para cualquier ramo de la tecnología, pero tenemos que recordar que al programar en FPGA, se está programando en *hardware*, es decir, cada vez que se descarga una aplicación en una tarjeta FPGA, en realidad se están reconfigurando las conexiones entre las compuertas lógicas de la tarjeta, que en esencia eso es lo que es una tarjeta FPGA, un conjunto de compuertas lógicas.

Por lo tanto cuando se programe en cualquier lenguaje ya sea, HDL Coder de Simulink, en lenguaje C o C++ o en lenguaje gráfico, es necesario programar orientado a *hardware*.

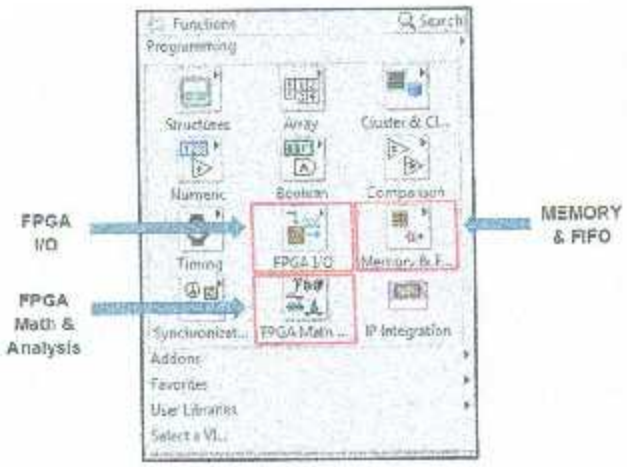


Figura 8.30 Funciones disponibles para FPGA en LabVIEW 2011.

Bloque de Memory & FIFO

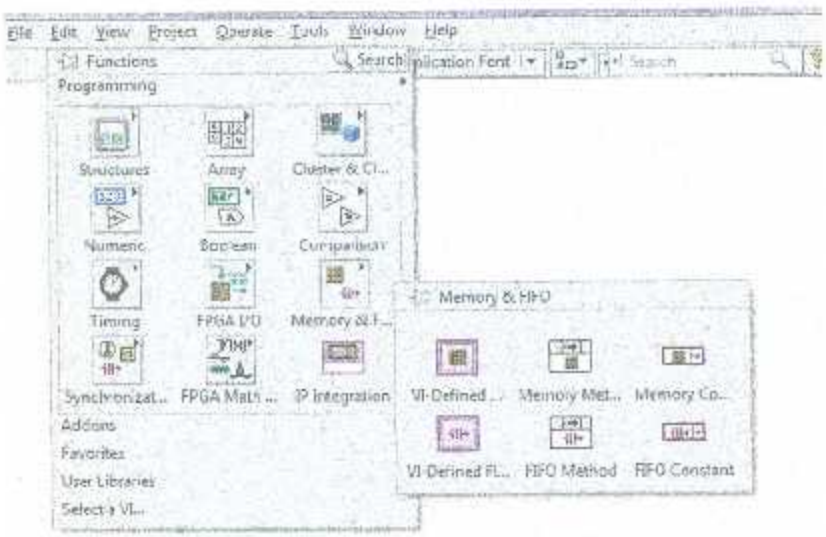


Figura 8.31 Bloques de memoria y FIFO'S.

Bloque de FPGA I/0

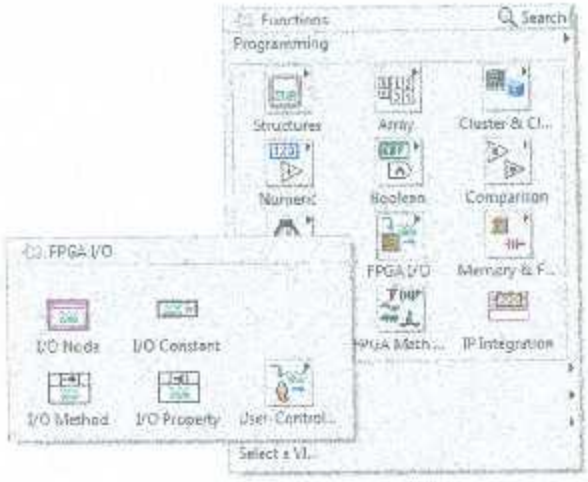


Figura 8.32 Bloques de Input/Output.

Bloque de FPGA Math & Analysis

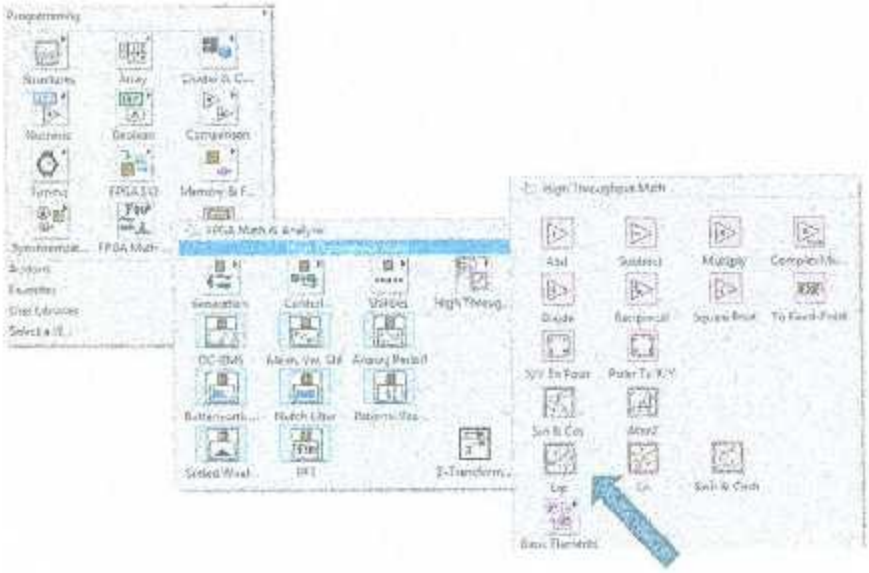


Figura 8.33 Operaciones matemáticas disponibles en LabVIEW 2011.

Realizar un programa ejemplo

1-. Abrir un nuevo proyecto en LabVIEW

2 - Agregar la tarjeta FPGA a utilizar.

3. Agregar los periféricos necesarios.

Agregar del SW0 al SW3

Agregar led del LD0 al LD3

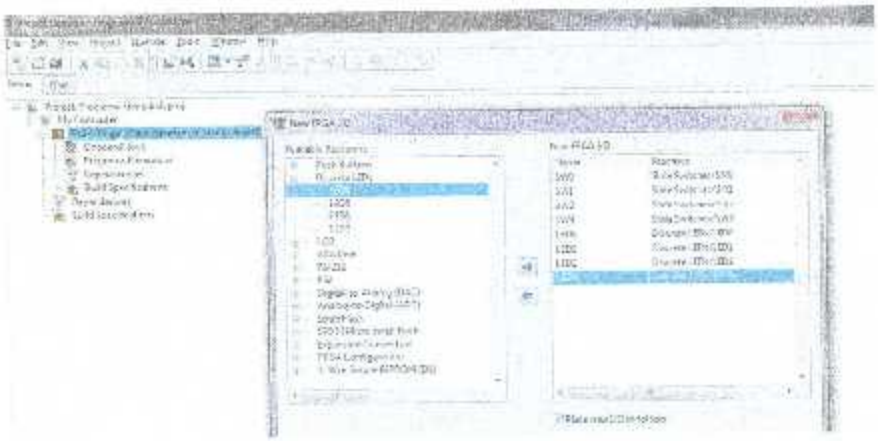


Figura 8.34 Programa ejemplo: agregar periféricos.

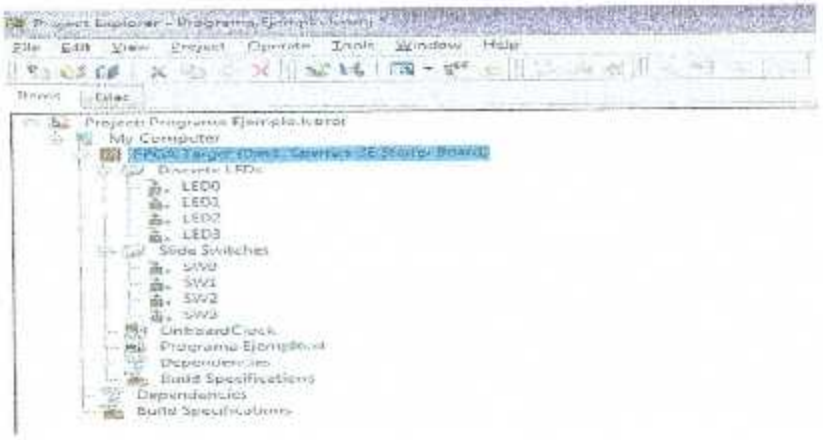


Figura 8.35 Programa ejemplo: vista del *Project Explorer*, periféricos agregados exitosamente.

4. Abrir un nuevo VI en la tarjeta previamente seleccionada e añadir compuertas lógicas e incluirlas dentro de un ciclo *while*.

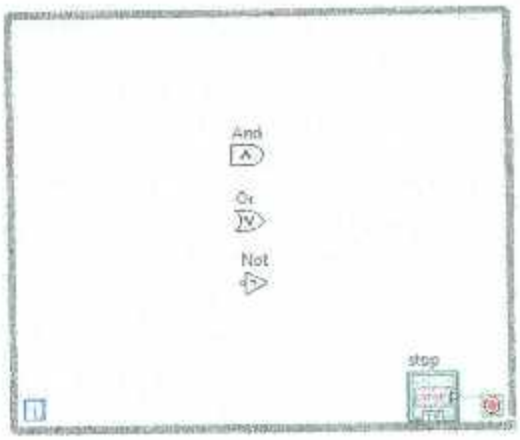


Figura 8.36 Programa ejemplo: Diagrama de bloques.

4.-Agregar 2 *I/O Node*.

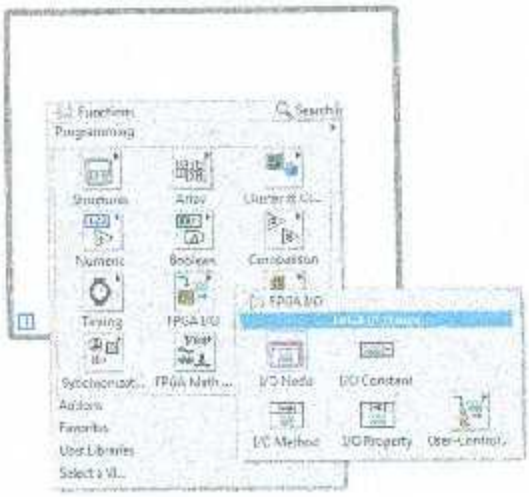


Figura 8.37 Programa ejemplo: Incluir bloques *I/O Node*

6. Agregamos los *switch* y los *leds*, en los bloques de *I/O Node*

Figura 8.38 Programa ejemplo: Selección de *switch* y *leds*

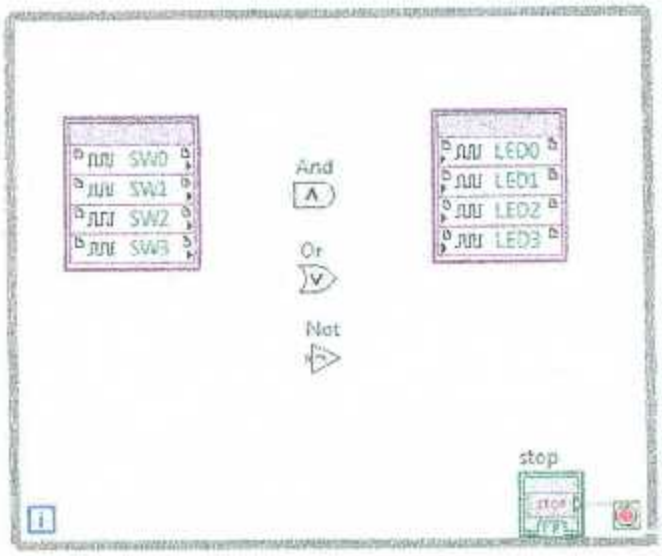


Figura 8.39 Programa ejemplo: Diagrama a bloques

7. Establecer las uniones deseadas.

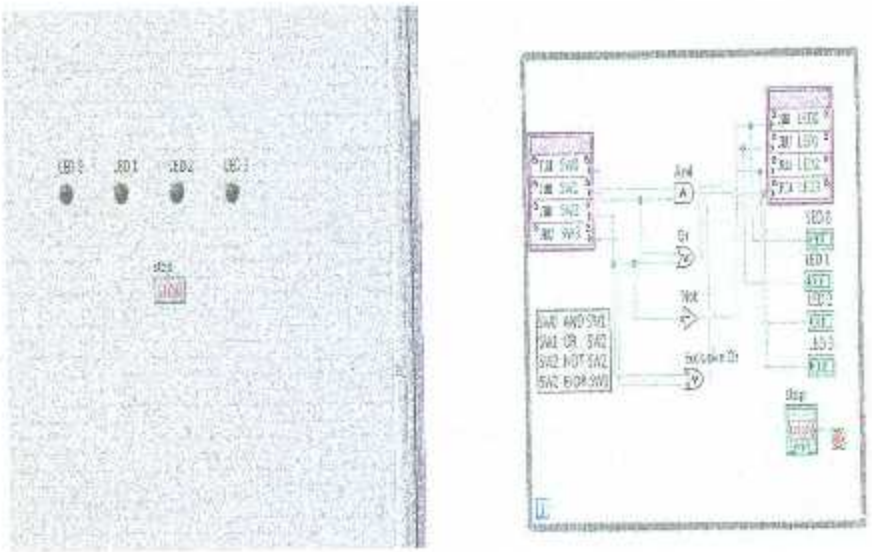


Figura 8.40 Programa ejemplo: Diagrama a bloques, conexiones.

8. Correr el programa.

Seleccionar el compilador.

Figura 8.41 Programa ejemplo: Inicio compilación.

Compilación del programa

Figura 8.42 Programa ejemplo: Compilación.

9. Reporte generado de cantidad de recursos lógicos utilizados.

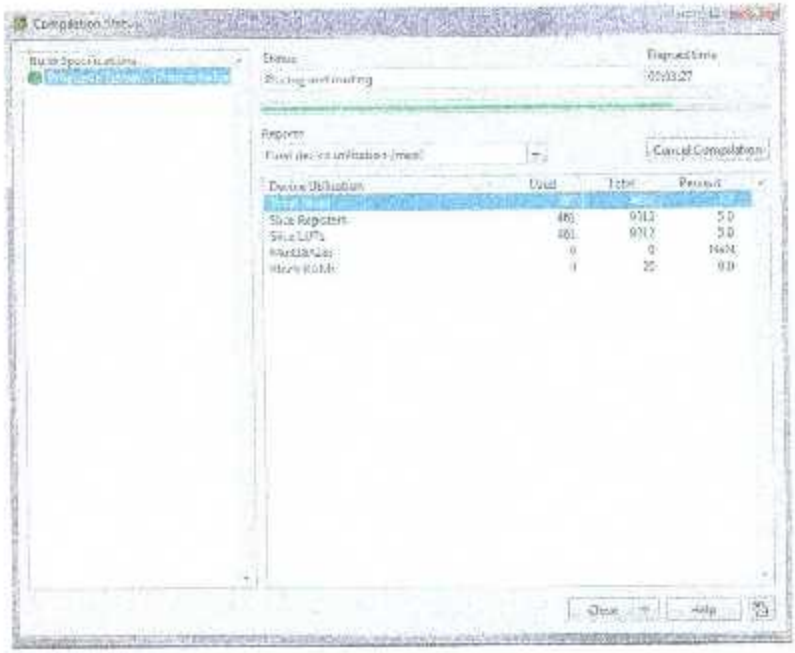


Figura 8.43 Programa ejemplo: Reporte final indicando el total de recursos lógicos usados.

10. Al finalizar la compilación, se muestra el siguiente mensaje., indicando que el programa está corriendo

Figura 8.44 Programa ejemplo. Compilación finalizada

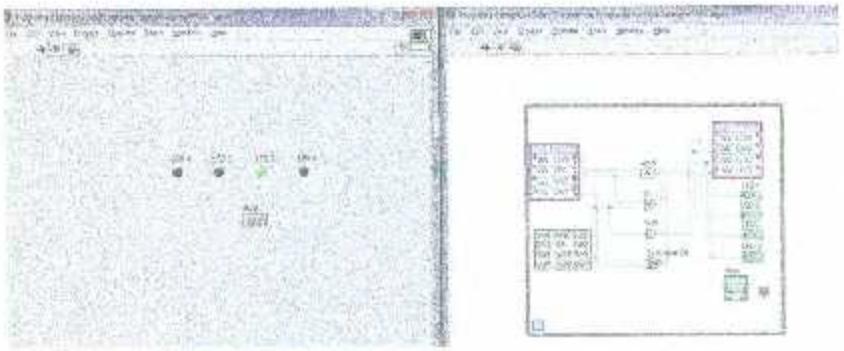


Figura 8.45 Programa ejemplo: Programa corriendo.

8.5 Implementación de *Host VI* para comunicar la tarjeta FPGA

Es posible mediante la programación de un *Host VI* y la creación de un VI adicional intercambiar información entre la FPGA y la computadora, en donde el VI HOST se ejecuta en el ordenador y el VI FPGA se ejecuta en la propia tarjeta.

Como se muestra en la siguiente ilustración:

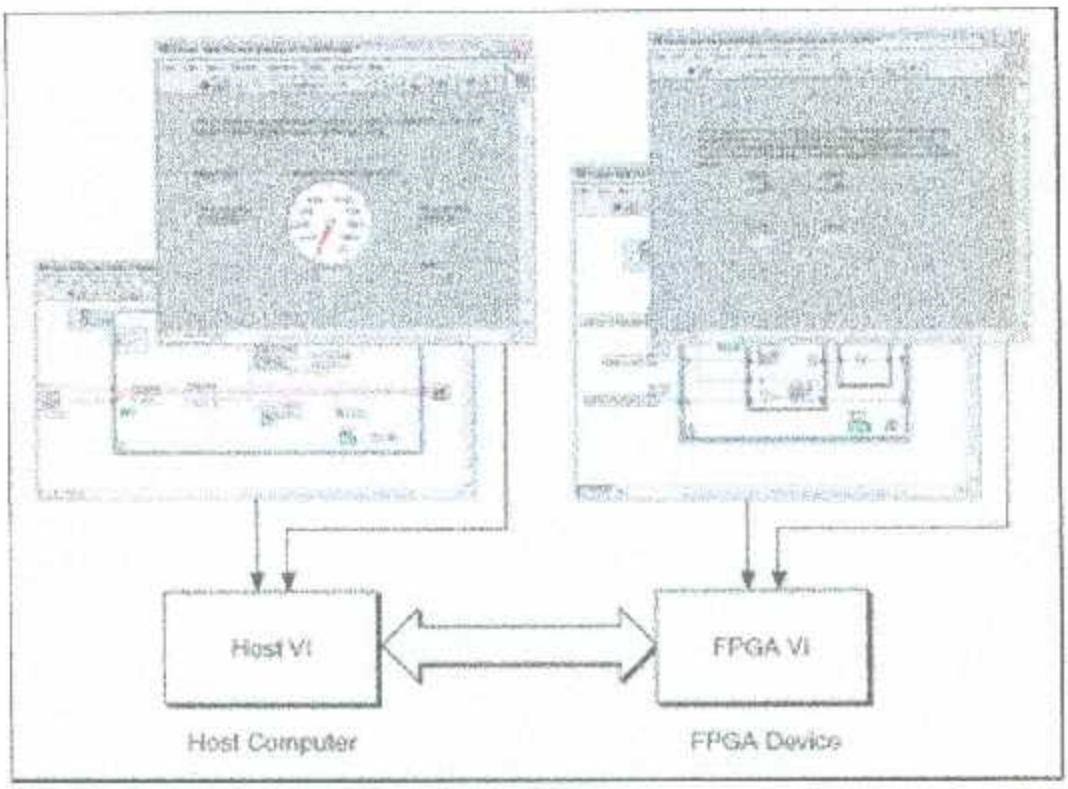


Figura 8.46 Interacción entre *Host VI* y FPGA VI

8.5.1 Ventajas de la utilización de un *Host VI.*

- Cuando la cantidad de recursos lógicos necesarios para el desarrollo de la aplicación supera la capacidad de procesamiento que la FPGA tiene disponible.

- Cuando es necesario llevar a cabo operaciones que no están disponibles en la tarjeta FPGA, como el uso de números de doble precisión o aritmética de punto flotante, ya que la tarjeta solo permite el uso de punto fijo.
- Cuando se desea controlar el tiempo y la secuencia de transferencia de datos
- Cuando es necesario correr diferentes programas en la tarjeta, pero no todos se encuentran almacenados al mismo tiempo, es decir, se va cargando a la FPGA el programa que se va requiriendo, con el objetivo de optimizar el espacio y recursos lógicos disponibles.

8.5.2 Programa ejemplo utilizando un Host VI:

Generar una señal PWM con salida al LED 0 de la tarjeta Spartan 3E.

XILINX *ISE DESING SUITE*

1.- Se realiza un programa en la plataforma de programación XILINX *ISE DESING SUITE*, lenguaje VHDL.

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;

-- Uncomment the following library declaration if using
-- arithmetic functions with Signed or Unsigned values
--use IEEE.NUMERIC_STD.ALL;

-- Uncomment the following library declaration if instantiating
-- any Xilinx primitives in this code.
--library UNISIM;
--use UNISIM.VComponents.all;

entity MD03_PWM_V1_VHDL_MODULE is

Port      ( pwmout : out  STD_LOGIC;
            clk : in  STD_LOGIC;
            refin : in  STD_LOGIC_VECTOR (10 downto 0));

end MD03_PWM_V1_VHDL_MODULE;

architecture Behavioral of MD03_PWM_V1_VHDL_MODULE is

signal cuenta : std_logic_vector(11 downto 0) := "000000000000";-- 50 MHz/ 25kHz= 2000/2=1000-1=999="001111100111"
signal cuentaPwm : std_logic_vector(10 downto 0) := "00000000000";

begin
process(clk)
begin
   if clk ='1' and clk'event then
      if cuenta <= "001111100111" then -- 50 MHz/ 25MHz= 2000/2=1000-1=999="001111100111"
         cuenta <= cuenta + 1;
         cuentaPwm <= cuentaPwm + 1;


      else
         cuenta <="000000000000";
      end if;
   end if;
end process;

process(refin, cuentaPwm)
begin
   if cuentaPwm <= refin then
      pwmout <= '1';
   else
      pwmout <= '0';
   end if;
end process;
end Behavioral;
```

Figura 8.47 Programa realizado en *Ise Desing Suite*, leguaje VHDL.

2.-Compilamos el programa en programación XILINX *ISE DESING SUITE* y hacemos la asignación de pines. Generación de archivo *.UCF

```
1
2  NET "clk"  LOC = "C9" ;#| IOSTANDARD = LVCMOS33 ;
3  NET "pwmout"  LOC = "D4"  ;#| IOSTANDARD = LVTTL | SLEW = SLOW | DRIVE = 6 ;
4  NET "refinc0"  LOC = "L13" ;#|IOSTANDARD = LVTTL | PULLDOWN ;
5  NET "refinc1"  LOC = "L14" ;#|IOSTANDARD = LVTTL | PULLDOWN ;
6  NET "refinc2"  LOC = "K12" ;#|IOSTANDARD = LVTTL | PULLDOWN ;
7  NET "refinc3"  LOC = "K13" ;#|IOSTANDARD = LVTTL | PULLDOWN ;
8  NET "refinc4"  LOC = "M13" ;#|IOSTANDARD = LVTTL | PULLDOWN ;
9  NET "refinc5"  LOC = "N13" ;#|IOSTANDARD = LVTTL | PULLDOWN ;
10 NET "refinc6"  LOC = "V4" ;# |IOSTANDARD = LVTTL | PULLDOWN ;
11 NET "refinc7"  LOC = "D14" ;#|IOSTANDARD = LVTTL | PULLDOWN ;
12
13
14
```

Figura 8.48 Asignación de periféricos de entrada y salida en *Ise Design Suite* 14.1

Enseguida continuamos con la programación en LabVIEW.

LABVIEW FPGA MODULE

1.- Abrimos un nuevo proyecto

2.- Agregamos la tarjeta SPARTAN 3E

3.- Agregamos un New VI FPGA LV

4.- En ***My Computer,*** inserto un nuevo VI, será nuestra *Host VI*:

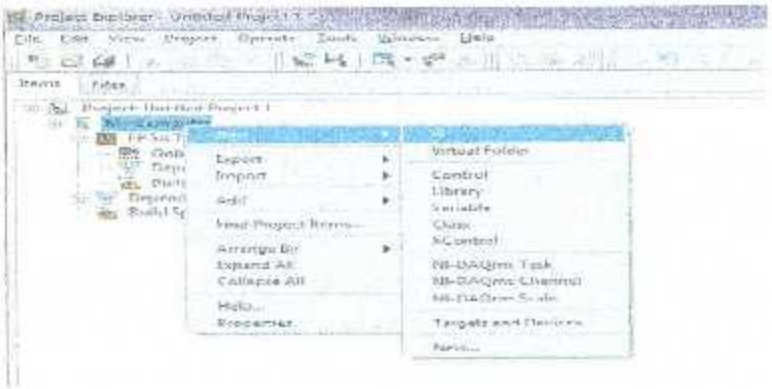


Figura 8.49 Crear un nuevo VI (instrumento virtual), el cual será el *Host VI*.

5.- En ***FPGA Target (Dev1, Spartan-3E Starter Board)***, inserto un nuevo VI:

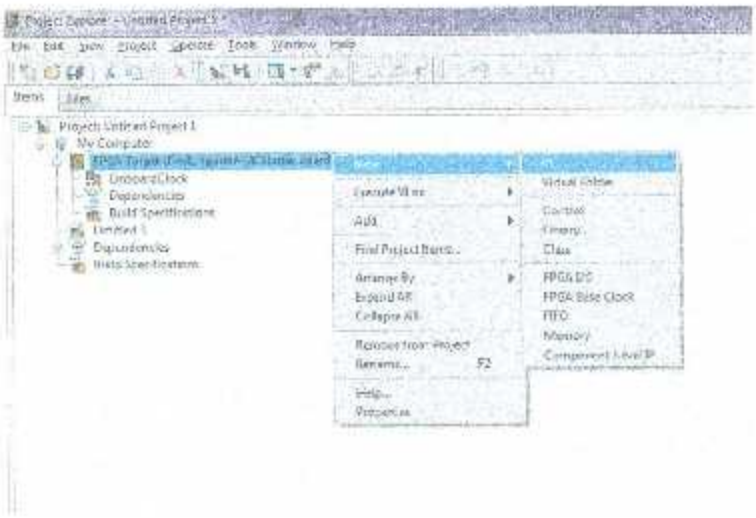


Figura 8.50 Creación de FPGA VI

6.- En ***FPGA Target (Dev1, Spartan-3E Starter Board)*** configuro la frecuencia del oscilador de la tarjeta FPGA.

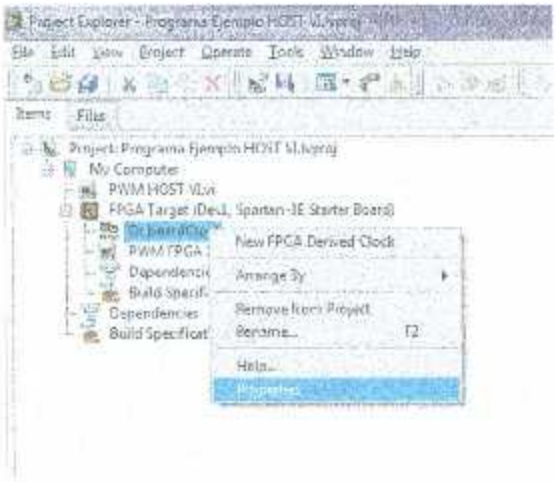


Figura 8.51 Configuración de oscilador interno.

Seleccionamos la frecuencia de trabajo del clock interno de la FPGA.

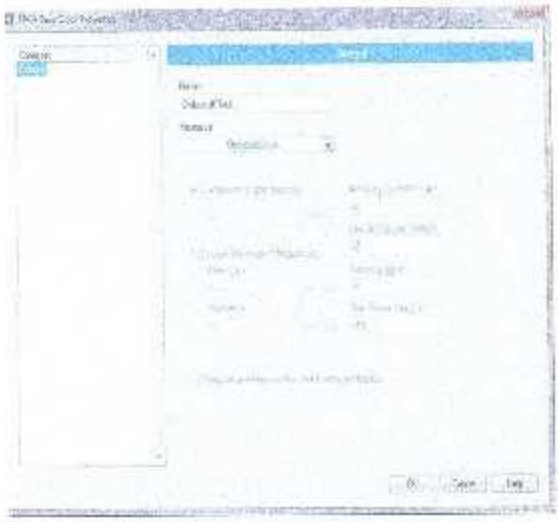


Figura 8.52 Precisar frecuencia de trabajo de la tarjeta FPGA.

7.- En ***FPGA Target (Dev1, Spartan-3E Starter Board),*** agrego los periféricos a utilizar. Para esta aplicación en particular, solo agrego el LED 0 como dispositivo de salida PWM de la FPGA.

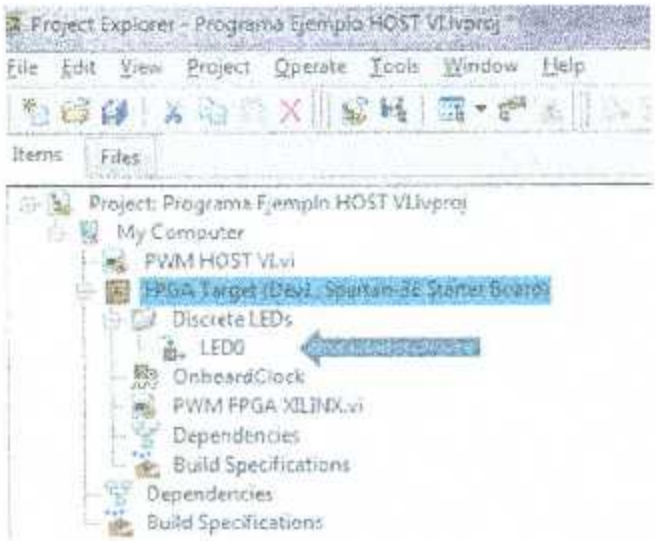


Figura 8.53 Cargar periféricos.

8.- En el diagrama a bloques del VI FPGA LV, agrego un IP. El cual nos permite seleccionar el archivo *.VHDL a ejecutar

Figura 8.54 Bloque IP *Integration*

>>Click en Add Synthesis File...

Selecciono el archivo **.*VHDL**

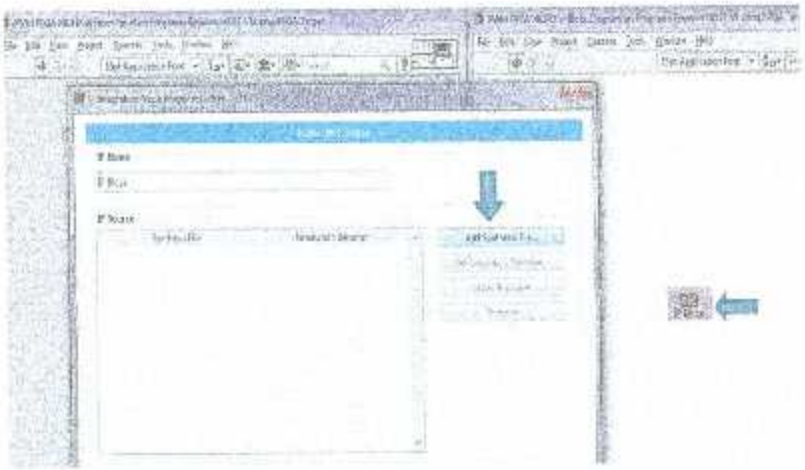


Figura 8.55 Agregar archivo *.vhdl

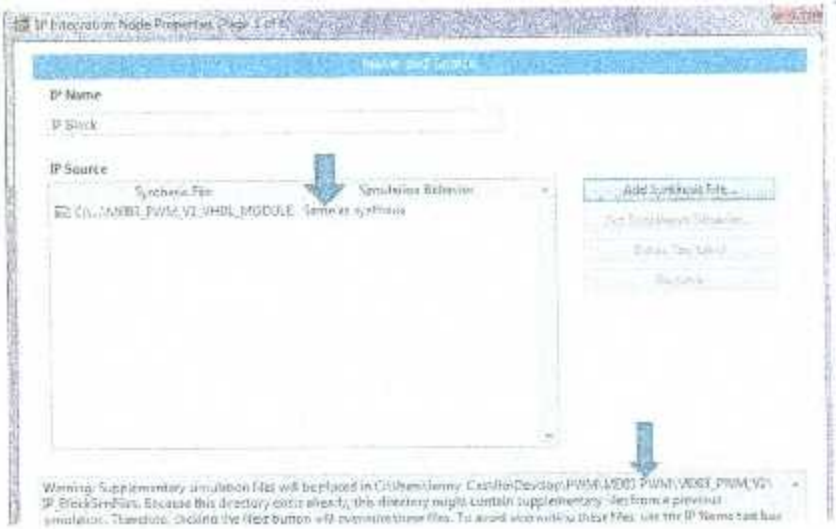


Figura 8.56 Selección del programa *.vhdl.

>>NEXT

9.- Delimito que el programa solo se pueda ejecutar en la tarjeta SPARTAN 3E.

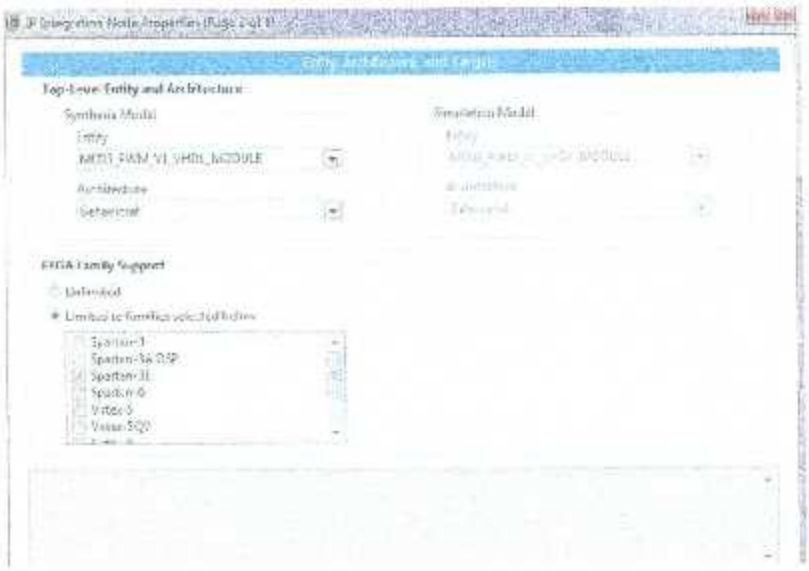


Figura 8.57 Limitar el programa a la tarjeta Spartan 3E.

10.- Primero verifico sintaxis del archivo *.VHDL, después *CLICK* en Generar.

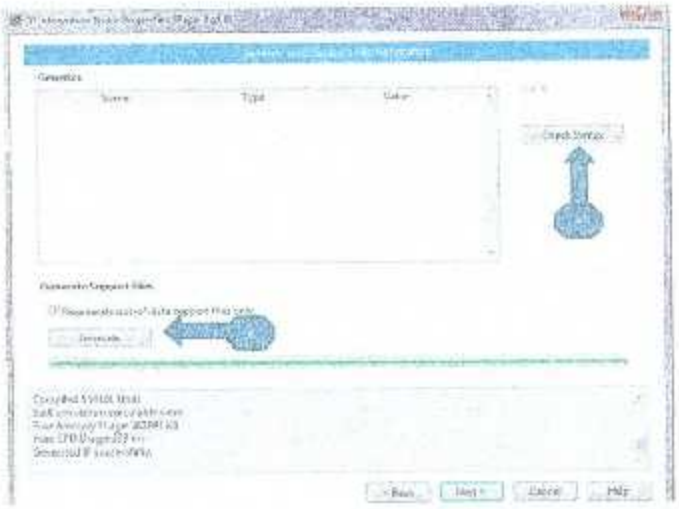


Figura 8.58 Verificar sintaxis en programa *.vhdl

>>NEXT

11.- La señal de CLK, se tomará directamente del oscilador interno de la propia tarjeta Spartan 3E es por eso que en esta ventana no se habilita nada.

Figura 8.59 *Clock and enable signals.*

12.- >>**NEXT**

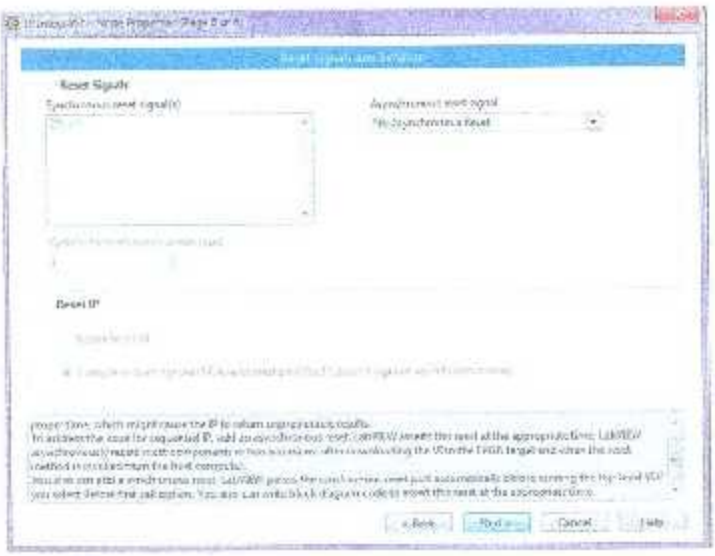


Figura 8 .60. *Reset Signals and Behavior*.

13.- Selecciono el tipo de datos.

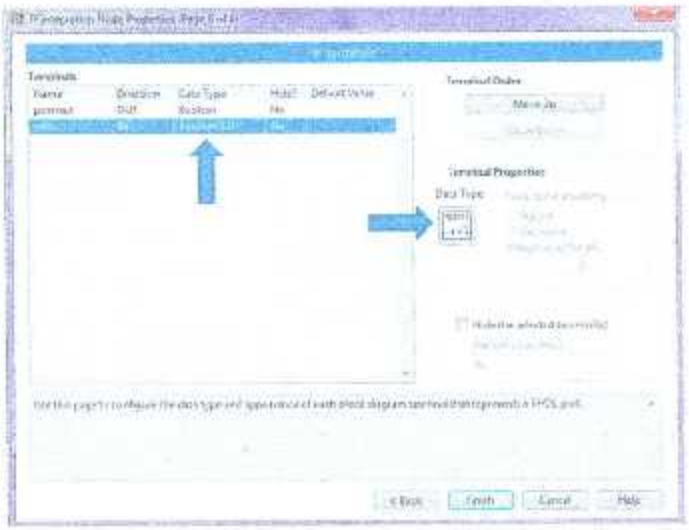


Figura 8.61 Configuración de tipo de datos a manejar.

14.- >>**FINISH**

15.- El diagrama a bloques queda de la siguiente forma:

Figura 8.62 *IP Block* en ventana de diagrama de bloques.

16.- Agrego un ***Timed Loop***, y selecciono la señal de CLK de la tarjeta.

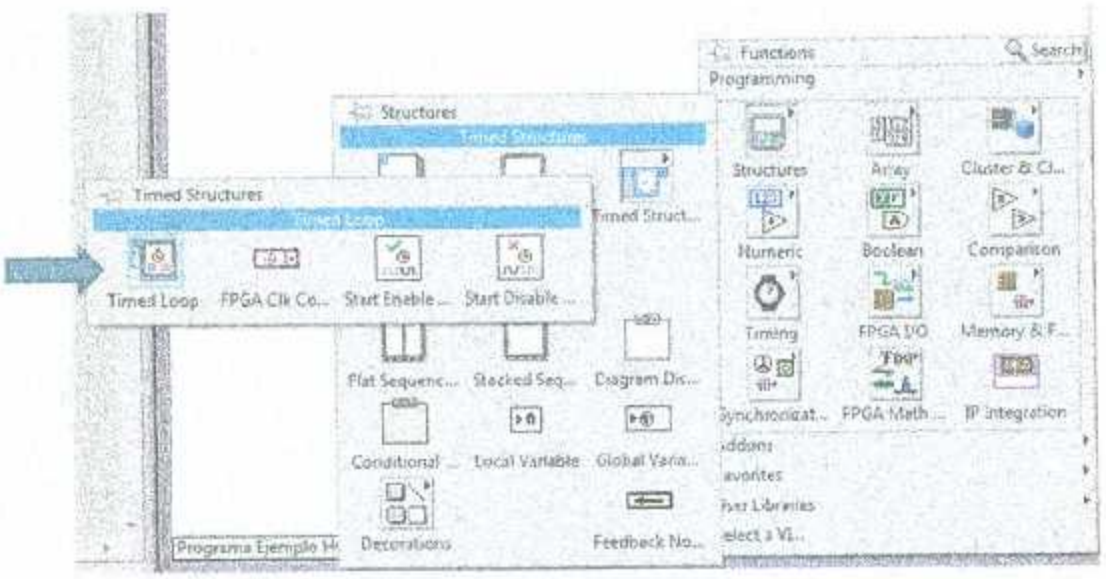


Figura 8.63. *Timed Loop*

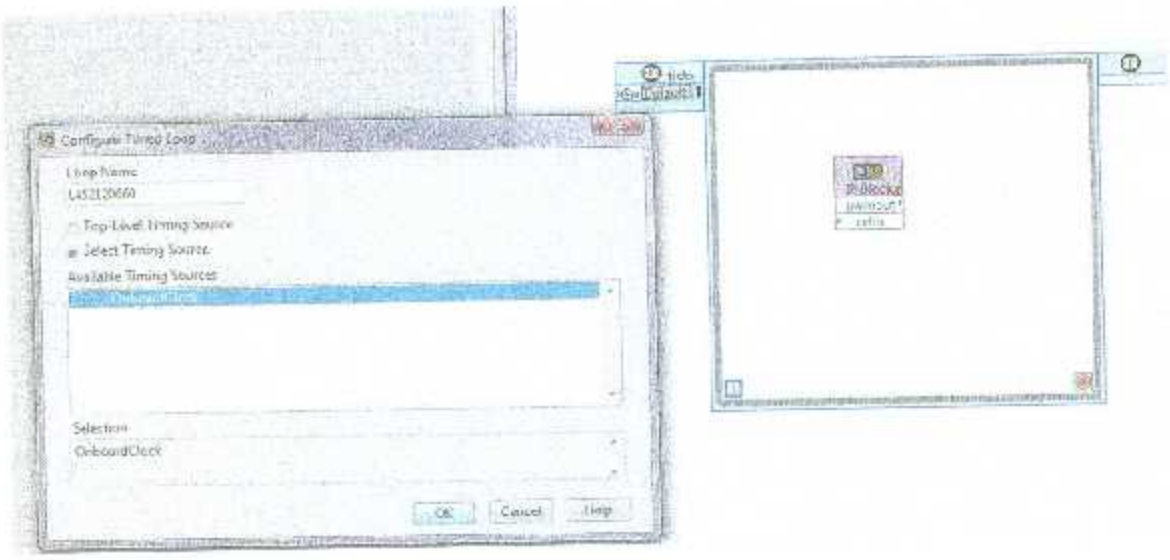


Figura 8. 64. Configure *Timed Loop*.

17.- Agrego un *I/O Node*, y selecciono el LED 0

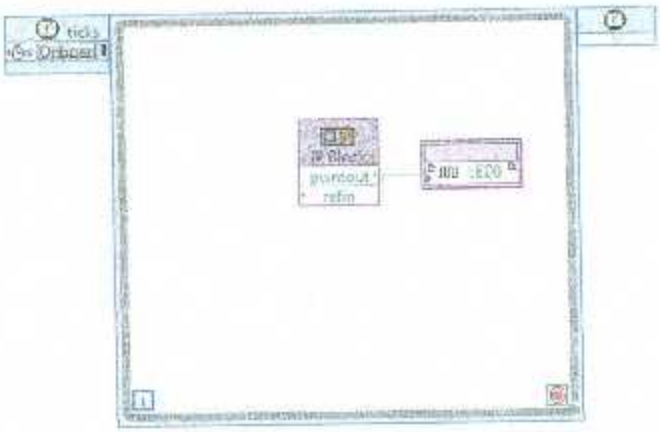


Figura 8.65 Unión de *IP Block* e *I/O Node*

18.- Realizo las siguientes conexiones.

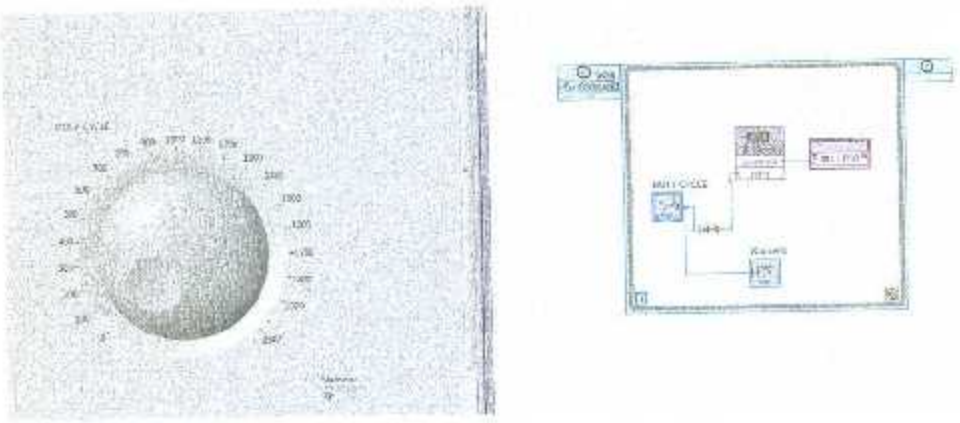


Figura 8.66 Conexión del diagrama a bloques

19.- Compilo el programa.

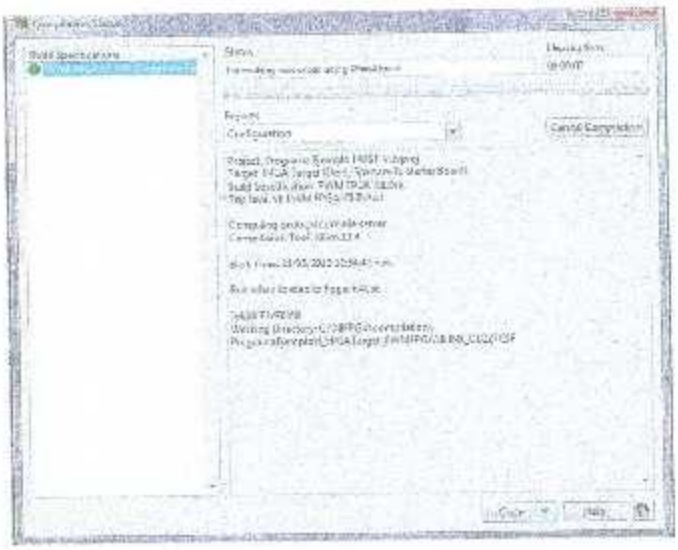


Figura 8.67 Inicio de compilación del programa.

20. - ***My Computer, Host VI.***

>>Agrego el bloque *Open FPGA*

>>Agrego el bloque *Read/Write FPGA*

>>Agrego el bloque *Close FPGA*

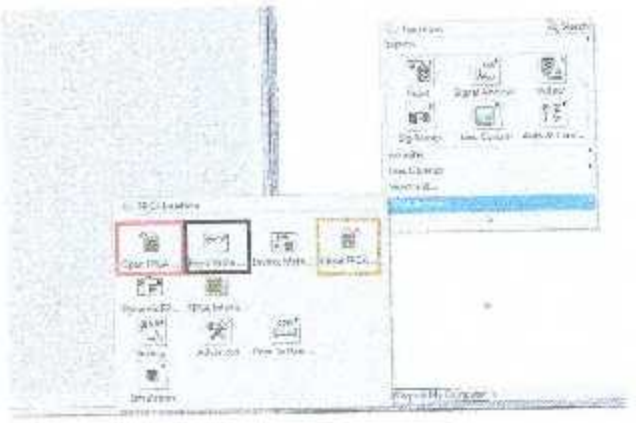


Figura 8.68 Programación *Host VI*

21.- Diagrama de bloques *Host VI*

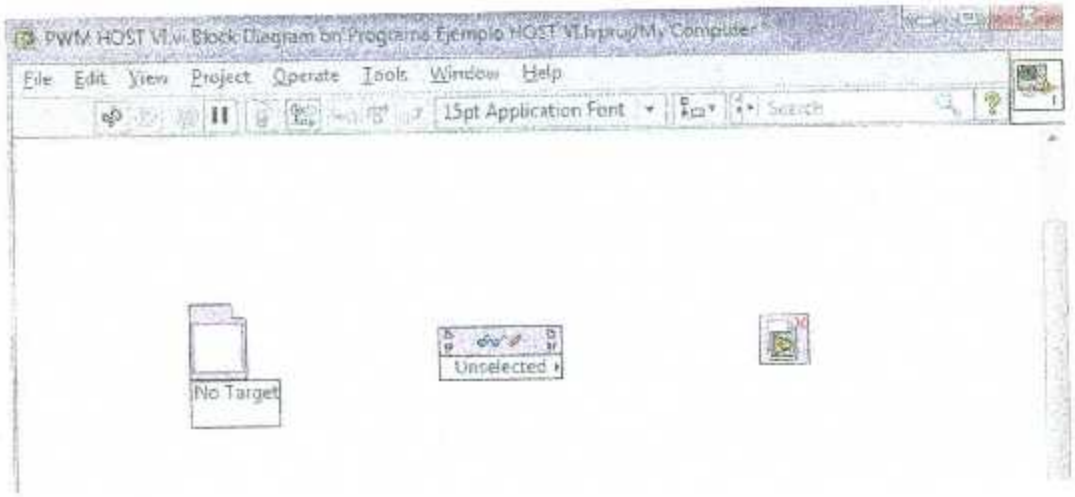


Figura 8.69. Diagrama a boques

22.- Cargo un VI en OPEN FPGA.

Selecciono la VI FPGA I.V

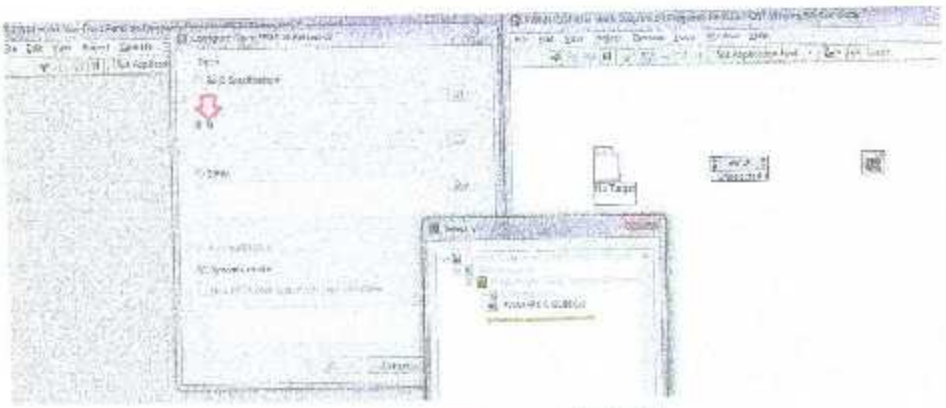


Figura 8.70 Cargar el programa VI FPGA

23.- Uniendo los tres bloques.

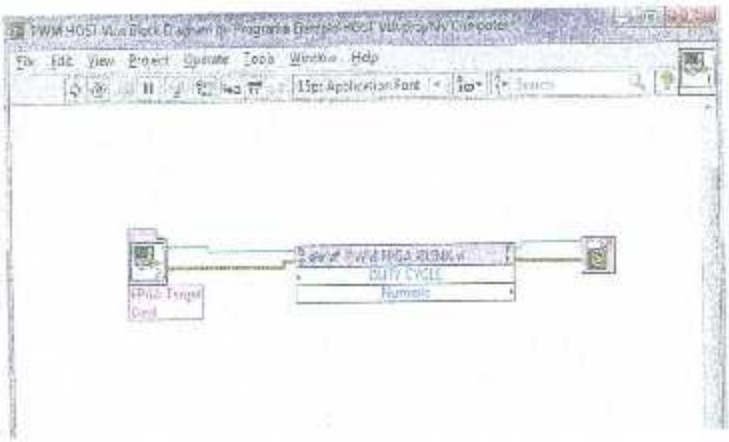


Figura 8.71 Diagrama a bloques, conexión entre elementos.

24. Agrego un *white* loop, para que el programa se ejecute hasta que el usuario presione el botón de Stop.

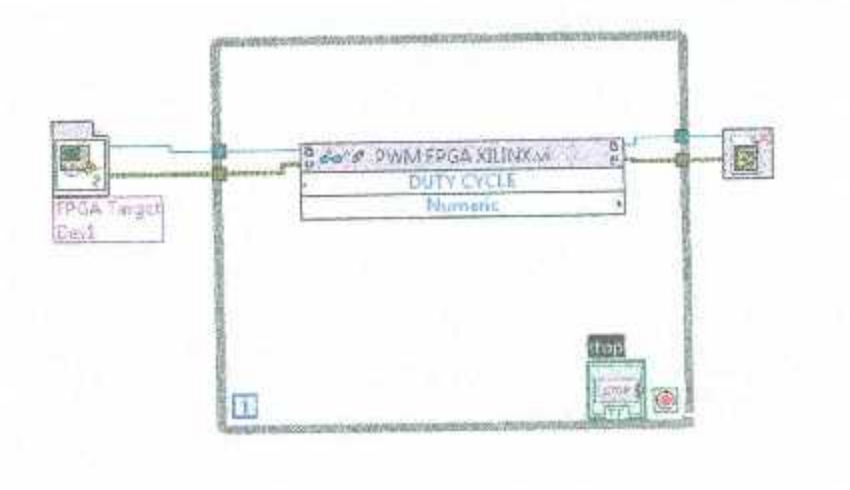


Figura 8.72 Estructura *While loop*

25.- Agrego un control para regular el PWM en el panel frontal.

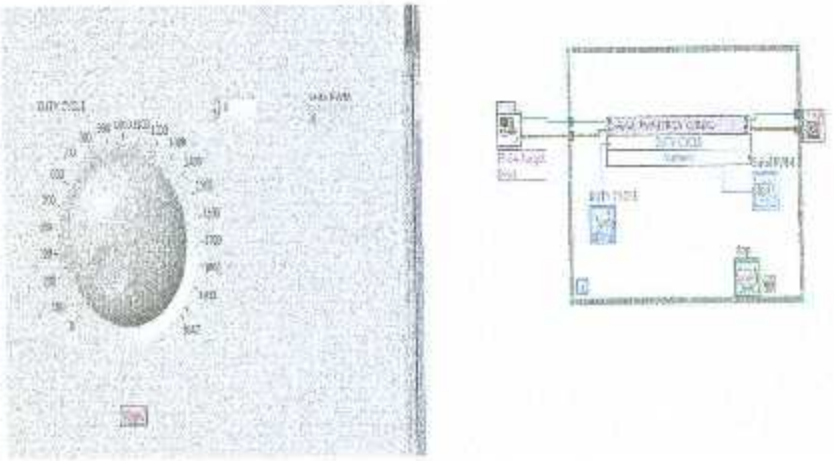


Figura 8.73 Panel frontal y diagrama a bloques

Capítulo IX. Lenguaje de descripción de *hardware*.

9.1 Programación en VHDL.

Como se mencionó anteriormente, el trabajo de investigación aquí presentado representa la unión de lenguaje de descripción de *hardware* en VHDL con lenguaje gráfico en LabVIEW, en lo que respecta al desarrollo del capitulo presente, este se centrará en la explicación de los programas realizados en VHDL.

9.2 Sensor Ultrasónico

Se hizo mención que era necesario el monitoreo de la posición del objeto que se encuentra dentro del tubo de acrílico, para ello se colocó el sensor ultrasónico SFR05 en el borde superior del tubo.

9.2.1 Funcionamiento del sensor SFR05

La señal necesaria para accionar el sensor (señal de disparo), se va a mandar a través de la línea llamada "Trigger Pulse Input", la cual es una señal cuadrada que durará 10 ms en alto seguida de 46 ms en bajo, esta señal se producirá con la tarjeta FPGA Spartan 3E.

El diagrama de tiempos del sensor SRF05 es el mostrado en la imagen 9.1, el cual es un diagrama base para el desarrollo del programa en lenguaje VHDL, ya que todo el programa es en base a tiempos, los cuales se deben de pasar a pulsos de reloj.

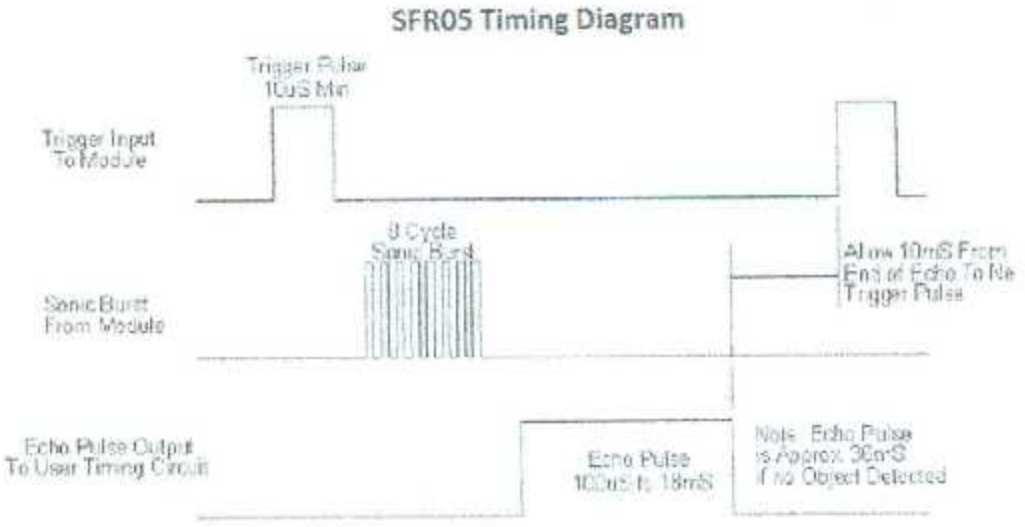


Figura 9.1 Diagrama de tiempo Sensor Ultrasónico SRF05 [18].

La señal que la tarjeta FPGA recibirá, proviene de la línea "Echo pulse", esta señal después de un cierto tiempo se pondrá en alto, y estará en alto hasta que el sensor detecte un objeto e inmediatamente cambiará de estado, por lo tanto el tiempo que la señal duró en alto es proporcional a la distancia a la que se encuentra el objeto

Librerías

Es necesario agregar las librerías necesarias ya que estas contienen las funciones predefinidas a utilizarse en el código del programa y los tipos de datos a manejar, por mencionar algunas de sus funciones.

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;
```

Figura 9.2 Librerías en VHDL. SFR05

Entidad: Señales de entrada/salida

La entidad es la parte del programa que define las entradas y salidas del circuito.

```
entity prueba7_vhdl_module is
Port      ( rst : in  STD_LOGIC;
            clk : in  STD_LOGIC;
            in_1: in std_logic;
            clk_5ms : out  STD_LOGIC;
            centimetros: out std_logic_vector (11 downto 0));
end prueba7_vhdl_module;
```

Figura 9.3 Señales de entrada/salida SFR05.

En el programa se consideraron las siguientes señales, una señal de reset (rst) , una señal de reloj (clk) la cual determina la frecuencia de trabajo del sistema, la señal de entrada del eco del sensor ultrasónico (in_1), una señal de disparo trigger (clk_5ms) y un acumulador de la distancia detectada en centimetros.

Señales auxiliares

```
architecture Behavioral of prueba7_vhdl_module is
--Contador de  30 MILISEGUNDOS (DISTANCIA MAXIMA)
signal aux_cont_30ms: STD_LOGIC_VECTOR (29 downto 0);--Cuenta hasta 1,499,999= "000101101011000011010111111"

--CENTIMETROS
--signal aux_cuentas_equivalen_centimetros: std_logic_vector (11 downto 0);
signal aux_cantidad_centimetros: std_logic_vector (11 downto 0):="000000000001";

--Pulso de disparo
--Contador 5 MILISEGUNDOS
signal aux_cont_5ms: STD_LOGIC_VECTOR (19 downto 0);--Cuenta hasta 249,999= "001111010000100011111"
signal aux_clk_5ms: STD_LOGIC;

--Contador de 50 MILISEGUNDOS
signal aux_cont_50ms: STD_LOGIC_VECTOR (25 downto 0):--Cuenta hasta 2,499,999= "001001100010010110111111"
```

Figura 9.4 Señales auxiliares SFR05.

Es necesario definir señales auxiliares, las cuales son para manejo interno del programa, es decir, no se relacionan directamente ni con las señales de entrada ni con las de salida.

Programa

```
begin

--GENERADOR DE 5 ms espaciados por 65 mS en bajo
process(rst, clk,aux_cont_50ms,aux_cont_5ms,aux_clk_5ms)
begin
if rst='1' then
aux_cont_50ms<=(others =>'0');
aux_cont_5ms<=(others =>'0');
aux_cantidad_centimetros<=(others =>'0');
else
if clk='1' and clk'event  then
   if (aux_cont_5ms< "00111101000010001111" ) then--Cuenta hasta 249,999= "00111101000010001111"
   aux_cont_5ms<= aux_cont_5ms+1;
   aux_clk_5ms<='1';
   else
   aux_clk_5ms<='0';

         if (in_1='1') then
            aux_cont_30ms<= aux_cont_30ms-1;
            if (aux_cont_30ms="00000000000010111011000010") then
            aux_cantidad_centimetros<=aux_cantidad_centimetros+1;
            aux_cont_30ms<=(others =>'0');
            centimetros<=aux_cantidad_centimetros;
            aux_clk_5ms<='0';
            end if;
         else
             if (in_1='0') then
                   if aux_cont_50ms<"00101011000100101101011111" then--Cuenta hasta 2,499,999
                   aux_cont_50ms<= aux_cont_50ms+1;
                   aux_clk_5ms<='0';
                   else
                   aux_cont_50ms<=(others =>'0');
                   aux_cont_5ms<=(others =>'0');
                   aux_cantidad_centimetros<=(others =>'0');
                   end if;
             end if;

          end if;
    end if;
end if;
end if;
clk_5ms<=aux_clk_5ms;
end process;

end Behavioral;
```

Figura 9.5 Programa detector de distancia SFR05.

La señal de disparo (Trigger) será programada la cual le indica al sensor SFR05 que debe de realizar una nueva lectura. Los centimetros acumulados se almacenan en la variable "centimetros".

9.3 Frecuencímetro

El objetivo del programa del frecuencímetro, como su nombre lo indica es detectar la frecuencia de giro de los motores, que fueron montados en el prototipo.

La tarjeta FPGA Spartan-3E, recibe una señal cuadrada por uno de sus periféricos de entrada, el programa contabiliza la cantidad de pulsos positivos capturados en 1 segundo, obteniendo de esta manera la frecuencia, ya que la definición de Hertz indica, que es la cantidad de ciclos ocurridos en 1 segundo, y en el programa realizado, un ciclo equivale a una revolución del eje del motor.

Librerías y definición de entidad: Señales de entrada/salida

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;


entity RPM_V1 is
Port (   clk : in  STD_LOGIC;
         reset: in STD_LOGIC;
         frecuencia_in: in STD_LOGIC;
         led: out std_logic;
         rps: out STD_LOGIC_VECTOR (19 DOWNTO 0));-- Contador hasta 4999 = "0001001110000111"
end RPM_V1;
```

Figura 9.6 Librerías y entidad de frecuencímetro.

Se definen las librerías necesarias para el correcto funcionamiento del programa. En la declaración de los puertos tanto de entrada como de salida, se agrega una señal de reloj (clk) para sincronizar los procesos, es añadida una señal de reset, la frecuencia que entrega el sensor

CNY70 a la FPGA es a través del pin llamado frecuencia_in, y el valor resultante de la frecuencia determinada por la tarjeta FPGA Spartan - 3E *Starter Kit* es almacenado en la señal "rps".

Señales auxiliares

```
architecture Behavioral of RPM_V1 is
--,50,000,000 -1= 49,999,999= "00101111101011110000011111111"
signal aux_contador: std_logic_vector (27 DOWNTO 0):="0000000000000000000000000000";
signal aux_rps: std_logic_vector (19 downto 0):="00000000000000000000";
signal aux_frecuencia_in: std_logic;
signal aux_led: std_logic:='0';
signal flag_1: std_logic:='0';
signal flag_2: std_logic:='0';
```

Figura 9.7 Señales auxiliares de programa frecuencímetro.

Es necesario declarar señales auxiliares para realizar de forma más precisa las operaciones de funcionamiento.

Programa

En la imagen 9.8, se muestra el primer proceso del programa, en el cual se genera la señal de monitoreo de 1 segundo.

```
begin
process (clk,reset,frecuencia_in)
begin
   if (reset='1') then
   aux_contador <=(others =>'0');
   else
      if (clk='1' and clk'event) then
         if aux_contador <= "00101111101011119000011111111" then
         aux_contador <= aux_contador +1;
         flag_1<='0';

         else
            flag_1<='1';
            if aux_led='1' then
            aux_led<='0';
            aux_contador <=(others =>'0');
            else
            aux_led<='1';
            aux_contador <=(others =>'0');
            end if;

         end if;
      end if;
   end if;
led<= aux_led;
end process;
```

Figura 9.8 Proceso 1 de frecuencímetro

En el proceso 2 (imagen 9), se estarán contabilizando flancos positivos de la señal de entrada de frecuencia (señal del sensor CNY70) durante el lapso de tiempo, indicado en el proceso 1.

```
--RPS
process (flag_1,flag_2, aux_frecuencia_in,aux_led)

begin
aux_frecuencia_in <= frecuencia_in;
if  clk='1' and clk'event then
   if flag_1='0'  and aux_frecuencia_in='1' and flag_2='0' then
   aux_rps<=aux_rps + 1;
   flag_2<='1';
   else
         if flag_1='0'  and aux_frecuencia_in='0' and flag_2='1' then
         flag_2<='0';
         end if;
   end if;

   if flag_1='1' then
   rps<=aux_rps;
   aux_rps<=(others =>'0');
   end if;
end if;
end process;

end Behavioral;
```

Figura 9.9 Proceso 2 de frecuencímetro.

9.4 Señal de control PWM

La modulación por ancho de pulso (o PWM, por sus siglas en inglés de *pulse-width modulation*) de una señal, es una técnica en la que se modifica el ciclo de trabajo de una señal quedando la frecuencia de trabajo fija.

Entidad: Señales de entrada/salida

Con el objetivo de crear una señal de control PWM, para la variación de la velocidad de los motores, es necesario declarar tres señales, donde dos de ellas son entradas y como única salida se tiene la señal PWM

```
entity MD03_PWM_V1_VHDL_MODULE is

Port      ( pwmout : out  STD_LOGIC;
            clk : in  STD_LOGIC;
            refin : in  STD_LOGIC_VECTOR (10 downto 0));

end MD03_PWM_V1_VHDL_MODULE;
```

Figura 9.10 Entidad programa PWM

Una de las entradas en la señal del reloj (clk), la cual proviene del oscilador interno de la propia tarjeta FPGA Spartan - 3E que funciona a 50 MHz, y nos ayudará a sincronizar las funciones. La señal de referencia llamada refino, nos indicará el valor del ciclo de trabajo.

Señales auxiliares

La frecuencia de trabajo de la señal PWM se estableció a 25 KHz, ya que por recomendaciones de los fabricantes de los *Drivers* utilizados, la frecuencia de PWM debe de ser superior a 20 KHz.

```
architecture Behavioral of MD03_PWM_V1_VHDL_MODULE is

signal cuenta : std_logic_vector(11 downto 0) := "000000000000";-- 50 MHz/ 25kHz= 2000/2=1000-1=999="001111100111"
signal cuentaPwm : std_logic_vector(10 downto 0) := "00000000000";
```

Figura 9.11 Señales auxiliares de PWM

Es necesario hacer un divisor de frecuencia de 50 MHz (la cual es la frecuencia de trabajo del oscilador interno de la tarjeta) a 25 KHz.

La señal "cuentaPwm" es la variable interna que almacenará de forma temporal el valor del ciclo de trabajo, el cual tiene una resolución de 0 a 2047, siendo 0 cuando el motor está apagado, el valor de 2047 indica una velocidad de giro del eje del motor máxima.

Programa

El programa fue dividido en dos partes, la primera de ellas es el divisor de frecuencia que fue generado a partir de la frecuencia de oscilación interna de la Spartan -3E.

```
begin
   process(clk)
   begin
      if clk ='1' and clk'event then
         if cuenta <= "001111100111" then -- 50 MHz/ 25MHz= 2000/2=1000-1=999="001111100111"
            cuenta <= cuenta + 1;
            cuentaPwm <= cuentaPwm + 1;


         else
            cuenta <="000000000000";
         end if;
      end if;
   end process;

   process(refin, cuentaPwm)
   begin
      if cuentaPwm <= refin then
         pwmout <= '1';
      else
         pwmout <= '0';
      end if;
   end process;
end Behavioral;
```

Figura 9.12 Programa para general señal de control PWM

En el segundo proceso, se altera el valor de la variable "cuentaPwm", para aumentar o disminuir la velocidad de giro del motor.

Capítulo X. Descripción del algoritmo

La mayor parte de la programación se realizó en el lenguaje grafico LabVIEW 2011, dicho lenguaje de programación se eligió porque permitió la interacción entre la plataforma de Xilinx Design Suit 14.1 y la programación de una interfaz gráfica de usuario.

10.1 Panel de control

El panel de control principal tiene seis modos de operación diferente, cada una de las pestañas mostradas en la imagen 10.1 juega un papel diferente en el control del prototipo, las pestañas de control son: condición de inicio, modo trabajo, modo entrenamiento, configuración, captura de valores y entrenamiento desde *Excel*.

A continuación se explicará el algoritmo de control detrás del funcionamiento de cada pestaña en el menu principal.

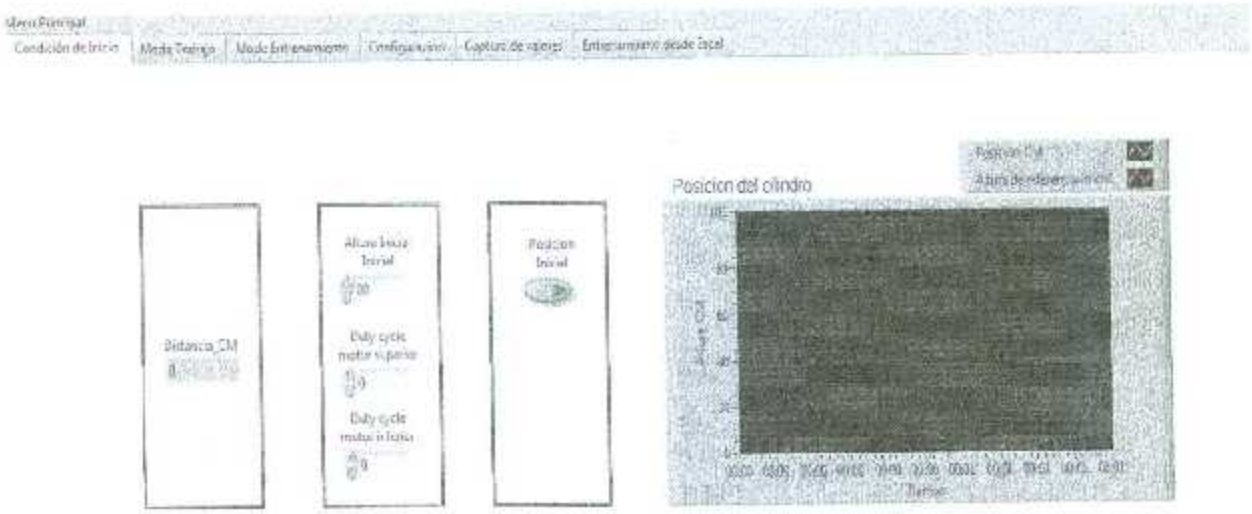


Figura 10.1 Menú principal.

El árbol de programación general, tiene una estructura en modo HOST es decir, por lo cual la cual está dividida entre dos secciones, donde algunas VI (*Virtual Instrument*) corren en la computadora y otras VI están diseñadas para que se ejecuten en la tarjeta FPGA Spartan - 3E, se detallará cada uno de los instrumentos virtuales programados figura 10.2.

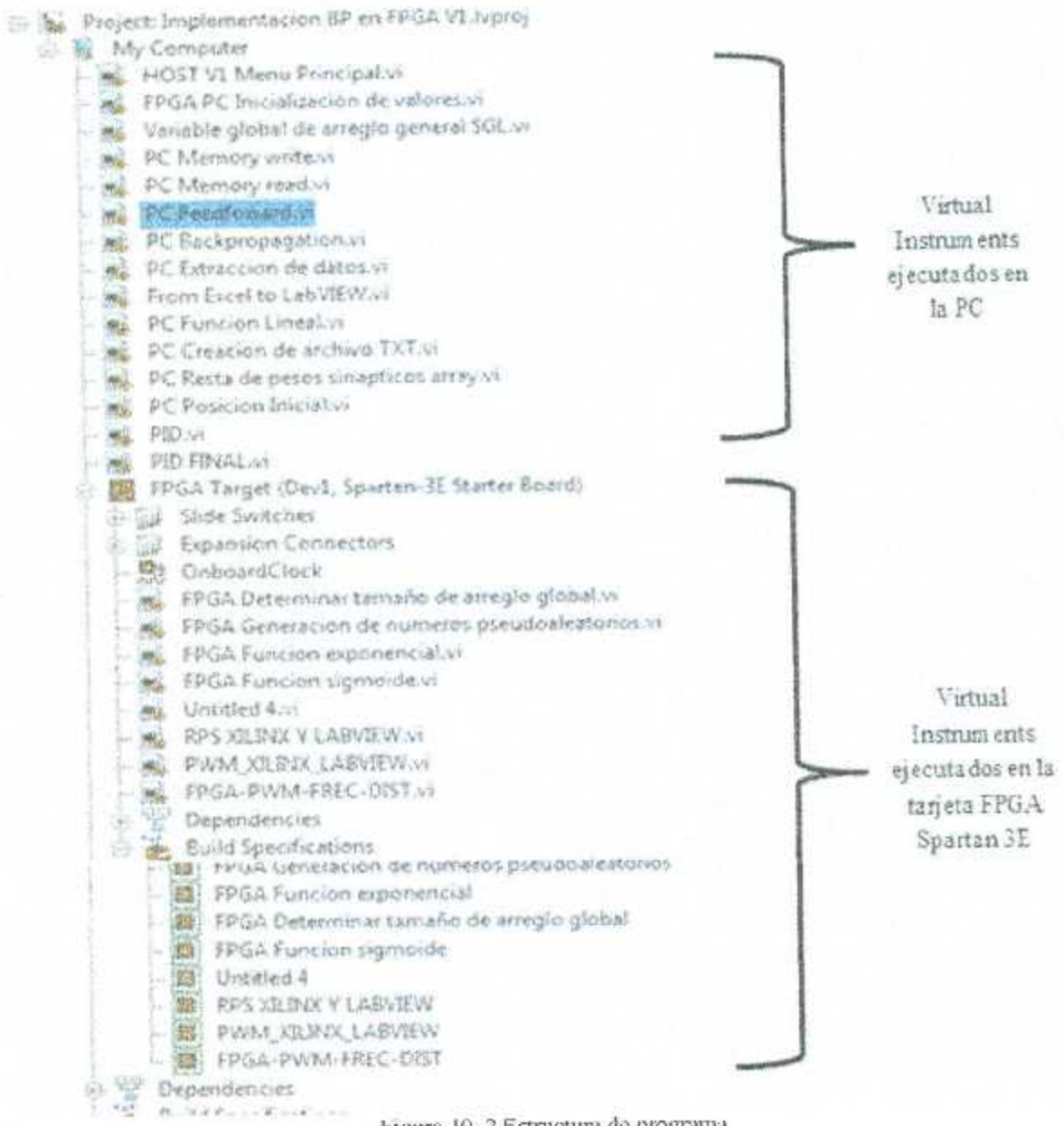


Figura 10. 2 Estructura de programa

10.2 Condición de inicio

El contenido de todos los programas forman parte de una estructura general *case*, donde dependiendo de la pestaña que este activa es el bloque de programa a ejecutar, imagen 10.3

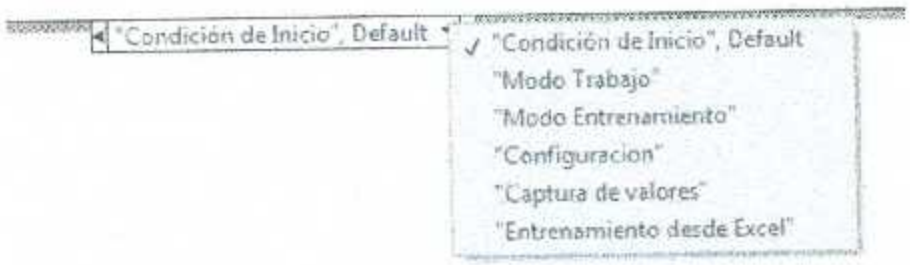


Figura 10.3 Estructura *case*.

La primer pestaña en el menú principal es el de condición de inició, originalmente se tenía pensado que previo a realizar el control de posicionamiento del cilindro dentro del tubo de acrílico, era necesario partir de una condición de inicio, es decir, que a partir de que el sistema alcanzara una altura inicial establecida por el usuario se continuara con el entrenamiento de la red neuronal y el controlador PID, pero afortunadamente la respuesta de los controladores es tan efectiva que no fue necesario iniciar el control a partir de una altura de estabilización, pero se decidió dejar este VI con el objetivo de que pudiera tener algún uso en posibles trabajos desarrollados a partir de este trabajo de investigación.

Figura 10.4 Condición de inicio panel frontal

En la figura 10.4, el usuario establece la altura inicial con un valor en centímetros, también ingresa el ciclo de trabajo del motor inferior, es monitoreado el valor de la posición actual del cilindro (lectura del sensor ultrasónico), se muestra el ciclo de trabajo del motor superior asignado por el control y al mismo tiempo es graficado el comportamiento del cilindro en relación al valor del set-point. Para dar inicio a esta rutina es necesario activar el botón de *posición inicial*.

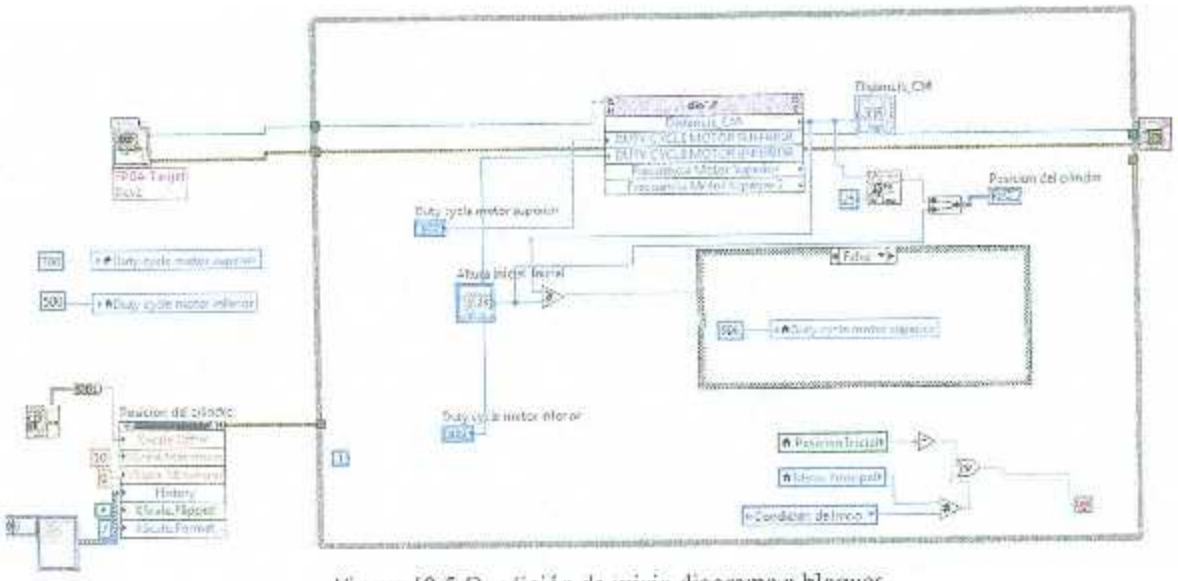


Figura 10.5 Condición de inicio diagrama a bloques.

En la imagen 10.5 se muestra el diagrama a bloques de la pestaña perteneciente a condición de inicio, en un principio se le asignan un ciclo de trabajo del 34.19% y 24.42% al motor superior e inferior respectivamente, posteriormente se abre el programa que se ejecutará en la tarjeta Spartan - 3E, el cual contiene los instrumentos virtuales (VI) encargados de monitorear las velocidades de ambos motores así como de la lectura de la posición del cilindro y el control PWM de cada motor, los cuales se explicarán más adelante. Básicamente el programa consiste en asignarles al motor superior un ciclo de trabajo del 31% o del 28% dependiendo si alcanzo o no la altura de referencia (set-point), como es un control bastante rudimentario y de respuesta lenta, solamente se utilizaría como condición inicial y no como control del sistema. También es graficado la respuesta del cilindro en relación al set-point, son modificadas algunas propiedades de las gráficas a efecto de una mejor visualización de la respuesta del

sistema, y para realizar esto es necesario agregar un property node de la gráfica llamada Posición del cilindro y modificar sus propiedades.

10.3 Modo de trabajo

El entrenamiento de la red neuronal, según la arquitectura del programa principal puede realizarse tanto en línea como fuera de línea, en caso de que el entrenamiento haya sido por lote (fuera de línea) es necesario ajustar los pesos sinápticos previamente en la pestaña llamada *modo de entrenamiento* una vez que se encontraron los pesos sinápticos ideales se detiene el entrenamiento de la red y se cambia a la pestaña de *modo de trabajo*.

En la planta en específico sobre la cual se trabajó en este proyecto de tesis, el entrenamiento de la red neuronal es realizado en línea, donde la red neuronal artificial trabaja en paralelo con un controlador PID, es por eso que el panel frontal de la imagen 10.6 cuenta con las especificaciones necesarias para que el controlador PID funcione.

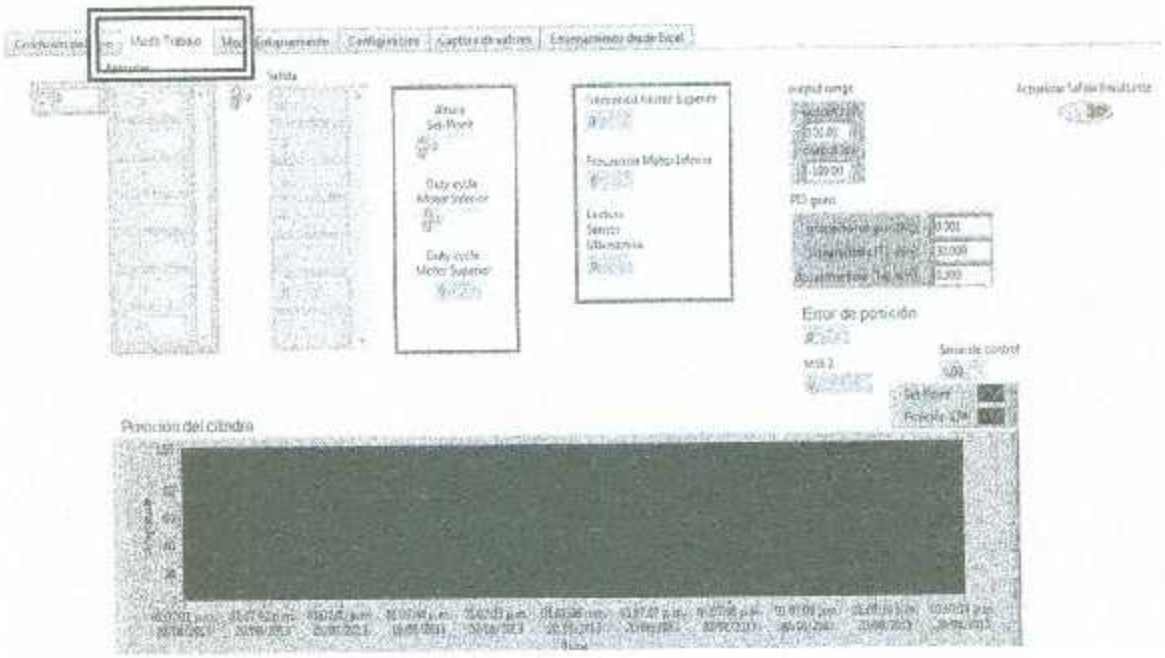


Figura 10.6 Modo de trabajo panel frontal.

En el panel frontal de la figura 10.6, es necesario especificar el set-point, el valor del ciclo de trabajo del motor inferior, así como las condiciones del trabajo del controlador PID, las cuales son explicadas en los capítulos de pruebas y resultados. Se grafica el comportamiento del cilindro.

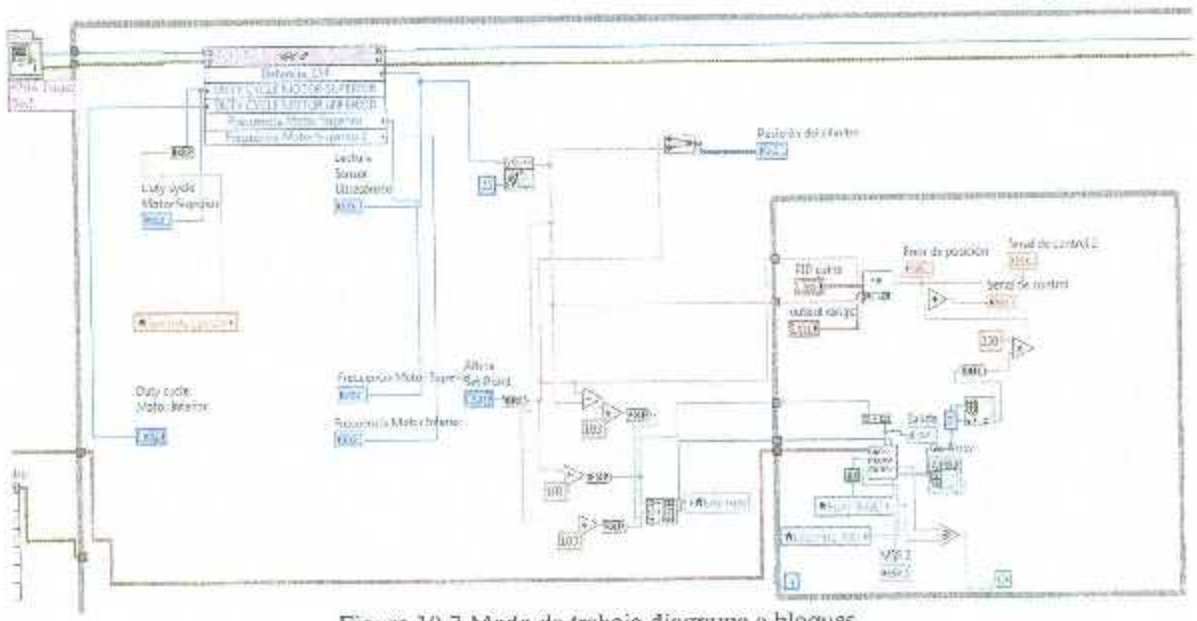


Figura 10.7 Modo de trabajo diagrama a bloques.

Dentro de un ciclo *while* es agregado un subVI el cual contiene el controlador PID y el subVI del algoritmo de *Backpropagation*, la salida de ambos controladores se suma y es asignado al ciclo de trabajo del motor superior.

La red neuronal es entrenada en línea con cada conjunto de datos capturados y monitoreados por la tarjeta FPGA Spartan - 3E.

10.3.1 Sub instrumento virtual *Backpropagation*

Fue implementado el algoritmo de entrenamiento *Backpropagation* el cual fue explicado en el capítulo 2, dicho algoritmo es una unión entre lenguaje de descripción de *hardware* y lenguaje gráfico.

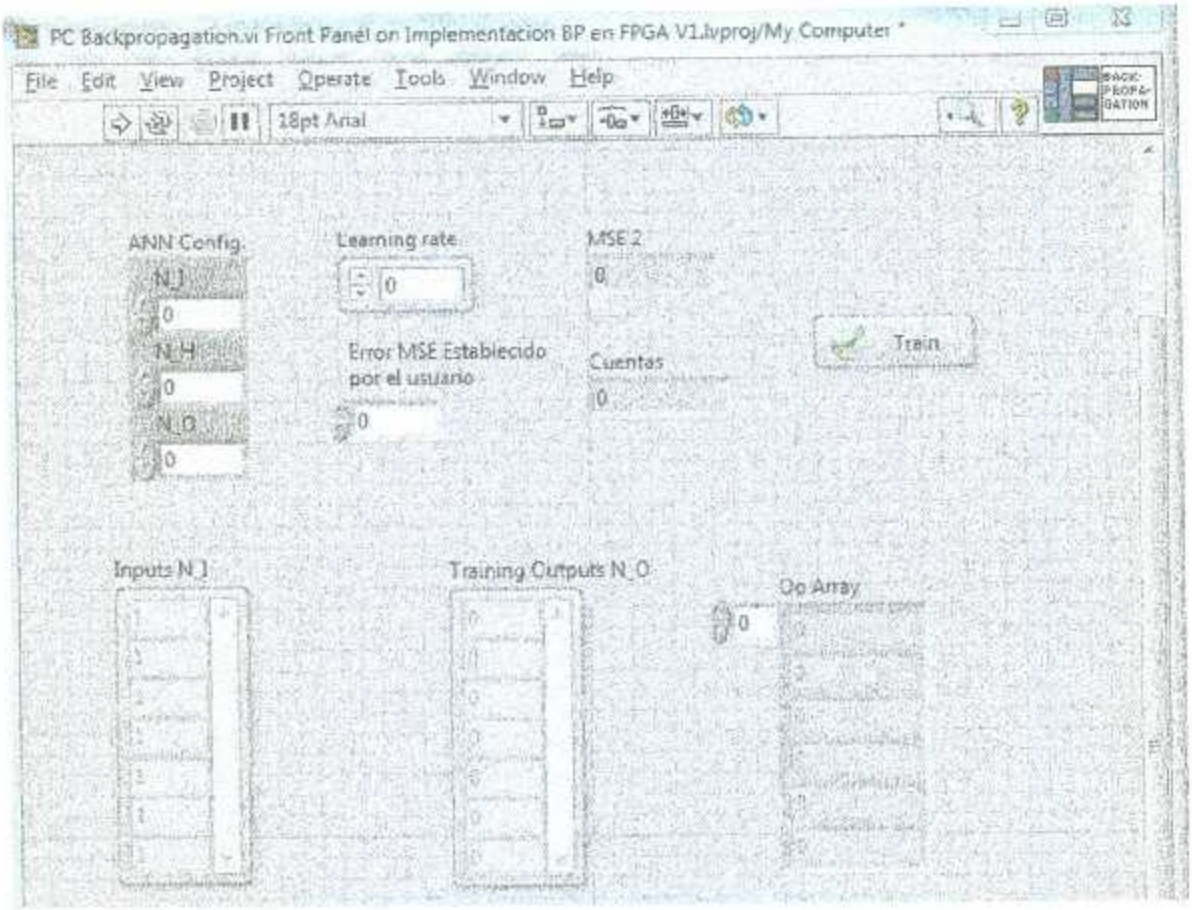


Figura 10.8 *Backpropagation* panel frontal.

Para realizar el entrenamiento de la red neuronal se manda llamar al sub-instrumento virtual (subVI) llamado *Backpropagation*, el cual necesita que sea especificado el número de neuronas de la capa de entrada, el número de neuronas de la capa oculta y el número en la capa de salida, estos datos son capturados en el ANN_Config.

Es necesario ingresar el valor del learning rate, así como el valor del MSE tanto el que determina el usuario como una constante como el que la red obtiene después de cada época de entrenamiento.

En los arreglos Inputs_NI contiene los valores de las entradas para el entrenamiento de la RNA, en el arreglo Training_Inputs N O se almacenan los valores de las salidas de entrenamiento. En el arreglo Oo_Array se muestra el valor de salida de la función de activación de la capa de salida. También se lleva un conteo del número de épocas que cada patrón de entrenamiento requiere antes de que se alcancen los pesos sinápticos ideales.

Se cuenta con un botón de activación llamado *Train* que da inicio al entrenamiento de la red neuronal.

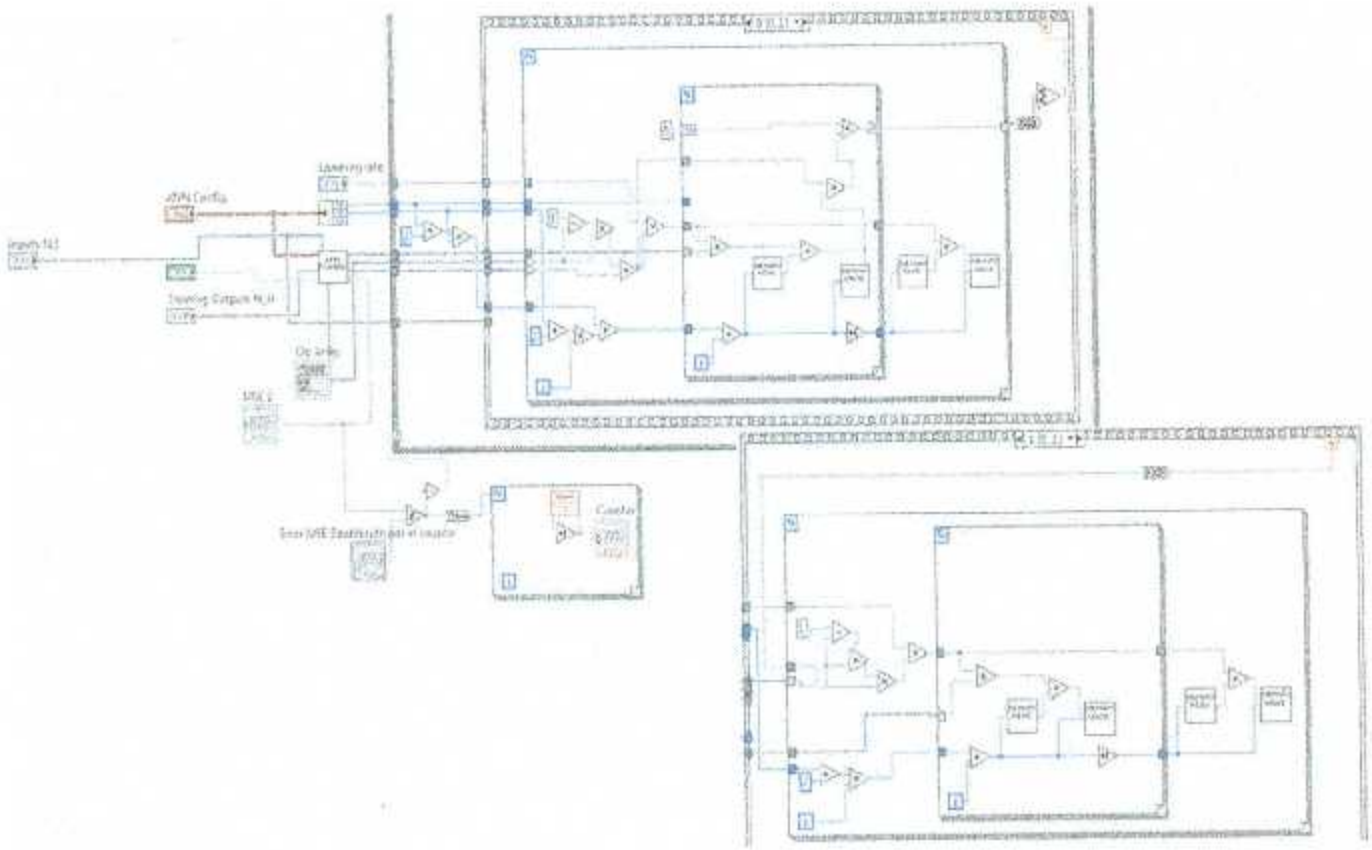


Figura 10.9 *Backpropagation* diagrama a bloques.

En la imagen 10.9, dentro de la estructura *case 0* se hace el cálculo de la suma ponderada de las neuronas de la capa intermedia y en *case 1* se realiza la suma ponderada de las neuronas que pertenecen a la capa de salida. El valor de los pesos sinápticos se almacenan en un arreglo general mandando llamar el subVI *Memory read* y cuando son recalculados dichos pesos se manda llamar a subVI *Memory write*, imagnes 10 y 11. Es necesario hacer la conversión de números de precisión simple SGL los cuales son de 32 bits, a números de punto fijos (fixed point) de 32 bits, siendo 16 bits para la parte entera y 16 bits para la parte flotante ya que la tarjeta FPGA Spartan - 3E no maneja números flotantes solamente almacena y manipula números de punto fijo.

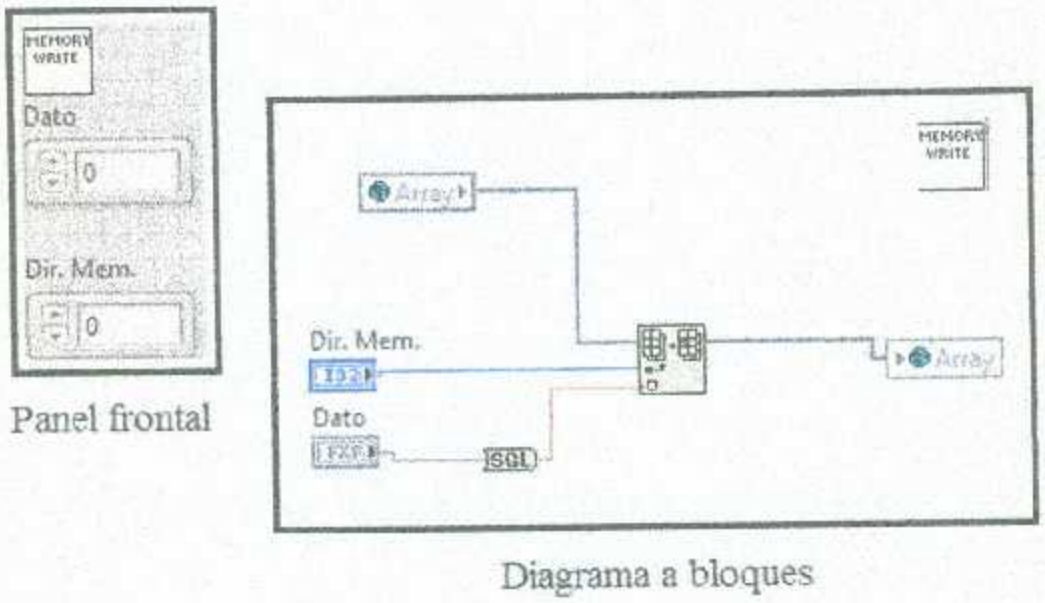


Figura 10.10 SubVI *Memory write*.

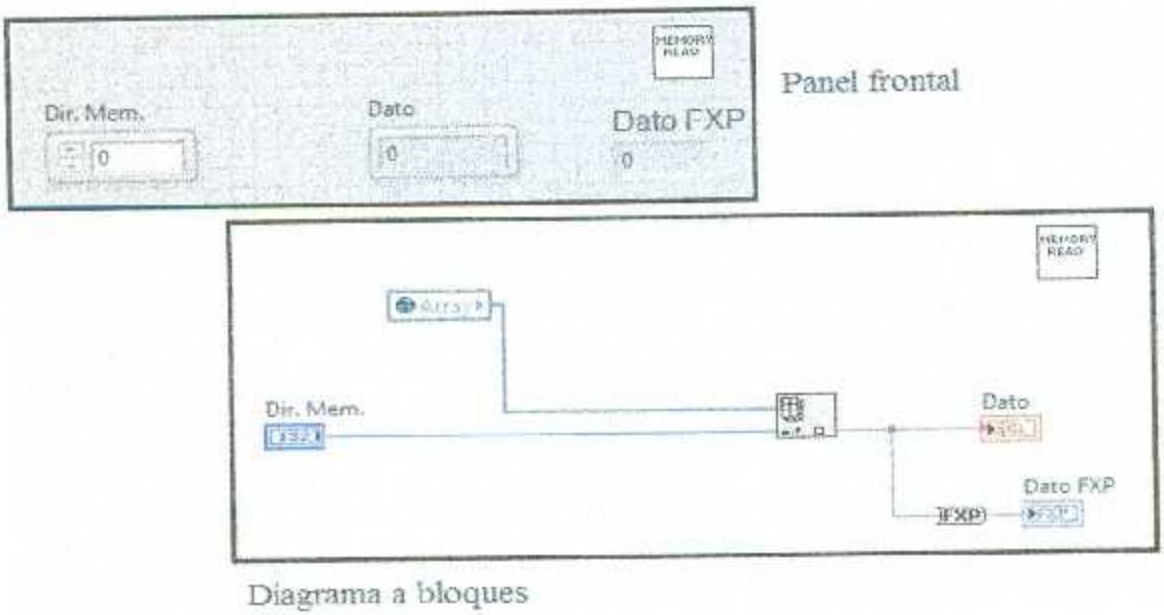


Figura 10.11 SubVI *Memory read*.

También en el subVI *Backpropagation* se manda llamear a otro sub instrumento llamado subVI *Feedforward* la cual es la primera etapa en el algoritmo de retropropagación o por su nombre en inglés. *Backpropagation*.

10.3.2 SubVI *Feedforward*

En la imagen 10.12, se presenta el panel frontal y diagrama a bloques del subVI *Feedforward*, en el cual se forma un cluster con tres controles numéricos referentes a la cantidad de neuronas de cada capa de la red neuronal.

El arreglo llamado Sh_Array almacena el valor de la suma ponderada de cada neurona de la capa oculta, dicha suma es ingresada a la función de activación sigmoide y es almacenada en el arreglo Oh_Array. Posteriormente el valor de la suma ponderada de la capa de salida se guarda en el arreglo So_Array y el valor numérico después de la función de activación la cual también es sigmoide va directo al arreglo Oo_Array.

En el arreglo llamado Entradas, es almacenado los valores de las variables que se considerarán para hacer el entrenamiento y en el arreglo Training Outputs N_O guarda los targets de cada conjunto de patrones de entrenamiento. En el modo de trabajo se manda llamar este sub-instrumento, con la diferencia que el arreglo llamado Training Outputs N_O no se almacena ningún valor.

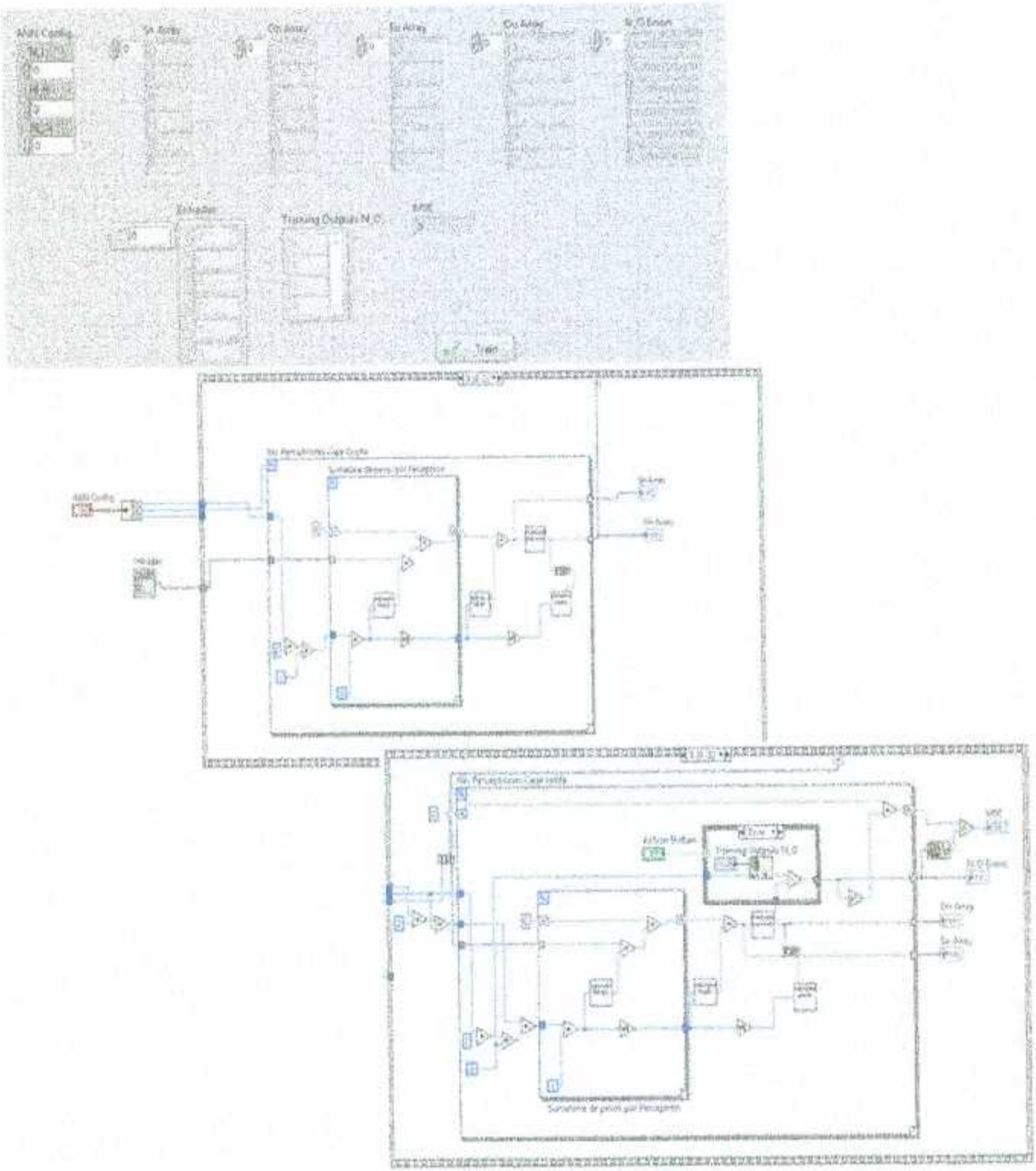


Figura 10.12 SubVI *Feedforward*

10.3.3 SubVI Función exponencial

La función de activación tanto de la capa oculta como de la capa de salida es la función sigmoide la cual necesita del número de Euler elevado a un número cualquiera tanto con valores positivos como negativos y es ejecutada en la tarjeta Spartan - 3E.Figura 10.13.

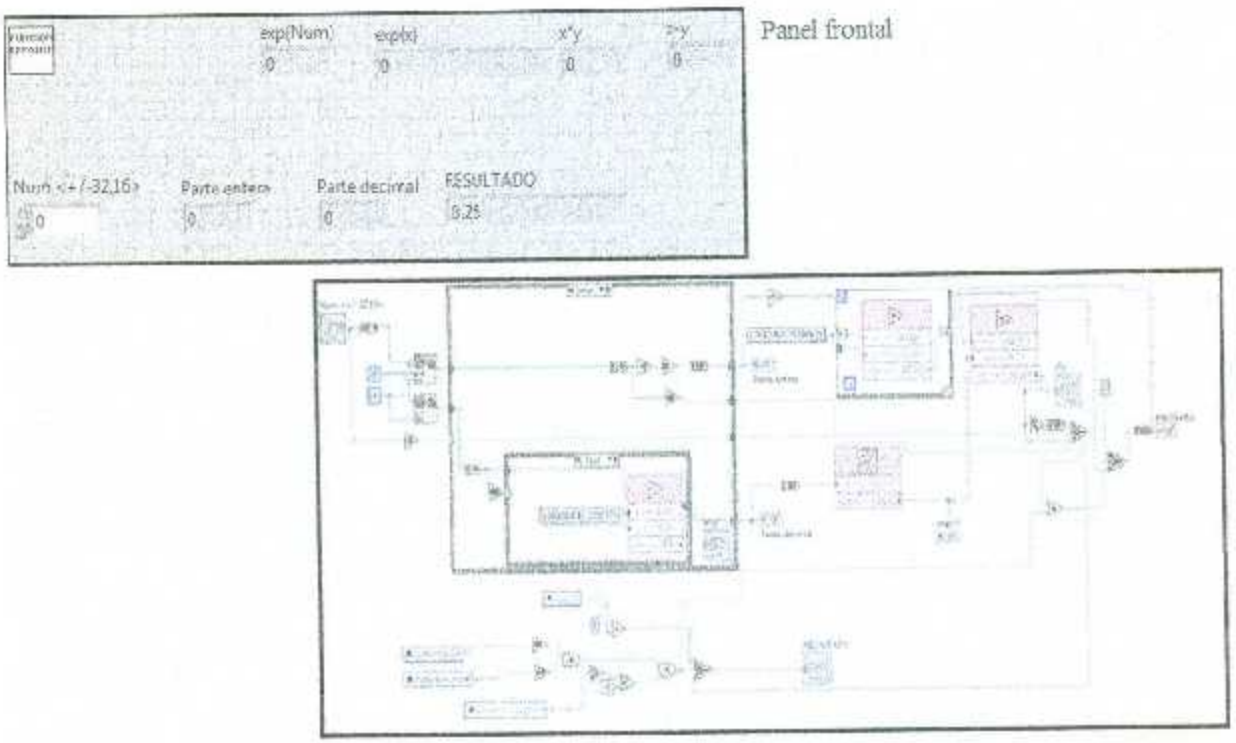


Figura 10.13 SubVI Función exponencial.

A pesar de que LabVIEW tiene el bloque del número de Euler para implementarse en la tarjeta FPGA, fue necesario realizar una serie de operaciones matemáticas ya que esta función de LabVIEW solo permite elevar el número de Euler a números entre 0 y 1, pero en el algoritmo de *Backpropagation* es necesario elevarlo al resultado de la suma ponderada de cada neurona de la RNA cualquiera que este valor sea.

10.3.4 SubVI Función sigmoide

La función de activación es la sigmoide imagen 10.14, la cual de una salida con un rango entre 0 y 1, dicha función es ejecutada en la tarjeta Spartan - 3E y manda llamar al SubVI de la función exponencial.

$$z = \frac{1}{1 + e^{-n}}$$

si n >> 0 entonces z → 1
si n << 0 entonces z → 0

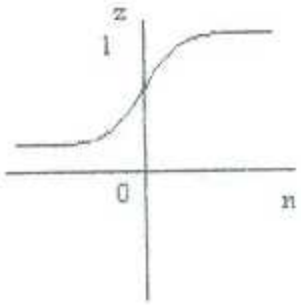


Figura 10.14 Función sigmoide.

En la imagen 10.15, se muestra tanto el panel frontal como el diagrama a bloques de la función sigmoide, en la cual se utilizaron algunos bloques específicos para implementarse en las tarjetas FPGA el cual nos permite delimitar el número de bits de cada posible resultado obtenido de una operación matemática, es decir, nos ayudan a truncar los valores de las diversas variables, ya que al hacer una implementación en *hardware* es muy importante delimitar el tamaño de los datos, porque la tarjeta FPGA tiene una cantidad de compuertas lógicas determinada las cual no puede expandirse, por lo tanto es necesario hacer estas de especificaciones en el tipo de datos a manejar.

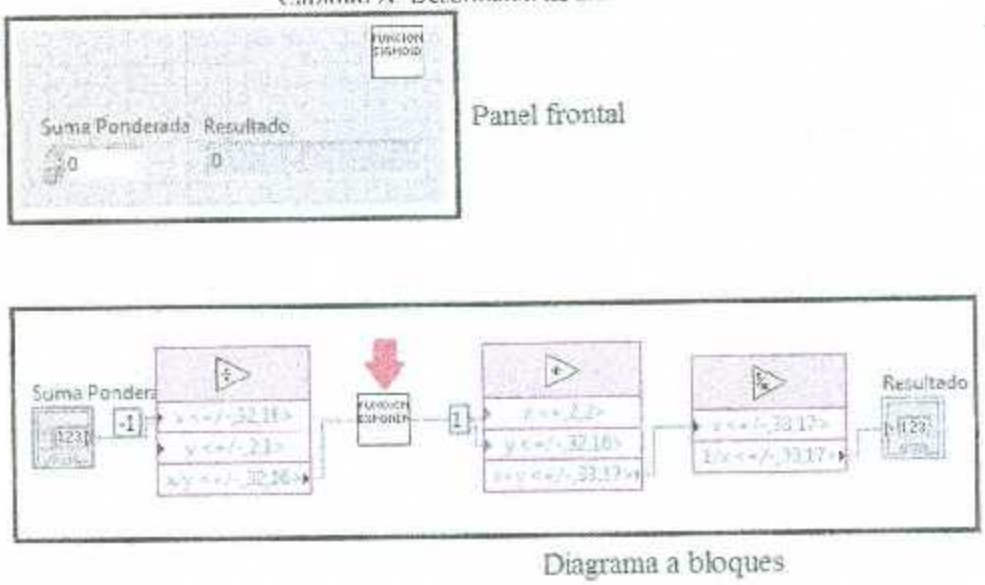


Figura 10.15 Función sigmoide panel frontal y diagrama a bloques.

10.3.5 SubVI PID

Fue tomado el bloque PID que LabVIEW 2011 SPI nos ofrece. Los valores de las ganancias fueron obtenidos en base a diversas pruebas que fueron realizadas anteriormente, quedando como la imagen 10.16 lo muestra.

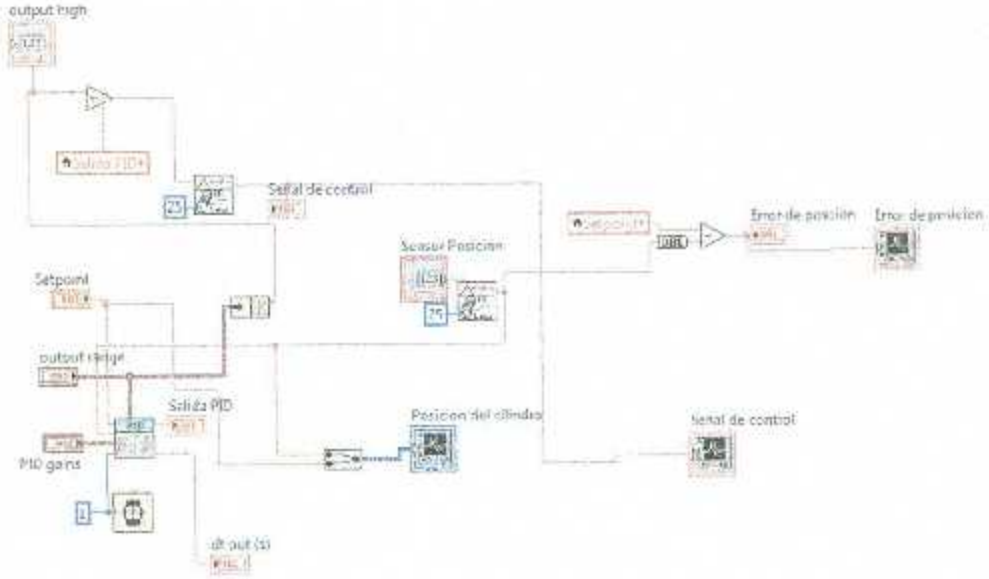


Figura 10.16 Diagrama a bloques de SubVI PID.

10.4 Modo entrenamiento

Esta pestaña dentro del tab-control, solamente es utilizada cuando el entrenamiento se va a realizar fuera de línea. El panel frontal cuenta con dos arreglos de datos los cuales están destinados a almacenar tanto los valore de las entradas y salidas de entrenamiento, así como un botón que da inicio a esta rutina llamado *Train* y un indicador led que muestra cuando el entrenamiento ha finalizado.

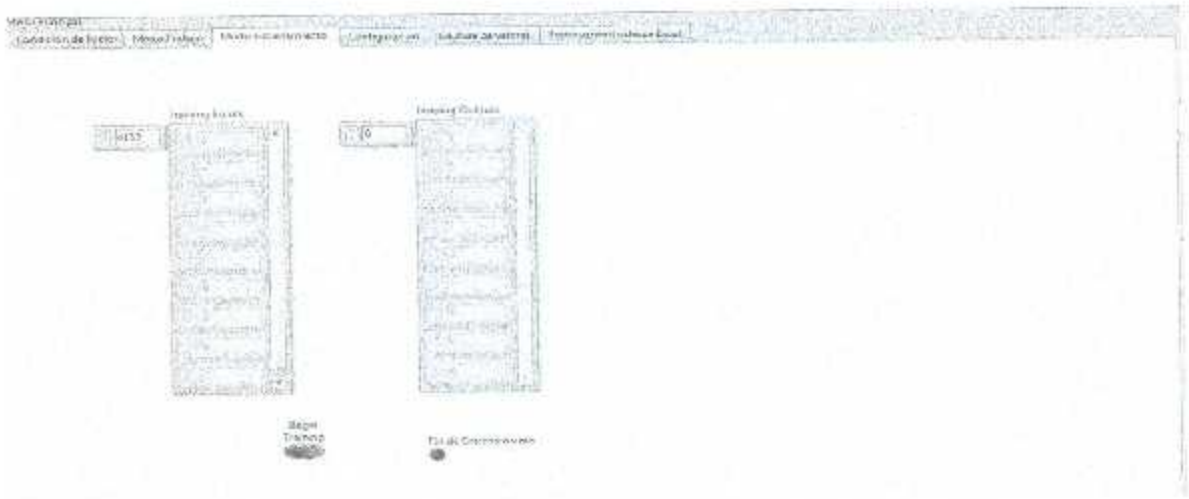


Figura 10.17 Modo de entrenamiento panel frontal.

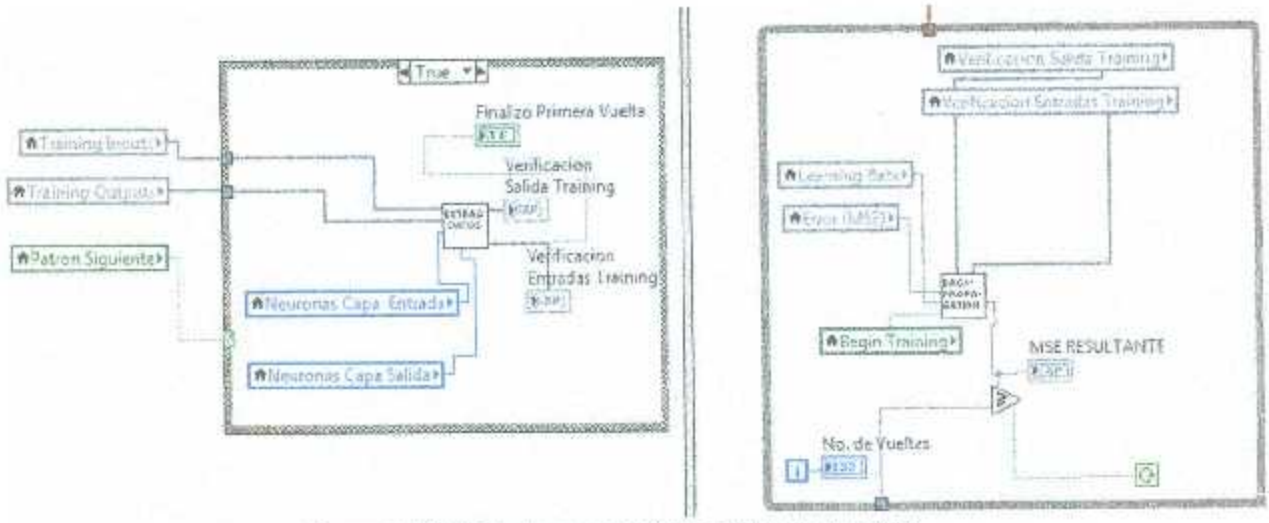


Figura 10.18 Modo de entrenamiento diagrama a bloques

En la imagen 10.18, se muestra la secuencia a seguir en el entrenamiento por lote de la red neuronal, en el cual previamente fueron capturados un conjunto de datos tanto de entrada y salida deseada. Es necesario ir tomando los datos por conjuntos del arreglo de *Training Inputs* y Training Outputs en cantidades iguales a la cantidad de neuronas en la capa de entrada y salida.

A la derecha de la imagen 10.18, se coloca dentro de un ciclo *while* el SubVI *Backpropagation* y el programa estará dentro del ciclo mientras que el error MSE obtenido al finalizar cada época sea mayor al MSE establecido por el usuario.

10.5 Configuración

La tercera pestaña dentro del tap control es la de configuración tanto de la arquitectura de la red y de las condiciones de entrenamiento de la RNA.

La arquitectura de la red neuronal tendrá la cantidad de neuronas en la capa de entrada en la misma cantidad que sensores se tengan montados en el prototipo, el número de neuronas en la capa intermedia se determina utilizando diversos teoremas y reglas heurísticas que fueron abordadas en el capítulo 2, y en el número de neuronas en la capa de salida es directamente proporcional a la cantidad de variables y/o actuadores a controlar.

Es también necesario ingresar los parámetros de entrenamiento, los cuales son: learning rate, MSE, y si los pesos sinápticos de las conexiones entre las neuronas tendrán un valor inicial establecido por el usuario o si se les asignará un valor aleatorio, imagen 10.19. El botón llamado Inicio asignación de pesos sinápticos da inicio a la rutina de configuración. El indicador led Final de asignación de pesos iniciales, indica cuando el fin de la rutina.

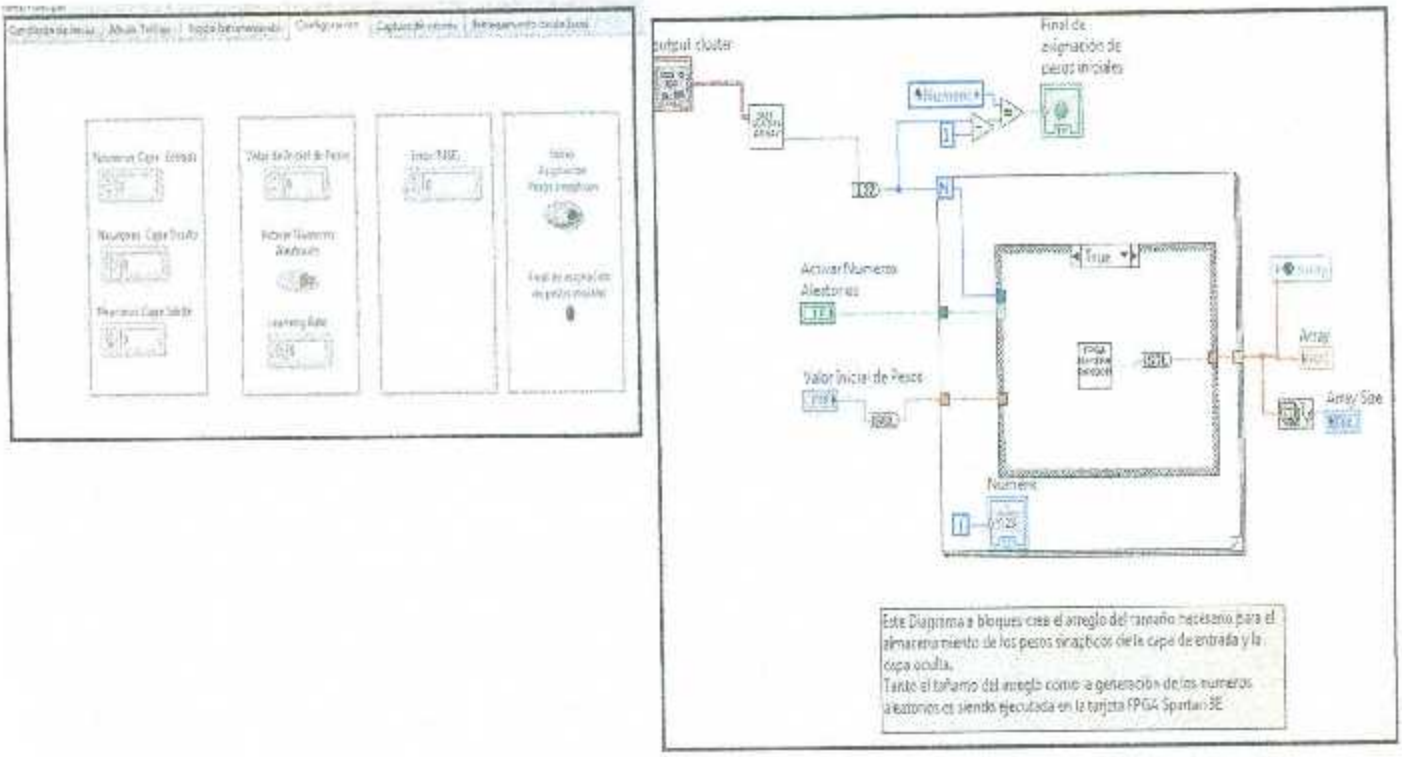


Figura 10 19 Configuración panel frontal

Para hacer la configuración de la red y la inicialización de los pesos sinápticos se mandan llamar dos SubVI, uno de ellos calcula el tamaño del arreglo global que contendrá los pesos sinápticos de todas las conexiones de la RNA, el segundo sub-instrumento genera números pseudoaletorios en la FPGA.

10.6 Captura de valores

La tercera pestaña en el tab-control captura los valores de los sensores así como el control PWM de los motores. Este VI interactúa directamente con el lenguaje de descripción de *hardware* VHDL de la plataforma de programación Xilinx *Ise Design Suite*.

El prototipo cuanta con dos frecuencímetros cada uno de ellos montados en el motor superior e inferior, posee un sensor ultrasónico para monitorear la posición del cilindro dentro del tubo

de acrílico, y tiene montados dos *Drivers* para el control de la velocidad de cada uno de los motores, la programación de todos estos elementos tanto de control como de monitoreo fueron programados en VHDL, en la figura 10.20 se muestra la programación en LabVIEW para el monitoreo de estos dispositivos.

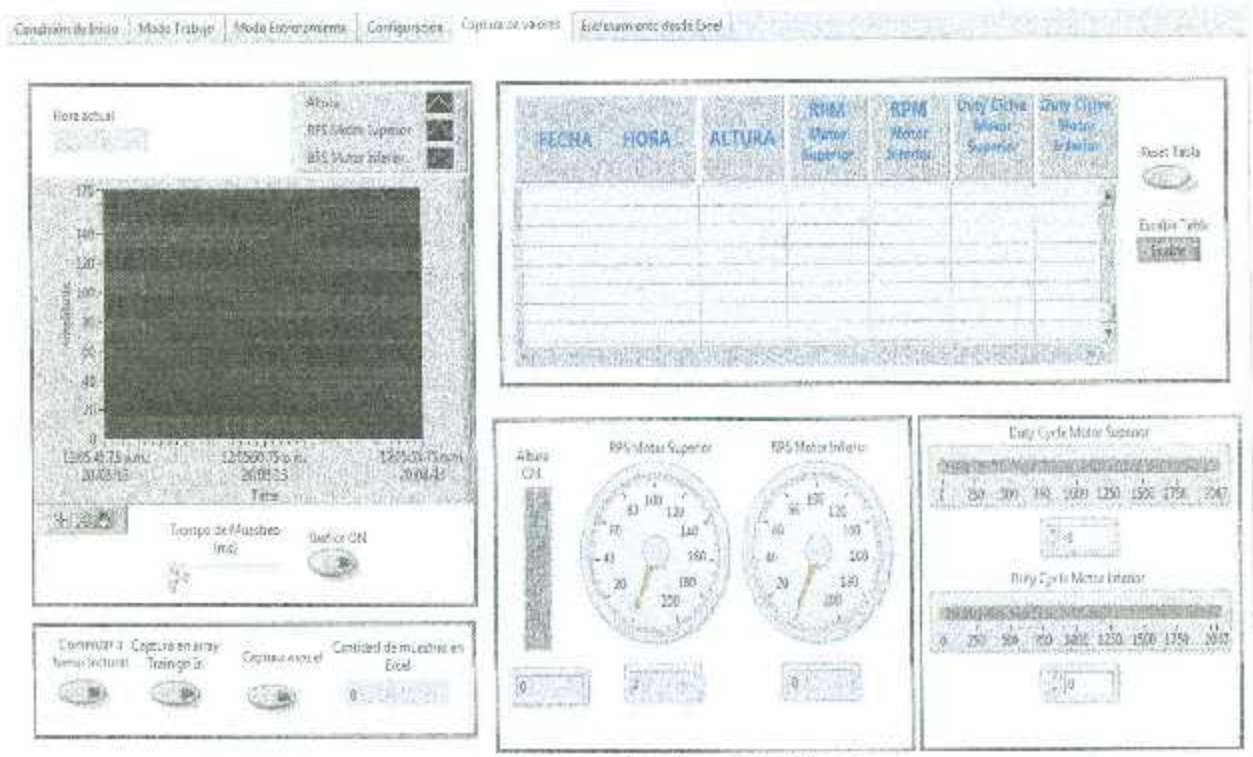


Figura 10.20 Captura de valores panel frontal

El panel frontal cuenta con los siguientes indicadores visuales: la gráfica indica el comportamiento del cilindro dentro del tubo, es decir, su posición en centímetros, también son graficados los cambios en las RPS de cada uno de los motores. Es posible modificar el tiempo de muestro en milisegundos, se tiene un *slide* el cual indica los cambios en la altura del cilindro, también se cuenta con dos *knob* que muestran las RPS de cada motor, los dos slides colocados a la derecha de la imagen 10.20 son utilizados para controlar la velocidad de cada motor.

Para dar inicio a la rutina de captura de datos, es necesario activar el botón de Comenzar a tomar lecturas. Este VI esta diseñado para hacer la captura de las velocidades de los motores y de la posición del cilindro y enviarlo directamente al arreglo de *Training Inputs* que está

colocado en la pestaña tres del tab-control llamado *Modo de entrenamiento*. De estar activo el botón llamado Captura en Excel, los datos son enviados a un libro de Excel.

10.7 Entrenamiento desde Excel

Con el objetivo de efectuar el entrenamiento de la red neuronal artificial fuera de linea es necesario sustraer los valores de Excel que fueron capturados previamente, en proporciones de datos iguales a la cantidad de neuronas de entrada y de salida.

En este VI es necesario especificar la cantidad de patrones de entrenamiento, ya que el programa está pensado de forma que si al usuario le interesa visualizar todos los patrones de entrenamiento que fueron capturados en una determina corrida del programa desde la interfaz en LabVIEW lo pueda hacer y que solo utilice la cantidad de patrones de entrenamiento que el considere conveniente, figura 10.21.

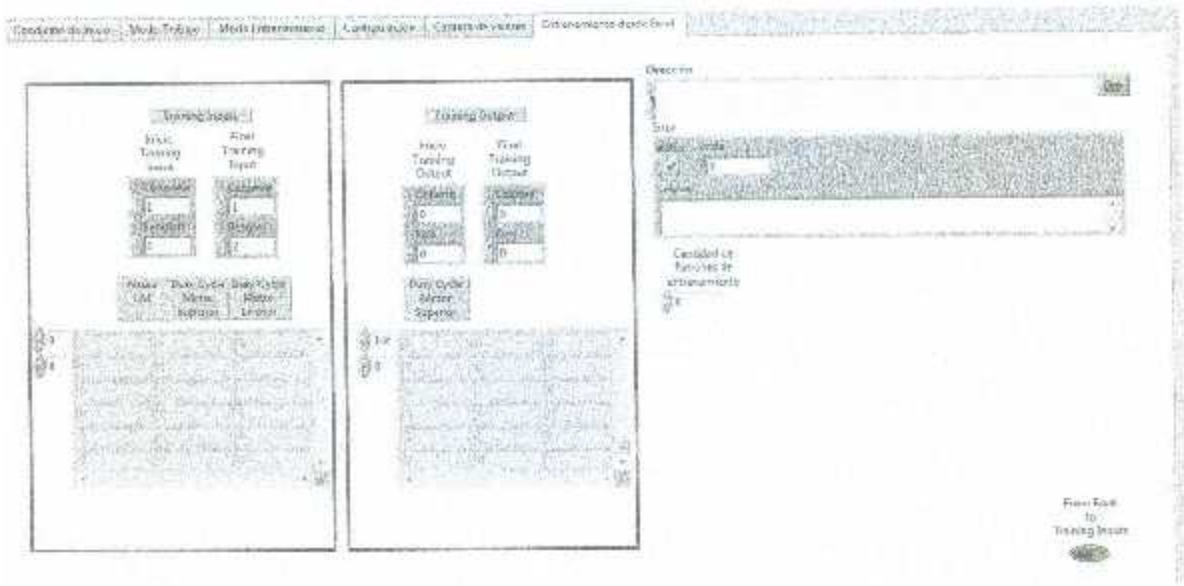


Figura 10.21 Entrenamiento desde Excel panel frontal.

En el apartado que dice *Dirección* se escribe la ruta de acceso al documento de Excel en el que se encuentra la información, es necesario especificar tanto las columnas como las filas en las que se encuentran los datos de entrada y salida para el entrenamiento de la red.

En el control numérico *Cantidad de patrones de entrenamiento* se indica el número de patrones que serán enviados a *Training Inputs* del modo de entrenamiento (pestaña 3 del tab-control).

Capítulo XI. Pruebas y resultados

11.1 Introducción

Desde el inicio del proyecto se planteó la idea de realizar el control total del sistema a través de la implementación de una red neuronal multicapa con un entrenamiento fuera de línea también llamado entrenamiento en lote, basado en el algoritmo de *Backpropagation*.

Posteriormente, al finalizar la construcción del prototipo y montaje de los sensores, se continuó con la etapa siguiente en el proyecto, la cual es la de pruebas y análisis de resultados obtenidos, para de esa manera realizar las modificaciones necesarias en el proyecto, refiriéndonos a modificaciones ya sea en la programación y/o alteraciones al prototipo.

En esta etapa fueron realizadas diversas pruebas que no mostraron los resultados esperados para el control del sistema basado solamente en la red neuronal multicapa, lo que nos llevó a explorar diversas configuraciones de controladores, de las cuales se hablará más a detalle en este capítulo obteniendo así resultados favorables concluyendo de forma exitosa el proyecto cumpliéndose el objetivo principal del mismo, el cual es mencionado en el capítulo uno de la tesis aquí expuesta

11.2 Configuraciones de controladores implementados

11.2.1 Red neuronal multicapa

La primera configuración propuesta es el control del sistema basado en una red neuronal multicapa la cual tendría la configuración de la imagen 11.1.

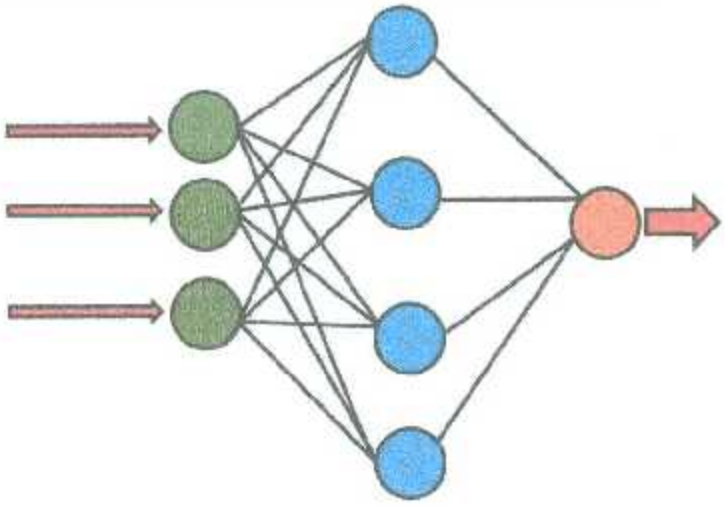


Figura 11.1.Configuración de red neuronal.

Es una red 3-4-1, es decir tres neuronas en la capa de entrada, cuatro neuronas en la capa de oculta y una neurona en la capa de salida, donde las neuronas de la capa de entrada corresponden al ciclo de trabajo del motor superior e inferior y la señal del sensor ultrasónico corresponde a la altura del cilindro confinado en el interior del tubo de acrílico, la capa de salida pertenece al ciclo de trabajo que se le asignará al motor superior.

El entrenamiento fuera de línea por lote, requiere que previo al entrenamiento se capture un conjunto de valores en forma de pares de entrada-salida y una vez que se considera que la red tiene el conjunto de pesos adecuado, se detiene el entrenamiento y se utiliza la red.
A continuación se muestran una serie de tablas para el análisis de los resultados obtenidos.

Tabla 11.1 Parte 1 captura de datos para entrenamiento de RNA.

No. de Iteración	Fecha	Hora	Altura	Duty Cycle Motor Superior	Duty Cycle Motor Inferior	RPS Motor Superior	RPA Motor Inferior
1	20/08/2013	11:50:28 a.m.	35	628	568	44	47
2	20/08/2013	11:50:28 a.m.	34	628	568	44	49
3	20/08/2013	11:50:28 a.m.	34	628	568	44	49
4	20/08/2013	11:50:28 a.m.	35	628	568	44	49
5	20/08/2013	11:50:28 a.m.	35	628	568	44	49

6	20/08/2013	11:50:28 a.m.	35	628	568	44	49
7	20/08/2013	11:50:28 a.m.	36	628	568	44	49
8	20/08/2013	11:50:28 a.m.	35	628	568	44	49
9	20/08/2013	11:50:28 a.m.	36	628	568	44	49
10	20/08/2013	11:50:28 a.m.	36	628	568	44	49
11	20/08/2013	11:50:28 a.m.	36	628	568	44	49
12	20/08/2013	11:50:28 a.m.	36	628	568	44	49
13	20/08/2013	11:50:28 a.m.	36	628	568	44	49
14	20/08/2013	11:50:28 a.m.	36	628	568	44	49
15	20/08/2013	11:50:28 a.m.	35	628	568	44	49
16	20/08/2013	11:50:28 a.m.	36	628	568	44	49
17	20/08/2013	11:50:29 a.m.	33	628	568	44	49
18	20/08/2013	11:50:29 a.m.	33	628	568	44	49
19	20/08/2013	11:50:29 a.m.	37	628	568	44	49
20	20/08/2013	11:50:29 a.m.	36	628	568	44	50
21	20/08/2013	11:50:29 a.m.	36	628	568	44	49
22	20/08/2013	11:50:29 a.m.	11	628	568	44	49
23	20/08/2013	11:50:29 a.m.	36	628	568	44	49
24	20/08/2013	11:50:29 a.m.	37	628	568	44	50
25	20/08/2013	11:50:29 a.m.	37	628	568	44	50
26	20/08/2013	11:50:29 a.m.	36	628	568	44	49
27	20/08/2013	11:50:29 a.m.	36	628	568	44	50

La tabla 11.1 se encuentra integrada por las siguientes columnas, donde la primera de ellas indica el número de muestra, posteriormente se captura la fecha y hora de captura de la información, después la columna que indica la altura del cilindro confinado dentro del tubo cuya medida es en centímetros, se indica de igual manera el ciclo de trabajo del motor superior e inferior, en donde el ciclo de trabajo del motor al 100% equivale al número de cuenta 2047, por lo tanto los números que se expresan en las columnas 5 y 6 equivalen al número/2047 para expresar el porcentaje del ciclo de trabajo, y por ultimo las revoluciones por segundo del motor superior e inferior, respectivamente.

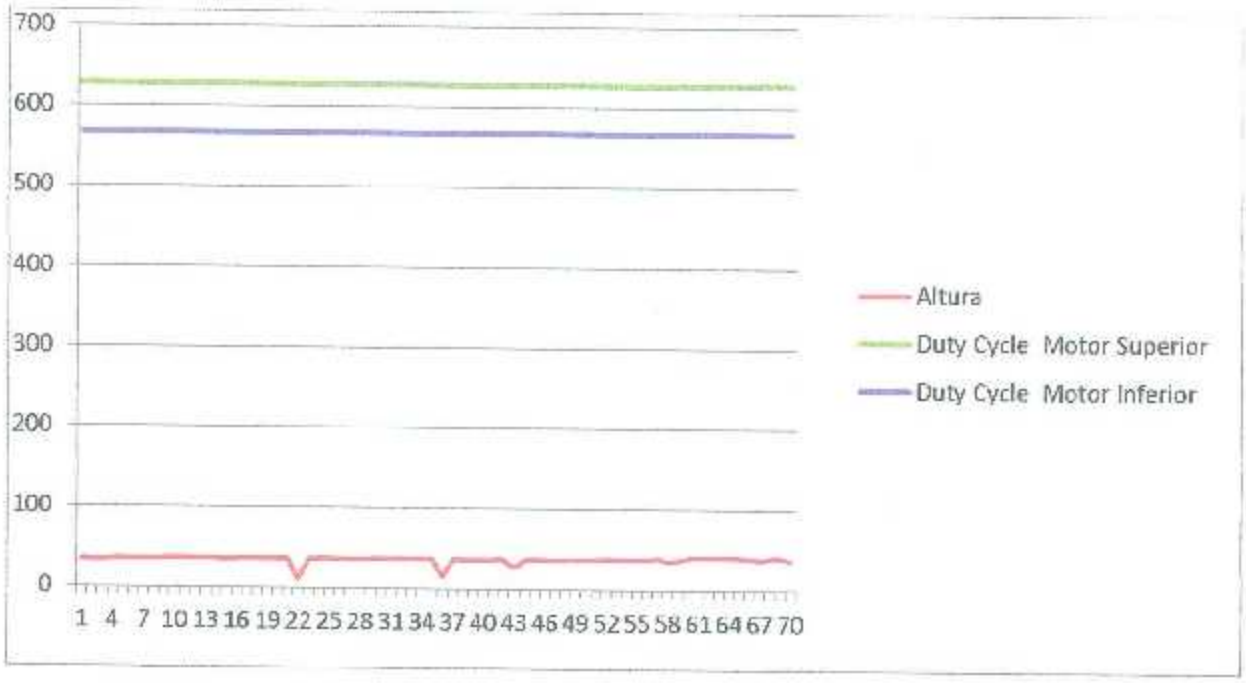
Se aprecia que la altura varía en el tiempo, a pesar de que el ciclo de trabajo tanto del motor superior e inferior se mantiene constante al igual que las revoluciones por segundo de ambos motores.

Tabla 11.2 Parte 2 captura de datos para entrenamiento de RNA.

No. de Iteración	Fecha	Hora	Altura	Duty Cycle Motor Superior	Duty Cycle Motor Inferior	RPS Motor Superior	RPA Motor Inferior
46	20/08/2013	11:50:30 a.m.	37	628	568	43	48
47	20/08/2013	11:50:30 a.m.	36	628	568	43	48
48	20/08/2013	11:50:30 a.m.	37	628	568	43	48
49	20/08/2013	11:50:30 a.m.	37	628	568	43	48
50	20/08/2013	11:50:30 a.m.	37	628	568	43	48
51	20/08/2013	11:50:30 a.m.	38	628	568	43	48
52	20/08/2013	11:50:30 a.m.	38	628	568	43	48
53	20/08/2013	11:50:31 a.m.	38	628	568	43	48
54	20/08/2013	11:50:31 a.m.	38	628	568	43	48
55	20/08/2013	11:50:31 a.m.	38	628	568	43	48
56	20/08/2013	11:50:31 a.m.	38	628	568	44	48
57	20/08/2013	11:50:31 a.m.	40	628	568	44	48
58	20/08/2013	11:50:31 a.m.	40	628	568	44	48
59	20/08/2013	11:50:31 a.m.	40	628	568	44	48
60	20/08/2013	11:50:31 a.m.	40	628	568	44	48
61	20/08/2013	11:50:31 a.m.	40	628	568	44	48
62	20/08/2013	11:50:31 a.m.	40	628	568	44	48
63	20/08/2013	11:50:31 a.m.	40	628	568	44	48
64	20/08/2013	11:50:31 a.m.	40	628	568	44	48
65	20/08/2013	11:50:31 a.m.	40	628	568	44	48
66	20/08/2013	11:50:31 a.m.	39	628	568	44	48
67	20/08/2013	11:50:31 a.m.	38	628	568	44	48
68	20/08/2013	11:50:31 a.m.	40	628	568	44	48
69	20/08/2013	11:50:31 a.m.	40	628	568	44	48
70	20/08/2013	11:50:31 a.m.	38	628	568	44	48
71	20/08/2013	11:50:32 a.m.	39	628	568	44	48

Debido a que la cantidad de datos que fueron capturados es muy grande, ya que el muestreo es realizado cada 10ms, solamente se analizan porciones del contenido de las tablas capturadas en Excel y que expresan los cambios más significativos en el sistema. En la segunda parte de la tabla 11.1, expresada en la tabla 11.2 pasamos a la iteración 46, donde se continúan apreciando las variaciones en la atura a pesar de que los ciclos de trabajo se mantenían constantes.

Grafica 11.1 Ciclo de trabajo constante de motor 1 y motor 2.



Se llegó a la conclusión de que aunque se mantuvieran las velocidades de ambos extractores, nunca se mantendría una altura constante ya que en esa parte es donde entra el control el cual estará variando las velocidades de un motor o de ambos para permanecer el objeto a una altura determinada.

Por lo tanto se decidió cambiar de estrategia y mantener un motor apagado y solamente variar la velocidad del extractor superior, obteniéndose los siguientes resultados, tabla 11.3.

Tabla 11.3 Parte 1 Captura de datos para entrenamiento de RNA.

No. de Iteración	Fecha	Hora	Altura	Duty Cycle Motor Superior	Duty Cycle Motor Inferior	RPS Motor Superior	RPA Motor Inferior
1	20/08/2013	11:46:15 a.m.	102	500	0	22	0
2	20/08/2013	11:46:15 a.m.	101	500	0	22	0
3	20/08/2013	11:46:15 a.m.	101	500	0	22	0

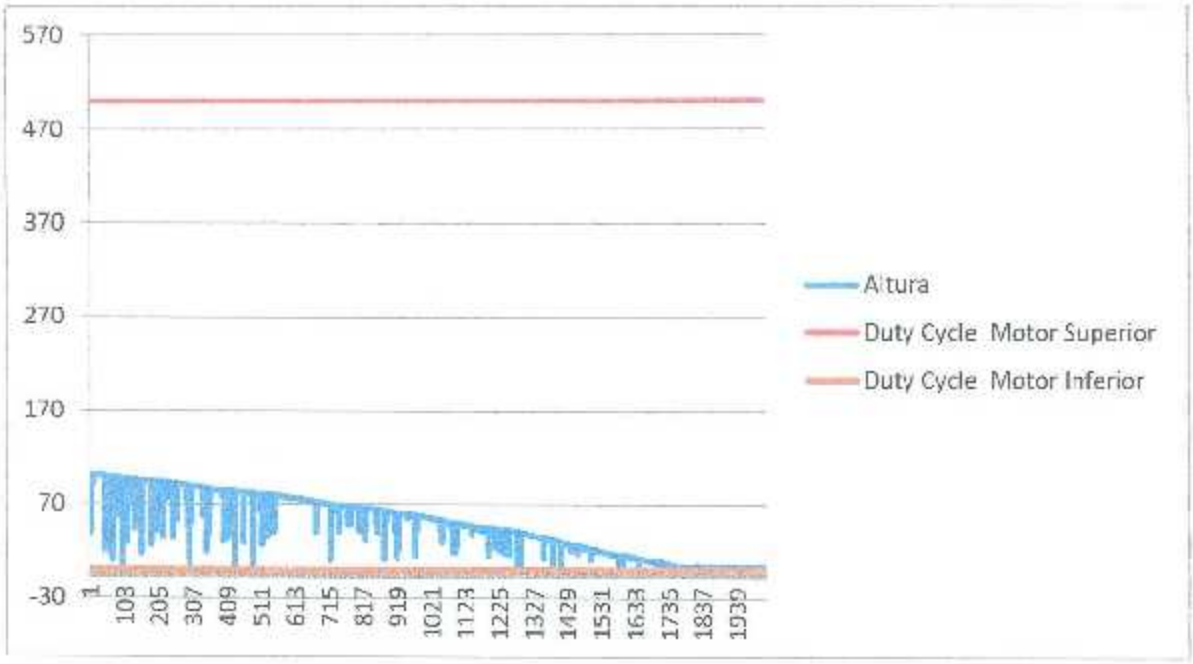
Fueron tomadas 2025 muestras, donde la altura final fue de 3 cm, como se indica en la última parte de la tabla original donde fueron extraídos los datos, tabla 11.4.

Tabla 11.4 Parte 2 Captura de datos para entrenamiento de RNA.

No. de Iteración	Fecha	Hora	Altura	Duty Cycle Motor Superior	Duty Cycle Motor Inferior	RPS Motor Superior	RPA Motor Inferior
2023	20/08/2013	11:48:03 a.m.	3	500	0	29	0
2024	20/08/2013	11:48:03 a.m.	3	500	0	29	0
2025	20/08/2013	11:48:03 a.m.	3	500	0	29	0

Para ejemplificar de una forma más general, fue generada una gráfica la cual contiene los valores de las 2025 muestras.

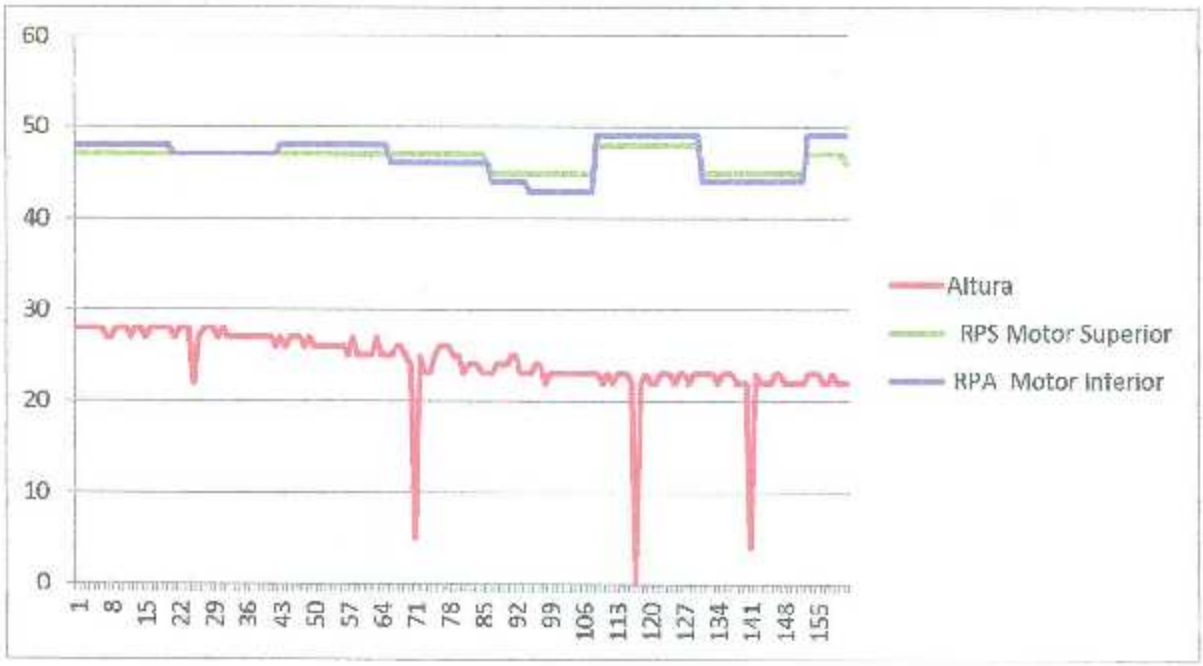
Grafica 11.2. Tendencia de valores de datos.



Analizando el comportamiento de los datos en la gráfica 11.2, tampoco se logró mantener una altura constante a pesar de que uno de los motores se encontraba apagado.

Se tenía la teoría de que si ambos motores giraban a una frecuencia similar el cilindro se posicionaría a la mitad del tubo, pero como se muestra en la gráfica 11.3, aunque tuvieran una frecuencia de giro similar los dos motores el cilindro siempre se encuentra inclinado hacia uno de los extremos.

Grafica 11.3 Ambos motores a frecuencia de giro similar.



También se realizaron pruebas en las cuales se alteraba de forma manual las velocidades de ambos motores con el objetivo de mantener una altura contante, pero lamentablemente el control que se necesita implementar para dar estabilización al sistema es demasiado rápido, por lo cual de hacerse de forma manual resulta casi imposible obtener datos veridicos.

A pesar de que los valores de las muestras capturadas anteriormente no mostraban estabilidad, se prosiguió a realizar el entrenamiento de la red neuronal artificial.

La planta propuesta consistió en una red neuronal entrenada fuera de linea con el objetivo de minimizar el error, el cual es la diferencia entre la salida deseada yd(t) también llamada set-point y la salida obtenida y(t), imagen 11.2.

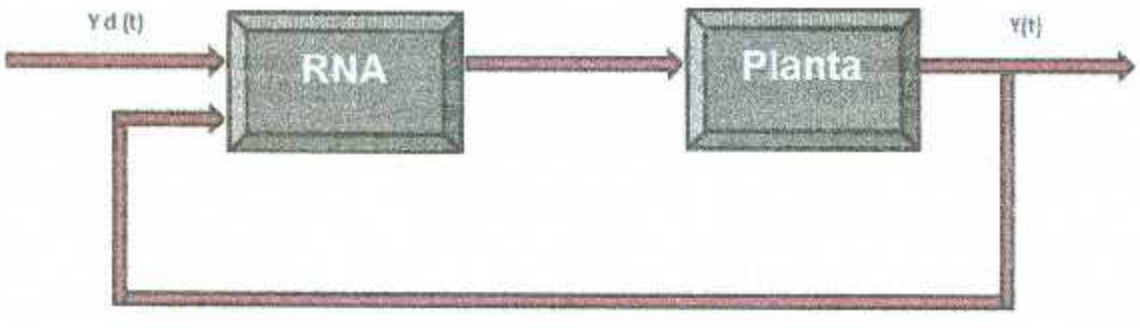


Figura 11.2 Control RNA.

Debido a que los datos que fueron capturados para obtener los patrones de entrenamiento de la red neuronal no le brindaron la información suficiente al control para que este actuara de forma adecuada ante cualquier perturbación y fuera capaz de posicionar el objeto dentro del tubo de acrílico en la altura que el usuario estableciera como set-point, por lo tanto se consideró necesario obtener más datos para el entrenamiento de la red neuronal multicapa implementando otro tipo de controlador.

11.3 Control con PID

Con el objetivo de obtener una mayor cantidad de información para realizar el entrenamiento fuera de línea de la red neuronal multicapa basada en el algoritmo de *Backpropagation*, se estableció un control PID para que regulara el funcionamiento de la planta, figura 11.3.

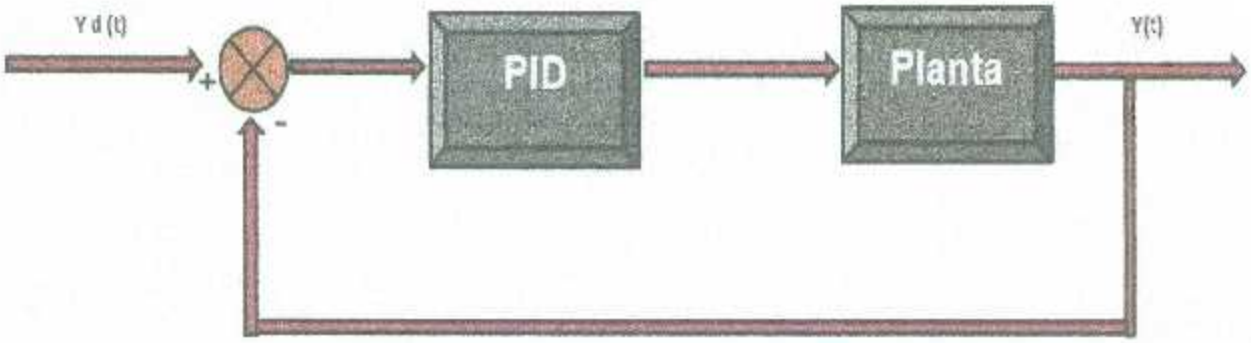


Figura 11.3 Control PID.

Enseguida fueron capturados los siguientes valores para su análisis, el muestreo se estableció cada 10 ms. Tabla 11.5.

Tabla 11.5 Parte 1 datos obtenidos control PID.

No. de Iteración	Fecha	Hora	Set-Point	Posición Actual	Error de posición	Señal de control	Ciclo de trabajo Motor Superior	Ciclo de trabajo Motor Inferior	RPS Motor Superior	RPA Motor Inferior
1	22/07/2013	09:54:46 a.m.	5	4	0	628.614	628.61	500	44	36
2	22/07/2013	09:54:46 a.m.	5	4	0	628.614	628.61	500	44	36
3	22/07/2013	09:54:46 a.m.	5	4	0	628.614	628.61	500	44	36
4	22/07/2013	09:54:46 a.m.	5	4	0	628.614	628.61	500	44	36
5	22/07/2013	09:54:46 a.m.	5	4	1	628.614	628.61	500	44	36
6	22/07/2013	09:54:46 a.m.	5	4	1	628.614	628.61	500	44	36
7	22/07/2013	09:54:46 a.m.	5	4	1	628.614	628.61	500	44	36
8	22/07/2013	09:54:46 a.m.	5	4	1	628.614	628.61	500	44	36
9	22/07/2013	09:54:46 a.m.	5	4	1	628.614	628.61	500	44	36
10	22/07/2013	09:54:46 a.m.	5	4	1	628.614	628.61	500	44	36
11	22/07/2013	09:54:46 a.m.	5	4	1	628.614	628.61	500	44	36
12	22/07/2013	09:54:46 a.m.	5	4	1	628.614	628.61	500	44	36
13	22/07/2013	09:54:46 a.m.	5	4	1	628.614	628.61	500	44	36
14	22/07/2013	09:54:46 a.m.	5	4	1	628.614	628.61	500	44	36
15	22/07/2013	09:54:46 a.m.	5	4	1	628.614	628.61	500	44	36
16	22/07/2013	09:54:46 a.m.	5	4	1	628.614	628.61	500	44	36
17	22/07/2013	09:54:46 a.m.	5	4	1	628.614	628.61	500	44	36
18	22/07/2013	09:54:46 a.m.	5	4	1	617.567	617.57	500	44	36

19	22/07/2013	09:54:46 a.m.	5	4	1	617.535	617.54	500	44	36
20	22/07/2013	09:54:46 a.m.	5	4	1	617.505	617.51	500	44	36

En la tabla 11.5, se aprecia el número de iteración, la fecha y hora de captura de los datos, el set-point que es la altura de referencia que el usuario ingresa (altura deseada) de forma manual al control, la columna siguiente corresponde a la lectura en un instante determinado del sensor ultrasónico (altura obtenida), la sexta columna corresponde al error de posición el cual es la diferencia entre la altura deseada y la obtenida, después se muestra el ciclo de trabajo del motor superior el cual es un valor cambiante ya que depende directamente de la salida del controlador PID, el ciclo de trabajo del motor inferior se mantuvo constante ya que en el proceso real solamente la velocidad del motor VTI o motor residual (motor superior en el prototipo) es cambiante y la velocidad el grupo de ventiladores de enfriamiento (motor inferior del prototipo) es contante, también fue capturada la frecuencia de giro de ambos motores en forma de revoluciones por segundo.

En la tabla 11.5, se observa que el valor de set point fijado es de 5 cm. y la lectura del sensor ultrasónico es de 4 cm, marcando como error de posición el valor de 1, a partir de la iteración 17 se aprecia un cambio en el valor del ciclo de trabajo del motor superior indicando que el control PID empieza a reaccionar al error de posición correspondiente, debido a que el cambio a la salida del controlador PID es cada 10ms, en la tabla 1 no se alcanza a apreciar el cambio en la velocidad del motor superior (motor 1), debido a que los motores responden de manera más lenta a los cambios en su velocidad.

La velocidad del motor inferior se mantiene constante con un ciclo de trabajo del 24.43%.

Tabla 11.6. Parte 2 Datos obtenidos control PID

No. de Iteración	Fecha	Hora	Set-Point	Posición Actual	Error de posición	Señal de control	Ciclo de trabajo Motor Superior	Ciclo de trabajo Motor Inferior	RPS Motor Superior	RPA Motor Inferior
60	22/07/2013	09:54:47 a.m.	5	4	1	616.314	616.31	500	44	36
61	22/07/2013	09:54:47 a.m.	5	4	1	616.284	616.28	500	44	36
62	22/07/2013	09:54:47 a.m.	5	4	1	616.253	616.25	500	45	40
63	22/07/2013	09:54:47 a.m.	5	4	1	616.222	616.22	500	45	40
64	22/07/2013	09:54:47 a.m.	5	4	1	616.192	616.19	500	45	40
65	22/07/2013	09:54:47 a.m.	5	4	1	616.161	616.16	500	45	40
66	22/07/2013	09:54:47 a.m.	5	4	1	616.131	616.13	500	45	40
67	22/07/2013	09:54:47 a.m.	5	4	1	616.1	616.1	500	45	40
68	22/07/2013	09:54:47 a.m.	5	4	1	616.07	616.07	500	45	40
69	22/07/2013	09:54:47 a.m.	5	4	1	616.039	616.04	500	45	40
70	22/07/2013	09:54:47 a.m.	5	4	1	616.009	616.01	500	45	40
71	22/07/2013	09:54:47 a.m.	5	4	1	615.978	615.98	500	45	40
72	22/07/2013	09:54:47 a.m.	5	4	1	615.947	615.95	500	45	40
73	22/07/2013	09:54:47 a.m.	5	4	1	615.917	615.92	500	45	40
74	22/07/2013	09:54:47 a.m.	5	4	1	615.886	615.89	500	45	40
75	22/07/2013	09:54:47 a.m.	5	4	1	615.856	615.86	500	45	40
76	22/07/2013	09:54:47 a.m.	5	4	1	615.825	615.83	500	45	40
77	22/07/2013	09:54:47 a.m.	5	4	1	615.795	615.79	500	45	40
78	22/07/2013	09:54:47 a.m.	5	4	1	615.764	615.76	500	45	40
79	22/07/2013	09:54:47 a.m.	5	4	1	615.734	615.73	500	45	40
80	22/07/2013	09:54:47 a.m.	5	4	1	615.703	615.7	500	45	40

Debido a que fueron capturados una gran cantidad de datos, solamente se presentaran los valores más significativos y que denotaron un cambio en el comportamiento del sistema, es por esto que en la tabla 11.5 la iteración final es la número 20 y en la tabla 11.6 se continua con la iteración número 60, este mecanismo de selección de datos continuará en las siguientes tablas de comportamiento. Pero es importante resaltar que el entrenamiento de la red se realizó con todos los valores de la tabla.

En la tabla 11.6, se aprecia que el valor de salida del controlador PID continua cambiando, sin embargo el valor del error de posición es el mismo por lo que la salida del PID se seguirá modificando hasta que el error de posición sea cero. En la iteración 62 se aprecia un cambio en la velocidad de motor superior del prototipo.

Tabla 11.7 Parte 3 datos obtenidos control PID

No. de Iteración	Fecha	Hora	Set-Point	Posición Actual	Error de posición	Señal de control	Ciclo de trabajo Motor Superior	Ciclo de trabajo Motor Inferior	RPS Motor Superior	RPA Motor Inferior
251	22/07/2013	09:54:49 a.m.	5	4	1	610.478	610.48	500	43	40
252	22/07/2013	09:54:50 a.m.	5	4	1	610.448	610.45	500	43	40
253	22/07/2013	09:54:50 a.m.	5	4	1	610.417	610.42	500	43	40
254	22/07/2013	09:54:50 a.m.	5	4	1	610.387	610.39	500	43	40
255	22/07/2013	09:54:50 a.m.	5	4	1	610.356	610.36	500	43	40
256	22/07/2013	09:54:50 a.m.	5	4	1	610.326	610.33	500	43	40
257	22/07/2013	09:54:50 a.m.	5	4	1	610.295	610.3	500	43	40
258	22/07/2013	09:54:50 a.m.	5	4	1	610.265	610.26	500	43	40
259	22/07/2013	09:54:50 a.m.	5	4	1	610.234	610.23	500	43	40
260	22/07/2013	09:54:50 a.m.	5	4	1	610.203	610.2	500	43	40
261	22/07/2013	09:54:50 a.m.	5	4	1	610.173	610.17	500	43	40
262	22/07/2013	09:54:50 a.m.	5	4	1	610.142	610.14	500	43	40
263	22/07/2013	09:54:50 a.m.	5	4	1	610.112	610.11	500	43	40
264	22/07/2013	09:54:50 a.m.	5	4	1	610.081	610.08	500	43	40
265	22/07/2013	09:54:50 a.m.	5	4	1	610.051	610.05	500	43	40
266	22/07/2013	09:54:50 a.m.	5	4	1	610.02	610.02	500	43	40
267	22/07/2013	09:54:50 a.m.	5	4	1	609.99	609.99	500	43	40

En la tabla 11.7, se aprecia que el ciclo de trabajo del motor superior sigue disminuyendo, pero el error de posición continúa siendo 1.

Posteriormente se cambia de forma dramática el valor del set-point pasando de 5 a 95 cm, con el objetivo de capturar todos los posibles errores de posición disponibles, o que en

determinado momento se le podían presentar al sistema, ya que esto dependería de la posición de referencia que el usuario le asignará.

Tabla 11.8 Parte 4 datos obtenidos control PID.

No. de Iteración	Fecha	Hora	Set-Point	Posición Actual	Error de posición	Señal de control	Ciclo de trabajo Motor Superior	Ciclo de trabajo Motor Inferior	RPS Motor Superior	RPA Motor Inferior
266	22/07/2013	09:54:50 a.m.	5	4	1	610.02	610.02	500	43	40
267	22/07/2013	09:54:50 a.m.	5	4	1	609.99	609.99	500	43	40
268	22/07/2013	09:54:50 a.m.	95	4	1	609.959	608.96	500	43	40
269	22/07/2013	09:54:50 a.m.	95	4	91	609.928	609.93	500	43	40
270	22/07/2013	09:54:50 a.m.	95	4	91	609.898	609.9	500	43	40
271	22/07/2013	09:54:50 a.m.	95	4	91	609.867	609.87	500	43	40
272	22/07/2013	09:54:50 a.m.	95	4	91	609.837	609.84	500	43	40
273	22/07/2013	09:54:50 a.m.	95	4	91	609.806	609.81	500	43	40
274	22/07/2013	09:54:50 a.m.	95	4	91	609.776	609.78	500	43	40
275	22/07/2013	09:54:50 a.m.	95	4	91	609.745	609.75	500	43	40
276	22/07/2013	09:54:50 a.m.	95	4	91	609.715	609.71	500	43	40
277	22/07/2013	09:54:50 a.m.	95	4	91	609.684	609.68	500	43	40
278	22/07/2013	09:54:50 a.m.	95	4	91	609.653	609.65	500	43	40
279	22/07/2013	09:54:50 a.m.	95	5	90	609.623	609.62	500	43	40
280	22/07/2013	09:54:50 a.m.	95	5	90	609.592	609.59	500	43	40
281	22/07/2013	09:54:50 a.m.	95	5	90	609.562	609.56	500	43	40

En la iteración 268 de la tabla 11.8, se cambia el valor del set-point a 95 cm obteniéndose como nuevo error de posición 91 posiciones, por lo que el controlador PID deberá de responder a este cambio tan grande en la magnitud del error de posición en el sistema.

Tabla 11.9 Datos obtenidos control PID.

281	22/07/2013	09:54:50 a.m.	95	5	90	609.562	609.56	500	43	40
282	22/07/2013	09:54:50 a.m.	95	5	90	0	0	500	43	0
283	22/07/2013	09:54:50 a.m.	95	4	91	0	0	500	43	0
284	22/07/2013	09:54:50 a.m.	95	4	91	0	0	500	43	0
285	22/07/2013	09:54:50 a.m.	95	4	91	0	0	500	43	0

286	22/07/2013	09:54:50 a.m.	95	4	91	0	0	500	43	3
287	22/07/2013	09:54:50 a.m.	95	4	91	0	0	500	43	0
288	22/07/2013	09:54:50 a.m.	95	4	91	0	0	500	43	0
289	22/07/2013	09:54:50 a.m.	95	4	91	0	0	500	43	0
290	22/07/2013	09:54:50 a.m.	95	4	91	0	0	500	43	0
291	22/07/2013	09:54:50 a.m.	95	4	91	0	0	500	43	2
292	22/07/2013	09:54:50 a.m.	95	4	91	0	0	500	43	0
293	22/07/2013	09:54:50 a.m.	95	4	91	0	0	500	43	0

En la iteración 282 de la tabla 11.9 se ilustra que la señal de control (columna 7) y el ciclo de trabajo del motor superior (columna 8) cambian a 0, hubo un cambio muy brusco en sus valores de pasar de 609 a 0, debido a que el error de posición alcanzo su máximo valor.

En este punto el motor superior se encuentra apagado y el motor inferior está girando de forma constante y como el diseño de las aspas de los motores están pensadas para que funcionen como extractores el objeto se encuentra completamente unido a la parte inferior del tubo de acrílico, por lo que será necesario ir aumentando el valor del ciclo de trabajo del motor superior para que pueda alcanzar el set-point de 95cm.

Tabla 11.10 Parte 5 datos obtenidos control PID.

No. de Iteración	Fecha	Hora	Set-Point	Posición Actual	Error de posición	Señal de control	Ciclo de trabajo Motor Superior	Ciclo de trabajo Motor Inferior	RPS Motor Superior	RPA Motor Inferior
328	22/07/2013	09:54:51 a.m.	95	10	85	0	0	500	43	39
329	22/07/2013	09:54:51 a.m.	95	11	84	0	0	500	43	39
330	22/07/2013	09:54:51 a.m.	95	11	84	0	0	500	43	39
331	22/07/2013	09:54:51 a.m.	95	11	84	0.10877	0.11	500	43	39
332	22/07/2013	09:54:51 a.m.	95	11	84	2.73635	2.74	500	43	39
333	22/07/2013	09:54:51 a.m.	95	11	84	2.82801	2.83	500	43	39
334	22/07/2013	09:54:51 a.m.	95	13	82	2.82801	2.83	500	43	39
335	22/07/2013	09:54:51 a.m.	95	13	82	2.82801	2.83	500	43	39
336	22/07/2013	09:54:51 a.m.	95	13	82	5.54725	5.55	500	43	39
337	22/07/2013	09:54:51 a.m.	95	13	82	11.3389	11.34	500	43	39
338	22/07/2013	09:54:51 a.m.	95	13	81	11.3389	11.34	500	43	39
339	22/07/2013	09:54:51 a.m.	95	13	81	11.3389	11.34	500	43	39

340	22/07/2013	09:54:51 a.m.	95	13	81	11.3389	11.34	500	43	39
341	22/07/2013	09:54:51 a.m.	95	13	81	11.3389	11.34	500	43	39
342	22/07/2013	09:54:51 a.m.	95	11	84	12.4425	12.44	500	43	39
343	22/07/2013	09:54:51 a.m.	95	17	78	11.3389	11.34	500	43	39
344	22/07/2013	09:54:51 a.m.	95	17	78	11.3389	11.34	500	43	39
345	22/07/2013	09:54:51 a.m.	95	17	78	11.3389	11.34	500	43	39
346	22/07/2013	09:54:51 a.m.	95	17	78	11.3389	11.34	500	43	39
347	22/07/2013	09:54:51 a.m.	95	18	77	11.3389	11.34	500	43	39
348	22/07/2013	09:54:51 a.m.	95	18	77	12.4425	12.44	500	43	39

En la tabla 11.10, en la iteración 331 se ejemplifica como el control PID empieza a aumentar el valor de su salida pasando de 0 a 0.10877 y continúa aumentado su magnitud como se documenta en la iteración 348 donde alcanza el valor 12.44. Ya se había mencionado en párrafos anteriores que la salida del controlador PID es directamente proporcional al ciclo de trabajo del motor superior por lo que las columnas 7 y 8 tendrán valores muy próximos unos del otro.

Tabla 11.11. Parte 6 datos obtenidos control PID.

No. de Iteración	Fecha	Hora	Set-Point	Posición Actual	Error de posición	Señal de control	Ciclo de trabajo Motor Superior	Ciclo de trabajo Motor Inferior	RPS Motor Superior	RPA Motor Inferior
350	22/07/2013	09:54:51 a.m.	95	18	77	14.4119	14.41	500	43	39
351	22/07/2013	09:54:51 a.m.	95	20	75	13.9971	14	500	43	39
352	22/07/2013	09:54:51 a.m.	95	20	75	12.4425	12.44	500	43	39
353	22/07/2013	09:54:51 a.m.	95	20	75	11.95	11.95	500	43	39
354	22/07/2013	09:54:51 a.m.	95	20	75	11.95	11.95	500	43	39
355	22/07/2013	09:54:51 a.m.	95	20	75	9.84548	9.85	500	43	39
356	22/07/2013	09:54:51 a.m.	95	22	73	9.84548	9.85	500	43	39
357	22/07/2013	09:54:51 a.m.	95	22	73	9.84548	9.85	500	43	39
358	22/07/2013	09:54:51 a.m.	95	22	73	9.84548	9.85	500	43	39
359	22/07/2013	09:54:51 a.m.	95	22	73	9.84548	9.85	500	43	39
360	22/07/2013	09:54:51 a.m.	95	25	70	9.84548	9.85	500	43	39
361	22/07/2013	09:54:51 a.m.	95	25	70	9.84548	9.85	500	43	39
362	22/07/2013	09:54:51 a.m.	95	25	70	9.84548	9.85	500	43	39
363	22/07/2013	09:54:51 a.m.	95	25	7	11.95	11.95	500	43	39
364	22/07/2013	09:54:51 a.m.	95	9	86	14.4554	14.46	500	43	39

365	22/07/2013	09:54:51 a.m.	95	27	68	16.9607	16.96	500	43	39
366	22/07/2013	09:54:51 a.m.	95	29	65	36.3388	36.34	500	22	42
367	22/07/2013	09:54:51 a.m.	95	29	66	36.3388	36.34	500	22	42
368	22/07/2013	09:54:51 a.m.	95	33	62	37.9141	37.91	500	22	42
369	22/07/2013	09:54:51 a.m.	95	33	62	38.722	38.72	500	22	42
370	22/07/2013	09:54:51 a.m.	95	33	62	40.2667	40.27	500	22	42
371	22/07/2013	09:54:51 a.m.	95	33	62	41.1051	41.11	500	22	42
372	22/07/2013	09:54:51 a.m.	95	33	62	42.6193	42.62	500	22	42
373	22/07/2013	09:54:51 a.m.	95	35	60	42.6193	42.62	500	22	42
374	22/07/2013	09:54:51 a.m.	95	35	60	46.1366	46.14	500	22	42
375	22/07/2013	09:54:51 a.m.	95	35	60	48.4281	48.43	500	22	42
376	22/07/2013	09:54:51 a.m.	95	35	60	50.7196	50.72	500	22	42
377	22/07/2013	09:54:51 a.m.	95	18	77	53.0111	53.01	500	22	42
378	22/07/2013	09:54:51 a.m.	95	38	57	55.3026	55.3	500	22	42
379	22/07/2013	09:54:51 a.m.	95	38	57	59.1865	59.19	500	22	42
380	22/07/2013	09:54:51 a.m.	95	38	57	61.4169	61.42	500	22	42
381	22/07/2013	09:54:51 a.m.	95	38	57	61.4169	61.42	500	22	42
382	22/07/2013	09:54:51 a.m.	95	41	54	63.6473	63.65	500	22	42

En la tabla 11.11, la columna 5 que corresponde a la señal del sensor ultrasónico, esta va aumentando su valor indicando oscilaciones en el comportamiento de la posición del objeto, es decir un sub-amortiguamiento. La columna 8 de igual manera muestra cambios de incremento y decremento en sus valores que de graficarse tendria una forma senoidal.

Hasta la iteración 366 se ven reflejados los cambios de la señal de control en la velocidad de giro del motor superior.

Tabla 11.12 Parte 7 datos obtenidos control PID.

No. de Iteración	Fecha	Hora	Set-Point	Posición Actual	Error de posición	Señal de control	Ciclo de trabajo Motor Superior	Ciclo de trabajo Motor Inferior	RPS Motor Superior	RPA Motor Inferior
482	22/07/2013	09:54:53 a.m.	95	99	-4	644.307	644.31	500	2	43
483	22/07/2013	09:54:53 a.m.	95	99	-4	664.659	664.66	500	2	43
484	22/07/2013	09:54:53 a.m.	95	101	-6	664.873	664.87	500	2	43
485	22/07/2013	09:54:53 a.m.	95	101	-6	664.873	664.87	500	2	43

486	22/07/2013	09:54:53 a.m.	95	101	-6	665.087	665.09	500	2	43
487	22/07/2013	09:54:53 a.m.	95	101	-6	665.3	665.3	500	2	43
488	22/07/2013	09:54:53 a.m.	95	57	37	708.052	708.05	500	2	43
489	22/07/2013	09:54:53 a.m.	95	101	-6	708.143	708.14	500	2	43
490	22/07/2013	09:54:53 a.m.	95	101	-6	708.143	708.14	500	2	43
491	22/07/2013	09:54:53 a.m.	95	101	-6	708.235	708.24	500	2	43
492	22/07/2013	09:54:53 a.m.	95	101	-6	708.327	708.33	500	2	43
493	22/07/2013	09:54:53 a.m.	95	87	-8	708.418	708.42	500	2	43
494	22/07/2013	09:54:53 a.m.	95	101	-6	729.87	729.87	500	2	43
495	22/07/2013	09:54:53 a.m.	95	101	-6	729.87	729.87	500	2	43
496	22/07/2013	09:54:53 a.m.	95	101	-6	729.901	729.9	500	2	43
497	22/07/2013	09:54:53 a.m.	95	101	-6	729.932	729.93	500	2	43
498	22/07/2013	09:54:53 a.m.	95	101	-5	729.962	729.96	500	2	43
499	22/07/2013	09:54:53 a.m.	95	101	-5	729.962	729.96	500	2	43
500	22/07/2013	09:54:53 a.m.	95	101	-6	762.95	762.95	500	2	43
501	22/07/2013	09:54:53 a.m.	95	101	-6	763.011	763.01	500	2	43
502	22/07/2013	09:54:53 a.m.	95	19	-6	763.072	763.07	500	2	43
503	22/07/2013	09:54:53 a.m.	95	101	-6	763.133	763.13	500	2	43

En la columna 6 de la tabla 11.12 se presenta el evento en el cual el error de posición es negativo, en este caso el motor superior (columna 8) deberá aumentar su velocidad para compensar la diferencia entre la altura deseada y la obtenida, además que al estar el objeto casi al ras de la superficie inferior la fuerza de succión que ejerce el motor inferior es mayor por lo que el motor superior deberá de compensar esta succión aumentando su velocidad de giro.

Tabla 11.13 Parte 8 datos obtenidos control PID.

No. de Iteración	Fecha	Hora	Set-Point	Posición Actual	Error de posición	Señal de control	Ciclo de trabajo Motor Superior	Ciclo de trabajo Motor Inferior	RPS Motor Superior	RPA Motor Inferior
537	22/07/2013	09:54:53 a.m.	95	99	-4	811.781	811.78	500	21	48
538	22/07/2013	09:54:53 a.m.	95	99	-4	811.781	811.78	500	21	48
539	22/07/2013	09:54:53 a.m.	95	98	-3	811.781	811.78	500	21	48
540	22/07/2013	09:54:53 a.m.	95	98	-3	811.781	811.78	500	21	48
541	22/07/2013	09:54:53 a.m.	95	98	-3	811.781	811.78	500	21	48
542	22/07/2013	09:54:53 a.m.	95	98	-3	811.781	811.78	500	21	48
543	22/07/2013	09:54:53 a.m.	95	0	-4	811.781	811.78	500	21	48

544	22/07/2013	09:54:53 a.m.	95	96	-1	811.781	811.78	500	21	48
545	22/07/2013	09:54:53 a.m.	95	96	1	811.781	811.78	500	21	48
546	22/07/2013	09:54:53 a.m.	95	96	-1	803.912	803.91	500	21	48
547	22/07/2013	09:54:53 a.m.	95	96	-1	803.759	500803.76		21	48
548	22/07/2013	09:54:53 a.m.	95	95	0	803.606	803.61	500	21	48
549	22/07/2013	09:54:53 a.m.	95	95	0	803.454	803.45	500	21	48
550	22/07/2013	09:54:53 a.m.	95	95	0	803.301	803.3	500	21	48
551	22/07/2013	09:54:54 a.m.	95	95	0	794.271	794.27	500	21	48
552	22/07/2013	09:54:54 a.m.	95	95	0	794.148	794.15	500	21	48
553	22/07/2013	09:54:54 a.m.	95	95	0	794.026	794.03	500	21	48
554	22/07/2013	09:54:54 a.m.	95	95	0	793.904	793.9	500	21	48
555	22/07/2013	09:54:54 a.m.	95	95	0	793.782	793.78	500	21	48
556	22/07/2013	09:54:54 a.m.	95	95	0	793.66	793.66	500	21	48
557	22/07/2013	09:54:54 a.m.	95	95	0	793.537	793.54	500	21	48
558	22/07/2013	09:54:54 a.m.	95	93	2	793.415	793.42	500	21	48
559	22/07/2013	09:54:54 a.m.	95	93	2	793.415	793.42	500	21	48
560	22/07/2013	09:54:54 a.m.	95	93	2	793.293	793.29	500	21	48
561	22/07/2013	09:54:54 a.m.	95	93	1	793.171	793.17	500	21	48
562	22/07/2013	09:54:54 a.m.	95	54	41	783.652	783.65	500	21	48
563	22/07/2013	09:54:54 a.m.	95	93	2	783.56	783.56	500	21	48
564	22/07/2013	09:54:54 a.m.	95	93	2	783.468	783.47	500	21	48
565	22/07/2013	09:54:54 a.m.	95	93	2	783.468	783.47	500	21	48
566	22/07/2013	09:54:54 a.m.	95	93	2	761.802	761.8	500	21	48
567	22/07/2013	09:54:54 a.m.	95	66	29	761.772	761.77	500	21	48
568	22/07/2013	09:54:54 a.m.	95	91	4	761.741	761.74	500	21	48
569	22/07/2013	09:54:54 a.m.	95	91	4	750.817	750.82	500	21	48
570	22/07/2013	09:54:54 a.m.	95	91	4	750.817	750.82	500	21	48
571	22/07/2013	09:54:54 a.m.	95	91	4	750.817	750.82	500	21	48
572	22/07/2013	09:54:54 a.m.	95	90	5	750.817	750.82	500	21	48
573	22/07/2013	09:54:54 a.m.	95	90	5	750.817	750.82	500	21	48
574	22/07/2013	09:54:54 a.m.	95	90	5	750.817	750.82	500	21	48
575	22/07/2013	09:54:54 a.m.	95	90	5	750.817	750.82	500	21	48
576	22/07/2013	09:54:54 a.m.	95	90	5	750.817	750.82	500	21	48
577	22/07/2013	09:54:54 a.m.	95	89	6	750.817	750.82	500	21	48
578	22/07/2013	09:54:54 a.m.	95	89	6	750.817	750.82	500	21	48
579	22/07/2013	09:54:54 a.m.	95	89	6	750.817	750.82	500	21	48
580	22/07/2013	09:54:54 a.m.	95	89	6	750.817	750.82	500	21	48
581	22/07/2013	09:54:54 a.m.	95	32	63	750.817	750.82	500	21	48
582	22/07/2013	09:54:54 a.m.	95	88	7	728.723	728.72	500	21	48
583	22/07/2013	09:54:54 a.m.	95	88	7	728.662	728.66	500	21	48

584	22/07/2013	09:54:54 a.m.	95	88	7	728.662	728.66	500	21	48
585	22/07/2013	09:54:54 a.m.	95	88	7	728.601	728.6	500	21	48
586	22/07/2013	09:54:54 a.m.	95	87	8	728.54	728.54	500	21	48
587	22/07/2013	09:54:54 a.m.	95	87	8	728.478	728.48	500	21	48
588	22/07/2013	09:54:54 a.m.	95	87	8	728.417	728.42	500	21	48

En la tabla 11.13 se aprecia que en la iteración 548 fue alcanzado el set-point deseado haciendo que el error de posición sea 0, sin embargo las oscilaciones continúan ocasionando que el error de posición aumente su valor de forma gradual.

Tabla 11.14 Parte 8 datos obtenidos control PID.

No. de Iteracion	Fecha	Hora	Set-Point	Posición Actual	Error de posición	Señal de control	Ciclo de trabajo Motor Superior	Ciclo de trabajo Motor Inferior	RPS Motor Superior	RPA Motor Inferior
1946	22/07/2013	09:55:12 a.m.	95	95	1	576.952	576.95	500	39	42
1947	22/07/2013	09:55:12 a.m.	95	95	0	576.952	576.95	500	39	42
1948	22/07/2013	09:55:12 a.m.	95	95	0	576.952	576.95	500	39	42
1949	22/07/2013	09:55:12 a.m.	95	95	0	576.952	576.95	500	39	42
1950	22/07/2013	09:55:12 a.m.	95	62	33	587.571	587.57	500	39	42
1951	22/07/2013	09:55:12 a.m.	95	95	0	587.571	587.57	500	39	42
1952	22/07/2013	09:55:12 a.m.	95	95	0	587.571	587.57	500	39	42
1953	22/07/2013	09:55:12 a.m.	95	95	0	587.571	587.57	500	39	42
1954	22/07/2013	09:55:12 a.m.	95	95	0	587.571	587.57	500	39	42
1955	22/07/2013	09:55:12 a.m.	95	95	0	587.571	587.57	500	40	40
1956	22/07/2013	09:55:12 a.m.	95	95	0	587.571	587.57	500	40	40
1957	22/07/2013	09:55:12 a.m.	95	95	0	587.571	587.57	500	40	40
1958	22/07/2013	09:55:12 a.m.	95	95	0	587.571	587.57	500	40	40
1959	22/07/2013	09:55:12 a.m.	95	95	0	587.571	587.57	500	40	40
1960	22/07/2013	09:55:12 a.m.	95	51	0	587.571	587.57	500	40	40
1961	22/07/2013	09:55:12 a.m.	95	94	0	587.571	587.57	500	40	40
1962	22/07/2013	09:55:12 a.m.	95	94	0	587.571	587.57	500	40	40
1963	22/07/2013	09:55:12 a.m.	95	94	0	587.571	587.57	500	40	40
1964	22/07/2013	09:55:12 a.m.	95	94	0	587.571	587.57	500	40	40
1965	22/07/2013	09:55:12 a.m.	95	95	0	587.571	587.57	500	40	40
1966	22/07/2013	09:55:12 a.m.	95	95	0	587.571	587.57	500	40	40
1967	22/07/2013	09:55:12 a.m.	95	95	0	587.571	587.57	500	40	40
1968	22/07/2013	09:55:12 a.m.	95	95	0	587.571	587.57	500	40	40

1969	22/07/2013	09:55:12 a.m.	95	95	0	587.571	587.57	500	40	40
1970	22/07/2013	09:55:12 a.m.	95	95	0	587.571	587.57	500	40	40
1971	22/07/2013	09:55:12 a.m.	95	95	0	587.571	587.57	500	40	40
1972	22/07/2013	09:55:12 a.m.	95	95	0	587.571	587.57	500	40	40
1973	22/07/2013	09:55:12 a.m.	95	95	0	587.571	587.57	500	40	40

En la tabla 11.14, se aprecia que el set-point fue alcanzado, pero las oscilaciones que presentó el sistema antes de llegar a la posición determinada provoco que la planta tomará más tiempo en estabilizarse.

Grafica 4. Error de posición.

En la gráfica 11.4, se observa que las oscilaciones que presenta el sistema son bastante pronunciadas.

Al inicio de la gráfica el error de posición es el máximo positivo, ya que pasó de una lectura real de 5 cm a un set-point de 95 cm, haciendo que el sistema experimente el mayor error de posición posible, después las oscilaciones van disminuyendo hasta que se alcanza la altura

deseada haciendo que el error de posición valga 0, sin embargo el control no se detiene y sigue modificando la salida de la planta pasando de un error de posición con valor de 0 a un error de posición con valores negativos.

11.4 Control con RNA + PID.

La tercera configuración de control propuesta se muestra en la imagen 11.4, en la cual el control de la planta es una tarea compartida entre la red neuronal multicapa en paralelo con un controlador PID, donde el entrenamiento de la red neuronal es en línea, a diferencia de las dos configuraciones propuestas previamente.

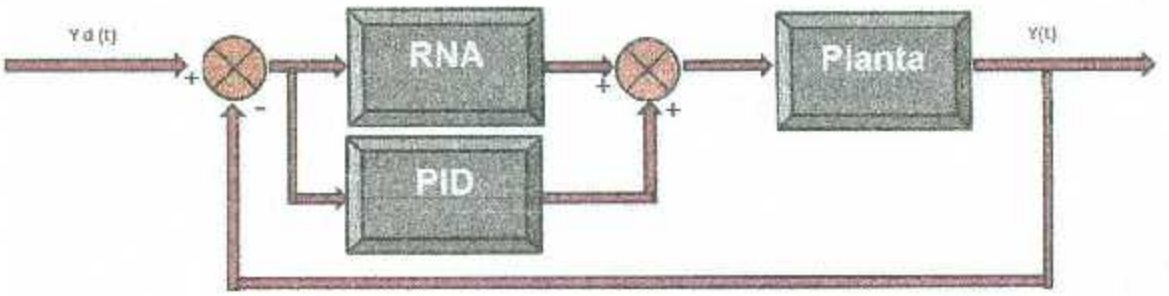


Figura 11.4 Control RNA + PID.

Es replanteada la arquitectura de la red neuronal según la imagen 11.5. Donde la capa de entrada corresponde a la lectura del sensor ultrasónico, al valor de set-point y al error de posición. En la capa de salida se tiene el valor del ciclo de trabajo que se le asignará al motor superior.

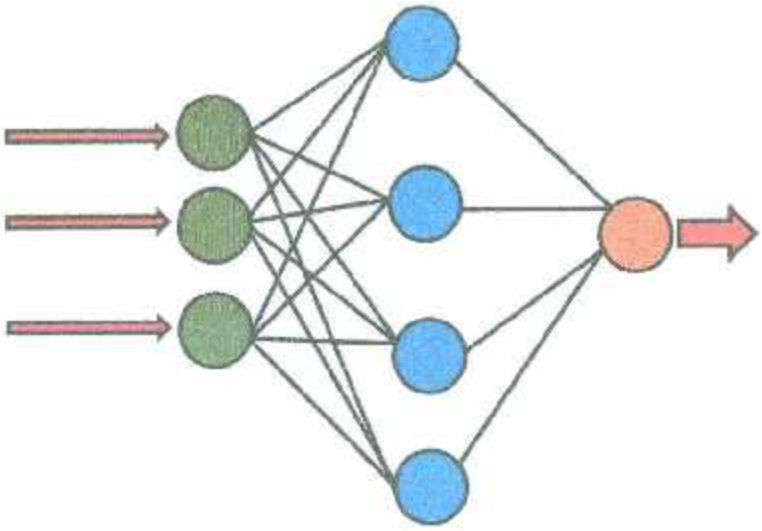


Figura 11.5 Red neuronal.

El entrenamiento se realizará en línea, es decir, la red neuronal cambiará el valor de los pesos sinápticos por cada patrón de comportamiento al que el sistema sea expuesto aprendiendo de un patrón a la vez, ya que cuando se realizaron las pruebas mencionadas en párrafos anteriores con el entrenamiento fuera de línea no resulto exitoso, ya que el tratar de mantener un conjunto de pesos sinápticos estáticos, no le brindaban al sistema la estabilidad necesaria ya que por cada posición del cilindro dentro del tubo de acrílico y debido a la naturaleza de la planta resultaba obsoleto e ineficiente la idea de mantener los mismos pesos sinápticos para todas las posibles condiciones de trabajo del prototipo, por lo que resulta lógico que el entrenamiento se realice en línea.

En la imagen 11.7, se muestran algunos patrones de comportamiento al que se sometió el sistema.

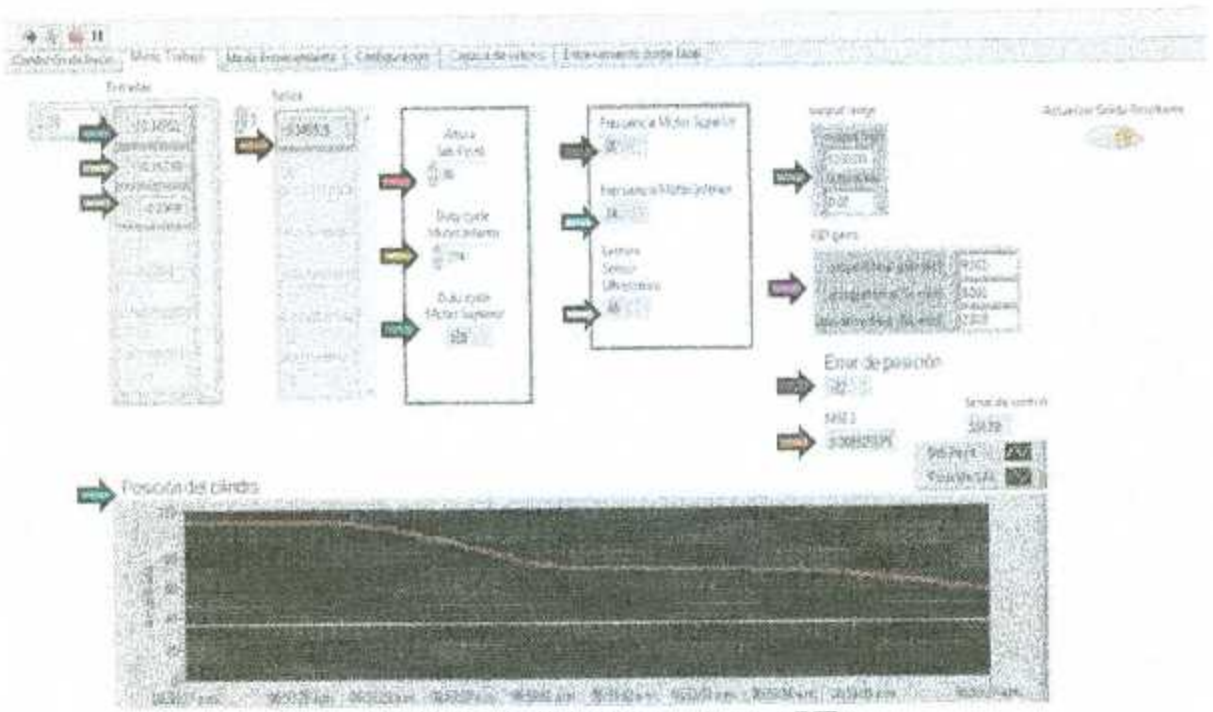


Figura 11.6 Control con RNA + PID

En la tabla 11.15 de muestran la nomenclatura de todos los elementos que intervienen en el control del sistema

Tabla 11.15 Panel de control.

Simbolo	Nombre de variable	Función
➡	Altura de referencia Set-Point	Magnitud de la altura de referencia también llamada Set-point normalizado, el cual es un valor ingresado por el usuario como la altura deseada. El valor es normalizado, es decir es dividido por la altura máxima que puede tener esta variable, el cual es 103 cm. La operación matemática es: set-point/103.
⇨	Altura	Es el valor normalizado que captura el sensor ultrasónico en un instante de

		tiempo determinado. Para ingresar el valor al arreglo es necesario normalizar su valor del sensor dividiéndolo entre la lectura máxima del sensor la cual es de 103 cm. La operación matemática es: valor del sensor ultrasónico/103cm.
⇨	Error de posición	Es la diferencia entre la altura de referencia y el valor real del sensor, en otras palabras es la diferencia entre el valor deseado y el valor obtenido. El error de posición es normalizado, realizando la siguiente operación matemática: error de posición/103.
⇨	Altura de referencia	La salida para realizar el entrenamiento en línea es el mismo valor de set-point normalizado.
⇨	Altura de referencia Set-Point	Es el valor en centímetros ingresado por el usuario como la altura deseada con un rango de entre 15-95 cm.
⇨	Duty cycle de motor inferior	El ciclo de trabajo del motor inferior depende del valor que el usuario determine. El sistema percibe los cambios en la velocidad del motor inferior como una perturbación donde independientemente del valor que tenga esta variable el sistema siempre buscara alcanzar la altura de referencia o set-point establecido.
⇨	Duty cycle de motor superior	Es el ciclo de trabajo necesario para que el cilindro alcance la altura deseada. El

		valor de esta variable lo determina el valor de salida del controlador PID y la salida de la red neuronal artificial.
➡	Frecuencia de motor superior	Es la frecuencia de giro del motor superior medida en Hz.
➡	Frecuencia de motor inferior	Es la frecuencia de giro del motor inferior medida en Hz.
➡	Lectura de sensor ultrasónico	Lectura del sensor ultrasónico en centímetros, el cual indica la posición del cilindro dentro del tubo de acrílico.
➡	Output range	La salida del controlador PID fue limitada a un valor entre 0 y 1100, ya que el valor total de giro del motor superior es de 2047 es cual representa un ciclo de trabajo del 100%, pero en diversas pruebas que fueron realizadas se comprobó que si era utilizado todo el rango de giro del motor de 0 a 2047 los cambios eran tan bruscos que provocaban una gran inestabilidad en el sistema.
➡	PID gains	Las ganancias del controlador PID, se mantuvieron estáticas durante todo el entrenamiento, sus valores fueron determinados en pruebas anteriores.
➡	Error de posición	El error de posición representa la diferencia entra la altura deseada y la obtenida. Su valor va disminuyendo y tendiendo a 0 conforme el cilindro se va acercando al set-point, puede tener valores tanto positivos como negativos,

		dependiendo de la posición del cilindro en un instante determinado con respecto a la altura de referencia.
➡	MSE	Es el error mínimo cuadrático asignado por el usuario, el cual determina cuando el entrenamiento de la red neuronal ha llegado a su fin.
➡	Posición del cilindro	Es una gráfica que indica de forma comparativa como la posición del cilindro (lectura del sensor ultrasónico) va alcanzado el valor de la posición deseada.

En la imagen 11.6, se aprecia como el set-point es de 36cm, el ciclo de trabajo del motor inferior es del 34% , se ingresan los valores al arreglo llamado *entradas* y al arreglo *salidas* de la forma que se explicó en la tabla 16, posteriormente se comienza con el entrenamiento de la red neuronal donde los pesos sinápticos cambiarán hasta que el MSE sea menor al valor establecido por el usuario y la salida de la red neuronal junto con la salida del controlador PID asignan el valor de ciclo de trabajo del motor superior.

En la gráfica de la imagen 11.6, se muestra como la línea que pertenece a la posición del cilindro va siguiendo al valor del set-point con el mínimo de oscilaciones, que si se hiciera solamente con el controlador PID o con la red neuronal sola.

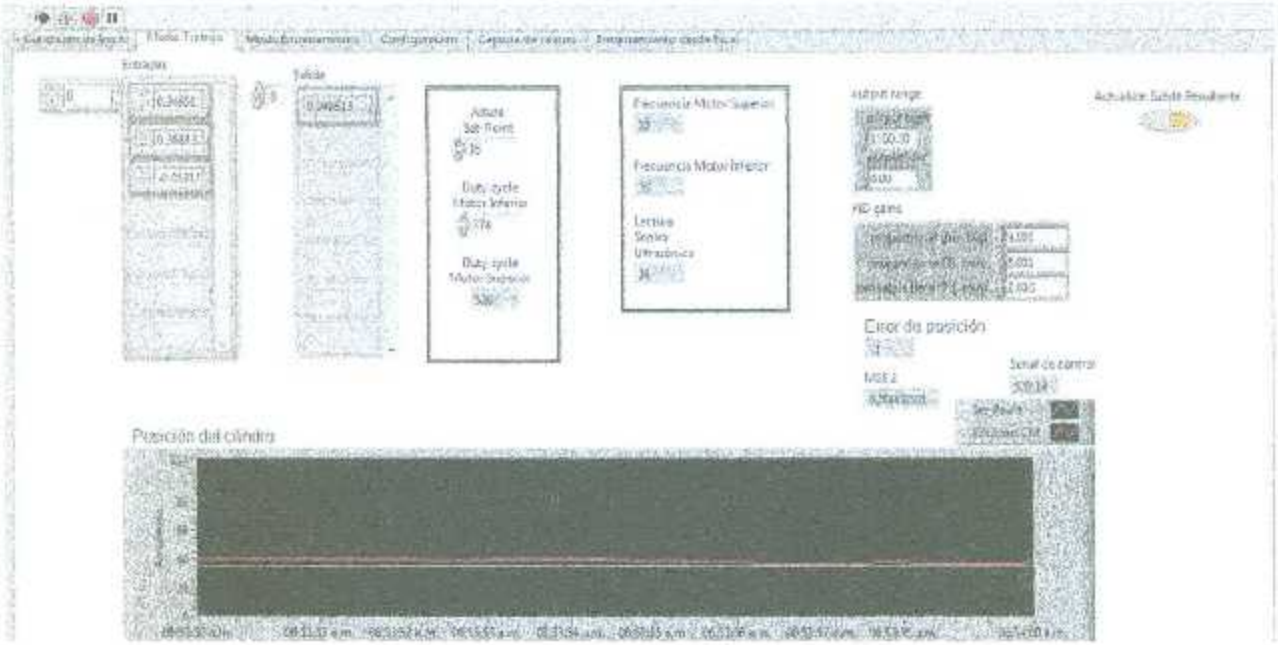


Figura 11.7. Control con RNA + PID.

En la figura 11.7 el error de posición tiende a 0, alcanzando el valor de -1. El ciclo de trabajo del motor superior es de 47.27% alcanzando al final de la gráfica el valor del set-point. Posteriormente se presenta una perturbación en el sistema aumentando la velocidad del motor inferior en un 18.27% y en la figura 11.8 se aprecia cómo se produce una oscilación en el comportamiento del cilindro, pero el sistema sigue en búsqueda de que se alcance el set-point deseado.

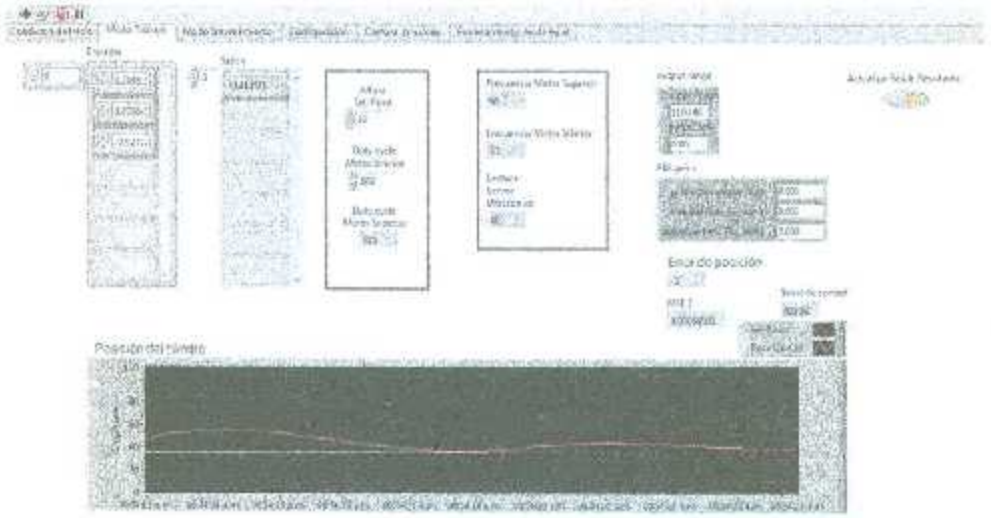


Figura 11.8 Control con RNA + PID

En la figura 11.9, la posición del cilindro ha alcanzado la altura de referencia fijada por el usuario, haciendo que el error de posición sea 0, con un aumento final en el ciclo de trabajo del motor superior del 9.54%, ademas el MSE es inferior al fijado por el usuario de 0.001.

Figura 11.9. Control con RNA + PID.

Enseguida es alterado el valor de referencia a 11.10 cm, manteniendo la velocidad del motor inferior constante, imagen 11.10.

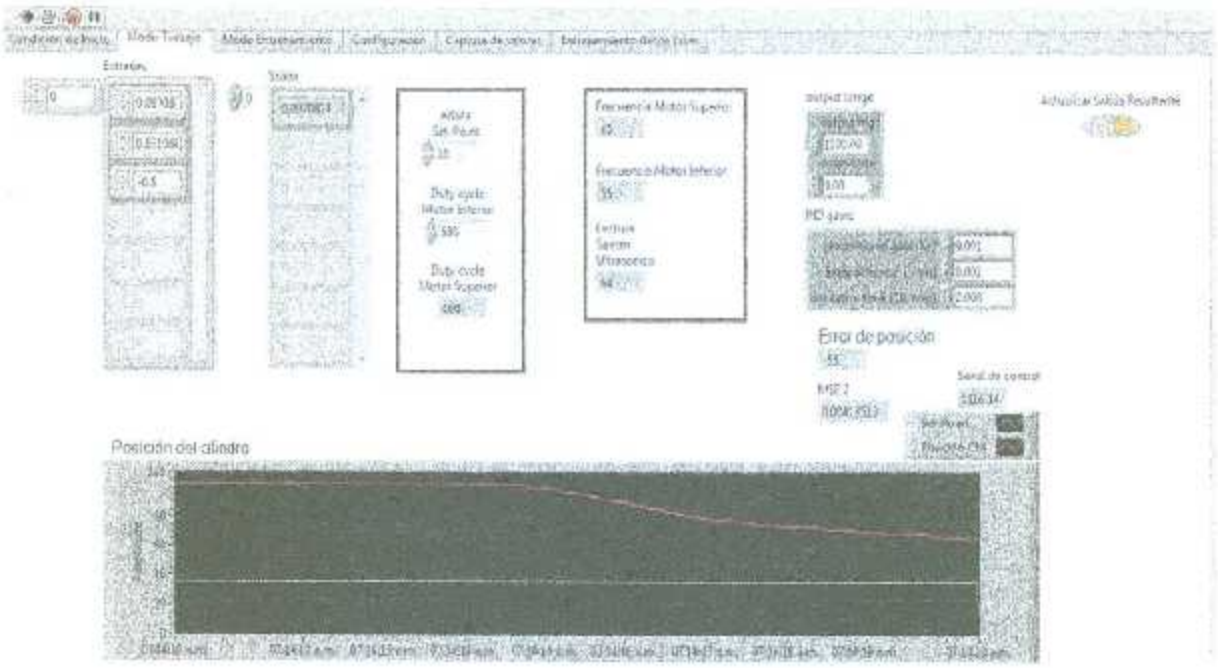


Figura 11.10 Control con RNA + PID

El error de posición en la imagen 11.10 es de -55, el valor del MSE es de 0.004 el cual es mayor al 0.001 fijado como estándar, por lo tanto la red neuronal se ve forzada a entrenarse nuevamente, ya que el patrón fue cambiado al alterar el set-point.

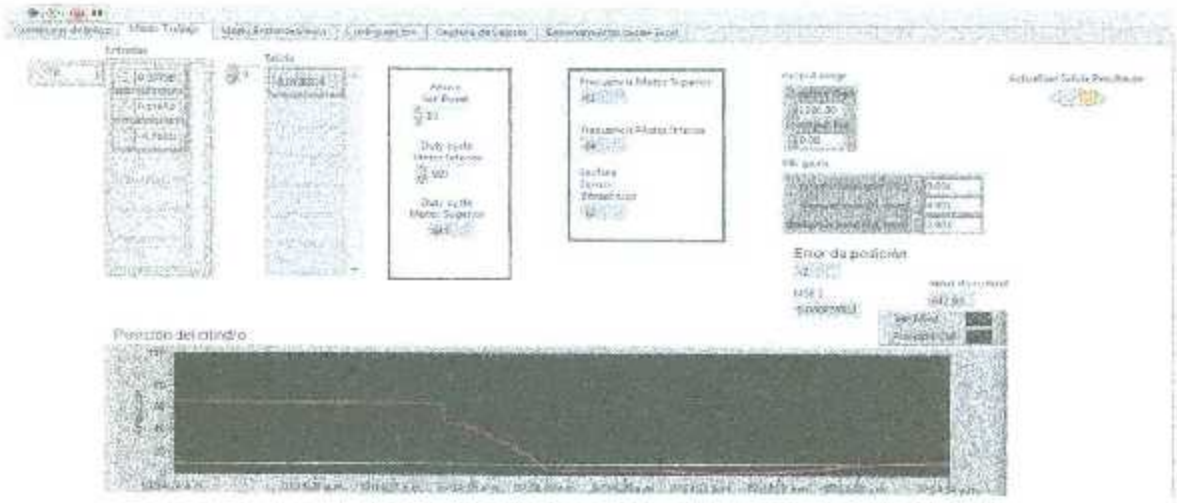


Figura 11.11 Control con RNA + PID

En la figura 11.11, se aprecia como la posición del cilindro cambio de forma rápida con pocas oscilaciones donde el error de posición es de -2. Finalmente en la imagen 11.12 el cilindro ha alcanzando la posición deseada.

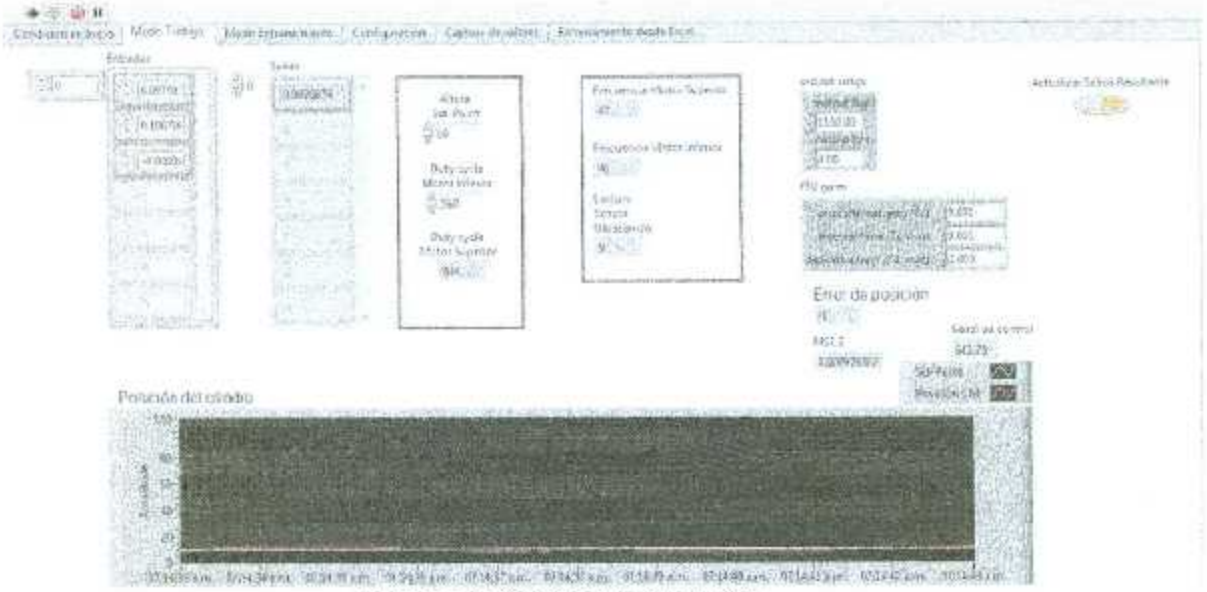


Figura 11.12 Control con RNA + PID

Se produce una perturbación disminuyendo la velocidad de succión del motor inferior en un 13.72%, produciendo una oscilación que es corregida rápidamente por el sistema.

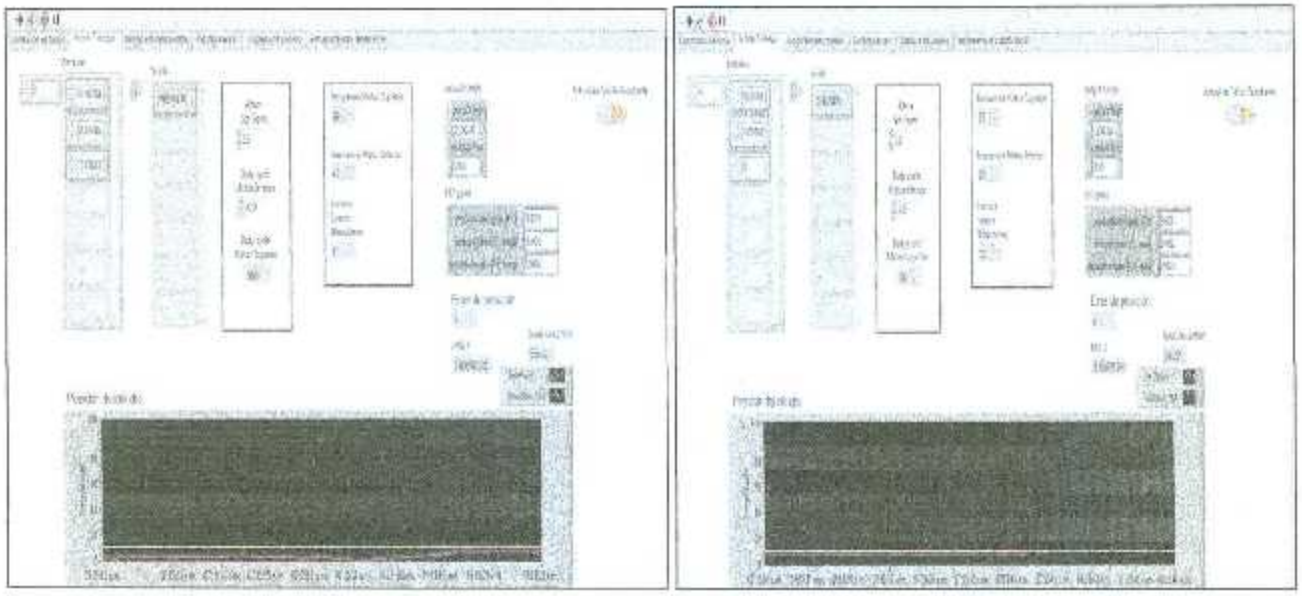


Figura 11.13. Control con RNA + PID

En la prueba realizada con PID + RNA, la función de transferencia tanto de la capa intermedia como de la capa de salida fue sigmoidea, donde el rango de valores de salida es entre 0 y 1.

Se cambia de forma drástica las condiciones de trabajo de la red neuronal como se muestra en la imagen 11.14.

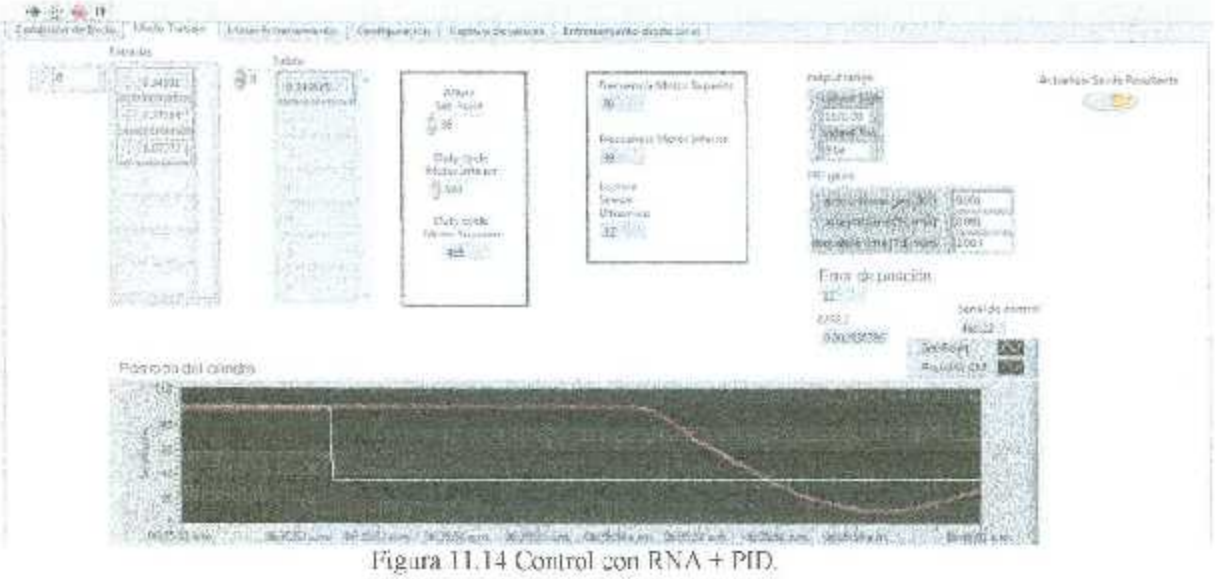


Figura 11.14 Control con RNA + PID.

La red neuronal se vuelve a entrenar, porque es un patrón nuevo, comportándose de forma eficiente ante el cambio en las condiciones de trabajo, imagen 11.15.

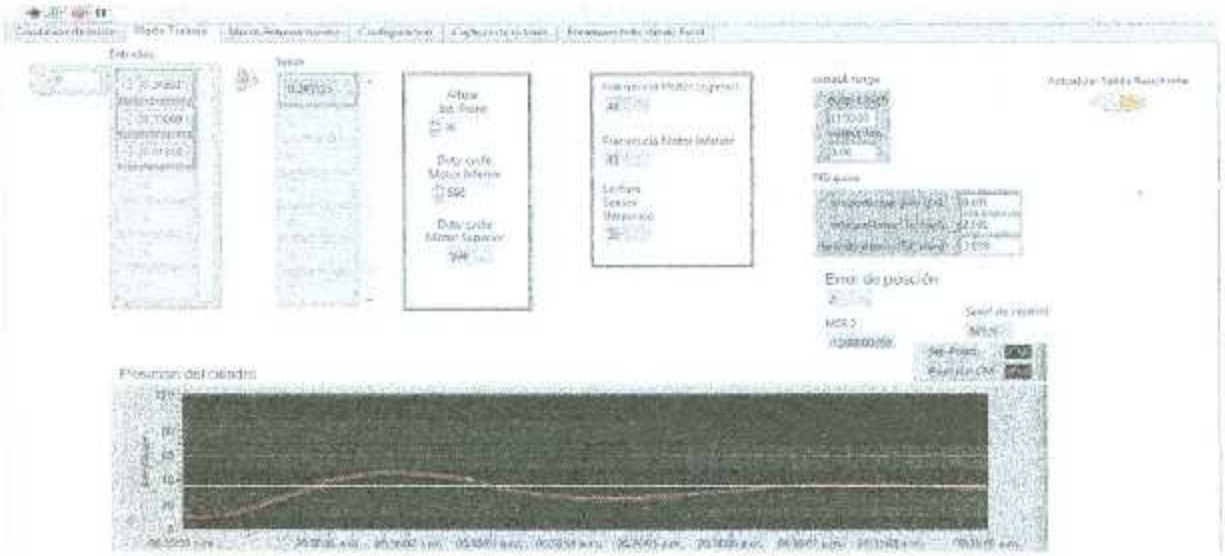


Figura 11.15 Control con RNA – PID.

Las ganancias del control PID se mantienen constantes, alcanzándose el set point como se ve en la figura 11.16.

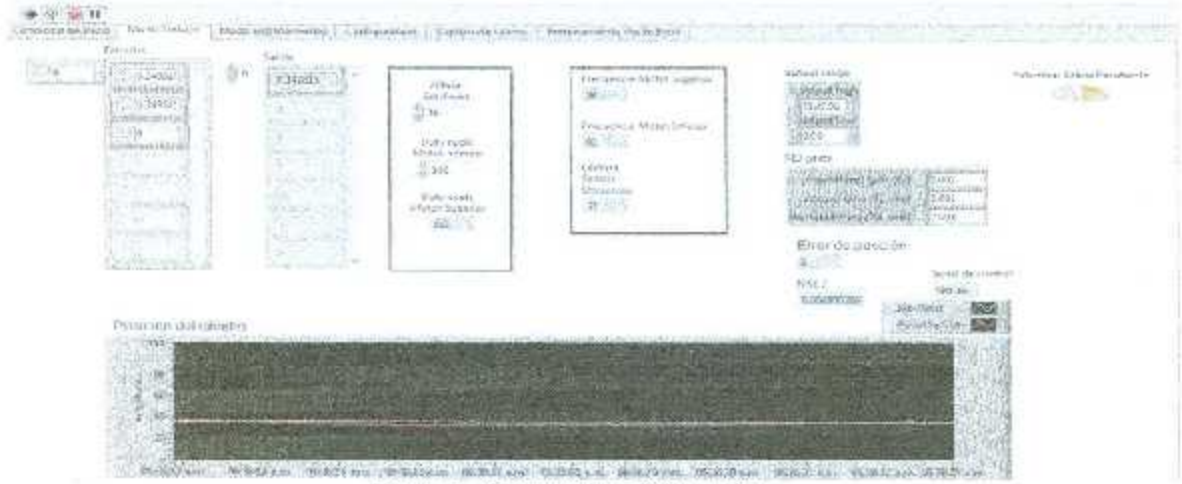


Figura 11.16 Control con RNA + PID

Por lo tanto se concluye que la forma más eficiente de mantener una altura determinada es el trabajo en paralelo de la red neuronal multicapa entrenada en linea con un controlador PID, ya que fueron implementadas diversas configuraciones de controladores, resultando la más exitosa la de RNA – PID.

La tabla 11.16 exhibe un conjunto de diversas lecturas realizadas con numerosas alturas de referencia deseadas, además del tiempo de estabilización del sistema después de que es modificado el valor de algún parámetro de control.

Tabla 11.16 Parámetros finales de control

No. Medición	Posición Deseada CM	Lectura (Sensor ultra-sónico) CM	Lectura Real (vernier) CM	Frecuencia Motor VTI RPS	Frecuencia Motor Enfria-miento (Perturba-ciones) RPS	Tiempo de respuesta (Segundo)	Error CM	Relación RPS VTI/ RPS PERT.
1	10	10	11	52	41	28	1	1.2683
2	13	13	12	52	41	30	-1	1.2683
3	16	16	17	51	42	31	1	1.2080
4	19	19	20	50	40	29	1	1.2500
5	22	22	21	48	39	30	-1	1.2308
6	25	25	24	47	39	30	-1	1.2051
7	28	28	27	46	39	30	-1	1.1795
8	31	31	30	44	40	29	-1	1.1000
9	34	34	33	41	40	30	-1	1.0250
10	35	35	35	52	52	28	0	1.0000
11	38	38	39	50	51	30	1	0.9744
12	41	41	42	50	51	30	1	0.9762
13	44	44	43	51	50	30	-1	1.0233
14	47	47	48	51	52	31	1	0.9792
15	50	50	51	49	52	30	1	0.9423
16	53	53	54	50	51	30	1	0.9815
17	56	56	55	52	51	29	-1	1.0182
18	59	59	60	51	52	28	1	0.9833
19	62	62	63	49	50	30	1	0.9841
20	65	65	66	43	10	30	1	4.3000
21	68	68	67	41	11	30	-1	3.7273
22	71	71	72	38	9	30	1	

23	74	74	75	37	10	31	1	3.7000
24	77	77	76	35	10	30	-1	3.5000
25	80	80	80	33	10	30	0	3.3000
26	83	83	84	30	8	29	1	3.7500
27	86	86	88	28	9	30	2	3.1111
28	89	89	90	25	10	30	1	2.5000
29	70	70	71	23	11	30	1	2.0909
30	73	73	74	22	12	31	1	1.8333
31	76	76	75	61	60	31	-1	1.0167
32	79	79	78	72	61	29	-1	1.1803
33	82	82	83	60	59	30	1	1.0169
34	85	85	84	58	60	30	-1	0.9667
35	66	66	65	68	60	30	1	1.1333
36	69	69	70	63	60	30	1	1.0500
37	56	56	54	67	59	30	-2	1.1356
38	43	43	44	70	59	31	1	1.1864
39	30	30	31	71	61	29	1	1.1639
40	17	17	18	74	60	29	1	1.2333
PROMEDIO						29.825	0.985	

Fueron verificadas las lecturas proporcionadas por el sensor ultrasónico con respecto a un instrumento de medición (vernier), para así obtener un margen de error entre la lectura que proporciona el sensor de distancia con respecto a la medición real, imagen 11.17.

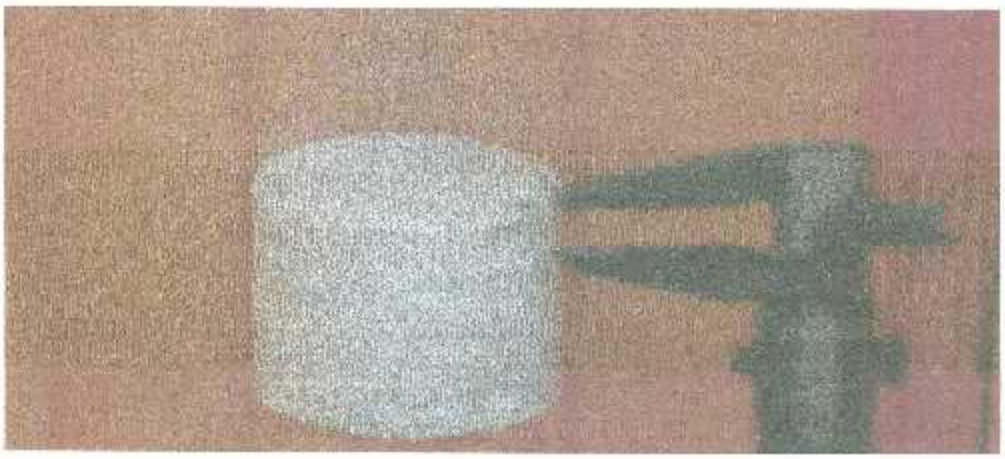


Figura 11.17 Medición efectuada con vernier.

En la tabla 11.16, la primera columna se refiere al número de iteración del sistema, fueron capturadas 40 muestras con distintos valores de set-point, la columna dos y tres de izquierda a derecha de la tabla 11.16 tiene valores iguales ya que el sistema modifica el valor del ciclo de trabajo del motor superior o VTI constantemente hasta alcanzar la altura deseada.

La columna titulada *Medición real*, fue capturada a partir de mediciones con un vernier como se muestra en la imagen 11.17, posteriormente se obtuvo un margen de error, el cual es la diferencia entra la altura deseada y el valor de medición del vernier.

Finalmente se calculó un error promedio el cual vale ±0.985 centímetros, es decir, la posición del cilindro dentro del tubo de acrílico se aproximara a la altura de referencia con un margen de error de ±0.985 centímetros con respecto a la altura deseada.

El tiempo de estabilización del sistema, es decir el tiempo transcurrido entre que se modifica el valor de set-point hasta que el sistema lo alcanza fue de 29.825 segundos, aproximadamente.

Capítulo XII. Conclusiones

Se implementó un sistema básico de control para motores de corriente directa, haciendo uso de diversas herramientas de *hardware* y *software* que tienen la capacidad de ser manipulados partiendo de una regla de control derivada de ejemplos de comportamiento.

La implementación de una red neuronal artificial para realizar el control de posicionamiento del cilindro dentro del tubo de acrílico se decidió principalmente por la dificultad que representaba obtener un modelo matemático no lineal del comportamiento de la planta y con el propósito de crear un mecanismo artificial que reaccione a perturbaciones en el sistema de manera similar a como lo haría el cerebro humano.

Se escogió la implementación de una red neuronal multicapa entrenada con el algoritmo de *Backpropagation* por la capacidad de respuesta que brinda ante aplicaciones que implican relaciones no lineal entre las entradas y salidas de la planta, como ocurre en el prototipo construido en este trabajo de investigación.

El entrenamiento de la RNA fue en línea mediante el algoritmo de aprendizaje *Backpropagation*. Así mismo fueron implementadas tres configuraciones distintas de controladores. Las configuraciones propuestas no fueron escogidas al azar, sino que responden a un proceso en el cual se parte de una configuración relativamente sencilla y conforme se van analizando los resultados obtenidos se incrementa el grado de complejidad de la arquitectura del controlador, hasta alcanzar el objetivo principal del proyecto

El sistema pudo haber sido emulado pero se prefirió la implementación física, debido a que las ecuaciones son abstracciones matemáticas del comportamiento físico y, por lo tanto, cuando se ponga a prueba el controlador seleccionado existe la posibilidad de que la respuesta real de la planta sea diferente a la que se pudo haber obtenido en la simulación de la computadora.

Es una planta real que fue construida para verificar la efectividad de las estrategias de control explicadas, básicamente se ajusta la velocidad del motor superior, y como resultado se obtiene

un flujo de aire dentro de un tubo de acrilico que elevará y mantendrá un cilindro a una altura determinada llamada set-point o altura de referencia.

El prototipo construido es un problema difícil de solucionar, debido a los efectos no lineales que rigen su dinámica, además como el retardo que existe desde que se da la orden al ventilador para que gire a una velocidad determinada hasta el momento cuando ya existe un flujo de aire proporcional a este valor dentro del tubo.

Además de los fenómenos físicos que ocurren dentro del sistema explicados por la ley de Bernoulli y el efecto Coandã, los cuales provocan un flujo de aire turbulento con cambios de presión dentro del tubo y la tendencia del cilindro girar sobre su propio eje.

Fue debido a la gran complejidad de la planta que fue necesario complementar el control de la RNA con un controlador PID, los cuales trabajarían en paralelo, además se observó que el entrenamiento fuera de línea no era eficiente ante los constantes cambios físicos de la propia naturaleza de la planta.

El haber realizado la implementacion de la red neuronal multicapa en una tarjeta FPGA, ayudo a que el entrenamiento fuera más rápido, debido a la forma de operar de estas tarjetas, las cuales nos permiten un mayor aprovechamiento de las propiedades que el control con redes neuronales nos ofrecen.

Gracias a la compatibilidad de comunicación que las tarjetas FPGA nos ofrecen, al permitirnos enlazarlas con diversos lenguajes de programación, específicamente el utilizado en este proyecto de investigación, que nos ayudó a unir el lenguaje gráfico con el lenguaje de descripción de *hardware*, y así obtener las mejores características de ambos lenguajes.

12.1 Trabajos futuros.

En lo que se refiere a trabajos futuros, la interfaz gráfica y el algoritmo de control fueron desarrollados pensando en que el usuario fuera capaz de configurar la arquitectura de la red neuronal de acuerdo a las necesidades de la aplicación que se le fuera a dar.

En el desarrollo de este trabajo de investigación, el algoritmo y la interfaz gráfica fueron solamente probados con el prototipo aquí construido para el control de posicionamiento de un cilindro dentro de un tubo de acrílico, sin embargo, el objetivo principal del proyecto es que el algoritmo y la interfaz de control fueran aplicables a proyectos de distinta naturaleza, de los cuales se espera que tengan una respuesta favorable, lo cual ya se propone como trabajo futuro debido a las limitaciones de tiempo con las que se cuenta.

[1] MUÑOZ, Mario A., LÓPEZ, Jesús A., CAICEDO, Eduardo F. Swarm intelligence: problem-solving societies (a review). *Ingeniería e investigación*. 28 (2): (119-130), Agosto de 2008.

[2] CORDOVA Reyes, Jairo, MARRUFO Cabanillas. Renzo, NOMBERTO Neyra, José, PÉREZ Pérez, Christian. Redes Neuronales y Computación Evolutiva. Universidad César Vallejo, Facultad de Ingeniería. Chimbote, Perú. [Fecha de consulta: 17 febrero 2013]. Disponible en: http://www.slideshare.net/renzomarrufocabanillas/computacion-evolutiva-y-computacion-neuronal

[3] DEL BRIO, Martín B., SANZ, Molina, Alfredo. Redes Neuronales y Sistemas Borrosos, 3ª Edición. México, Alfa-Omega, 2007. 4p, 18p, 19p.

[4] KASABOV, Nikola K., BOOK, Bradford A., Foundations of Neural Networks, Fuzzy Systems, and Knowledge Engineering. The MIT Press Cambridge, Massachusetts, 2da Edición. London, England, 1998. 251p.

[5] FLORES, López, Raquel, FERNÁNDEZ, Fernández, José M. Redes neuronales artificiales. Fundamentos teóricos y aplicaciones prácticas. España, Netbiblo S.L., 2008.

[6] KELLY, Rafael, SANTIBÁNEZ, Víctor. Control de movimiento de robots manipuladores, España, Prentice Hall, Pearson Educación, 2003.

[7] CASTILLO, Vázquez, María L. Control de motores de C.D. con aprendizaje por imitación basado en redes neuronales. Tesis para obtener el grado de Maestra en ciencias de la computación. México, D.F.: Instituto Politécnico Nacional, Centro de Investigación en Computación, febrero 2010.

[8] SERRANO, Antonio J., SORIA, E., Martín J.D. Redes Neuronales Artificiales. Programa 3er Ciclo. Escuela Técnica Superior de Ingeniería.

[9] MAXINEZ, David G., ALCALÁ, Jarra, Jessica. VHDL El arte de programar sistemas digitales. Tercera Reimpresión. México, CECSA, 2005.

[10] ZAMORANO, Vargas, Jose, Alberto. Diseño de un sistema de medición de distancia ultrasónico con labview FPGA. Tesis para obtener el grado de Maestro en ciencias de Ingeniería Eléctrica. Torreón, Coahuila: Instituto Tecnológico de la Laguna, División de estudios de posgrado e investigación, noviembre 2009, 4p.

[11] AGUILAR, Torres, Anna, Leonor. Sistema domótico de red inalámbrica para monitoreo y control del gasto electro-doméstico. Tesis para obtener el grado de Maestra en ciencias de Ingeniería Eléctrica. Torreón, Coahuila: Instituto Tecnológico de la Laguna, División de estudios de posgrado e investigación, marzo 2011, 37p.

[12] CARPIO, Pardo, Fernando, GRAU, Boluda, José A. VHDL Lenguaje para síntesis y modelado de circuitos, 2ª Edición. México, Alfa-omega, 2004.

[13] ZUÑIGA, Guerra, Juan, Felipe. Estudio del control de presión invertida en horno usando técnicas avanzadas de control e instrumentación. Tesis para obtener el grado de Maestro en ciencias de Ingeniería Eléctrica. Torreón, Coahuila: Instituto Tecnológico de la Laguna, División de estudios de posgrado e investigación, enero 2012, 14p, 22p, 25p-27p, 34p.

[14] RAMOS, Ramayo, Myriam. Dimensionamiento de un horno rotativo para la producción de *clinker* de cemento Portland. Tesis para obtener el grado de Ingeniera química. Universidad de Cádiz, Facultad de Ciencias, Febrero 2008, 13-14p.

[15] Spartan - 3E *Starter Kit Board User Guide*, March 9, 2006, UG230 (v1.0).

[16] KILTS, Steve. Advanced FPGA Design Architecture, Implementation, and Optimization. 2ª Edición, Minneapolis, Minnesota, John Wiley & Sond, INC, 2007

[17] WU, Xiang, YE, Jing. EPSRC Islay Project. Work Package 3. Agosto 2009. [Fecha de consulta. 16 Mayo 2013].

Disponible en: http://www.eng.ed.ac.uk/~idcomislay/WP3/index.html

[18] SRF05 - Ultra-Sonic Ranger, Technical Specification. [Fecha de consulta: 27 de Febrero 2013]. Disponible en: http://www.robotstorehk.com/sensors/doc/srf05tech.pdf

Manual del usuario

En el capítulo presente se explicará la forma de operación del prototipo construido en el desarrollo de esta tesis así como el manejo de la interfaz gráfica de usuario diseñada para su control.

Configuración de la red

La configuración de la red neuronal y las condiciones de entrenamiento de la RNA se hacen en la pestaña de Configuración del menú principal.

Figura 1. Menú principal configuración de la red.

1.- Ingresar la cantidad de neuronas en la capa de entrada, en la capa intermedia y en la capa de salida

2.- Inicialización de pesos sinápticos

a) Si se decide inicializar los pesos sinápticos con un valor en especifico, es necesario ingresar el valor con un rango entre 0 y 1 en el control numérico llamado *valor inicial de pesos* y posteriormente activar el botón *Inicio asignación pesos sinápticos.*

Figura 2. Asignación de pesos sinápticos

b) Si se decide que los pesos sinápticos iniciales sean pseudoaletorios, se activa el botón de *Activar Números Aleatorios pesos* y activar el botón *Inicio asignación pesos sinápticos.*

Figura 3. Asignación de pesos sinapticos aleatorios.

3.- Se iluminará un indicador led, indicando que se ha finalizado con la rutina de activación de pesos sinápticos.

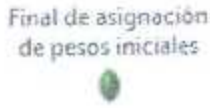


Figura 4. Final de asignación de pesos iniciales

Captura de valores

La captura de valores del prototipo es realizada en la sexta pestaña del menú principal, imagen 5.

Figura 5 Menú principal configuración

1.- Para inicial la rutina de monitoreo de variables del prototipo es necesario activar el botón de *Comenzar a tomar lecturas.*

Figura 6. Inicio de rutina de monitoreo

2.- Si se decide capturar los valores para el entrenamiento de la red neuronal fuera de linea directamente del prototipo se activa el botón *Captura en Array Training Inputs.*

Figura 7.Captura de datos en training input

3.- Para hacer la captura de datos en Excel se activa el botón de *Captura en* Excel

Figura 8. Captura de datos en Excel

4.- Modificación de tiempo de muestreo en milisegundos.

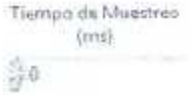


Figura 9. Tiempo de muestreo.

5.- Modificación de velocidad de motor superior e inferior.

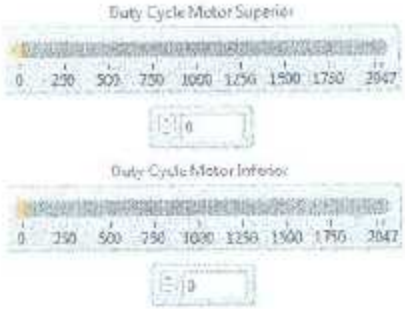


Figura 10. Variación de velocidad de motores.

6.- Inicio/final de graficar la velocidad de ambos motores así como las variaciones en la altura.

Figura 11. Inicio/final gráfica.

Entrenamiento desde Excel

Si el entrenamiento de la RNA es fuera de linea, es necesario haber capturado un conjunto de patrones de entrada y salida del prototipo, figura 12.

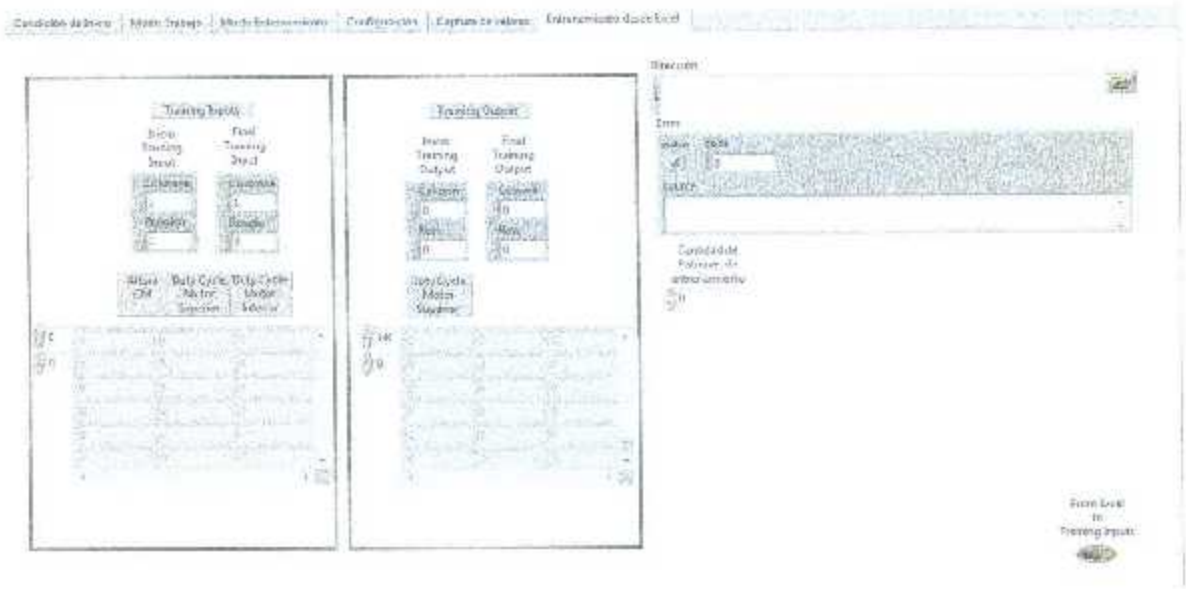


Figura 12. Menú principal entrenamiento desde Excel.

1.-Ingresar la ruta de acceso al documento incluyendo el nombre del archivo con extensión *.xls.

Figura 13. Ruta de acceso al documento de Excel.

2.- Ingresar la cantidad de patrones a utilizar para el entrenamiento de la RNA.

Figura 14. Numero de patrones de entrenamiento.

3.- Training inputs

A) Ingresar el número de columna y el número de renglón de inicio de las entradas.

B) Ingresar el número de columna y el número de renglón de final de las entradas.

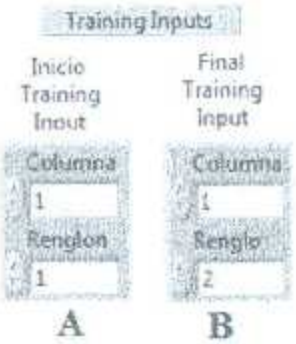


Figura 15. Training inputs

4.- Training outputs

A) Ingresar el número de columna y el número de renglón de inicio de las salidas.

B) Ingresar el número de columna y el número de renglón de final de las salidas.

Figura 16. Training outputs

5.- Ejecutar programa

Activar el botón de inicio de rutina.

Figura 17. From Excel to training inputs

Modo entrenamiento

Una vez que se ha realizado la configuración de la RNA y que se han llenado de datos los arreglos de *Training Inputs* y training outputs, solamente es necesario activar el botón de *Begin training* y esperar que el led *Fin de entrenamiento se ilumine*, figura 18.

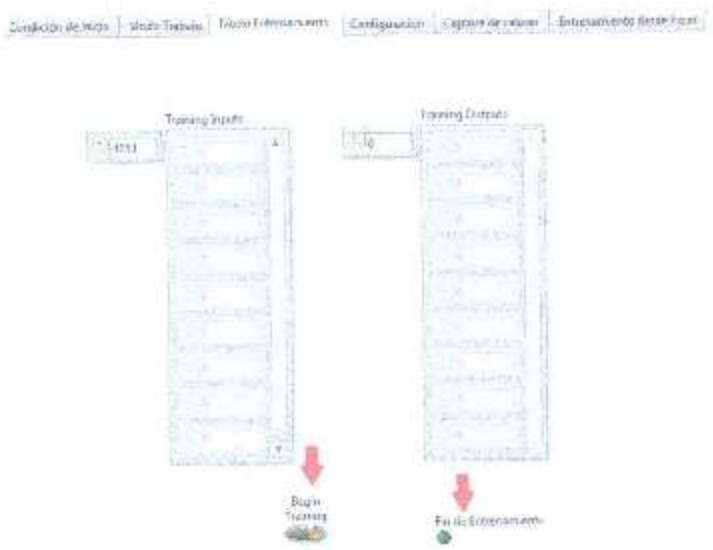


Figura 18. Menú principal modo de entrenamiento

Modo trabajo

Una vez que se ha entrenado la red se continúa con el modo de trabajo, pero si el entrenamiento de la red será en linea es necesario llenar todo el panel frontal de la forma en que fue indicada previamente

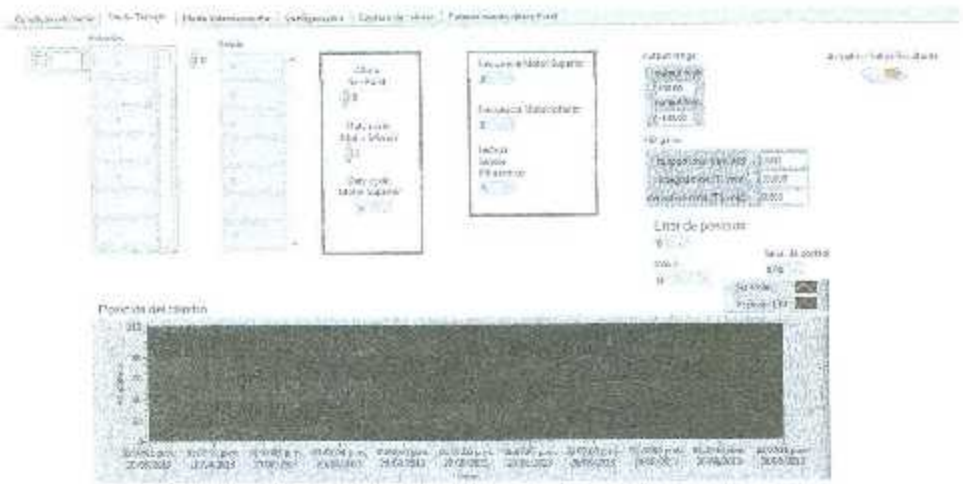


Figura 19. Menú principal modo de trabajo

Es necesario activar el botón *Activar salida resultante.*

Figura 20. Botón de inicio de rutina modo de trabajo