SEP
SECRETARÍA DE EDUCACIÓN PÚBLICA

TECNOLÓGICO NACIONAL DE MEXICO
Instituto Tecnológico de la Laguna

DIVISIÓN DE ESTUDIOS DE POSGRADO E INVESTIGACIÓN

" Implementación del Circuito de Control Digital de un Motor de DC con Encoder Usando VHDL y LABVIEW"

POR

Ing. Samuel Hernández García

TESIS

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

DIRECTOR DE TESIS

Dr. Juan Sifuentes Mijares

ISSN: 0188-9060

RIITEC: (12)-TMCIE-2016

Torreón, Coahuila. México
Diciembre 2016

SEP
SECRETARÍA DE EDUCACIÓN PÚBLICA

TECNOLÓGICO NACIONAL DE MÉXICO
Instituto Tecnológico de La Laguna

Torreón, Coah., 15/Noviembre/2016
Dependencia: DEPI/CPCIE
Oficio: DEPI/CPCIE/204/2016
Asunto: Autorización de impresión de tesis.

C. Samuel Hernadez García
CANDIDATO AL GRADO DE MAESTRO EN CIENCIAS EN INGENIERÍA ELÉCTRICA.
PRESENTE

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

"Implementación del Circuito de Control Digital de un Motor de CD con Encoder usando VHDL y LabView"

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 **RIITEC: (12)-TMCIE-2016**, para que proceda a la impresión del mismo.

ATENTAMENTE
EDUCACIÓN TECNOLÓGICA FUENTE DE INNOVACIÓN

SECRETARÍA DE EDUCACIÓN PÚBLICA
INSTITUTO TECNOLÓGICO
Instituto Tecnológico de La Laguna
División de Estudios de Posgrado e Investigación

M.I.I. ARMANDO LONGORIA DE LA TORRE
Jefe de la División de Estudios de Posgrado e Investigación
del Instituto Tecnológico de la Laguna

ALT/RHSR

Torreón, Coah., 14/Noviembre/2016

M.I.I. ARMANDO LONGORIA DE LA TORRE
JEFE DE LA DIVISIÓN DE ESTUDIOS DE POSGRADO E INVESTIGACIÓN

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

"Implementación del Circuito de Control Digital de un Motor de CD con Encoder usando VHDL y LabView"

Desarrollado por el **C. Samuel Hernadez García**, con número de control **M1313071** 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
EDUCACIÓN TECNOLÓGICA FUENTE DE INNOVACIÓN

Dr. Juan Sifuentes Mijares
Asesora/Directora de Tesis

Dr. Mario F. Cepeda Rubio
Comité Tutorial

M.C. Sergio F. Salas Huerta
Comité Tutorial

Dr. José A. Pamanes García
Comité Tutorial

INDICE

INDICE DE FIGURAS

Capítulo 1. Introducción

Uno de los periodos más asombros en la historia de la humanidad se presentan en el siglo XXI debido a los grandes cambios que logró la revolución industrial. Tales cambios son del orden demográfico, económico, ambiental a tal grado que la población mundial se ha triplicado desde el siglo pasado. Sin embargo, tales cambios han causado adversidades en diferentes aspectos de nuestra vida, entre ellos la incidencia de enfermedades como el cáncer [Santisteban, 2006].

No obstante, durante el transcurso de los años han existido diferentes métodos para el tratamiento de cáncer, estos tratamientos han sido modificados a lo largo de los años dando diferentes efectos secundarios que dejan marcado el cuerpo humano. Los métodos actuales para el tratamiento de cáncer tienen efectos secundarios severos; es por esta razón que nuevos métodos alternativos para el tratamiento de cáncer son importantes. Siendo uno de estos la termoterapia o terapia por ablación termal [Lavista et al, 2008].

La terapia por hipotermia se basa en el hecho de que las células cancerígenas son más sensibles al incremento de temperatura que las células normales. Esto es llevar el tumor a temperaturas entre 41 y 42 grados centigrados llegando al límite de recuperación de las células normales y llevando el incremento de temperatura en los rangos de 50 a 70 grados centígrados que provoca la destrucción de las células cancerígenas con daño irreversible, que resulta en la eliminación del tumor o la reducción o detenimiento de su crecimiento. Este tipo de métodos incluyen la ablación por radiofrecuencia, termoterapia por ultrasonido enfocado, terapia de laser inducido, ablación térmica magnética [El-Sherbin et al, 2011]. Si bien estos métodos han sido usados contemporáneamente; se requiere de un especialista para poder posicionar y maniobrar la terapia por ablación. Es por esto la importancia de los robots manipuladores y lo que han logrado ya que estos pueden posicionarse con una gran precisión y exactitud que se ha venido perfeccionando a lo largo de los años.

La creación o la existencia de robots manipuladores se debe a que se decidió construir máquinas que realicen tareas repetitivas, extenuantes o difíciles de

realizar. La tarea de un robot que se ocupa esta investigación es posicionar una antena de ablación por microondas. De la manera óptima, se pretende a través de un robot manipulador direccionar una antena para ablación; es decir, el brazo robótico se encargará de direccionar tanto la posición como la orientación de la antena con la mayor precisión y exactitud.

La Asociación de Industrias Robóticas o denominada RIA por su acrónimo en inglés define a un robot industrial como: "un manipulador multifuncional reprogramable, capaz de mover materias, piezas, herramientas o dispositivos especiales, según trayectorias variables, programadas para realizar tareas diversas" [Barrientos et al., 1997]. A su vez, La Federación Internacional de Robótica o IFR por su acrónimo, define al robot industrial como: "Un manipulador controlado automáticamente, reprogramables en tres o más ejes los cuales pueden ser fijos o móviles de acuerdo a su aplicación en la industria".

Un robot manipulador está compuesto por eslabones rígidos articulados entre sí. Los eslabones son los elementos mecánicos que forman la estructura mecánica que al interconectarse forman una cadena cinemática. Las cadenas cinemáticas de un robot manipulador pueden ser abiertas o cerradas dependiendo del arreglo de los eslabones. Un robot manipulador puede tener una anatomía del brazo humano y debido a esto se hace referencia a los distintos elementos que componen un robot usando términos como *cuerpo, brazo, codo y muñeca* [Barrientos et al., 1997]. Por último, el modelado de un robot manipulador consiste en obtener una representación matemática ideal del robot.

El modelado matemático de un robot resulta más complejo a medida que aumenta el número de eslabones del robot. El modelado se divide en lo que es la cinemática y la dinámica.

En la presente investigación se aplica solamente el modelo del robot considerando. La cinemática es la ciencia que estudia el movimiento de los cuerpos sin importar las fuerzas que lo causan. En la investigación del modelado cinemático se estudia la posición y todas las derivadas de alto orden sobre la posición, es por

eso que la cinemática de manipuladores se refiere a todas las propiedades geométricas y de tiempo basadas en las propiedades del movimiento [Craig, 2006].

1.1 Justificación

La conveniencia de realizar esta investigación parte en la importancia del objetivo principal de la tesis. El cual es diseñar un robot capaz de orientar y posicionar una antena. Se vuelve importante debido a que la antena para ablación requiere de óptima precisión que un robot manipulador puede proporcionar. Debido a esto, el campo de investigación tanto electrónico como médico tendrá más información para futuras investigaciones. La idea es que futuros trabajos tomen parte de esta investigación y sean una fuente confiable de análisis para proyectos futuros.

Es importante tener en cuenta un factor, el cual es el aumento y el uso de robots en los últimos años. Es decir, las aplicaciones con robots han ido creciendo de manera rápida debido a las habilidades que tienen estos como realizar tareas repetitivas y peligrosas. Su precisión es un punto clave y elemental para realizar una tarea como la que se pretende en esta investigación la cual es posicionar y orientar una antena.

Para lograr sustentar el manejo de robots manipuladores en esta investigación se consultaron diversas fuentes de información referentes al control de robots manipuladores que existen hoy en día. Sin embargo, el acercamiento que se tiene en la presente investigación se aborda de manera un tanto diferente al modo convencional el cual es empleando microntroladores y PLCs. El control del robot manipulador será de manera diferente tomando en cuenta los dispositivos lógicos programables llamados FPGA los cuales aunado a esto se tiene la interacción con Labview y sus instrumentos virtuales siendo esto una aplicación tanto novedosa desde el punto de vista digital.

1.2 Consideraciones Metodológicas

1.2.1 Objetivo general

El objetivo general original de esta investigación, era la implementación de un brazo robótico con la habilidad para posicionar una antena en el espacio tridimensional, así como dar la orientación adecuada a la antena.

Debido a la falta de tiempo solo se obtuvo parte de este objetivo siendo descrita la parte obtenida en los objetivos específicos logrados.

1.2.2 Objetivos específicos logrados

Se tienen a continuación los siguientes objetivos logrados:

- Diseñar e implantar un circuito lector de encoder para motores en VHDL.
- Diseñar e implantar un circuito sumador restador en VHDL
- Diseñar e implantar un circuito multiplicado en VHDL.
- Diseñar e implantar un circuito generador de PWM en VHDL.
- Diseñar e implantar un circuito de control proporcional en VHDL.
- Diseñar e implantar un circuito derivativo en VHDL.
- Diseñar e implantar un circuito de control proporcional derivativo en VHDL.
- Realizar pruebas y ajustes de ganancias a los controladores.
- Enlazar los controladores diseñados con LabView
- Presentar los resultados y conclusiones.

1.2.3 Organización de la tesis

En la primera parte de la investigación, el capítulo 2 aborda el tema de la teoría e implementación de circuitos VHDL sobre el FPGA. Los resultados de la investigación aplicada fueron basados en el diseño de circuitos digitales empleando el lenguaje VHDL. El capítulo 3 aborda la explicación acerca de los módulos de circuitos digitales para el control de un motor de DC el cual será parte futura de un brazo robótico. Aquí se da a conocer que paquete de software se usa y que componentes se requieren para la implementación del control de un brazo manipulador a través de VHDL. El capítulo 4 muestra los resultados obtenidos en una FPGA y usando instrumentos virtuales en Labview. Los resultados son

mostrados en graficas donde se observa la posición actual del eje del motor para diferentes ganancias. Por último, capitulo 5 se dan las conclusiones a los resultados obtenidos sobre la implementación del control.

Capítulo 2. Implementación de circuitos en VHDL sobre FPGA.

2.1 Morfología del robot

Un robot manipulador es un dispositivo móvil o estacionario dotado de uno o varios brazos mecánicos, controlados por computadora y que realizan tareas de servicio. La estructura mecánica de un robot manipulador serial está formada por una serie de eslabones unidos mediante articulaciones los cuales pueden lograr un movimiento relativo entre cada dos eslabones unidos.

El número de articulaciones determina los grados de libertad del robot manipulador.

Los manipuladores son diseñados para una variedad de aplicaciones y dependiendo de su aplicación se tiene una configuración determinada. Tales configuraciones son cartesiana, cilíndrica, polar, SCARA, brazo articulado. Cada configuración presenta una determinada combinación de articulaciones de revolución y/o prismáticas. Para efectos de esta investigación, se tiene en mente un robot de brazo articulado donde su configuración son articulaciones de revolución, conectados en forma serial. Esta configuración es la que más se asemeja al brazo humano y se denomina antropomórfica. En esta configuración, los errores de posicionamiento aumentan cuando el brazo se extiende. La posición de los actuadores requiere de una mayor compensación de cargas debido a que están directamente ubicados sobre las articulaciones.

2.2 Aspectos del VHDL

Las siglas VHDL denominan VHDL-> VHSIC + HDL, donde la sigla V es VHSIC o "Very High Speed Integrated Circuit" y la H denomina hardware, D denomina descripción y L es lenguaje. VHDL es un lenguaje de programación directamente orientado a la descripción de hardware. Cabe destacar que posee características de software clásico tales como C o JAVA. El concepto de tipo de datos es una de las características que VHDL comparte con los lenguajes de programación clásico. Además de esto, el usuario puede definir los propios tipos de datos según las necesidades del programador.

Una de las cualidades más importantes del VHDL es el manejo del control de flujo a través de las sentencias tales como IF y las sentencias iterativas como FOR o WHILE. Esto hace más sencilla la descripción de los procesos que se desean implementar. Otra de las ventajas de la programación en VHDL es la creación de bibliotecas de diseño que contienen funciones, operadores y tipos de datos que es con el fin de que el usuario pueda hacer y declarar un conjunto para una tarea específica.

2.2.1 Estructura básica de diseño

El lenguaje VHDL posee tres características que se enfocan en describir el hardware. La figura 2.1 muestra la estructura para describir el hardware en VHDL.

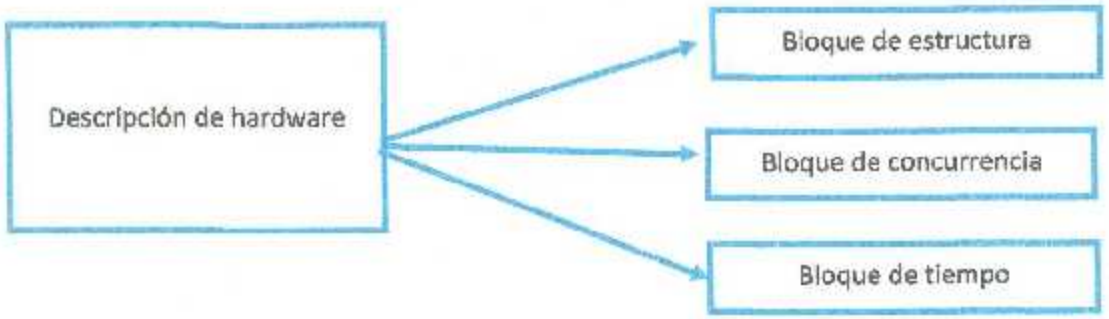


Figura 2.1 Estructura de descripción de hardware

La primera característica es el bloque de estructura donde se hace referencia al diseño digital, y se tiene en cuenta sus partes y la forma en que estas se interconectan, además estas partes se modelan de tal manera y forma estructural a

través de sus componentes o de forma funcional usando lo que se llama como algoritmos.

Para describir cualquier elemento en VHDL se deben definir las entradas y las salidas que son los puertos del dispositivo o entidad. Una característica importante que maneja el VHDL es la de usar un elemento ya definido como parte de uno más grande, a través del concepto de componente. Es decir, cuando se crea un componente este componente es usado en otro elemento más grande dentro de la estructura.

Para llevar a cabo esto se debe definir el elemento que se va a usar y conectar su interfaz de terminales a los puntos adecuados para el diseño que se está realizando. Esto quiere decir que se asemeja como si fuera un montaje de electrónica en el que se interconectan elementos básicos para formar un sistema más complicado, y por último cabe destacar que el agregar un componente no importa la complejidad no tendría problemas dentro del sistema.

El bloque de concurrencia es la segunda característica y se presenta con el fin de modelar el paralelismo en un sistema. El VHDL posee un elemento denominado proceso. Un proceso puede definirse como un programa y se compone de sentencias y define datos para su uso exclusivo. Se puede usar para subprogramas, es decir, describe un comportamiento a través de un código secuencial, con la salvedad que todos los procesos del elemento VHDL se ejecutarán en forma paralela. Un proceso se ubica en la arquitectura del elemento que se está programando.

Para que los procesos que conforman un sistema de VHDL puedan actuar de forma sincronizada entre ellos, se emplea un elemento denominado señal. Cada proceso tiene un grupo de señales a las que es sensible, estas son determinadas por la persona que realiza el proyecto, ser sensible quiere decir que el proceso se ejecutara si se presenta algún cambio en las señales, este comportamiento se ejecuta desde el principio de manera cíclica.

Por último, se tiene el bloque de tiempo y se usa debido a que el modelado en VHDL se orienta también hacia la simulación, se requiere definir el concepto de tiempo para llevar a cabo este procedimiento. Una simulación se une a los eventos, es decir, posee una lista de cambios en las entradas y salidas del proyecto en VHDL, y de sus señales internas a lo largo del tiempo de simulación. La función del simulador consiste en calcular las consecuencias de cada cambio en la lista de eventos actual, además de agregarlos a la lista de eventos futuros, esto se realiza llevando a cabo los procesos del proyecto VHDL que se está simulando.

Este proceso termina al completar el tiempo de simulación establecido por el usuario o cuando los eventos son nulos. Cabe destacar que las estructuras de simulación son denominadas bancos de trabajo, los cuales poseen algunas diferencias en la sintaxis con los proyectos normales de VHDL, además de contener sentencias de tiempo exclusivas. [Pérez, Cristobal; 2007].

2.2.2 Sintaxis básica de VHDL

La sintaxis básica dentro del VHDL se divide en la entidad y la arquitectura. La entidad define lo que es la visión externa del dispositivo donde su función es determinar la interfaz del dispositivo que se va a implementar con su entorno, dado que la entidad es usada como declaración del componente en otro diseño. La sintaxis básica de la entidad se describe en la figura 2.2.

```
ENTITY {nombre del dispositivo} IS
PORT ( {Lista de puertos de entrada}: IN {tipo dato};
        {Lista de puertos bidireccionales}: INOUT {tipo dato}; (
        Lista de puertos de salida}: OUT {tipo dato};
        {Lista de puertos de salida}: BUFFER {tipo dato}; );
        END {nombre del dispositivo};
```

Figura 2.2 Sintaxis básica de entidad en VHDL

Un FPGA es un dispositivo que un diseñador de sistemas digitales puede programar, después que está soldado en el circuito impreso, para que funcione de un modo determinado. Los FPGAs son fabricados con conexiones y lógica programables. El diseñador desarrolla su sistema digital usando herramientas tipo EDA (Electronics Design Automation), sean dibujos esquemáticos o lenguaje de descripción de hardware (como VHDL), para poder plasmar el sistema en lógica digital. Luego de simular satisfactoriamente el sistema digital comprobando su funcionalidad se usan herramientas específicas del vendedor del FPGA para crear un archivo de configuración del FPGA, el cual describe todas las conexiones, interconexiones y lógica que necesita ser programada dentro del FPGA para poder implementar el sistema digital desarrollado. Entonces, a través de un cable USB se conecta el FPGA o el circuito impreso en cual está soldado el FPGA, a una PC y usando el software de configuración del FPGA se descarga el archivo de configuración. Una vez comprobado el correcto funcionamiento del sistema en el FPGA se graba el archivo de configuración en una memoria no volátil que el FPGA leerá y usará para auto-configurarse cada vez que se aplica la tensión de alimentación al FPGA o cada vez que se desee re-configurar el FPGA.

El uso de arquitecturas específicas de procesado implementadas sobre FPGAs presenta como principales ventajas costo-rendimiento y un ciclo de desarrollo muy corto a comparación de otras formas de desarrollo. Los bloques básicos combinacionales de ciertas FPGAs están formados por pequeñas memorias que actúan como tablas de búsqueda capaces de implementar cualquier función de un determinado número de entradas. Las herramientas de diseño suministradas por el fabricante permiten interferir el uso de memoria a partir de descripciones VHDL genéricas en la etapa de síntesis y seleccionar el tipo de memoria a utilizar en la etapa de implementación. [Fajardo, 2011]. En la última década la implementación sistemas de control sobre FPGA se ha convertido en la solución alternativa, ' dado que este tipo de circuitos proporcionan bajo consumo de potencia y mayor estabilidad en el desempeño [Jung, 2007].

2.3.1 Arquitectura básica del FPGA

La arquitectura de un FPGA varía de un fabricante a otro. Resulta ser una discusión permanente, en la que cada fabricante se atribuye que tiene la mejor arquitectura. Se ejecutan cientos de estudios comparativos o denominados benchmarks para mostrar que tan viable y buena es la arquitectura ofrecida por cada fabricante.

A pesar de tratar de diferenciarse uno del otro, en realidad los FPGAs de los diferentes fabricantes tienen muchos componentes en común. Algunos ejemplos son bloques lógicos programables, bloques de memoria de doble puerto (dual port), bloques para ejecución de MACs (Multiplicador-Acumulador o bloques DSPs), bloques de control de reloj (generación de frecuencias a partir de una frecuencia base, corrimiento de fase), bloques de entrada/salida, etc.

La empresa Xilinx ofrece al mercado dos tipos de FPGAs: una de bajo costo, desempeño medio y otra de alto costo con desempeño alto-muy alto. La primera se denomina la familia de los FPGA Spartan, y la segunda los FPGA Virtex. A su vez, y debido a la gran competitividad del mercado, estas familias se van renovando cada dos o tres años. Tal es así que, por ejemplo, la familia Spartan fué evolucionando como Spartan, Spartan 2, Spartan 3 y la recientemente lanzada Spartan 6. Del mismo modo para la Virtex, comenzando con Virtex, Virtex-E, Virtex 2, Virtex 2Pro, Virtex 4, Virtex 5 y la reciente Virtex 6.

La figura 2.4 muestra cómo están organizados los distintos componentes en un FPGA Spartan 3 en donde se puede observar el anillo de bloques de E/S tales como Input/Output Blocks, IOBs los cuales rodea el arreglo matricial de los bloques lógicos configurables (Configurable Logic Blocks, CLBs).

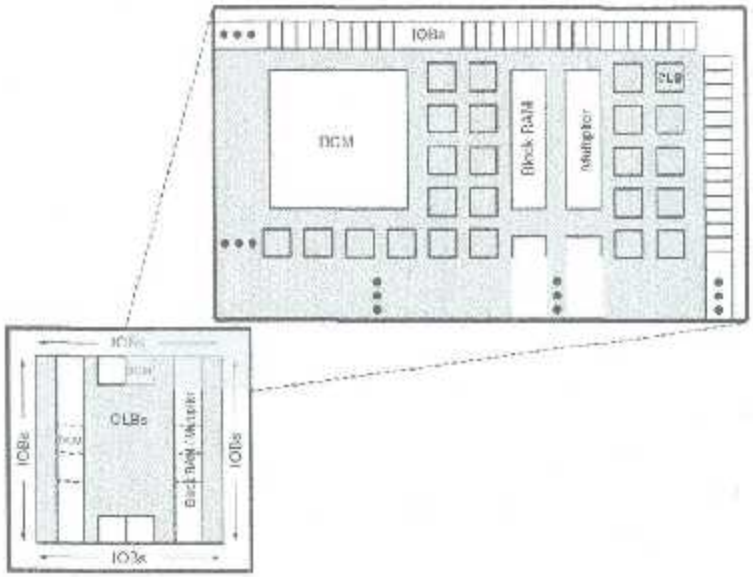


Figura 2.4 Arquitectura Spartan 3 FPGA

Los bloques de memoria RAM en realidad constan de varios bloques de memoria RAM de 18K bits. Cada bloque de memoria RAM está asociado a un multiplicador dedicado. Los controladores de reloj (Digital Clock Manager, DCM) están distribuidos de manera que hay dos en la parte superior, dos en la inferior y uno en cada uno de los costados del FPGA. En la Figura 7 no se muestran la gran cantidad de recursos destinados a la interconexión de los diferentes componentes del FPGA. A continuación, se verán en detalle los diferentes componentes mencionados, que son parte de los FPGAs. [Sisterna, 2000].

2.4 Modulación PWM

La modulación de ancho de pulso que se abrevia como PWM, es un método de transmisión de la información sobre una serie de pulsos. Los datos se están que se están transmitiendo se codifican en el ancho de estos pulsos para controlar la cantidad de energía que es enviada a una carga. Es decir, la modulación de ancho de pulso es una técnica de la modulación para generar pulsos variables para representar la amplitud de una señal análoga o de una onda de la entrada. También el uso popular de la modulación PWM, es para modular la potencia de los motores, otra aplicación interesante de PWM es en donde el ancho de pulso es usado para controlar la cantidad de energía que es enviada a una carga.

La modulación de ancho de pulso se utiliza para reducir la energía total entregada a una carga fuera dando por resultado la pérdida, que ocurre normalmente cuando una fuente de energía es limitada por un elemento resistivo. El principio que se provee en el proceso entero es que la energía media entregada es directamente proporcional al ciclo de deber de la modulación. Si el cociente de modulación es alto, es posible lograr el tren de pulso usando los filtros electrónicos pasivos y recuperar una forma análoga media de la onda.

En la modulación por ancho de pulsos PWM, setransforma una señal análoga en una señal digital conancho de pulso proporcional al valor de la señal análoga previamente muestreada. El muestreo de la señal debeser por lo menos el doble de su ancho de banda, paragarantizar la correcta modulación de la señal.

Entre las técnicas tradicionales para modular en PWM seencuentra el control todo/nada, el cual requiere de un contador módulo N, de un comparador con una señal de referencia y un registro. La figura 2.5 muestra la estructura dentro de un control PWM.

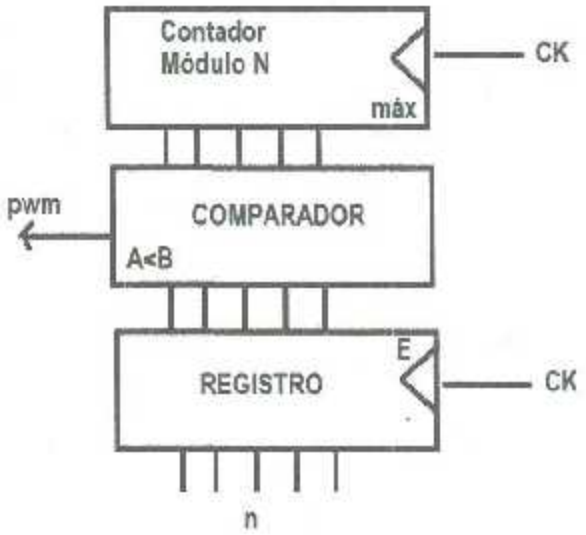


Figura 2.5 Estructura PWM

2.4.1 Máquinas de estado finito

Una máquina de estado se define como un circuito secuencial síncrono que emplea la lógica combinacional y flipflops y donde el valor de los estados corresponde al valor de las salidas de los flipflops. Las máquinas de estado finito denominadas FSM son separadas en Mealy y Moore. En una máquina de estados Moore, la salida o salidas solamente dependen del valor del estado en que se encuentra. En cambio, una máquina Mealy la salida se ve cambiada también por el valor de la señal de entrada.

El diagrama de la máquina de estados contiene los estados, las salidas, las transiciones y las condiciones de transición. Para pasar de un estado a otro es necesario que exista una transición entre los dos en el sentido adecuado y que se cumpla con la condición de la transición.

Una máquina de estados diseñada correctamente debe cumplir con las siguientes características

- La suma de las condiciones de salida de cada estado debe ser igual a uno.
- Ninguna combinación de las señales de entradas puede repetir en dos o más condiciones de salida de un estado.

En el diagrama de máquina de estados se identifica su tipo si es Moore cuando las salidas están dentro de los estados y se distingue de tipo Mealy si las salidas están en las transiciones.

Para el diseño e implementación se muestra en la subsección 2.5.1 donde se tiene el circuito de encoder en VHDL en donde nos servirá la máquina de estados y usar este tipo de máquinas de estado para codificar dentro de VHDL.

2.4.2 Encoders

Los codificadores rotatorios que se conocen usualmente como encoders son mecanismos utilizados para entregar la posición, velocidad y aceleración del rotor de un motor. Sus principales aplicaciones incluyen aplicaciones en robótica, lentes fotográficas, aplicaciones industriales que requieren medición angular, militares, etc.

Un codificador rotatorio es un dispositivo electromecánico que convierte la posición angular de un eje, directamente a un código digital. Los tipos más comunes de encoders se clasifican en: absolutos y relativos (conocidos también como incrementales). Los encoders absolutos pueden venir codificados en binario o gray. Dentro de los encoders incrementales, se encuentran los encoders en cuadratura, ampliamente utilizados en motores de alta velocidad y en aplicaciones en las que interesa conocer la dirección del movimiento del eje. El tipo común de encoder incremental consiste de un disco solidario al eje del motor que contiene un patrón de marcas o ranuras que son codificados por un interruptor óptico (par led/fotodiodo o led/ fototransistor) generando pulsos eléctricos cada vez que el patrón del disco interrumpe y luego permite el paso de luz hacia el interruptor óptico a medida que el disco gira.

La resolución de un encoder típico es del orden de 1000 pulsos por revolución. Desde un encoder incremental no se puede determinar la posición angular absoluta del eje. Para poder determinar la posición relativa a un punto de referencia (cero), el encoder debe incluir una señal adicional que genera un pulso por revolución, denominada índice.

El encoder de cuadratura es un tipo de encoder incremental que utiliza dos sensores ópticos posicionados con un desplazamiento de un cuarto de ranura el uno del otro, generando dos señales de pulsos digitales desfasada en noventa grados o en cuadratura. A estas señales de salida, se les llama comúnmente A y B, mediante ellas es posible suministrar los datos de posición, velocidad y dirección de rotación del eje. Si se incluye la señal de referencia, se le denomina índice. Usualmente, si la señal A adelanta a la señal B (la señal A toma valor lógico "1" antes que la señal B, por ejemplo), se establece el convenio de que el eje está

rotando en sentido horario, mientras que, si B adelanta a A, el sentido será podría estar acoplado a través de un sistema de transmisión con una proporción conocida de reducción o elevación. antihorario.

El número de pulsos del encoder se determina por los pulsos generado en una revolución. Así, midiendo la frecuencia de dichas señales y conociendo la manera de cómo se encuentra acoplado, es posible determinar la velocidad de giro del eje. [Venegas, 2009].

2.5 Circuitos creados.

Para la obtención del control mediante VHDL, se tiene que tener una serie de códigos que en conjunto generan el control PWM propuesto como un objetivo especifico. Se divide en secciones para poder detallar y explicar su funcionamiento y a su vez plasmar sin complejidad el código VHDL y su circuito creado a base de hardware dentro de una FPGA.

2.5.1. Decodificador de encoder.

El decodificador de encoder se basa en diagramas de estado que ayudaran en la construcción de un decodificador de dos puertos de un encoder de cuadratura. Se toma la señal denominada 'a' y 'b' y a través de diagramas de estado se enfoca en saber el valor actual y el valor anterior para poder hacer la suma o conteo de pulsos del encoder a su disposición en tiempo real del motor de directa.

ESQUEMÁTICO:

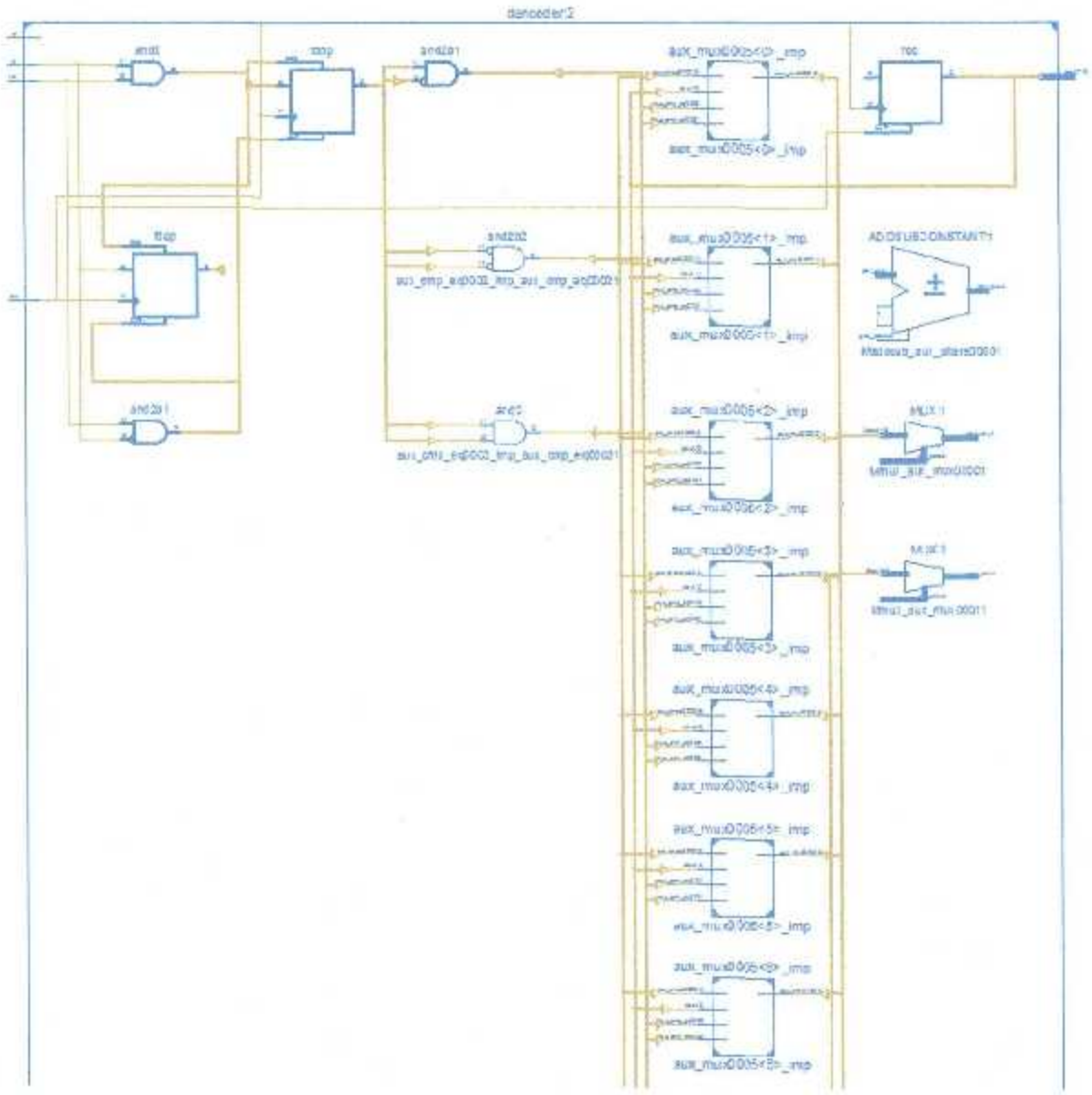


Figura 2.6 Esquemático del decodificador de encoder

CÓDIGO DEL PROGRAMA:

Código decodificador de encoder:

```
entity dencoder is
    Port ( clk : in  STD_LOGIC;
rst : in  STD_LOGIC;
a : in  STD_LOGIC;
b : in  STD_LOGIC;
dout : out  STD_LOGIC_VECTOR (7 downto 0));
end dencoder;

architecture Behavioral of dencoder is
signal aux: STD_LOGIC_VECTOR (7 downto 0);
signal ab_pas,ab_pre : STD_LOGIC_VECTOR (1 downto 0);
begin
process(rst,clk,a,b)
begin
      if rst='1' then
      aux <= "00000000";
      ab_pas<= b&a;
      ab_pre<= b&a;
      elsifclk'event and clk='1' then
      ab_pre<= b&a;
      ab_pas<= ab_pre;
           case ab_pre is
       when "00" =>
          if ab_pas = "01" then
                aux <= aux+1;
                elsifab_pas = "10" then
                aux <= aux-1;
                end if;
       when "01" =>
          if ab_pas = "11" then
                aux <= aux+1;
                elsifab_pas = "00" then
                aux <= aux-1;
                end if;
       when "10" =>
          if ab_pas = "00" then
                aux <= aux+1;
                elsifab_pas = "11" then
                aux <= aux-1;
                end if;
       when "11" =>
          if ab_pas = "10" then
                aux <= aux+1;
                elsifab_pas = "01" then
                aux <= aux-1;
                end if;
       when others =>
          aux <= "00000000";
```

Declaración de puertos donde se tiene el reloj, el botón reset, la entrada de encoder a y b y su salida dout de resolución de 8 bits

Esta parte de código indica cómo se hace la decodificación del encoder donde es un diagrama de estados en donde verifica el estado actual y el anterior para poder decodificar el encoder y hacer el conteo de pulsos.

```
            end case;
        end if;
end process;
dout<= aux;
end Behavioral;
----------------------------------------------------
```

2.5.2. Circuito restador.

El código restador considera como referencia y la posición actual de los pulsos de encoder y los resta en una variable final denominada 'par'.

CÓDIGO DEL PROGRAMA:

Código de resta:

```
----------------------------------------------------
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 instantiating
---- any Xilinx primitives in this code.
--library UNISIM;
--use UNISIM.VComponents.all;

entity resta is
    Port ( ref : in  STD_LOGIC_VECTOR (7 downto 0);
pos : in  STD_LOGIC_VECTOR (7 downto 0);
par : out  STD_LOGIC_VECTOR (7 downto 0));
end resta;
```

Declaración de puertos donde se tiene la referencia de 8 bits de resolución al igual que la posición y la salida par de 8 bits

```
 ----------------------------------------------------
architecture Behavioral of resta is

begin
process(ref,pos)
begin
par <= ref - pos;
end process;
end Behavioral;
----------------------------------------------------
```

Esta parte de código indica cómo se hace la resta entre la referencia dada y la posición que da como salida la variable par.

ESQUEMÁTICO:

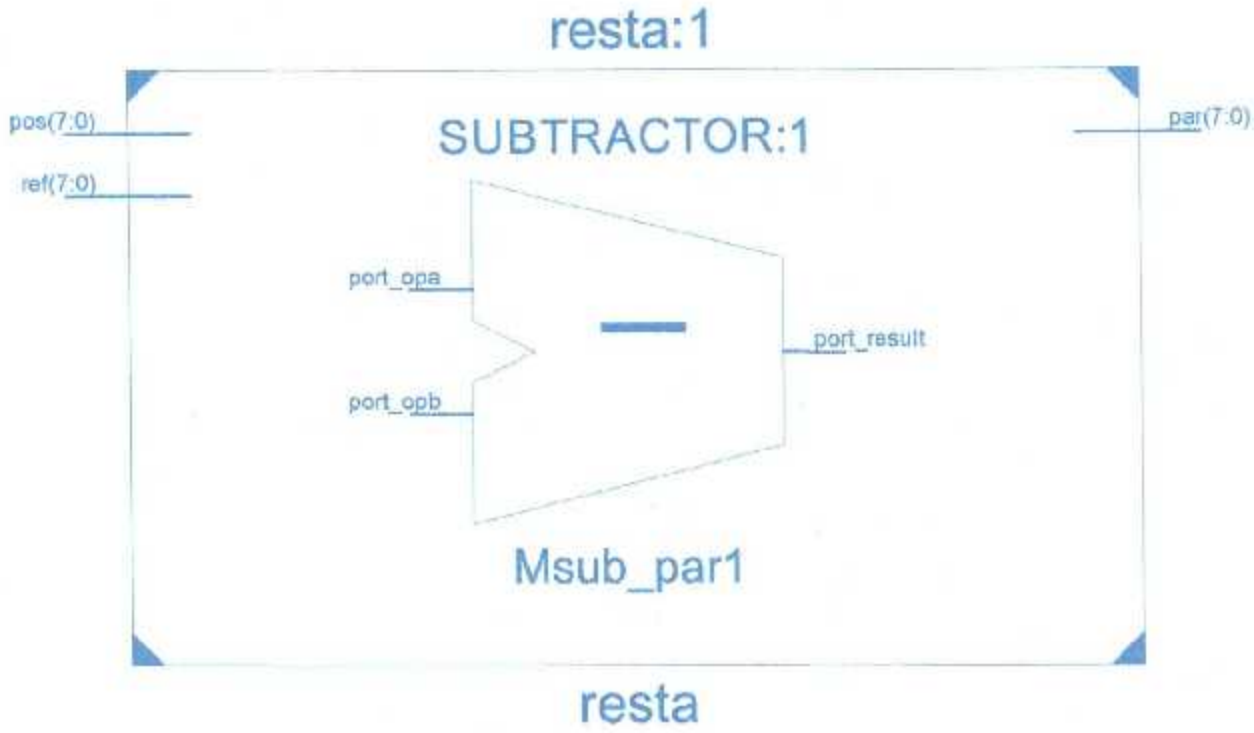


Figura. 2.7 Esquemático de resta

2.5.3. Circuito multiplicador

El circuito multiplicador tiene entrada dinla cual es multiplicada por una ganancia proporcional que es la que nos da el controlador proporcional. Esta ganancia denominada 'kp' da como resultado una multiplicación y una salida que se hace en el código multiplicador mostrado a continuación.

ESQUEMÁTICO:

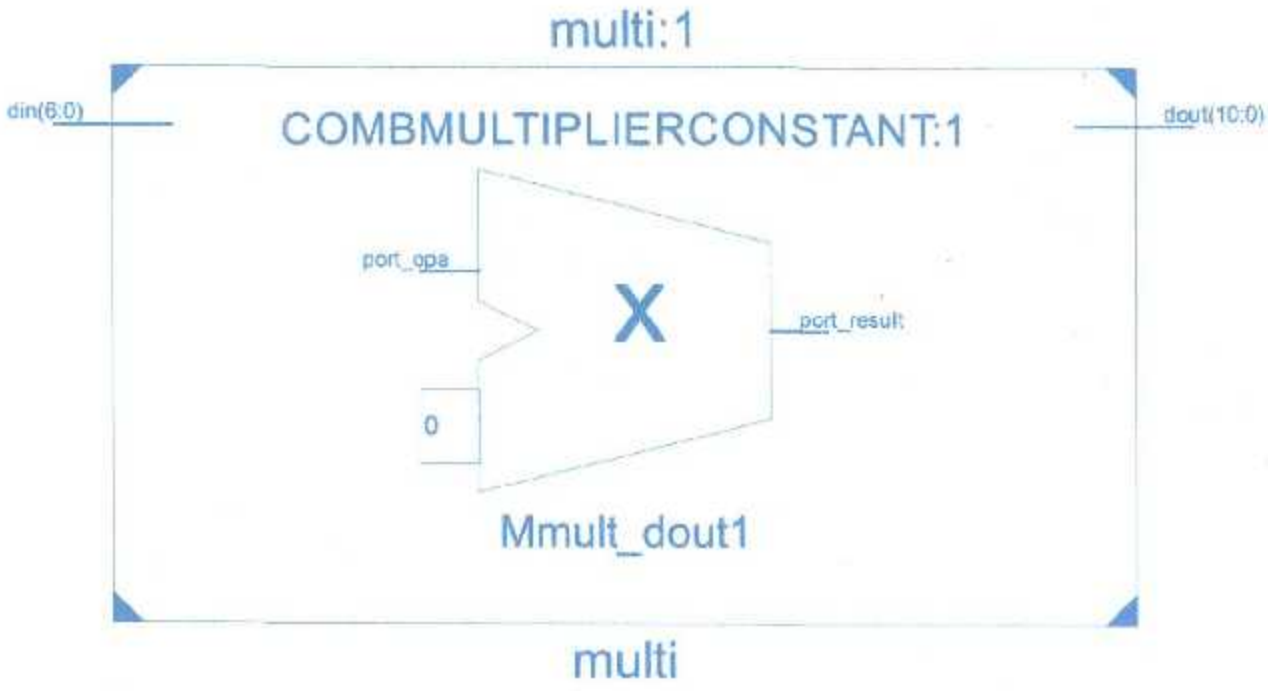


Figura2.8 Esquemático de multiplicador

CÓDIGO DEL PROGRAMA:

Código de multiplicador:

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


entity multi is
    Port ( din : in  STD_LOGIC_VECTOR (6 downto 0);
dout : out  STD_LOGIC_VECTOR (10 downto 0));
end multi;
```

Declaración de puertos donde se tiene la entrada 'din' y la salida dout de 11 bits de resolución

```
architecture Behavioral of multi is
constant kp: STD_LOGIC_VECTOR (3 downto 0):="101C";
begin
process(din)
begin
dout<= kp * din ;
endprocess;
endBehavioral;
```

Esta parte de código indica cómo se hace la multiplicación de la entrada din de 7 bits de resolución por una ganancia proporcional kp y se da a una salida dout

2.5.4 Circuito obtención de absoluto.

ESQUEMÁTICO:

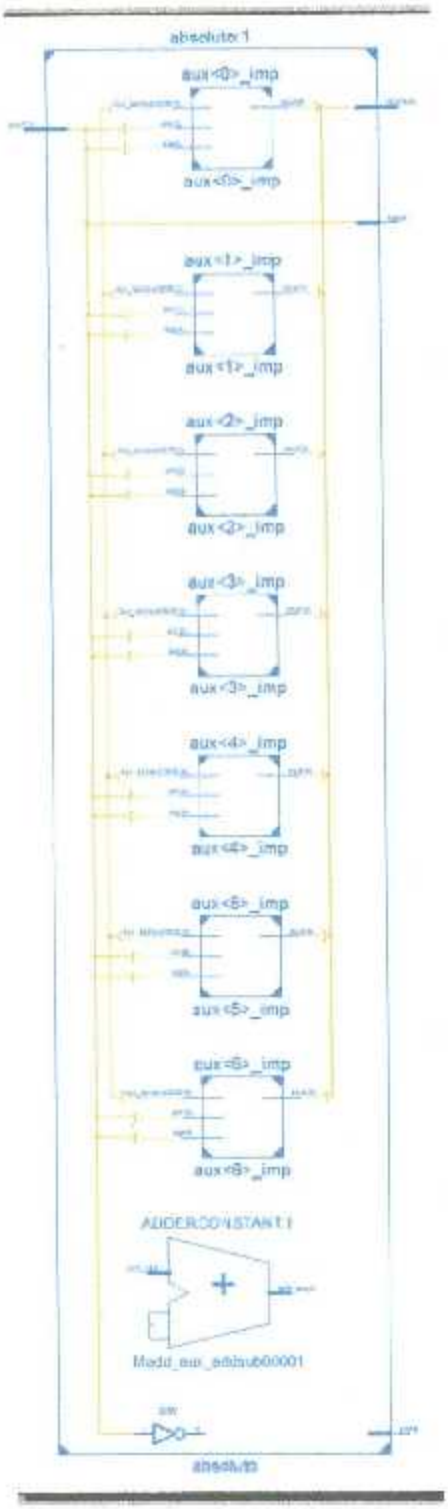


Figura 2.9 Esquemático absoluto

CÓDIGO DEL PROGRAMA:

Código de absoluto:

```
----------------------------------------------------

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

entity absoluto is
    Port ( signA : out  STD_LOGIC;
                signB : out  STD_LOGIC;
din :  in  STD_LOGIC_VECTOR (7 downto 0);
dout : out  STD_LOGIC_VECTOR (6 downto 0)
----------------------);-------------------------------
end absoluto;

architecture Behavioral of absoluto is
signal aux: STD_LOGIC_VECTOR (7 downto 0);
begin
process(din)
begin
     if din(7)='1' then
          aux <= not(din) +1;
     else
          aux <= din;
     end if;
end process;
signA<= din(7);
signB<= not din(7);
dout<= aux(6 downto 0);
end Behavioral;
```

Declaración de puertos se tiene la entrada SignAsignB y din. La salida dout tiene una resolución de 7 bits

Código de programa donde se hace el proceso de valor de absoluto.

2.5.5. Generador de PWM.

El generador de PWM se encarga de controlar el motor. El generador da la salida del controlador en modo de PWM en donde se tiene una señal de salida de un bit. El PWM es generado empleando un contador y mediante sentencias IF para su comparación y así construir la señal de salida.

ESQUEMÁTICO:

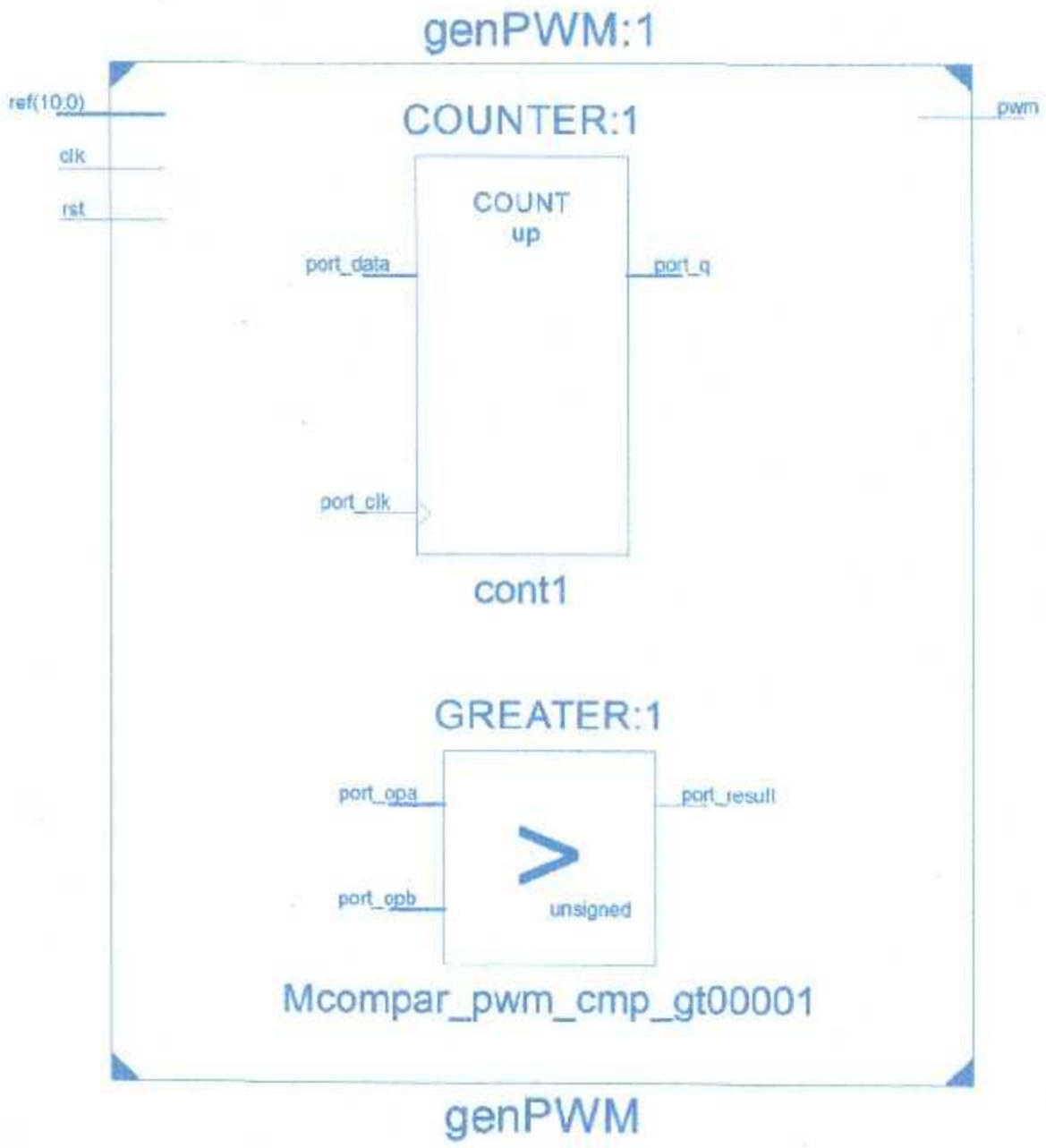


Figura 2.10 Esquemático de PWM

CÓDIGO DEL PROGRAMA:

Código de PWM:

```
----------------------------------------------------
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 instantiating
---- any Xilinx primitives in this code.
--library UNISIM;
--use UNISIM.VComponents.all;

entity genPWM is
    Port ( clk : in  STD_LOGIC;
rst : in  STD_LOGIC;
ref : in  STD_LOGIC_VECTOR (10 downto 0);

pwm : out  STD_LOGIC);
end genPWM;
----------------------------------------------------
architecture Behavioral of genPWM is
signal cont: STD_LOGIC_VECTOR (7 downto 0);
signal ref_aux :STD_LOGIC_VECTOR (7 downto 0);
begin
ref_aux<= ref(10 downto 3);
process(clk,rst,cont)
begin
     if rst = '1' then
          cont<= "00000000";
     elsif clk'event and clk='1' then
          cont<= cont +1;
     end if;
end process;

process(ref_aux,cont)
begin
     if ref_aux>cont then
     pwm<= '1';
     else
     pwm<= '0';
     end if;
end process;
--dout<= cont;
endBehavioral;
```

Declaración de puertos se tiene el reloj, reset, la referencia de 11 bits y la salida pwm de 1 bit

Código de programa donde se tiene la generación de PWM. Donde a través de una sentencia if y un contador interno se tiene la salida de un bit de PWM.

Capítulo 3. Implementación de circuitos de control en VHDL.

Diferentes mecanismos de aplicación industrial y en el hogar son movidos mediante motores de corriente directa. Una buena parte de sistemas robóticos también utilizan motores para generar sus movimientos. Los sistemas de eslabones conectados en forma serial tipo brazo manipulador no son la excepción, es por esto que el manejo y en especial el control de motores de corriente directa es importante. Así también lo es la creación de circuitos digitales que nos permitan controlar los movimientos de los motores de una mejor forma y que sean de creación propia que es una parte muy importante de la aplicación de la tecnología y la teoría de control, de la asimilación y creación del conocimiento de un país.

Para el control de motores de corriente directa se emplea la técnica de PWM. PWM o "Pulse Width Modulation" modulación en ancho de pulso es la base para el control en electrónica de potencia. Teóricamente las técnicas de PWM se usan en diferentes aplicaciones tales como drivers variables de velocidad, cambiadores de frecuencia, fuente de poder sin interrupción entre otros [JianSun, 2012]. En la presente investigación la técnica de PWM se emplea para controlar los motores de corriente que actuaran como las articulaciones del brazo robótico. El proyecto futuro de la implementación de un brazo manipulador se soporta en el control de los motores de corriente directa los cuales darán movimiento y los eslabones del brazo manipulador, siendo estos eslabones las articulaciones del robot mencionado.

3.1 Implementación del circuito digital en VHDL

El circuito digital fue implementado en lenguaje de descripción de hardware VHDL el cual fue sintetizado mediante el software Xillinx ISE. Este código de VHDL se codifico en el software *Xilinx ISE 14.6*. El software Xilinx ISE 14.6 tiene un ambiente conformado por una serie de programas para editar, simular e implementar diseños digitales a un dispositivo FPGA. Todas las herramientas usan una interfaz gráfica que ayuda y permite a los programadores ejecutar los programas con facilidad. El mismo software Xilinx nos permite simular a través de codificación para VHDL,

Verilog y diseños de lenguaje mixtos. El software Xilinx ISE permite a través de un programa llamado *Project Navigator* para codificar en una interfaz gráfica y simular. [Xilinx, 2011].

La figura 3.1 muestra la ventana del programa *Project Navigator* en donde se trabajará para programar el control del PWM en un lenguaje VHDL.

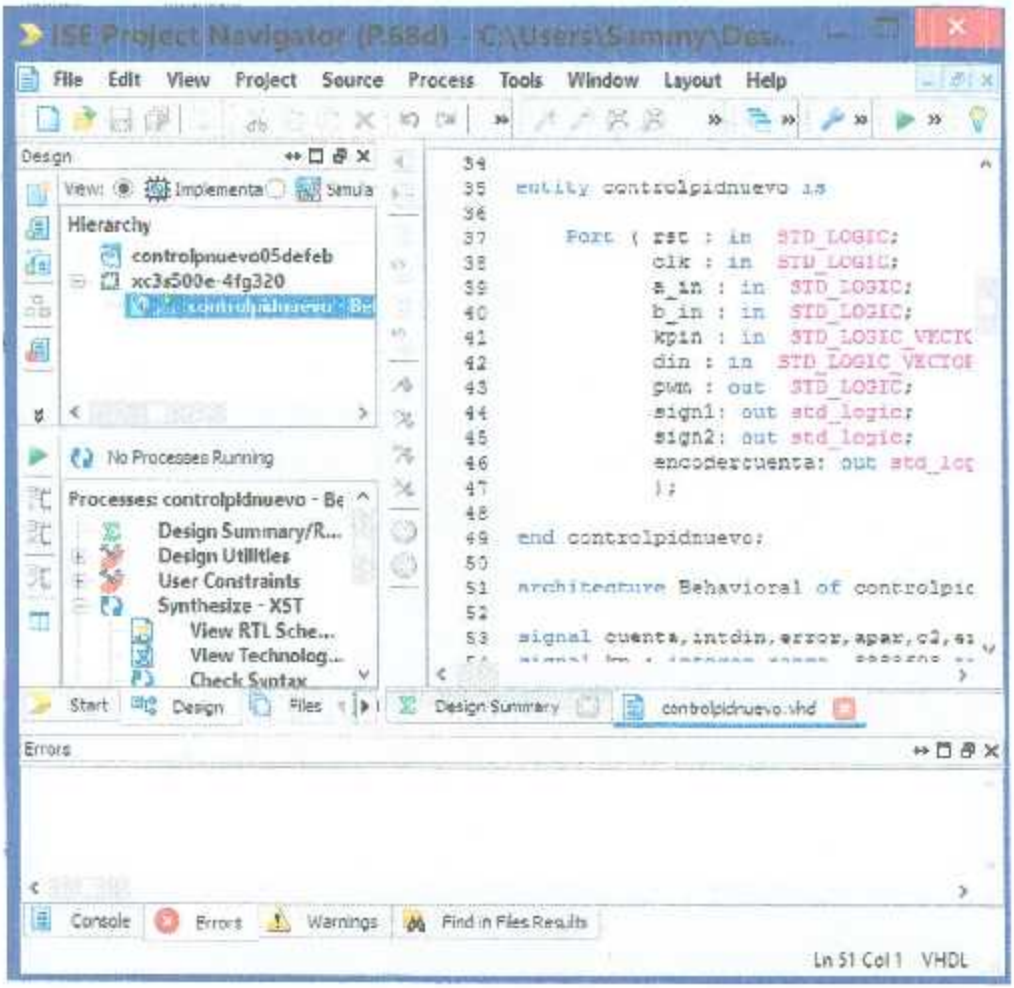


Figura 3.1 Ventana principal del Project Navigator

Para lograr la implementación del control se requiere tener distintos procesos dentro de la lógica de VHDL para poder llevar a cabo el control. El primer punto esencial para llevar acabo esto, es definir las distintas entradas y salidas del puerto de la FPGA. La tabla 3.1 muestra el puerto de entradas y salidas implementadas.

Tabla 3.1 Puerto de entradas y salidas del control del PWM

Entradas	Definincion
Rst	boton de reset para el sistema
Clk	reloj de FPGA a 50 Mhz
a_in	salida del sensor hall del encoder a
b_in	salida del sensor hall del encoder b
kpin	ganancia proporcional
din	direccion de pwm de 16 bits
Salidas	
pwm	salida pwm
sign1	señal de signo para salida del motor
sign2	señal invertida de signo para salida del motor
encodercuenta	cuenta de pulsos del encoder

El código en VHDL dispone de dos procesos en los cuales a través del encoder dado por el motor empleado se tiene el control proporcional completo mediante los pulsos de salida en PWM. Las entradas *a_in* y*b_in* son las salidas que proporciona en el encoder las cuales son tomadas como entradas en el proceso para el control del PWM. El apéndice A.1 muestra el código implementado para lograr el control de PWM mediante un lenguaje VHDL.

Los motores, que se pretende sean posteriormente usados en las articulaciones de un robot manipulador, son motores de DC con encoder incluido y caja de engranajes. Este motor es un motor que trabaja a 12 V con una caja de engranajes de 29:1 y mientras el encoder de cuadratura nos da una resolución de 64 conteos por revolución, por lo que multiplicando por la reducción se vuelve 1856 conteos por revolución. Los motores poseen un diámetro de 6 mm. La figura 3.2 muestra los motores a emplear.

Figura 3.2 motores para la implementación del movimiento articular

Los cables del motor son 6 cables donde se tiene la alimentación del motor del lado positivo y negativo. A Su vez se tiene los cuatro cables correspondientes al sensor Hall del encoder donde dos son su alimentación y los dos restantes son la salida que proporciona. La figura 3.3 muestra la distribución de los cables del motor de DC implementado.

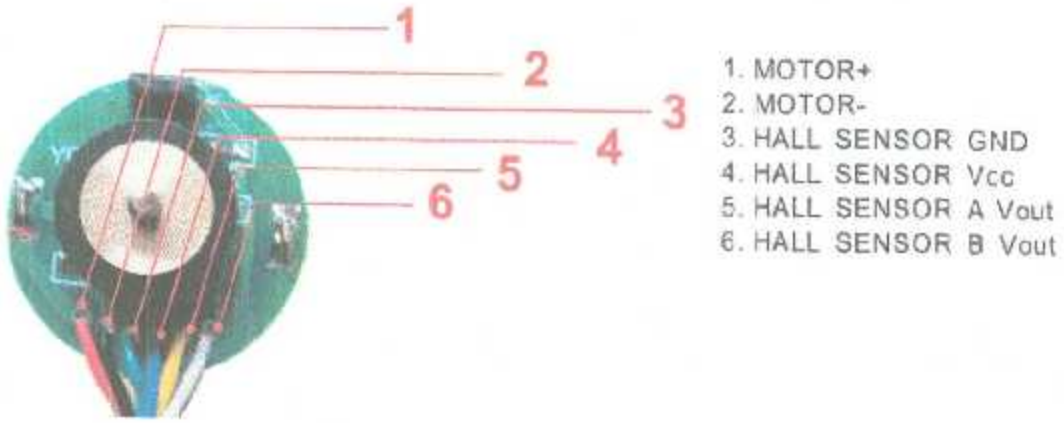


Figura 3.3 distribución de cables del motor DC

Existen dos tipos de dispositivos lógicos programables. Uno es llamado FPGA (*Field Programmable gate array)* y el otro un dispositivo lógico complejo CPLD. Una FPGA es un dispositivo semiconductor que contiene bloques de lógica donde su interconexión se configura mediante un lenguaje de descripción especializado. La

lógica programable puede reproducir desde funciones tan sencillas como llevarlas a una puerta lógica o sistema combinacional.

En la presente investigación se hace uso de una FPGA para programar el código del control de PWM. Se empleó una tarjeta *Spartan 3E XC3S*. La figura 3.4 muestra la FPGA usada para la implementación.

Figura 3.4 FPGA Spartan 3E usada para la implementación

La Spartan 3E posee 500,000 compuertas. Posee 10,476 celdas lógicas como 73K de RAM Bits. Posee distintos conectores tales como: conector FX2 Hirose de 100 pines, tres conectores Pmod de seis pines, DB15HD VGA, conector PS/2 teclado, dos conectores DB9 RS-232, RJ-45 conector Ethernet, conector SMA para entradas de reloj altas. Para la programación la Spartan 3E posee programación JTAG vía el puerto USB2 [Xilinx, 2006].

3.2 Implementación en Labview

Para lograr una interfaz gráfica junto con el código del circuito de control digital en VHDL se usó como software de interfaz el Labview diseñado especialmente para manejo de esta tarjeta FPGA. Es decir, se instaló un módulo especial para FPGAs en Labview para poder trabajar con la Spartan 3E. Con el módulo Labview FPGA

se pueden crear instrumentos virtuales (VI) que corren en dispositivos autorizados RIO o como el caso de la Spartan 3E que es también uno de los dispositivos autorizados. Dependiendo del dispositivo del que se trate se tendrán más o menos entradas y salidas que incluyen recursos digitales o análogos tales como convertidores análogo a digital y viceversa [National Instruments, 2004].

Para lograr una interfaz virtual y una implementación usando un control de posición de un motor con el código VHDL se usa la tarjeta Spartan 3E. La Spartan 3E servirá contendrá el circuito digital desarrollado en el código en VHDL y el instrumento virtual será creado en Labview. El circuito de control creado en VHDL se enlazará al Labview y Labview a través de sus bloques dará la salida deseada dentro de la interfaz y en la Spartan 3E. La figura 3.5 muestra el diagrama sobre la implementación del movimiento del motor de DC usando Labview y el código en VHDL.

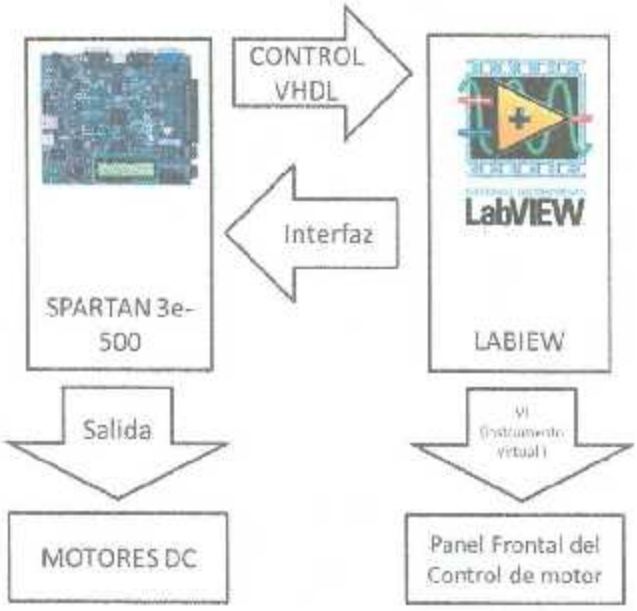


Figura 3.5 Diagrama sobre implementación de movimiento del brazo

La figura 3.6 muestra el explorador de proyectos utilizado cuando se tiene el módulo de FPGA instalado en Labview. En el explorador de proyectos se notan los componentes de la FPGA que se ultilizarán tales como los *LEDs, Conectores de expansión, PushButtons y Switches.* Aparte de los elementos a usar se tiene el

instrumento virtual dentro de la FPGA para poder trabajar con el código VHDL en el ambiente gráfico de Labview.

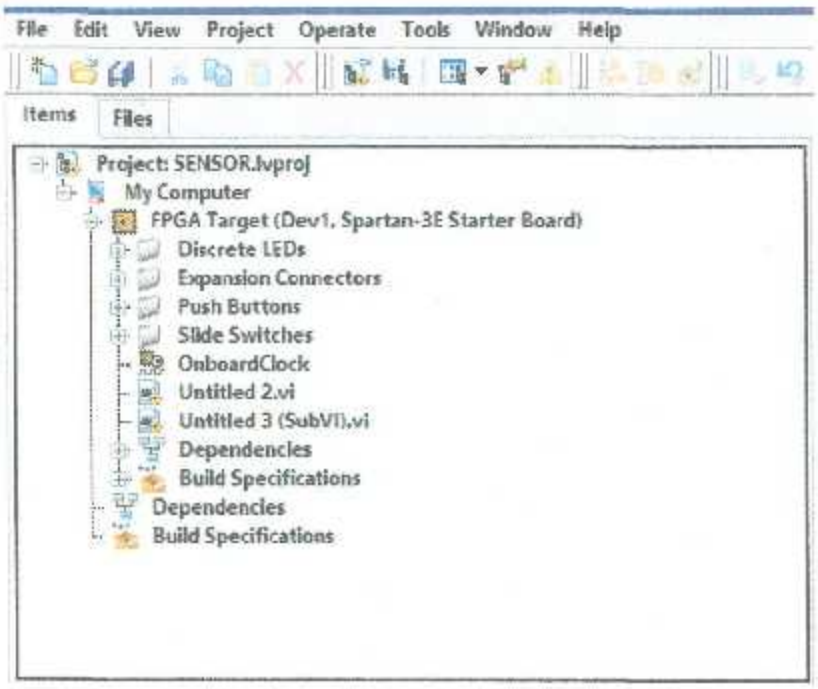


Figura 3.6 Explorador de proyectos de Labview para el control de PWM

En el panel frontal se muestra la dirección y potencia de PWM que se le da por parte del usuario y la ganancia kp. Cabe destacar que el control es solamente un control proporcional por lo que el usuario debe de dar la ganancia proporcional. A su vez, en el panel frontal se tiene los pulsos que genera el encoder y lograr ver el error. La figura 3.7 muestra como se muestra el panel frontal del control de PWM.

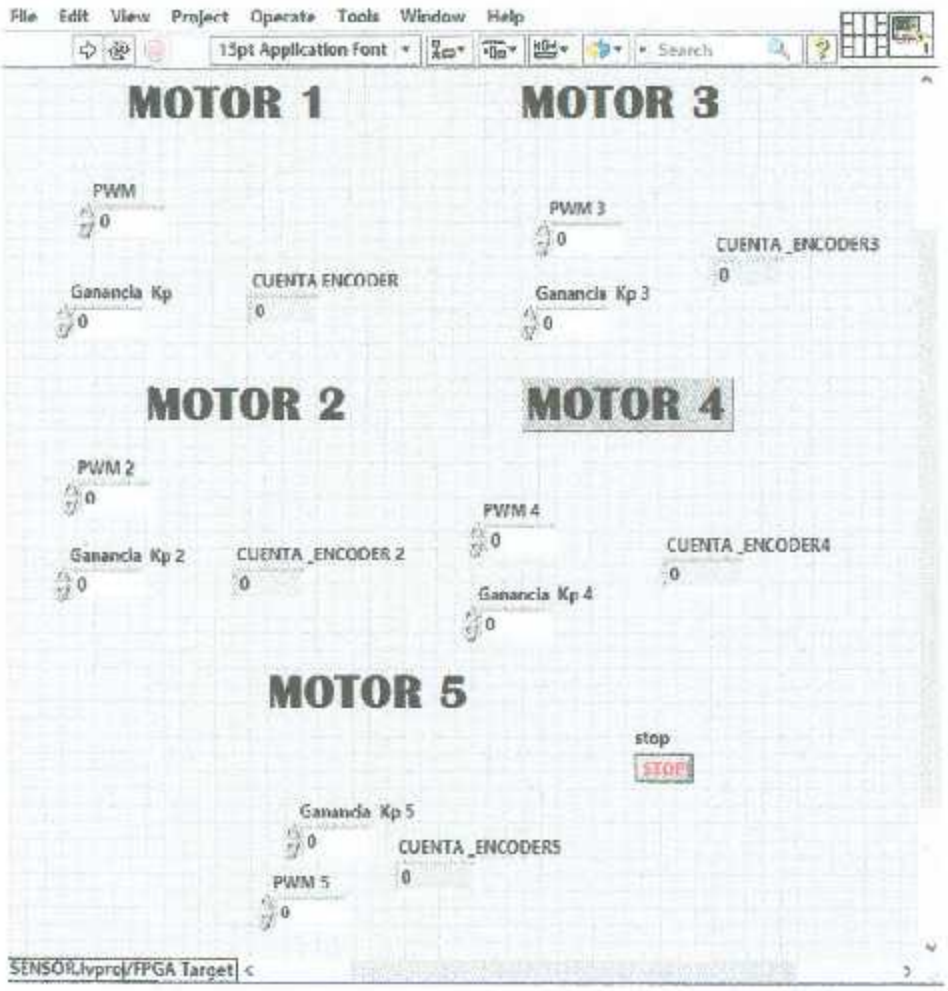


Figura 3.7 Panel frontal del control PWM

Los bloques usados en el programa son los que proporciona el módulo FPGA. Son esenciales los bloques del módulo de FPGA para lograr el control de PWM. Uno de los bloques usados es el nodo FPGA I/O. Este nodo mostrado en la figura 6.8 se puede configurar para entrada o salida de la FPGA tales como LEDs, interruptores, conectores de expansión.

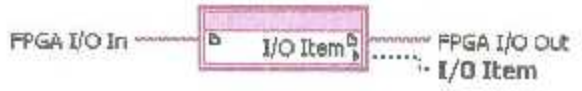


Figura 3.8 Nodo de entrada y salida

Otro nodo o bloque esencial es el nodo de integración IP. Este nodo integra IPs al instrumento virtual de VI. Es decir, es este nodo en donde se carga el archivo

VHDL con el código para integrarlo al instrumento virtual de Labview. La figura 3.9 muestra cómo se visualiza el nodo con el código VHDL integrado.

Figura 3.9 Nodo de integración IP con el código PWM

En el apéndice A.2 se muestra el diagrama completo de bloque donde se tiene el control de PWM mediante un código VHDL y usando los distintos bloques de Labview

3.3 Circuito usado para la implementación

Debido a que la FPGA genera salidas en sus conectores de expansión alrededor de 4 a 4.5V, no es posible levantar el motor de DC con estas especificaciones. Es por esto que se requiere de un driver de potencia para poder manejar con facilidad el motor a través del código dado en la subsección 3.2 de PWM. El driver es básicamente el circuito integrado o chipo denominado "puente H" o L293D. Se usó este circuito debido a que posee diodos de protección que evita daños producidos por los picos de voltaje que llegará a producir el motor.

La figura 3.10 muestra el diagrama de conexiones usado para el control de PWM junto con las salidas de la FPGA conectadas al circuito L293D.

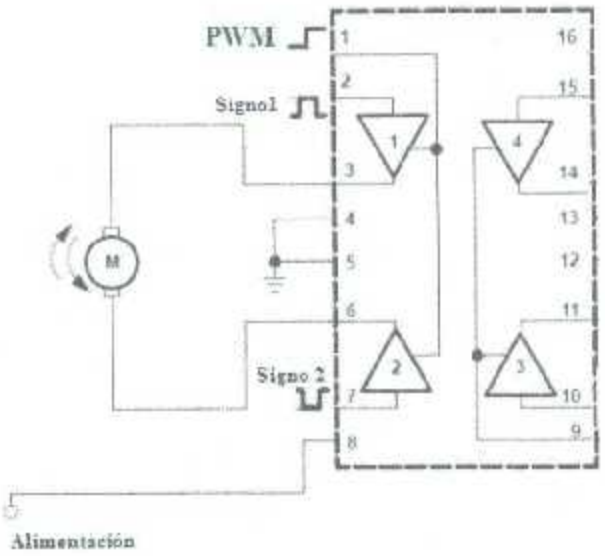


Figura 3.10 Diagrama de conexión del circuito integrado L293D

Las salidas que se emplean para el manejo del motor de DC son las salidas generada por la FPGA PWM, Signo1 y Signo 2. PWM es la salida final de ancho de pulso que manejara la velocidad del motor. Signo 1 y Signo2 manejaran el sentido de la velocidad del motor.

Se logró una implementación básica del manejo de los motores que actuaran como articulaciones del brazo robótico diseñado y modelado en los anteriores capítulos. Para futuras investigación se plantea llevar lo estudiado y lograr tener un modelo físico junto con los eslabones y articulaciones del robot.

Capítulo 4 Resultados

4.1 Resultados del controlador proporcional

En conjunción con Labview, el controlador proporcional solamente hace uso de la ganancia proporcional denominada Kp. Esta ganancia es introducida por el usuario y se puede manipular desde la ventana o panel frontal del archivo host como se muestra en la Figura 4.1.

Figura 4.1 lugar de manipulación de ganancia proporcional

En la Figura 4.1 muestra la sección en donde el usuario podrá introducir una ganancia proporcional que desee. A su vez para las diferentes pruebas empleadas ya sea para la prueba proporcional, proporcional derivativa y proporcional/derivativa/ integral se optó por usar diferentes códigos de programa VHDL usados en un bloque IP dentro del sistema de Labview. Es decir, para la prueba proporcional se empleó solamente un código VHDL que solo acepte como entrada la ganancia proporcional y deshabilite las ganancias derivativa e integral. La Figura 4.2 muestra la sección del código de VHDL donde muestra la deshabilitación de las ganancias derivativa e integral.

```
signal cuenta,intdin,error,apar,c2,errorpasado,par1,par2,par3,par4,M,acumerror, par : integer range -8388608 to 8388608;
signal kp : integer range -8388608 to 8388608;
signal KV : integer range -8388608 to 8388608:=0;
signal ki : integer range -8388608 to 8388608:=0;
signal AB, AB_pas : std_logic_vector (1 downto 0);
signal clk2: std_logic;
signal sign: std_logic;
signal c1 : integer range 0 to 25;
```

Figura 4.2 Código VHDL para deshabilitación de ganancias integral y derivativa.

Para lograr ver el comportamiento del motor con respecto a una señal de entrada, se empleó un bloque en Labview para generar una señal cuadrada. Esta señal será guardada con respecto al tiempo en una tabla y podrá ser comparada con la salida que genere el encoder del motor de DC. El Labview tiene la facilidad de importar datos generados en las gráficas de salida del mismo sistema de Labview. Estos datos importados y generados de Labview son llevados a Excel en donde se generan dos columnas. La figura 4.3 muestra el lugar en donde se encuentra la opción de importar los datos de la gráfica a Excel.

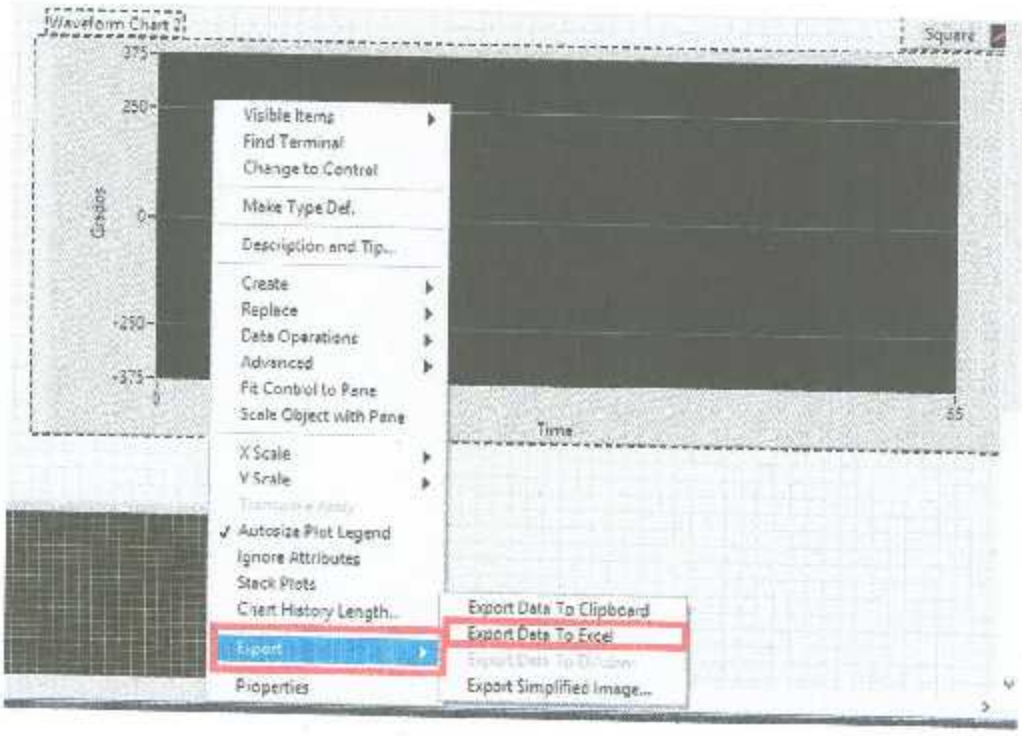
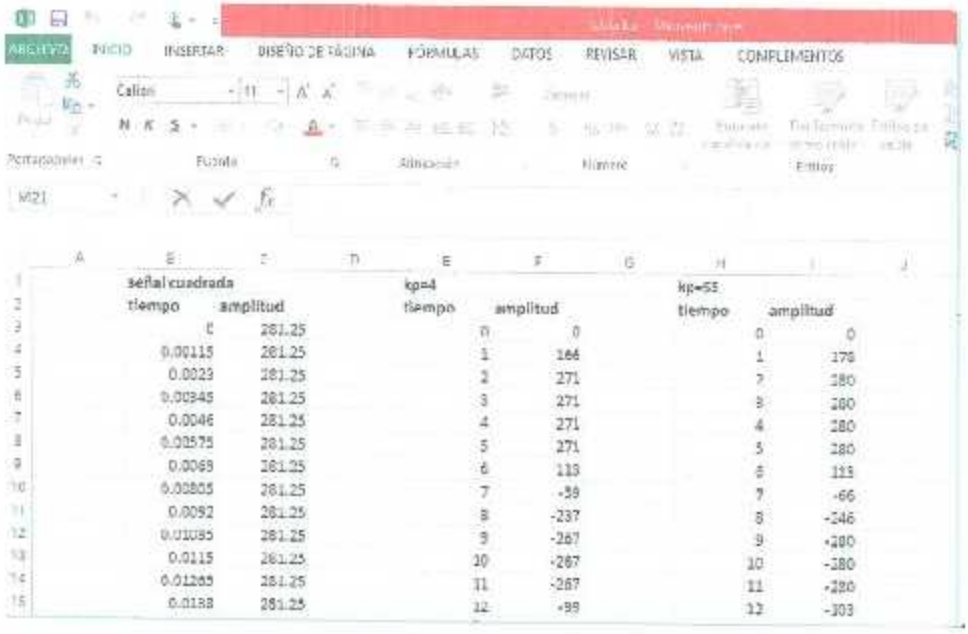


Figura 4.3 Lugar para exportar datos a Excel

La figura 4.4 muestra la ventana de Excel que genera la opción de importación de datos de Labview. La primera columna es definida como el tiempo y sus intervalos que se reflejan en la pantalla de Labview. Es decir, el rango que dispone la columna importada es el rango del eje x de la gráfica importada desde Labview. La segunda columna importada de Excel es el resultado que arroja el motor de DC



	señal cuadrada			kp=4			kp=55	
	tiempo	amplitud		tiempo	amplitud		tiempo	amplitud
3	0	281.25		0	0		0	0
4	0.00115	281.25		1	166		1	178
5	0.0023	281.25		2	271		2	280
6	0.00345	281.25		3	271		3	280
7	0.0046	281.25		4	271		4	280
8	0.00575	281.25		5	271		5	280
9	0.0069	281.25		6	133		6	133
10	0.00805	281.25		7	-59		7	-66
11	0.0092	281.25		8	-237		8	-246
12	0.01035	281.25		9	-267		9	-280
13	0.0115	281.25		10	-267		10	-280
14	0.01265	281.25		11	-267		11	-280
15	0.0138	281.25		12	-99		12	-303

Figura 4.4 Exportación de datos a Excel

Los datos de la segunda columna están en términos de pulsos, por lo que es necesario realizar una conversión de pulsos a grados o radianes. Esta conversión depende de las características del motor DC con encoder.

Los resultados se reflejan en las tablas B.1, B.2 y B.3. La tabla B.1 es la tabla de datos exportados a Excel referente a la señal de entrada cuadrada, generada con un bloque de señal en Labview. La tabla B.2, es generada por la salida de pulsos de Labview con una ganancia proporcional de 4 Nm/ grad. La tabla B.3 muestra los datos obtenidos de Labview exportados a Excel con una ganancia proporcional de 55 Nm/grad. Cabe destacar que la obtención de cada tabla de valores fue debido a una entrada cuadrada con una amplitud constante de 1500 pulsos del encoder, siendo aproximadamente de 280 grados, con un periodo relativamente lento para que el sistema pudiera alcanzar la referencia de la entrada sea reflejada en la gráfica.

Con la ayuda de Matlab, fue posible unir las tres tablas generadas en Labview para poder ver con mayor claridad el funcionamiento del sistema de control del motor de corriente directa cuando se tiene como entrada una señal cuadrada

constante. Con la ayuda de un código de Maltab se pudo lograr esto y da como resultado la figura 4.5.

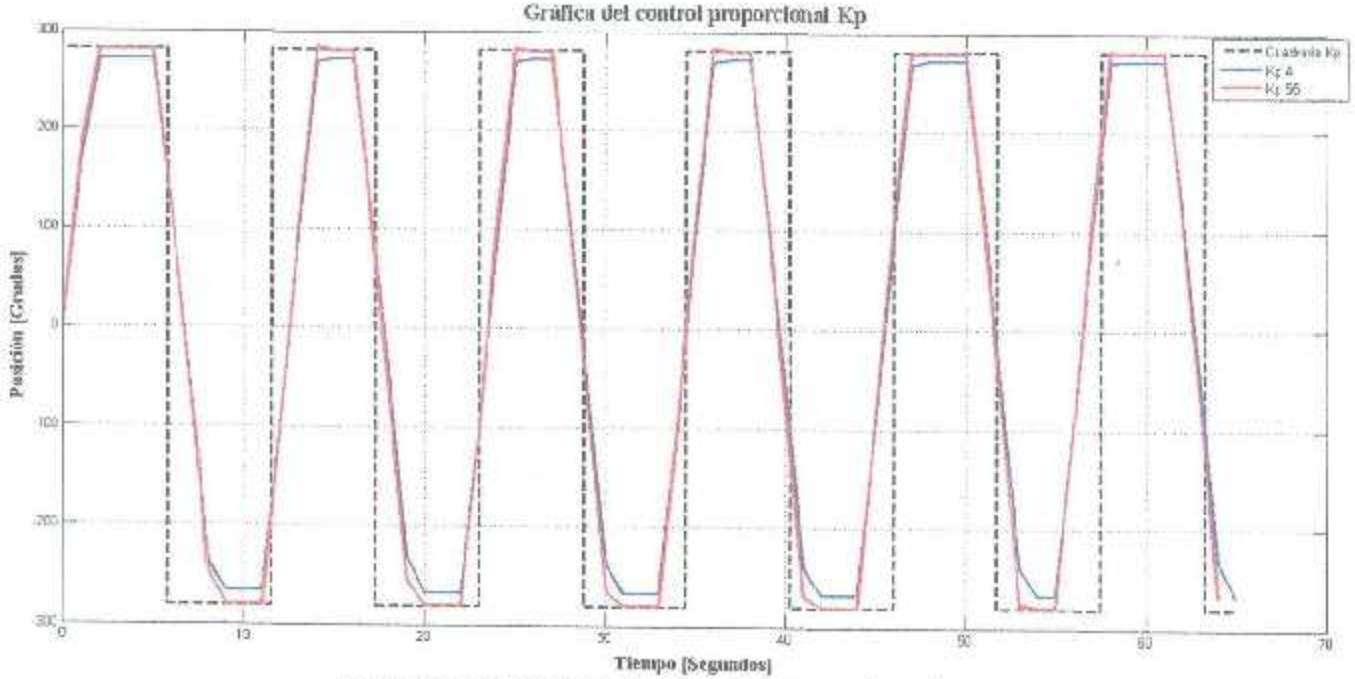


Figura 4.5 Gráfica de control proporcional

La figura 4.5 muestra la señal cuadrada en forma de color negro y lineas punteadas y dos salidas respectivas con su ganancia proporcional. La salida de color azul refleja una ganancia proporcional kp de 4 Nm/ grad. La salida de color rojo refleja una ganancia de 55 Nm/grad. El tiempo de respuesta es aceptable y se denota que conforme avanza la ganancia de valor se acerca más a la señal de entrada. Esto se nota debido a que la salida roja por cierto tiempo llega a alcanzar el valor máximo y mínimo de la señal de entrada mientras que la salida azul no logró ese objetivo. Teniendo un control proporcional es posible tener un sistema adecuado para el banco de articulaciones propuesto en esta investigación.

4.2 Resultados del controlador proporcional/derivativo

Siguiendo la misma metodología de la sección 4.1, se obtiene el sistema de control proporcional y derivativo en donde en el código de programa se habilita la ganancia derivativa que previamente estaba deshabilitada. La figura 4.6 muestra la habilitación de la ganancia derivativa.

```
architecture Behavioral of controlPI is
signal cuenta,intdin,error,apar,c2,errorpasado,par1,par2,
signal kp : integer range -8388608 to 8388608;
signal kv : integer range -8388608 to 8388608;
signal ki : integer range -8388608 to 8388608:=0;
signal AB, AB_pas : std_logic_vector (1 downto 0);
signal clk2: std_logic;
signal sign: std_logic;
signal c1 : integer range 0 to 25;
```

Figura 4.6 Habilitación de ganancia derivativa

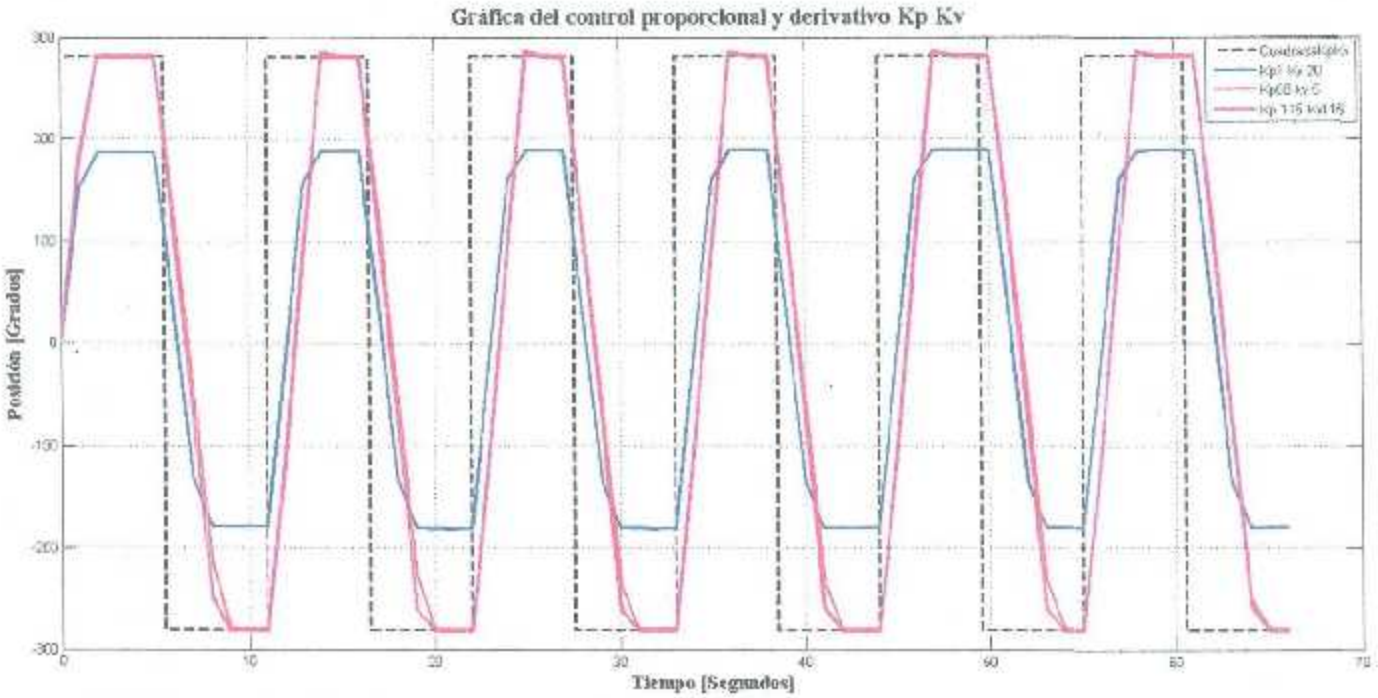
La figura 4.7 muestra el lugar de introducción de las ganancias proporcional y derivativa dentro del programa de Labview en donde se reflejarán los resultados y se exportaran usando la misma metodología propuesta en la sección 4.1.

Figura 4.7 Lugar para introducción de ganancias

Las tablas C.1, C.2, C.3, C.4 son el resultado del sistema proporcional derivativo propuesto con tres distintas ganancias kp (proporcional) y kv (derivativo). Usando de nuevo la ayuda de Matlab se logró graficar todas las salidas con sus ganancias comparándolas con la señal de entrada cuadrada.

La tabla C.1 es la tabla de datos exportados de la señal de entrada cuadrada con una amplitud de 280. La tabla C.2 es la tabla de salida correspondiente a una ganancia proporcional de 1 Nm/grad y ganancia derivativa de 20 Nm*seg/grad. La tabla C.3 tiene una kp de 50 Nm/grad y 5 Nm*seg/grad de kv. Por último, la tabla C.4 muestra la salida de una ganancia proporcional de 115 Nm/grad al igual que la ganancia derivativa.

La figura 4.8 muestra la gráfica en Matlab donde se unen las tablas para ver su comparación con respecto a la señal de entrada cuadrada. La señal cuadrada se refleja con una línea de color negro en trazos. La salida azul refleja el sistema más limitado con una ganancia kp de 1 Nm/grad y kv de 20 Nm*seg/grad. Las otras dos salidas tuvieron un comportamiento más apegado a la señal de entrada sin embargo, la señal violeta con una ganancia de 115 para ambas ganancias fue la



que obtuvo un mejor desempeño en el sistema.

Figura 4.8 Grafica de control proporcional- derivativo

Capítulo 5 Conclusiones

Las pruebas experimentales del sistema de control digital para diferentes ganancias nos muestran, a través de las gráficas obtenidas, que el sistema funciona. Buenos resultados, son obtenidos de los diferentes experimentos que fueron llevados a

cabo. El enlace que se hace mediante el LabView permite tener una interfaz visual donde se pueda interactuar con el controlador de forma interactiva en línea.

Existen varias contribuciones dentro de la presente investigación tales como:

- Implementación de diferentes circuitos VHDL a sobre una FPGA.
- Enlace para monitoreo y parámetros de control usando LabView.
- Control de motores de DC a través de circuitos de VHDL
- Se presentó el artículo nombrado: "Diseño de un brazo robótico para el posicionamiento de antenas para ablación por microondas" en congreso CIESLAG llevado a cabo en el Instituto Tecnológico de la Laguna.

Como trabajo de investigación futura queda pendiente llevar lo implementado a un nivel más arriba en la práctica. Es decir, construir un robot manipulador, sus eslabones y poner el control diseñado en los motores del robot manipulador para lograr como finalidad alcanzar la posición y orientación deseada con elemento final que es la antena de ablación.

Bibliografía

[Aníbal Ollero, 2001]. Ollero Aníbal. "Robótica: Manipuladores y robots móviles". Barcelona, Marcombo, 2001.

[Barrientos et al., 1997]. Barrientos A., Peñín L. F., Balaguer C., Aracil R., "Fundamentos de Robótica", Ed. Mc Graw-Hill, 1997.

[Cöltekin et al., 1999]. Cöltekin, Arzu. "Studying Geometry, Color and Texture in VRML" Institute of Photogrammetry and Remote Sensing. Helsinki University of Technology, 1999.

[Craig, 2006]. J. Craig, John. "Introduction to Robotics Mechanics & Control" Addison- Wesley Publishing Company, 2006.

[Denavit – Hartenberg, 1955]. Denavit, J. y Hartenberg E. "A kinematic notation for lower-pair mechanisms base on matrices". Journal of Applied Mechanics, vol. 77, pp 215-221, 1955.

[El-Sherbin et al,. 2011] Sherwin, C.M. 2011. "Can invertebrates suffer? Or how robust is argument-by-analogy?" Animal Welf. 10, S103-S118, 2011.

[Jian Sun, 2012]. Sun, Jian. "Pulse Width Modultation". Department of Electrical, Computer, and Systems Engineering, Rensselaer Polytechnic Institute, Troy, NY, 2012.

[Jung, 2007]. J. Kim, H. Jeon, S. Jung, "Hardware implementation of nonlinear PID controller with FPGA based on floating point operation for 6-DOF manipulator robot arm, in Proceedings of the International Conference on Control, Automation and Systems", ICCAS '07., pp.1066-1071, 17-20 Oct. 2007.

[Kai, Lui, 1993]. Liu Kai, Fitzgerald John M. & Lewis Frank L. "Kinematic Analysis of a Stewart Platform Manipulator". IEEE Transactionson Industrial Electronics, Vol. 40, No. 2, 1993.

[Lavista et al, 2008]. Lavista LG, Vázquez, Flores- Balcazar, CH., Méndez-Probst CE, "Actas Urol Esp.", 32(10) pp. 985-988, 2008.

[Lobo, 2002] Lobo, P.J. "APLICACIONES DEL LENGUAJE VRML (VIRTUAL REALITY MODELLING LANGUAGE) A LA CIENCIA DE MATERIALES" EUIT de Telecomunicación, Ctra. de Valencia, Madrid. España, 2002.

[Mathworks, 2015]. Matlab& Simulink. "Simulink 3D Animation User's Guide", The MathWorks, Inc. Marzo 2015.

[Moore, 2007]. Moore, Holly. "MATLAB para ingenieros". Pearson Education, Inc. Prentice Hall, 2007.

[Ramírez, 2008]. Ramirez Woo C., "Modelador dinámico y control en modo par del robot Mitsubishi PA10-7CE". Tesis de Maestría, Instituto Tecnológico de la Laguna, División de Estudios de Posgrado e Investigación, 2008.

[Santisteban, 2006]. Solidoro Santisteban, Andrés. "Cáncer en el Siglo XXI, Cáncer in XXIst century". Acta Med Per. 23(2), pp 112- 118, 2006.

[Xilinx, 2011]. Xillinx. "ISE Simulator (ISim) In-Depth Tutorial". Marzo 1, 2011.

[Xilinx, 2006]. Xillinx. "Spartan-3E Starter KitBoardUserGuide". Marzo 9, 2006.

[Venegas, 2009]. "Encoders" Venegas, Javier Requena, 2009.

APÉNDICE A Implementación del brazo robótico

A.1 Código VHDL para el control de motor mediante PWM

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

```
use IEEE.std_logic_arith.all;
use IEEE.std_logic_signed.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 controlpidnuevo is

   Port ( rst : in  STD_LOGIC;
clk : in  STD_LOGIC;
a_in : in  STD_LOGIC;
b_in : in  STD_LOGIC;
kpin : in  STD_LOGIC_VECTOR (7 downto 0);
                        din : in  STD_LOGIC_VECTOR (15 downto 0);
pwm : out  STD_LOGIC;
                         sign1: out std_logic;
                         sign2: out std_logic;
                        encodercuenta: out std_logic_vector(23 downto 0)
        );

end controlpidnuevo;

architecture Behavioral of controlpidnuevo is
```

```
signalcuenta,intdin,error,apar,c2,errorpasado,par1,par2,par3,par4,m,acumerror, par : integerrange -
8388608 to 8388608;
signal kp : integer range -8388608 to 8388608;
signal kv : integer range -8388608 to 8388608:=0;
signal ki : integer range -8388608 to 8388608:=0;
signal AB, AB_pas :std_logic_vector (1 downto 0);
signal clk2: std_logic;
signal sign: std_logic;
signal c1 : integer range 0 to 25;


begin

process(rst,clk,AB,AB_pas)
begin
        if rst = '1' then
                AB <= a_in&b_in;
                AB_pas<= AB;
                cuenta<= 0;
        elsifclk='1' and clk'event then
                AB <= a_in&b_in;
                AB_pas<= AB;
                case AB is
                        when "00" =>
                                if AB_pas = "01" then cuenta<= cuenta + 1;
                                elsifAB_pas = "10" then cuenta <= cuenta - 1;
                                end if;
                        when "01" =>
                                if AB_pas = "11" then cuenta<= cuenta + 1;
```

```
                    elsifAB_pas = "00" then cuenta <= cuenta - 1;
                    end if;
                when "11" =>
                    if AB_pas = "10" then cuenta<= cuenta + 1;
                    elsifAB_pas = "01" then cuenta <= cuenta - 1;
                    end if;
                when "10" =>
                    if AB_pas = "00" then cuenta<= cuenta + 1;
                    elsifAB_pas = "11" then cuenta <= cuenta - 1;
                    end if;
                when others =>
                    cuenta<= cuenta;
            end case;
        end if;
end process;

intdin<= conv_integer(din);
apar<= -par when par<0 else par; -- error absoluto
sign<= '1' when par<0 else '0'; -- error absoluto
--ref<= CONV_STD_LOGIC_VECTOR(apar, 10); -- escalamiento
pwm<= '1' when apar> c2 else '0';
--error <= intdin- cuenta;  --calculo del error
--par <= kp * error; -- Kp por el error
kp<= conv_integer(kpin);


process(rst,clk,c1)
begin
        if rst = '1' then
            c1 <= 0;
```

```
            clk2 <= '1';
        elsifclk'event and clk='1' then
            if c1 < 25 then
                c1 <= c1 +1;
            else
                clk2 <= not (clk2) ;
                c1 <= 0;
            end if;
        end if;
end process;

process(rst,clk2) -- Pulsosporrevolucion
begin
        if rst = '1' then
            c2 <= 0; --(others => '0' );
        elsif clk2'event and clk2='1' then
            if c2 < 1000 then
                c2 <= c2 + 1;
            else
                c2 <= 0;
                errorpasado<= error;
                error<= intdin-cuenta;
                par1<= error*kp;
                m<= error-errorpasado;
                par2<= m* kv;
                acumerror<= acumerror + error;
                par3<= (acumerror* ki)/1000;
            end if;
        end if;
endprocess;
```

```
par4<= par1- par2;
par<= par4+ par3;
sign1<= sign;
sign2<= not sign;
encodercuenta<= CONV_STD_LOGIC_VECTOR(cuenta,24);

end Behavioral;
```

A.2 Diagrama de bloque del control PWM en Labview

Debido a que el diagrama de bloques es largo se dividió en dos partes el diagrama que se dan en la figura A.1 y A.2.

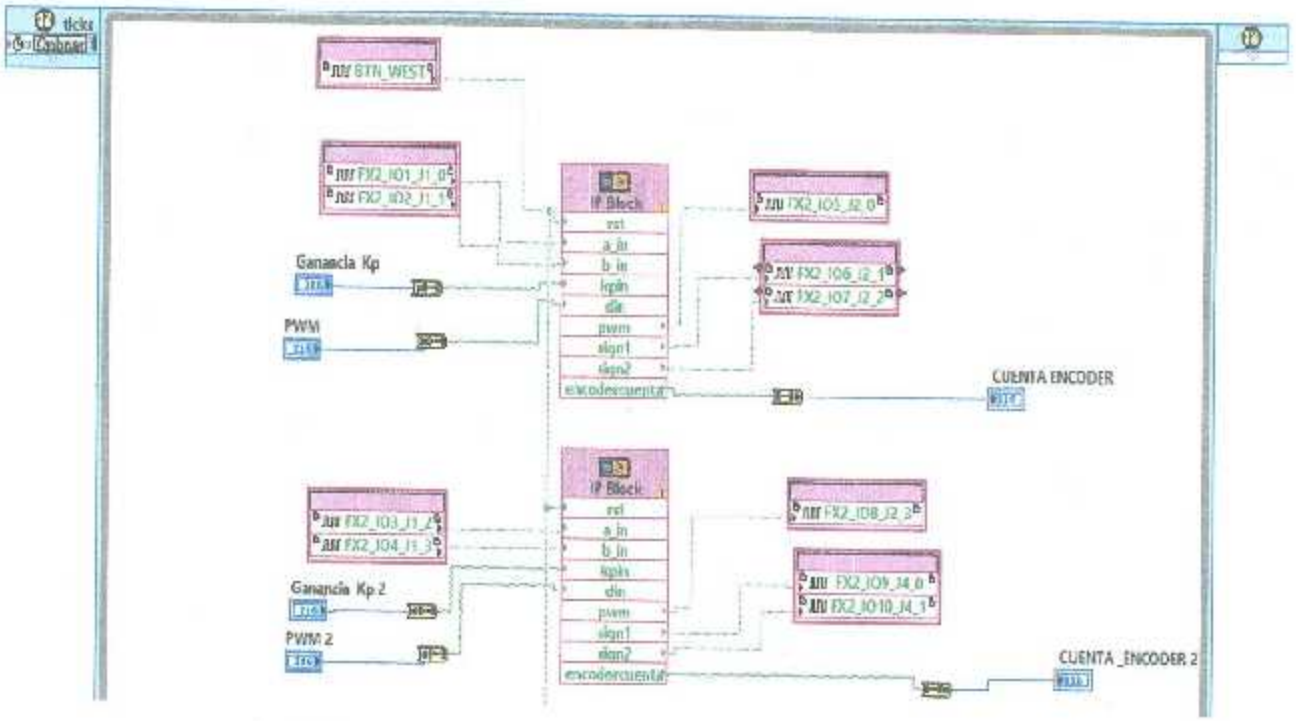


Figura A.1 Primera parte del diagrama de bloques

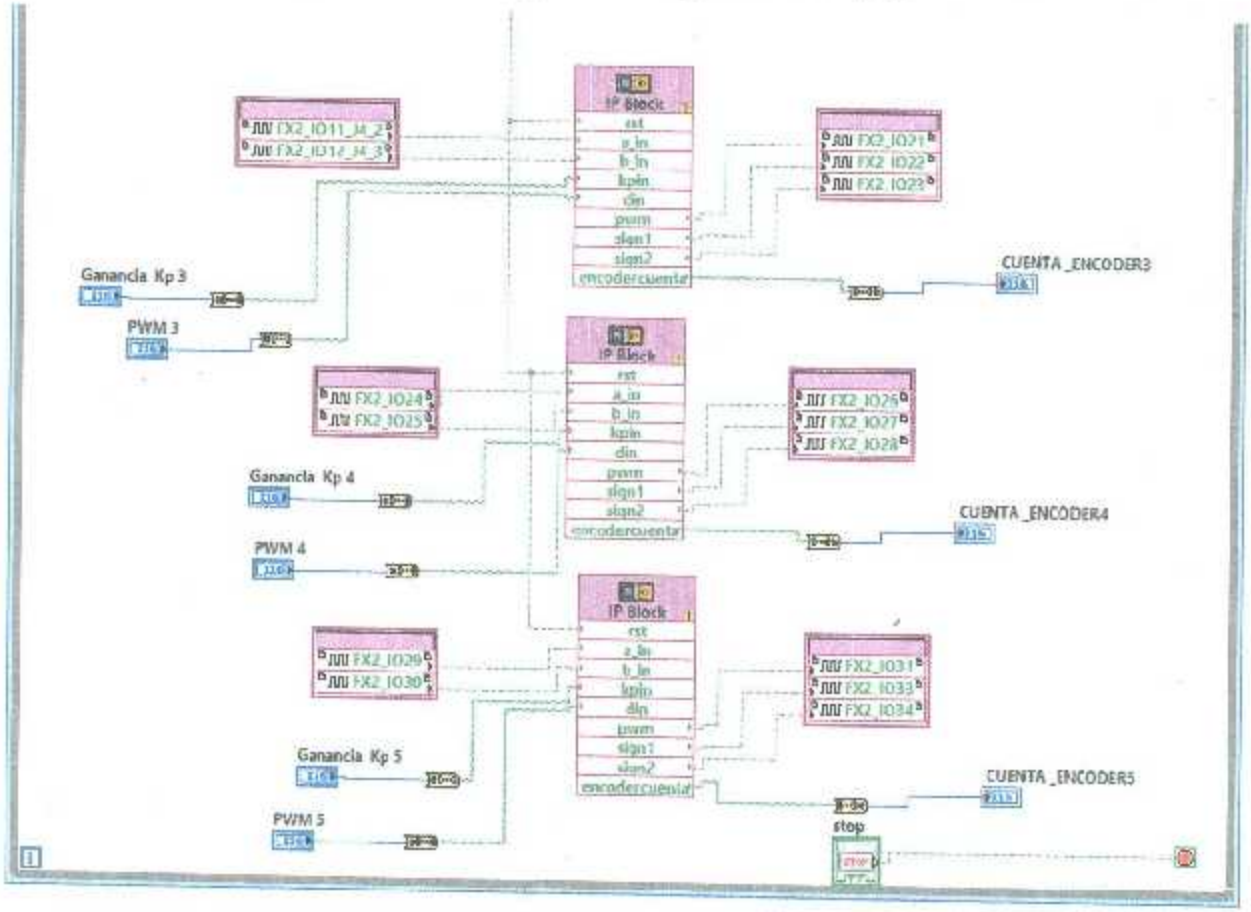


Figura A.2 Segunda parte del diagrama de bloques

B.1 Tabla de datos de señal cuadrada

señal cuadrada	
tiempo	amplitud
0	281.25
0.00115	281.25
0.0023	281.25
0.00345	281.25
0.0046	281.25
0.00575	281.25
0.0069	281.25
0.00805	281.25
0.0092	281.25
0.01035	281.25
0.0115	281.25
0.01265	281.25
0.0138	281.25
0.01495	281.25
0.0161	281.25
0.01725	281.25
0.0184	281.25
0.01955	281.25

0.0207	281.25
0.02185	281.25
0.023	281.25
0.02415	281.25
0.0253	281.25
0.02645	281.25
0.0276	281.25
0.02875	281.25
0.0299	281.25
0.03105	281.25
0.0322	281.25
0.03335	281.25
0.0345	281.25
0.03565	281.25
0.0368	281.25
0.03795	281.25
0.0391	281.25
0.04025	281.25
0.0414	281.25
0.04255	281.25
0.0437	281.25
0.04485	281.25
0.046	281.25
0.04715	281.25
0.0483	281.25
0.04945	281.25
0.0506	281.25
0.05175	281.25
0.0529	281.25
0.05405	281.25
0.0552	281.25
0.05635	281.25
0.0575	281.25
0.05865	281.25
0.0598	281.25
0.06095	281.25
0.0621	281.25
0.06325	281.25
0.0644	281.25
0.06555	281.25
0.0667	281.25

0.06785	281.25
0.069	281.25
0.07015	281.25
0.0713	281.25
0.07245	281.25
1.26615	281.25
1.2673	281.25
1.26845	281.25
1.2696	281.25
1.27075	281.25
1.2719	281.25
1.27305	281.25
2.3759	281.25
2.37705	281.25
2.3782	281.25
2.37935	281.25
2.3805	281.25
2.38165	281.25
2.3828	281.25
3.91115	281.25
3.9123	281.25
3.91345	281.25
3.9146	281.25
3.91575	281.25
3.9169	281.25
7.84185	-281.25
7.843	-281.25
7.84415	-281.25
7.8453	-281.25
7.84645	-281.25
7.8476	-281.25
9.3035	-281.25
9.30465	-281.25
9.3058	-281.25
9.30695	-281.25
11.5069	281.25
11.50805	281.25
11.5092	281.25
11.51035	281.25
11.5115	281.25
11.51265	281.25

11.5138	281.25
15.54225	281.25
15.5434	281.25
15.54455	281.25
15.5457	281.25
15.54685	281.25
15.548	281.25
20.87135	-281.25
20.8725	-281.25
20.87365	-281.25
20.8748	-281.25
20.87595	-281.25
20.8771	-281.25
25.5001	281.25
25.50125	281.25
25.5024	281.25
25.50355	281.25
25.5047	281.25
25.50585	281.25
25.507	281.25
32.9751	-281.25
32.97625	-281.25
32.9774	-281.25
32.97855	-281.25
32.9797	-281.25
36.3009	281.25
36.30205	281.25
36.3032	281.25
36.30435	281.25
36.3055	281.25
41.97615	-281.25
41.9773	-281.25
41.97845	-281.25
41.9796	-281.25
41.98075	-281.25
46.13455	281.25
46.1357	281.25
46.13685	281.25
46.138	281.25
46.13915	281.25
49.0981	281.25

49.09925	281.25
49.1004	281.25
49.10155	281.25
49.1027	281.25
49.10385	281.25
53.8407	-281.25
53.84185	-281.25
53.843	-281.25
53.84415	-281.25
53.8453	-281.25
53.84645	-281.25
53.8476	-281.25
60.3658	281.25
60.36695	281.25
60.3681	281.25
60.36925	281.25
60.3704	281.25
64.99225	-281.25
64.9934	-281.25
64.99455	-281.25
64.9957	-281.25
64.99685	-281.25
64.998	-281.25
64.99915	-281.25

B.2 Tabla de datos de salida de encoder con ganancia proporcional de 4 Nm/rad

kp=4	
tiempo	**amplitud**
0	0
1	166
2	271
3	271
4	271
5	271
6	113
7	-59
8	-237
9	-267

10	-267
11	-267
12	-99
13	91
14	269
15	272
16	272
17	115
18	-56
19	-232
20	-268
21	-268
22	-268
23	-101
24	91
25	269
26	272
27	272
28	114
29	-60
30	-238
31	-268
32	-268
33	-268
34	-99
35	91
36	269
37	272
38	272
39	116
40	-62
41	-240
42	-268
43	-268
44	-268
45	-101
46	88
47	268
48	272
49	272
50	272

51	116
52	-65
53	-240
54	-268
55	-268
56	-97
57	96
58	271
59	272
60	272
61	272
62	120
63	-56
64	-233
65	-268

B.3 Tabla de datos de salida de encoder con ganancia proporcional de 55 Nm/rad

kp=55	
tiempo	amplitud
0	0
1	178
2	280
3	280
4	280
5	280
6	113
7	-66
8	-246
9	-280
10	-280
11	-280
12	-103
13	99
14	285
15	280
16	280
17	112
18	-71

19	-255
20	-280
21	-280
22	-280
23	-99
24	108
25	283
26	280
27	280
28	109
29	-76
30	-266
31	-280
32	-280
33	-280
34	-99
35	104
36	283
37	280
38	280
39	110
40	-80
41	-269
42	-280
43	-280
44	-280
45	-93
46	110
47	280
48	280
49	280
50	280
51	108
52	-82
53	-276
54	-279
55	-279
56	-97
57	108
58	282
59	280

60	280
61	280
62	110
63	-77
64	-268

C.1 Tabla de datos de señal cuadrada

señal cuadrada	
tiempo	amplitud
0	281.25
0.0011	281.25
0.0022	281.25
0.0033	281.25
0.0044	281.25
0.0055	281.25
0.0066	281.25
0.0077	281.25
0.0088	281.25
0.0099	281.25
0.011	281.25
0.0121	281.25
0.0132	281.25
0.0143	281.25
0.0154	281.25
4.466	281.25
4.4671	281.25
4.4682	281.25
4.4693	281.25
4.4704	281.25
4.4715	281.25
4.4726	281.25
4.4737	281.25
4.4748	281.25
4.4759	281.25
4.477	281.25
9.8758	-281.25
9.8769	-281.25
9.878	-281.25
9.8791	-281.25
9.8802	-281.25

9.8813	-281.25
9.8824	-281.25
9.8835	-281.25
9.8846	-281.25
9.8857	-281.25
9.8868	-281.25
9.8879	-281.25
9.889	-281.25
9.8901	-281.25
14.2131	281.25
14.2142	281.25
14.2153	281.25
14.2164	281.25
14.2175	281.25
14.2186	281.25
14.2197	281.25
14.2208	281.25
14.2219	281.25
14.223	281.25
14.2241	281.25
20.4721	-281.25
20.4732	-281.25
20.4743	-281.25
20.4754	-281.25
20.4765	-281.25
20.4776	-281.25
20.4787	-281.25
20.4798	-281.25
20.4809	-281.25
20.482	-281.25
35.6378	281.25
35.6389	281.25
35.64	281.25
35.6411	281.25
35.6422	281.25
35.6433	281.25
35.6444	281.25
35.6455	281.25
35.6466	281.25
35.6477	281.25
35.6488	281.25

35.6499	281.25
39.259	-281.25
39.2601	-281.25
39.2612	-281.25
39.2623	-281.25
39.2634	-281.25
39.2645	-281.25
39.2656	-281.25
39.2667	-281.25
39.2678	-281.25
39.2689	-281.25
44.0814	281.25
44.0825	281.25
44.0836	281.25
44.0847	281.25
44.0858	281.25
44.0869	281.25
51.7814	-281.25
51.7825	-281.25
51.7836	-281.25
51.7847	-281.25
51.7858	-281.25
51.7869	-281.25
51.788	-281.25
65.9923	-281.25
65.9934	-281.25
65.9945	-281.25
65.9956	-281.25
65.9967	-281.25
65.9978	-281.25
65.9989	-281.25
66	-281.25

C.2 Tabla de datos de salida de encoder con ganancia proporcional de 1 Nm/grad y ganancia derivativa de 20 Nm*seg/grad

kp=1 kv=20	
tiempo	amplitud
0	0

1	152
2	187
3	187
4	187
5	187
6	34
7	-131
8	-179
9	-179
10	-179
11	-179
12	-14
13	157
14	189
15	189
16	189
17	36
18	-134
19	-180
20	-181
21	-182
22	-182
23	-12
24	160
25	189
26	189
27	189
28	33
29	-135
30	-180
31	-181
32	-182
33	-182
34	-11
35	159
36	189
37	189
38	189
39	31
40	-136
41	-180

42	-180
43	-180
44	-180
45	-6
46	162
47	189
48	189
49	189
50	189
51	33
52	-135
53	-180
54	-180
55	-180
56	-9
57	160
58	188
59	189
60	189
61	189
62	33
63	-135
64	-180
65	-180
66	-180

C.3 Tabla de datos de salida de encoder con ganancia proporcional de 50 Nm/grad y ganancia derivativa de 5 Nm*seg/grad

kp=50 kv=5	
tiempo	amplitud
0	0
1	177
2	280
3	280
4	280
5	280
6	128
7	-40

8	-213
9	-280
10	-280
11	-280
12	-110
13	83
14	281
15	280
16	280
17	128
18	-47
19	-227
20	-280
21	-280
22	-280
23	-108
24	90
25	284
26	280
27	280
28	122
29	-55
30	-234
31	-280
32	-280
33	-280
34	-108
35	88
36	285
37	280
38	280
39	124
40	-52
41	-232
42	-280
43	-280
44	-280
45	-111
46	90
47	286
48	280

49	280
50	280
51	122
52	-52
53	-231
54	-280
55	-280
56	-105
57	99
58	287
59	280
60	280
61	280
62	118
63	-65
64	-250
65	-280
66	-280

C.4 Tabla de datos de salida de encoder con ganancia proporcional de 115 Nm/grad y ganancia derivativa de 115 Nm*seg/grad

kp=110 kv=115	
tiempo	amplitud
0	0
1	180
2	282
3	282
4	282
5	282
6	116
7	-65
8	-250
9	-282
10	-282
11	-282
12	-104

13	96
14	286
15	282
16	282
17	115
18	-73
19	-261
20	-282
21	-282
22	-282
23	-102
24	98
25	286
26	282
27	282
28	115
29	-74
30	-260
31	-282
32	-282
33	-282
34	-101
35	98
36	285
37	282
38	282
39	114
40	-74
41	-258
42	-282
43	-282
44	-282
45	-103
46	97
47	286
48	282
49	282
50	282
51	114
52	-73
53	-260

54	-282
55	-282
56	-103
57	98
58	285
59	282
60	282
61	282
62	113
63	-73
64	-257
65	-282
66	-282