UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO

PROGRAMA DE MAESTRÍA Y DOCTORADO EN INGENIERÍA
INGENIERÍA ELÉCTRICA – SISTEMAS ELECTRÓNICOS

CONTROLADOR PARA EL DESPLAZAMIENTO DE UNA PLATAFORMA MÓVIL EN APLICACIONES DE NAVEGACIÓN UTILIZANDO FPGA.

TESIS
QUE PARA OPTAR POR EL GRADO DE:
MAESTRO EN INGENIERÍA

PRESENTA:
ING. JUAN ANTONIO GÓMEZ MEDINA

TUTOR PRINCIPAL
Dr. JUAN MARIO PEÑA CABRERA, INSTITUTO DE INVESTIGACIONES EN MATEMÁTICAS APLICADAS Y EN SISTEMAS

MÉXICO, D. F. AGOSTO 2014

JURADO ASIGNADO:

Presidente:	DR. MARTÍNEZ LÓPEZ JOSÉ ISMAEL
Secretario:	DR. RODRÍGUEZ CUEVAS JORGE
Vocal:	DR. PEÑA CABRERA JUAN MARIO
1 er. Suplente:	M.I. HARO RUÍZ LUIS ARTURO
2 do. Suplente:	M.I. ÁLVAREZ CASTILLO JESÚS

LUGAR DONDE SE REALIZÓ LA TESIS: INSTITUTO DE INVESTIGACIONES EN MATEMÁTICAS APLICADAS Y EN SISTEMAS, UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO

TUTOR DE TESIS:

DR. PEÑA CABRERA JUAN MARIO

FIRMA

(Segunda hoja)

DEDICATORIAS:

A mi familia, por respetar mis decisiones y apoyar en todo momento a esta oveja gris de la familia.

A mis padres, por sus apoyos, consejos y regaños a lo largo de mi vida.

A mis amigos, por su apoyo y ayuda incondicional para seguir adelante en todo momento tanto en lo profesional como lo personal.

A las personas que forman parte fundamental en mi vida, por su apoyo y cariño en todo momento.

Al próximo Dr. Lucio Fidel Rebolledo y al Dr. Juan Mario Peña, por su apoyo tanto de la licenciatura por parte del Dr. Lucio, como a lo largo de la maestría por parte del Dr. Mario, en ayudarme a desarrollarme profesionalmente.

Gracias.

AGRADECIMIENTOS

- A la Universidad Nacional Autónoma de México, por su apoyo y permitirme crecer, tanto personal como intelectualmente.

- Al Instituto de Investigaciones en Matemáticas Avanzadas y en Sistemas, Por el apoyo otorgado en la realización de este trabajo y su confianza.

- Al Dr. Juan Mario Peña Cabrera, por su apoyo incondicional tanto en el logro de esta meta, como en mi desarrollo profesional.

- Al Dr. José Ismael Martínez López, Dr. Jorge Rodríguez Cuevas, M.I. Luis Arturo Haro Ruíz y M.I. Jesús Álvarez Castillo por el tiempo dedicado a la revisión de este trabajo.

- Al Programa de Posgrado de Ingeniería en Sistemas Electrónicos, por el seguimiento y orientación a lo largo de mis estudios de posgrado.

- A la Coordinación de Estudios de Posgrado (CEP), por el apoyo económico otorgado durante mis estudios de posgrado.

RESUMEN

En el trabajo se presenta el diseño de un controlador electrónico de desplazamiento aplicado a una plataforma móvil omnidireccional, con la característica de poder desplazarse a cualquier punto en dos dimensiones. Para ello se utilizan técnicas de inteligencia artificial (IA) y hardware programable.

El diseño del control de desplazamiento considera el movimiento en 8 direcciones y una configuración de 4 ruedas, con el propósito de aprender del comportamiento de la movilidad de la plataforma para poder aplicarlos a sistemas de navegación.

El algoritmo de control utiliza la Lógica difusa, esta técnica de inteligencia artificial permite diseñar un esquema de control sin conocer el modelo matemático del sistema (Planta); pero considerando para su diseño el conocimiento de un experto, y las pruebas físicas para adaptar el sistema a un ambiente real.

Para la implementación del diseño del sistema de control se utiliza un FPGA (arreglo de compuertas programables de campo), que permite la programación a nivel lógico de sistemas digitales y reutilizar código de sistemas con el soporte del lenguaje de programación de descripción de hardware (VHDL); otra ventaja de utilizar FPGA, es poder diseñar arquitecturas con un enfoque de paralelismo en la implementación de los sistemas, lo que proporciona características de flexibilidad y respuesta rápida para los dispositivos que los utilizan.

En este trabajo de investigación se desarrolla en el *Instituto de Investigaciones en Matemáticas Avanzadas y en Sistemas* (IIMAS).

ÍNDICE

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

ÍNDICE DE ECUACIONES

Capítulo 1

INTRODUCCIÓN

Los sistemas de navegación están conformados por unidades de localización y desplazamiento, el módulo de localización está encargado de calcular tanto la ubicación del móvil, como el punto de arribo y la trayectoria a seguir del vehículo para llegar a esa ubicación. La unidad de desplazamientos se encarga con los datos recibidos del módulo de localización, de la trayectoria a seguir con el accionar de los elementos necesarios para seguir esa ruta y por lo consiguiente llegar al punto de arribo designado.

Para el desplazamiento existen varias formas de ejecución que están ligadas con las plataformas móviles que el sistema contenga para seguir la ruta designada, que está dada por el módulo de localización del sistema de navegación.

El propósito de este proyecto es el diseño y desarrollo de un sistema de control de desplazamiento de una plataforma móvil utilizando hardware y dotarlo de autonomía para ser utilizado en conjunto con un sistema de navegación, para propósitos de aplicaciones en robots móviles

Para el desarrollo de este control de desplazamiento se utiliza una plataforma móvil omnidireccional por su flexibilidad de movimiento para seguir diferentes rutas y por su reciente utilización en áreas de investigación concernientes a sistemas robotizados.

Para el diseño e implementación del sistema, se utilizaron piezas mecánicas para el armado de la plataforma móvil ofrecida por una empresa especializada en sistemas robóticos (Andymark). Este conjunto de piezas contiene todo lo necesario para el ensamble de una plataforma móvil omnidireccional. El sistema de control para los motores utiliza un dispositivo manejado mediante modulación por ancho de pulso (PWM), facilitando la operación de los motores (Talon SR). En la medición de desplazamiento longitudinal se ocupa un codificador de cuadratura (codificador óptico modelo E4P). En la parte de hardware electrónico se utiliza una tarjeta de desarrollo del fabricante Xilinx, llamada Spartan-3E FPGA Starter Kit Board, en donde se implementa el sistema de control para el desplazamiento e implementado en el lenguaje de programación VHDL. Para la comunicación se utiliza un módulo de comunicación bluetooth (RN-24) con una aplicación desarrollado para celulares inteligentes con sistema operativo Android.

La tesis está organizada en 5 capítulos, el primer capítulo describe una introducción al tema de la tesis, sus objetivos y el alcance, así como, información del trabajo previo realizado por diferentes autores.

El capítulo 2 presenta información general de los dispositivos y algoritmos utilizados para la realización del proyecto, para proporcionar al lector los antecedentes del tema.

El capítulo 3 muestra el desarrollo del sistema, describiendo las partes que lo componen, el diseño de cada bloque que lo conforma y la implementación en FPGA y hardware externo.

En el capítulo 4, se describen las pruebas físicas realizadas y los resultados experimentales obtenidos.

En el capítulo 5 se muestran las conclusiones y trabajos futuros.

1.1. Planteamiento

El problema abordado es una propuesta de solución con respecto a sistemas de control de desplazamiento basados en métodos de inteligencia artificial con aplicaciones de navegación para plataformas móviles omnidireccionales.

Planteamiento del problema

El problema al que nos enfocamos es el desplazamiento, componente importante de un sistema de navegación en el área de plataformas móviles. En el diseño de un sistema de navegación, las partes que conforman la plataforma móvil que son utilizadas para el desarrollo de un sistema de desplazamiento capaz de seguir la ruta trazada por el resto de los componentes del sistema de navegación y que en conjunto conforman el accionamiento.

El problema general es el desplazamiento basado en el sistema de navegación completo, ya que se utilizan modelos altamente estudiados de plataformas móviles, y que tienen la cualidad de ser fácilmente controlados; El problema radica en como desplazar una plataforma móvil con una configuración mecánica que hace difícil el seguimiento de una ruta, este trabajo se orienta a obtener una solución en ese sentido.

Planteamiento de la solución

La propuesta de solución para este trabajo es el uso de FPGA (*Arreglo de compuertas programables de campo)* para la implementación del módulo de control de desplazamiento y con lógica difusa en una configuración de plataforma móvil recientemente utilizada denominada *plataforma móvil omnidireccional*.

1.2. Justificación

Desde el punto de vista mecánico, la mayoría de las aplicaciones que se realizan en esta área, se implementan basándose en una configuración diferencial, configuración sencilla de implementar tanto mecánicamente como en su control, derivado de su amplio estudio y pruebas técnicas de control (odometría, lógica difusa y neurodifusa entre otros) [1], [2], [3], [7], [8], [9], [10], [11] y [17].

Por otro lado, el estudio de las plataformas con configuración mecánica omnidireccional empieza a tener importancia para el diseño e implementación de algoritmos para su control, debido al comportamiento en cuanto a su movilidad. En este tipo de configuración se encuentra el problema del control de cuatro ruedas omnidireccionales de forma simultánea para efectuar los desplazamientos con precisión y repetitividad en el seguimiento de las rutas, y aprovechar las cualidades con las que cuenta este tipo de configuración mecánica con respecto a otras.

Para una configuración mecánica de una plataforma móvil omnidireccional y considerando su funcionamiento básico, se plantea un sistema que contenga las siguientes características:

- Manejo de paralelismo para control.
- Independencia de software.
- Manejo de control mediante usuario (otro sistema o dispositivo).
- Hardware de reciente utilización.

- Adaptable (no necesitar abundantes datos del sistema).
- Modular.

Cumpliendo con estas características y considerando los estudios actuales en el ramo sobre *plataformas móviles omnidireccionales* [14] [22], se propone un sistema implementado en un dispositivo de *Arreglo de compuertas programables de campo* (FPGA), que cuenta con las características de ser un dispositivo que se puede utilizar para implementación de sistemas de control que requieren de alta velocidad de respuesta y el manejo de concurrencia por ser un dispositivo con fundamento de programación concurrente, además de reciente utilización en varias áreas de la investigación y es modular y reprogramable.

En cuanto a las características de adaptabilidad se considera como propuesta un control difuso, el cual, se basa en la experiencia para su implementación y puede ser adaptable en el diseño y la integración de otras técnicas como Redes Neuronales Artificiales o Algoritmos Genéticos. En este caso, es importante aprovechar la velocidad de ejecución del FPGA considerando un sistema elemental de control difuso. El diseño e implementación de este algoritmo de control con este tipo de dispositivos y sus diferentes formas de integrarlos mediante programación son de reciente estudio.

Considerando el *manejo de control mediante usuario,* se propone utilizar comunicación mediante Bluetooth, tomando en cuenta las nuevas tendencias de utilización de dispositivos móviles para el control de diferentes elementos en distintas áreas en el desarrollo tecnológico.

1.3. Objetivos

Objetivo General

El objetivo es desarrollar un controlador de desplazamiento mediante FPGA utilizando técnicas de inteligencia artificial (Lógica difusa) aplicado a una plataforma móvil omnidireccional limitado a un número definido de posiciones y para demostrar la movilidad de este tipo de plataformas con respecto a otros modelos y las ventajas de uso de lógica difusa en hardware (FPGA) para el control de una plataforma móvil omnidireccional.

Objetivos Específicos

Los objetivos de este tema de investigación que se proponen son los siguientes:

- Estudio de controladores basados en FPGA.
- Estudio de controladores con lógica difusa.
- Diseño de "*Controlador para el desplazamiento de una plataforma móvil en aplicaciones de navegación utilizando FPGA"*.
- Elaboración de aplicaciones para celulares móviles.
- Comunicación *celulares móviles -FPGA* vía bluetooth.
- Implementación del diseño del controlador en FPGA.

Con base en estos puntos, se dio forma al proyecto de investigación con el propósito de cumplir con los objetivos y metas planteadas, al igual, de obtener resultados satisfactorios del mismo para utilidad de futuras investigaciones tanto dentro como fuera del área de interés de este trabajo.

1.4. Metodología

La metodología utilizada para alcanzar los objetivos tanto generales como específicos de este tema es la siguiente:

- Estudio sobre plataformas móviles.
- Estudio sobre métodos de desplazamiento de plataformas móviles.
- Estudio sobre controladores difusos.
- Estudio de controladores difusos implementados en FPGA.
- Estudio acerca de codificadores.
- Propuesta de un modelo para el "*Controlador para el desplazamiento de una plataforma móvil en aplicaciones de navegación utilizando FPGA"*.
- Elección del material para el diseño.
- Consideraciones e implementación del diseño.
- Caracterización de cada elemento a utilizar.
- Pruebas experimentales del diseño.
- Ajustes y correcciones.

Con esta metodología se diseñó el sistema de control de desplazamiento, que nos permite conocer nuevos elementos y técnicas a utilizar, al mismo tiempo de poder hacer una elección con base al conocimiento obtenido y las metas a alcanzar.

1.5. Estado del Arte

En el campo de robótica existen varias áreas, las cuales, se centran principalmente en la investigación y desarrollo de nuevos sistemas para el mejor manejo, mayor eficiencia y menor tamaño concerniente a los diferentes tipos de robots y sus aplicaciones en diversas áreas de conocimiento o implementación en las que se utilizan.

Una de las áreas de robótica emergentes son los robots de servicio, los cuales, son los encargados de realizar o ayudar en las tareas de la vida cotidiana del ser humano. Estos están presentes en varios sectores, por ejemplo: en educación, medicina, industria y prestaciones de servicios entre otras.

Una de las partes centrales de investigación en robótica de servicio y robótica en general es en sistemas de navegación, lo cual nos lleva a interpretar varios métodos matemáticos que representan posicionamiento y trayectorias. Uno de los métodos utilizados y que ampliamente son temas de investigación y desarrollo se basan en ecuaciones odométricas entre otros métodos como: Filtros de Kalman, chip ADN, C-espacio, Localización y mapeo simultáneo y lógica difusa [1], [2], [3], [9], [8], [10], [11], [13], [16], [17], [18], [19] y [20].

La odometría es un procedimiento de posicionamiento de un robot móvil, que utiliza la cinemática de propulsión y ecuaciones geométricas que surgen de la relación entre los componentes del sistema de propulsión y codificación rotativa en las ruedas, que al tener un sistema de posicionamiento por odometría preciso, se pueden reducir costos por tener menos actualizaciones de la posición. Su método

de calibración consiste en la medición directa de los parámetros del modelo cinemático del móvil [1] [9] [10].

El Filtro Extendido de Kalman (EKF) es un método que estima la posición del robot a partir de las ecuaciones de odometría y de medida de sensores. Mediantes una etapa de predicción, que tiene en cuenta la odometría y una etapa de corrección donde se utilizan las medidas de sensores y la distribución de errores [2], [8], [9], [13], [19] y [20].

Los chips ADN consisten en un gran número de cadenas de ADN fijadas ordenadamente sobre un sustrato sólido, formado por arreglos matriciales; dentro de un punto del chip, se encuentran implantadas miles de cadenas de nucleótidos de la misma secuencia; en cada punto hay secuencias diferentes, las cuales, se denominan cadena de pruebas. Un chip ADN emulado electrónicamente codifica la información de entrada como cadenas binarias de n bits correspondientes a las lecturas de un ADC (Convertidor Analógico-Digital) u otro sistema. Sus ventajas son el paralelismo y memoria asociativa; al ser asociado en un sistema electrónico reconfigurable (FPGA) constituye una alternativa eficiente al procesar datos con un elevado número de variables [17].

El C- espacio (Espacio de configuraciones) se considera como el conjunto de todas las posibles configuraciones en que puede posicionarse un robot. Cada punto del C-espacio es una tupla de una cierta dimensión, donde se especifican los valores de los parámetros correspondientes con los grados de libertad del robot. Una de las ventajas en este proceso es la planificación de un solo punto mientras una planificación de un espacio de trabajo planificas cada punto de movimiento; de lo que hace esta planificación de C-espacio una tarea sencilla realizar [3].

El método de Localización y mapeo simultaneo o SLAM (sus siglas en ingles), el cual, consiste en lograr que el robot móvil sea capaz de estimar su pose (posición y orientación), por lo tanto, el sistema debe de usar un sensor para hacer una estimación del mapa del ambiente y al mismo tiempo localizarse a sí mismo sobre este [16], [19] y [20].

La lógica difusa se considera una lógica multivaluada que proporciona un medio para enfrentarse a situaciones del mundo real, situaciones complejas y dinámicas que son fácilmente caracterizadas por palabras que por modelos matemáticos. Estos son sistemas basados en el conocimiento, el cual, a su vez están sintetizado en una base de reglas, que hace este; un sistema utilizable para la navegación u desplazamiento de un sistema móvil con base a designación de condiciones y desconocimiento de valores del sistema para alcanzar este fin [9], [11] y [18].

Otro de los métodos utilizados para el sistema de navegación es mediante Modelos ocultos de Markov (HMM). Es un proceso estocástico doble, en el que, uno de los proceso es observable. La secuencia es un modelo estadístico obtenido del análisis de un sistema que posee la propiedad de Markov en el cual; existe una variable aleatoria conocida y una desconocida que mediante la definición de estados y la definición del modelo inicial se puede obtener un sistema de navegación mediante este método [21].

Pasando del sistema de navegación hay que hacer referencia que, estos sistemas están ligados sus modelos al tipo de estructura que tiene la plataforma y el tipo de comportamiento con el cual, esta ópera y su clasificación dependiendo de sus puntos de apoyo y configuración mecánica para movimiento como las plataformas Khepera, Nomad 200, Magelan, B21R, DANI, Nexus omnidireccional, XL-TERABOT, P-METIN entre otros [6], [7], [9], [10], [11] y [19].

La plataforma con configuración diferencial, por lo general, cuenta con 2 motores para propulsión y con un punto de referencia, el cual, sea un soporto para la dirección, este puede ser conformado por una o 2 ruedas. Este armado puede ser fácil de aplicar pero al mismo tiempo puede ser difícil de manejar por la necesidad de controlar las ruedas de propulsión adecuadamente [10] y [14].

Otras de las configuraciones es llamada de Ackermann, está basada principalmente en el movimiento de los automóviles, en que la parte trasera esta combinada para darle propulsión al sistema mientras la parte delantera está dispuesta a darle dirección al mismo, aunque el inconveniente de este sistema es el giro sobre su mismo eje el cual no puede realizar [14].

Hablando de desarrollos en el área de estructuras de plataformas móviles las configuraciones mencionadas anteriormente tiene la deficiencia de no poder desplazarse libremente en cualquier posición, lo cual, una configuración omnidireccional puede realizar. Esta cuenta con ruedas, las cuales, están construidas con rodillos que se pueden desplazar libremente. Para utilizarlas en una plataforma de 3 puntos de apoyo las ruedas deben de tener rodillos de 90 grados mientras una plataforma de 4 puntos de apoyo debe de tener rodillos de una colocación de 45 grado, lo cual, le permite con diferentes tipo de accionamiento en cada rueda moverse en cada dirección, aunque el inconveniente que tiene este tipo de rueda es la dificulta de manejo en línea recta y su difícil implementación de sistema de desplazamiento [14].

En la parte de control existen varias técnicas, una de ellas es la de lógica difusa. Esta pretende manejar la imprecisión del razonamiento humano. La lógica difusa tiene la capacidad de expresar el conocimiento en una forma lingüística, lo cual permite que un sistema pueda ser descrito por simple reglas humanas a comparación de técnicas como controladores PID digitales. Esta técnica está siendo aplicada en FPGA para la reducción de carga computacional y por la velocidad de reacción del dispositivo [4], [5] y [14].

En el apartado de medición en plataforma móvil se tiene como elementos en ellos de sistemas de estereovisión y de ultrasonido al igual de sistemas infrarrojos, los cuales, son utilizados ampliamente en cada una de las metodologías de navegación antes mencionadas por la utilidad de cada uno de los campos donde fue utilizado, por ejemplo, la estereovisión está basada principalmente en el desarrollo de un mapa completo con base en imágenes tomadas y procesando estas para tomar un sistema de referencia, lo cual, ubique tanto así mismo como la trayectoria para su lugar de destino, mientras, las técnica de medición de distancia son utilizadas principalmente en métodos donde se toman referencia en función de mediciones para reducción de errores de desplazamiento en cada uno de los métodos de navegación [9], [10], [11], [12], [15], [19] y [20].

Capítulo 2

TEMAS RELACIONADOS EN EL SISTEMA DE CONTROL

En este capítulo se presentan los temas relacionados con el proyecto.

En este entorno se abordan temas relacionados al diseño del sistema, con la finalidad de resaltar los métodos, elementos o teorías utilizadas y dar un concepto general de su uso.

2.1. Plataforma móvil

En este apartado se habla sobre la parte mecánica del proyecto, esto es, la plataforma móvil, mejor conocida como robot móvil.

Esta parte trata la clasificación general de las plataformas móviles con base en su modo de tracción o por su disposición de las ruedas para la forma de tracción de forma muy general.

Se habla de forma extensa de la configuración omnidireccional, poniendo énfasis en sus cualidades tanto en tracción y dirección como en la disposición de las ruedas.

Clasificación de plataformas móviles

La clasificación de las plataformas móviles se deriva principalmente de dos criterios, los cuales son:

- Tracción y dirección.
- Disposición de ruedas.

La parte de tracción y dirección se clasifica los robots con base en los movimientos que estos puedan realizar y en como los realizan.

En la parte de disposición de las ruedas; la clasificación se realiza con base al tipo de rueda y su forma de ser colocadas en un robot para brindar movimiento.

En la parte de tracción y dirección se clasifica de la siguiente forma:

- Diferencial.
- Síncrono.
- Ejes independientes.
- Omnidireccional.

Esta clasificación aplica solamente a robots con locomoción mediante ruedas.

La configuración diferencial trata de tener 2 ruedas fijas con manejo independiente en cada lado del robot, aparte de otra rueda (loca o sueca), colocada dependiendo de la configuración de la disposición de las ruedas; esta forma necesita 3 puntos de apoyo como mínimo para su operación; este tipo de tracción y dirección es muy fácil de armar mecánicamente pero un tipo de control complicado a causa del manejo independiente de dos de sus ruedas [14] y [22]. Esta configuración se observa en la figura 2.1.

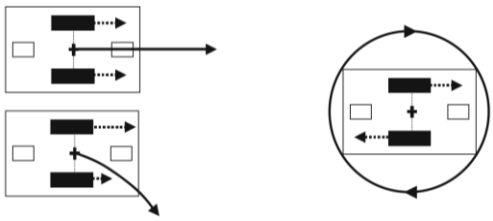


Figura 2.1. Tracción y rotación de configuración de manejo diferencial [22].

El manejo síncrono se basa en la dirección y desplazamiento del móvil mediante sus tres puntos de apoyo (ruedas), estos dependiendo de si se desplaza o direcciona; giran hacia un mismo sentido igual depende como es configurado, por ejemplo, si es un motor designado para dirección y otro para desplazamiento. Este tipo de manejo está limitado a 2 grados de libertad, aparte de no poder hacer las acciones de desplazamiento y dirección en el mismo instante [22]. En la figura 2.2 se muestra una representación gráfica de esta configuración.

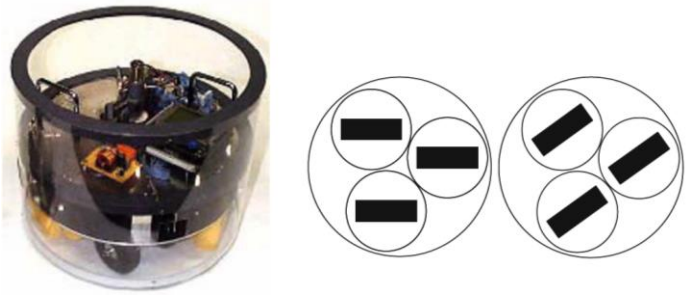


Figura 2.2. Manejo síncrono (Robot Xenia, Universidad de Kaiserslautern, con diagrama esquemático) [22].

Por último en esta clasificación que está realizada mediante tracción y dirección se menciona el manejo por ejes independientes o mejor conocido como “Direccionamiento Ackermann”. Esta forma de manejo se rige por la división de tarea de direccionamiento y desplazamiento en el dispositivo móvil con base en la cantidad de ruedas que tenga (Donde una parte estará dedicada para moverse y otra para la dirección), esta configuración es igual al utilizado por vehículos [14], [22] y [28]. Este tipo de manejo se puede apreciar de mejor manera en la figura 2.3.

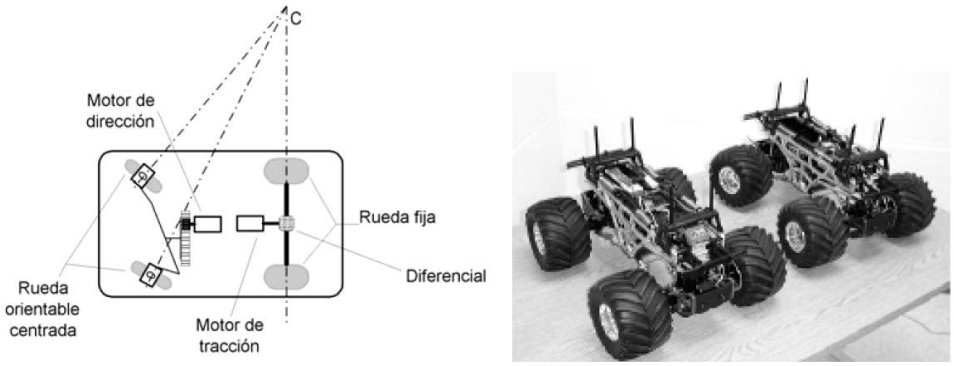


Figura 2.3. Sistema de manejo de ejes independientes [28].

Mientras, en la clasificación por disposición de las ruedas es la siguiente:

- Uniciclo.
- Triciclo.
- Cuatriciclo (Ackermann).
- Omnidireccional.

La clasificación por disposición de ruedas se basa en la estructura de colocación de ruedas en el medio físico y su cantidad o mejor conocido por algunos autores como puntos de apoyo, lo cual difiere en algunos aspectos a la clasificación hecha por tracción y dirección del móvil.

La configuración uniciclo se presenta como una configuración de 2 ruedas colocadas de forma lineal de un extremo a otro y controladas independientemente y una rueda loca, la cual ayuda a la estabilidad del móvil, esta configuración es homologa a la de manejo diferencial mostrada en la clasificación mediante dirección – tracción; la única diferencia es que esta configuración es forzosamente un manejo de 3 punto de apoyo, mientras, la diferencial puede ser utilizada en elementos con 4 puntos de apoyo si se cumplen estas condiciones [28]. La configuración uniciclo se observa en la figura 2.4.

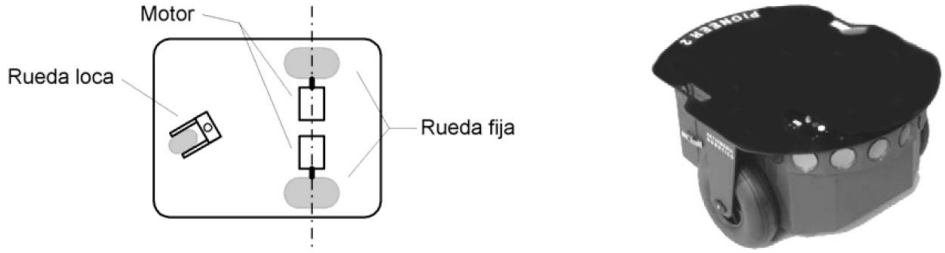


Figura 2.4. Sistema Uniciclo (Robot Pionner) [28].

En la configuración triciclo tiene ruedas fijas sobre un mismo eje y una rueda centrada orientable, que tiene la función de tracción – dirección, este tipo de móviles cuenta con tres puntos de apoyo [14] [28]. Esta configuración se ilustra en la figura 2.5.

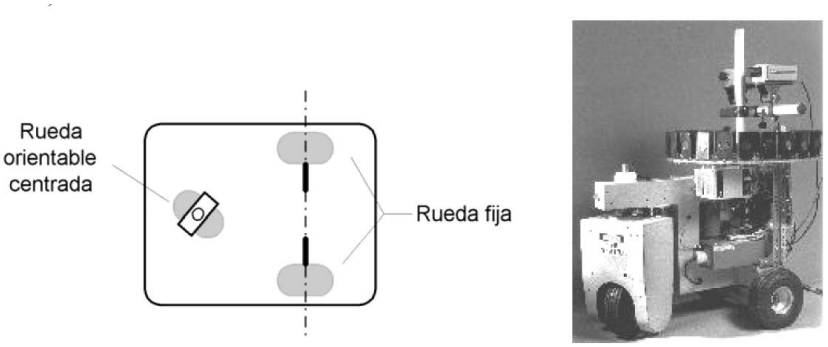


Figura 2.5. Sistema triciclo (Robot Neptune) [28].

La configuración cuatriciclo es el homólogo del manejo por “ejes independientes”, abordado anteriormente en la clasificación por tracción – dirección y corresponde con la imagen mostrada en la figura 2.3 [14], [22] y [28].

Tanto la configuración uniciclo como triciclo son muy utilizadas en el ámbito científico para pruebas de algoritmos de control por su facilidad mecánica que estas contiene y su cierta complicación que pueden tener a la hora de controla su comportamiento.

El sistema omnidireccional mencionado en las partes de clasificación, tanto en la parte de tracción – direccionamiento, como en la parte de disposición de ruedas, se trata en el apartado 2.1.2 llamado "Móvil omnidireccional", esto por ser una de las partes utilizadas para la elaboración de este trabajo y por lo tal dar información detallada del tema.

Móvil omnidireccional

La gran flexibilidad de movimiento de un robot omnidireccional está ligado a las características de las ruedas llamadas "*Mecanum Wheels*" (Ruedas Mecánicas), que esta configuración utiliza. Las "*Mecanum Wheels*" fueron desarrollas por *Swedish Company Mecanum AB* con Bengt Ilon en 1973 [22].

En la imagen 2.6 se muestra los diferentes diseños de las "*Mecanum Wheels*".

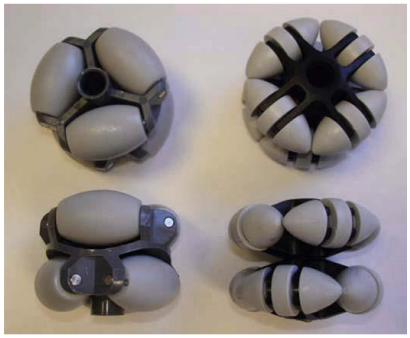


Figura 2.6. *Mecanum Wheels,* tipos de diseño con rodillos a 90º [22].

Como se observa en la figura 2.6, la superficie está cubierta por rodillos libres que contienen una forma cilíndrica, en estos diseños, existen 2 tipos de "*Mecanum Wheels*" que son los que contiene rodillos libres de 90º y los que contiene rodillos libre con un ángulo de 45º; las ruedas con rodillos de 45º tiene versiones de orientación izquierda y derecha, mientras lo de 90º no necesita tener ese tipo de versiones.

Un robot omnidireccional puede ser construido con 3 o 4 puntos de apoyo con manejo independiente, los vehículos con 3 puntos de apoyo requieren de ruedas con ángulos de 90º del eje de la rueda, mientras, los diseños basados en 4 puntos de apoyo requieren ruedas con rodillos que tengan una inclinación de 45º. Para la construcción de un robot con 4 "*Mecanum Wheels*" se requiere además un par de ruedas con orientación a la izquierda y otro hacia la derecha. Este tipo de configuraciones se pueden observar en la figura 2.7.

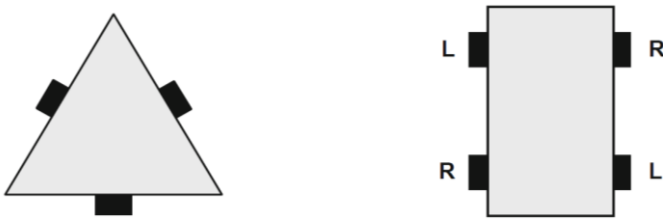


Figura 2.7. Vehículos omnidireccionales de 3 y 4 ruedas [22].

En la figura 2.8 se puede ver el principio mecánico de las "*Mecanum Wheels*" y su descomposición en vectores.

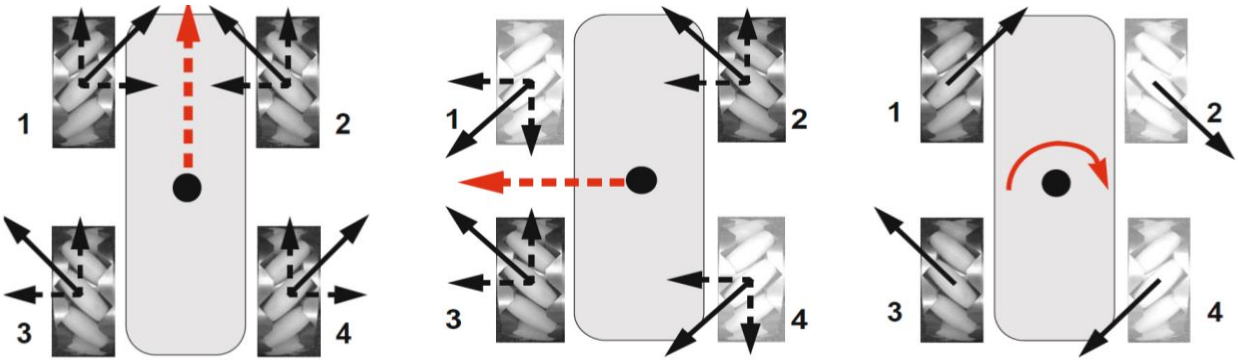
Rueda izquierda
Vista desde abajo

Rueda derecha
Vista desde abajo

Figura 2.8. Descomposición de vectores de rueda omnidireccional [22].

En la parte de manejo de un robot omnidireccional, los movimientos en 4 puntos de apoyo son en específico con base en anulación de vectores de trayectoria y la fuerza perpendicular ejercida para la generación de movimiento en cada dirección de un robot omnidireccional.

Algunos ejemplos de esta anulación de vectores de trayectorias y adicción de fuerzas perpendiculares se manifiestan en las siguientes imágenes, donde en la figura 2.9 se muestra el dibujo de los vectores de cada rueda para manejar hacia adelante, mientras en la figura 2.10 muestra el movimiento hacia la izquierda de la plataforma móvil con base al movimiento de las ruedas y sus vectores de desplazamiento. En la figura 2.11 se muestra la imagen para la rotación en su mismo eje de un robot móvil direccional y la dirección de los vectores que toman con base al movimiento de las 4 ruedas.



A) Enfrente B) Lateral C) Rotación

Figura 2.9. Movimientos básicos de un móvil omnidireccional [22].

En el capítulo 3 se muestra de forma específica el movimiento y de cada rueda para el diseño e implementación de varias partes del sistema de control.

La cinemática directa de este tipo de configuraciones mecánicas de robots móviles se representa en la ecuación 2.1, la cual, incluye las velocidades del vehículo en forma lineal, perpendicular y de rotación con base en las velocidades individuales de cada rueda mecánica y la distancia de cada rueda en la plataforma.

$$\begin{bmatrix} V_X \\ V_Y \\ \omega \end{bmatrix} = 2\pi r \begin{bmatrix} \frac{1}{4} & \frac{1}{4} & \frac{1}{4} & \frac{1}{4} \\ -\frac{1}{4} & \frac{1}{4} & \frac{1}{4} & -\frac{1}{4} \\ -\frac{1}{2(d+e)} & \frac{1}{2(d+e)} & -\frac{1}{2(d+e)} & \frac{1}{2(d+e)} \end{bmatrix} \cdot \begin{bmatrix} \dot{\theta}_{FL} \\ \dot{\theta}_{FR} \\ \dot{\theta}_{BL} \\ \dot{\theta}_{BR} \end{bmatrix} \quad (2.1)$$

Donde:

$\dot{\theta}_{FL},\ \dot{\theta}_{FR},\ \dot{\theta}_{BL}, y\ \dot{\theta}_{BR}$	4 velocidades individuales de las ruedas en revoluciones por segundo,
r	radio de la rueda,
d	distancia entre los pares derecha e izquierda de las ruedas,
e	distancia entre la parte frontal y trasera del par de ruedas,
V_X	velocidad del vehículo en dirección delantera,
V_Y	velocidad del vehículo en dirección lateral,
ω	velocidad de rotación del vehículo.

En la cinemática inversa es una formula matricial, en la cual, se obtiene las velocidades individuales de las ruedas con base al proporcionar la velocidades delantera, lateral y rotacional ($V_X, V_Y\ y\ \omega$) y puede ser encontrada con base en la inversión de la formula matricial de la cinemática directa [22], se representa en la ecuación 2.2.

$$\begin{bmatrix} \dot{\theta}_{FL} \\ \dot{\theta}_{FR} \\ \dot{\theta}_{BL} \\ \dot{\theta}_{BR} \end{bmatrix} = \frac{1}{2\pi r} \begin{bmatrix} 1 & -1 & -(d+e)/2 \\ 1 & 1 & (d+e)/2 \\ 1 & 1 & -(d+e)/2 \\ 1 & -1 & (d+e)/2 \end{bmatrix} \cdot \begin{bmatrix} V_X \\ V_Y \\ \omega \end{bmatrix} \quad (2.2)$$

2.2. Lógica difusa

La idea de la logia difusa es similar a los sentimientos del ser humano y procesos de inferencia. A diferencia de una estrategia de control, que es un control de punto a punto, el control difuso es un rango puntos o una gama de rangos. La salida de un control difuso es dada desde la fuzzificación de entradas y salidas usando la relación de las funciones de membresía [40] y [41].

La idea de lógica difusa fue inventada por el profesor Lotfi A. Zadeh de la Universidad de California en Berkeley en 1965 [39] y [40]. Esta teoría no era muy reconocida hasta que el Dr. Ebrahim H. Mamdani, que es un profesor de la Universidad de Londres, donde aplico la lógica difusa para el control de una máquina de vapor en 1974 [39] y [40].

Para la implementación de lógica difusa para una aplicación real se requiere de los siguientes tres pasos [41]:

- Fuzzificación: convierte la entrada de valores numéricos a datos difusos o funciones de membresía.
- Proceso de inferencia difusa: combina las funciones de membresía con el control de reglas para generar una salida difusa.
- Defuzzificación: usa diferentes métodos para convertir la salida difusa en una salida numérica.

En resume, un proceso difuso es un proceso numérico- difuso- numérico para un sistema real.

Conjuntos difusos

Para definir a un conjunto difuso se debe considerar primero que es un conjunto. Un conjunto se puede definir como la agrupación de objetos por ciertas características o cualidades, por ejemplo, un médico para dar un diagnostico si un paciente si tiene "fiebre alta" o "sin fiebre alta" los clasifica con base en la temperatura corporal del paciente, esto se puede demostrar en la figura 2.22 [14].

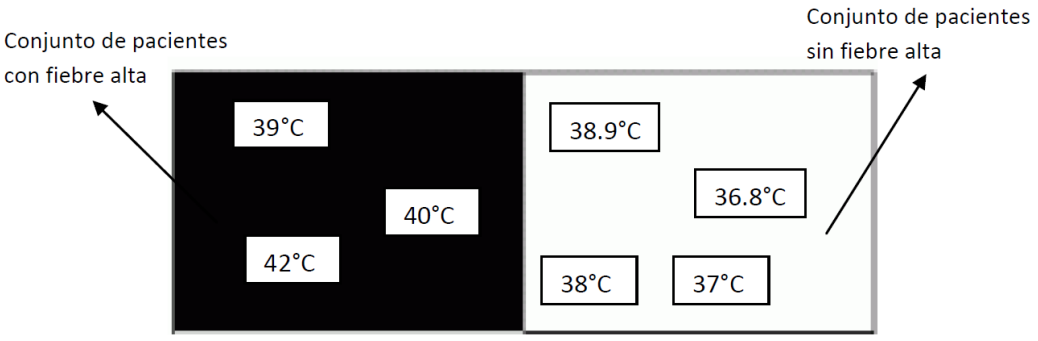


Figura 2.10. Conjunto de "fiebre alta" y "sin fiebre alta" del ejemplo del diagnóstico de los pacientes [14].

Como se observa en la figura 2.10, se considera con fiebre alta a pacientes con una temperatura igual o mayor a 39ºC y menor a esta se consideran sin fiebre. En la parte de teoría de conjuntos difusos esta clasificación cambia en el caso puede tener un grado de pertenecía entre los dos conjuntos un objeto; en el caso del diagnóstico de los pacientes con "fiebre alta" y "sin fiebre alta", aplicando la teoría de conjuntos difusos, las temperaturas corporales puedan tener un grado de pertenecía al conjunto de "fiebre alta" y otro grado de pertenecía al conjunto de "sin fiebre alta", de forma muy diferente a la teoría de conjuntos.

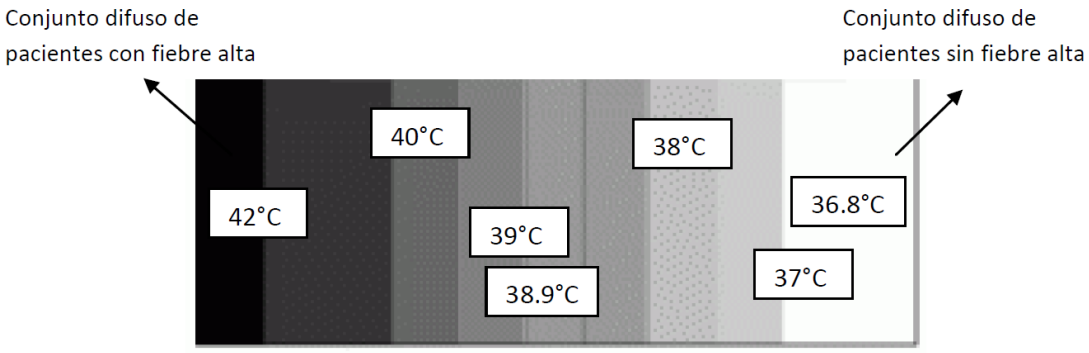


Figura 2.11. Conjunto difuso de "fiebre alta" y "sin fiebre alta" del ejemplo del diagnóstico de los pacientes [14].

Como se observa en la figura 2.11, a la temperatura corporal para el diagnóstico que si el paciente tiene "fiebre alta" o "sin fiebre alta" se le da una cantidad de valide entre estos dos conjuntos, esto con la finalidad de obtener la mayor información posible con una representación lingüística de expresión humana.

Algunos conceptos básicos asociados a los conjuntos difusos son [14]:

- **Universo de discurso**: es el intervalo de todos los valores aplicables a una variable del sistema.

- **Entrada real**: son entradas proveniente del mundo exterior al sistema difuso.
- **Función de membresía**: función matemática que nos indica el grado de pertenecía de las variables de entrada y salida.
- **Grado de membresía**: es el grado de competitividad de una entrada con una función de membresía en un intervalo de 0 a 1, donde el cero es de no pertenencia y el 1 es de pertenencia total, los valores entre estos 2 valores son una pertenecía parcial.
- **Etiqueta**: es el nombre descriptivo dado a la función de membresía.
- **Dominio**: es el intervalo de valores donde se ubica el ancho de una función de membresía.

En la figura 2.12 se puede ver de manera gráfica los conceptos asociados a un conjunto difuso.

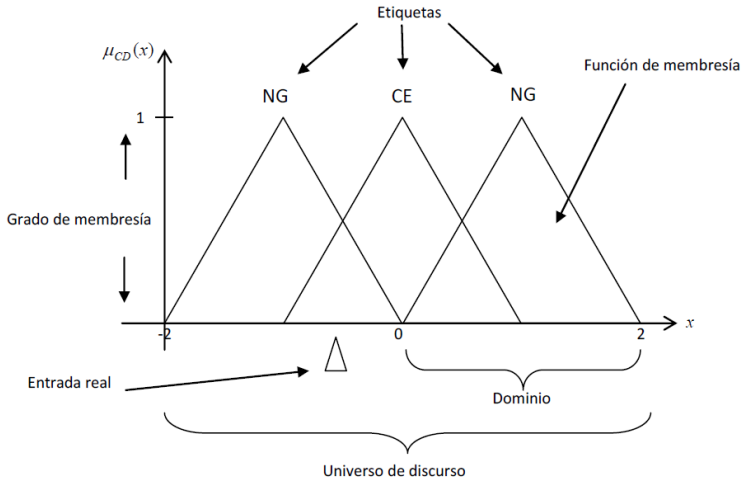


Figura 2.12. Representación gráfica de los conceptos asociados a los conjuntos difusos [14].

Tipos de función de membresía

En lógica difusa existen diferentes tipos de funciones de membresías, de las cuales, las más importantes son definidas a continuación [38]:

- **Triangular:** esta función es definida por tres parámetros (a, b y c), los cuales determinan los limites superior e inferior (a y c) y el centro de la figura (m) que es representada matemáticamente de la siguiente forma:

$$Triangulo\ (x, a, m, c) = Max\ \left[Min\ \left(\frac{x-a}{m-a}, \frac{c-x}{c-m}\right), 0\right] \quad (2.3)$$

- **Trapezoidal:** está definida por cuatro parámetros (a, b, c, d), que son el límite inferior y superior (a y d) y los límites de sus soportes inferior y superior (b y c) y es determinado por la siguiente expresión:

$$Trapezoidal\ (x, a, b, c, d) = Max\ \left[Min\ \left(\frac{x-a}{b-a}, 1, \frac{d-x}{d-c}\right), 0\right] \quad (2.4)$$

- **Función S**: está definida por sus límites inferior y superior (a, b) y el valor m o punto de inflexión, tal que a < m < b , que tiene la siguiente representación matemática:

$$S\,(x,a,b,m)\begin{cases} 0 & para \quad x \le a \\ 2\left(\frac{x-a}{b-a}\right)^2 & para \quad a \le x \le m \\ 1-2\left(\frac{x-b}{b-a}\right)^2 & para \quad m \le x \le b \\ 1 & para \quad x > b \end{cases} \tag{2.5}$$

El crecimiento es más lento cuando mayor sea la distancia a-b.

- **Singleton**: esta función de membresía se caracteriza por tener el valor donde se encuentre en el universo de discurso.

A continuación se presenta la forma gráfica de estas funciones de membresía.

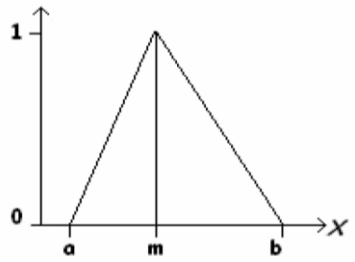


Figura 2.13. Función de membresía triangular.

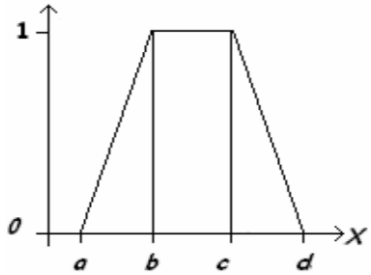


Figura 2.14. Función de membresía trapezoidal.

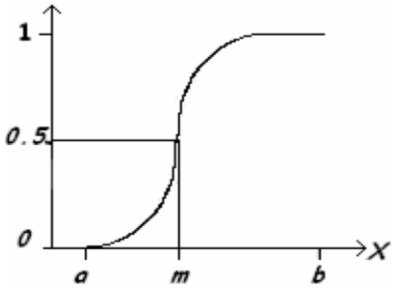


Figura 2.15. Función de membresía S.

Variable lingüística

Una variable lingüística es una variable que toma como valor palabras del lenguaje natural, donde las palabras están caracterizadas por conjuntos difusos definidos en un universo de discurso en el cual la variable está definida [36].

Para el diseño de la variable difusa se sigue el siguiente esquema:

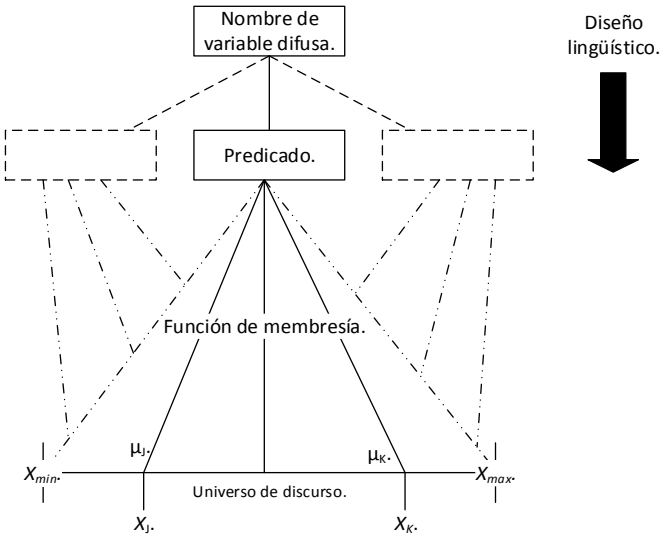


Figura 2.16. Esquema de diseño de la variable lingüística.

Para el diseño de una variable lingüística se debe de tomar en cuenta tanto la selección de palabras, que se utilizaran tanto para identificar la variable lingüística como los predicados que estará puesto en las etiquetas de los conjuntos difusos. Las funciones de membresía que tendrán los conjuntos que formen a la variable y el universo de discurso de la variable lingüística.

Operaciones de conjuntos difusos

Se definen dos conjuntos difusos $\underset{\sim}{A}$ y $\underset{\sim}{B}$ en un universo de discurso *X*. Para obtener un elemento *x* del universo de discurso, las siguientes operaciones de la teoría de funciones en teoría de conjuntos difusos de unión, intersección y complemento se definen para $\underset{\sim}{A}$ y $\underset{\sim}{B}$ en X [49].

En la operación de unión calcula el grado de membresía de la unión de los conjuntos $\underset{\sim}{A}$ y $\underset{\sim}{B}$, esto es representado matemáticamente de la siguiente manera:

$$\mu_{\underset{\sim}{A}\cup\underset{\sim}{B}}(x) = Max(\mu_{\underset{\sim}{A}}(x), \mu_{\underset{\sim}{B}}(x)) \tag{2.6}$$

La operación de intersección calcula el grado de membresía de la intersección de los conjuntos $\underset{\sim}{A}$ y $\underset{\sim}{B}$. Lo que lo vuelve lo contrario al operador de unión.

$$\mu_{\underset{\sim}{A}\cap\underset{\sim}{B}}(x) = Min(\mu_{\underset{\sim}{A}}(x), \mu_{\underset{\sim}{B}}(x)) \tag{2.7}$$

Por último, en el operador de complemento, en esta operación se obtiene el inverso del conjunto difuso operado.

$$\mu_{\underset{\sim}{\bar{A}}}(x) = 1 - \mu_{\underset{\sim}{A}}(x) \tag{2.8}$$

En las siguientes figuras se observa la descripción grafica de cada uno de los operadores difusos.

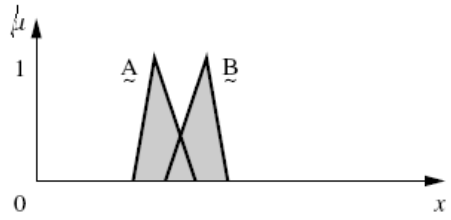


Figura 2.17. Operador de Unión entre los conjuntos difusos $\underset{\sim}{A}\ y\ \underset{\sim}{B}$ [49].

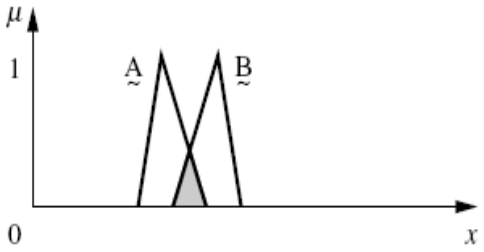


Figura 2.18. Operador de Intersección entre los conjuntos difusos $\underset{\sim}{A}\ y\ \underset{\sim}{B}$ [49].

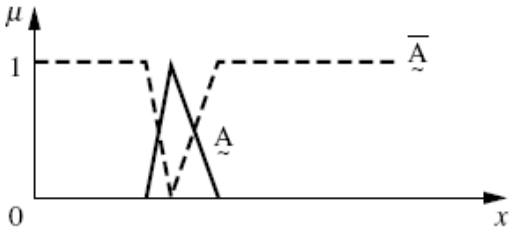


Figura 2.19. Operador de complemento del conjunto difuso $\underset{\sim}{A}$ [49].

Tabla de decisión

En el diseño de un sistema difuso se cuenta tanto como variables de salida como variables de entrada, al igual de reglas para toma de una decisión, existen en este caso muchas forma de formar esas reglas, una de ellas es por tabla de decisión.

La tabla de decisión es un herramienta en la que se da un conjunto de consecuentes y un conjunto de acciones, en donde, el valor que tenga la condición se ejecute una acción, en este caso tendrá un conjunto de condiciones con base al valor de las variables lingüísticas de entrada y tendrá como conjunto de acciones los valores de la variable lingüísticas de salida.

Con base en esta tabla se forma la base de reglas del sistema difuso.

Defuzzificación

Este proceso se define como la parte en donde convertimos una variable lingüística de salida a un valor numérico.

Para el proceso de defuzzificación existen varios métodos para implementar de los cuales mencionaremos los más utilizados a continuación [37] [40]:

- **Método de media de los máximos (MOM)**: genera una acción de control que representa el valor medio de todas las acciones de control, cuyos valores de pertenencia alcancen el máximo, este es expresado como:

$$Z_0 = \sum_{j=1}^{k} \frac{Z_j}{k} \tag{2.9}$$

Donde:

$Z_j : acciones\ de\ control\ cuya\ funciones\ de\ membrecia\ llegan\ al\ maximo.$

$k = numero\ de\ acciones\ de\ control.$

- **Método centro del área (COA)**: esta estrategia genera el centro de gravedad de la distribución de posibilidad de un conjunto difuso C, esto se representa de la siguiente forma:

$$Z_0 = \frac{\sum_{j=1}^{n} \mu_C(Z_j) * Z_j}{\sum_{j=1}^{n} \mu_C(Z_j)} \tag{2.10}$$

Funcionamiento de un sistema difuso

Los pasos para el funcionamiento de un sistema difuso se describen a continuación:

- **Entrada numérica:** en este paso se adquieren las entradas de una medición u cualquier otro tipo dato que necesite el sistema difuso.
- **Fuzzificación:** explicado anteriormente este proceso tiene como finalidad convertir los datos numéricos recibidos a grados de membresía de una variable lingüística, con el fin de utilizarlo en el sistema.
- **Selección de reglas:** en esta parte, con base en los conjuntos difusos con sus grados de membresía de las variables lingüísticas del sistema, se realiza una selección de estas reglas para la toma de decisión.
- **Razonamiento difuso:** con base en las reglas activadas que se obtiene de la selección de reglas se aplica una de las operaciones difusas para determinar el valor de salida que tendrá el sistema difuso, en este caso es con la o las variables de salidas lingüísticas que posea el sistema.
- **Salida difusa:** en esta caso es la verificación de que conjunto de salidas tiene el sistema, en forma lingüística.
- **Defuzzificación:** en esta parte del funcionamiento del sistema difuso se aplica un método numérico para convertir el valor de las membresías de los conjuntos difusos de las variables de salida a un valor numérico.
- Salida numérica: en esta parte del sistema difuso se da como resultado del proceso del sistema difuso una salida numérica para su utilización.

Los objetivos que se obtiene del funcionamiento del sistema difuso y su aplicación en el control de diferentes sistemas se clasifican a continuación [39]:

1. Mejorar la robustez que se obtienen con los métodos clásicos de control lineal.

2. Diseño de control simplificado para modelos complejos.
3. Implementación simplificada.
4. Autonomía
5. Adaptabilidad.
6. En control difuso, no es necesario un modelo matemático de la planta.

2.3. FPGA

Arquitectura

El FPGA (field programmable gate array) es un dispositivo lógico que contiene 2 dimensiones de arreglos de celdas lógicas genéricas e interruptores programables [26].

La arquitectura de un FPGA consiste de CLBs (bloques lógicos configurables), bloques I/O configurables y programación de interconexión para la designación de las rutas de las señales entre los CLBs y los bloques de I/O [25] [27]. Aparte de circuito de reloj entre otros elementos.

En la figura 2.20 se observa de forma gráfica la estructura interna del FPGA.

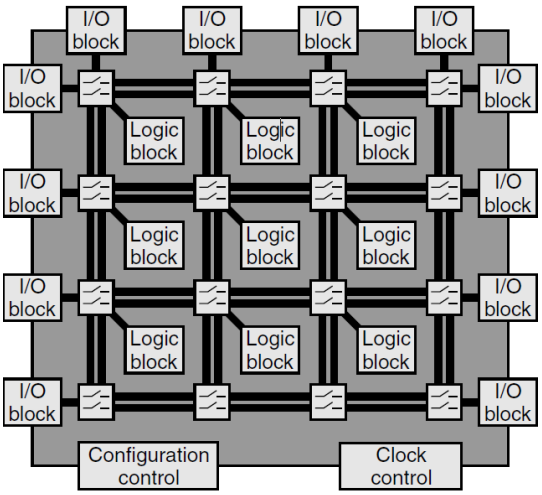


Figura 2.20. Arquitectura interna de un FPGA [24].

Los CLBs contienen la lógica programable para el FPGA. El diagrama mostrado en la figura 2.21 se puede observar la estructura de un CLB, el cual contiene, una memoria RAM para la creación de combinaciones lógicas, flip-flops para el almacenamiento y multiplexores.

Los CLB tienen en su arquitectura 2 memorias de 4 entradas, a estas memorias se les conoce como LUTs (Look-Up Tables).

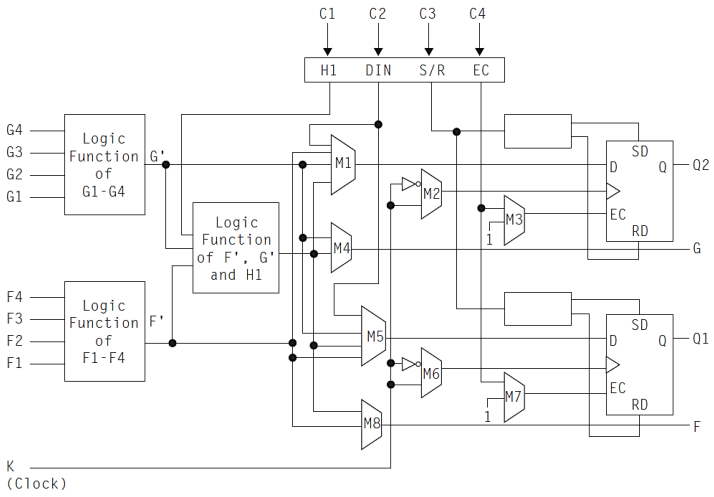


Figura 2.21. Arquitectura interna de un CLB [25].

Otras de las partes en la arquitectura de un FPGA es el bloque de configuración de I/O, son usados para las interacciones de las señales utilizadas con el chip, en la figura 2.22, se observa el esquema que compone a esta parte, en donde la el buffer de entrada tiene la capacidad de configuración del buffer en tri-estado o colector abierto y el control de velocidad de activación. Estos controles permiten al FPGA tener como salidas voltajes TTL o CMOS, mientras, el buffer de entrada puede ser programado para diferentes voltajes de umbral, tales como TTL o CMOS, para la interconexión de dispositivos TTL o CMOS [25].

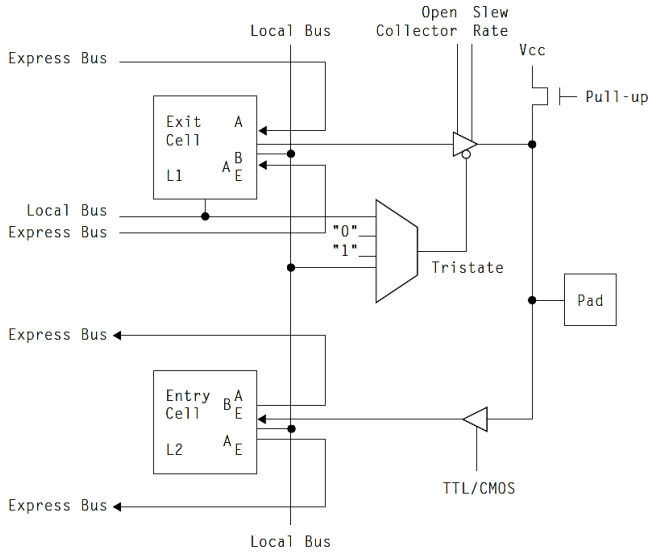


Figura 2.22. Arquitectura interna de un bloque de configuración de I/O [25].

Para el sistema de interconexión existen 3 tipos diferentes de enrutamiento, el primero es una interconexión entre CLBs cercanos, este lo que tiene como consecuencia es la creación de lógica compleja para que pueda ser contenida en el chip. El ruteo por conexión de CLBs por matrices de interconexión, trata de la interconexión de un numero predefinido de CLBs mediante estos bloques; la desventaja de este método es el tiempo de ejecución que puede tener si los CLBs quedan distantes entre ellos. En el tipo

de ruteo por “línea larga” está dado en se pueden hacer interconexiones de CLBs lejanos sin tener muchos retardos, esta “línea larga” esta puesta de extremo a extremo del arreglo y no pasa por las matrices de interrupción [25].

Por último, en la parte de programación, existen dos métodos para programar FPGAS, el primero es mediante SRAM, el cual se implementa mediante bits estáticos RAM para la programación, estos pueden ser combinados en una memoria y usados con un LTU para la implementación de la lógica combinacional y usarse de forma individual para controlar por separado diferentes tipos de elementos en el arreglo. El segundo tipo de programación se da por medio de “Anti fusibles”, que consiste en dos conductores separados por un materia aislante; cuando es aplicado un voltaje alto, el materia aislante se derrite, lo cual permite la interconexión de los dos conductores [25].

VHDL (VHSIC Hardware Description Language)

Very high-speed integrated circuit hardware description language (VHSIC HDL o VHDL) se desarrolló en 1980, para el Departamento de Defensa de los Estados Unidos, con el propósito de desarrollar diseños digitales con una metodología común, proveyendo de la habilidad de auto-documentado y reutilización con tecnología nueva. En 1983 inicio el desarrollo del lenguaje VHDL [23].

Finalmente, en el año 1987, el lenguaje VHDL se convierte en la norma IEEE-1076. Como todas las normas IEEE, se somete a revisión periódica, por lo que en 1993, 2000 y 2002[23].

Las características con las que cuenta este lenguaje de programación se describen a continuación:

- Jerarquía.
- Soporte para la utilización de bibliotecas de diseño.
- Diseño genérico.
- Concurrencia.
- Estilos de descripción.
    - Estructural (“Structural”).
    - Comportamiento (“Behavioral”).
    - Flujo de datos (“Dataflow”).
- Soporte para simulación, modelado y síntesis.
- VHDL sintetizable es un subconjunto del VHDL simulable.

La descripción estructural se basa en describir la estructura del circuito en términos de compuertas lógica, esto, utilizando la interconexión entre las compuertas lógicas como una lista de conexiones de circuitos (netlist) [23]. Esta descripción se basa principalmente en la utilización de procesos y de declaraciones secuenciales, las cuales, le permiten modelar la función con rapidez [29].

La descripción por comportamiento describe el comportamiento del diseño en términos del circuito y el funcionamiento del sistema mediante algoritmos. Esta descripción a alto nivel usa construcción de lenguaje que se asemeja en cierta manera a lenguaje de alto nivel en programación de software [23].

La descripción por flujo de datos describe la trasferencia de datos desde entrada a salida y entre señales [23]. Sin la necesidad de una declaración secuencia, este puede ser utilizado mediante 2 formatos los cuales son: mediante instrucciones "*when-else"* o por medio de ecuaciones booleanas [29].

La estructura de un programa en VHDL esta forma por módulos o unidades de diseño, de estos, existen 5 unidades de diseño en VHDL: declaración de entidad, arquitectura, configuraciones, declaración de paquetes y cuerpo del programa [29] [30]. En el desarrollo de programas en VHDL son indispensables los bloques de entidad y arquitectura, que son indispensables para la estructura de un programa [29].

Algunas ventajas del uso de VHDL para descripción de hardware son [30]:

- VHDL permite diseñar, modelar y comprobar un sistema desde un alto nivel de abstracción hasta un nivel de definición estructura de compuertas.
- Circuitos descritos mediante VHDL, pueden ser utilizados para crear e implementar circuitos.
- Los modelos creados en VHDL pueden utilizarse en diferentes diseños, permitiendo la reutilización de código. Además, la misma descripción puede utilizarse para diferentes tecnologías.
- VHDL permite diseño tipo Top-Down, es decir, describir bloques en alto nivel, analizarlos y refinar la funcionalidad en alto nivel antes de llevarlo a un nivel bajo de abstracción para su implementación.

2.4. Codificador óptico incremental.

El primer codificador óptico fue desarrollado en mediado de los 40s por la Baldwin Piano Company. Esta división tiene 2 tipos de dispositivos de codificadores: incremental y absoluto [33] [34].

El codificador óptico es un dispositivo formado por un elemento lineal o un disco, que contiene 2 zonas iguales y distribuidas en su superficie, lo cual al incidir un haz de luz que puede producir cierto número de pulsos por cada eje de revolución [33] [34].

El codificador óptico pueden estar basados en sectores opacos y transparentes, en sectores reflectores y no reflectores, o en franjas de inferencia, en cualquier caso, la lectura fija del cabezal hay siempre una fuente de luz, normalmente un led infrarrojo, y un foto-detector [34] [22].

Cuando es emplea sectores opacos y transparentes, el emisor y el detector deben situarse uno a cada lado del elemento móvil. En cambio, cuando se emplea sectores reflectores y no reflectores, el emisor y el detector deben de estar en el mismo lado [34].

En la clasificación de los codificadores ópticos se encuentra los incrementales y absolutos que se muestra en la figura 2.23.

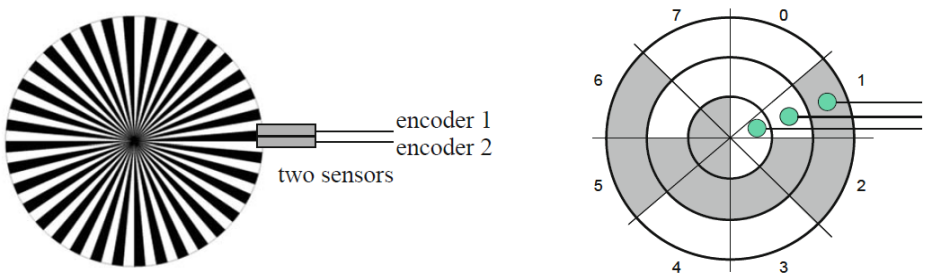


Figura 2.23. Codificador ópticos: incremental y absoluto [22].

En la figura 2.24 se muestra la estructura básica de un codificador incremental.

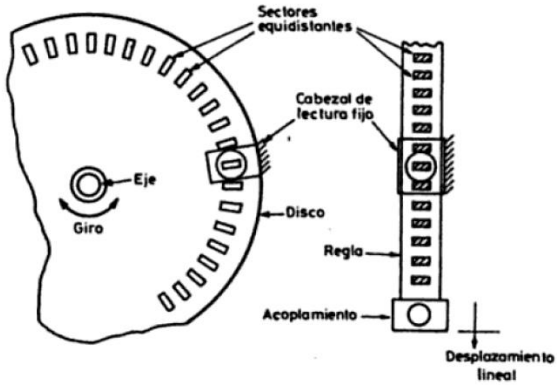


Figura 2.24. Estructura de un codificador incremental (para posición lineal y angular) [34].

Los codificadores incrementales se clasifican a su vez en canal simple y en cuadratura de fase.

El problema con los codificadores de canal simple en la medición es que no son capaces de determinar la dirección de rotación, lo cual no son usados como sensores de posición. El codificador en cuadratura de fase resuelve este problema al integrar un canal de salida más con un desfase de 90 grados, esto al saber que canal se activa primero se puede conocer el sentido de giro, su estructura y funcionamiento se observa en la figura 2.25 [33].

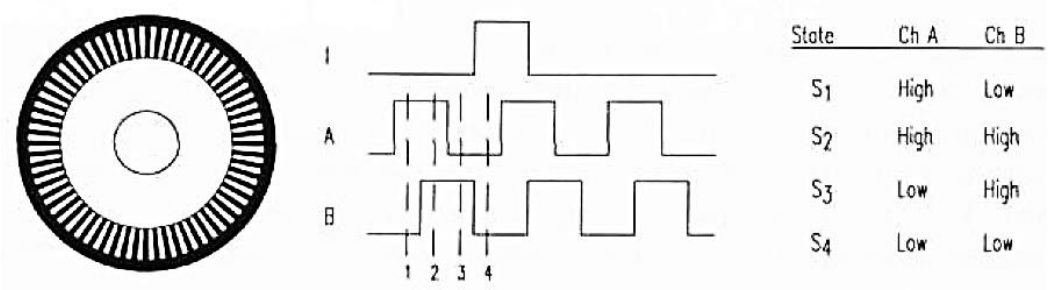


Figura 2.25. Funcionamiento de un codificador incremental de cuadratura (para posición lineal y angular) [33].

Capítulo 3

DESARROLLO DEL SISTEMA DE CONTROL

En este capítulo se plantea y muestra el proceso de diseño e implementación del controlador de desplazamiento, con base en la estructura general del sistema, de la que se propone una estructura para el diseño del controlador y con la finalidad de mostrar el alcance del proyecto en general y dar razones del diseño propuesto.

Después de presentar la estructura para el diseño del controlador, se menciona el material relacionado y utilizado para el diseño e implementación del controlador.

3.1. Estructura del sistema

Para el desarrollo del proyecto de tesis, como primer paso se desarrolló un diagrama general, el cual nos sirve de referencia para saber la magnitud del proyecto que queremos llevar a cabo y delimitarlo adecuadamente con respecto a tiempos y objetivos a obtener.

Este diagrama es el que se presenta a continuación:

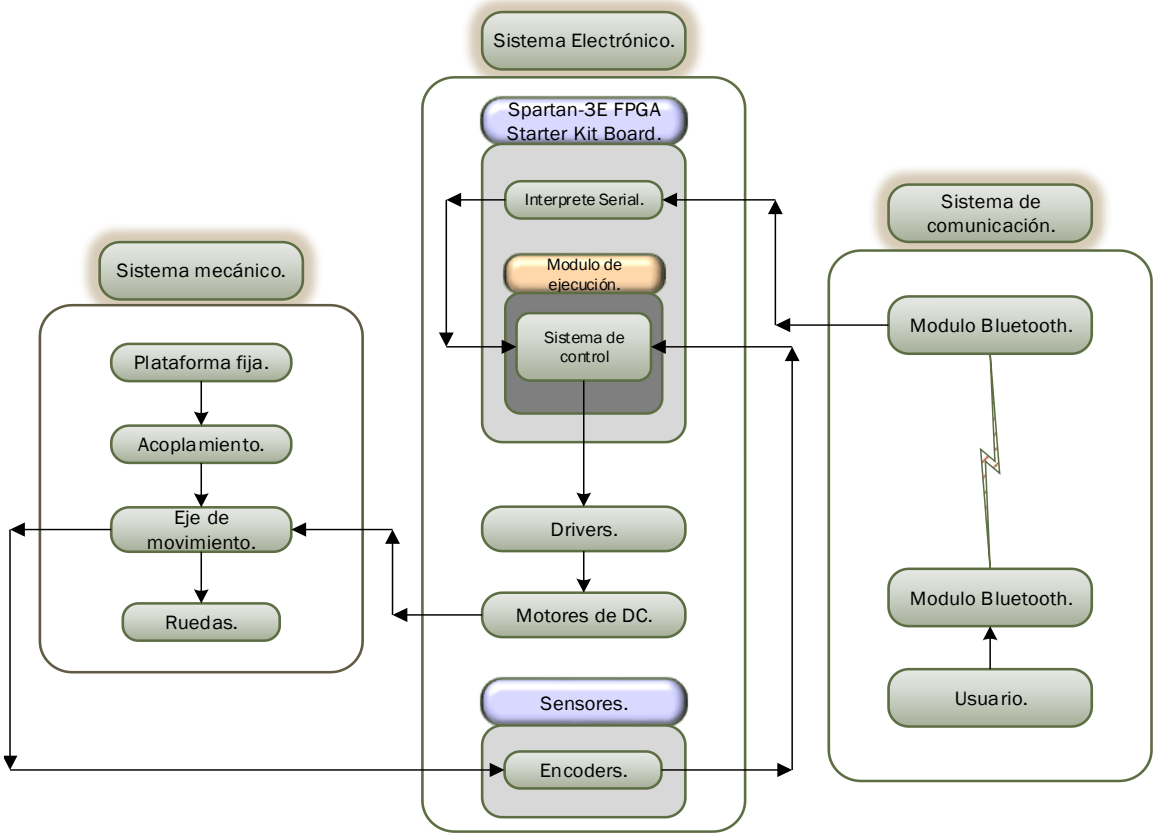


Figura 3.1. Sistema general del dispositivo móvil omnidireccional.

En la Figura 3.1 se observa la división que se propone con respecto a la actividad de cada uno de los elementos en el sistema, los cuales son:

- Sistema mecánico.
- Sistema electrónico.
- Sistema de comunicación.

Este trabajo es basado en el sistema electrónico; por el desarrollo de un sistema electrónico dedicado al control de desplazamiento.

3.2. Estructura de diseño para el "*Controlador para el desplazamiento de una plataforma móvil en aplicaciones de navegación utilizando FPGA*"

Con base en estudio sobre plataformas móviles omnidireccionales, se toma en cuenta los elementos que contiene y con respecto a ello se elige uno para trabajar, el cual, presentamos un esquema que representa de forma global lo que se tiene en mente realizar y como se estructura.

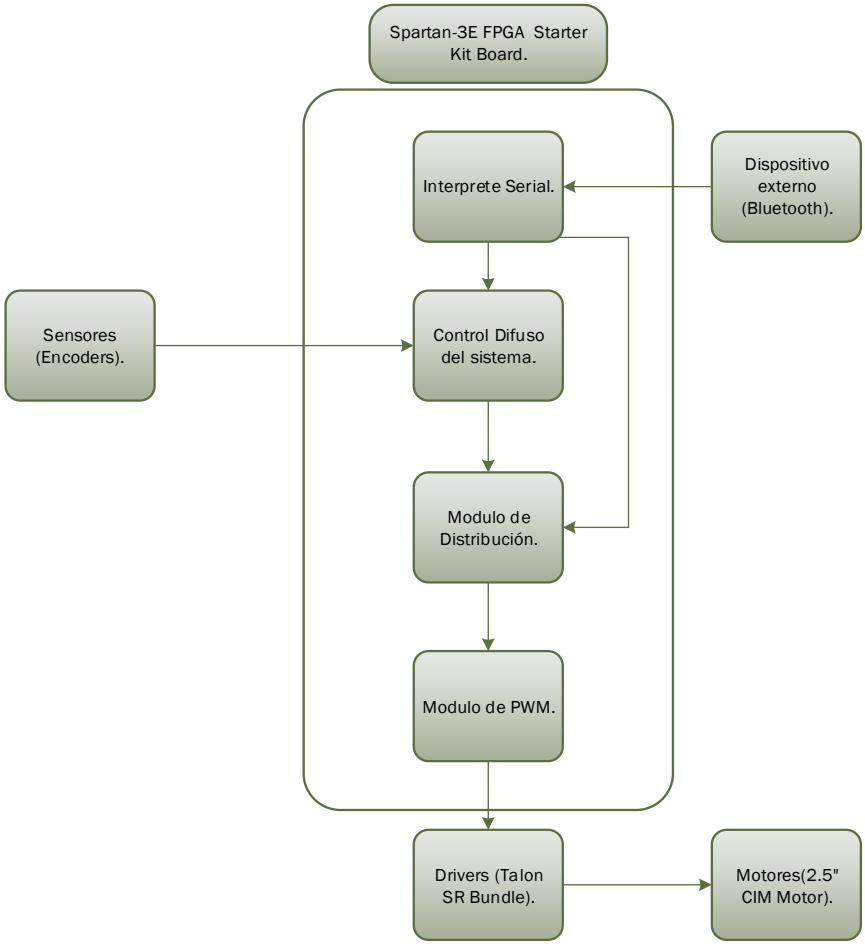


Figura 3.2. Diagrama del proyecto.

En la figura 3.2 se presenta un diagrama general del sistema desarrollado. En este diagrama se desglosa del esquema general de la figura 3.1, que contiene las partes que conforman una plataforma móvil omnidireccional; incluyendo la parte que trata el diagrama anterior.

En la siguiente figura se presenta el modelo del sistema a implementar, con respecto al estudio sobre el tema y una propuesta de solución para el sistema de control de desplazamiento.

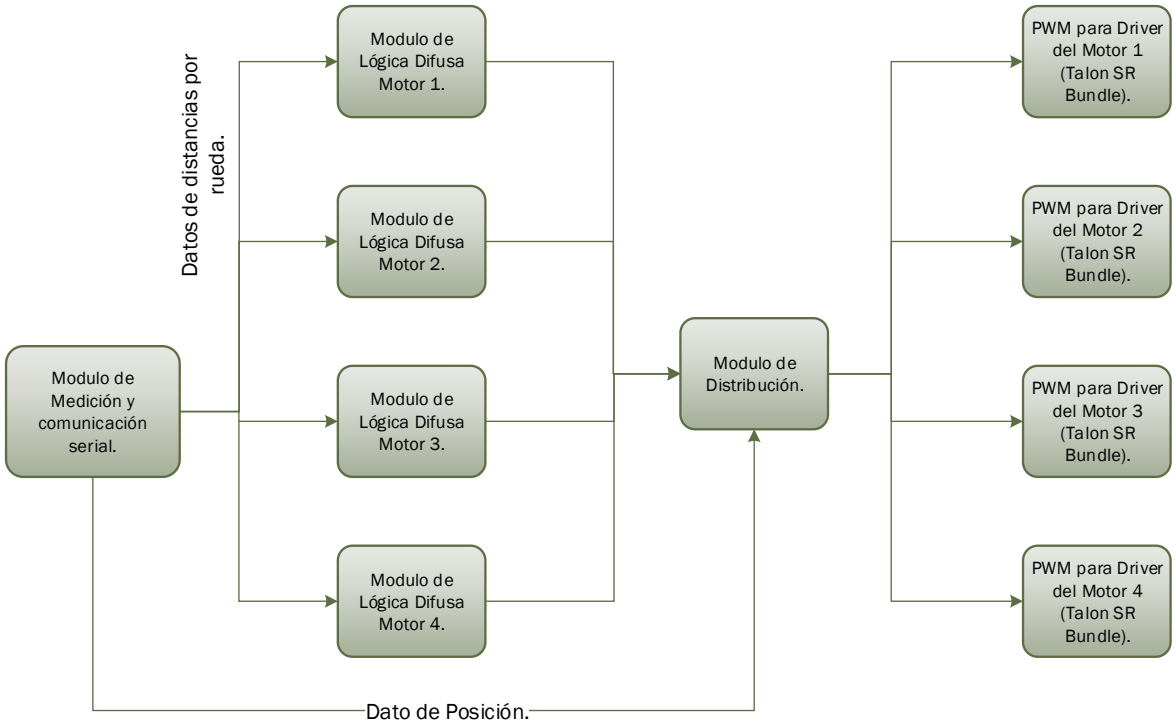


Figura 3.3. Modelo propuesta para el tema.

La figura 3.3 muestra el modelo de diseño propuesto para el control de desplazamiento, que consta:

- Módulo de sistema de comunicación y medición.
- Módulos de lógica difusa.
- Módulo de distribución.
- Módulos de PWM (modulación por ancho de pulsos).

De los cuales, nuestro diseño se enfoca:

- Módulos de lógica difusa.
- Módulo de distribución.
- Módulos de PWM (modulación por ancho de pulsos).

El *módulo de lógica difusa* tiene como tarea tener la lógica de control necesaria para el desplazamiento, con base, en la medición de los codificadores y el comando que se le envié. Este se implementa para cada rueda de la plataforma móvil.

El *módulo de distribución* tiene como tarea, considerando los datos recibidos de los sistemas de lógica difusa y del comando de dirección del sistema de comunicación, determina los accionamientos de los motores tanto en velocidad como en el sentido del giro del motor; otra de las tareas es dar la facilidad de calibrar el sistema.

El *módulo de PWM* (Pulse Width Modulation) tiene como finalidad generar los anchos de pulso para el manejo de los sistemas de control de los motores, calculados entre la etapa de control difuso y la distribución.

Considerando lo expresado en el objetivo de este trabajo se presentan a continuación 2 tablas, donde, la primera estarán las direcciones propuestas para el control de desplazamiento. En la segunda tabla están reflejados los comandos que representaran a estas direcciones.

Tabla 3.1. Comportamiento de la plataforma móvil en 8 direcciones.

Dirección de desplazamiento.	Rueda 1.		Rueda 2.		Rueda 3.		Rueda 4.	
	Dirección	Estado	Dirección.	Estado	Dirección	Estado	Dirección	Estado
↑	↑	True.	↑	True.	↑	True.	↑	True.
↓	↓	True.	↓	True.	↓	True.	↓	True.
→	↑	True.	↓	True.	↓	True.	↑	True.
←	↓	True.	↑	True.	↑	True.	↓	True.
↗	↑	True.	—	False.	↑	True.	—	False.
↖	—	False.	↑	True.	—	False.	↑	True.
↙	↓	True.	—	False.	↓	True.	—	False.
↘	—	False.	↓	True.	—	False.	↓	True.

En la elaboración de la tabla 3.1 se considera la velocidad de giro de los motores como constante para llevar acabo la dirección definida con respecto al sentido que tenga el giro del motor [46] [47].

Tabla 3.2. Comandos propuestos para el manejo del vehículo omnidireccional.

Comandos de ejecución para direccionamiento.		
Dirección.	Modo Manual.	Modo semiautomático.
	61	91
	62	92
	63	93
	64	94
	65	95
	66	96
	67	97
	68	98

En la tabla 3.2 se muestran las propuestas de comandos utilizados para dar la dirección de desplazamiento de la plataforma móvil, el cual, es recibido por medio de comunicación serial para los 2 tipos de operación que se muestra en la tabla (*Modo Manual* y *Modo Semiautomático*) que se tratan en el diseño e implementación del *Módulo de Distribución* a detalle.

Otro elemento de suma importancia para el diseño del sistema de control de desplazamiento es el sistema de control del motor (Talon SR), que es tratado a detalle en el tema siguiente y en temas donde se muestra su relación directa al diseño, del mismo modo de las direcciones y los comandos de operación.

En la parte de comunicación se propone la secuencia de transmisión/recepción de datos que tiene el siguiente orden: comando identificador, comando de dirección y distancia; dependiendo del comando de dirección (Tabla 3.2), se utiliza o no la distancia recibida. Esta secuencia tiene como finalidad de tener datos necesarios para el manejo de la plataforma en sus 2 modos de operación.

3.3. Elementos a utilizar

Plataforma móvil omnidireccional

El *C-Base Frame Wide, Toughbox Mini Gearboxes, Direct Drive, 8" Mecanum Wheels* es un paquete para el armado de la plataforma móvil omnidireccional, el cual contiene [47]:

- Un C-Base Chassis Kit.
- Un conjunto de ruedas omnidireccionales de 8 pulgadas.
- 4 ToughBox Mini.
- 4 2.5" CIM Motor.
- 2 Toughbox Mini para C-Base Mounting Kit.
- Una hoja perforada de policarbonato.
- 4 595x500 espaciadores.
- 4 bujes de conexión modelo 500.

Y en conjunto tiene las siguientes especificaciones.

- Ancho: 37.25".
- Longitud: 26.25".
- Altura: 8".
- Distancia al suelo: 1.25".
- Peso estimado: 32.5 libras.
- Capacidad estimada de carga: 320 libras (Basada en las ruedas).
- Máxima velocidad estimada: 11 [ft/sec].
- Máximo toque en las ruedas estimado: 5100 [oz-in].

En la figura 3.4 se muestra la representación de la plataforma móvil adquirida, mostrada en la página de internet del proveedor AndyMark.

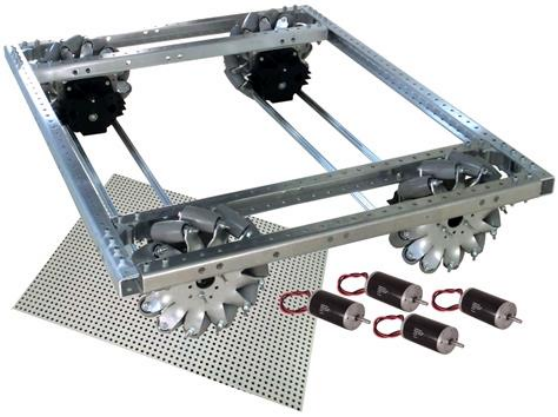


Figura 3.4. C-Base Frame Wide, Toughbox Mini Gearboxes, Direct Drive, 8" Mecanum Wheels (am-0987) [47].

Spartan-3E FPGA Starter Kit Board

La *Spartan-3E FPGA Starter Kit Board* es una tarjeta de desarrollo distribuida por el fabricante Xilinx, es utilizada para aplicaciones orientadas al conocimiento del manejo de dispositivos programables (FPGA) y su utilización en diversas áreas de conocimiento.

Esta tarjeta de desarrollo cuenta con las siguientes características [43]:

- LCD 16X2.
- Conector VGA DB 15 de 4.096 colores.
- Conector puerto RS232 DB9 hembra – macho.
- Conector Ethernet 10/100 PHY.
- Conector JTAG USB para descarga.
- Conector PS/2.
- 4 interruptores.
- 4 interruptores Push Button.
- 8 Led`s.
- 3 expansiones de 6 pines de puertos de conexión.
- Conector FX2 de 100 pines.
- Spartan-3E FPGA (XC3S500E-4FG320C).
- CoolRunner-ll CPLD (XC2C64A-5VQ44C).
- Plataforma flash (XCF04S-VO20C).
- 128 Mbit Intel StrataFlash.
- 16 Mbit SPI Serial Flash.
- 64 Mbyte DDR SDRAM.
- 4 Mbit de memoria de configuración PROM.
- Reloj de 50MHz.
- 4 salidas SPI para DAC (Convertidor Analógico - Digital).
- 2 Entradas SPI para ADC (Convertidor Digital - Analógico).
- Encoder rotatorio con Push Button.
- Conector de entrada de reloj SMA.

Está orientada para proyectos con propósitos de aprendizaje y de uso general, esto, por los componentes que el hardware contiene y lo hace de un dispositivo flexible.

En la figura 3.5 se muestra una imagen de la tarjeta de desarrollo, con la finalidad de dar a conocer la forma del dispositivo.

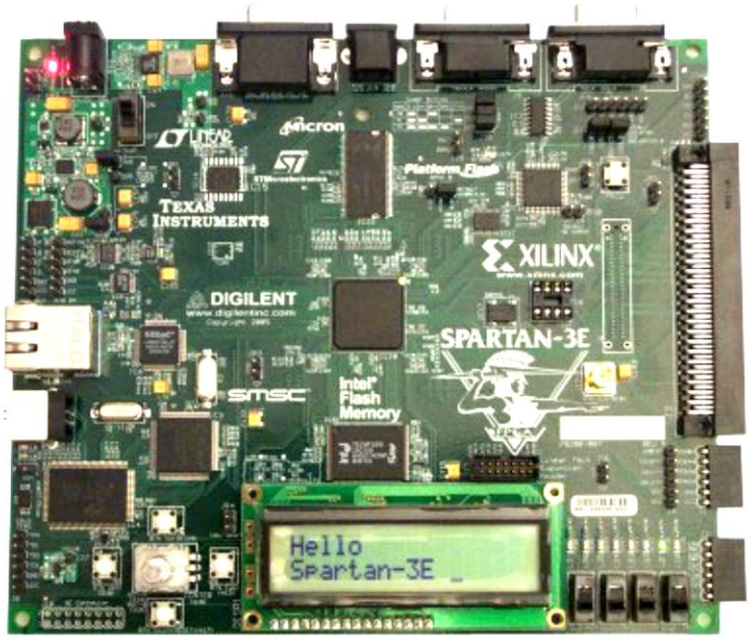


Figura 3.5. Tarjeta de desarrollo Spartan-3E FPGA Starter Kit Board [43].

La tarjeta ilustrada en la figura 3.5 es utilizada para tener la responsabilidad de hacer el proceso de datos y control del sistema en este trabajo.

Modulo RN-24

El elemento de comunicación *RN-24* es un dispositivo de comunicación bluetooth, cuenta con los siguientes elementos [42].

- Bluetooth 2.1/2.0/1.2/1.1 y versión 2.0+EDR.
- Alimentación de 3 a 24 [VCD].
- Hardware TTL.
- UART baud rate: 1200 [bps] hasta 921.6 [Kbps].
- Seguridad de comunicación, 128 bits de encriptación.
- 2 canales de 8 bits ADC (5 [Hz], 0 – 1.8 [VCD]).
- LED indicador Tx/Rx.
- 9 pines de uso general de entrada – salida.
- Bajo consumo (30 [mA] conectado y < 10 [mA] en reposo).

Una de las cuestiones a tratar en el manejo de este dispositivo es con qué tipo de hardware se interconectara y que tipo de comunicación maneja, esto, para saber si utilizamos comunicación serial tipo UART o utilizamos el TTL; en este caso se decidió manejar el TTL por el hardware al cual se conectara el dispositivo. Para ello se debe remover la R7 del dispositivo, para desconectar el UART (Pin 13 y 16) y conectar el hardware TTL (Pin 8 y 11), esta parte se puede observar en la figura 3.6, en la cual, contiene una descripción de lo antes mencionado en inglés.

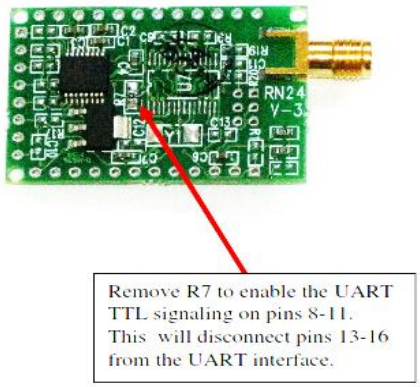


Figura 3.6. RN-24 y descripción de R7 [42].

Una de las características a mencionar son las entradas de alimentación que cuenta este elemento, el cual uno de ellos es regulado a 3.3 [V] y el otro es no regulado desde 4[V] hasta 24[V], se eligió la configuración de 3.3 [V] por el voltaje que maneja nuestro dispositivo.

Los pines que utilizamos en esta ocasión son el pin 1, 2, 3, 8, 9, 10 y 11 para la conexión y manejo de forma TTL del dispositivo, al igual, que se interconectaran los pines 8 y 10 (CTS y RTS) para una comunicación serial asíncrona [42].

Para el propósito de este trabajo se utilizó el modo esclavo, que permite la conexión de cualquier dispositivo a este elemento.

Las forma de configurar la comunicación entre este dispositivo y otros elementos se clasifican mediante comunicación serial, vía remota o física directamente, de las cuales, ninguna utilizaremos en esta ocasión y trabajaremos con configuración de fábrica.

La configuración de fábrica es la que se presenta a continuación:

- Baud Rate 115200.
- 8 bits.
- No paridad.
- 1 bit de paro.
- Control de flujo por hardware.

En la parte de emparejamiento el nombre con el cual aparecerá el dispositivo es el de “FireFly - ABCD”, donde “FireFly” es el tipo de dispositivo de Roving Networks y “ABCD” son los últimos 4 nibbles de la dirección MAC del bluetooth [42].

En el proceso de emparejamiento se requerida una clave de acceso, para ello puede ser reprogramada mediante código o no usar. El código por defecto es el “1234”.

Talon SR y 2.5” CIM Motor

En la cuestión del *Talon SR* tiene las siguientes características [44]:

- Disipador de calor.

- Bloqueo anti fase de rectificación.
- Señal de magnitud síncrona de rectificación.
- 15 [KHz] de frecuencia de oscilación.
- 6 – 28 [VCD] entrada.
- Calibración simple.
- Hasta 100 [A] Pico y 60 [A] de corriente continua.
- Manejo mediante PWM (modulación por ancho de pulso).
- Respuesta lineal de aceleración.
- 10 – bit de Resolución en entrada y salida.
- 4% Neutra banda muerta.
- Selección entre estado de bloqueo o estado libre.

Presenta el siguiente funcionamiento [44]:

- Duración de señal entre 1 – 2 [ms].
- Punto neutro de 1.5 a 1.55 [ms] de duración.

En su funcionamiento se interpreta que cuenta con cambio de giro, esto, por el tiempo neutro que tiene. Las peculiaridades de funcionamiento del Talon SR se mencionan a continuación:

- De 1ms a menos de 1.5 [ms] se cuenta con un giro de motor.
- 1.5 a 1.55 [ms] punto neutro de operación.
- Mayor a 1.55 [ms] y menor o igual a 2 [ms] se cuenta con el giro contrario del motor.
- Linealidad de velocidad con respecto a la variación de la duración de la señal en cada giro del motor.

Su manejo se basa en la modulación por ancho de pulso (PWM) utilizado para el control de posición de servomotores. Los tiempos mencionados anteriormente de los sentidos de giro fueron tomados experimentalmente, estos, son utilizados como referencia en los módulos que forman al sistema de control (Control difuso, Distribución y PWM) para su diseño.

En la figura 3.7, se muestra la forma física del Talon SR, dispositivo para manejar los motores de DC y en la que se ilustran algunas partes físicas mencionadas anteriormente.

Figura 3.7. Talon SR [44].

El modelo matemático que representa el funcionamiento del Talon SR se describe de la siguiente forma:

$$i_1(\theta) = \frac{V_b}{R} - \frac{V_b}{R}\left[1 + \tanh\left(\frac{\pi}{2q}\right)\right] e^{-\theta/q}, \qquad 0 \leq \theta \leq \pi.$$

$$i_2(\theta) = \frac{V_b}{R}\left[1 + \tanh\left(\frac{\pi}{2q}\right)\right] e^{-(\theta-\pi)/q} - \frac{V_b}{R}, \qquad 0 \leq \theta \leq \pi.$$

Estas ecuaciones son consideradas para interpretar el funcionamiento interno del *Talon SR*. Es una configuración *"Square-Wave Inverter with Inductive Load"*, donde $i_1(\theta)$ como $i_2(\theta)$ son las corrientes de carga cuando se activan en pares los interruptores (SW1 y SW3 para $i_1(\theta)$ y SW2 y SW4 para $i_2(\theta)$) que componen al inversor para su control, en este caso del motor, que se considera en la ecuación como una carga inductiva y considera una señal cuadrada de operación de activación y desactivación de los pares de interruptores que componen a esta configuración.

En el caso del motor de DC utilizado es el modelo "*2.5" CIM Motor*", el cual cuenta con las siguientes características:

- Voltaje: 12 [VCD].
- RPM (Revoluciones por minuto) sin carga: 5,310 (+/- 10%).
- Corriente en manejo libre: 2.7 [A].
- Potencia máxima: 337 [W] (a 2655 [rpm], 172 [oz-in] y 68 [A]).
- Par máximo: 2.42 [N-m] (343.4 [oz-in]).
- Corriente máxima: 133 [A].

La parte visual del motor se muestra en la figura 3.8.

Figura 3.8. 2.5" CIM Motor.

Sensor modelo E4P

El sensor modelo E4P que es un codificador reflexivo miniatura que cuenta con las siguientes características [45]:

- 100 a 360 ciclos por revolución (CPR).
- 400 a 1440 pulsos por revolución (PPR).
- Alimentación de 5[V].
- Tolerancia de fuera del eje de .010".

- Acepta +/- .020” juego en el eje.
- Acepta salida del eje desde 2 [mm] hasta 6.350 [mm].
- Tamaño pequeño.

Este sensor es de la división de codificadores ópticos absoluto de cuadratura, el cual cuenta con una salida con un voltaje de 5[V].

En la figura 3.9 se muestra el sensor modelo E4P.

Figura 3.9. Sensor modelo E4P (codificador óptico miniatura) [45].

Al ser un codificador de cuadratura, este tiene entre sus pines dos salidas, las cuales cuenta con un cierto desfase que se muestra en el diagrama de la figura 3.10.

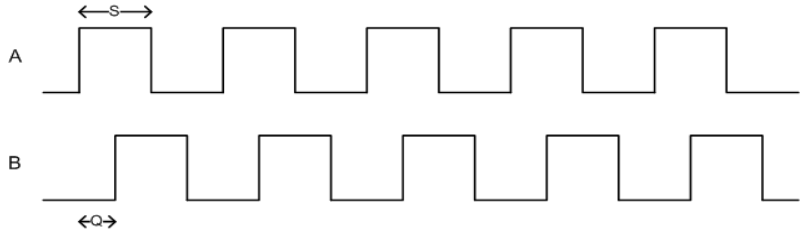


Figura 3.10. Grafica de relación de fase de las salidas [45].

En este modelo de sensor cuenta con 2 tipos de funcionamiento que son: modo diferencial y el modo simple. En este caso se maneja el dispositivo de modo simple, el cual contiene una sola salida A y B.

En la siguiente tabla se muestran los pines correspondientes tanto en la versión simple como en la versión diferencial del sensor.

Tabla 3.3. Configuraciones de pines del sensor modelo E4P.

4 Pin versión Simple		6 Pin versión Diferencial	
Pin	Descripción	Pin	Descripción
1	5 [VCD]	1	Tierra
2	Canal A	2	Canal A
3	Tierra	3	Canal -A
4	Canal B	4	5 [VCD]
		5	Canal B
		6	Canal -B

3.3. Implementación de plataforma móvil omnidireccional

En el armado de la plataforma móvil omnidireccional se utilizan los elementos que ofrece AndyMark para la implementación de una plataforma móvil omnidireccional, que es el *C-Base Frame Wide, Toughbox Mini Gearboxes, Direct Drive, 8" Mecanum Wheels* que se habla en el apartado anterior [47].

En el ensamble del vehículo se inicia por la parte de las ruedas, las cuales, se armó los juegos de engranes junto a su acoplamiento con los motores, como se observa en la figura 3.11.

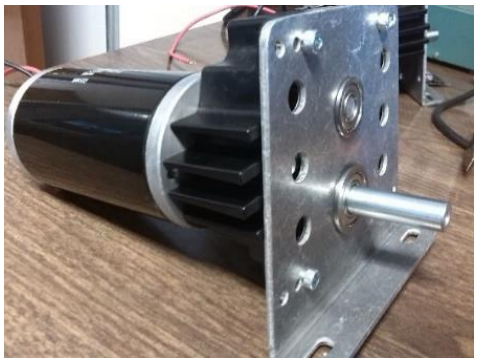


Figura 3.11. Juegos de engranes (*Toughbox Mini*) y motor.

Con el armado de los juegos de engranes con el acoplamiento de los motores se prosiguió con el armado de las ruedas omnidireccionales que se muestran en la figura 3.12.

Figura 3.12. Rueda omnidireccionales de 8 pulgadas.

La rueda que se muestra en la figura 3.12 tiene rodillos en colocación de 45°, esto, considerando los puntos de apoyo que tiene la plataforma móvil para su desplazamiento.

Teniendo ensamblados los juegos de engranes con motores y las ruedas, se procede al armado del cuerpo de la plataforma y el acoplamiento de las ruedas con el juego de engranes.

En esta parte se arma el marco externo de la plataforma y por lo consiguiente se colocó el cuadros de engranes en las barras internas que se unen al marco externo, con esto, se colocan las barras niveladoras de los juegos de engranes y las ruedas. El resultado se muestra en la figura 3.13.

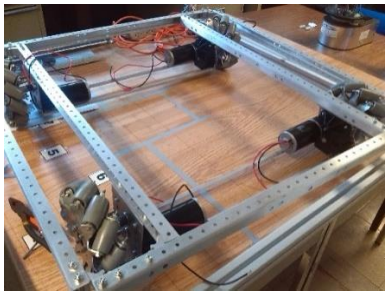


Figura 3.13. Plataforma móvil omnidireccional (Estructura metálica).

Por ultimo colocamos una hoja perforada de policarbonato, esto, para la colocación de objetos en la plataforma. El resultado final de este procedimiento se aprecia en la figura 3.14.

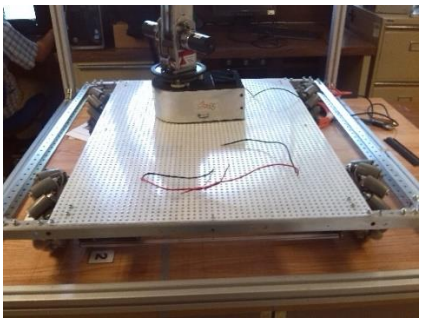


Figura 3.14. Plataforma móvil omnidireccional completa.

El procedimiento a detalle del armado de la plataforma móvil lo encuentra en el anexo A1 llamado "*PLATAFORMA MÓVIL OMNIDIRECCIONAL"* que se encuentra al final de este trabajo.

Diseño e Implementación del Módulo de Lógica Difusa

En el diseño del *Módulo de Lógica Difusa* se considera para su diseño temas como la "*Teoría de lógica difusa*" y el controlador del motor, aparte de otros elementos que se mencionan a continuación.

Para el diseño del *Módulo de Lógica Difuso* se toma en cuenta lo siguiente:

- Variables que afectan al sistema (Entradas y Salidas).
- Dispositivos a usar.
- Dispositivos externos para su funcionamiento.

Considerando estos puntos los elementos que afectan directamente al diseño son:

- Controlador del motor (*Talon SR*).
- Medición (Sensor modelo E4P).
- Hardware (*Spartan-3E FPGA Starter Kit Board*).

El funcionamiento del Talon SR es mencionado en el apartado anterior con todas sus características, del cual, se utilizan los datos de su manejo para el control del motor en ambos sentidos de giro del motor.

Otros de los elementos a considerar es el reloj interno de trabajo que cuenta el *Spartan-3E FPGA Starter Kit Board,* que es de 50MHz y el valor de distancia obtenido de la medición del sensor modelo E4P.

Diseño del módulo difuso

En este diseño se proponen las siguientes variables difusas:

- Entradas.
  - Referencia (Medidas de longitud).
  - Sensor (Medidas de longitud).
- Salida.
  - Ciclos (número de ciclos).

Para el diseño del universo de discurso(rango donde la variable es válida) de entrada, se toma en cuenta la distancia de desplazamiento máxima considerada (200 [cm]), mientras, el universo de discurso de salida se toma en cuenta el reloj interno del *Spartan-3E FPGA Starter Kit Board* y los tiempos de operación del controlador del motor (*Talon SR*). Para el cálculo del universo de discurso de la variable de salida se considera el tiempo en que ejecuta una instrucción el FPGA, que se muestra en la ecuación 3.1.

$$Periodo = \frac{1}{Frecuencia} = \frac{1}{F} = \frac{1}{50MHz} = 20ns \tag{3.1}$$

Donde:

$$1\ Ciclo\ \rightarrow 20\ ns$$

El tiempo de duración de los sentidos de giro del motor está determinados por el tipo de señal que se requiere generar para el manejo del controlador del motor (*Talon SR)*, este valor es calculado mediante estas representaciones:

$$Rango\ de\ tiempo\ de\ giro\ de\ motor\ = punto\ neutro - limite\ inferior$$

$$= 1.5\ ms - 1\ ms = .5\ ms \tag{3.2}$$

$$Rango\ de\ tiempo\ de\ giro\ contrario\ del\ motor = limite\ superior - punto\ neutro$$

$$= 2\ ms - 1.5\ ms = .5\ ms \tag{3.3}$$

Tomando en cuenta que los resultados de la ecuación 3.2 y 3.3. Se toma el valor como límite máximo para el universo de discurso para la variable lingüística de salida, aunque, por comodidad y mejorando la precisión del controlador se convierte a una representación de ciclos de ejecución para el universo de discurso, que se lleva a cabo en la ecuación 3.4.

$$Nciclos = \frac{Tiempo\ de\ duración\ deseado}{Periodo} = \frac{.5\ ms}{20\ ns} = 25000\ ciclos \tag{3.4}$$

En la ecuación 3.4, se toma como referencia de conversión el tiempo de duración de 1 ciclo de ejecución de una instrucción del FPGA para el cálculo.

Para el diseño del módulo de control difuso se toma en cuenta una parte de las características que debe de tener la señal de controlador del motor, estas son complementadas en el *Modulo de Distribución* para la generación de la señal modulada que se forma en el *Módulo PWM*.

Una parte importante que mencionar es la utilización de *MATLAB®* con el Apps *Fuzzy Logic Design*, con la finalidad de ver el comportamiento del sistema difuso propuesto [48].

La variable lingüística *Referencia* tiene estos elementos.

- Universo de discurso dado en centímetros.
- 5 conjuntos difusos.
    - Cero
    - Corta.
    - Media.
    - Larga.
    - Máxima.
- Funciones de membresía de cada conjunto tiene una forma triangular.

El número de conjunto que se propone, están relacionados con la precisión que se quiere obtener en el controlador.

La forma triangular de la función de membresía de los conjuntos difusos es tener una membresía total en un solo punto y obtener una ambigüedad intermedia entre conjuntos de la variable difusa.

A continuación se muestra la variable difusa de *Referencia*.

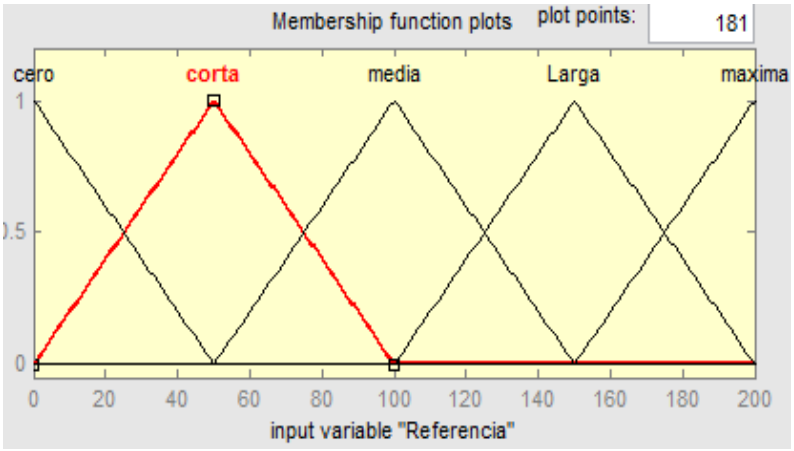


Figura 3.15. Variable lingüística de entrada *Referencia* en *MATLAB®*.

En la figura 3.15, se puede observar tanto la forma de los cinco conjuntos difusos (cero, corta, media, larga y máxima), como su colocación en el universo de discurso.

En la parte de colocación de conjuntos difusos en el universo de discurso se tiene presenta las intercesiones que tiene entre ellos, lo cual, hace del sistema difuso tomar un rango de valores no absolutos en comparación a la algebra booleana.

La variable lingüística *Sensor* tiene estos elementos.

- Universo de discurso dado en centímetros.
- 5 conjuntos difusos.

- Cero.
- Corta.
- Media.
- Larga.
- Máxima.

- Funciones de membresía de cada conjunto tiene una forma triangular.

Esta variable lingüística es un duplicado de la variable *Referencia*, esto se debe a la propuesta del diseñador para una fácil manipulación y respuesta del sistema, considerando la operación del controlador de motor (*Talon SR*).

La representación gráfica de la variable difusa *Sensor* se presenta a continuación.

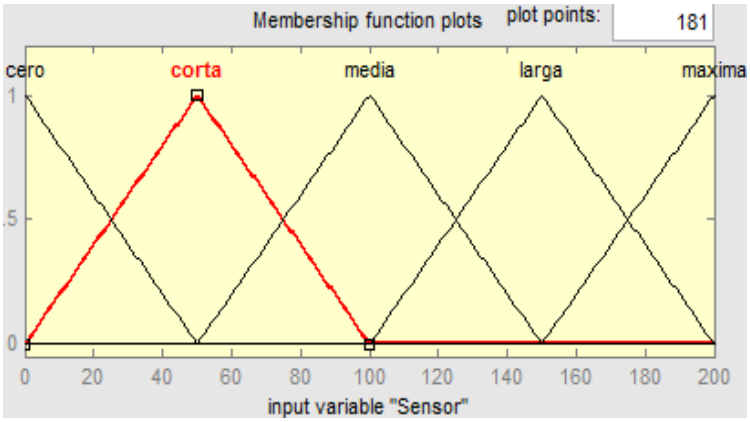


Figura 3.16. Variable lingüística de entrada *Sensor* en *MATLAB®.*

Como se puede apreciar en la figura 3.16, tiene la misma forma y la misma colocación que la variable lingüística *Referencia*.

Ahora que se dio una explicación detallada de las variables de entrada, proseguimos con la variable lingüística de salida *Ciclos*.

La variable lingüística *Ciclos* tiene estos elementos.

- Universo de discurso dado en número de ciclos.
- 5 conjuntos difusos.
  - Zero.
  - Baja.
  - Media.
  - Alta.
  - Máxima.
- Funciones de membresía de cada conjunto tiene la forma singleton (Valor único).

La forma de los conjuntos difusos en esta variable está dada por la reacción del sistema de control, que, al tener un único valor en la parte de defuzzificación, se pretende tener una rápida respuesta del sistema difuso.

Por lo tanto, la variable lingüística *Ciclos* de salida queda de la siguiente manera.

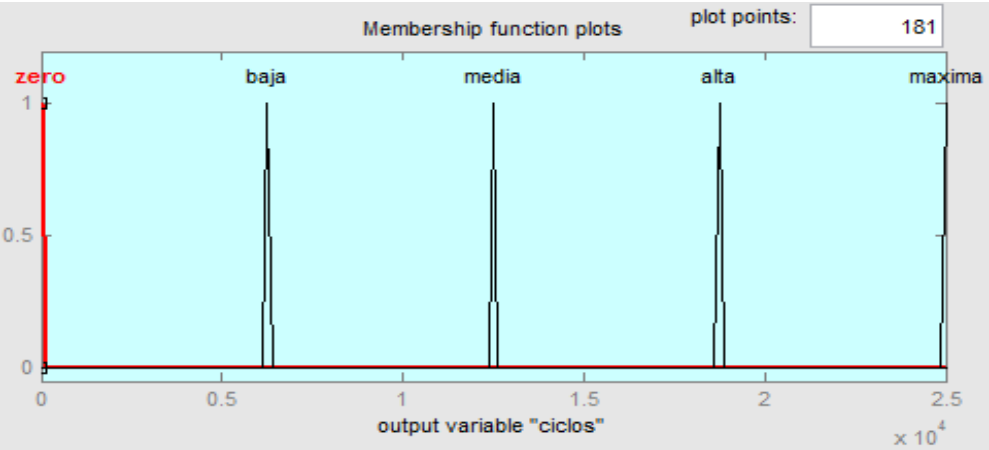


Figura 3.17. Variable lingüística de salida *ciclos* en *MATLAB®*.

Como se observa en la figura 3.17, la forma que contiene esta variable es de un solo punto máximo de valor (membresía = 1), el cual, para un sistema de control en tiempo real y considerando las operaciones de conversión de una variable lingüística a numérica (*defuzzificación*), es viable para nuestros propósitos.

En la parte de elaboración de la tabla de decisión es diseñada por el experto, que cuenta con los conocimientos necesarios del tema para implementar el diseño, esta se muestra a continuación:

Tabla 3.4. Tabla de decisión del sistema difuso.

Tabla de decisión.		
Referencia.	Sensor.	Ciclos.
Cero	Cero	Zero
Cero	Corta	Zero
Cero	Media	Zero
Cero	Larga	Zero
Cero	Máxima	Zero
Corta	Cero	Baja
Corta	Corta	Zero
Corta	Media	Zero
Corta	Larga	Zero
Corta	Máxima	Zero
Media	Cero	Baja
Media	Corta	Baja
Media	Media	Zero
Media	Larga	Zero
Media	Máxima	Zero
Larga	Cero	Alta
Larga	Corta	Media
Larga	Media	Baja
Larga	Larga	Zero
Larga	Máxima	Zero

Máxima	Cero	Máxima
Máxima	Corta	Alta
Máxima	Media	Media
Máxima	Larga	Baja
Máxima	Máxima	Zero

En la tabla 3.4 se muestra todas las combinaciones entre las variables de entrada y salida.

En la elaboración de la tabla de decisión se toma en cuenta para la selección de la salida con respecto a la combinación de entrada; la opinión del experto decide qué tipo de salida es mejor para la combinación de entradas.

Siguiendo con el diseño de este módulo; pasamos al diseño de las reglas difusas. En esta parte de diseño es solamente traducir lo realizado en la tabla 3.4 a una estructura del tipo "*If – Then".* Las reglas para este sistema se muestran a continuación:

1. If (Referencia is cero) and (Sensor is cero) then (cyclos is baja) (1)
2. If (Referencia is cero) and (Sensor is corta) then (cyclos is zero) (1)
3. If (Referencia is cero) and (Sensor is media) then (cyclos is zero) (1)
4. If (Referencia is cero) and (Sensor is larga) then (cyclos is zero) (1)
5. If (Referencia is cero) and (Sensor is maxima) then (cyclos is zero) (1)
6. If (Referencia is corta) and (Sensor is cero) then (cyclos is baja) (1)
7. If (Referencia is corta) and (Sensor is corta) then (cyclos is zero) (1)
8. If (Referencia is corta) and (Sensor is media) then (cyclos is zero) (1)
9. If (Referencia is corta) and (Sensor is larga) then (cyclos is zero) (1)
10. If (Referencia is corta) and (Sensor is maxima) then (cyclos is zero) (1)
11. If (Referencia is media) and (Sensor is cero) then (cyclos is baja) (1)
12. If (Referencia is media) and (Sensor is corta) then (cyclos is baja) (1)
13. If (Referencia is media) and (Sensor is media) then (cyclos is zero) (1)
14. If (Referencia is media) and (Sensor is larga) then (cyclos is zero) (1)
15. If (Referencia is media) and (Sensor is maxima) then (cyclos is zero) (1)
16. If (Referencia is Larga) and (Sensor is cero) then (cyclos is alta) (1)
17. If (Referencia is Larga) and (Sensor is corta) then (cyclos is media) (1)
18. If (Referencia is Larga) and (Sensor is media) then (cyclos is baja) (1)
19. If (Referencia is Larga) and (Sensor is larga) then (cyclos is zero) (1)
20. If (Referencia is Larga) and (Sensor is maxima) then (cyclos is zero) (1)
21. If (Referencia is maxima) and (Sensor is cero) then (cyclos is maxima) (1)
22. If (Referencia is maxima) and (Sensor is corta) then (cyclos is alta) (1)
23. If (Referencia is maxima) and (Sensor is media) then (cyclos is media) (1)
24. If (Referencia is maxima) and (Sensor is larga) then (cyclos is baja) (1)
25. If (Referencia is maxima) and (Sensor is maxima) then (cyclos is zero) (1)

Figura 3.18. Reglas propuestas para el modulo difuso en *MATLAB®.*

En la figura 3.18 se muestra tanto la estructura de las reglas y su forma en las cual pueden ser creadas.

Ahora definimos la parte de "*defuzzificación*", que es uno de los procesos que depende el método difuso para la precisión como la rapidez de ejecución en su funcionamiento práctico.

Se considera el método de defuzzificación mediantes *Centro del Área* (COA), la cual, tiene la siguiente representación matemática:

$$Z_0 = \frac{\sum_{j=1}^{n} \mu_C(Z_j) * Z_j}{\sum_{j=1}^{n} \mu_C(Z_j)} \qquad (3.5)$$

Las cualidades para aplicar el modo de defuzzificación de la ecuación 3.5 son las siguientes:

- Mayormente aplicado a figuras simétricas.
- Tener conjuntos difusos simétrico.
- Facilidad de aplicar mediante una sucesión de sumas en cualquier dispositivo de programación electrónico.

Por estas cualidades, se propone este método aunque hay más que realizan esta misma tarea pero con condiciones de operación diferentes.

Por último mostramos el sistema genera del diseño con sus variable de entrada, salida y su módulo de inferencia.

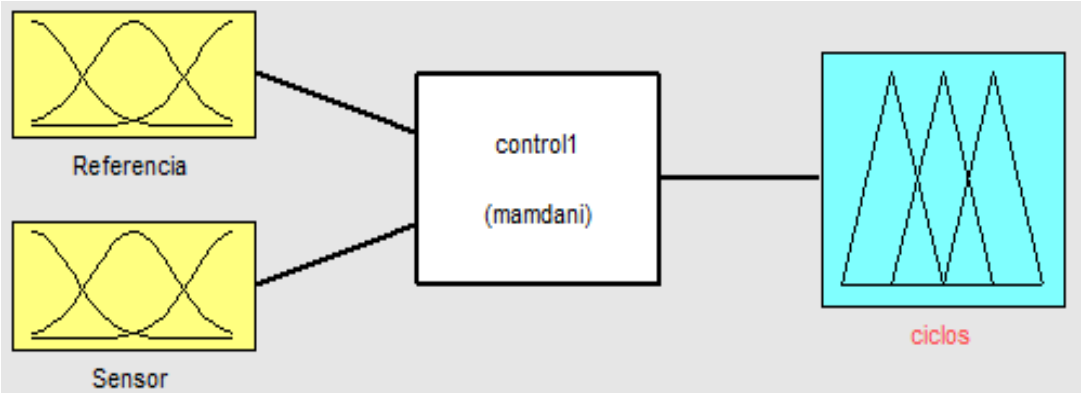


Figura 3.19. Esquema general del módulo difuso en *MATLAB®*.

Como se observa en la figura 3.19, está la estructura general del sistema, tanto las consideraciones generales de diseño como los bloques que representa a cada una de las parte del sistema.

En la figura 3.20 se muestra comportamiento del sistema con respecto a la entrada de valores entre los rangos propuestos en el universo de discurso. Se visualiza la reacción de la salida con respecto a los valores de entrada dados y los conjuntos difusos seleccionados.

En la figura 3.21 se muestra un diagrama general con los posibles comportamientos del sistema difuso tomando en cuenta los universos de discurso de las variables.

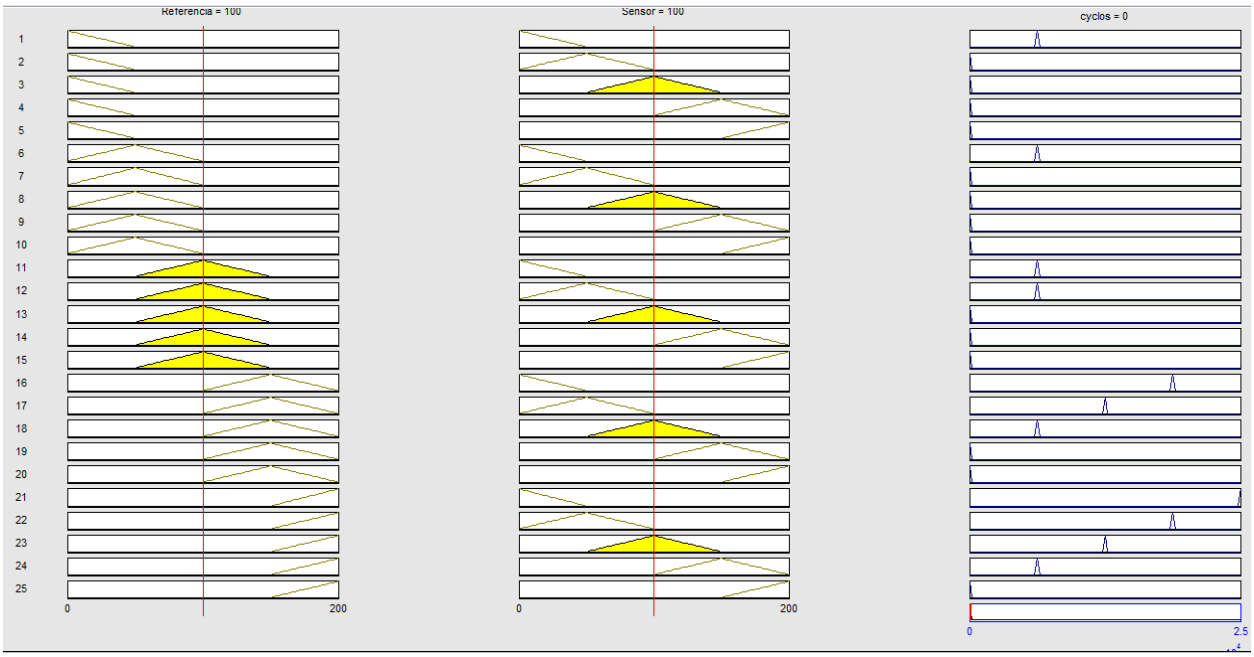


Figura 3.20. Comportamiento de la variable de salida con respecto a las variables de entrada y las reglas en *MATLAB®.*

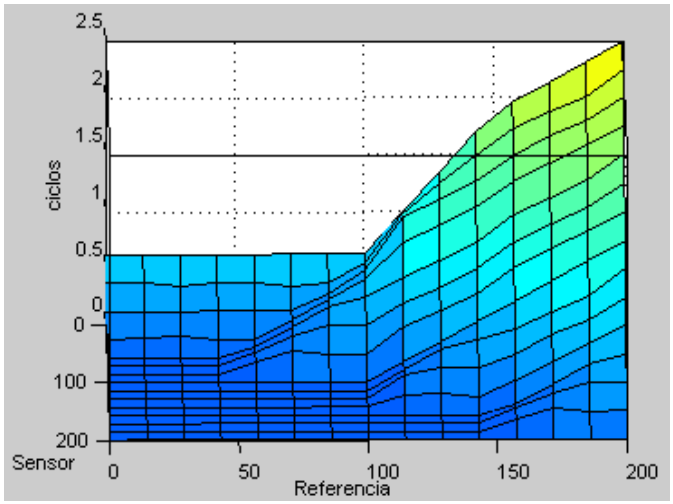


Figura 3.21. Comportamiento del sistema difuso en *MATLAB®*.

Con la utilización de *MATLAB®* podemos observar el comportamiento del diseño del módulo difuso para su aplicación en el controlador de desplazamiento.

Implementación del módulo de control difuso

Considerando información recopilada [5], [14], [52], [53] y [54], una de las formas más utilizadas para la implementación de un sistema difuso en un FPGA es mediante la realización de tabla de valores, en donde, mediante las entradas se considera una salida determinada. Uno de los inconvenientes de este forma de

implementación es que estas ligado el margen de error del sistema al número de valores considerados en la elaboración de la tabla, lo cual, al realizar una modificación, se tiene que hacer en su totalidad de la tabla, esto limita la modificación rápida y efectiva del sistema.

Para la implementación del módulo de control en nuestro caso se considera como primer objetivo el transforma en ecuaciones las figuras de las funciones de membresía de las variables ligústicas de entrada.

En la parte de salida como tenemos un punto máximo en una sola cantidad no es necesario esta transformación con esta variable.

Para las variables difusas de entrada consideramos su representación mediante un sistema de ejes de coordenadas "x" y "y" en donde las "x" son los valores que toma el universo de discurso y las "y" los valores de membresía.

Tomando las formas triangulares de las funciones de membresía como un conjunto de rectas, utilizamos la representación matemática 3.6 para la formación de nuestras ecuaciones.

$$y = mx + b \tag{3.6}$$

Considerando que no conocemos la pendiente de la recta en donde corta con el eje "y", se calcular utilizando la fórmula 3.7.

$$m = \frac{y_2 - y_1}{x_2 - x_1} \tag{3.7}$$

Con las ecuaciones 3.6 y 3.7, podemos determinar la ecuación característica de cada una de las rectas que forman a los triángulos de las funciones de membresía. Para ello tendrán diferentes pendientes y valores de corte en el eje "y".

Una de las consideraciones para la creación de las ecuaciones es el cambio de rango de la función de membresía, en vez de 0 a 1 lo tomaremos de 0 a 100, como nueva propuesta de diseño para aplicación de sistemas difuso tanto en hardware como software que cuente con limitaciones.

Las ecuaciones que representan los conjuntos difusos de las variables lingüísticas se presenta a continuación.

$$\begin{cases} 0 < x \leq 50 & \begin{cases} y = 2x \\ y = -2x + 100 \end{cases} \\ 50 < x \leq 100 & \begin{cases} y = -2x + 200 \\ y = 2x - 100 \end{cases} \\ 100 < x \leq 150 & \begin{cases} y = -2x + 300 \\ y = 2x - 200 \end{cases} \\ 150 < x \leq 200 & \begin{cases} y = -2x + 400 \\ y = 2x - 300 \end{cases} \end{cases} \tag{3.8}$$

La ecuación 3.8 será utilizada para las 2 variables lingüísticas de entrada, por sus características que comparte entre ellas.

Con el conjunto de ecuaciones 3.8, proponemos un modelo, el cual, se basa en la recreación del sistema básico de lógica difuso, con la diferencia del cambio de rangos de las funciones de membresía; con la

finalidad de facilitar la implementación del sistema tanto en FPGA como en cualquier otro dispositivo con limitaciones de módulos de operación o manejo del punto flotantes.

El sistema que se propone en este trabajo se basa como se mencionó anteriormente en el modelo básico de un sistema difuso, con la diferencia del cambio de rango de las funciones de membresías de la variable difusa, con la finalidad de evitar el manejo de valores en punto flotante; otra de las cuestiones es el manejo del conjunto de operaciones de la ecuación 3.8 para tener una amplia gama de obtención de valores y eliminar la limitación de tener valores en ciertos rango.

Con estas consideraciones se obtuvo en primer lugar, la creación de un sistema con un código reducido y fácilmente modificable; otras de las ventajas es no tener una limitante de valores excepto la de los universo de discurso en el diseño del sistema.

El margen de error del sistema está ligado solamente al manejo de los puntos flotantes en el método de *defuzzificación*, esto, por la aplicación del redondeo del resultado obtenido de sucesiones de restas emulando la división que en hardware.

El desarrollo del algoritmo para la implementación del sistema difuso se planteó con base en el esquema que se muestra en la figura 3.22, donde, se estructura las partes de funcionamiento de un sistema difuso con respecto al sistema que se diseñó para este trabajo.

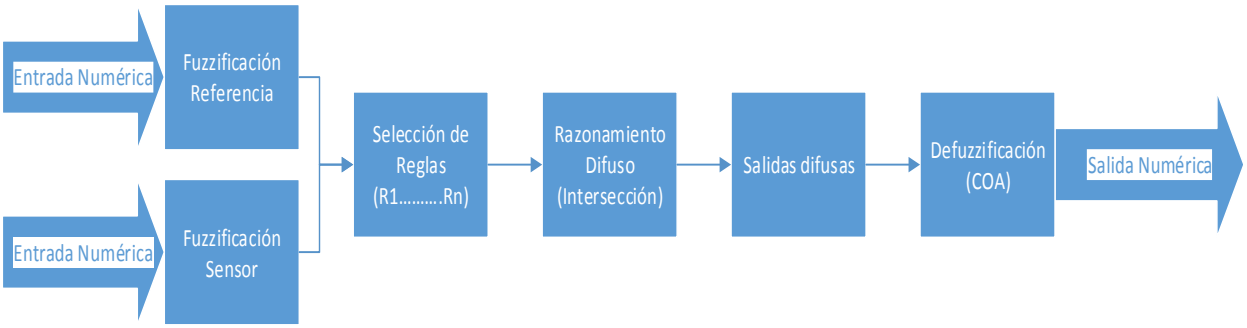


Figura 3.22. Esquema de funcionamiento del *módulo de lógica difusa.*

Con base en la figura 3.22 para la implementación en FPGA del *módulo de lógica difusa*, este consiste en el siguiente procedimiento:

- En el inicio del proceso se toman los valores de las variables de entrada (*Referencia* y *Sensor*) y se les aplica el proceso de "*fuzzificación*" para convertirlas de una variable física a una variable lingüística con un valor de membresía modificado de 0 a 100.
- Al obtener los valores del sistema de "*fuzzificación*" en donde se obtiene tanto el valor lingüístico, como, su membresía se procede a utilizar el sistema de elección de reglas mostrado para la elección de las reglas que se acoplan a las variables de entrada difusa.
- Como siguiente paso se hace e *razonamiento difuso*, el cual, consiste en la ejecución de las reglas mediante las operaciones de intersección, unión, complemento entre otras (en este caso intersección) para su ejecución en las reglas mediante la aplicación de estos operadores para la obtención de las salidas difusas.

- Al terminar el proceso de razonamiento y tener las salidas difusas, se pasa a convertirlas a salidas numéricas, esto se puede obtener con el proceso de "*defuzzificación*" que en este caso se utiliza el método de *Centro de Área* para ello.

Con el esquema de funcionamiento y procedimiento de operación del *módulo de lógica difusa* para su implementación, se pasa a su aplicación en VHDL, mediante *XILIX ISE 14.6*. El esquemático obtenido se presenta en la figura 3.23.

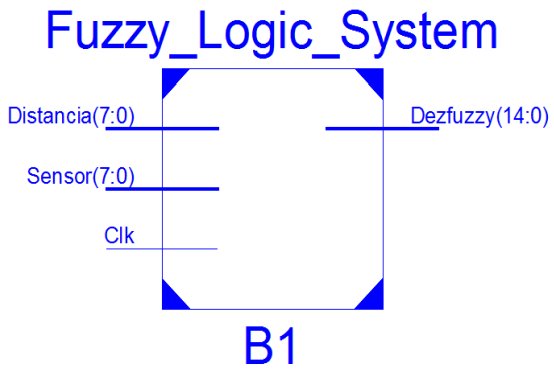


Figura 3.23. Esquema obtenido del *Módulo de Lógica Difusa* en *Xilinx ISE 14.6*.

En este esquemático se puede observar que, aparte de las entradas y salidas designadas, se cuenta con una entrada de reloj, esto es para el procesos de la operación de división, este no puede ser realizado directamente en el dispositivo, por lo tanto, se aplica un algoritmo con la finalidad de realizar esta operación mediante una señal de reloj.

El código de la implementación lo puede consultar en el anexo A2 con el título "*BLOQUE DE SISTEMA DIFUSO*".

3.4. Diseño e Implementación del Módulo de Distribución

En el *Modulo de Distribución* se realiza la tarea de indicar sentido de giro y velocidad de los motores, con base en las especificaciones del controlador del motor (*Talon RS*) y las direcciones tomadas en cuenta para este proyecto.

Para ello se consideran las tablas 3.1 y 3.2 mostradas en la estructura de diseño del sistema de control de desplazamiento. Mediante esas tablas se desglosan 2 modos de operaciones que se utilizan para el desplazamiento de la plataforma móvil, los cuales son:

Modo Manual: donde se selecciona solamente la dirección de la plataforma móvil para su desplazamiento sin control.

Modo Semiautomático: este se elige tanto la dirección de la plataforma móvil como la distancia que tiene que recorrer; en esta opción entra en operación el sistema de control de desplazamiento.

En la parte de implementación de este "*Modulo de Distribución*", se toma en cuenta las entradas que le proporciona datos de la parte de comunicación y del "*Módulo de Lógica Difusa*", para que, en la salida

proporcione el número de ciclos necesarios a los módulos de PWM para la generación de las señales para el manejo del controlador del motor (*Talon SR*) en el accionamiento de los motores.

Para su implementación se maneja el punto medio de operación del dispositivo de control del motor (1.5 ms). Aplicamos la ecuación 3.4 para conocer el número de ciclos que corresponde a este tiempo.

$$Nciclos\ de\ punto\ medio = \frac{1.5\ ms}{20\ ns} = 75000\ ciclos \tag{3.9}$$

Considerando la ecuación 3.9, se propone una operación de suma o resta entre el número de ciclos del *Punto Neutro* y los datos obtenidos de los controladores, dependiendo del movimiento que tenga que realizar la plataforma móvil mediante el comando de dirección dado.

Se propone 3 ecuaciones para el cálculo de los ciclos necesarios para la creación de las señales de manejo del controlador del motor en los módulos de *PWM.* La utilización de las tres ecuaciones depende del comando utilizado para la dirección del que se desplazará la plataforma móvil. Estas tres ecuaciones son las siguientes:

$$T(PWM[Ciclos]) = Punto\ Medio(Ciclos) + Control\ Difuso(Ciclos) \tag{3.10}$$

$$T(PWM[Ciclos]) = Punto\ Medio(Ciclos) - Control\ Difuso(Ciclos) \tag{3.11}$$

$$T(PWM[Ciclos]) = Punto\ Medio(Ciclos) \tag{3.12}$$

Las ecuaciones 3.10 y 3.11 se aplican solamente en los comandos del "*modo semiautomático"* de operación.

Con estas 3 relaciones, y considerando las tablas 3.1 y 3.2, se hace una comparación del comando recibido con nuestra lista de comparaciones para la selección y ejecución de una de las ecuaciones dadas anteriormente (Ecuaciones 3.10, 3.11 y 3.12); dependiendo que acción de desplazamiento se quiera utilizar.

En la parte de los comandos del "*Modo Manual"* se toma aparte del "*Punto Neutro"* de operación la mitad del tiempo calculado en las ecuaciones 3.2 y 3.3 que tienen el mismo valor (.5[ms]). Esto para operar el sistema sin control y con un valor arbitrario de velocidad.

Tomando como tiempo de duración de la señal .25 [ms] para generar una velocidad arbitraria al utilizar el "*Modo Manual*" de operación, se realiza la conversión de este tiempo a ciclos utilizando la ecuación 3.4, la cual nos arroja el siguiente resultado.

$$Nciclos\ de\ .25\ ms = \frac{.25\ ms}{20\ ns} = 12500\ ciclos \tag{3.13}$$

Con la ecuación 3.13, se proponen para el *Modo Manual* estas 2 ecuaciones más aparte la ecuación 3.12 para la generación de la señal de manejo del controlador del motor.

$$T(PWM[Ciclos]) = Punto\ Medio(Ciclos) + .25\ ms(Ciclos) \tag{3.14}$$

$$T(PWM[Ciclos]) = Punto\ Medio(Ciclos) - .25\ ms(Ciclos) \tag{3.15}$$

Dependiendo el comando ejecutado del Modo Manual, se aplicara una de las ecuaciones anteriores (3.12, 3.14 y 3.15) o en conjunto para desplazar el móvil.

En el siguiente esquema es obtenido de *Xilinx ISE 14.6*. Resultado de la implementación del diseño en programación de descripción de hardware.

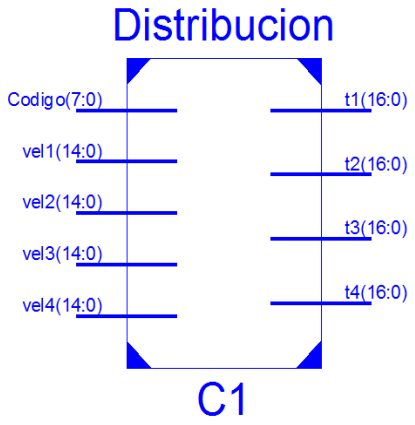


Figura 3.24. Esquema obtenido del *Módulo de Distribución* en *Xilinx ISE 14.6*.

En la figura 3.24 se observa el esquemático resultante del proceso de síntesis en la programación mediante descripción de hardware, aplicado en *Xilinx ISE 14.6* para ser usado en la *Spartan-3E FPGA Starter Kit Board* como parte del sistema de control de desplazamiento.

En el anexo A3 con el nombre "*BLOQUE DE DISTRIBUCIÓN*" se puede observar el código fuente utilizado para la implementación del módulo de distribución.

3.5. Diseño e Implementación del Módulo PWM

La unidad de PWM, mediante los datos recibidos del módulo de distribución, tiene la tarea de generar una señal modulada para el manejo del controlador del motor (*Talon SR)*.

Considerando la señal de control requerida para operar el *Talon SR* se toma las siguientes consideraciones:

- Ancho de pulso bajo constante de 2 ms.
- Ancho de pulso alto es dado directamente por el dispositivo de distribución.

El diseño de este módulo requiere solamente de conocer los ciclos equivalentes de 2 ms, que se obtiene mediante la ecuación 3.4, el cual, el resultado se presenta en la siguiente representación matemática.

$$Nciclos\ de\ 2\ ms = \frac{2\ ms}{20\ ns} = 100000\ ciclos \tag{3.16}$$

Con el resultado obtenido de la ecuación 3.16, podemos definir el número de ciclos necesarios para tener un tiempo en bajo, mientras, el tiempo en alto es dado directamente por el módulo de distribución.

En la implementación de este módulo se toma en cuenta los datos recibidos del módulo de distribución y el dato obtenido de la ecuación 3.16, con ellos genera una señal cuadra con esos tiempos de duración tanto en la parte positiva como en la parte negativa.

El proceso para la generación de la señal modulada en VHDL es mediante un proceso, en donde cada ciclo de reloj se realiza una cantidad de incrementos dada dependiendo del estado de la señal, con la finalidad de controlar la duración de la señal generada y su cambio de estado.

Con base en el funcionamiento descrito anteriormente. El funcionamiento del bloque para la generación de la señal modulada es implementado mediante lenguaje de descripción de hardware en el software de programación *Xilinx ISE 14.6.* En la figura siguiente se presenta el esquema obtenido del proceso de síntesis.

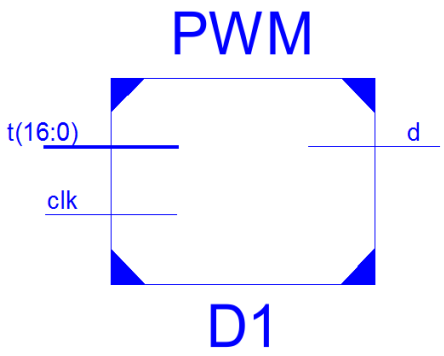


Figura 3.25. Esquema de *Xilinx ISE 14.6* del *Módulo PWM.*

Como se observa en la figura 3.25, este módulo cuenta con 2 entradas y una salida, donde, las entradas son las señal de reloj y el ciclo de duración en alto de la señal generada, obtenidos de los bloques anteriores, la salida es la señal generada deseada para el control del *Talo SR*.

En el anexo A4 con el nombre de "*BLOQUE PWM*" se muestra el código fuente del programa hecho por descripción de hardware para la generación de la modulación de los anchos de pulso.

3.6. Diseño e Implementación de Aplicación para Android

Para simular la interacción entre la parte de ubicación y trayectorias con la parte de control de desplazamiento en un sistema de navegación, se propone un sistema remoto mediante comunicación Bluetooth utilizando el sistema operativo "*Android*" y programada en el ambiente de programación "*Processing*".

Esta parte, el usuario funge como el sistema de navegación, el cual, tendrá el control del desplazamiento del móvil en base a decidir la distancia y dirección tendrá la plataforma móvil.

Como parte de esta interacción se usan teléfonos celulares inteligentes (*Smartphone*), los cuales, contiene un sistema operativo para realiza diferentes funciones. Con esta tecnología se utiliza la comunicación Bluetooth, que contiene la mayor parte de estos elementos.

Para este proyecto se elabora una aplicación, que nos permita controlar de forma inalámbrica la plataforma móvil.

Para el diseño de esta aplicación se utiliza el siguiente diagrama de flujo.

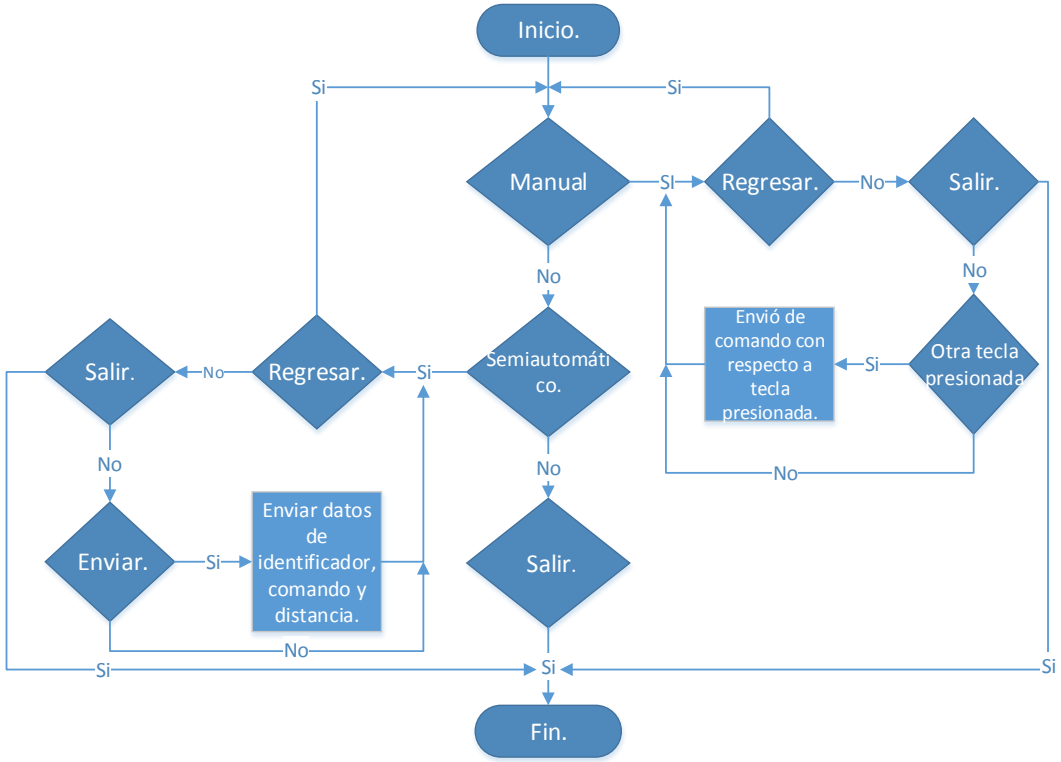


Figura 3.26. Diagrama para la programación de la aplicación Android del *"Controlador para el desplazamiento de una plataforma móvil en aplicaciones de navegación utilizando FPGA".*

El diagrama de la figura 3.26 es programado en el compilador "*Processing*", que cuenta con un conjunto de instrucciones basadas en diseño gráfico, además, de un sistema de soporte para varios tipos de plataformas; en este caso utilizamos la ofrecida para *Android* para la programación de la aplicación, en la

cual, se utiliza la librería *"Ketai"* que contiene conjunto de instrucciones para el manejo de los elementos de comunicación [35], [50] y [51].

Con base en el diagrama de la figura 3.26 y las herramientas de programación antes mencionadas se elaboró una aplicaciones para el sistema operático *Android.* La parte grafica de esta aplicación se presentan a continuación en las siguientes figuras.

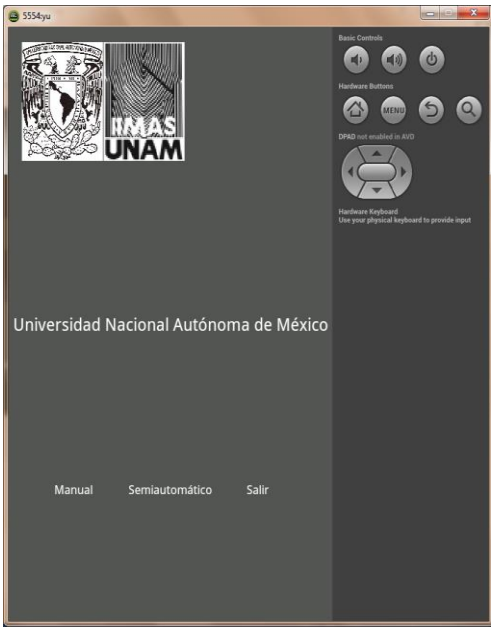


Figura 3.27. Pantalla de inicio de la aplicación de comunicación para el control de desplazamiento de la plataforma móvil.

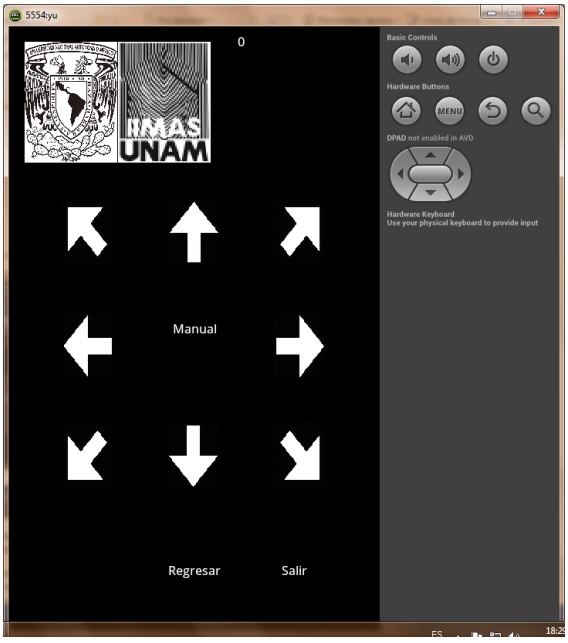


Figura 3.28. Pantalla de "*Control Manual*" de la aplicación de comunicación para el control de desplazamiento de la plataforma móvil.

Figura 3.29. Pantalla de "*Control Semiautomático*" de la aplicación de comunicación para el control de desplazamiento de la plataforma móvil.

La figura 3.27 es la pantalla de inicio, se da la elección al usuario que quiere realizar, si quiere manejar manualmente la plataforma móvil (*Modo Manual*) o quiere utilizar el control de desplazamiento (*Modo Semiautomático*).

La figura 3.28 muestra el modo de operación manual de la plataforma móvil, este no tiene la necesidad de datos de distancia a recorrer. La figura 3.29 tiene la ejecución semiautomática, esta se le proporcionan datos de distancia y dirección para el desplazamiento del móvil al dar enviar para la ejecución de la acción.

En cada una de las pantallas mostradas en las figuras 3.27, 3.28 y 3.29 se tienen un botón de salida y en las figuras 3.28 y 3.29 se tiene un botón de regreso para poder retornar al menú principal.

Los comandos que esta aplicación envía a la plataforma móvil están relacionado con la tabla 3.2.

Para el código fuente generado mediante el diagrama mostrado en la figura 3.26 se muestra en el anexo A5 que contiene el nombre de "*APLICACIÓN ANDROID*" para su estudio y revisión.

3.7. Partes complementarias de diseño

En este apartado se trata sobre los elementos utilizados como complemento para el diseño del sistema de control de desplazamiento, estos, se dividen en 3 partes que son:

- Comunicación serial.
- Alimentación.
- Acoplamiento y adquisición de datos del codificador óptico modelo E4P.

Comunicación Serial

Considerado las características del módulo bluetooth modelo RN-24 para comunicarse con dispositivos externos y las cualidades del FPGA, se consideró la implementación de una recepción serial asíncrono para la obtención de los datos recibidos del dispositivo RN-24 mediante comunicación bluetooth del teléfono celular inteligente.

La forma de operación de una comunicación serial asíncrona se basa principalmente en la recepción de pulsos, los cuales tiene un cierto tiempo duración y un cierto orden, este se puede observar en la siguiente imagen.

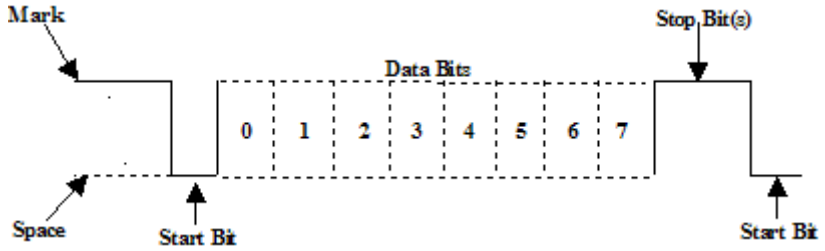


Figura 3.30. Formato de comunicación Serial Asíncrona.

En la figura 3.30 se muestra el formato utilizado tanto para transmisión como recepción de una comunicación del tipo serial asíncrona de 8 bits, utilizado para la comunicación en nuestro sistema de control.

El método que utilizamos para la lectura de datos del protocolo serial es la espera del bit de inicio que ocurre en una transición descendente del pulso; al suceder este evento se determina un tiempo de lectura, el cual, es calculado mediante la velocidad de transmisión que se maneje entre 2, esto, con la finalidad de tomar la lectura en la mitad de cada pulso para tener un dato confiable al terminar de recibir la trama; al terminar de leer los 8 bits se espera el bit de stop para espera un nuevo dato y mostrar el dato de forma paralela y con una señal extra se determina la validez del dato con base en el estado de la señal de comunicación.

Para determina los tiempos de espera en forma de ciclos para la toma de los datos en el formato serial, se calcula mediante la siguiente ecuación:

$$numeros\ de\ ciclos = \frac{1}{\frac{Baud\ rate}{Reloj\ FPGA}} = \frac{1}{\frac{115200}{50x10^6}} = 434\ ciclos \qquad (3.17)$$

En la ecuación 3.17 se toma en cuenta la velocidad de comunicación en bits por segundo y la frecuencia del reloj de la tarjeta de desarrollo.

Otra parte del área de comunicación es en concepto de almacenamiento y distribución de datos recibidos.

Con base en el almacenamiento y distribución de datos, como primer paso se espera la señal que indica que hay un nuevo dato del módulo de comunicación, después, se hace una comparación para verificar el comando identificador con la finalidad de mantener un orden de recepción. Al comprobar el identificador que sea correcto se espera de nuevo el bit de dato nuevo para almacenar y distribuir el dato a los demás componentes y así con el siguiente dato, para después iniciar de nuevo el proceso.

Al terminar el proceso de almacenamiento y distribución de datos se activa una señal, con la finalidad de reiniciar las mediciones de los codificadores de cuadratura, independientemente del comando de dirección recibido. La secuencia como se envía los datos se menciona en la estructura de diseño del sistema de control.

Con base en la explicación anterior las figuras 3.31, 3.32 y 3.33 muestran la implementación del sistema de comunicación para el control de desplazamiento en FPGA mediante lenguaje VHDL.

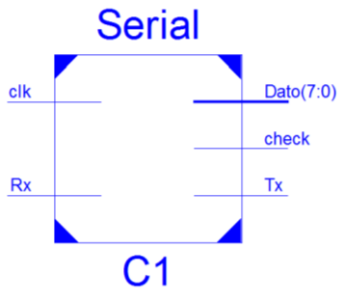


Figura 3.31. Esquemático de la interfaz de comunicación serial.

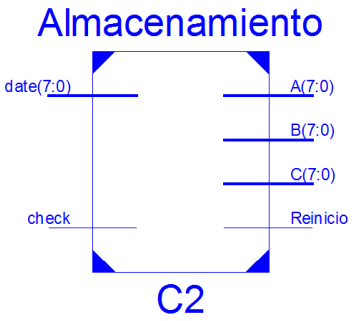


Figura 3.32. Esquemático del módulo de almacenamiento.

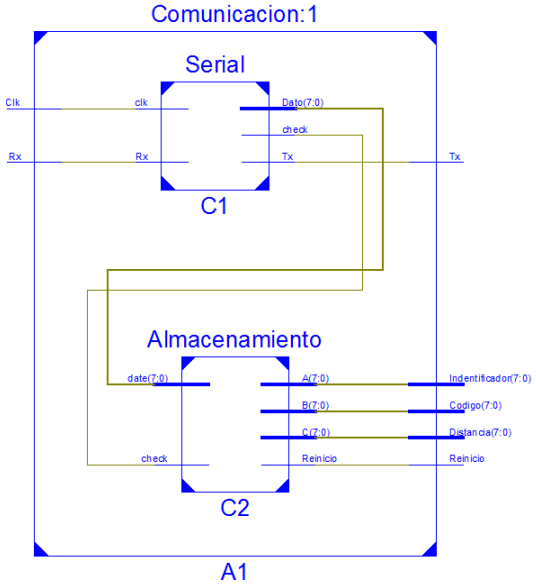


Figura 3.33. Esquemático del módulo de comunicación del sistema de control.

En la figura 3.30 se muestra la implementación de la comunicación serial, mientras, en la figura 3.31 se muestra el modulo obtenido del bloque de almacenamiento de los datos; por último, la figura 3.32 muestra el bloque de comunicación que es incluido en el sistema de control, en el cual, se encuentra la comunicación serial y el almacenamiento.

Los códigos fuentes del módulo de comunicación serial mediante descripción de hardware se muestra en el anexo A6 con el nombre de "*COMUNICACIÓN SERIAL*" al final de este escrito; esto incluye el código de comunicación serial, almacenamiento e interconexión entre estas partes.

Alimentación

El diseño del módulo de alimentación para el sistema electrónico que contiene la plataforma móvil se hace la consideración que el medio de alimentación sea constante y portátil, para eso, se usan los siguientes elementos:

- Batería sellada de 12V 34 [Ah].
- Regulador LM309K.
- Capacitores de 1 y 2.2 [µF].

Esto para poder tener un sistema portátil de alimentación para el desplazamiento de la plataforma móvil y poder utilizarlo en espacios más amplios para pruebas y desarrollos de nuevos diseño.

El regulador LM309K tiene como especificaciones eléctricas una entrada de voltaje de 6 a 38 [V], mientras, la corriente de salida tiene como máxima de 3 [A] con un voltaje de salida de 5.20 [V].

Para la alimentación de los motores de DC se realiza directamente desde la batería, mientras, la parte electrónica del sistema se realiza mediante el regulador (LM309K), esto, por cuestiones de alimentación del hardware.

El diagrama de conexión del regulador de voltaje se muestra en la siguiente figura.

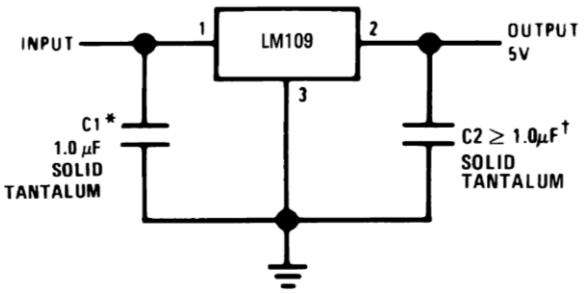


Figura 3.34. Conexión del regulador de voltaje LM109 (homólogo del LM309K).

Como resultado se obtuvo la finalidad plateada para el desarrollo de este sistema, el cual, consiste en la movilidad en la plataforma móvil y su flexibilidad de uso.

En el anexo A7 con el nombre de "*ESQUEMA DE ALIMENTACIÓN*" contiene el esquema general del módulo de alimentación utilizado para la plataforma móvil omnidireccional.

Acondicionamiento y adquisición de datos del sensor modelo E4P

El sensor modelo E4P es un codificador óptico de cuadratura, que mediante 2 canales puede identificar la dirección de giro, aparte de facilitar la detección de números de pulsos que genera.

En la parte de acondicionamiento, el sensor tiene un voltaje de salida de 5 [V], para lo cual, acoplarlo a las entradas que contiene la *Spartan-3E FPGA Starter Kit Board*, que soportan una entrada nominal de voltaje de 3.3 [V] se plantea el siguiente esquema que se presenta a continuación:

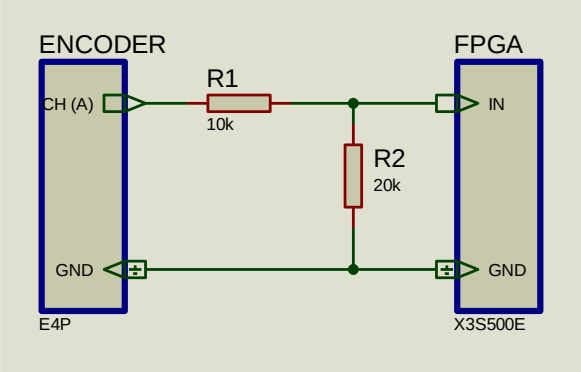


Figura 3.35. Esquema de conexión de acoplamiento entre el sensor y el FPGA.

La forma de acondicionar mostrada en la figura 3.35 se basa en un diviso de voltaje, el cual, reparte la cantidad de voltaje entre sus 2 resistencia, permitiéndonos utilizar la señal de salida del sensor con niveles aceptables al FPGA.

La siguiente ecuación se modela un sistema de división de voltaje que se utilizó para obtener estos valores de resistencia.

$$V_{OUT} = V_{IN}\frac{R_2}{R_1+R_2} = 5V\ \frac{20K\Omega}{30\ K\Omega} = 3.3V \tag{3.18}$$

Como se muestra en la ecuación 3.18, se calcula los valores de cada resistencia para el acondicionamiento de la señal de un voltaje de 5 [V] a un voltaje de 3.3 [V] para ser procesada y utilizada para medición.

Proseguimos en la parte de traducir los pulsos a una medida que nos sea de utilidad, para ello se utiliza la siguiente ecuación.

$$S = \frac{R*2\pi}{PPV} * C \tag{3.19}$$

Donde:

- S: Distancia recorrida.
- R: Radio de la rueda.
- PPV: Pulsos por vuelta.
- C: Pulsos Recorridos

Considerando las constante de la ecuación 3.19 se replantea para su utilización, esto, se refleja a continuación.

$$S = SL * C \quad donde \;\; SL = \frac{R*2\pi}{PPV} \tag{3.20}$$

Para el diseño del módulo para la medición de distancia, utilizamos la ecuación 3.20 entre otras características que se desglosan a continuación:

- Comparación de las señales de entrada: en esta parte se toman los valores de los canales A y B del sensor para determinar si es '0', '1' o el valor anterior de la medición.
- Evento de rotación: en esta parte se compara un evento anterior y actual de lo obtenido en la comparación de señales de entrada, con el fin de saber si se llevó a cabo un pulso para el conteo.
- En esta última parte se utiliza la ecuación 3.20 para saber la distancia que recorre la rueda, al igual de una señal para el reinicio de la cuenta de la medición.

Respecto a lo descrito anteriormente se implementó una unidad de medición para un sensor codificador de cuadratura, esto en el lenguaje VHDL y en el software de *Xilinx ISE 14.6*, el cual, nos arroga el siguiente esquemático.

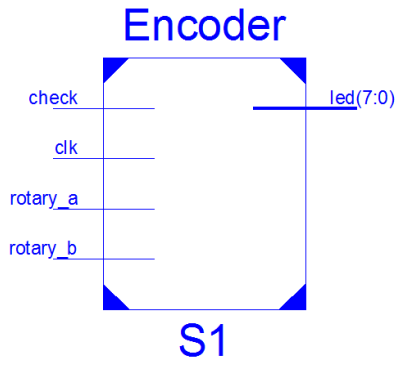


Figura 3.36. Esquema del sistema de medición para el sensor modelo E4P.

Como se observa en la figura 3.36, el bloque cuenta con una salida de reloj, el reinicio de cuenta y los dos canales que cuenta el sensor como entrada, mientras, en la salida cuenta con una señal de 8 bits.

Mediante al esquema se crea un bloque de medición, el cual, incluye cuatro módulos de medición de los sensores, para su integración con el sistema de control de desplazamiento.

Los códigos de implementación del esquema mostrado en las figuras 3.36 y el armado del módulo de medición se muestran en el anexo A8 llamado "*ADQUISICIÓN*" para su consulta.

En el anexo A9 llamado "*SISTEMA DE CONTROL*" se muestra tanto el código relacionado a la interconexión y el esquema resultante con base en los diseños mostrados en este apartado.

Capítulo 4

PRUEBAS Y RESULTADOS EXPERIMENTALES

Las pruebas hechas al control de desplazamiento son aplicadas principalmente en los términos de sentido de giro de las ruedas, distancia recorrida y dirección del móvil en el desplazamiento. Estas pruebas se llevaron a cabo con la plataforma móvil suspendida en el aire y en piso.

Las pruebas hechas al sentido de giro de las ruedas para el direccionamiento de la plataforma móvil, son llevadas con la finalidad de ver si cumplen con lo establecido en el diseño de la plataforma, cuando está suspendida y en pisos, con la finalidad de observar si llevan a cabo el direccionamiento de la plataforma de forma general. Los resultados obtenidos son el correcto funcionamiento de sentido de los giros del motor para el direccionamiento de la plataforma, aparte, de cumplir con las direcciones definidas para el diseño de control (enfrente, reversa, laterales y diagonales), aunque, se calibro para que la desviación por factores externos (ruedas, juegos de engranes y piso) en la direcciona de la plataforma sea mínima.

Las pruebas hechas para la distancia que la plataforma móvil recorre son con la finalidad de poder determinar el grado de error que esta pueda tener en dos tipos de distancias, 50 y 100 cm, con 8 repeticiones hechas en piso, mientras suspendida es de corroboración de funcionamiento de los codificadores incrementales de cuadratura. En la parte de verificación del funcionamiento de los sensores se tuvo problemas con el acoplamiento al eje de la rueda, los cuales, fueron solucionados para su correcto funcionamiento.

La elección de las distancias y numero de repeticiones consideradas para las pruebas del controlador son tomadas con respecto al desarrollo del proyecto como las distancias tomadas en algunos sistemas de navegación para sus pruebas [14].

Los resultados obtenidos se muestran en las siguientes tablas.

Tabla 4.1. Valores obtenidos de desplazamiento frontal de plataforma móvil.

Desplazamiento frontal de plataforma móvil omnidireccional		
Sucesiones	**50 cm**	**100 cm**
1	49.70 cm	99.15 cm
2	49.30 cm	98.20 cm
3	49.40 cm	99.45 cm
4	49.90 cm	100.40 cm
5	50.70 cm	100.40 cm

6	49.75 cm	101.00 cm
7	51.00 cm	100.70 cm
8	50.40 cm	100.20 cm
Promedio	50.01 cm	99.93 cm
Error promedio	± .01 cm	± .07 cm
Error porcentual	.02 %	.07 %

Tabla 4.2. Valores obtenidos de desplazamiento de la plataforma móvil de reversa.

Desplazamiento de plataforma móvil omnidireccional de reversa		
Sucesiones	**50 cm**	**100 cm**
1	49.10 cm	99.80 cm
2	50.80 cm	99.35 cm
3	49.30 cm	100.30 cm
4	50.20 cm	100.30 cm
5	50.60 cm	101.00 cm
6	50.10 cm	100.90 cm
7	50.30 cm	100.29 cm
8	49.40 cm	100.50 cm
Promedio	49.97 cm	100.30 cm
Error promedio	± .03 cm	± .3 cm
Error porcentual	.06 %	.3 %

En las siguientes graficas se muestran los resultados que se tiene en las tablas, aparte la de error absoluto que se tiene con cada medición con respecto a la medición deseada.

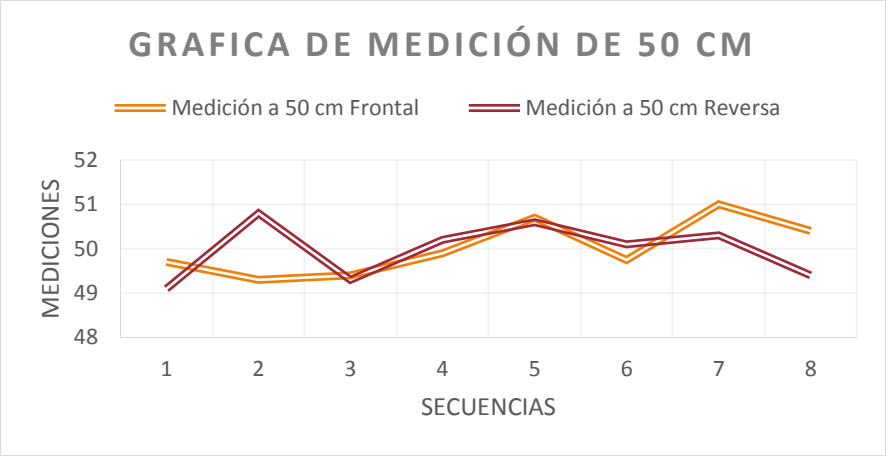


Figura 4.1. Grafica de mediciones hechas para las prueba de 50 cm.

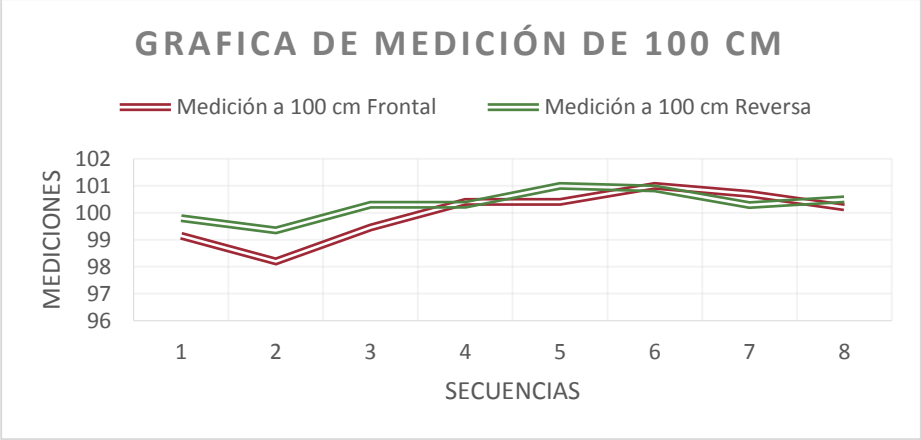


Figura 4.2. Grafica de mediciones hechas para las prueba de 100 cm.

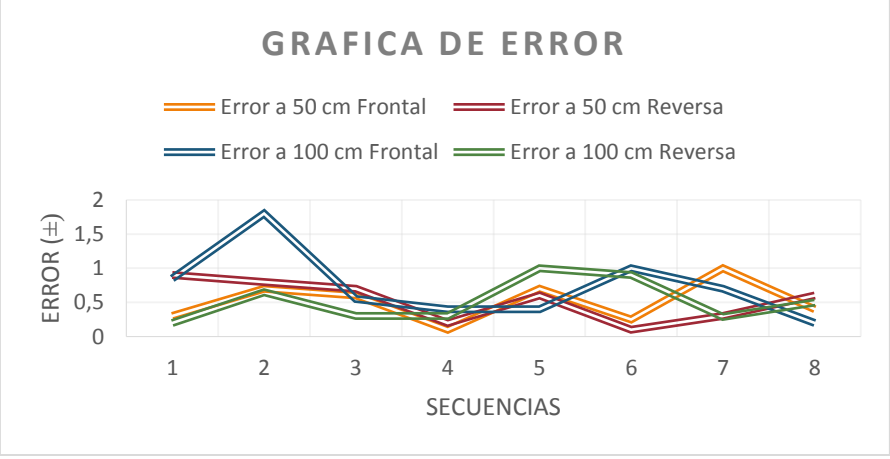


Figura 4.3. Grados de error de las mediciones hechas.

Estas mediciones pueden tener variaciones con respecto al piso en donde se realicen las pruebas, lo cual está ligado a grietas, desniveles entre otros.

Capítulo 5

CONCLUSIONES

Con respecto al estudio, elección, desarrollo e implementación del sistema, se elabora una evaluación de los objetivos planteados con los objetivos alcanzados en este trabajo.

En el sistema de control propuesto para el desplazamiento de la plataforma móvil omnidireccional se puede concluir en el cumplimiento del control de movimiento en diferentes direcciones (8 direcciones diferentes), con un error máximo de .3% en un desplazamiento a 100 cm y .06% para 50cm con 8 repeticiones en 2 de las 8 direcciones controladas considerando el sistema de control utilizado, en donde, no es necesario un modelo matemático para su diseño se puede decir de su fiabilidad en comparación a otras técnicas de control, las cuales, necesitan un modelo matemático del sistema donde se implementa, por lo cual, en su aplicación práctica suelen tener un margen de error mayor por diferencias entre el modelo matemático con el modelo físico.

En la parte de aplicación de *Lógica Difusa* en FPGA se obtuvo una forma de implementación sencilla, código reducido y fácil de implementar tanto en FPGA como en otros dispositivos con limitaciones; en este caso muestra una alta fidelidad de resultados con el mismo sistema difuso aplicado en software y sin la necesidad de ocupar mucho espacio en hardware en comparación a otras forma de implementación, [5], [14], [52], [53] y [54], haciendo de esta forma de implementación de utilidad para ser utilizadas en sistemas de mayor tamaño y procesamiento de datos.

En el sistema de comunicación mediante Bluetooth entre la plataforma móvil y el teléfono celular, por medio de una aplicación desarrollada para SO Android, se alcanzó el objetivo de poder enviar datos al sistema de control para obtener una acción de la plataforma como se había propuesto, al igual, se encontraron forma de utilizar nuevas tecnologías para diversos proyectos y sean de utilidad para mejorar como sacar mejores resultados de diversos proyectos donde sea utilizable las cualidades de los teléfonos inteligentes.

En cuestión de la plataforma móvil omnidireccional, se concluye que es una de las plataformas que tiene mayor libertad de movimiento, en comparación a otras configuraciones mecánicas básicas, al mismo tiempo conlleva a una mayor dificultad en la aplicación de sistemas de control, por lo cual, en estos momentos es menos utilizada en comparación de otros tipos de plataforma, que en años siguientes pueden cambiar en gran mediad gracias a aplicaciones de técnicas de control con inteligencia artificial.

5.1. Trabajo futuro

Con base en los conocimientos obtenidos en el desarrollo e implementación de plataformas móviles, los trabajos plateados para el seguimiento y desarrollo de una línea de investigación sólida y de impacto social al considerar los avances de la tecnología en la actualidad se ponen en análisis los siguientes temas de investigación y lo que se pretende obtener de ellos.

Sistema de reconocimiento y localización de objetos con propósitos de ubicación y desplazamiento en robótica omnidireccional: En este tema se pretende el estudio de reconocimiento mediante a contornos y profundidad de objetos por medio de inteligencia artificial, a su vez es implementado en un hardware programable (GPU o FPGA) y estimar su localización en un plano, esto con el uso de cámaras aparte de la utilización de una plataforma omnidireccional, la cual, se desplace a la posición que se encuentre el objeto buscado; esto con la finalidad de ser utilizado en robótica de servicio ara la asistencia del ser humano.

Sistema de navegación basado en hardware para robots omnidireccionales: Este tema pretende el desarrollo de un sistema de navegación mediante inteligencia artificial para el desplazamiento de una plataforma móvil omnidireccional en un área desconocida mediante el uso de cámaras para su ubicación y su trazado de rutas de un punto de inicio a un punto de llegada designado, todo esto aplicado a un dispositivo programable (GPU o FPGA) y un sistema de evasión de objetos.

BIBLIOGRAFÍA

[1] Danilo Navarro, Gines Benet Gilabert, Luis Hernando Ríos, Maximiliano Bueno L., "*MEJORAS DE LA LOCALIZACION ODOMÉTRICA DE UN ROBOT DIFERENCIAL MEDIANTE LA CORRECCION DE ERRORES SISTEMATICOS",* Universidad Tecnológica de Pereira, Pereira Colombia 2007.

[2] Iñaki Navarro Oiza, "*Localización de un Robot Móvil con el Filtro Extendido de Kalman",* 14 de julio de 2005.

[3] Francisco Javier Blanco Rodríguez, "*CÁLCULO DEL C-ESPACIO DE UN ROBOT MÓVIL: CONVOLUCIÓN JERÁRQUICA*", UNIVERSIDA DE SALAMANCA, Abril 2003.

[4] López R., Leonardo J., Franco m., Zulay E., Pateti M., Antonio S.,"*METODOLOGÍA DE IMPLEMENTACIÓN DE UN CONTROLADOR PID DIFUSOEN UN FPGA*", Universidad Nacional Experimental Politécnica "Antonio José de Sucre", Julio 2006.

[5] V. H. Grisales, J. E. Bonilla, M. A. Melgrado, "*DISEÑO E IMPLEMENTACION DE UN CONTROLADOR DIFUSO BASADO EN FPGA*", Universidad Distrital José de Caldas, Santafé de Bogotá, Colombia.

[6]Nelson David Muños, Carlos Andrés Andrade, Nelson Londoño Ospina, "*Diseño y construcción de un robot móvil orientado a la enseñanza e investigación*", Universidad de Antioquia, Colombia, Enero-Junio 2006.

[7]Francisco J. Arjonilla García, "*Desarrollo e implementación de plataforma móvil en entorno distribuido*", UPM, 18 de octubre de 2011.

[8]Zheng Wang, Yuqing He, Jianda Han, "*Simultaneous Locating and Calibrating Pseudolite Navigation System for Autonomous Mobile Robots*", Chinese Academy of Sciences, septiembre 2009.

[9]M.A. Zamora, L.M. Tomás-Balibrea, H. Martínez, A.G. Skarmeta, "*Navegación Planificada de un Robot Móvil en Entornos Interiores Desconocidos*", Universidad de Murcia.

[10]Jhonny A. Valencia V., Alejandro Montoya O., Luis Hernando Ríos, "*Modelo Cinemático de un Robot Móvil tipo Diferencial y de Navegación a partir de la Estimación Odométrica*", Universidad Tecnológica de Pereira, Mayo de 2009.

[11]Hernando Parra, Luis Hernando Ríos, Maximiliano Bueno L.," *Navegación de Robots Móviles Mediante Comportamiento Utilizando Lógica Difusa*", Universidad Tecnológica de Pereira, Mayo de 2007.

[12]Wei-Song Lin, Ming-Kang Chuang, Glorious Tien, "*Autonomous Mobile Robot Navigation Using Stereovision*", Department of Electrical Engineering National Taiwan University, July 2005.

[13]Le-Jie Zhang, Zeng-Guang Hou, Min TAN, "*Kalman Filter and Vision Localization Based Potential Field Method for Autonomous Mobile Robots"*, the Chinese of Sciences, July 2005.

[14]Belmar García García, "*Diseño y construcción de un robot móvil controlado con técnicas de lógica difusa implementadas en un FPGA*", Instituto Politécnico Nacional, septiembre 2012.

[15]Adriana Riveros Guevara, Cindy Natalia Salas López, Leonardo Solaque Guzmán, *"Aproximación a la Navegación Autónoma de una Plataforma Móvil, Mediante Visión Estereoscópica Artificial*", Universidad Militar Nueva Granada, Bogotá, Colombia, Diciembre 2012.

[16]Jonathan Eduardo Cruz Ortiz, Carlos Andrés Vásquez Torrez,"*Sistema de Navegación Autónoma para la Plataforma Robótica móvil (DANI) del Grupo de Investigación ROMA Basado en Métodos De Control Reactivo*", Universidad Distrital Francisco José de Caldas, Bogotá, Enero 2012.

[17]Borrero Guerrero H., Delgado Rivera A. *"Evolución de chip ADN emulado con algoritmo genético en FPGA para el control de navegación de un robot móvil*", Universidad Nacional de Colombia, Mayo 2008.

[18]C.G. Rusu, I.T. Birou, E. Szöke, "*Fuzzy Based Obstacle Avoidance System for Autonomous Mobile Robot*", Technical University of Cluj-Napoca.

[19]Narváez Tipantaxi Victor Julio, Yandún Narváez Francisco Javier, *"Diseño e Implementación de un Sistema de Localización y Mapeo Simultaneo (SLAM) para la Plataforma Robótica ROBOTINO*", Escuela Politécnica Nacional, Quito, Junio 2013.

[20]Seo-Yeon Hwang, Joong-Tae Park, Jae-Bok Song, "*Autonomous Navigation of a Mobile Robot Using an Upward-Looking Camera and Sonar Sensors*", IEEE, 2010.

[21] María del Roció Ortiz Albarrán, "*Implementación de Modelos Ocultos de Markov en la navegación de un robot móvil autónomo*", Universidad Nacional Autónoma de México, 2011.

[22] Thomas Bräunl, "*Embedded Robotics, Mobile Robot Design and Application with Embedded systems*", Springer, tercera edición, 2008.

[23] Ian Grout, "*Digital Systems Design with FPGAs and CPLDs*", ELSEVIER, 2008.

[24] Donald G. Bailey, "*DESIGN FOR EMBEDDED IMAGEN PROCESSING ON FPGAS*", IEEE, Jhon Whiley & Sons, 2011.

[25] Bob Zeidman, "*Designing with FPGAs and CPLDs*", CMP Books, 2002.

[26] Pong P. Chu, "*FPGA PROTOTYPING BY VERILOG EXAMPLES*", Jhon Wiley & Sons, 2008.

[27] Eduardo Augusto Bezerra, Djones Vinicius Lettnin, "*Synthesizable VHDL Design for FPGAs*", Springer, 2014.

[28] Autor anónimo, "*Una Introducción a los Robots Móviles*", 2008.

[29] David G. Maxinez, Jessica Alcalá, "*VHDL El arte de programar sistemas digitales*", Tec de Monterrey, CECSA, 2002.

[30] Fernando Pardo, Jose A. Boluda, "*VHDL Lenguaje para síntesis y modelado de circuitos*", Ra-Ma, 2011.

[31] Kourosh Kalantar-zadeh, "*Sensors An Introductory Course*", Springer, 2013.

[32] Syed Kamrul Islam, Mohammad Rafiqul Haider, "*Sensors and Low Power Signal Processing*", Springer, 2010.

[33] H.R. Everett, "*Sensors for Mobile Robots Theory and Application*", A K Peters, Ltd. Wellesley, Massachusetts, 1995.

[34] Ramón Pallás Areny, "*SENSORES y ACONDICIONADORES de SEÑAL*", cuarta edición, marcombo, 2003.

[35] http://ketai.googlecode.com/svn/trunk/ketai/reference/ketai/net/bluetooth/KetaiBluetoo th.html

[36] Floriberto Ortiz Rodríguez, "*Modelo y control de PD-difuso en tiempo real para el sistema barra-esfera*", centro de investigaciones y estudios avanzados del instituto politécnico nacional, octubre 2004.

[37] Kwang H.Lee, "First Course on Fuzzy Theory and Applications", Springer, 2005.

[38] Garcia Ortega Marco Antonio, "*Diseño de un controlador difuso de velocidad en un conjunto motor-generado de CD*", Universidad Autónoma de Estado de Hidalgo, 2006.

[39] Pedro Ponce Cruz, "INTELIGENCIA ARTIFICIAL CON APLICACIONES A LA INGENIERÍA", Alfaomega, México, julio 2010.

[40] Kevin M.Passino, Stephen Yurkovich, "*Fuzzy Control*", ADDISON-WESLEY, 1998.

[41] Ying Bai, Hanqi Zhuang, Dali Wang, "*Advances in Industrial Control*", Springer, 2006.

[42] Roving networks, "*RN-24/ RN-25*", California, Agosto 2010.

[43] Xilinx, "*Spartan-3E FPGA Starter Kit Board User Guide*", Versión 1.2, 20 de Enero 2011.

[44] Cross the Road Electronics, LLC, "*Talon and Talon SR User Manual*", Versión 1.3, 4 de Febrero 2013.

[45] US DIGITAL, "E4P OEM Miniature Optical Kit Encoder", Washington USA, 16 de Enero 2012.

[46]NEXUS robot, "*4WD 100mm Mecanum wheel robot kit 10011*", Página de internet: www.nexusrobot.com.

[47] AndyMark, "C-Base Frame Wide, Toughbox Mini Gearboxes, Direct Drive, 8" Mecanum Wheels (am-0987)", Página de internet: //www.andymark.com/product-p/am-0987.htm

[48] The MathWorks,"*Fuzzy Logic Toolbox For Use with MATLAB*", Enero 1999.

[49] Timothy J. Ross,"*FUZZY LOGIC WITH ENGINEERING APPLICATIONS*", John Wiley & Sons, Ltd, 2004.

[50] Danicl Sautcr, "*Rapid Android Development Build Rich, Sensor-Based Applications with Processing*", The Pragmatic Bookshelf, Dallas, Texas, 2012.

[51] Casey Reas, Ben Fry, "*Getting Started with Processing*", O'REILLY, 2010.

[52] S. Sánchez Solano, A. Cabrera, C. J. Jiménez, P. Brox, I. Baturone, A. Barriga, "*IMPLEMENTACIÓN SOBRE FPGAS DE SISTEMAS DIFUSOS PROGRAMABLES*", Instituto de Microelectrónica de Sevilla, Centro Nacional de Microelectrónica.

[53] Juan Carlos Herrera Lozano, "*IMPLEMENTACIÓN DE UN CONTROLADOR DIFUSO EN UN FPGA, UTILIZANDO LENGUAJE DE DESCRIPCIÓN DE HARDWARE*", Centro de investigaciones de computación, instituto politécnico nacional, México 2002.

[54] Domingo José Gómez Meléndez, "*SISTEMA DE CONTROL BORROSO PARA RIEGO LOCALIZADO BASADO EN ARREGLOS DE COMPUERTAS PROGRAMABLES EN CAMPO*", Universidad Autónoma de Querétaro, Facultad de Ingeniería, Santiago de Querétaro Abril 2008.

Anexo A1

PLATAFORMA MÓVIL OMNIDIRECCIONAL

Para el armado de la plataforma móvil omnidireccional, es requerido del el armado de varias partes (Juego de engranes con el actuador, ruedas y cuerpo de la plataforma) para su funcionamiento, por esto, en este anexo hablamos de forma detallada de los procedimientos seguidos para el armado de todas las mecánicas involucradas para el movimiento y buena operación del móvil omnidireccional.

En el paquete de AndyMark para el armado del Toughbox Mini; contiene los siguientes elementos:

- Una mini placa para el eje.
- Una mini tapa.
- Una barra pequeña del eje hexagonal.
- Un eje hexagonal de salida.
- 4 10-32 tuercas con retención plástica.
- 4 10-32 X 3/4” tornillo largo de cabeza hueca.
- Una roldanas de 1/4 de pulgadas con .63 pulgadas de diámetro de orificio.
- Un tornillo de cabeza larga hueca de 1/4-20 x 1/2".
- Un clip externo de 1/2 pulgada para eje.
- 2 clips retenedores de 8mm.
- 2 baleros blindados de 3/8 de pulgada.
- Un balero con pestaña blindado de 3/8 de pulgada.
- Un balero con pestaña blindado de 1/2 pulgada.
- 2 engranes CIM de 14 dientes.
- Una barra aseguradora de acero de 1/8 X 1/8 X 0700.
- Una barra aseguradora de acero de 2 X 2 X 10mm.
- Un pequeño engrane de 14 dientes.
- Un engrane de 50 dientes.
- Un engrande de salida de 50 dientes.

Con estos elementos, a continuación explicamos los pasos para el armado del juego de engranes encargados del toque.

Como primer paso colocamos el balero de 3/8 y 1/2 con pestaña en la mini placa del eje, como se muestra en la figura A.1.

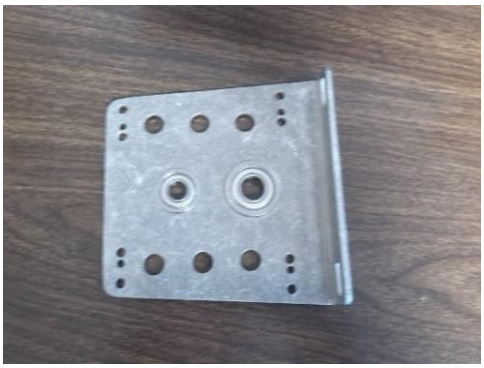


Figura A.1. Mini placa del eje con los baleros de pestaña de 1/2 y 3/8.

Después, proseguimos con la colocación de los baleros de 1/2 normales en la tapa del *Toughbox Mini* como se observa en la figura A.2.

Figura A.2. La Mini tapa del *Toughbox Mini* con los baleros de 1/2.

Después, colocamos la barra pequeña del eje hexagonal en el balero que se encuentra en la tapa, ubicado en el orificio descubierto, como se presenta en la figura A.3.

Figura A.3. La Mini tapa del *Toughbox Mini* con la pequeña barra del eje hexagonal colocado en el balero.

El siguiente paso es colocar el engrane de 50 dientes en la barra pequeña del eje hexagonal con su saliente circular hacia abajo contra el rodamiento, como se puede apreciar en la figura A.4.

Figura A.4. La Mini tapa del *Toughbox Mini* con el engrane de 50 dientes colocado en la pequeña barra del eje hexagonal.

Con lo observado en la figura A.4, pasamos a colocar el pequeño engrane de 14 dientes en la pequeña barra del eje hexagonal; sobre el engrane de 50 dientes antes puesto, el cual se aprecia en la figura A.5.

Figura A.5. Pequeño engrane de 14 dientes puesto en la pequeña barra del eje hexagonal y sobre el engrane de 50 dientes.

Al terminar de poner el pequeño engrane de 14 dientes, pasamos a eje hexagonal de salida, que cuenta con una ranura en la parte hexagonal, en la cual, pondremos un seguro delimitador como se ve en la figura A.6.

Figura A.6. Eje hexagonal de salida con seguro delimitador.

Al tener completo el eje hexagonal de salida, proseguimos en colocarlo en el balero que se encuentra libre en la mini tapa *Toughbox Mini* que se ejemplifica en la figura A.7.

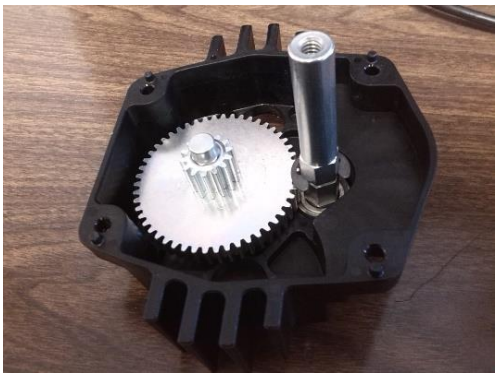


Figura A.7. Mini tapa *Toughbox Mini* con eje hexagonal de salida.

Teniendo el eje de salida en la mini tapa *Toughbox Mini,* colocamos el engrane de salida en el eje de salida, encontrado este con el engrane pequeño de 14 dientes, esto se puede ver en la imagen presentada en la figura A.8.

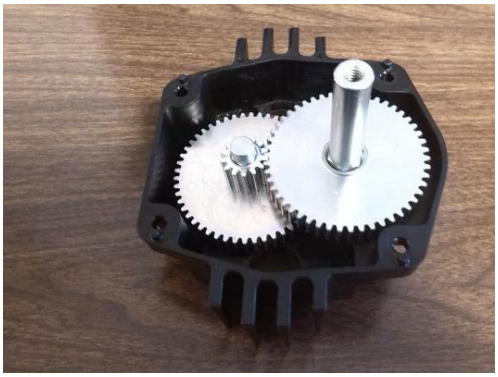


Figura A.8. Engrane de salida puesto en el reto del conjunto de juego de engranes.

Con el juego de engranes completo, se realiza el engrasado de estas partes, el cubrimiento de la mini tapa con la mini placa para el eje y sujete ambas parte con las tuercas con retención plástica y los tornillos, como es ilustra en la figura A.9.

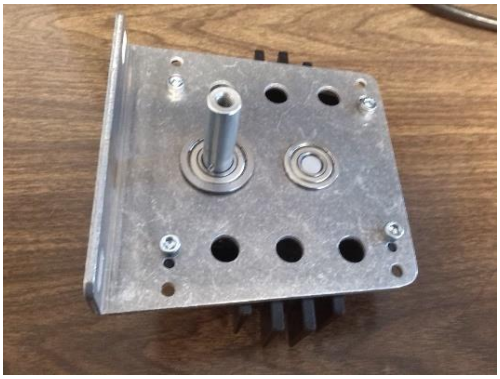


Figura A.9. Unión de la tapa con la placa del eje y sujeción con las tuercas y tornillos con retención plástica.

Para la colocación de las tuercas y tornillos con retención plástica, la tapa cuenta con unos orificios con forma hexagonal para la colocación de las tuercas y en la parte de la placa, solamente se tiene que verificar en qué punto concuerdan los orificios de las tuercas con los orificios de la placa para la sujeción.

Con esto finalizamos el armado del *Toughbox Mini* que nos sirve para el toque de cada motor.

Este juego de engranes (*Toughbox Mini*) está hecho para ser acoplado con 2 motores de DC; en este caso, se utilizara con un *motor 2.5" CIM*.

En el siguiente tema hablamos del acoplamiento del motor al *Toughbox Mini* para la cuestión de fuerza.

Acoplamiento de motores de DC 2.5 CIM a *Toughbox Mini*

Para este acoplamiento se utilizó los siguientes elementos:

- Un motor *2.5" CIM.*
- Un engrane CIM de 14 dientes.
- 2 roldanas de 1/4 de pulgadas con .63 pulgadas de diámetro de orificio.
- 2 tornillo de cabeza larga hueca de 1/4-20 x 1/2".
- Una barra aseguradora de acero de 2 X 2 X 10mm.
- Un clip retenedor de 8mm.

Lo primero paran realizar el acoplamiento es la colocación de las 2 roldanas en la flecha del motor como se muestra en la figura A.10.

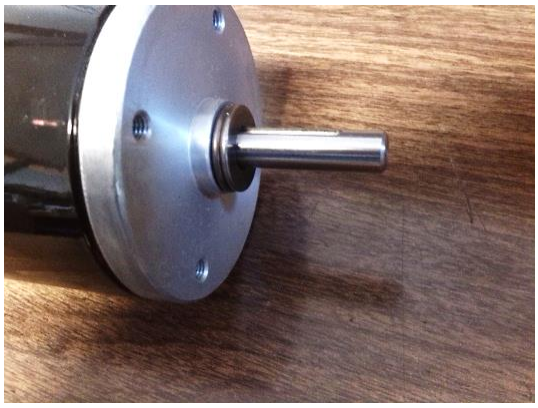


Figura A.10. Flecha del motor con las 2 roldanas.

Ya que tenemos las 2 roldanas en la flecha, pasamos a colocar la barra seguradora en la pequeña ranura que se encuentra en la flecha del motor, como se puede ver en la figura A.11.

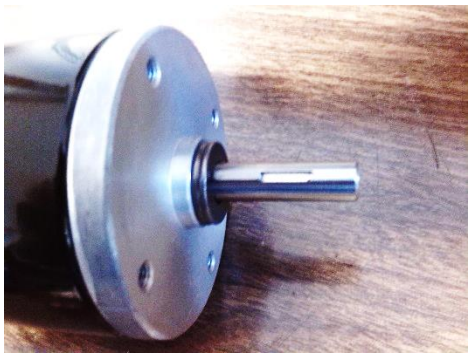


Figura A.11. Ranura de la flecha del motor y barra aseguradora.

Ahora pasamos a colocar el engrane CIM, el cual, cuenta con una ranura; esta se hará coincidir con la ranura donde está la barra y se colocara al último el clip asegurador de 8[mm]. Este paso se puede apreciar en la figura A.12.

Figura A.12. Motor con engrane CIM y clip retenedor.

Por último, introducimos la flecha del motor a uno de los 2 grandes orificios que cuenta el *Toughbox Mini* y lo unimos a él con 2 tornillos huecos, esto se puede ver en la figura A.13.

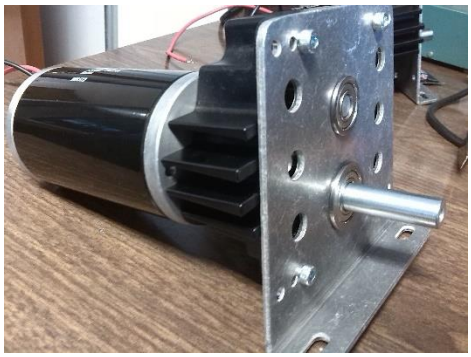


Figura A.13. Motor y *Toughbox Mini*.

Con este procedimiento, incluimos la parte del motor con la parte de torque para el movimiento de la plataforma omnidireccional.

En los siguientes temas, se habla sobre la construcción de la rueda omnidireccional y la integración de la misma al mecanismo que tenemos armado con los procedimientos anteriores.

Armado de ruedas omnidireccionales de 8 pulgadas

Para el armado de las ruedas omnidireccionales se utiliza los siguientes elementos:

- 2 placas laterales (Derecha o Izquierda).
- 12 rodillos con núcleo negro y 2 cojinetes.
- 12 tubos de latón.
- Un espaciador modelo 1600.
- 18 tornillos largos de #10-32x3.0 pulgadas.
- 18 tuercas con sujeción plástica de #10-32.
- 24 roldanas de 1/4 de pulgadas con .04 pulgadas de orificio.
- Un buje de conexión modelo 500.

Este material es para el armado de una sola rueda, para el armado de las 4 ruedas se debe de tener 4 veces el material antes mencionado excepto en las placas laterales donde 4 de ellas deben de ser derechas y las otras 4 izquierdas por la configuración del móvil.

Como primer paso es colocar una de las placas laterales con la cara interior (Indicado con una etiqueta en la placa) hacia arriba en una mesa. Colocamos 6 tornillos en los 6 orificios cercanos al orificio central de la placa desde abajo hacia arriba como se muestra en la figura A.14.

Figura A.14. Placa lateral y tornillos desde los orificios más cercanos al orificio central.

Se debe de mencionar que las placas utilizadas deben de ser del mismo tipo (Derechas o Izquierdas).

Pasamos a colocar el espaciador en la paca lateral a través de los 6 tornillos que están colocados en los 6 agujeros como se ilustra en la figura A.15.

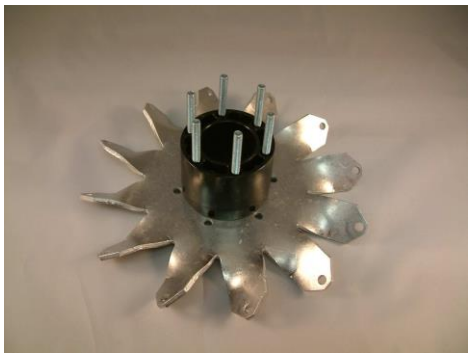


Figura A.15. Placa lateral con espaciador.

Ya que tenemos el espaciador en su lugar, pasamos a colocar la otra placa lateral en la parte superior, la cual su "cara interior" debe de encontrarse con su homólogo de la placa que se encuentra en la parte inferior; colocamos las tuercas sin apretar en cada uno de los tornillos como se presenta en la figura A.16.

Figura A.16. Segunda placa lateral colocada y las tuercas con los tornillos centrales.

Después armamos los rodillos, en el que cada rodillo insertamos los tubos de latón, esto se puede ver en la figura A.17.

Figura A.17. Rodillo con el tubo de latón.

Ya que tenemos los rodillos listos, pasamos a colocarlos en la estructura de la rueda, para ello, en cada extremo del rodillo colocaremos una roldana, después introduciremos el rodillo en las partes perimetrales formadas por las placas laterales y donde se encuentran los orificios en partes metálicas con un ángulo de

45º. Pasamos a alinear los agujeros de las placas con los del rodillo e introducimos el tornillo que atraviese el rodillo desde los orificios de las placas y al último colocamos la tureca y sujetamos.

Lo antes mencionado se repita para el resto de rodillos, el resultado final se muestra en la figura A.18.

Figura A.18. Resultado final de colocación de rodillos.

Ahora que hemos colocado todos los rodillos, retiramos las tuercas y colocamos el buje de conexión, estos se puede observar en la figura A.19.

Figura A.19. Colocación de Buje de conexión a la rueda.

Ya que hemos hecho ese paso solo queda poner las tuercas y sujetarlas, de modo equilibrado.

En la siguiente sección, se tratara la colocación de la rueda en el juego de engranes para el torque (*Toughbox Mini*).

Este procedimiento es para tanto armar las ruedas con placas laterales con cara derecha como con cara izquierda.

Este procedimiento difiere un poco al manual presentado por el vendedor por la cuestión de la inclusión del buje de conexione a la rueda.

Acoplamiento de las ruedas al *Toughbox Mini*

Para el acoplamiento de la rueda al *Toughbox Mini* se necesitan los siguientes elementos extras:

- Un espaciador de 595x500.
- Una barra aseguradora de acero de 1/8 X 1/8 X 0700.

- Una roldanas de 1/4 de pulgadas con .63 pulgadas de diámetro orificio.
- Un tornillo de cabeza larga de 1/4-20 x 1/2".

Con este tipo de material; conectaremos la parte de la rueda con la parte de toque del motor.

El primer paso para la unión es la colocación del espaciador en la flecha que se muestra en la parte del cuadro de engranes, como se aprecia en la figura A.20.

Figura A.20. Colocación del espaciador a la flecha del *Toughbox Mini.*

Ahora que se colocó el espaciador, proseguimos con la colocación de la barra aseguradora, esta es colocada en la ranura que se encuentra en la flecha y haciendo contacto con el espaciador como se aprecia en la figura A.21.

Figura A.21. Colocación de la barra aseguradora en la ranura de la flecha.

Ya que contamos con la colocación de la barra aseguradora, se realiza la puesta de la llanta; se pone está haciendo coincidir la barra que tiene la flecha con la ranura que tiene el buje de conexión que contiene la rueda omnidireccional; para evitar cualquier falla que pueda tener a causa del movimiento.

Por último pasamos a la sujeción de la rueda con el juego de engranes; esto se realiza con base en atornillar estas con a roldana y tornillo de cabeza larga. Esto se realiza en el orificio que se encuentra en la parte superior de la fleca, la cual entra en el agujero del buje de conexión de la llanta; esto sirve para evitar que salga la llanta de su posición de forma horizontal, esto se puede observar con mayor detalle en la figura A.22 que se presenta a continuación.

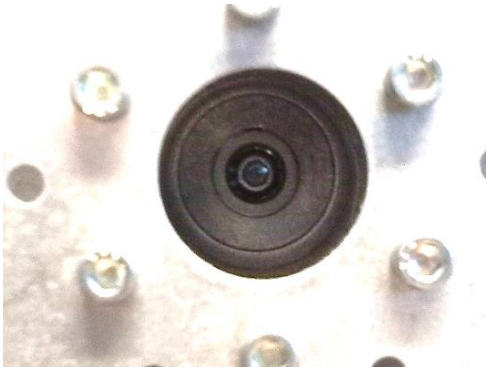


Figura A.22. Unión con la tuerca y roldana de forma horizontal para la conexión de la rueda y el cuadro de engranes.

Y el resultado obtenido de este procedimiento se muestra a continuación.

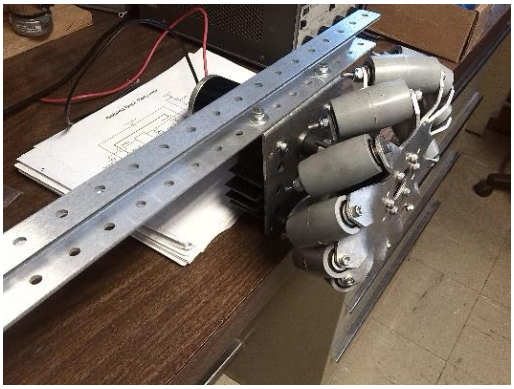


Figura A.23. Terminado de la unión de la rueda con la parte de torque.

Considerando lo hecho hasta ahora, solo resta la colocación de este elemento a la estructura de la plataforma, que en el tema posteriores hablamos de ellos.

Este método debe de ser repetido para cada una de las llantas y juegos de engranes que conformaran la plataforma.

Armado del cuerpo de la plataforma

Para el armado del cuerpo de la plataforma es necesario lo siguiente:

- 6 conectores de esquina.
- 6 canaletas de 35 pulgadas de largo.
- 32 1/4 -20 X 3/4" Tornillo largo de cabeza hueca.
- 40 1/4 -20 tuercas con retención plástica.
- 4 tubos de forma hexagonal tipo 500.
- 8 1/4 -20 X .625" Tornillo largo de cabeza hueca.
- 8 1/4 -20 X 1” Tornillo de cabeza hexagonal.
- Una hoja perforada de policarbonato.

El primer paso para el armado del cuerpo de la plataforma móvil, donde estarán las ruedas omnidireccionales con los *Toughbox Mini* es la decisión de qué tipo de configuración a utilizar. Si una configuración estrecha (37.25” X 27.25”), ancha (26.19” X 37.25”) o una cuadrada (37.25” X 37.25”). En esta ocasión se decidió por una configuración cuadrada para primero experimentos. Esta decisión fue tomada en consideración de obtener conocimientos acerca de esta nueva forma de desplazamiento con ruedas omnidireccionales.

En la figura A.24 se muestran las configuraciones estrecha y ancha que tiene este tipo de plataformas, para ser utilizadas en un futuro para aplicaciones específicas.

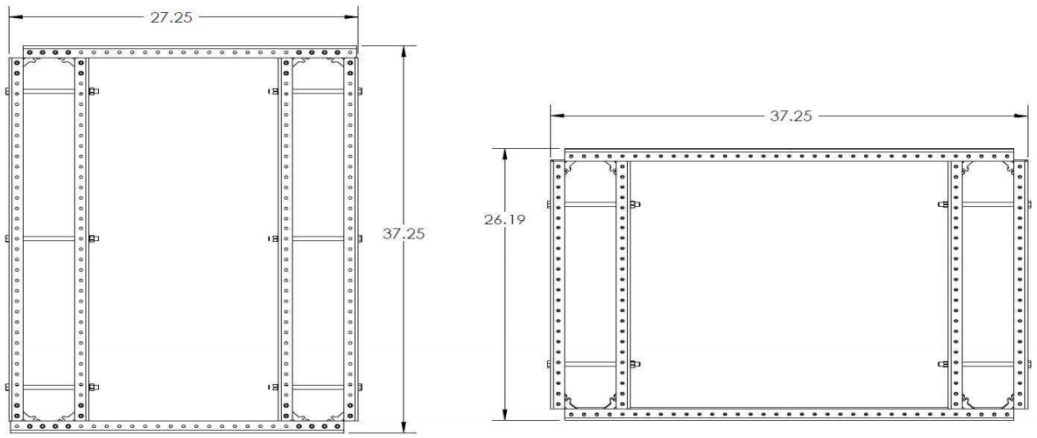


Figura A.24. Plataforma tipo Estrecho (izquierda) y plataforma tipo Ancho (Derecha).

Ya con la parte de configuración para el armado de la plataforma ya decidida, pasamos al armado de cada una de las partes que la conforman.

Primero armamos el marco externo, el cual, cuenta con cuatro canaletas de 35” y 4 conectores de esquina. Con esto nos podemos dar una idea general del tamaño total de la plataforma y sacar medidas para la colocación de los demás elementos de la plataforma.

Las canaletas cuentas con varios orificios para la colocación de los conectores de esquina, los cuales, se deben de alinear con ellos para asegurarlos; estos conectores se ajustan con un juego de 2 tornillos largos con cabeza hueca y con 2 tuercas con retención plástica.

En la figura A.25 se observa el cuadro externo de la plataforma.

Figura A.25. Cuadro externo de la plataforma.

Ya que tenemos el cuadro externo de la plataforma, armamos las 2 canaletas restantes para colocarles las partes de movimiento de la plataforma (ruedas, juego de engranes y motores).

En la figura A.26 se puede ver la colocación del cuadro de engranes a la canaleta sin la rueda, esto para observar la forma de unión que es mediante dos tornillos huecos de 1/4 -20 X .625" con dos tuercas entre el *Toughbox Mini* y la canaleta que formara parte después del cuerpo de la plataforma.

En la figura A.26 y A.27 se pueden percibir en la primera el trabajo final de la unión de la parte de movimiento y la canaleta, esto incluyendo la rueda. En la figura A.28 se percibe la canaleta lista para ser colocada al cuadro externo y tomando mediciones para saber el lugar de conexión entre el cuadro y la canaleta con la parte de torque para evitar cualquier tipo de contacto de la rueda con la estructura.

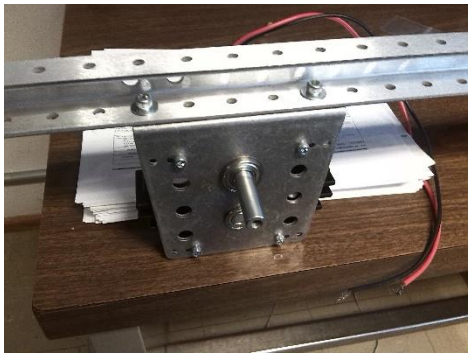


Figura A.26. Conexión del *Toughbox Mini* y la canaleta para la plataforma.

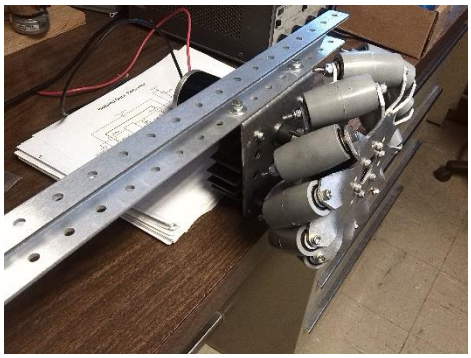


Figura A.27. Conexión del *Toughbox Mini* y rueda omnidireccional a la canaleta para la plataforma.

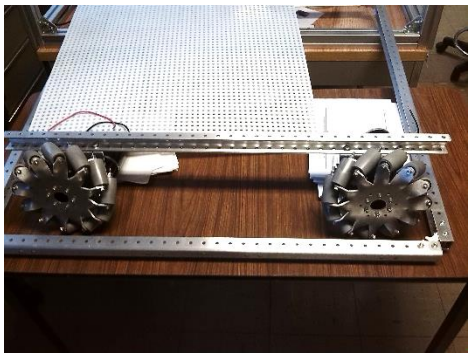


Figura A.28 Canaleta completa (Con 2 Ruedas) y cuadro Externo.

Para la colocación de las ruedas en cada canaleta, como se aprecia en la figura, se debe de tomar en cuenta que existe un par de ruedas derechas e izquierdas; en esto se debe colocar una de cada una en cada canaleta. Esto para el movimiento que se pretende de buscar y la orientación de los rodillos de 45º. Dado todo esto por el fabricante (AndyMark).

Ya con las canaletas armadas, pasamos a la medición y colocación de los tubos hexagonales, que harán el trabajo de ser un estilo de retenedores, para mantener alineadas las ruedas. En este caso, en la medición, se obtiene que se debe de cortar los tubos a 25.20" cada uno. Después de ser cortados se colocaron en cada par de placas del eje del *Toughbox Mini* en la parte inferior con tornillos con cabeza hexagonal como se muestra en la figura A.29.

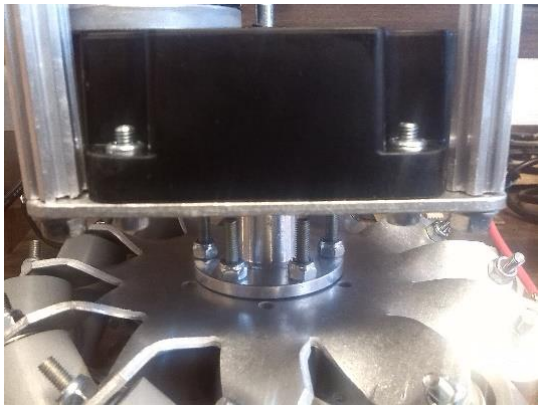


Figura A.29. Tubos hexagonales en la parte inferior del *Toughbox Mini*.

La parte de unión por las barras hexagonales de los juegos de torque de cada canaleta se puede ver en la figura A.30.

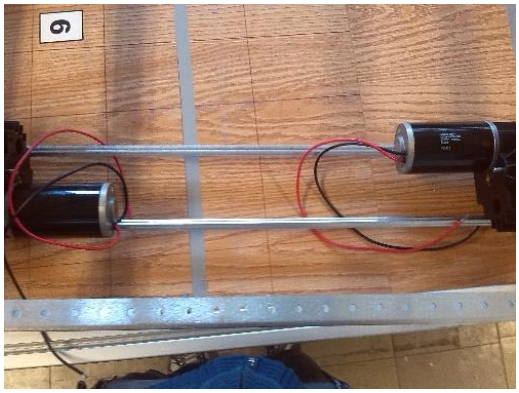


Figura A.30. Barra de equilibrio de las ruedas.

Ahora se arma la estructura de la plataforma; la unión de las canaletas con las ruedas y el cuadro externo, para ello, se utilizan los conectores de esquina; considerando lo ancho y largo de cada rueda; el trabajo final de esto se puede observar en la figura A.31.

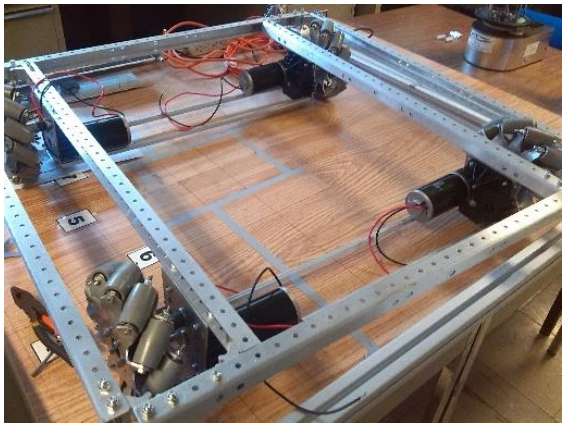


Figura A.31. Cuerpo completo de la plataforma omnidireccional.

Como parte final de este trabajo se colocó la tapa de la plataforma, para ello, se utiliza una hoja perforada de policarbonato, su colocación se basa; al tener toda la estructurar; retiramos los tornillos que intervenga con el tamaño de la hoja, después hacer coincidir los orificios de la hoja con los de las canaletas y reintegrando los tornillos retirados y colocando extras para mantener firme la placa, esto se ve en la figura A.32.

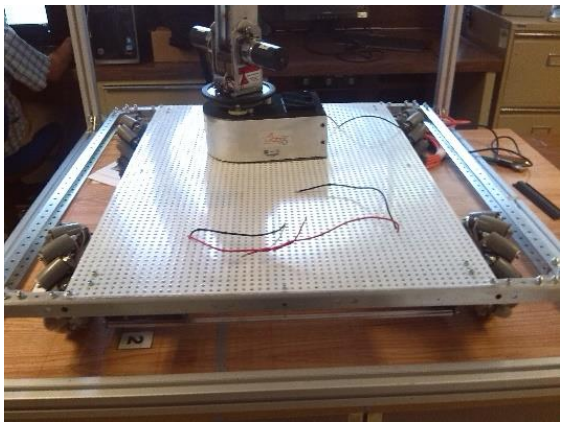


Figura A.32. Plataforma omnidireccional Completa.

Anexo A2

BLOQUE DE SISTEMA DIFUSO

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

entity Fuzzy_Logic_System is
                Port (  Clk : in STD_LOGIC;
                        Sensor   : in   STD_LOGIC_VECTOR (7 downto 0);
                        Distancia : in   STD_LOGIC_VECTOR (7 downto 0);
                        Dezfuzzy  : out  integer range 0 to 25000);
end Fuzzy_Logic_System;

architecture Behavioral of Fuzzy_Logic_System is

type inferencia is (corta,media,larga,cero,maxima);
Signal Dis,Sen: integer range 0 to 255;
Signal VAA,VAB,VBA,VBB: integer range 0 to 100;
Signal Dist,Sens: inferencia;
Signal Dist1,Sens1: inferencia;
Signal op1,op2: integer;
constant zero   : integer := 0;
constant baja   : integer := 6250;
constant media1 : integer := 12500;
constant alta   : integer := 18750;
constant maxima1: integer := 25000;
--------------------------------------------------------------------------------
FUNCTION Razon (SIGNAL a,b: integer) RETURN INTEGER IS
Variable resultado: integer;
BEGIN
        if a < b then
        resultado := a;
        else
        resultado := b;
        end if;
        RETURN resultado;
END Razon;
--------------------------------------------------------------------------------
FUNCTION Salida (SIGNAL VLD,VLS: inferencia) RETURN INTEGER IS
Variable Valor : integer;
BEGIN
        if VLD = cero and VLS = cero  then
                Valor := zero;
        elsif VLD = cero and VLS = corta  then
                Valor := zero;
        elsif VLD = cero and VLS = media  then
                Valor := zero;
        elsif VLD = cero and VLS = larga  then
                Valor := zero;
        elsif VLD = cero and VLS = maxima then
                Valor := zero;
```

```
        elsif VLD = corta and VLS = cero then
                Valor := baja;
        elsif VLD = corta and VLS = corta then
                Valor := zero;
        elsif VLD = corta and VLS = media  then
                Valor := zero;
        elsif VLD = corta and VLS = larga  then
                Valor := zero;
        elsif VLD = corta and VLS = maxima then
                Valor := zero;
        elsif VLD = media and VLS = cero   then
                Valor := baja;
        elsif VLD = media and VLS = corta  then
                Valor := baja;
        elsif VLD = media and VLS = media  then
                Valor := zero;
        elsif VLD = media and VLS = larga  then
                Valor := zero;
        elsif VLD = media and VLS = maxima then
                Valor := zero;
        elsif VLD = larga and VLS = cero   then
                Valor := alta;
        elsif VLD = larga and VLS = corta  then
                Valor := media1;
        elsif VLD = larga and VLS = media  then
                Valor := baja;
        elsif VLD = larga and VLS = larga  then
                Valor := Zero;
        elsif VLD = larga and VLS = maxima then
                Valor := zero;
        elsif VLD = maxima and VLS = cero  then
                Valor := maxima1;
        elsif VLD = maxima and VLS = corta  then
                Valor := alta;
        elsif VLD = maxima and VLS = media  then
                Valor := media1;
        elsif VLD = maxima and VLS = larga  then
                Valor := baja;
        elsif VLD = maxima and VLS = maxima then
                Valor := zero;
        end if;
        RETURN Valor;
END Salida;
--------------------------------------------------------------------------------
begin
process (Distancia)
begin
        Dis <= conv_integer(Distancia);
        if Dis >= 0 and Dis <= 50 then
                VAA   <= 2 * Dis;
                VAB   <= 100 - (2*Dis);
                Dist  <= corta;
                Dist1 <= cero;
        elsif Dis > 50 and Dis <= 100 then
                VAA   <= 200 -(2*Dis);
                VAB   <= (2*Dis)- 100;
                Dist  <= corta;
                Dist1 <= media;
```

```
        elsif Dis > 100 and Dis <= 150 then
                VAA   <= 300 - (2*Dis);
                VAB   <= (2*Dis)- 200;
                Dist  <= media;
                Dist1 <= larga;
        elsif Dis > 150 and Dis <= 200 then
                VAA   <= 400 - (2*Dis);
                VAB   <= (2*Dis)- 300;
                Dist  <= larga;
                Dist1 <= maxima;
        else
                VAA   <= 0;
                VAB   <= 0;
                Dist  <= cero;
                Dist1 <= cero;
        end if;
end process;

process (Sensor)
begin
        Sen <= conv_integer(Sensor);
        if Sen >= 0 and Sen <= 50 then
                VBA   <= 2 * Sen;
                VBB   <= 100 - (2*Sen);
                Sens  <= corta;
                Sens1 <= cero;
        elsif Sen > 50 and Sen <= 100 then
                VBA   <= 200 -(2*Sen);
                VBB   <= (2*Sen)- 100;
                Sens  <= corta;
                Sens1 <= media;
        elsif Sen > 100 and Sen <= 150 then
                VBA   <= 300 - (2*Sen);
                VBB   <= (2*Sen)- 200;
                Sens  <= media;
                Sens1 <= larga;
        elsif Sen > 150 and Sen <= 200 then
                VBA   <= 400 - (2*Sen);
                VBB   <= (2*Sen)- 300;
                Sens  <= larga;
                Sens1 <= maxima;
        else
                VBA   <= 0;
                VBB   <= 0;
                Sens  <= cero;
                Sens1 <= cero;
        end if;
end process;

process (VAA,VAB,VBA,VBB,Sens,Sens1,Dist,Dist1)
Variable SV1,SV2,SV3,SV4: integer;
Variable CS1,CS2,CS3,CS4: integer;
begin
SV1 := Razon(VAA,VBA);
SV2 := Razon (VAA,VBB);
SV3 := Razon (VAB,VBA);
SV4 := Razon (VAB,VBB);
CS1 := Salida (Dist,Sens);
```

```
CS2 := Salida (Dist,Sens1);
CS3 := Salida (Dist1,Sens);
CS4 := Salida (Dist1,Sens1);
op1 <= (CS1*SV1)+(CS2*SV2)+(CS3*SV3)+(CS4*SV4);
op2 <= SV1+SV2+SV3+SV4;
end process;

process (Clk)
Variable A1,A2,Cont : integer;
begin
        if Clk ='1' and Clk'event then
                if A1 < A2 or A1 = 0 or A2 = 0 then
                        A1 := op1;
                        A2 := op2;
                        Dezfuzzy <= Cont;
                        Cont:=0;
                else
                        Cont:= Cont +1;
                        A1 := A1 - A2;
                end if;
        end if;
end process;

end Behavioral;
```

Anexo A3

BLOQUE DE DISTRIBUCIÓN

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

entity Modulo1 is
                Port ( posicion        : in  integer range 0 to 255;
              vel1,vel2,vel3,vel4 : in  integer range 0 to 25000;
                        t1,t2,t3,t4 : out integer range 0 to 100000);
end Modulo1;

architecture Behavioral of Modulo1 is
constant neutro: integer:=75000;
constant promedio: integer:=12500;
begin

process(posicion,vel1,vel2,vel3,vel4)
begin
        if (posicion = 91)then
                t1 <= neutro - vel1;
                t2 <= neutro - vel2;
                t3 <= neutro - vel3;
                t4 <= neutro - vel4;
        elsif (posicion = 92)then
                t1 <= neutro + vel1;
                t2 <= neutro + vel2;
                t3 <= neutro + vel3;
                t4 <= neutro + vel4;
        elsif (posicion = 93)then
                t1 <= neutro - vel1;
                t2 <= neutro + vel2;
                t3 <= neutro - vel3;
                t4 <= neutro + vel4;
        elsif (posicion = 94)then
                t1 <= neutro + vel1;
                t2 <= neutro - vel2;
                t3 <= neutro + vel3;
                t4 <= neutro - vel4;
        elsif (posicion = 95)then
                t1 <= neutro - vel1;
                t2 <= 75000;
                t3 <= neutro - vel3;
                t4 <= 75000;
        elsif (posicion = 96)then
                t1 <= 75000;
                t2 <= neutro - vel2;
                t3 <= 75000;
                t4 <= neutro - vel4;
        elsif (posicion = 97)then
                t1 <= neutro + vel1;
```

```
                t2 <= 75000;
                t3 <= neutro + vel3;
                t4 <= 75000;
        elsif (posicion = 98)then
                t1 <= 75000;
                t2 <= neutro + vel2;
                t3 <= 75000;
                t4 <= neutro + vel4;
        elsif(posicion = 61)then
                t1 <= neutro - promedio;
                t2 <= neutro - promedio;
                t3 <= neutro - promedio;
                t4 <= neutro - promedio;
        elsif (posicion = 62)then
                t1 <= neutro + promedio;
                t2 <= neutro + promedio;
                t3 <= neutro + promedio;
                t4 <= neutro + promedio;
        elsif (posicion = 63)then
                t1 <= neutro - promedio;
                t2 <= neutro + promedio;
                t3 <= neutro - promedio;
                t4 <= neutro + promedio;
        elsif (posicion = 64)then
                t1 <= neutro + promedio;
                t2 <= neutro - promedio;
                t3 <= neutro + promedio;
                t4 <= neutro - promedio;
        elsif (posicion = 65)then
                t1 <= neutro - promedio;
                t2 <= 75000;
                t3 <= neutro - promedio;
                t4 <= 75000;
        elsif (posicion = 66)then
                t1 <= 75000;
                t2 <= neutro - promedio;
                t3 <= 75000;
                t4 <= neutro - promedio;
        elsif (posicion = 67)then
                t1 <=  neutro + promedio;
                t2 <= 75000;
                t3 <=  neutro + promedio;
                t4 <= 75000;
        elsif (posicion = 68)then
                t1 <= 75000;
                t2 <=  neutro + promedio;
                t3 <= 75000;
                t4 <=  neutro + promedio;
        else
                t1 <= 75000;
                t2 <= 75000;
                t3 <= 75000;
                t4 <= 75000;
        end if;
end process;

end Behavioral;
```

Anexo A4

BLOQUE PWM

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;
entity PWM is
        Port ( clk : in  STD_LOGIC;
               d : inout  STD_LOGIC;
               t : in integer range 0 to 100000);
end PWM;

architecture Behavioral of PWM is
signal g,r : INTEGER RANGE 0 TO 100000;

begin

process(d,clk)
begin
if(clk='1' and clk 'event)then
                if(g=r) then
                        g<=0;
                                if(d='1') then
                                d<='0';
                                r<= 100000;
                                else
                                d<='1';
                                r<= t;
                                end if;
                else
                        g<= g +1;
                end if;

        end if;
end process;
end Behavioral;
```

Anexo A5

APLICACIÓN ANDROID

```
import android.content.Intent;
import android.os.Bundle;
import ketai.net.bluetooth.*;
import ketai.ui.*;
import ketai.net.*;

PFont fontMy;
boolean bReleased = true;
KetaiBluetooth bt;
boolean isConfiguring = true;
String info = "";
KetaiList klist;
ArrayList devicesDiscovered = new ArrayList();

PImage bg,b,f1,f2,f3,f4,f5,f6,f7,f8,f9,f10;
int estado = 0;
int dato =0, distancia =0,i=0;
byte[] data = new byte [3];

void setup()
{
  frameRate(30);
 orientation(PORTRAIT);
 bg = loadImage("logo1.png");
  b = loadImage("logo.jpg");
 f1 = loadImage("f1.jpg");
 f2 = loadImage("f2.jpg");
 f3 = loadImage("f3.jpg");
 f4 = loadImage("f4.jpg");
 f5 = loadImage("f5.jpg");
 f6 = loadImage("f6.jpg");
 f7 = loadImage("f7.jpg");
 f8 = loadImage("f8.jpg");
 f9 = loadImage("f9.jpg");
 f10 = loadImage("f10.jpg");
bt.start();
isConfiguring = true;
fontMy = createFont("SansSerif", 40);
 textFont(fontMy);
}

void draw()
{
 switch (estado)
 {
```

```
   case 0:
   dispositivo();
    break;
   case 1:
   inicio();
    break;
   case 2:
   manual();
    break;
   case 3:
   semiautomatico();
    break;
   case 4:
   salir();
    break;
  }
}
void mouseReleased()
{
  switch (estado)
  {
   case 1:
    inicioB();
    break;
   case 2:
    manualB();
    break;
   case 3:
    semiautomaticoB();
    break;
  }
}
void mousePressed()
{
  switch (estado)
  {
   case 2:
    manualC();
    break;
   case 3:
    semiautomaticoC();
    break;
  }
}
void onCreate(Bundle savedInstanceState) {
 super.onCreate(savedInstanceState);
 bt = new KetaiBluetooth(this);
}
void onActivityResult(int requestCode, int resultCode, Intent data) {
 bt.onActivityResult(requestCode, resultCode, data);
}
void onKetaiListSelection(KetaiList klist) {
```

```
 String selection = klist.getSelection();
 bt.connectToDeviceByName(selection);
 //dispose of list for now
 klist = null;
}
void dispositivo ()
{
  if (isConfiguring)
 {
  ArrayList names;
  background(78, 93, 75);
  klist = new KetaiList(this, bt.getPairedDeviceNames());
  isConfiguring = false;
 }
   else
   {
     estado = 1;
   }
}
void inicio() {
  background(84);
  image(bg,width/4+21 , 20,width/4,height/5);
  image(b, 20, 20, width/4,height/5);
  textSize(width/20);
  fill(255);
  textAlign(CENTER, CENTER);
  text("Universidad Nacional Autónoma de México",width/2,height/2);
  textSize(width/28);
  text("Manual",width/5, height-height/4.5);
  text("Semiautomático",3*width/6, height-height/4.5);
  text("Salir",5*width/6.5, height-height/4.5);
}
void manual() {
  background(0);
  image(bg,width/4+21 , 20,width/4,height/5);
  image(b, 20, 20, width/4,height/5);
  textAlign(CENTER, CENTER);
  image(f8,0+width/7, height-height/1.4, width/7, width/5.5);
  image(f1,3*width/7, height-height/1.4, width/7, width/5.5);
  image(f5,5*width/7, height-height/1.4, width/7, width/5.5);
  image(f4,0+width/7, height-height/1.9, width/7, width/5.5);
  image(f2,5*width/7, height-height/1.9, width/7, width/5.5);
  image(f7,0+width/7, height-height/2.9, width/7, width/5.5);
  image(f3,3*width/7, height-height/2.9, width/7, width/5.5);
  image(f6,5*width/7, height-height/2.9, width/7, width/5.5);
  fill(255);
  textSize(width/28);
  text("Manual",width/2,height/2);
  text("Regresar",3*width/6, height-height/10);
  text("Salir",5*width/6.5, height-height/10);
  text(dato,2*width/4+60,20);
}
```

```
void semiautomatico() {
  background(84);
  image(bg,width/4+21 , 20,width/4,height/5);
  image(b, 20, 20, width/4,height/5);
  textAlign(CENTER, CENTER);
  image(f8,0+width/7, height-height/1.4, width/7, width/5.5);
  image(f1,3*width/7, height-height/1.4, width/7, width/5.5);
  image(f5,5*width/7, height-height/1.4, width/7, width/5.5);
  image(f4,0+width/7, height-height/1.9, width/7, width/5.5);
  image(f2,5*width/7, height-height/1.9, width/7, width/5.5);
  image(f7,0+width/7, height-height/2.9, width/7, width/5.5);
  image(f3,3*width/7, height-height/2.9, width/7, width/5.5);
  image(f6,5*width/7, height-height/2.9, width/7, width/5.5);
  tint(255, 126);
  image(f10,0+width/7, height-height/5, width/7, width/9);
  image(f9,3*width/7, height-height/5, width/7, width/9);
  noTint();
  fill(255);
  textSize(width/28);
  text("Semiautomático",width/2,height/2);
  text("Enviar",width/5, height-height/12);
  text("Regresar",3*width/6, height-height/12);
  text("Salir",5*width/6.5, height-height/12);
  text(dato,2*width/4+60,20);
  text(distancia,5*width/6.5, height-height/6);
}
void salir(){
    data[0]=byte(100);
    data[1]=byte(0);
    data[2]=byte(0);
    bt.broadcast(data);
  exit();
}
void inicioB()
{
  if(mouseX > 0+width/7 && mouseX < 0+width/7 + width/7 && mouseY > height-height/4 && mouseY < height-
height/4+ width/9)
  {
    estado = 2;
    dato = 0;
  }
  else if(mouseX > 3*width/7 && mouseX < 3*width/7 + width/7 && mouseY > height-height/4 && mouseY <
height-height/4+ width/9)
  {
    estado = 3;
    dato = 0;
    distancia =0;
  }
  else if(mouseX > 5*width/7 && mouseX < 5*width/7 + width/7 && mouseY > height-height/4 && mouseY <
height-height/4+ width/9)
  {
    estado = 4;
```

```
  }
}
void manualB()
{
  if(mouseX > 3*width/7 && mouseX < 3*width/7 + width/7 && mouseY > height-height/9 && mouseY < height-
height/9+ width/9)
  {
    estado = 1;
  }
  else if(mouseX > 5*width/7 && mouseX < 5*width/7 + width/7 && mouseY > height-height/9 && mouseY <
height-height/9+ width/9)
  {
    estado = 4;
  }
    else
  {
    dato = 0;
    data[0]=byte(100);
    data[1]=byte(dato);
    data[2]=byte(0);
    bt.broadcast(data);
  }
}
void semiautomaticoB()
{
  if(mouseX > 0+width/7 && mouseX < 0+width/7 + width/7 && mouseY > height-height/9.5 && mouseY < height-
height/9.5+ width/9)
  {
    data[0]=byte(100);
    data[1]=byte(dato);
    data[2]=byte(distancia);
    bt.broadcast(data);
  }
  else if(mouseX > 3*width/7 && mouseX < 3*width/7 + width/7 && mouseY > height-height/9.5 && mouseY <
height-height/9.5+ width/9)
  {
    estado = 1;
    data[0]=byte(100);
    data[1]=byte(0);
    data[2]=byte(0);
    bt.broadcast(data);
  }
  else if(mouseX > 5*width/7 && mouseX < 5*width/7 + width/7 && mouseY > height-height/9.5 && mouseY <
height-height/9.5+ width/9)
  {
    estado = 4;
  }
}
void manualC(){
  if(mouseX > 0+width/7 && mouseX < 0+width/7 + width/7 && mouseY > height-height/1.4 && mouseY < height-
height/1.4+ width/5.5)
  {
```

```
   dato = 66;
  }
  else if(mouseX > 3*width/7 && mouseX < 3*width/7 + width/7 && mouseY > height-height/1.4 && mouseY <
height-height/1.4+ width/5.5)
  {
   dato = 61;
  }
  else if(mouseX > 5*width/7 && mouseX < 5*width/7 + width/7 && mouseY > height-height/1.4 && mouseY <
height-height/1.4+ width/5.5)
  {
   dato = 65;
  }
  else if(mouseX > 0+width/7 && mouseX < 0+width/7 + width/7 && mouseY > height-height/1.9 && mouseY <
height-height/1.9+ width/5.5)
  {
   dato = 64;
  }
  else if(mouseX > 5*width/7 && mouseX < 5*width/7 + width/7 && mouseY > height-height/1.9 && mouseY <
height-height/1.9+ width/5.5)
  {
   dato = 63;
  }
  else if(mouseX > 0+width/7 && mouseX < 0+width/7 + width/7 && mouseY > height-height/2.9 && mouseY <
height-height/2.9+ width/5.5)
  {
   dato = 67;
  }
  else if(mouseX > 3*width/7 && mouseX < 3*width/7 + width/7 && mouseY > height-height/2.9 && mouseY <
height-height/2.9+ width/5.5)
  {
   dato = 62;
  }
  else if(mouseX > 5*width/7 && mouseX < 5*width/7 + width/7 && mouseY > height-height/2.9 && mouseY <
height-height/2.9+ width/5.5)
  {
   dato = 68;
  }
  data[0]=byte(100);
  data[1]=byte(dato);
  data[2]=byte(0);
  bt.broadcast(data);

}
void semiautomaticoC(){
  if(mouseX > 0+width/7 && mouseX < 0+width/7 + width/7 && mouseY > height-height/1.4 && mouseY < height-
height/1.4+ width/5.5)
  {
   dato = 96;
  }
  else if(mouseX > 3*width/7 && mouseX < 3*width/7 + width/7 && mouseY > height-height/1.4 && mouseY <
height-height/1.4+ width/5.5)
  {
```

```
   dato = 91;
  }
  else if(mouseX > 5*width/7 && mouseX < 5*width/7 + width/7 && mouseY > height-height/1.4 && mouseY <
height-height/1.4+ width/5.5)
  {
   dato = 95;
  }
  else if(mouseX > 0+width/7 && mouseX < 0+width/7 + width/7 && mouseY > height-height/1.9 && mouseY <
height-height/1.9+ width/5.5)
  {
   dato = 94;
  }
  else if(mouseX > 5*width/7 && mouseX < 5*width/7 + width/7 && mouseY > height-height/1.9 && mouseY <
height-height/1.9+ width/5.5)
  {
   dato = 93;
  }
  else if(mouseX > 0+width/7 && mouseX < 0+width/7 + width/7 && mouseY > height-height/2.9 && mouseY <
height-height/2.9+ width/5.5)
  {
   dato = 97;
  }
  else if(mouseX > 3*width/7 && mouseX < 3*width/7 + width/7 && mouseY > height-height/2.9 && mouseY <
height-height/2.9+ width/5.5)
  {
   dato = 92;
  }
  else if(mouseX > 5*width/7 && mouseX < 5*width/7 + width/7 && mouseY > height-height/2.9 && mouseY <
height-height/2.9+ width/5.5)
  {
   dato = 98;
  }
   else if(mouseX > 3*width/7 && mouseX < 3*width/7 + width/7 && mouseY > height-height/5 && mouseY <
height-height/5+ width/9)
  {
   if(distancia > 0)
   {
   distancia = distancia - 1;
   }
  }
  else if(mouseX > 0+width/7 && mouseX < 0+width/7 + width/7 && mouseY > height-height/5 && mouseY <
height-height/5+ width/9)
  {
    if(distancia < 200)
   {
   distancia = distancia + 1;
   }
  }

}
```

Anexo A6

COMUNICACIÓN SERIAL

Interprete serial:

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

entity serie is
   Port ( Rx,clk : in  STD_LOGIC;
                  Tx : out STD_LOGIC;
          check : out STD_LOGIC;
          Dato   : out  STD_LOGIC_VECTOR (7 downto 0));
end serie;

architecture Behavioral of serie is

type curse is (init,one,date,done);
signal cur_state: curse := init;
signal byte: integer range 0 to 10 :=0;
signal i: integer range 0 to 10000000 :=0;
constant c1: integer := 434;

begin
Tx <= '1';
process (clk)
begin
            if (clk = '1' and clk'event) then

                        case cur_state is
                                    when init =>
                                                check <= '0';
                                                if Rx ='0' then
                                                cur_state <= one;
                                                else
                                                cur_state <= init;
                                                end if;

                                    when one =>
                                                if i = c1/2 then
                                                cur_state <= date;
                                                i <= 0;
                                                else
                                                cur_state <= one;
                                                i <= i+1;
                                                end if;

                                    when date =>
                                                if i = c1 then
                                                            if byte > 7 then
                                                                        cur_state <= init;
                                                                        byte <=0;
                                                                        i <= 0;
```

```
                                        check <= '1';
                                    else
                                        cur_state <= date;
                                        i <= 0;
                                        Dato(byte) <= Rx;
                                        byte <= byte +1;
                                    end if;
                                else
                                cur_state <= date;
                                i <= i+1;
                                end if;

                        when done =>
                                cur_state <= done;

                end case;
        end if;
end process;

end Behavioral;
```

Almacenamiento y distribución de datos de intérprete serial:

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

entity Almacenamiento is
        Port (   date : in  STD_LOGIC_VECTOR (7 downto 0);
                 Reinicio : out  STD_LOGIC;
      check : in  STD_LOGIC;
      A,B,C : out  STD_LOGIC_VECTOR (7 downto 0));
end Almacenamiento;

architecture Behavioral of Almacenamiento is

type curse is (init,date1,date2,done);
signal cur_state: curse := init;

begin

process (check)
variable dato: integer :=0;
 begin
                dato := conv_integer(date);

 if (check = '1' and check'event) then

                case cur_state is
                        when init =>
                                A <= date;
                                if dato = 100 then
                                cur_state <= date1;
                                else
                                cur_state <= init;
                                end if;
```

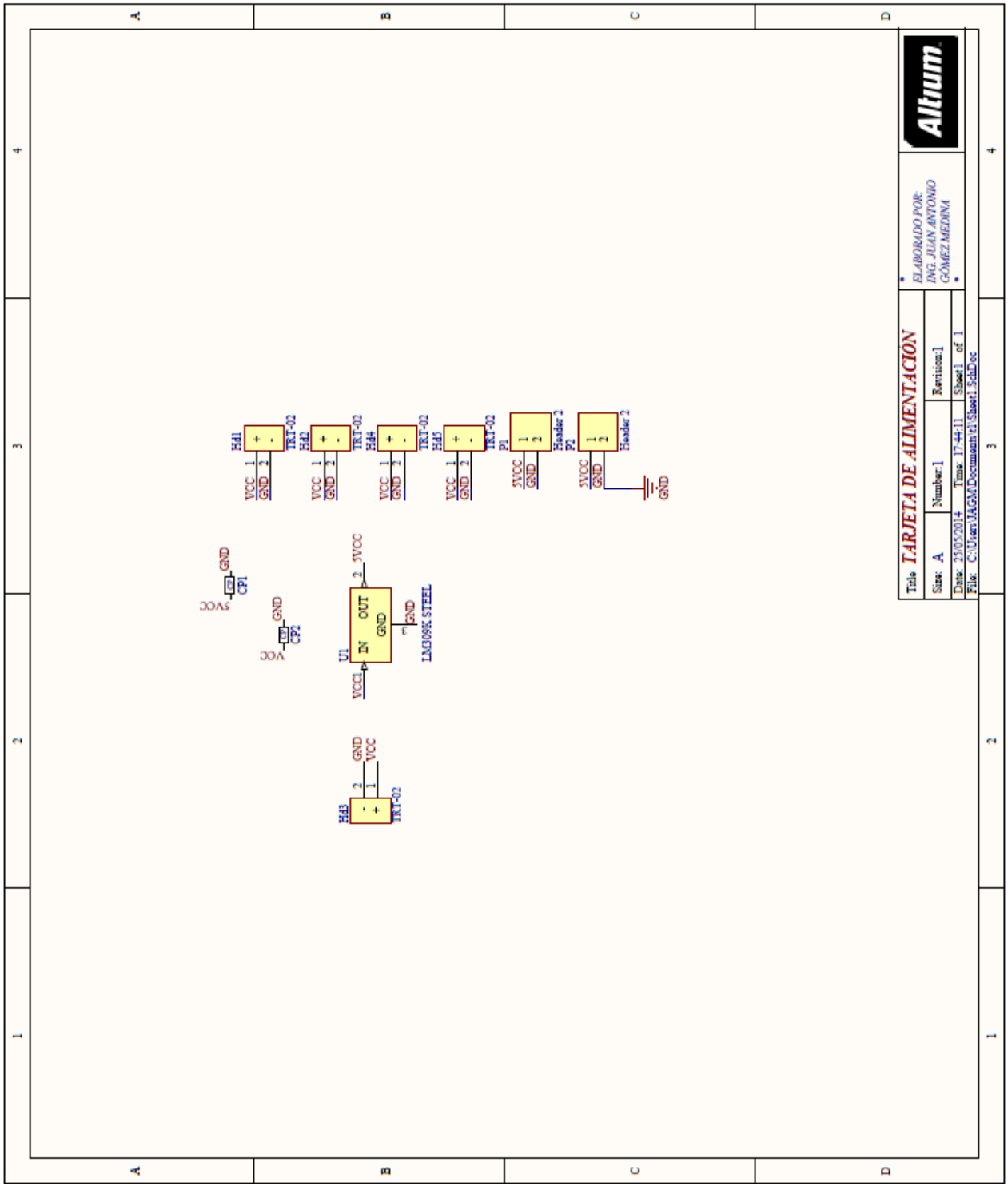
```
                when date1 =>
                                Reinicio <= '1';
                                B <= date;
                                cur_state <= date2;

                when date2 =>
                                Reinicio <= '0';
                                C <= date;
                                cur_state <= init;

                when done =>
                                cur_state <= done;

                end case;
        end if;
 end process;
end Behavioral;
```

Archivo de interconexión entre módulo de intérprete serial con el módulo de comunicación:

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

entity Comunicacion is
   Port (       Rx,Clk : in  STD_LOGIC;
          Tx,Reinicio : out  STD_LOGIC;
        Indentificador : out  STD_LOGIC_VECTOR (7 downto 0);
             Codigo : out  STD_LOGIC_VECTOR (7 downto 0);
           Distancia : out  STD_LOGIC_VECTOR (7 downto 0));
end Comunicacion;

architecture Structural of Comunicacion is

component Serial is
        Port ( Rx,clk : in  STD_LOGIC;
                                                Tx : out STD_LOGIC;
                                        check : out STD_LOGIC;
       Dato   : out  STD_LOGIC_VECTOR (7 downto 0));
end component;

component Almacenamiento is
        Port (   date : in  STD_LOGIC_VECTOR (7 downto 0);
                 Reinicio : out  STD_LOGIC;
         check : in  STD_LOGIC;
         A,B,C : out  STD_LOGIC_VECTOR (7 downto 0));
end component;

Signal check : STD_LOGIC;
Signal datos : STD_LOGIC_VECTOR (7 downto 0);

begin

C1: Serial port map(Rx,Clk,Tx,check,datos);
C2: Almacenamiento port map(datos,Reinicio,check,Indentificador,Codigo,Distancia);

end Structural;
```

Anexo A7

ESQUEMA DE ALIMENTACIÓN



Anexo A8

ADQUISICIÓN

Lector de Sensor E4P:

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

entity Encoder is
Port (          led : out std_logic_vector(7 downto 0);
               check : in std_logic;
                                                              rotary_a : in std_logic;
            rotary_b : in std_logic;
               clk : in std_logic);
end Encoder;

architecture Behavioral of Encoder is

signal     rotary_a_in : std_logic;
signal     rotary_b_in : std_logic;
signal      rotary_in : std_logic_vector(1 downto 0);
signal      rotary_q1 : std_logic;
signal  delay_rotary_q1 : std_logic;
signal    rotary_event : std_logic;
signal                                    cnt : integer range 0 to 255 := 0;
constant                ciclos : integer := 3;
signal      led_drive : std_logic_vector(7 downto 0);

begin

 process(clk)
 begin
  if clk'event and clk='1' then
   rotary_a_in <= rotary_a;
   rotary_b_in <= rotary_b;
   rotary_in <= rotary_b_in & rotary_a_in;
   case rotary_in is
     when "00" => rotary_q1 <= '0';
     when "01" => rotary_q1 <= rotary_q1;
     when "10" => rotary_q1 <= rotary_q1;
     when "11" => rotary_q1 <= '1';
     when others => rotary_q1 <= rotary_q1;
   end case;
  end if;
 end process;

 process(clk)
 begin
  if clk'event and clk='1' then
```

```
    delay_rotary_q1 <= rotary_q1;
    if rotary_q1='1' and delay_rotary_q1='0' then
      rotary_event <= '1';
     else
      rotary_event <= '0';
    end if;
  end if;
 end process;

 process(clk)
 begin
  if clk'event and clk='1' then
    if rotary_event='1' then
                                    cnt <= cnt + 1;
                                            if cnt = ciclos then
                                                    cnt <= 0;
                                                    led_drive <= led_drive + "00000001";
                                            end if;
    end if;

                        if check = '1' then
                                    led_drive <= "00000000";
                                    cnt <= 0;
                        end if;
                        led <= led_drive;
  end if;
 end process;
end Behavioral;
```

Conexión entre los lectores de cada sensor E4P al sistema:

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

entity Sensores is
   Port (       Clk : in  STD_LOGIC;
        Reinicio   : in  STD_LOGIC;
        Entradas_A : in  STD_LOGIC_VECTOR (3 downto 0);
        Entradas_B : in  STD_LOGIC_VECTOR (3 downto 0);
        A,B,C,D    : out  STD_LOGIC_VECTOR (7 downto 0));
end Sensores;

architecture Structural of Sensores is
component Encoder is
              Port (      led : out std_logic_vector(7 downto 0);
          check : in std_logic;
                                      rotary_a : in std_logic;
        rotary_b : in std_logic;
            clk : in std_logic);
end component;

begin
S1: Encoder port map(A,Reinicio,Entradas_A(0),Entradas_B(0),Clk);
S2: Encoder port map(B,Reinicio,Entradas_A(1),Entradas_B(1),Clk);
S3: Encoder port map(C,Reinicio,Entradas_A(2),Entradas_B(2),Clk);
S4: Encoder port map(D,Reinicio,Entradas_A(3),Entradas_B(3),Clk);

end Structural;
```

Anexo A9

SISTEMA DE CONTROL

En este apartado se muestra el sistema de control completo implementado en *Xilinx ISE 14.6* que contendrá los esquemáticos del sistema tanto interno como externo, aparte del código de conexión entre todas las partes involucradas, que fueron abordadas anteriormente para la creación del sistema de control.

En la figura A.33, se muestra el esquema general del sistema de control, donde se observa tanto sus entradas como sus salidas que fueron utilizadas físicamente.

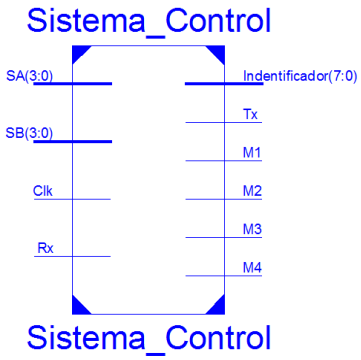


Figura A.33. Esquema general de "*Sistema de Control*".

La figura A.34 se presenta la composición interna del "Sistema de control", donde se observan los bloques abordados en el capítulo 4, en específico los que son diseñados e implementados en VHDL y sus códigos se encuentran en los anexos A2, A3, A4, A6 y A9.

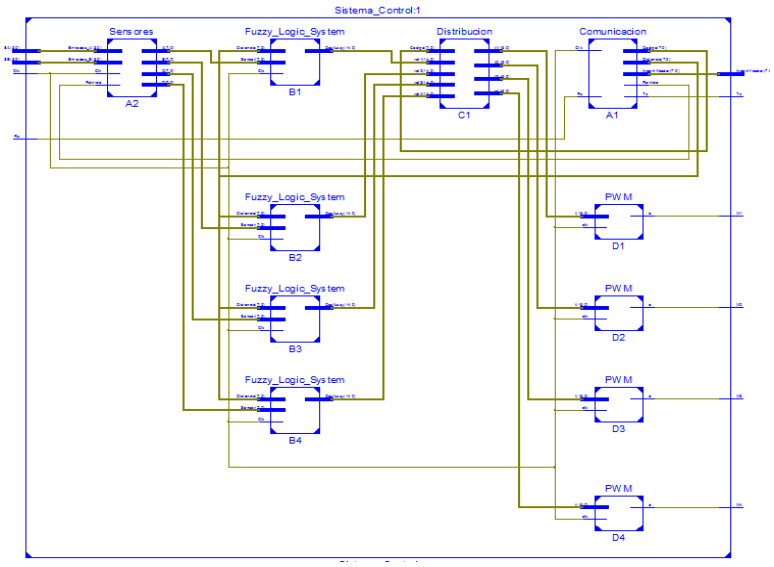


Figura A.34. Esquema interno del "*Sistema de Control*".

A continuación se muestra el código de la implementación general.

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

entity Sistema_Control is
   Port ( Clk,Rx : in  STD_LOGIC;
          Tx : out  STD_LOGIC;
          M1,M2,M3,M4 : inout  STD_LOGIC;
          SA,SB : in  STD_LOGIC_VECTOR (3 downto 0);
          Indentificador : out  STD_LOGIC_VECTOR (7 downto 0));
end Sistema_Control;

architecture Structural of Sistema_Control is

component Comunicacion is
   Port (       Rx,Clk : in  STD_LOGIC;
         Tx,Reinicio : out  STD_LOGIC;
       Indentificador : out  STD_LOGIC_VECTOR (7 downto 0);
            Codigo : out  STD_LOGIC_VECTOR (7 downto 0);
          Distancia : out  STD_LOGIC_VECTOR (7 downto 0));
end component;

component Sensores is
   Port (       Clk : in  STD_LOGIC;
       Reinicio   : in  STD_LOGIC;
       Entradas_A : in  STD_LOGIC_VECTOR (3 downto 0);
       Entradas_B : in  STD_LOGIC_VECTOR (3 downto 0);
       A,B,C,D    : out  STD_LOGIC_VECTOR (7 downto 0));
end component;

component Fuzzy_Logic_System is
                        Port (  Clk : in STD_LOGIC;
                                Sensor   : in   STD_LOGIC_VECTOR (7 downto 0);
                                Distancia : in   STD_LOGIC_VECTOR (7 downto 0);
                                Dezfuzzy  : out  integer range 0 to 25000);
end component;

component Distribucion is
                        Port (    Codigo : in  STD_LOGIC_VECTOR(7 downto 0);
                                  vel1,vel2,vel3,vel4 : in  integer range 0 to 25000;
                                  t1,t2,t3,t4 : out integer range 0 to 100000);
end component;

component PWM is
    Port ( clk : in  STD_LOGIC;
           d : inout  STD_LOGIC;
           t : in integer range 0 to 100000);
end component;

Signal Reinicio : STD_LOGIC;
Signal Codigo,Distancia : STD_LOGIC_VECTOR (7 downto 0);
Signal D_A,D_B,D_C,D_D : STD_LOGIC_VECTOR (7 downto 0);
signal Vel_A,Vel_B,Vel_C,Vel_D : integer range 0 to 25000;
signal T_A,T_B,T_C,T_D : integer range 0 to 100000;
```

```
begin

A1: Comunicacion port map(Rx,Clk,Tx,Reinicio,Indentificador,Codigo,Distancia);
A2: Sensores port map(Clk,Reinicio,SA,SB,D_A,D_B,D_C,D_D);
B1: Fuzzy_Logic_System port map(Clk,D_A,Distancia,Vel_A);
B2: Fuzzy_Logic_System port map(Clk,D_B,Distancia,Vel_B);
B3: Fuzzy_Logic_System port map(Clk,D_C,Distancia,Vel_C);
B4: Fuzzy_Logic_System port map(Clk,D_D,Distancia,Vel_D);
C1: Distribucion port map(Codigo,Vel_A,Vel_B,Vel_C,Vel_D,T_A,T_B,T_C,T_D);
D1: PWM port map(Clk,M1,T_A);
D2: PWM port map(Clk,M2,T_B);
D3: PWM port map(Clk,M3,T_C);
D4: PWM port map(Clk,M4,T_D);

end Structural;
```