BENEMÉRITA UNIVERSIDAD AUTÓNOMA DE PUEBLA

FACULTAD DE CIENCIAS DE LA ELECTRÓNICA

SIMULACIÓN DEL MOVIMIENTO DE UN VEHÍCULO TERRESTRE PARA TRAYECTORIAS PROGRAMADAS APLICANDO CONTROL DIFUSO

Tesis que presenta:

EDUARDO VALADEZ CAMPOS

Para obtener el grado de:

LICENCIADO EN ELECTRÓNICA

Director de la tesis:

Dr. GUSTAVO MENDOZA TORRES

Puebla. Pue. Abril 2018

AGRADECIMIENTOS

Me gustaría expresar mi más sincero agradecimiento al Dr. Gustavo Mendoza y al Dr. Eduardo Mendoza, por brindarme su apoyo a lo largo de todos estos años, sin el cual no podría haber concluido este trabajo ni haber llegado a ser la persona que soy actualmente. Acabar la universidad jamás habría sido posible sin su apoyo, ejemplo y confianza.

A mi esposa Daniela Reyes por su compañía, paciencia y apoyo durante la realización de este trabajo.

A mi madre María Teresa Campos por su cuidado y cariño, tal vez no me dio todas las cosas que ella hubiera querido darme pero me enseño a tener el valor suficiente para enfrentarme al mundo real cuando fue necesario.

También me gustaría agradecer a todos aquellos pocos amigos que conocí en la universidad. Su amistad y compañía fueron invaluables e hicieron más soportable el estrés, los problemas, las desveladas y el trabajo a lo largo de mi tiempo en la universidad.

Como último quisiera agradecer al DJ Armin Van Buuren y a su equipo de trabajo de "*A State Of Trance"*. Su música y programa de radio me acompañó en mis largas horas de estudio en la universidad y durante la realización de esta tesis.

RESUMEN

El presente trabajo de tesis tiene como objetivo la implementación de un programa en Visual Basic de un simulador del movimiento de un vehículo terrestre no tripulado que podrá ejecutar el seguimiento de trayectorias en 2-D de manera autónoma sobre un mapa, y usando teoría de control no-lineal basado en lógica difusa para el control de su dirección. Además se desarrolla un subsistema de adquisición de datos usando un smartphone y una brújula externa para transmitir posición y dirección al programa de simulación y comparar la salida del controlador difuso con datos reales.

El algoritmo de seguimiento de trayectorias se realiza usando un método denominado *Path Planning* (planeo de ruta), el cual consiste en que el usuario proporcione una serie de puntos que conforman una ruta, la cual el vehículo terrestre (simulado en este caso) deberá de seguir.

La motivación del proyecto se basa en la importancia que tiene en el diseño de vehículos terrestres no tripulados para diversas aplicaciones como en desastres naturales, zonas de guerra o exploración espacial.

Contenido

AGRADECIMIENTOS 1
RESUMEN 2
INTRODUCCIÓN 8
JUSTIFICACIÓN DE LA PROPUESTA 9
OBJETIVO GENERAL 9
OBJETIVOS ESPECÍFICOS 9
1. ESTADO DEL ARTE 11
1.1. HISTORIA DE LOS UGV 11
1.2. CARACTERÍSTICAS DE LOS UGV 13
2. FUNDAMENTOS TEÓRICOS 15
2.1. CONCEPTOS BÁSICOS DE LOS CONJUNTOS DIFUSOS 16
2.2. OPERACIONES SOBRE CONJUNTOS DIFUSOS 18
2.3. LÓGICA DIFUSA 19
2.4. LÓGICAS MULTIVALUADAS 23
2.5. PROPOSICIONES DIFUSAS 25
2.6. INFERENCIA DE LAS PROPOSICIONES CONDICIONALES DIFUSAS 25
2.7. REGLAS DE INFERENCIA 26
3. CONTROL DIFUSO 29
3.1. CONCEPTOS BÁSICOS 29
3.2. REGLAS DIFUSAS 32
3.3. INFERENCIA DIFUSA 33
3.3.1. INFERENCIA DE MAMDANI 33
3.4. FUSIFICACIÓN 33
3.5. DEFUSIFICACIÓN 34
3.6. RAZONAMIENTO APROXIMADO 36
3.6.1. SISTEMA DIFUSO EXPERTO 37
3.7. RAZONAMIENTO APROXIMADO MULTICONDICIONAL 38

4. PLATAFORMA DE PROGRAMACIÓN 40
4.1. SIMULACIONES POR COMPUTADORA 42
4.2. WEB MAPPING 43
5. DESCRIPCIÓN GENERAL DEL ALGORITMO 44
5.1. OBTENCIÓN DE LA TRAYECTORIA 45
5.2. CÁLCULO DE LA DIRECCIÓN Y DISTANCIA HACIA CADA PUNTO DE LA TRAYECTORIA 47
5.3. CONTROL DIFUSO DE LA DIRECCIÓN DEL VEHÍCULO TERRESTRE 53
5.3.1. FUNCIONES DE PERTENENCIA 55
5.3.2. REGLAS DE INFERENCIA E INFERENCIA DIFUSA 60
5.3.3. DEFUSIFICACIÓN 63
5.3.4. SIMULACIÓN FIS (Fuzzy Inference Systems) 63
6. ADQUISICIÓN DE DATOS 66
6.1. SMARTPHONES 68
6.1.1. DESARROLLO DE SOFTWARE EN SMARTPHONES 70
6.2. SISTEMA DE POSICIONAMIENTO GLOBAL (GPS) 71
6.3. SENSOR BRÚJULA EXTERNO 74
6.4. INTERFAZ GRÁFICA 79
CONCLUSIONES 84
7. TRABAJO ACTUAL 84
8. TRABAJO FUTURO 85
9. REFERENCIAS 86

ÍNDICE DE FIGURAS

Figura 1. Robot eliminador de bombas TALON 4 de los Estados Unidos..... 11

Figura 2. El sojourner de la NASA, el primer vehículo no tripulado en Marte. 12

Figura 3. A la derecha, MER (Mars Exploration Rover). A la izquierda, MSL (Mars Science Laboratory)..... 13

Figura 4. Funciones de pertenencia que representan los conceptos de adulto y viejo.18

Figura 5. Funcionamiento de un controlador difuso 31

Figura 6. Funciones de pertenencia para la variable lingüística T (temperatura) = {Muy frio, Frio, Tibio, Caliente, Muy caliente}..... 34

Figura 7. Simulación de la órbita de transferencia de un satélite [9]..... 43

Figura 8. Diagrama de flujo del programa de simulación para el seguimiento de trayectorias..... 45

Figura 9. Plano en donde se ven representados los vectores de dirección del vehículo y la dirección deseada. 46

Figura 10. Trayectoria del vehículo en donde se aprecia cómo cambian los vectores de dirección y sus ángulos en cada uno de los checkpoints de la trayectoria 47

Figura 11. Determinación del ángulo formado entre la recta hacia el primer punto de la trayectoria y la recta que va de norte a sur cuando Lat2 (Latitud del punto de destino) es MAYOR a Lat1 (Latitud del vehículo) 48

Figura 12. Determinación del ángulo formado entre la recta hacia el primer punto de la trayectoria y la recta que va de norte a sur cuando Lat2 (Latitud del punto de destino) es MENOR a Lat1 (Latitud del vehículo)..... 49

Figura 13. Representación de una trayectoria sobre la superficie de la Tierra. 50

Figura 14. Representación gráfica de la tolerancia de un GPS comercial de $\pm 2\ m$..... 51

Figura 15. Ruta del vehículo aplicando las tolerancias de los GPS comerciales. A la izquierda es una tolerancia de $\pm 1\ m$, a la derecha una tolerancia de $\pm\ 2\ m$ 52

Figura 16. Diagrama de bloques del control difuso 53

Figura 17. A la izquierda función sigmoid y a la derecha función gaussiana..... 56

Figura 18. Conjuntos difusos que representan los estados para la variable lingüística dirección (d) = {izquierda, Centro, Derecha} 58

Figura 19. Conjuntos difusos que representan los estados para la variable lingüística Velocidad de giro (v) = {bajo, medio, alto}..... 59

Figura 20. Conjunto difuso *Hecho*, asociado a la variable h..... 62

Figura 21. Herramienta FIS de MATLAB, en donde se ingresa un valor numérico de entrada y se obtiene el resultado del sistema difuso. 64

Figura 22. Herramienta FIS de MATLAB, en donde se definen los conjuntos difusos para la variable de entrada. 65

Figura 23. Herramienta FIS de MATLAB, en donde se definen los conjuntos difusos para la variable de salida. 65

Figura 24. Diagrama de bloques de una conversión analógica/digital 67

Figura 25. Smartphone HTC Desire 626s [20]. 69

Figura 26. Ejemplo de una constelación de 24 satélites GPS. Las líneas rojas indican los satélites que están siendo usados para obtener la posición en un determinado punto de la Tierra. 72

Figura 27. Módulo GPS con antena 73

Figura 28. Representación gráfica de las coordenadas geográficas en una esfera. 73

Figura 29. Sensor HMC5883L en un módulo GY-271 75

Figura 30. Latitud y longitud de uno de los edificios de la facultad de electrónica de la Benemérita Universidad Autónoma de Puebla. 76

Figura 31. Módulo HC-05 77

Figura 32. Diagrama de bloques de la brújula digital bluetooth 78

Figura 33. Diagrama de flujo del programa para el sensor brújula bluetooth 78

Figura 34. Diagrama de flujo del programa de adquisición de datos 79

Figura 35. Captura de pantalla del software de simulación en funcionamiento. 80

Figura 36. Panel de control del programa de simulación. 80

ÍNDICE DE TABLAS

Tabla 1. Tipos de funciones de pertenencia más usados en lógica difusa. 17
Tabla 2. Comparación de 2 códigos para calcular el número de Fibonacci. 41
Tabla 3. Pseudo-código para obtener los puntos de la trayectoria. 46
Tabla 4. Pseudo-código donde se indica los pasos usados para dibujar el polígono. .. 54
Tabla 5. Pseudo-codigo de la actualización de la posición y dirección del polígono 55
Tabla 6. Tabla comparativa de los resultados obtenidos por el control difuso y los obtenidos por la herramienta FIS de MATLAB. 64
Tabla 7. Especificaciones técnicas del smartphone HTC Desire 626s [20]. 70
Tabla 8. Características generales del sensor HMC5883L en el módulo GY-271 75
Tabla 9. Características generales del módulo bluetooth HC-05 77
Tabla 10. Datos de la dirección del vehículo. 83

INTRODUCCIÓN

En los años noventa comienza el auge de los robots móviles con el fin de realizar operaciones de propósito general. A esta clase de robots, se les dotó con un mayor grado de inteligencia y percepción que a los robots convencionales, así como de un sistema de locomoción versátil (principalmente ruedas); permitiéndoles así operar fuera del ámbito industrial donde las condiciones del entorno sufren cambios inesperados.

Una definición correcta de robot móvil plantea la capacidad de movimiento sobre entornos no estructurados, mediante la interpretación de la información suministrada a través de sus sensores y del estado actual del vehículo.

Con el constante avance de la electrónica y la miniaturización de componentes electrónicos, sensores, etc., los vehículos terrestres no tripulados se están volviendo una tendencia a nivel mundial; el uso de estos aparatos ha tomado una gran relevancia, no solo hablando de los usos militares o de vigilancia urbana, sino de aplicaciones civiles más comerciales. Por lo tanto, el objetivo es darle funcionalidad a esta clase de sistemas en desastres naturales y casos de emergencia al proporcionar información útil al personal de ayuda.

En este proyecto planteamos el diseño de un software para simular la trayectoria de un vehículo móvil, que controla su dirección mediante control difuso. El recorrido que la simulación ejecutada será visible y utiliza un mapa del servidor de Google Maps. La compañía Google provee diferentes clases de código abierto para poder incorporar servicios de mapas online en el diseño de software. Al mismo tiempo se implementará de manera física un sistema de adquisición de datos y el sistema de comunicación inalámbrica hacia la computadora del usuario usando un dispositivo móvil.

El uso del dispositivo móvil en el sistema de transmisión nos da la ventaja de que contiene en su hardware una cámara de alta resolución, un punto de acceso Wifi para la transmisión de la telemetría y un módulo de GPS para el constante monitoreo de la posición del vehículo.

JUSTIFICACIÓN DE LA PROPUESTA

En la actualidad existen tareas que el hombre no es capaz de llevarlas a cabo, por ejemplo: ingresar en áreas contaminadas por alguna sustancia química, en riesgo de derrumbe, etc. Este tipo de tareas así como algunas más pueden ser realizados por un vehículo que cubra cierta área o recorra una trayectoria programada.

Las pruebas y simulaciones de cada uno de los sistemas son de suma importancia, más tratándose de control autónomo de trayectoria, por tal motivo los programas de simulación ahorran recursos en el desarrollo final de los UGV.

OBJETIVO GENERAL

Desarrollar el software capaz de simular en 2-D el movimiento de un vehículo terrestre no tripulado (UGV por su siglas en ingles) que recorre de forma autónoma una trayectoria previamente marcada de manera manual por el usuario, y para controlar el movimiento del vehículo, aplicamos un algoritmo de control basado en Lógica Difusa.

OBJETIVOS ESPECÍFICOS

- Mostrar sobre un mapa digital la posición y dirección proporcionado por la salida del sistema, simulando a un vehículo terrestre no tripulado, el mapa será proporcionado por los servicios de Google Maps para uso libre. El usuario será capaz de marcar en el mapa manualmente la ruta que la simulación del vehículo deberá seguir.
- El software de simulación mostrará los resultados del ángulo de dirección y su referencia en todo momento durante su ejecución.
- Desarrollar un subsistema de adquisición de datos usando un smartphone y un microcontrolador, que contenga un magnetómetro (brújula electrónica) y GPS que transmita su telemetría a la computadora

y poder visualizar en un ambiente gráfico su ubicación y dirección en tiempo real.

- Diseñar el algoritmo de control, basado en Lógica Difusa que corrija el movimiento y proporcione la información al vehículo, para corregir los movimientos en todo momento y así obtener los movimientos adecuados.

CAPÍTULO 1

1. ESTADO DEL ARTE

1.1. HISTORIA DE LOS UGV

Los estudios para el desarrollo de UGV se incrementaron cuando se estudió su potencial para cumplir ciertos objetivos en situaciones de combate. En una etapa de diseño general, a los UGV se les consideraba para participar en tareas de carga, plataformas de armamento remoto, reconocimiento, vigilancia y adquisición de blancos, por mencionar algunas. Es por esto que los UGV han llamado la atención de numerosas investigadores y organizaciones, tanto federales como privadas.

De hecho los UGV fueron usados en puntos de inspección en Iraq y Afganistán (Figura 1) y en labores de rescate durante el desastre en el World Trade Center en el 2001. Sin embargo, pese al haber ganado algo de éxito, los UGV aún estaban lejos de ser confiables, ya que según estudios estadísticos presentados durante la Conferencia Internacional de Robótica y Automatización IEEE en el 2004, el tiempo medio de funcionamiento antes de presentarse una falla es entre 6 a 24 horas.

Figura 1. Robot eliminador de bombas TALON 4 de los Estados Unidos.

Los componentes de estado sólido en la electrónica ha sido otro de los factores a considerar durante esta evolución, reduciendo el tamaño considerablemente

de las computadoras y componentes electrónicos; y ya en los años setenta, la NASA inicio un programa de cooperación con el Jet Propulsión Laboratory (JPL) para desarrollar plataformas capaces de explorar terrenos hostiles. El primer fruto de esta alianza seria el MARS-ROVER, que estaba equipado con un brazo mecánico tipo STANFORD, un dispositivo telemétrico láser, cámaras estéreo y sensores de proximidad (Figura 2). En los ochenta aparece el CART del SRI que trabaja con procesado de imagen estéreo, más una cámara adicional acoplada en su parte superior. También en la década de los ochenta, el CMU-ROVER de la Universidad Carnegie Mellon incorporaba por primera vez una rueda timón, lo que permite cualquier posición y orientación del plano.

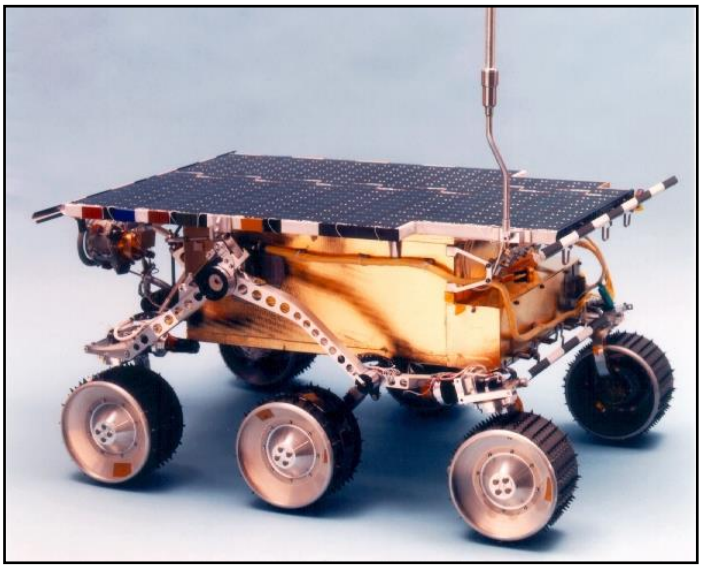

Figura 2. El sojourner de la NASA, el primer vehículo no tripulado en Marte.

Citando uno de los vehículos terrestres más avanzados está el Mars Science Laboratory, más comúnmente conocido como Curiosity. El vehículo llegó a la superficie de Marte en Agosto del 2012 y actualmente (o por lo menos hasta el 28 de Enero del 2018) se encuentra en funcionamiento (Figura 3). Debido a la distancia entre la Tierra y Marte, el vehículo terrestre debe ser capaz de realizar tareas autónomas básicas como su control y las instrucciones de su trayectoria son determinadas en la estación de control, analizando imágenes satelitales y las condiciones climáticas, para después ser enviadas hasta el vehículo terrestre. Es por eso que todos los vehículos terrestres que se han enviado se mueven a muy baja velocidad, para evitar una pérdida parcial o total de la misión.

.

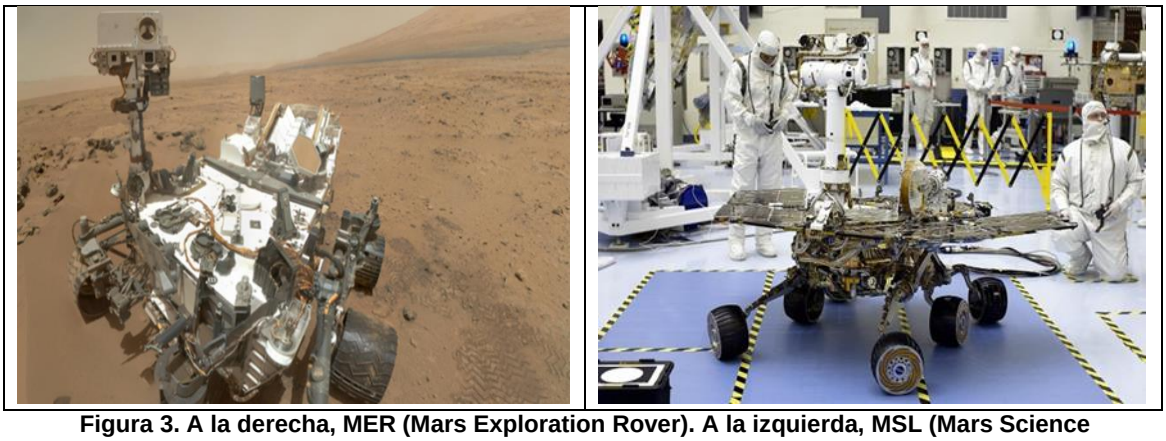

Figura 3. A la derecha, MER (Mars Exploration Rover). A la izquierda, MSL (Mars Science Laboratory).

1.2. CARACTERÍSTICAS DE LOS UGV

Una de las características principales de los UGV es que tienen una conexión "inteligente" entre las operaciones de percepción y acción, es decir que definen su comportamiento y le permiten cumplir con los objetivos programados sobre entornos con cierta incertidumbre. El grado de autonomía que pueda tener el robot dependerá en gran parte de la facultad de obtener información de su entorno y convertir dicha información en órdenes. De este modo, las dos grandes características que lo alejan de cualquier otro tipo de vehículo se relacionan a continuación.

- Percepción: El robot móvil debe ser capaz de determinar la relación con su entorno de trabajo, mediante el sistema sensorial a bordo. La capacidad de percepción se traduce en la síntesis de toda la información provista por los sensores, con el objeto de generar mapas globales y locales del entorno de acuerdo a los diversos niveles de control.

- Razonamiento: El robot móvil debe ser capaz de decidir qué acciones son requeridas en cada momento, según el estado del robot y el de su entorno, para alcanzar su(s) objetivo(s). La capacidad de razonamiento del robot móvil se traduce en la planificación de trayectorias globales seguras y en la habilidad para modificarlas en presencia de obstáculos inesperados (control local de trayectoria) para permitirle, al robot, la consecución de los objetivos encomendados.

Para poder cumplir con el objetivo del seguimiento de una trayectoria, el UGV debe poder cumplir ambas características.

La computadora de abordo (OBC por sus siglas en inglés) es el subsistema encargado de tomar toda la información recabada por los sensores y determinar sus movimientos. Generalmente todos los demás subsistemas van conectados a la OBC, y dependiendo del tiempo de funcionamiento y/o las misiones que el vehículo deba cumplir, es necesario tener dos computadoras redundantes por si ocurrieran fallos.

Para el procesamiento de los datos de los sensores, comúnmente se utiliza un microcontrolador para recibir la información mandada por un GPS y un módulo de medición inercial (IMU por sus siglas en ingles). El microcontrolador procesará dicha información para obtener su posición global (en latitud y longitud) y su dirección con respecto al norte.

El principal problema a resolver en un UGV es generar trayectorias y guiar su movimiento según éstas, en base a la información proveniente del sistema de sensores externos (ultrasonidos, láser, visión), permitiendo al vehículo desplazarse entre dos puntos del ambiente de trabajo de manera segura, sin colisiones. Esto exige diseñar sistemas de control de trayectorias (posición, dirección, velocidad) en diversos niveles jerárquicos, de manera que el procesamiento de la información proveniente de los sensores externos asegure la mayor autonomía posible

CAPÍTULO 2

2. FUNDAMENTOS TEÓRICOS

El concepto de conjuntos difusos fue concebido por Lofti Zadeh, profesor de la Universidad de California en Berkeley, quien inconforme con la rigidez de los conjuntos clásicos (crisp sets) que sólo permiten la pertenencia o no de un elemento a dicho conjunto, presentó una forma de procesar información, permitiendo pertenencias parciales a conjuntos en contraposición a los clásicos, a estos los llamo conjuntos difusos (fuzzy sets).

Este concepto fue expuesto en 1965 por Zadeh en un artículo ahora clásico de la literatura de la lógica difusa [2]. Se introducen los elementos formales que acabarían componiendo el cuerpo de la lógica difusa y sus aplicaciones, tal como se conoce en la actualidad.

En 1974, el Británico Ebrahim Mamdani, muestra que la lógica difusa puede ser aplicada en el control, desarrolla el primer sistema de control difuso práctico para la regulación de un motor de vapor. El comienzo de las aplicaciones de la lógica difusa en la teoría de control se debió al incremento en la capacidad de los procesadores computacionales.

El profesor Zadeh mostró que la gente no requiere información numérica precisa del medio que lo rodea para desarrollar tareas de control con las cuales se obtiene resultados altamente aceptables, por ejemplo conducir un automóvil o caminar por una acera sin chocar con obstáculos, personas, etcétera. Si los controladores convencionales, en esencia realimentados, se pudieran programar para aceptar entradas imprecisas, podrían trabajar de forma más eficiente y como consecuencia su implementación es muy sencilla.

En Estados Unidos por razones culturales, el concepto de lógica difusa no tuvo mucho impacto, mientras que en Japón y algunos países europeos aceptaron sin complicación esta idea y desde la década de los 80 han estado construyendo aplicaciones que funcionan basados en la lógica difusa. Por ejemplo en 1986 Yamakawa publicó “Fuzzy Controller Hardward system” y

desarrollo controladores difusos en circuitos integrados [1]. En 1987 se inaugura en Japón el tren subterráneo de Sendai, una de las aplicaciones más espectaculares de sistemas de control difuso creados por el hombre [3]. Desde entonces el controlador inteligente ha mantenido los trenes funcionando eficientemente. En al año de 1987, se comercializan muchos productos basados en la lógica difusa, sobre todo en Japón, a lo que se la llama el FUZZY BOOM.

2.1. CONCEPTOS BÁSICOS DE LOS CONJUNTOS DIFUSOS

Se sabe que dados un conjunto universal X y A un subconjunto de X, (A ⊆ X) el conjunto A puede ser representado mediante una función, llamada función característica, denotada por XA la cual está definida por:

Definición 1 *La función* $X_A : X \to \{0; 1\}$ *es una función que caracteriza al subconjunto A, esto ocurre si y solo si, para todo x,*

$$X_A = \begin{cases} 1 \; si \; x \in A \\ 0 \; si \; x \in A. \end{cases} \quad (1)$$

la cual es llamada función característica.

Esta función puede ser generalizada, al asignar a cada elemento del conjunto universal un grado de pertenencia, que en lógica difusa sería llamado grado de membresía o pertenencia. Dicha función recibe el nombre de función de membresía o de pertenencia y el conjunto que se obtiene al evaluar la función de pertenencia sobre los elementos de un conjunto universal es nombrado conjunto difuso. La cercanía del valor de la función a 1 indica un mayor grado de pertenencia [4]. Los diferentes tipos de funciones de membresía que son comúnmente usados son: singleton, triangular, trapezoidal y gaussiana. Y dependiendo del tipo de respuesta que se requiera en la salida, es como el usuario selecciona que funciones usar.

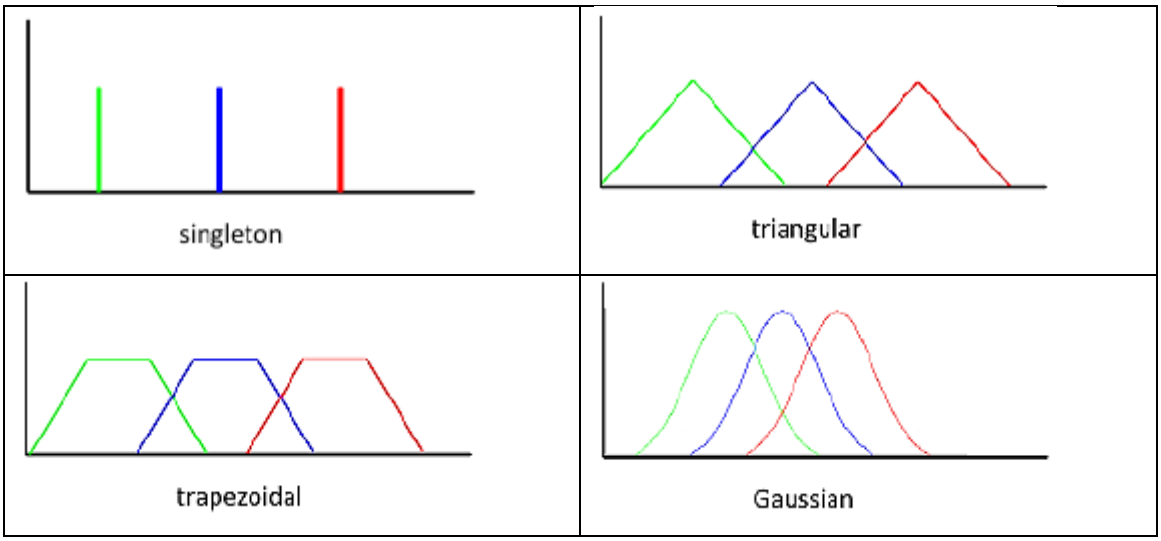

Tabla 1. Tipos de funciones de pertenencia más usados en el control por lógica difusa.

Cada conjunto difuso está completamente y unívocamente determinado por la función de membresía y el intervalo en el que se asignan los valores de la función de membresía es [0, 1].

Si X es el conjunto universal y x ∈ X, entonces un conjunto difuso A en X es definido como el conjunto de los pares ordenados

$$A = \{(x;\ A(x))/x \in X\} \quad (2)$$

donde A(x) es llamado grado de membresía de x.

Los conjuntos difusos representan conceptos lingüísticos tales como joven, adulto y viejo son empleados para definir estados de una variable, la cual es llamada variable lingüística [4].

Ejemplo:

Para ilustrar los conceptos de los conjunto difusos, se consideran tres conjuntos difusos (figura 1) que representan los conceptos de hombre joven, adulto, viejo. La forma de presentar estos conjuntos difusos será en forma trapezoidal. Estas funciones se definen en el intervalo [0, 80], la variable x son años.

$$B_1(x) = \begin{cases} 1 & \text{si} \quad x \leq 20 \\ \frac{35-x}{15} & \text{si} \quad 20 < x < 35 \\ 0 & \text{si} \quad 35 \leq x \end{cases} \tag{3}$$

$$B_2(x) = \begin{cases} 0 & \text{si} \quad \leq 20 \text{ ó } x \geq 60 \\ \frac{x-20}{15} & \text{si} \quad 20 < x < 35 \\ \frac{60-15}{15} & \text{si} \quad 45 < x < 60 \\ 1 & \text{si} \quad 35 \leq x \leq 45 \end{cases} \tag{4}$$

$$B_3(x) = \begin{cases} 0 & \text{si} \quad x \leq 45 \\ \frac{x-45}{15} & \text{si} \quad 45 < x < 60 \\ 1 & \text{si} \quad 60 \leq x \end{cases} \tag{5}$$

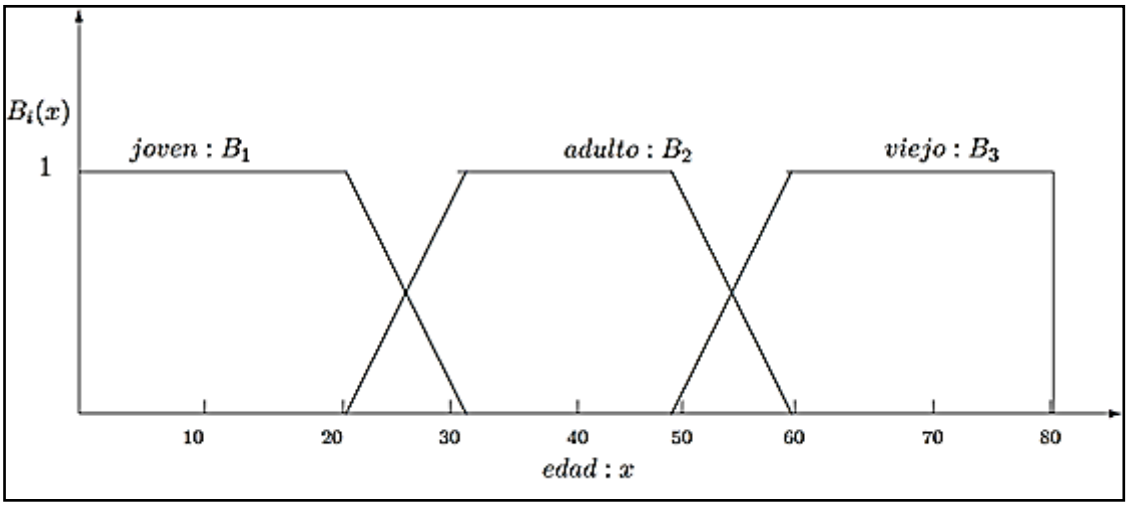

Figura 4. Funciones de pertenencia que representan los conceptos de joven adulto y viejo.

2.2. OPERACIONES SOBRE CONJUNTOS DIFUSOS

En la teoría de conjuntos se establecen operaciones entre estos, la unión, intersección, complemento, etc. Se definirán estas operaciones para conjuntos difusos, se verá que estas operaciones no son únicas, por ejemplo se tienen varios tipos de uniones e intersecciones para conjuntos difusos.

Asumiendo que A, y B son dos conjuntos difusos de x, tenemos las siguientes operaciones [4]:

Complemento de A $$\bar{A}(x) = 1 - A(x) \tag{6}$$

Intersección de A y B $$(A \cap B)(x) = \min[A(x), B(x)] \tag{7}$$

Unión de A con B	$(A \cup B)(x) = \max[A(x), B(x)]$	(8)
Complemento relativo de B con respecto a A	$(A - B)(x) = \max[0, A(x) - B(x)]$	(9)
Suma limitada de Ay B	$(A \oplus B)(x) = \min[1, A(x) + B(x)]$	(10)

Para cada operación existe una clase de funciones cuyos elementos son calificados como la generalización de las operaciones clásicas. Cada clase es caracterizada por un conjunto de axiomas justificados apropiadamente.

Aunque se pueden definir varios complementos, intersecciones y uniones difusas, las estándar poseen ciertas propiedades que le dan un significado especial.

2.3. LÓGICA DIFUSA

En el lenguaje cotidiano su utilizan expresiones tales como mexicanos al grito de guerra, los autos de la calle, Arturo es más alto que José, sino trabajo no tengo dinero, etcétera. Algunas de estas expresiones pueden ser calificadas como falsas o verdaderas, otras simplemente son expresiones. La lógica es una disciplina que se encarga de estudiar la estructura, fundamento y el uso de las expresiones del conocimiento humano. Las expresiones que pueden ser calificadas como falsas o verdaderas son llamadas proposiciones lógicas. Para el estudio de las proposiciones lógicas tenemos la lógica proposicional, la cual construye proposiciones arbitrarias por la combinación de variables, estas variables son llamadas variables lógicas (letras proposicionales). Cada variable representa una proposición, la cual aplicada a un caso particular toma cualquiera de los valores de verdad.

Se tiene que para n variables lógicas $v_1, v_2, \dots, v_n$, es posible definir una nueva variable lógica dada por una función que asigna un valor de verdad a la nueva variable, para cada combinación de valores de verdad de las variables lógicas. Esta función es llamada función lógica. Las funciones lógicas que tienen una o dos variables son llamadas operaciones lógicas o primitivas lógicas.

Se dice entonces que un conjunto de primitivas es completo, si cualquier función de variables $v_1, v_2, \ldots, v_n$ (para n finito) puede ser formada por un número finito de estas primitivas [4].

Los conjuntos de primitivas que predominan en la lógica proposicional son:

i) negación y conjunción.
ii) negación y disyunción.
iii) negación e implicación.

Por ejemplo, de la combinación de la negación, conjunción y disyunción (empleadas como primitivas) en una expresión algebraica apropiada, referida como fórmula lógica, se puede formar otra función lógica.

Las fórmulas lógicas son definidas recursivamente de la siguiente forma:

1. Si v es una variable lógica, entonces v y $\bar{v}$ son fórmulas lógicas.

2. Si a y b denotan fórmulas lógicas, entonces $a \wedge b$ y $a \vee b$ también son fórmulas lógicas. Toda formula lógica define una función lógica de la composición de las tres funciones primarias (negación, conjunción, disyunción).

Para representar una función de acuerdo al orden en el cual se da la composición individual, se pueden tomar varios caminos. El más común es el uso de paréntesis como en cualquier expresión algebraica.

Una fórmula lógica y la variable lógica asociada representan una función lógica, diferentes fórmulas pueden representar la misma función y variable lógica. Cuando las fórmulas lógicas a y b son equivalentes escribimos $a = b$.

Si una fórmula lógica es siempre verdadera, sin importar el valor asignado a las variables que componen la fórmula, esta es llamada tautología, en caso contrario es llamada contradicción.

Por ejemplo, dadas las 2 fórmulas a y b las cuales son equivalentes, entonces $a \leftrightarrow b$ es una tautología sin importar el valor asignado a a y a b, sin embargo $a \wedge b$ es una contradicción. Las tautologías son importantes para el razonamiento deductivo, porque representan fórmulas lógicas que son

verdaderas, algunas de estas tautologías son usadas como reglas de inferencia deductiva, es de referirse a ellas como reglas de inferencia.

Tautologías usadas como reglas inferencia en la lógica proposicional y otras lógicas, son [5]:

$(a \wedge (a \longrightarrow b)) \longrightarrow b$	Modus ponens
$(\bar{b} \wedge (a \longrightarrow b)) \longrightarrow \bar{a}$	Modus tollens
$(a \longrightarrow b) \wedge (b \longrightarrow c) \longrightarrow (a \Longrightarrow c)$	Silogismo hipotético

En modus ponens son dadas 2 proposiciones a y $a \rightarrow b$ verdaderas (premisas), la conclusión es la proposición b que es verdadera, por inferencia. Toda tautología permanece como tal cuando cualquiera de sus variables es sustituida por una fórmula lógica arbitraria, esta propiedad de la tautología es un ejemplo y de ahí su importancia como reglas de inferencia y también como reglas de sustitución.

La lógica proposicional basada en un conjunto de variables lógicas es isomorfa a la teoría de conjuntos, bajo una correspondencia particular entre los componentes de estas dos teorías. Más aún, estos son isomorfos al álgebra booleana. El isomorfismo entre el álgebra booleana, la teoría de conjuntos y la lógica proposicional nos garantiza que todo teorema en un sistema tiene un homólogo en cada una de las otras teorías.

La lógica proposicional estudia las relaciones lógicas, las cuales son proposiciones compuestas de otras proposiciones mediante operadores lógicos, las proposiciones expresan oraciones, cada oración representa un asunto, estado, situación, etc. Esta oración está formada por un sujeto y un predicado, debido a esto toda proposición tiene una forma general, la cual es llamada forma canónica de una proposición: x es P , donde x es el símbolo que representa al sujeto y P al predicado. Más aún podemos generalizar la forma canónica, donde x es cualquier tema contenido en un conjunto universal X, el predicado P toma el papel de una función definida en X, con lo que cada valor

de x forma una proposición, la cual es una función que se llamará predicado y se denotará $P\,(x)$. Claramente una proposición puede ser representada por la función predicado, la cual puede ser verdadera o falsa, lo que dependerá del valor que x tome del conjunto universal X.

El predicado puede estar dado de dos formas. En la primera forma existe un predicado *n*-ario $P(x_1, x_2, \dots, x_n)$, cuando $n = 1$ representa una propiedad y para $n = 2$ una relación entre dos temas dados en su respectivo conjunto universal $X_i (i \in N)$, por ejemplo:

x_1 es ciudadano de x_2

Es un predicado binario, donde x_1 es una persona del conjunto universal X_1y x_2 es una ciudad del conjunto de ciudades X_2.

Otra forma de ver un predicado, es extendiendo su alcance, cuantificando su valor respecto del conjunto de variables. Hay dos formas de cuantificar los predicados, las cuales son el cuantificador existencial y el cuantificador universal [5].

El cuantificador existencial de un predicado $P\,(x)$, el cual es expresado por la forma $(\exists x)P\,(x)$ que representa la sentencia: “Existe un individuo *x*” (en el conjunto universal X de la variable x) “tal que x es P ” (o la sentencia equivalente: “Algunos $x \in X$ son P ”). El símbolo $\exists$ se llama cuantificador existencial y cumple la igualdad

$$(\exists x)(Px) = \bigvee_{x \in X} P(x). \qquad (11)$$

El cuantificador universal de un predicado $P\,(x)$, tiene la forma $(\forall x)P(x)$, significa que “Para todo $x \in X, x\ es\ P$ ”. El símbolo $\forall$ se llama cuantificador universal y cumple la igualdad.

$$(\forall x)(Px) = \bigwedge_{x \in X} P(x). \qquad (12)$$

2.4. LÓGICAS MULTIVALUADAS

En la lógica proposicional clásica toda proposición es verdadera o falsa, en el caso en que las proposiciones se relacionan con eventos futuros o inciertos, no se puede decir que sea totalmente verdadero o totalmente falso, por lo tanto los valores de verdad son indeterminados con menor o mayor prioridad para el evento. Para la evaluación de proposiciones de este tipo y dado que en la lógica bi-valuada no se contemplan estos casos, puede extenderse a la lógica tri-valuada en la que se aplican la verdad, la falsedad y lo indeterminado, denotados por $1, 0, 1/2$ respectivamente. Algunas de las lógicas tri-valuadas más aceptadas por su utilidad, se generalizan a las lógicas *n*-valuadas. Por ejemplo, puede definir al conjunto T_n de valores de verdad de una lógica *n*-valuada se puede definir:

$$T_n = \left\{0 = \frac{0}{n-1}, \frac{1}{n-1}, \frac{2}{n-1}, \dots, \frac{n-2}{n-1}, \frac{n-1}{n-1} = 1\right\} \tag{13}$$

Estos valores pueden ser interpretados como grados de verdad. La primer serie de lógica n –valuadas, para $n \geq 2$*;* fue propuesta por Lukasiewicz en los años 30's [3], como una generalización de su lógica *tri*-valuada. Utilizó los valores de verdad dados en T_n y definió las primitivas mediante las siguientes ecuaciones:

$$\bar{a} = 1 - a \tag{14}$$

$$a \wedge b = \min(a, b) \tag{15}$$

$$a \vee b = \max(a, b) \tag{16}$$

$$a \rightarrow b = \min(1, 1 + b - a) \tag{17}$$

$$a \leftrightarrow b = 1 - |a - b| \tag{18}$$

Lukasiewicz, utilizó la negación y la implicación como primitivos y definió las otras operaciones lógicas en términos de estas, como se muestran a continuación:

$$a \vee b = (a \rightarrow b) \rightarrow b \tag{19}$$

$$a \wedge b = \overline{\bar{a} \vee \bar{b}} \tag{20}$$

$$a \leftrightarrow b = (a \rightarrow b) \wedge (b \rightarrow a) \tag{21}$$

Para $n \geq 2$, la lógica *n*-valuada de Lukasiewicz se denota por L_n, sus valores de verdad de L_n dados por los valores del conjunto T_n. La sucesión $(L_1, L_2, \ldots, L_\infty)$ de lógicas tiene como extremos las lógicas L_2 y L_∞. La lógica L_2 es la lógica clásica 2-valuada y la lógica L_∞ es una lógica *infinita*-valuada, la cual toma sus valores de verdad de todos los números racionales del conjunto contable T_∞, contenido en el intervalo $[0; 1]$. Ahora supóngase que no sólo toma valores de verdad del conjunto T_∞, sino que puede tomar cualquier valor real en el intervalo [0*;* 1] como valor de verdad, se obtiene otra lógica *infinita*-valuada.

A pesar de esta diferencia es posible decir que estas lógicas son equivalentes, en el sentido de que representan las mismas tautologías. Sin embargo, esta equivalencia sólo se cumple para fórmulas lógicas que involucren proposiciones, para fórmulas predicados con cuantificadores puede haber algunas diferencias entre las lógicas tratadas.

Se establece un isomorfismo entre la lógica *n*-valuada L_∞ y la lógica *infinita*-valuada que toma sus valores de verdad de los números reales en el intervalo $[0,1]$, la cual es la lógica estándar de Lukasiewicz(L_1). L_1 es isomorfa a la teoría de conjuntos difusos, basada en los operadores difusos estándar, son isomorfas en el mismo sentido que son isomorfas la lógica 2-valuada y la teoría de conjuntos. En realidad el grado de membrecía $A(x)$ para $x \in X$, de un conjunto difuso A definido en el conjunto universal X, puede ser interpretado como el valor de verdad de la proposición "x es un miembro del conjunto A" en L_1. Recíprocamente los valores de verdad para todo $x \in X$ de una proposici´on "x es P " en $L1$, donde "P" es un predicado vago (difuso) tal como alto, joven, costoso, peligroso, etc., pueden ser vistos como los grados de membrecía $P(x)$ en el cual el conjunto difuso es caracterizado por la propiedad P definida en X. Se establece el isomorfismo entre las operaciones de L_1, que tienen la misma forma que las operaciones estándar de los conjuntos difusos. Para cada lógica

infinita-valuada se podría verificar que es isomorfa a las operaciones estándar de los conjuntos difusos.

La lógica de Lukasiewicz es una de las lógicas *infinitas*-valuadas con las que se puede establecer un isomorfismo con una de las teorías de conjuntos difusos, que es una de las variedades de las teorías de conjuntos difusos, estas difieren una de otra por las operaciones de conjuntos empleadas. La insuficiencia de una lógica *infinita*-valuada se asocia con la noción de un conjunto completo de primitivas lógicas [2]. Esto es, se sabe que existe un conjunto completo no finito de primitivas lógicas para una lógica *infinita*-valuada. Por lo tanto dado un conjunto finito de primitivas se puede definir una lógica *infinita*-valuada, sólo se obtiene un subconjunto de todas las funciones de las variables lógicas primarias.

2.5. PROPOSICIONES DIFUSAS

La diferencia entre las proposiciones clásicas y las proposiciones difusas es el rango de los valores de verdad. Mientras que una proposición clásica sólo toma valores de verdad o falsedad, en una proposición difusa el valor de verdad es expresado por un número en el intervalo $0,1$.

En esta sección se mencionan los tipos de proposiciones difusas, las cuales se clasifican en:

1. Proposiciones no condicionales y no calificadas.
2. Proposiciones no condicionales y calificadas.
3. Proposiciones condicionales y no calificadas.
4. Proposiciones condicionales y calificadas.

2.6. INFERENCIA DE LAS PROPOSICIONES CONDICIONALES DIFUSAS

En la lógica difusa las reglas de inferencia se utilizan para facilitar el razonamiento aproximado, en esta sección se generalizan las tres reglas de inferencia de la lógica clásica, las cuales como se sabe son: modus ponen,

modus tollens y silogismo hipotético. Estas generalizaciones están basadas en reglas proposicionales de inferencia.

La relación R es introducida en una proposición condicional difusa p de la forma

$$\boldsymbol{p}: \textit{Si } X \textit{ es } A;\ \textit{entonces } Y \textit{ es } B$$

Es determinado para todo $x \in X$ y todo $y \in Y$ por la fórmula

$$R(x, y) = \mathcal{J}|A(x), B(x)| \tag{22}$$

Donde $\mathcal{J}$ es una implicación difusa.

Si se da otra proposición q de la forma

$$\boldsymbol{q}: X \textit{ es } A'$$

Se concluye que Y es B' por la regla composicional de indiferencia.

Este procedimiento es llamado generalización de *modus ponens* [6].

Dada una proposición p como una regla y la proposición q como un factor, la generalización de modus ponen es dada de la siguiente forma:

Regla : Si X es A , entonces Y es B

Antecedente : Y es A'

Conclusión : Y es B'

2.7. REGLAS DE INFERENCIA

El eje fundamental del uso de lógica difusa para un sistema de control es la versatilidad para escribir reglas que provengan del sentido común.

Las reglas difusas acoplan conjuntos difusos de entrada (pueden ser uno o más) llamados premisas, y los ligan con un conjunto difuso de salida llamado consecuente.

Con las reglas difusas es posible expresar la relación completa entre las premisas y el consecuente, para ello es necesario contar con varias reglas (base de reglas).

La base de reglas se representa con una FAM (*Fuzzy Associative Memory*). Las FAM son matrices donde el consecuente de cada regla queda representado para cada combinación de a par de las entradas, es decir, muestra la correspondencia entre la variable lingüística de salida y las variables lingüísticas de entrada [7].

Un sistema de control difuso se construye con una base de reglas de la forma

SI {entrada/situación} ENTONCES {salida/acción}

Se denomina a esta forma de reglas como de tipo Mamdani.

En el formato Mamdani se comienza por escribir reglas básicas y después con la experiencia del experto depurarlas.

A propósito de los sistemas descritos con múltiples entradas y una sola salida, éstos se conocen como MISO (*Multiple Input Single Output*).

La inferencia difusa es el proceso mediante el cual se obtiene como conclusión un conjunto difuso a partir de premisas tomadas de las reglas de inferencia.

La inferencia difusa permite interpretar las reglas de tipo SI-ENTONCES de una base de reglas, para obtener valores de salida a partir de los valores de entrada del sistema.

Uno de los métodos más usados en este tipo de aplicaciones (cuando se tiene un número reducido de variables) es el método de Mamdani, ya que tiene una estructura muy simple de operaciones “min-max”.

Se puede caracterizar a la inferencia difusa como la generalización del *modus ponens* o *modus ponens difuso*, tenemos el *modus ponens difuso* representado como:

Si x es A, entonces y es B

x es A

---

y es B

- A y B son conjuntos difusos.
- "x es A" representa x es "algo parecido" a A (pertenencia parcial).
- "y es B" representa y es "algo parecido" a B (pertenencia parcial).

Un ejemplo ilustrativo es el siguiente:

Si la curva es muy cerrada, reducir la velocidad.

La curva es ligeramente cerrada.

Reducir un poco la velocidad.

CAPÍTULO 3

3. CONTROL DIFUSO

En este capítulo se presenta una Introducción a la implementación de la lógica difusa en un sistema de control; en este trabajo el control difuso se basa en la generalización del modus ponnens para conjuntos difusos.

3.1. CONCEPTOS BÁSICOS

Los sistemas expertos de control difuso basado en reglas, conocido como controladores difusos (FLC - Fuzzy Logic Controller por sus siglas en ingles), son sin duda la aplicación más extendida de la lógica difusa [6][7]. Un primer bloque realiza un pre-procesado de las variables de entrada, que proporciona el vector de entradas al controlador difuso. El controlador difuso aplica la entrada que recibe a la base de reglas, para obtener la salida. Finalmente, esta salida puede requerir un procesado final, con el fin de adecuarla al proceso que se ha de controlar.

La estructura básica de un controlador difuso, consta de un primer elemento llamado fusificador, que realiza la conversión de valores clásicos a términos difusos. Su salida es utilizada por el dispositivo de inferencia difuso para aplicarla a cada una de las reglas de la base de reglas, siguiendo el método de inferencia seleccionado. La salida de este bloque pueden ser conjuntos difusos o bien un conjunto difuso. Finalmente, el desfusificador transformar estos conjuntos difusos en un valor no difuso o clásico.

En general un FLC puede ser presentado de la siguiente manera:

$$u(k) = F\,(e(k), e(k-1), \dots, e(k-v), u(k-1), u(k-2), \dots, u(k-v)) \tag{23}$$

Donde la función F, las normas de control son descritas en una acción que describe la relación entre la variable de entrada y la salida del control.

Es importante resaltar que un FLC no es una función de transferencia de ecuaciones diferenciales.

La base de información del FLC dicta el uso limitado del valor del error e y el control u, porque es el radio razonable para el manejo del enunciado lingüístico para $e(k-3;\ e(k-4);\ u(k-3);\ u(k-4), etc.$

Es resumen, diferentes combinaciones de valores de $e(\cdot)$ y $u(\cdot)$ con significado físico, por ejemplo cambio de errores:

$$\Delta e(k) = e(k) - e(k-1) \tag{24}$$

Suma de errores:

$$\sum e(k) = \sum_{i=0}^{k} e(i-1) \tag{25}$$

Cambio de control:

$$\Delta u(k) = u(k) - u(k-1) \tag{26}$$

Pueden ser considerados en el FLC.

Un FLC típico describe la relación entre el cambio de control $\Delta u(k) = u(k) - u(k-1)$ de un lado, el error $e(k)$ y el cambio $\Delta e(k) = e(k) - e(k-1)$ del otro lado, tal que una ley de control puede ser formalizada como:

$$\Delta u(k) = F(e(k), \Delta e(k)) \tag{27}$$

Que es la representación general de un FLC.

La actual salida del controlador $u(k)$ es obtenida desde los valores previos del control $u(k-1)$.

$$u(k) = u(k-1) + \Delta u(k) \tag{28}$$

Cada regla del FLC es caracterizada con un *SI*, el cual se llama antecedente y con un *ENTONCES* llamado consecuente. El antecedente de una regla debe cumplir con un conjunto de condiciones, el consecuente contiene una conclusión.

Cada regla trabaja de la siguiente forma: *SI* la condición del antecedente es satisfecha, *ENTONCES* la conclusión del consecuente se aplica.

El FLC es un sistema, que tiene como entradas las variables, estas son incluidas en el antecedente de las reglas y las salidas de la variable son incluidas en el consecuente. El error $e(k)$ y el cambio $\Delta e(k)$ son entradas, el cambio del control $\Delta u(k)$ la salida del FLC, representada por la ecuación (27).

Las salidas y las entradas del FLC son los estados del sistema controlado, esto es el FLC es un estado de las variables controladas por una familia de reglas y mecanismos de inferencia difusa.

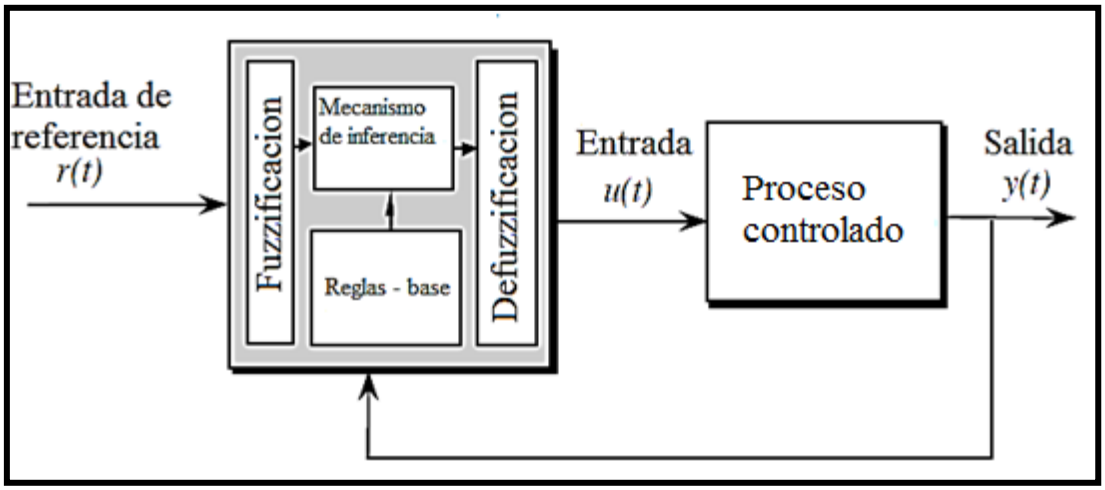

Figura 5. Funcionamiento de un controlador difuso

En general un controlador difuso (Figura 5) es un sistema experto, ya que utiliza información que provee un operador humano, dicha información es expresada en términos de reglas de inferencia difusa y una inferencia apropiada para resolver el problema [6][7].

Una forma típica de estas reglas se ejemplifica por la regla:

SI la temperatura es muy alta *Y* la presión es ligeramente baja *ENTONCES* el cambio de calor es ligeramente negativo.

Donde la temperatura y la presión son variables del proceso y el cambio de calor es la acción que ejecuta el controlador, los términos muy alto, ligeramente bajo y ligeramente negativo son representados por conjuntos difusos.

Un controlador difuso consiste de cuatro módulos:

- Una base de reglas difusas.
- Una inferencia difusa.
- Un módulo de fusificación.
- Un módulo de defusificación.

El controlador opera por repetición en ciclos de los cuatro pasos anteriormente descritos.

3.2. REGLAS DIFUSAS

Para entender como formular lo que se conoce como reglas difusas, supongamos que tenemos a un humano experto en el manejo de cualquier vehículo, y por otro lado tenemos a otro humano, sin ningún tipo de experiencia previa, al que se le pide llegar a determinado punto en su zona de trabajo. Para que este humano sin experiencia pueda completar su tarea, el humano experto le proporciona una serie de instrucciones o reglas de cómo es mejor controlar el vehículo en base a ciertas variables de entrada (como la posición o la velocidad). Estas reglas vienen en un lenguaje lingüístico natural, por ejemplo inglés y deben ser lo más simples posibles para que el conductor pueda realizar la tarea asignada.

Estas reglas no son el modelado del comportamiento del vehículo, más bien son reglas que capturan la toma de decisiones del humano experto de cómo es mejor el manejo de la planta en cuestión.

Una regla difusa es una declaración del tipo IF-THEN con una condición y una conclusión, dependiendo de la cantidad de variables lingüísticas y sus respectivos valores lingüísticos, es la cantidad de reglas posibles. Por ejemplo,

si tenemos dos variables lingüísticas y 5 valores lingüísticos por cada una, hay como máximo $5^2 = 25$ posibles reglas (Todas las combinaciones posibles).

3.3. INFERENCIA DIFUSA

Al proceso por el cual se obtiene un valor de salida, a partir de cierto valor de entrada usando teoría de conjuntos difusos, se le conoce como Inferencia difusa. Hay dos tipos de inferencia usados ampliamente: el modelo de Mamdani y el de TSK (Takagi, Sugeno y Kang).

3.3.1. INFERENCIA DE MAMDANI

Es el método más ampliamente usado, ya que se presta más a la representación del conocimiento experto de una forma más intuitiva. Propuesto en el año de 1975 por Ebrahtm Mamdani y consiste en 4 pasos:

- Fusificación de las variables de entrada
- Evaluación de las reglas
- Agregación de las salidas de las reglas
- Defusificación

3.4. FUSIFICACIÓN

Es el proceso por el cual se toman los valores de entrada y mediante el uso de las funciones de pertenencia, se determina el grado de pertenencia de cada valor.

Las funciones de pertenencia son usadas tanto en la fusificación como en la defusificación para cuantificar los términos lingüísticos. El motor o mecanismo de inferencia opera con dichos conjuntos. Hay que notar que una importante característica de la lógica difusa es que un valor numérico no tiene que ser fusificado usando únicamente una función de pertenencia. En pocas palabras, un valor puede pertenecer a múltiples conjuntos difusos al mismo tiempo, teniendo un grado de pertenencia diferente para cada conjunto.

En la figura 6 se muestra una gráfica ejemplo de unas funciones de pertenencia para los términos lingüísticos de un medidor de temperatura, de acuerdo a la gráfica podemos notar que un valor de temperatura puede ser considerado

“frio” y “Muy frio” al mismo tiempo, con diferentes grados de pertenencia en cada caso.

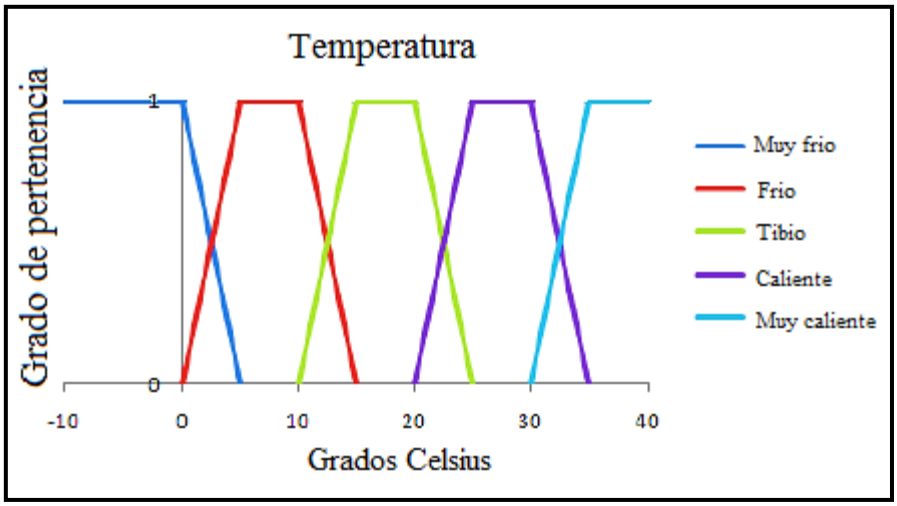

Figura 6. Funciones de pertenencia para la variable lingüística T (temperatura) = {Muy frio, Frio, Tibio, Caliente, Muy caliente}

3.5. DEFUSIFICACIÓN

Tomando primero el concepto de fusificación, el cual consiste en evaluar los valores de entrada no difusos en los conjuntos difusos delimitados por un rango específico, dichos conjuntos tienen una etiqueta lingüística que los identifica.

Los conjuntos difusos son definidos primeramente por el conocimiento del experto, sin embargo éstos no necesitan ser simétricos, ni estar uniformemente extendidos dentro de los rangos dados.

Diferentes formas de fusificación pueden definirse para diferentes variables, en este trabajo sólo se usa el método directo. El cual consiste en tomar la función de fusificación (la función que define a los conjuntos difusos) y operarla con las variables no difusas de entrada para expresar la incertidumbre (o grado de pertenencia) en los conjuntos difusos [6][7]. El propósito de la fusificación es interpretar las mediciones del acelerómetro expresadas en números reales, con una aproximación difusa respecto a esos números reales.

La defusificación es una función que transforma un conjunto difuso, salido de una implicación difusa, en un valor no difuso, para lo cual emplea los siguientes métodos.

Estos métodos transforman valores difusos que se obtienen del motor de inferencia y se convierten en valores reales.

Defusificador por media de centros de área [7]. Para el caso continuo, el valor es calculado por la fórmula:

$$d_{CA}(C) = \frac{\int_{i=1}^{n} zC(z)dz}{\int_{i=1}^{n} C(z)dz} \quad (29)$$

Dónde:

- $z \in Z$ (Z conjunto universal).
- $C(z)$ es el grado de pertenencia a algún conjunto difuso A.

Para el caso discreto en el que C es definido en el conjunto universal $\{z_1, z_2, z_3, \dots, z_n\}$ la fórmula es:

$$d_{CA}(C) = \frac{\sum_{i=1}^{n} z_{oi} C_i(z)}{\sum_{i=1}^{n} C_i(z)} \quad (30)$$

Dónde:

$z_{oi} \in Z$ es el centro del intervalo para el cual está definido $z \in Z$.

Centro del método máximo. En este método el valor defusificado $d_{CM}(C)$ es definido como el promedio de los valores más pequeños y los valores más grandes de v para el cual $C(z)$ es la altura, $h(C)$ de C:

$$d_{CM}(C) = \frac{\inf M + \sup M}{2} \quad (31)$$

Dónde:

$$M = \{z\epsilon[-c, c]/C(z) = h(C)\}$$

Para el caso discreto:

$$d_{CM}(C) = \frac{\min \{z_k/z_k \epsilon M\} + \max\{z_k/z_k \epsilon M\}}{2} \quad (32)$$

Dónde:

$$M = \{z_k / C(z_k) = h(C)\}$$

Media del método máxima. Este método se usa sólo para el caso discreto, el valor defusificado $d_{MM}(C)$, es el promedio de todos los valores en el conjunto clásico M definido anteriormente, esto es:

$$d_{MM}(C) = \frac{\sum_{z_k \epsilon M} z_k}{|M|} \quad (33)$$

3.6. RAZONAMIENTO APROXIMADO

Para que un sistema pueda realizar ciertas funciones complejas, una programación clásica no es suficiente, ya que el sistema no estaría explícitamente programado para que realice dichas funciones, por lo que el sistema tendría que "razonar" su respuesta.

Por ejemplo si el sistema conoce los siguientes hechos: "La Tierra tiene agua", "la Tierra es un planeta" y le planteáramos la siguiente pregunta: "¿Todos los planetas tienen agua?", el sistema, al no estar explícitamente programado para contestar dicha pregunta, debe razonar para dar una respuesta. Sin embargo a primera vista podemos darnos cuenta que el sistema no podría darnos una respuesta correcta, ya que no conoce el porqué de que la Tierra tiene agua en primer lugar.

Por lo tanto podemos deducir que, cuando el número de hechos y reglas con los que se cuentan es grande, el sistema será capaz de entregar una respuesta más aproximada a la correcta.

El razonamiento aproximado se utiliza para razonar con conocimiento expresado en forma de primitivas reglas, enunciadas en lenguaje natural. Por ejemplo: "La temperatura tiene un valor negativo grande". Es necesario conocer estos fundamentos, ya que son esenciales para el diseño de motores de inferencia para sistemas difusos expertos.

3.6.1. SISTEMA DIFUSO EXPERTO

Un sistema experto es una base de datos computarizados que intenta emular el proceso de razonamiento humano en un dominio de conocimientos (información) específicos. Los sistemas expertos son construidos con el fin de tomar la experiencia, el entendimiento y resolver problemas con la información obtenida de un experto en un tema en especial, por una persona que no necesariamente sea experta en el tema. Algunos de los temas para los que se diseñan sistemas expertos son: consultoría, diagnósticos médicos, lectura, toma de decisiones, investigación y desarrollo.

El núcleo de un sistema experto consiste en una base de conocimientos (información), una base de datos y un motor de inferencia. Estas tres unidades, junto con un interfaz para comunicarse con el usuario, forman la configuración mínima de un sistema experto.

La base de conocimiento contiene información general que pertenece al dominio de conocimiento para el problema. En un sistema experto difuso, el conocimiento es representado por un conjunto de reglas difusas, las cuales conectan el antecedente con el consecuente, premisas con conclusiones o condiciones con acciones. En un sistema experto difuso, tiene la forma “SI A, ENTONCES B”, donde A y B son conjuntos difusos [3].

La base de datos es un almacén de estos, dispuestos para ser utilizados por el sistema experto, estos datos establecen un diálogo entre el sistema experto y el usuario. Otros datos pueden ser obtenidos por la inferencia del sistema experto.

El motor de inferencia de un sistema experto difuso que opera en una serie de reglas de producción y realiza la inferencia difusa. Existen dos aproximaciones para evaluar las reglas de producción. La primera es el manejo de datos y es ejemplificado por la regla de inferencia modus ponens. En este caso, provee los datos disponibles al sistema experto, los cuales son evaluados en las reglas de producción y obtenida la posible conclusión. Un método alternativo de evaluación es el manejo de meta; este es generalizado por la regla de

inferencia *modus tollens*: El motor de inferencia también puede utilizar el conocimiento de la base en las reglas de producción. Este tipo de conocimiento cuyo nombre es *metaconocimiento* se localiza en la base de conocimiento.

3.7. RAZONAMIENTO APROXIMADO MULTICONDICIONAL

La forma general del razonamiento aproximado multicondicional es:

Regla 1: Si X es A_1 entonces Y es B_1

Regla 2: Si X es A_2 entonces Y es B_2

•

•

•

Regla n: Si X es A_n entonces Y es B_n

Hecho : X es A'

_____

Conclusión: Y es B'

Dadas n reglas "*si – entonces"* donde A' , $A_j \in F(X)$ para todo $j \in Nn_n$ y X, Y conjuntos de valores de las variables X y Y, esta forma de razonamiento es típica en los controladores construidos con lógica difusa. Este es un ejemplo de la generalización de la regla de inferencia *modus ponnes* para n-reglas difusas, de la forma SI-ENTONCES.

El método común para determinar B', es el método de interpolación, el cual consiste de dos pasos.

PASO 1.- Calcular el grado de consistencia de $rj_j(A')$, entre el hecho dado y el antecedente de cada *j*-regla "*si – entonces"* en términos de la intersección de los conjuntos asociados A' y A_j, esto es para cada $j \in N_n$, luego entonces:

$$r_j(A') = h(A' \cap A_j) \tag{34}$$

Usando la intersección difusa estándar:

$$r_j\ (A') = \sup_{x \in X} min[A'(x), A_j(x)] \tag{35}$$

PASO 2.- Calcular la conclusión B' por truncamiento de cada conjunto B_j por los valores de $rj_j\ (A')$, el cual expresa el grado con el cual el antecedente A_j es compatible con el hecho A', y tomando la unión de conjuntos truncados, esto es:

$$B' = \sup_{j \in N_n} min[r_j(A', B_j(y))] \tag{36}$$

Para todo $y \in Y$.

El método de interpolación es una caso especial de la regla de inferencia composicional, para ver esto mostramos que si R es una relación difusa en $X \times Y$ definida por

$$R(x, y) = \sup_{j \in N_n} min[A_j(x), B_j(y)] \tag{37}$$

Para todo $x \in X$y $y \in Y$, entonces la B' es igual a $A' \circ R$ donde "$\circ$" denota la composición ***sup−min***.

CAPITULO 4

4. PLATAFORMA DE PROGRAMACIÓN

En este capítulo se explica de manera general lo que es una plataforma de programación y su uso en la generación de software para control y simulación.

Un lenguaje de programación es lenguaje formal que nos permite interactuar con una computadora, microcontrolador, procesador, etc. Mediante el uso de un set de instrucciones que pueden ser usadas para implementar algoritmos y producir varios tipos de salidas. De acuerdo a algunos sitios web [13] [14], entre los lenguajes de programación más usados están: Python, C, C++, C#, Java, Swift, JavaScript, entre otros.

La dificultad para realizar cierto tipo de programas varía dependiendo del nivel del lenguaje, es decir, que tan cercano están los comandos o funciones de dicho lenguaje a las instrucciones del procesador en cuestión, generalmente esto se refiere a que tan cercano está lo que se escribe al código máquina (ensamblador), ver tabla 2. En el caso de los microcontroladores, es normal que se trabaje en lenguaje en C/C++ y para optimizar el código, ciertos segmentos se trabajen a nivel de ensamblador, por otro lado, en las computadores y dispositivos móviles, es muy común que se usen lenguajes de más alto nivel.

Lenguage ensamblador en MASM x86

```
fib:
    mov edx, [esp+8]
    cmp edx, 0
    ja @f
    mov eax, 0
    ret

    @@:
    cmp edx, 2
    ja @f
    mov eax, 1
    ret

    @@:
    push ebx
    mov ebx, 1
    mov ecx, 1
```

C++

```
unsigned int fib(unsigned int n) {
    if (n <= 0)
        return 0;
    else if (n <= 2)
        return 1;
    else {
        unsigned int a,b,c;
        a = 1;
        b = 1;
        while (1) {
            c = a + b;
            if (n <= 3) return c;
            a = b;
            b = c;
            n--;
        }
    }
}
```

```
    @@:
        lea eax, [ebx+ecx]
        cmp edx, 3
        jbe @f
        mov ebx, ecx
        mov ecx, eax
        dec edx
    jmp @b

    @@:
    pop ebx
    ret
```

Tabla 2. Comparación de 2 códigos para calcular el número de Fibonacci.

Esta selección de lenguaje suele ser definida dependiendo del tiempo que uno deseé dedicar al desarrollo del software, la eficiencia de ejecución del algoritmo y los recursos económicos para el pago de licencias, permisos, etc. Lenguajes de programación de muy alto nivel como MATLAB, LABVIEW suelen requerir unas cuantas líneas de código para realizar algoritmos complejos como el procesamiento de imágenes, sin embargo el tiempo de ejecución del programa, así como el costo de las licencias, lo hacen ineficiente en aplicaciones prácticas de índole profesional.

Algunos lenguajes permiten realizar software con interfaces gráficas en un nivel no tan bajo, es decir que aún nos permiten interactuar con ciertos procesos de la computadora para optimizar la ejecución del código, pero sin ser lenguajes enteramente de alto nivel que con solo un enunciado nos da un resultado sin la posibilidad de interactuar con los procesos que llevaron a obtener dicho resultado.

Ahora para escribir y compilar código, es preciso seleccionar una plataforma de programación. En las ciencias de la computación, una plataforma es un sistema que sirve como base para crear líneas de instrucciones o comandos con el fin de hacer funcionar diferentes módulos de hardware o software con los que la plataforma es compatible. Comúnmente se encuentran relacionadas directamente a un sistema operativo (Linux, Windows, etc.), sin embargo, también están ligadas a una familia de lenguajes de programación. También existen plataformas que permiten crear software ejecutable en múltiples sistemas operativos, por ejemplo Eclipse IDE o Netbeans son plataformas de programación para Java, que a su vez pueden usarse tanto en sistema

operativo Linux y Windows; por el contrario Visual Studio sería la plataforma para C++, C# y Visual Basic para crear software exclusivamente en el sistema operativo Windows.

Así como con las plataformas, algunos lenguajes de programación también pueden usarse en diferentes sistemas operativos, por ejemplo Python, Java, JavaScript, C++, entre otros. Lo que cambia en ocasiones es la ubicación o los nombres de los módulos importados del sistema (librerías, headers, etc.) o los nombres del sistema para referirse a periféricos de salida (nombres de puertos USB o Serial).

4.1. SIMULACIONES POR COMPUTADORA

Actualmente para probar que una máquina funciona de manera eficiente, es necesario que se realicen simulaciones por computadora de estos mismos. Ya sea de una parte o de varias partes de todo un sistema, las simulaciones nos permiten obtener un análisis de su comportamiento en diferentes ambientes antes de que se implemente en un sistema real.

El comportamiento de un sistema se realiza mediante un modelado matemático en un programa de computadora, que pueden ser tan pequeños como para ser ejecutado en dispositivos embebidos como micro-controladores, o programas de gran escala que son ejecutados por horas o días en súper-computadoras.

Cuando el programa de simulación es ejecutado, el algoritmo forma un análogo del comportamiento de sistema real, con los resultados presentados en forma de datos. Una simulación también puede tomar la forma de una imagen de gráficos por computadora que representa los procesos dinámicos en una secuencia animada [8].

Las simulaciones por computadora son generalmente usadas para el estudio del comportamiento dinámico de objetos o de sistemas en respuesta a condiciones que no pueden ser aplicadas de manera fácil o segura. Por ejemplo una explosión nuclear puede ser descrita por modelos matemáticos que incorpora variables como calor, velocidad y emisiones radiactivas. Otro

ejemplo es la reproducción de los movimientos de drones o sondas espaciales en vuelo (figura 7) [8] [9].

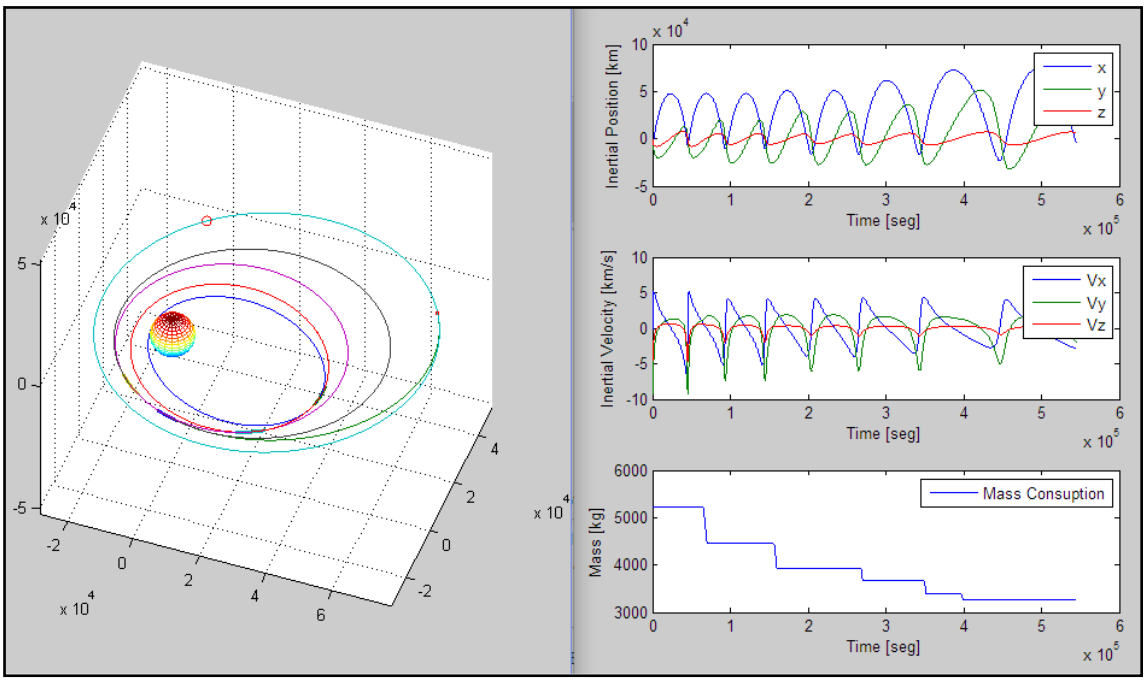

Figura 7. Simulación de la órbita de transferencia de un satélite [9]

4.2. WEB MAPPING

El “web mapping” es el proceso de usar mapas entregados por un Sistema de Información Geográfica (GIS por sus siglas en inglés). Los usuarios de esta clase de servicio tienen la posibilidad de obtener un determinado mapa de las coordenadas deseada [10]. El uso del internet para uso de las aplicaciones GIS ha estado creciendo en popularidad, parcialmente debido a la efectividad y aceptación de los usuarios [11]. Entre los servicios GIS Web más usados actualmente están OpenStreetMaps, Bing Maps y Google Maps.

CAPITULO 5

5. DESCRIPCIÓN GENERAL DEL ALGORITMO

El algoritmo para el seguimiento de la trayectoria es ejecutado en de la siguiente manera (Figura 8): Cada uno de los puntos (en latitudes y longitudes) de la trayectoria son proporcionados por el usuario y dependiendo de la posición GPS del vehículo así como de la orientación, calcula la distancia hacia el primer punto que conforma la trayectoria y el ángulo formado con la dirección del vehículo (Figura 9 y 10). El software posteriormente actualizará sobre el mapa la dirección y posición del gráfico que representa al vehículo, de manera que se pueda apreciar de manera visual los resultados del control.

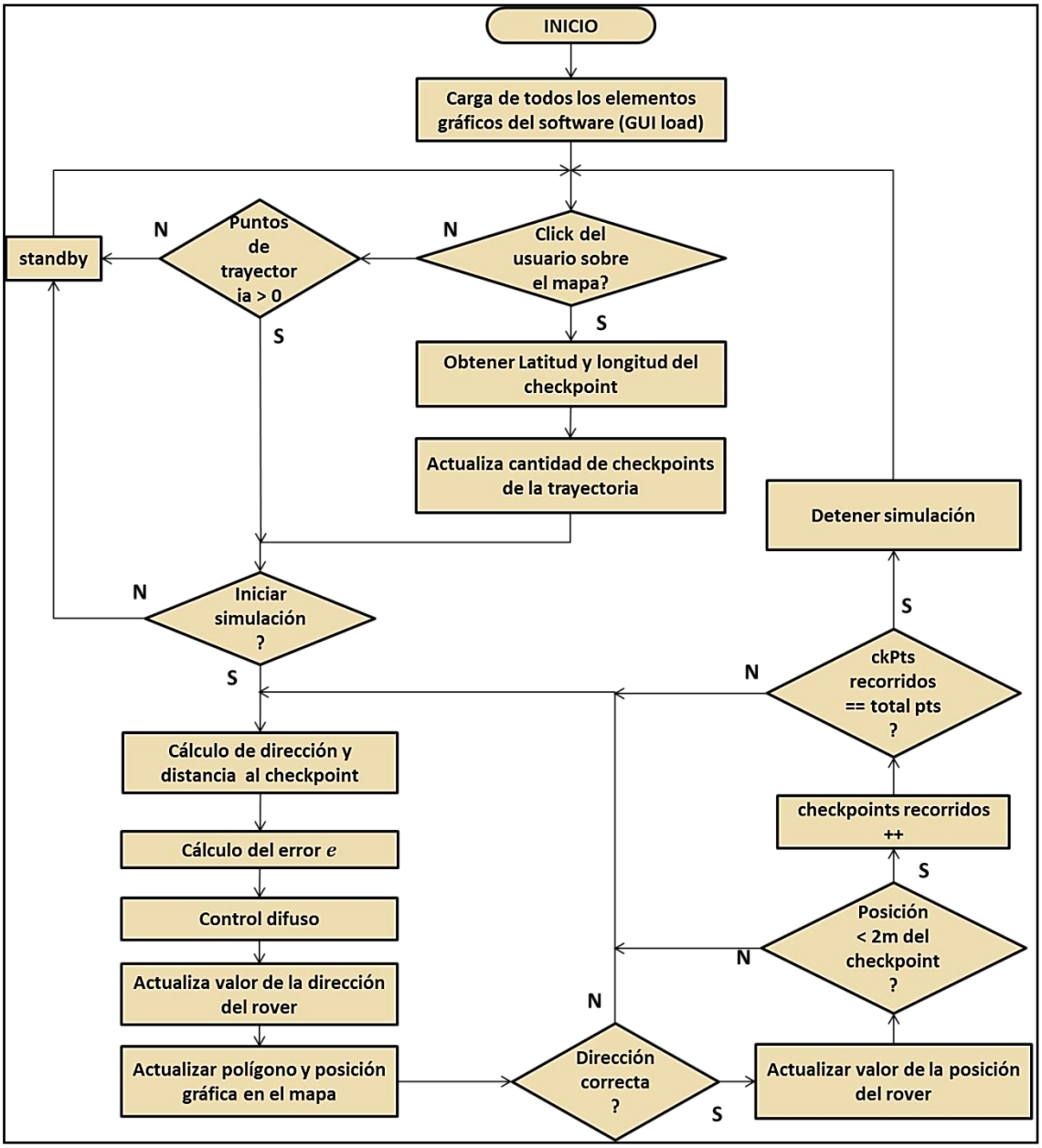

Figura 8. Diagrama de flujo del programa de simulación para el seguimiento de trayectorias

5.1. OBTENCIÓN DE LA TRAYECTORIA

En el programa de simulación, tanto la dirección como las coordenadas iniciales son proporcionadas por el usuario de manera manual en una caja de texto. Posteriormente cada punto o “checkpoint” del recorrido, que se desea que el vehículo terrestre simulado siga, también es proporcionado por el usuario usando el apuntador del mouse. Cada doble click sobre el mapa nos da la posición en latitud y longitud del apuntador, dicha posición se guarda en un array.

En la tabla 3 se muestra un pseudo-código de cómo se obtiene la trayectoria, los pasos son los siguientes: Paso 1: se obtiene las coordenadas del apuntador en latitud y longitud. Paso 2: Función para dibujar una ruta sobre el mapa usando las coordenadas actuales del apuntador (variable currentPosition) y las últimas coordenadas que se ingresaron (variable lastPosition). Paso 3: Guardamos en una arreglo las coordenadas. Paso 4: Actualizamos la variable lastPosition. Paso 5: Actualizamos el número de puntos que conforman la trayectoria (variable numOfPoints)

```
function googleMap_clickOnMap(var as mousePosition)
{
   currentPosition = var.getLatLng;                          //paso 1
   googleMap.addPath(currentPosition, lastPosition);         //paso 2
   arrayOfCheckPoints[numOfPoints] = currentPosition;        //paso 3
   lastPosition = currentPosition;                           //paso 4
   numOfPoints = numOfPoints + 1;                            //paso 5
}
```

Tabla 3. Pseudo-código donde se exponen los pasos para obtener los puntos de la trayectoria.

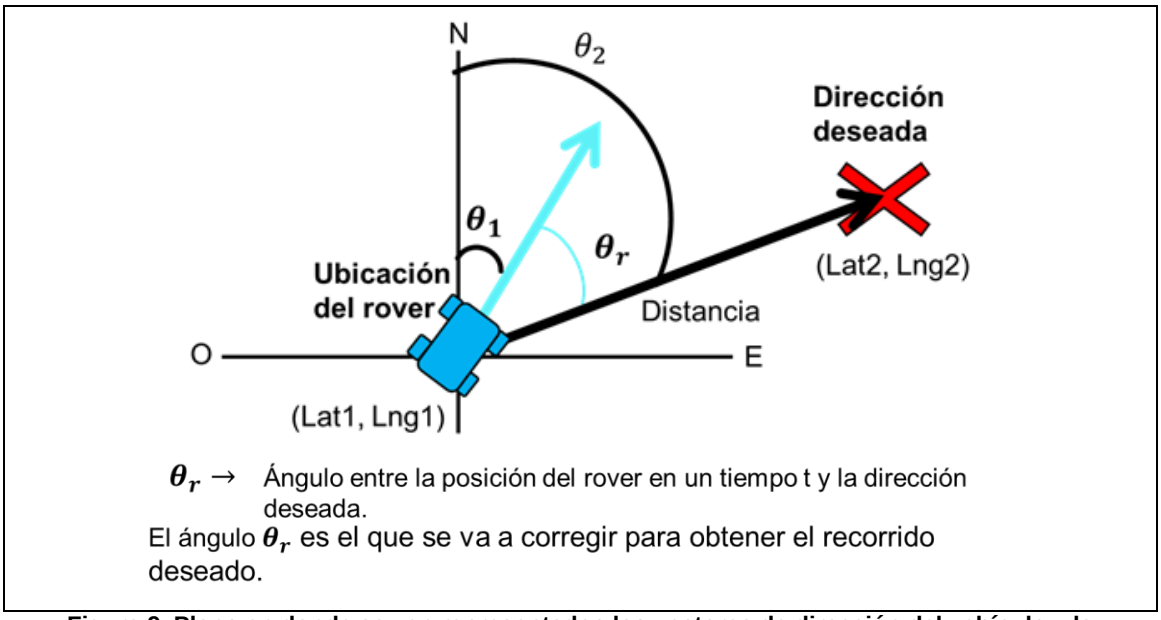

Figura 9. Plano en donde se ven representados los vectores de dirección del vehículo y la dirección deseada.

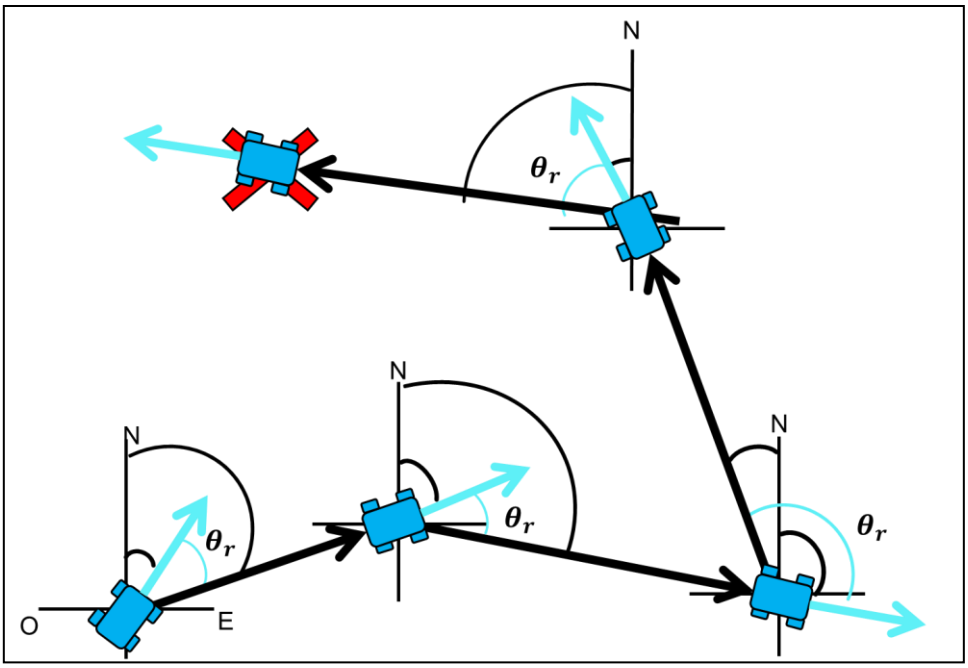

Figura 10. Trayectoria del vehículo en donde se aprecia cómo cambian los vectores de dirección y sus ángulos en cada uno de los checkpoints de la trayectoria

5.2. CÁLCULO DE LA DIRECCIÓN Y DISTANCIA HACIA CADA PUNTO DE LA TRAYECTORIA

Cuando queremos llegar a un determinado lugar, generalmente iniciamos tratando de ubicar nuestra posición de partida, ya sea una dirección, una calle, coordenadas en un mapa, etc. El siguiente paso sería determinar en qué dirección está nuestro punto de destino, para esto requerimos alguna referencia fija, un lugar o posición que no se mueva independientemente de nuestra posición o movimientos.

Los puntos de referencia más usados son las estrellas, el sol y el polo norte magnético de la Tierra, y los instrumentos desarrollados para usar dichos puntos de referencia son el sextante y la brújula respectivamente. El sextante es el instrumento que permite usar al sol y/o un astro para la navegación, este instrumento reemplazó al astrolabio por tener mayor precisión y ha sido durante varios siglos de gran importancia en la navegación marítima y también en la navegación aérea. La brújula es el instrumento que, mediante una aguja imantada, se orienta al norte magnético de la Tierra, y mediante el ángulo de declinación magnética se puede determinar el norte geográfico.

Ambos instrumentos se han implementado en su versión electrónica, dando lugar a los sensores brújulas usados ampliamente en sistemas autónomos y a los sensores de sol y estrellas que se usan en los satélites y sondas espaciales [13].

En nuestro software de simulación, el punto de inicio es determinado por el usuario, ingresando las coordenadas de manera manual. Los servicios de Web Mapping tienen como referencia fija el polo norte geográfico, por lo que en nuestra simulación no será requerido ningún tipo de corrección de declinación magnética. En trabajos futuros, si se quiere implementar este algoritmo a un vehículo autónomo real, será necesario tener que considerar dichas correcciones.

Ya con el punto de inicio, se toma el primer punto que conforma a la trayectoria, luego se calcula el ángulo con respecto al norte de dicha recta (figura 11), donde la latitud y longitud de cada punto están en unidades de ángulos.

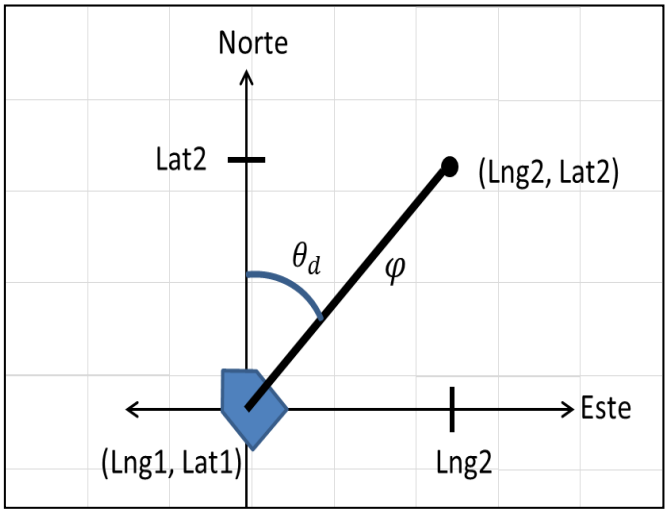

Figura 11. Determinación del ángulo formado entre la recta hacia el primer punto de la trayectoria y la recta que va de norte a sur cuando Lat2 (Latitud del punto de destino) es MAYOR a Lat1 (Latitud del vehículo)

De acuerdo a la figura 11 y usando trigonometría, determinamos la magnitud de φ entre los dos puntos

$$\varphi = \sqrt{(Lng_2 - Lng_1)^2 + (Lat_2 - Lat_1)^2} \quad (38)$$

Luego calculamos el ángulo θ_d

$$\Delta Lng = Lng_2 - Lng_1 \tag{39}$$

$$\sin\theta_d = \frac{\Delta Lng}{\varphi} \tag{40}$$

$$\theta_d = \sin^{-1}\left(\frac{\Delta Lng}{\varphi}\right) \tag{41}$$

Con esto tenemos la dirección en grados con respecto al norte hacia el primer punto de la trayectoria. Se usa función seno en vez de la función tangente porque tendríamos una indefinición cuando la latitud es cero, es decir, cuando la trayectoria va enteramente al Este o al Oeste.

Sin embargo, este razonamiento solo nos permite calcular adecuadamente el ángulo con respecto al norte siempre y cuando $Lat_2 > Lat_1$. Por lo que, para determinar el ángulo correcto cuando $Lat_2 < Lat_1$ (Figura 12) se realiza lo siguiente:

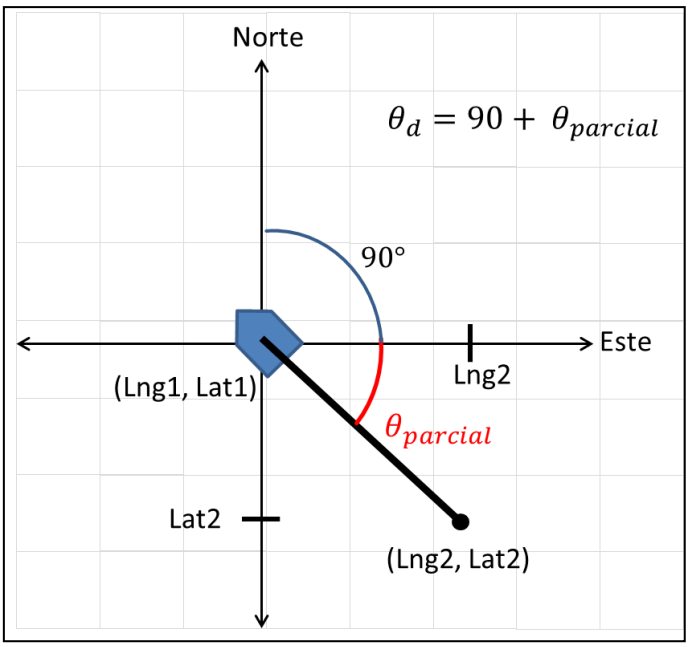

Figura 12. Determinación del ángulo formado entre la recta hacia el primer punto de la trayectoria y la recta que va de norte a sur cuando Lat2 (Latitud del punto de destino) es MENOR a Lat1 (Latitud del vehículo)

Se determina el ángulo parcial y luego el ángulo entre el norte y la recta:

$$\Delta Lng = Lng_2 - Lng_1 \tag{42}$$

$$\cos(\theta_{parcial}) = \frac{\Delta Lng}{\varphi} \tag{43}$$

$$\theta_{parcial} = \cos^{-1}\left(\frac{\Delta Lng}{\varphi}\right) \quad (44)$$

$$\theta_d = 90 + \theta_{parcial} \quad (45)$$

Con esto cubrimos el cálculo del ángulo formado entre el norte y el primer punto de la trayectoria.

Para saber la distancia que debe de recorrer el vehículo en metros, es preciso entender primero que si bien calculamos la magnitud de φ con la ecuación (38), en realidad estamos calculando el ángulo formado entre dos puntos, separados por una sección de la circunferencia terrestre (Figura 13).

Aunque hagamos a los mapas de manera plana, en términos generales la Tierra se puede representar como una esfera. Por lo tanto, para determinar la distancia en unidades de longitud entre dos puntos usando coordenadas, de debe de calcular el arco, usando el radio de la Tierra y el ángulo formado entre el punto de inicio y el primer punto de la trayectoria (magnitud de φ).

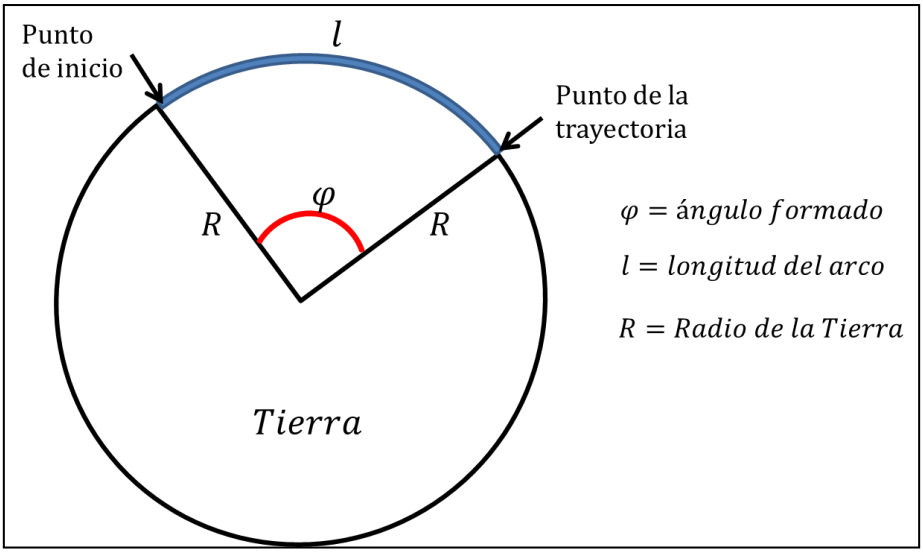

Figura 13. Representación de una trayectoria sobre la superficie de la Tierra.

Usando la ecuación del arco de un ángulo, tenemos:

$$dl = Rd\varphi \quad (46)$$

Integramos y evaluamos con l como longitud y φ como el ángulo en radianes:

$$\int_0^l dl = R\int_0^\varphi d\varphi \tag{47}$$

$$l = R\varphi \tag{48}$$

Para el cálculo final de la distancia en metros, recordemos que la magnitud de φ se debe de convertir a radianes y usar el radio de la Tierra en metros ($6{,}371{,}000.00\ m$).

Cabe mencionar que podríamos decir que en distancias cortas, este ángulo φ es muy chico en comparación con el radio terrestre, podríamos pensar en tomar los valores obtenidos de la ecuación (38) como distancias en metros de manera directa, sin embargo si existe un margen de error lo suficientemente grande como para realizar el cálculo del arco en metros.

Tomemos como ejemplo un GPS comercial con $\pm 1\ m$ de tolerancia en su ubicación. Podemos decir que tenemos un círculo dibujado en una superficie, con 1 m de radio (Figura 14). Con este dato podemos hacer el cálculo de la misma tolerancia pero en las unidades de latitud y longitud, es decir en grados.

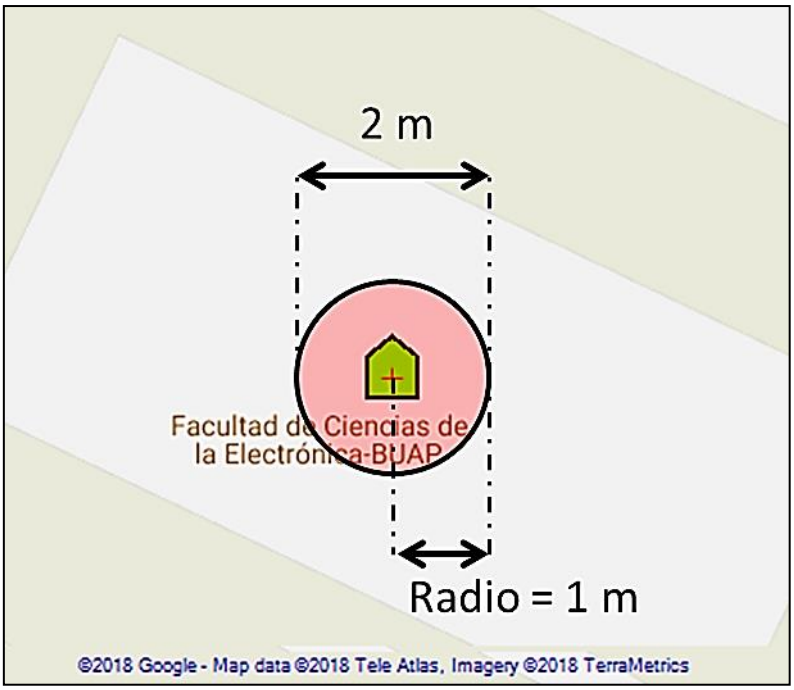

Figura 14. Representación gráfica de la tolerancia de un GPS comercial de $\pm 2\ m$

Tomando la ecuación (48) despejando φ para calcular su valor en radianes y posteriormente en grados:

$$\varphi = \frac{l}{R} \tag{49}$$

$$\varphi = \frac{(1\,m)}{(6371000\,m)} = 1.5696 \times 10^{-7}\ radianes \tag{50}$$

$$\varphi = (1.5696 \times 10^{-7})\left(\frac{180}{3.1416}\right) = 0.00000899° \tag{51}$$

$$\varphi \approx 0.00001\ ° \tag{52}$$

Entonces la tolerancia de $\pm 1\,m$ en grados de un GPS comercial es de aproximadamente $\pm\ 0.00001°$

Para darnos una idea de lo importante que puede ser estas variaciones, ingresaremos este dato a los valores de posición para poder dibujar la ruta del vehículo terrestre simulado y visualizar las variaciones que tendría en el mundo real. Como vemos en la figura 15, al aumentar la tolerancia del GPS, también se verá afectado el seguimiento de la trayectoria, por lo que es si resulta importante tomar a considerar el radio de la Tierra para minimizar los errores de seguimiento.

Figura 15. Ruta del vehículo aplicando las tolerancias de los GPS comerciales. A la izquierda es una tolerancia de $\pm 1\,m$, a la derecha una tolerancia de $\pm\ 2\,m$

Al final tenemos la distancia en metros y la dirección con respecto al norte en grados hacia el primer punto. Una vez que se llegue al primer punto, se repite el proceso del cálculo del ángulo y distancia, actualizando las coordenadas de inicio y fin.

5.3. CONTROL DIFUSO DE LA DIRECCIÓN DEL VEHÍCULO TERRESTRE

A continuación se presenta un diagrama de bloques del control difuso de la dirección del vehículo en el programa de la simulación.

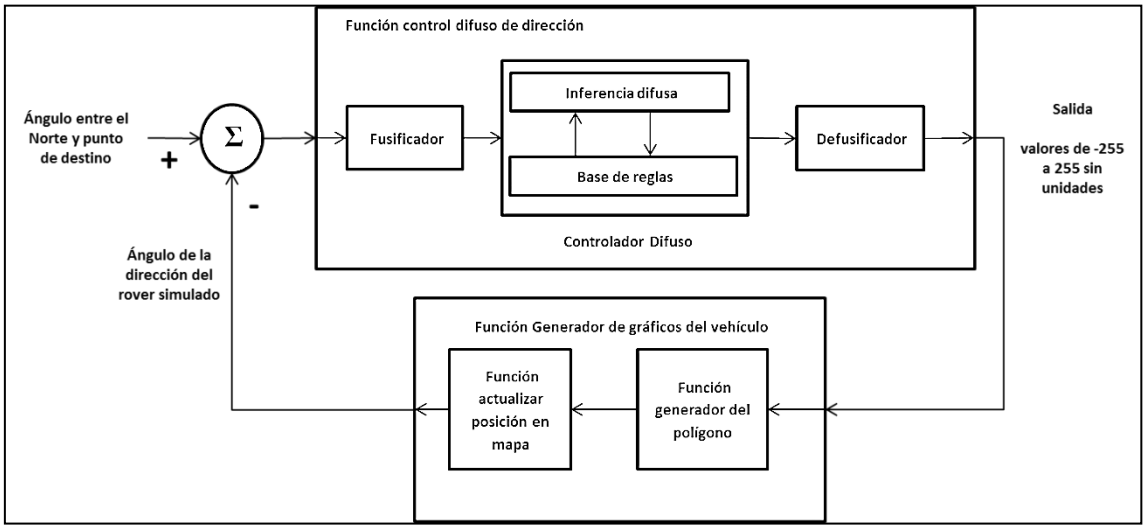

Figura 16. Diagrama de bloques del control difuso

Si observamos la figura 16, observamos que la variable de entrada es el resultado de resta entre la referencia (dirección deseada) y la posición del vehículo (dirección de la brújula), también vemos que los valores de la salida son usados en una parte del código que se dedica a actualizar al polígono del vehículo y su posición en el mapa de google. Al no tener un vehículo real con actuadores reales, debemos generar un gráfico que nos pueda mostrar que en efecto el control está funcionando al cambiar las características visuales de dicho gráficos, es decir, que lo haga “girar” y cambiar su posición en el mapa en la dirección que nosotros deseamos.

Nuestro polígono no necesariamente debe de tener forma de algún tipo de vehículo terrestre y como en este trabajo no se están tomando en cuenta factores como la deformación de las ruedas al movimiento o el terreno en sí, basta con solo saber a qué dirección se dirige cuando se esté moviendo, por lo

que se genera un polígono tipo flecha de 5 puntos (Tabla 4), cada punto debe de ser dibujado en función de lo valores de la variable “compass_”, que sería la brújula, de manera que un cambio de dirección haga girar cada punto del polígono.

```
Private bitmap drawRoverPolygon(compass_)
{
   compass_ = compass_ * 3.1416 / 180; //Convertir a radianes
   pt1.X = centerPt.X + (r1_ * Math.Cos(1.5708 + compass_)); //primer punto del
   pt1.Y = centerPt.Y - (r1_ * Math.Sin(1.5708 + compass_)); //polígono

   pt2.X = centerPt.X + (r2_ * Math.Cos(2.4434 + compass_)); //segundo punto del
   pt2.Y = centerPt.Y - (r2_ * Math.Sin(2.4434 + compass_)); //polígono

   pt3.X = centerPt.X + (r2_ * Math.Cos(3.8397 + compass_)); //tercer punto del
   pt3.Y = centerPt.Y - (r2_ * Math.Sin(3.8397 + compass_)); //polígono

   pt4.X = centerPt.X + (r2_ * Math.Cos(5.5856 + compass_)); //cuarto punto del
   pt4.Y = centerPt.Y - (r2_ * Math.Sin(5.5856 + compass_)); //polígono

   pt5.X = centerPt.X + (r2_ * Math.Cos(0.6981 + compass_)); //quinto punto del
   pt5.Y = centerPt.Y - (r2_ * Math.Sin(0.6981 + compass_)); //polígono

   Point polygonPoints[5][5]={pt1, pt2, pt3, pt4, pt5}; //arreglo de 5 puntos xy
   //crea un nuevo gráfico en un bitmap en blanco
   Graphics g = new graphics.fromImage(bitmap);
   //pinta al polígono de verde
   graphics.FillPolygon(Brushes.LawnGreen, polygonPoints);
   //dibuja al contorno de negro
   graphics.DrawPolygon(Pen.Black, polygonPoints);

   return bitmap; //regresa un mapa de bits del polígono dibujado
}
```

Tabla 4. Pseudo-código donde se indica los pasos usados para dibujar el polígono.

Como se mencionó en el capítulo 5, la posición del vehículo inicial será asignada por el usuario al inicio y será actualizada por el programa con una simple suma de valores a dicha posición, ya en un vehículo real, esta posición será proporcionada por un módulo GPS. Como se manejan los mismos valores de tolerancia que un GPS comercial ($de \pm 2\ a\ \pm 5\ metros$), solo se implementará el control difuso a la dirección, cuando su posición simulada se encuentre a 2 metros o menos del punto, se detendrá (Tabla 5).

```
If (checkPoints < totalCheckPoint)
{
   If (beta  != 0) //referencia diferente a cero, no está en la dirección correcta
   {
      //Usa la salida del control difuso, para cambiar el giro del vehículo terrestre
      roverCompass = roverCompass + outputFLC ;
   }
   Else
   {
      If (distanceToTgt >= 2) //distancia al checkpoint igual o mayor a 2 metros
      {
         r = 20; //incremento deseado en metros
         rad  = (roverCompass * 3.1416) / 180;     //convertir a radianes
         dist_y = r * math.Cos(rad);               //incremento en metros en 'y'
         dist_x = r * math.Sin(rad);               //incremento en metros en 'x'
         incLat = dist_y / earthRadius            //convierte a rad en latitud
         incLng = dist_x / earthRadius            //convierte a rad en longitud

          //suma esos incrementos en Lat y Lng a la posición del rover
          roverLatitude = roverLatitude +( incLat*180/math.Pi)
          roverLongitude = roverLongitude + (incLng*180/math.Pi)
      }
      Else
      {   //llegó, actualiza el contador de checkpoints
          checkPoints = checkPoints + 1;
      }
   }
else
{ //El conteo de checkpoints es mayor a num total, llego al final de la trayectoria
}
drawRoverPolygon(roverCompass);    //actualize al poligono del vehículo terrestre
```

Tabla 5. Pseudo-codigo de la actualización de la posición y dirección del polígono en el mapa

5.3.1. FUNCIONES DE PERTENENCIA

Para poder definir a nuestros conjuntos difusos, es preciso que primero se escoja que variable será la entrada al sistema, es decir, la referencia que se usará. Comúnmente, la referencia más usada en navegación es la cantidad de grados a la que nosotros nos encontramos del punto hacia dónde queremos ir, por lo que dicho ángulo en grados será nuestra entrada y se calcula mediante el punto suma justo antes de entrar al bloque del control (Figura 16).

El siguiente paso es seleccionar el tipo de funciones de pertenencia que se tendrán para cada conjunto difuso y el intervalo en el que quedarán definidas.

Entonces decimos que un vehículo puede estar mirando hacia el centro cuando se encuentra en dirección frontal al punto de destino, y si durante el trayecto se desvía, este puede girar o a la izquierda o a la derecha. Dependiendo que tan a

la izquierda o a la derecha se pudiera desviar, es la velocidad de giro que el vehículo deberá de tener para posicionarse en la dirección correcta lo más rápido posible.

Siguiendo este razonamiento, podemos declarar la variable lingüística de la entrada como "*Dirección*" y sus valores lingüísticos como: "*Izquierda, Centro, Derecha*". Se escogen como funciones de pertenencia las de tipo gaussiana y sigmoidal (Figura 17).

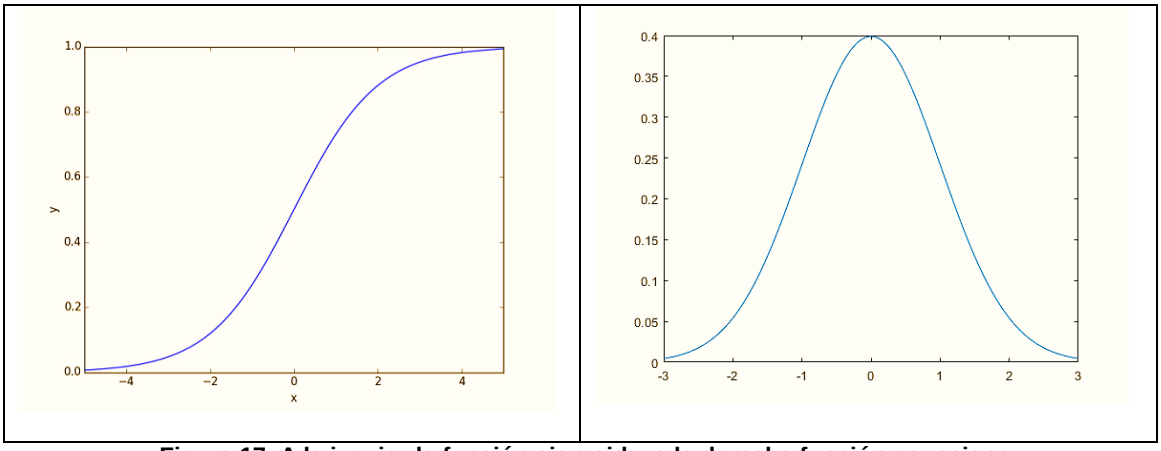

Figura 17. A la izquierda función sigmoid y a la derecha función gaussiana

Los intervalos en donde estas funciones estarán definidas son los grados de movimiento con respecto al norte que tendrá tanto el vehículo terrestre como la dirección hacia cada uno de los puntos de la trayectoria. Tomamos como punto central el valor de ángulo 0, lo que quiere decir que estamos mirando justo en la dirección que deseamos ir, si nos desviamos girando a la derecha consideraremos todos eso ángulos con valor positivo, por el contrario si nos desviamos girando a la izquierda, consideraremos todos esos ángulos con valor negativo.

Entonces podemos definir 3 intervalos de valores, cuando consideramos que estamos dirigiéndonos a la izquierda (usando una función sigmoid invertida), cuando estamos en el centro (usando una función gaussiana) y cuando nos dirigimos a la derecha (usando una función sigmoid).

Las funciones de pertenencia con sus respectivos intervalos se muestran en la siguiente ecuación:

$$Direccion(x) = \begin{cases} \frac{1}{1+e^{x}}; & para\ x < a \\ e^{-x^{2}}; & para\ a \leq x \leq b \\ \frac{1}{1+e^{-x}}; & para\ x > b \end{cases} \quad (53)$$

Sin embargo estas funciones mapean su dominio de valores de entrada a un rango de valores de salida, es decir que aunque se evalúen dichas ecuaciones con los valores de los ángulos que se manejan en nuestro control, el resultado será inadecuado. Por lo que es requerido realizar una conversión de escalas de los valores de entrada $[-180, 180]$ en un rango de valores que nos den las curvas deseadas para que estas ecuaciones nos den un resultado en el intervalo de $[0, 1]$.

Tratándose de solamente una conversión de un rango de valores a otro de manera proporcional, una conversión lineal es suficiente. Usando la ecuación de línea característica en su forma punto-pendiente podemos realizar dicha conversión.

$$y - y_1 = m(x - x_1) \quad (54)$$

$$y - y_1 = \left(\frac{y_2 - y_1}{x_2 - x_1}\right)(x - x_1) \quad (55)$$

Tanto para las funciones sigmoid como para la función gaussiana, el rango de valores que se usan para obtener la curva van de $[-5, 5]$. Entonces convertimos el rango de valores $[-180, 180]$ en cada uno de dichos rangos de manera proporcional.

Evaluamos la función de línea:

$$y - (-5) = \left(\frac{5-(-5)}{180-(-180)}\right)(x - (-180)) \quad (56)$$

$$y + 5 = \left(\frac{10}{360}\right)(x + 180) \quad (57)$$

$$y = \left(\frac{10}{360}\right)(x + 180) - 5 \tag{58}$$

Al final la ecuación requerida para el escalamiento de los valores de la dirección al rango de valores usados en las funciones queda:

$$y = \frac{10}{360}x \tag{59}$$

Donde "x" son los grados de la dirección y "y" los valores en el rango de las funciones de pertenencia. Esta conversión de escala se debe de hacer cada vez que se calcule el grado de pertenencia de la entrada.

En la figura 18 se muestra una gráfica de las funciones de pertenencia con el escalamiento de valores para uso en un intervalo de $-180\ a\ 180$.

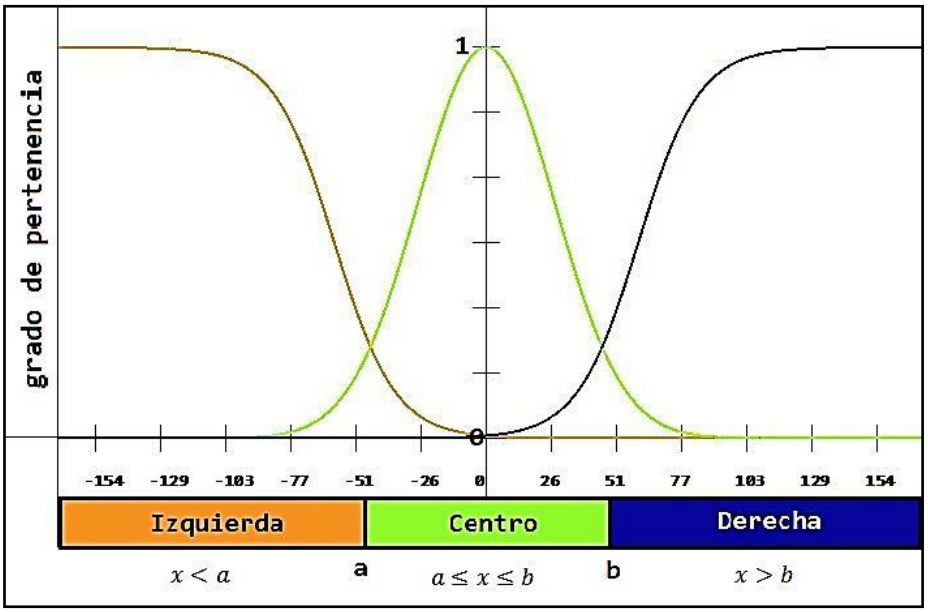

Figura 18. Conjuntos difusos que representan los estados para la variable lingüística dirección (d) = {izquierda, Centro, Derecha}

El mismo procedimiento se aplica para las funciones de pertenencia de la variable de la salida. Hasta este punto, solo hemos definido lo que deseamos controlar (la dirección del vehículo terrestre simulado), falta por definir la variable de salida.

En la vida real, para el control de la velocidad o los grados de dirección de algunos servos y motorreductores, generalmente se usan señales de modulación por ancho de pulso (PWM) y algunas librerías de programación de

microcontroladores usan valores de 0 a 255 (un entero de 8-bit sin signo) para representar un mínimo y un máximo de dicho ancho del pulso. Por lo que se escoge este intervalo de valores para la variable de salida, con la notable diferencia de que el rango abarcará de $[-255, 255]$. El signo negativo realmente no significa que un pulso de PWM sea negativo o invertido, solamente le indicará al programa si el vehículo terrestre debe de girar a la izquierda.

Entonces podemos definir la variable lingüística de la salida como "*Velocidad de giro*" y sus variables lingüísticas como "*Bajo, Medio y Alto*", los valores en donde estarán definidas las funciones de pertenencia son los valores de -255 a 255 para su uso con librerías de PWM y así variar la velocidad de un motor en el mundo físico, el signo negativo solo indicará la inversión de giro del motor. En la figura 19 se muestra una gráfica de las funciones de pertenencia con el escalamiento de valores para uso en el intervalo $[-255, 255]$.

Es importante mencionar que cuantificar la dinámica de un proceso con variables lingüísticas no es una tarea fácil, esta requiere de una depuración durante varias pruebas para obtener la mejor respuesta posible. El tener un mejor entendimiento de la dinámica de los procesos lleva a una mejor cuantificación de las variables lingüísticas que puedan adecuadamente medir la dinámica del sistema para que el controlador difuso pueda tomar la decisión correcta al tiempo adecuado [15].

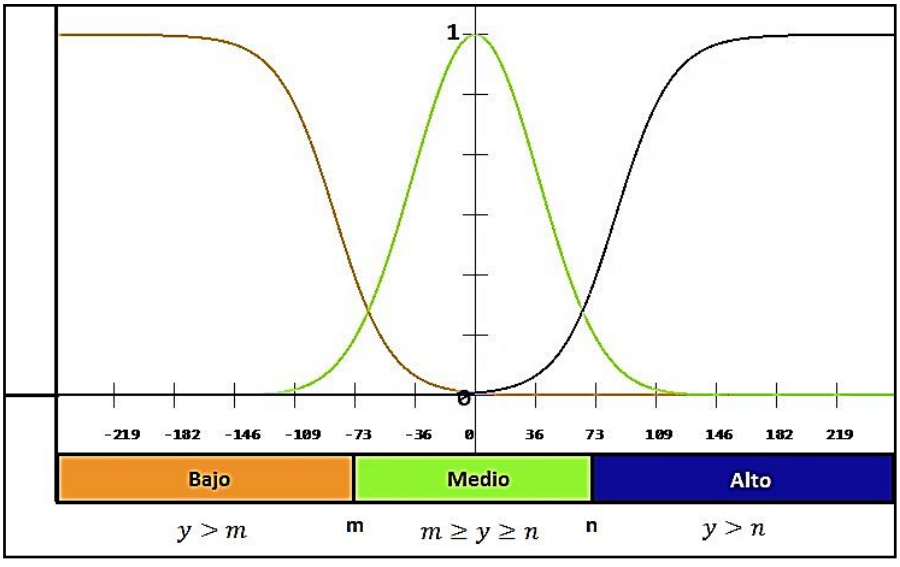

Figura 19. Conjuntos difusos que representan los estados para la variable lingüística Velocidad de giro (v) = {bajo, medio, alto}

5.3.2. REGLAS DE INFERENCIA E INFERENCIA DIFUSA

Como se mencionó en la sección 3.2., la forma de las reglas de inferencia es del tipo Mamdani, es decir, tienen la forma de IF {entrada/antecedente} THEN {salida/consecuente}.

En el formato Mamdani se comienza por escribir reglas básicas y después con la experiencia del experto depurarlas. La inferencia difusa es el proceso mediante el cual se obtiene como conclusión un conjunto difuso a partir de premisas tomadas de las reglas de inferencia.

La inferencia difusa permite interpretar las reglas de tipo IF-THEN de una base de reglas, para obtener valores de salida a partir de los valores de entrada del sistema.

En nuestro sistema podemos definir el número de reglas posibles como:

$$N_R = {N_l}^{N_v} \tag{60}$$

Donde:

$N_R - n\acute{u}mero\ de\ reglas$

$N_l - n\acute{u}mero\ de\ valores\ ling\ddot{u}isticos\ por\ cada\ variable$

$N_v - n\acute{u}mero\ de\ variables\ ling\ddot{u}isticas\ del\ sistema$

Entonces tenemos un total de 2 variables lingüísticas, que son "*dirección*" como la variable de entrada y "*Velocidad* de giro" como la salida. Por cada variable tenemos 3 valores posibles, "*izquierda, centro, derecha*" y "*bajo, medio y alto*" respectivamente. Por lo que el número total de reglas posibles sería:

$$N_R = (3)^2 = 9 \tag{61}$$

Un total de 9 reglas para todos los casos posibles.

Las reglas para todos los casos posibles son las siguientes:

	Set de reglas			
1	IF	*izquierda*	THEN	*bajo*
2	IF	*izquierda*	THEN	*medio*
3	IF	*izquierda*	THEN	*alto*
4	IF	*centro*	THEN	*bajo*
5	IF	*centro*	THEN	*medio*
6	IF	*centro*	THEN	*alto*
7	IF	*derecha*	THEN	*bajo*
8	IF	*derecha*	THEN	*medio*
9	IF	*derecha*	THEN	*alto*

Ya que no existe otra variable en el sistema, los valores de pertenencia únicos k_i son los valores de definición para *bajo, medio* y *alto* fusificados. El número del subíndice de las k's es determinado por el orden en que se escribieron las reglas de inferencia, así la k con subíndice 7 corresponde a la relación que genera la séptima regla de inferencia que evalúa una dirección actual *derecha.*

Para este trabajo, los valores de definición $C_1, C_2\ y\ C_3$ sin fusificar para bajo, medio y alto son:

Bajo	C_1	-200
Medio	C_2	0
Alto	C_3	200

Como ya se ha mencionado, el cuantificar valores no es tarea fácil [15], por lo que estos valores se pueden ajustar dependiendo de las necesidades del sistema a controlar.

Una vez obtenido todas las k_i, se realiza una comparación entre éste y un nuevo conjunto difuso propuesto.

El nuevo conjunto difuso llamado *Hecho* (figura 20) permite una atenuación en la salida, esto es importante pues disminuye posibles sobresaltos en el sistema, por ejemplo si se piensa usar para controlar a un vehículo real, las vibraciones ocasionadas por el terreno o si existe interferencia magnética, la brújula y el GPS pueden ocasionar dichos sobresaltos. De la misma forma que

los conjuntos anteriores el *Hecho* es un conjunto difuso tipo gaussiano (Figura 20).

Al fusificar la entrada d encontrando su valor de pertenencia en los conjuntos *Izquierda*, *Centro* y *Derecha* respectivos, se encuentra también el valor de pertenencia de dichas entradas en el conjunto difuso *Hecho*. El programa fusifica de forma paralela la entrada d usando el conjunto *Hecho* y almacena los valores de pertenencia correspondientes en h.

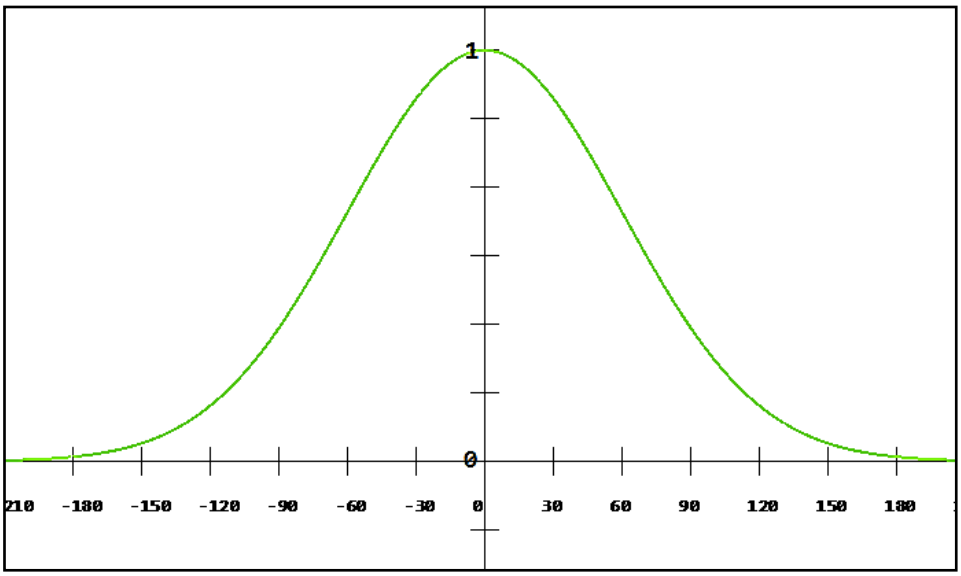

Figura 20. Conjunto difuso *Hecho*, asociado a la variable h

La comparación entre k_i y h se realiza usando el operador intersección como lo muestra la ecuación (62).

$$u_i = \min(k_i, h) \tag{62}$$

Donde

$u_i = consecuente\ final\ y\ salida\ no\ fusificada\ del\ sistema$

Esta operación se realiza para cada uno de los valores de pertenencia únicos obtenidos de la evaluación del set de reglas.

5.3.3. DEFUSIFICACIÓN

Para la elección de un método de defusificación, los aspectos referentes a la eficiencia computacional y a la facilidad de adaptación son clave para vehículos no tripulados; por estas razones, para nuestro sistema, la opción más apropiada es el método de centro de áreas. Los valores de u obtenidos y el número total de reglas de inferencia se emplean en la defusificación de la salida.

Se usa la siguiente ecuación, explicada en la sección 3.5., para obtener el centro del área, es decir, la salida fusificada del sistema:

$$d_{CA}(u) = \frac{\sum_{i=1}^{n} z_{oi} u_i(z)}{\sum_{i=1}^{n} u_i(z)} \tag{63}$$

Al tener 3 valores de definición para *bajo, medio y alto* ($C_1, C_2\ y\ C_3$) y 9 valores consecuentes resultado de la base de reglas ($u_1, u_2, \dots, u_9$), tenemos entonces:

(64)

$$d_{CA}(u) = \frac{u_1 C_1 + u_2 C_2 + u_3 C_3 + u_4 C_1 + u_5 C_2 + u_6 C_3 + u_7 C_1 + u_8 C_2 + u_9 C_3}{u_1 + u_2 + u_3 + u_4 + u_5 + u_6 + u_7 + u_8 + u_9}$$

El resultado obtenido de la ecuación (64) ya puede ser usado en librerías de PWM para microcontroladores que ocupen valores de 0 255, tomando en cuenta que si l resultado es negativo, solo indica que requiere una inversión de giro de los motores para cambiar la dirección hacia donde se dirige el vehículo terrestre.

Este resultado va directamente hacia la simulación, cambiando el valor de la variable brújula del polígono, haciendo “girar” al vehículo terrestre.

5.3.4. SIMULACIÓN FIS (Fuzzy Inference Systems)

Para corroborar que los datos de salida del control difuso son los que se esperan, se comprueba con una herramienta ya estandarizada de MATLAB llamada Fuzzy Inference System (FIS), en donde se ingresa el tipo de función

de pertenencia, el dominio en donde estas funciones están definidas y los valores de definición (Figura 21, 22 y 23). En la Tabla 6 se muestran los resultados de la herramienta FIS

Valor de entrada	Salida del control difuso	Salida FIS MATLAB
157	-100	-99.9
69.2	-99	-98.4
-64	45	-42

Tabla 6. Tabla comparativa de los resultados obtenidos por el control difuso y los obtenidos por la herramienta FIS de MATLAB.

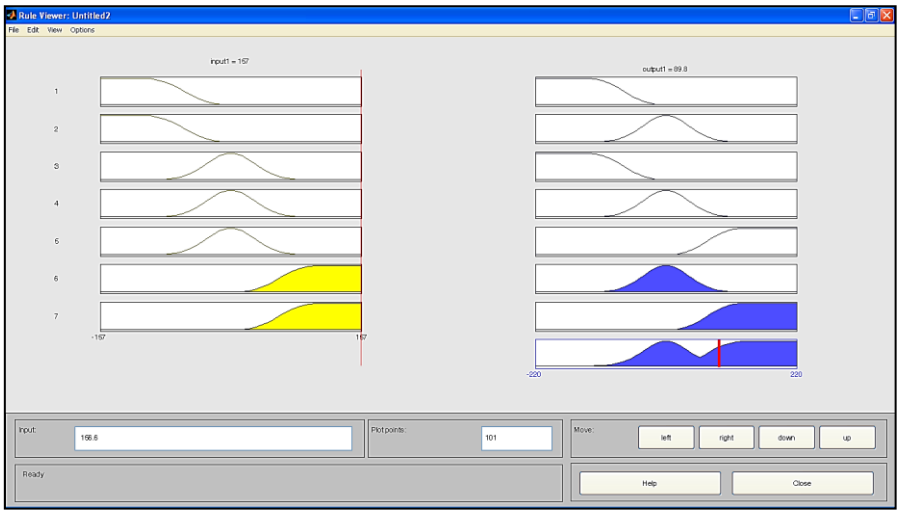

Figura 21. Captura de pantalla de la herramienta FIS de MATLAB, en donde se ingresa un valor numérico de entrada y se obtiene el resultado del sistema difuso.

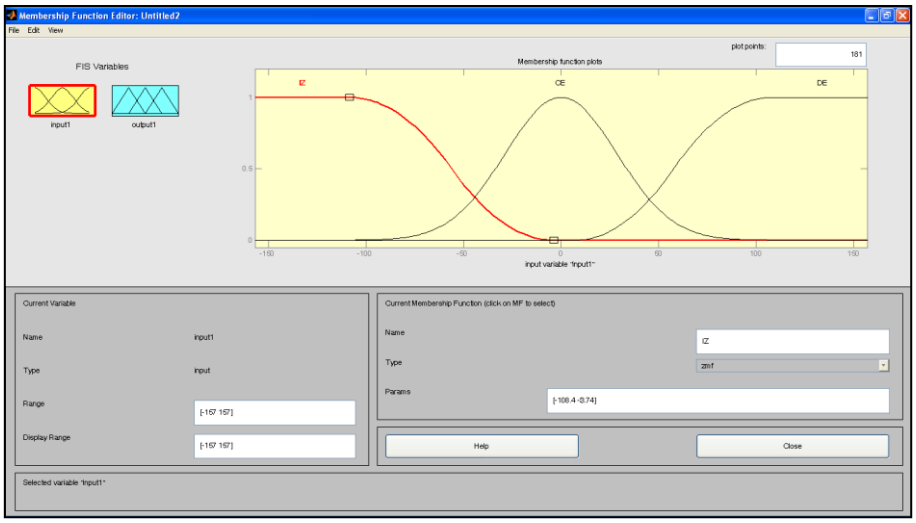

Figura 22. Captura de pantalla de la herramienta FIS de MATLAB, en donde se definen los conjuntos difusos para la variable de entrada.

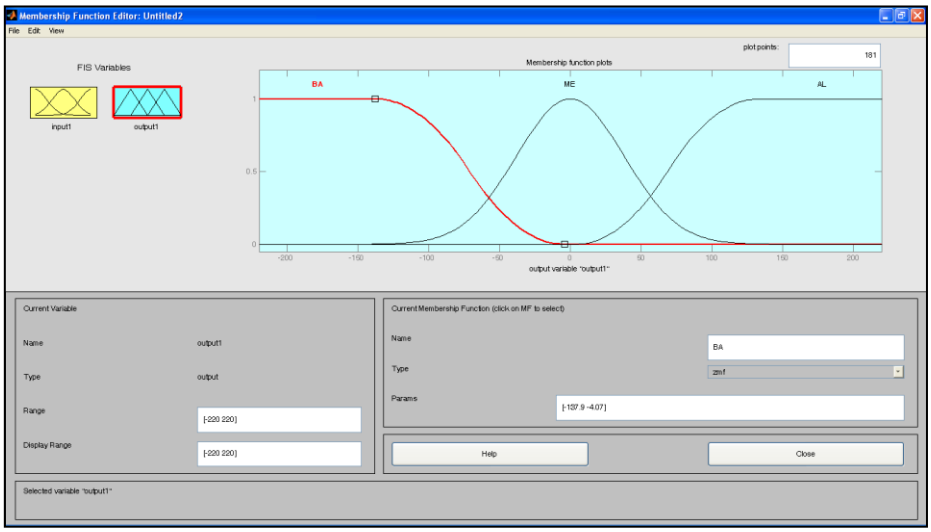

Figura 23. Captura de pantalla de la herramienta FIS de MATLAB, en donde se definen los conjuntos difusos para la variable de salida.

CAPITULO 6

6. ADQUISICIÓN DE DATOS

Un subsistema se puede definir como una serie de componentes que trabajan en conjunto para conseguir un fin común pero que no proveen una función útil por si solos, ya que se requieren integrar con otros subsistemas para conformar un sistema [16].

En este capítulo describiremos de manera general el desarrollo de un subsistema de adquisición y transmisión de datos usando un sensor brújula, un GPS y un celular con sistema operativo android. Esto se realiza con el fin de obtener datos reales para alimentar a nuestro control difuso y comparar los datos de la salida con los obtenidos de la simulación, sin embargo se hace la aclaración que ESTE SUBSISTEMA SIRVE ÚNICAMENTE PARA COMPROBAR EL FUNCIONAMIENTO DEL CONTROL DIFUSO y no para comprobar el seguimiento de las trayectorias. Para comprobar la simulación del seguimiento de trayectorias por completo, es requerido más subsistemas para desarrollar un vehículo terrestre físico.

La adquisición de datos es el proceso por que cual medimos un determinado fenómeno físico del mundo real (aceleración, distancia, campo magnético) usando una unidad digital de procesamiento, sea una PC, microcontrolador, DSP, etc. El proceso inicia convirtiendo una cierta condición física, como la temperatura, en una señal eléctrica analógica usando un transductor. Esta señal eléctrica analógica entra a un convertidor analógico/digital (ADC), donde se muestrea, cuantifica y codifica para obtener una señal digital que ya puede ser manipulada por una computadora.

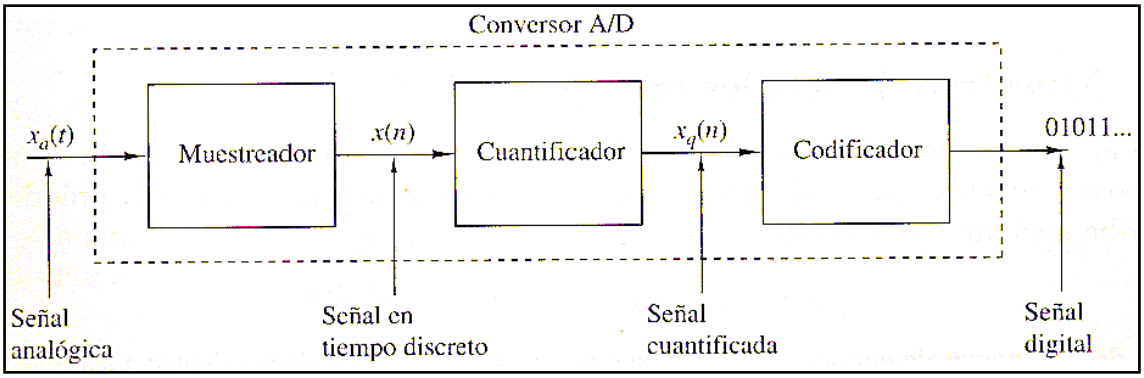

Figura 24. Diagrama de bloques de una conversión analógica/digital

La mayoría de los microcontroladores ya tienen incluido en su arquitectura módulos convertidores analógicos/digitales, sin embargo también se venden por separado como chips o módulos. En la actualidad (marzo 2018) muchos fabricantes de componentes electrónicos ya venden sensores como módulos que, entre otros componentes para acondicionar la señal, incluyen un microcontrolador para procesar la señal y entregar la información mediante algún protocolo de comunicación digital (SPI, I2C, UART, etc.). Esta clase de sensores-modulo son más populares porque son menos susceptibles al ruido electrónico y porque liberan de trabajo de computo al microcontrolador que recibe los datos.

Ahora, los sistemas y subsistemas de adquisición de datos pueden ser tan sencillos como un único microcontrolador junto con un sensor o más elaborados como estaciones completas de monitoreo de múltiples sensores digitales comunicándose con procesadores ARM. Por ejemplo, los celulares son mini-PCs que reciben y procesan constantemente al información recabada de sus sensores para mejorar la interacción del usuario con las diversas funciones del sistema operativo (voltear la pantalla, ubicación en mapas, aplicaciones varias), y se puede desarrollar software de uso específico para estos dispositivos para que su única función sea la de recabar y transmitir su información hacia otra PC. Cuando los sensores incluidos en los celulares no son suficientes, existen sensores con conexión USB o Bluetooth para interactuar con el software del dispositivo.

En el subsistema de adquisición de datos que se desarrolla, se escogen dos dispositivos que trabajarán en conjunto.

El primer dispositivo es un smartphone por las características mencionadas anteriormente, además que ya vienen con módulos de comunicación inalámbrica WIFI, lo que permite implementar métodos de envió/recepción de datos usando protocolos TCP/IP o UDP. No solo eso, también tienen al menos una cámara de video de buena resolución, que aunque no se tenga un propósito en este trabajo, tiene buenas perspectivas futuras en trabajos posteriores.

El segundo es un microcontrolador ATMega328 con un sensor digital brújula HMC5883L, ya que la mayoría de los celulares no contienen una brújula incluida, por lo que se agregará este sensor al subsistema de adquisición de datos de manera externa para obtener la dirección.

Es importante mencionar que en el desarrollo de un UGV real, se requieren más tipos de sensores como por ejemplo acelerómetros o sensores medidores de distancia (ultrasónicos, laser) para impedir obstáculos. Esto para aumentar la percepción del vehículo con respecto a su entorno y en el caso de los vehículos semi-autónomos o totalmente autónomos, contar con más información para la toma de decisiones.

6.1. SMARTPHONES

Un Smartphone puede ser definido como una mini-PC para uso personal que actualmente tiene poderosas capacidades de adquisición de datos mediante los diversos sensores que tiene embebidos (acelerómetros, GPS, barómetro, etc.) así como sus altas capacidades de comunicación, ya sea usando la red celular, el punto de acceso Wi-Fi o el bluetooth [17] [18].

Cuando una de estas mini-computadoras es agregada a los vehículos no tripulados, son capaces de recolectar valiosos datos que pueden ser guardados y posteriormente recuperados junto con el dispositivo, o pueden ser

transmitidos in situ, esta parte del subsistema se conoce como OBC (computadora de abordo por sus siglas en ingles). La selección de la OBC de cualquier vehículo no tripulado depende de la cantidad de datos que se planean obtener, procesar, guardar y/o enviar; siendo la cantidad de memoria RAM, el procesador y la cantidad de periféricos de comunicación (junto con los respectivos tamaños de los buffers de I/O) los factores con mayor relevancia a considerar.

Para el subsistema desarrollado en este trabajo, se usa un smartphone HTC Desire 626s (Figura 25) [20], aparte de tener buenas características (Tabla 7) para cumplir con las demandas del subsistema, es el dispositivo móvil con el que se cuenta al momento, sin embargo para trabajos posteriores no se descarta el uso de otros smartphones para optimizar el funcionamiento.

Los sensores requeridos para probar los resultados de la simulación del control difuso son un GPS y el sensor brújula. Este último no viene incluido en el hardware del smartphone, por lo que se deberá de implementar el sensor brújula por separado. Para la transmisión de datos, se usa el módulo Wi-Fi como punto de acceso usando el protocolo UDP para transmisión de grandes paquetes de datos vía Ethernet/Wi-Fi.

Figura 25. Smartphone HTC Desire 626s [20].

Tamaño	146,9 x 70,9 x 8,19 mm
Procesador	Qualcomm® Snapdragon™, de cuatro núcleos 4 a 1.1 GHz
Memoria	ROM: 8 GB / RAM: 1 GB ranura para tarjeta microSDXCTM, hasta 2 TB
Conectividad	Bluetooth® 4.1 Wi-Fi®: 802.11 b/g/n (2,4 GHz) Conector de sonido estéreo de 3,5 mm Puerto micro-USB 2.0 (5 patillas)
Sensores	Sensor de luz ambiental Sensor de proximidad Acelerómetro
Ubicación	GPS/AGPS + GLONASS
Cámara	Cámara principal: 8 MP, enfoque automático, sensor BSI, grabación de vídeo a 720p Cámara frontal: 2 MP, sensor BSI, grabación de vídeo a 720p
Batería	Capacidad: 2000 mAh

Tabla 7. Especificaciones técnicas del smartphone HTC Desire 626s [20].

Como ya se ha mencionó, una de las grandes ventajas de usar smartphones como OBC es que muchos de sus sensores ya vienen embebidos en su electrónica, reduciendo considerablemente el tiempo que tomaría el diseñar e integrar de manera individual cada uno de los sensores a una tarjeta PCB, dejando solo el trabajo en desarrollo de software.

6.1.1. DESARROLLO DE SOFTWARE EN SMARTPHONES

El desarrollo de software en esta clase de dispositivos dependerá del sistema operativo con el que este funciona. Cada compañía celular tiene un sistema operativo para cada uno de sus smartphones, por ejemplo los dispositivos de la compañía finlandesa Nokia venían con Symbian OS, los BlackBerry de la compañía canadiense BlackBerry Limited tienen BlackBerry OS, la compañía Apple tiene iOS, y así otros más.

Sin embargo, Android OS de la compañía Google es actualmente el sistema operativo más usado por múltiples compañías fabricantes de smartphones, tablets, pantallas inteligentes, etc.; con su uso en más del 60% de los dispositivos móviles a nivel mundial [19].

Android OS es basado en una versión modificada del kernel de Linux, tiene la enorme ventaja de ser open-source, es decir, que el usuario puedes ver y modificar el código fuente si así lo deseas para crear una propia versión de android. Al software desarrollado para smartphones se le denomina comúnmente como “software application” o simplemente “app”, que no es más que un programa de computadora que puede ser ejecutado únicamente en dispositivos móviles.

Estas apps son escritas usando el SDK (*Software Development Kit*) de Android, que Google provee en su respectivo sitio web, el SDK contiene múltiples herramientas como librerías para interactuar con el hardware del smartphone y puede ser usado en diferentes plataformas de programación, usualmente en Java como lenguaje de programación. Sin embargo, existen plataformas que permiten hacer uso de otros lenguajes como Visual Basic, C# o Python para crear aplicaciones, solo que, al momento de compilar el código, dichas plataformas tienen que convertir forzosamente el código a Java para que se pueda hacer uso de las herramientas del SDK.

A diferencia de versiones de Linux como Raspbian para la mini-PC Raspberry Pi, Android no tiene de manera nativa “X Window System (X11)” que es un sistema de despliegue de ventanas como en los sistemas operativos para PC de escritorio, ni tampoco tiene soporte para todo el set de librerías GNU estándar, haciendo difícil migrar software existente de Linux hacia Android.

6.2. SISTEMA DE POSICIONAMIENTO GLOBAL (GPS)

El GPS, originalmente llamado Navstar GPS, es un sistema de radionavegación basado en satélites, del cual el gobierno de los Estados Unidos es dueño y es operado por su fuerza aérea. Este sistema de navegación provee la geolocalización así como la información del tiempo a un usuario mediante un receptor GPS y funciona de manera global mientras esté en línea de visión directa con al menos 3 o más satélites (Figura 26). El GPS

no requiere que un usuario transmita algún tipo de dato y opera de manera independiente a cualquier tipo de sistema telefónico o de internet.

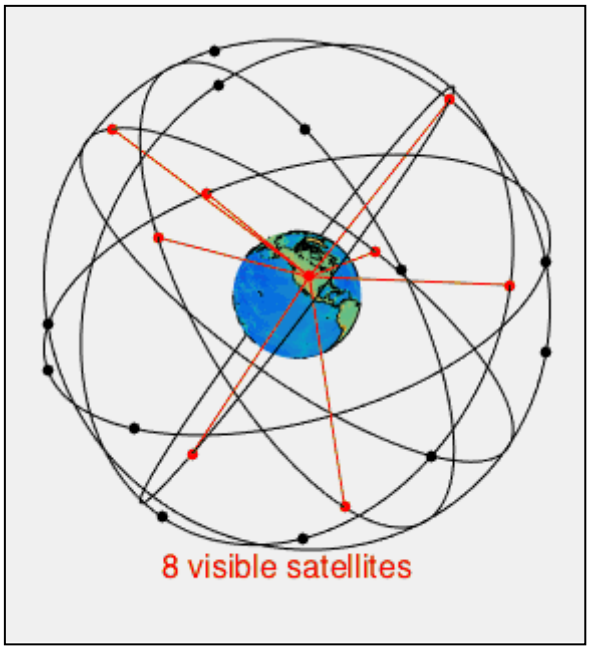

Figura 26. Ejemplo de una constelación de 24 satélites GPS. Las líneas rojas indican los satélites que están siendo usados para obtener la posición en un determinado punto de la Tierra.

En términos generales, un receptor GPS (Figura 27) está compuesto por una antena, un receptor de radio calibrado a la misma frecuencia a la que transmite los satélites, un procesador y un reloj altamente estable (generalmente un oscilador de cristal). Los smartphones, tablets y algunas computadoras tienen un módulo GPS ya embebido en su electrónica, sin embargo se pueden adquirir módulos GPS externos para integrarlos en diversos diseños de vehículos no tripulados, globos meteorológicos, etc.

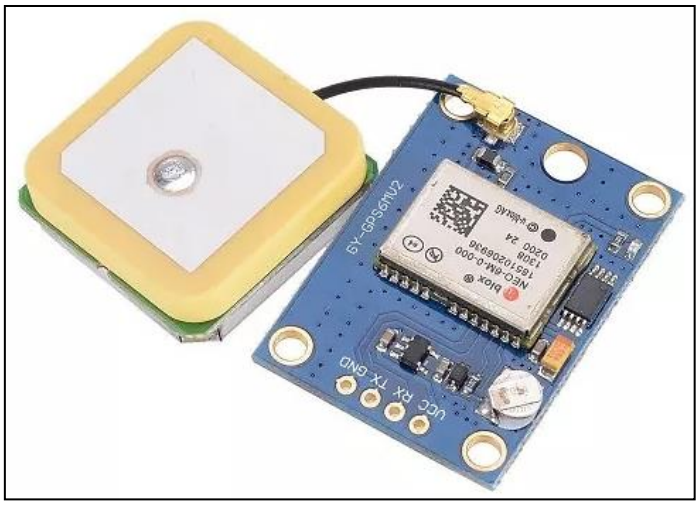

Figura 27. Ejemplo de un módulo GPS con antena

Los datos que otorgan los módulos GPS ya están procesados, de tal manera que estos nos entregan la información en coordenadas geográficas de posición horizontal (latitud y longitud), que es un sistema de coordenadas angulares esféricas cuyo centro es el centro de la Tierra y suelen expresarse en grados sexagesimales (Figura 28).

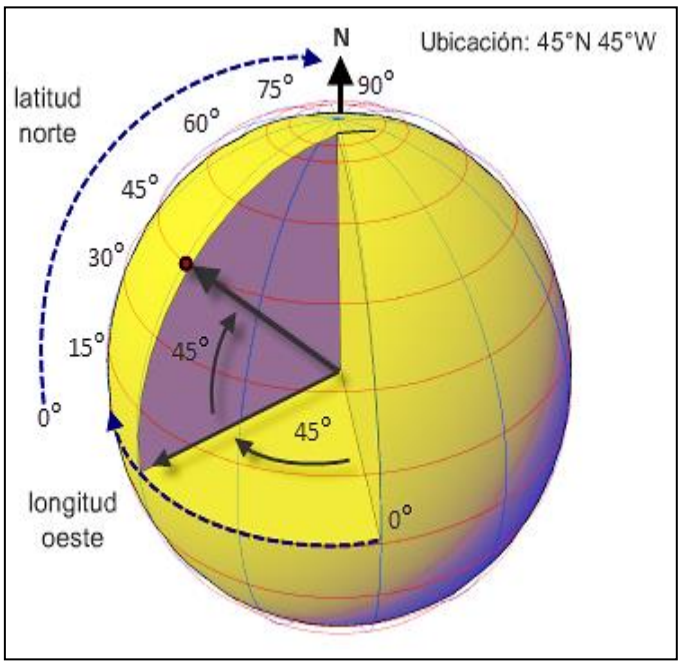

Figura 28. Representación gráfica de las coordenadas geográficas en una esfera.

6.3. SENSOR BRÚJULA EXTERNO

En algunos modelos de smartphones, el sensor brújula no viene agregado en el hardware, las aplicaciones que hacen uso de google maps sí requieren tener dicho sensor para orientar adecuadamente al usuario, pero en caso de no contar con él, usan el GPS y los cambios de posición y de aceleración para estimar la dirección en la que el usuario se está moviendo, claro que no puede distinguir si realmente está mirando en dicha dirección.

En el subsistema, si requerimos tener una brújula en todo momento ya que el control difuso depende de ese dato para determinar la referencia. Por lo que es necesario ingresar esa información al software del subsistema para que lo transmita a la computadora. Los dispositivos seleccionados para esta tarea son:

- Sensor brújula HMC5883L
- Microcontrolador Atmega328 en un Arduino Nano
- Módulo Bluetooth HC-05

El sensor HMC5883L (Figura 29) es un magnetómetro de 3 ejes digital (figura 29), producido por la compañía Honeywell, permite medir tanto la dirección como la magnitud del campo magnético de la Tierra, materiales magnéticos cercanos y campos magnéticos producidos por corrientes eléctricas. Funciona cuando un campo magnético exterior modifica ligeramente el modo en que la corriente pasa a través de sus sensores magneto-resistivos dentro del chip, permitiendo así el cálculo del vector de fuerza magnética. Felizmente la conversión analógica-digital y los cálculos del vector los realiza el mismo chip del sensor, únicamente nos entrega el resultado. El componente se comunica mediante el protocolo serial I2C, por lo que para proporcionarle la dirección al smartphone es necesario un microcontrolador para leer los datos del vector y convertirlos a grados de dirección con respecto al norte. Actualmente Arduino cuenta con una librería para convertir los datos del sensor HMC5883L a grados.

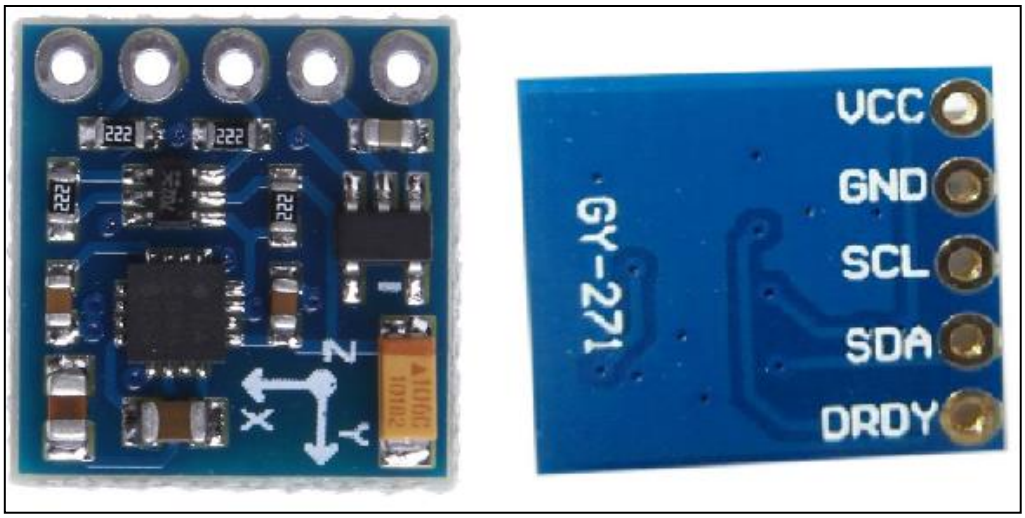

Figura 29. Sensor HMC5883L en un módulo GY-271

Consumo	3V-5V DC Operaciones de bajo voltaje (2.16 to 3.6V) y consume de baja corriente (100 µA)
Comunicación	Interfaz digital I2C
Velocidad I2C	Fast 160 Hz Maximum Output Rate
Rango de medición	± 1.3-8 Gauss
Dimensiones (Modelo GY-271)	14.8 x 13.5 x 3.5mm

Tabla 8. Características generales del sensor HMC5883L en el módulo GY-271

Nosotros tenemos la idea de que el campo magnético de la Tierra es generalmente estable en cualquier parte del globo, es decir que el valor de la declinación magnética es siempre el mismo sin importar en donde uno se encuentre. Sin embargo esta suposición es incorrecta, el campo magnético medido en cualquier punto de la superficie es una suma de varios campos magnéticos generados de manera interna en el núcleo exterior de la Tierra. Este campo principal varía de intensidad lentamente con el tiempo y también su influencia local varía dependiendo del lugar, es tan irregular que debe de ser medido en diferentes lugares para obtener una distribución correcta y esto se realiza usando satélites, aproximadamente 200 realizan operaciones de observación a nivel mundial [21].

Para calibrar correctamente el ángulo de dirección en los vehículos no tripulados, se debe tomar en cuenta el lugar en donde estarán operando, algunos traen una base de datos en donde, dependiendo de su posición, hacen

sus calibraciones. En nuestro diseño, solo se incluye la declinación magnética de la ciudad de Puebla en el programa de Arduino para hacer la corrección de ángulo. La página web de la NOAA (National Oceanic and Atmospheric Administration) de los Estados Unidos provee una calculadora del ángulo de declinación, solo es requerido ingresar la latitud y longitud del lugar.

Se escoge la latitud y longitud de la facultad de electrónica de la BUAP (Figura 30), es el punto de inicio de la simulación por default y el que ingresaremos a la calculadora:

Figura 30. Latitud y longitud de uno de los edificios de la facultad de electrónica de la Benemérita Universidad Autónoma de Puebla.

El resultado de la declinación magnética es de:

4.27° ESTE ± 0.31°

Es verdad que el ángulo es realmente pequeño comparado con otras declinaciones en otros lugares, dependiendo de la aplicación, uno puede despreciarlo si así se prefiere. En el caso de este trabajo, si se incluirá en el programa de Arduino. Una vez con la dirección en grados de -180 a 180, estos se enviarán vía bluetooth a la OBC, que es el smartphone, donde nuevamente se enviará junto con la información del GPS a la computadora del usuario.

El módulo bluetooth HC-05 (Figura 31) es un dispositivo electrónico que permite realizar una interfaz inalámbrica transparente entre un

microcontrolador, FPGA, DSP, etc. y una computadora que también tenga bluetooth. Este modelo solo permite ser configurado como *Esclavo*. Usa protocolo serial, por lo que es relativamente fácil de usar (Figura 32).

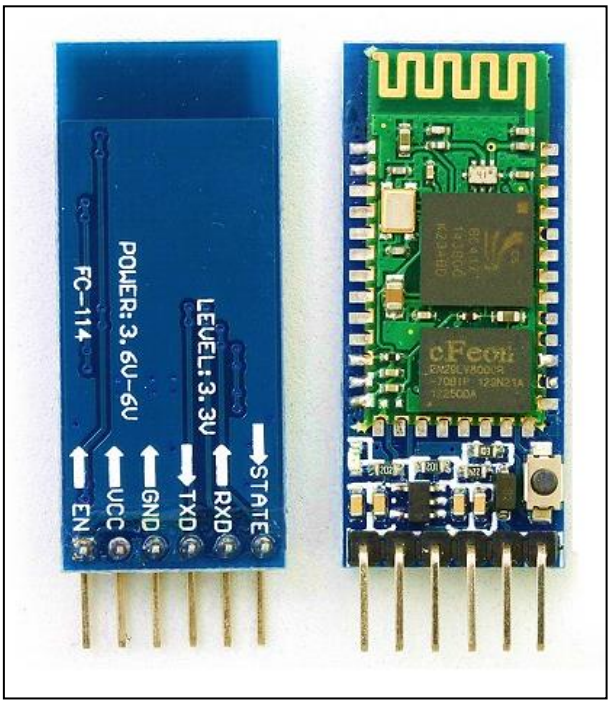

Figura 31. Módulo HC-05

Consumo	3.3V a 5V
Sensibilidad	-80 dBm
Potencia de transmisión RF	Hasta +4dBm
Comunicación	UART
Antena	Integrada en PCB
Baudrates permitidos	9600, 19200, 38400, 57600, 115200, 230400, 460800

Tabla 9. Características generales del módulo bluetooth HC-05

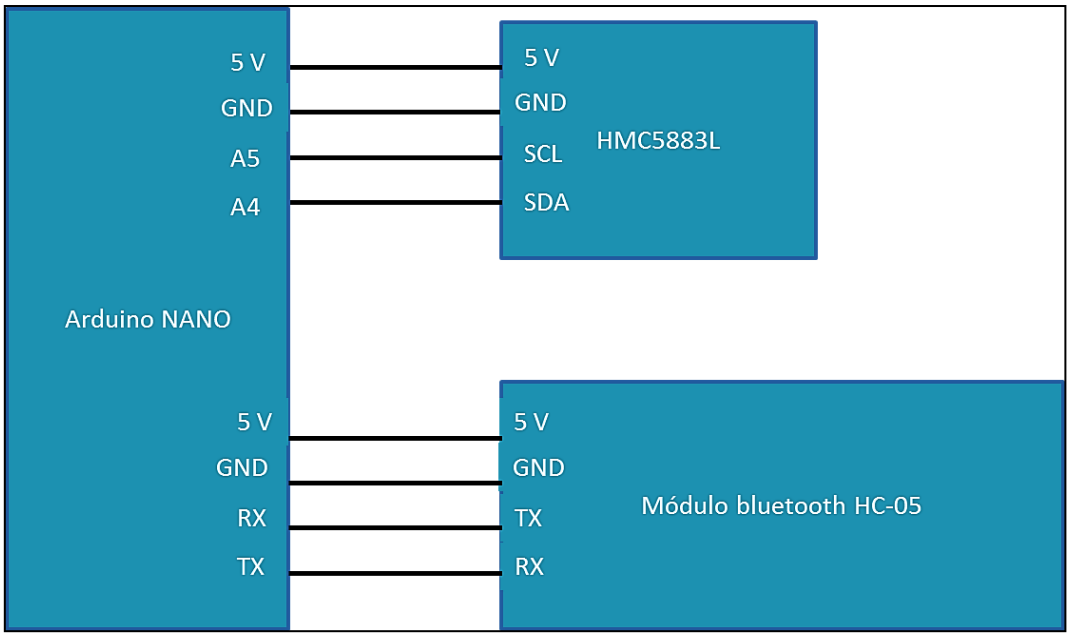

Figura 32. Diagrama de bloques de la brújula digital bluetooth

A continuación en la figura 33 se muestra un diagrama de flujo del programa de arduino y en la figura 34 el diagrama de flujo del programa de adquisición de datos en el smartphone.

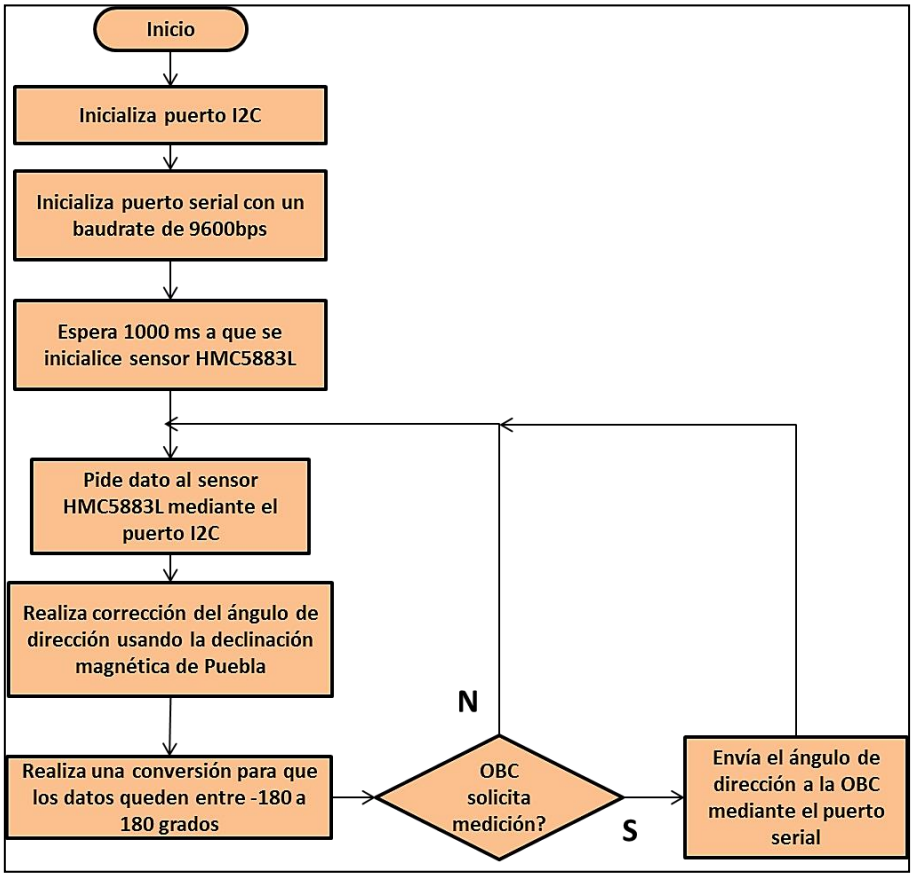

Figura 33. Diagrama de flujo del programa para el sensor brújula bluetooth

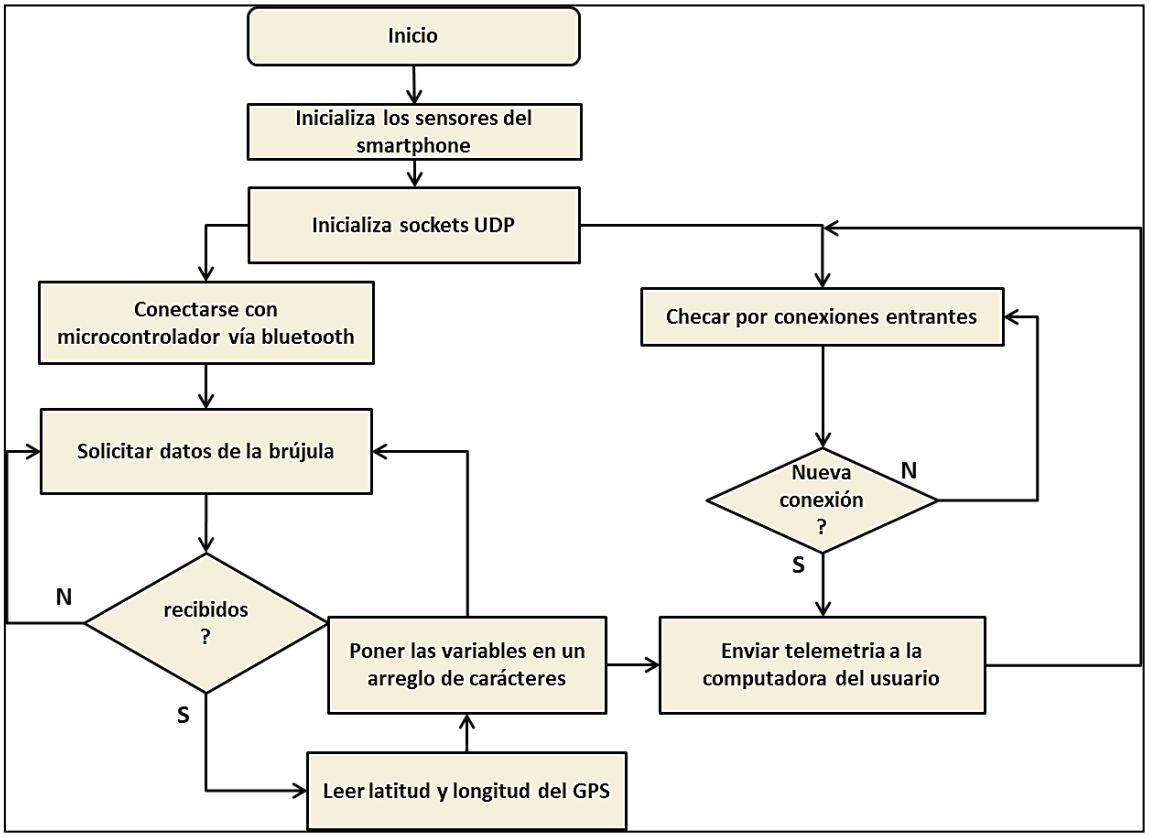

Figura 34. Diagrama de flujo del programa de adquisición de datos en el smartphone

6.4. INTERFAZ GRÁFICA

La interfaz gráfica del software de simulación se desarrolla en el lenguaje de programación Visual Basic incluido en la plataforma Visual Studio 2010 versión Express. Se escogió este lenguaje de programación porque al tratarse de un lenguaje que pertenece a la plataforma .Net, está en el top 10 de los más usados a nivel profesional [13] [14]. El servicio de web mapping usado para obtener los mapas, viene de Google Maps ya que permite el uso de sus servicios de manera gratuita, En la figura 35, los checkpoints de la trayectoria son los puntos naranja, la trayectoria a seguir también es de color naranja. El polígono verde tipo flecha es la representación del vehículo, la punta indica la dirección. En la figura 36, el panel de la izquierda muestra los controles en donde se inserta la latitud, longitud y la dirección de inicio del polígono, ya que al tratarse de una simulación, el usuario debe ingresar dichos datos iniciales. A la derecha de la figura 36, se muestra el panel con indicadores de la posición del vehículo, su distancia y la dirección a la que se encuentra del Checkpoint al

que se dirige. También se muestra el valor de la referencia (Ref), que es la entrada al control y a su vez su salida (Output). En la tabla 10 se muestran los datos de la dirección, el valor de entrada al control y su valor de salida obtenidos de la simulación del seguimiento de la trayectoria mostrada en la figura 35.

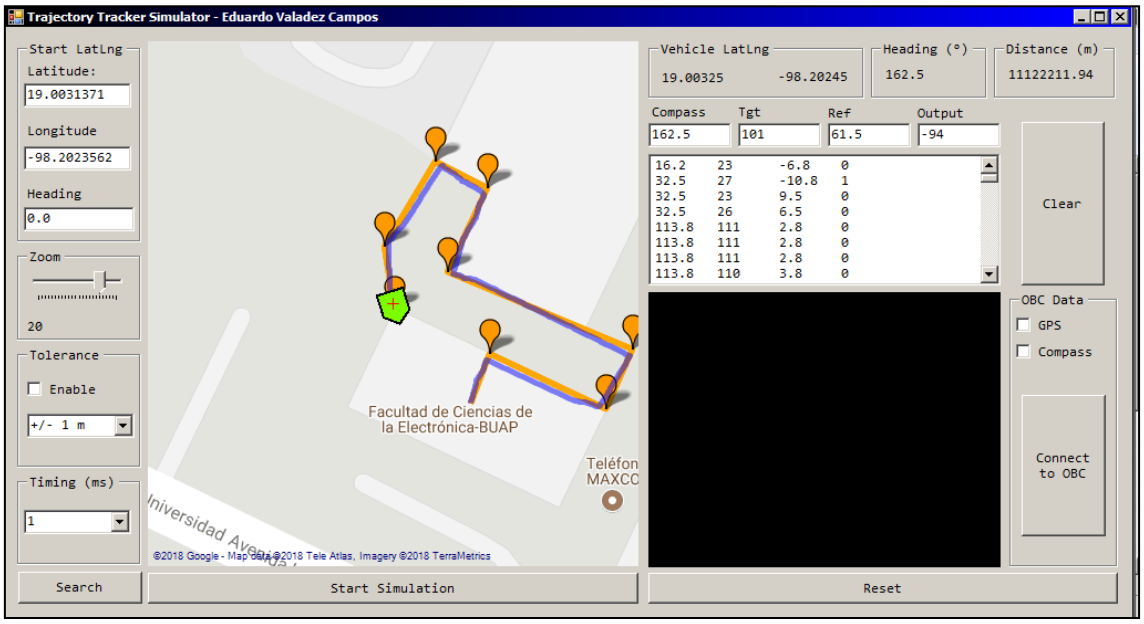

Figura 35. Captura de pantalla del software de simulación en funcionamiento.

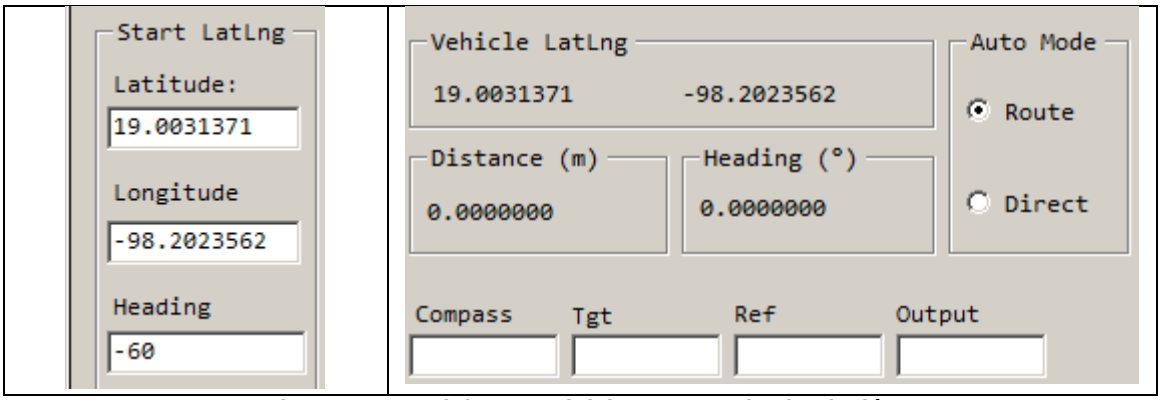

Figura 36. Panel de control del programa de simulación.

Dirección vehículo	Dirección Checkpoint	Referencia	Salida
16.2	23	-6.8	0
32.5	27	-10.8	1
32.5	23	9.5	0
32.5	26	6.5	0
113.8	111	2.8	0
113.8	111	2.8	0
113.8	111	2.8	0
113.8	110	3.8	0
113.8	110	3.8	0
113.8	109	4.8	0
113.8	108	5.8	0
113.8	106	7.8	0
97.5	103	-5.5	0
97.5	106	-8.5	0
113.8	104	9.8	-1
32.5	31	17.8	-1
32.5	31	1.5	0
32.5	31	1.5	0
32.5	30	2.5	0
32.5	29	3.5	0
32.5	25	7.5	0
-32.5	-66	49.8	-60
-65	-66	1	0
-65	-67	2	0
-65	-67	2	0
-65	-67	2	0
-65	-67	2	0
-65	-67	2	0

-65	-67	2	0
-65	-67	2	0
-65	-68	3	0
-65	-68	3	0
-65	-68	3	0
-65	-69	4	0
-65	-69	4	0
-65	-70	5	0
-65	-72	7	0
-65	-74	9	0
-81.2	-72	-9.2	0
-81.2	-75	-6.2	0
-16.2	23	-55.5	82
16.2	23	-6.8	0
16.2	24	-7.8	0
16.2	25	-8.8	0
32.5	26	6.5	0
32.5	24	8.5	0
16.2	22	-5.8	0
16.2	25	-8.8	0
32.5	25	7.5	0
-48.8	-57	24.5	-3
-65	-59	-6	0
-65	-57	-8	0
-48.8	-56	7.2	0
-65	-58	-7	0
-48.8	-56	7.2	0
-113.8	-143	45.5	-41
-146.2	-143	-3.2	0
-146.2	-143	-3.2	0

-146.2	-142	-4.2	0
-146.2	-142	-4.2	0
-146.2	-141	-5.2	0
-146.2	-140	-6.2	0
-146.2	-137	-9.2	0
-130	-138	8	0
-146.2	-138	-8.2	0
-113.8	175	-305	41
0	175	-191.2	100
113.8	175	-77.5	100
178.8	175	3.8	0
178.8	174	4.8	0
178.8	173	5.8	0
178.8	172	6.8	0
178.8	170	8.8	0
162.5	169	-6.5	0
178.8	171	7.8	0
162.5	101	61.5	-94
162.5	101	61.5	-94
162.5	101	61.5	-94

Tabla 10. Datos de la dirección del vehículo.

CONCLUSIONES

7. TRABAJO ACTUAL

En este trabajo se propone un modo de implementar de manera autónoma el seguimiento de trayectorias mediante checkpoints en una ruta sobre un mapa, el cual usa un sistema de control difuso para el control de la dirección de un UGV. Esta propuesta es probada mediante el diseño de un software de simulación que nos permita visualizar de manera gráfica el comportamiento aproximado que tendría un vehículo terrestre al usar este algoritmo. La aportación en esta clase de programas es la posibilidad de probar diferentes ajustes de tolerancias de GPS y cómo esto afectaría la precisión del algoritmo, también poder probar la eficiencia del control de dirección usando teoría de conjuntos difusos, teoría de control difuso y técnicas de programación.

El adoptar el control de lógica difusa en vez de un controlador clásico (basado en modelos matemáticos) se basa en el hecho que el primero presenta cierta tolerancia a la imprecisión y a la habilidad de poder tomar decisiones bajo ciertas incertidumbres. Además puede abordar problemas de algunos sistemas que no pueden ser aproximados de manera sencilla a un modelo matemático estándar o lineal, de manera que puedan resolverse usando un razonamiento similar al de un ser humano.

No puede decirse que un tipo de control sustituya al otro; depende del sistema con el que se trabaje, si se cuenta con el modelo matemático (incluso si es no-lineal) del proceso a controlar, se puede optar por un control clásico. Sin embargo, en la naturaleza la mayoría de los comportamientos son no-lineales, por lo que no se cuenta con un modelo matemático o es igualmente difícil de obtener. En esta clase de casos, el control difuso es una alternativa viable.

En cuanto al uso de servicios de web mapping, la tendencia del mundo actual va en dirección de los servicios web, desde sistemas de comunicación hasta el

modo en que nos entretenemos; basta con decir que los servicios de transporte privados (Uber, Cabify, etc.) se han vuelto muy populares debido a la personalización del servicio que estas empresas brindan a los usuarios. Por lo que el usar herramientas online en diversas aplicaciones puede ayudar a mejorar la experiencia del usuario y optimizar algunas tareas.

Por último el uso de smartphones Android como plataformas de telemetría y computadoras de abordo en vehículos no tripulados ya es una realidad; no solamente por las versatilidades que su sistema operativo basado en Linux puede brindar a la hora de desarrollar aplicaciones o los múltiples sensores y componentes electrónicos que ya vienen embebidos en la electrónica de cada dispositivo, sino por la constante mejora. Las grandes compañías que fabrican smartphones inviertes millones de dólares en la constante optimización de sus productos, haciéndolos más eficientes y resistentes a las condiciones de uso; convirtiendo a los smartphones en mini-PCs robustas.

8. TRABAJO FUTURO

Las futuras aplicaciones para este trabajo son la implementación del algoritmo en un vehículo no tripulado real, inicialmente probarlo con un vehículo terrestre, para después realizar los cambios necesarios para implementarse en un UAV (Unmmaned Aerial Vehicle), mejor conocido como un drone.

En el caso de la implementación en un vehículo terrestre, el diseño del mismo, el terreno donde va a operar y los obstáculos como paredes, zanjas, etc. ya forman parte de los aspectos a considerar para ajustar adecuadamente el algoritmo.

En el caso del drone, es preciso realizar mucho más ajustes, ya que ahora se trata de un movimiento en 3-D, incluyendo a la altura como variable de posición. La inclinación, la masa de las baterías, la resistencia del viento, etc., son algunos de los factores a considerar si se piensa adaptar el algoritmo para su uso en esta clase de máquinas.

9. REFERENCIAS

[1] Yamakawa T., (1988). High-speed fuzzy controller hardware system: The mega-FIPS machine. Inf. Sci. Vol. 45, No.2, pp. 113-128.

[2] Zadeh L.A., (1985). Fuzzy Sets, Information and Control, Vol. 8, No. 3 pp. 338-353, june 1985.

[3] Nguyen Hung T. et al (2003) A first course in fuzzy and neural control. Chapman & Hall/CRC. Estados Unidos de América. p 249.

[4] Albertos P., y Sala A., (2004). El control borroso: una metodología integradora. RI-AII, Vol. 1, No. 2, pp. 22-31.

[5] Guillermo Morales Luna, (1998). Informe técnico. Lógica difusa. CINVESTAV. México, D.F.

[6] Klir, Yuan. (1995). Fuzzy sets and fuzzy logic. Theory and applications. Prentice Hall PTR

[7] Martín del Brío B., y Sanz Molina A., (2002). Redes neuronales y sistemas difusos. 2da. Edición, Alfa Omega Grupo Editor. México.

[8] Britannicacom. 2017. *Encyclopedia Britannica.* (Online). (26 Diciembre 2017). Disponible en: https://www.britannica.com/technology/computer-simulation

[9] Mathworkscom. 2017. Mathworkscom. (Online). (28 Diciembre 2017). Disponible en: https://www.mathworks.com/matlabcentral/fileexchange/57132-satellite-orbit-transfer-simulation

[10] Kraak, Menno Jan (2001): *Settings and needs for web cartography*, in: Kraak and Allan Brown (eds), Web Cartography, Francis and Taylor, New York, p. 3–4.

[11] Peng, Z.-R. & M-H. Tsou (2003), *Internet GIS - Distributed Geographic Information Services for the Internet and Wireless Networks*, John Wiley & Sons, New Jersey.

[12] C.C. Liebe (1993), *Pattern Recognition of Star Constellation for Spacecraft Applications*, lEEE Aerospace & Electronics Sys. Mag., pp. 31-39.

[13] The 2017 Top Programming Languages. (Online). (3 de marzo 2018). Disponible en: https://spectrum.ieee.org/computing/software/the-2017-top-programming-languages

[14] TIOBE Index for February 2018. (Online). (3 de marzo 2018). Disponible en: https://www.tiobe.com/tiobe-index/

[15] Passino, K.M & Yurkovich, S (1998). *Fuzzy Control*. California: Addison Wesley Longman, Inc. pp. 30

[16] Nasagov. 2018. Nasagov. [Online]. [13 March 2018]. Available from: https://www.nasa.gov/pdf/598887main_Auburn_PowerPoints_SE.pdf

[17] Cai, L, Machiraju, S & Chen, H. 2009. *Defending against sensor-sniffing attacks on mobile phones*. Proceedings of the 1st ACM workshop on Networking, systems, and applications for mobile handhelds. pp. 31-36.

[18] Reininger M, Miller S, Zhuang Y and Cappos J, *A first look at vehicle data collection via smartphone sensors*, 2015 IEEE Sensors Applications Symposium (SAS), Zadar, 2015, pp. 1-6.

[19] Mybroadbandcoza. 2018. Mybroadbandcoza. [Online]. [19 March 2018]. Available from: https://mybroadband.co.za/news/software/232485-the-most-popular-operating-systems-for-smartphones-and-pcs.html

[20] Htccom. 2018. *HTC*. [Online]. [19 March 2018]. Available from: http://www.htc.com/latam/smartphones/htc-desire-626s/

[21] National centers for environmental information (ncei). 2018. *Noaagov.* [Online]. [19 March 2018]. Available from: https://www.ngdc.noaa.gov/geomag/faggeom.shtml

[22] González Jiménez, Javier y Ollero Baturone, Aníbal, “Estimación de la posición de un robot móvil”, Informática y Automática, vol.29-4, España, 1996.

[23] Lozano-Pérez, Tomás “Foreword: Mobile Robot and Robotics”. Autonomous Robot Vehicles. Editores I.J.Cox y G.T. Wilfong. Springer-Verlag. pp vii-xi, 1990.

[24] Arkin, Ronald “Behavior-Based Robotics”. The MIT Press. pp 1-29, 1998.

[25] Campion, G., Bastin, G and D’Andréa-Novel. “Structural properties and classifications of kinematics and dynamics models of wheeled mobile

robots”. IEEE Transactions on Robotics and Automation. Vol. 12, No.1. February 1996.

[26] Jones, J.L. and Flynn, A.M. “Mobile Robots. Inspiration to Implementation”. Editorial. A.K. Peters, Ltd. USA. 1993. Ch. 6, pp.139-161

[27] Adamowski, J.C., Simoes, M.G. y Gozman, F.G., "Desenvolvimento de un robo móvel", Escola Politecnica de la Universidade de Sao Paulo, San Pablo, Brasil, 1990.

[28] Cox, I.J., "Blanche - An experiment in guidance and navegation of an autonomous robot vehicle", IEEE, Trans RA, vol. 7, número 2, 1991.

[29] Connell, J. y Viola, P., "Cooperative control of a semi-autonomous mobile robot", IBM T.J. Watson Research Center, 1990.

[30] Leopoldo Calderón Estévez, Ramón Ceres Ruiz, José Nó Sánchez de León y José Ramón Alique López. “Sensores de distancia en robótica” Revista Robótica. No.12. Marzo-Abril. 1985. España.

[31] World-nuclearorg. 2018. *World-nuclearorg.* [Online]. [25 March 2018]. Available from: http://www.world-nuclear.org/information-library/safety-and-security/safety-of-plants/chernobyl-accident.aspx