BENEMÉRITA UNIVERSIDAD AUTÓNOMA DE PUEBLA

FACULTAD DE CIENCIAS DE LA ELECTRÓNICA

DESARROLLO E IMPLEMENTACIÓN DE SOFTWARE PARA MANTENER HORIZONTAL UNA PLATAFORMA ANTE MOVIMIENTOS EXTERNOS BASADO EN LÓGICA DIFUSA

Tesis que presenta

Alan Osorio Orduña

Para obtener el grado de

Licenciado en Ingeniería Mecatrónica

Director de la Tesis: **M.C. Gustavo Mendoza Torres**

Puebla, Pue. **Febrero 2014**

AGRADECIMIENTOS

Quiero expresar mi sincero agradecimiento al M.C. Gustavo Mendoza Torres y a la Dra. Leticia Gómez Esparza, por haberme brindado su apoyo y su conocimiento. Gracias por sus correcciones, sus consejos y su confianza.

A mis padres Anselmo Osorio Mirón y Guadalupe Orduña Huerta, por ser las personas que siempre han estado; a mi padre por mostrarme la pauta de lo que se puede y se debe hacer en el mundo del entendimiento y la conciencia; a mi madre que me enseñó que dentro de las dificultades hay también esfuerzo y satisfacción.

A Odeth Osorio Orduña por abrirme un horizonte que no conocía hasta que ella me lo mostró, la literatura también es una conquista.

A José Luis Espinoza Molina y Marco Antonio Tierra Axalco por su amistad y su complicidad dentro y fuera de la universidad.

A Misael Mendoza Tlalolini (Dios me libre!) por hacer más amenas las clases, los proyectos y las prácticas en la facultad.

A Miguel Ángel Ortega porque sin su ayuda no sería posible este trabajo.

Por su puesto a Blanca Areli Martínez Treviño, una persona muy singular, paciente e inteligente que me concede su amistad. *Tu sais bien ça.*

Gracias además a todas las personas con las que conviví en estos años, amigos y familiares, por los buenos y por los muy buenos momentos que compartí a su lado.

Finalmente espero que, a pesar de los detalles que se puedan encontrar en estas notas, para algo hayan de ser útiles a los estudiantes de la F.C.E.

RESUMEN

El presente trabajo de tesis tiene como objetivo la creación de un programa basado en lógica difusa y una interfaz gráfica de usuario para la estabilización de una plataforma en posición vertical, con tres grados de libertad, ante movimientos externos. El control difuso permite controlar sistemas no lineales con relativa facilidad, gracias a la sencillez de sus cálculos, (Martín, 2002).

El programa estará compilado en un microcontrolador, éste a su vez se encontrará montado en una placa de desarrollo, que estará conectada directamente con los actuadores de la plataforma. Para el diseño de la interfaz gráfica de usuario se eligió el ambiente de programación *Processing*, para proporcionar un entorno visual sencillo del sistema.

La motivación del proyecto se basa en la importancia que tiene su aplicación en diversas ramas de la ingeniería tanto civil como militar, la robótica móvil, y los sistemas de posicionamiento de cámaras y sensores, son algunos ejemplos.

Contenido

Índice de figuras

Índice de tablas

INTRODUCCIÓN

Actualmente, el grado de desarrollo científico y tecnológico en diversas instituciones educativas propicia la formación de recursos humanos especializados en diversas áreas técnico-científicas con un potencial de innovación que permite adquirir, adaptar y desarrollar tecnología en el menor tiempo posible. En este contexto, el desarrollo de tecnología de bajo costo para aplicarse en diversos campos técnicos, industriales, educativos, sociales o ambientales, entre otros, es uno de los fines principales dentro de las instituciones de educación superior. Este trabajo se inserta en el campo de desarrollo de tecnología, con diversas aplicaciones tales como puede verse en la literatura, desde control industrial hasta medicina, pasando por áreas como computación, sistemas expertos, recuperabilidad en edificios y viviendas, confiabilidad, etc. La realización este proyecto nos permitirá generar conocimiento para el desarrollo de un algoritmo de control aplicando diversas técnicas de diseño basadas en lógica y teoría de conjuntos difusos.

Los controladores basados en lógica difusa se han empleado con bastante éxito en la industria, principalmente en Japón. Entre las principales aplicaciones se tienen; sistemas de control de sistemas de aire acondicionado, sistemas de iluminación automático en cámaras fotográficas, optimización de sistemas de control industrial, sistemas de reconocimiento de escritura, mejora en la eficiencia del uso de combustible en motores, sistemas expertos del conocimiento (simular el comportamiento de un experto humano), bases de datos difusas, almacenar y consultar información imprecisa, para este punto, por ejemplo, existe el lenguaje FSQL; modelado con red neuronal y lógica difusa de un sistema experto que permite decidir a personas inexpertas sobre la recuperabilidad de los edificios y viviendas luego de un fuerte sismo, la agencia del espacio de la NASA se dedica a aplicar la lógica difusa para maniobras complejas. En medicina, los campos médicos que han sido estudiados desde el punto de vista de la lógica difusa han sido clasificados según Mahfour y Col, (2001), en cuatro categorías: 1) disciplinas conservadoras, 2) medicina invasiva, 3) disciplinas médicas definidas regionalmente, 4) Procesado de imágenes y señales. (D'Negri y De Vito, 2006). En este sentido, el empleo de control difuso es recomendable en procesos muy complejos, cuando es complicado obtener un modelo matemático o si el procesamiento basado en el conocimiento de un experto (en términos lingüísticos) puede ser aplicado. Por otro lado, el empleo del control difuso no es una buena idea si el control convencional teóricamente rinde un resultado satisfactorio y si existe un modelo matemático fácilmente soluble y adecuado, (Pratihar, 2008; Du y Swamy, 2006; Albertos y Sala, 2004; Martín y Sanz, 2002).

Se tiene una plataforma mecánica de tres grados de libertad a escala de banco ("workbench") y una placa de desarrollo "Arduino Uno" con un microcontrolador ATmega328. Se busca desarrollar un software con una interfaz gráfica, para el control del sistema en lazo cerrado, plataforma-controlador difuso. La interfaz permitirá la

visualización de las señales del acelerómetro, parámetros y/o variables del sistema. Las señales del acelerómetro (aceleraciones de ejes coordenados) se emplearan en una fórmula trigonométrica, para obtener los grados de inclinación de la plataforma.

En el capítulo 1, se presentan los fundamentos de la lógica difusa y los conjuntos difusos, para construir un control difuso; así también se muestra el hardware que se emplea para la programación de de dicho control y sus dispositivos.

En el capítulo 2, se describen los conceptos de fusificación, defusificación, base de reglas e inferencia difusa, todo esto para mostrar el desarrollo y aplicación teórica de un control difuso.

En el capítulo 3, se presenta la plataforma sobre la cual se implementa el control difuso programado en la placa Arduino, tomando los datos del acelerómetro y dando como resultado el movimiento de los servomotores, para obtener la posición horizontal de dicha plataforma. También se describe la interfaz gráfica que muestra el comportamiento del acelerómetro como sensor de inclinación.

Por último se muestran los resultados y conclusiones del trabajo.

Como apéndices A, B y C se encuentra el código para el empleo del acelerómetro, él código para el empleo de los servomotores y el código de la interfaz gráfica.

OBJETIVO GENERAL

Diseñar un algoritmo de control basado en lógica difusa que permita mantener en posición horizontal la parte superior de una plataforma de dos grados de libertad, ante movimientos externos reflejados en la base de ésta. El control será tipo lazo externo (control difuso), lazo interno (control del servo).

OBJETIVOS ESPECÍFICOS

1. Caracterizar dispositivos, tales como acelerómetro y motores.

2. Hacer un programa empleando la lógica difusa y la teoría de conjuntos difusos, para finalmente obtener el control difuso.

3. Construir una interfaz gráfica bajo el entorno de desarrollo Processing.

4. Implementar el programa con la interfaz gráfica y el control difuso en la plataforma.

JUSTIFICACIÓN

La motivación del estudio del problema de control de una plataforma mecánica, se basa en la importancia que tiene para su aplicación en robótica móvil y en aplicaciones en donde se desea incorporar antenas, cámaras o sensores externos del tipo sonar, radar, telémetro, infrarrojos y similares donde la línea de mira del sensor debe mantenerse estable frente a los movimientos del vehículo donde se asienta. En sistemas de navegación puede aplicarse en equipos donde se debe mantener contacto visual estable con elementos externos al buque, tal como GPS, antenas parabólicas para comunicaciones vía satélite, detección de obstáculos, faros, puntos de referencia en la costa, etc. En aplicaciones militares, se puede utilizar en sistemas de defensa antiaérea. En estas aplicaciones la medición del sensor se puede degradar por el movimiento aleatorio de la superficie sobre la que se asienta. Para compensar esto, se deben instalar actuadores lentos o rápidos, utilizando la retroalimentación de la variable medida (por ejemplo, la velocidad angular) para el diseño de controladores con mayor refinamiento que mantengan márgenes de fase aceptables con el fin de evitar inestabilidades y oscilaciones (Gómez-Stern, 2002).

Así, este trabajo se enfoca en el diseño, programación y aplicación de un control difuso, la implementación en una plataforma mecánica, es un compromiso extra, para verificar el software diseñado. Se dispone del equipo a escala "workbench" y una plataforma de hardware conocida como placa "Arduino Uno" con un microcontrolador ATmega328. El desarrollo del software con interfaz amigable al usuario, permitirá evaluar el desempeño del sistema en lazo cerrado, plataforma-controlador difusa, ante perturbaciones tales como el ruido en las señales que se coloque sobre la plataforma.

El programa que se pretende diseñar y construir, contribuirá al desarrollo científico e institucional, que será capaz de implementar nuevos algoritmos de control para caracterizar un mejor desempeño en el ámbito de sistemas de estabilización.

El software y el conocimiento que se generarán al realizar este proyecto serán las bases que conformarán algún proyecto de una magnitud y relevancia mayor en un futuro. También se pretende alcanzar el desarrollo de tecnología mexicana, consiguiendo así la elaboración de nuevos sistemas control para su estudio en instituciones educativas y científicas.

Capítulo 1

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 (Zadeh L., Fuzzy Sets, 1965). 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 (Yamakawa, Miki 1986). 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 (Nguyen et al 2003). 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 (Sangali 2013).

Parte del interés de este trabajo es introducir al lector en la teoría de la lógica difusa.

1. Teoría de los Conjuntos Difusos

1.1. Conceptos básicos de los conjuntos difusos

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

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

$$X_A = \begin{cases} 1\ si\ x \in A \\ 0\ si\ x \in A. \end{cases} \tag{1}$$

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 membrecía. Dicha función recibe el nombre de función de membrecía y el conjunto que se obtiene al evaluar la función de membrecía 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.

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

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

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

donde *A*(*x*) es llamado grado de membrecía 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.

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.

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

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

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

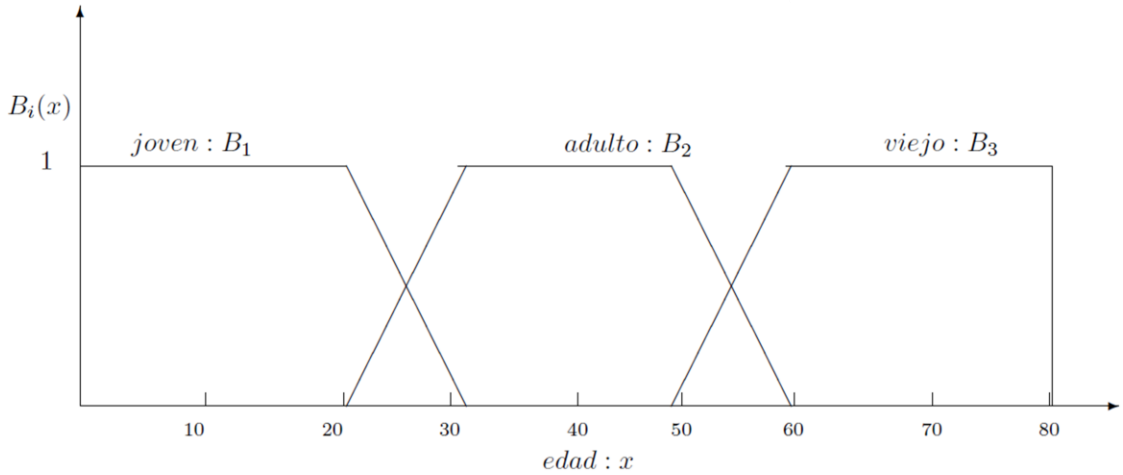


Figura 1. Funciones de membrecía que representan los conceptos de joven adulto y viejo.

1.2 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:

Complemento de A	$\bar{A}(x) = 1 - A(x)$	(6)
Intersección de A y B	$(A \cap B)(x) = \min\,[A(x), B(x)]$	(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.

1.3 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 tiene 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 $v_1, v_2, \dots, v_n$ (para n finito) puede ser formada por un número finito de estas primitivas.

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 formula 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:

$(a \wedge (a \longrightarrow b)) \longrightarrow b$	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.

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

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

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

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

1.4 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 lógicas n −valuadas, para $n \geq 2$; fue propuesta por Lukasiewicz en los años 30's, como una generalización de su lógica *tri*-valuada. Utilizó los valores de verdad dados en T_ny 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 estan dados por los valores del conjunto T_n. La sucesión $(L_1, L_2, \dots, L_\infty)$ 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.

Establecemos 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. 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.

1.5 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 verán 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.

1.6 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

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

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

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

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

Este procedimiento es llamado generalización de *modus ponens*.

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'

1.7 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.

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* presentado en el capítulo 1.6. Siguiendo la misma línea de ese capítulo 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.

1.8 Componentes Electrónicos y Lenguajes de Programación

1.8.1 Arduino UNO

El Arduino UNO (figura 2) es una placa de hardware con microcontrolador basada en el ATmega328 de 8-bits. Cuenta con un entorno de desarrollo propio, y al igual que su hardware su software también es libre.

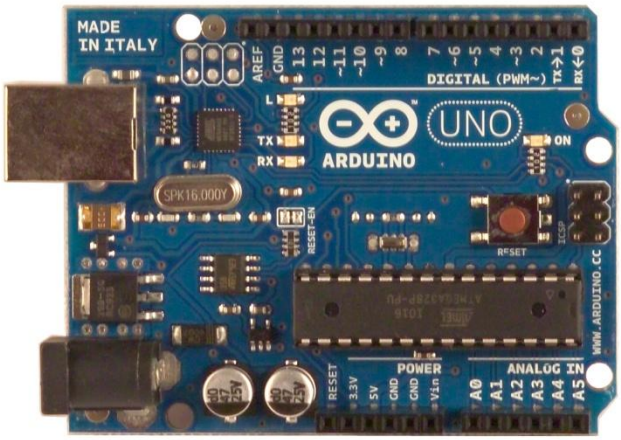


Figura 2. Placa de desarrollo Arduino UNO.

El Arduino UNO puede ser alimentado vía USB o con una fuente de alimentación externa, como pueden ser un transformador o una batería. La placa puede trabajar con una alimentación de entre 6 y 20 volts, aun que el rango recomendado es de 7 a 12 volts.

En el aspecto de la comunicación con la PC el Arduino UNO la facilita en varios aspectos pues el microcontrolador ATmega328 cuenta con comunicación vía serie UAR TTL y además soporta la comunicación I^2C y SPI.

Respecto a la programación como se mencionó anteriormente el Arduino UNO tiene su propio lenguaje ("Arduino"), sin embargo es posible usar una gran variedad de lenguajes y aplicaciones para programarlo, como Java, Flash, C, C++, Matlab o Processing. Siendo éste último el lenguaje de programación usado para crear la interfaz gráfica de este proyecto.

A continuación se presentan a manera de resumen algunas de las características de las características de la placa Arduino UNO y del microcontrolador ATmega328:

• Microcontrolador	ATmega328
• Voltaje de funcionamiento	5V
• Voltaje de entrada (recomendado)	7-12V
• Voltaje de entrada (limite)	6-20V
• Pines E/S digitales	14 (6 proporcionan salida PWM)
• Pines de entrada analógica	6
• Intensidad por pin	40 mA
• Intensidad en pin 3.3V	50 mA
• Memoria Flash	32 KB (ATmega328) de los cuales 2 KB las usa el gestor de arranque(bootloader)
• SRAM	2 KB (ATmega328)
• EEPROM	1 KB (ATmega328)
• Velocidad de reloj	16 MHz

1.8.2 Acelerómetro

El ADXL345 (figura 3) es un acelerómetro de 3 ejes pequeño y de alta resolución, que permite hacer mediciones de hasta ±16g.

Figura 3. Acelerómetro ADXL345

La razón principal de usar un acelerómetro en este proyecto y no un giroscopio de la misma gama, es porque los giroscopios de cualquier clase presentan un pequeño error (deriva) acumulable con el tiempo. En efecto, a medida que transcurre el tiempo, la lectura del giroscopio tiene cierta variación que provoca que el ángulo calculado difiera del ángulo real.

Cuenta con salida de datos digital con un formato de 16 bits a través del bus Interfaz de Periféricos Serie (SPI por sus siglas en ingles) o del bus de comunicaciones en serie Inter-Circuitos Integrados (referido como I^2C).

A continuación se presentan a manera de resumen algunas de las características del acelerómetro ADXL345:

- Voltaje de alimentación 2.0-3.6v.
- Consumo de ultra baja intensidad de corriente eléctrica: 40uA en medición y 0.1uA en *stand by* en 2.5v.
- Detección de caída libre.
- Interfaces SPI e I^2C.

Existen en el mercado una amplia gama de inclinómetros ópticos, electrónicos, mecánicos y de burbuja. Cada uno tiene sus propias ventajas y desventajas.

Muchos inclinómetros electrónicos están basados en acelerómetros como los de la serie Murata. Algunos otros están basados en la deformación o torsión de una varilla de acero y generalmente están fabricados para el uso industrial (incluyen una protección metálica externa), lo cual aumenta considerablemente su costo respecto a los propios acelerómetros, como los sensores de inclinación de la serie ASM.

En cuanto al resto de los inclinómetros la mayor parte de los consultados en el mercado son fabricados para el empleo directo del usuario y carecen de un protocolo de comunicación para conectarse con la placa Arduino o no cumplen con las dimensiones físicas para acoplarse a la plataforma.

Hay que remarcar que para la medición de aceleraciones estáticas el acelerómetro es muy fiable.

Los acelerómetros miden las fuerzas de aceleración, éstas pueden ser estáticas, como la gravedad que empuja a los cuerpos al centro de la Tierra, o dinámicas como el movimiento o la vibración del acelerómetro (Hoja de datos adxl345).

La unidad de medida de éste acelerómetro tanto en aceleraciones dinámicas como estáticas es precisamente la fuerza de gravedad ($9.8m/s^2$), representada por "1g". El adxl345 puede medir hasta 16g, lo que, expresado en el Sistema Internacional de Unidades serían $156.8\ m/s^2$, que es resultado de multiplicar 16 por $9.8m/s^2$. Así mismo puede medir aceleraciones tan pequeñas como 3.9 mg.

Los acelerómetros son sistemas microelectromecánicos (MEMS por sus siglas en ingles), consistiendo interiormente en una masa sísmica y un cierto tipo de desviación que detecta el trazado de circuito, bajo la influencia de la gravedad o la aceleración la masa sísmica desvía su posición neutral.

Otra forma de explicar lo anterior sería imaginar un acelerómetro como una caja cerrada en forma de cubo con una masa esférica en su interior (figura 4).

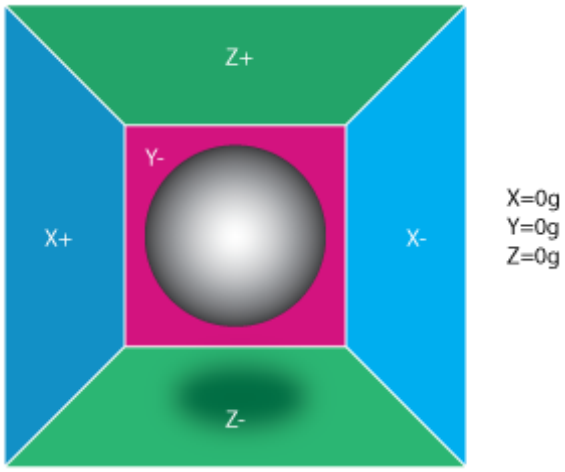


Figura 4. Un acelerómetro se asemeja a una pequeña masa dentro de una caja.

Idealmente se podría aislar la caja y la masa de la figura 4 de todo campo de gravitación o de cualquier otro que las afecte, de tal forma que, la pequeña masa esférica flotaría en el medio de la caja. Si repentinamente una fuerza externa moviera la caja hacia la izquierda y le produjera una

aceleración de 9.8m/s^2 o lo que es lo mismo 1g. La esfera en el interior golpearía la pared derecha de la caja (X-) como se aprecia en la figura 5; la pared detectaría esta presión y el acelerómetro arrojaría un valor de salida de -1g sobré el eje X de la caja.

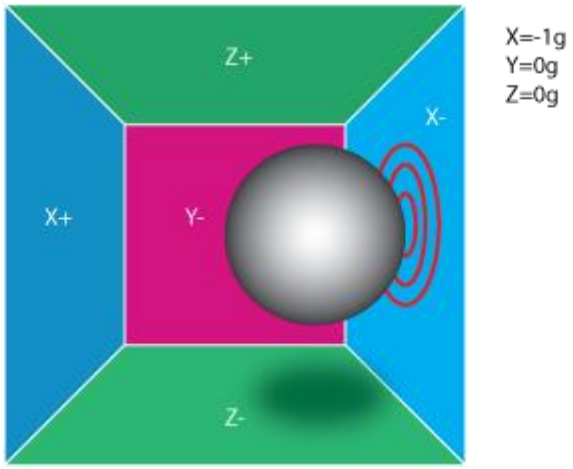


Figura 5. La masa hace presión en la cara opuesta al movimiento de la caja.

Si se girara la caja 45° en sentido horario y de nueva cuenta una fuerza externa la moviera y le produjera una aceleración de 9.8m/s^2 de manera que la masa tocara las paredes Z- y X- al mismo tiempo (figura 6), las salidas que arrojaría el acelerómetro en los ejes X y Z serían una fracción de 1g. Las operaciones para deducir este número están determinadas por el fabricante.

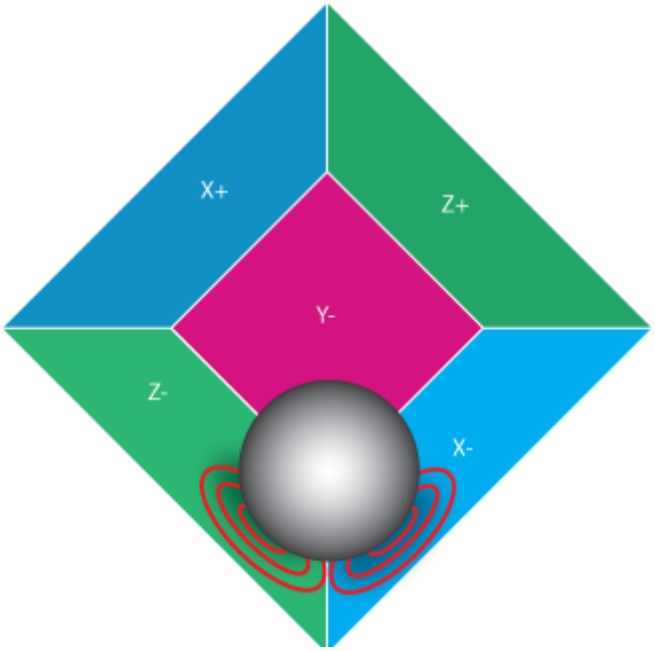


Figura 6. La masa toca dos paredes de la caja simultáneamente.

Los acelerómetros se utilizan para medir la vibración en coches, máquinas, edificios, sistemas de control y sistemas de movimiento.

En los últimos años los acelerómetros se están incorporando cada vez más en dispositivos electrónicos personales, tales como: celulares, videojuegos, contadores de pasos, etc.

El acelerómetro ADXL345 resulta ideal para medir aceleración estática en aplicaciones de sensado de inclinación (como es el caso de este proyecto).

Asimismo posee varias funciones especiales como detección de movimiento y caída libre. También percibe si la aceleración excede un umbral determinado por el usuario en alguno de los ejes.

1.8.3 Servomotores

Los actuadores precisados para este proyecto fueron dos servomotores de la marca HexTronik modelo HX12K (figura 7). Estos servomotores digitales cuentan con un engranaje metálico, un peso de 55g, torque máximo de 10Kg.cm y velocidad de 0.16seg/60grad. Cualidades suficientes para cumplir con los propósitos del presente trabajo.

El primer servo ajustado a la base inferior de la plataforma sostiene al segundo servo (ajustado a la base superior) y a la base superior. La plataforma no soporta ninguna carga en la base superior; la carga total soportada por el primer servo se encuentra dentro de los parámetros permitidos para el buen funcionamiento del servomotor. El segundo servo, al tener una carga menor que el primero de igual forma se encuentra dentro de los parámetros permitidos para su correcto funcionamiento.

Por el número de pruebas realizadas y el movimiento constante de los servos, se seleccionó el modelo anterior (HexTronik HX12K) para los dos actuadores. Las razones principales fueron el torque máximo permitido y el engranaje metálico; éste último porque presenta mayor resistencia al desgaste por fricción en comparación con los engranajes de plástico.

Figura 7. Servomotor HexTronik HX12K.

La mayoría de los servomotores comerciales pueden rotar entre 90° y 180°. Tienen un voltaje de alimentación que va de 4.8 a 7v en CD.

De acuerdo con las especificaciones del fabricante el servomotor HexTronik HX12K cuyo rango de desplazamiento va de 0° a 180°, podría moverse en espacios de menos de 1°, ya que puede modificarse el ancho de pulso de control del servo y escalarlo de 0 a 255, que es la salida máxima de frecuencia PWM que nos entrega la placa Arduino. Esto es, tener 256 posiciones distintas para un desplazamiento máximo de 180°, lo que daría una fracción de grado por cada posición. Sin embargo, la librería de Arduino usada para controlar al servomotor (*servo.h*) únicamente le permite moverse en espacios de 1°. Luego entonces, definiendo la resolución de un motor como el paso más pequeño al que puede moverse éste, el servo HexTronik HX12K tendrá una resolución de 1° para los propósitos manejados en este proyecto.

A continuación se presentan a manera de resumen algunas de las características del servomotor HexTronik HX12K:

- Engranaje Metálico.
- Torque máximo 10kg.
- Voltaje de alimentación de 5v a 7v.
- Peso 55g.
- Velocidad 0.12/60°
- Tipo de control digital.

1.8.4 Lenguaje Arduino

El microcontrolador en la placa Arduino se programa mediante el lenguaje de programación Arduino (basado en Wiring) y el entorno de desarrollo Arduino (basado en Processing). Los proyectos hechos con Arduino pueden ejecutarse sin necesidad de conectar a un ordenador, si bien tienen la posibilidad de hacerlo y comunicar con diferentes tipos de software (p.ej. Flash, Processing, MaxMSP).

Arduino se puede utilizar para desarrollar objetos interactivos autónomos o puede ser conectado a software del ordenador (por ejemplo: Macromedia Flash, Processing, Max/MSP, Pure Data). Las placas se pueden montar a mano o adquirirse. El entorno de desarrollo integrado libre se puede descargar gratuitamente.

En la figura 8 se muestra el ambiente de programación de la versión Arduino 1.0.5 (usada en este proyecto).

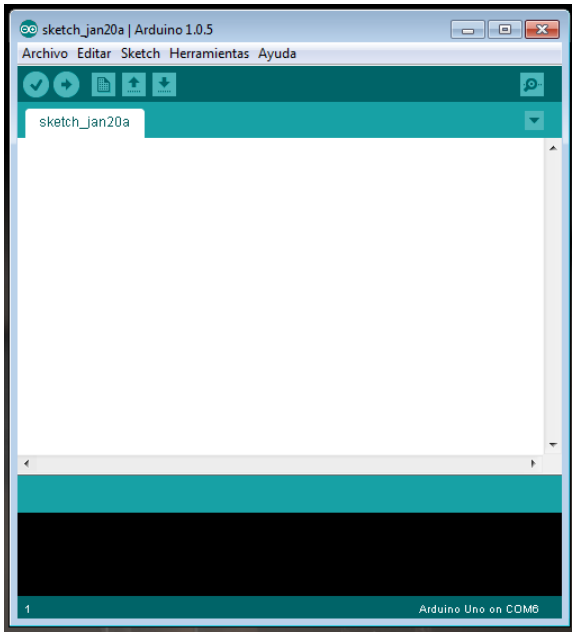


Figura 8. Arduino 1.0.5 basado en Processing.

1.8.5 Software de la interfaz gráfica (Processing)

Processing es un lenguaje de programación y entorno de desarrollo integrado de código abierto basado en Java, de fácil utilización, y que sirve como medio para la enseñanza y producción de proyectos multimedia e interactivos de diseño digital.

Este lenguaje de programación es usado para crear la interfaz gráfica del proyecto. La figura 9 muestra el ambiente de programación de la versión Processing 2.1 (usada en ese proyecto), el cual es muy similar al de Arduino.

Processing es compatible con distintas versiones de Windows, Mac OS X y Linux.

Gracias a la política de software libre de Processing, es posible exportar y descargar las librerías disponibles bajo su licencia, lo que significa que se puede hacer uso de las librerías e incluirlas dentro de un proyecto sin necesidad de abrir el propio código del proyecto al público en general.

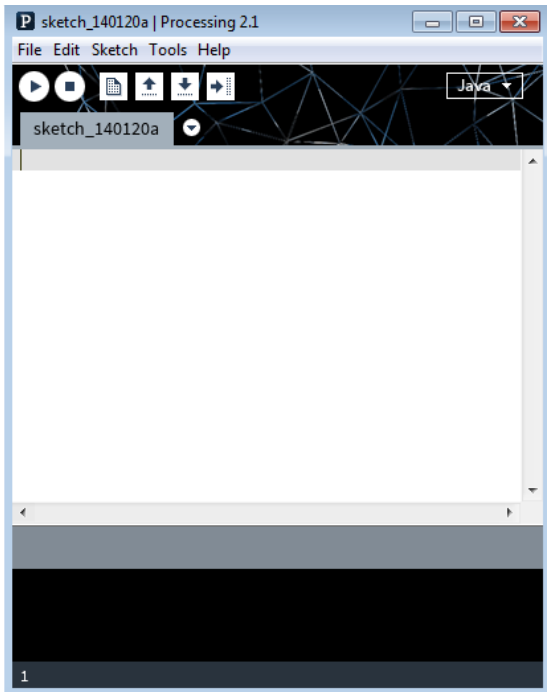


Figura 9. Processing 2.1

Tanto Arduino como Processing son proyectos de software libre. La manera de programar es muy similar en ambos, ya que comparten una gran cantidad de comandos y operadores. Esto que permite migrar de un entorno a otro sin necesidad de hacer grandes modificaciones al programa.

La aplicación hecha en processing de la interfaz gráfica se ejecuta directamente en la PC, tiene comunicación directa con la placa Arduino, pero no incide de ninguna forma en el control de la plataforma. Lo mismo sucede con la propia PC, que si bien es un medio para programar en la placa Arduino, al momento de correr el programa todas las funciones recaen en esta última.

La interfaz gráfica es presentada en el apéndice C.

Capítulo 2

CONTROL DIFUSO

En este capítulo se presenta una breve introducción a la construcción de un controlador basado en lógica difusa, llamado control difuso; en este trabajo el control difuso se basa en la generalización del modus ponnens para conjuntos difusos.

2.1 Conceptos básicos

Los sistemas expertos de control difuso basados en reglas, conocidos como controladores difusos, son sin duda la aplicación más extendida de la lógica difusa [Martín 2002]. Un primer bloque realiza un preprocesado 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 CLD puede ser presentado en forma similar a la de un control clásico.

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

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

Un CLD no es una función de transferencia de ecuaciones diferenciales.

La base de información del CLD 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 CLD.

Un CLD 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 CLD.

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 CLD es caracterizada con un *SI*, el cual se llama antecedente y con un *ENT* 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 CLD 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 CLD, representada por (27).

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

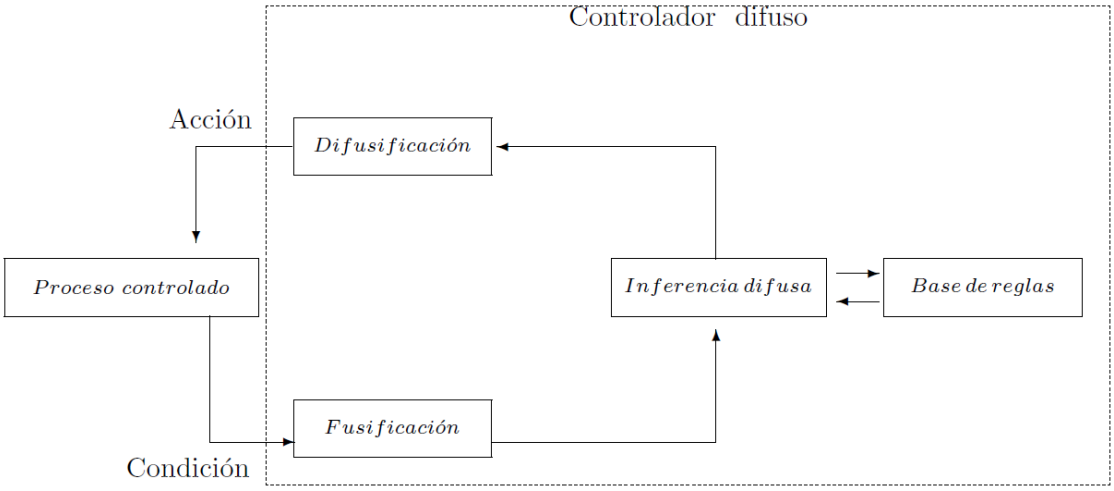


Figura 10. Funcionamiento de un controlador difuso.

En general un controlador difuso (figura 10) 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 [Klir y Yuan 1995].

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 modulo de fusificación.
- Un modulo de defusificación.

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

2.2 Razonamiento Aproximado

2.2.1 Sistema difuso experto

El propósito de este capítulo es cubrir los fundamentos del razonamiento basado en reglas difusas de producción, que es referido como razonamiento aproximado. Este material es esencial para el diseño de motores de inferencia para sistemas difusos expertos.

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 [Nguyen et al 2003].

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.

2.3 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
Antecedente: 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:

$$r_j\,(A') = h(A' \cap A_j) \tag{29}$$

usando la intersección difusa estándar,

$$r_j\,(A') = \sup_{x \in X} min\,[A'(x), A_j(x)] \tag{30}$$

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:

$$B' = \sup_{j \in N_n} min\,[r_j(A', B_j(y))] \tag{31}$$

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

$$R(x, y) = \sup_{j \in N_n} min\,[A_j(x), B_j(y)] \tag{32}$$

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*.

2.4 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, véase figura 25.

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 [Klir y Yuan 1995]. 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 [Klir y Yuan 1995]. Para el caso continuo, el valor es calculado por la fórmula:

$$d_{CA}(C) = \frac{\int_{i=1}^{n} zC(z)dz}{\int_{i=1}^{n} C(z)dz} \tag{33}$$

donde:

- $z \in Z$ (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:

$$d_{CA}(C) = \frac{\sum_{i=1}^{n} z_{oi} C_i(z)}{\sum_{i=1}^{n} C_i(z)} \tag{34}$$

donde:

- $z_{oi} \in Z$ 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:

$$d_{CM}(C) = \frac{\inf M + \sup M}{2} \tag{35}$$

donde:

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

Para el caso discreto:

$$d_{CM}(C) = \frac{\min\{z_k/z_k\epsilon\, M\} + \max\{z_k/z_k\epsilon\, M\}}{2} \tag{36}$$

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

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:

$$d_{MM}(C) = \frac{\sum_{z_k\epsilon M} z_k}{|M|} \tag{37}$$

Como se menciona anteriormente, este es el modelo del control difuso, que en el presente trabajo funcionará como el lazo externo del control.

Capítulo 3

PLATAFORMA

El presente proyecto plantea la estabilización de una plataforma de 2 grados de libertad. La plataforma se considera estabilizada cuando se registre 0 grados de inclinación tanto en el eje x como en el eje y en un sistema dextrógiro, en otras palabras, cuando el plano xy de la base superior de la plataforma sea paralelo a la superficie de la Tierra. Si durante el tiempo de funcionamiento de la plataforma, se registran valores de inclinación de -5° a 5° en cualquiera de los ejes, la plataforma se considera estabiliza. El prototipo tiene una posición inicial estable ($x = 0$, $y = 0$) y desde este punto comenzará a detectar las variaciones en la inclinación. El giro de la plataforma se efectuará sobre los ejes x y y; dicho giro irá de -45° a 45° para el eje x y de -45° a 30° par el eje y, esto último debido a las limitaciones físicas de la plataforma.

Una imagen del prototipo en CAD se muestra en la figura 11.

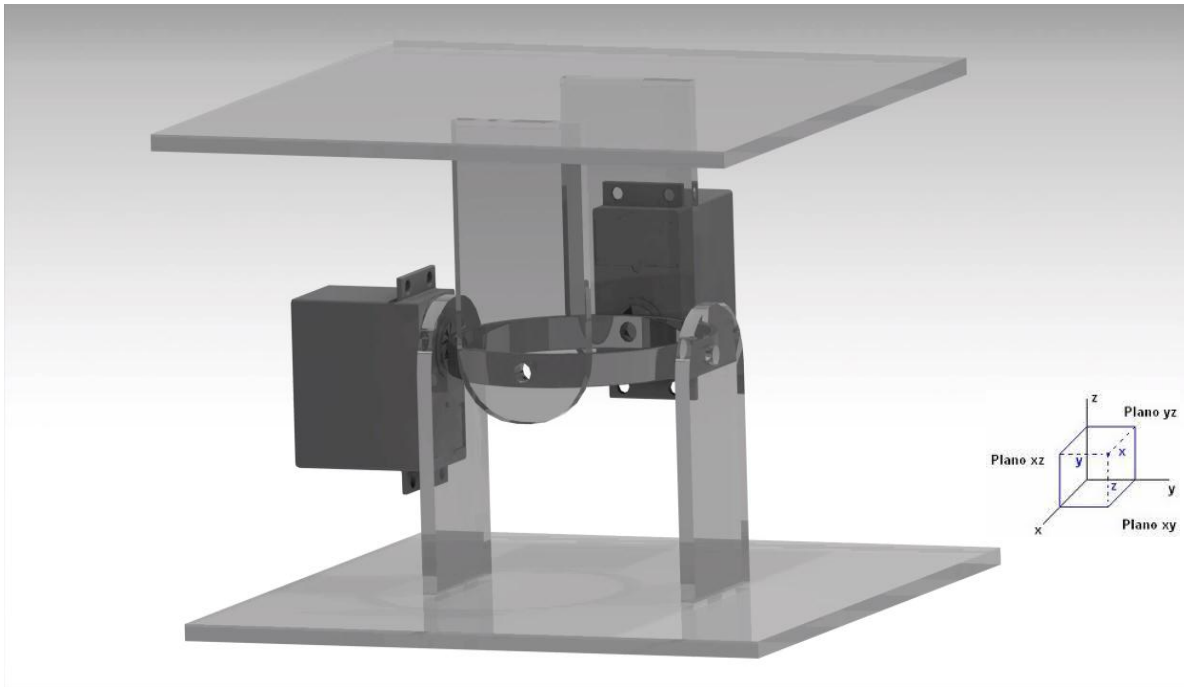


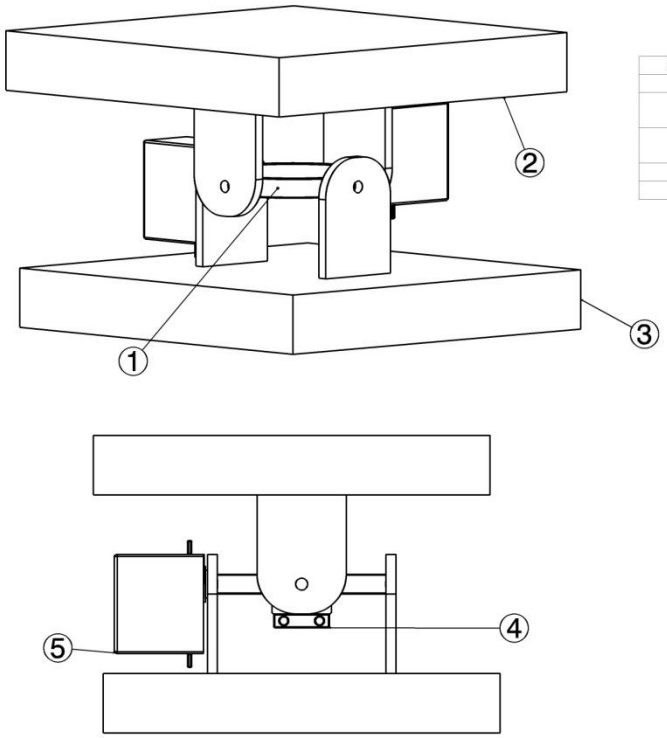
Figura 11. Plataforma de tres grados de libertad con anillo al centro.

La estabilización se lleva a cabo gracias a dos servos colocados en la estructura, los cuales al hacer rotar sus ejes modifican la inclinación de la base superior de la plataforma.

La base inferior de la plataforma, idealmente debiera estar sujeta a un cuerpo externo y éste es el que debiera moverse generando desequilibrio; sin embargo, por razones prácticas e ilustrativas la base se mantendrá independiente de dicho cuerpo, para que el usuario pueda manipularla a libertad. Las bases de la plataforma son exactamente iguales, tanto en material de construcción como en dimensiones.

Las figuras 12 y 13 muestran algunas de las especificaciones mecánicas de los componentes de la plataforma (las dimensiones están dadas en milímetros.)

Plataforma



Marca	Cantidad	Nomenclatura	Material
1	1	Anillo	Acero
2	1	Base Superior	Acrílico
3	1	Base Inferior	Acrílico
4	1	Servo_A	--------
5	1	Servo_B	--------

Figura 12. Componentes de la plataforma.

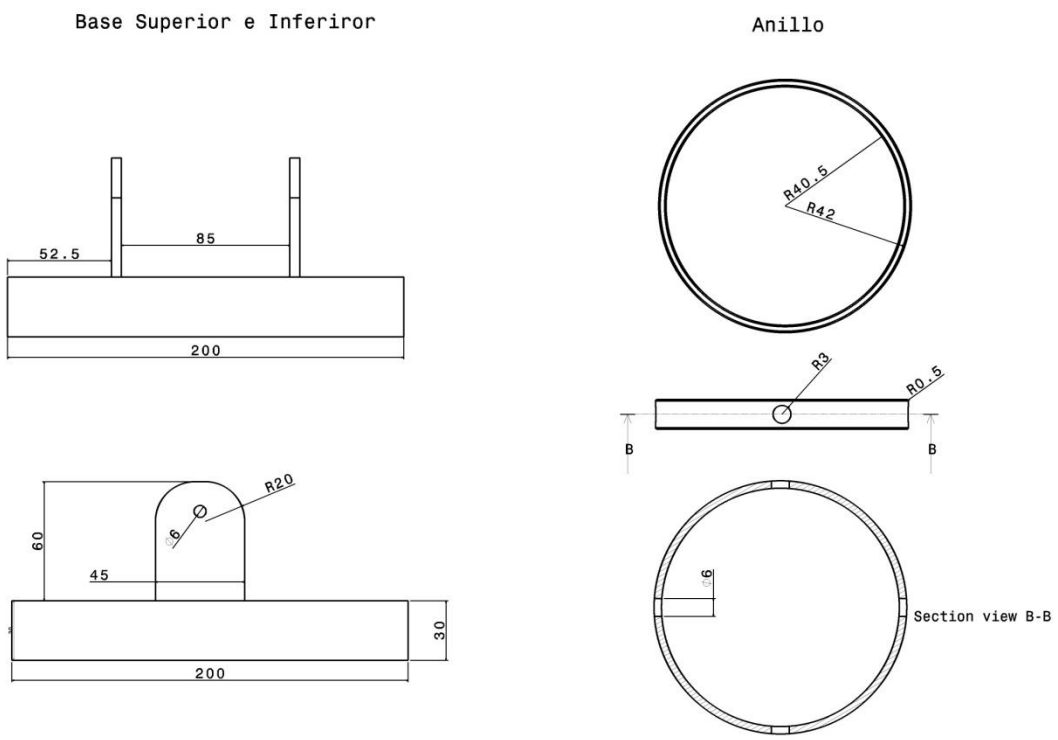


Figura 13. Dimensiones de los componentes.

Los servomotores que se encuentran unidos físicamente a las bases de acrílico de la plataforma por un anillo de acero; dicho anillo es un eslabón (un sujetador que une o conecta las partes de la plataforma) que está unido a las bases superior e inferior mediante articulaciones de rotación. Teniendo así una articulación para el movimiento de la base inferior y otra para el movimiento de la base superior.

Se considera a la plataforma como un robot, pues es un sistema que cuenta con estructura mecánica y de transmisión, sensores, etapa de potencia, sistemas de control y elementos terminales (Barrientos et al. 2007).

Cualquier movimiento independiente que una articulación pueda realizar respecto a otra anterior, constituye un grado de libertad.

Una cadena cinemática, es una serie de eslabones unidos por. La estructura mecánica de la plataforma constituye una cadena cinemática dónde los eslabones son las bases superior e inferior unidos por articulaciones de rotación. Si en una cadena cinemática se puede llegar desde un eslabón a otro por al menos dos caminos, se dice que es una cadena cinemática cerrada. Por el contrario que sólo exista una ruta posible, la cadena cinemática será abierta.

La figura 14 muestra algunas de las articulaciones usadas en mecanismos. En este trabajo sólo se hace uso de las articulaciones de rotación.

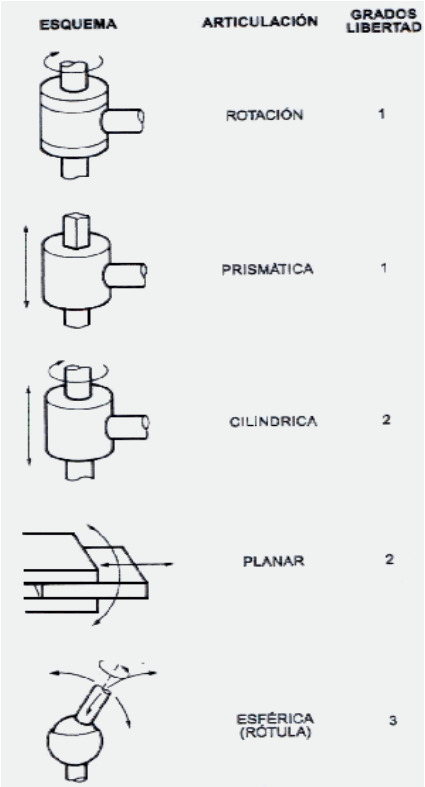


Figura 14. Articulaciones: movimientos y grados de libertad.

3.1 Caracterización de los Componentes Electrónicos

3.1.1 Caracterización del acelerómetro

Antes de poder usar el acelerómetro se necesita crear una conexión acelerómetro-arduino que permita leer los datos que nos envía el sensor.

Esta conexión se implementa mediante el bus I^2C. Con el circuito de la figura 15 conectamos el acelerómetro a la placa arduino UNO.

Las terminales SDA (*Serial Data*) y SCL (*Serial Communications Clock*) del ADXL345 van conectadas a los pines A4 y A5 de la placa Arduino respectivamente.

Se coloca un par de resistencias pull up de 10Kw, una colocada entre SDA y el pin de 3.3v del Arduino; la segunda va de SCL al pin de 3.3v de nueva cuenta. Lo anterior se sugiere en la hoja de datos del ADXL345 para la apropiada operación en I^2C.

SDO (Alternate I^2CAddress Select) se conecta a tierra; CS (*Chip Selec*t) se conecta al pin de 3.3v, finalmente GND va a tierra y VCC a 3.3v. Diagrama de conexión en la figura 15.

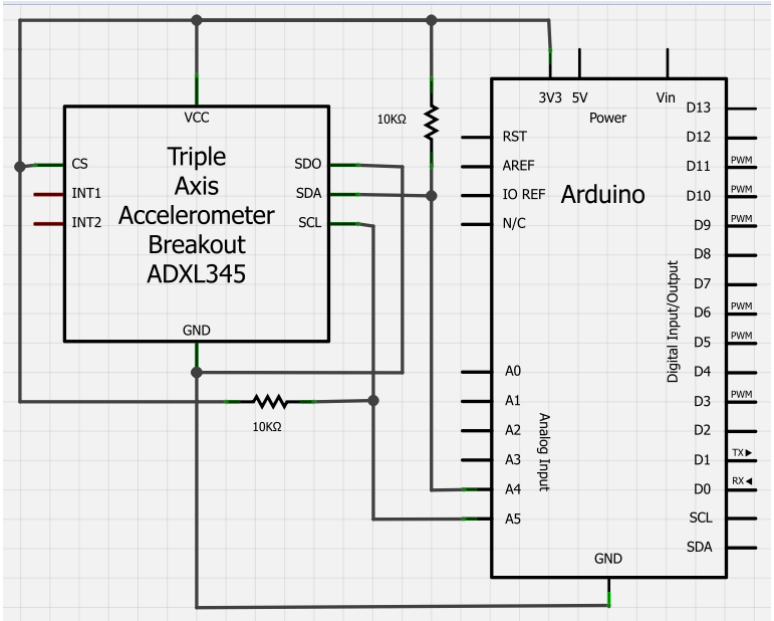


Figura 15. Conexión acelerómetro-arduino para la comunicación en serie I^2C.

El circuito puede verse de manera *física* en la figura 16 (sólo con fines ilustrativos).

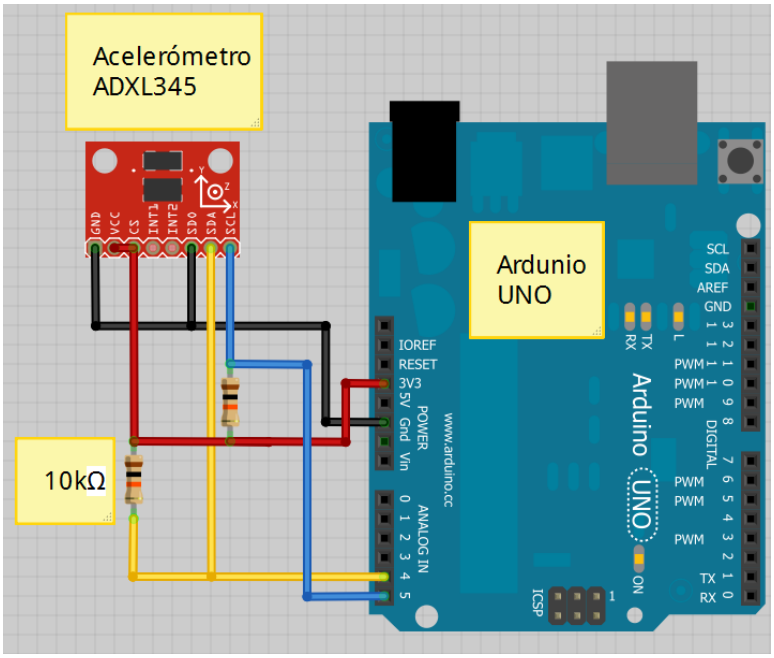


Figura 16. Conexión *física* acelerómetro-arduino.

Una vez que el circuito anterior está montado, se procede a escribir el código que permita obtener y manipular los valores que entrega el sensor.

En la hoja de datos del ADXL345 (pág.18) se muestra la información para la configuración de comunicación con el protocolo I^2C.

Existen dos formas para comunicarse con el sensor, los protocolos SPI e I^2C. En este proyecto se opta por el protocolo I^2C por reducir el cableado a sólo dos líneas de comunicación; una para datos y otra para la señal de reloj (SDA y SCL respectivamente). En contraste, el protocolo SPI requiere de 3 a 4 líneas dependiendo la configuración.

Arduino cuenta con la librería *Wire* que permite la comunicación con componentes que implementen el protocolo I^2C.

El acelerómetro ADXL345 tiene un número determinado de registros que deben ser configurados para poder hacer uso de éste. Los detalles de ésta configuración, así como el código con el que se implementa en la placa Arduino se encuentran en el apéndice A.

3.1.2 Acelerómetro como sensor de inclinación

Un acelerómetro puede ser usado para determinar la inclinación de un cuerpo en el espacio.

Como lo indica su nombre un acelerómetro mide aceleraciones, éstas pueden ser dinámicas (una vibración por ejemplo) o estáticas (la atracción de la gravedad). Se aprovecha la última para calcular la inclinación respecto a la superficie de la Tierra.

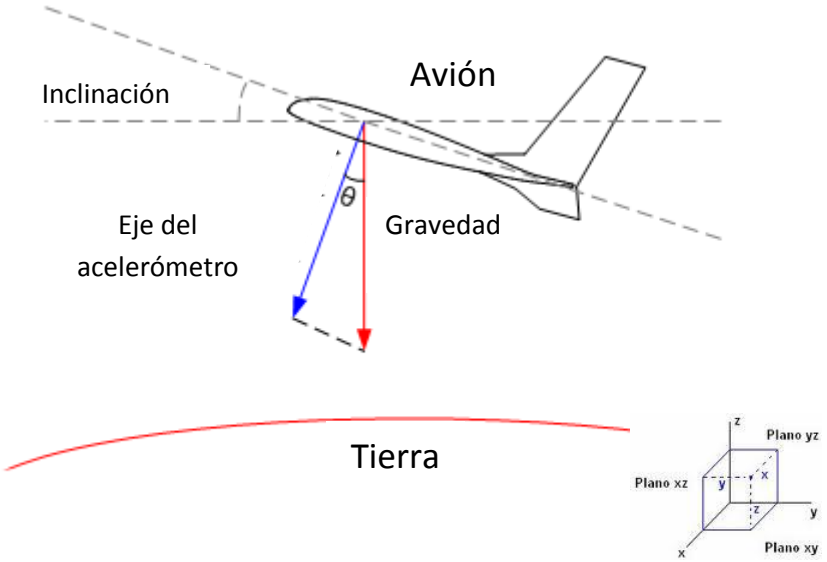


Figura 17. Cálculo de la inclinación de un cuerpo usando un acelerómetro.

En la figura 17 la flecha roja representa la gravedad de la Tierra, la flecha azul representa la aceleración estática en uno de los ejes del acelerómetro, producida por la fuerza de gravedad. La flecha azul es perpendicular a la base del avión. El acelerómetro se coloca de tal forma que su plano XY sea paralelo al plano XY del avión (la base).

El ángulo teta que se encuentra entre la flecha roja (gravedad) y la flecha azul (eje del acelerómetro) se relaciona con la inclinación del avión respecto a la superficie terrestre con la ecuación (36).

$$Inclinación = \theta + \frac{\pi}{2} \tag{38}$$

Conociendo teta y la magnitud de la gravedad, podremos conocer la inclinación del cuerpo con algunos cálculos.

Tenemos la aceleración estática

$$aceleración = gravedad * \cos(\theta) \tag{39}$$

Luego

$$\theta = arccos(aceleración/gravedad) \tag{40}$$

Dado que

$$Inclinación = \theta + \frac{\pi}{2}$$

Y

$$\arccos(x) = \frac{\pi}{2} - arcsen(x) \tag{41}$$

$$sen(y) = x \leftrightarrow y = \pi - arsen(x) + 2k\pi$$

Entonces

$$inclinación = \pi - arcsen(aceleración/gravedad)$$

$$inclinación = arcsen(aceleración/gravedad)$$

Como

$$gravedad = 1g$$

Finalmente

$$inclinación = arcsen(aceleración) \tag{42}$$

La respuesta del acelerómetro depende de la orientación que tenga respecto a la gravedad, como se aprecia en figura 18.

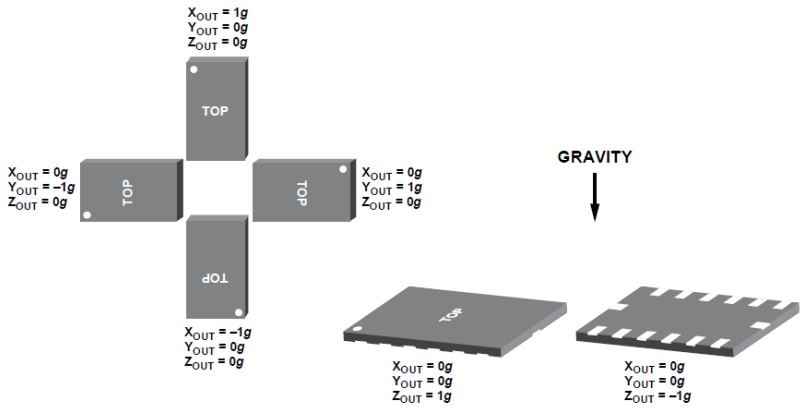


Figura 18. Respuesta de salida contra la gravedad.

3.1.3 Caracterización de servomotores

Los servomotores contienen circuitos de control y potenciómetros (resistencias variables), estos se encuentran en el eje central del motor. Los potenciómetros permiten al sistema de control, supervisar el ángulo actual del motor. Si éste se encuentra en el ángulo correcto, entonces el motor se mantendrá estable y no se moverá. En cambio si el sistema detecta que el ángulo es incorrecto, el motor girará en la dirección adecuada hasta llegar a dicho ángulo deseado. El eje de los servomotores es capaz de alcanzar los 180 °.

Para implementar el control de un servomotor sólo hace falta enviar una señal codificada en la línea de entrada, aplicando dicha señal, el motor cambiará y mantendrá la posición angular hasta que exista otro cambio en la señal de entrada. La señal codificada se usará para comunicar el ángulo determinado. Esta codificación es manejada por la modulación del ancho del pulso (PWM por sus siglas en inglés), el servo refresca la posición cada 20 milisegundos (ms). La longitud del pulso determinará el giro del motor. Teniendo un pulso de 1.5 ms, este llevará el servomotor a la posición central (90°), si el pulso es menor a 1.5 ms el motor se acercará a los 0° y si el pulso es mayor a 1.5 ms, este se acercará a la posición de 180°. Como lo muestra la figura 19.

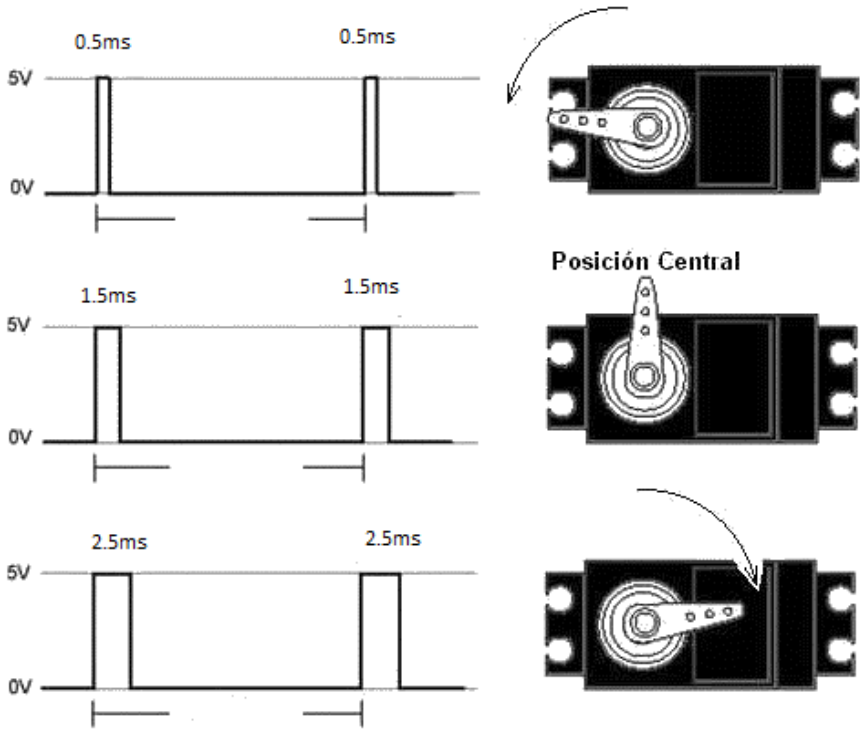


Figura 19. Control de un servomotor.

Los servomotores tienen tres cables: alimentación, tierra, y señal. El cable de alimentación suele ser rojo, y debe ser conectado a una fuente de alimentación externa de 5v. El cable a tierra es normalmente de color negro o marrón y se debe conectar al pin de tierra de la placa Arduino. El cable de señal es generalmente de color amarillo, naranja o blanco y debe ser conectado a un pin digital con salida PWM (D9 y D11 para cada servo) en la placa Arduino. Se debe conectar la tierra del Arduino y la fuente de alimentación externa. El diagrama de la figura 20 muestra la conexión de los servomotores con la placa Arduino.

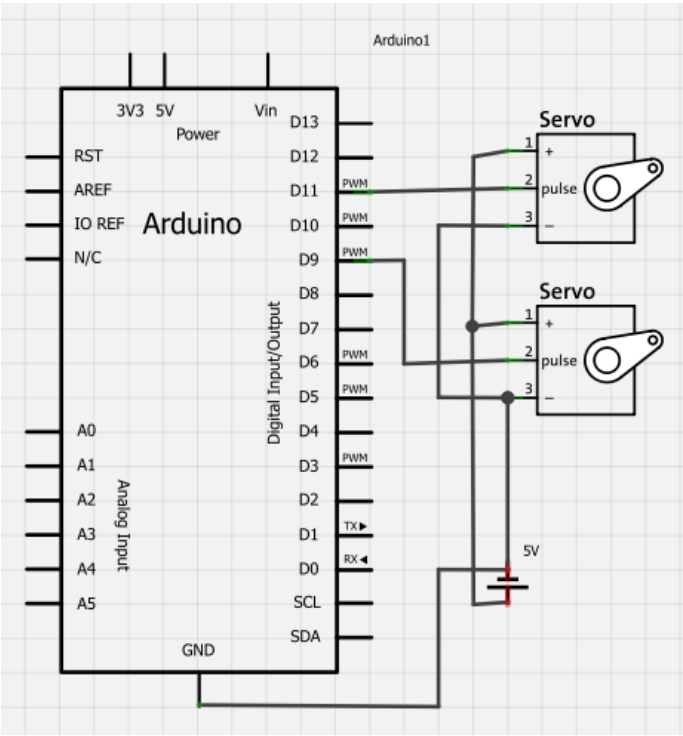


Figura 20. Conexión de servos sobre la placa Ardunio.

En la figura 21 puede verse de manera *física* la conexión de los servomotores a la placa Arduino UNO (sólo con fines ilustrativos).

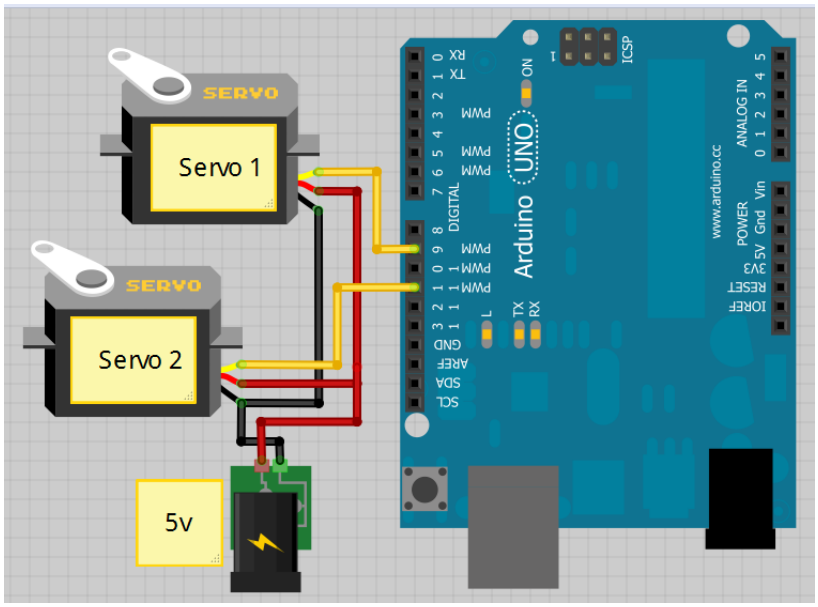


Figura 21. Conexión *física* de servos en Arduino.

Arduino cuenta con una librería (*Servo*) que permite controlar servomotores de manera sencilla. Esta librería tiene una función (*write()*)que realiza una analogía entre los grados de rotación del servo (180° en este caso) y las longitudes de los pulsos PWM necesarios para moverlo. Así por ejemplo si un pulso de 1.5ms lleva al servo a una posición central, dentro del programa se escribiría entonces 90, pues es el punto medio entre 0 y 180 grados.

No todos los fabricantes siguen la misma norma en cuanto a los tiempos de anchura de pulso, incluso un mismo fabricante puede variarla según el modelo del servomotor. Es por eso que antes de hacer uso de éste, debe de hacerse una prueba de su comportamiento.

Los detalles de la configuración de los servomotores, así como el código con el que se implementan las pruebas de comportamiento en la placa Arduino, se encuentran en el apéndice B.

3.2 Control Difuso de la Plataforma

Se presenta el diagrama general de la planta (figura 22), del cual la presente sección sólo se enfocara en la descripción del controlador difuso. Luego se mostrarán los conjuntos difusos y la base de reglas implementada, también se explicará el método de defusificación y la forma de escritura de las reglas.

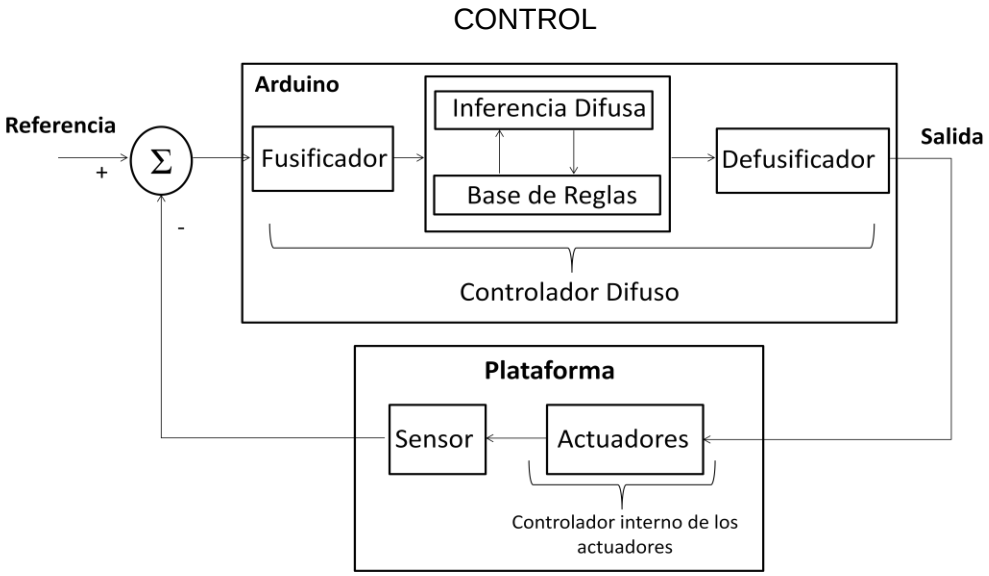


Figura 22. Diagrama General del Sistema.

Dado que el control de la plataforma se basa en el movimiento que producen los dos servomotores, es importante decir que cada servo recibe una señal que proviene de un sistema de control independiente, es decir que a pesar de que las señales para los actuadores provienen de un mismo dispositivo (Arduino UNO), hay una sección del código del programa para cada servo dedicada exclusivamente al control de su eje y de ningún otro. Es importante mencionar que cada servo tiene un controlador interno, en este trabajo es el lazo interno (figura 22), el cual funciona de forma separada del control difuso de la plataforma; esto se debe a la naturaleza del circuito implementado por el fabricante en el servo, que refresca su posición cada cierto tiempo (20 ms).

El *loop* principal del programa contiene las funciones que aparecen en orden de ejecución en la tabla 1.

Tabla 1. Funciones principales del programa de control difuso.

No.	Nombre de la función	Comentarios
1	Lectura de aceleración	Recibe y transforma los datos del acelerómetro.
2	Conjuntos	Asigna valores de pertenencia a los datos de la función 1.
3	Reglas	Realiza las comparaciones con los valores de pertenencia donde intervienen la inferencia difusa y la base de reglas**.**
4	Defusificación	Convierte los valores borrosos trabajados en valores no borrosos.
5	Salida	Envía los datos no borrosos al cable de control de los servos.

De acuerdo con la tabla 1 es posible dividir el programa en Arduino en tres etapas principales: Inicio, Controlador Difuso y Salida. Las cuales se resumen en el diagrama de flujo de la figura 23.

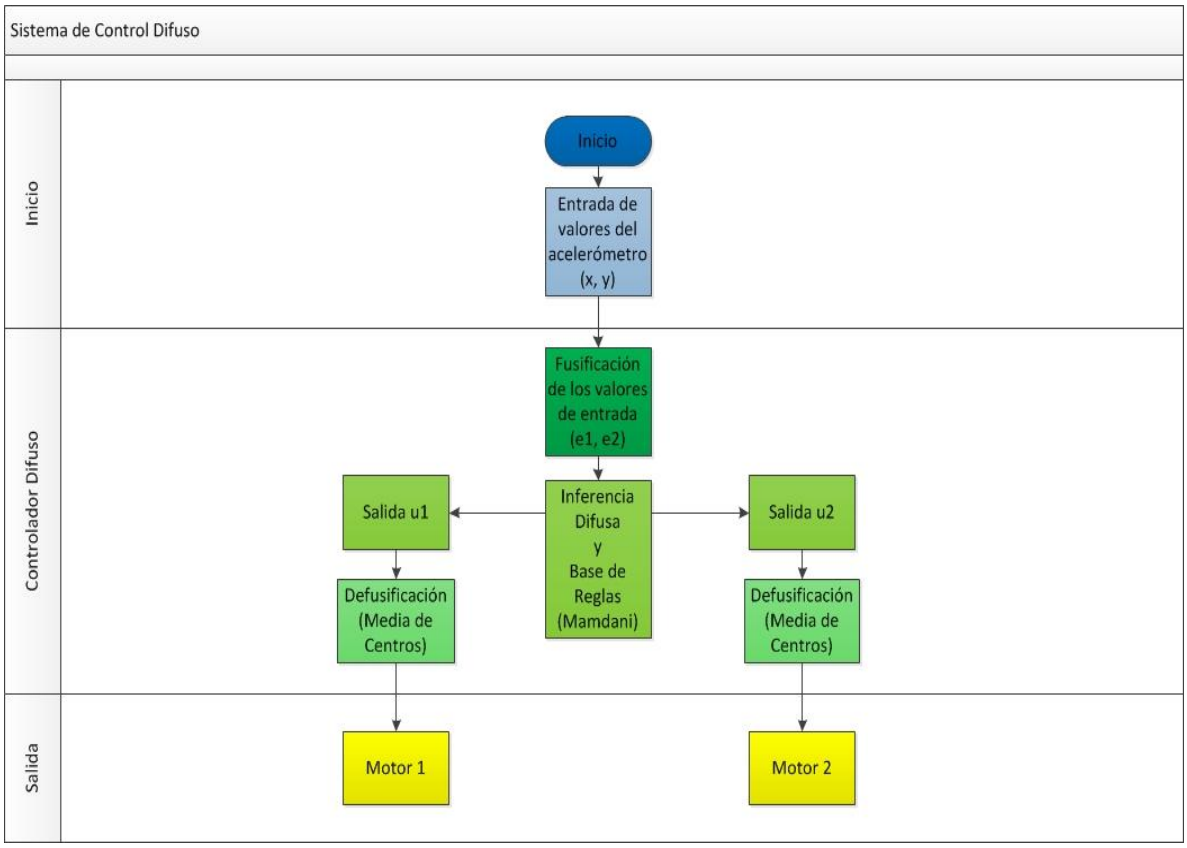


Figura 23. Diagrama de flujo del programa en Arduino.

En la etapa de Inicio se reciben los datos (con unidades en g's) enviados por el acelerómetro, se procesan para convertirlos a radianes (capítulo 3.1.2) y posteriormente a grados. Es también en esta etapa dónde se inicializa el puerto serial para comunicar la placa Arduino con la interfaz gráfica en la PC.

En la etapa de Controlador Difuso las variables que guardan la posición de la plataforma (en grados) se fusifican en los conjuntos difusos Bajo, Medio, Alto y en el conjunto difuso Hecho. Después, las variables fusificadas entran al motor de inferencia difusa donde se encuentra la base de reglas tipo Mandani de la cual obtenemos las salidas no defusificadas del sistema.

En la etapa de Salida se defusifican los valores difusos (obtenidos en la etapa de Controlador Difuso) por el método de Media de Centros usando los conjuntos difusos *singleton* BAJO', MEDIO' y ALTO'. Al obtenerse los valores de la defusificación se envían como señales PWM a los servomotores de la plataforma.

Cada servomotor se ocupa de un solo movimiento de la plataforma, ya sea alrededor del eje X o Y, sin embargo ambos pueden moverse a la vez combinando sus trayectorias, dándole así mayor versatilidad a la plataforma.

El número de reglas de inferencia se mantienen en 9 por las combinaciones posibles de los conjuntos difusos *e* y ê

Se refiere a la frecuencia de muestreo como el número de muestras por unidad de tiempo que se toman de una señal continua para producir una señal discreta, durante el proceso necesario para convertirla de analógica en digital. El número de muestras promedio es una elección del diseñador, pero el acelerómetro lo estandariza en 10 muestras cada 100ms, esto es para velocidades de trabajo de 100Hz o superior, siendo la primera la que se maneja en este proyecto.

En cuanto al tiempo de respuesta del sistema, podemos aproximarlo midiendo el tiempo de ejecución del programa, el cual es de 300ms por ciclo completado.

Un sistema en tiempo real es aquel que debe producir respuestas concretas en un intervalo de tiempo definido. Si el tiempo de respuesta excede ese límite, se produce una alteración en el funcionamiento.

El estimado de la respuesta de la plataforma ante cambios en su inclinación es de 0.5s. Con esto no se afirma que la plataforma alcance la posición horizontal ante una perturbación en medio segundo, sino que los actuadores responderán a esa perturbación en ese lapso o menor.

Debido a la naturaleza del control difuso y a que este trabajo se considera un primer acercamiento al manejo de la plataforma aquí presentada, este sistema debe de ubicarse, según la teoría de sistemas en tiempo real, dentro de los sistemas de tiempo real no estricto. Dichos sistemas permiten la pérdida ocasional de las especificaciones del tiempo, aunque deben cumplirse de manera general.

La memoria del microcontrolador posee una capacidad de alrededor de 32Kb, de las cuales el programa ocupa 30Kb aproximadamente. Esto indica que el programa está apenas por debajo de la capacidad total permitida. Si se decidiera hacerle modificaciones, en el sentido de aumentar el número de reglas o conjuntos difusos, se correría el riesgo de sobrepasar la capacidad de memoria disponible.

El ambiente de programación Arduino (figura 24) muestra al momento de la compilación los datos referentes al tamaño del código y la capacidad de memoria del microcontrolador de la placa Arduino UNO. En la figura 24 aparece el ambiente de programación Arduino donde se muestra el tamaño del código usado en el control de la plataforma.

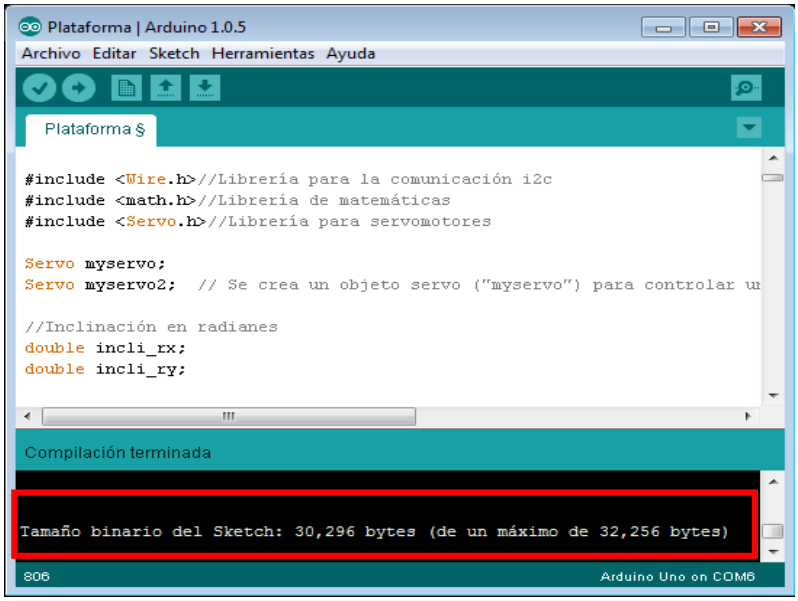


Figura 24. Tamaño del programa enmarcado en rojo.

Se hace mención que tanto la interfaz gráfica como la propia PC no tienen mayor incidencia en el control, que la de monitorear y presentar de forma visual los cambios en la posición de la plataforma.

3.2.1 Conjuntos Difusos

Algunos aspectos para considerar el tipo y el número de conjuntos difusos se basan tanto en la literatura como en trabajos realizados anteriormente.

Se escoge a los conjuntos difusos tipo "S" por ser continuos, en los intervalos que se caracterizaron motores y el acelerómetro.

Las condiciones son los ángulos en los que se desplaza la plataforma y el recorrido de los motores. Estos datos permiten proponer los parámetros en los que se construyen los conjuntos difusos.

Los conjuntos difusos tipo "S" usados para describir los distintos niveles de inclinación de la se describen con la función 43.

$$A(x) = \begin{cases} 0 & x \leq a \\ 2\{(x-a)/(b-a)\}^2 & x \in (a,m] \\ 1-2\{(x-b)/(b-a)\}^2 & x \in (m,b) \\ 1 & x \geq b \end{cases} \tag{43}$$

La función S (figura 24) es definida por sus límites inferior y superior a y b respectivamente; además del valor del punto de inflexión m, tal que $a < m < b$.

Un valor común de m es $(a + b)/2$. El incremento de la función es más lento conforme mayor sea la distancia $a - b$.

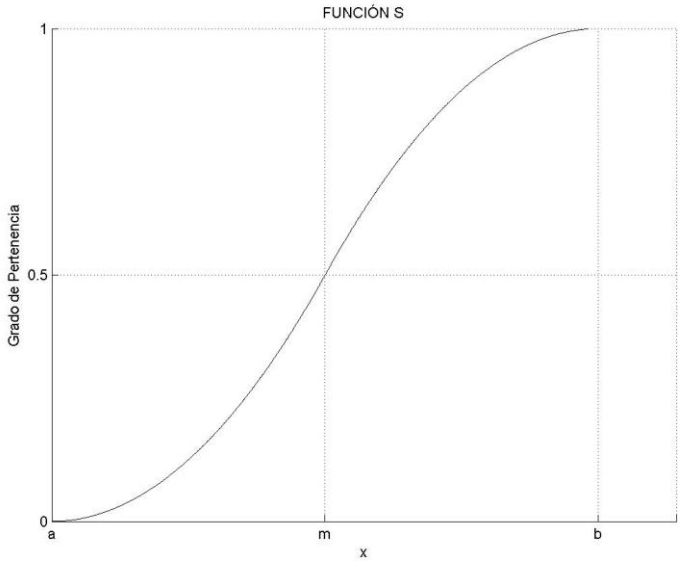


Figura 25. Función de pertenencia tipo S.

Los datos referentes a la inclinación del eje x que provienen del acelerómetro se guardan en la variable $x°$, la cual está asociada al servomotor 1. Dicha variable se fusifica al ser evaluada en los conjuntos difusos tipo S que aparecen en la figura 25. El valor de pertenencia de la variable fusificada se almacena ahora en la variable e_1.

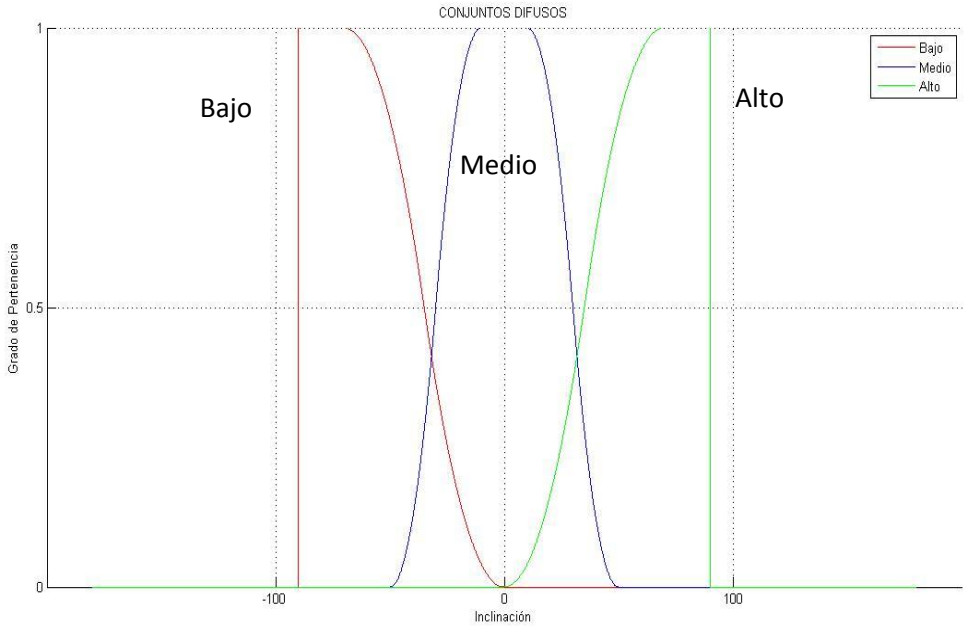


Figura 26. Conjuntos difusos Bajo, Medio y Alto asociados a la variable de entrada x° y y°.

Los datos referentes a la inclinación del eje y se guardan en la variable $y°$ que está asociada al servomotor 2. De igual forma que en $x°$ el valor de pertenencia de la variable fusificada $y°$ se almacena ahora en e_2.

Los conjuntos difusos tipo S (Bajo, Medio y Alto) asociados a la variable e_2 tienen los mismos valores que los asociados a su homóloga e_1.

Los valores usados para la representación de los conjuntos difusos Bajo, Medio y Alto se encuentran en la tabla 2.

Tabla 2. Valores de definición de los conjuntos de fusificación.

Conjuntos Difusos	**a**	**b**	**a_2**	**b_2**
Bajo	-70	0	-	-
Medio	-50	-10	10	50
Alto	0	70	-	-

Dadas las entradas ($x°$ y $y°$) se obtienen los distintos valores de pertenencia para cada una. A esto se le llama "fusificación de la entrada". Si la variable fusificada tiene más de un valor de pertenencia, a continuación se aplica algún operador (intersección en este caso) obteniendo un único valor de pertenencia.

Como en el sistema de control la única magnitud con la que se trabaja es la inclinación, ésta se compara continuamente consigo misma durante la ejecución del programa. Los valores de las variables x° y y°se actualizan regularmente, con lo que los valores próximos pasados se almacenan en variables *espejo*. Así, cada entrada x° y y° tienen una variable espejo x_0 y y_0 asociadas respectivamente, las cuales se definen como:

$$x_0 = 0 - x^{\circ} \tag{44}$$

$$y_0 = 0 - y^{\circ} \tag{45}$$

Dadas x_0 y y_0, se obtienen los valores de pertenencia para cada una de ellas, evaluándolas en los conjuntos difusos Bajo, Medio y Alto. Lo que resulta de la fusifiación anterior son las nuevas vaiables $\hat{e}_1$ y $\hat{e}_2$.

Tanto e como $\hat{e}$ son dos premisas que se usarán para formar las reglas de inferencia del sistema.

Las variables e y $\hat{e}$ deben entenderse como variables difusas que representan la inclinación de la plataforma durante la ejecución del programa; en el instante actual (e) y en el instante anterior ($\hat{e}$).

Al inicio del programa el primer valor fusificado de la inclinación se almacena en la varible e, éste se compara con el valor de la inclinación en un instante anterior almacenado en la variable $\hat{e}$; sin embargo en el primer ciclo de ejecución del programa no existe propiamente un *instante anterior*, por lo que el valor de $\hat{e}$ es igual a cero. Una vez completado el primer ciclo del programa $\hat{e}$ tendrá el valor de la fusificación de x_0 o y_0 según sea el caso. Todo lo anterior para cualquiera de los dos ejes x y y.

Como se tienen dos términos (e_n y $\hat{e}_n$), el valor de pertenencia único se determina aplicando el operador intersección entre ellos, como lo muestra la ecuación (46).

$$k_m = min\,[e_n, \hat{e}_n] \tag{46}$$

Donde

- k_m es el valor de pertenencia único.
- $m = 1, 2, 3 \ldots, 9.$
- $n = 1, 2.$

Las variables e_n y $\hat{e}_n$ al ubicarse en los conjuntos difusos Alto, Medio o Bajo generan 9 posibles combinaciones según las reglas de inferencia; de esto pueden obtenerse 9 posibles valores para k dependiendo de la combinación anterior. Es decir, si e_1 perteneciera al conjunto difuso Alto y $\hat{e}_1$ perteneciera al conjunto difuso Bajo, k_7 será el resultado de aplicar el operador intersección al par [Alto, Bajo] según la ecuación (46). Si por el contrario en el siguiente ciclo del programa e_1 perteneciera al conjunto difuso Medio y $\hat{e}_1$ perteneciera al conjunto difuso Medio, k_5 será el resultado de aplicar el operador intersección al par [Medio, Medio] según la misma ecuación.

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 (no a la regla misma) que evalúa una inclinación actual *alta* contra una inclinación *baja* medida anteriormente.

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

El nuevo conjunto difuso llamado Hecho (figura 26) permite una atenuación del movimiento del motor en la salida, esto es importante pues disminuye posibles sobresaltos en el sistema ocasionado por vibraciones en el ambiente. De la misma forma que los conjuntos anteriores el Hecho es un conjunto difuso tipo S.

Al fusificar las entradas $x°$ y $y°$ encontrando su valor de pertenencia en los conjuntos Bajo, Medio y Alto respectivos, se encuentra también el valor de pertenencia de dichas entradas en el conjunto difuso Hecho. En otras palabras, el programa fusifica de forma paralela las entradas $x°$ y $y°$ usando el conjunto Hecho, y almacena los valores de pertenencia correspondientes en la variables h_1 (para $x°$) y h_2 (para $y°$).

Es importante mencionar que a cada una de las entradas $x°$ y $y°$ le corresponde un conjunto difuso Hecho exclusivo que, sin embargo, comparte las mismas características y los mismos valores que su homólogo de otra entrada.

La variable h es una premisa que al operarlo de forma indirecta con e y $\hat{e}$ se obtiene el valor numérico del consecuente último de la base de reglas.

Si bien h y el conjunto difuso Hecho son fundamentales para la estabilización del sistema, no tienen una incidencia apreciable en la creación de las reglas de inferencia como se verá en el capítulo 3.2.2.

La comparación entre k_m y h_n se realiza usando el operador intersección como lo muestra la ecuación (47).

$$u_m = min[k_m, h_n] \tag{47}$$

Donde

- u_m es el consecuente último y salida no defusificada del sistema.
- $m = 1, 2, 3 \ldots, 9.$
- $n = 1, 2.$

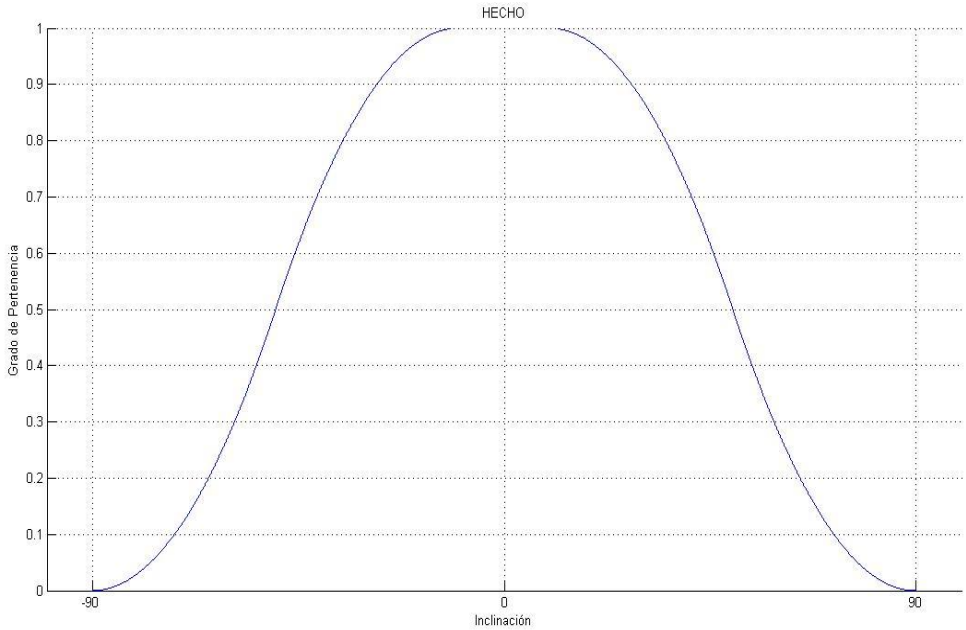


Figura 27. Conjunto Difuso Hecho asociado a la variable h.

Los valores usados para la representación del conjunto difuso Hecho se encuentran en la tabla 3.

Tabla 3. Valores de definición del conjunto difuso Hecho.

Conjuntos Difusos	a	b	a_2	b_2
Hecho	-90	-10	10	-90

La función de defusificación usa las variables de salida u's correspondientes a los servomotores 1 y 2 por separado. Los conjuntos difusos usados para esta función son tipo *singleton*.

La función *singleton* (figura 27) tiene un valor único de 1 para un punto *a* y 0 para todos los demás valores.

Es muy usado en sistemas difusos simples para fijar los conjuntos de las variables de salida. Esta función simplifica las operaciones y usa poca memoria de almacenamiento (dentro del microcontrolador) en comparación con otros tipos de funciones.

Tenemos entonces:

$$A(x) = \begin{cases} 1 & x = a \\ 0 & x \neq a \end{cases} \tag{48}$$

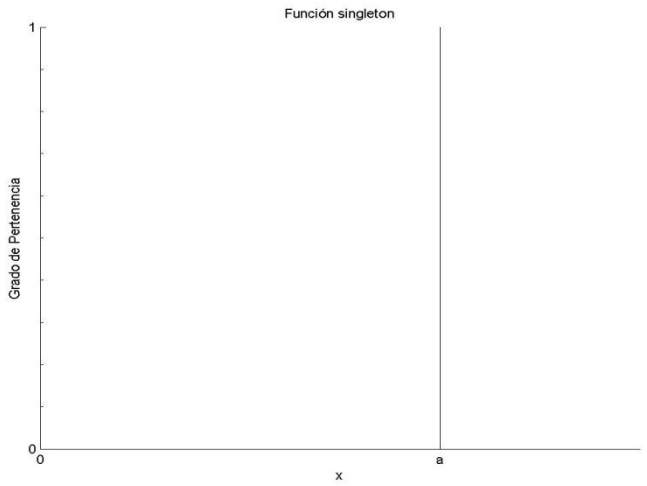


Figura 28. Gráfica de la función de pertenencia *singleton*.

Los conjuntos *singleton* usados para la defusificación se muestran en la figura 28.

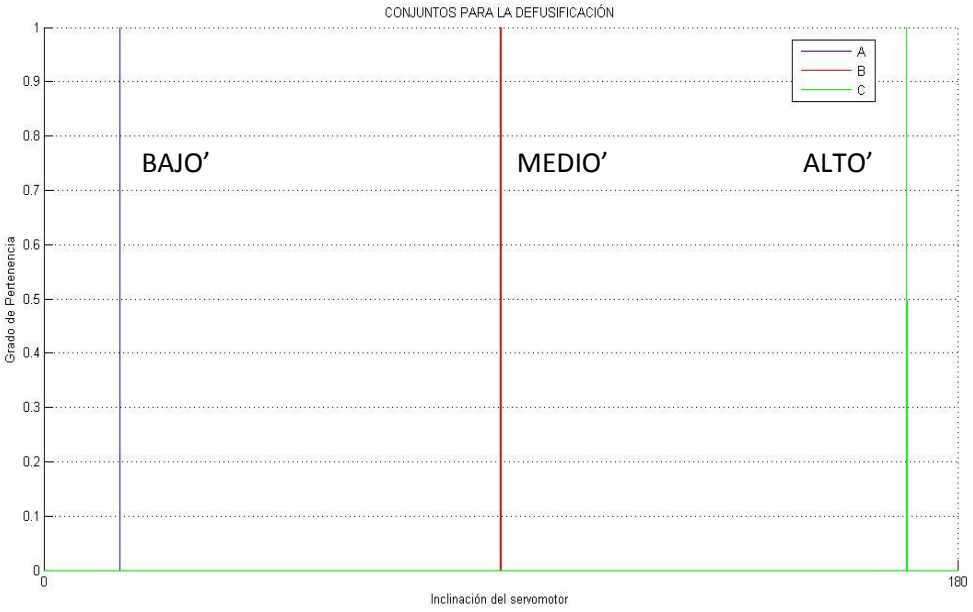


Figura 29. Conjuntos difusos *singleton* asociados a la variable de salida u.

Los valores usados para la representación de los conjuntos difusos *singleton* anteriores se encuentran en la tabla 4.

Tabla 4. Valores de definición de los conjuntos difusos de salida.

Conjuntos Difusos	a
BAJO’	20
MEDIO’	90
ALTO’	160

Los conjuntos difusos singleton (BAJO’, MEDIO’ y ALTO’) asociados a las variables u’s correspondientes al servo 1 tienen los mismos valores que sus homólogas correspondientes al servo 2.

Nótese que el domino de los conjuntos difusos anteriores tiene mayor relación con las salidas de los motores (0° - 180°) que con la propia inclinación de la plataforma.

3.2.2 Reglas de Inferencia e Inferencia Difusa

El consecuente de una regla de inferencia no depende necesariamente de todas las premisas o antecedentes. Esto es importante ya que aunque la variable difusa h es una premisa de las reglas de inferencia, la incidencia que tiene en el consecuente de las mismas es nula.

Lo anterior se desprende de que el conjunto difuso Hecho está formado por una única función de pertenecía, la cual es continua en el espacio de trabajo [-90,90]; no así los conjuntos de e y $\hat{e}$ los cuales necesitan tres funciones de pertenencia para definir cada variable.

La función principal de h es amortizar el traslado de la plataforma permitiendo un movimiento suave y regular; si bien es cierto que influye en el valor numérico del consecuente u, afecta de igual forma a los conjuntos de $\hat{e}$ y e, es por eso que sólo se toman en cuenta estos últimos para precisar la base de reglas.

Las reglas de inferencia se crean a partir del conocimiento de un experto, dado que h es una premisa permanente e “invariable” de las reglas de inferencia, se obvia con fines prácticos al momento de definir el consecuente de las mismas, ya que al experto no le dice nada una variable “amortizadora”. Sin embargo se menciona h en la base de reglas tan solo para no confundir el consecuente último u con la variable k del capítulo 3.2.1.

Las reglas de inferencia tipo mamdani empleadas en el sistema de control difuso se muestra en la tabla 5.

Tabla 5. Reglas de inferencia.

h			
$\hat{e}$ \ e	BAJO	MEDIO	ALTO
BAJO	BAJO'	BAJO'	BAJO'
MEDIO	BAJO'	MEDIO'	MEDIO'
ALTO	BAJO'	MEDIO'	ALTO'

Base de reglas desprendida de la tabla 5:

1. Si e es **BAJO** y $\hat{e}$ es **BAJO** entonces u es **BAJO'**.
2. Si e es **BAJO** y $\hat{e}$ es **MEDIO** entonces u es **BAJO'**.
3. Si e es **BAJO** y $\hat{e}$ es **ALTO** entonces u es **BAJO'**.
4. Si e es **MEDIO** y $\hat{e}$ es **BAJO** entonces u es **BAJO'**.
5. Si e es **MEDIO** y $\hat{e}$ es **MEDIO** entonces u es **MEDIO'**.
6. Si e es **MEDIO** y $\hat{e}$ es **ALTO** entonces u es **MEDIO'**.
7. Si e es **ALTO** y $\hat{e}$ es **BAJO** entonces u es **BAJO'**.
8. Si e es **ALTO** y $\hat{e}$ es **MEDIO** entonces u es **MEDIO'**.
9. Si e es **ALTO** y $\hat{e}$ es **ALTO** entonces u es **ALTO'**.

La razón para considerar tres conjuntos difusos para cada variable difusa (e, $\hat{e}$ y u), es porque permite formular las reglas de inferencia de forma intuitiva y mantener un número manejable de ellas, tanto para el experto como para el programa. Con un número menor de conjuntos difusos no se podría contemplar un número de casos suficientes para la descripción del movimiento de la plataforma. Un número mayor de conjuntos difusos para cada variable llevaría a complicar la declaración de las reglas de inferencia, ya que dificultaría al experto definir un consecuente para todas las combinaciones de los antecedentes. El número de combinaciones aumenta exponencialmente conforme aumenta el número de conjuntos difusos. Además, requeriría de un espacio mayor en la memoria del microcontrolador, al tener que almacenar las variables necesarias para definir cada conjunto.

Idealmente cualquiera de las variables e o $\hat{e}$ debería mantenerse en alguno de sus conjuntos MEDIO, para asegurar la posición horizontal con el menor número de movimientos del motor. Esto se traduce en que la señal de salida PWM tome valores que varíen el giro del motor ±10°; recordando que los servos tienen un rango de movimiento de 0 a 180° y que la señal PWM toma valores de 0 a 255.

Cuando las variables e o $\hat{e}$ se encuentran en un estado diferente al descrito en el párrafo anterior, es decir, cuando ninguna ellas se encuentre ubicada en el conjunto MEDIO, la señal de salida PWM puede tomar valores que varíen el giro del motor hasta por ±30°, lo que es un aumento considerable en la señal con respecto al caso anterior. Por ejemplo, si e se ubicara en el conjunto BAJO y $\hat{e}$ se ubicara en el conjuno ALTO (extremos opuestos de la plataforma), el movimiento del servo se vuelve más pronunciado en comparación con otras combinaciones de las variables. Lo anterior según las pruebas realizadas con la plataforma.

La implicación difusa, después de creada la base de reglas, se desprende considerando los casos que ocurren en la plataforma y su resultado esperado.

3.2.3 Defusificación

Para la elección de un método de defusificación se consideran principalmente los aspectos referentes a la eficiencia computacional y a la facilidad de adaptación; por estas razones, para nuestro sistema basado en arduino la opción más apropiada es el método de centro de áreas (33).

Los valores de u obtenidos por medio de la ecuación (47) y el número total de reglas de inferencia se emplean en la defusicación de la salida.

Todos los consecuentes u independientemente de su valor numérico pueden asociarse con alguno de los tres conjuntos difusos de salida, gracias a la base de reglas.

Al obtener el valor numérico de u con (47), se determina de forma paralela la salida que tiene este consecuente conforme a las reglas de inferencia, es decir, sin importar el resultado de la intersección entre k y h, se comparan dentro de la base de reglas las premisas e y $\hat{e}$ para determinar a qué conjunto difuso de salida pertenece el consecuente u.

Se usa (33) para obtener la salida desfusificada del sistema, tenemos entonces:

$$d_{CA}(u) = \frac{\sum_{i=1}^{9} z_{oi} u_i(z)}{\sum_{i=1}^{9} u_i(z)} \tag{49}$$

Nótese que en (32) se ha cambiado C por u, pues esta última representa la salida de nuestro sistema y n se ha cambiado por 9, esto debido al número total de reglas de inferencia.

El resultado de (49) se envía directamente a los actuadores, es un número entre 20 y 160 aproximadamente, quedando dentro del rango de giro de los servomotores (0°-180°).

3.2.4 Interfaz Gráfica

La interfaz gráfica muestra el comportamiento (variaciones y fluctuaciones) de la inclinación de la plataforma en tiempo real. Además de presentar un desplegado de tres parámetros: inclinación sobre el eje x, inclinación sobre el eje y y el reloj interno de la PC.

Una imagen del entrono de la interfaz se muestra en la figura 29.

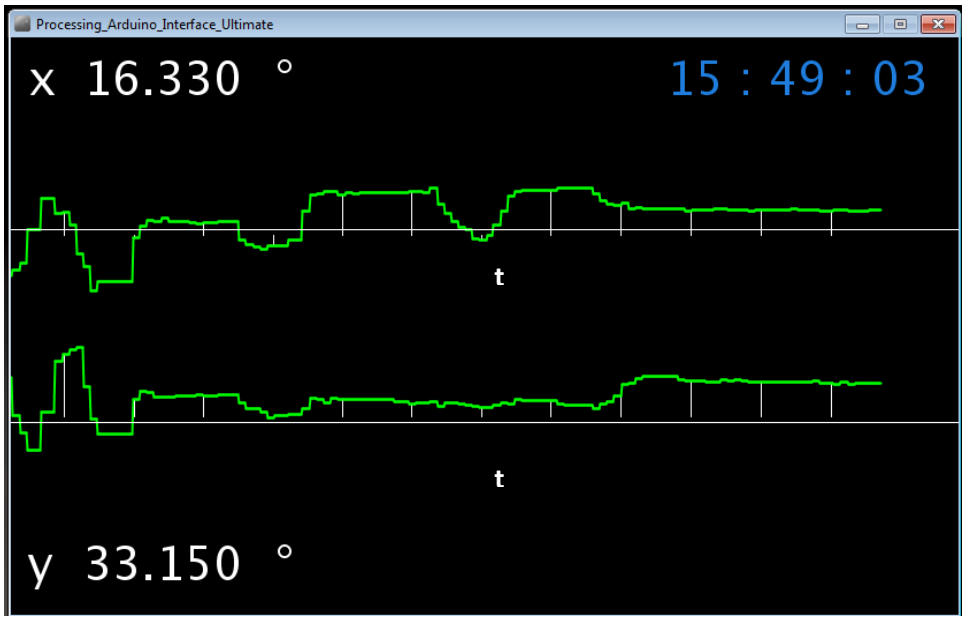


Figura 30. Ejecución de la interfaz gráfica.

Los primeros dos parámetros muestran la inclinación de los ejes en grados, tanto positivos como negativos.

El tercer parámetro, el reloj interno de la PC, se presenta para evidenciar la velocidad de lectura y transmisión de datos del acelerómetro al Arduino y de éste a la PC.

Como se aprecia en la figura 29 el fondo es de color negro y en las esquinas superior e inferior izquierda, así como en la superior derecha se localizan los tres parámetros antes mencionados.

Dentro de la ventana de la interfaz aparecen dos ejes de color blanco que cruzan la pantalla del monitor en forma horizontal; estos son los ejes de referencia a cero, el de la parte superior

perteneciente al eje x y el de la parte inferior perteneciente al eje y. Debajo de cada eje de referencia aparece una “t” que hace referencia al tiempo.

Al arrancar el programa surgen dos líneas de color verde (una por cada eje) que se desplazan de forma paralela a los ejes de referencia, dejando un rastro que señaliza la inclinación de la plataforma. Cada segundo aparece una línea blanca perpendicular a los ejes de referencia, dicha línea es proporcional a la inclinación mostrada por las líneas de color verde y se desplaza sobre la pantalla conforme a éstas.

En las figuras 30 y 31 se señalizan las partes que conforman la interfaz gráfica.

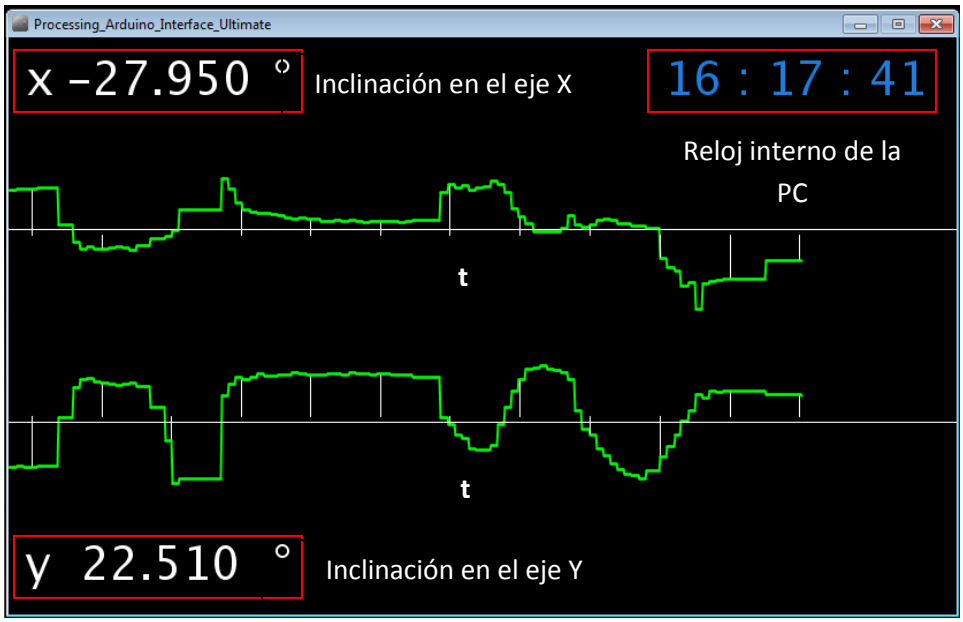


Figura 31. Parámetros de la interfaz.

Desde el aspecto de comunicación con Arduino, la interfaz únicamente recibe los datos de inclinación ya procesados del acelerómetro; con éstos imprime las gráficas que se ven en las figuras 30 y 31 e imprime también el valor numérico de la inclinación en el tiempo en que se ejecuta el programa en la placa Arduino. Se remarca que el rol de la PC en el control de la plataforma es nulo ya que sólo ejecuta la interfaz gráfica y la presenta a través de la pantalla del monitor.

Si se decidiera prescindir de la interfaz gráfica por ejemplo para hacer el sistema portable, no tendría que reescribirse el programa del microcontrolador; pues sólo en un par de líneas del código se hace el envío de datos a la PC, y no afecta desde el punto de vista práctico al control de la plataforma.

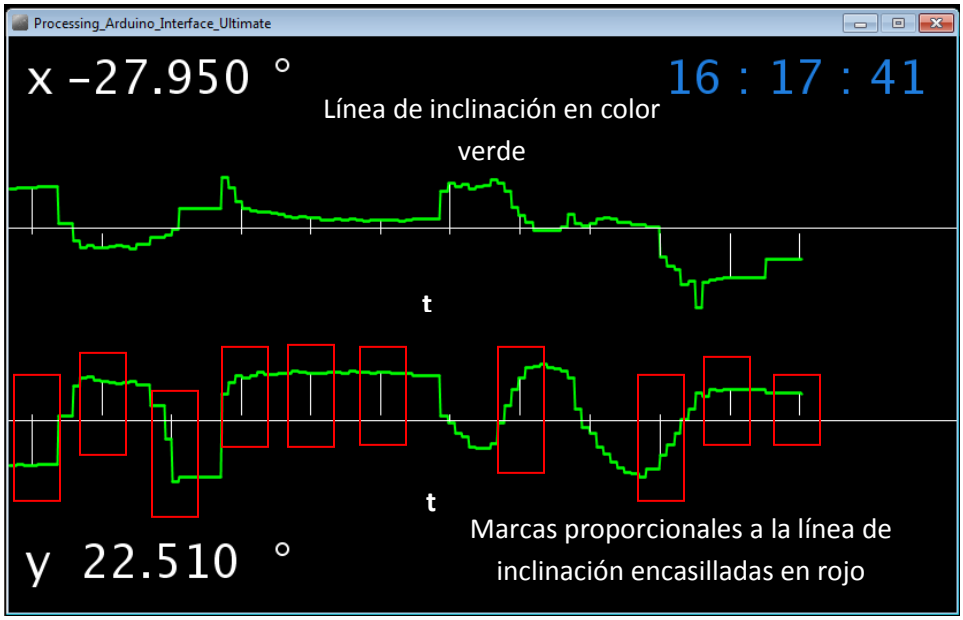


Figura 32. Marcas de tiempo y línea de inclinación en la interfaz.

El código de la interfaz puede verse en el apéndice C.

3.2.5 Resultados

En esta sección se muestran los resultados obtenidos del controlador difuso a través de la interfaz gráfica, una vez que el sistema de control y el montaje de la plataforma fueron terminados.

Se logró que los servomotores tomen la posición deseada, usando los datos del acelerómetro procesados en el programa basado en lógica difusa y conjuntos difusos implementado en Arduino.

Para monitorear el comportamiento del acelerómetro se programó una interfaz gráfica que recibe los valores de inclinación en los ejes x y y de la plataforma.

Se implementó el acelerómetro en la plataforma como un dispositivo que genera datos de entrada al control; los servomotores como dispositivos que reciben datos de salida de la placa Arduino, para obtener la posición horizontal de dicha plataforma.

En la figura 32 se muestra la plataforma en operación estabilizando su base superior.

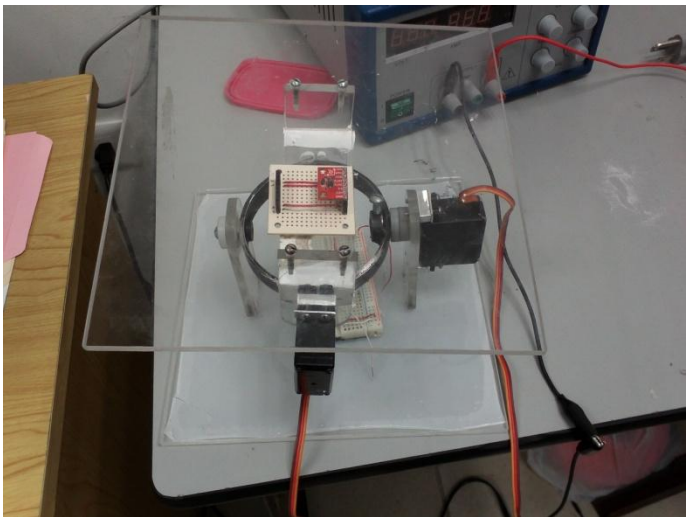


Figura 33. Montaje de la plataforma con acelerómetro y actuadores.

La plataforma en funcionamiento se mantuvo dentro de los límites permitidos para la estabilización, y respondió a los cambios de inclinación de forma casi instantánea.

En la figura 33 se muestra en detalle la unión del eje del servomotor con el anillo central de la plataforma.

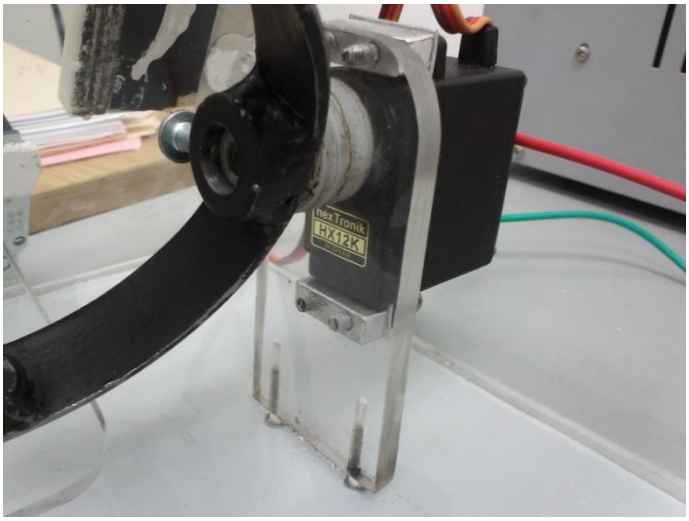


Figura 34. Detalle de una de las articulaciones de la plataforma.

CONCLUSIONES

Este trabajo se considera como un intento de abordar el estudio del control de una plataforma paralela en el marco de la lógica difusa, visto como un enfoque alternativo a las técnicas de control moderno. La aportación principal de este trabajo es la elaboración de un programa de control en el problema de la cinemática inversa de un robot paralelo de tres grados de libertad, usando teoría de conjuntos difusos, teoría de control difuso y técnicas de programación. En el estudio del control de la plataforma se analizaron los casos en los que el movimiento del prototipo es planar (rotación sobre los ejes x o y) y los casos en que el movimiento es espacial (rotación simultanea de los ejes x y y). Se consideró la aplicación de la lógica difusa, en primera instancia al control del movimiento planar, con cambio de posición en la base inferior del prototipo, recordando que el punto de referencia es la superficie de la Tierra. Posteriormente se consideraron los movimientos no planares. En el caso del movimiento planar, se obtuvieron regiones de inoperatividad debido a la posición de los motores en la plataforma, las cuales quedan fuera del espacio de trabajo propuesto en el capítulo 3. La simulación en la interfaz gráfica muestra que se puede alcanzar un grado de estabilidad aceptable, mediante la modificación de algunos de los valores de los conjuntos difusos, sin cambiar su tipo. Finalmente, también se consideró el problema donde el prototipo presentaba oscilación en su movimiento y no se estabilizaba, por lo que se añadió una línea al código del programa que sólo procesa los datos del acelerómetro que estén dentro de ciertas cotas.

El hecho de adoptar el control difuso en oposición al control clásico (basado en modelo) se basa en el hecho que el primero presenta tolerancia a la imprecisión y la habilidad de tomar decisiones bajo cierta incertidumbre. Además puede abordar problemas de sistemas que no tienen un modelo matemático estándar o lineal, de manera que puedan resolverse usando un razonamiento similar al *humano*.

Sin embargo cuando la dinámica de los sistemas o procesos a controlar es lineal o tiene un modelo matemático definido, el control clásico tiene la capacidad de responder a variaciones o interferencias en el sistema, ocasionadas por oscilaciones, retardos en el tiempo de muestreo etc., con diversos mecanismos como el PID con un alto porcentaje de eficacia y rapidez.

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, es siempre preferible trabajar con él dentro de las pautas del control clásico. Esto debido a su exactitud y precisión.

Cuando los sistemas requieran de ajustarse a condiciones cambiantes difíciles de predecir, no se cuente con un modelo matemático o sea igualmente difícil de obtener, el control difuso es una alternativa viable.

En cuanto a la selección del número de reglas de inferencia y a los consecuentes que arrojan las mismas en sus comparaciones, no se modificaron estos parámetros aún cuando los resultados no fueron los esperados al inicio de las pruebas. Esto se debe a que es más sencillo modificar los conjuntos difusos asociados a las variables e y $\hat{e}$, que añadir o eliminar reglas de inferencia o que incluso modificar sus consecuentes.

Al modificar los conjuntos difusos de entrada (Bajo, Medio y Alto) no se habla de cambiar su tipo, sino sólo de modificar sus proporciones, es decir, hacerlos más estrechos con formas menos anchas y evidenciando más las intersecciones entre los mismos. Ya que los conjuntos difusos están declarados como funciones dentro del programa, es relativamente fácil manipular los valores de entrada de dichas funciones y obtener variedad de resultados.

PANORAMA FUTURO

Las perspectivas para trabajos futuros se basan principalmente en el estudio del control de plataformas de más de tres grados de libertad, con variación o intercambio de los sensores de movimiento.

Otra línea de trabajo de interés es la reducción de oscilaciones y ruido en las mediciones. El ruido afecta principalmente a los estimados de fallas, por lo que se pretende proponer filtros físicos y por software para atenuarlo. También se considera de forma importante, la aplicación directa del control de la plataforma en otros sistemas mecánicos y/o electromecánicos con objeto de implementarlos en equipos industriales.

Apéndice A

Configuración del acelerómetro en la placa Arduino

Este apéndice aborda los detalles para el uso efectivo del acelerómetro adxl345 en la plataforma de este trabajo de tesis. Sin más preámbulos presentamos la información.

Al configurar el acelerómetro se designa la dirección del dispositivo con que se identificará ante el Arduino, esta dirección es 0x53.

Después debe escribirse sobre los registros del sensor, esto se hace colocando datos en cada byte de los registros (*single-byte write*) siguiendo los pasos descritos a continuación:

- Iniciar transmisión al dispositivo usando la dirección 0x53.
- Escribir la dirección del registro deseado.
- Escribir el dato que se quiera colocar sobre el registro anterior.
- Finalizar la transmisión.

Los pasos anteriores se resumen en la figura 34 (hoja de datos ADXL345 pág. 18).

SINGLE-BYTE WRITE									
MASTER	START		SLAVE ADDRESS + WRITE		REGISTER ADDRESS		DATA		STOP
SLAVE				ACK		ACK		ACK	

Figura 35. I^{2}C Device adaressing.

El código en arduino que escribirá un valor sobre un registro específico del dispositivo se expresa en la función *writeTo*.

```
void writeTo(byte registro, byte val)

{
  Wire.beginTransmission(0x53);
  Wire.write(registro);
  Wire.write(val);
  Wire.endTransmission();
}
```

En el código anterior la función *writeTo()* escribe un valor sobre un registro. La función *Wire.beginTransmission()* inicia la transmisión al dispositivo con la dirección 0x53, después *Wire.write()* coloca la dirección del registro deseado además de escribir el valor requerido sobre el registro anterior; *Wire.endTransmission()* finaliza la transmisión.

Para completar la comunicación acelerómetro-arduino es necesaria también una función de lectura. La función de lectura es un poco más larga, aunque usa conceptos similares.

Esta vez será necesario usar una función que permita leer múltiples bytes al mismo tiempo (*multiple-byte read*), esto es así por la exigencia de leer datos de 6 bytes del sensor (2 bytes por cada eje); de lo contrario el valor de un eje variaría mientras se leen los otros.

Para efectuar la lectura de múltiples bytes se singuen los pasos descritos a continuación:

- Iniciar transmisión al dispositivo usando la dirección 0x53.
- Escribir la dirección del registro de *inicio*, esto es, la dirección por la que se quiera comenzar a leer.
- Iniciar transmisión al dispositivo de nueva cuenta usando la dirección 0x53.
- Leer los bytes deseados.
- Finalizar la transmisión.

Los pasos anteriores se resumen en la figura 35 (hoja de datos ADXL345 pág. 18).

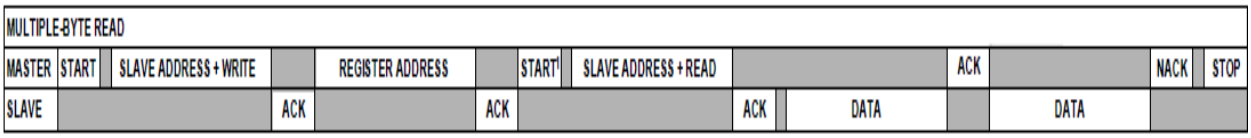


Figura 36. I²C Device addressing.

El código en arduino que escribirá un valor sobre un registro específico del dispositivo se expresa en la función *readFrom*:

```
void readFrom(byte 68omunicac, int num, byte _buff[])

{
  Wire.beginTransmission(0x53);

  Wire.write(registro);

  Wire.endTransmission();

  Wire.beginTransmission(0x53);

  Wire.requestFrom(0x53, num);

  Int i = 0;

While (Wire.available())

  {
    _buff[i] = Wire.read();
    I++;
  }
  Wire.endTransmission();
}
```

En el código anterior la función *readFrom()* lee y guarda los datos de 6 bytes del sensor. La *función Wire.beginTransmission()* inicia la transmisión al dispositivo con la dirección 0x53, después *Wire.write()* envía la dirección del registro de inicio y *Wire.endTransmission()* finaliza la transmisión. *Wire.beginTransmission()* inicia de nueva cuenta la transmisión a la dirección 0x53; en *Wire.requestFrom()* la variable "num" solicita 6 bytes de lectura (2 bytes por casa eje) al dispositivo; una subrutina se ejecuta siempre que haya 6 bytes de lectura disponibles, estos se guardan en la matriz "_buff"; *Wire.endTransmission()* finaliza la transmisión.

Después de presentar las funciones *writeTo* y *readFrom* es necesario describir brevemente algunas de las funciones de la librería de arduino *Wire* que se usan aquí.

Wire: Esta librería permite comunicar arduino con dispositivos I^2C. En la placa Arduino UNO, SDA (línea de datos) está en el pin analógico 4, y SCL (línea de reloj) está en el pin analógico 5.

Begin(): Inicializa la librería Wire y configura el bus I^2C como maestro o esclavo.

beginTransmission(address): Comienza una transmisión a un dispositivo I2C esclavo con la dirección dada (address). Posteriormente, prepara los bytes a transmitir con la función *send()* y los transmite llamando a la función *endTransmission()*.

endTransmission(): Finaliza una transmisión a un esclavo que fue empezada por *beginTransmission()*.

Read(): Lee un byte que se transmite de un dispositivo esclavo de un maestro después de llamar a *requestFrom()* o se transmite de maestro a un esclavo.

Write(value): Escribe los datos de un dispositivo esclavo en respuesta a una petición de un maestro, o colas de bytes para la transmisión de un maestro a un dispositivo esclavo (se coloca entre las llamadas a *beginTransmission()* y *endTransmission()*).

requestFrom(address, quantity): Solicita bytes de otro dispositivo. Los bytes pueden ser recibidos con la función *available()*.

Available(): Devuelve el numero de bytes disponibles para recuperar con *receive()*. Debería ser llamada por un maestro después de llamar a *requestFrom()*.

Una vez contempladas las funciones esenciales, se describe el resto del programa que hace la comunicación con el sensor.

```
#include <Wire.h>
#include <mat.h>

#define DEVICE (0x53)

Byte _buff[6];

Char POWER_CTL = 0x2D;
```

```
Char DATA_FORMAT = 0x31;
char DATAX0 = 0x32;
char DATAX1 = 0x33;
char DATAY0 = 0x34;
char DATAY1 = 0x35;
char DATAZ0 = 0x36;
char DATAZ1 = 0x37;
```

En el código anterior la librería *Wire*, comunica al arduino con el acelerómetro por I^2C. La librería de matemáticas *mat* es necesaria para emplear funciones trigonométricas.

En *"DEVICE ()"* se define la dirección con la cual será identificado el acelerómetro en el arduino.

Se crea una matriz *_buff]* de 6 bytes que almacenará los datos entregados por el acelerómetro (2 bytes por cada eje).

El resto de las declaraciones refiere a las direcciones de los registros propios del acelerómetro, dichos registros pueden verse en la hoja de datos del ADXL345.

La siguiente parte del programa corresponde a las funciones *setup* y *loop* de arduino, la primera coloca los valores de inicio de ciertas variables y funciones (sólo se ejecuta al inicio del progama), en tanto que en la segunda se encuentra el programa principal (es en un ciclo infinito).

```
void setup()

{
  Wire.begin();

 Serial.begin(57000);

 Serial.print("init");

  writeTo(DATA_FORMAT, 0x00);

  writeTo(POWER_CTL, 0x08);
}


void loop()
{
  readAccel();
  Delay(500);
}
```

En la función *setup()* se inicia formalmente el programa, en ella se colocan la mayoría de los valores de inicio del programa; esta función sólo se ejecuta una vez. Dentro del *setup()* la función *Wire.begin()* inicia la comunicación con el acelerómetro por I^2C; *Serial.begin()* inicia la comunicación serial con la PC a 57000 baudios, aunque sólo es importante para la interfaz, luego *Serial.print* imprime la cadena de caracteres *init* para saber que la transmisión con la PC está lista.

Hay dos llamados la función *writeTo()*, el primero es para colocar el acelerómetro en el rango de +/-2g escribiendo 0x00 en el registro *DATA_FORMAT*; el segundo coloca al acelerómetro en modo medición escribiendo 0x08 en el registro *POWER_CTL*.

Dentro de la función *loop()* la cual es un ciclo infinito, se encuentra el programa principal.

La función *readAccel* transforma los valores dados en bits por el acelerómetro en ángulos medidos en grados.

```
Void readAccel()

{
  uint8_t howManyBytesToRead = 6;

  readFrom( DATAX0, howManyBytesToRead, _buff);


  Int x = (((int)_buff[1]) << 8) | _buff[0];
  int y = (((int)_buff[3]) << 8) | _buff[2];
  int z = (((int)_buff[5]) << 8) | _buff[4];

Double x1 = x ;
double y1 = y ;

x1 = ((x1)/256);
y1 = ((y1)/256);

    incli_rx = asin(x1);
    incli_ry = asin(y1);


    incli_gx = ((incli_rx * 180)/3.1416);
    incli_gy = ((incli_ry * 180)/3.1416);

   Serial.print("X: ");
   Serial.print( incli_gx );
   Serial.print("\t");

   Serial.print("Y: ");
   Serial.println( incli_gy );
   Serial.print(" ");
}
```

La función *readAccel()* lee la aceleración del ADXL345. Cada eje tiene una resolución de salida de 10 bits o 2 bytes ie (integer endian). Se convierten los 2 bytes de datos de cada eje en un entero. Los enteros en arduino almacenan un valor de 16 bits o 2 bytes.

Como los valores de x, y, z son enteros es preferible trabajarlos como punto flotante doble para mejorar la precisión.

Se prescinde de los valores del eje z, pues no son necesarios para el control de la plataforma.

Como la resolución de salida es de 10 bits se tendrán 1024 valores, que irán de -512 (-2g) hasta +512 (2g). La sensibilidad en +/-2g es de 256 LBS/g, lo que significa que cada lectura de 256 bits será igual a 1g.

Se convierten los valores de bits a g's y se calcula la inclinación en radianes en los ejes *x* y *y*; se convierten los radianes a grados y se muestran los resultados en pantalla.

Apéndice B

Configuración de los servomotores en la placa Arduino

Este apéndice aborda los detalles para el uso efectivo del acelerómetro adxl345 en la plataforma de este trabajo de tesis.

El siguiente programa es uno de los ejemplos que ofrece Arduino en su página web para hacer un test de prueba de los servomotores.

```
#include <Servo.h>

Servo myservo;

int pos = 0;

void setup()
{
  myservo.attach(9);
}


void loop()
{
  for(pos = 0; pos < 180; pos += 1)
  {

    myservo.write(pos);

   delay(15);
  }
  for(pos = 180; pos>=1; pos-=1)

  {
   myservo.write(pos);

   delay(15);                        }
}
```

En el programa anterior se incluye la librería *Servo* que contiene las funciones necesarias para controlar los actuadores del sistema.

Se crea un objeto servo para el control de un servomotor. Cada servomotor necesita su propio objeto, así para un segundo motor puede usarse la sentencia "Servo myservo2". La variable entera *pos* almacena la posición del servomotor.

Dentro del *setup* la sentencia *myservo.attach()* asigna al servomotor el pin 9 de la placa arduino, este pin tiene salida pwm. Dentro de la función *loop* se encuentra un ciclo *for* que realiza un barrido de 0° a 180° en el giro del motor en movimientos de 1°.

La función *myservo.write()* indica al servo la posición a la cual ir, dicha posición está almacenada en la variable *pos*. Cuando el giro del servo llega a 180°, se inicia un nuevo ciclo *for* que invierte la dirección del giro, es decir, el motor ahora se mueve de 0° a 180°.

En este ejemplo se hace un *barrido* de todas las posiciones que puede tomar un servo, de 0 a 180 grados, haciendo uso de la función mencionada anteriormente. Si hay buena compatibilidad entre la norma que tiene el servo y el control que hace la función *write(),* el eje del motor debería situarse completamente a la izquierda cuando el valor sea 0, totalmente a la derecha cuando sea 180 y en medio cuando sea 90. En caso de no haber buena compatibilidad, puede escribirse la duración de los pulsos de manera manual con la función *writeMicroseconds()* de Arduino.

A manera de resumen se describen brevemente algunas de las funciones de la librería *Servo*.

attach(pin): Asocia la variable Servo a un pin.

***write(ángulo)*:** Escribe un valor en el servo, controlando el eje en consecuencia. En un servo estándar ajustará el ángulo del eje (en grados), moviendo el eje hasta la orientación deseada.

***writeMicroseconds()*:** Escribe un valor en microsegundos (uS) en el servo, controlando el eje en consecuencia. En un servo estándar seleccionará el ángulo del eje. En servos estándar un valor de parámetro de 1000 situará el eje completamente a la izquierda, 2000 totalmente a la derecha y 1500 en el medio.

Apéndice C

Código de la Interfaz Gráfica

```
import processing.serial.*;   //Librería para la comunicación serial
int[] PC_Time = new int[3];  // Variable para registrar la hora
String curr_time;            // Variable para la obtención del tiempo

int retardo;  //Variable usada en la impresión de las líneas verticales sobre los ejes de referencia

Serial myPort;  // Crear objeto de la clase Serial
String val;      // La vairable val almacena los datos recibidos del puerto serie

int posicion=0; // Posición horizontal de la gráfica

float[] ejes;  // Matriz de punto flotante que almacena la cadena de caracteres de "val"
float x, y;    // Varibles que almacenan el valor de la inclinación
```

```
float xAnt=height/6;     // Variables auxiliares para realizar las gráficas
float yAnt=height/6;    // de inclianción

//Función de inicialización
void setup() {

 size(displayWidth -200, displayHeight-100);  //Tamaño de la ventana en donde corre el programa
 background(0);      // Fondo negro
 smooth();            // Calidad del dibujo en "fino"

  String portName = Serial.list()[0]; //Carga el pueto serial que esté en uso por el Arduino
  myPort = new Serial(this, portName, 9600); //Inicia comunicación serial
  myPort.clear(); //Borra cualquier dato recibido antes de lo esperado
  myPort.bufferUntil('\n'); // Coloca un tope al almacenamiento de datos recibidos

  retardo=millis(); // Variable que almacena milisegundos
}

//Bucle principal del programa
void draw(){

 //Impresión de coordenas en X

  stroke(0);//Color del contorno del rectángulo
  fill(0);//Color del relleno del rectángulo
  rect(0, 0, 300, 50);  // Rectángulo negro

  textSize(40);// Tamaño de la fuente
  fill(#FFFFFF);//Color de fuente

  text("x",15, 50); //Posición en la pantalla de caracteres y datos de la inclinación en X
  text("°",230, 50);
  text( x, 50, 50);

 //Impresión de coordenas en Y

  fill(0);//Color del relleno del rectángulo
  rect(0, 430, 300, 70);

  textSize(40);  //Tamaño de la fuente
  fill(#FFFFFF);//Color de fuente

  text("y",15, 470); //Posición en la pantalla de caracteres y datos de la inclinación en Y
  text("°",230, 470);
  text( y, 50, 470);
```

```
//Impresión del signo de señalización del tiempo "t"

  textSize(20);//Tamaño de fuente
  fill(#FFFFFF);//Color de fuente

  text("t",370, 200);//Posición en la pantalla del carácter “t”
  text("t",370, 370);

//Hora del reloj interno de la PC

  textSize(40);//Tamaño de fuente

  curr_time = PC_Time();  // Cargamos en la variable el valor obtenido al llamar la función
                          //  "PC_Time()"

  fill(0);//Color del relleno del rectángulo
  rect(400, 0, 400, 50);

   // Escritura de la hora

  fill(31, 125, 222);  // Color de fuente
  text(curr_time, 570, 50);  //Se imprime y se posiciona el contenido de la variable

//Impresión de las lineas verticales que se dibujan cada segundo

  if (millis() - retardo > 1000){//Cada 1000ms (1 segundo) aparece una línea vertical

    strokeWeight(1); // Grosor de las lineas
    stroke(255); // Lineas verticales en color blanco

    line(posicion, height/3-xAnt, posicion, height/3+5);//Posición de las líneas
    line(posicion, height*2/3-yAnt, posicion, height*2/3-5);// a lo largo de la pantalla

    retardo=millis();

  }

//Ejes de referencia

  strokeWeight(1);
  stroke(255);
  line(0, height/3, width, height/3); //Eje de referencia de X
  line(0, height*2/3, width, height*2/3);//Eje de referencia de Y
```

```
//Gráficas de la inclinación

 strokeWeight(2);
 stroke(0, 255, 0);
 line(posicion, height/3-xAnt, posicion+1, height/3-x);//Inclinación en X
 line(posicion, height*2/3-yAnt, posicion+1, height*2/3-y);//Inclinación en Y

 xAnt=x;//Variables auxiliares para realizar las gráficas de inclinación
 yAnt=y;

//Al cruzar toda las gráfica toda la pantalla, se borran las gráficas y se  inicia nuevamente el ciclo

  if(posicion==width)
  {
    posicion=0;
    background(0);
  }
  else{
    posicion++;
   }
}
//Función "evento serial"
//Recibe los datos del Arduino por el puerto serial

void serialEvent(Serial myPort){
if ( myPort.available() > 0){  // Si el dato está disponible ejecuta
       val = myPort.readString(); // Leer y almacenar en la cadena de caracteres "val"
         if(val != null){ //Si el dato es nulo ejecuta
                  val = trim(val);
                  float[] ejes = float(splitTokens(val));
                      if (ejes.length >=2){//Asigna cada dato de “val” a “x”  y  ”y”
                           x=ejes[0];
                           y=ejes[1];
                                      }
                             }
                  }
        }

// Función para obtener la hora de la PC

String PC_Time(){
PC_Time[2] = second();  // Valores de 0 a 59
PC_Time[1] = minute();  // Valores de 0 a 59
PC_Time[0] = hour();    // Valores de 0 a 23
return join(nf(PC_Time, 2), " : ");  // Devolvemos la hora completa indicando que entre cada
                                     //  número se impriman ":"
}
```

BIBLIOGRAFÍA

[Albertos 2004] Albertos P., y Sala A., (2004). El control borroso: una metodología integradora. RI-AII, Vol. 1, No. 2, pp. 22-31. http://personales.upv.es/asala/publics/papers/R17riai043fuzzy integradoraPreprint.pdf. 2/11/2012.

[Alzate 2007] Alzate Gómez A., López López A., Restrepo Patiño C. (2007). Control difuso de una plataforma móvil para el seguimiento de trayectorias. Scientia Et Tchnica, Vol. XIII, No. 35, pp. 169-174.

[D.Negri 2006] D.Negri C. E., y De Vito E. L., (2006). Introducción al razonamiento aproximado: lógica difusa. Revista Argentina de Medicina Respiratoria, No. 4, pp. 126-136. http://www.ramr.org.ar/archivos/numero/ano_6_3_dic_2006/mere3_6.pdf

[Barrientos et al 2007] Barrientos Antonio et al. Fundamentos de robótica. Mc Graw Hill. España.

[Du 2006] Du K. L., y Swamy M. N. S., (2006). Neural Networks in a Softcomputing Framework. Springer Verlag. Alemania.

[Gómez-Estern 2002] Gómez-Estern A. F., (2002). Control de sistemas no lineales basado en la estructura hamiltoniana. Tesis doctoral, Universidad de Sevilla, España. Disponible en: http://www.esi2.us.es/~fabio/tesis.pdf. 4/11/2012.

[Jiménez 2007] Jiménez M. R., (2007). Diseño de un controlador lógico difuso, aplicado al control de posición de un servomotor de C.D., usando un algoritmo genético. Tesis de Maestría. Universidad Veracruzana. Disponible en: http://cdigital.uv.mx/bitstream/123456789/1736/1/Tesis-Jimenez-Madrigal-Rene.pdf 4/11/2012.

[Klir y Yuan 1995] Fuzzy sets and fuzzy logic. Theory and applications. Prentice Hall PTR 1995

[Mahfouf 2001] Mahfouf M., Abbod M. F., Linkens D. A., (2001). A Survey of Fuzzy Logic Monitoring and Control Utilization in Medicine Articial Intelligence in Medicine, Vol. 21, pp. 27-42.

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

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

[Pratihar 2008] Pratihar, D. K., (2008). Soft Computing. Alfa Science International. Reino Unido.

[Rairán 2006] Rairán A. J. D., y Urrego R. L. S., (2006). Implementación de un controlador difuso para la regulación de posición de un cilindro hidráulico lineal. Tecnura, Vol. 10, No. 19, pp. 18-28.

[Rubio 2010] Rubio F. R., Ortega M. G. y Gordillo F., (2010). Control de Posición e Inercial de Plataforma de Dos Grados de Libertad. RIAII, Vol. 7, No. 4, pp. 65-73. http://recyt.fecyt.es/index.php/RIAII/article/view/ RIAI.2010.04.09/7509. 4/11/2012.

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

[Zadeh 1985] Zadeh L.A., .Fuzzy Sets, Information and Control, Vol. 8, No. 3 pp. 338-353, june 1965.