SISTEMA DE CONTROL PARA UN ROBOT SELF-BALANCING

DIEGO MAURICIO MUÑOZ GARCIA

Universidad Pedagógica Nacional
Facultad de Ciencia y Tecnología
Departamento de Tecnología
Licenciatura en Electrónica
Bogotá D.C
2013

RESUMEN ANALÍTICO EN EDUCACIÓN - RAE

1. Información General	
Tipo de documento	Monografía – Trabajo de grado.
Acceso al documento	Universidad Pedagógica Nacional. Biblioteca Central
Titulo del documento	SISTEMA DE CONTROL PARA UN ROBOT SELF-BALANCING.
Autor(es)	Muñoz García Diego Mauricio.
Director	Mg Alberto Morales.
Publicación	Bogotá. Universidad pedagógica Nacional. 2014. 51 p.
Unidad Patrocinante	Universidad Pedagógica Nacional
Palabras Claves	Control difuso, Estabilidad, Equilibrio, potencia.

2. Descripción

El desarrollo y creación de prototipos que permitan la estabilidad de sistemas sobre una superficie, péndulo invertido, es una propuesta que se ha ido desarrollando en la universidad pedagógica como herramienta para promover en el estudiante el interés por nuevas tecnologías en el área de control. Por esta razón el presente trabajo busca incentivar la investigación en varias áreas de la electrónica apoyándose en los sistemas físicos y los algoritmos de control sobre movilidad y equilibrio que se desarrollan en estos estudios.

Para lograr el objetivo propuesto, principalmente se identifican los temas fundamentales que permitan el desarrollo del sistema de control con el fin de profundizar en los conceptos que cursan los estudiantes de la licenciatura en electrónica como Potencia, control tradicional y control difuso.

Para el diseño del sistema de control, inicialmente se construye la estructura mecánica que está enfocada en el diseño de péndulo invertido, buscando corregir el método de movilización sobre dos ruedas. Para esto se incorpora una tercera rueda y se establece una separación entre ellas de 60°, permitiendo desplazar la estructura sobre la esfera sin necesidad de frenar. Para corregir la posición se diseña el circuito que permite establecer el ángulo de inclinación de la estructura

mecánica, el circuito de potencia para el funcionamiento de los motores, el control de velocidad y el control difuso que establece las reglas de comportamiento consecuentes a las perturbaciones de la posición de la estructura para así mantenerla en un ángulo de 90° sobre la superficie terrestre.

3. Fuentes

GARCIA BREIJO EDUARDO, Compilador c ccs y simulador PROTEUS para Microcontroladores PIC. México, junio de 2008.

MIGUÉLEZ GARCÍA RUBÉN. Estudio diseño y desarrollo de una aplicación de tiempo real y de un simulador para su comprobación: péndulo invertido. Disponible en internet: http://marte.unican.es/projects/xrubenx/pendulum.pdf

MIGUEL Varga Juan y GHENZI Néstor. Introducción a la dinámica no lineal: péndulo invertido forzado. Disponible en internet: http://www.ib.cnea.gov.ar/~experim2/informes2006/info16.pdf
KUMAGAI Masaaki y GAKUIN Tohoku University. A Robot That Balances on a Ball (Un robot que balancea sobre una pelota) Disponible en internet: http://spectrum.ieee.org/automaton/robotics/robotics-software/042910-a-robot-that-balances-on-a-ball

POZO ESPIN, David Fernando, Diseño y construcción de una plataforma didáctica para medir ángulos de inclinación usando sensores inerciales como acelerómetro y giroscopio. Quito, febrero 2010

Mínguez, G (2009). Integración Kalman de sensores inerciales INS con GPS en un UAV. (Tesis de Pregrado). Universitat Politécnica de Catalunya. España.

Wang, Rizos y Li. Application of a Sigma-point Kalman filter for alignment of MEMS-IMU (artículo). University of New South Wales (sydney). Australia.

4. Contenidos

El presente documento consta de tres secciones en los que se presenta el desarrollo de toda la implementación realizada. En sección I se expone el contexto investigativo y los objetivos que se propusieron para dar respuesta y solución a la problemática de control planteada.

En la sección II se expone la metodología utilizada para el desarrollo de la construcción e implementación de la planta y el modelo de control y se describe minuciosamente la ruta de modelamiento difuso sugerida.

Finalmente en la sección III se presentan los análisis de los resultados obtenidos durante la implementación de la metodología difusa y las reflexiones finales que deja este trabajo de desarrollo.

5. Metodología

Como metodología a implementar se utiliza RUP (Rational Unified Process) la cual es una metodología de desarrollo de software que sin embargo brinda herramientas y estructuras útiles para el desarrollo e implementación del proyecto.

Tiene cuatro fases de implementación. La fase de inicio está orientada al estudio, diseño y fabricación de los circuitos encargados para la toma de datos de los sensores y la estructura mecánica en donde interactuaran los motores DC con la superficie esférica. Fase de diseño, en esta fase se hace un énfasis en el diseño del modelo matemático del sistema dinámico. Al igual que el diseño de la simulación del comportamiento del modelo. Fase de implementación en esta fase se va a implementar el sistema de control que permitirá realizar las pruebas de estabilización al sistema esfera-barra. Fase de transición, en esta fase busca garantizar que se tiene un prototipo preparado para su entrega final.

El RUP es un producto de Rational (IBM). Se caracteriza por ser iterativo e incremental, estar centrado en la arquitectura y guiado por los casos de uso. Incluye artefactos (que son los productos tangibles del proceso) y roles (papel que desempeña una persona en un determinado momento, una persona puede desempeñar distintos roles a lo largo del proceso).
La estructura dinámica de RUP es la que permite que éste sea un proceso de desarrollo fundamentalmente iterativo.

6. Conclusiones

Se diseñó y construyó el sistema de estabilidad “Self-Balancing sobre una esfera” cumpliendo con los objetivos propuestos y los módulos básicos para su funcionamiento. Su aplicación está orientada al desarrollo de controladores difusos que permitan estabilizar plataformas no lineales que se plantean como proyectos en algunos espacios académicos de la licenciatura en electrónica. Este prototipo permite la simulación de distintos sistemas de controladores difusos, gracias a su versatilidad en la parametrización y sintonización de las variables que lo rigen.
Un aspecto importante que se obtuvo como resultado, es el comportamiento del sensor IMU, puesto que es el sensor que parametriza las variables de entrada al controlador difuso y la vez es el que determina la estabilidad de todo el sistema. Por lo que se dio a conocer la importancia de la corrección de los ángulos por medio del algoritmo del filtro kalman y se brindan herramientas que permiten su análisis y diseño, para realizar posteriores trabajos sobre estabilización en donde aplique estos parámetros de comportamiento.
Los componentes mecánicos juegan un papel fundamental en los sistemas self-balancing debido a que la estabilización de la plataforma depende del movimiento y reacción adecuado de cada elemento que interactúa con el sistema, como lo es en este caso la superficie esférica, la cual da un grado de inestabilidad mayor al del modelo de péndulo invertido sobre dos ruedas. Por esta razón las ruedas omnidireccionales permitieron el correcto funcionamiento del proyecto planteado, permitiendo la libre movilidad de la estructura sobre la esfera, sin generar ninguna perturbación por fricción.
Otro aspecto interesante a valorar, es el hecho de poder observar las características en tiempo real del sistema en lazo abierto, en donde se identifica los puntos de inestabilidad más

predominantes en la planta, y de esta manera poder entablar las variables, los conjuntos y las reglas que empleara el controlador difuso para poder estabilizar la posición de la planta.	
Elaborado por:	Diego Mauricio Muñoz García.
Revisado por:	Mg Alberto Morales.

Fecha de elaboración del Resumen:	11	02	2014

SISTEMA DE CONTROL PARA UN ROBOT SELF-BALANCING

DIEGO MAURICIO MUÑOZ GARCIA

Trabajo de grado para optar al título de Licenciatura en Electrónica

Director
Mg ALBERTO MORALES

Universidad Pedagógica Nacional
Facultad de Ciencia y Tecnología
Departamento de Tecnología
Licenciatura en Electrónica
Bogotá D.C
2013

tabla de contenido **Pag.**

FIGURAS

1. Introducción.

Las temáticas abordadas en diferentes áreas de la electrónica como lo son sistemas de potencia, control III (control difuso) y diseños electrónicos, se agrupan para dar desarrollo a este proyecto. Aun así la formación en las aulas sobre lógica difusa se enfoca hacia el desarrollo temático, sin profundizar en el diseño e implementación de hardware.

Este proyecto brinda herramientas que permiten ampliar los conocimientos sobre dicho tema, para la elaboración de diseños que hagan uso de sistemas de control difuso en modelos no lineales. Dichas herramientas no son solo datos teóricos y gráficos sino también un hardware que permite constatar los análisis previamente realizados.

Esto ayuda desde la didáctica de la pedagogía a tener otros puntos de vista y análisis sobre un determinado tema, puesto que ya no se iría de lo teórico a lo práctico, sino que se analizaría un sistema real. En la educación en tecnología y electrónica el hecho de poder verificar los análisis teóricos en cosas reales hace que el aprendizaje sea significativo.

En este proyecto se desarrolla un controlador de posición mediante la metodología de lógica difusa, su objetivo es controlar un sistema de auto estabilidad el cual se moviliza sobre un plano esférico y debe mantenerse a 90° grados con respecto al plano terrestre.

Dicho sistema consta de una estructura cilíndrica y tres motores los cuales generan el movimiento dependiendo de la posición que indique el sensor IMU, unidad de medida inercial.

2. PLANTEAMIENTO DEL PROBLEMA

Las diferentes investigaciones en robótica se plantean alrededor de tres campos de estudio: Swarm robotics, Field Robotics y Behavior Robotics. En este último campo se estudia el desarrollo de plataformas y su dinámica y cinemática buscando el diseño de controladores que brinden estabilidad, Tracking y Robustez.

Los robot tipo selfbalancing son una oportunidad interesante para profundizar en la teoría de control. Estos sistemas son similares a un péndulo invertido, el cual es un sistema físico compuesto por una barra cilíndrica que oscila libremente alrededor de un eje fijo. El eje está montado sobre una pieza móvil que se desplaza en dirección horizontal. La barra naturalmente tiende a caerse desde la posición vertical, ya que es una posición de equilibrio inestable.

El robot planteado en este estudio se compone de un eje (péndulo) el cual está realizando el equilibrio sobre una esfera. En este caso es necesario diseñar un sistema de control que permita llevar al punto de equilibro el robot.

3. ANTECEDENTES.

Estudio y desarrollo de una estrategia para controlar la estabilidad de una plataforma bípeda con proporciones antropométricas e implementación de un prototipo capaz de caminar en un trayecto plano. TE-12782

Este documento muestra de primera mano un análisis matemático por el método numérico de Euler del péndulo invertido real con un motor, un arreglo de piñones y una banda para el desplazamiento del péndulo, este último realizado en simulink®. Estos dos estudios se realizaron en pro del modelamiento de la marcha bípeda en un solo plano (plano x). El cual se plantea una estrategia de control por un arreglo fuzzy hibrido deslizante dividido en varios bloques fuzzy.

Desarrollo de un sistema de control de equilibrio para un móvil con dos ruedas de apoyo ubicadas lateralmente –robo 1.0- TE-13904

El documento muestra las pautas que se tuvieron en cuenta para llevar a cabo el proyecto de desarrollo "Desarrollo de un sistema de control de equilibrio para un móvil con dos ruedas de apoyo ubicadas lateralmente –robo 1.0-" el cual consiste en controlar un péndulo invertido sobre dos ruedas como instrumento de medida, para detectar el ángulo de inclinación; utiliza una IMU (unidad de medida inercial) y dos moto-reductores con encoders, donde los moto-reductores sirven como actuadores y los encoders como sensor de realimentación de la velocidad.

Diseño de un móvil con cuatro puntos de apoyo para terrenos irregular con sistema para control de estabilidad electrónico digital. TE-12312

Conseguir un sistema total acoplado entre parte de control y mecánica que le permita al móvil desplazarse por terreno irregular ejecutando maniobras para mantenerse en equilibrio, independiente de las condiciones de inclinación a los que lo someta el medio, teniendo en cuenta obviamente condiciones límites de operación.

4. JUSTIFICACIÓN.

El desarrollo y creación de prototipos que permitan la estabilidad de sistemas sobre una superficie (péndulo invertido) es una propuesta que se desarrolla en la universidad pedagógica como herramienta para promover en el estudiante el interés por la robótica y la teoría de control; buscando por medio de esta motivación incentivar la investigación en varias áreas de la electrónica.

Como propuesta y continuación a proyectos sobre péndulo invertido y sistemas de auto estabilidad ya desarrollados en la universidad, este estudio se enfoca en el diseño e implementación de un controlador que permita a un sistema no lineal mantenerse con un Angulo de 90° sobre una esfera y permitirle movilizarse a la vez en varias direcciones sin cambiar su orientación.

El robot tipo selfbalancing sobre una esfera es de tipo no lineal e inestable; con lo cual mantener en equilibrio el sistema se convierte en una tarea compleja. Para ello se enfocan los esfuerzos en la teoría de control difuso

Este prototipo busca implementar un nuevo método de movilización, en los cuales los sistemas de péndulo invertido sobre dos ruedas se detienen y giran para cambiar de dirección, este diseño sobre una esfera permitirá que el robot se desplace en cualquier dirección inmediatamente sin tener que parar, permitiendo que el uso que se le pueda dar después de implementado el controlador sea también a nivel industria para la movilización de objetos.

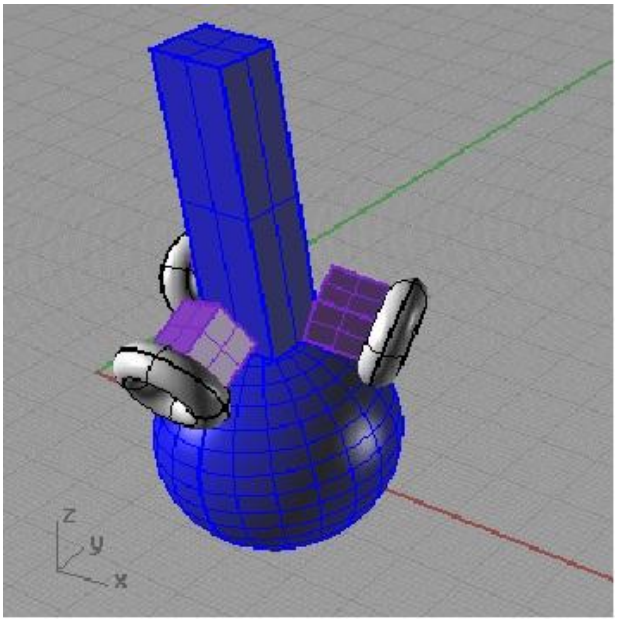


Figura 1 Modelo del robot en 3D

5. Objetivo General.

Diseñar e implementar un sistema de control que permita mantener el sistema mecánico en equilibrio en un ángulo de 90° sobre una esfera ante pequeñas perturbaciones.

5.2 Objetivos Específicos.

- Diseñar y construir una estructura mecánica basada en una esfera-barra, en la cual los motores DC interactúan con la esfera.

- Diseñar e implementar la etapa de instrumentación para identificar cambios de velocidad e inclinación en el sistema. (giroscopios y acelerómetros).

- Diseñar un sistema de control que permita estabilizar el sistema barra-esfera.

6. Desarrollo del sistema

6.1 Construcción de la estructura.

Para el desarrollo e implementación del prototipo como fase inicial, se diseña la estructura mecánica que va a soportar los componentes y los circuitos que rigen el funcionamiento del sistema.

En la construcción del cuerpo de la estructura se escoge el material acrílico y el aluminio por sus características de ligereza, duración, maleabilidad y que son materiales no tóxicos. Este cuerpo tiene una forma cilíndrica permitiendo dar estabilidad y firmeza a la planta, se construyeron 3 secciones en las cuales se alojaran los distintos circuitos y componentes que permitirán el correcto funcionamiento del controlador.

Figura 2. Cuerpo de la estructura.

La base de la estructura se encuentra diseñada en aluminio y tiene una forma triangular como se muestra en la siguiente imagen.

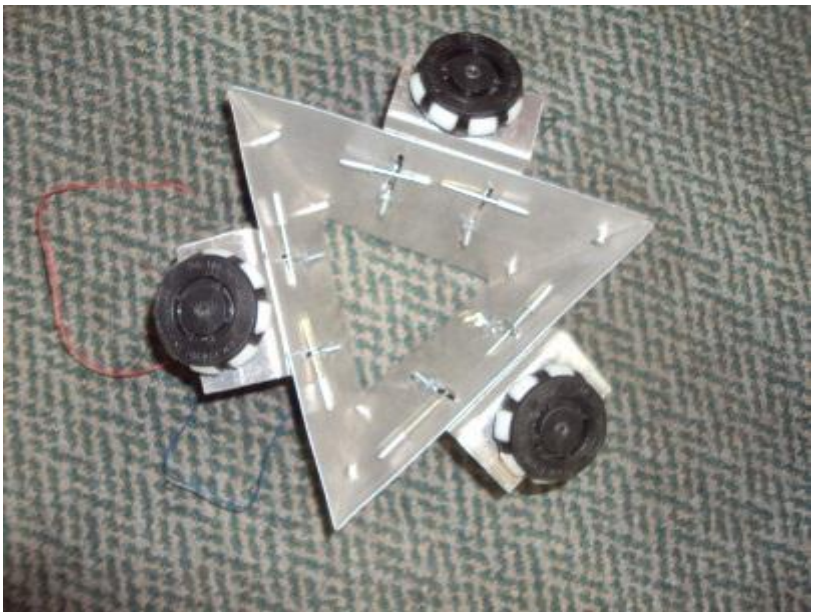


Figura 3. Base de los motores

Dicha base permite ajustar los tres motores en ángulos de 120°, lo cual permitirá que al colocar la estructura sobre una esfera los motores pueda manipular la esfera en distintas direcciones para así permanecer estable.

6.2 Actuadores.

En la sección de los actuadores que permitirán la interacción entre la estructura y la esfera para poderse estabilizar, se utiliza tres motores 100:1 Metal Gearmotor con encoder, con el cual se calcula la velocidad a la que se encuentra funcionando cada motor para realizar un control de velocidad y mantener el sistema estabilizado y preparado para soportar pequeñas perturbaciones en cuanto a su posición.

6.3 Características de los motores utilizados.

Los motores utilizados son los motores de 12 V DC 100:1 Metal Gearmotor 37Dx57L mm los cuales tienen una caja de engranajes de metal lo cual permite que el torque del motor soporte un peso de 16 Kg-CM, lleva un codificador en cuadratura integrada que proporciona una resolución de 64 recuentos por revolución del eje del motor, lo que corresponde a 6.533 recuentos por revolución del eje de salida de la caja de cambios,

Características técnicas del motor 100:1 Metal Gearmotor 37Dx57L mm

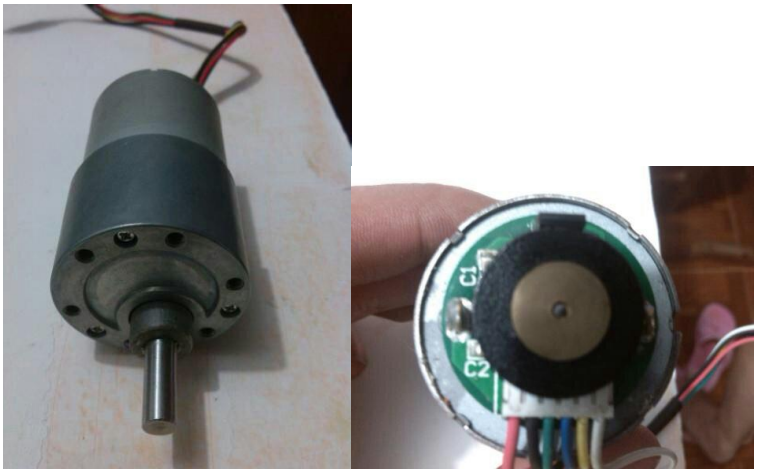


Figura 4. Motor 100:1 Metal Gearmotor

Dimensiones	
Tamaño	37D * 69L mm
Peso	7,9 oz
Diámetro del eje	6 mm

Especificaciones Generales	
Relación de reducción	100:1
RPM del motor de velocidad a 6v	50 rpm
Corriente del motor a 6v	250 mA
Torque a 6v	110 oz
RPM del motor a 12 V	100 rpm
Corriente del motor a 12v	300 mA
Corriente del motor a 12v con carga	5000 mA

Para el desarrollo de este proyecto se utilizara el motor con una velocidad de 100 RPM, para lo cual se hace necesario implementar una fuente que satisfaga las características de corriente

según el fabricante con una carga de 4 Kg. Estas características son importantes para el diseño del controlador de velocidad, debido a que se establecen los parámetros de revoluciones por minuto relacionado con el voltaje y la corriente.

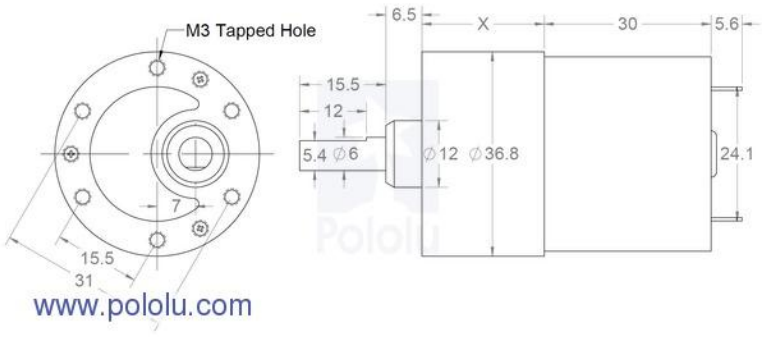


Figura 5. Dimensiones y estructura del motor. *Tomado de: http://www.pololu.com/category/116/37d-mm-gearmotors*

6.4 Sensor Encoder de cuadratura.

Una de dos canales de efecto Hall codificador se utiliza para detectar la rotación de un disco magnético en una protuberancia posterior del eje del motor. El codificador de cuadratura proporciona una resolución de 64 pulsos por revolución del eje del motor cuando se cuentan ambos bordes de ambos canales. Para calcular los conteos por revolución de la salida de la caja de cambios, multiplicar la relación de transmisión en un 64. El motor / Encoder tiene seis codificadas por colores: 11 "(28 cm) conduce termina con un conector hembra 1 × 6 con un 0,1" de paso.

El sensor Hall requiere una tensión de entrada, VCC, entre 3,5 y 20 V y dibuja un máximo de 10 mA. Las salidas A y B son ondas cuadradas de 0 V a Vcc aproximadamente 90 ° fuera de fase. La frecuencia de las transiciones le indica la velocidad del motor, y el orden de las transiciones le indica la dirección. La siguiente captura de osciloscopio muestra los A y B salidas del codificador (amarillo y blanco) con un voltaje del motor de 12 V y un sensor Hall Vcc de 5 V:

Contando tanto los flancos ascendente y descendente tanto de la salida A y B, es posible obtener 64 recuentos por revolución del eje del motor. El uso de un solo borde de uno de los

canales resultados en 16 recuentos por revolución del eje del motor, por lo que la frecuencia de la salida A en la captura osciloscopio anterior es 16 veces la frecuencia de rotación del motor. Esta información del motor fue extraída de la página del fabricante.

Figura 6. Señal de las salidas del Encoder.

Para el diseño del controlador que se implementara en este proyecto se utiliza solo una salida del Encoder lo cual permitirá que se trabaje con un modelo de controlador por conversión de frecuencia voltaje, para así poder realizar la parametrización del motor obteniendo el modelo matemático y de esta manera poder diseñar el control de velocidad para cada uno de los motores.

6.5 Rueda Transwheel (Rueda Sueca)

Como la estructura se desplaza sobre un plano esférico se hace necesario analizar el comportamiento de las ruedas debido a que se necesita un desplazamiento en diferentes direcciones sin frenar el movimiento de la estructura y girar lentamente hasta posicionarse adecuadamente, como en el movimiento clásico de los prototipos self-ballancing “segway”.

Para lo cual se utilizaran las ruedas Transwheel las cuales tienen un diseño con ocho rodillos de giro libre colados a 90 grados en el eje, alrededor de la periferia, permitiendo realizar movimientos rotatorios en cualquier dirección, ruedas omnidireccionales, estas ruedas son ideales para el desarrollo del controlador, debido a que al desplazar la estructura sobre la esfera

se necesita que las ruedas cuando estén girando en sentidos contrarios no generen una fricción que frene y desestabilice la planta, con estas ruedas se tiene desplazamiento sobre la superficie esférica en cualquier dirección sin que se generen perturbaciones en el movimiento que puedan cambiar el comportamiento planteado en el controlador.

Figura 7. Rueda Transwheel.

7.5 Etapa de potencia

La etapa de potencia permite acoplar los actuadores con los componentes digitales sin afectar ningún elemento del circuito, debido a que el funcionamiento de los controladores que se implementan opera con un voltaje máximo de 5v.

Para la construcción de la etapa de potencia, se utiliza un circuito puente H, el cual permitirá aislar la etapa de control y de potencia. El puente H que se utiliza es el integrado L298 el cual es un circuito permite una fuente de alimentación máxima de 46 voltios y hasta 4 Amperios de corriente.

Se realizaron pruebas de funcionamiento de los motores con una fuente de alimentación de 12 voltios y una carga de 4 a 5 Kg y se pudo identificar que la corriente nominal del sistema con los tres motores es de 4 amperios para una carga total del sistema de 5kg, esta corriente se debe a que se divide la carga en cada uno de los tres motores.

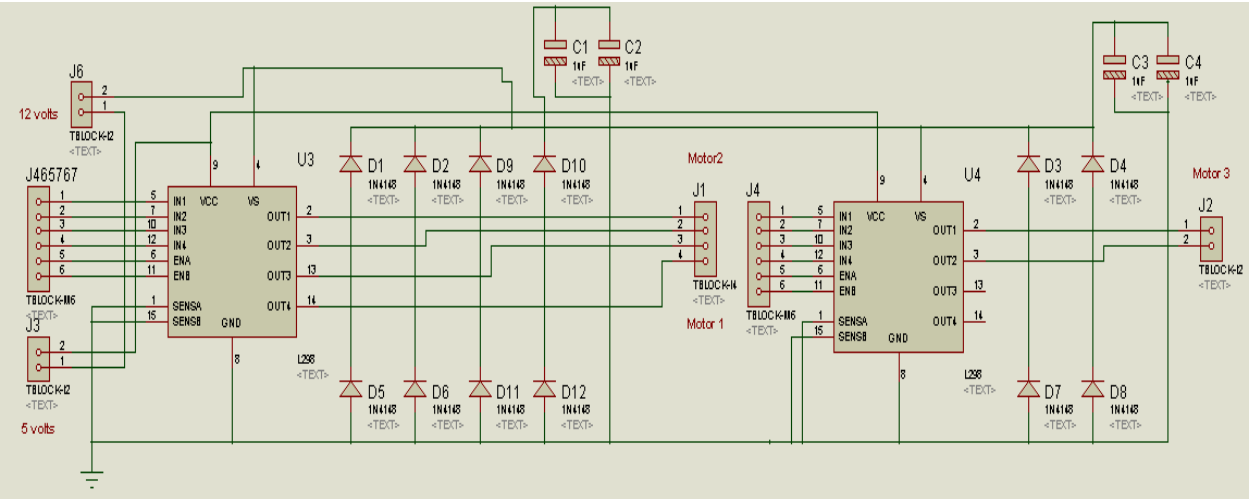


Figura 8. Circuito de Potencia para los tres motores.

7.6 Diseño del control de velocidad

Para el control de velocidad se utiliza el sensor Encoder el cual entrega una señal en forma de cuadratura para así realizar un proceso de conversión frecuencia voltaje para identificar la curva característica del motor; de esta manera realizar un modelo matemático que se aproxime al funcionamiento del motor y diseñar el control de velocidad, mejorando las características de funcionamiento del motor: respuesta en el tiempo, estabilidad y sobre-pico, que se generan cuando el motor entra en funcionamiento.

6.8 Tratamiento de la señal Encoder

La salida del Encoder de cada motor tiene las siguientes características:

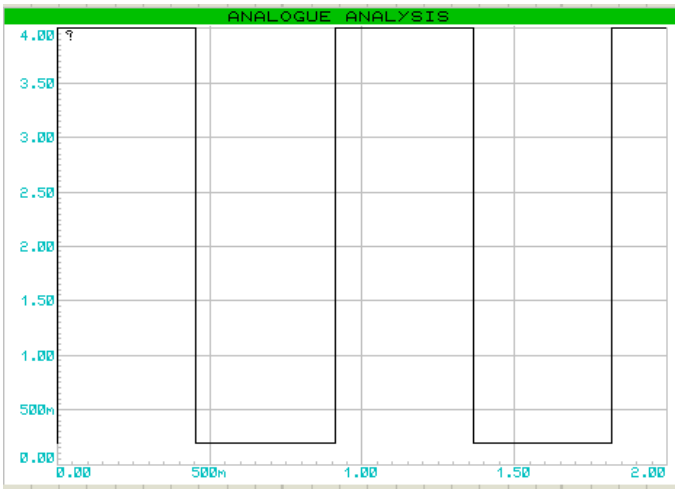


Figura 9. Señal de salida del Encoder.

El integrado LM331 o KA331 es un convertidor frecuencia voltaje el cuales un dispositivo que trabaja con tensiones de alimentación de 4V a 40 V y un consumo de corriente Max de 156 uA, este integrado permite realizar varias configuraciones de funcionamiento; una de ellas es la configuración convertidor frecuencia a tensión en donde se relaciona un valor de frecuencia asignado a la entrada y un valor de voltaje que se da en la salida.

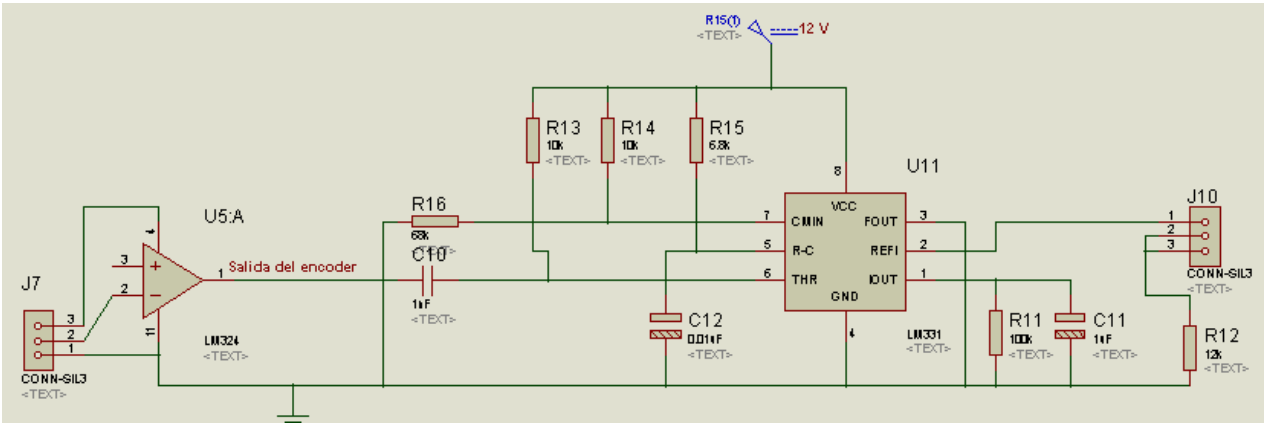


Figura 10. LM331 en configuración convertidor Frecuencia-voltaje

Esta configuración nos permite realizar una conversión de la frecuencia de pulso del Encoder a una salida expresada en voltios; la cual por medio de una tarjeta de adquisición de datos captura la información del comportamiento del motor y de esta manera se realiza la identificación del motor.

6.9 Modelamiento matemático del motor

Para el modelamiento matematico de los motores se utiliza un microcontrolador 16f877a programado para recibir datos analogos y expresarlo en un rango digital de 8 bits, los cuales se envian a un computador por medio del puerto serial para ser almacenados en un editor de texto, con esta informacion se utiliza la herramienta MATLAB para procesar los datos e identificar la funcion de transferencia de los motores en donde puede observar el respectivo comportamiento de cada motor.

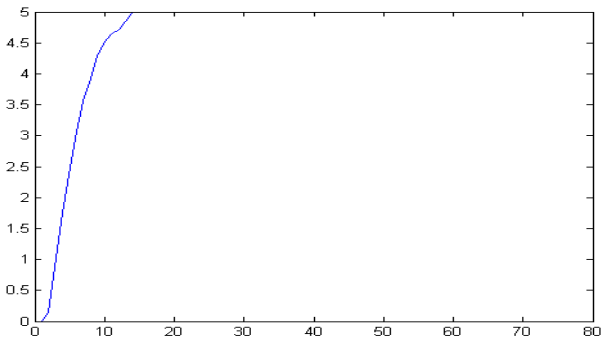


Figura 11. Grafico de datos del motor. Eje X datos capturados, eje Y voltaje convertidor frecuecnia voltaje

Grafica obtenida con los datos del funcionamiento del motor en lazo abierto, capturados por medio del sensor encoder. Los valores en el eje Y son los voltajes de funcionamiento del motor y el eje X representa los valores de tiempo de muestreo con el que se realizo la captura de datos del encoder, en donde se puede observar las caracteristicas del tiempo de respuesta, sobre-pico en el arranque del motor y el tiempo en el que se estabiliza el motor.

Utilizando la ToolBox System Identification de Matlab se obtiene la función de transferencia que más se aproxime a la respuesta experimental del motor.

Funcion de transferencia del motor:

$$G(s) = \frac{1.001}{0.004669s+1}$$

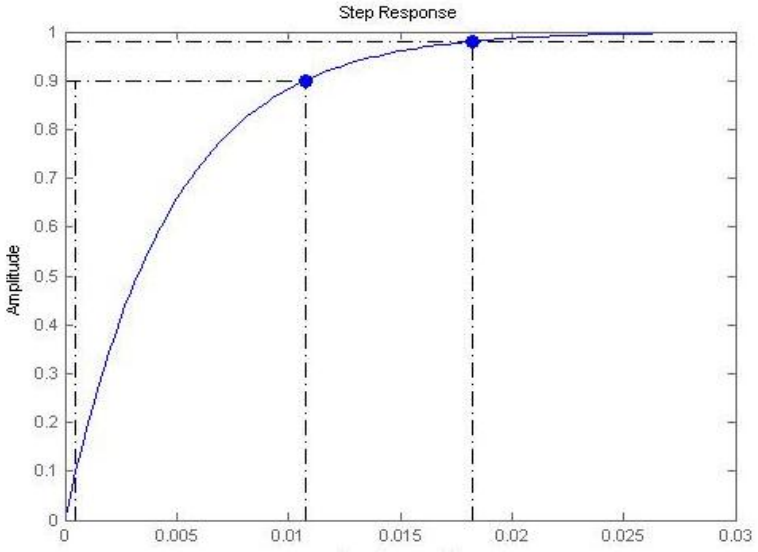


Figura 12. Curva del modelo parametrico del motor normalizada.

En la figura 12 se observa el grafico que representa la funcion de tranferencia calculada con un 98% de proximidad con los datos obtenidos del motor, en los cuales se puede apreciar el tiempo de respuesta en un 0.0016, el estado estable en un 0.019 y no presenta un sobre-pico de señal, comportandose como un sistema de orden 1.

De igual manera se realiza la identificación de los 3 motores los cuales tienen las siguientes funciones de transferencia:

$$Gm1(s) = \frac{1}{0.004446s+1}$$

$$Gm2(s) = \frac{1.001}{0.004669s+1}$$

$$Gm3(s) = \frac{1}{0.004669s+1}$$

Como se puede observar las tres funciones son muy semejantes, esto debido a que los tres motores son de la misma referencia y sus componentes mecánicos son muy similares.

6.10 Control PID

La metodología que se implementara para mantener la velocidad en RPM de los motores, será el controlador PID implementado en el Arduino. El control PID está compuesto por una acción proporcional, una acción derivativa y una acción integradora.

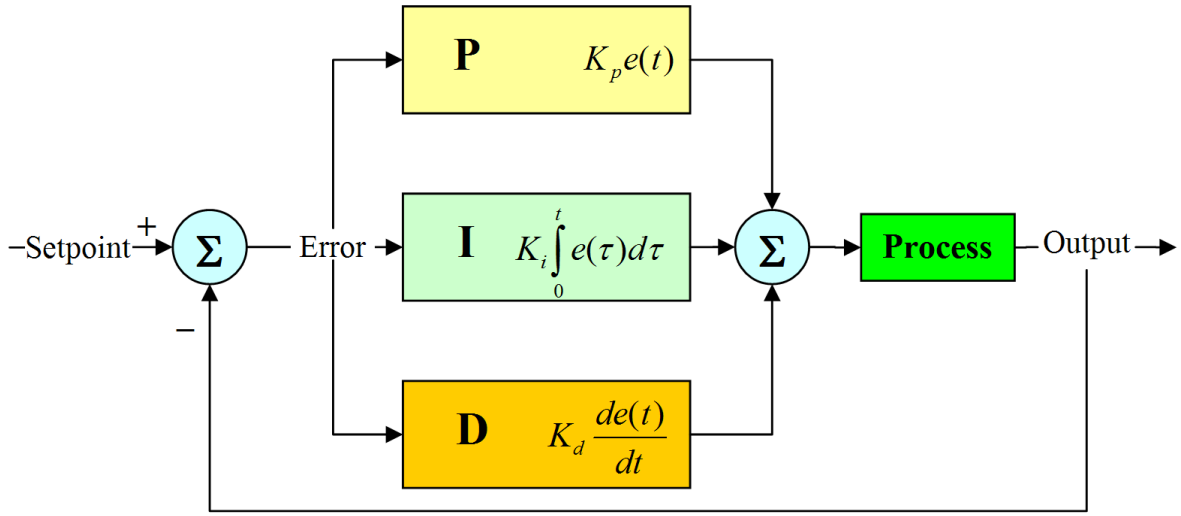


Figura 13 Esquema de composición PID.

Para la implementación en el Arduino se utilizara la suma de las 3 acciones:

- Acción proporcional.

P= kp*error;

El término proporcional realiza un cambio en la salida que es proporcional al valor actual, en donde permite mejorar el tiempo de respuesta del sistema. Por encima de un valor limite el cual es el que se debe sintonizar, el sistema comenzara a oscilar.

- Acción derivada.

D= Kd*(error – error_anterior);

Esta acción proporcional a la variación de error, esta acción calcula que tan rápido se produce un cambio en el sistema.

- Acción integral

Integrador_error + = error;

I= ki * (integrador_error);

El término integral es el que permite el cálculo de ajuste para el valor de aterrizaje del sistema "error de estado estacionario". Para calcular todo el controlador PID se realiza la suma de las 3 medidas:

```
Int error = ( int punto_entrada, set_point);
P= kp*error;
Integrador_error + = error;
I= ki * ( integrador_error);
D= KD*(error – error_anterior);
Error_anterior= error;
Return  ((P+I+D));
```

- Sintonización manual

Sintonización manual, coloque las tres contantes a cero, variar el valor de KP hasta que se produce la oscilación.

Luego subir Kd hasta oscilación desaparece. Ajuste KD hasta que el sistema este críticamente amortiguado, y luego aumentar el Ki hasta que el error de estado estacionario tiende a cero en un tiempo razonable.

A continuación se muestra un diagrama de flujo de cómo se realiza el proceso de control PID en

El compilador Arduino.

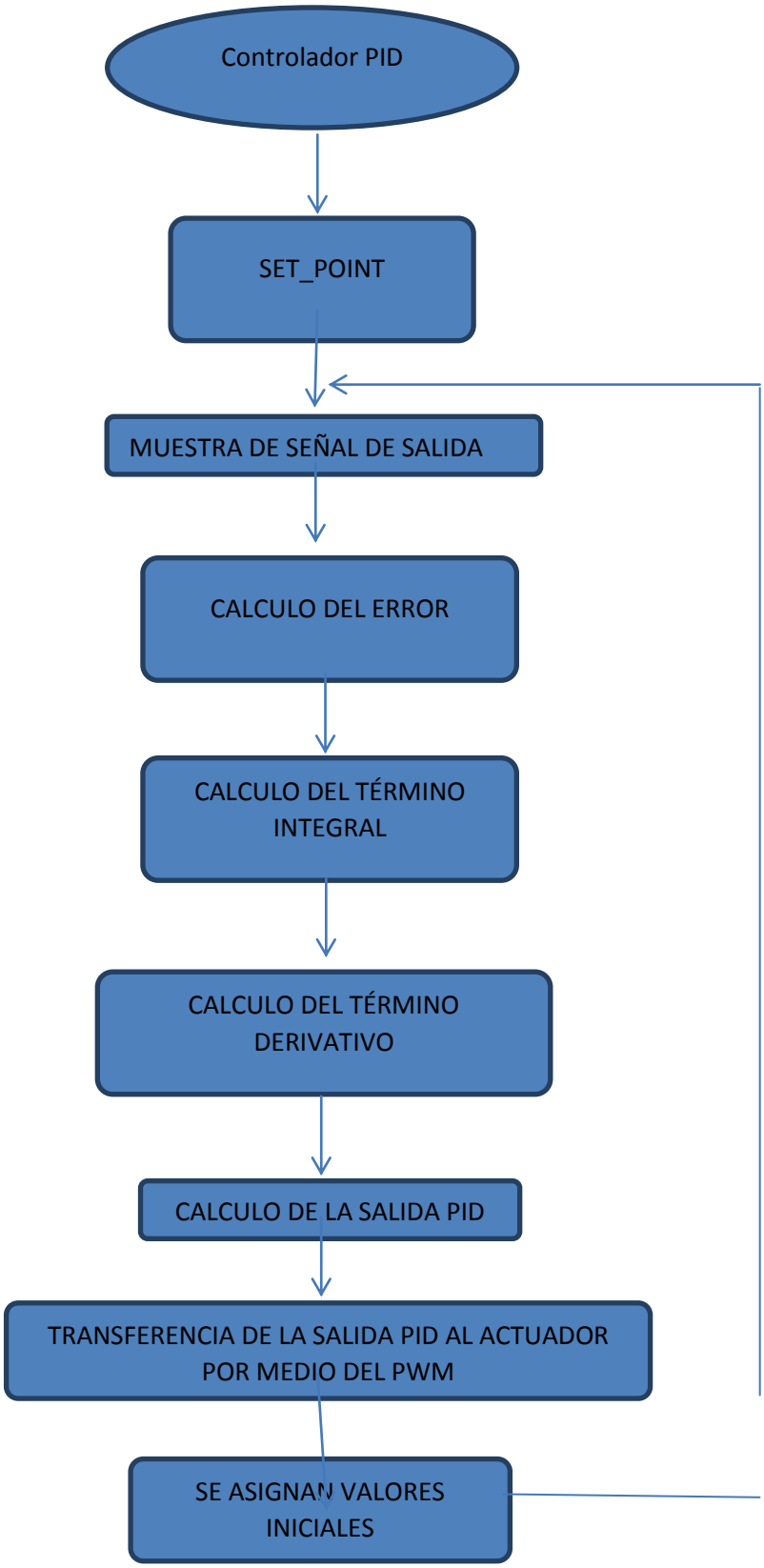


Figura 14 Diagrama de flujo del controlador PID.

6.11 Sensor IMU.

La estructura utiliza los datos del Angulo de inclinación tanto en el eje X y el eje Y el cual es el sensor principal por medio del cual el controlador difuso toma la decisión de dirección velocidad a la que deben girar los motores. El sensor es una IMU, Unidad de Medida Inercial, el cual está compuesto por el acelerómetro ADXL335 y dos giroscopios: LPR530AL y LY530ALH.

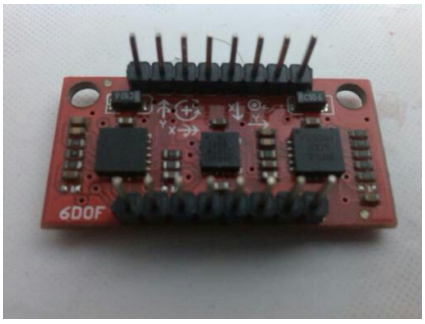


Figura 15.IMU utilizada en este desarrollo

El acelerómetro ADXL335 el cual es un integrado que trabaja con un voltaje de 3.3 voltios de alimentación con un consumo de corriente de 0.7 mA y 1.1 mA. Trabaja con los ejes “X” y “Y”, siendo capaz de calcular tanto la aceleración dinámica, la vibración, como la aceleración estática, La gravedad, permitiendo ser utilizado con sensor de inclinación. La señal de salida son voltajes análogos proporcionales a la aceleración hasta 1.7° y -1.7°, la sensibilidad de la salida tiene la característica de ser una señal proporcional a la fuente de alimentación.

Los giroscopios LPR530AL y LY530ALH son dispositivos que permiten conocer como varia un ángulo en el tiempo, mientras este encuentra rotando, velocidad angular, con respecto al sistema de referencia fijo. Estos giroscopios tienen un rango a escala de 500°/s, con una sensibilidad de 2.0 mV°/s en estado estático, velocidad angular igual a 0, en sensor entrega en la salida un voltaje de 1.5, con un consumo de corriente de 9.5mA en reposo.

6.12 Filtro Kalman

El filtro kalman es un algoritmo que fue desarrollada por Rudof E. Kalman en 1960 y describe una solución recursiva para problemas de filtrado de datos discretos. Es un estimador que puede implementarse de manera sencilla en sistemas lineales como no lineales, y cuyo

procesamiento de datos es de carácter recursivo, recibiendo y procesando todas las mediciones posibles y en base a estas, estima el valor actual de las variables de interés. Con el fin de llevar a cabo es necesario conocer el sistema en el cual va a hacer implementado el algoritmo, conocer las condiciones iniciales de las variables más importantes que rigen el modelamiento del sistema. El filtro discreto es un algoritmo mediante el cual se realiza un proceso de predicción y otro de corrección mediante la medición de un grupo de variables presentes en el sistema, dicho conjunto de variables forman el vector de estados. Buscando obtener una variable estimada que se acerque lo más posible a la realidad.

Para generar el filtro Kalman se establecen unas constantes para los datos en X y Y:

```
float Q_angle  =  0.001; //0.001
float Q_gyro   =  0.003;  //0.00
float R_angle  =  0.03;  //0.03
```

Los cuales son constantes ya establecidas por el fabricante, se declaran las condiciones iniciales del sistema y de los valores que almacenaran los datos

```
float x_angle = 0;
float x_bias = 0;
float xP_00 = 0, xP_01 = 0, xP_10 = 0, xP_11 = 0;
float xK_0, xK_1;
```

Se crea una secuencia que realizara la estimación de los datos con los valores de sintonización para obtener el ángulo real, con la corrección a la IMU.

6.13 Control difuso

El control que se implementa para desarrollar el proceso de estabilización de la planta es el control difuso o lógica borrosa, el cual tiene una ventaja sobre el control clásico y el control digital, esto debido a que el control difuso permite de manera un poco más sencilla analizar una cantidad alta de variables sin tener que regirse bajo un modelo matemático estricto.

Tiene la característica de agrupar datos en conjuntos clásicos por medio de expresiones con imprecisión generando modelos intuitivos, esa agrupación se realiza medio del grado

pertenencia que se le asigna a cada variable de entrada al sistema permitiendo determinar la decisión o acción que debe tomar la salida del controlador y obtener un modelo aproximado del sistema. Todo esto en un intervalo de tiempo considerablemente pequeño.

En comparación con el control análogo PID, los tiempos de respuesta del controlador difuso son mucho menores, al manejar una gran cantidad de variables y reglas que rigen el sistema el error en estado estable más pequeño que en los otros métodos de control, el controlador difuso responde mejor a las perturbaciones y en la parte de implementación de hardware en el controlador PID se requiere de un modelo matemático que terminaría siendo altamente complejo, mientras que en el controlador difuso se requiere de menos cantidad de hardware puesto que está más enfocado al procesamiento de datos por software, esto dando una alternativa económica para implementación de este proyecto.

Por estas razones se desarrollara este proyecto con la metodología de lógica difusa la cual es implementada en un Arduino Duemilanove.

El Arduino Duemilanove es una placa con un micro controlador ATMega328p, tiene 14 pines entradas/salidas digitales, 6 de las cuales pueden ser usadas como salidas PWM para los actuadores del sistema, 6 entradas analógicas las cuales permitirán capturar los datos de la IMU, y con esos datos estimar a través del filtro Kalman el ángulo al que se encuentra el robot; un cristal oscilador a 16Mhz, conexión USB, entrada de alimentación, una cabecera ISCP, y un botón de reset. Contiene todo lo necesario para acceder a los recursos del microcontrolador.

Para el desarrollo matemático y de procesamiento de datos para el controlador difuso se utilizara las librerías Efll, *Embedded Fuzzy Logic Library*, la cual tiene una licencia "*Creative Commons Reconocimiento-SemDerivados 3.0 Unported*." La cual permite copiar y redistribuir el material en cualquier medio o formato, brindando herramientas para un modelamiento adecuado de la sintaxis y el lenguaje de la lógica difusa gracias a que es un código generado en C ++ por lo cual se puede utilizar en los procesadores que comprendan este lenguaje, como en el caso de este proyecto el Arduino Deumilanove.

El desarrollador de la librería indica que esta librería no tiene limitaciones explicitas sobre la cantidad de Reglas, conjuntos y universos, solo está limitado a la capacidad de procesamiento y almacenamiento de cada microcontrolador. A continuación se realizara un recuento de la documentación brindada por el desarrollador sobre la librería a utilizar en el integrado Atmega.

6.8 Funcionamiento librería eFLL

Para el desarrollo de este proyecto se utilizaron las librerías eFLL para facilitar la implementación en el procesador. Es importante tener claridad de algunos términos:

Fuzzy Object - Este objeto incluye todo el sistema difuso, a través de él, puede manipular los conjuntos borrosos, las reglas lingüísticas, entradas y salidas.

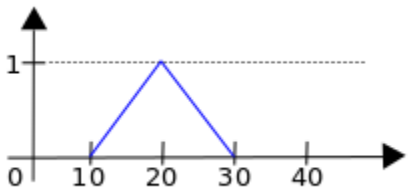
Objeto FuzzyInput grupos Este objeto todas las entradas literalmente Fuzzy que pertenecen al mismo dominio -

Objeto FuzzyOutput - Este objeto es similar a FuzzyInput, se utiliza para todos los grupos de salida Fuzzy pertenecientes al mismo dominio.

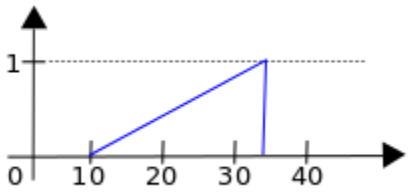
Objeto FuzzySet - Este es uno de los principales objetos de la biblioteca Fuzzy, con cada conjunto puede modelar el sistema en cuestión. Actualmente, la biblioteca es compatible con las funciones de triangular, singleton y relevancia trapezoidal, que se ensamblan sobre la base de los puntos A, B, C y D que se pasa como parámetro en el constructor **FuzzySet (float a, float b, c float, float)**

Ejemplos de declaración de funciones de pertenencia:

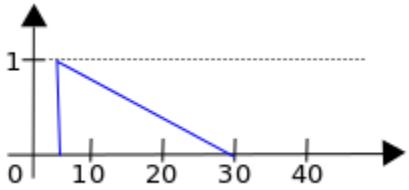
*Función de pertenencia triangular***:**



FuzzySet * fs = FuzzySet (10, 20, 20, 30);

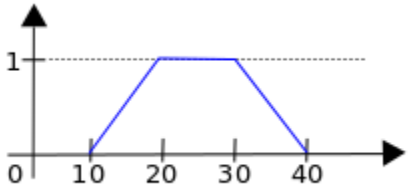


FuzzySet * fs = FuzzySet (10, 33, 33, 33);

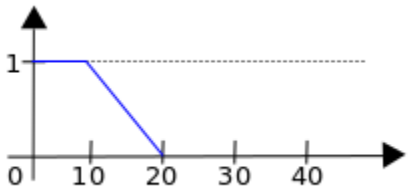


FuzzySet FuzzySet * fs = (5, 5, 5, 30);

Función de pertenencia trapezoidal:

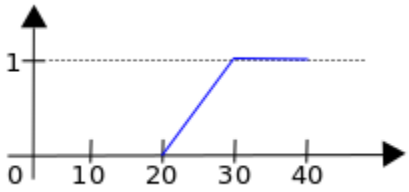


FuzzySet * fs = FuzzySet (10, 20, 30, 40);



FuzzySet FuzzySet * fs = (0, 0, 10, 20);

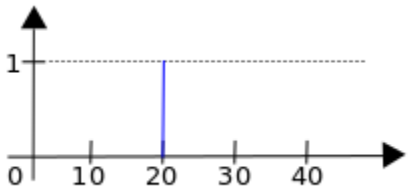
Cualquier valor por debajo de 10 tendrá relevancia = 1



FuzzySet * fs = FuzzySet (20, 30, 40, 40);

Cualquier valor por encima de 30 tendrá relevancia = 1

Función de pertenencia Singleton:



FuzzySet * fs = FuzzySet (20, 20, 20, 20);

Objeto FuzzyRule - Este objeto se utiliza para construir la base de reglas difusas del objeto, que contiene uno o más de esto. Instancia con **FuzzyRule fr = new FuzzyRule (id, antecedente, consecuente)**

Objeto FuzzyRuleAntecedent - Este objeto se utiliza para componer el objeto FuzzyRule, responsable del montaje del antecedente de la expresión condicional de un FuzzyRule ejemplos:

"SI ENTONCES distancia = pequeña = velocidad lenta"

FuzzyRuleAntecedent ifDistanceSmall * = nuevo FuzzyRuleAntecedent (); ifDistanceSmall-> joinSingle (pequeño);

El método joinSingle (FuzzySet * fuzzySet), se utiliza para preparar las expresiones simples SI ENTONCES que este. Para componer expresiones más complejas existen otros métodos especiales.

"IF = temperatura caliente y la presión = alto riesgo = grande ENTONCES"

FuzzyRuleAntecedent ifTemperatureHotAndPressureHight * = nuevo FuzzyRuleAntecedent ();

ifTemperatureHotAndPressureHight-> joinWithAND (frío, altura);

"IF = temperatura o la presión caliente = alto riesgo = grande ENTONCES"

FuzzyRuleAntecedent ifTemperatureHotAndPressureHight * = nuevo FuzzyRuleAntecedent ();

ifTemperatureHotAndPressureHight-> joinWithOR (frío, altura);

Los métodos joinWithAND (FuzzySet fuzzySet1 *, * FuzzySet fuzzySet2) y joinWithOR (FuzzySet fuzzySet1 *, * FuzzySet fuzzySet2); pueden hacer composiciones lógicas entre conjuntos borrosos. Estos métodos todavía tienen variaciones más avanzadas que permiten mejorar aún más la expresión, que son:

bool joinWithAND (FuzzySet * fuzzySet, FuzzyRuleAntecedent fuzzyRuleAntecedent *);

bool joinWithAND (FuzzyRuleAntecedent * fuzzyRuleAntecedent, FuzzySet fuzzySet *);

bool joinWithOR (FuzzySet * fuzzySet, FuzzyRuleAntecedent fuzzyRuleAntecedent *);

bool joinWithOR (FuzzyRuleAntecedent * fuzzyRuleAntecedent, FuzzySet fuzzySet *);

bool joinWithAND (FuzzyRuleAntecedent * fuzzyRuleAntecedent1, FuzzyRuleAntecedent fuzzyRuleAntecedent2 *);

bool joinWithOR (FuzzyRuleAntecedent * fuzzyRuleAntecedent1, FuzzyRuleAntecedent fuzzyRuleAntecedent2 *);

Si la aplicación de estos métodos, cualquier expresión puede estar equipado, desde un objeto FuzzyRuleAntecedent se puede utilizar para componer otro objeto FuzzyRuleAntecedent, de muchas maneras diferentes.

FuzzyRuleAntecedent speedHightAndDistanceSmall * = nuevo FuzzyRuleAntecedent ();

speedHightAndDistanceSmall-> joinWithAND (altura, pequeño);

/ / Este objeto es FuzzyRuleAntecedent que será utilizado para componer la FuzzyRule objeto

FuzzyRuleAntecedent ifSpeedHightAndDistanceSmallOrFuelLow * = nuevo FuzzyRuleAntecedent ();

ifSpeedHightAndDistanceSmallOrFuelLow-> joinWithOR (speedHightAndDistanceSmall, bajo);

Objeto FuzzyRuleConsequente - Este objeto se utiliza para componer el objeto FuzzyRule, responsable del montaje de la expresión de salida de unos ejemplos FuzzyRule:

"SI ENTONCES distancia = pequeña = velocidad lenta"

FuzzyRuleConsequent thenSpeedSlow * = nuevo FuzzyRuleConsequent ();

thenSpeedSlow-> addOutput (lento);

Esto daría lugar a un objeto FuzzyRule:

FuzzyRule fuzzyRule * = nuevo FuzzyRule (2 ifDistanceSmall, thenSpeedSlow);

"Si (Alta velocidad = distancia E = pequeño) O ENTONCES velocidad = menos combustible = pequeña = bajo consumo E"

FuzzyRuleConsequent thenSpeedSmallAndFeedTine * = nuevo FuzzyRuleConsequent ();

thenSpeedSmallAndFeedSmall-> addOutput (pequeño);

thenSpeedSmallAndFeedSmall-> addOutput (diente);

FuzzyRuleConsequent objeto a toda la expresión está equipado con método addOutput (FuzzySet fuzzySet *);

Esto daría lugar a un objeto FuzzyRule:

FuzzyRule fuzzyRule * = nuevo FuzzyRule (2 ifSpeedHightAndDistanceSmallOrFuelLow, thenSpeedSmallAndFeedTine);

Después de montar un objeto FuzzyRule, utilice el método (FuzzyRule * fuzzyRule), el objeto difuso addFuzzyRule para añadir la regla a la base de reglas, repita el mismo proceso para todas las reglas.

Función SET.

Estos son todos los objetos de la biblioteca eFLL que se utilizan en el proceso. El siguiente paso es interactivo y, en general manipulado por los métodos del objeto Fuzzy, la primera,

SetInput bool (int id, valor flotante);

Se utiliza para pasar el valor de la entrada crispe al sistema, tenga en cuenta que el primer parámetro es el ID de objeto FuzzyInput a la que se destina el valor del parámetro.

fuzzify bool ();

Se utiliza para iniciar el proceso de fuzzyficación, la composición y la inferencia.

Y, por último: float defuzzify (int id);

Se utiliza para finalizar el proceso fuzzyficación en cuenta que el ID que se pasa por parámetro es el ID de la FuzzyOutput que para obtener el valor desfuzzificado.

Consejo: A veces es necesario conocer la importancia con la que algunos o cada conjunto difuso se activó. Para ello, utilice el getPertinence float (método), el objeto de Objet FuzzySet este valor, por ejemplo:

* = FuzzySet caliente nueva FuzzySet (30, 50, 50, 70);

. / / Después fuzzyficación con -> fuzzyfy ();

float en caliente pertinenceOfHot => getPertinence ();

O si una regla en particular está habilitado, utilice el método **bool isFiredRule (int ReglaID)** objeto borroso.

FuzzyRule fuzzyRule * = nuevo FuzzyRule (2 ifDistanceSmall, thenSpeedSlow)

/ / Después fuzzyficación con -> fuzzyfy ();

bool = fuzzy-wasTheRulleFired> isFiredRule (2);

6.14 Conjuntos difusos

Se establecen 8 universos en discurso que rigen el funcionamiento del sistema self ballancing sobre la esfera, los cuales son dos para la entrada y 6 universos de salida los cuales se describen a continuación.

Entradas del sistema

Para los conjuntos de entrada tomamos los datos entregados por la IMU y el filtro Kalman:

- **Angulo *x***

AnguloX				
Phx	20	30	50	50
Plx	2,5	10	25	35
Zerox	-20	-5	5	20

Nlx	-35	-25	-10	-2,5
Nhx	-50	-50	-30	-20

Tabla 1 entrada Angulo x

El ángulo *x* se divide en 5 conjuntos los cuales describen el comportamiento que se desplaza de -50 a 50 grados.

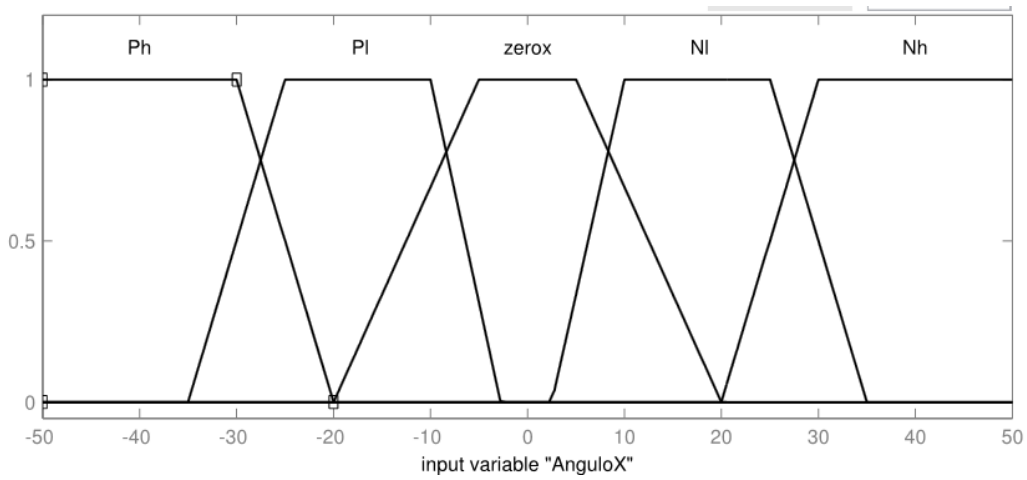


Figura 15. Funciones de pertenencia entrada Angulo X realizadas con FUZZY- Matlab

- **Angulo *y***

AnguloY				
Phy	20	30	50	50
Ply	2,5	10	25	35
Zeroy	-20	-5	5	20
Nly	-35	-25	-10	-2,5
Nhy	-50	-50	-30	-20

Tabla 2 Conjunto señal de entrada ángulo *Y*.

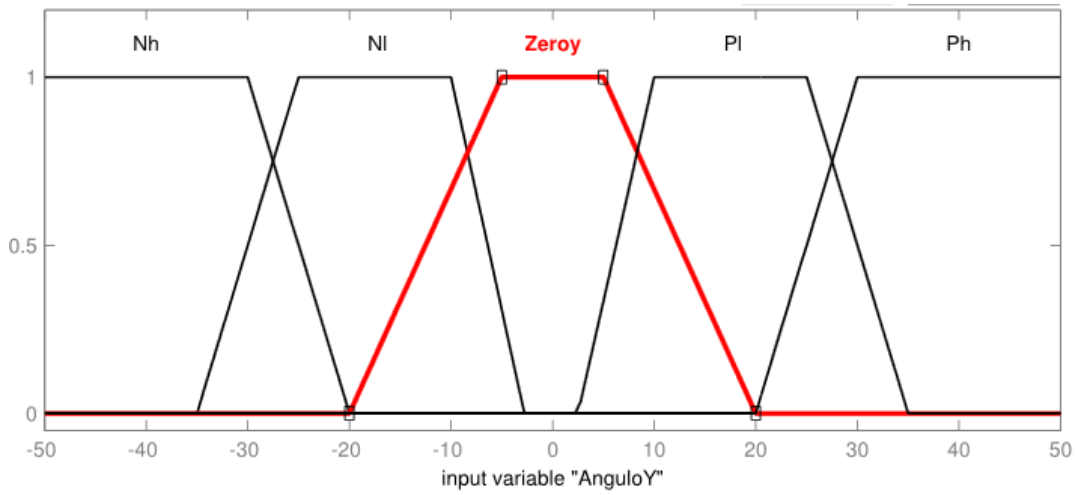


Figura16. Funciones de pertenencia entrada Angulo Y realizadas con FUZZY Matlab.

El ángulo y se divide en 8 conjuntos los cuales describen el comportamiento que se desplaza de -50 a 50 grados.

Salidas del sistema.

- **Actuadores**

Motor				
neutro1	0	0	80	127
medio1	24	127	127	230
rapido1	127	180	255	255

Tabla 3. *Salidas de los actuadores en valores de PWM.*

Para cada uno de los motores se crea un universo en discurso distinto, con tres conjuntos difusos que rigen el comportamiento del motor en cuanto a variables de velocidad establecidas por un PWM con una resolución de 8 bits.

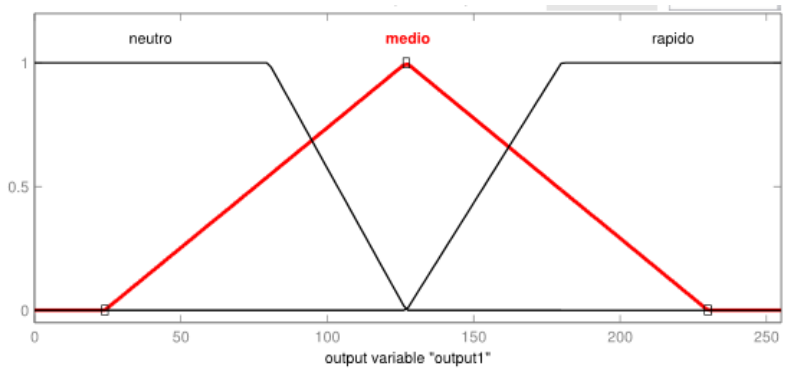


Figura17. Funciones de pertenencia que representan la velocidad de los motores.

Dirección

Dirección				
izquierda1	1	1	1	1
quieto1	0	0	0	0
derecha1	3	3	3	3

Tabla 4 *Salidas de los actuadores en valores de representación de la dirección.*

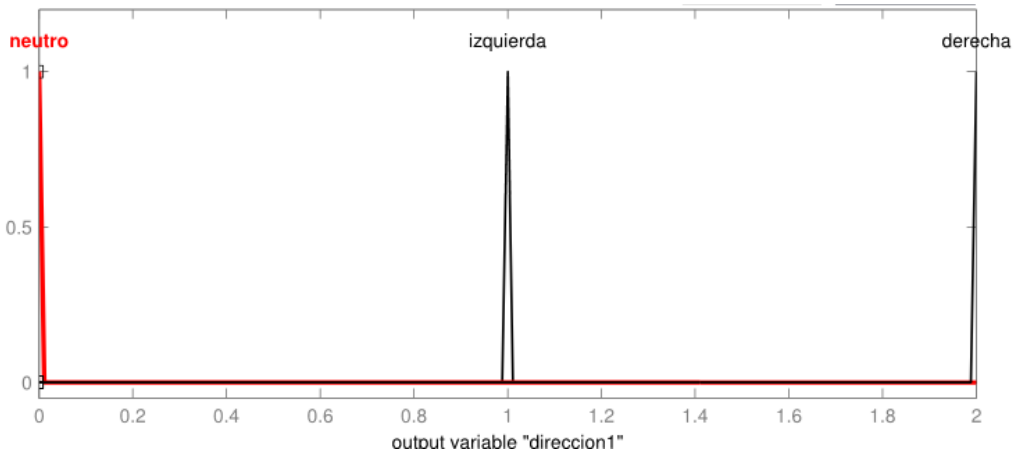


Figura18. Funciones de pertenencia que representan la dirección de los motores.

Para la dirección se establece 3 conjuntos difusos tipo singleton para darle un valor unitario a cada movimiento.

Luego de establecer los conjuntos de entrada y salida que permiten simular de manera aproximada el comportamiento de la planta, se procede a realizar las reglas que determinaran los valores y las acciones que realizaran los actuadores para estabilizar la planta. La combinación de esos dos universos genera los antecedentes del controlador.

Para la construcción de las reglas se genera una tabla con los antecedentes y los consecuentes para cada uno de los casos controlador:

- Reglas difusas.

AngX	AngY	Motor1	Motor2	Motor3	Direccion1	Direccion2	Direccion3
Zerox	Zeroy	neutro1	neutro2	neutro3	quieto1	quieto2	quieto3
Zerox	Ply	medio1	medio2	neutro3	derecha1	izquierda2	izquierda3
Plx	Zeroy	medio1	medio2	neutro3	izquierda1	derecha2	derecha3
Nlx	Zeroy	neutro1	medio2	medio3	izquierda1	izquierda2	derecha3
Phx	Zeroy	rapido1	rapido2	neutro3	izquierda1	derecha2	derecha3
Nhx	Zeroy	neutro1	rapido2	rapido3	izquierda1	izquierda2	derecha3
Zerox	Nly	medio1	medio2	neutro3	izquierda1	derecha2	derecha3
Plx	Ply	neutro1	medio2	medio3	derecha1	derecha2	izquierda3
Phx	Phy	neutro1	rapido2	rapido3	derecha1	derecha2	izquierda3
Nhx	Nhy	rapido1	rapido2	neutro3	derecha1	izquierda2	izquierda3
Nlx	Nly	medio1	medio2	neutro3	derecha1	izquierda2	izquierda3
Zerox	Nhy	rapido1	rapido2	neutro3	derecha1	izquierda2	izquierda3
Zerox	Phy	rapido1	rapido2	neutro3	izquierda1	derecha2	derecha3
Nlx	Ply	medio1	medio2	neutro3	derecha1	derecha2	derecha3
Nhx	Phy	rapido1	rapido2	neutro3	derecha1	derecha2	derecha3

Estos algunos de los movimientos más predominantes en la estructura para que esta permanezca estable frente a pequeñas perturbaciones.

En la construcción de las reglas se tuvo en cuenta todos los posibles movimientos con respecto al ángulo de inclinación entregado por el sensor IMU, En donde en algunas ocasiones un motor se movía en dirección contraria a los otros dos y con distinta velocidad, todo esto para poder mover la esfera en la dirección adecuada que permitiera estabilizar el sensor IMU en el conjunto y grado de pertenencia adecuado para que la estructura no alcanzara una inclinación en donde se desestabilizara hasta caerse de la esfera.

En la configuración de cada una de las reglas se hace necesario conocer en detalle la acción o consecuente que ve a realizar la combinación de los antecedentes para obtener el resultado adecuado, en donde se involucra bastante la cantidad de variables y de conjuntos que se establezcan, entre más conjuntos y más variables el error de estado estable disminuirá, pero a su vez se generara una carga de procesamiento en los datos que realizara el microcontrolador y se verá afectado y reflejado en tiempos de respuesta.

Ejemplo de la declaración de una regla difusa por medio de la librería eFLL.

AngX	AngY	motor1	motor2	motor3	direccion1	direccion2	direccion3
Zerox	Zeroy	neutro1	neutro2	neutro3	quieto1	quieto2	quieto3

```
/*Creación  del Antecedente 1*/
FuzzyRuleAntecedent*ifAnguloXzeroxAndAnguloYzeroy = new FuzzyRuleAntecedent();
ifAnguloXzeroxAndAnguloYzeroy->joinWithAND(zerox,zeroy);

/*Creación  del consecuente */
FuzzyRuleConsequent*thenmotor1neutro1ANDmotor2neutro2ANDmotor3neutro3= new
FuzzyRuleConsequent();
thenmotor1neutro1ANDmotor2neutro2ANDmotor3neutro3->addOutput(neutro1);
thenmotor1neutro1ANDmotor2neutro2ANDmotor3neutro3->addOutput(neutro2);
thenmotor1neutro1ANDmotor2neutro2ANDmotor3neutro3->addOutput(neutro3);
```

```
/*base de datos reglas */
FuzzyRule*fuzzyRule1= new
FuzzyRule(1,ifAnguloXzeroxAndAnguloYzeroy,thenmotor1neutro1ANDmotor2neutro2ANDmoto
r3neutro3);
fuzzy->addFuzzyRule(fuzzyRule1);

/*Creación  del Antecedente 1*/
FuzzyRuleAntecedent*ifAnguloXzeroxAndAnguloYzeroy = new FuzzyRuleAntecedent();
ifAnguloXzeroxAndAnguloYzeroy->joinWithAND(zerox,zeroy);

/*Creación  del consecuente  */
FuzzyRuleConsequent*thendireccion1quieto1ANDdireccion2quieto2ANDdireccion3quieto3=
new FuzzyRuleConsequent();
thendireccion1quieto1ANDdireccion2quieto2ANDdireccion3quieto3->addOutput(quieto1);
thendireccion1quieto1ANDdireccion2quieto2ANDdireccion3quieto3->addOutput(quieto2);
thendireccion1quieto1ANDdireccion2quieto2ANDdireccion3quieto3->addOutput(quieto3);

/*base de datos reglas */
FuzzyRule*fuzzyRule1= new
FuzzyRule(1,ifAnguloXzeroxAndAnguloYzeroy,thendireccion1quieto1ANDdireccion2quieto2AND
direccion3quieto3);
fuzzy->addFuzzyRule(fuzzyRule1);
```

Como se puede observar para la creación de una regla if AnguloX = zerox AND anguloy = zeroy, entonces la dirección del motor 1 = quieto1 and direccion2 = quiero2 y la direccion3 = quieto3.

Es necesario declarar el antecedente, declarar el consecuente, declarar la regla y almacenarla en una matriz, lo cual permitirá al controlador verificar esta información e identificar cual es la acción a realizar. Dicha información se almacena en la EEPROM del Arduino.

7.14 Método de comprobación de las reglas difusas.

Para este proyecto se implementaron dos metodologías para probar las reglas difusas:

1. Utilizando la herramienta FUZZY de Matlab®.
2. Utilizando la herramienta set de Arduino.

Con la herramienta fuzzy de Matlab® se realizaron las siguientes etapas:

Declarar los universos en discurso y los conjuntos que rigen el comportamiento de cada uno.

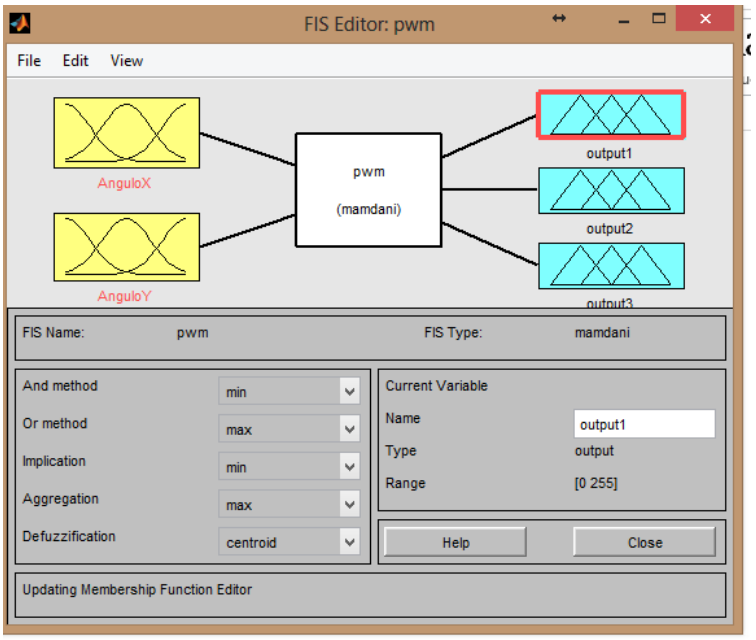


Figura 19. Control difuso del PWM en Matlab®.

Después de declarar cada uno de los universos y dichos conjuntos se procede con la creación de las reglas para lo cual se establecen los antecedentes y la acción que desata el consecuente.

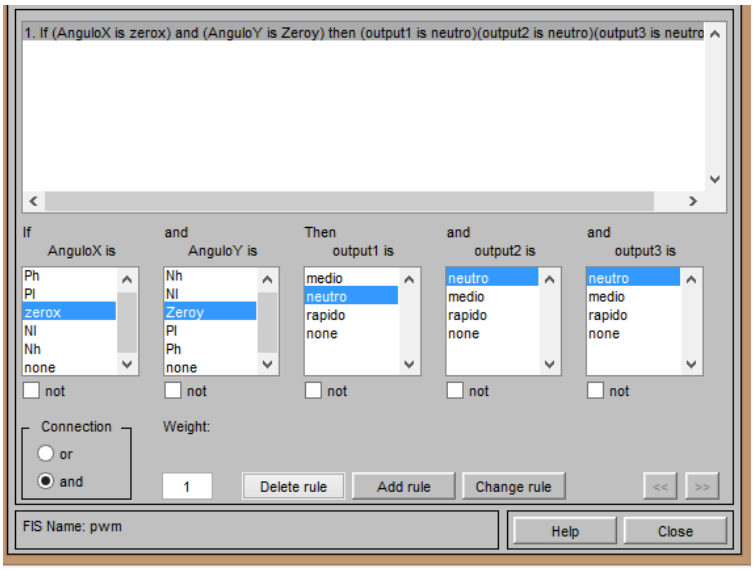


Figura 20. Declaración de reglas en Matlab®

Y se ejecuta la herramienta view rules, en donde se puede desplazar la guía por los conjuntos de entrada y salida asignándoles un valor a dichas variables y colocando los grados de pertenencia correspondientes al antecedente y a la acción relacionada con ese antecedente.

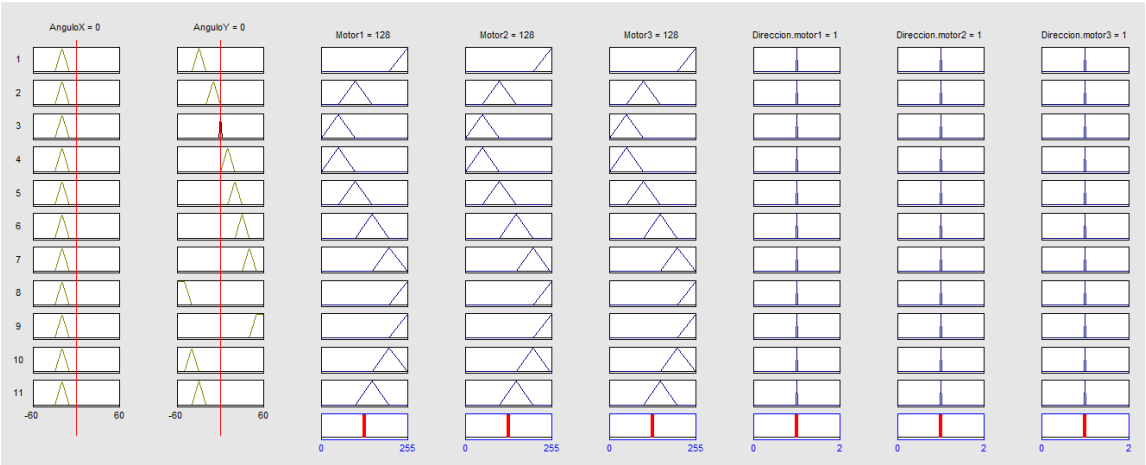


Figura 21. Tabla de verificación de reglas en Matlab®.

Con la herramienta de compilación Arduino EFLL por medio de una rutina se fijan las variables de entrada asignándoles un valor seleccionado que se encuentre dentro de los conjuntos difusos establecidos en el universo en discurso de entrada. Se da un ejemplo del código que se implementó para realizar dichas pruebas.

Asignación del valor 0 en las dos entradas para verificar la regla planteada anteriormente.

```
If (AnguloX = zerox And AnguloY=zeroy)

    fuzzy->setInput(1, 0);  //asignación de la variable zero al AnguloX
    fuzzy->setInput(2, 0); //asignación de la variable zero al AnguloY
```

Proceso de desfusificación

```
    fuzzy->fuzzify();

    float pwm1 = fuzzy->defuzzify(1);

    float pwm2 = fuzzy->defuzzify(2);

    float pwm3 = fuzzy->defuzzify(3);
```

Impresión de los datos por el puerto serial, para este ejemplo solo se imprime el valor de salida pmw1.

```
   Serial.print("ax: ");

   Serial.print(actAngleX);
   Serial.print(",   ay: ");
   Serial.print(actAngleY);
   Serial.print(",   pwm1: ");
   Serial.print(pwm1);
   Serial.print(",   pwm2: ");
   Serial.print(pwm2);
   Serial.print(",    pwm3: ");
   Serial.println(pwm3);
```

Después de que se fijan dichas variables se imprimen por el terminal serial las variables defusificadas.

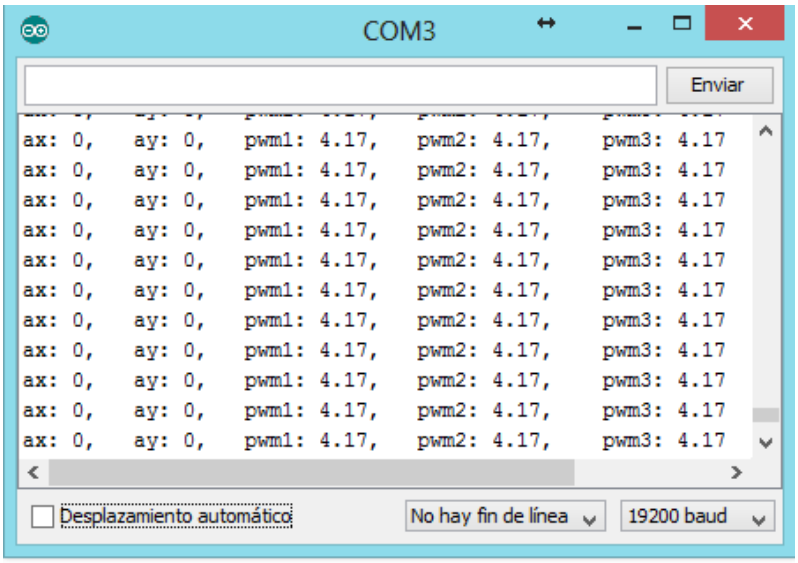


Figura 22. Salida serial del compilador Arduino.

Como se puede observar se cumple la regla, puesto que si las variables aX y aY son iguales a cero los PWM para cada uno de los motores está en el rango de valores asignados al conjunto Neutro, esto significa que el PWM de salida para cada motor es igual a 4.17 los motores están prácticamente quietos.

Las ventajas del proceso de verificación con la herramienta de compilación de Arduino es que se pueden realizar de manera más rápida la comprobación de las reglas y a su vez obtener la información en tiempo real, lo cual nos permite observar el comportamiento del sistema frente a la implementación de las reglas, mientras en el simulador Matlab se obvian muchas variables debido a que el modelo de simulación que implementa es ideal y asume procedimientos que en algunas ocasiones no se aplican en la estructura física.

6.17. Fase de transición

- División del control difuso para el PWM y control difuso para la dirección

Mientras se realizaron las pruebas por el método set de la librería eFLL, se pudo identificar que el procesador del Arduino Duemilanove no permitía realizar todos los procesos de control en paralelo, control difuso, PID, filtro, Kalman, por esta razón fue necesario realizar el procedimiento de separado de actividades en dos Arduino de la misma referencia para poder

liberar procesamiento del circuito y de esa misma manera aumentar el tiempo de respuesta de dichos procesos.

Se separan las funciones en dos Arduino Duemilanove los cuales se distribuyen de la siguiente manera:

- Arduino 1: Con lleva circuito difuso PWM, PID, filtro Kalman.
- Arduino 2: con lleva control difuso direcciones, Impresión serial IMU, Filtro Kalman.

Como podemos observar los dos Arduino con llevan el procedimiento del Filtro Kalman esto debido a que ambos controladores difusos dependen de los datos del ángulo en x como el ángulo en *y*, por ende el circuito de la IMU va conectado a los dos Arduino en paralelo. Este proceso permite mejorar el tiempo de respuesta de la planta. Este procedimiento se facilita debido a que el módulo de potencia viene separado cuatro pines son el direccionamiento del motor y 2 pines son la habilitación del motor en el cual se asigna el PWM entregado por el controlador.

Un posible inconveniente que se podría presentar en el funcionamiento de la planta con este procedimiento de división de actividades en diferentes microcontroladores, es si alguno de los dos Arduino realiza un proceso con retraso de tiempo en la salida, la planta no tendrá el mismo tiempo de respuesta en la estabilización del sistema.

Por esta razón no se implementó la comunicación serial entre los arduinos, no solo por tiempos de sincronización sino también por la fiabilidad de los paquetes enviados y los paquetes recibidos, si uno de los paquetes enviados se dé sincroniza con los paquetes recibidos cambia el proceso del controlador y su resultado no será el planteado.

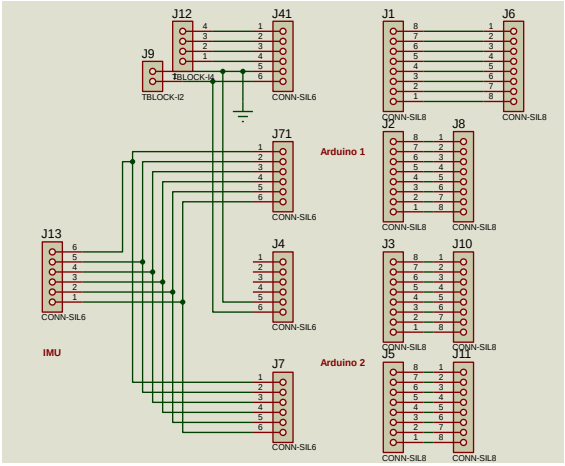


Figura 23. Esquema de conexión en paralelo de la IMU con los Arduino.

Interacción del controlador difuso con el control PID de velocidad.

El controlador difuso nos indicara cual es el valor del PWM en un rango de 8 bit de o a 255 que debe tener asignado el motor en el momento en el que se esté realizando una variación en los antecedentes esta valor PWM es ingresado al procedimiento PID el cual corregirá algún problema de disminución por fricción o rozamiento de las llantas con la esfera, este valor expresado en un PWM se inyecta en el circuito de potencia en el pin de habilitación el cual activa la velocidad de giro al motor.

6.18 Pruebas

Para la realización de pruebas se utiliza el aplicativo processing es una herramienta de código abierto GPL, la cual permite la comunicación del Arduino y el ordenador, para visualizar o guardar algunos datos recogidos por el Arduino y enviarlos por el puerto USB.

Con la herramienta processing se imprime en pantalla los datos de salida del sensor IMU y los datos de salida del filtro Kalman para poder observar la corrección del ángulo ambos en tiempo real.

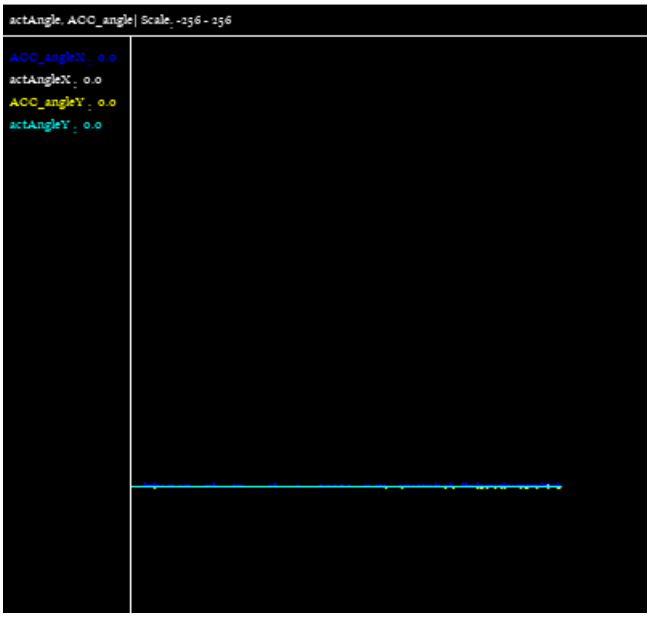


Figura 24. Grafica de salida del programa processing.

Se tienen cuatro salidas:

- La azul representa la salida del ángulo X del sensor IMU.
- La blanca representa la salida del ángulo ya procesada por el filtro Kalman.
- La azul representa la salida del ángulo Y del sensor IMU.
- La blanca representa la salida del ángulo ya procesada por el filtro Kalman.

Para poder observar la corrección de manera adecuada se ajusta la IMU para que se posicione en los valores x =-6.0 y Y=2.0 y observamos la salida del filtro Kalman.

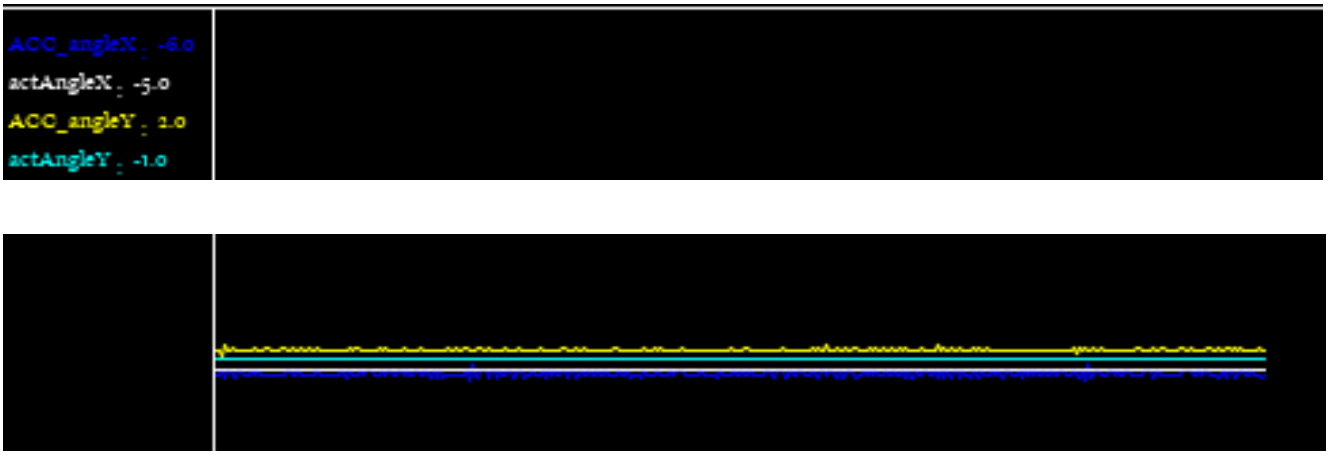


Figura 25. Grafica de angulos en procesing.

Como se puede observar en la figura 25 las salidas de la IMU presentan pequeñas distorsiones al momento de tomar el dato de inclinación del acelerómetro y el giroscopio, provocando un cambio en la magnitud del valor del ángulo. Mientras que en la salida procesada por el algoritmo del filtro Kalman se observa una corrección significativa en el ángulo el cual no tiene dicha distorsión generada por el sensor IMU. Por lo cual podemos observar la importancia del filtro Kalman en el controlador de estabilidad, permitiendo tener un número mínimo de variaciones en el sistema por señales de interferencia o ruido generado por el sensor en la entrada del controlador difuso. Ahora se aplicaran variaciones de posición en ambos ángulos para observar más en detalle la corrección del filtro Kalman.

En la figura 26 se observa la inclinación que se realiza en el ángulo Y del sensor en un lapso de tiempo de 2 segundos hasta volver a su posición inicial, se puede observar que la señal sin filtrado genera bastante distorsión variando la magnitud del ángulo frecuentemente, de igual manera se evidencia variación del sensor en el ángulo x, la cual se corrige en gran porcentaje por el algoritmo Kalman.

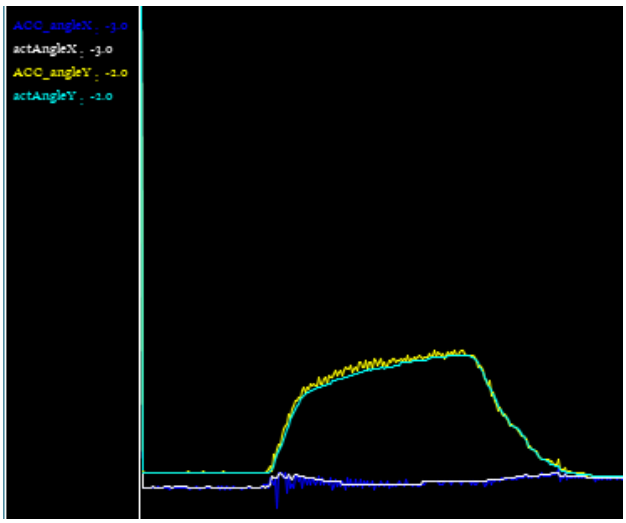


Figura 26. Variación en el ángulo Y.

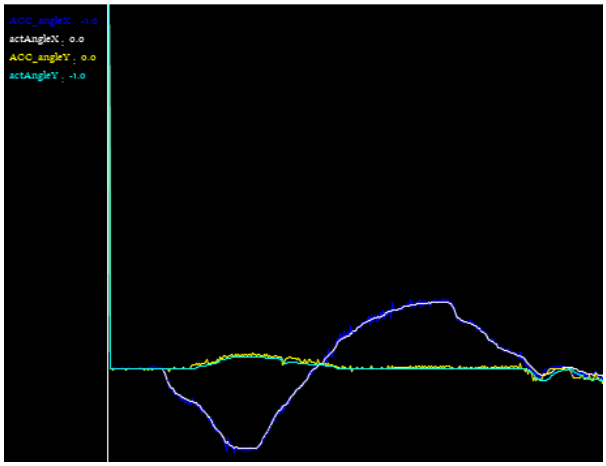


Figura 27. Variación de la posición en x en los valores positivos y negativos.

Después de comprobado el funcionamiento del sensor y el filtro Kalman los cuales van a establecer los antecedentes del comportamiento de la planta física, se realiza unas pruebas de funcionamiento sin aplicar el controlador difuso al sistema.

Esta prueba se realiza situando la estructura sobre la superficie esférica y enviándola en distintas posiciones en donde los motores no van a realizar ninguna acción, la estructura caerá en distintas direcciones sin corrección alguna y así poder observar el comportamiento en lazo abierto.

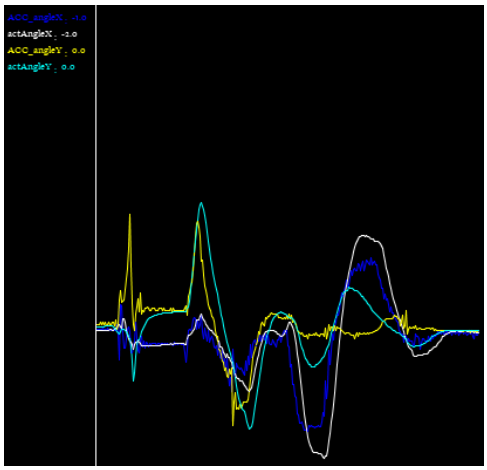


Figura 27. Grafica de las salidas de la planta en pruebas de lazo abierto.

Como se puede observar en la figura 28 las salidas graficadas en el processing, la planta presenta inestabilidad en su posición debido a que no hay actuadores mecánicos que permitan que la estructura corrija los cambios de posición de manera automática.

En donde el diseño del control difuso permite corregir dichos cambios de posición presentados en el sistema de lazo abierto, para lo cual se captura la información del filtro Kalman, se procesan estos datos en el controlador difuso, se desfusifica la salida del controlador y se inyecta esta información en los actuadores, permitiendo tomar una serie de decisiones y acciones a la planta de manera automática.

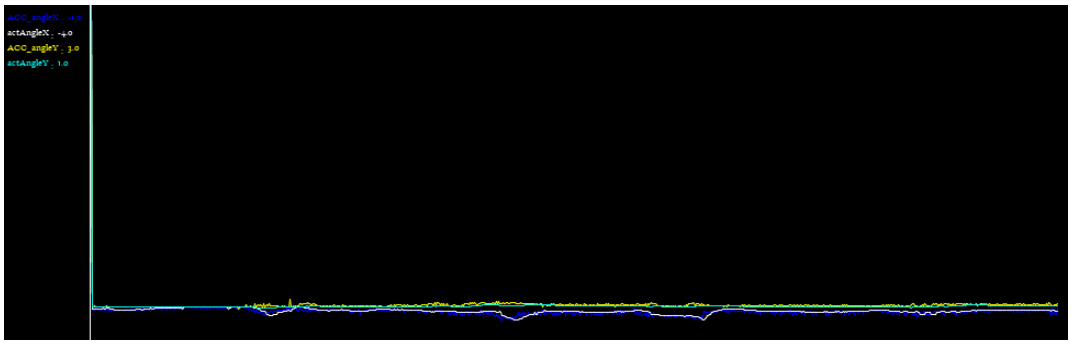


Figura 28. Respuesta del sistema en lazo cerrado.

7. CONCLUSIONES.

Se diseñó y construyó el sistema de estabilidad “Self-Balancing sobre una esfera” cumpliendo con los objetivos propuestos y los módulos básicos para su funcionamiento. Su aplicación está orientada al desarrollo de controladores difusos que permitan estabilizar plataformas no lineales que se plantean como proyectos en algunos espacios académicos de la licenciatura en electrónica. Este prototipo permite la simulación de distintos sistemas de controladores difusos, gracias a su versatilidad en la parametrización y sintonización de las variables que lo rigen.

Un aspecto importante que se obtuvo como resultado, es el comportamiento del sensor IMU, puesto que es el sensor que parametriza las variables de entrada al controlador difuso y la vez es el que determina la estabilidad de todo el sistema. Por lo que se dio a conocer la importancia de la corrección de los ángulos por medio del algoritmo del filtro Kalman y se brindan herramientas que permiten su análisis y diseño, para realizar posteriores trabajos sobre estabilización en donde aplique estos parámetros de comportamiento.

Los componentes mecánicos juegan un papel fundamental en los sistemas self-balancing debido a que la estabilización de la plataforma depende del movimiento y reacción adecuado de cada elemento que interactúa con el sistema, como lo es en este caso la superficie esférica, la cual da un grado de inestabilidad mayor al del modelo de péndulo invertido sobre dos ruedas. Por esta razón las ruedas omnidireccionales permitieron el correcto funcionamiento del proyecto planteado, permitiendo la libre movilidad de la estructura sobre la esfera, con baja perturbación por fricción.

Otro aspecto interesante a valorar, es el hecho de poder observar las características en tiempo real del sistema en lazo abierto, en donde se identifica los puntos de inestabilidad más predominantes en la planta, y de esta manera poder entablar las variables, los conjuntos y las reglas que empleara el controlador difuso para poder estabilizar la posición de la planta.

8. Referencias Bibliografías

GARCIA BREIJO EDUARDO, Compilador c ccs y simulador PROTEUS para Microcontroladores PIC. México, junio de 2008.

MIGUÉLEZ GARCÍA RUBÉN. Estudio diseño y desarrollo de una aplicación de tiempo real y de un simulador para su comprobación: péndulo invertido. Disponible en internet: http://marte.unican.es/projects/xrubenx/pendulum.pdf

MIGUEL Varga Juan y GHENZI Néstor. Introducción a la dinámica no lineal: péndulo invertido forzado. Disponible en internet: http://www.ib.cnea.gov.ar/~experim2/informes2006/info16.pdf

KUMAGAI Masaaki y GAKUIN Tohoku University. A Robot That Balances on a Ball (Un robot que balancea sobre una pelota) Disponible en internet: http://spectrum.ieee.org/automaton/robotics/robotics-software/042910-a-robot-that-balances-on-a-ball

POZO ESPIN, David Fernando, Diseño y construcción de una plataforma didáctica para medir ángulos de inclinación usando sensores inerciales como acelerómetro y giroscopio. Quito, febrero 2010

Mínguez, G (2009). Integración Kalman de sensores inerciales INS con GPS en un UAV. (Tesis de Pregrado). Universitat Politécnica de Catalunya. España.

Wang, Rizos y Li. Application of a Sigma-point Kalman filter for alignment of MEMS-IMU (artículo). University of New South Wales (sydney). Australia.