COMPARACIÓN CUANTITATIVA DE MÉTODOS DE CONTROL DE ORIENTACIÓN Y POSICIÓN PARA APLICACIONES DE PRECISIÓN DE UN CUADRICÓPTERO

JOHN FREDY CASTILLO
JESUS VIVEROS DELGADO

UNIVERSIDAD DE NARIÑO
FACULTAD DE INGENIERÍA
PROGRAMA DE INGENIERÍA ELECTRÓNICA
SAN JUAN DE PASTO
2014

COMPARACIÓN CUANTITATIVA DE MÉTODOS DE CONTROL DE ORIENTACIÓN Y POSICIÓN PARA APLICACIONES DE PRECISIÓN DE UN CUADRICÓPTERO

JOHN FREDY CASTILLO
JESUS VIVEROS DELGADO

Trabajo de grado presentado como requisito parcial para optar al título de Ingeniero Electrónico.

Director:
PH.D. ANDRÉS DARIO PANTOJA BUCHELI

UNIVERSIDAD DE NARIÑO
FACULTAD DE INGENIERÍA
PROGRAMA DE INGENIERÍA ELECTRÓNICA
SAN JUAN DE PASTO
2014

NOTA DE RESPONSABILIDAD

Las ideas y conclusiones aportadas en el siguiente trabajo son responsabilidad exclusiva del autor.

Artículo 1ro del Acuerdo No. 324 de octubre 11 de 1966 emanado del Honorable Consejo Directivo de la Universidad de Nariño.

Nota de aceptación:

_____
_____
_____
_____
_____
_____

_____
Firma del Presidente de tesis

_____
Firma del jurado

_____
Firma del jurado

San Juan de Pasto, Abril de 2014

RESUMEN

Resumiendo en este documento se presenta en la sección 1 el problema, su descripción y formulación; en la sección 2 los objetivos del proyecto son descritos, tanto el general como los específicos; en la sección 3 se presenta las bases teóricas sobre los controladores y las técnicas externas utilizadas en este trabajo; la sección 4 describe paso a paso el proceso realizado para cumplir los objetivos del proyecto; y finalmente la sección resultados muestra el consolidado dela comparación cuantitativa y la selección final, simulaciones sobre trayectorias en el espacio de este sistema de control, la implementación de algunos de los controladores diseñados y además se propone un nuevo esquema de control adaptativo basado en Replicator Dynamics a nivel de simulación.

ABSTRACT

Summarize in this document is presented in Section 1 the problem description and formulation; in section 2 the aims of the project are described, both general and specific; in section 3 the theoretical basis on drivers and external techniques used in this work is presented; Section 4 describes the step by step process performed to meet the objectives of the project; and finally the results section shows the consolidated dela quantitative comparison and final selection, simulations of trajectories in the space of this control system, the implementation of some of the drivers and also designed a new scheme based on adaptive control is proposed Replicator Dynamics simulation level.

CONTENIDO

Pág.

LISTA DE GRAFICAS

Pág.

LISTA DE TABLAS

Pág.

LISTA DE ANEXOS

Pág.

GLOSARIO

AUVs: Unmanned Aerial Vehicles (Vehículos Aéreos no tripulados) Cuadricóptero: AUV de cuatro hélices.

UKF: Unscenced Kalman Filter.

IMU: Digital Combo Board, que incluye Giroscopio y Acelerómetro

ICE: Integral del error cuadrático.

IAET: Integral del valor absoluto del error ponderado en el tiempo.

SO: Sobre impulso.

ST: Tiempo de establecimiento.

CrrFFT: Correlación del espectro.

Robustez: Capacidad de un sistema de permanecer estable.

PID: Controlador Proporcional Integral Derivativo.

MPC: Model Predictive Control.

SMC: Sliding Mode Control.

RD: Replicator Dynamics.

Roll: Angulo de inclinación sobre el eje x_b en el sistema de referencia del cuadricóptero.

Pitch: Angulo de inclinación sobre el eje y_b en el sistema de referencia del cuadricóptero.

Yaw: Angulo de inclinación sobre el eje z_b en el sistema de referencia del cuadricóptero.

Roll-rate (p): Velocidad angular alrededor del eje x_b en el sistema de referencia del cuadricóptero.

Pitch-rate (q): Velocidad angular alrededor del eje y_b en el sistema de referencia del cuadricóptero.

Yaw-rate (r): Velocidad angular alrededor del eje z_b en el sistema de referencia del cuadricóptero.

***Drifting**:* fenómeno que causa el incremento continuo del offset de una señal debido a errores de integración discreta.

$\boldsymbol{\zeta_{real}}$**:** Medida real de ángulos de inclinación.

ζ_{imu}**:** Medida de ángulos de inclinación por medio del IMU.

X,Y,Z: Coordenadas en el espacio del cuadricóptero desde el sistema de referencia terrestre.

θ, ϕ, ψ**:** Ángulos de Euler, medidos desde un norte.

V_x, V_y, V_z: Velocidades lineales vistas desde el sistema de referencia terrestre.

SGO: Subsistema general de orientación del cuadricóptero.

L, M, N, Γ:Aceleraciones angulares y lineal

d: Factor de arrastre.

b: Factor de empuje.

SPH: Subsistema de Posición Horizontal.

SPV: Subsistema de Posición Vertical.

INTRODUCCION

Unmanned Aerial Vehicles(UAVs) pueden ser muy útiles al ser usados en operaciones de búsqueda y rescate, tareas de inspección y reconocimiento.[1]. Además estos dispositivos pueden ser utilizados para el transporte de carga[2]. La evolución de estos dispositivos ha sido posible gracias a los recientes avances en electrónica, baterías, sensores y materiales[3].

Existen diversos tipos de UAVs, uno de ellos en particular es el Cuadricoptero, un vehículo de 4 hélices que utiliza las leyes fundamentales de la física para realizar sus maniobras de movimiento y estabilización. El cuadricoptero posee dinámicas no lineales que pueden convertir la tarea de controlar sus estados en experiencia muy enriquecedora. Todas estas características hacen que los UAVs, en particular los Cuadricopteros, sean aptos para trabajos de navegación a zonas de difícil acceso, reconocimiento 3D de terrenos, y puede llegar a ser una muy buena herramienta en el sector académico, ya que permite aplicar sobre él distintos métodos de control, tanto lineales como no lineales.

Este documento describe el desarrollo del proyecto investigación titulado "Comparación cuantitativa de métodos de control de orientación y posición para aplicaciones de precisión de un cuadricóptero" el cual compara y analiza el rendimiento de distintos sistemas de control con el fin de manipular los estados dinámicos de orientación y posición de un cuadricóptero. El vehículo utiliza el empuje generado por el giro de las hélices para mantenerse en vuelo y la conservación del momento angular de sus hélices opuestas girando en sentido contrario para mantenerse orientado. Su representación en ecuaciones de estado incluye 12 variables, las cuales pueden clasificarse en sub-sistemas de orientación y posición Horizontal y Vertical. Estos sub-sistemas pueden ser analizados de forma independiente hasta cierto punto, sin embargo siendo el sub-sistema de posición Horizontal dependiente directamente del sistema orientación es necesario un análisis del sistema completo para analizar sus dinámicas de forma exhaustiva.

El diseño y la validación de los controladores requieren un modelo dinámico del cuadricóptero con sus parámetros a la par con los del vehículo real a controlar, lo que hace necesaria una estimación de parámetros. En este trabajo se realiza un modelamiento teórico del vehículo y una estimación experimental de parámetros basada en Unscented Kalman Filter (UKF) y búsqueda en línea.

Existen varias técnicas de control para este tipo de vehículos en la actualidad. El proyecto tuvo como objetivo realizar una comparación del rendimiento de 4 controladores para el sub-sistema de orientación y 3 para el sub-sistema de posición y una selección de

[1] CASTILLO, Di Gennaro, and JURADO, F. "Trajectory Tracking for a Quadrotor via Fuzzy Regulation". México: Centro de Investigación y de Estudios Avanzados del I.P.N., 45015 Zapopan.

[2] SREENATH, K; LEE, T; KUMAR, V. & SO, R. (n.d.). Geometric Control and Differential Flatness of a Quadrotor UAV with a Cable-Suspended Load. Mexico: s.n. 1243000(3).

[3] PINES, D. and BOHORQUEZ, F. "Challenges facing future micro air vehicle development," EEUU: AIAA Journal of Aircraft, Vol. 43, No. 2. 2006.

2 de éstos (1 por sub-sistema) para su implementación y validación. Además se propone un sistema de ponderación de controladores a nivel de simulación basado en Replicator Dynamics con el fin de obtener un controlador el cual tome las mejores características de diferentes controladores y las fusiona.

Los diferentes controladores requieren una estimación de los estados dinámicos del vehículo, por tanto se desarrolló un sistema de Visión Artificial para la estimación de los estados del sistema trabajando en conjunto con la información proporcionada por un IMU, el cual incluye Acelerómetro y Giroscopio. Las herramientas de MATLAB y Simulink fueron utilizadas para la Simulación, Modelamiento y validación del Modelo del UAV. El montaje real del vehículo se realizó mediante una interfaz MATLAB- Xbee-Arduino.

La investigación sobre nuevos métodos de control trae un aporte significativo para los estudiantes de la Universidad de Nariño y para la comunidad investigativa en general. La existencia de un cuadricóptero controlado con precisión abre el paso a aplicaciones como navegación, reconocimiento topográfico, y reconstrucción 3D de terrenos; además con el desarrollo de este proyecto se pretende motivar a estudiantes y profesores de la Universidad de Nariño a la realización de futuros proyectos que utilicen los resultados de esta investigación.

Resumiendo en este documento se presenta en la sección 1 el problema, su descripción y formulación; en la sección 2 los objetivos del proyecto son descritos, tanto el general como los específicos; en la sección 3 se presenta las bases teóricas sobre los controladores y las técnicas externas utilizadas en este trabajo; la sección 4 describe paso a paso el proceso realizado para cumplir los objetivos del proyecto; y finalmente la sección resultados muestra el consolidado dela comparación cuantitativa y la selección final, simulaciones sobre trayectorias en el espacio de este sistema de control, la implementación de algunos de los controladores diseñados y además se propone un nuevo esquema de control adaptativo basado en Replicator Dynamics a nivel de simulación.

PROBLEMA

Descripción del problema

Para aplicaciones de gran interés investigativo en nuestra región tales como navegación, reconocimiento topográfico y reconstrucción visual 3D, es muy práctica la utilización de un vehículo aéreo: cuadricóptero. Estos modelos aéreos en la actualidad son muy comerciales y de bajo costo, características que los hacen atractivos a todo público. Sin embargo requieren sistemas de control para obtener las dinámicas deseadas.

En el laboratorio programa de Ingeniería Electrónica de la Universidad de Nariño escasean herramientas para el desarrollo de actividades prácticas que faciliten la comprensión de las teorías del Análisis de Sistemas Dinámicos y Sistemas de Control. Es por ello que la disponibilidad de un vehículo de este tipo, modelado y controlado en los laboratorios de Ingeniería electrónica, aporta en gran medida al desarrollo de conocimientos en las áreas mencionadas.

Para cualquiera de las aplicaciones del vehículo ya mencionadas, se requiere un sistema de control que lleve al cuadricóptero a comportarse de forma adecuada, y el sistema de control a su vez requiere el modelo del UAV para facilitar el diseño. En la actualidad existe gran variedad de métodos de control, tanto lineales, como no lineales. Cuando surge la idea de implementar un sistema de vuelo para alguna aplicación en particular, da cabida la duda sobre cuál de estos métodos de control utilizar para obtener resultados más precisos y un comportamiento más estable; lo que conlleva a la necesidad de realizar una comparación de diferentes métodos de control y evaluar su comportamiento ante diferentes condiciones.

Formulación del problema:

¿Qué método de Control aplicado a un cuadricóptero prototipo proporciona un mejor rendimiento respecto a precisión y robustez?

OBJETIVOS

Objetivo general

Comparar al menos tres métodos aplicados al cuadricóptero para seleccionar una metodología apropiada para controlar los estados dinámicos de posición teniendo en cuenta parámetros de desempeño tales como el ICET y IAET.

Objetivos específicos:

- Seleccionar los métodos de control apropiados para someterse a comparación en el sistema cuadricóptero con base en revisión bibliográfica de técnicas exitosas en el control de posicionamiento, trayectoria.

- Establecer el modelo matemático del cuadricóptero e identificar los parámetros principales del sistema adaptados al modelo de prueba real.

- Diseñar los controladores escogidos, tanto para posición como para orientación, para realizar una comparación cuantitativa e identificar la metodología con mejores parámetros de desempeño.

- Implementar la estrategia con mejor desempeño en el sistema de prueba real por medio de un sistema embebido y un sistema de medición inalámbrico.

1. MARCO TEORICO

1.1 MÉTODOS DE CONTROL

1.1.1 Pid. El método de control proporcional-integral-derivativo PID, es una de las estrategias de control más usadas en la actualidad por su fácil implementación y calibración, se podría decir que es un dispositivo de control genérico donde el diseñador sintoniza o ajusta unos parámetros para controlar un sistema. Se dice que más de la mitad de los controladores industriales usan el modelo PID o algunas de sus posibles modificaciones[4], haciendo de esta estrategia una de las más confiables al momento de implementar un método de control. "En general el controlador PID es un compensador en adelanto y retraso, donde el compensador de adelanto es proporcional derivativo y el compensador de retraso es proporcional integral.Llo importante es que se pueden obtener un controlador del producto de estos dos compensadores, con la característica de poseer dos raíces reales o imaginarias, un polo en el origen y una ganancia"[5].

Considere el siguiente diagrama de bloques de un sistema SISO (una entrada una salida) como el de la figura 1.

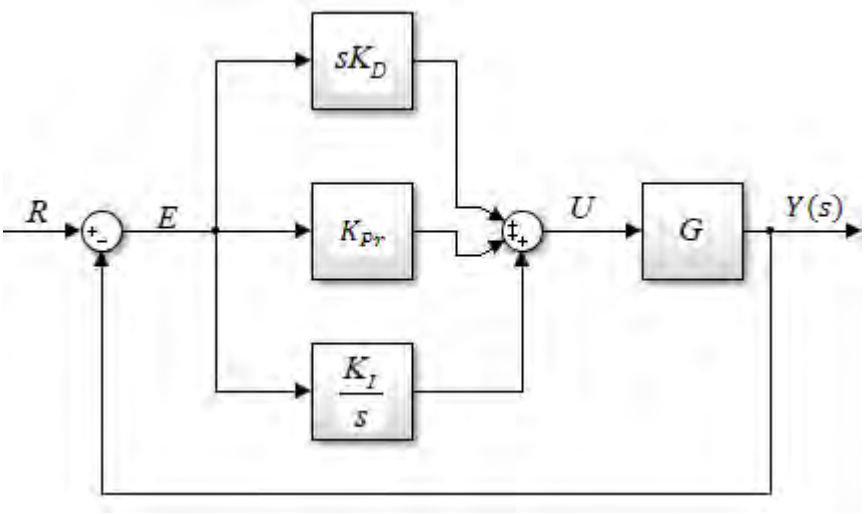


Figura 1. Diagrama de bloques PID.

La señal de control $U(s)$ del PID está definida de la siguiente manera:

$$\boldsymbol{U(s) = \left(K_{Pr} + \frac{k_I}{s} + sK_D\right) E(s) \leftrightarrow u(t) = K_p\left(e(t) + \frac{1}{T_I}\int_0^t e(\tau)d\tau + T_D\frac{d(e)}{dt}\right)}$$

Ecuación 1. Señal de Control para PID.

[4] OGATA, K. "Ingeniería de Control Moderna". 3ra ed. Minnesota: Pearson Educación, 2000.
[5] GIL, J. DÍAZ, A. "Ingeniería de Control, Control de Sistemas Continuos". EEUU: Universidad De Navarra, Escuela Superior de Ingenieros, Unicopia C.B, 2004.

Donde k_{Pr} es una ganancia ajustable que da una salida con mayor rapidez, pero con más oscilaciones al controlador y es proporcional al error, K_I es el parámetro del controlador relacionado con la integral del error acumulado, lo que implica un modo de controlar el error de estado estacionario y K_D es la está relacionada con el tiempo derivativo del error y tiene un carácter de eliminar las oscilaciones de sobrepaso aplicando un sobre-amortiguamiento a la señal de salida en el menor tiempo posible.

Al combinar todas las características de las constantes del PID podemos sintonizar el controlador de manera que se puede llegar a una señal de referencia en poco tiempo, con poco sobrepaso y sin error de estado estacionario.

Existen diferentes métodos para sintonizar un controlador PID, es decir encontrar los mejores valores para K_P, T_I y T_D con las cuales se cumplen las especificaciones de desempeño; Ziegler-Nichols propusieron ciertas reglas para determinar el valor de estos parámetros basándose en la respuesta transitoria del sistema, se considera un sobrepaso del 25% para la respuesta a la señal de referencia escalón. El primer método que sugieren Ziegler-Nichols es basar los cálculos de K_P, T_I y T_D mediante una tabla de valores basada en el tiempo de retardo L y la constante de tiempo T los cuales se calculan trazando una recta tangente en el punto de inflexión de la curva de respuesta al escalón unitario. Otro método sugerido por los mismos autores es calcular los parámetros del PID determinado experimentalmente una ganancia crítica K_{CR} y un periodo crítico P_{CR} para los cuales la salida presenta oscilaciones sostenidas.

Un método analítico de sintonizar los parámetros del PID consiste en ubicar los polos del sistema en lazo cerrado en posiciones que garanticen un comportamiento adecuado en estado estacionario. Para esto se obtiene la función de transferencia del sistema en lazo cerrado y se iguala el denominador de esta función a otro que contiene los polos deseados para calcular K_P, T_I y T_D que satisfacen la ecuación. Cabe mencionar que la sintonización de los parámetros para PID se puede hacer manualmente.

En el caso de un cuadricóptero el sistema es MIMO, pero es posible utilizar PID en los subsistemas propuestos ya que los pares entrada salida funcionan en forma independiente. La equivalencia de la ecuación teórica de $U(s)$ usando el paquete de Matlab es:

$$\boldsymbol{U(s) = P + I\frac{1}{s} + D\frac{N}{1 + N\frac{1}{s}}}$$

Ecuación 2. Ecuación PID usada en Matlab.

Donde $P = K_{pr}$,$I = K_I$, $D = K_D$ y Nes el coeficiente del filtro pasabajos para posibilitar la implementación de la acción derivativa.

1.1.2 MPC (Model Predictive Control). MPC es un método basado en algoritmos que computa una serie temporal discreta de variables tratadas a futuro con el objetivo de optimizar las dinámicas propias de un sistema el cual se quiere controlar. La estrategia MPC requiere el modelo de la planta para prever su comportamiento, la optimización se

basa en la evolución predictiva del comportamiento de la planta y el control es aplicable a sistemas multi-variable complejos con restricciones.

Este método ha sido diseñado teniendo en cuenta los ideales de predecir las salidas del proceso del sistema en futuros instantes del tiempo, obtener unas señales de control a partir de una optimización de una función objetivo y utilizar una estrategia deslizante para desplazar cada instante hacia el futuro.

MPC consta de un modelo predictivo que constituye el elemento principal del método, a su vez éste se complementa con un modelo de proceso y un modelo de perturbaciones que incluye efectos del ruido, errores de modelado y entradas no medibles.[6]

Los modelos de procesos más comunes son los que se describen a continuación:

FIR está fundamentado en la respuesta impulsional finita, pero tienen la limitante de que los procesos del sistema tienen que ser estables. La relación entre las variables de entrada y salida del sistema la rige la siguiente ecuación (con t discreta):

$$\boldsymbol{y(t) = \sum_{i=1}^{N} h_i u(t-i) = H(Z^{-1})u(t)}$$

Ecuación 3. Relación entre las variables de entrada y salida del sistema FIR.

Donde los valores de h_i son calculados a partir de la medición de las variables de salida, cuando se aplica un impulso unitario de amplitud igual a la del tempo de muestreo.

Paralelo al anterior modelo de proceso está el que se aplica una señal de escalón unitario a la entrada para relacionar las variables de entrada y salida por medio de la expresión:

$$\boldsymbol{y(t) = y_0 + \sum_{i=1}^{N} g_i \Delta u(t-i) = y_0 + H(Z^{-1})\Delta u(t)}$$

Ecuación 4. Relación entre las variables de entrada y salida del sistema con entrada escalón.

Donde $\Delta = (1 - Z^{-1})$, los coeficientes g_i se obtienen midiendo la salida ante la entrada del escalón unitario y y_0 es el offset que comúnmente es nulo.

También están los modelos de proceso que utilizan las funciones de transferencia para relacionar las variables de salidas con entradas así:

$$\boldsymbol{y(t) = \frac{B(Z^{-1})}{A(Z^{-1})} u(t)}$$

Ecuación 5. Relación entrada salida con funciones de transferencia.

[6] RODRÍGUEZ, D. "Perspectiva General del Control Predictivo" Tesis Doctoral. Bogotá: s.n. s.f.

$$\text{Con } B(Z^{-1}) = b_1 Z^{-1} + b_2 Z^{-2} + \cdots + b_{nb} Z^{-nb}$$
$$A(Z^{-1}) = a_1 Z^{-1} + a_2 Z^{-2} + \cdots + a_{na} Z^{-na}$$

Otros modelos de proceso se caracterizan por usar espacios de estados y modelos no lineales como redes neuronales, fuzzy etc.

El modelo para las perturbaciones más común es el denominado *Controlled Auto-Regressive and Integrated Moving Average* [6] en donde las salidas del proceso y la calculada mediante el modelo está dada por (con t discreta):

$$\boldsymbol{n(t) = \frac{C(Z^{-1})e(t)}{D(Z^{-1})}}$$

Ecuación 6. Modelo de relación entrada salida con perturbaciones.

Donde $D(Z^{-1})$ incluye un integrador de la forma $\Delta = 1 - Z^{-1}$, $e(t)$ es un ruido blanco de media cero y $C(Z^{-1})$ es un polinomio que por lo general es uno; el resultado es una señal de control capaz de rechazar perturbaciones.

Se conoce que una de las limitantes de MPC tiene que ver con obtener un modelo preferiblemente lineal del sistema; todo esto con el fin de asegurar la validez de la robustez y convergencia del MPC, además de hacer rápido el cálculo computacional. Esta es una condición requerida en estos modelos, ya que estos cálculos se basan en la resolución de problemas de optimización con restricciones usando métodos numéricos aplicables a sistemas lineales. Por lo tanto se puede concluir que MPC requiere precisión en el modelo lineal para poder predecir con más confiabilidad las señales de control requeridos en el proceso de control.

1.1.3 Controlador fuzzy. La técnica de lógica difusa o Fuzzy está basada en la inteligencia artificial y su funcionamiento se basa en expresar el conocimiento lingüístico cualitativo en una expresión matemática regido por la teoría de conjuntos difusos y funciones de pertenecía asociados a estos[7].

Se ha demostrado que aplicar la técnica Fuzzy a problemas de control no lineal presenta cualidades en cuanto a la flexibilidad en las imprecisiones del sistema ya que se basa en un lenguaje natural de aprendizaje. Otra ventaja es que cuando los datos no son muy precisos por la complejidad del sistema o por distorsiones de ruido y disminuye la capacidad de realizar afirmaciones precisas, Fuzzy permite asignar funciones de membresía adecuada que corrigen este problema.

Para este método se define un conjunto difuso o borroso el cual es un conjunto donde la pertenencia de sus elementos se define mediante una función de membresía o de pertenencia, ésta determina si un valor pertenece totalmente, parcialmente o se excluye del conjunto (figura No 2), la diferencia con otros métodos es que los valores intermedios pueden ser considerados como miembros del conjunto borroso.

[7] MOROTA, F. “Controlador Fuzzy de un Quadrotor”. Tesis Doctoral. Madrid: Universidad Complutense de Madrid, 2009.

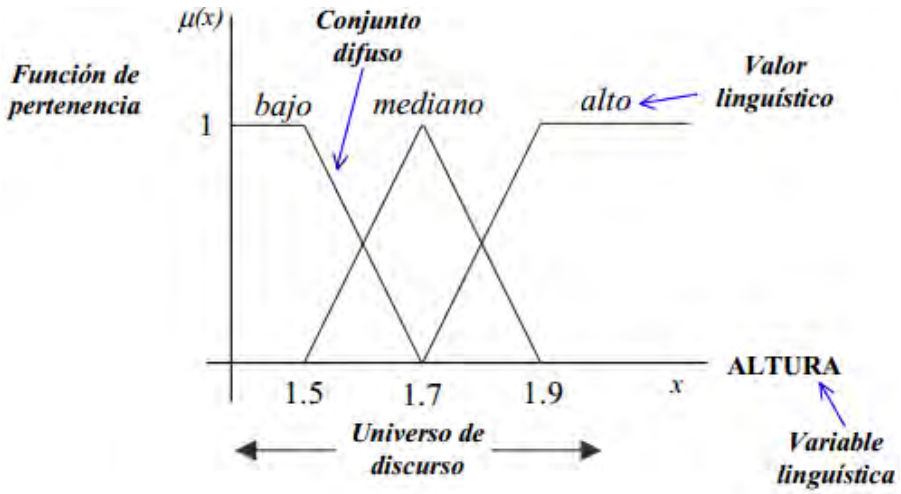


Figura 2. Función de Membresía para un conjunto difuso.

Las funciones de membresía más usadas por su estructura lógica para definir su valor lingüístico asociado, son la función gamma (Γ), L, triangular, trapezoidal, S y Z entre otras.[8]

Para desarrollar un control basado en Fuzzy lo que es necesario asignar reglas con premisas con un cierto grado de veracidad y cuantificadas por las funciones de membresía, a las variables de entrada para modificar las variables de salida que a su vez afectaran las señales de control de planta, esta es su principal diferencia con los otros métodos basados en modelos matemáticos rigurosos para tomar una decisión de control. Las reglas o funciones de decisión son del tipo *SI condición ENTONCES acción* y están basadas en el conocimiento de los expertos que serían personas que conocen muy bien el funcionamiento del sistema.

El primer paso del procedimiento seria calcular el grado de pertenencia de las variables involucradas en el proceso de control, a los conjuntos difusos que se han seleccionado. Esto se conoce como fuzzificacion, que se resumiría en cuantificar los conjuntos borrosos por medio de las funciones de membresía. El segundo paso es obtener el grado de verdad o *peso* para cada una de las reglas basado en la veracidad de sus antecedentes. Ahora a cada salida también le corresponde una función de membresía y a cada una de ellas se les asigna un grado de aplicabilidad que corresponde al máximo valor entre todas las reglas que la mencionan para que al finalizar toda la evaluación de reglas las salidas tengan un valor. Finalmente se hace un promedio de todas las salidas correspondientes a cada variable, debido a que el procedimiento genera más de una salida para cada variable de entrada[9].

1.1.4 SMC (Sliding Mode Control). El control de Modos Deslizantes (Sliding Mode Control) es una estrategia diseñada con el fundamento de estructura variable en

[8] Ibid.

[9] GÓMEZ, J. "Fuzzy Control". Bogotá: Universidad Tecnológica Nacional – FRBA, s.f.

realimentación de estados que cambia el comportamiento de un sistema no lineal forzando una señal de control bivalente. SMC se ha diseñado para ser robusto y versátil ante sistemas con un modelo matemático muy complejo.

La idea básica de este tipo de controlador es manipular las variables de estado de un sistema, de modo que estos se muevan en dirección a una superficie de deslizamiento $s(t) = 0$ que es la referencia del sistema de control. Para lograr esto, el sistema es forzado con una señal de control discontinua, lo que empujará las trayectorias del sistema en la dirección de la superficie de deslizamiento, las trayectorias del sistema oscilará alrededor de la misma superficie y la amplitud de las oscilaciones es mucho más pequeña cuanto mayor es la frecuencia de la señal de control[10].

Para el diseño de SMC hay que considerar dos principios básicos, el primero es la elección de una superficie de deslizamiento $s(t)$ a la cual las trayectorias del sistema deben converger ya que el comportamiento del sistema en realimentación depende de dicha superficie y en segundo lugar se debe considerar elegir una ley de control en función de la superficie de deslizamiento $s(t)$.

Considere un sistema de segundo orden como el que se muestra a continuación[11]:

$$\dot{x}_1 = x_2$$
$$\dot{x}_2 = h(x) + g(x)u$$

Ecuación 7. Sistema de segundo orden.

Con h y g funciones no lineales desconocidas. Se desea diseñar una ley de control que limite el movimiento del sistema sobre una superficie $s(t) = ax_1 + x_2 = 0$, donde el movimiento es gobernado por $\dot{x}_1 = -a_1 x_1$, como la idea es llevar las trayectorias del sistema a la superficie $s(t)$ y mantenerlas ahí, entonces se debe cumplir que:

$$\dot{s} = a_1\dot{x}_1 + \dot{x}_2 = a_1 x_2 + h(x) + g(x)u$$

Ecuación 8.

La anterior ecuación requiere que h y g satisfagan la siguiente inecuación:

$$\left|\frac{a_1 x_2 + h(x)}{g(x)}\right| \leq k_1$$

Ecuación 9.

Se necesita conocer el valor $k_1$que satisfaga la inecuación de tal manera que la superficie de deslizamiento sea independiente de h y g, cumpliendo criterios de estabilidad de

[10] KHALIL, Michigan. “Nonlinear System” State University. 3ra ed. México: Pearson Educación, 2000

[11] Ibíd.

Lyapunov lo cual coincide con una característica propia de SMC. Es robusto con respecto a estas funciones.

Las funciones $h(x)$ y $g(x)$son del modelo a controlar y la señal de control generalmente se puede tomar como:

$$u = -ksing(s)$$
$$\text{Con } k > k_1.$$

Ecuación 10. Señal de control para Sliding Mode Control.

1.1.5 Replicator Dynamics. Es un método de la teoría evolutiva de juegos, en donde se describe la evolución de una población donde cada individuo perteneciente a dicha población, cambia su comportamiento o algún rasgo característico, por imitación a otros individuos que se consideran con un mejor fitness. Es una descripción de la evolución de las frecuencias de las estrategias que se adoptan en una población, provee un apoyo dinámico al concepto de estabilidad evolutiva bajo la suposición de que se puede describir el sistema de forma continua[12].

Si una población está dividida en n tipos, los cuales cada uno posee una frecuencia de estrategia x_i dentro de la población global, además consideremos que cada tipo tiene su función de fitness $f_i(\vec{x})$, entonces se puede pensar en que cada generación se puede mezclar de tal forma que $\vec{x}(t)$ puede evolucionar de manera diferenciable, es decir que para cada tipo tiene una tasa de crecimiento así:

$$\frac{\dot{x}_i}{x_i}$$

Ecuación 11. Tasa de crecimiento para cada tipo de la población.

La anterior ecuación representa el éxito reproductivo y en consecuencia de su fitness en relación con el fitness de la población dado por:

$$\frac{\dot{x}_i}{x_i} = fitness_i - fitness\ media = f_i(\vec{x}) - f(\vec{x})$$

Ecuación 12. Relación tasa de crecimiento y Fitness.

De donde despejando $\dot{x}_i$ podemos reescribir como:

$$\dot{x}_i = x_i([f(\vec{x}) - f(\vec{x})]$$

Ecuación 13. Ecuación del replicador.

[12] ABRAMSON, G. "Introducción a la Teoría de Juegos". México: Centro Atómico Bariloche, Instituto Balseiro, 2006.

1.2 UNSCENTED KALMAN FILTER (UKF)

UKF pertenece a una clase general de filtros llamados Sigma-Point Kalman filters o Linear Regression Kalman Filters, los cuales hace uso de la técnica de linealización estadística[13]. Esta técnica es usada para linealizar un sistema no-lineal de una variable aleatoria a través de una regresión lineal entre n puntos escogidos de la distribución a priori de la variable aleatoria y así obtener una estimación de los estados futuros del sistema. Debido a que se considera la propagación de la variable , la técnica tiende a ser más precisa que la Linearizacion de Taylor.

El filtro extetendido de Kalman (EKF (Extended Kalman Filter)) es utilizado para estimación de estados de sistemas no lineales al igual que UKF, sin embargo, en EKF la distribución del estado es propagada analíticamente a través de la lineraizacion de primer orden del sistema no lineal, razón por la cual las covarianzas y medias podrían verse corruptos[14]. UKF es una alternativa libre de derivadas, supera este problema usando un enfoque de muestreo determinístico. La distribución del estado está representada usando un minimo conjunto de muestras escogidas cuidadosamente, llamadas sigma points. UKF al igual que EKF y KF consiste en los mismos dos pasos: Pronosticación de estado, y asimilación de datos, excepto que para UKF estos pasos están precedidos por otro encargado de la selección de los sigma points.

UKF está basado en la intuición de que es más fácil aproximar una distribución probabilística que aproximar una función o transformación no lineal arbitraria. Los sigma points son escogidos tal que su media y covarianza sean exactamente x_{k-1}^{a} y P_{k-1}. Luego cada sigma point es propagado a través de la nolinearidad, produciendo una nube puntos transformados. Las nueva media y covarianza son calculadas basadas en sus estadísticas. Este proceso es llamado Unscented Transformation que es un método para el cálculo de estadísticas de una variable aleatoria que experimenta una transformación no lineal.

Para ilustrar el proceso de estimación de estados mediante UKF considérese el siguiente sistema no-lineal, descrito por las ecuaciones de diferencias y el modelo de observación con ruido aditivo.

$$x_k = f(x_{k-1}) + w_{k-1}$$
$$z_k = h(x_k) + v_k$$

Ecuación 14. Modelo no lineal del sistema x_k y modelo de observación con ruido z_k.

El proceso de UKF consta de una serie de etapas que son aplicadas en cada ciclo de funcionamiento de un sistema de estimación de estados e iterativamente sobre el conjunto de mediciones para la estimación de parámetros asi:

---

[13] TEREJANU, G. Unscented Kalman Filter tutorial. Department of Computer Science and Engineering. Tomado de http://www.cse.sc.edu/~terejanu/files/tutorialUKF.pdf. 2011.

[14] Ibíd.

El estado inicial $x_0$es un vector aleatorio con media conocida $\mu_0 = E[x_0]$ y covarianza $P_0 = E[(x_0 - \mu_0)(x_0 - \mu_0)^T]$.

Selección de los *Sigma Points:*

Sea X_{k-1} el conjunto de $2n + 1$ sigma points (donde n es la dimensión del esapcio de estados) y sus pesos asociados:

$$X_{k-1} = \{(x_{k-1}^j, W^j) | j = 0 \dots 2n\}$$

Ecuación 15. Conjunto de puntos aleatorios.

Considérese la siguiente selección de Sigma Points. Selección que incorpora la información de mayor orden en los puntos seleccionados.

$$x_{k-1}^0 = x_{k-1}^a$$
$$-1 < W^0 < 1$$
$$x_{k-1}^i = x_{k-1}^a + \left(\sqrt{\frac{n}{1 - W^0} P_{k-1}}\right)_i \; para\ todo\ i = 1 \dots . n$$
$$x_{k-1}^{i+n} = x_{k-1}^a - \left(\sqrt{\frac{n}{1 - W^0} P_{k-1}}\right)_i \; para\ todo\ i = 1 \dots . n$$
$$W^j = \frac{1 - W^0}{2n} \; para\ todo\ j = 1 \dots 2n$$

Ecuación 16.

Donde los pesos deben satisfacer la condición:

$$\sum_{j=0}^{2n} W^j = 1$$

W^0 controla la posición de los sigma points: con $W^0 \geq 0$ los puntos tienden a alejarse del origen, mientras que en el caso contrario tienden a acercarse. Información general sobre la selección de los sigma points puede encontrarse en[15].

Pronosticación del modelo:

Cada sigma point es propagado a través del modelo del proceso no lineal:

$$x_k^{f,j} = f\left(x_{k-1}^j\right)$$

Ecuación 17.

Los puntos transformados son usados para calcular la media y la covarianza de la pronosticación de x_k:

[15] WAN, E. and VAN, R. The Unscented Kalman Filter. EEUU: Wiley Publishing, 2001

$$x_k^f = \sum_{j=0}^{2n} W^j x_k^{f.j}$$

Ecuación 18.

$$P_k^f = \sum_{j=0}^{2n} W^j \left(x_k^{f,j} - x_k^f\right)\left(x_k^{f,j} - x_k^f\right)^T + Q_{k-1}$$

Ecuación 19.

Seguidamente se propaga los sigma points a través del sistema no-lineal de observación:

$$z_{k-1}^{f,j} = h(x_{k-1}^j)$$

Ecuación 20.

Y se calcula su media y covarianza asi:

$$z_{k-1}^f = \sum_{j=0}^{2n} W^j z_{k-1}^{f.j}$$

Ecuación 21.

$$\boldsymbol{cov}\left(\tilde{\boldsymbol{z}}_{\boldsymbol{k-1}}^{\boldsymbol{f}}\right) = \sum_{\boldsymbol{j=0}}^{\boldsymbol{2n}} \boldsymbol{W}^{\boldsymbol{j}} \left(\boldsymbol{z}_{\boldsymbol{k-1}}^{\boldsymbol{f,j}} - \boldsymbol{z}_{\boldsymbol{k-1}}^{\boldsymbol{f}}\right)\left(\boldsymbol{z}_{\boldsymbol{k-1}}^{\boldsymbol{f,j}} - \boldsymbol{z}_{\boldsymbol{k-1}}^{\boldsymbol{f}}\right)^{\boldsymbol{T}} + \boldsymbol{R}_{\boldsymbol{k}}$$

Ecuación 22.

La covarianza cruzada entre $\tilde{x}_k^f$ y $\tilde{z}_{k-1}^f$ es:

$$cov(\tilde{x}_k^f, \tilde{z}_{k-1}^f) = \sum_{j=0}^{2n} W^j \left(x_k^{f,j} - x_k^f\right)\left(z_{k-1}^{f,j} - z_{k-1}^f\right)^T$$

Ecuación 23.

Asimilación de datos:

El siguiente paso consiste en combinar la información obtenida en la etapa de pronosticación, con la nueva medida z_k. Se asume, al igual que en el filtro Kalman, que la estimación de estados tiene la siguiente forma:

$$x_k^a = x_k^f + K_k(z_k - z_{k-1}^f)$$

Ecuación 24.

Donde la ganancia de Kalman K_k esta dada por:

$$K_k = cov(\tilde{x}_k^f, \tilde{z}_{k-1}^f\ )cov^{-1}(\tilde{z}_{k-1}^f\ )$$

Ecuación 25. Calculo de ganancia de kalman.

La actualización de la covarianza se calcula en seguida a partir de la siguiente expresión:

$$P_k = P_k^f - K_k\ cov(\tilde{z}_{k-1}^f\ )K_k^T$$

Ecuación 26. Actualización de la covarianza.

Este proceso se repite en cada iteración de un sistema de estimación de estados. Para el caso de la estimación de parámetros, el proceso se repite hasta alcanzar la convergencia de los parámetros a estimar. El método es útil bajo las condiciones de este trabajo debido a que los sistemas de estimación de parámetros presentan un comportamiento no lineal y utilizar técnicas basadas en linealización como EKF puede resultar en estimaciones poco precisas.

2. DESARROLLO DEL TRABAJO

El desarrollo del trabajo cada incluye cada etapa del proceso para establecer el control del vehículo escogido:

- Construcción de un cuadricóptero prototipo
- El diseño e implementación de un Sistema de Visión artificial que permita estimar los estados dinámicos del vehículo
- La estimación de parámetros, obtención y validación del modelo del prototipo
- El diseño, simulación, comparación cuantitativa y selección de los distintos métodos de control seleccionados
- La implementación real del sistema de control definitivo
- La proposición de un sistema de ponderación de controladores basado en Replicator Dynamics a nivel de simulación.

2.1 CONSTRUCCIÓN DEL PROTOTIPO

2.1.1 Estructura del cuadricóptero. La estructura del cuadricóptero está compuesta por 4 soportes metálicos cuadrados; en uno de los extremos de cada uno de ellos se fija el motor y el otro extremo se asegura al chasis del vehículo. Cada uno de los motores consta de elementos que permite sujetar las hélices a su eje.

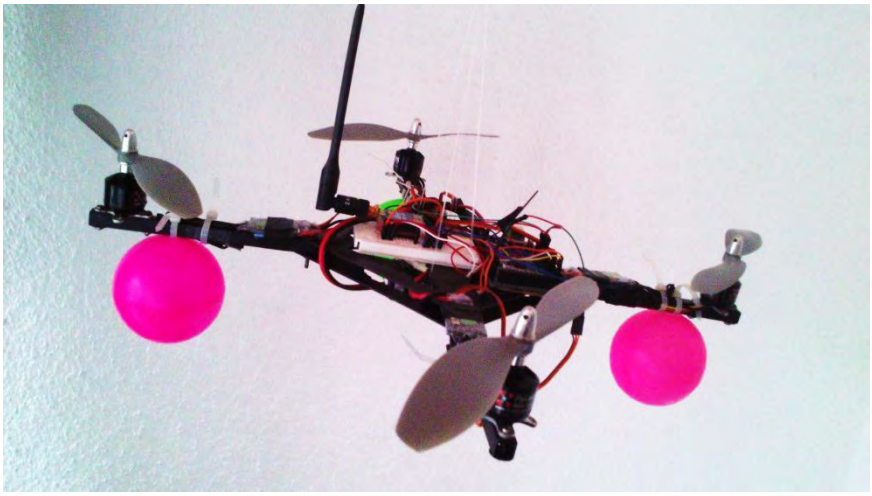


Figura 3.Construcción del Prototipo.

En la parte superior del chasis se ubicó una board donde se insertó previamente el IMU, el dispositivo de comunicaciones (Xbee) y el sistema de procesamiento (embebido Arudino). En la parte inferior del chasis se encuentra la batería Lipo y la conexión eléctrica principal a los motores. El cableado se distribuye de forma conveniente a lo largo de la estructura.

Cabe anotar que para distribuir la corriente desde la batería Lipo a los motores, se usó cable termo-resistente de alto amperaje para prevenir posibles recalentamientos en el conductor, lo que podría provocar cortocircuitos puesto que la corriente máxima a circular será de aproximadamente 12 amperios(3A por motor).

2.1.2 Componentes de la estructura:

Motores Brushless: Son motores eléctricos que se diferencian de los demás por no usar escobillas para realizar el cambio de polaridad en el rotor, ni colector, lo cual evita inconvenientes como recalentamiento, rozamiento entre componentes y partículas de carbón entre otras que disminuyen el rendimiento del motor. Los motores Brushless están conformados por un rotor móvil donde se encuentran los imanes permanentes y el estator que es la parte fija donde están los embobinados de hilo conductor. La corriente pasa directamente por los embobinados, generando un campo electromagnético que interactúa con el campo magnético generado por los imanes del rotor, provocando una fuerza que hace girara el eje del rotor. Para garantizar que el rotor gire, sea cual sea su posición, se cuenta con un variador eléctrico que verifica la posición del rotor en cada momento para comprobar que la corriente que es suministrada, sea la adecuada para mantener en giro al rotor; dicho variador procesa la información de sensores que están monitoreando el comportamiento de la corriente del rotor en tiempo real para controlar el rotor[16].

Un parámetro importante es el denominado *kV(kiloVolt)*, el cual indica el número de revoluciones por minuto que puede girar el motor, por cada voltio de electricidad que se aplique, pero hay que considerar que a mayores valores de este parámetro menores valores de par se obtienen.

Figura 4. Motor Bruhsless.

Soportes metálicos: estos soportes son tubos cuadrado hechos en aluminio, sus dimensiones son de 1 cm x 1 cm x 19 cm, en uno de sus extremos se ubica el motor Brushless y en el otro se sujeta al cuerpo del montaje.[17]

[16] Disponible en Internet: http://www.kkmulticopter.kr/

[17] Ibíd.

Figura 5. Soporte Metálico.

Hélices plásticas: están hechas de material de plástico resistente, tienen una longitud de 10 pulgadas, paso de 4.7 pulgadas, un adaptador de 4,5 mm y son del tipo propulsor. Para el montaje completo del cuadricóptero se necesitan cuatro hélices dos que giran en sentido anti horario y otras dos en sentido horario[18].

Figura 6. Hélices plásticas del cuadricóptero.

Batería Lipo: Son una variación de las baterías de iones de litio. Sus características son muy similares, pero permiten una mayor densidad de energía, así como durabilidad en la carga mucho superior a las baterías comunes. Otro factor determinante de estas baterías es tamaño, mucho más reducido respecto a las de otros componentes.

Por lo regular cada celda tiene un voltaje nominal de 3.7 V, voltaje máximo 4.2 y mínimo 3, este último debe respetarse rigurosamente ya que la pila se daña irreparablemente a voltajes inferiores a 3 voltios.

Sus principales ventajas están relacionadas con la mayor densidad de carga, lo que se refleja en un menor tamaño de la batería y buena tasa de descarga. Como desventaja cabe mencionar la inutilidad de la batería al si se descarga por debajo de los 3 voltios.

[18] Ibíd.

Figura 7. Batería LIPO utilizada en cuadricóptero

Controlador ESC: Es un dispositivo electrónico con microcontrolador que sirve para controlar la velocidad del motor brusless. Cuenta con terminales para la entrega de potencia al motor y para alimentación desde la batería. También posee 3 pines de los cuales 2 de ellos se usan para recibir la señal de control desde Arduino[19].

El funcionamiento del ESC consiste en entregar una potencia al motor Brushless proporcional a la duración de un tren de pulsos entre 1060uS-1500uS a una frecuencia de 50Hz.

Figura 8. Controlador de velocidad para motor Brushless (ESC).

2.1.3 Sistema de procesamiento y transmisión de datos. El embebido Arduino fue utilizado para ejecutar labores de procesamiento de información proveniente de sensores de orientación, y el sistema de visión artificial implementado en MATLAB. Además se utilizó un sistema comunicación basado en Xbee que sirve de puente para el intercambio de datos entre MATLAB y Arduino. El embebido también está encargado del cálculo y la ejecución de señales de control.

Módulo XBee: Son instrumentos de comunicaciones inalámbricas para transmisión de datos. Los módulos XBee poseen dos formas de comunicación: transmisión serial modo AT y el modo API. El modo AT se caracteriza por ser transparente al usuario, razón por la cual facilita el envío y la recepción de datos, sin embargo el modo API permite transmisión a mayor velocidad. Los módulos Xbee pueden ser configurados desde el PC utilizando el programa X-CTU o desde microcontrolador. Algunas sus principales características son[20]:

- Alcance: hasta 1600 mts en línea vista.

[19] Ibíd.

[20] Ibíd.

- 6 entradas ADC de 10 bits
- 8 entradas/salidas digitales.
- Potencia de salida 17 dBm.
- Alimentación 3.3v a 295 mA.
- Interfaz serial.

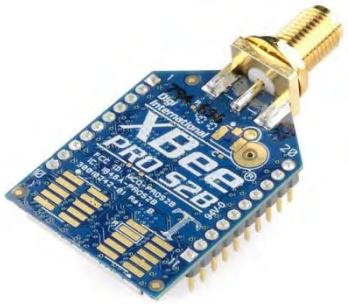


Figura 9. Módulo XBee para comunicación inalámbrica.

Arduino Mega 2560: El embebido Arduino Mega 2560 fue utilizado para servir de última interfaz entre los controladores y los motores. El Arduino permanece acoplado al vehículo y su tarea consiste en:

- Obtener la información de medición de estados de orientación desde el IMU y enviarlos al PC para procesamiento.
- Aplicar la señal de control hacia los motores mediante pulsos PWM entre 1060uS y 1500uS
- Recibir información sobre referencias y datos de medición de estados enviados desde el PC (Sistema de Visión Artificial) mediante comunicación XBee.

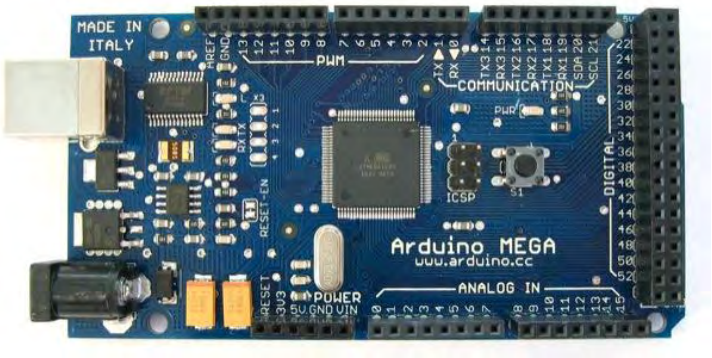


Figura 10. Arduino Mega 2560.

2.2 SISTEMA DE MEDICIÓN DE ESTADOS

El control del vehículo requiere la medición de sus estados dinámicos para realizar control y seguimiento del vehículo y se hace necesario diseñar y ajustar un método para medirlos y/o estimarlos. Para este fin un IMU digital Combo Board, que incluye Giroscopio y

Acelerómetro, fue adquirido para funcionar en conjunto con un sistema de Visión Artificial.

2.2.1 Calibración del IMU. El IMU digital Combo Board integra el giroscopio ITG3200 y el acelerómetro ADXL345 lo que lo hace capaz de obtener información directa sobre velocidad angular alrededor de los 3 ejes, así como la aceleración lineal a lo sobre los 3 ejes. Dichos datos pueden ser fusionados para obtener información sobre la inclinación Pitch y Roll, alrededor de los ejes x_b y y_b, respectivamente. Los datos de Pitch, Roll y Yaw, así como sus respectivas razones de cambio (velocidades angulares) Pitch-rate, Roll-rate, Yaw-rate, son proporcionados mediante comunicación serial hacia el sistema de procesamiento (Arduino) haciendo uso de las librerías facilitadas por el distribuidor del producto. Sin embargo la medición de Yaw presenta *drifting* debido a que el sensor no integra una brújula magnética sino que la librería utiliza solamente una aproximación basada en la integral de Yaw-rate.

A continuación se muestra una imagen del IMU utilizado:

Figura 11. IMU Digital Combo board

El filtro de Kalman es utilizado por la librería para obtener una medida más confiable de los ángulos de inclinación.

Teniendo en cuenta las necesidades del sistema se configuró el sensor para trabajar a máxima resolución (13bits) y con un rango máximo de $\pm 4g$. Una vez configurado el IMU está listo para enviar datos hacia Arduino.

Haciendo uso de los comandos proporcionados por el fabricante en manual de funcionamiento del IMU se tomaron medidas preliminares de los ángulos de inclinación Pitch, Yaw. Sin embargo estas medidas requieren calibración para asegurar la precisión del sistema de medida y por ende el del sistema de control. La medida proporcionada por el IMU puede describirse de la siguiente manera:

$$\zeta_{real} = q_p \zeta_{imu} + b_p$$

Ecuación 27. Relación lineal entre ángulos de inclinación real y medida.

Donde q_p representa la correspondencia proporcional entre la medida del IMU y el valor real de medición; b_p representa el *bias* con la cual están relacionadas las dos medidas. Estos parámetros tienen valores ideales $q_p = 1 \; y \; b_p = 0$. A continuación se muestra un gráfico de $\zeta_{real} \; vs \; \zeta_{imu}$ y los valores obtenidos por medio de regresión lineal para las constantes de calibración $q_p \; y \; b_p$ tanto para le medición de Pitch y Roll.

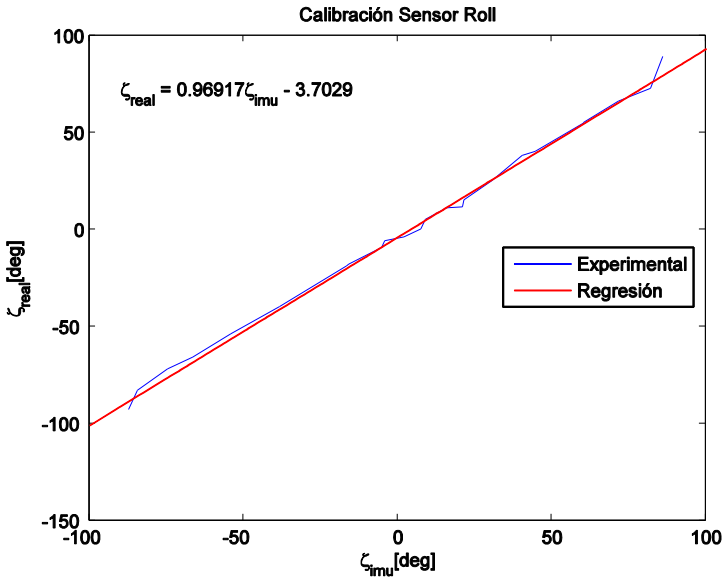


Figura 12. Calibración IMU Roll.

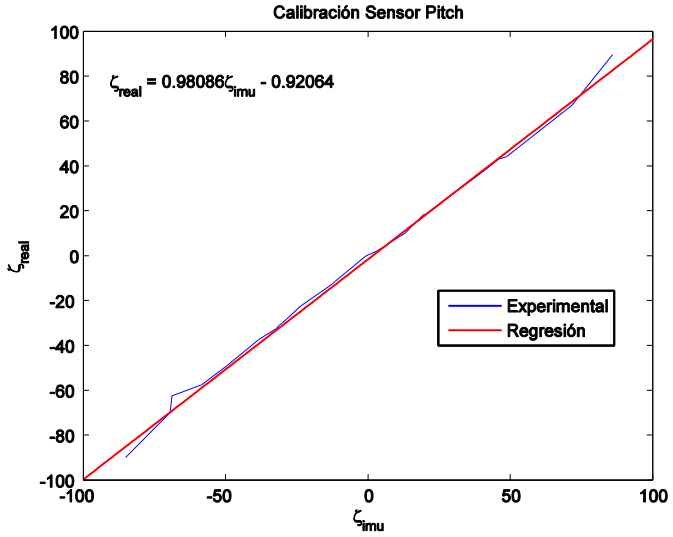


Figura 13. Calibración IMU Pitch.

Los datos obtenidos a partir de la regresión lineal se consignaron en la tabla I.

Tabla 1. Datos obtenidos de la regresión lineal para calibración imu-pitch.

Sensor	q_p	b_p
Roll	0.96917	−3.0729
Pitch	0.98086	−0.92064

La corrección de los ángulos medidos por el IMU fue agregada al código fuente de Arduino para su uso posterior.

2.2.2 Acondicionamiento de medidas del Giróscopo. Las medidas del giróscopo fueron procesadas mediante un filtro IIR para evitar ruido de alta frecuencia. A continuación se muestra los resultados de esta etapa para la velocidad angular pitch-rate como ejemplo.

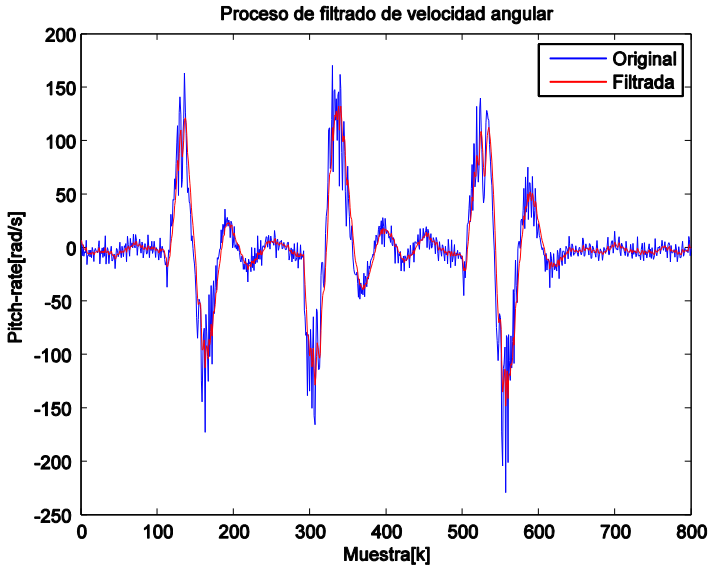


Figura 14. Filtro a señales del giroscopio.

2.2.3 Sistema de Visión Artificial. Los sistemas de visión artificial son los mecanismos que permiten replicar la forma en que los organismos vivos reconocen y localizan objetos en el medio ambiente por medio de procesamiento de imágenes con el fin de aplicarlos a procesos automáticos o maquinas[21]. El principal objetivo de la visión artificial es la de extraer características de una imagen para su descripción e interpretación por medio ordenador. En el presente trabajo un sistema de visión artificial es utilizado para estimar la posición y la velocidad del vehículo en el espacio. Existe gran variedad de métodos para lograr este objetivo. La mayoría de ellos con sistemas de visión estéreo. En este trabajo se propone un método de estimación de posición que utiliza solamente una cámara.

[21] GOMEZ, G. “Visión Computacional”. Mexico: s.n. s.f.

El sistema de Visión artificial consta de varias etapas, las cuales una a una aportan al objetivo de estimar los estados de posición X,Y y Z; las velocidades Lineales V_x, V_y y V_z; y por último la rotación alrededor del eje Z, Yaw.

A continuación se muestra un esquema general del Sistema de Visión Artificial.

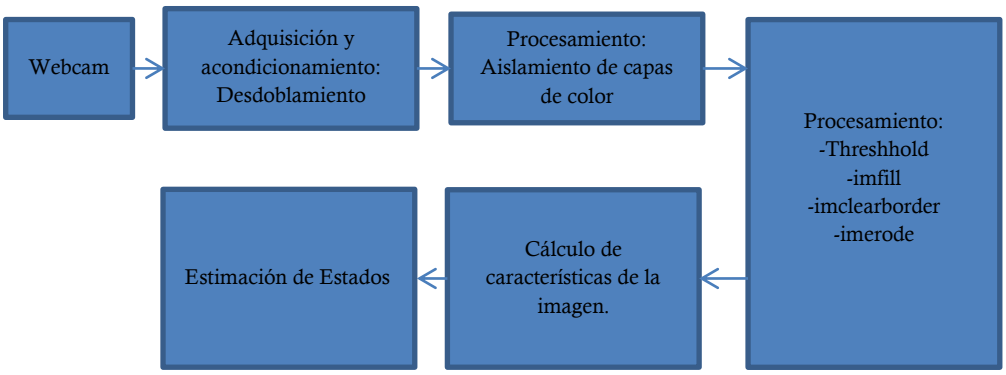


Figura 15. Esquema general del sistema de visión artificial.

Adquisición y acondicionamiento: El sistema de visión artificial hace uso de una cámara y 3 marcadores de Color para estimar parte de los estados dinámicos del vehículo. Para ampliar el rango de visión de la cámara se agregó un lente Fish-Eye como se muestra en la foto:

Figura 16. Lente Fish-Eye sobre cámara web.

Los marcadores se construyeron utilizando pelotas de plástico de colores fuertes y sin brillo, a las cuales se les inserto LEDs de alta intensidad en su interior como se muestra en la siguiente figura.

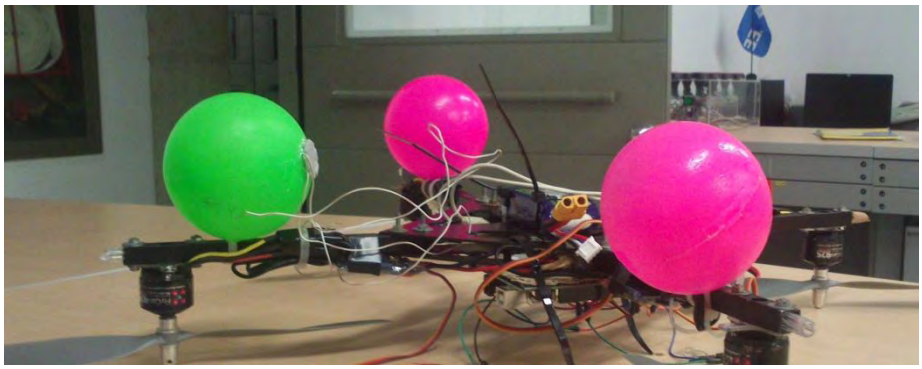


Figura 17. Marcadores visión artificial.

El lente Fish-Eye produce distorsión radial en la toma de imágenes por lo cual fue necesario implementar un sistema de desenvolvimiento de la imagen mediante el modelado del sistema de captura de imágenes proporcionado por el ToolBox OmniCalib para MATLAB.

$a)$ $b)$

Figura 18. a) Imagen proveniente de lente Fish-Eye b) imagen aplicando desenvolvimiento.

La imagen sin distorsión sirve de insumo para la siguiente etapa del sistema: la Etapa de procesamiento.

Procesamiento: La etapa de procesamiento tiene como objetivo aislar y resaltar los marcadores de color ligados al vehículo. Para tal fin la imagen pasa nuevamente por una serie de procesos que se ilustran en la figura 19.

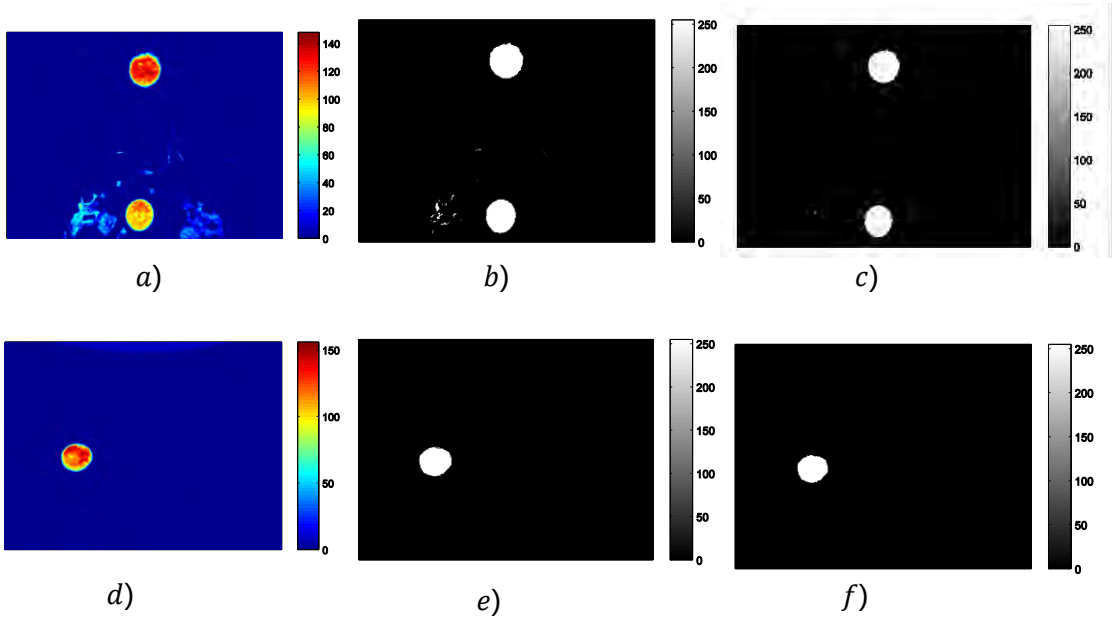


Figura 19. Proceso de etiquetado:a) Aislamiento componente rosa de la imagen b) Imagen binaria rosa Threshold=50 c) Imagen binaria rosa procesada d) Aislamiento componente verde de la imagen e) Imagen binaria verde Threshold=50 f) Imagen binaria verde procesada

Medición directa de posición: A partir de la imagen Binaria obtenida en la etapa anterior se obtiene información útil para estimar la posición X,Y,Z del vehículo. Considerando la siguiente imagen ejemplo:

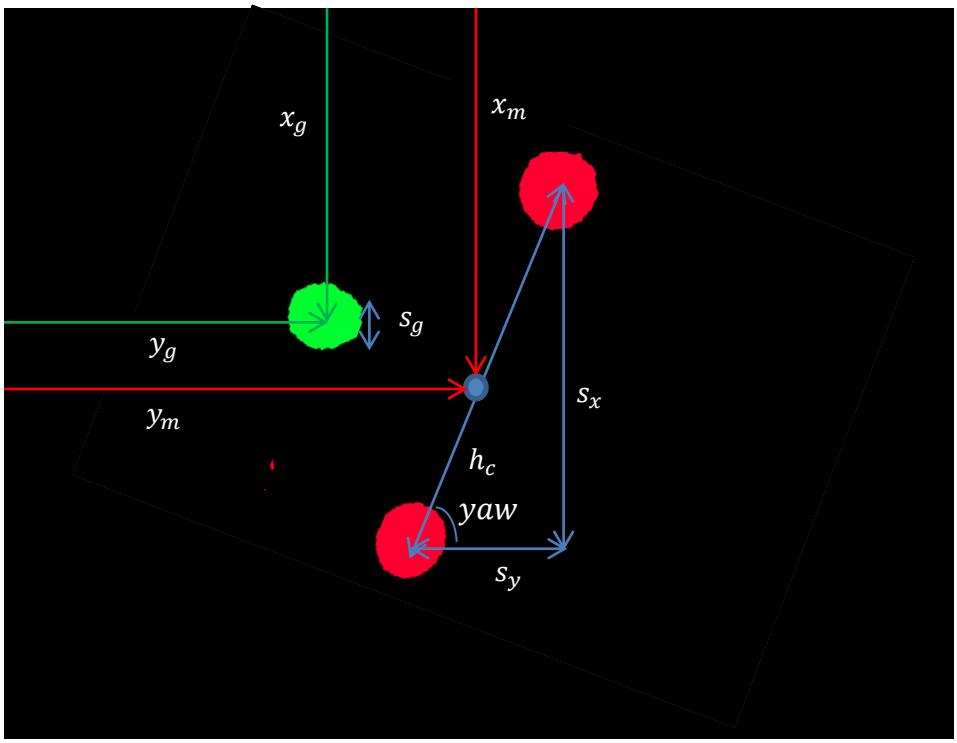


Figura 20. Estimación de la posición X,Y,Z.

Defínase las variables:

$$x_m: El\ promedio\ de\ la\ posición\ en\ X\ de\ todos\ los\ puntos\ rojos\ de\ la\ imagen$$
$$y_m: El\ promedio\ de\ la\ posición\ en\ Y\ de\ todos\ los\ puntos\ rojos\ de\ la\ imagen$$
$$s_x: La\ desviación\ absoluta\ media\ sobre\ el\ eje\ X\ de\ los\ puntos\ rojos\ de\ la\ imagen$$
$$s_y: La\ desviación\ absoluta\ media\ sobre\ el\ eje\ Y\ de\ los\ puntos\ rojos\ de\ la\ imagen$$
$$x_g: El\ promedio\ de\ la\ posición\ en\ X\ de\ todos\ los\ puntos\ verdes\ de\ la\ imagen$$
$$y_g: El\ promedio\ de\ la\ posición\ en\ Y\ de\ todos\ los\ puntos\ verdes\ de\ la\ imagen$$
$$s_g: La\ desviación\ absoluta\ media\ sobre\ el\ eje\ X\ de\ los\ puntos\ verdes\ de\ la\ imagen$$

Ahora considerar un modelo lineal de la cámara como el que se muestra a continuación.

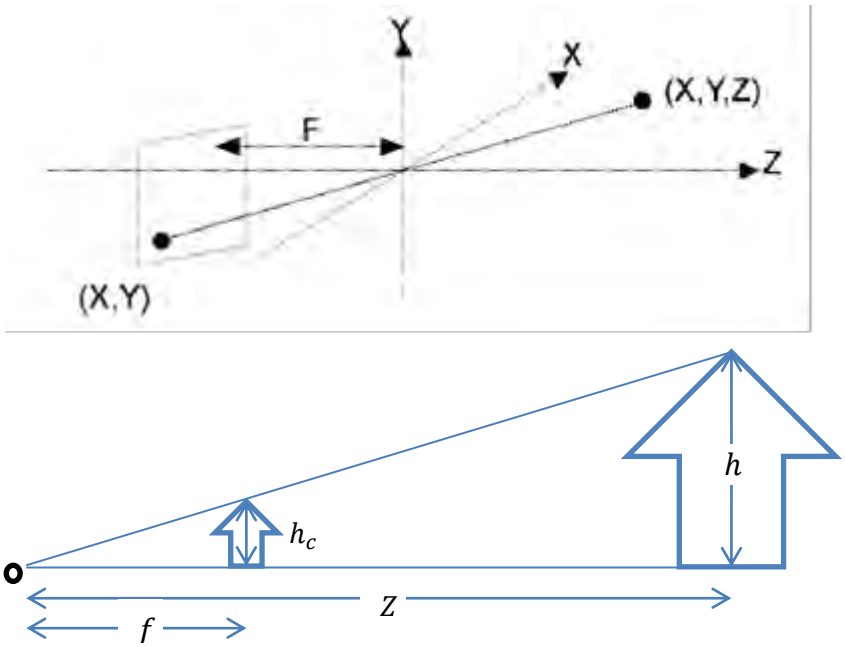


Figura 21. Modelo pinole [14] Modelo lineal de la cámara.

Del cual podemos obtener la siguiente razón:

$$\frac{h}{Z} = \frac{h_c}{f}$$

Ecuación 28. Relación de triángulos semejantes.

Donde f representa el producto de la distancia focal de la cámara y el factor de escalamiento pixels-metros.

Ahora si h representase la distancia real entre los marcadores rosa en metros, de la ecuación 28 se puede obtener Z conociendo la distancia entre marcadores en pixeles en la cámara h_c, la cual puede ser calculada como se muestra en la figura 20 de la siguiente manera.

$$h_c = \sqrt{s_x^2 + s_y^2}$$

Ecuación 29.

En resumen:

$$Z = \frac{k_z h f}{\sqrt{s_x^2 + s_y^2}}$$

Ecuación 30.

Donde k_z es un parámetro de calibración.

Ahora tomando un nuevo enfoque de la ecuación 28 tenemos que:

$$X = k_x Z \frac{x_m}{f} \; y \; Y = k_y Z \frac{y_m}{f}$$

Ecuación 31.

Donde k_x y k_y son parámetros de calibración.

La calibración de los parámetros k_x, k_y y k_z fue realizada de manera empírica.

A continuación se muestran un ejemplo de medición de posición realizada por medio del sistema de visión artificial.

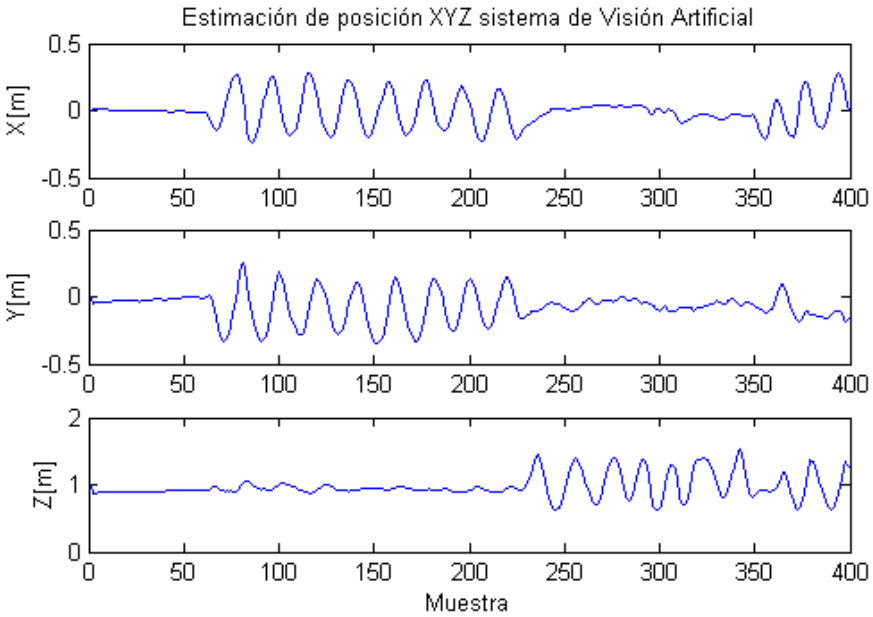


Figura 22. Estimación de posición con sistema de visión artificial

Estimación de la rotación alrededor del eje Z, Yaw(ψ): Si se define un Norte de referencia paralelo al eje Y, la rotación sobre el eje Z respecto a tal Norte puede calcularse mediante la relación:

$$yaw = sign(x_g - x_m) * \tan^{-1}\left(\frac{s_x - s_g}{s_y - s_g}\right)$$

Ecuación 32.

En la práctica s_g se ajusta a un valor ligeramente menor para evitar singularidades en el cociente si se trabaja con sistemas embebidos que no soportan la asignación de *infinito* a una variable. MATLAB soporta dicha asignación por lo que no fue necesario realizar este ajuste.

A continuación se muestra un ejemplo de medición de Yaw realizada por medio del Sistema de Visión Artificial.

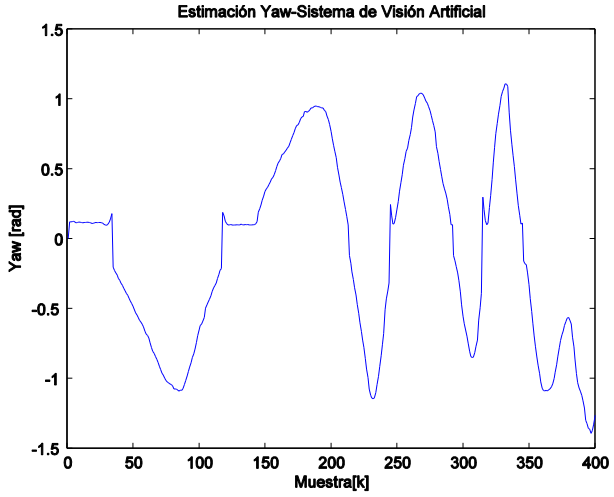


Figura 23. Medición de Yaw realizada por medio del Sistema de Visión Artificial.

Estimación de Velocidad Lineal: Diferentes métodos de estimación de Velocidad Lineal fueron estudiados:

-Mediante la integración de la Aceleración Lineal medida por el IMU.
-Mediante diferenciación de la posición del Sistema de Visión Artificial.

Los dos métodos fueron probados obteniendo mejores resultados con la diferenciación de la posición como se muestra en los siguientes resultados:

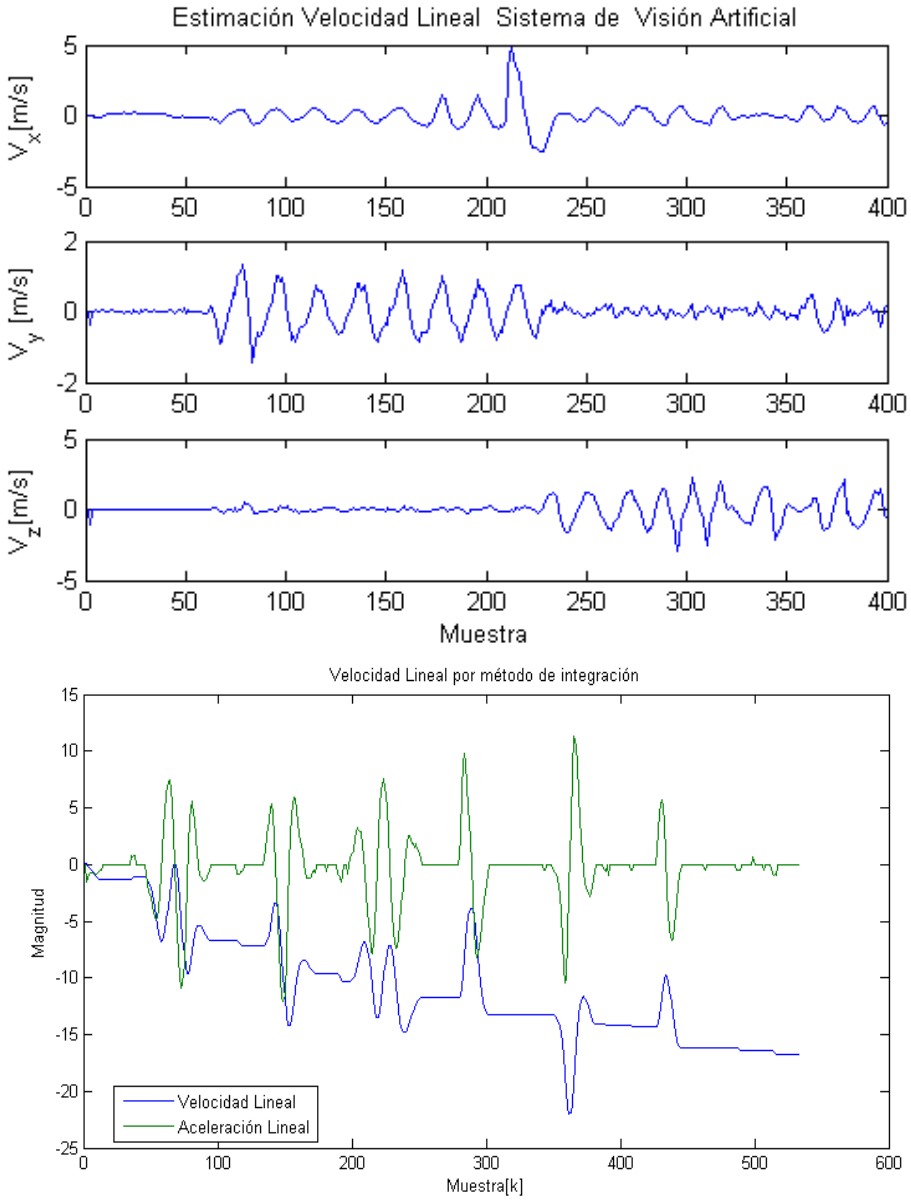


Figura 24. Comparación estimación de velocidad mediante los diferentes métodos. Arriba: mediante sistema de visión artificial. Abajo: mediante integración.

Se observa que la estimación mediante integración presenta un incremento de su nivel de offset a medida que pasa el tiempo, fenómeno conocido como *drifting.* Para solucionar este problema es posible utilizar un filtro de corrección de offset, sin embargo debido a que el offset generado no es constante es posible perder información de la señal al implementar dicho filtro. Por otro lado, la estimación del sistema de visión artificial no tiene ese problema.

2.3 MODELO CUADRICÓPTERO: OBTENCIÓN Y VALIDACIÓN

El cuadricóptero presenta una planta rígida, su modelo matemático está gobernado bajo las leyes fundamentales de la física, especialmente por la segunda ley de movimiento de Newton para la translación y la rotación, la cual expresa que las derivadas de los momentos lineales y angulares de un cuerpo rígido son equivalentes a las sumatorias de las fuerzas externas e inerciales respectivamente.

Existen varios análisis y puntos de vista para la consecución de las ecuaciones de estado del vehículo, sin embargo uno de ellos se ajusta de mejor forma a las necesidades del presente trabajo [15] debido a que presenta el sistema tomando como estados justo los estados de interés de este trabajo, los cuales son las posiciones y velocidades angulares, y la relación entre estos y la posición lineal mediante los ángulos de orientación de Euler.

Con el objetivo de desarrollar las ecuaciones del movimiento se estableció un sistema de ejes referenciado al cuerpo del vehículo (x_b, y_b, z_b) y otro referenciado a tierra (*X,Y,Z*) como se muestra en la figura 25. La masa total del vehículo es m_b y el peso total $m_b g$ actúa en el centro O1 en la dirección de z_b. Debido a la simetría de la configuración la inercia alrededor de x_b y y_b tiene el mismo valor de *Ix*, y la inercia alrededor de z_b tiene el valor de *Iz* que es igual 2*Ix*, y todo el producto de inercia es cero.

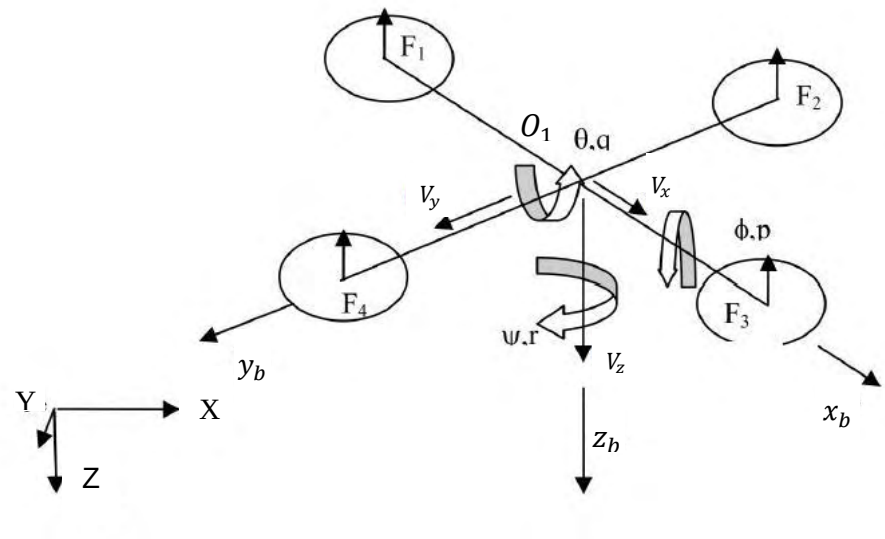


Figura 25. Modelo Cuadricóptero.

Además de la posición angular (phi, theta, psi), las velocidades angulares: roll rate-p (alrededor de x_b), pitch rate-q (alrededor de y_b), yaw rate-r (alrededor de z_b), las velocidades lineales con respecto a los sistema de ejes terrestre: V_x (sobre el eje *X*), V_y (sobre el eje *Y*), V_z (sobre el eje *Z*), y las posición *X,Y,Z* en el marco de referencia terrestre se requieren para definir completamente el comportamiento dinámico del cuadricóptero, por lo tanto, tomando p, q, r, phi, theta, psi, V_x, V_y, V_Z, *X,Y,Z* como variables del vector de

estados y cuatro entradas $\Gamma, L, M\ y\ N$ como variables del vector de control, las ecuaciones del movimiento del vehículo pueden ser descritas como se muestra a continuación:

$$\dot{p} = k_3 qr + L$$
$$\dot{q} = -k_3 pr + M$$
$$\dot{r} = N$$
$$\dot{\phi} = p\,cos\psi - q\,sen\psi$$
$$\dot{\theta} = p\frac{sen\psi}{\cos\phi} + q\frac{cos\psi}{cos\phi}$$
$$\dot{\psi} = psentan\phi + qcos\psi tan\phi + r$$
$$\dot{V_x} = -k_1 V_x^2 - \Gamma\, cos\phi\, sen\theta$$
$$\dot{V_y} = -k_1 V_y^2 + \Gamma sen\phi$$
$$\dot{V_z} = g - \Gamma cos\phi\, cos\,\theta$$
$$\dot{X} = V_x$$
$$\dot{Y} = V_y$$
$$\dot{Z} = V_z$$

Ecuación 33. Ecuaciones del movimiento del cuadricóptero.

Donde p, q, r son las velocidades angulares respecto a los 3 ejes x_b , y_b y z_b respectivamente, k_1, k_3 son parámetros físicos del cuadricóptero, g es la aceleración de la gravedad, L, M, N, Γ son las entradas del sistema; θ, ϕ, ψ son los ángulos de orientación de Euler; V_x, V_y, V_Z representan las velocidades lineales sobre los 3 ejes; y X, Y, Z simbolizan la posición del vehículo. Las entradas del sistema están relacionadas con las velocidades angulares de los motores del vehículo de la siguiente manera:

$$L = d * (w_3^2 - w_1^2)$$
$$M = d * (w_4^2 - w_2^2)$$
$$N = d * (w_4^2 + w_2^2 - w_3^2 - w_1^2)$$
$$\Gamma = b * (w_4^2 + w_2^2 + w_3^2 + w_1^2)$$

Ecuación 34. Entradas del sistema.

Donde w_1, w_2, w_3, w_4 son las velocidades angulares de los motores del cuadricóptero; $d\ y\ b$ son los factores de arrastre y de empuje respectivamente. Los parámetros del vehículo pueden ser calculados como sigue a continuación[22].

$$k_1 = \frac{\rho A_D C_D}{2m_b}$$
$$A_D = 4\pi r_a^2$$
$$k_3 = \frac{I_y}{I_x} - \frac{I_z}{I_x}$$

Ecuación 35. Calculo de los parámetros del vehículo.

Donde ρ es la densidad del aire, C_D es el coeficiente de arrastre, m_b es la masa del vehículo, r_a es la longitud de las hélices. I_x, I_y, I_z Son la inercia alrededor del eje x_b, y_b y

[22] RANGAJEEVA, S., & WHIDBORNE, J. Linear parameter varying control of a quadrotor. Mexico: Industrial and Information Systems, 2011.

z_b respectivamente. Si se asume que el vehículo es lo suficientemente simétrico entonces $I_x = I_y$, $I_z = 2I_x$.

2.3.1 Subsistemas de Orientación. El sub-sistema general de orientación del cuadricóptero (SGO) está referido al conjunto de dinámicas del cuadricóptero que están relacionadas con los ángulos de rotación alrededor de sus propios ejes, es decir, los ángulos de inclinación de Euler y las velocidades angulares. El sub-sistema general de orientación esta descrito como se muestra a continuación:

Tabla 2. Sub-sistema general de orientación

Sub-Sistema	SGO
Modelo	$\dot{p} = k_3 qr + L$ $\dot{q} = -k_3 pr + M$ $\dot{r} = N$ $\dot{\phi} = p\,cos\psi - q\,sen\psi$ $\dot{\theta} = p\frac{sen\psi}{\cos\phi} + q\frac{cos\psi}{cos\phi}$ $\dot{\psi} = psen\psi tan\phi + qcos\psi tan\phi + r$ $L = d_0(w_3^2 - w_1^2)$ $M = d_0(w_4^2 - w_2^2)$ $N = d_0(w_4^2 + w_2^2 - w_3^2 - w_1^2)$ $d_0 = k^2 d$
Parámetros a estimar y/o calcular	k_3, d

w_i en la aplicación en particular, esta proporcionada como una proporción de la velocidad angular de los motores generado por el embebido Arduino como un escalar entre 0 y 500 (1000uS-1500uS) como se describe en la sección "Construcción del prototipo" por tanto podemos describir el sistema como:

$$L = d(w_{e3}^2 - w_{e1}^2)$$
$$M = d(w_{e4}^2 - w_{e2^2})$$
$$N = d(w_{e4}^2 + w_{e2}^2 - w_{e3}^2 - w_{e1}^2)$$
$$w_{ei} = kw_i$$

Ecuación 36.

Donde w_{ei} representa la velocidad angular de los motores medida como el escalar proporcionado por el embebido Arduino y d incluye tanto la proporción d_0 como la proporción generada por el escalamiento de la variable en el embebido k.

2.3.1.1 Estimación de Factor de Arrastre d**.** Para la estimación del parámetro d se tiene en cuenta el SGO para las variables $p\ y\ \phi$, con el sistema bajo las condiciones dadas por:

$$\psi = 0, \theta = 0, r = 0\ , q = 0, L_d = w_{e3}^2 - w_{e1}^2$$

$$\dot{p} = dL_d = L$$
$$\dot{\phi} = p$$

Ecuación 37.

Se montó una estructura para garantizar las condiciones establecidas a partir de trípodes topográficos y madera. La transmisión de las medidas de los estados de roll y roll-rate fue realizada mediante la comunicación serial proporcionada por la interfaz Arduino-Xbee-MATLAB. Las señales de entrada directa (w_e) de los motores fueron proporcionadas por un mando de videojuegos operado de forma manual y enviadas a través de la misma conexión mencionada.

A continuación se muestra el montaje realizado.

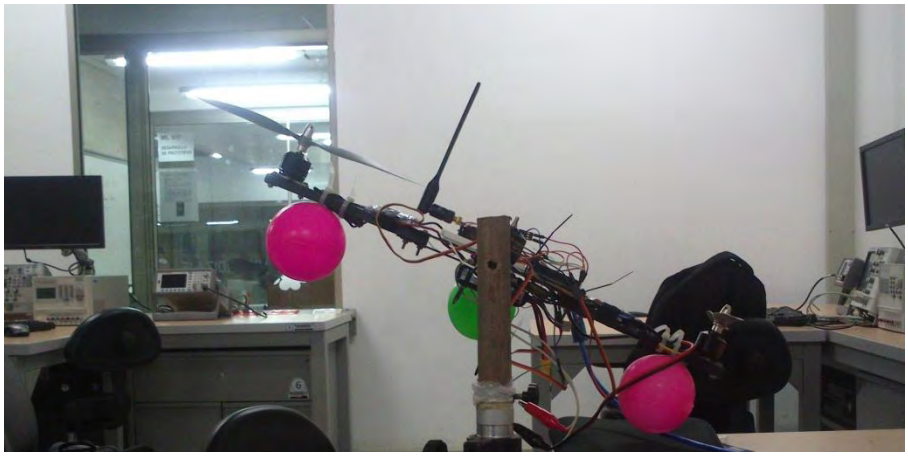


Figura 26. Montaje para estimación facto de arrastre.

Se tomaron mediciones del estado ϕ con valores conocidos para la entrada $L(w_{e1}, w_{e3}\ , d)$ midiendo el estado equivalente roll.

Esto es posible debido a que bajo las condiciones dadas, es decir, como se muestra en la ecuación 37 p representa el valor roll-rate, y $\dot{\phi}$ Phi-rate, por tanto $roll = \phi$.

Las mediciones se muestran a continuación:

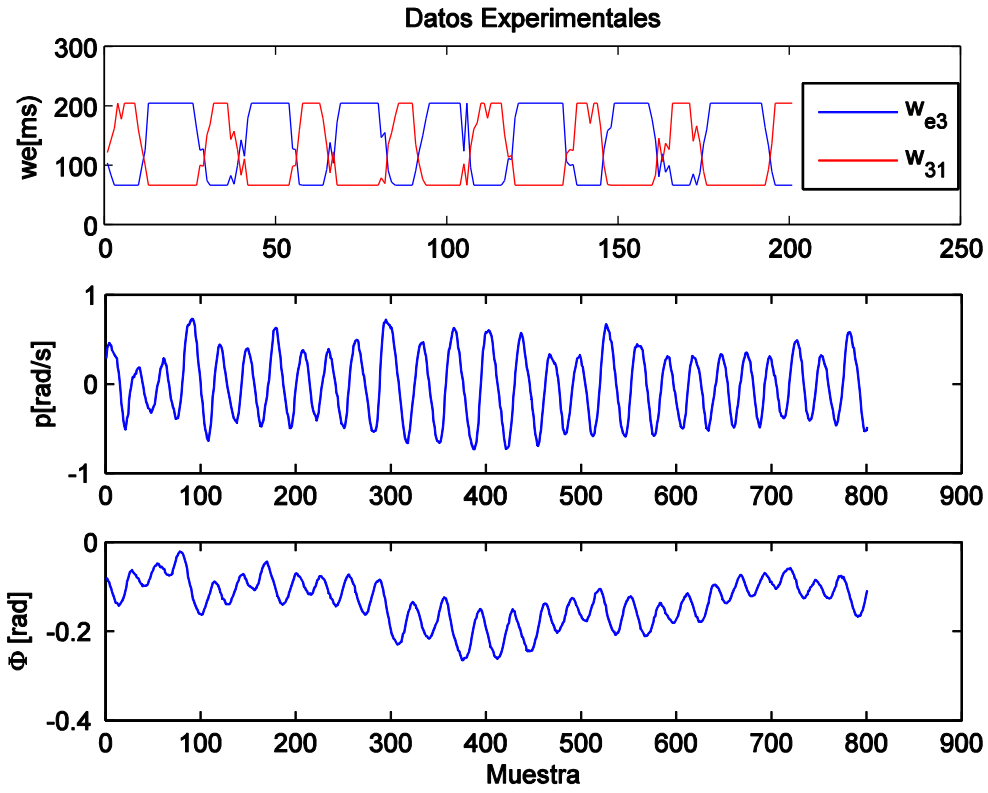


Figura 27. Datos Experimentales de w_e, p, ϕ.

En este punto se utilizó el algoritmo Unscented Kalman Filter (UKF) para estimar d al proponer el sistema aumentado discreto:

$$p_{k+1} = p_k + d_k L_{d\,k}\,\Delta t$$
$$\phi_{k+1} = \phi_k + p_k \Delta t$$
$$d_{k+1} = d_k$$
$$L_d = w_{e3}^2 - w_{e1}^2$$

Ecuación 38.

Los insumos del algoritmo fueron los datos experimentales mostrados en la Figura 27. El tiempo de muestreo fue estimado mediante una búsqueda en línea tomando la relación entra la velocidad angular (roll-rate) y la posición angular (roll). Una incorrecta estimación del tiempo de muestreo real puede confluir en una estimación falsa del factor de arrastre debido a que Δt está multiplicando a dicho factor como se muestra en la ecuación 38.

El algoritmo UKF fue programado en MATLAB siguiendo los pasos descritos en la sección Marco Teórico: Algoritmo UKF. Como resultado se tiene valor estimado d mediante la ejecución del script UKF adjunto. A continuación se muestra gráficos del proceso de evolución del algoritmo.

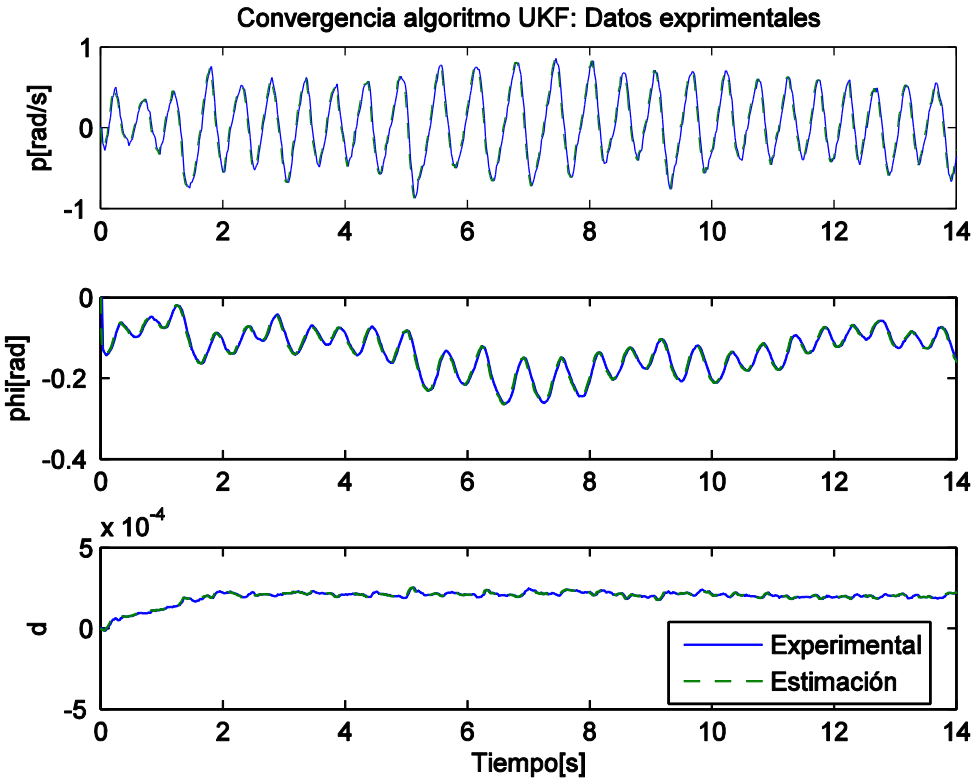
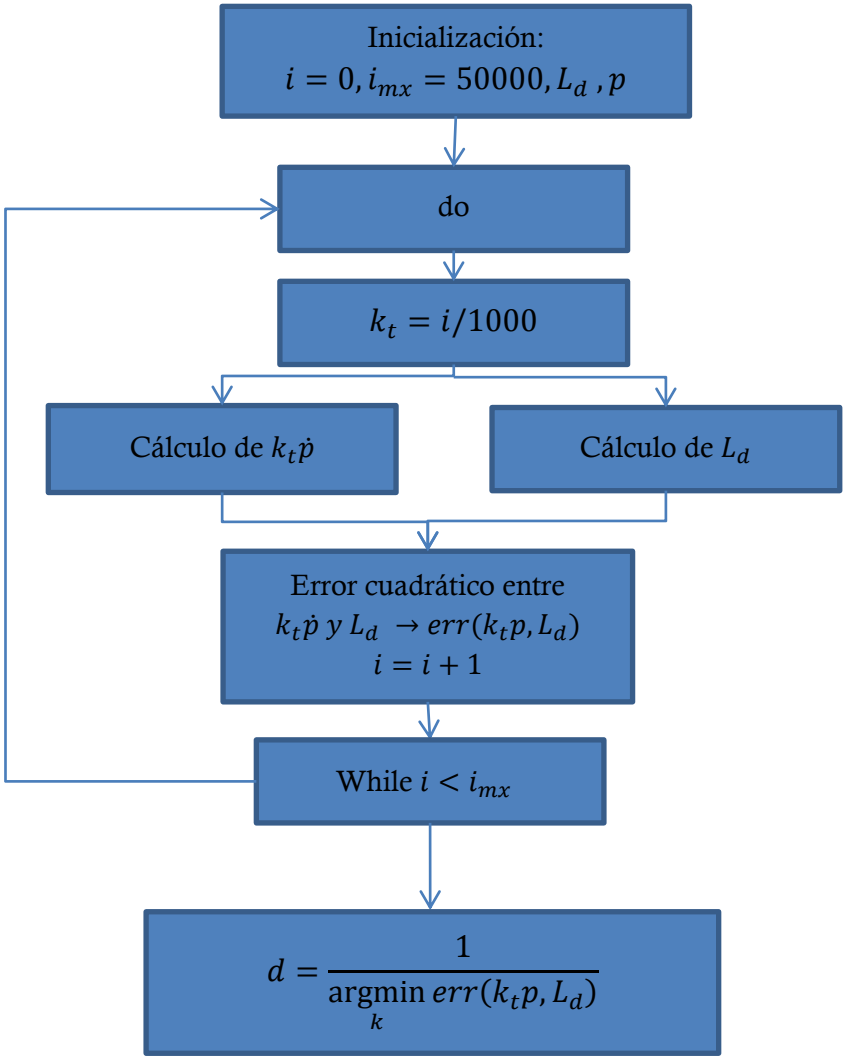


Figura 28. Convergencia UKF, Medidas experimentales, $d = 1.825x10^{-4}$, calculado mediante promedio de la zona de convergencia final.

Se propuso también un método de estimación denominado búsqueda en línea para la estimación del factor de arrastre, que consiste en obtener un graficó del error cuadrático entre la señal estimada y la señal real versus el cambio del parámetro sometido a estimación, y seguidamente concluir que el parámetro es aquel que confluye en un menor error cuadrático.

A continuación se muestra un esquemático de los procesos realizados por este método de estimación



.

Figura 29. Algoritmo de búsqueda en línea

Una búsqueda en línea fue utilizada también para la estimación del parámetro, haciendo uso de la relación entre la derivada de la velocidad angular (*p*) y la entrada L como se muestra a continuación:

$$\dot{p} = dL_d = \frac{L_d}{k_t}$$

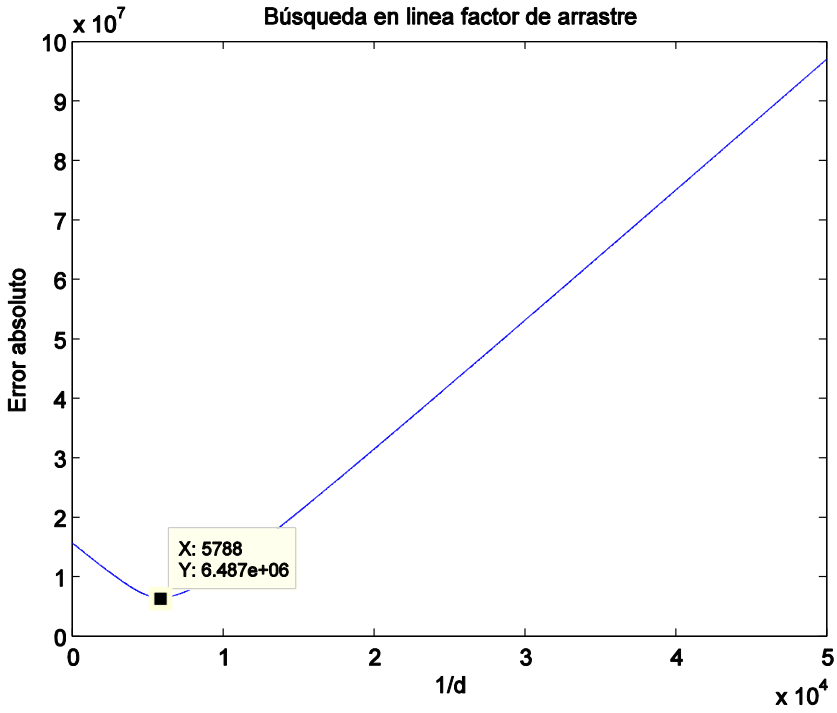


Figura 30. Búsqueda en Línea de Inverso de Factor de Arrastre ($\frac{1}{d} = 5788\,, d = \ 1.7277x10^{-4}$).

A continuación se muestra la validación del Factor de Arrastre mediante los dos métodos usando la derivada de la velocidad angular (37).

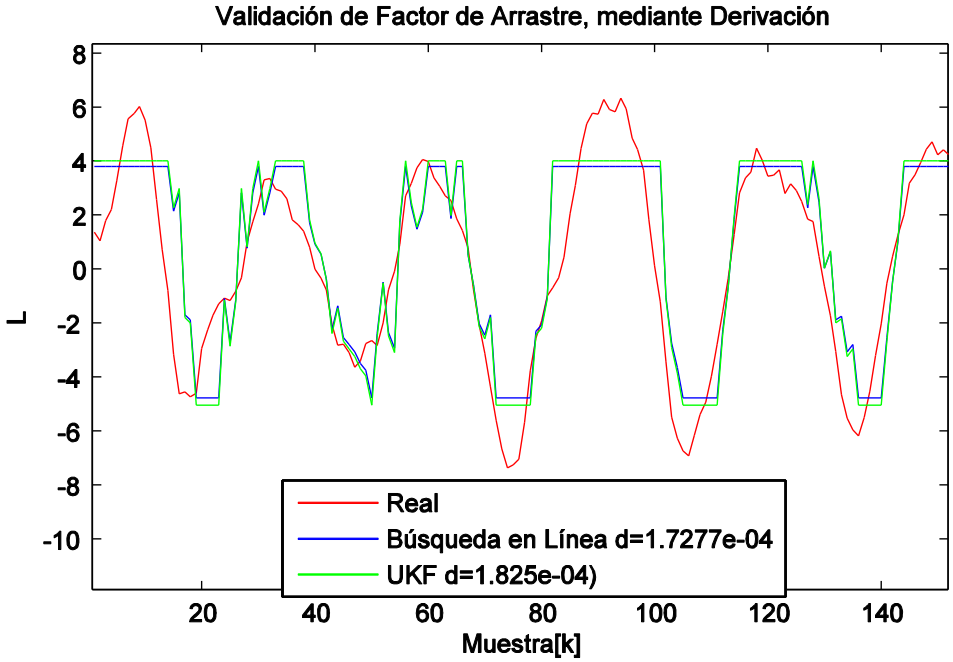


Figura 31. Validación de Factor de Arrastre.

Podemos calcular d como el promedio de los dos valores estimados:

$$d = \frac{1.72770^{-4} + 1.825x10^{-4}}{2} = 1.7763e-04$$

2.3.2 Subsistema de Posición. El sub-sistema general de posición agrupa las dinámicas del vehículo que están relacionadas con la posición Lineal del Vehículo, es decir, sus coordenadas en el espacio (X, Y, Z) y sus respectivas Velocidades Lineales (V_x, V_y, V_Z). Además el sub-sistema general puede ser dividido nuevamente en dos sub-sistemas más: El sub-sistema de Posición Horizontal (SPH) y el Sub-sistema de posición Vertical (SPV). El SPH tiene como entradas directas las inclinaciones $\phi\ y\ \theta$ para la traslación en el eje X y Y respectivamente; y el SPV tiene como entrada directa la aceleración por empuje Γ para el levantamiento del vehículo a una altura Z. A continuación se muestra las dinámicas de los sub-sistemas de posición.

Tabla 3. Dinámicas de los sub-sistemas de posición.

Sub-Sistema	SPH	SPV
Modelo	$\dot{V_x} = -k_1 V_x^2 - \Gamma\, cos\phi\ sen\theta$ $\dot{V_y} = -k_1 V_y^2 + \Gamma sen\phi$ $\dot{X} = V_x$ $\dot{Y} = V_y$	$\dot{V_z} = g - \Gamma cos\phi\ cos\ \theta$ $\dot{Z} = V_z$ $\Gamma = b(w_4^2 + w_2^2 + w_3^2 + w_1^2)$
Parámetros a estimar y/o calcular	k_1	b

2.3.2.1 Estimación de Factor de Empuje b. El SPV descrito como se muestra en la Tabla III requiere la estimación de un solo parámetro b, el factor de empuje. Para su estimación se procedió como muestra el siguiente razonamiento.

Γ Está definida por (34); Γ representa la aceleración proporcionada al vehículo por acción de las hélices. Si consideramos solo el aporte de un motor (w_1) tendremos:

$$\Gamma = b * w_{e1}^2$$
$$F = m_b \Gamma = b_o w_{e1}^2$$

Ecuación 39.

Donde $b = \frac{b_o}{m_b}, m_b = Masa\ del\ Vehículo,$
$w_{e1} = k_w(w_1) = Velocidad\ angular\ embebido,$
$F = fuerza\ Generada\ por\ la\ aceleración\ \Gamma$

Se implementó un arreglo para lograr medir el Empuje, mediante el traslado de la fuerza ejercida por los motores por medio de una palanca(los brazos del vehículo) hasta una balanza como lo muestra la Figura 32.

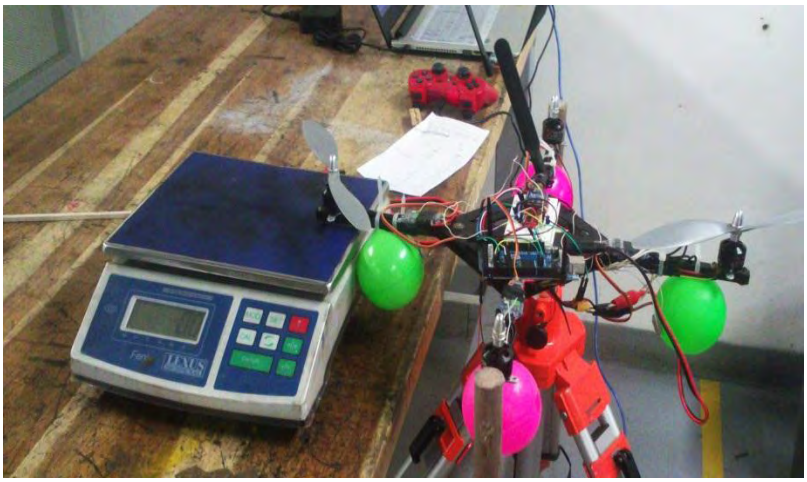


Figura 32. Montaje Medición Factor de Empuje.

La salida del sistema nos arroja una medida $m^* = \frac{F}{g}$ en gramos, donde g representa la aceleración de la gravedad, para obtener entonces:

$$\Gamma = m^* g / m_b$$

Lo que nos brinda una medida experimental de Γ .Usando esta medida y las mediciones de w_{e1} es posible realizar regresión cuadrática, lo que nos ofrece información sobre el factor de empuje. Cabe aclarar que estas mediciones se realizan para un solo motor por lo tanto las mediciones de Γ deben multiplicarse por 4 (4 Motores) para realizar la regresión cuadrática.

A continuación se muestra los resultados de este proceso:

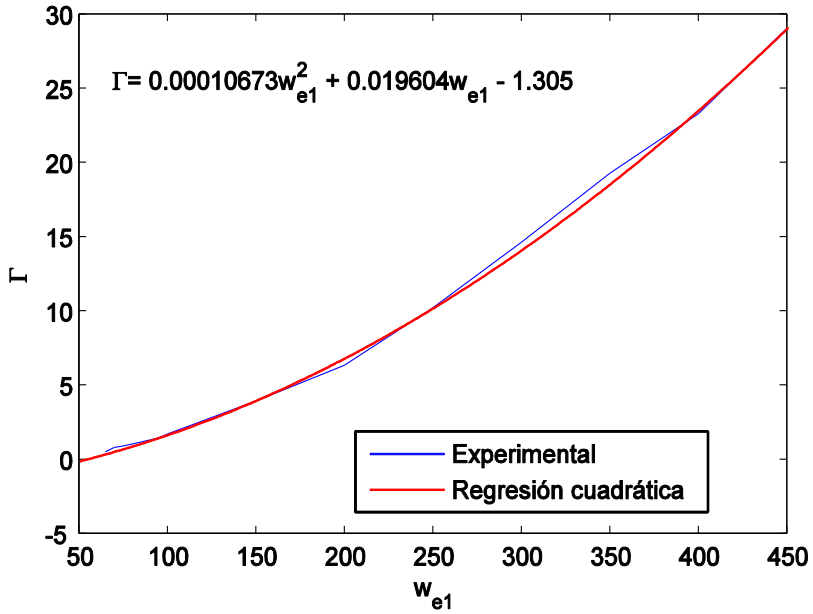


Figura 33. Regresión cuadrática de Γ respecto a w_{e1} para estimación de factor de empuje b.

La regresión puede ser reescrita en forma canónica completando cuadrados para obtener el offset de aceleración Γ y de w_e asi:

$$\Gamma = a_1 w_e^2 + a_2 w_e + a_3$$
$$\Gamma = a_1 \left(w_e^2 + \frac{a_2}{a_1} w_e + \frac{a_3}{a_1} \right)$$
$$\Gamma = a_1 \left(w_e + \frac{a_2}{2a_1} \right)^2 + a_3 - \frac{a_2^2}{(2a_1)^2}$$

Ecuación 40.

$$d = a_1 = factor\ de\ arrastre$$

Las cantidades: $\frac{a_2}{2a_1}$ y $a_3 - \frac{a_2^2}{(2a_1)^2}$ son valores de ajuste de offset en las variables w_e y Γ respectivamente para $a_1 = 1.0673\text{x}10^{-4}, a_2 = 1.9604\text{x}10^{-2}, a_3 = -1.305.$

La regresión cuadrática suministró información suficiente para el cálculo del factor de empuje y el offset generado tanto en la variable w_e como en la variable Γ necesarios para realizar la programación del controlador en el embebido Arduino.

2.3.2.2 Cálculo de k_1. El parámetro k_1 se calculó como se describe en las ecuaciones 35 y su resultado fue:

$$k_1 = 9.9567x10^{-4}$$

2.3.3 Resumen Modelamiento del cuadricóptero. La estimación total de parámetros dio como resultado:

Tabla 4. Estimación total de parámetros

Sub-Sistema	Modelo	Parámetros
SGO	$\dot{p} = k_3 qr + L$ $\dot{q} = -k_3 pr + M$ $\dot{r} = N$ $\dot{\phi} = p\,cos\psi - q\,sen\psi$ $\dot{\theta} = p\frac{sen\psi}{\cos\phi} + q\frac{cos\psi}{cos\phi}$ $\dot{\psi} = psentψtan\phi + qcos\psi tan\phi + r$	$L = d(w_{e3}^2 - w_{e1}^2)$ $M = d(w_{e4}^2 - w_{e2}^2)$ $N = d(w_{e4}^2 + w_{e2}^2 - w_{e3}^2 - w_{e1}^2)$ $d = 1.7763x10^{-4}$ $k_3 = -1$
SPH	$\dot{V}_x = -k_1 V_x^2 - \Gamma\, cos\phi\, sen\theta$ $\dot{V}_y = -k_1 V_y^2 + \Gamma sen\phi$ $\dot{X} = V_x$ $\dot{Y} = V_y$	$\Gamma = b(w_{e1}^2 + w_{e2}^2 + w_{e3}^2 + w_{e4}^2)$ $b = 0.064659$ $k_1 = 9.9567x10^{-4}$
SPV	$\dot{V}_z = g - \Gamma cos\phi\, cos\,\theta$ $\dot{Z} = V_z$	

2.3.4 Simplificación del SGO del cuadricóptero. La rotación alrededor del eje Z, Yaw (ψ) tiene la característica de simplificar el SGO del cuadricóptero al hacerse cero y mantenerse en dicho valor si se considera despreciable la acción inercial causada por k_1, es decir si:

$$\psi = 0\; y\; r = 0, k_1 = 0$$

Entonces el modelo se simplifica como se muestra en la siguiente tabla:

Tabla 5. Características SGO simplificado

Sub-Sistema	Modelo Simplificado	Parámetros
SGO'	$\dot{p} = L$ $\dot{q} = M$ $\dot{\phi} = p$ $\dot{\theta} = \frac{q}{cos\phi}$	$L = d(w_{e3}^2 - w_{e1}^2)$ $M = d(w_{e4}^2 - w_{e2}^2)$ $d = 1.7763\text{x}10^{-4}$

2.4 DISEÑO Y SIMULACION DE CONTROLADORES

2.4.1 Control de orientación. El modelo de SGO mostrado en el conjunto de ecuaciones mostrado en la tabla IV, puede simplificarse si se mantiene una rotación Yaw constante e igual a cero, como se describió anteriormente. Por tanto se decidió realizar un controlador Sliding Mode cuya única tarea consista en regular a cero la orientación Yaw. Se procedió a realizar el diseño y la simulación de los controladores sometidos a prueba teniendo en cuenta esta premisa.

Los controladores puestos a prueba para este subsistema fueron:

-PID
-FUZZY
-MPC
-Sliding Mode Control

Al final de la sección se muestra gráficos comparativos de la respuesta de los distintos controladores.

2.4.1.1 PID para SGO. Se diseñó controladores PID para los subsistemas de orientación (ϕ, θ) descritos enla tabla IV resultando en un sistema de control como se muestra en la ecuación 2. Las constantes del controlador se sintonizaron usando métodos empíricos con la ayuda de la herramienta PID Tunning de MATLAB.

Los valores finales de las constantes del controlador fueron:

Tabla 6. Valores finales del controlador PID para SGO

Controlador:	Constantes			
	P	I	D	N
$\boldsymbol{\phi}$	8.929	1.000	12.289	20
$\boldsymbol{\theta}$	8.364	0.628	10.114	20

Simulaciones del comportamiento del controlador fueron realizadas utilizando el modelo en Simulink de la planta.

2.4.1.2 Controlador FUZZY para SGO. Para el diseño del controlador Fuzzy se utilizó el bloque *Fuzzy Logic Controller*. y la herramienta *Fuzzy Logic Design* de MATLAB.

Los parametros requeridos por el bloque Fuzzy son:

Tabla 7. Requerimientos para el bloque fuzzy

Parámetro	Método
AND	Función *min*
Implicación	Función *min*
Agregación	Función *max*
Defusificación	Centroide(Centro de Masa)

El esquema general de un controlador Fuzzy requiere como entradas el error del estado frente a una referencia (e) y la derivada del mismo (de). Sin embargo, teniendo en cuenta que se tiene medición de la velocidad angular (ω) haciendo uso del giróscopo integrado en el IMU utilizado, se decidió hacer uso de ella en reemplazo de la derivada del error (de) y diseñar la tabla de normas Fuzzy teniendo en cuenta tal aspecto. Cabe resaltar que de depende tanto de la vairación del a referencia como dela variación del estado, en cambio ω solo cuantifica la variación del estado.

Para el controlador se realizó la siguiente tabla de normas Fuzzy para las variables difusas *e y* ω.

Tabla 8. Tablas de normas fuzzy para e y ω.

e \ ω	NN	N	O	P	PP
NN	O	N	NN	NN	NN
N	P	O	N	NN	NN
O	PP	P	O	N	NN
P	PP	PP	P	O	N
PP	PP	PP	PP	P	O

Convenciones	
Muy positivo	PP
Positivo	P
Neutral	O
Negativo	N
Muy negativo	NN

Se utilizó 5 valores lingüísticos pues utilizar mas no presentó mejoras significativas en la etapa de simulación.

Se sintonizó los parámetros de funciones de membresía requeridas para el proceso de Fuzificación y defuzificucación del controlador mediante métodos empíricos. Los parámetros que brindaron un mejor desempeño fueron:

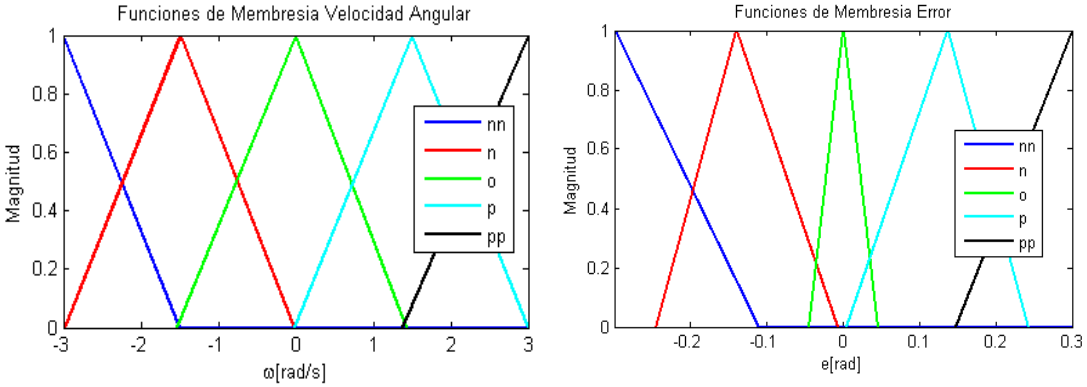


Figura 34. Funciones de Membresía de Entradas del controlador Fuzzy, Error en el Angulo y Velocidad Angular.

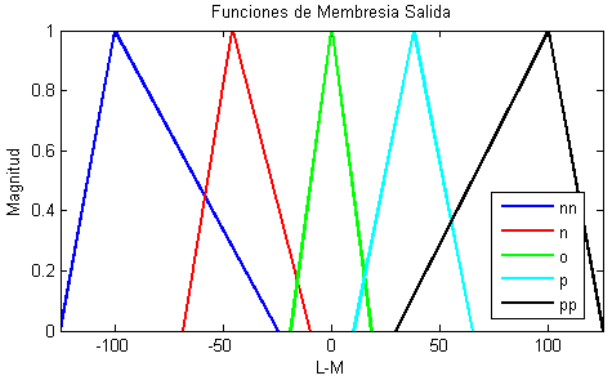


Figura 35. Funciones de Membresía Salida del controlador Fuzzy (L, M)

Se resalta que las funciones de membresía para e y ω (Figura 34) entran en saturación una vez superado sus entradas tanto en el eje positivo como el negativo.

Las simulaciones del controlador fueron realizadas tanto para la variable θ como para ϕ. La regulación de ψ (Yaw) se realizó mediante controlador Sliding Mode. No se implementó componente integral para el controlador Fuzzy debido a que se decidió evitar al máximo posible la programación de integradores en el sistema embebido debido al *drifting* generado por la integración discreta.

A continuación se muestra el mapeo obtenido para este controlador:

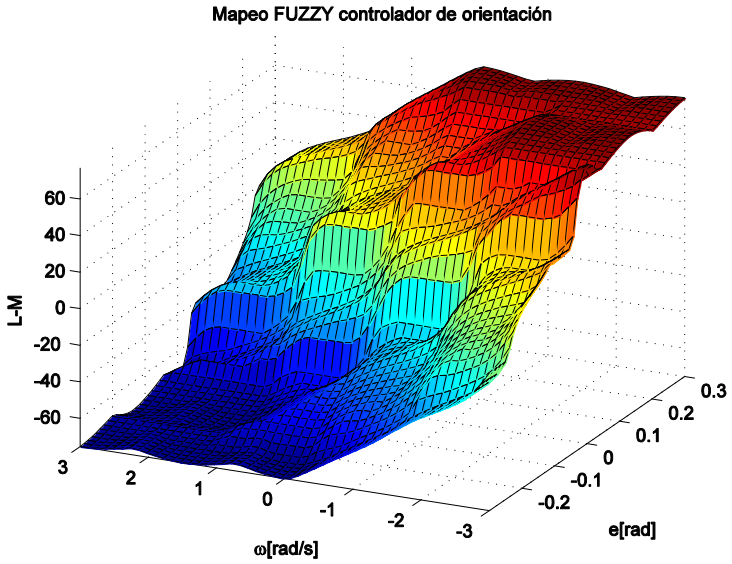


Figura 36. Mapeo obtenido por el controlador.

Una vez diseñado se procedió a realizar diferentes pruebas de rendimiento.

2.4.1.3 Sliding-Mode Control (SMC) para SGO'. Para controlar los ángulos ϕ, θ y ψ los cuales afectan directamente la orientación del vehículo tanto en los ejes horizontales como el vertical, se aplica la estrategia de control de modos deslizantes (ver Figura 37) donde la señal de control está definida como $U = -kS$. Para el caso en particular S es una función definida como sigue $S = sat(a(x_1 - ref) + bx_2)[9]$. $sat(x)$ se define como una función saturación (41), y su argumento se denomina "superficie deslizante". La función $sat(x)$ evita la conmutación extrema de las entradas lo cual favorece a la aplicación del controlador en la planta ya que los motores no pueden realizar ese tipo de maniobras.

$x_1 - ref$ representa el error de estado, ref la señal referencia y x_2 es la derivada del estado x_1.

$$sat(x) = \begin{Bmatrix} x & -1 < x < 1 \\ -1 & x \leq -1 \\ 1 & x \geq 1 \end{Bmatrix}$$

Ecuación 41. Función Saturación

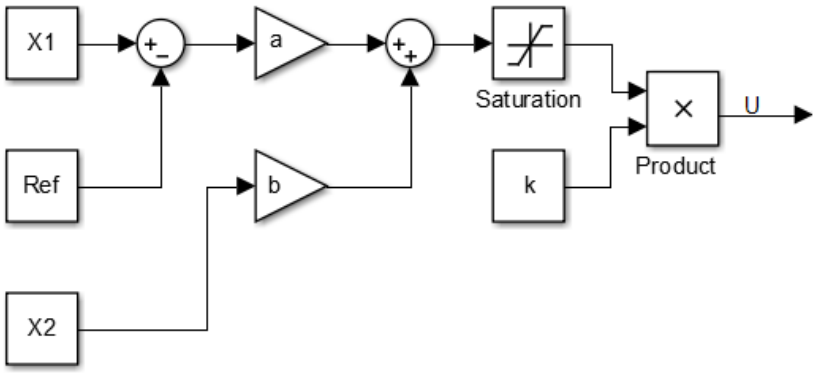


Figura 37. Diagrama de bloques Sliding Mode.

Los valores de las constantes k, a y b se calcularon teniendo en cuenta el modelo general para aplicar Sliding Mode Control que se muestra a continuación[23]:

$$\dot{x}_1 = x_2$$
$$\dot{x}_2 = h(x) + g(x)u$$

Como se puede observar este modelo es semejante al que se describe para el subsistema simplificado SGO' (4).

$$\left|\frac{\mathrm{a}x_1 + h(x)}{g(x)}\right| \leq k$$

Ecuación 42.

Los valores de las constantes $k, a\ y\ b$ se sintonizan para que cumplan esta inecuación y así asegurar la convergencia y estabilidad del sistema de control[24].

Tabla 9. Parámetros controlador sliding mode para control de orientación

Constante	a	b	k
Valor	7	-0.9	30

Para comprobar que los valores escogidos en la tabla cumplen la inecuación (42), retomamos las ecuaciones del modelo del cuadricóptero que modelan las derivadas de los estados Pitch-Rate y Roll-Rate, las cuales son:

$$\dot{p} = k_3 qr + L$$
$$\dot{\phi} = p$$

Ecuación 43.

[23] Ibíd.

[24] Ibíd.

Como los parámetros k_3 y r tienden a cero y considerando la constante b que está relacionada con $g(x) = \frac{1}{\mathrm{b}}$ podemos rescribir el sistema anterior como:

$$\dot{p} = L$$
$$\dot{\phi} = \frac{p}{\mathrm{b}}$$

Ecuación 44.

En este caso $h(x)$ es cero y x_2 relacionada con L,x_1 toma valores en el rango $[\frac{-\pi}{4}, \frac{\pi}{4}]$, por lo tanto se verifica el cumplimiento de la inecuación:

$$\left| \frac{7\left(\frac{\pi}{4}\right) + 0}{\frac{1}{(-0.9)}} \right| \leq 30$$
$$4.98 \leq 30$$

Se verifica el cumplimiento de la inecuación.

2.4.1.4 Model Predictive Control (MPC) para SGO'. El control mediante MPC (Model Predictive Control) es un método de control basado en sistemas de optimización. El controlador hace uso del modelo para predecir estados futuros y así tomar la mejor decisión en la entrada del sistema a controlar. El sistema requiere de un buen modelo matemático para su funcionamiento y en general su teoría es aplicable a sistemas lineales. Su uso en sistemas no lineales está sujeto a considerar que la no-linealidad no es demasiado fuerte.

Para su simulación se utilizó el bloque MPC Controller de Simulink (adjunto) y para su diseño la aplicación MPC Design de MATLAB. Los parámetros se calibraron de manera empírica y se tuvo en cuenta las restricciones físicas del Vehículo.

A continuación se muestra un resumen de los parámetros y restricciones utilizados para el diseño del controlador.

Tabla 10. Parámetros controlador MPC para orientación

Parámetro:	Valor
Intervalo de control	0.05s
Horizonte de Predicción	5
Horizonte de Control	2
Restricciones en el estado	[-0.6 rad,0.6 rad]
Restricciones en la Entrada L,M	[-200, 200]

Las restricciones en el estado (los ángulos de inclinación *Theta* y *Phi*) se implementan con el fin de evitar que el vehículo realice movimientos inadecuados y presente comportamiento inestable. Las restricciones en la Salida ([-100,100]) se establecieron

teniendo en cuenta las restricciones de los motores del cuadricóptero. Sin embargo se adecuaron a [-200,200] para lograr una mejor respuesta del controlador sin sobrepasarlas observando las medidas de la simulación de L y M las cuales permanecieron en el rango [-100,100].

2.4.1.5 Gráficos comparativos de controladores de orientación. Se realizó la simulación de los controladores usando MATLAB y Simulink realizando su análisis respectivo. Se estudió el comportamiento ante entradas escalón, sinusoidal y perturbaciones (ver figura 38). Evaluar la respuesta de los controladores ante las condiciones mencionadas es de gran importancia pues provee información que sirve de insumo para realizar el cálculo de los parámetros de rendimiento, lo que permite realizar la eleccion del mejor controlador.

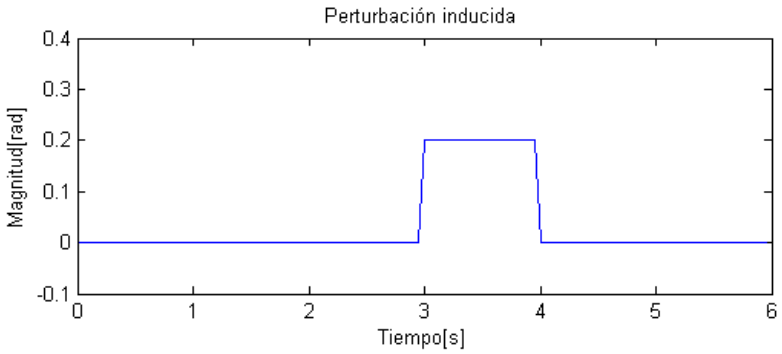


Figura 38. Perturbacion inducida

Las respuestas de los controladores sobre la variable θ y ϕ son muy similares, razón por la cual continuación se muestra como ejemplo solo la respuesta de los controladores sobre la variable ϕ.

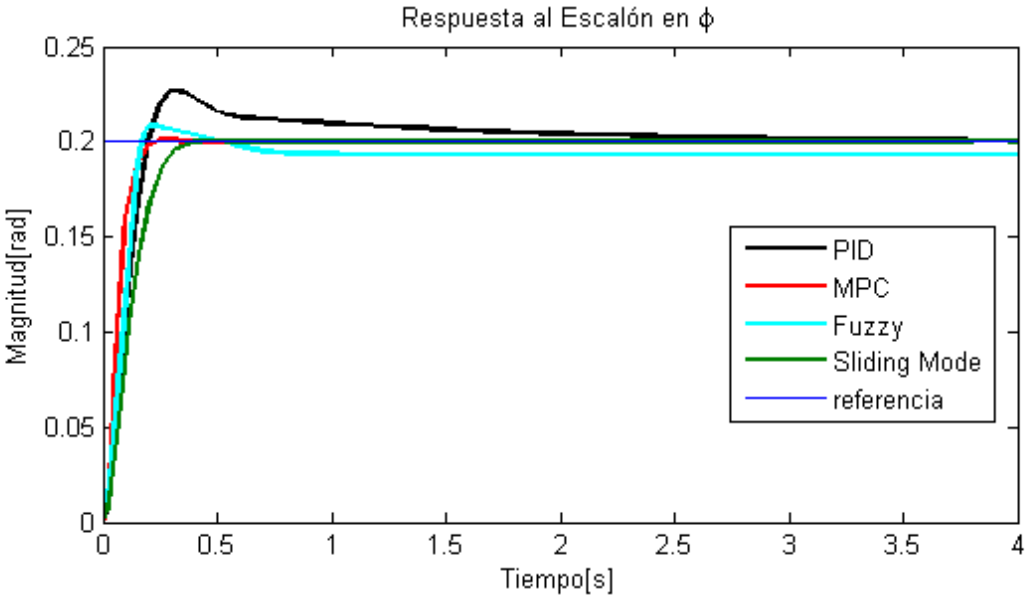


Figura 39. Respuesta al escalón en ϕ.

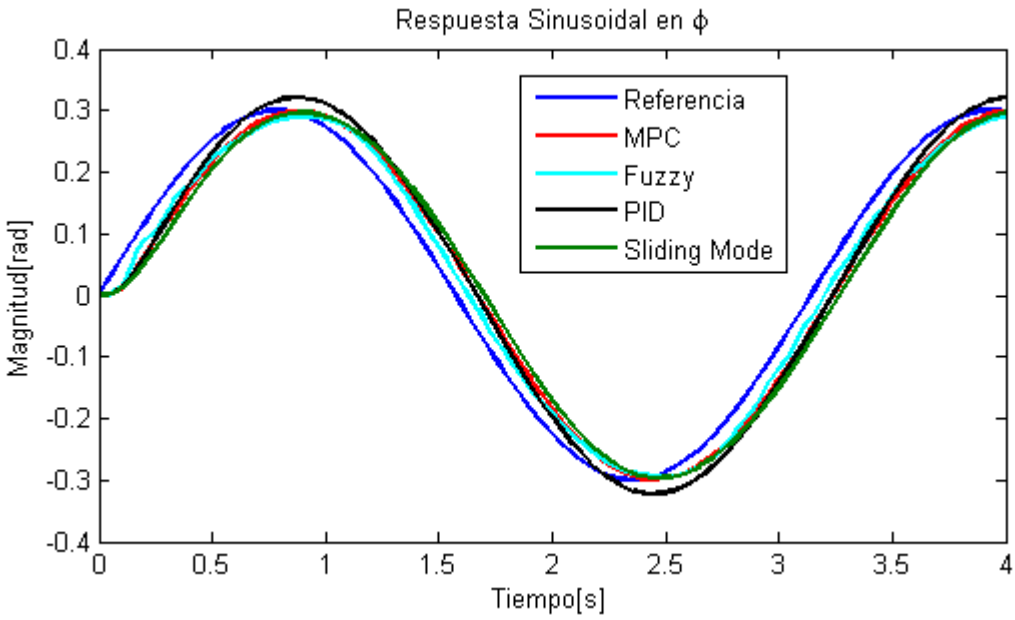


Figura 40. Respuesta Sinusoidal en ϕ.

En la figura 39 es posible observar que todos los controladores cumplen con el objetivo de un error de estado estacionario menor del 5% pues estos son cero en 3 de los casos y despreciable en el restante. A simple vista es difícil decidir cuál de ellos presenta un mejor comportamiento. Por ejemplo MPC parece tener un rendimiento mayor al resto de controladores y si la escogencia solo estuviera basada en esta respuesta, MPC sería el escogido sin embargo es posible observar en la figura 41 que MPC tiene problemas al estabilizar el sistema ante perturbaciones.

En la figura 40 es aún más complicado obtener un criterio de selección cualitativo pues las respuestas de los diferentes controladores son muy similares, tanto que están sobrepuestas unas con otras. Este hecho pone en claro que la escogencia de un controlador por medio de parámetros cuantitativos es necesaria

A continuación se muestra un comparativo de la capacidad de reacción ante perturbaciones de los controladores.

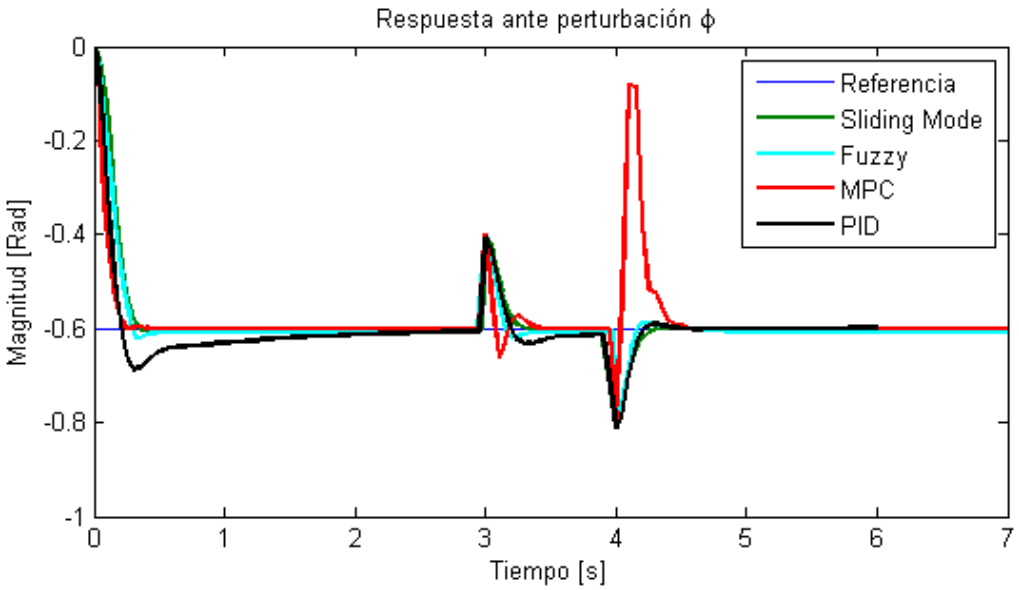


Figura 41. Respuesta ante perturbaciones.

Como se mencionó anteriormente MPC tuvo problemas al lidiar con las perturbaciones como se muestra en la figura 41; hecho que le quita puntos a la hora de realizar la comparación cuantitativa como se verá más adelante.

2.4.2 Control de posición. Se eligieron diferentes controladores para ser puestos a prueba tanto para el sistema de control de posición horizontal (SPH) como para el sistema de posición vertical (SPV).

Los controladores puestos a prueba fueron:

SPV: FUZZY, MPC,Sliding Mode Control
SPH: FUZZY, Sliding Mode Control

Las dinámicas verticales del sistema presentan una no linealidad debido a las variables $\theta\ y\ \phi$. Se propone una linealización por realimentación como se muestra en la Tabla XI para evitar fallas no deseadas en el controlador MPC, el cual brinda un mejor desempeño para sistemas lineales, y para mejorar el rendimiento en general de los controladores restantes.

Tabla 11. Linealización por realimentación

SPV no-lineal	$\dot{V}_z = g - \Gamma cos\phi\ cos\theta$ $\dot{Z} = V_z$
SPV Lineal	$\dot{V}_z = g - \Gamma_l$ $\dot{Z} = V_z$ $\text{Donde}: \Gamma = \frac{\Gamma_l}{cos\phi\ cos\theta}$ y $\phi\ y\ \theta$ son estimadas mediante el IMU.

2.4.2.1 Control FUZZY para SPV. Para el diseño del controlador Fuzzy se utilizó el bloque *Fuzzy Logic Controller*. y la herramienta *Fuzzy Logic Design.*

Los parámetros requeridos por el bloque Fuzzy, al igual que el controaldor FUZZY para orientación, son los mostrados en la Tabla VII.

El controlador Fuzzy requiere como entradas el error del estado frente a una referencia (*ez*) y la derivada del mismo (*dez*). Análogamente al sistema de control FUZZY de orientación la derivada del error fue reemplazada por velocidad lineal (V_z), la cual se mide mediante la estimación proporcionada por el Sistema de Visión Artificial.

Para el controlador de Posición Vertical de altura (Z) se realizó la siguiente tabla de normas Fuzzy para las variables difusas *ez y* V_z.

Tabla 12. Normas Fuzzy para las variables ez y V_z.

ez \ V_z	NN	N	O	P	PP
NN	O	P	PP	PP	PP
N	N	O	P	PP	PP
O	NN	N	O	P	PP
P	NN	NN	N	O	P
PP	NN	NN	NN	N	O

Convenciones	
Muy positivo	PP
Positivo	P
Neutral	O
Negativo	N
Muy negativo	NN

Se sintonizó los parámetros de Funciones de membresía requeridas para el proceso de Fuzzyficación y defuzzyficación del controlador mediante métodos empíricos. Los parámetros que brindaron un mejor desempeño fueron:

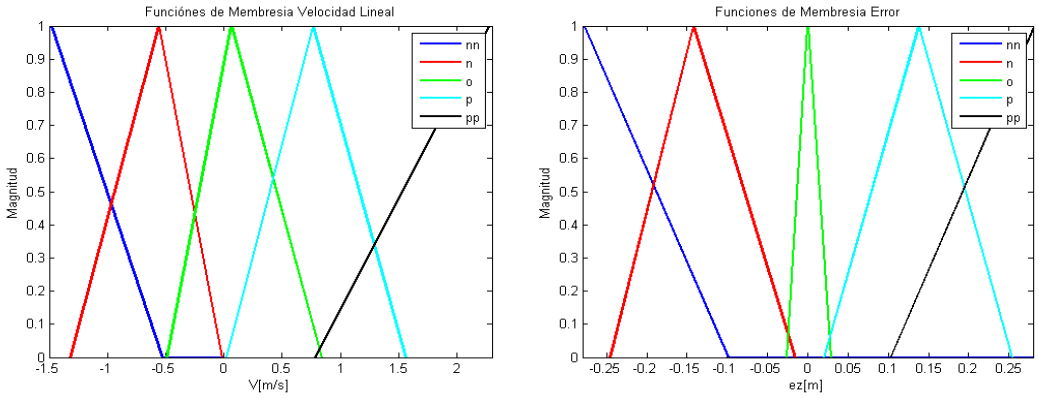


Figura 42. Funciones de Membresía de Entradas del controlador Fuzzy, Error en la altura (Z) y Velocidad Lineal en Z.

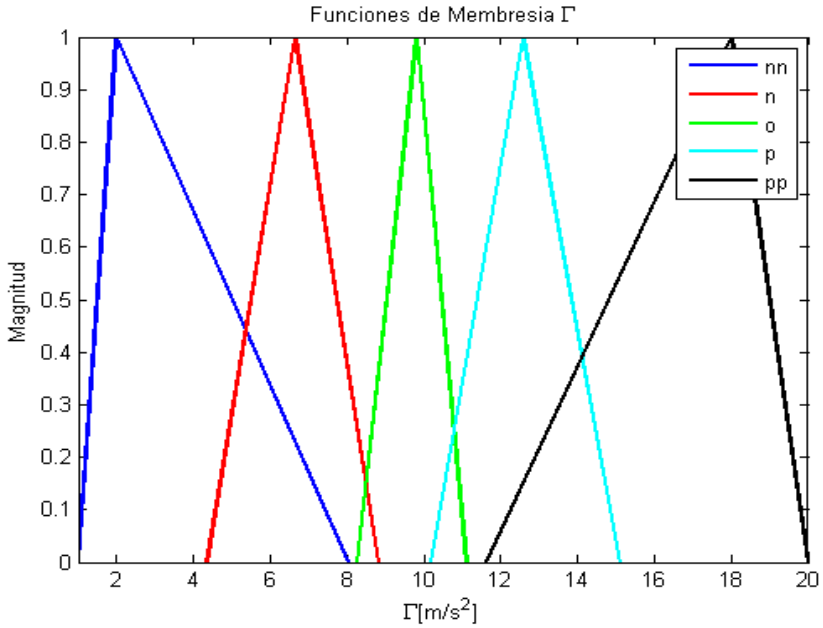


Figura 43. Funciones de Membresía Salida del controlador Fuzzy (Γ)

Se resalta que las funciones de membresía para *ez y* V_z*(*Figura 42) entran en saturación una vez superado sus entradas tanto en el eje positivo como el negativo.

A continuación se muestra el mapeo FUZZY generado por este controlador:

Mapeo FUZZY para controlador de posición vertical

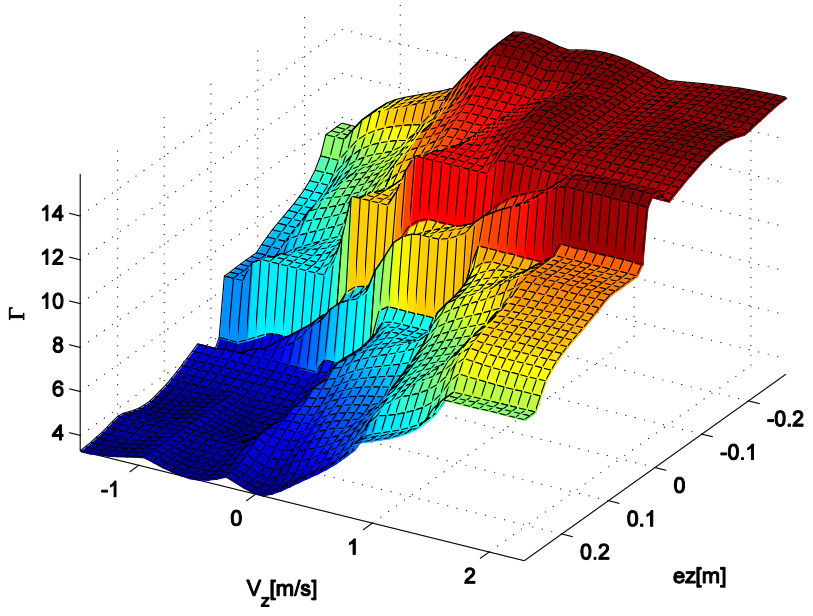


Figura 44. Mapeo generado por el controlador FUZZY.

2.4.2.2 Controlador FUZZY para SPH. El controlador Fuzzy para Posición Horizontal se diseñó de manera similar a sus semejantes y se utilizó las mismas herramientas. La tabla de reglas Fuzzy utilizadas se muestra en la tabla XII. Las funciones de membresía de entradas y salida, las cuales fueron sintonizadas de forma empírica, se muestran a continuación.

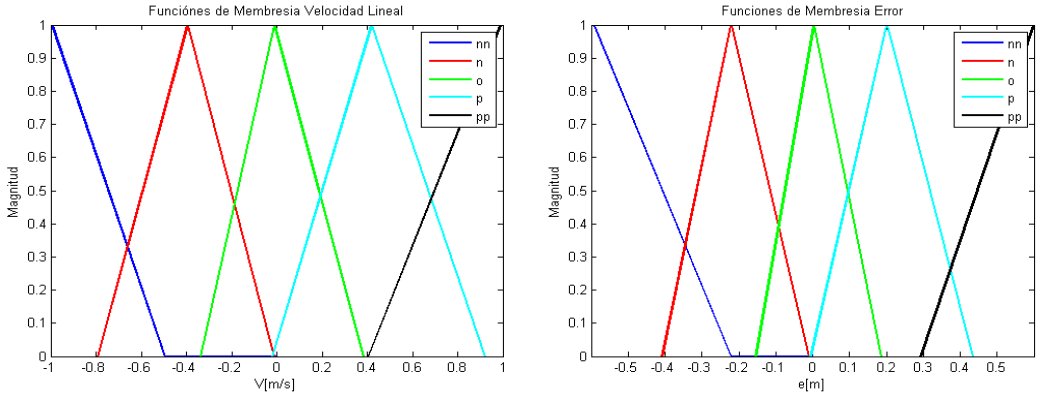


Figura 45. Funciones de Membresía de Entradas del controlador Fuzzy, Error en la posición XY y Velocidad Lineal.

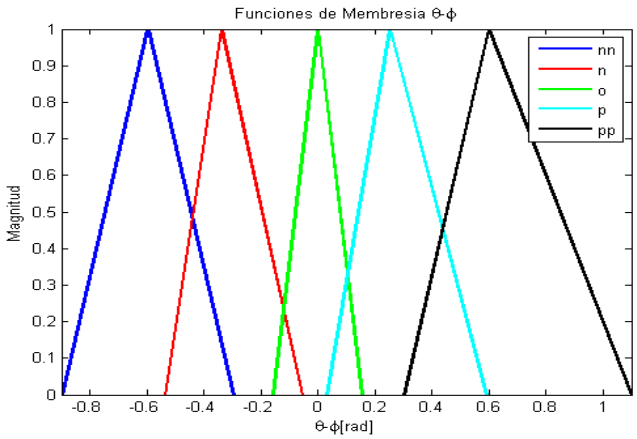


Figura 46. Funciones de Membresía Salida del controlador Fuzzy (Theta y Phi)

Se resalta que las funciones de membresía ara e_h $y\ V_h$ (Figura 45) entran en saturación una vez superado sus entradas tanto en el eje positivo como el negativo.

El mapeo FUZZY generado por este controlador se muestra a continuación:

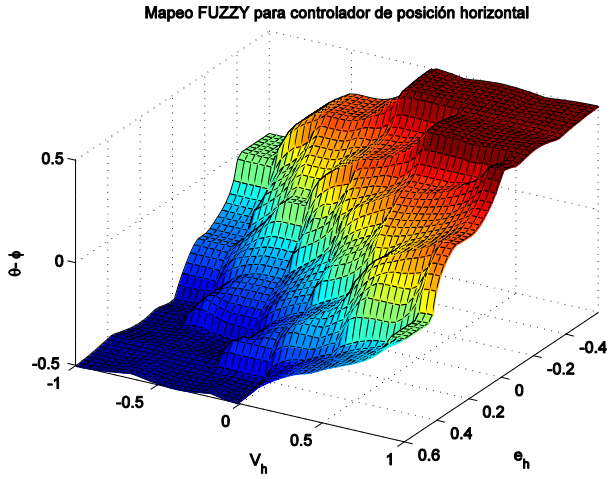


Figura 47. Mapeo FUZZY para control de posición horizontal.

2.4.2.3 Model Predictive Control (MPC) para SPV. Para su simulación se utilizó el bloque *MPC Controller* de Simulink y para su diseño la aplicación *MPC Design* de MATLAB. Los parámetros se calibraron de manera empírica y se tuvo en cuenta las restricciones físicas del Vehículo.

A continuación se muestra un resumen de los parámetros y restricciones utilizados para el diseño del controlador.

Tabla 13. Parámetros controlador MPC para posición vertical

Parámetro:	Valor
Intervalo de control	0.05s
Horizonte de Predicción	10
Horizonte de Control	1
Restricciones en el estado	[-50 0]
Restricciones en la Entrada G	[5 50]

MPC utiliza un modelo linealizado del sistema de movimiento en Z. Este sistema Linealizado no tiene en cuenta los cambios del modelo que ocurren al solicitar maniobras en X y Y ,lo que ocasiona que el controlador no sea capaz de estimar la señal de control necesaria para realizar su tarea. En la figura 48 se muestra un ejemplo de este hecho.

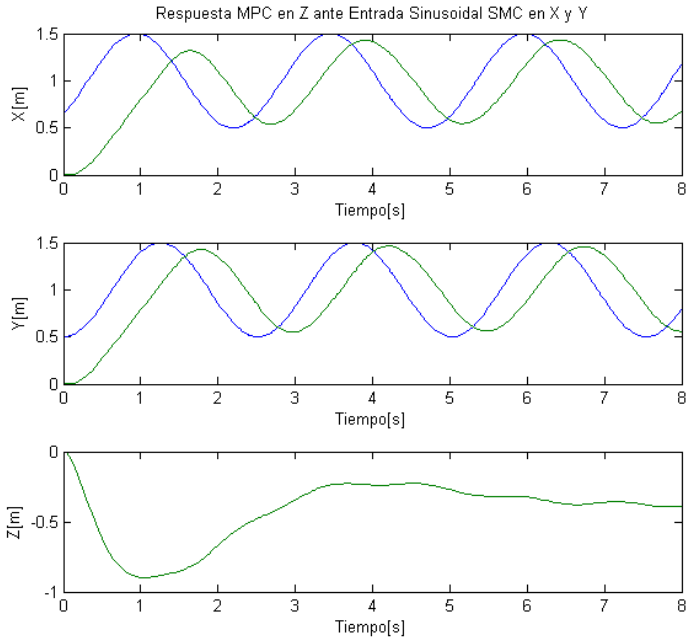


Figura 48. Respuesta escalón MPC en Z, Ante solicitud de movimiento de Sliding Mode Control en X y Y.

Una solución para este tipo de comportamientos es el propuesto con la linealización por realimentación descrita en la Tabla XI.

Los resultados del método propuesto se muestran a continuación.

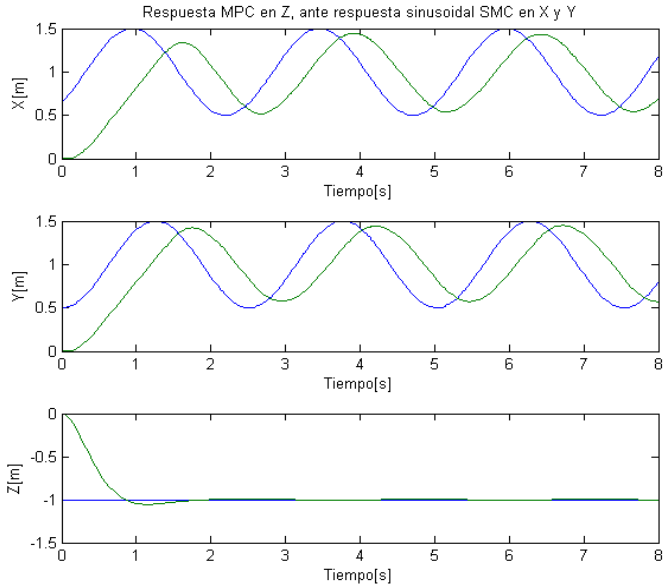


Figura 49. Respuesta escalón MPC en Z, Ante solicitud de movimiento en X y Y, Planta linealizada por realimentación.

2.4.2.4 Sliding Mode Control Para SPH y SPV. Se planteó una estrategia similar de diseño del controlador de Sliding Mode para Control Horizontal y Vertical, a las aplicadas en los controladores de Orientación.

Se calcularon los parámetros del controlador cumpliendo con la inecuación 42. Los resultados de los cálculos realizados fueron calibrados manteniendo la validez de la inecuación para obtener el comportamiento deseado.

A continuación se muestra los valores finales obtenidos para estos controladores.

Tabla 14. Parámetros Sliding Mode controlador para posición vertical y horizontal

Controlador	a	b	k
Vertical	11	-2	9
Horizontal X	0.62	-1	0.6
Horizontal Y	0.62	-1	-0.6

2.4.2.5 Gráficos comparativos de controladores de posición. Se realizó pruebas sobre el comportamiento a la respuesta al escalón de los distintos controladores para posición Vertical y Horizontal. Pruebas ante perturbaciones fueron realizadas también. A continuación se muestra los resultados obtenidos y el proceso de elección del mejor controlador.

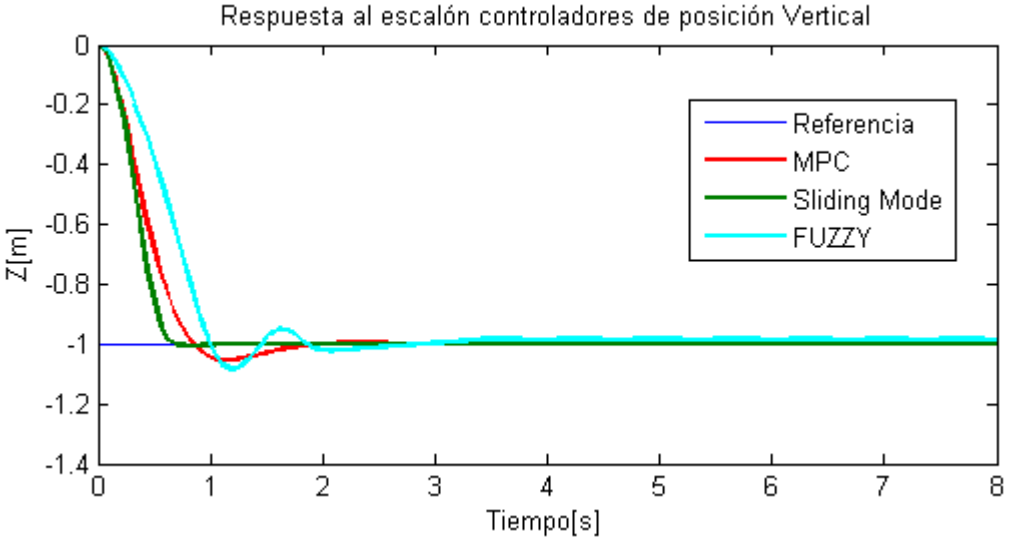


Figura 50. Controladores de Posición Vertical, Escalón en X y Y en 3s.

Sliding Mode Control obtuvo muy buenos resultados en su respuesta al escalón sobre el controlador de posición Z, es el único sin sobrepaso, esto repercutirá positivamente en la comparación cuantitativa.

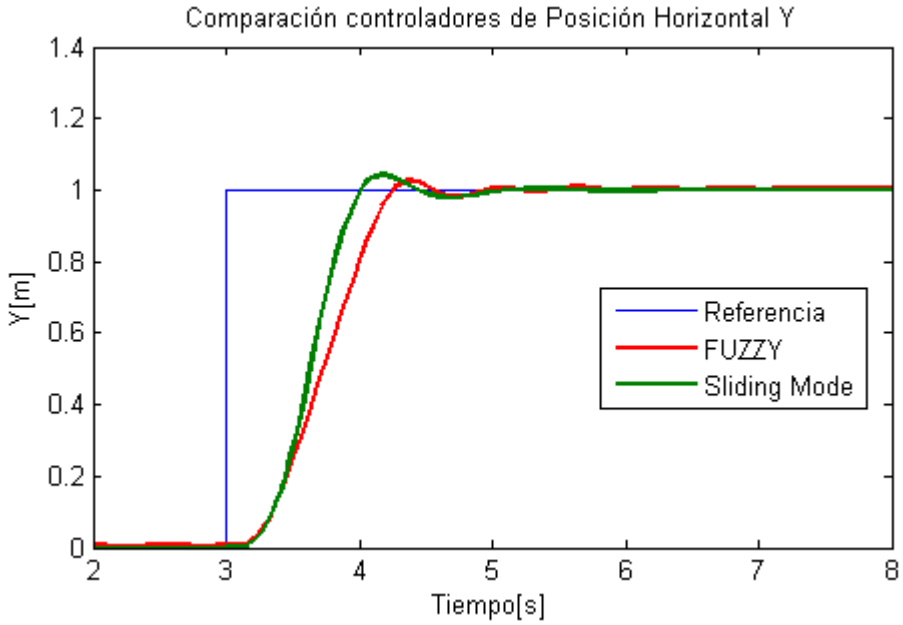


Figura 51. Controladores de Posición Horizontal Y.

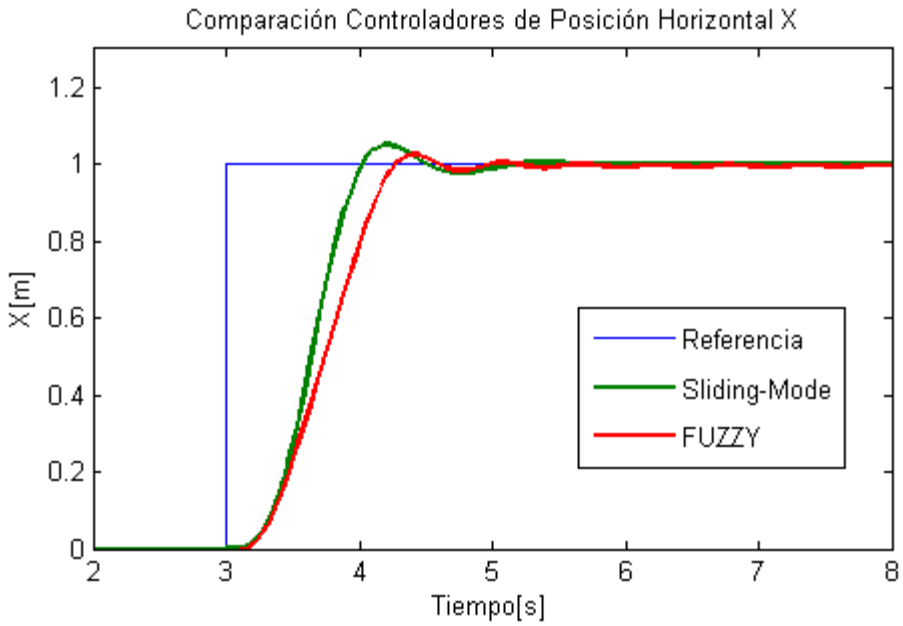


Figura 52. Controladores de Posición Horizontal X.

La respuesta al escalón de los controladores de posición horizontal muestra a un controlador FUZZY con menor sobrepaso que SMC, e incluso el tiempo de establecimiento parece ser similar. La comparación cuantitativa despejará las dudas sobre cuál de los dos es el adecuado para realizar el trabajo.

Los resultados de la respuesta de los controladores ante entradas Sinusoidales se muestran a continuación:

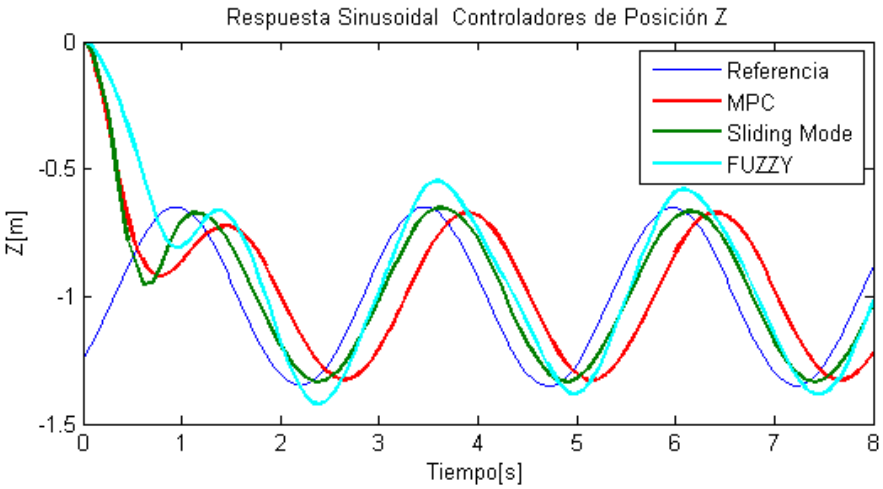


Figura 53. Controladores de Posición Vertical Entrada Sinusoidal, Escalón en X y Y en 3s.

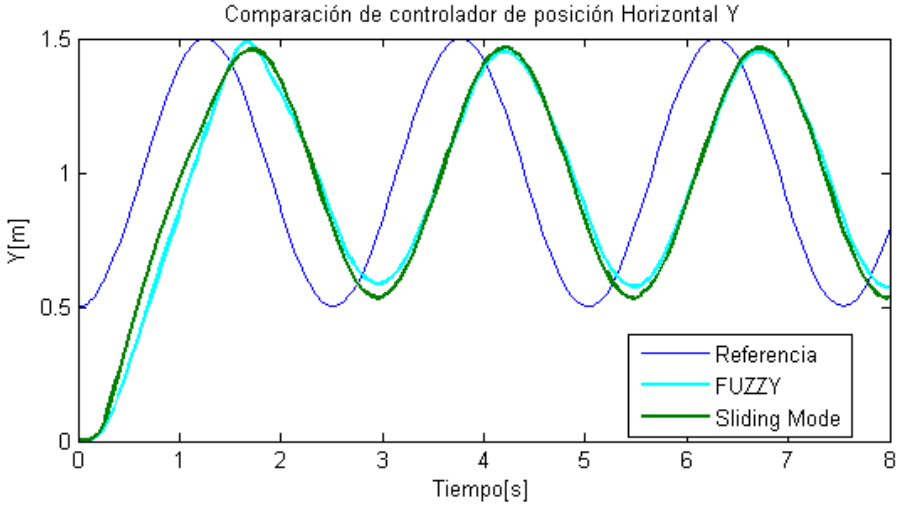


Figura 54. Controladores de Posición Horizontal Y, Entrada Sinusoidal.

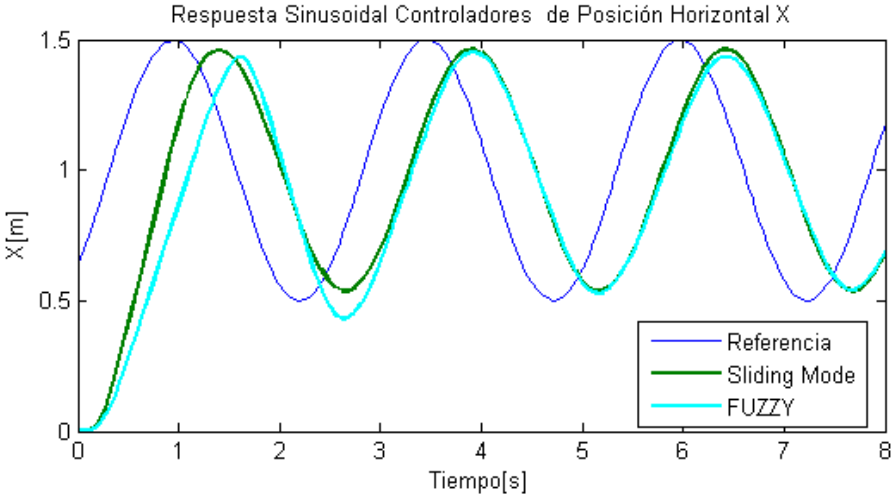


Figura 55. Controladores de Posición Horizontal X, Entrada Sinusoidal.

La respuesta sinusoidal de los controladores solo difiere en el estado transitorio inicial, después de eso los controladores se comportan prácticamente igual. Nuevamente será la comparación cuantitativa la herramienta que permita un análisis más a fondo del comportamiento de estos controladores.

La perturbación mostrada en la figura 38 fue inducida sobre la variable Z. Se simuló la respuesta de cada controlador ante esta perturbación.

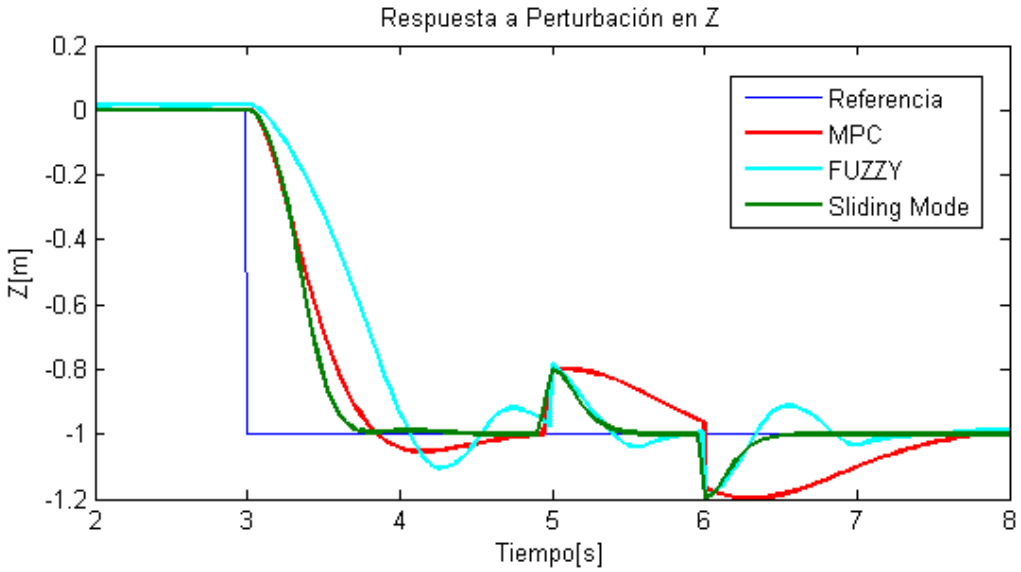


Figura 56. Respuesta ante perturbación en Z.

Como se observa en la figura 56, MPC nuevamente tiene un rendimiento inferior que los controladores FUZZY y SMC respecto a la estabilización ante perturbaciones. La falla no es tan exagerada como en el caso de los controladores de orientación sin embargo no deja de ser un inconveniente de la aplicación en particular del MPC.

La respuesta a perturbaciones de los controladores de posición horizontal X y Y se muestra a continuación.

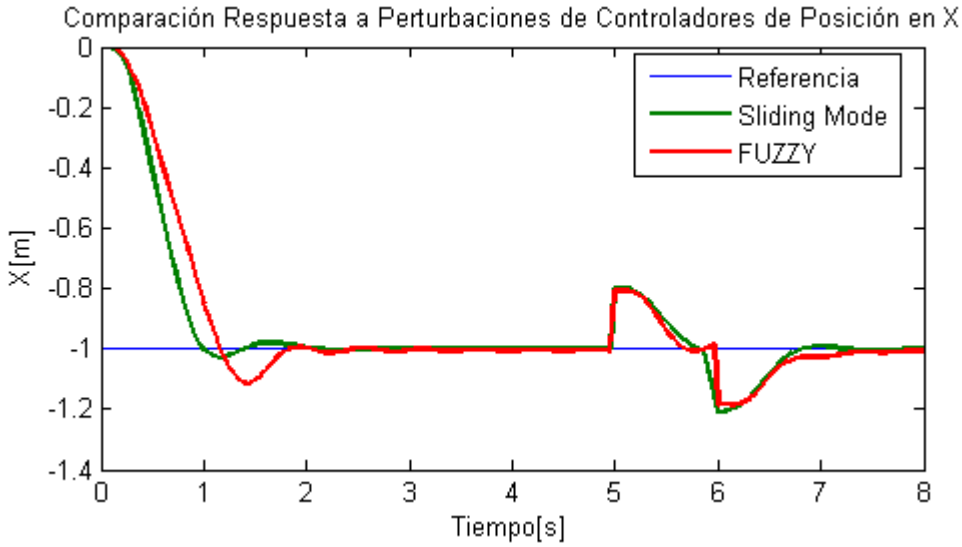


Figura 57. Respuesta ante perturbación en X.

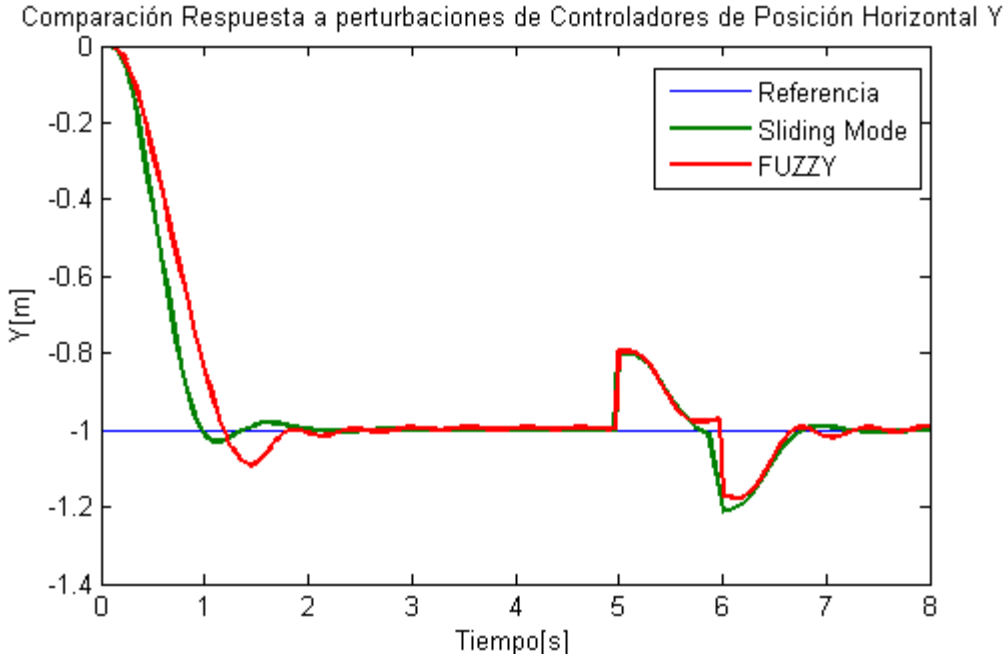


Figura 58. Respuesta ante perturbación en Y.

2.5 COMPARACIÓN CUANTITATIVA DE CONTROLADORES

Para realizar la comparación cuantitativa de los controladores diseñados fueron utilizados algunos parámetros de conocimiento común como lo son el IAET (Integral del valor absoluto del error ponderado en el tiempo), ICE (integral del error cuadrático), tiempo de establecimiento y sobreimpulso (sobrepaso), así como la correlación del espectro (*crrFFT*) el cual es un parámetro de proposición propia.

2.5.1 Descripción de parámetros de Rendimiento:

2.5.1.1 Integral del valor absoluto del error ponderado en el tiempo (IAET). Para el cálculo del parámetro IAET se utilizó la siguiente ecuación [16] aplicada a la respuesta al escalón. Se programó un script en MATLAB para calcular el parámetro.

$$IEAT = \int_0^6 t * |e(t)| dt$$

Ecuación 45.

Donde e es la señal de error ante la respuesta escalón.

2.5.1.2 Sobreimpulso (OS). Sea t_p el tiempo en estado transitorio donde la salida temporal alcanza el primer máximo como respuesta a una referencia.A la diferencia entre ese valor máximo y el valor de régimen permanente que se alcanza cuando el error es nulo o es limitado ante una entrada escalón, se llama sobreimpulso máximo OS Analíticamente se puede calcular como[25]:

$$OS = \frac{C(t_p) - C(\infty)}{C(\infty)}$$

Ecuación 46.

Donde $c(t)$ la respuesta al escalón del sistema.

A continuación se muestra un gráfico que ilustra el concepto de sobrepaso.

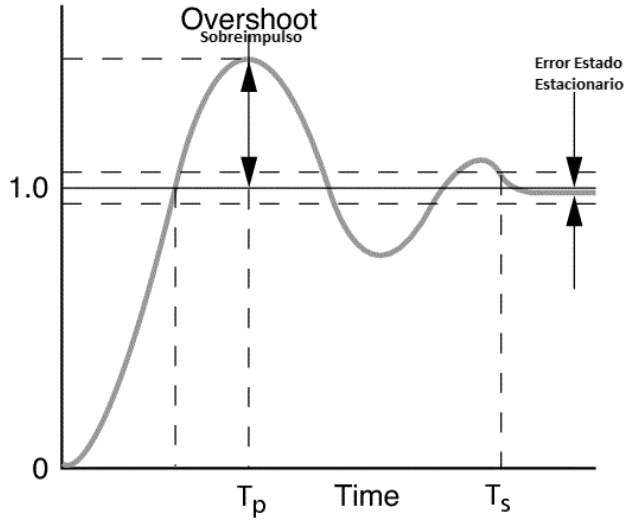


Figura 59. Ilustración concepto de sobrepaso.

[25] GIL, y DÍAZ, Op. Cit.

2.5.1.3 Tiempo de establecimiento(ST). Se define como el instante a partir del cual la respuesta transitoria oscila dentro de una franja alrededor del valor en estado estacionario no mayor al 2%[26].

2.5.1.4 CrrFFT(Correlación del Espectro). Se propone el parámetro de rendimiento: Correlación del Espectro, el cual mide que tan parecida es la forma de la respuesta del sistema con respecto a la referencia sin importar el retardo de fase generado por el tiempo de respuesta mediante la siguiente ecuación.

$$CrrFFT(ref, f(t)) = crr\left(\left|FFT\left(ref(t)\right)\right|, \left|FFT\left(f(t)\right)\right|\right)$$

Ecuación 47.

$$Crr(x, y) = \frac{\sum_{i=1}^{N}(x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\sum_{i=1}^{N}(x_i - \bar{x})^2}\sqrt{\sum_{i=1}^{N}(y_i - \bar{y})^2}}$$

Este cálculo se realiza sobre la respuesta de un sistema ante referencias variables; en los casos en estudio descritos en este trabajo, las pruebas de respuesta a entrada sinusoidal.

2.5.1.5 Integral del error cuadrático (ICE). El parámetro de rendimiento ICE[27] se comporta de manera similar al IAET, solamente que no pondera respecto del Tiempo. El ICE fue utilizado para evaluar el rendimiento de los controladores en respuesta a perturbaciones, ya que causa una mayor penalización entre mayor sea el error. De igual forma que el IAET se calculó mediante un script en MATLAB.

$$ICE = \int_0^8 e(t)^2 dt$$

Ecuación 48.

2.5.2 Comparación y elección de controlador de orientación. La elección del controlador de Orientación se llevó a cabo realizando una ponderación de las características Cuantitativas. Los parámetros de rendimiento Cuantitativos se normalizaron sobre el más óptimo de cada grupo para hacer posible su ponderación. Para evaluar el rendimiento de los controladores respecto a perturbaciones se utilizó el parámetro ICE(P).

[26] Ibíd.

[27] LOPEZ, A.M.; MILLER, J.A.; SMITH, C.L., y P.W. Tuning Controllers with Error-Integral Criteria. EEUU: Murrill, Instrumentation Technology, 1976.

Tabla 15. Parámetros de rendimiento controladores de orientación

Controlador	Características Cuantitativas				
	IAET (15%)	OS (10%)	ST (30%)	Crr FFT (15%)	ICE(P) (30%)
MPC	6.5316e-04	0.0061	0.2	0.9789	0.0515
PID	0.0295	0.1188	2.12	0.9991	0.0348
FUZZY	0.1185	0.0412	0.39	0.9989	0.0382
SLIDING MODE	0.0020	0.0023	0.3	0.9998	0.0458

Los valores de los parámetros de rendimiento fueron normalizados entre 0 y 100. Correspondiendo 100 a la mejor respuesta y 0 a la peor respuesta. Para obtener una medida del desempeño total de cada controlador los valores normalizados de sus parámetros fueron ponderados como se muestra en la siguiente tabla.

Tabla 16. Parámetros de rendimiento normalizados

Controlador	Características Cuantitativas					TOTAL(0-100)
	IAET (15%)	OS (10%)	ST (30%)	Crr FFT (15%)	ICE(P) (30%)	
MPC	100	96.74	100	0	0	54.674
PID	74.96	0	0	96.65	100	55.7415
FUZZY	0	66.61	90	95.69	79.64	71.9065
SLIDING MODE	98	100	94.79	100	34.13	78.376

Al obtener mayor puntuación de acuerdo a la ponderación escogida sobre los parámetros de rendimiento el control elegido para realizar las tareas de orientación fue Sliding Mode Control.

2.5.3 Comparación y Elección de controlador de Posición Vertical. La elección del controlador de Posición Vertical se llevó a cabo realizando una ponderación de las características Cuantitativas. Los parámetros de rendimiento se normalizaron sobre el más óptimo de cada grupo para hacer posible su ponderación. Para evaluar el rendimiento de los controladores respecto a perturbaciones se utilizó el parámetro ICE(P).

Tabla 17. Parámetros de rendimiento controladores de posición vertical

Controlador	Características Cuantitativas				
	IAET (15%)	OS (10%)	ST (30%)	Crr FFT (15%)	ICE(P) (30%)
MPC	0.1441	0.0269	1.85	0.9658	0.3180
FUZZY	0.7075	0.0418	1.63	0.9789	0.4736
SLIDING MODE	0.0696	0.0033	0.62	0.9844	0.2603

Tabla 18. Parámetros de rendimiento normalizados controladores de posición vertical

Controlador	Características Cuantitativas					TOTAL(0-100)
	IAET (15%)	OS (10%)	ST (30%)	Crr FFT (15%)	ICE(P) (30%)	
MPC	88.32	61.29	0	0	72.94	41.26
FUZZY	0	0	17.88	70.43	0	15.93
SLIDING MODE	100	100	100	100	100	100

Los resultados de la comparación ponderada muestran que Sliding Mode Control presenta un mejor rendimiento para realizar el tracking de Posición Vertical.

2.5.4 Comparación y Elección de controlador de Posición Horizontal. La elección del controlador de Posición Horizontal se llevó a cabo realizando una ponderación de las características Cuantitativas. Los parámetros de rendimiento se normalizaron sobre el más óptimo de cada grupo para hacer posible su ponderación.

Tabla 19. Parámetros de rendimiento controladores de posición horizontal

Controlador	Características Cuantitativas									
	IAET (15%)		OS (10%)		ST (30%)		Crr FFT (15%)		ICE(P) (30%)	
	X	Y	X	Y	X	Y	X	Y	X	Y
FUZZY	2.5096	2.4673	0.0535	0.0604	1.8	1.8	0.9827	0.9916	0.5412	0.5393
SLIDING MODE	2.1333	2.0855	0.0205	0.0145	1.9	1.3	0.9722	0.9816	0.4512	0.4472

Tabla 20. Parámetros de rendimiento normalizados controladores de posición horizontal

Controlador	Características Cuantitativas										TOTAL (0-100)	
	IAET (15%)		OS (10%)		ST (30%)		Crr FFT (15%)		ICE(P) (30%)			
	X	Y	X	Y	X	Y	X	Y	X	Y	X	Y
FUZZY	0	0	0	0	100	0	100	100	0	0	45	15
SLIDING MODE	100	100	100	100	0	100	0	0	100	100	55	85

Los resultados de la comparación ponderada muestran que Sliding Mode Control presenta un mejor rendimiento para realizar el tracking de Posición Horizontal tanto sobre el eje X como sobre el eje Y.

2.6 IMPLEMENTACIÓN REAL

El controlador definitivo basado en Sliding Mode Control mostrado en la Tabla XXI fue programado en Arduino. A continuación se muestra un esquema general de la implementación.

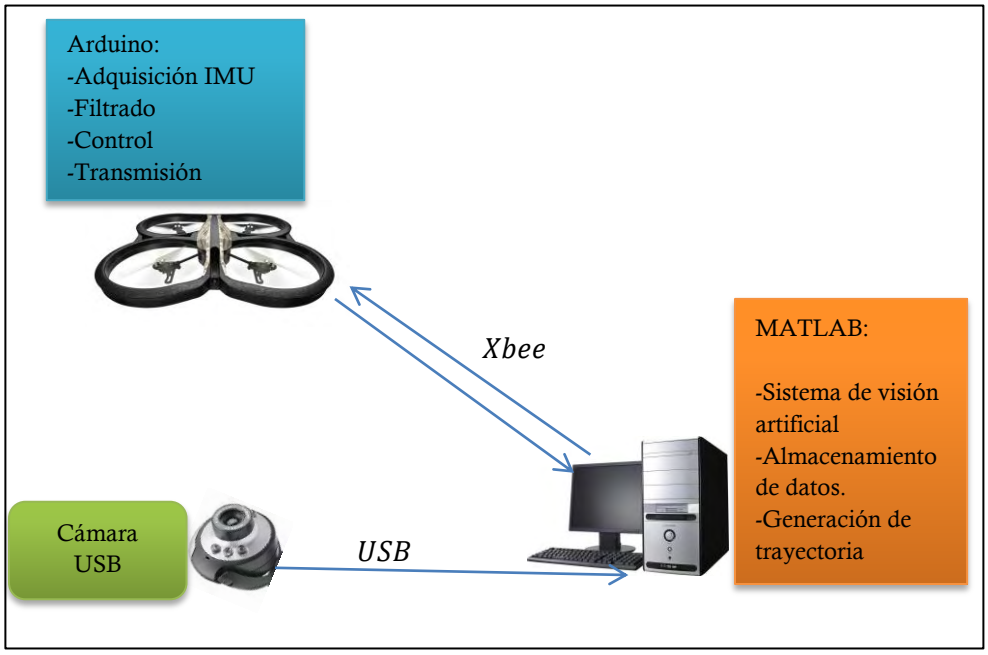


Figura 60. Esquema general de la implementación.

2.6.1 Arduino. Como se mencionó en secciones anteriores el embebido Arduino es el encargado de realizar las siguientes tareas:

- Adquisición IMU
- Filtrado
- Control
- Transmisión Xbee

A continuación se muestra el diagrama de flujo de los procesos realizados por Arduino:

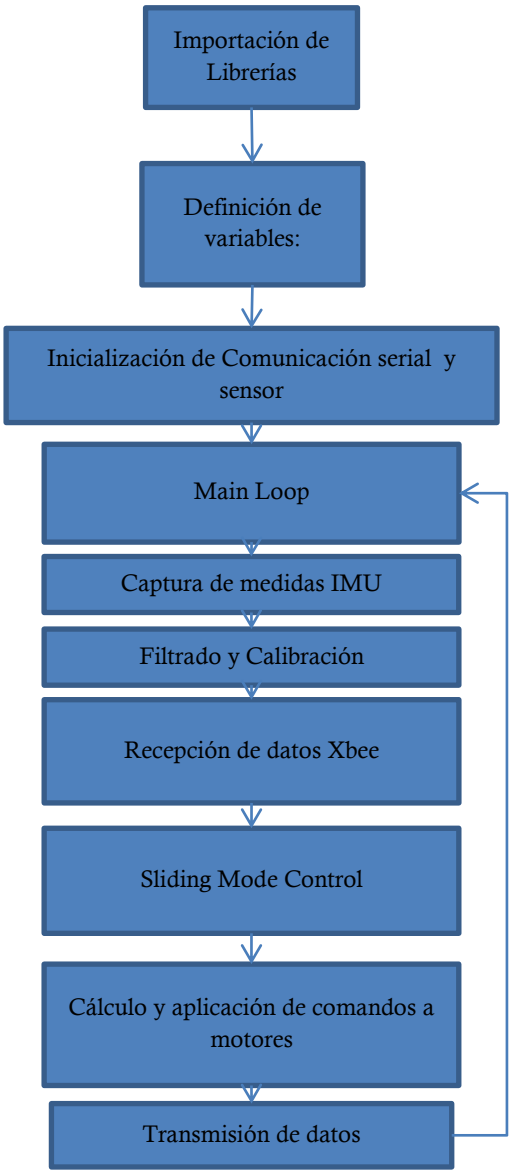


Figura 61. Diagrama de flujo de los procesos realizados por Arduino.

Importación de librerías: Las librerías utilizadas para la programación así como su uso y fuente se muestran a continuación.

Definición de Variables: Se inicializan todas las variables y constantes a utilizar en cada

una de las etapas subsiguientes, incluyendo variables y constantes requeridas por el objeto IMU(*sixDOF*) y objetos tipo *servo* de los motores brushless .

Inicialización de comunicación serial y sensor: Esta etapa se codifica en la sección *setup()* del *skectch* de Arduino y se encarga de inicializar la comunicación serial a 115200 baudios para el XBee; inicializar la comunicación *wire(I2C)* para la comunicación con el IMU Digital Combo Board; configurar el IMU para trabajar a resolución máxima(13 bits) y un rango de $\pm 4g$ e inicializarlo; realizar la función de armado de los motores, para lo cual se creó un procedimiento *arm()* que realiza esta tarea ;

Capturas de medidas IMU: El distribuidor del producto adquirido (IMU Digital Combo Board) proporcionó las librerías para su utilización. Los comandos *sixDOF.gyro.readGyro(yprp)* y *sixDOF.getYawPitchRoll(ypr)* capturan los vectores de velocidad y posición angular en los vectores *yprp* y *ypr* respectivamente.

Filtrado y calibración: Las señales provenientes de la captura de IMU fueron filtradas con un filtro simple IIR para evitar ruido de alta frecuencia. Seguidamente, se realizó el ajuste de calibración realizado en la sección Calibracion de sensores.

Recepción de datos XBee: Si hay datos disponibles en el buffer del puerto serial XBee, se leen estos datos y se almacenan en sus respectivas variables. Los datos recibidos son: posición X,Y y Z; orientación yaw; y trayectoria; los cuales provienen del sistema de visión artificial.

En el caso de operación manual se reciben datos solamente de señales de referencia de orientación.

Sliding Mode Control: Esta etapa se encarga de calcular $\mathrm{L}, \mathrm{M}, \mathrm{N}, \Gamma$ a partir de las leyes de control obtenidas en la etapa de diseño.

Cálculo y aplicación de comandos a motores: A partir de L, M, N, Γ se calcula w_{e1}, w_{e2}, w_{e3}y w_{e4} teniendo en cuenta las ecuaciones de la tabla IV .Además se realiza los cambios de variable necesarios para realizar el ajuste de offset obtenido en la sección estimación de parámetros.

$$w_{e1} = \sqrt{\frac{\Gamma - \Gamma_0}{b} - \frac{N}{d} - \frac{2L}{d}} + w_{e0}$$

$$w_{e2} = \sqrt{\frac{\Gamma - \Gamma_0}{b} + \frac{N}{d} - \frac{2M}{d}} + w_{e0}$$

$$w_{e3} = \sqrt{\frac{\Gamma - \Gamma_0}{b} - \frac{N}{d} + \frac{2L}{d}} + w_{e0}$$

$$w_{e4} = \sqrt{\frac{\Gamma - \Gamma_0}{b} + \frac{N}{d} + \frac{2L}{d}} + w_{e0}$$

Seguidamente se aplica a cada motor i una señal de control que equivale a $w_{ei} + 1060$, ya que el ESC asociado a los motores acepta tren de pulsos de duración 1060uS y 1500 uS a una frecuencia de $50Hz$ para llevar a los motores desde velocidad mínima (0) hasta la máxima respectivamente. El comando utilizado en Arduino para realizar esta tarea es: $myservo1.writeMicroseconds(w_{e1});$

Transmisión de datos: Finalmente se realiza una transmisión de estados de orientación a través de XBee hacia MATLAB para su visualización y almacenamiento.

2.6.2 Matlab. El sistema de visión artificial, el sistema de generación de trayectoria y el sistema almacenamiento de datos fueron programados en un script de MATLAB. Inicialmente MATLAB también se encargaba de permitir la visualización de los estados dinámicos del cuadricóptero, sin embargo los procesos de las funciones *plot e imshow* de MATLAB tenían un alto costo computacional, lo que se veía reflejado en una estimación de estados a bajas frecuencias de muestreo.

3. RESULTADOS

3.1 CONTROLADOR DEFINITIVO

A continuación se muestra un consolidado de los controladores escogidos a partir de la comparación cuantitativa.

Tabla 21. Consolidado de controladores.

Sub-Sistema	Controlador	Parámetros		
		a	***b***	***k***
SGO	Sliding Mode Control	7	0.9	30
SPH-X	Sliding Mode Control	0.62	-1	0.6
SPH-Y	Sliding Mode Control	0.62	-1	-0.6
SPV	Sliding Mode Control	11	-2	9

Simulaciones del controlador definitivo ante la solicitud de diferentes trayectorias fueron realizadas. A continuación se muestra ejemplos de ellas.

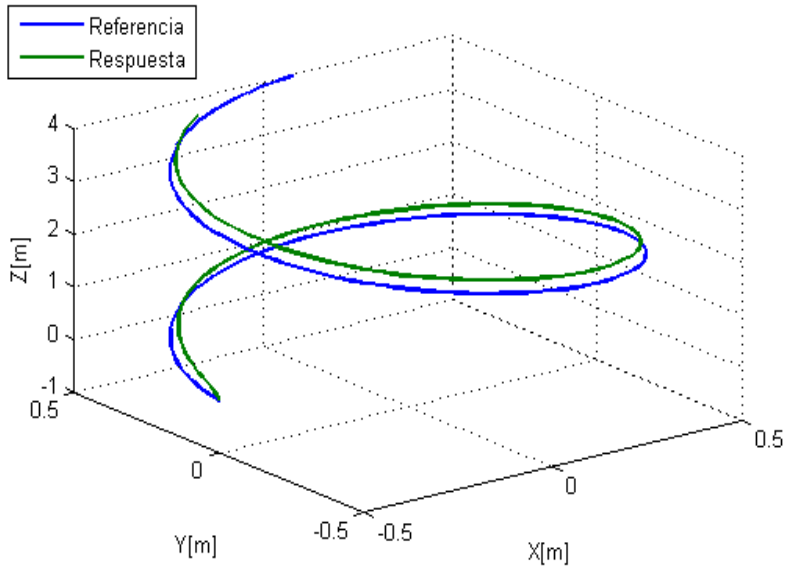


Figura 62. Simulación respuesta a trayectoria Variable XYZ Controlador Definitivo.

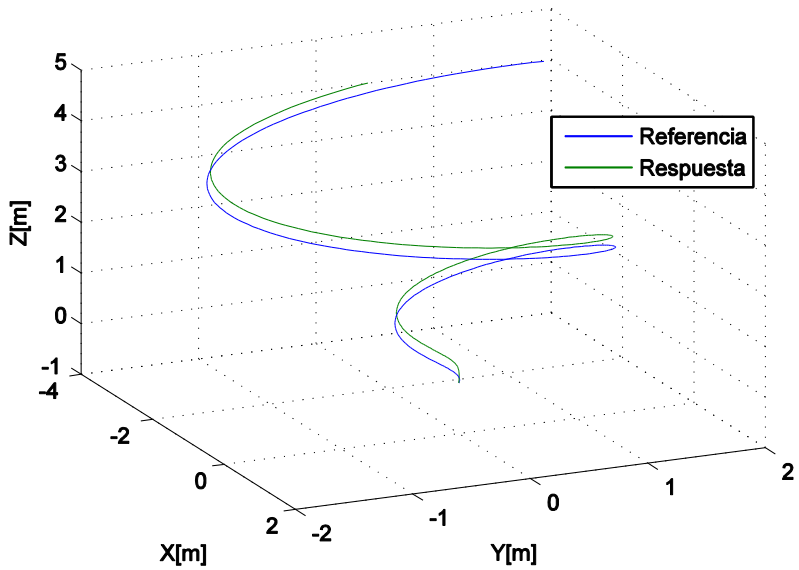


Figura 63. Simulación respuesta a trayectoria Variable XYZ Controlador Definitivo.

Es posible observar en la figura 62 y 63 que el controlador SMC fue capaz de realizar la trayectoria solicitada por la referencia, sin embargo no lo hace a la par, es decir, presenta un retardo de fase ya observado en la sección de gráficos comparativos.

El controlador fue programado en el embebido ARDUINO y puesto a prueba en un entorno real. La respuesta real de controlador se muestra en la siguiente sección.

3.2 RESULTADOS IMPLEMENTACIÓN REAL

El controlador definitivo mostrado en la tabla XXI. Fue programado en el embebido ARDUINO. Pruebas de los controladores de orientación fueron realizadas inicialmente por separado. Los parámetros de los controladores Sliding Mode tuvieron que ser ajustados para obtener el comportamiento deseado.

A continuación se muestra la respuesta del controlador SGO-Roll:

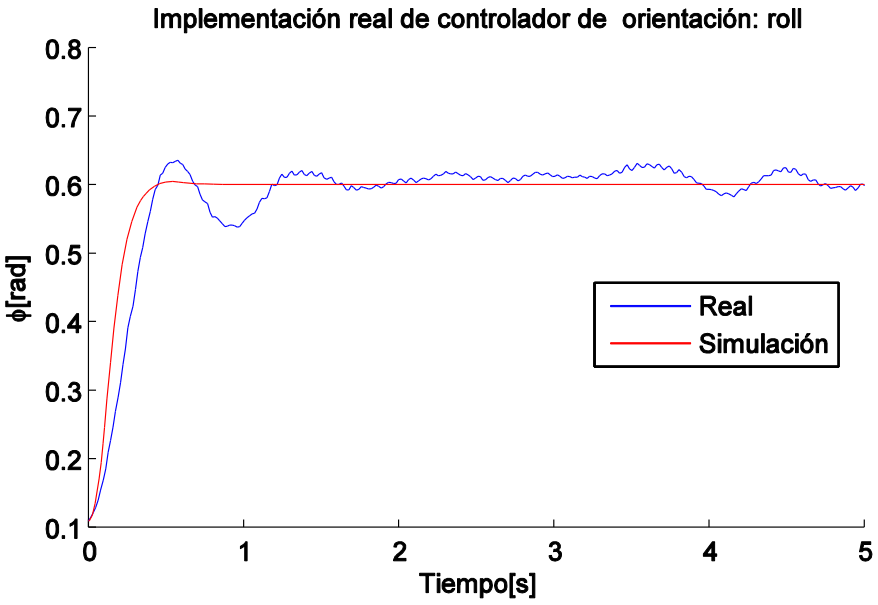


Figura 64. Respuesta del controlador SGO-Roll.

A continuación se muestra la respuesta del controlador SGO-Pitch:

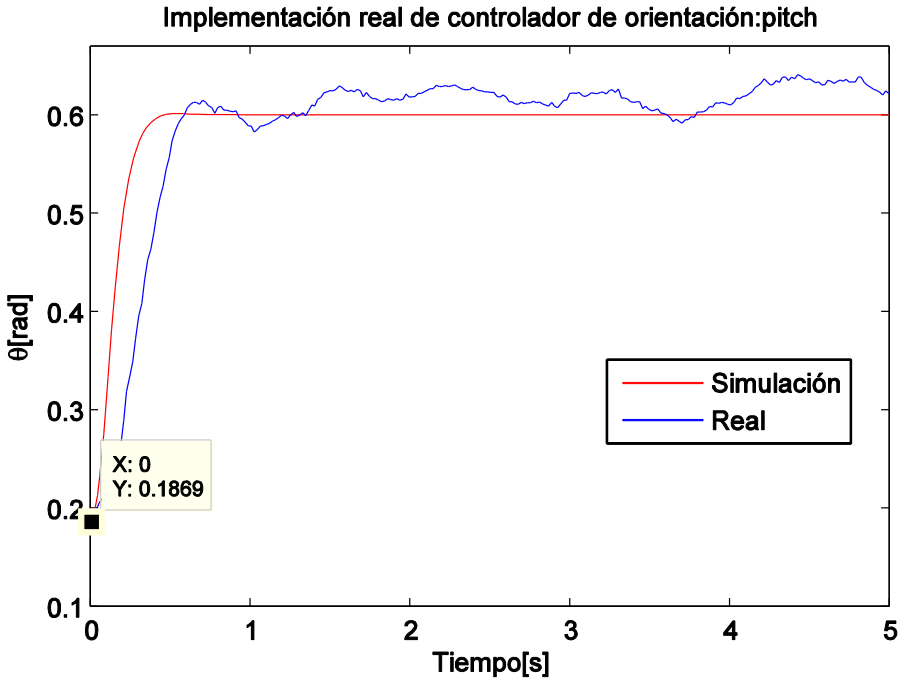


Figura 65. Respuesta del controlador SGO-Pitch.

El sistema de control implementado en el embebido Arduino muestra un comportamiento similar a los resultados obtenidos mediante simulación, sin embargo es posible apreciar un rizado en estado estable debido al ruido de vibración de los motores, el cual fue filtrado, sin embargo no fue posible su eliminación total pues filtros más selectivos generaban un retardo de fase indeseable

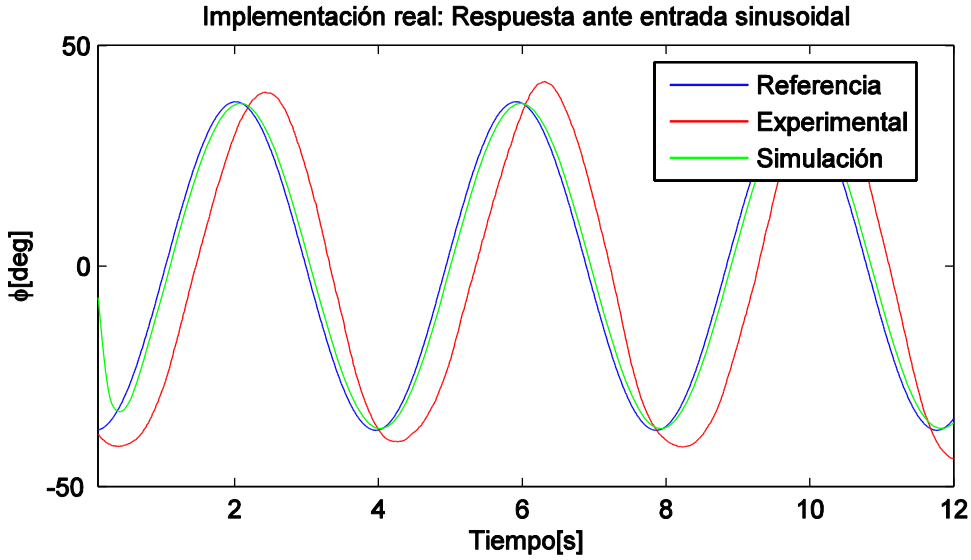


Figura 66. Respuesta del controlador SGO-Pitch ante entrada sinusoidal.

La respuesta ante entrada sinusoidal del sistema de control implementado presenta un retardo de fase mayor que el obtenido en simulación, además de responder con una amplitud ligeramente mayor, no obstante se considera un comportamiento aceptable. El ruido de vibración de los motores no afectó esta prueba pues el vehículo se encontraba en constante movimiento.

3.3 REPLICATOR DYNAMICS (RD)

Con el objetivo de proponer un controlador con mejores características se diseñó un ponderador de controladores basado en Replicator Dynamics. Las características que se quieren mejorar con el nuevo diseño a nivel de simulación son: menor tiempo de establecimiento, mayor robustez ante entradas fijas, mejor seguimiento de referencias variables.

Defínase los siguientes modos de funcionamiento y sus respectivas características.

Tabla 22. Modos de funcionamiento y características

MODO		Condición	Característica
Modo PI	**Tracking**	$dr \geq \beta$ $er \leq \alpha$	Disminución de error en Tracking
	Equilibrio	$dr < \beta$ $er \leq \alpha$	Robustez ante perturbaciones
Modo Arranque		$er > \sigma$	Bajo tiempo de levantamiento
Dónde: $\boldsymbol{\alpha, \sigma\ y\ \beta}$ son parámetros de sintonización y definen las zonas donde cada Modo debe tener mayor interés mediante las funciones de fitness asociadas a cada modo. $\boldsymbol{dr}$ Simboliza la diferencia absoluta entre el estado actual y anterior de la señal de referencia, $\boldsymbol{er}$ simboliza el error absoluto de estado del sistema.			

Defínase un ente llamado "CONTROLADOR" cuya tarea sea escoger una combinación lineal de 3 conjuntos de controladores en base a su condición de funcionamiento. Es decir la escogencia de una ley de control u como la que se muestra a continuación.

$$u = \tau_1 u_1 + \tau_2 u_2 + \tau_3 u_3$$

Ecuación 49.

Donde $0 < \tau_i \leq 1$ representa el porcentaje con el cual la señal de control u_i es aplicada.

τ_i es un porcentaje de ponderación por tanto se cumple la siguiente condición:

$$\sum_{i=1}^{3} \tau_i = 1$$

El CONTROLADOR elige a un número finito m de individuos (agentes) racionales los cuales están gobernados por Replicator Dynamics y se va a valer de la observación del comportamiento de dicha masa (m) para realizar su tarea. Cada uno de los agentes puede realizar la escogencia de una estrategia (Modo) basado en RD, estando cada Modo asociado a una función de bienestar (fitness) como se muestra en la Tabla XXII.

Adaptando la información descrita en la sección Marco Teórico sobre Replicator Dynamics al caso de estudio en particular, podemos describir las RD mediante la ecuación diferencial 51:

$$\dot{x}_i = \aleph x_i [f(x_i) - f_m] \;\; i = 1,2,3$$

Ecuación 50.

Dónde:

x_i Representa la porción de la masa m siguiendo la estrategia i.
$f(x_i)$ es el bienestar de los individuos siguiendo la estrategia i.
f_m es el promedio ponderado de los fitness asociados a las 3 estrategias.

$$f_m = \frac{1}{m}\sum_{i=1}^{3} x_i f(x_i)$$

Ecuación 51.

ℵ Es un parámetro de sintonización el cual escala la rapidez con la cual las porciones de la masa m cambian de estrategia.

En este orden de ideas es posible calcular τ_i mediante la siguiente expresión.

$$\tau_i = \frac{x_i}{m}$$

Cada modo de funcionamiento está ligado a un controlador Sliding Mode que cumple con la característica del Modo. De esta forma se tiene entonces 3 conjuntos de controladores diferentes para Posición y Orientación. El CONTROLADOR toma una suma ponderada de estos 3 grupos de controladores y la aplica al Sistema, es decir computa la entrada u asociada con cada una de las entradas de los subsistemas del cuadricóptero.

Para resolver el algoritmo para el sistema de control Sliding-Mode se propuso el siguiente diagrama de flujo.

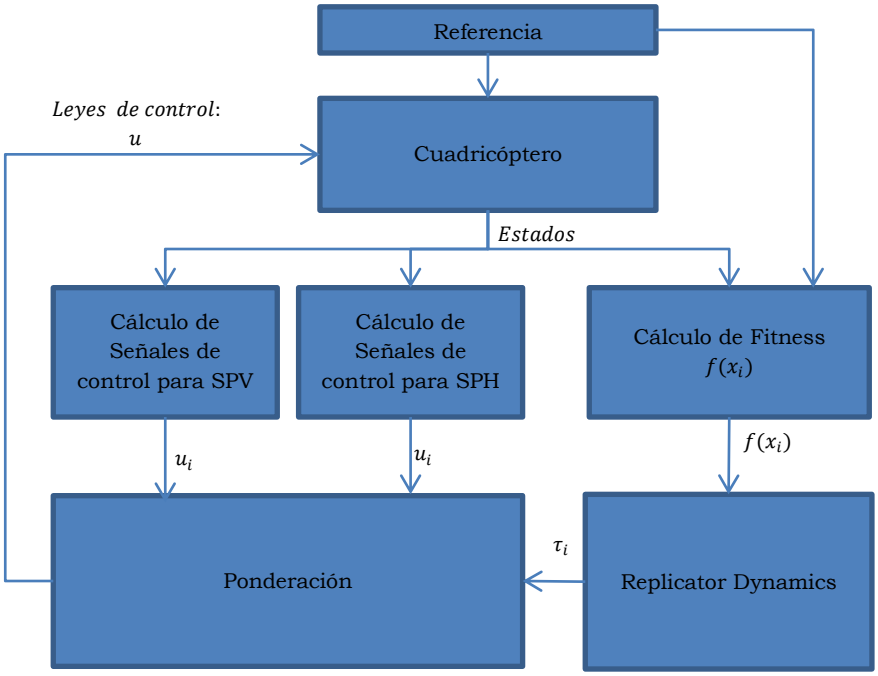


Figura 67. Diagarama de flujo para resolver el algoritmo para el sistema de control Sliding-Mode.

Se construyó 3 funciones de fitness haciendo uso de funciones exponenciales. Dichas funciones de bienestar incrementan su valor cuando se cumplen las condiciones del

modo asociado descritas en la Tabla XXII,y disminuye hasta convertirse en cero cuando en el caso contrario.

A continuación se muestra las funciones fitness asociadas con cada modo:

Tabla 23. Función fitness y parámetros de cada modo

Modo	**Función de Fitness($f(x_i)$)**	**Parámetros Fitness**					
		Sis.	α	β	γ	off	odd
Tracking ($i=1$)	$\left[\left(1-e^{\frac{10er-\alpha-off}{\gamma}}\right)\right]_+\left[\left(1-e^{\frac{-10dr+\beta-odd}{\gamma}}\right)\right]_+$	SP V	5	0	0.1	0.2	0.2
		SP H	4	$1.39x10^{-5}$	0.1	0.04	0.005
Equilibrio ($i=2$)	$\left[\left(1-e^{\frac{10er-\alpha-off}{\gamma}}\right)\right]_+\left[\left(1-e^{\frac{10dr-\beta-odd}{\gamma}}\right)\right]_+$	SP V	5	0	0.1	0.2	0.2
		SP H	4	$1.39x10^{-5}$	0.1	0.04	0.005
Arranque ($i=3$)	$\left[1-e^{\frac{-10er+\sigma-off}{\gamma}}\right]_+$	SP V	5	-	0.1	0.2	-
		SP H	8	-	0.1	0.04	-
$[x]_+ = \boldsymbol{max}(\mathbf{0}, x)$							

A continación se muestran graficos de las funciones fitness vs las variables $dr\ y\ er$:

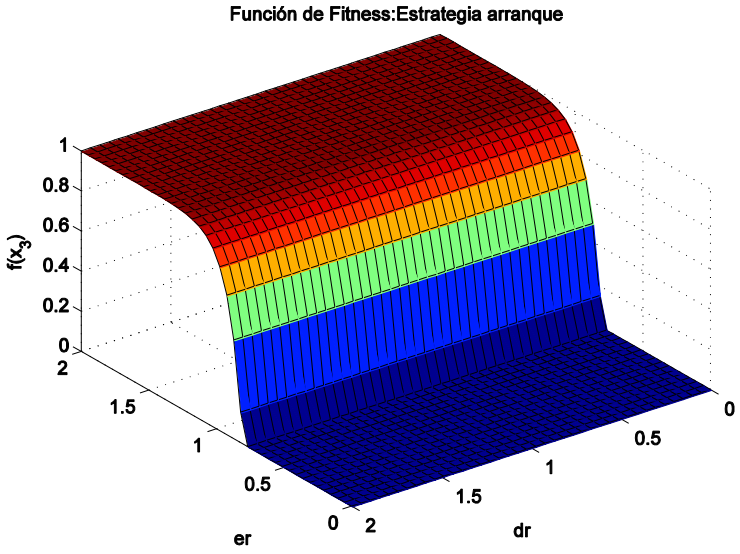


Figura 68. Función Fitness modo arranque.

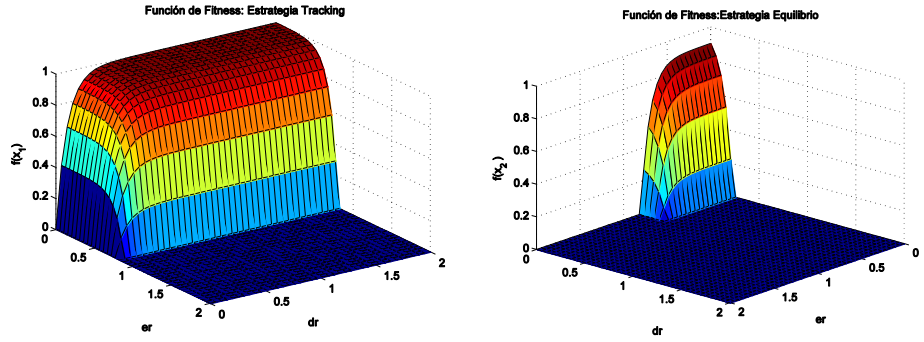


Figura 69. Función Fitness modo arranque traking y equilibrio.

Pruebas del la respuesta del controaldor ante diferntes trayectorias en el espacio fueron realizadas. A continuación se muestra los resultados.

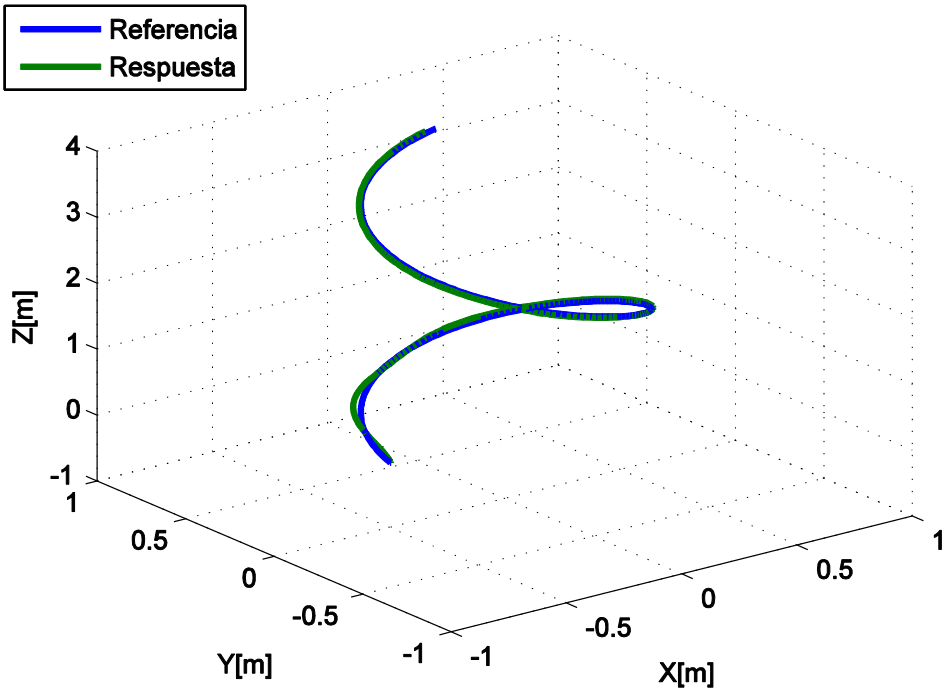


Figura 70. Respuesta del controlador trayectoria en el espacio.

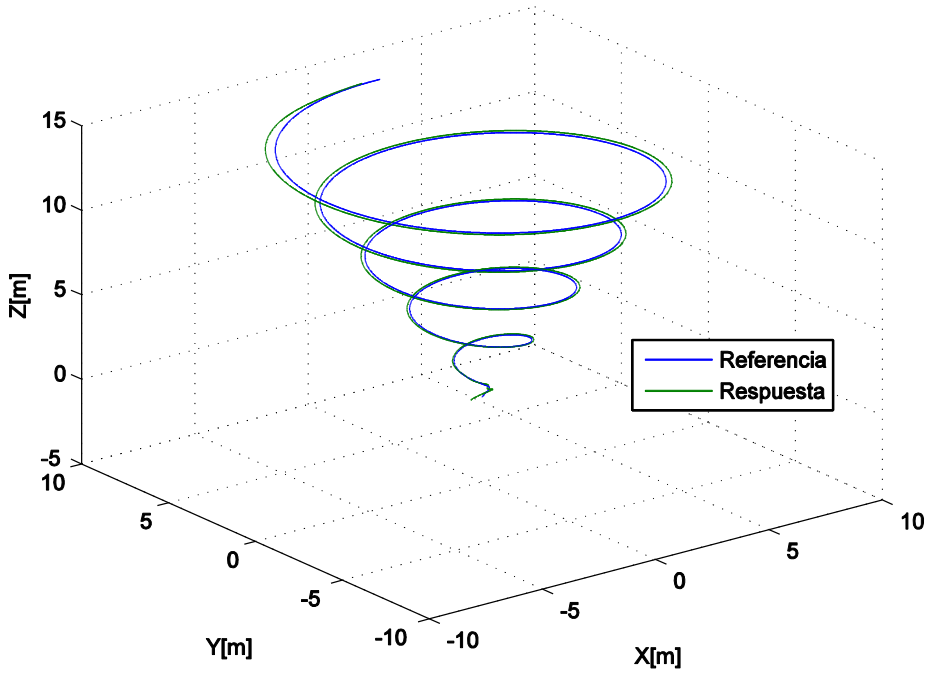


Figura 71. Respuesta del controlador trayectoria en el espacio.

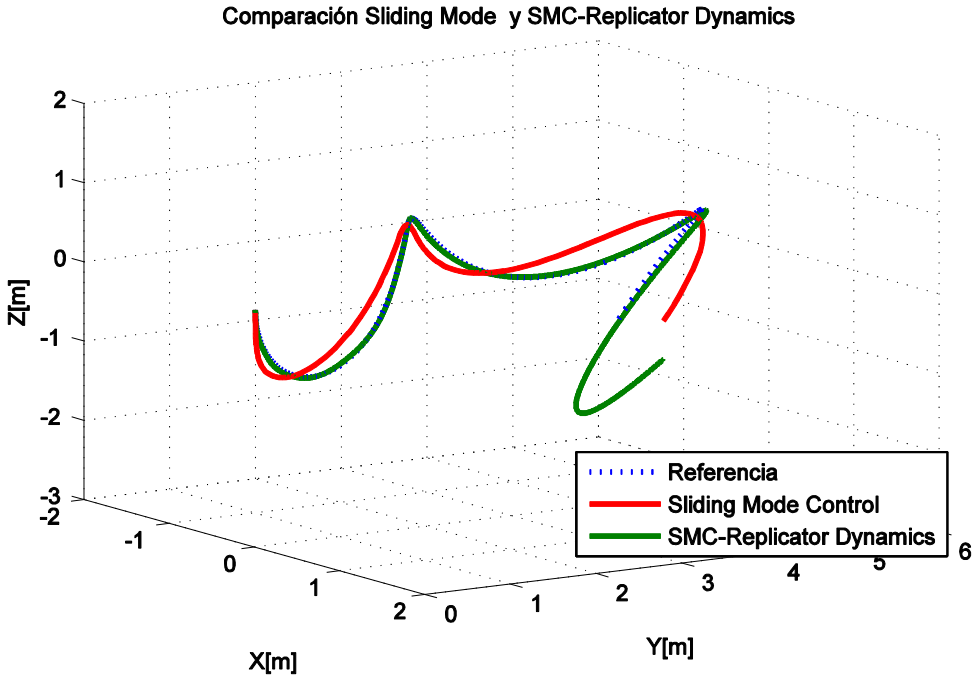


Figura 72.Comparación de Sliding Mode Control regular y SMC-RD

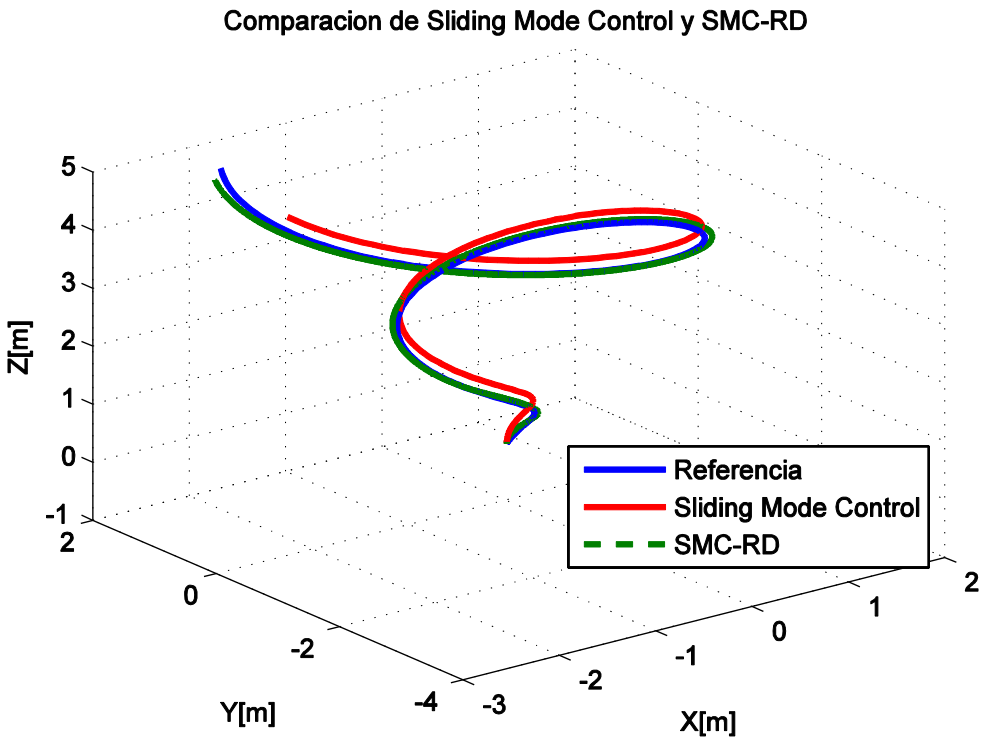


Figura 73.Comparación Sliding Mode Control y SMC-RD

Es posible observar claramente la mejoría generada a partir de la fusión SMC-RD ante SMC. El controlador SMC-RD logra un seguimiento a la par de la referencia mientras que el controlador SMC tiene algunas dificultades para seguir la trayectoria de variaciones continuas.

4. CONCLUSIONES

El peso de los materiales a usar para la construcción de un cuadricóptero fue un factor de gran importancia, pues afecta de forma directa los parámetros físicos del modelo del vehículo y el diseño de los controladores. Las restricciones influyen directamente a la hora de realizar una implementación real de los controladores. La falta de la distribución de masa del modelo puede causar desbalances en las señales de control lo que ocasiona comportamientos no deseados. Además la simetría entre los ejes permite considerar que las inercias sobre los ejes del cuadricóptero son iguales, lo que simplifica el modelo y la estimación de algunos parámetros.

La calibración de sensores tanto de orientación (IMU) como de posición (sistema de visión artificial) llevó a obtener medidas más precisas de los estados dinámicos del vehículo, ofreciendo así un control más confiable. Además proporcionó información vital para la realización de la implementación real de los controladores.

El modelamiento matemático del vehículo, así como la estimación de sus parámetros físicos jugó un papel importante en el diseño de los controladores, pues permitió la correcta simulación y sintonización de los controladores, en especial aquellos que dependen específicamente del modelo como los MPC y Sliding Mode.

El proceso de diseño realizado de forma empírica fue un proceso que llevó una cantidad de tiempo considerable, sin embargo permitió conocer más a fondo las dinámicas del vehículo. Los controladores basados en Sliding Mode son simples de sintonizar pues so lo tienen 3 parámetros de sintonización y no requiere de derivadores ni integradores en su estructura.

La consideración de la linealidad del modelo del vehículo en sus dinámicas de altura llevó a obtener resultados no deseados en la respuesta de controladores como los MPC. Esto fue corregido con la linealización exacta por realimentación de la planta que no provee un modelo lineal aproximado del vehículo, sino que convierte las dinámicas no lineales del cuadricóptero a dinámicas simples y lineales. La aplicación de este método corrigió la respuesta no deseada del MPC y ayudó a mejorar el comportamiento de los demás controladores. Además la separación del modelo total en subsistemas fue una herramienta muy útil en lo que al diseño de los controladores se refiere, pues permite la sintonización y el estudio de las dinámicas por separado, haciendo más simple el proceso de diseño.

La elección del mejor controlador depende del criterio escogido para la aplicación en particular que la realice. Sin embargo de acuerdo con los objetivos planteados fue posible la realización de una ponderación de los parámetros de rendimiento del vehículo, lo cual concluyó en que Sliding Mode control era la mejor opción en cada grupo para realizar las tareas de control. Cabe aclarar que las ventajas del Sliding Mode Control no solo se limitan a su buen comportamiento y robustez, sino que se extienden a la etapa de implementación, pues la programación de las dinámicas de dicho controlador en sistemas embebidos resulta bastante cómoda y simple. Sin embargo fue necesaria una re-calibración de los parámetros de sintonización de sus parámetros, después de la cual se obtuvo los resultados deseados.

Aunque el sistema de control completo basado en Sliding Mode cumple con los objetivos planteados, fue posible realizar mejoras a nivel de simulación del sistema con la inclusión de Replicator Dynamics.

La fusión entre Replicator Dynamics y Sliding Mode Control resultó en una arquitectura de un controlador adaptativo y robusto.

Arduino es una herramienta que facilitó en gran magnitud la implementación del sistema de control basado en Sliding Mode, así como todas sus dependencias (sensores, filtros, comunicación).

Como un resumen general del proyecto se tiene:

Una revisión bibliográfica fue realizada para seleccionar algunos métodos de control que al ser aplicados a un cuadricóptero, puedan ser sometidos a comparación.

Se realizó modelamiento matemático de la planta basado en la revisión bibliográfica. Se realizó la estimación y/o el cálculo de los parámetros físicos de éste, mediante diferentes métodos, las estimaciones fueron validadas experimentalmente.

Un método control de orientación del vehículo fue escogido (Sliding Mode) entre 4 candidatos, el cual presentó un error de estado estacionario nulo, y supero al resto en base a los parámetros de rendimiento: IEAT,ICE, Sobreimpulso, Tiempo de establecimiento y Correlación del espectro.

Se realizó una comparación Cuantitativa de 4 métodos de control de Orientación para un cuadricóptero, necesaria para el control de posición del vehículo, para el cual una comparación de 3 métodos de control fue realizada. Además se propuso un sistema de control basado en Sliding Mode y Replicator Dynamics la cual proporciona características adaptativas a las dinámicas de control del vehículo.

El controlador basado en Sliding Mode para orientación fue implementado en el embebido Arduino y fue puesto a prueba bajo algunas trayectorias. Se diseñó e implementó un Sistema de Estimación de estados mediante la fusión en un Sistema de Visión Artificial y las medidas de un Acelerómetro-Giróscopo.

5. RECOMENDACIONES

Se recomienda usar materiales de bajo peso para la construcción de vehículos aéreos. Un mayor peso del vehículo exigirá mayor potencia en los motores lo que confluye en un mayor gasto de energía, disminuyendo el tiempo de vuelo. Utilizar baterías de mayor capacidad podría ser una solución para incrementar el tiempo de vuelo, sin embargo las baterías de mayor capacidad pesan mucho más, y el incremento de carga disponible podría no compensar el gasto de energía generado por el peso extra añadido. Por lo tanto, se recomienda usar baterías de baja carga y bajo peso.

Por lo general el fabricante proporciona librerías para el uso de sus productos, sin embargo estas pueden contener funciones que no son necesarias para alguna aplicación en particular, ocupando espacio en memoria inútilmente. Para aplicaciones donde el tiempo de respuesta del microcontrolador o embebido no sea critico este hecho podría no ser de importancia, sin embargo si se trabaja con procesadores de capacidades limitadas y se requiere la máxima rapidez de procesamiento es recomendable codificar librerías propias, lo cual además de ser más eficiente puede llevar a realizar mejores estimaciones.

La estimación de parámetros se llevó a cabo mediante una búsqueda en línea y mediante el algoritmo UKF. Sin embargo no es siempre posible realizar una búsqueda en línea directa de los parámetros de una planta, para estos casos es recomendable utilizar algoritmos de optimización basados en foraging y semejantes. Para el algoritmo UKF es muy importante tener información acertada sobre la covarianza del ruido gaussiano que contienen las mediciones y el proceso.

Es posible realizar una sintonización empírica de los controladores, sin embargo la utilización de algoritmos de optimización, tomando como variables a optimizar los parámetros de los controladores y los parámetros de rendimiento de interés como función de costo puede ser también una alternativa para conseguir este objetivo. Conocer la planta y sus limitaciones también es una herramienta útil a la hora de diseñar controladores para entornos reales.

Los métodos usados para la selección cuantitativa, acorde con los objetivos planteados son suficientes para determinar la mejor estrategia de control.

Con respecto a los integradores numéricos programados en Matlab para calcular los parámetros de rendimiento se aconseja que no se trate de atacar el inconveniente del error generado por la integración numérica puesto que es irrelevante debido a que son medidas comparativas y el error numérico generado afecta a las dos medidas por igual.

En cuestión a la implementación real, antes de hacer pruebas del control definitivo sobre el vehículo, se recomienda inicialmente, hacer pruebas por separado para cada subsistema definido, verificando el funcionamiento y si es necesario hacer correcciones o ajustes individuales con lo cual posteriormente se puede aplicar los controladores en conjunto.

REFERENCIAS BIBLIOGRÁFICAS

ABRAMSON, G. “Introducción a la Teoría de Juegos”. México: Centro Atómico Bariloche, Instituto Balseiro, 2006.

B. CASTILLO, Di Gennaro, and JURADO, F. “Trajectory Tracking for a Quadrotor via Fuzzy Regulation”. México: Centro de Investigación y de Estudios Avanzados del I.P.N., 45015 Zapopan.

Disponible en Internet: http://www.kkmulticopter.kr/

Disponible en Internet: http://www.sparkfun.com

GIL, J. DÍAZ, A. “*Ingeniería de Control, Control de Sistemas Continuos”.* EEUU: Universidad De Navarra, Escuela Superior de Ingenieros, Unicopia C.B, 2004.

GOMEZ, G. “*Visión Computacional*”. Mexico: s.n. s.f.

GÓMEZ, J. “Fuzzy Control”. Bogotá: Universidad Tecnológica Nacional – FRBA, s.f.

KHALIL, Michigan. “*Nonlinear System*” State University. 3ra ed. México: Pearson Educación, 2000.

LOPEZ, A.M.; MILLER, J.A.; SMITH, C.L., y P.W. Tuning Controllers with Error-Integral Criteria. EEUU: Murrill, Instrumentation Technology, 1976.

MOROTA, F. “Controlador Fuzzy de un Quadrotor”. Tesis Doctoral. Madrid: Universidad Complutense de Madrid, 2009.

OGATA, K. “*Ingeniería de Control Moderna*”. 3ra ed. Minnesota: Pearson Educación, 2000.

PINES, D. and BOHORQUEZ, F. “Challenges facing future micro air vehicle development,” EEUU: AIAA Journal of Aircraft, Vol. 43, No. 2. 2006.

RANGAJEEVA, S., & WHIDBORNE, J. Linear parameter varying control of a quadrotor. Mexico: Industrial and Information Systems, 2011.

RODRÍGUEZ, D. “*Perspectiva General del Control Predictivo*” Tesis Doctoral. Bogotá: s.n. s.f.

SREENATH, K; LEE, T; KUMAR, V. & SO, R. (n.d.). Geometric Control and Differential Flatness of a Quadrotor UAV with a Cable-Suspended Load. Mexico: s.n. 1243000(3).

TEREJANU, G. Unscented Kalman Filter tutorial. Department of Computer Science and Engineering. Tomado de http://www.cse.sc.edu/~terejanu/files/tutorialUKF.pdf. 2011.

WAN, E. and VAN, R. The Unscented Kalman Filter. EEUU: Wiley Publishing, 2

ANEXOS

ANEXO 1. CÓDIGO ARDUINO

```
#include <Servo.h>
#include <FreeSixIMU.h>
#include <FIMU_ADXL345.h>
#include <FIMU_ITG3200.h>
#include <Wire.h>
int dato,dato2;
float yaw;
float const alpha2=0;
float const alpha=0.95;
float const alpha3=0.95;
float const alphaz=0.7;
float g,gc,az,azz,gcz,azf,axf,ayf,aff,aza,uz,ki,uzr,uzz;
float angles[6]; // yaw pitch roll
float ypr[3],yprp[3];
Servo myservo,myservo1,myservo2,myservo3;
FreeSixIMU sixDOF = FreeSixIMU();
float L,M,N,G;
float const dx=1.8248e-04;
float const bb=0.00010673;
float const aaa=0.00010673;
float const bbb=0.019604;
float const ccc=-1.305;
float const a1=7/4;//6.7
float const b1=-0.9/3.1;//-3.8
float const k1=30/3.9;//30//12
float const a=1;//6.7
float const b=-0.9/2.6;//-3.8
float const k=30/3.2;//30//12
float const a2=2.7;//6.7
float const b2=-0.01;//-3.8
float const k2=-60000*1.77e-4;//30//12
float const a3=11;//6.7
float const b3=-2/4;//-3.8
float const k3=10;//30//12
float w1,w2,w3,w4;
int off=84;
float dt1=0,dt2=0;
float dt,dtas,dta;
int band=0;
void setup() {
Serial3.begin(115200);
Serial.begin(115200);
//pinMode(8, OUTPUT);
//pinMode(9, OUTPUT);
//pinMode(10, OUTPUT);
//pinMode(11, OUTPUT);
 myservo.attach(8) ;
myservo1.attach(9);
myservo2.attach(10);
myservo3.attach(11);
arm();
```

```
Wire.begin();
  delay(5);
  sixDOF.init(); //begin the IMU
 sixDOF.acc.setFullResBit(true);
 sixDOF.acc.setRangeSetting(1);
  delay(5);
}
void loop() {
  // put your main code here, to run repeatedly:
   dt=(micros()-dta)/1000000;
   dta=micros();
   sixDOF.gyro.readGyro(yprp);
    sixDOF.getYawPitchRoll(ypr);
   if (Serial3.available()) {
    yaw=Serial3.parseFloat();
   //    analogWrite(8, dato);
//    analogWrite(9, dato);
//    analogWrite(10, dato);
//    analogWrite(11, dato);
// if (Serial3.available())
//    {
//     dato2=abs(Serial3.parseInt());
//    }
    }
    else
    {yaw=ypr[0]*PI/180;
    }
          //dato=map(dato,0,32767,0,30);
//         dato2=map(dato2,-32767,32767,-30,30);
//         if(float(millis())*1000.0>10)
//         {
//         dato=0.6*180/PI;
//       }
//         else
//         {
//         dato=0;
//       }
//      dato=0.6*180/PI;
    dato=0;
    sixDOF.getValues(angles);
    gc=9.8;
    az=-angles[2]*gc*1.0265/236.0753;
    gcz=10.1*cos(ypr[1]*PI/180)*cos(ypr[2]*PI/180);
    azz = az + gcz;
    aff= alphaz *aff + (1 - alphaz) * azz;
    if(millis()>6000 || band==1)
{band=1;
//  if(abs(aff*1000)<500)
//  {
//aff=0;
//  }
  if (aza*aff>0)
  {  uz=uz+dt*aza;
  uz=uz +dt*(aff-aza)/2;
```

```
  }
 else if(aza*aff<0)
{ki=-aff/aza;
 dt1=dt/(ki+1);
 dt2=dt-dt1;
uz=uz+aza*dt1/2+aff*dt2/2;
}
uzr= alpha3 *uzr + (1 - alpha3) * uz;
uzz = uz - uzr;
aza=aff;
//   sliding mode
ypr[1]=ypr[1]*0.9808-0.9207+0.6;
ypr[2]=ypr[2]*0.9617-3.0729;
//yprp[1]=yprp[1]*0.9808-0.9207;
azf= alpha2 *azf + (1 - alpha2) * yprp[0];
axf= alpha2 *axf + (1 - alpha2) * yprp[1];
ayf= alpha2 *ayf + (1 - alpha2) * yprp[2];
//ayf= alpha2 *ayf + (1 - alpha2) * yprp[2];
 M=k1*(sat(b1*(-axf*PI/180)+a1*(dato-ypr[1])*PI/180,1));//+0.00467;
 L=k*(sat(b*azf*PI/180+a*(dato-ypr[2])*PI/180,1));//-0.02365;
  g= alpha *g + (1 - alpha) * ypr[0];
  ypr[0]=ypr[0]-g;
 //M=k*(sat(b*yprp[2]+a*(dato-ypr[2])))
 N=k2*(sat(b2*ayf*PI/180+a2*(dato*PI/180-yaw),1))-30000*1.7e-4;
 //N=-30000*1.77e-4;
 G=29+k3*(sat(b3*-aff+a3*(dato+uzz),1))-(ccc-bbb*bbb/(4*aaa));
 //w1=0;
 // w2=0;d
 //  w3=0;
//    w1=1180;
 w1=int(sqrt(pos((G/bb-N/dx-2*L/dx)/4)))+1060-bbb/(8*aaa)+32/2;//+25.5
 w2=int(sqrt(pos((G/bb+N/dx-2*M/dx)/4)))+1060-bbb/(8*aaa)-16/2;
 w3=int(sqrt(pos((G/bb-N/dx+2*L/dx)/4)))+1060-bbb/(8*aaa)-32/2;
 w4=int(sqrt(pos((G/bb+N/dx+2*M/dx)/4)))+1060-bbb/(8*aaa)+16/2;//16
  myservo.writeMicroseconds(w1);
  myservo1.writeMicroseconds(w2);
  myservo2.writeMicroseconds(w3);
  myservo3.writeMicroseconds(w4);
//   Serial.print(azf);
//   Serial.print(",");
//   Serial.print(axf);
//   Serial.print(",");
   Serial3.print(4*uzz);
   Serial3.print(",");
   Serial3.println(aff);
//   Serial.print(",");
//   Serial.print(ypr[1]);
//   Serial.print(",");
//   Serial.println(ypr[2]);
 dtas=micros();
delay(1);
}
}
void arm(){
```

```
  delay(200);
int t=1500;
//    analogWrite(8, t);
//    analogWrite(9, t);
//    analogWrite(10, t);
//    analogWrite(11, t);
  myservo.writeMicroseconds(t);
  myservo1.writeMicroseconds(t);
  myservo2.writeMicroseconds(t);
  myservo3.writeMicroseconds(t);
t=1000;
delay(800);
    myservo.writeMicroseconds(t);
    myservo1.writeMicroseconds(t);
    myservo2.writeMicroseconds(t);
    myservo3.writeMicroseconds(t);
   delay(1200);
}
float sat(float u,float a)
{float s;
  if (u>1)  {
    s=1;  }
   else if(u<-1) {
  s=-1;
  }
  else
  {
  s=u*a;
  }
return s;
}
float pos(float u)
{float s;
  if(u<0)
 {s=0;
 }
 else
 {s=u;
 }
return s;
}
```

ANEXO 2. CÓDIGO VISIÓN ARTIFICIAL.

```
clear all;
vid = videoinput('winvideo', 1, 'YUY2_640x480');
src = getselectedsource(vid);
 s2 = serial('COM9','BaudRate',115200,'Terminator','CR/LF','InputBufferSize',84);
%%
close all
load('Omni_Calib_Results_old3.mat')
vid.FramesPerTrigger = 1;
triggerconfig(vid, 'manual');
vid.ReturnedColorspace = 'rgb';
% vid.timerPeriod = 0.1;
% vid.FrameGrabInterval = 0.5;
set(vid,'TriggerRepeat',Inf);
start(vid);
fopen(s2);
data=zeros(480,640,3);
Imc=zeros(480,640,3);
fc = 1.9;
ocam_model=calib_data.ocam_model;
Nwidth =640; %size of the final image
Nheight = 480;
Nxc = Nheight/2;
Nyc = Nwidth/2;
Nz  = -Nwidth/fc;
width = ocam_model.width;
height = ocam_model.height;
%The ocam_model does not contain the inverse polynomial pol
ocam_model.pol = findinvpoly(ocam_model.ss,sqrt((width/2)^2+(height/2)^2));
%distance of the plane from the camera, change this parameter to zoom-in or out
display = 0;
Nimg = zeros(Nheight, Nwidth, 3);
[i,j] = meshgrid(1:Nheight,1:Nwidth);
Nx = i-Nxc;
Ny = j-Nyc;
Nz = ones(size(Nx))*Nz;
M = [Nx(:)';Ny(:)';Nz(:)'];
mm = world2cam_fast( M , ocam_model );
scrsz = get(0,'ScreenSize');
 figure('Name','Video Segmentation','NumberTitle','off','Position',[1 0 scrsz(3)/2 scrsz(4)]);

subplot(4,2,[5 6]);legend('X-Pos','Y_Pos','Z_Pos');ylim([-1.5,1.5]);
subplot(4,2,[7 8]);legend('X-Pos','Y_Pos','Z_Pos');
 mm = world2cam_fast( M , ocam_model );
% Kal,man position and Speed
n=500;
zzk=zeros(1,n);
vvk=zeros(1,n);
yyk=zeros(1,n);
vvyk=zeros(1,n);
yawk=zeros(1,n);
xxk=zeros(1,n);
```

```
vvxk=zeros(1,n);
vvzk=zeros(1,n);
%---------------------------
%Quad Camera Position
xx=zeros(1,n);
yy=zeros(1,n);
zz=zeros(1,n);
zz(1)=1;
xx(1)=0;
yy(1)=0;
%---------------------------
vv=0.3;
qk=0.2;rk=0.025;hk=[1 0];say=[0;0];saz=[0.8;0];sax=[0;0];dt=0.225;
pz=qk^2*[dt^4/4 dt^3/4;dt^4/4 dt^2];py=qk^2*[dt^4/4 dt^3/4;dt^3/4 dt^2];
px=qk^2*[dt^4/4 dt^3/4;dt^3/4 dt^2];
img=zeros(480,640,3);
key1=mm';
% %% Controller
%    running = 1;
%
%     % Initialize two joysticks
%     JoyMEX('init',0);
%     JoyMEX('init',1);
%
% %%
profile on
pause(1)
dt2=0.1225;
i=1;
off=[0;0;0];
tic;
while(i<=400)
i=i+1;
      trigger(vid);
     %data = getdata(vid,1);

     img= getsnapshot(vid);
      startime=tic;
%   Imc=imread(url);
   %Imc = data(:,:,:,1);
    %r=Imc(:,:,1);
 %image=undistort(calib_data.ocam_model,r,1.9,0);
 %[rr,gg,bb] = get_color_from_imagepoints( img, mm' );
height = size(img,1);
width  = size(img,2);
key1 = round(key1);
% Correct points which are outside image borders
indH = find( key1(:,1)<1 | key1(:,1)>height | isnan(key1(:,1)) );
key1(indH,1) = 1;
key1(indH,2) = 1;
indW = find( key1(:,2)<1 | key1(:,2)>width  | isnan(key1(:,2)) );
key1(indW,1) = 1;
key1(indW,2) = 1;
```

```
im1(1,1,1) = 0;
im1(1,1,2) = 0;
im1(1,1,3) = 0;
RI = img(:,:,1);
GI = img(:,:,2);
BI = img(:,:,3);
rr = RI(sub2ind( [height,width], key1(:,1), key1(:,2) ));
gg = GI(sub2ind( [height,width], key1(:,1), key1(:,2) ));
bb = BI(sub2ind( [height,width], key1(:,1), key1(:,2) ));
Nimg(:,:,1) = reshape(rr,Nwidth,Nheight)';
Nimg(:,:,2) = reshape(gg,Nwidth,Nheight)';
Nimg(:,:,3) = reshape(bb,Nwidth,Nheight)';
Nimg=uint8(Nimg);
 r=Nimg(:,:,1);
 g=Nimg(:,:,2);
 b=Nimg(:,:,3);
%justGreen=(g-r/1.5-b/1.5
   justGreen=(g-r/1.5-b/1.5);
   r=255-Nimg(:,:,1);
   g=255-Nimg(:,:,2);
  b=255-Nimg(:,:,3);
justRed=(g-r/1.5-b/1.5);
  %____________  Position capture
th =10;
bw=justGreen>th;
n=3;
m =ones(n,n);
%bw=conv2(single(bw),m);
 bw = imfill(bw, 'holes');
 bw=imclearborder(bw, 4);
 seD = strel('diamond',1);
 bw = imerode(bw,seD);
 bw = imerode(bw,seD);
[uug,vvg]=find(bw);
xmg=mean(uug);
ymg=mean(vvg);
bw=justRed>th;
n=3;
%bw=conv2(single(bw),m);
 bw = imfill(bw, 'holes');
  bw=imclearborder(bw, 4);
 seD = strel('diamond',1);
 bw = imerode(bw,seD);
 bw = imerode(bw,seD);
%subplot(4,2,[1 2 3 4]);%imshow(255*bw);
[uur,vvr]=find(bw);
xmr=mean(uur);
ymr=mean(vvr);
sx=std(uur);
sy=std(vvr);
yaw=sign(ymg-ymr)*atan(sx/sy);
 xm=xmr;
 ym=ymr;
%
```

```
% if(sy>4*sx)
%     xm=xmr;
%     ym=ymr;
%     sy=std(vvr);
%     sx=std(uur);
%
% %sx=sx*(1-0.3*exp(-(150./(200-xm)).^2));
% sy=sy;
%
% else
%     xm=xmg;
%     ym=ymg;
%     sx=std(uug);
%     sy=std(vvg);
%
% sx=sx;
% %sy=sy*(1-0.3*exp(-(150./(200-ym)).^2));
% end
kll=0.87;
klm=0.9;
kll2=0.95;
off=[0;0;0];
zz(1,i)=62.632/sqrt(sx^2+(sy/0.9)^2)+0.0963;
xx(1,i)=(220-xm)*zz(1,i)/(klm*406.15);
yy(1,i)=(ym-320)*zz(1,i)/(kll*406.15);
%:_:::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::
%
 if zz(1,i)==Inf || isnan(zz(1,i))
    zz(1,i)=zz(1,i-1);
 end
if (abs(zz(i)-zz(i-1))>1)
        zz(i)=zz(i-1);
    else
        zz(i)=zz(i);
end

 if yy(1,i)==Inf || isnan(yy(1,i))
    yy(1,i)=yy(1,i-1);
 end
if (abs(yy(i)-yy(i-1))>0.8)
        yy(i)=yy(i-1);
    else
        yy(i)=yy(i);
end
 if xx(1,i)==Inf || isnan(xx(1,i))
    xx(1,i)=xx(1,i-1);
 end
if (abs(xx(i)-xx(i-1))>0.8)
        xx(i)=xx(i-1);
    else
        xx(i)=xx(i);
end
% Kalman Filtering for Estimation
```

```
az=fscanf(s2,'%f,%f,%f');
while(numel(az)<3)
az=fscanf(s2,'%f,%f,%f');%recepcion de datos
end
dt=0.2;% Tiempo de Muestreo
axx=az(1); % Aceleración en los 3 ejes
ayy=-az(2);
azz=az(3);
% vv=azz*dt+vv;
% vvm(1,i)=vv;
Ex=qk^2*[dt^4 0;0 dt^2];
A=[1 dt;0 1]; B=[(dt^2)/2 ;dt];% (dt^2)/2 Modelo de Aceleración
% Estimacion de VZ
sbz=A*saz+B*azz;
pz=A*pz*A'+Ex;
k=pz*hk'*inv(hk*pz*hk'+rk);
saz=sbz+k*(zz(1,i)-hk*sbz);
pz=(eye(2)-k*hk)*pz;
zzk(1,i)=saz(1)-off(3);
% Estimacion de VY
sby=A*say+B*ayy;
py=A*py*A'+Ex;
k=py*hk'*inv(hk*py*hk'+rk);
say=sby+k*(yy(1,i)-hk*sby);
py=(eye(2)-k*hk)*py;
yyk(1,i)=say(1)-off(2);
% Estimacion de VX
sbx=A*sax+B*axx;
px=A*px*A'+Ex;
k=px*hk'*inv(hk*px*hk'+rk);
sax=sbx+k*(xx(1,i)-hk*sbx);
px=(eye(2)-k*hk)*px;
xxk(1,i)=sax(1)-off(1);
% fprintf(s2,'B');
% fwrite(s2,vvk(1,i)*100,'int8');
% fwrite(s2,zzk(1,i)*100,'int8');
%_____________________________________________________________________
if(i==2)
off=[xxk(1,i);yyk(1,i);0];
end
%:_:::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::
%
 if zzk(1,i)==Inf || isnan(zzk(1,i))
    zzk(1,i)=zzk(1,i-1);
 end
if (abs(zzk(i)-zzk(i-1))>1)
        zzk(i)=zzk(i-1);
    else
        zzk(i)=zzk(i);
end

 if yyk(1,i)==Inf || isnan(yyk(1,i))
```

```
    yyk(1,i)=yyk(1,i-1);
 end
if (abs(yyk(i)-yyk(i-1))>0.8)
        yyk(i)=yyk(i-1);
    else
        yyk(i)=yyk(i);
end
 if xxk(1,i)==Inf || isnan(xxk(1,i))
    xxk(1,i)=xxk(1,i-1);
 end
if (abs(xxk(i)-xxk(i-1))>0.8)
        xxk(i)=xxk(i-1);
    else
        xxk(i)=xxk(i);
end
vvxk(1,i)=(xxk(1,i)-xxk(1,i-1))/dt2;
yawk(1,i)=yaw;
vvzk(1,i)=(zz(1,i)-zz(1,i-1))/dt2;
vvyk(1,i)=(yy(1,i)-yy(1,i-1))/dt2;
subplot(5,2,[1 2 3 4 5 6]),plot(xx,'black');hold on;grid on;
plot(yy,'green');hold on;
plot(zz); hold off;%plot(zzk,'red');
ylim([-1.5,1.5])
subplot(5,2,[7 8]),plot(vvxk,'black');hold on;
plot(vvyk,'green');hold on;
plot(vvzk); hold off;%plot(zzk,'red');
subplot(5,2,[9 10]),plot(yawk,'black');hold on;
hold off;
pause(0.00000000000001)
  dt2 = toc(startime);
end
stop(vid)
 fclose(s2);
 disp('Finalizado')
profile viewer
```

ANEXO 3. CÓDIGO DE ADQUISICIÓN DE DATOS

```
function [datos, entrada,dt]=Serialcontrol
clear all;
    running = 1;
    % Initialize joystick
    joymex2('open',0);
    delete(instrfind({'Port'},{'COM9'}));
    s2 = serial('COM9','BaudRate',115200,'Terminator','CR/LF','InputBufferSize',100);
    % Create figure and attach close function
    figure('CloseRequestFcn',@onClose);
    % Create plots
    subplot(2,2,1)
    p1=plot([0],[0],'x'); hold on;
    p2=plot([0],[0],'+r');
    axslims = [double(intmin('int16'))-10 double(intmax('int16'))+10];
    set(gca,'xlim',axslims,'ylim',axslims); axis square
    fopen(s2);
    i=0;
    datos=zeros(10001,8);
    dt=zeros(10001,1);
    entrada=zeros(10001,2);
    while(i<5000 )
        i=i+1;
        % Query postion and button state of joystick 0
        a = joymex2('query',0);
        % Update the plots
        % Notice the usage of minus signs to invert certain axis values.
        % Depending on the device you may want to change these
        % (The plots are originally configured for a Xbox 360 Controller)
       % set(p1,'Xdata',a.axes(1),'Ydata',-a.axes(2));%izquierda
        %set(p2,'Xdata',a.axes(5),'Ydata',-a.axes(4));%derecha
%         iz=int2str(-a.axes(2));
%         fprintf(s2,iz);
%         entrada(i,1)=-a.axes(2);
%         iz=int2str(-a.axes(4));
%         fprintf(s2,iz);
%         entrada(i,2)=-a.axes(4);
  b=fscanf(s2,'%f,%f,%f,%f,%f,%f,%f,%f');
%   while(numel(b)<8)
%  b=fscanf(s2,'%f,%f,%f,%f,%f,%f,%f,%f');
%  end
       datos(i,:)=b;
        % Force update of plot
        %drawnow
        pause(0.0001)
    end
    running=0;
    % Clear MEX-file to release joystick
    clear joymex2
    fclose(s2);
    %delete(src);
    function onClose(src,evt)
```

```
        % When user tries to close the figure, end the while loop and
        % dispose of the figure
        running = 0;
        delete(src);
    end
end
```

ANEXO 4. FUNCIÓN UKF

```
function [x,P]=ukf(fstate,x,u,P,hmeas,z,Q,R)
L=numel(x);                          %numer of states
m=numel(z);                          %numer of measurements
alpha=1e-5;                          %default, tunable
ki=5;                                %default, tunable
beta=5;                              %default, tunable
lambda=alpha^2*(L+ki)-L;             %scaling factor
c=L+lambda;                          %scaling factor
Wm=[lambda/c 0.5/c+zeros(1,2*L)];    %weights for means
Wc=Wm;
Wc(1)=Wc(1)+(1-alpha^2+beta);        %weights for covariance
c=sqrt(c);
X=sigmas(x,P,c);                     %sigma points around x
[x1,X1,P1,X2]=ut(fstate,X,u,Wm,Wc,L,Q);   %unscented transformation of process
% X1=sigmas(x1,P1,c);                %sigma points around x1
% X2=X1-x1(:,ones(1,size(X1,2)));    %deviation of X1
[z1,Z1,P2,Z2]=ut(hmeas,X1,u,Wm,Wc,m,R);   %unscented transformation of measurments
P12=X2*diag(Wc)*Z2';                 %transformed cross-covariance
K=P12*inv(P2);
x=x1+K*(z-z1);                       %state update
P=P1-K*P12';                         %covariance update
function [y,Y,P,Y1]=ut(f,X,u,Wm,Wc,n,R)
%Unscented Transformation
%Input:
%       f: nonlinear map
%       X: sigma points
%      Wm: weights for mean
%      Wc: weights for covraiance
%       n: numer of outputs of f
%       R: additive covariance
%Output:
%       y: transformed mean
%       Y: transformed smapling points
%       P: transformed covariance
%      Y1: transformed deviations
L=size(X,2);
y=zeros(n,1);
Y=zeros(n,L);
for k=1:L
   Y(:,k)=f(X(:,k),u);
   y=y+Wm(k)*Y(:,k);
end
Y1=Y-y(:,ones(1,L));
P=Y1*diag(Wc)*Y1'+R;
function X=sigmas(x,P,c)
%Sigma points around reference point
%Inputs:
%       x: reference point
%       P: covariance
%       c: coefficient
%Output:
```

```
%       X: Sigma points
A = c*chol(P)';
Y = x(:,ones(1,numel(x)));
X = [x Y+A Y-A];
```

ANEXO 5. CÓDIGO ESTIMACIÓN UKF

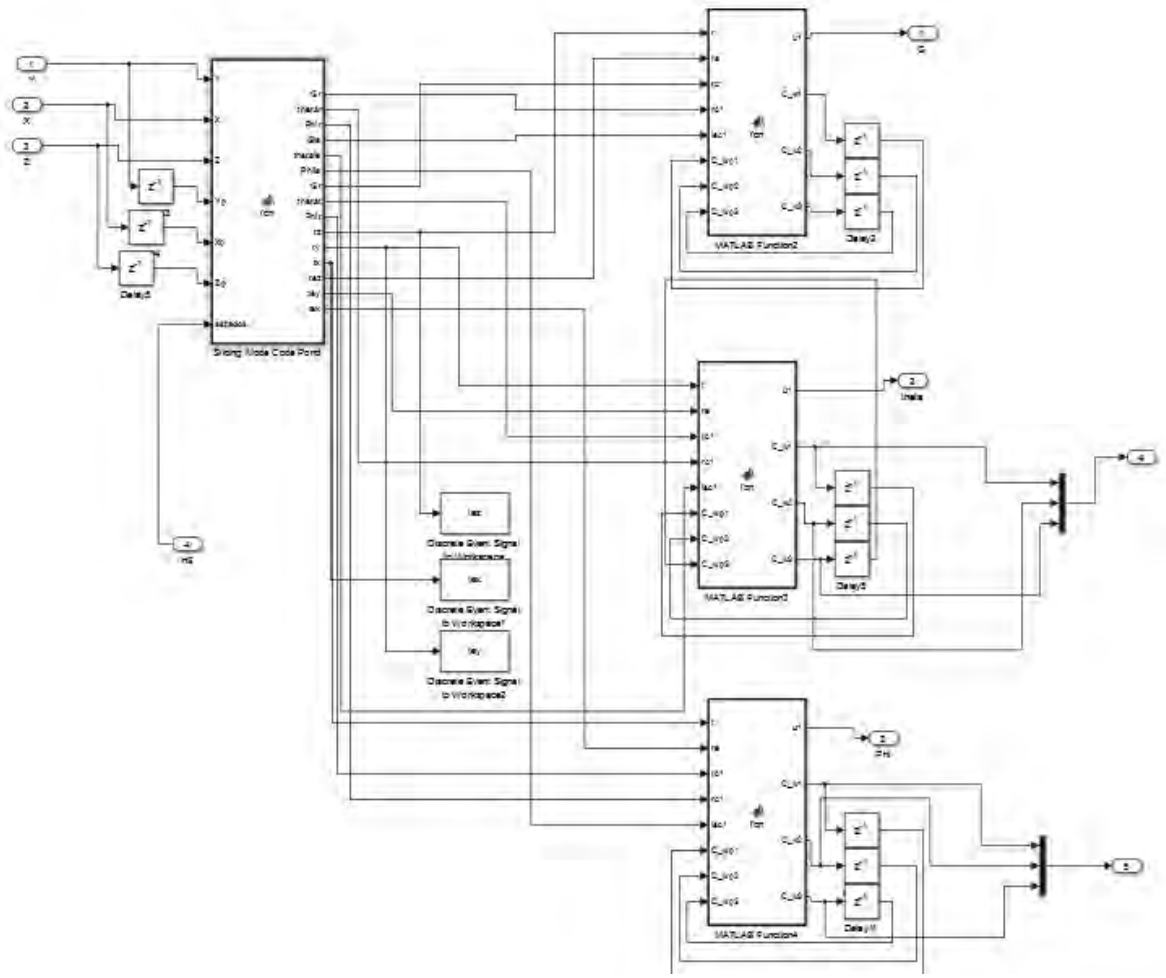
```
load('arratredefinitivo.mat')
 off=0;
 close all;
bbb=0.019604;
aaa=0.00010673;
  Ll=(datos(200:1000,3)-1060+bbb/(8*aaa)-16).^2-(datos(200:1000,1)-1060+bbb/(8*aaa)).^2;
 Pl=-(datos(200:1000,2)'*pi/180);
 PHI=datos(200:1000,5)*pi/180;
 dt=0.018;
 Time=0:dt:dt*(800);

n=3;      %number of state
q=1;    %std of process
r=1;    %std of measurement
%Q=q^2*eye(n); % covariance of process
Q=[0.1 0 0;0 0.1 0; 0 0 0.000001];
R=[(0.6672) 0 0; 0 (0.0619) 0;0 0 0.0000001];        % covariance of measurement
f=@(x,u)[(u*x(3))*(dt)+x(1);x(1)*dt+x(2);x(3)];  % nonlinear state equations
h=@(x,u)[x(1) x(2) x(3)];                          % measurement equation
s=[0;0;0];                              % initial state
x=s; %initial state
P = eye(n);                              % initial state covraiance
N=numel(Time);                                    % total dynamic steps
xV = zeros(n,N);          %estmate
sV = zeros(n,N);          %actual
zV = zeros(3,N);
for k=2:N-9;
 % z = h(s) + r*randn;                    % measurments
  z=[Pl(k+9); PHI(k+9);xV(3,k-1)];
  sV(:,k)= s;                           % save actual state
  zV(:,k)  = z;                           % save measurment
  [x, P] = ukf(f,x,Ll(k),P,h,z,Q,R);            % ekf
  xV(:,k) = x;                          % save estimate
            % update process s
  s =[xV(1,k);xV(2,k);xV(3,k)];
end
for k=1:3                               % plot results
  subplot(3,1,k)
  plot(Time, sV(k,:), '-', Time, xV(k,:), '--')
end
figure
plot(Time,Ll)
```

ANEXO 6. CÓDIGO BÚSQUEDA EN LÍNEA

```
close all;
clear all;
 load('arratredefinitivo.mat')
 off=0;
 close all;
 Ll=sqrt(datos(200:1000,1))-sqrt(datos(200:1000,3));
%  Ll=smooth(((sqrt(gamma(6500:9000,1))-sqrt(gamma(6500:9000,2)))+2-0.13)/10);
w=(-datos(200:1000,2)'*pi/180);
 th=datos(200:1000,5)*pi/180;
w=smooth(w)';
wi=zeros(1,numel(w));
piy=zeros(1,numel(w)+1);
piy2=zeros(1,numel(w)+1);
li=zeros(1,numel(w)+1);
li2=zeros(1,numel(w)+1);
thi=zeros(1,numel(w)+1);
err=zeros(1,100);
err2=zeros(1,1000);
err3=zeros(1,1000);
dt=zeros(1,100);
k=zeros(1,numel(w));
for j=1:1:1000
    dt(j)=j/1000;
for i=2:1:numel(w)
wi(i)=(th(i)-th(i-1))/dt(j);
end
%wi=smooth(wi)';
err(j)=sqrt((wi-w)*(wi-w)');
end
 plot(dt,err); figure
 [v k]=min(err);
 dtf=dt(k);
 for i=2:numel(w)
wi(i)=(th(i)-th(i-1))/dtf;
thi(i)=thi(i-1)+th(i)*dtf;
 end
 wi=smooth(wi)';
 for j=1:1:50000
    k(j)=j/1000;
for i=3:3000
li(i)=k(j)*(wi(i-1)-wi(i-2))/dtf;
piy(i)=piy(i-1)+Ll(i)*dtf/k(j);
end
li=smooth(li)';
err2(j)=sum(abs((li-Ll')));
err3(j)=sum(abs((piy-w)));
 end
 plot(k,err2); figure
 [v oo]=min(err2);
 kf=k(oo);
 for i=2:numel(w)
```

```
li(i)=kf*(w(i)-w(i-1))/dtf;
li2(i)=1.2*(w(i)-w(i-1))/dtf;
 end
 li=smooth(li);
 for i=2:numel(w)
piy(i)=piy(i-1)+(Ll(i))*dtf/(1);
piy2(i)=piy2(i-1)+(Ll(i))*dtf/(kf);
 end
 %li=smooth(li,15);
 plot(wi);hold on;plot(w,'r');plot(piy,'g');plot(piy2,'c');figure;plot(th);hold
on;plot(thi,'r');figure;plot(Ll);hold on;plot(li,'r');plot(li2,'g')
```

ANEXO 7. DIAGRAMA DE SLIDING MODE CONTROL Y VARIABLES DE REPLICATOR.



ANEXO 8. CÓDIGO CALCULO SLIDING MODE.

```
function [Gr,thetar,Phir,Gie,thetaie,Phiie,Gt,thetat,Phit,tz,ty,tx,tez,tey,tex] = fcn(Y, X,
Z,Yp,Xp,Zp,estados)
estado=zeros(1,6);
estado=estados;
%controlador 1
ez=(Z-estado(6));
ey=(Y-estado(5));
ex=(X-estado(4));
tez=abs(ez)*10;
tey=abs(ey)*10;
tex=abs(ex)*10;
tx=10*abs(X-Xp);
ty=10*abs(Y-Yp);
tz=10*abs(Z-Zp);
vz=estado(3);
vy=estado(2);
vx=estado(1);
% Robust
a1=2;%5
a2=-11;%-1%-11
ksl=-9;%-9
u1=a2*ez+a1*vz;
if(abs(u1)>1)
   Gr=-ksl*sign(u1)+9.8;
else
   Gr=-ksl*u1+9.8;
end
a1=0.62;%5
a2=-1;%-1%-11
ksl=-0.6;%-9
u1=a2*ey+a1*vy;
if(abs(u1)>1)
   Phir=ksl*sign(u1);
else
   Phir=ksl*u1;
end
a1=0.62;%5
a2=-1;%-1%-11
ksl=-0.6;%-9
u1=a2*ex+a1*vx;
if(abs(u1)>1)
   thetar=-ksl*sign(u1);
else
   thetar=-ksl*u1;
end
%inital step
a1=2;%5
a2=-11;%-1%-11
ksl=-9;%-9
u1=a2*ez+a1*vz;
if(abs(u1)>1)
```

```
    Gie=-ksl*sign(u1)+9.8;
else
    Gie=-ksl*u1+9.8;
end
a1=3;%5
a2=-3;%-1%-11
ksl=-0.6;%-9
u1=a2*ey+a1*vy;
if(abs(u1)>1)
    Phiie=ksl*sign(u1);
else
    Phiie=ksl*u1;
end
a1=3;%5
a2=-3;%-1%-11
ksl=-0.6;%-9
u1=a2*ex+a1*vx;
if(abs(u1)>1)
    thetaie=-ksl*sign(u1);
else
    thetaie=-ksl*u1;
end
% tracking
a1=1;%5
a2=-15;%-1%-11
ksl=-9;%-9
u1=a2*ez+a1*vz;
if(abs(u1)>1)
    Gt=-ksl*sign(u1)+9.8;
else
    Gt=-ksl*u1+9.8;
end
a1=0.3;%5
a2=-5;%-1%-11
ksl=-0.6;%-9
u1=a2*ey+a1*vy;
if(abs(u1)>1)
    Phit=ksl*sign(u1);
else
    Phit=ksl*u1;
end
a1=0.48;%5
a2=-5;%-1%-11
ksl=-0.6;%-9
u1=a2*ex+a1*vx;
if(abs(u1)>1)
    thetat=-ksl*sign(u1);
else
    thetat=-ksl*u1;
end
```

ANEXO 9. CÓDIGO REPLICADOR

```
function [u1 ,C_w1,C_w2,C_w3] = fcn( t,te,tc1,rc1,iec1,C_wp1,C_wp2,C_wp3)
    tc=zeros(1,3);
    rc=zeros(1,3);
    iec=zeros(1,3);
    tc=tc1;
    rc=rc1;
    iec=iec1;
    off=0.2;%
    beta2=0.1;
  e=0.8*10;%
  alpha= 1;%8 Tracking
    C_fit=zeros(1,3);
    C_w=[C_wp1 C_wp2 C_wp3];
    C_fits=((1-exp((-t+alpha-off)/beta2))>0)*(1-exp((-t+alpha-off)/beta2));
    C_fitpi=((1-exp((t-alpha-off)/beta2))>0)*(1-exp((t-alpha-off)/beta2));
    C_fit(1)=C_fits;%modo Seguimiento
    C_fit(2)=C_fitpi*((1-exp((-te+e-off)/beta2))>0).*(1-exp((-te+e-off)/beta2));%Modo Arranque
    C_fit(3)=C_fitpi*((1-exp((te-e-off)/beta2))>0).*(1-exp((te-e-off)/beta2));%MOdo Regulados
    fit_m=sum(C_fit.*C_w)/99;
    C_w=C_w+5*C_w.*(C_fit-fit_m)/99;
     C_w1=C_w(1);
     C_w2=C_w(2);
     C_w3=C_w(3);
     u1=(C_w(1)*tc(1)+(C_w(2))*iec(1)+C_w(3)*rc(1))/99;
```

ANEXO 10. DIAGRAMA PLANTA