Diseño e implementación de un algoritmo de seguimiento de trayectorias para el minidron Parrot mambo utilizando – Simulink/Stateflow.

Estive Leandro Trujillo Loaiza

Universidad Antonio Nariño

Facultad de Ingeniaría Mecánica, Electrónica y Biomédica

Bogotá D.C, Colombia.

2022

Diseño e implementación de un algoritmo de seguimiento de trayectorias para el minidrone Parrot mambo utilizando – Simulink/Stateflow

Estive Leandro Trujillo Loaiza

Proyecto de grado presentado como requisito parcial para optar al título de:

Ingeniero Mecatrónico

Director:

PhD. Christian Camilo Erazo Ordoñez

Universidad Antonio Nariño

Facultad de Ingeniería Mecánica, Electrónica y Biomédica

Bogotá D.C, Colombia

2022

Agradecimientos.

El cariño, amor y esfuerzo es invaluable, cualidades características de mi familia, agradezco a mis padres que me educaron con mucha osadía durante todos estos años, para mi siempre han sido un pilar fundamental en la vida, una vida llena de bendiciones y aunque en algunos casos no fue sencillo, siempre he podido contar con ellos, agradezco a mis hermanos que siempre me apoyaron en los momentos difíciles y me han impulsado a salir adelante. De lo mas profundo de mi corazón les doy las gracias por estar siempre conmigo.

Agradezco mucho por la ayuda de mis maestros, mis compañeros, mis amigos y a mi asesor de tesis el Ing. Christian Camilo Erazo Ordoñez y a la universidad Antonio Nariño en general que me otorgaron grandes experiencias en la vida y el conocimiento para llegar a la culminación de mi tesis.

Resumen

En este documento se presenta la propuesta de diseño e implementación de un algoritmo de seguimiento de trayectorias que se incorporará al sistema de control de vuelo del minidrone Parrot Mambo, con el objetivo de realizar vuelos autónomos. El algoritmo de seguimiento se desarrollará en tres fases: inicialmente se procesará una secuencia de imágenes por medio de tres procesos que son; binarización, segmentación, y conteo de pixeles, luego se diseñará el algoritmo por medio del diseño de la máquina de estados para la adquisición y procesamiento de datos y finalmente se validará el algoritmo a través de Simulink/Stateflow considerando cuatro escenarios donde se pondrá a prueba el funcionamiento.

Asimismo en el desarrollo de este documento se observó una efectividad del 93.33% de éxito en las diferentes pruebas de rendimiento del algoritmo, y parte de la eficiencia se presenta en la recolección de datos los pulsos emitidos por los distintos sensores que se generaron a partir de la imagen, la duración de cada pulsación es variable y puede rondar desde 9 s hasta finalizar a los 18 s, y el tiempo de respuesta visto en el movimiento final del dron dura exactamente el tiempo que demoran los pulsos.

PALABRAS CLAVE: UAVs, RC, Algoritmo, Imagen, Resolución, RGB, coordenadas.

Abstract

This document presents the proposal for the design and implementation of a trajectory tracking algorithm that will be incorporated into the flight control system of the Parrot Mambo minidrone, with the aim of performing autonomous flights. The tracking algorithm will be developed in three phases: initially a sequence of images will be processed through three processes that are; binarization, segmentation, and pixel counting, then the algorithm will be designed through the design of the state machine for data acquisition and processing and finally the algorithm will be validated through Simulink/Stateflow considering four scenarios where it will be tested the performance.

Likewise, in the development of this document, an effectiveness of 93.33% success was demonstrated in the different performance tests of the algorithm, and part of the efficiency is presented in the data collection, the pulses emitted by the different sensors that were generated from the image, the duration of each pulse is variable and can range from 9 s to end at 18 s, and the response time seen in the final movement of the drone lasts exactly the time that the pulses take.

KEY WORDS: UAVs, RC, Algorithm, Image, Resolution, RGB, coordinates.

Contenido

Lista de figuras

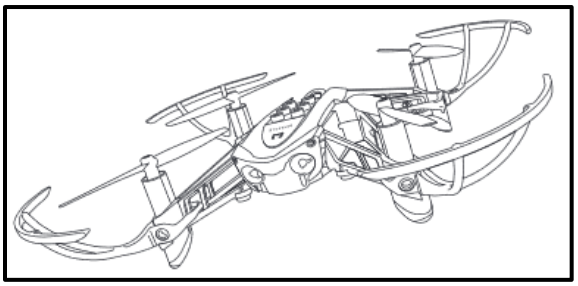
Lista de diagramas

Lista de tablas

1 Introducción

Durante los últimos años, el diseño de controladores automáticos de vehículos aéreos no tripulados (drones) ha impactado en diversas aplicaciones que van desde ambientes domésticos como fotografía, diversión, etc. hasta ambientes profesionales tales como industria militar, seguridad, transporte [1], [2]. Una de las ventajas de trabajar con drones es la maniobrabilidad y el manejo autónomo que se puede programar fácilmente por un usuario a través de diversas plataformas como Matlab, Python entre otras, según V. Sgurev *et al.* [3]. La estructura de los drones puede variar según el número de rotores, tamaño, rango de vuelo, etc. y los hay desde el tricopter a octacopter, siendo los más utilizados el quadcopter o cuadricóptero [4], ver figura 1-1.

Figura 1-1: Esquema Dron Parrot MAMBO

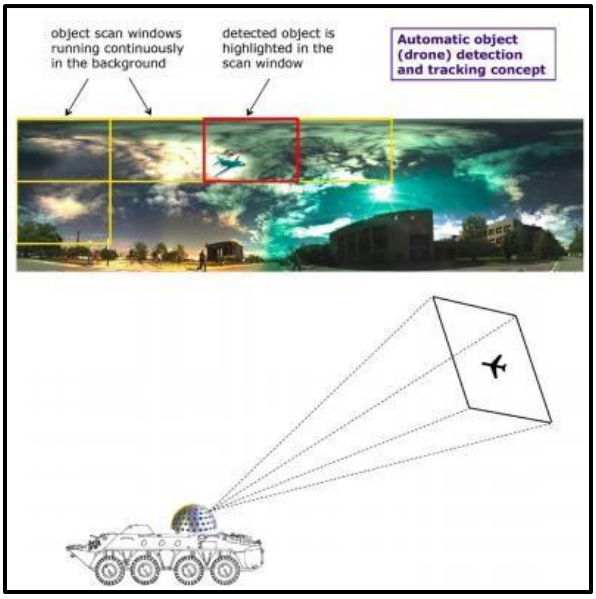


Fuente: [5]

1.1 Estado del arte

Anteriormente, los drones hacían uso de sistemas de navegación satelital para identificar objetivos o para realizar tareas de monitoreo. En muchos casos, debido a que los drones contaban con una memoria limitada, se recolectaba los datos y luego se procesaba la información off-line utilizando softwares especializados, como es el caso de B. Demir *et al* [6] que en su artículo científico realizan el procesamiento de imágenes para detectar objetivos en vuelo con un dispositivo que cuenta con 16 cámaras para recolectar a información suministrada en 360° a un rango de 700m de distancia, proceso mediante el cual se recolectan fotogramas que pasan a una PC para procesarlo con una muy alta resolución como se ve en la Figura 1-2.

Figura 1-2: Visualización del concepto general del sistema de detección de drones



Fuente: [6]

Hoy en día, nuevos desarrollos tecnológicos han hecho posible la incorporación de técnicas de visión por computador, obteniendo así vuelos autónomos y confiables que permiten desarrollar tareas de monitoreo y reconocimiento. R. Collins *et al.*[7], diseñaron un algoritmo capaz de identificar un objeto en movimiento, algo que se observa con el vehículo en la Figura 1-3. En este trabajo, al controlador de vuelo se incorporó un algoritmo que delimita en un área el objetivo y por medio del procesamiento de los fotogramas es posible seguir el objeto.

Figura 1-3 Captura del programa empleado para seguir un auto

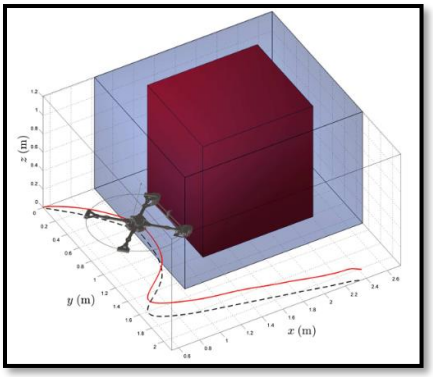
Fuente: [7]

En [8] se utilizó un dron para realizar un levantamiento cartográfico para el instituto nacional de geografía de korea, obteniendo imágenes de mayor resolución, ver Figura 1-5. Un control por realimentación de la salida para seguir las trayectorias utilizadas comúnmente en la

industria del cine junto con un algoritmo de planeamiento de trayectorias fue propuesto en [9]. Este trabajo es desarrollado en [10] para seguir trayectorias automáticamente en tiempo real mientras el dron evita obstáculos dinámicamente. En el área de agricultura de precisión, un sistema de control PID para guiar vehículos aéreos autónomos en la inspección de cultivos fue presentado en [11]. En este trabajo se desarrolló un método basado en la orientación de las texturas del cultivo, extraídas por el sistema de visión incorporado en el dron.

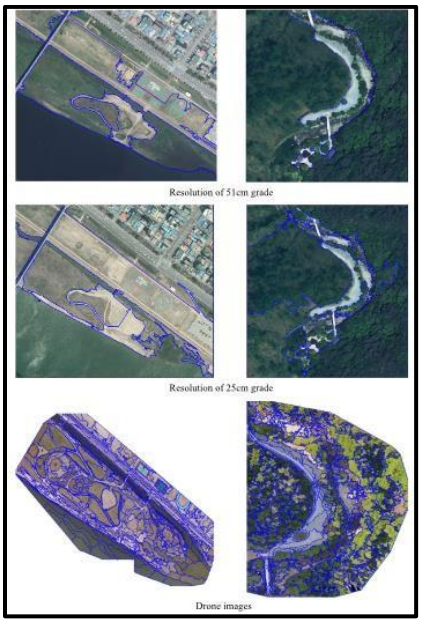
En [6] se desarrolló un controlador de vuelo predictivo basado en modelo, que evita obstáculos de manera robusta, ver Figura 1-4

Figura 1-4: Evaluación de la evasión de obstáculos.



Fuente:[12]

Figura 1-5: Captura con dron terreno Korea



Fuente:[8]

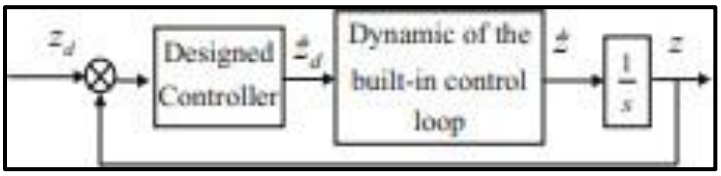
Entrando más puntualmente en la programación existen diversos algoritmos de control utilizados en los drones como es el caso del mini dron mambo programación explicada por

V. Sgurev *et al.* [3] donde explican que este dron en particular se puede controlar cambiando la fuerza de elevación total U ver Figura 1-6 y Figura 1-7, también que La dinámica del cuadrotor a lo largo del canal de altitud junto con el bucle de control incorporado se puede estimado como una función de transferencia de primer orden:

$$G_z^{in}(s) = \frac{\dot{z}(s)}{\dot{z}d\ (s)} = \frac{k_z}{Tzs+1} \qquad (1.1)$$

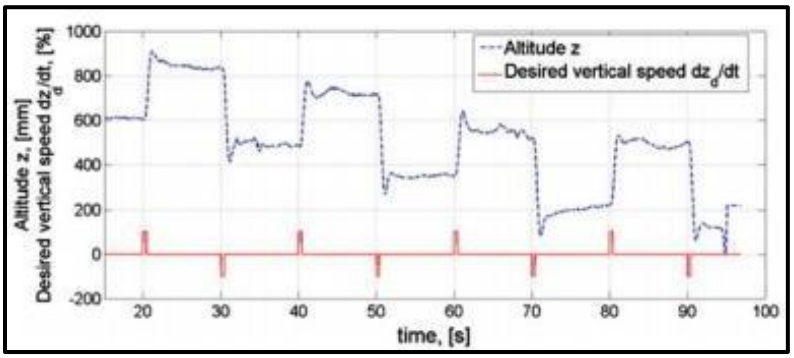
Esto con el fin de crear un algoritmo de control para generar el movimiento básico vuelo del dron

Figura 1-6 Lazo de control diseñado para el desplazamiento vertical



Fuente: [3]

Figura 1-7: Resultados experimentales para el modelado de dinámica de movimiento vertical



Fuente: [3]

1.2 Planteamiento del problema

Diseñar e implementar sistemas de navegación autónoma de drones, es una tarea compleja, que incluye diferentes tareas como son la localización, el mapeo de escenarios, el planeamiento de trayectorias y la evasión de obstáculos, como se mencionó anteriormente.

Crear estos programas y organizar una correcta secuencia de instrucciones entre la coordenadas y análisis de las imágenes recolectadas requiere de un adecuado procesamiento teniendo en cuenta que la resolución proporcionada por los sensores ópticos de estos dispositivos es muy baja, razón por la cual se hace necesario el desarrollo de plataformas que faciliten la programación de los sistemas de navegación de naves no tripuladas.

Si bien en la literatura, se encuentran diversos estudios sobre el control de posición de drones y técnicas de procesamiento, no se reportan muchos trabajos que relacionen estas dos áreas.

Es por esto que en este proyecto se busca dar respuesta a la siguiente pregunta. ¿Es posible implementar un sistema de seguimiento autónomo de trayectorias utilizando el minidrone Parrot mambo?

Vale la pena aclarar que el dron cuenta con un sistema de control de posición funcional, por lo tanto, en este proyecto se busca desarrollar solamente el algoritmo de seguimiento de trayectorias y acoplarlo a este sistema de control.

1.3 Objetivos

1.3.1 Objetivo general

Implementar un sistema de seguimiento de trayectorias a través del minidrone Parrot mambo, mediante la herramienta *Simulink/Stateflow*.

1.3.2 Objetivos específicos

- Realizar un preprocesamiento de las imágenes capturadas por la cámara del dron.
- Diseñar un algoritmo de seguimiento de trayectorias basado en la lectura de los sensores ópticos del dron.
- Validar el sistema automático de navegación empleando 4 diferentes trayectorias.

1.4 Metodología

Fase 1: Preprocesamiento de las imágenes capturadas.

Para el procesamiento de las imágenes se hará uso del software *Matlab – Simulink*, donde a partir de complementos del programa en la sección de procesamiento de imágenes y visión por computadora, se discretizará la secuencia de imágenes por medio del complemento *color Thresholder*, app permite aislar el objetivo por medio de selección de color. Este proceso se realiza con el fin de obtener la orientación en la que el dron debe desplazarse para realizar las tareas de seguimiento. En particular se utilizará el bloque *Blob-análisis* el cual mediante un análisis estadístico etiqueta regiones en una imagen binaria. El bloque devuelve cantidades como el centroide, el cuadro delimitador, la matriz de etiquetas y el recuento de manchas [13]. El bloque *Blob-Análisis* admite señales de entrada y salida de tamaño variable.

Fase 2: Diseño del algoritmo de seguimiento.

Dado que el dron cuenta con varios lazos de control de altitud y de posición desarrollados en *Simulink*, en esta etapa se pretende desarrollar un algoritmo de seguimiento compatible con el sistema ya implementado. Para el diseño del algoritmo se utilizará la herramienta *Stateflow*, donde a partir de la información de la posición del dron y del procesamiento de la secuencia de imágenes capturadas por la cámara del minidrone se pretende construir una

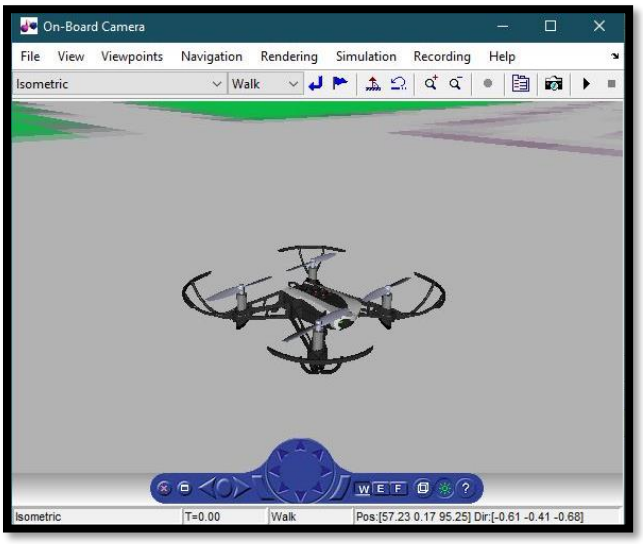
máquina de estado que permita que el dron sea capaz de seguir un objetivo dado. Se realizarán diferentes pruebas con diversos controladores con el fin de obtener el mejor rendimiento, de modo que la orientación del dron sea la adecuada.

Fase 3: Validación del algoritmo

Mediante el procesamiento de los datos adquiridos de las simulaciones y experimentación en *MATLAB* por medio del complemento *Simulink* y empleando las herramientas de *Stateflow* y *Blob-Análisis*. se puede realizar el diseño del algoritmo al realizar pruebas en deferentes escenarios, esto permite imitar entornos de prueba y permite realizar trayectos virtuales simulados por computador (en este caso se simulará el dron en vuelo).

Al establecer los entornos de prueba como es el caso de un camino calle o carretera se extraen las simulaciones numéricas tales como la precisión con la cual el dron sigue una trayectoria y la precisión del tiempo que demora en realizar el recorrido establecido dentro del programa ver Figura 1-8.

Figura 1-8: Ambiente de prueba visto en el programa.



Fuente propia.

Así mismo se establecen factores como el campo de visión del dron, altura optima, que tipos de trayectoria se emplearan en el dron, entre otros, el seguimiento realizado por el dron se lleva a cabo empleando un trayecto delimitado en mi zona residencial o universidad mediante una línea de color (inicialmente rojo) de un grosor inicial de 10cm cuya longitud dependerá del punto al cual se desea que el dron llegue, ya que la cámara del dron tiene un campo de visión de 120° se inicia con una altura de 1m permitiendo establecer límites idóneos para el manejo inicial de este dispositivo.

Teniendo en cuenta que este tipo de dispositivos cuentan con una capacidad limitada de memoria de la cual dispone el dron como se menciona en la referencia [6] se espera validar

el algoritmo de seguimiento de trayectorias de manera que no afecte el funcionamiento de este.

Se espera que en la práctica y etapa de experimentación realizar más pruebas de campo mucho más extensas en el cual se pondrán a prueba las capacidades del algoritmo seleccionando una variedad de objetivos, en específico 4 trayectorias grandes o pequeños delimitando la capacidad de seguir un rastro al asignar ángulos de giro, altura y grosor del trayecto a seguir.

1.5 Justificación

Este trabajo se enfoca primordialmente en la implementación de un algoritmo que permita la autonomía del dron al momento de realizar una trayectoria establecida por el usuario. Este método es certero y genera ventajas como tener una mejor perspectiva de a qué se enfrenta el usuario al momento de llegar a su objetivo teniendo en cuenta el trayecto, dado que puede utilizar el dron para trazar y visualizar una ruta mientras realiza otra operación o actividad generando mayor autonomía.

Emplear un dron es más eficiente que emplear otro tipo de dispositivo UAVs, ya que tiene mayor estabilidad y es más controlable que un helicóptero RC (por dar un ejemplo), dando una notable maniobrabilidad y acceso a lugares complejos.

En Bogotá se están empleando drones que permiten identificar situaciones de orden público, esto se explica en las referencias [14] y [15] en el cual los autores expresan que una manifestación puede volverse rápidamente en una situación de total inseguridad, donde se están implementando drones para la fuerza pública con el objetivo de supervisar zonas de alta aglomeración o vigilar accesos en las estaciones de transporte público, siendo el uso del dron una operación efectiva.

2 Marco Teórico

2.1 Modelo del dron

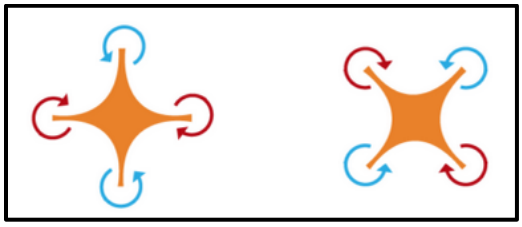
2.1.1 Funcionamiento

Los drones más específicamente los de tipo quadcopter, son drones que funcionan mediante cuatro hélices que rotan a gran velocidad generando así una presión equivalente en cada una de las aspas lo cual permite que se genere una aceleración ascendente que supera la gravedad. Para que funcione este tipo de drones es necesario que el movimiento de dos de los rotores se muevan en el sentido de las agujas de reloj y dos en sentido contrario siendo esta característica la que garantiza un aterrizaje seguro [4], ver Figura 2-1 y Figura 2-2.

Figura 2-1: Vista aérea del minidrone parrot mambo fly

Fuente:[16]

Figura 2-2: Sentido correcto de los 4 rotores tipo + y X



Fuente: [4]

Dependiendo del trabajo simultaneo de las cuatro hélices es la fuerza que se crea para realizar el empuje que lleva el dispositivo hacia arriba también es posible variar dicho empuje que se ejerce respecto a cada hélice para conseguirla estabilidad completa del dispositivo. Cabe anotar que el peso total del cuadricóptero se divide entre cada uno de los motores y modificando el par de cada uno de ellos, podremos controlar el movimiento en vuelo del dron. Un cuadricóptero dispone de cuatro tipos de movimientos los cuales son [17]:

- **Guiñada:** hacia la derecha o izquierda del eje vertical
- **Inclinación:** hacia la derecha o izquierda del eje longitudinal
- **Cabeceo:** rotación hacia delante o hacia con respecto al eje transversal
- **Altitud:** elevación en vertical.

2.1.2 Ecuación descriptiva del dron

2.1.2.1 Modelo no lineal

Los movimientos del dron en vuelo están dados por un modelo matemático que permite ver cada una de las variables físicas presentes en el dron y como afecta el rendimiento respecto al movimiento de este. En la Figura 2-3 los cuadros I y B indican la inercia dextrógira en el marco de referencia y marco de referencia fijo del cuerpo con respecto al origen como el centro de masa (CoM), respectivamente. En este caso $\zeta^I = [x, y, z]^T$ es la posición y $\eta^I = [\phi, \theta, \psi]^T$ (balanceo, cabeceo y guiñada, respectivamente) es la actitud del minidrone. La matriz de rotación R define la orientación del marco fijo del cuerpo con respecto a el marco inercial. Sea v la velocidad lineal y ω la velocidad angular de la siguiente manera: [18], [19].

$$\begin{aligned} v = \dot{\zeta} = R \cdot v^B \\ \omega = \dot{\eta} = R \cdot \omega^B \end{aligned} \qquad (2.1)$$

Donde, T esta dada por:

$$T = \begin{bmatrix} 1 & s(\phi)t(\theta) & c(\phi)t(\theta) \\ 0 & c(\phi) & -s(\phi) \\ 0 & \frac{s(\phi)}{c(\phi)} & \frac{c(\phi)}{c(\phi)} \end{bmatrix} \qquad (2.2)$$

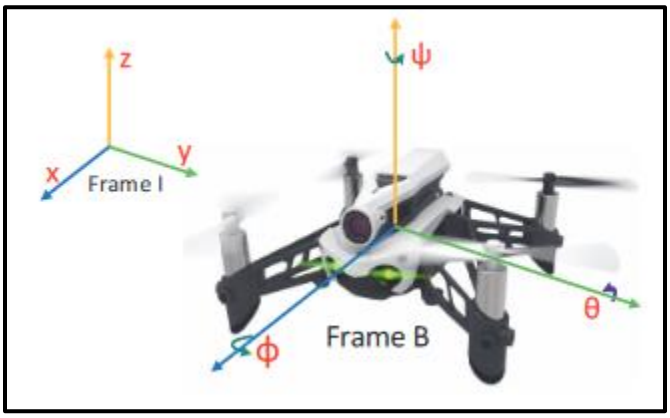
Con la variables c(.), s(.) y t(.) se pueden denotar las funcoines trigonometricas coseno, seno y tangente respectivamente.

Si empleamos la formulacion de Newton-Euler podemos adquirir la dinamica UAV del dron siendo dada por:

$$\begin{aligned} m \cdot \dot{v}^I = F = F^I_{Empuje} + F^I_{Gravedad} \\ I_B \cdot \dot{\omega}^B = \dot{\eta} = \tau^B_{Rotor} \end{aligned} \qquad (2.3)$$

Donde m es la masa del quadcopter, F^Ies la fuerza externa del quadcopter, $I_B$es la matriz de inercia referido al centro de masa del minidrone (CoM) así como $\tau^B_{Rotor} = [\tau_\phi \quad \tau_\theta \quad \tau_\psi]^T$ son los torques aplicados por los rotores.

Figura 2-3: Parrot mambo minidrone con los marcos de coordenada



Fuente: [18]

La fuerza externa del minidrone ejercida sobre el centro de masa esta dada por:

$$F^I_{Gravedad} = [0,0,-mg]^T \tag{2.4}$$

Las fuerzas que ejercen los rotores sobre el minidrone son:

$$F^I_{Empuje} = RF^B_{Empuje} = [0 \quad 0 \quad b\sum_1^4 \Omega_i^2]^T \tag{2.5}$$

Donde b es el coeficiente de empuje y $\Omega_i$es la velocidad angular del rotor i

De las ecuaciones anteriores se deduce que las ecuaciones dinámicas del movimiento son:

$$\dot{v}^I = \begin{bmatrix} \ddot{x} \\ \ddot{y} \\ \ddot{z} \end{bmatrix} = \frac{b}{m}\sum_1^4 \Omega_i^2 \begin{bmatrix} s(\phi)s(\psi) + c(\phi)s(\theta)c(\psi) \\ -s(\phi)c(\theta) + c(\phi)s(\theta)s(\psi) \\ c(\phi)c(\theta) \end{bmatrix} - \begin{bmatrix} 0 \\ 0 \\ g \end{bmatrix} \tag{2.6}$$

$$\dot{\omega}^B = I_B^{-1}\tau^B_{Rotor} = \begin{bmatrix} \frac{1}{I_X} bl(\Omega_4^2 + \Omega_3^2 - \Omega_2^2 - \Omega_1^2) \\ \frac{1}{I_y} bl(\Omega_3^2 + \Omega_2^2 - \Omega_1^2 - \Omega_4^2) \\ \frac{1}{I_z} d(-\Omega_1^2 + \Omega_2^2 - \Omega_3^2 + \Omega_4^2) \end{bmatrix} \tag{2.7}$$

Donde l es la distancia entre el centro del quadcopter y los rotores y d es el coeficiente de empuje o desplazamiento que genera del motor del rotor.

2.1.2.2 Modelo lineal

El modelo lineal del minidrone parrot mambo deriva de forma similar a las ecuaciones de [19], ya que las entradas son definidas como:

$$U = [U_1 \quad U_2 \quad U_3 \quad U_4] = \left[F_{Empuje}^{B} \quad \tau_\phi \quad \tau_\theta \quad \tau_\psi\right]^T \tag{2.8}$$

Para derivar el modelo linealizado el UAV tiene que estar en condición de vuelo utilizado como punto de operación.

$$\begin{gathered} U^0 = [{U_1}^0 \quad {U_2}^0 \quad {U_3}^0 \quad {U_4}^0] = [mg \quad 0 \quad 0 \quad 0] \\ \omega^0 = [p^0 \quad q^0 \quad r^0] = [0 \quad 0 \quad 0] \\ v^0 = [u^0 \quad v^0 \quad w^0] = [0 \quad 0 \quad 0] \\ \eta^0 = [\phi^0 \quad \theta^0 \quad \psi^0] = [0 \quad 0 \quad \psi^0] \\ \zeta^0 = [x^0 \quad y^0 \quad z^0] = [x^0 \quad y^0 \quad z^0] \end{gathered} \tag{2.9}$$

Por lo tanto:

$$U_1^0 = b\sum_1^4 \Omega_i^2 = mg, \quad \Omega_i^0 = \sqrt{\frac{mg}{4b}} =: \Omega_h \tag{2.10}$$

Así, aplicando las ecuaciones (2.7) y (2.8) podemos reformular la ecuación:

$$U = \begin{bmatrix} U_1 \\ U_2 \\ U_3 \\ U_4 \end{bmatrix} = \begin{bmatrix} b & b & b & b \\ -bl & -bl & bl & bl \\ -bl & bl & bl & -bl \\ -d & d & -d & d \end{bmatrix} \begin{bmatrix} \Omega_1^2 \\ \Omega_2^2 \\ \Omega_3^2 \\ \Omega_4^2 \end{bmatrix} \tag{2.11}$$

Utilizando la serie de Tylor podemos linealizar la ecuación (2.11) dando como resultado:

$$U_i = U_i^0 + \frac{dU_i}{d\Omega_i}\Big|_{\Omega_i^0} (\Omega_i - \Omega_i^0) \begin{cases} \Delta U_1 = 2b\Omega_h(\Delta\Omega_1 + \Delta\Omega_2 + \Delta\Omega_3 + \Delta\Omega_4) \\ \Delta U_2 = 2bl\Omega_h(\Delta\Omega_3 + \Delta\Omega_4 - \Delta\Omega_2 - \Delta\Omega_1) \\ \Delta U_3 = 2bl\Omega_h(\Delta\Omega_2 + \Delta\Omega_3 - \Delta\Omega_1 - \Delta\Omega_4) \\ \Delta U_4 = 2d\Omega_h(-\Delta\Omega_1 + \Delta\Omega_2 - \Delta\Omega_3 + \Delta\Omega_4) \end{cases} \tag{2.12}$$

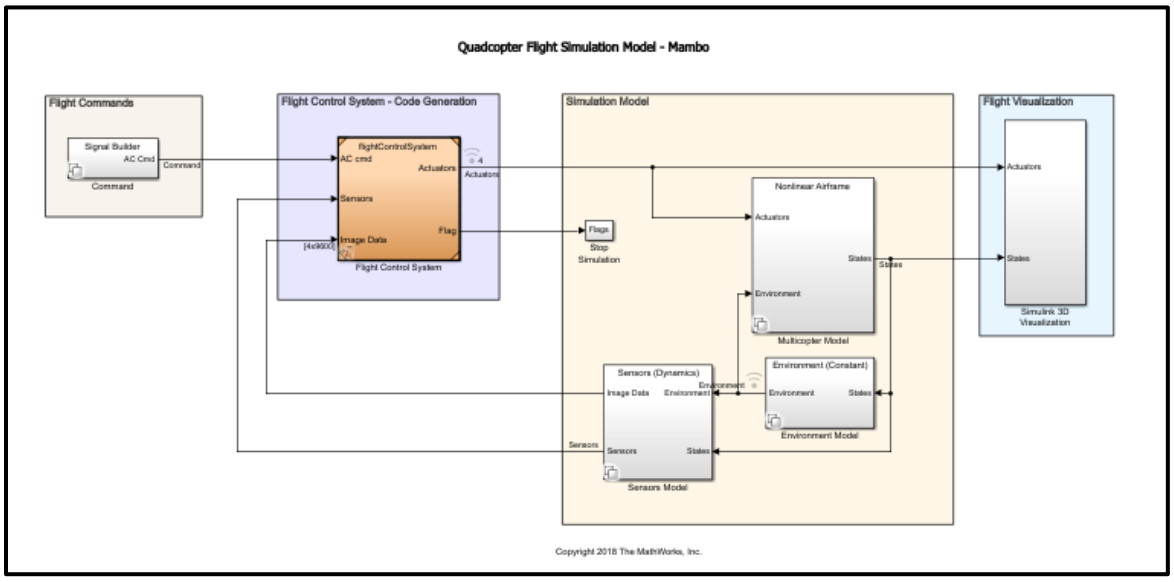
Donde $\Delta\Omega_i = \Omega_i - \Omega_h$ y $\Delta U_i = U_i - U_i^0$.

2.1.3 diagrama de bloques del simulador del dron

El ambiente de simulación de Matlab-Simulink permite descargar un template cuya información está distribuida en cuatro secciones que son las siguientes: Flight Commands, Flight Control System - Code Generation, Simulation Model y Flight Visualization, estos bloques son los que proporcionan básica para poder realizar un modelo virtual de vuelo del minidrone véase Figura 2-4. El diagrama base permite diseñar y crear algoritmos de control de vuelo para minidrones Parrot ya que facilita la implementación de algoritmos de forma inalámbrica por medio de Bluetooth, este ambiente de simulación permite el acceso a los

sensores integrados, como los sensores ultrasónicos, acelerómetros, giroscopios y de presión de aire, así como la cámara orientada hacia abajo [20].

Figura 2-4: Diagrama de bloques del minidrone parrot mambo



Fuente propia.

2.2 Stateflow

La herramienta o Toolbox Stateflow que nos proporciona Matlab permite modelar, editar y simular de manera lógica ciertas decisiones que pueden ser vistas como circuitos secuenciales ya que puede representar sistemas dinámicos que van cambiando de estado a otro, así mismo su interfaz de diagramas de flujo permite el uso de comandos simplemente seleccionando el bloque gráfico que se desee programar, una explicación un poco más profunda de ello nos dice:

> *Stateflow* proporciona un lenguaje gráfico que incluye diagramas de transición de estado, diagramas de flujo, tablas de transición de estado y tablas de verdad. *Stateflow* se puede emplear para describir cómo reaccionan los algoritmos de *MATLAB* y los modelos de *Simulink* a las señales de entrada, los eventos y las condiciones basadas en el tiempo.
>
> Stateflow permite diseñar y desarrollar control de supervisión, planificación de tareas, gestión de fallos, protocolos de comunicación, interfaces de usuario y sistemas híbridos.
>
> Con *Stateflow*, podrá simular lógica de decisión combinatoria y secuencial que se puede simular como un bloque dentro de un modelo de *Simulink* o se puede ejecutar como un objeto en *MATLAB*. La animación gráfica permite analizar y depurar la lógica durante la ejecución. Las comprobaciones en tiempo de edición y en tiempo de ejecución garantizan que el diseño sea coherente y esté completo antes de la implementación [21].

Esta herramienta es altamente compatible con los demás complementos de *Matlab* lo que ayuda a una mayor diversidad de alternativas a la hora de programar, cabe aclarar que este

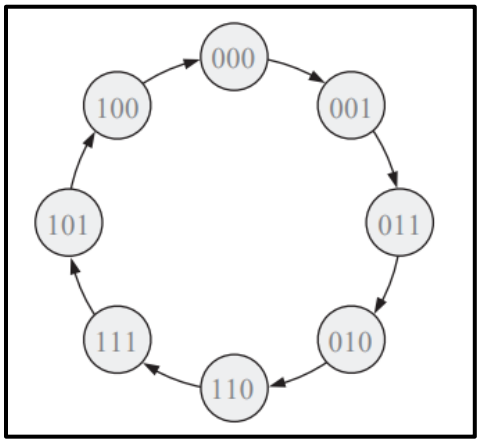
toolbox es un entorno de programación grafico basado en máquinas de estados finito, es decir que es un modelo de comportamiento que consiste en acciones y estados de transiciones a otros estados.

2.2.1 Máquina de estados

Una máquina de estados es un sistema lógico que muestra una secuencia de estados condicionada por la lógica interna y las entradas externas. Cualquier circuito secuencial que exhibe una determinada secuencia de estados.

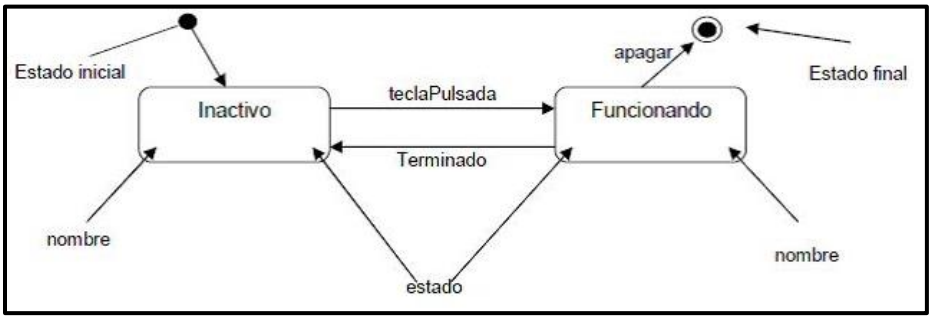
- **Diagrama de estados** según Thomas L. Floyd [22] un diagrama de estados muestra el avance de estados en un sistema cuando se aplica una señal además nos muestra de manera grafica la secuencia de estados en un y las condiciones de cada estado además de las transiciones entre cada uno de ellos. Básicamente muestra la secuencia de estados y las distintas condiciones de una secuencia. En los sistemas digitales como las computadoras, las combinaciones de los dos estados, denominadas códigos, se emplean para representar números, símbolos, caracteres alfabéticos y otros tipos de datos ver Figura 2-5 y Figura 2-6.

Figura 2-5: Diagrama de estados para un contador en código Gray de 3 bits.



Fuente: [22]

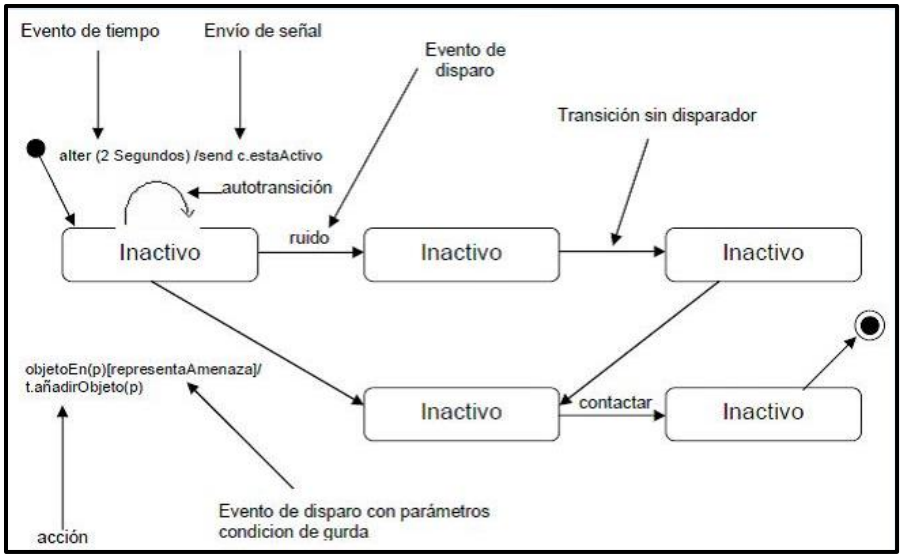
Figura 2-6: Estado



Fuente: [23]

- **Transición de estado** Una transición es una relación entre dos estados lo que implica que al suceder ciertas acciones o evento especifico cambia al segundo estado, según [23] cuando ocurra un evento específico y se satisfagan unas condiciones mínimas se produce el cambio de estado.
- **Leyes de transición** así mismo según [23] Una transición tiene cinco partes, estado origen, evento disparado, condición de guarda, acción y estado de destino; una transición se representa con una línea continua dirigida desde el estado origen hacia el destino y una auto transición es una transición cuyos estados origen y destino son los mismos ver Figura 2-7.

Figura 2-7: Transiciones



Fuente: [23]

- **Para que se utiliza** las máquinas de estado se emplean para simplificar y optimizar gran diversidad de secuencias lógicas, según [24] es un proceso creativo para traducir la descripción del lenguaje en una descripción tabular formal, es realizar un proceso de síntesis bien definido para realizar un dispositivo de lógica programable.

Mas adelante utilizando la máquina de estados se construirá un algoritmo que defina los estados del dron cuando se encuentra sobre la trayectoria que está siguiendo y dependiendo de ciertas condiciones si se encuentra dentro o no de la trayectoria va a cambiar de estado.

2.2.2 Modelizar máquinas de estados finitos

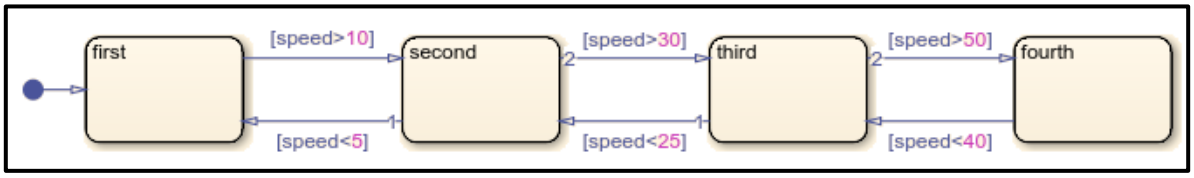
Esta herramienta al ser basada en estados finitos permite examinar errores en el diseño y probar diferentes escenarios de simulación para generar el código desde la máquina de

estados. En [25] nos dicen que las máquinas de estados finitos son representaciones de sistemas dinámicos que pasan de un modo de operación a otro.

Las ventajas de emplear este Toolbox como máquina de estado finito van desde dar un inicio más eficiente a un proceso de software complejo, compactar el interfaz de bloques gráficos para cambiar a otro mono y ayuda en el diseño y creación de modelos más concisos que en algunos casos pueden llegar a ser demasiado complejos

El sistema de control en base a máquinas de estados finitos parte en gran medida la organización de la lógica compleja y es empleado en distintos tipos de diseños ya sean robots aeronaves o automóviles, véase la Figura 2-8, donde se observa el proceso del sistema de transmisión automática de los cuatro cambios de un automóvil, que dependiendo de él margen de velocidades usado genera un salto en el estado y ya que estos estados representan las marchas hace que sean únicos lo que nos indica que solo puede estar un estado activo a la vez.

Figura 2-8: lógica de cambio de marcha del sistema de transmisión automática de cuatro velocidades de un automóvil.



Fuente: [25]

2.2.3 Descripción general Mealy y Moore

Según [26] cumple exactamente la función de una máquina de estados finitos, lo cual significa que actualizar un dato local hace que el estado actual cambie a un nuevo estado, la representación matemática empleada para este caso es el siguiente:

$$X(n+1) = f(X(n), u) \qquad (2.13)$$

Donde:

- $X(n)$ = representa el estado en el paso de tiempo n
- $X(n+1)$ = representa el estado en el siguiente paso de tiempo $n+1$
- u = representa las entradas.

El estado persiste de un paso de tiempo al siguiente paso de tiempo.

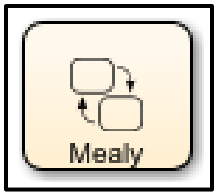
2.2.4 Modelamiento Grafico

Así mismo en [26] se nos muestra que si se desea crear un gráfico de *Stateflow*, el modelo para crear la máquina de estado se llama gráfico clásico hibrido, los cuales combinan la

semántica de los gráficos de *Mealy* y *Moore* con la semántica extendida del gráfico de *Stateflow*.

Crear uno de estos gráficos es sencillo ya que solo es necesario utilizar la línea de código *sfnew -Mealy* o *sfnew -Moore* ver Figura 2-9 y Figura 2-10.

Figura 2-9: Grafico tipo Mealy



Fuente: [26]

Figura 2-10: Grafico tipo Moore

Fuente: [26]

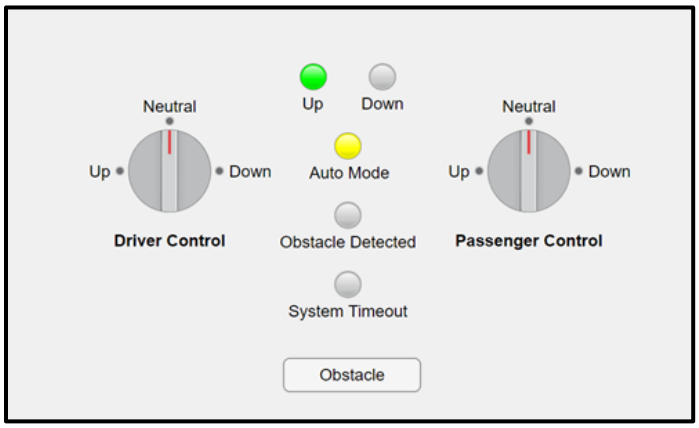
Las ventajas de emplear estos tipos de gráficos son las siguientes:

- Se puede seleccionar el gráfico que mejor se adapte según se desee que cumpla determinada regla semántica empleando el tipo *Mealy* y *Moore* que crea funcione de acuerdo a su criterio.
- Los gráficos de *Moore* son más eficientes respecto a los gráficos clásicos para objetivos como C/C++ y HDL.
- Es importante tener en cuenta que el gráfico de *Moore* es mejor para modelar un ciclo de retroalimentación ya que las entradas no tienen alimentación directa. En cambio, los gráficos *Mealy* y *Classic* tienen alimentación directa y producen un error cuando se presenta un bucle algebraico.

2.2.5 Diseño y control en diagramas de bloques con Stateflow

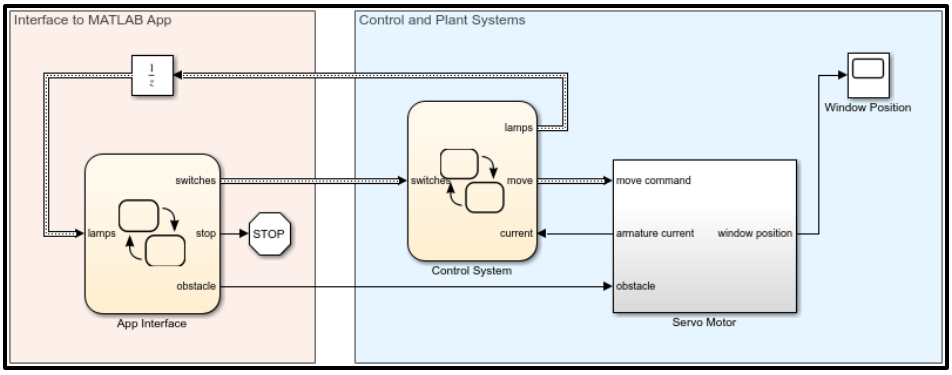
Stateflow se puede emplear para definir máquinas de estado que controlen el comportamiento de las apps de Matlab, en [27] se explica que gráficos de *Stateflow* pueden monitorizar sus interacciones con un app y activar y desactivar los widgets de apps, así mismo para gráficos en Simulink usando el comando *this* se establece una conexión del app y el grafico. ver Figura 2-11 y Figura 2-12.

Figura 2-11: Model a Power Window Controller.



Fuente: [27]

Figura 2-12: command to the power window control system.



Fuente: [27]

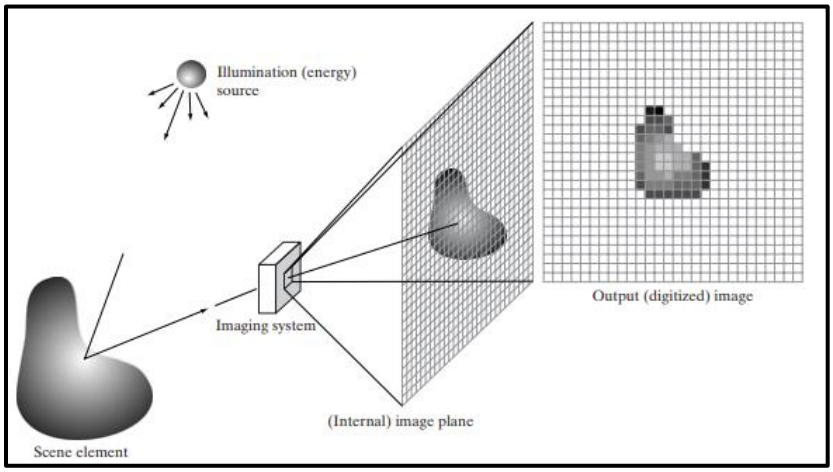
2.3 Binarización y segmentación de imágenes

La binarización y segmentación de imágenes es un medio por el cual se procesa una representación visual del mundo que nos rodea, este proceso se utiliza más frecuentemente en programación a la hora de estudiar una imagen o video de forma digital o analógica y tiene la peculiaridad que para este tipo de prácticas existen diversas topologías de programas y procesos que permiten profundizar en el estudio de una imagen dependiendo del problema al que este sujeto y que se desee resolver.

- **Binarización:** La binarización es una técnica de descomposición que se aplica a una imagen, se refiere a determinar el umbral y suprimir las partes de la imagen que no se desean utilizar, para la binarización una imagen pude ser definida como una función bidimensional la cual presenta una intensidad de luz $f(X,Y)$, donde X y Y representan las coordenadas espaciales y el valor de f en cualquier punto (X,Y) es proporcional al brillo o escala de grises de la imagen en ese punto (cuyos valores están comprendidos entre 0 y 255). Esta técnica puede considerarse como una matriz,

y a este proceso de distribución digital se le denomina *elementos de la imagen* o más comúnmente *pixeles* [28]. La binarización más precisamente consiste en desintegrar o fragmentar una imagen cromática o monocromática en más elementos para realizar un análisis más sustancioso del cual se pueda extraer la información necesitada es decir una imagen binaria Ver Figura 2-13.

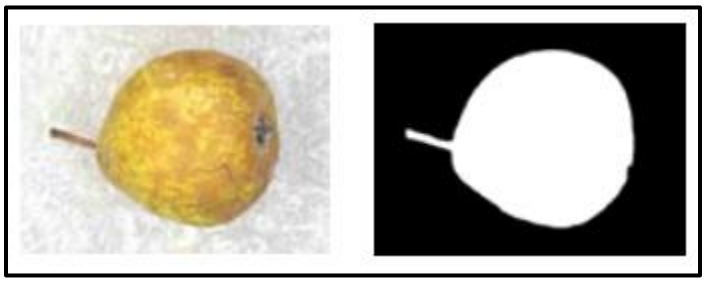
Figura 2-13: Un ejemplo del proceso de adquisición de imágenes digitales.



Fuente: [28]

- **Segmentación:** La segmentación subdivide una imagen en sus partes constituyentes u objetos, con el fin de separar las partes de interés del resto de la imagen, por lo tanto el nivel al que se lleva a cabo esta subdivisión depende de la binarización y del problema a resolver, también se puede establecer la segmentación como la clasificación de los puntos de la imagen (pixeles), indicando las clases a la que pertenecen los diferentes pixeles [28], [29], ver Figura 2-14.

Figura 2-14: Segmentación de imagen

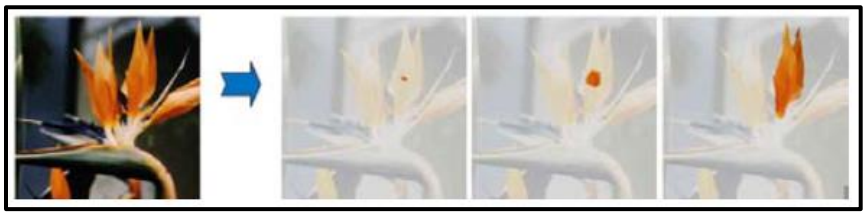


Fuente: [24]

Existen diferentes tipos de fundamentos teóricos para desarrollar un algoritmo de segmentación como la umbralización, detección de discontinuidades, crecimiento de regiones, watershed, transformada de Fourier, segmentación piramidal y la Transformada de Hough siendo el primer fundamento teórico el más sencillo de trabajar [30].

Barea [8] citado por [29] presenta que existen diversas técnicas de segmentación, las que se pueden agrupar en tres: técnicas orientadas al pixel, a los bordes y a las regiones. También se puede examinar algunos métodos como: línea divisoria de aguas (watershed) que a partir de los mínimos en la imagen se aumenta gradualmente el nivel de gris, como si fuera agua que se vierte en un valle, hasta encontrar sus valles vecinos; detección de bordes de las regiones mediante la búsqueda de máximos en el gradiente de la imagen o cruces por cero en la segunda derivada de la imagen; filtros en los que se optimiza una función de costo que considera la exactitud en la posición del borde y la cantidad de bordes detectados; y detección de regiones mediante agrupación de pixeles vecinos con características similares (Región Growing) véase la Figura 2-15.

Figura 2-15: Segmentación orientada a regiones



Fuente: [24]

2.3.1 Blob analysis

Blob análisis es un Toolbox de Matlab que calcula estadísticas para regiones etiquetadas en una imagen binaria, este bloque devuelve información como el centroide, el cuadro delimitador, la matriz de etiquetas y el recuento de blobs (sección de interés de la imagen) así mismo permite señales de entrada y de salida que pueden tener un tamaño variable, los puertos son especificados como variable o matriz y el tipo de datos que maneja es datos de tipo booleano [13].

En blob analysis se hace un proceso de segmentación para en el cual se divide una imagen en segmentos característicos generalmente homogéneos respecto a alguna particularidad. En un fotograma simple o afectado los segmentos pueden corresponder directamente a los fotogramas, pero para un fotograma complejo no necesariamente será el caso. Una simple clasificación en dos conjuntos lleva a una clasificación binaria. Comúnmente esto se logra aplicando un test de *threshold* a la imagen para determinar los valores de los pixeles[31].

2.3.1.1 Funcionamiento

- **Output área** Número de píxeles en regiones etiquetadas, devuelto como vector, ayuda a descartar áreas pequeñas en el procesamiento de los pixeles.
- **Output centroide** Las filas representan las coordenadas del centroide de cada región y M representa el número de blobs.

Por ejemplo, hay dos blobs donde las coordenadas de fila y columna de sus centroides son X_1, Y_1 y X_2, Y_2. Donde la matriz M por 2 de M número de blobs. El bloque genera:

$$\begin{bmatrix} X_1 & Y_1 \\ X_2 & Y_2 \end{bmatrix}$$

- **Output coordenadas** Son las coordenadas del cuadro delimitador, devueltas como una matriz de M por 4 de M cuadros delimitadores para blobs. Cada fila de la matriz define un cuadro delimitador como un vector de cuatro elementos $[x, y, width, height]$ en coordenadas de píxeles. Las filas representan las coordenadas de cada cuadro delimitador, donde M representa el número de blobs.

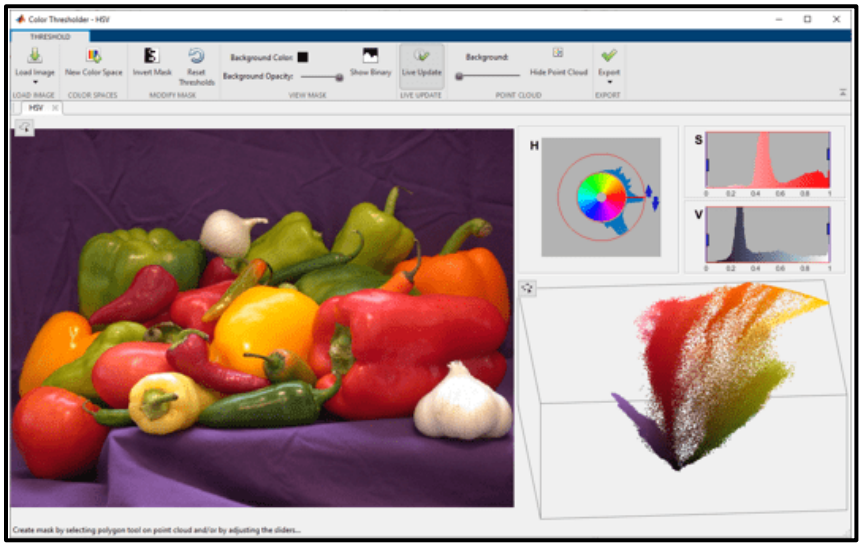
  Por ejemplo, hay dos blobs, donde x e y definen la ubicación de la esquina superior izquierda del cuadro delimitador, y w y h definen el ancho y el alto del cuadro delimitador. Las salidas del bloque [13]

$$\begin{bmatrix} x_1 & y_1 & w_1 & h_1 \\ x_2 & y_2 & w_2 & h_2 \end{bmatrix}$$

2.3.2 Color Thresholder

La app Color Thresholder consiente en la segmentación de imágenes con umbrales para los canales de color en función de diferentes espacios de color, esta app ayuda a crear una máscara de segmentación binaria para una imagen en color permitiendo usar cuatro espacios de color donde se muestra la imagen con sus tres canales de color y el valor de cada color de los pixeles como puntos de una gráfica de espacio de color 3D, así se puede emplear la selección de colores trazando una región de interés (ROI) en la imagen o grafica [32].

Figura 2-16: Color Thresholder app



Fuente: [32]

3 Metodología

Para el desarrollo de este algoritmo su tuvo en cuenta ciertas características del dron (véase anexo A), como la capacidad de la cámara, la memoria, las dimensiones y particularmente el sistema de navegación del dron efectuado por los sensores; el sensor de ultrasonido, sensores de estabilización (acelerómetro de 3 ejes y giroscopio de 3 ejes) y el sensor de flujo óptico.

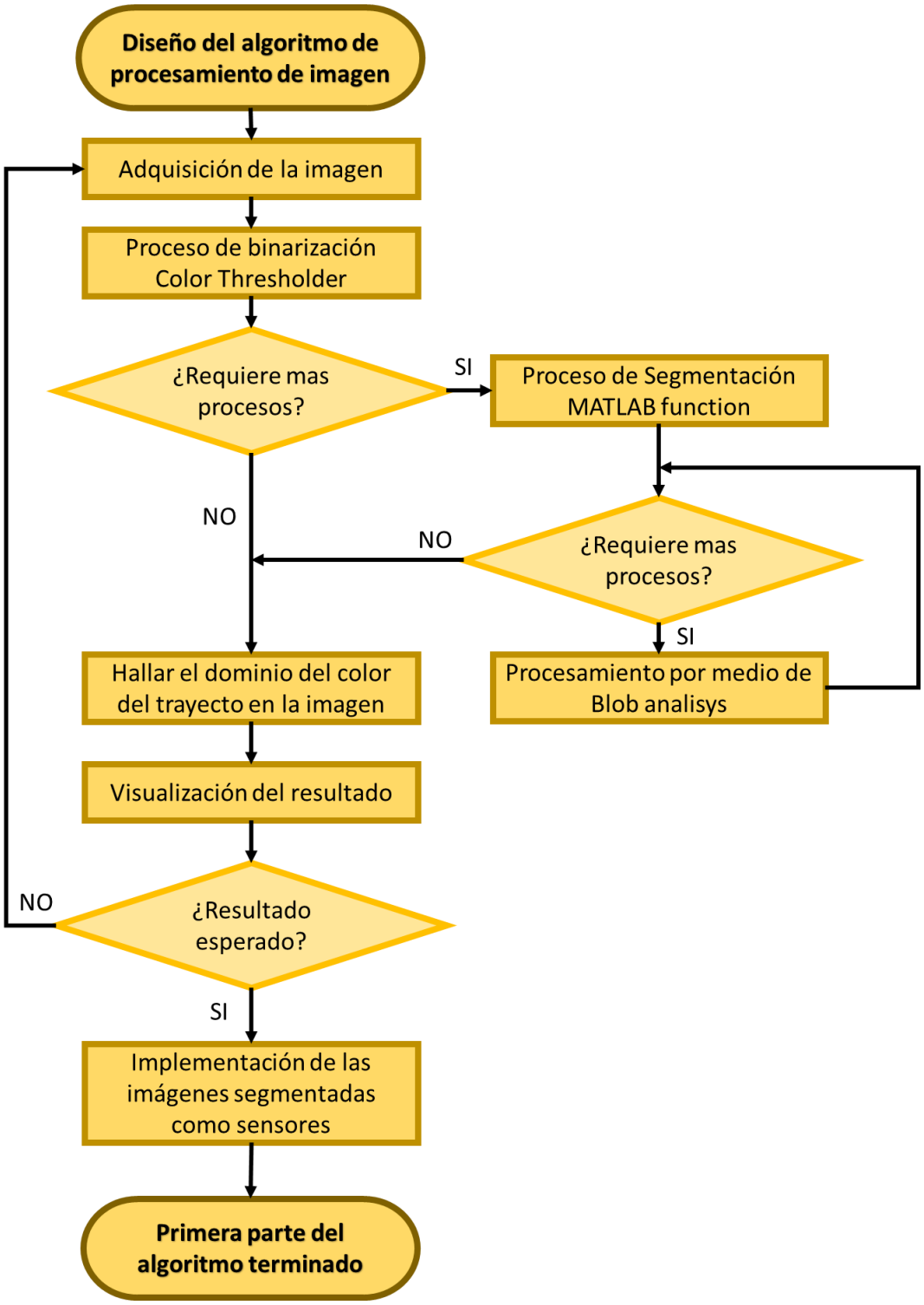
El sensor de flujo óptico (sensor de la cámara del dron) tiene la particularidad de tener integrado el sistema de procesamiento de imágenes y un procesador preprogramado por la empresa (en el caso del minidrone mambo por la empresa Parrot), el procesador se encarga de comparar los datos obtenidos por la cámara y el sensor de ultrasonido para reconstruir el movimiento en los ejes X, Y y Z, permitiendo así la detección de movimiento y segmentación de objetos, este tipo de sensores no requieren ningún tipo de procesador adicional para llevar a cabo su funcionamiento.

3.1 Algoritmo de procesamiento y segmentación de imágenes capturadas por el dron

Para realizar el algoritmo de procesamiento y segmentación de imágenes capturadas por el dron se precisó plantear el proceso lógico a seguir según los tipos de bloques y apps de *Simulink-Matlab*, herramientas imprescindibles para el desarrollo de proyecto (el software *Matlab* fue proporcionado por la Universidad Antonio Nariño)

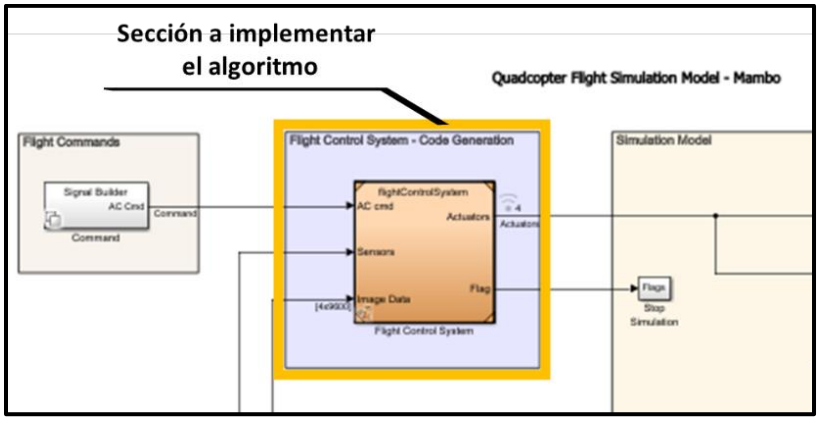
El esquema base para la resolución del procesamiento y segmentación de imágenes se muestra en el siguiente diagrama de flujo:

Diagrama 3-1: Esquema base para el procesamiento y segmentación de imágenes



Cabe aclarar que del diagrama de bloques del minidrone parrot mambo (véase Figura 3-1), la única sección a modificar es Flight Control System - Code Generation ya que el objetivo del proyecto es el diseño e implementación de un algoritmo de seguimiento de trayectorias para el minidrone Parrot mambo.

Figura 3-1: Bloque sin implementar el algoritmo



Fuente propia.

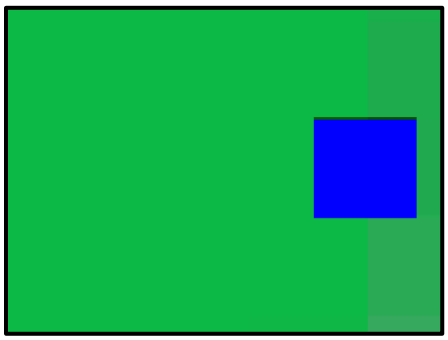
La adquisición de las imágenes se realizó por medio del bloque correspondiente a la cámara del dron llamado *image data*, el cual especifican las intensidades de los componentes rojo, verde y azul del color de la imagen en formato RGB.

3.1.1 Binarización por medio de la app Color Thresholer

3.1.1.1 Procesamiento de un fotograma empleando binarización.

Inicialmente se definió una imagen base para realizar el primer estudio del algoritmo de binarización de imágenes, esta "imagen base" llamada *azul* se extrajo del entorno de simulación del dron en vuelo del minidrone Parrot mambo, y fue seleccionada por la fiabilidad al aplicar el proceso de binarización ya que los colores que posee *azul* tienen variaciones mínimas ver Figura 3-2.

Figura 3-2: Imagen *azul* antes de ser binarizada.



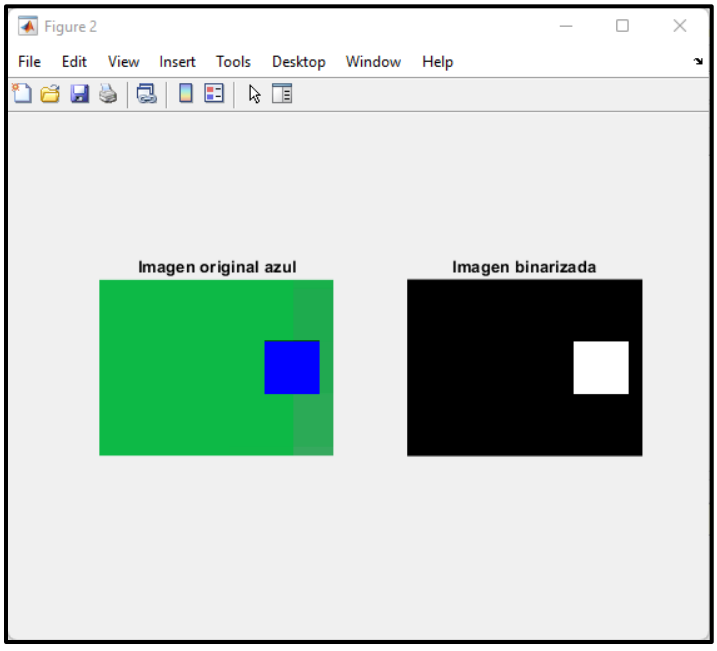
Fuente propia.

Para realizar el proceso de binarización se hace uso de la app Color Thresholder cuya interfaz (véase Figura 2-16) permite cargar una imagen (en este caso *azul*) para realizar el proceso de binarización. Esta app permite escoger entre cuatro modelos o sistemas de colores, el modelo de interés para este proyecto es el RGB por sus siglas en inglés (Red, Green, Blue) ya que el procesamiento de las imágenes capturadas por el dron hace uso de este modelo siendo el más común y más usualmente utilizado.

Posterior a la selección del tipo de sistema de color se delimita la gama de colores aislando el color de interés, en este caso el color azul y finalmente se hace uso de la herramienta binarizar, proceso en el cual cada pixel de la imagen adquiere un valor de 0 o 1 dependiendo del valor asignado a cada color entre 0 y 255.

Con el fin de verificar el resultado de la imagen binarizada, desde la misma app *Color Thresholder es* posible exportar la información al ambiente de programación de Matlab como una variable que puede ser llamada, se realiza una confirmación de las variables empleando el script del apéndice B generando el primer resultado preliminar del algoritmo de binarización como se observa en la Figura 3-2:

Figura 3-3: Resultado preliminar de la binarización 1.



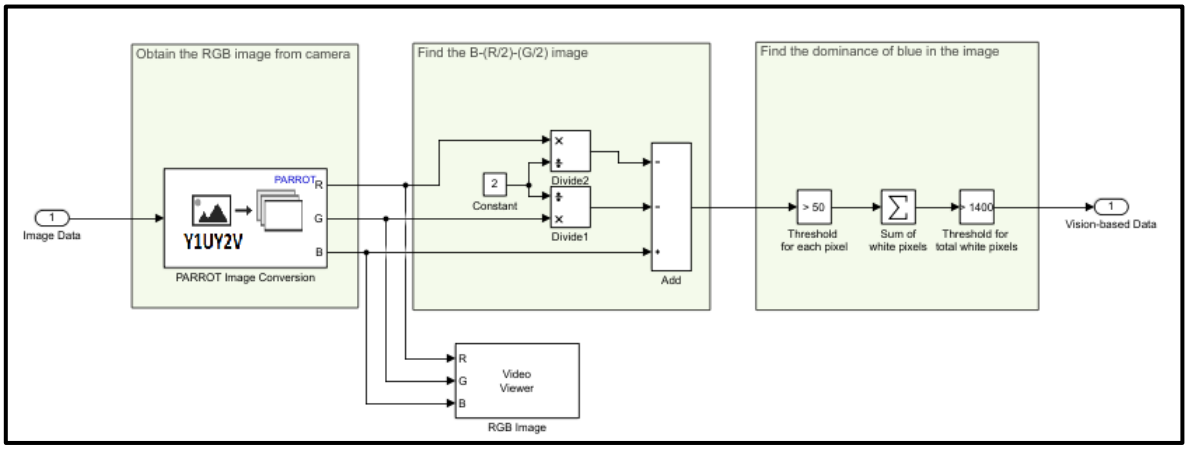
Fuente propia.

3.1.1.2 Procesamiento de video empleando binarización.

Para el segundo estudio se definió como señal de entrada el sensor óptico que permite la adquisición de la imagen RGB obtenida por el dron, configuración que viene por defecto en *Flight Control System - Code Generation* del diagrama de bloques (véase Figura 3-4). Con el fin de realizar un análisis de una trayectoria, se extrae un fotograma del video

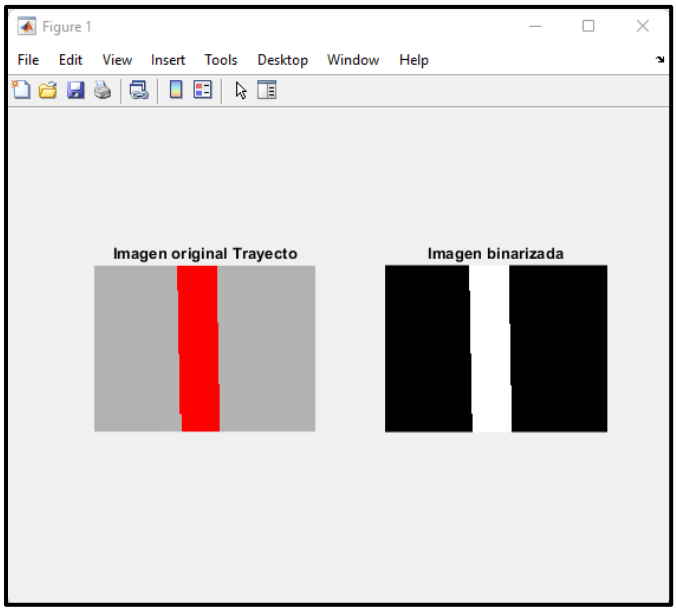
proporcionado por la cámara del dron (sensor óptico) como un ensayo a la que se aplicó el proceso de binarización, el proceso a seguir es el mismo realizado en el primer procesamiento de binarización con la pequeña diferencia del color y la delimitación de la gama de colores que en este caso están enfocados al color rojo, proporcionando así el segundo resultado preliminar, ver Figura 3-5.

Figura 3-4: Configuración por defecto.



Fuente propia.

Figura 3-5: Resultado preliminar de la binarización 2.



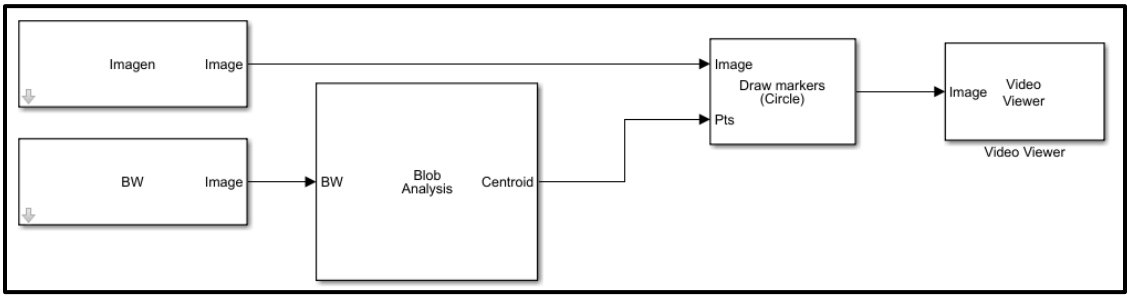
Fuente propia.

3.1.2 Procesamiento de la imagen empleando Blob Analysis

3.1.2.1 Procesamiento de un fotograma empleando Blob Analysis.

Para el algoritmo de segmentación se emplea la herramienta *Simulink* de *Matlab*, cuya interfaz permite usar *Toolbox* como *Blob Analysis* mencionado en el capítulo 2.3.1. Se planteó el diseño de diagrama de bloques (véase Figura 3-6), como un esquema inicial para el análisis teniendo la imagen base *azul* y la imagen binarizada de *azul* a la cual se le asigno el nombre de BW (el proceso de binarización fue realizado en la sección 3.1.1). La imagen binarizada sé procesa mediante el bloque de comando *Blob Analysis* el cual calcula estadísticas de la imagen seleccionando los pixeles característicos.

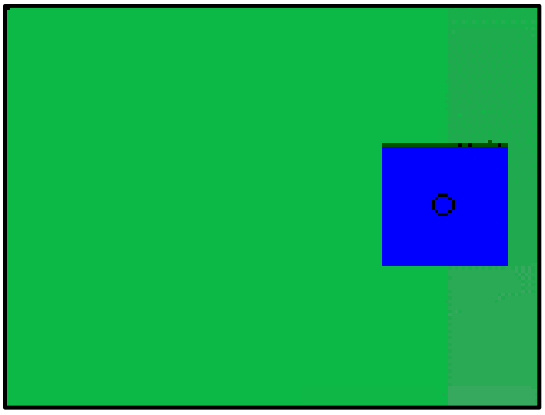
Figura 3-6: Diagrama de bloques empleando Blob Analysis.



Fuente propia.

Al ejecutar el diagrama de bloques el bloque *Blob Analysis* entrega distintos tipos de información en este caso se seleccionó el centroide de BW, dicha información se procesa en otro bloque junto a la imagen base (*azul*) lo cual permite que se dibuje una marca circular en el centro de la imagen y posteriormente se conecta a otro bloque de comando que permite la visualización consiguiente, véase Figura 3-7.

Figura 3-7: Resultado preliminar de la segmentación 1.

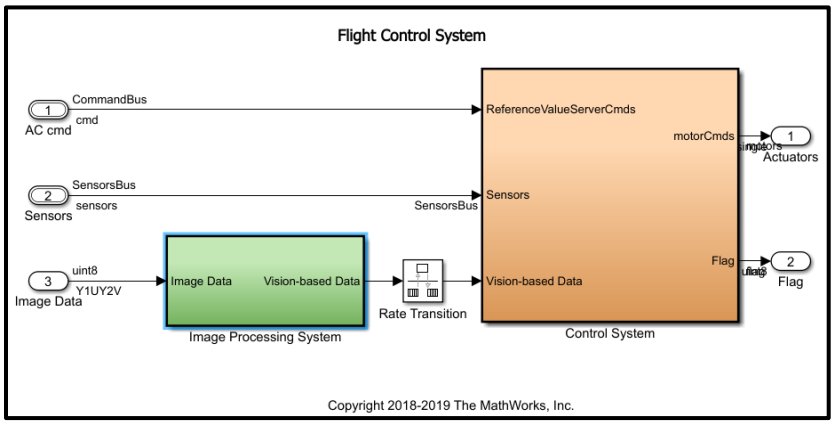


Fuente propia.

3.1.2.2 Procesamiento de video empleando Blob Analysis aplicando Binarización.

En este proceso se implementó el diagrama de bloques de la figura 3-6 al bloque llamado *Image processing system* dentro de la sección *Flight Control Sistem* que se observa en la Figura 3-8, en este caso se realiza un análisis teniendo en cuenta la señal de entrada del sensor óptico mencionado en el proceso de binarización, proceso que permite la adquisición de las imágenes obtenidas por el dron. Teniendo presente lo mencionado anteriormente se extrae el fotograma del trayecto enfatizando que el video adquirido es una vista aérea del dron en vuelo en el ambiente de simulación.

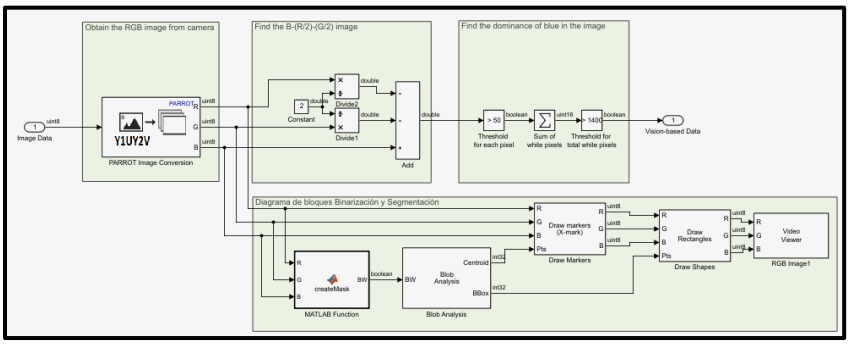
Figura 3-8: Subsistema del control de vuelo.



Fuente Propia

Subsiguientemente para realizar el proceso de binarización por medio de la app *Color Thresholder*, se empleó la herramienta *MATLAB function* permitiendo así obtener el script con la información RGB de uno de los fotogramas del video, la señal que entrega esta herramienta es una matriz con los datos binarios de la imagen original (BW) el script de este proceso se encuentra en el apéndice C. Este script fue modificado para que fuera compatible con el bloque que permite procesar las imágenes y así obtener la segmentación deseada de cada fotograma capturado por la cámara del dron, al emplear la herramienta *MATLAB function* permite editar y configurar el código facilitando que la función sea llamada al ambiente de simulación *Simulink - Matlab* que se observa en la Figura 3-9.

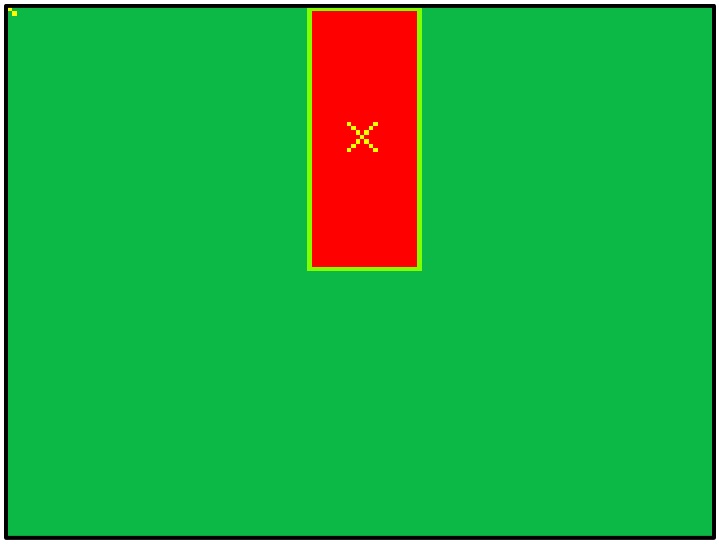
Figura 3-9: Sección del procesamiento de imágenes (Image processing system)



Fuente propia

Una vez realizado el procedimiento de binarización se realiza el mismo procesamiento visto en la Figura 3-6 adicionando en este caso al bloque Blob Analysis un dato adicional de salida, siendo este dato el cuadro delimitador observable en la en la Figura 3-9, así mismo se generó el resultado preliminar del proceso como se observa a continuación en la Figura 3-10:

Figura 3-10: Resultado preliminar del algoritmo de procesamiento y segmentación de imágenes.

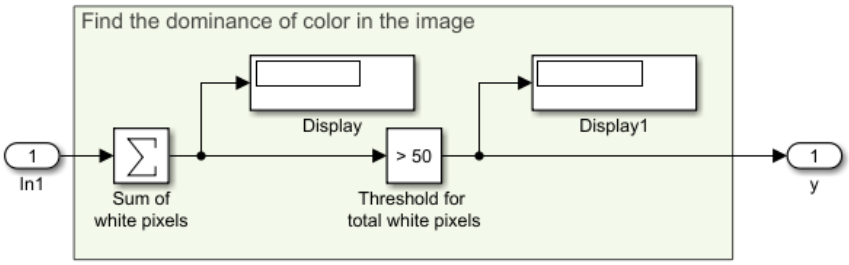


Fuente propia

3.1.3 Procesamiento del dominio de la sección segmentada

Teniendo el fotograma segmentado es posible realizar la sumatoria de los pixeles, en este caso los pixeles blancos que representan el color binarizado, de este modo se logra realizar la comparación de la región deseada (expresada en cantidad de pixeles), dato válido para asignar el valor de verdadero o falso en la señal de salida por medio de una comparación, este proceso se observa en la siguiente figura (ver Figura 3-11).

Figura 3-11: Sumatoria y comparación de pixeles



Fuente propia

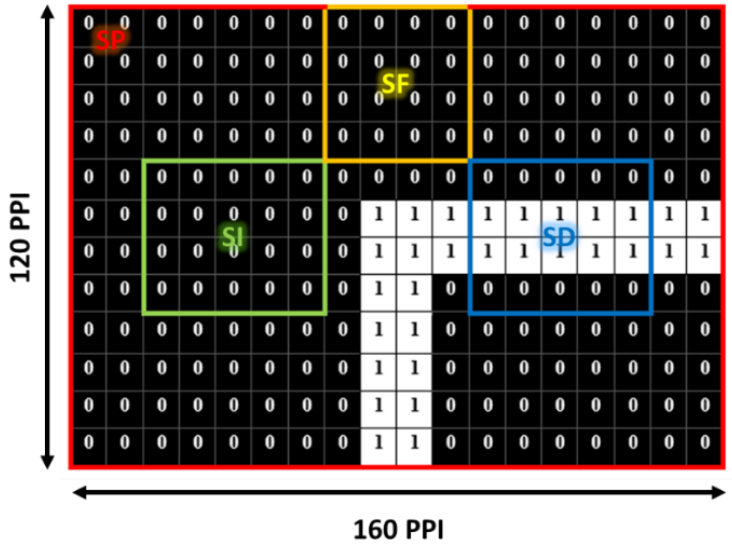
Este proceso es muy importante ya que con los valores de salida es posible trasformar la segmentación de una imagen en una transición para la máquina de estados.

3.1.4 Segmentación del fotograma principal como un nuevo sensor

Ya realizado el procesamiento de binarización, segmentación y establecimiento del dominio por medio de la sumatoria de pixeles es posible procesar la información específica de una sección del fotograma. La sección puede ser establecida delimitando un área de la imagen por medio de matrices, por medio del bloque *video viewer* es posible adquirir las dimensiones de la imagen en pixeles siendo estas las dimensiones del sensor óptico del dron de 120x160 pixeles, sabiendo esto se procede a realizar la fragmentación de la imagen y se puede especificar una región de $(x \leq 120, y \leq 160)$ dimensiones.

En este caso los nuevos sensores están dados por la segmentación de la imagen capturada por el sensor óptico del dron el cual será el sensor principal. Como se evidenció en la sección 3.1.2.2 por sí solo el sensor principal (SP) entrega únicamente como salida de datos el dato binario de la imagen (0 y 1) ver Figura 3-12, para obtener más sensores fue preciso realizar un proceso de división de la imagen para adquirir más secciones y así poder efectuar el estudio de binarización y segmentación a cada una de estas nuevas partes de la imagen.

Figura 3-12: Idea inicial con tres divisiones de imagen representados como sensores



Fuente propia.

Basado en la Figura 3-12 se realiza la delimitación de la sección que será un nuevo sensor por medio de la función $sensor = (x: x_n, y: y_n)$, ya que se parte de una imagen principal se realiza el llamado de la imagen a segmentar en este caso (BW) y se realizan las especificaciones como se observa en la siguiente Figura (véase Figura3-13), los parámetros de las dimensiones de cada sensor pueden variar dependiendo del tamaño del trayecto o de la función que cumpla el sensor.

Figura 3-13: Script de la función de delimitación de nuevo sensor

```
function SensorFrontal = SectionUp(BW)
I=BW;
NewImagen1 = I(1:40,54:107);
SensorFrontal = NewImagen1;

```

Fuente propia.

Una vez realizado el proceso de fragmentación de la imagen principal en diferentes secciones es posible realizar la comparación binaria de cada uno de los sensores permitiendo así que al detectar un valor especifico en un sensor se active una señal referente a una ordene especifica dependiendo de la máquina de estados.

En el caso de la detección del destino o punto de llegada del dron (punto de aterrizaje circular) se realiza un proceso adicional de comparación de variables por medio de las señales que entrega el bloque *Blob Analysis* teniendo en cuenta que los valores a comparar son el área y el perímetro de un círculo se puede realizar la siguiente operación.

$$Area\ de\ un\ circulo = \pi r^2$$
$$Perimetro\ de\ un\ circulo = 2\pi r$$

Si el perímetro del círculo se eleva al cuadrado podemos decir que:

$$Dato = \frac{(2\pi r)^2}{\pi r^2} = \frac{4\pi^2 r^2}{\pi r^2}$$

$$Dato = 4\pi$$

Independientemente del tamaño del círculo el resultado de la operación será 4π por lo tanto se puede establecer los parámetros de la función como se observa en la Figura 3-14 con el fin de obtener la señal de salida deseada para la implementación en la máquina de estados (1 si detecta el circulo, 0 si no lo detecta).

Figura 3-14: Script operación lógica definir la señal de salida del sensor del circulo

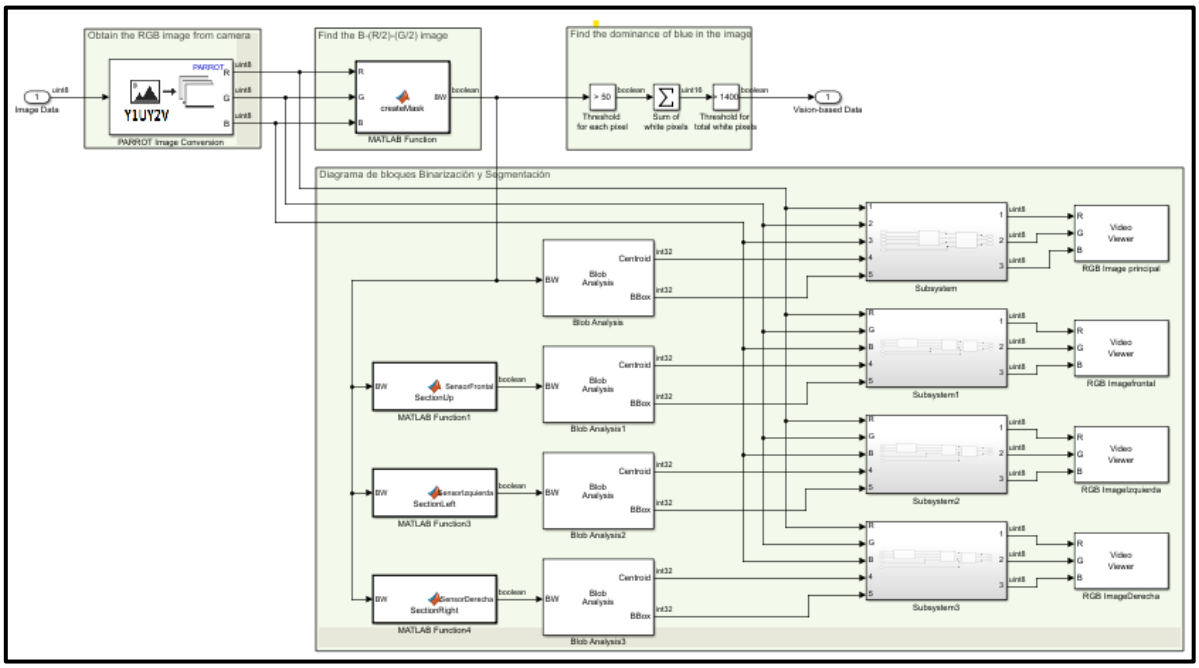
```
function Centro = Stop(Area, Perimetro)
%Perimetro=2pi*r
%Area=pi*(r)^2
Dato=Perimetro^2/Area;
if Dato<14
    Centro=1;
else
    Centro=0;
end
```

Fuente propia

Estableciendo ya todos los parámetros de los sensores y las funciones base de comparación, en el ambiente de simulación de *Simulink-Matlab* se implementan los bloques

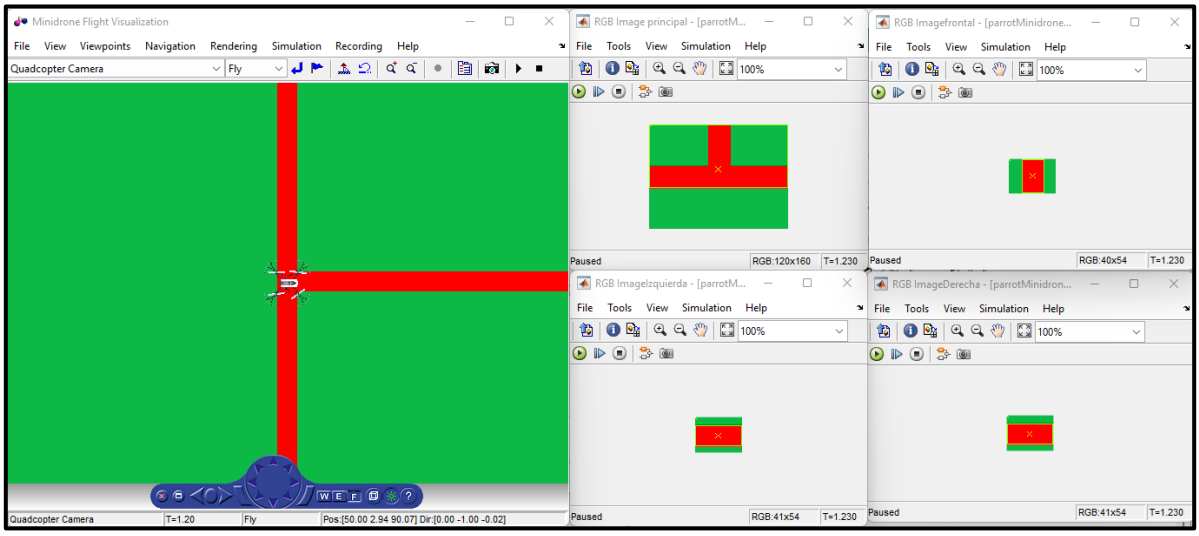
correspondientes a cada sensor véase la Figura 3-15. Para este proceso se finalizó con una visualización previa que muestra de maera visual los 4 sensores segmentados de la imagen principal véase la figura 3-16, en este caso el dron se ubicó en estado de vuelo en el ambiente de simulación cuyo trayecto presenta tres caminos (frontal, izquierdo y derecho) permitiendo visualizar el correcto funcionamiento de los sensores.

Figura 3-15: Implementación de los primeros 4 sensores



Fuente propia.

Figura 3-16: Vista previa de los sensores funcionando en el ambiente de simulación



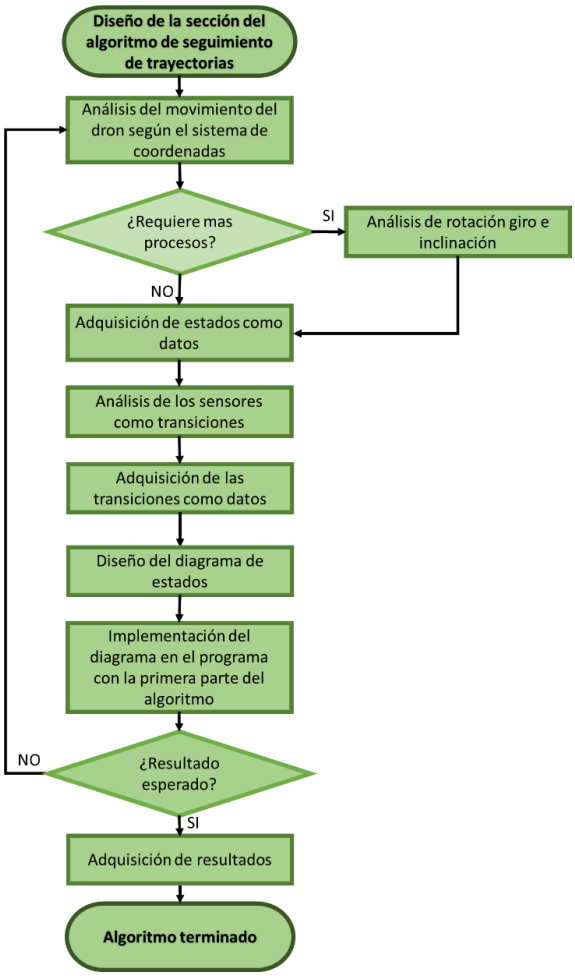
Fuente propia.

3.2 Algoritmo de seguimiento de trayectorias basado en la lectura de los sensores ópticos del dron

Para realizar el diseño del algoritmo de seguimiento fue necesario primero realizar un análisis dependiendo de los requerimientos de movimiento del dron o dicho de otra forma el diseño de la máquina de estados, se observó en una primera instancia que el dron cuenta con seis estados base (puede tener más dependiendo del tipo de análisis realizado), y dichos estados son los que representan las acciones a realizar por el dron, este proceso está sujeto al estado actual y las transiciones que se apliquen, básicamente describen los movimientos ejecutados por el dron.

Igual que en el capítulo 3.1. sé realizo el proceso lógico por medio de un diagrama de flujo en el cual se presentó el esquema base del algoritmo de seguimiento de trayectorias basado en la lectura de los sensores ópticos del dron. Teniendo en cuenta el análisis anterior se sigue el proceso que se indica en el diagrama 3-2:

Diagrama 3-2: Diseño del algoritmo de seguimiento de trayectorias

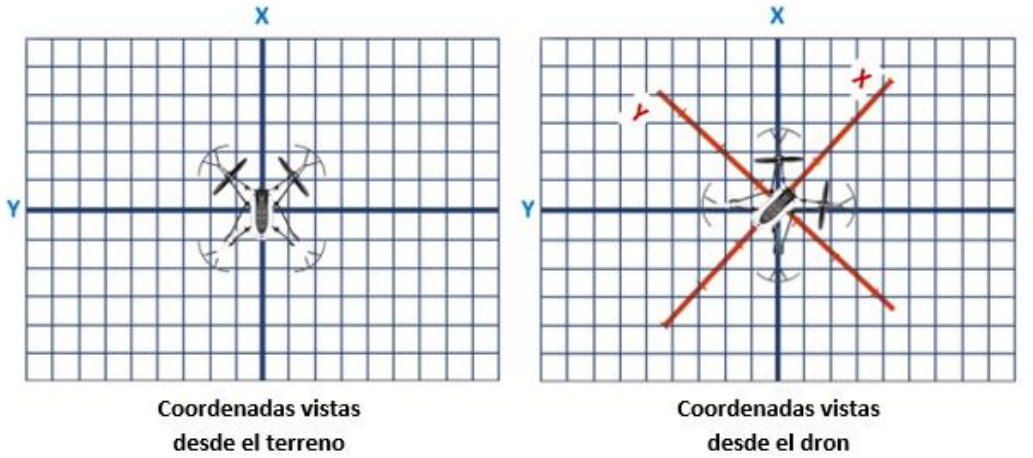


3.2.1 Movimiento del dron a partir de las coordenadas:

Para el diseño de la máquina de estados se tuvo en cuenta las acciones que realizo el dron desde su estado inicial partiendo del suelo a su estado final que es cuando termina de recorrer el trayecto, llega a un punto especifico, se detiene y termina su operación al llegar al suelo. Se inicia con el diseño teórico de un trayecto para visualizar los diferentes estados, una vez establecidos el trayecto de prueba se posiciona el dron en el inicio del trayecto y se representa gráficamente el movimiento del dron en las secciones relevantes del recorrido.

Inicialmente se define el movimiento del dron a partir del sistema de coordenadas, el ambiente donde se desplaza el dron tiene su propio sistema de coordenadas y desde la perspectiva del dron el sistema de coordenadas es diferente (véase la Figura 3-17).

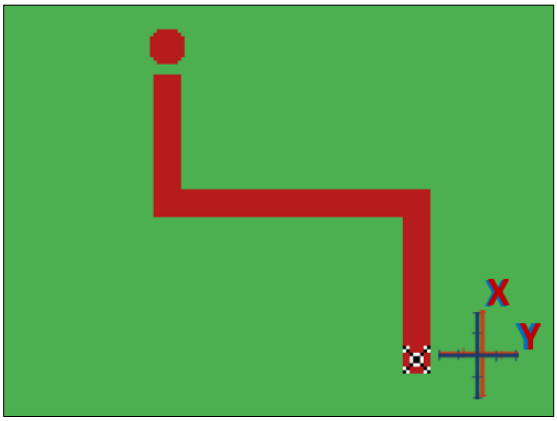
Figura 3-17: Coordenadas de movimiento.



Fuente Propia

El movimiento es tomado a partir de las coordenadas espaciales vistas desde la perspectiva del terreno y del dron, el proceso razonado y realizado es el siguiente:

Figura 3-18: Trayecto y dron teórico posicionado en el trayecto



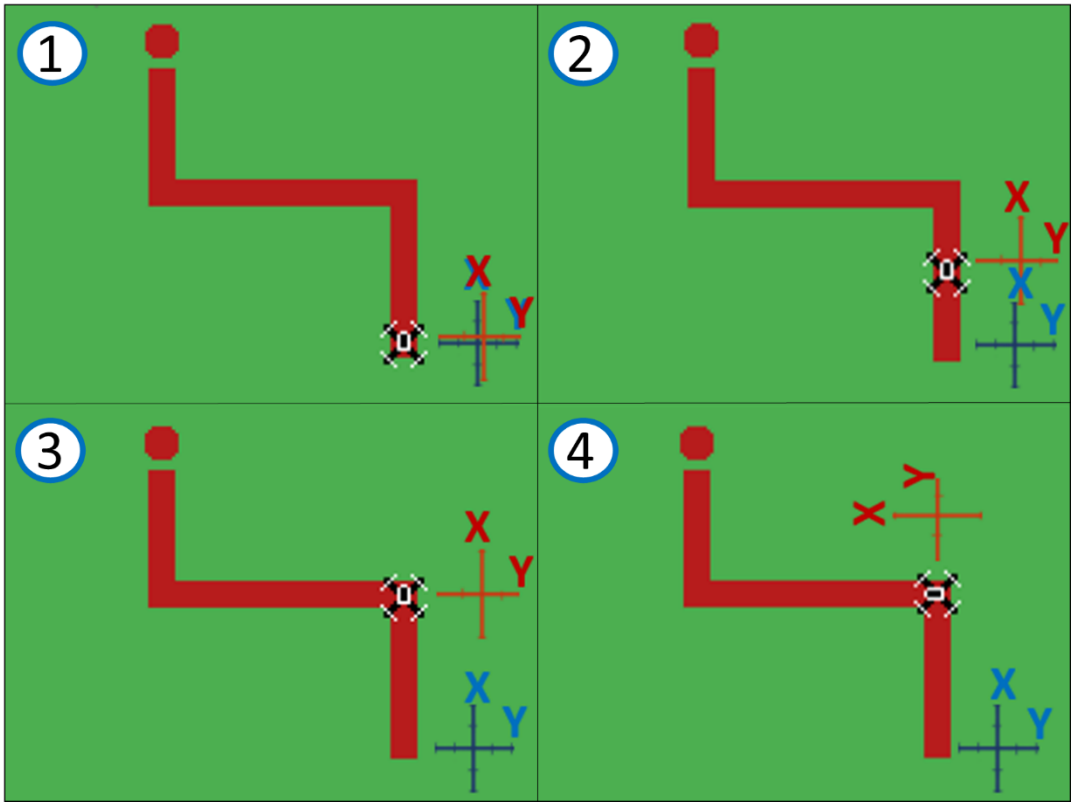
Fuente propia.

Una vez posicionado el dron en el trayecto se analiza el movimiento en la primera seccion del trayecto (vease la Figura 3-19).

Se observa que el primer estado (E1) es el dron en vuelo cuyo desplazamiento se presenta en Z positivo visto en el primer recuadro, posteriormente el siguiente estado (E2) es el de avanzar hacia el frente cuyo deslpazamiento se presenta en X positivo visto en el segundo recuadro.

Subsiguientemente se observó que el dron tiene que detenerse al encontrar un giro por lo tanto este seria el estado (E3) cuyo desplazamiento es nulo en las coordenadas X,Y y Z (permanece estatico en la misma posición). El estado que le sigue (E4) se presenta en el momento que el dron da un giro en este caso a la izquierda lo que indica un desplazamiento rotcional respecto al eje Z dando como resultado un desplazamiento angular en el eje Y positivo.

Figura 3-19: Movimiento del dron primera sección del trayecto



Fuente Propia.

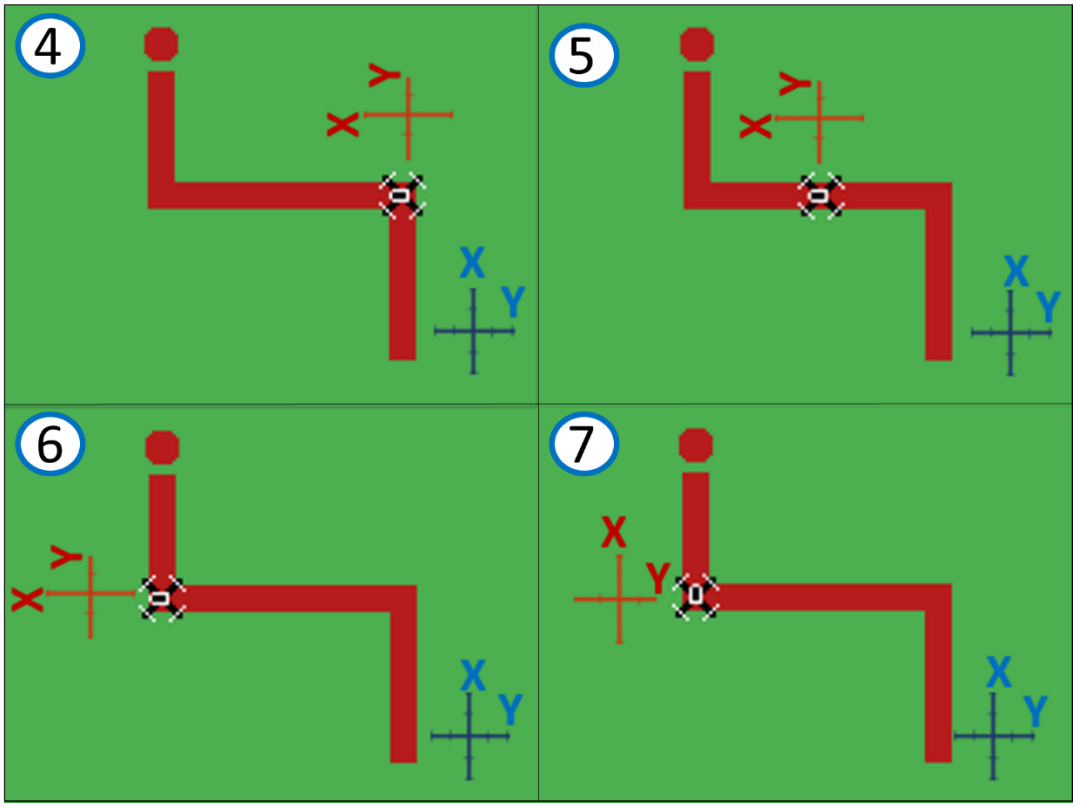
Consecutivamente se analiza la segunda sección del trayecto (véase la figura 3-20), y se observa que el dron luego del desplazamiento angular visto en el cuarto recuadro tiene que desplazarse hacia al frente visto en el quinto recuadro lo que indica que el estado (E2) se repite ya que, aunque presenta un desplazamiento negativo en el eje Y visto desde las coordenadas del terreno y el desplazamiento en X es positivo visto desde la perspectiva del

dron independientemente de que coordenada se emplee sigue presentado el estado de avanzar, por lo tanto, no es necesario agregar un estado diferente al que ya se encuentra.

Una vez llega al punto de giro visto en el sexto recuadro se observa que el dron tiene que detenerse por lo que también se repite el estado (E3) independientemente de las coordenadas que se tomen ya que presenta las mismas condiciones de desplazamiento nulo en las coordenadas X, Y y Z.

Se observa que el dron en el séptimo cuadrante da una rotación sobre el eje Z indicando así un desplazamiento angular negativo sobre el eje Y el cual puede establecerse como un nuevo estado (E5) de rotación.

Figura 3-20: Movimiento del dron segunda sección del trayecto

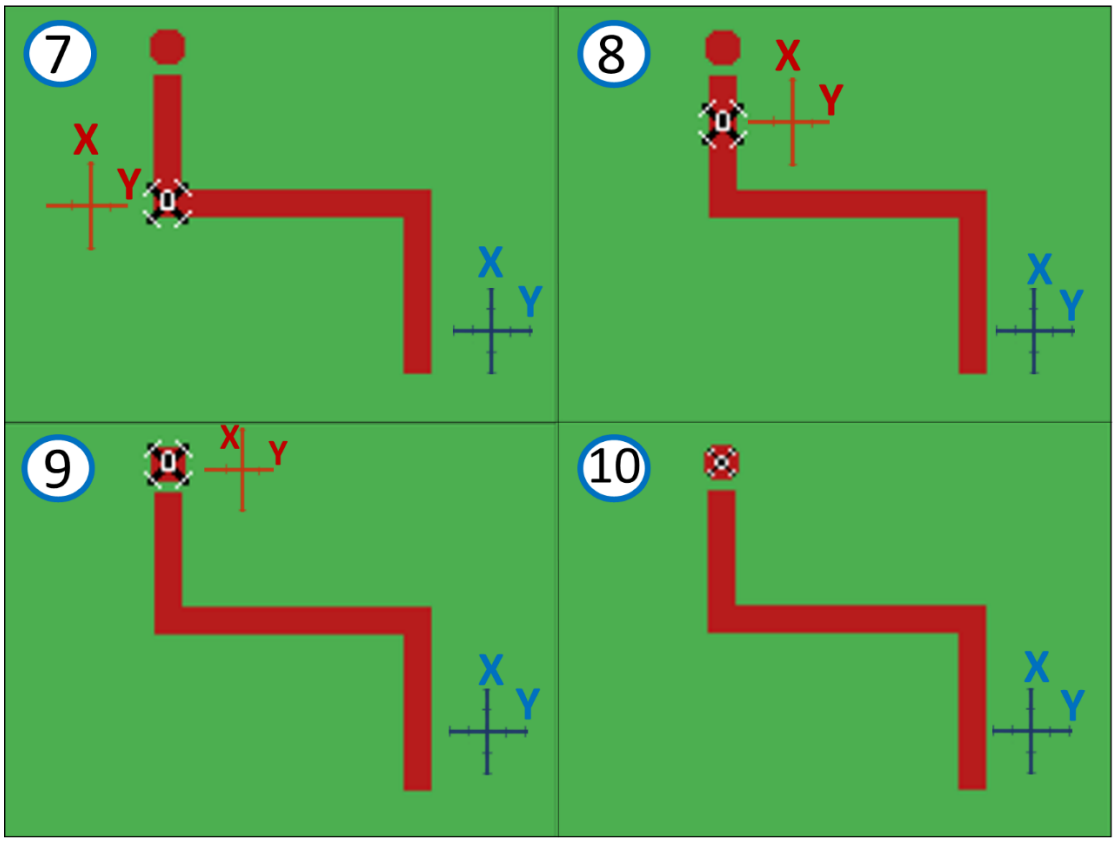


Fuente Propia

A continuación se puede observar en la Figura 3-21 que luego de que el dron realizara la rotacion negativa (E5) presenta un desplazamiento frontal repitiéndose el estado (E2) por lo cual no es necesario agregar otro estado.

Finalmente se observa que el dron llegó a su destino por lo cual se detiene empleando nuevamente el estado (E3) de detenerse y luego presenta un desplazamiento en la coordenada Z negativo indicando que este es el último estado (E6) de fin de operación donde el dron llega a su destino y finaliza su programa.

Figura 3-21: Segundo trayecto y segundo giro



Fuente Propia

Como resultado del estudio se obtuvo la siguiente tabla con las especificaciones de los estados:

Tabla 3-1: Estados y su descripción como datos.

Estado	Descripción
E1	Inicio de vuelo (desplazamiendo vertical)
E2	Avance frontal del dron
E3	Detener el dron en estado de vuelo
E4	Giro hacia la izquierda
E5	Giro hacia la deracha
E6	Fin de vuelo (fin de los procesos)

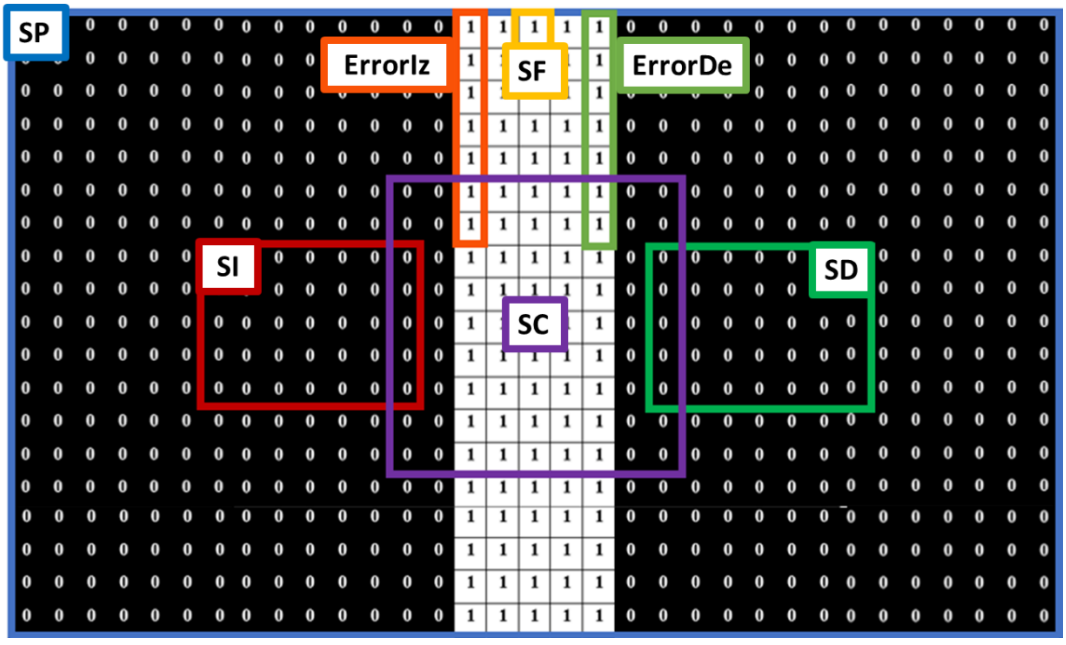
3.2.2 Análisis de los sensores ópticos como transiciones.

Como se mencionó anteriormente en este capítulo, la imagen procesada por medio de la binarización presenta 2 estados (0 o 1) y en una máquina de estados una sola transición no permite un correcto funcionamiento ya que la transición podría activar un estado no deseado en un momento no especificado, por lo tanto y teniendo en cuenta el análisis del movimiento del dron y sus estados se realizaron subdivisiones de la imagen capturada por el sensor principal para obtener los suficientes datos que serán variables de comparación para el diseño de las transiciones.

El objetivo de dividir la imagen es adquirir más funciones de tipo sensor, importancia que recae en que una transición representa la relación de dos estados siendo la transición el medio por el cual se realiza dicha acción especifica y sin una transición correcta el proceso fallará dando cabida a errores no deseados.

En la sección 3.1.4, se realizó una implementación previa de cuatro sensores para analizar el movimiento del dron respecto a los estados, razón por la cual se identificó que era necesario adicionar otros dos sensores para centrar el dron en las diferentes secciones del trayecto, ya que la rotación en Y positivo o negativo no genera el desplazamiento angular exacto y por ende el dron no se centra en todas las partes del recorrido. La idea aproximada empleando todos los sensores se observa en la siguiente figura:

Figura 3-22: Imagen teórica con todos los sensores a emplear ubicados en lugares estratégicos.



Fuente propia.

Basado en la información de las delimitaciones de los distintos sensores adquiridos de la imagen principal se realiza la obtención de las variables que serán empleados en el diseño de las transiciones de estados:

Tabla 3-2: Sensores como transiciones y su descripción

Sensor:	Descripción:
SP	Segmentación del color rojo en la imagen principal
SI	Segmentación del cuadrante izquierdo de la imagen
SD	Segmentación del cuadrante derecho de la imagen
SF	Segmentación del frontal de la imagen
SC	Segmentación del centro de la imagen
ErrorIz	Segmentación de la parte izquierda del trayecto
ErrorDe	Segmentación de la parte derecha del trayecto

3.2.2.1 Implementación de las transiciones:

Como se ha mencionado anteriormente las transiciones son las que dan paso de un estado a otro, en ese orden de ideas teniendo los estados y los datos de transición solo es cuestión de asignar un proceso a cada transición, es decir en el caso de la transición 1 del estado (E1) dron en vuelo, al estado (E2) avanzar, se requiere que el proceso que se lleve a cabo sea que la imagen principal (SP) detecte el color rojo dando paso así a la transición.

El análisis para todas las transiciones aplica el mismo proceso dependiendo del estado que se desea ejecutar primero y como resultado de este análisis se obtuvo la siguiente tabla:

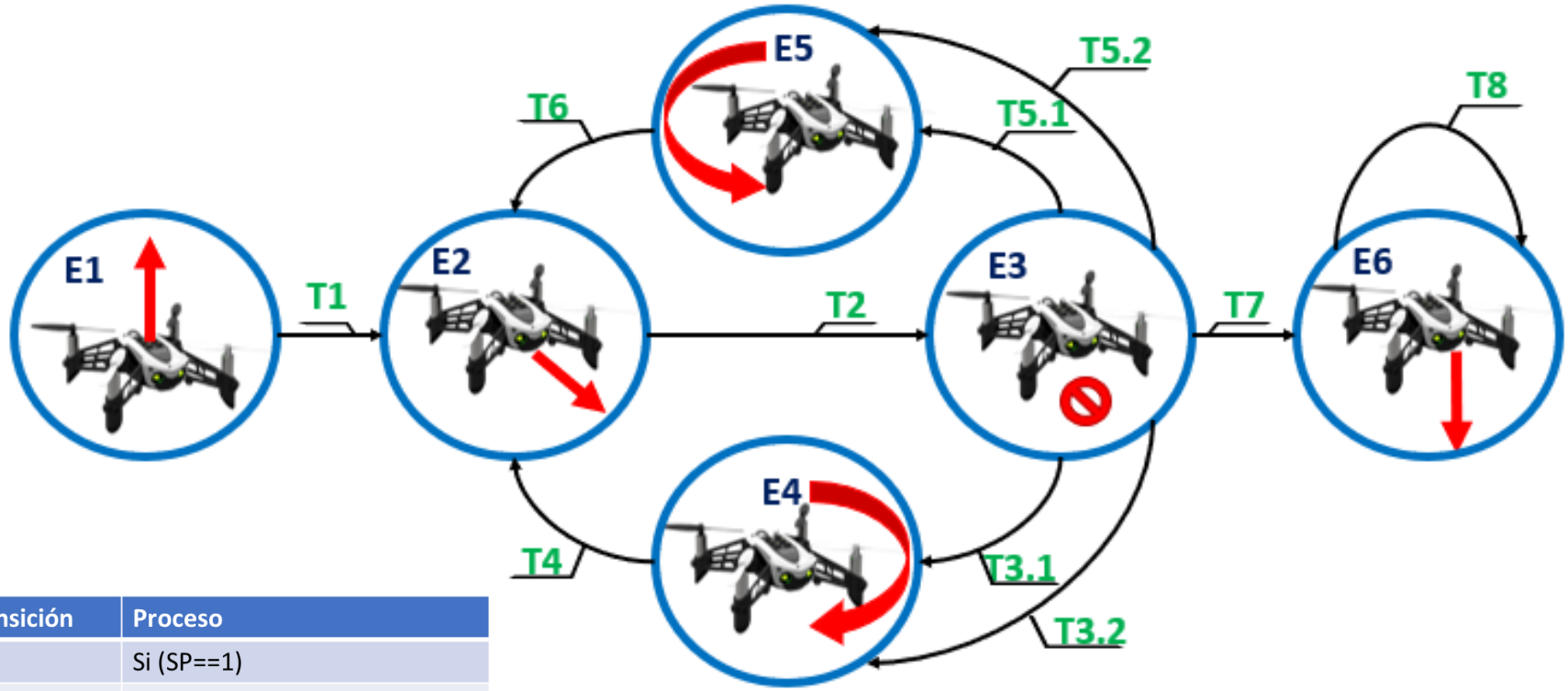
Tabla 3-3: Tabla de transiciones segun su proceso

Transición	Proceso
T1	Si [SP==1]
T2	Si [SP==1 && SF==0]
T3.1	Si [SF==0 && SI==1]
T3.2	Si [ErrorIz==0 && ErrorDe==1]
T4	Si [SF==1]
T5.1	Si [SF==0 && SD==1]
T5.2	Si [ErrorIz==1 && ErrorDe==0]
T6	Si [SF==1]
T7	Si [SF==1 && SD==0 && SI==0]
T8	Si [SC==1] acción {zout=0;}

3.2.3 Diseño del diagrama de estados:

Para el diseño del diagrama de estados se tiene en cuenta la información de las Tabla 3-1 para los datos de los estados y la Tabla 3-2 para los datos de los sensores y la Tabla 3-3 para el proceso de transiciones. El diseño de este proceso fue basado en el paso a paso del recorrido preliminar del dron del capítulo 3.2.1 para lograr el movimiento correcto. Con esa información se realizó el diagrama de estados generando la representación gráfica del modelo, (véase el Diagrama 3-3). Empleando este diagrama el algoritmo podrá seguir cualquier trayecto que tenga parámetros parecidos al utilizado en el ejemplo del capítulo 3.2.1.

Diagrama 3-3: Diagrama máquina de estados



Transición	Proceso
T1	Si (SP==1)
T2	Si (SP==1 && SF==0)
T3.1	Si (SF==0 && SI==1)
T3.2	Si (ErrorIz==0 &&ErrorDe==1)
T4	Si (SF==1)
T5.1	Si (SF==0 && SD==1)
T5.2	Si (ErrorIz==1 &&ErrorDe==0)
T6	Si (SF==1)
T7	Si (SF==1 && SD==0 && SI==0)
T8	Si (SC=1) Acción {zout=0}

Estado	Descripción
E1	Dron en vuelo
E2	Dron avanzando
E3	Dron en vuelo estático
E4	Giro a la izquierda
E5	Giro a la derecha
E6	Fin de operación

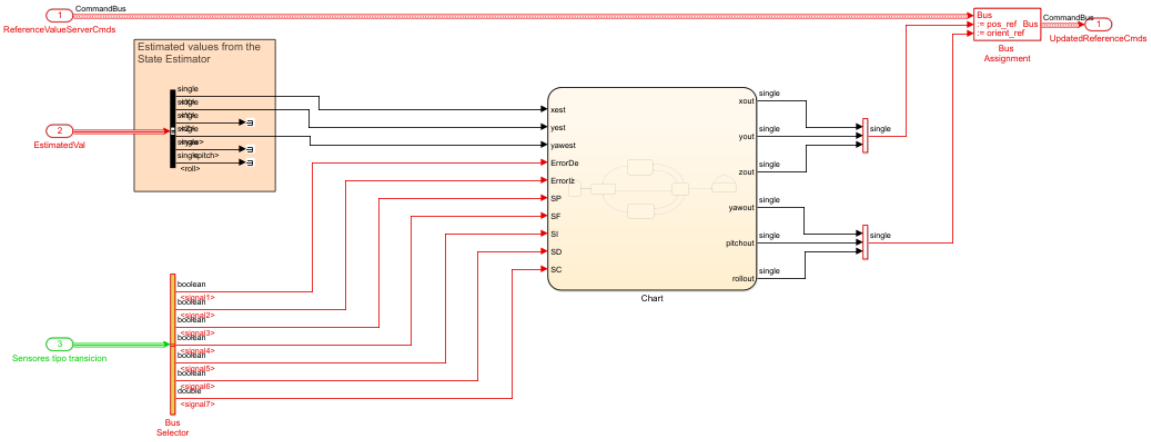
Sensor	Descripción
ErrorDe	Fuera del trayecto derecho
ErrorIz	Fuera del trayecto izquierdo
SP	Sensor imagen principal
SF	Sensor frontal
SI	Sensor izquierda
SD	Sensor derecha
SC	Sensor circulo

3.2.4 Implementación de los dos algoritmos en el programa

Una vez obtenido el diagrama base de estados se implementa el mismo diagrama en el algoritmo de seguimiento de trayectorias en el ambiente de simulación de *Simulink-Matlab*, para esto es necesario abrir la sección de control *system - Path planning* (véase Figura 3-8), el proceso de creación del diagrama consistió en especificar la acción que se desea que el dron realice en cada uno de los bloques que en este caso representan los estados, los bloques son muy versátil por lo que la implementación del mismo permite especificar qué tipo de acción tiene que realizar el estado antes de ejecutar la transición, proceso realizado por medio de variables adquiridas en los anteriores capítulos.

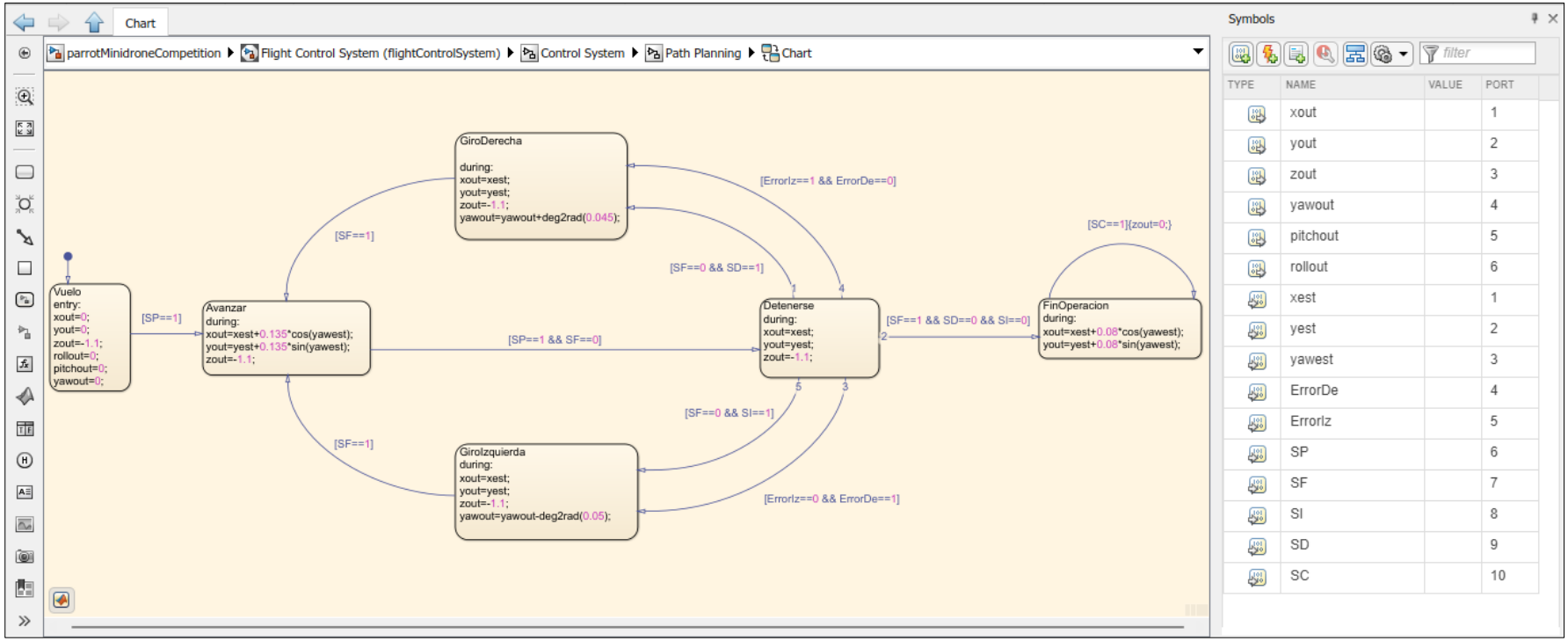
Revisando los parámetros del capítulo 3 el proceso de la implementación de la primera parte del algoritmo (procesamiento y segmentación de imágenes capturadas por el dron) al algoritmo de seguimiento de trayectorias resulta de la unión de valores de salida los bloques del procesamiento de imagen a la máquina de estados, el proceso final que se realizó fue indicar los parámetros de entrada de datos los sensores al bloque *State flow-state chart* y especificar los datos de salida y especificar (véase Figura 3-23), con este proceso se finalizó el algoritmo (véase Figura 3-24).

Figura 3-23: Unión de los datos de procesamiento de imagen con la máquina de estados.



Fuente propia

Figura 3-24: Maquina de estados en el ambiente de simulación Simulink-Matalab Resultado final



Fuente Propia.

4 Resultados y validación del sistema autónomo de seguimiento de trayectorias

En esta sección se observa el comportamiento del posicionamiento y desplazamiento del dron respecto a las perturbaciones en los ejes X y Y debido al desplazamiento angular de los ejes (roll, pitch, yaw), así mismo se analiza como las señales de los sensores son los que generan dicha perturbación y se realiza el análisis del desplazamiento estimado (el desplazamiento final) comparado con el desplazamiento programado por el dron.

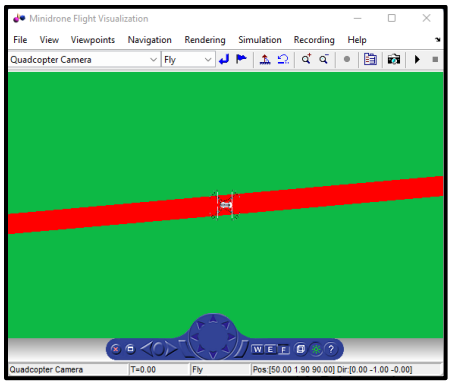
A continuación, se analiza gráficamente la distancia recorrida junto con una visualización grafica del resultado final del movimiento realizado por el dron después de haber ejecutado el algoritmo.

Una vez realizado este proceso se realiza el mismo análisis en tres pistas con variación angular presente en distintas secciones del trayecto y variación en la distancia total recorrida, esta información es determinante con el fin de evidenciar el correcto funcionamiento del algoritmo y demostrar que su eficiencia al realizar este proceso.

4.1 Posicionamiento y desplazamiento del dron

En una primera instancia se creó una sección de trayecto cuya longitud aproximada es de 20 m a una inclinación de 5° y con una operación de vuelo del dron de 30 s para observar su comportamiento, el trayecto analizado se observa en la siguiente figura:

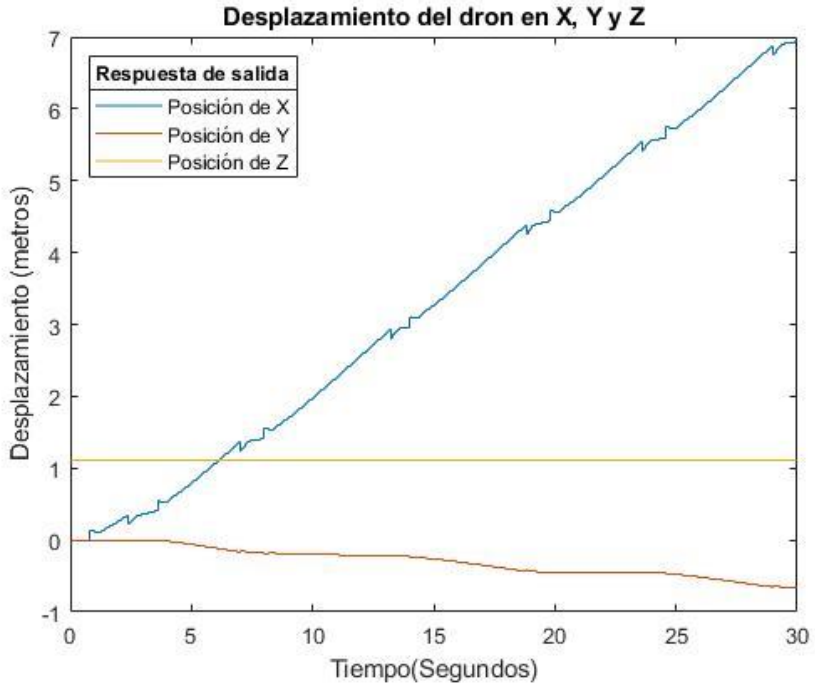
Figura 4-1: Trayecto de única sección con variación angular de 5°



La Figura 4-2 muestra los datos de salida de las señales X, Y y Z, estas señales son las ordenes que realiza la máquina de estados dependiendo del trayecto binarizado y segmentado, así mismo la señal de Z es constante con un valor de 1.1 m de elevación, ya que representa el desplazamiento realizado en dicha posición (según los ejes de coordenadas del dron, el movimiento de elevación se representa con un valor negativo, esta variable se multiplico por -1 para hacer una representación más acorde al movimiento).

Se observa que el desplazamiento de la posición del dron en X es continuamente creciente en el desplazamiento positivo y en Y es continuamente creciente en el desplazamiento negativo, esto indica que al no presentar una variación significativa el dron esta avanzando por una sola sección (recorre una sección recta del trayecto).

Figura 4-2: Señal generada de la máquina de estados en X, Y y Z. para la sección del trayecto con ángulo de 5°



Las perturbaciones presentes en la gráfica de X y Y se deben a un ligero cambio en el desplazamiento generado por la rotación del dron, como se observa en la Figura 4-3 en las gráficas de movimiento del desplazamiento en X y Y aproximadamente entre los 7.01 s y los 8.01 s en el valor de rotación se observa que del punto máximo (-0.03142 rad) y mínimo (-0.1861 rad) de la pendiente presenta una rotación del dron aproximada de 0.15468 rad, si transformamos este valor a grados tenemos que:

$$-0.03142 - (-0.1861) = 0.15468\, rad$$

$$0.15468\, rad = \frac{x° * \pi\, rad}{180°}$$

Despejando esta ecuación obtenemos:

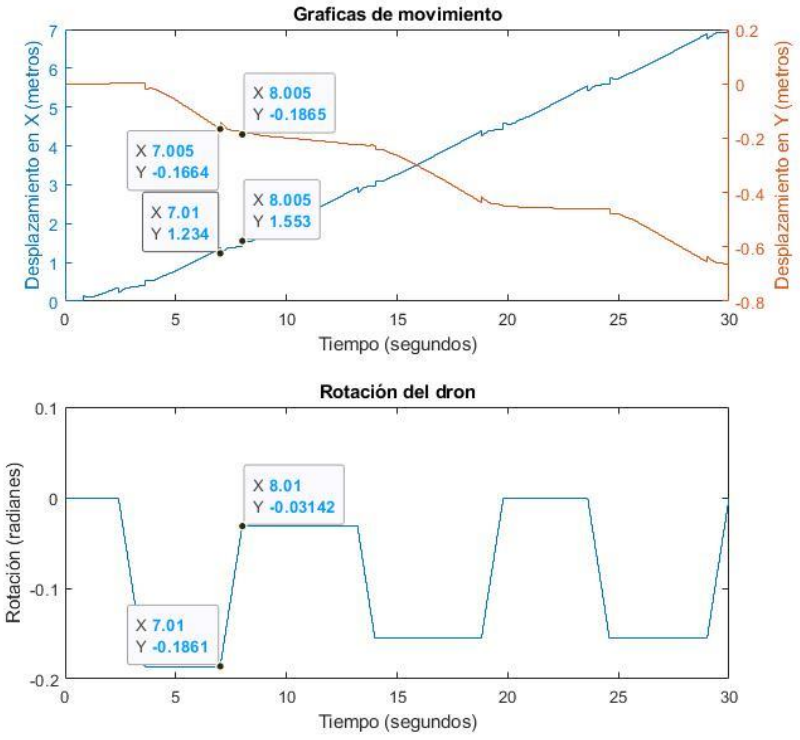
$$\frac{0.15468\ rad * 180°}{\pi\ rad} = x°$$

$$x° = 8.8625°$$

Aplicando el mismo proceso a cada una de las pendientes tenemos que P1=10.70°, P2=8.86°, P3=4.96°, P4=8.81°, P5=8.82°, P6=8.81°, realizando el promedio de todos los valores angulares de las pendientes tenemos que la precisión de redireccionamiento angular es de 8.49°.

Este valor es coherente ya que si el dron presenta un desplazamiento angular inicial este tiende realizará desplazamientos angulares cercanos al ángulo de inclinación de la trayectoria que en este caso es de 5°, en la gráfica se observan distintas pendientes con valores que tienden a 5°.

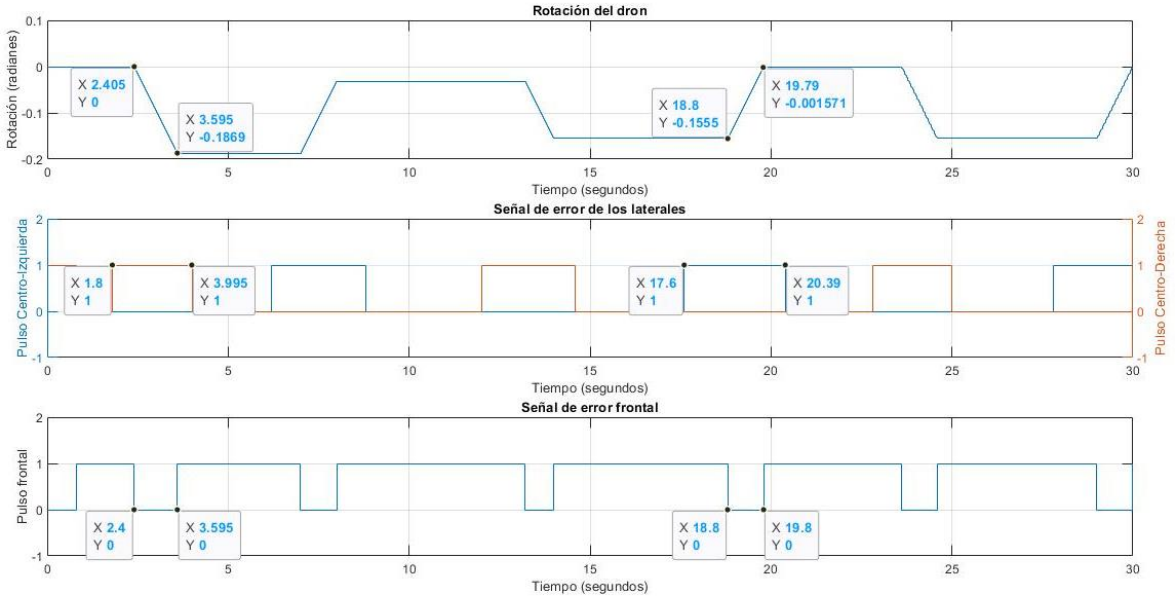
Figura 4-3: Perturbaciones del desplazamiento en los ejes X y Y debido al desplazamiento rotacional con ángulo de 5°.



El desplazamiento angular de tipo rotacional se genera a partir de las señales binarias (para este caso particular) de los sensores centro izquierda, centro derecha y del sensor frontal como se observa en la Figura 4-4, la gráfica superior de esta figura muestra la rotación la del dron mencionada anteriormente, la gráfica central presenta los pulsos de los sensores centro izquierda y centro derecha, y la gráfica inferior muestra el pulso del sensor frontal, estos pulsos indican el momento en el que se presenta un error y lo corrige al generar el giro dependiendo de los pulsos activos en determinado tiempo, un ejemplo de esto se evidencia

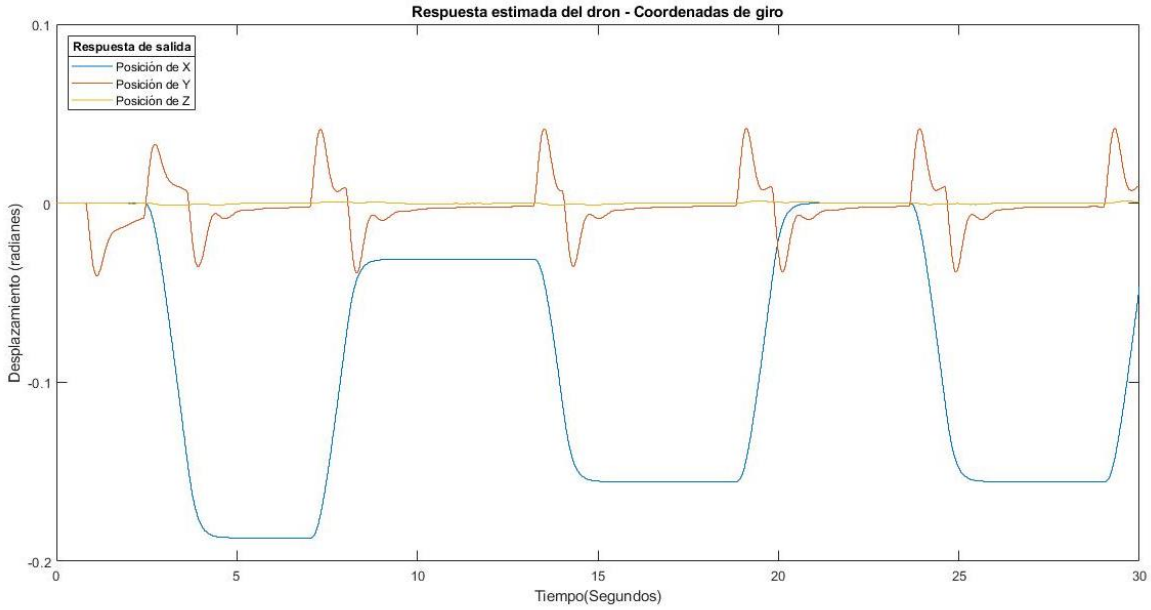
en los tiempos de 2.4 s a 3.4 s cuando el sensor frontal genera un pulso igual a 0 y el sensor del centro derecha genera un pulso de 1 en intervalos mayores o iguales al del sensor frontal, esto genera que la señal de rotación (Yaw) del dron presente una pendiente negativa (indica que el dron tiene que rotar), así mismo se presenta una situación similar en el tiempo de 18.8 s a 19.9 s con la diferencia de que la señal de rotación del dron se genera al recibir un pulso frontal igual a 0 y el cenzor de centro izquierda presenta un valor mayor o igual al intervalo del sensor frontal en ese periodo de tiempo generando en este caso una pendiente positiva.

Figura 4-4: Señal de rotación debido a pulsos emitidos por los sensores (ángulo de 5°)



En la Figura 4-5 se observan el desplazamiento final realizado por el dron, en el cual se aprecia las tres señales de movimiento angular las cuales son rotación (Yaw), inclinación (Pitch) y giro (Roll), de los tres el proceso que conlleva más movimiento es la rotación ya que visto desde el dron es el que permite el movimiento angular para avanzar en los ejes X y Y, el proceso que consiguiente es el de inclinación, movimiento generado cuando el dron se detiene y genera un pequeño retroceso antes de seguir con su operación y finalmente el proceso que presenta un movimiento casi nulo es el de giro el cual es un movimiento que no se emplea en el algoritmo.

Figura 4-5: Respuesta estimada del desplazamiento angular del dron (ángulo de 5°)



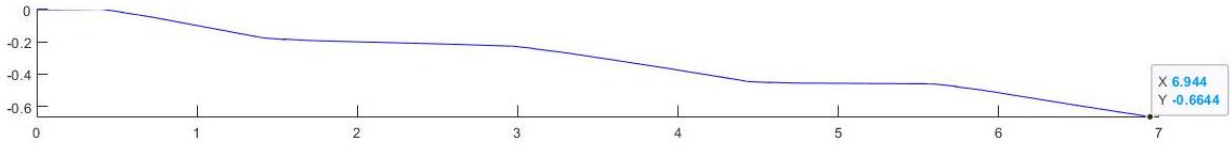
La distancia aproximada recorrida por el dron se puede obtener aplicando el teorema de Pitágoras al sistema de coordenadas presente al final de la sección del trayecto en este caso como solo se presenta una sección y las coordenadas de posicionamiento final del dron son (6.944, -0.6644), véase la figura 4-6, se puede obtener la siguiente información para recorridos o secciones de recorrido que presenten cierto grado de inclinación.

$$c = \sqrt{a^2 + b^2}$$

Donde c = recorrido; a = -0.6644; b=6.944, por lo tanto, tenemos que:

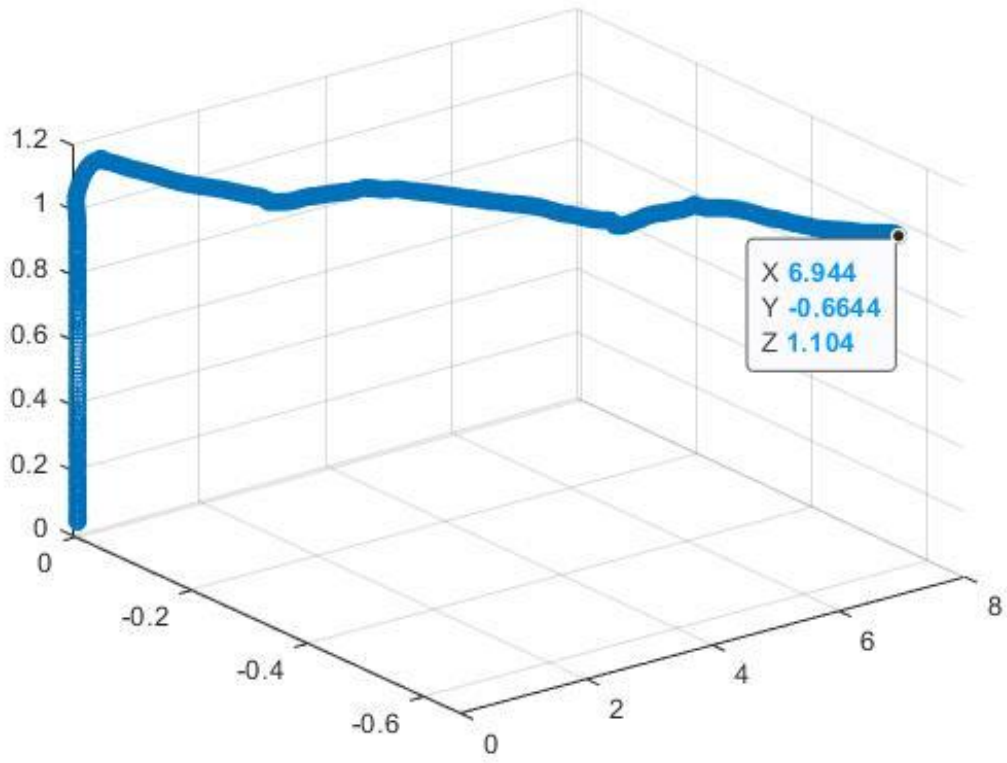
$$Recorrido = \sqrt{-0.6644\ m^2 + 6.944\ m^2} \quad ==> \quad Recorrido = 6.912\ m$$

Figura 4-6: Trayecto y las distancias recorridas en cada sección con (ángulo de 5°)



El resultado final de la operación de la máquina de estados se observa en la Figura 4-7 donde se muestra de forma gráfica el recorrido de la sección del trayecto analizado.

Figura 4-7: Visualización de la representación gráfica del movimiento realizado por el dron (ángulo de 5°)



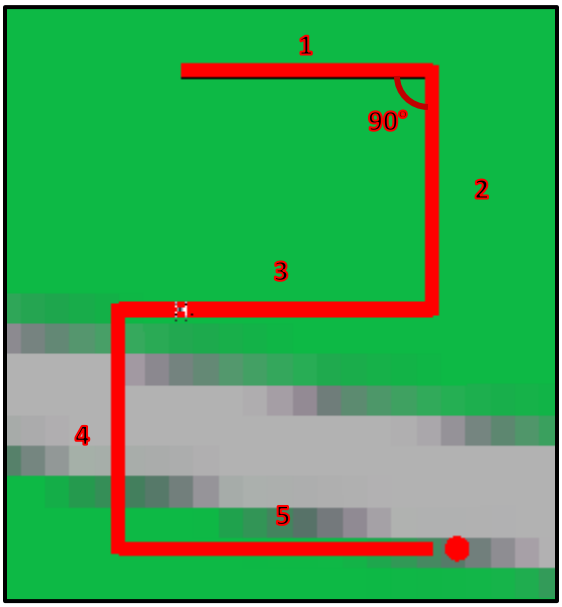
4.2 Análisis y resultados en 3 pistas diferentes con trayecto variable:

Teniendo en cuenta el análisis de la sección de 4.1 se realiza el mismo proceso de análisis a tres 3 pistas diferentes cuyo trayecto presenta variaciones específicas para cada caso, estas variaciones fueron seleccionadas con el fin de determinar el funcionamiento del diseño de cada sensor evaluando así su eficiencia en cada caso.

4.2.1 Trayecto con variaciones angulares de 90°

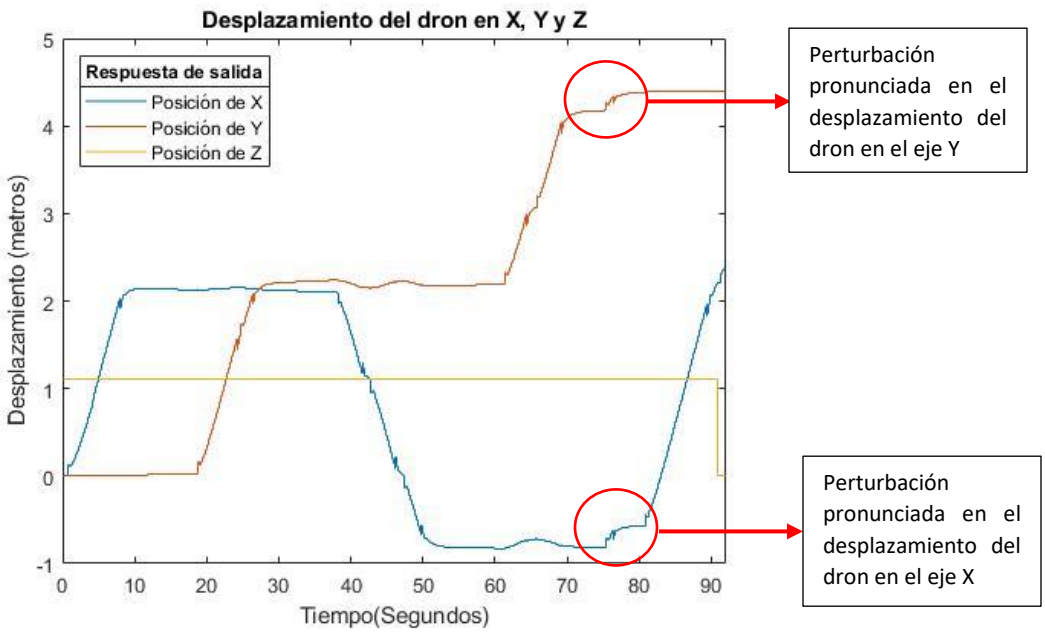
Se realizo el diseño de la primera pista con el recorrido que realiza el dron, véase la Figura 4-8 donde se muestra un recorrido que se divide en 5 secciones y presentan ángulos de 90°, el tiempo estimado del recorrido es de 90.86 s, este recorrido emplea todos los sensores que se diseñaron, destacando principalmente el uso del sensor frontal, sensor derecho y sensor izquierdo.

Figura 4-8: Trayecto de múltiples secciones con variación angular de 90°



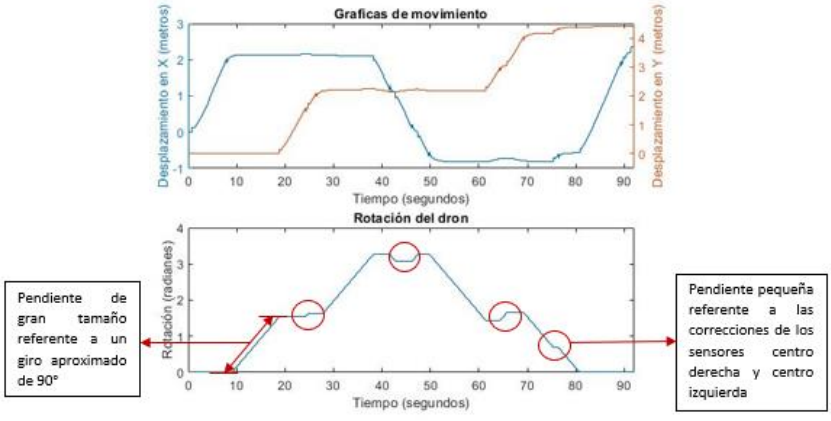
La Figura 4-9 presenta una notable perturbación de la señal del desplazamiento representado en los ejes X y Y visible de una forma más pronunciada en los tiempos de 70 s a 80 s, este tipo de perturbación fue causado por el ángulo de desplazamiento con el que se trasladaba el dron antes de llegar al ángulo de rotación, es decir, la posición relativa del dron no está completamente acoplado respecto a la posición del trayecto (véase la figura 3-17).

Figura 4-9: Señal generada de la máquina de estados en X, Y y Z. para la sección del trayecto con ángulos de 90°



Se puede observar en la Figura 4-10 la relación de las perturbaciones en las gráficas de desplazamiento X y Y con la rotación realizada por el dron, se observa que el trayecto al presentar ángulos de 90° genera pendientes de gran tamaño indicando que el valor de desplazamiento rotacional tiende a ese valor, adicionalmente se observa pequeñas pendientes ocasionadas por los sensores de centro derecha y centro izquierda que corrigen el curso del recorrido del dron cada pendiente representa una perturbación en esta grafica de movimiento. Aproximadamente el dron presento en este trayecto 9 perturbaciones visibles.

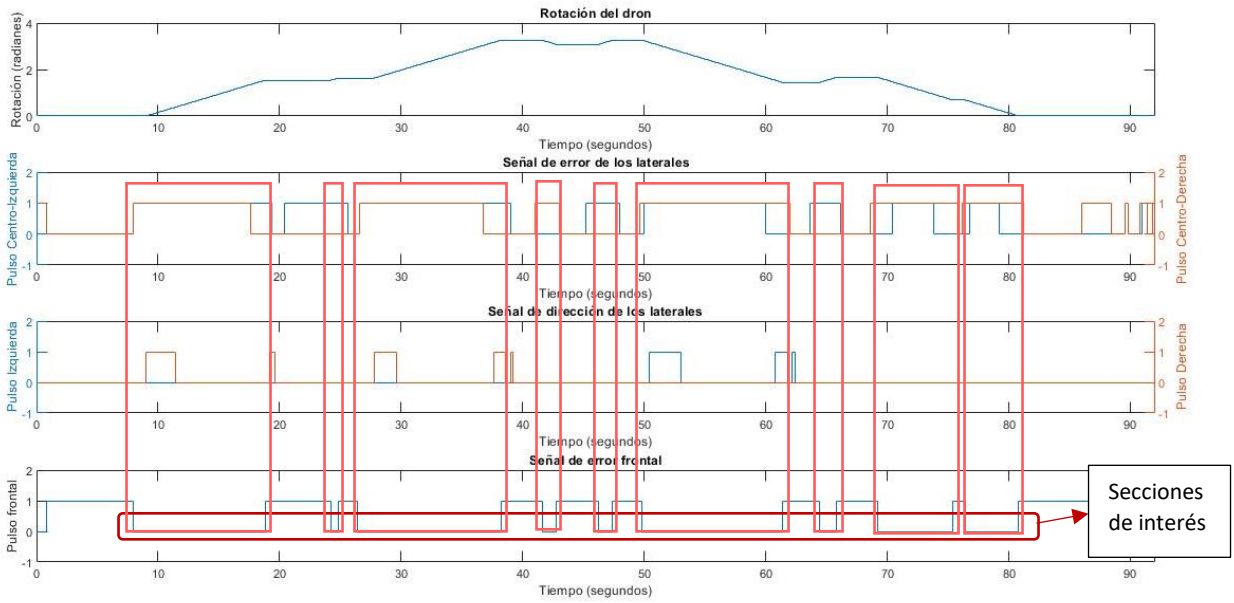
Figura 4-10: Perturbaciones del desplazamiento en los ejes X y Y debido al desplazamiento rotacional con ángulos de 90°.



Se observa que en la Figura 4-11 es posible visualizar el dato de la cantidad de pendientes revisando los datos de la señal de error frontal, donde el pulso frontal presenta 9 intervalos

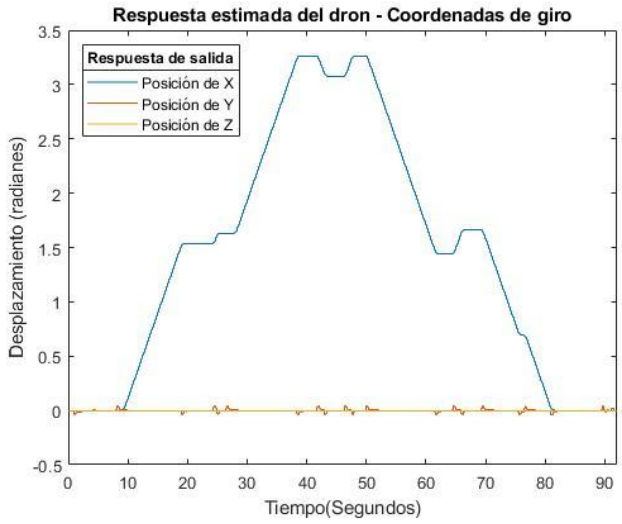
con valor de 0 que al ser comparado con los demás pulsos de los sensores de direccionamiento dan como resultado los desplazamientos que se pueden observar en la gráfica de rotación del dron.

Figura 4-11: Señal de rotación debido a pulsos emitidos por los sensores (ángulos de 90°)



En la figura 4-12 se evidencia que el desplazamiento final del dron es casi idéntico a la figura 4-10 observando que los movimientos angulares de giro e inclinación casi no presentan variaciones respecto al eje Y indicando que el dron casi no hace uso pronunciado de esos desplazamientos angulares.

Figura 4-12: Respuesta estimada del desplazamiento angular del dron (ángulos de 90°)



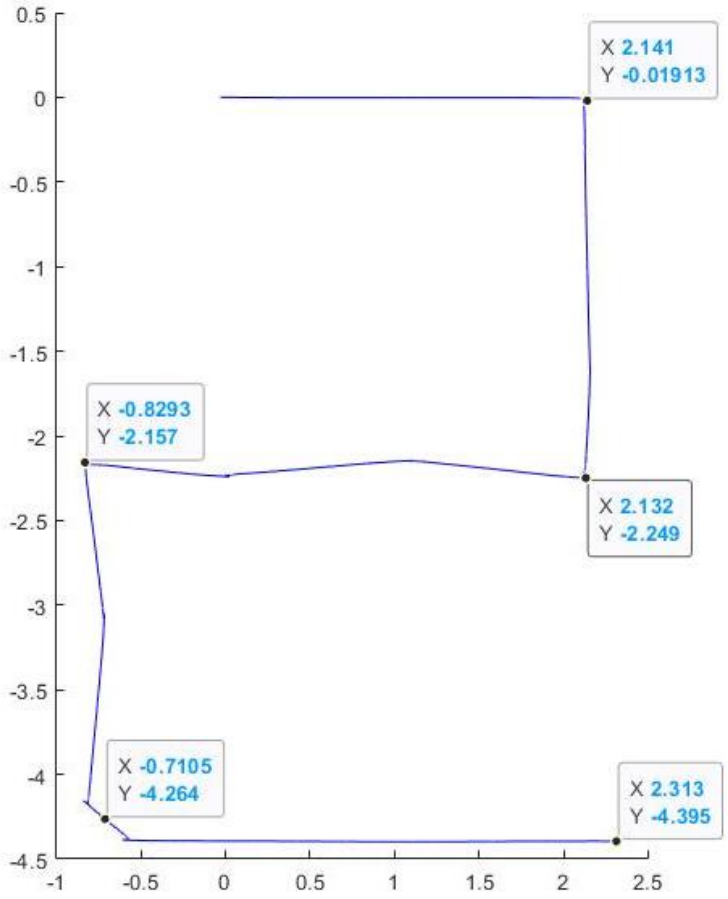
Teniendo en cuenta el análisis realizado en la sección 4.1 se puede deducir rápidamente que, de la figura 4-13 se pueden extraer la siguiente información:

$$Distancia\ aproximada\ recorrida = 2.141\ m + 2.249\ m + 2.961\ m + 2107\ m + 3.024\ m$$

$$Distancia\ aproximada\ recorrida = 10.375\ m$$

Ya que no presenta movimiento transversal (de forma pronunciada) en el plano simplemente se realiza la suma de las distancias para tener el valor aproximado.

Figura 4-13: Trayecto y las distancias recorridas en cada sección (ángulos de 90°)



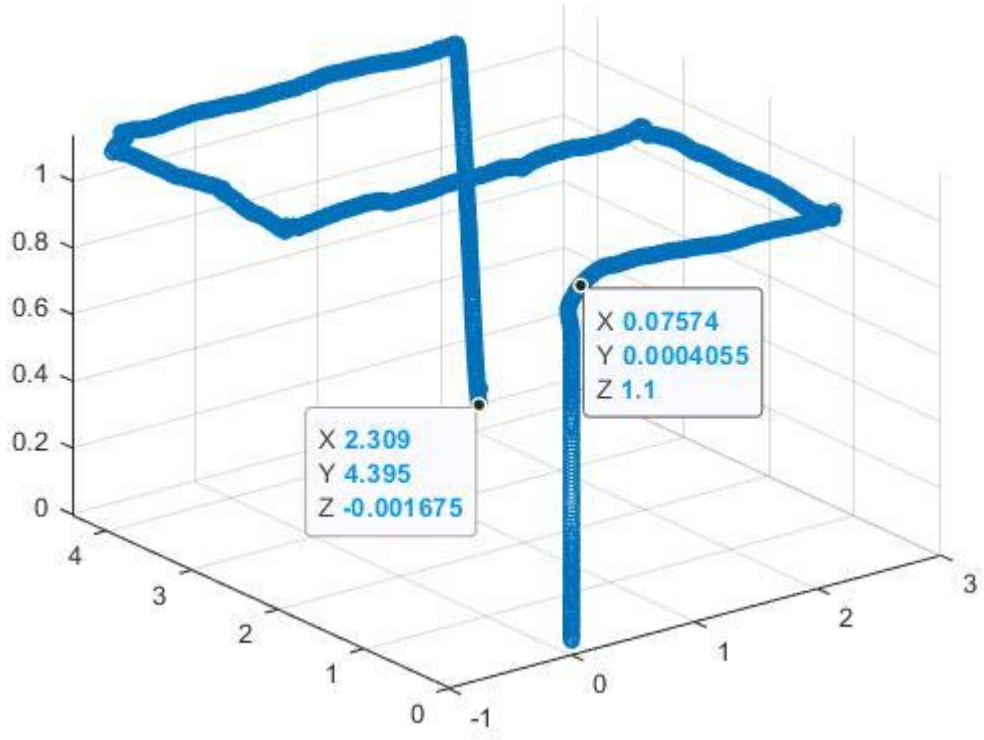
El resultado final de la operación en un trayecto con rotaciones angulares de 90° empleando la máquina de estados se observa en la Figura 4-7 donde se muestra de forma gráfica el recorrido del trayecto analizado que realizo el dron.

Como resultado de este proceso se obtuvo la siguiente tabla:

Tabla 4-1: Datos del análisis con desplazamientos angulares de 90°.

Datos	Valor
Tiempo	90.86 s
Distancia aproximada recorrida	10.375 m
Numero de secciones	5
Numero de perturbaciones en la señal	9

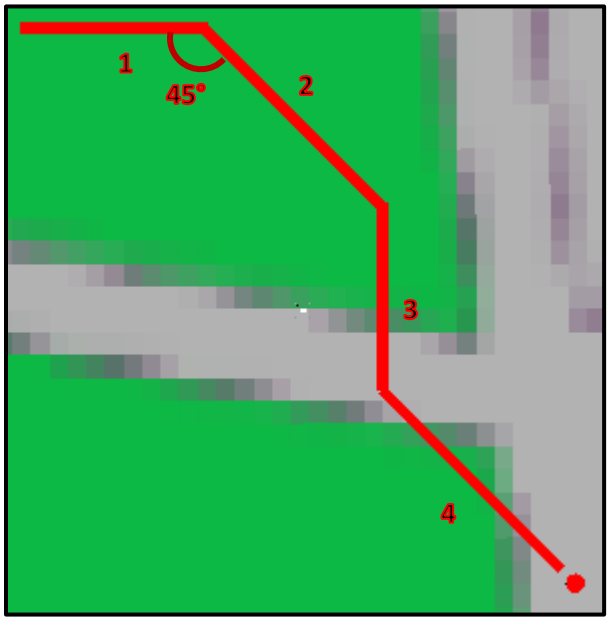
Figura 4-14: Visualización de la representación gráfica del movimiento realizado por el dron (ángulos de 90°)



4.2.2 Trayecto con variaciones angulares de 45°

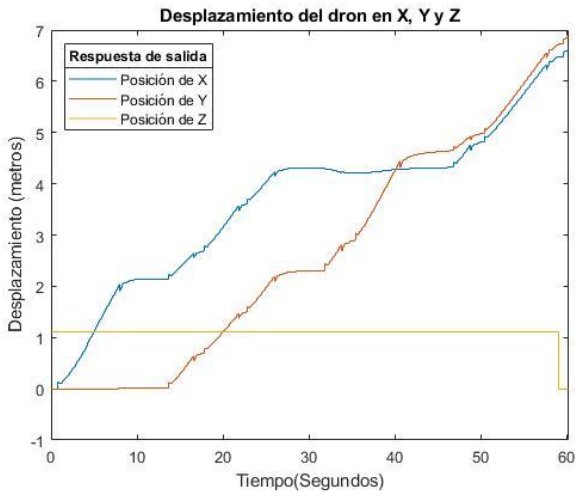
El diseño de la segunda pista, véase la Figura 4-15, muestra un recorrido que se divide en 4 secciones que presentan Angulos de 45°, el tiempo estimado del recorrido es de 60.01 s. Este recorrido emplea todos los sensores menos el sensor de la derecha y el sensor de la izquierda, ya que en ningún punto detectan el camino por la forma como se diseñaron.

Figura 4-15: Trayecto de múltiples secciones con variación angular de 45°



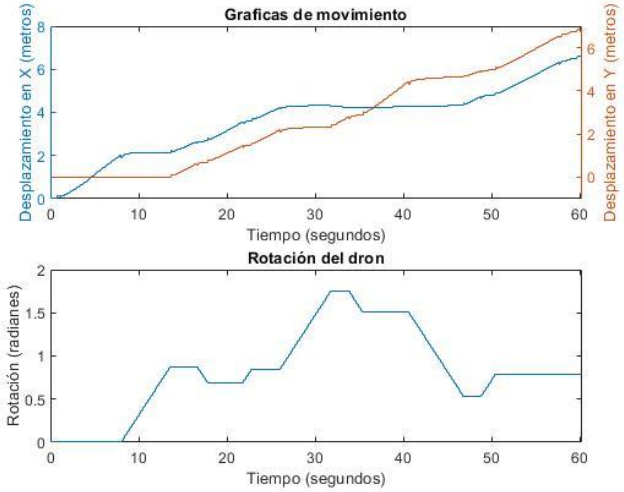
La Figura 4-16 presenta perturbaciones en X y Y, algunas perturbaciones son más notables en el desplazamiento de 2 m a 3 m y de 4.5 m a 5m, como se mencionó anteriormente este tipo de perturbación fue causado por el ángulo de desplazamiento con el que se desplazaba el dron antes de llegar al ángulo de rotación.

Figura 4-16: Señal generada de la máquina de estados en X, Y y Z. para la sección del trayecto con ángulos de 45°.



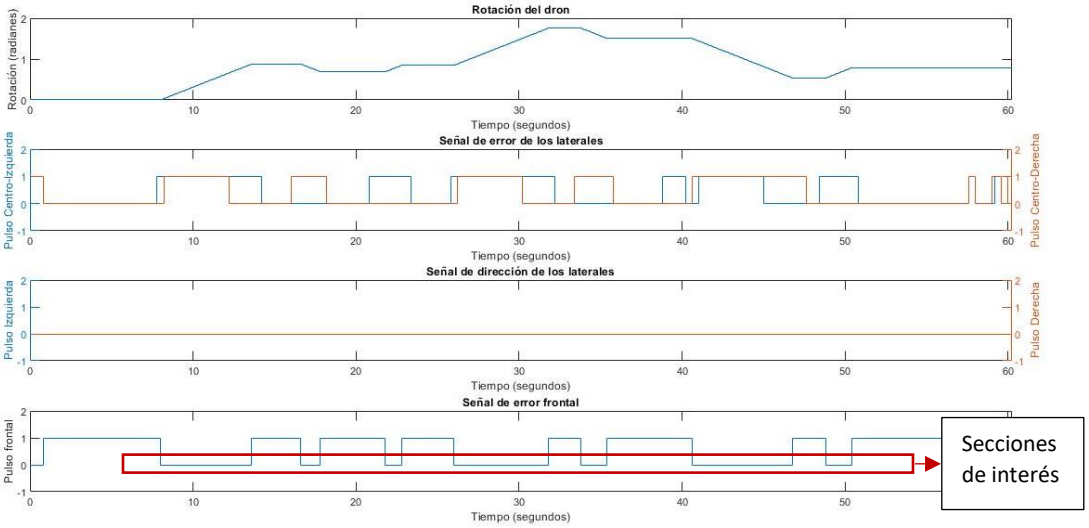
Se puede observar en la Figura 4-17 que, igual que en el anterior análisis, aunque no son tan pronunciadas las perturbaciones en las gráficas de desplazamiento X y Y se puede comparar con la gráfica de rotación del dron ya que cada pendiente presente indica que la rotación realizada genero perturbación en el desplazamiento. Aproximadamente el dron presento en este trayecto 7 perturbaciones visibles.

Figura 4-17: Perturbaciones del desplazamiento en los ejes X y Y debido al desplazamiento rotacional con ángulos de 45°.



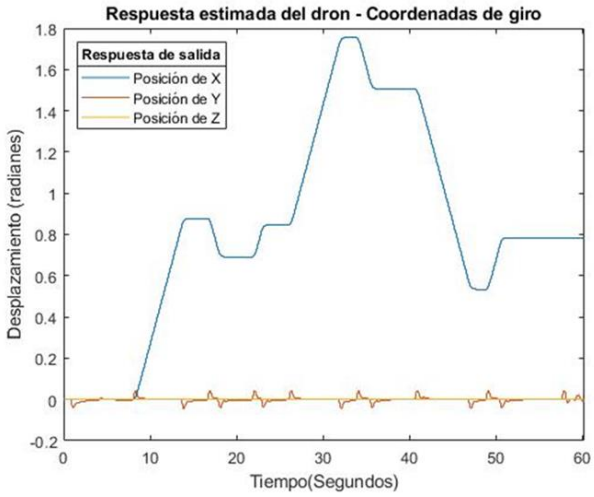
Se observa que en la Figura 4-18 se puede extraer la cantidad de pendientes de la rotación del dron visualizando la señal de error frontal, donde el pulso frontal presenta 7 intervalos con valor de 0 que al ser comparado con los demás pulsos de los sensores de direccionamiento dan como resultado los desplazamientos que se pueden observar en la gráfica de rotación del dron.

Figura 4-18: Señal de rotación debido a pulsos emitidos por los sensores (ángulos de 45°)



Se evidencia que los movimientos angulares de giro e inclinación casi no presentan variaciones respecto al eje Y indicando que el dron casi emplea este tipo de desplazamientos angulares, la señal de desplazamiento rotacional conserva muy bien el ángulo de rotación ya que se evidencia muy pocas interrupciones en el desplazamiento (véase Figura 4-19).

Figura 4-19: Respuesta estimada del desplazamiento angular del dron (ángulos de 45°)



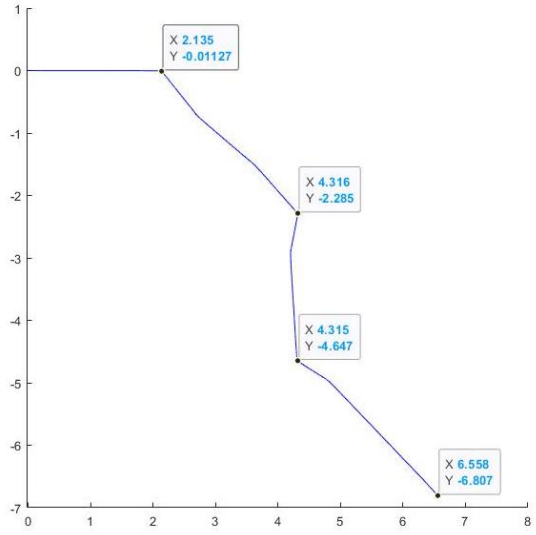
Ya que el trayecto presenta secciones rectas y trasversales simplemente se suma el recorrido de las secciones rectas con el proceso matemático para calcular la distancia de las secciones transversales mencionado en la sección 4.1.

De la figura 4-20 se pueden extraer la siguiente información:

$$Distancia\ aproximada\ recorrida = 2.135\ m + \sqrt{2.181\ m^2 + -2.285\ m^2} + 2.362\ m + \sqrt{2.243\ m^2 + -2.16\ m^2}$$

$$Distancia\ aproximada\ recorrida = 10.769\ m$$

Figura 4-20: Trayecto y las distancias recorridas en cada sección (ángulos de 45°)



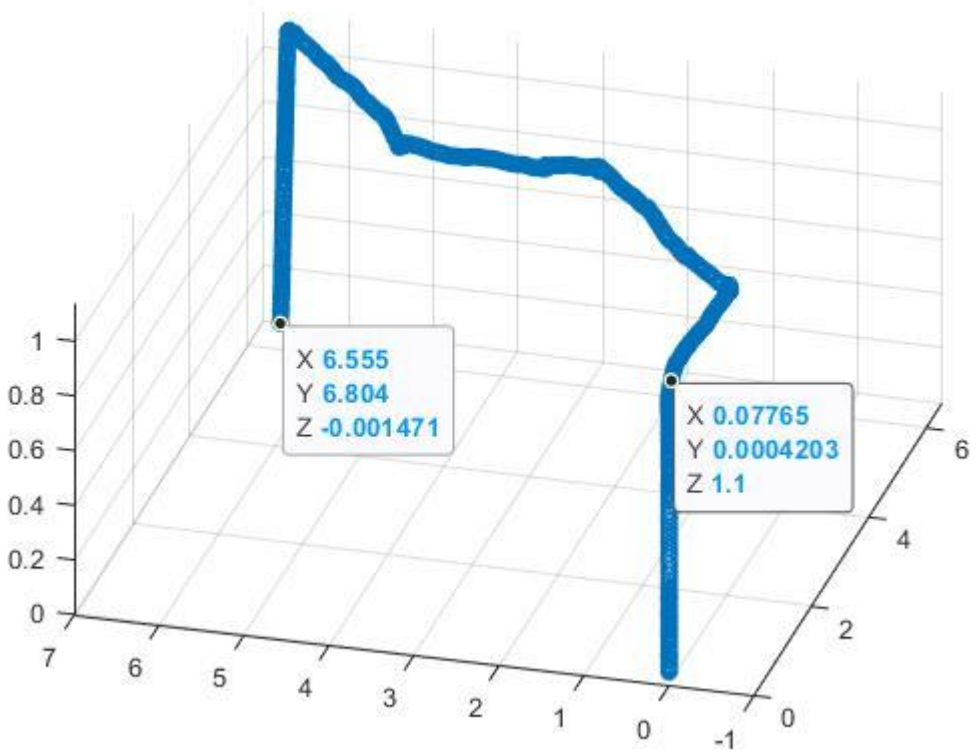
El resultado final de la operación en un trayecto con rotaciones angulares de 45° empleando la máquina de estados se observa en la Figura 4-21 donde se muestra de forma gráfica el recorrido del trayecto analizado.

Como resultado de este proceso se obtuvo la siguiente tabla:

Tabla 4-2: Datos del análisis con desplazamientos angulares de 45°.

Datos	Valor
Tiempo	60.01 s
Distancia aproximada recorrida	10.769 m
Numero de secciones	4
Numero de perturbaciones en la señal	7

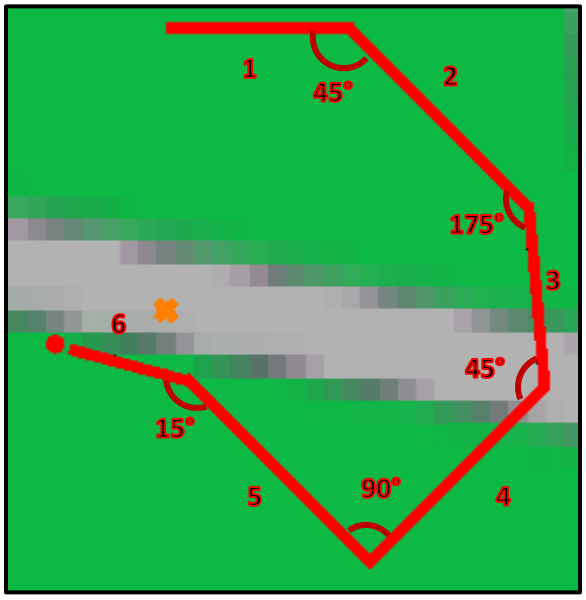
Figura 4-21: Visualización de la representación gráfica del movimiento realizado por el dron (ángulos de 45°)



4.2.3 Trayecto con variaciones angulares de diferentes grados y variación de la distancia de cada sección de trayecto.

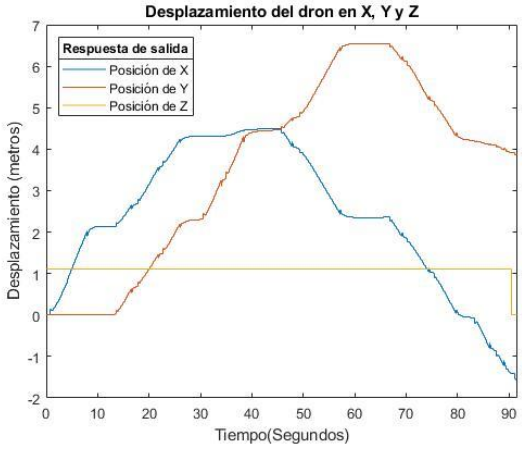
El diseño de la tercera pista, véase la Figura 4-22 muestra un recorrido que se divide en 6 secciones que presentan ángulos de rotación variados (90°, 45°,175°, 15°), el tiempo estimado del recorrido es de 90.23 s. Este recorrido emplea todos los sensores y muestra el rendimiento en un trayecto con dimensiones aleatorias.

Figura 4-22: Análisis de un trayecto con secciones de ángulos de rotación variados.



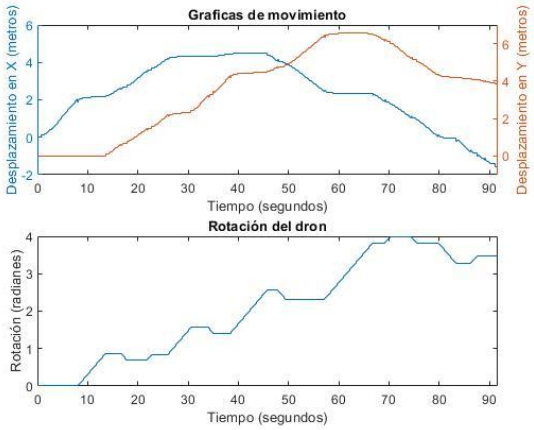
La Figura 4-23 presenta perturbaciones en X y Y, algunas perturbaciones son más notables en el desplazamiento de 2 m a 3 m y de 4.5 m a 5m, como se mencionó anteriormente este tipo de perturbación fue causado por el ángulo de desplazamiento con el que se desplazaba el dron antes de llegar al ángulo de rotación.

Figura 4-23: Señal generada de la máquina de estados en X, Y y Z. para la sección del trayecto con ángulos de variados.



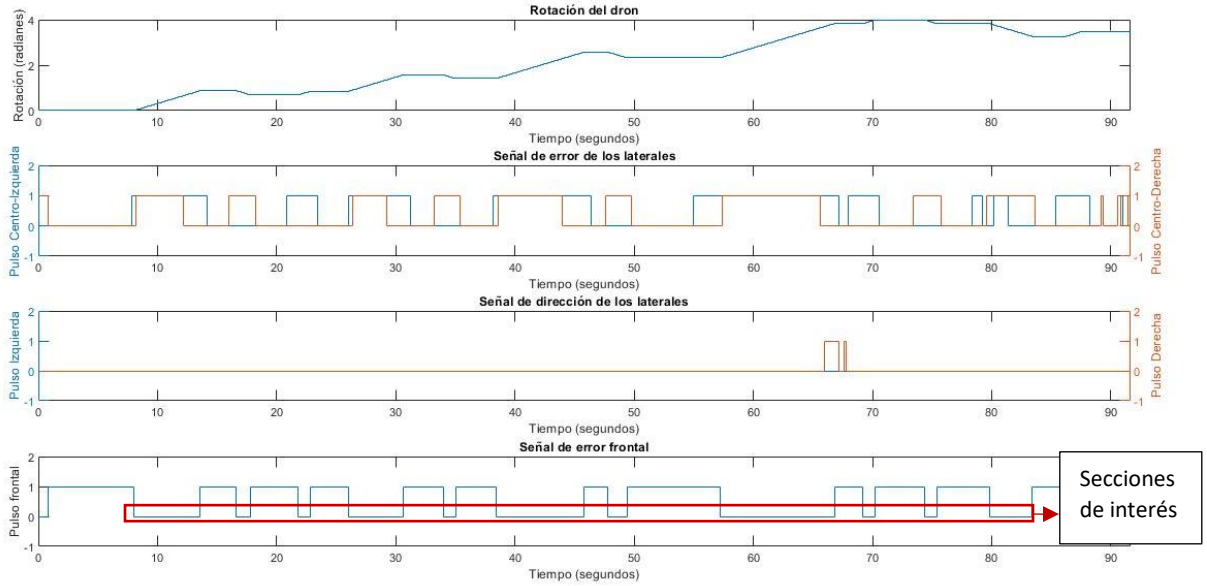
Se puede observar en la Figura 4-24 que, igual que en análisis anteriores se puede comparar con la gráfica de rotación del dron con la gráfica de movimiento del dron respecto al desplazamiento en X y Y, ya que cada pendiente presente indica que la rotación realizada genero perturbación en el desplazamiento. Aproximadamente el dron presento en este trayecto 12 perturbaciones visibles.

Figura 4-24: Perturbaciones del desplazamiento en los ejes X y Y debido al desplazamiento rotacional. (ángulos variados)



Se observa que en la Figura 4-25 se puede extraer la cantidad de pendientes de la rotación del dron visualizando la señal de error frontal, donde el pulso frontal presenta 12 intervalos con valor binario de 0 que al ser comparado con los demás pulsos de los sensores de direccionamiento dan como resultado los desplazamientos que se pueden observar en la gráfica de rotación del dron.

Figura 4-25: Señal de rotación debido a pulsos emitidos por los sensores (ángulos variados)



La señal de desplazamiento rotacional presenta 5 pendientes que en condiciones ideales no existirían, esto indica que el movimiento a través del trayecto presenta cierto nivel de error respecto a el desplazamiento que centra al dron. Se evidencia que los movimientos angulares de giro e inclinación casi no presentan variaciones respecto al eje Y indicando que el dron casi no emplea este tipo de desplazamientos angulares, (véase Figura 4-26).

Figura 4-26: Respuesta estimada del desplazamiento angular del dron (ángulos variados)

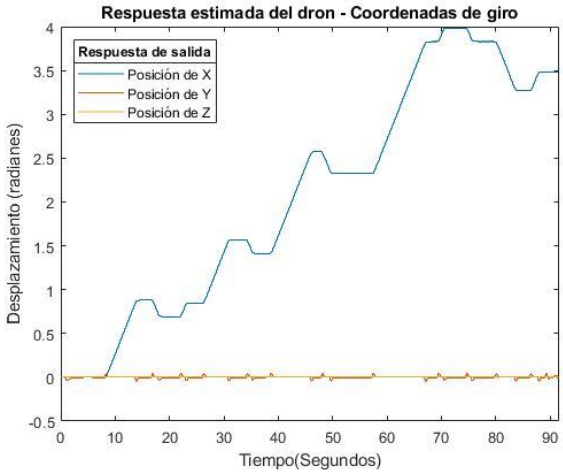


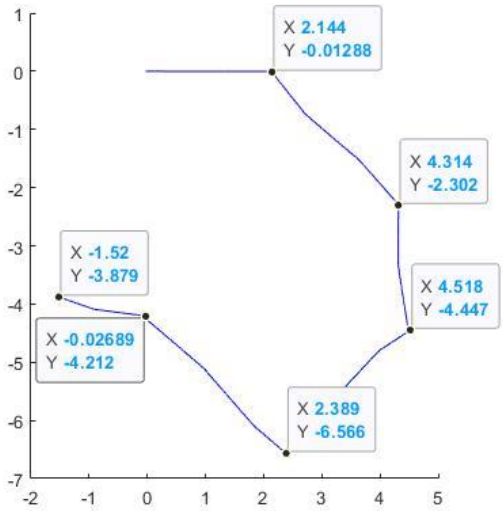
Figura 4-27: Trayecto y las distancias recorridas en cada sección (ángulos variados)

De la figura 4-27 se pueden extraer la siguiente información:

$$Distancia\ aproximada\ recorrida = \ 2.144\ m + \sqrt{-2.302\ m^2 + 2.17\ m^2} + 2.145\ m$$

$$+\sqrt{-2.119\ m^2 + -2.129\ m^2} + \sqrt{2.362\ m^2 + 2.354\ m^2} + \sqrt{2.155\ m^2 + -0.333\ m^2}$$

$$Distancia\ aproximada\ recorrida = \ 15.97\ m$$



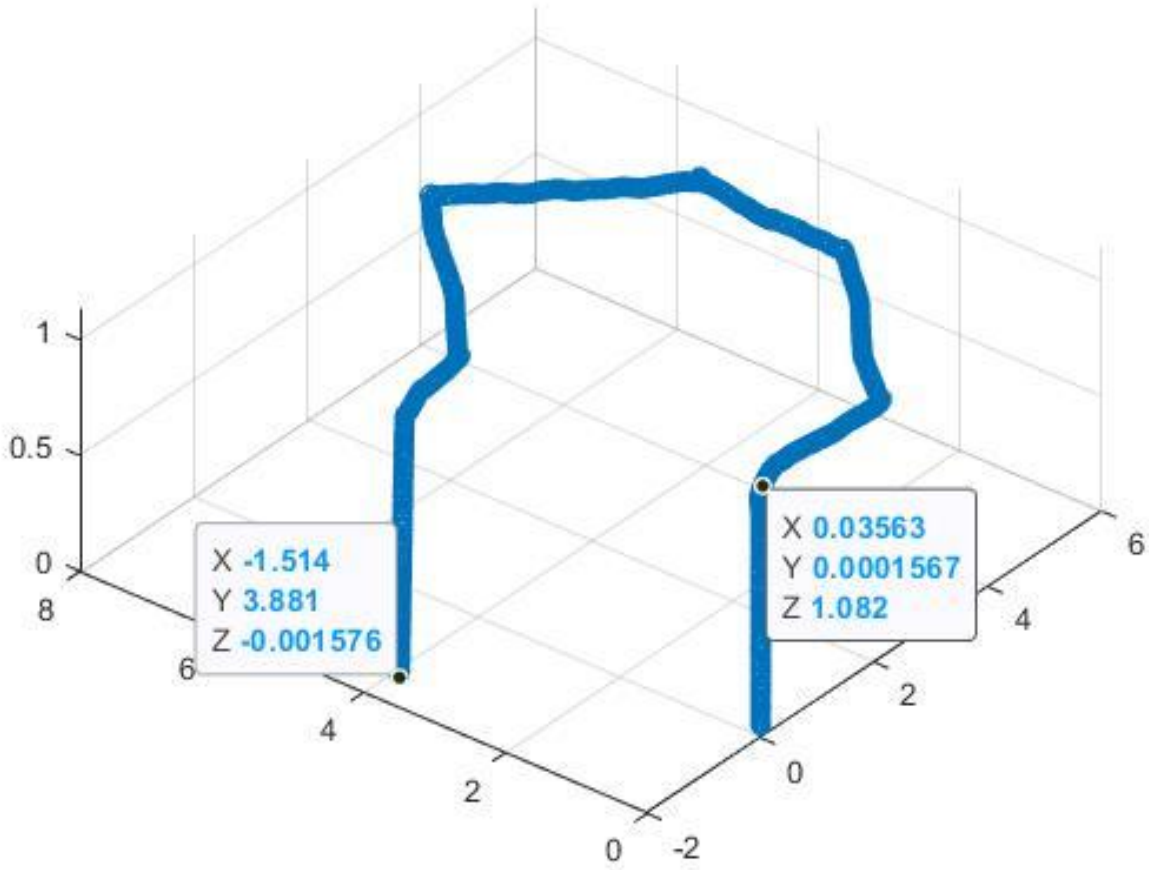
El resultado final de la operación en un trayecto con rotaciones angulares de 45° empleando la máquina de estados se observa en la Figura 4-21 donde se muestra de forma gráfica el recorrido del trayecto analizado.

Como resultado de este proceso se obtuvo la siguiente tabla:

Tabla 4-3: Datos del análisis con desplazamientos angulares variados.

Datos	Valor
Tiempo	90.23 s
Distancia aproximada recorrida	15.97 m
Numero de secciones	6
Numero de perturbaciones en la señal	12

Figura 4-28: Visualización de la representación gráfica del movimiento realizado por el dron (ángulos variados)



Con la Tabla 4-1, Tabla 4-2 y Tabla 4-3 obtenidas del análisis de las tres pistas se puede decir que:

Tabla 4-4: Funcionamiento promedio del algoritmo

	Tiempo	Distancia aproximada recorrida	Numero de secciones	Numero de perturbaciones en la señal
Tabla4-1	90.86 s	10.375 m	5	9
Tabla4-2	60.01 s	10.769 m	4	7
Tabla4-3	90.23 s	15.97 m	6	12
Promedio	80.36 s	12.37 m	5	9

Según la Tabla 4-4 Como resultado aproximado del funcionamiento del algoritmo para cualquier trayecto se tiene que idealmente para un trayecto de 12.37 m el tiempo requerido para realizar el recorrido será de aproximadamente 80.36 s y el número de perturbaciones en la señal será de 9 siempre y cuando se tenga un promedio de 5 secciones.

5 Conclusiones y recomendaciones

5.1 Conclusiones

Los resultados de las velocidades promedio de cada uno de los recorridos debería ser igual para todos los casos, sin embargo, solo se tuvo en cuenta el tiempo total (con el proceso de pausa en los giros), si se realiza el análisis correspondiente al tiempo que demora en dar un ángulo se puede establecer las pausas que realiza el dron dependiendo del giro y así obtener la velocidad promedio.

De acuerdo con los resultados demostrados en el capítulo 5, aunque en el análisis de las distintas pistas sean relativamente similar se pueden apreciar diferentes características como mayor o menor tiempo, distintos ángulos de rotación y diferentes distancias, aun así el algoritmo realizara el proceso de seguir la trayectoria hasta el punto final de forma correcta, demostrando así que el diseño cumple con los requisitos para seguir cualquier pista.

Con la información obtenida se puede establecer que el correcto proceso de los distintos parámetros puede mejorar significativamente la precisión de la obtención de los datos, un ejemplo de esto es el procesamiento de la imagen en el capítulo 3 sección 3.1 y 3.2, en la adquisición de datos binarios por medio de Color Theresholder se evidencia que ajustar mal los parámetros de color RGB genera pequeñas imprecisiones en la delimitación de la adquisición de datos, esto a su vez puede generar que el proceso de segmentación presente fallos y analice las secciones que no son de interés y en consecuencia los datos entregados al resto del algoritmo presenten fallos.

Para la obtención de la imagen fragmentada del capítulo 4.2 se puede destacar que dependiendo del tipo de fragmentación que se realice a la imagen se tendrá que realizar un análisis distinto equivalente a el tipo de sensores que se deseen utilizar, es decir que si se realiza una fragmentación de una cantidad x de cuadrantes todo el algoritmo cambiario por completo, en consecuencia, de esto se demostraría un resultado similar con un proceso totalmente diferente.

Teniendo en cuenta los resultados adquiridos de las diferentes pruebas, y teniendo presente que el archivo funcional que contiene el algoritmo de seguimiento de trayectorias para el minidrone parrot mambo posee un tamaño de 1 KB, se puede establecer que efectivamente el dron cuenta con la capacidad para ejecutar, y llevar a cabo el procesamiento del algoritmo

ya que según las características del dron cuenta con una capacidad de memoria de 1GB y un procesador interno programado por la empresa Parrot.

En la sección 4.1 y 4.3 se puede concluir que el diseño del diagrama de estados puede adquirir cualquier forma dependiendo de la cantidad de estados que se implementen, generando un resultado con más variables empleables que pueden ayudar con el movimiento.

En el desarrollo del capítulo 4.1 de la figura 4-3 se observó que el promedio de todas las pendientes era de 8.49°, este valor es coherente con el valor estimado de giro ya que el trayecto de ese ensayo, presenta una inclinación de 5°, suponiendo un trayecto infinito con un desplazamiento del dron infinito eventualmente la suma de todos las pendientes tendería a 5° ya que las correcciones que hace el dron se van ajustando más a medida que pasa el tiempo indicando que el dron coincide con las coordenadas del trayecto.

En el capitulo 4.2.1 observando la figura 4-11 se distingue el tiempo de duración de cada pulso, el análisis de esta pista muestra los tiempos de duración de cada señal emitida de los sensores que indican el momento exacto de giro que debe realizar el dron. Principalmente el sensor más importante es el sensor frontal, visualizando este sensor se observa que el primer pulso emitido presenta una duración de 9 s iniciando desde 9 s hasta finalizar a los 18 s, el tiempo de respuesta visto en la figura 4-19 (movimiento final del dron), indica que el tiempo que demoro en ejecutar la operación es idéntico con una duración del primer pulso de 9 s a 18 s, esto mismo pasa con las graficas de todas las pistas indicando así que la eficiencia del dron es bastante buena porque no presenta retraso en la ejecución de los comandos de la máquina de estados.

5.2 Recomendaciones

Se sugiere usar el programa en un ambiente que no genere mucho ruido en la imagen, si el trayecto es rojo, y el resto de la superficie es o tiene un color similar el algoritmo puede no detectar la imagen correctamente segmentada y consiguientemente puede generar que realice cualquier operación menos la deseada, también se propone el uso de distintos colores para los trayectos dependiendo del ambiente si no es posible cambiar el lugar de implementación, dicho esto el único proceso a realizar seria cambiar los valores del bloque *MATLAB function* correspondiente al procesamiento del color por medio de la app *Color Thresholder*.

Como se mencionó en el capítulo 4.2 emplear la misma transición como transición para todos los estados puede generar incoherencias particularmente en este caso el objetivo del sensor SP es detectar la variación de color programado siendo empleado para procesar la información administrada por la trayectoria, se recomienda implementar más sensores no necesariamente solo de la imagen para adquirir mucha más precisión, un ejemplo seria implementar un sensor que de la indicación de cuál fue el ángulo de giro, si es el correcto ángulo de giro continuar, si no aplicar correcciones y seguir con el siguiente proceso, así mismo se puede implementar otro sensor que dé más velocidad cuando realice los procesos de desplazamientos ya sea angular o de posicionamiento con más rapidez.

En trabajos futuros se recomienda la implementación de un sistema de control más robusto y precisos para que las operaciones sean mucho más eficientes, del mismo modo se recomienda implementar una I.A. (Inteligencia artificial), si se realiza un correcto procedimiento en la implementación del código se puede llegar a una generación que realice las ordenes de la forma más precisa posible para cualquier tipo de trayectoria sin importar el tipo de ángulo de rotación o distancia.

Se recomienda en trabajos futuros evaluar el algoritmo para terrenos no planos, que presenten rampas o inclinaciones con el objetivo de que generar un algoritmo capaz de transitar por cualquier recorrido.

Bibliografía

[1] T. de Swarte, O. Boufous, and P. Escalle, “Artificial intelligence, ethics and human values: the cases of military drones and companion robots,” *Artif. Life Robot.*, vol. 24, no. 3, pp. 291–296, 2019, doi: 10.1007/s10015-019-00525-1.

[2] H. Zimmermann, L. Vidiasova, I. Tensina, and C. Lee, *Electronic Governance and Open Society: Challenges in Eurasia*, vol. 947. 2019.

[3] V. Sgurev, V. Piuri, and V. Jotsov, *Learning Systems: From Theory to Practice*, vol. 756. 2018.

[4] “¿QUÉ TIPOS DE DRONES EXISTEN? - areadron.com.” https://www.areadron.com/que-tipos-de-drones-existen/ (accessed Oct. 13, 2020).

[5] “Support - Parrot Mambo Fly | Sitio Web Official de Parrot.” https://support.parrot.com/es/support/productos/parrot-mambo-fly (accessed Nov. 02, 2020).

[6] B. Demir *et al.*, “Real-time high-resolution omnidirectional imaging platform for drone detection and tracking,” *J. Real-Time Image Process.*, vol. 17, no. 5, pp. 1625–1635, 2020, doi: 10.1007/s11554-019-00921-7.

[7] R. Collins, X. Zhou, and S. K. Teh, “An Open Source Tracking Testbed and Evaluation Web Site,” *IEEE Int. Work. Perform. Eval. Track. Surveill.*, vol. 5, pp. 3769–3772, 2005.

[8] Y. G. Han, S. H. Jung, and O. Kwon, “How to utilize vegetation survey using drone image and image analysis software,” *J. Ecol. Environ.*, vol. 41, no. 1, pp. 1–6, 2017, doi: 10.1186/s41610-017-0035-2.

[9] J. Fleureau, Q. Galvane, F. L. Tariolle, and P. Guillotel, “Generic drone control platform for autonomous capture of cinema scenes,” *DroNet 2016 - Proc. 2nd Work. Micro Aer. Veh. Networks, Syst. Appl. Civ. Use, co-located with MobiSys 2016*, pp. 35–40, 2016, doi: 10.1145/2935620.2935622.

[10] T. Nageli, J. Alonso-Mora, A. Domahidi, D. Rus, and O. Hilliges, “Real-time motion planning for aerial videography with real-time with dynamic obstacle avoidance and viewpoint optimization,” *IEEE Robot. Autom. Lett.*, vol. 2, no. 3, pp. 1696–1703, 2017, doi: 10.1109/LRA.2017.2665693.

[11] A. Castillo, “Visual Control of an Autonomous Aerial Vehicle for Crop Inspection,”

vol. 33, no. 1, pp. 1–8, 2014.

[12] K. Alexis, C. Papachristos, R. Siegwart, and A. Tzes, “Robust Model Predictive Flight Control of Unmanned Rotorcrafts,” *J. Intell. Robot. Syst. Theory Appl.*, vol. 81, no. 3–4, pp. 443–469, 2016, doi: 10.1007/s10846-015-0238-7.

[13] “Statistics for labeled regions - Simulink - MathWorks América Latina.” https://la.mathworks.com/help/vision/ref/blobanalysis.html (accessed Nov. 02, 2020).

[14] “Drones para la seguridad en Bogotá | Bogota.gov.co.” https://bogota.gov.co/mi-ciudad/seguridad/drones-para-la-seguridad-en-bogota (accessed Nov. 02, 2020).

[15] “Seguridad y control de manifestaciones y eventos con drones | Embention.” https://www.embention.com/es/news/seguridad-y-control-de-manifestaciones-y-eventos-con-drones/ (accessed Nov. 02, 2020).

[16] “Review: Parrot Mambo Mini Quadcopter Drone with Camera | Best Buy Blog.” https://blog.bestbuy.ca/toys/drones/review-parrot-mambo-mini-quadcopter-drone-with-camera (accessed May 04, 2022).

[17] “▷ Funcionamiento de un Drone - Cuadricóptero.” https://www.muydrones.com/funcionamiento-del-cuadricoptero/ (accessed May 04, 2022).

[18] M. R. Kaplan, A. Eraslan, A. Beke, and T. Kumbasar, “Altitude and Position Control of Parrot Mambo Minidrone with PID and Fuzzy PID Controllers,” *ELECO 2019 - 11th Int. Conf. Electr. Electron. Eng.*, pp. 785–789, 2019, doi: 10.23919/ELECO47770.2019.8990445.

[19] S. Abdelmoeti and R. Carloni, “Robust control of UAVs using the parameter space approach,” *IEEE Int. Conf. Intell. Robot. Syst.*, vol. 2016-Novem, no. 600958, pp. 5632–5637, 2016, doi: 10.1109/IROS.2016.7759828.

[20] “Parrot Minidrones Support from Simulink - Hardware Support - MATLAB & Simulink.” https://la.mathworks.com/hardware-support/parrot-minidrones.html (accessed Nov. 02, 2020).

[21] “Stateflow - MATLAB & Simulink.” https://la.mathworks.com/products/stateflow.html (accessed Nov. 03, 2020).

[22] T. L. Floyd, *Digital Fundamentals with PLD Programming*. 2006.

[23] “Máquinas de Estado | LENGUAJE UNIFICADO DE MODELADO UML.” https://unadzsurlab.com/UML/U3/mquinas_de_estado.html (accessed May 02, 2022).

[24] J. Wakerly, “Diseño Digital,” *Angew. Chemie Int. Ed. 6(11), 951–952.*, pp. 5–24, 1967.

[25] “Modelizar máquinas de estados finitos - MATLAB & Simulink - MathWorks América Latina.” https://la.mathworks.com/help/stateflow/gs/finite-state-

machines.html (accessed May 02, 2022).

[26] “Overview of Mealy and Moore Machines - MATLAB & Simulink - MathWorks América Latina.” https://la.mathworks.com/help/stateflow/ug/overview-of-mealy-and-moore-machines.html (accessed May 02, 2022).

[27] “Diseño de apps - MATLAB & Simulink - MathWorks América Latina.” https://la.mathworks.com/help/stateflow/app-design-logic.html (accessed May 02, 2022).

[28] J. M. Blackledge, *Image Restoration and Reconstruction*. 2005.

[29] N. La Serna Palomino and U. Román Concha, “Técnicas de Segmentación en Procesamiento Digital de Imágenes,” *Rev. Ing. Sist. e Informática*, vol. 6, no. 2, pp. 9–16, 2009.

[30] P. F. I. N. D. E. Carrera, “Uso de ontologías para guiar el proceso de segmentación de imágenes,” 2007.

[31] P. I. Corke, “The machine vision toolbox: A MATLAB toolbox for vision and vision-based control,” *IEEE Robot. Autom. Mag.*, vol. 12, no. 4, pp. 16–25, 2005, doi: 10.1109/MRA.2005.1577021.

[32] “Umbrales de imágenes en color - MATLAB - MathWorks América Latina.” https://la.mathworks.com/help/images/ref/colorthresholder-app.html?searchHighlight=threshold&s_tid=srchtitle_threshold_2 (accessed May 04, 2022).

Apéndice A

Algunas características vitales para el funcionamiento del algoritmo:

Parrot mambo	Característica
Sensor de estabilidad	Acelerómetro de 3 ejes y giroscopio de 3 ejes
Sensor de ultrasonido	Estabilización vertical
Sensor de flujo óptico	Sistema de procesamiento de imágenes y un procesador preprogramado integrado (para estabilización horizontal)
Cámara	60 Fps

Tomado de [16]

Apéndice B

Script de verificación de las variables:

```
Imagen = imread('azul.bmp');

figure
subplot(1,2,1)
imshow(Imagen,[]);
title('Imagen original azul');
subplot(1,2,2)
imshow(maskedRGBImage)
title('imagen binarizada azul');
```

Apéndice C

Script de la función BW.

```
function BW  = createMask(R,G,B)

%createMask  Threshold RGB image using auto-generated code from
colorThresholder app.
%  [BW,MASKEDRGBIMAGE] = createMask(RGB) thresholds image RGB using
%  auto-generated code from the colorThresholder app. The colorspace and
%  range for each channel of the colorspace were set within the app. The
%  segmentation mask is returned in BW, and a composite of the mask and
%  original RGB images is returned in maskedRGBImage.

% Auto-generated by colorThresholder app on 02-Jun-2022
%------------------------------------------------------

% Define thresholds for channel 1 based on histogram settings
channel1Min = 250.000;
channel1Max = 255.000;

% Define thresholds for channel 2 based on histogram settings
channel2Min = 0.000;
channel2Max = 0.000;

% Define thresholds for channel 3 based on histogram settings
channel3Min = 0.000;
channel3Max = 0.000;

% Create mask based on chosen histogram thresholds
sliderBW = (R >= channel1Min ) & (R <= channel1Max) & ...
    (G >= channel2Min ) & (G <= channel2Max) & ...
    (B >= channel3Min ) & (B <= channel3Max);
BW = sliderBW;

end
```

Apéndice D

Script para la extracción de datos del ambiente de simulación Simulink a Workspace

```
 %Desplazamiento X,Y y Z  del dron
figure (1)
plot(SalidaX,'DisplayName','Desplazamiento en X');
hold on;
plot(SalidaY,'DisplayName','Desplazamiento en Y');
plot(SalidaZ*-1,'DisplayName','Desplazamiento en Z');
hold off;
title('Desplazamiento del dron en X, Y y Z')
xlabel('Tiempo(Segundos)')
ylabel('Desplazamiento (metros)')
lgd = legend ({'Posición de X','Posición de Y','Posición de
Z'},'Location','northwest')
title(lgd,'Respuesta de salida')
xlim([0 91.655])

%%Desplazamiento angular}
figure(2)
plot(SalidaYaw1,'DisplayName','Desplazamiento en X');
hold on;
plot(SalidaPitch1,'DisplayName','Desplazamiento en Y');
plot(SalidaRoll1,'DisplayName','Desplazamiento en Z');
hold off;
title('Respuesta estimada del dron - Coordenadas de giro')
xlabel('Tiempo(Segundos)')
ylabel('Desplazamiento (radianes)')
lgd = legend ({'Posición de X','Posición de Y','Posición de
Z'},'Location','northwest')
title(lgd,'Respuesta de salida')
xlim([0 91.655])


%%Grafica  rotacion respecto a las señales
figure(3)
subplot(4,1,1)
plot(SalidaYaw)
title('Rotación del dron')
xlabel('Tiempo (segundos)')
ylabel('Rotación (radianes)')
xlim([0 91.655])

subplot(4,1,2)
```

```
yyaxis right
plot(SalidaErrorDe);hold on;
title('Señal de error de los laterales')
xlabel('Tiempo (segundos)')
ylabel('Pulso Centro-Derecha')
xlim([0 91.655])
ylim([-1 2])
yyaxis left
plot(SalidaErrorIz);hold off;
ylabel('Pulso Centro-Izquierda')
xlim([0 91.655])
ylim([-1 2])

subplot(4,1,3)
yyaxis right
plot(SD);hold on;
title('Señal de dirección de los laterales')
xlabel('Tiempo (segundos)')
ylabel('Pulso Derecha')
xlim([0 91.655])
ylim([-1 2])
yyaxis left
plot(SI);hold off;
ylabel('Pulso Izquierda')
xlim([0 91.655])
ylim([-1 2])

subplot(4,1,4)
plot(SF)
title('Señal de error frontal')
xlabel('Tiempo (segundos)')
ylabel('Pulso frontal')
xlim([0 91.655])
ylim([-1 2])

%%Grafica del desplazamiento X y Y
figure(4)
yyaxis left
plot(SalidaX)
title('Graficas de movimiento')
xlabel('Tiempo (segundos)')
ylabel('Desplazamiento en X (metros)')
yyaxis right
plot(SalidaY)
title('Graficas de movimiento')
ylabel('Desplazamiento en Y (metros)')
xlabel('Tiempo (segundos)')
xlim([0 91.655])

%Grafica comparativa del desplazamiento respecto a el ángulo de giro
subplot(2,1,1)
yyaxis left
plot(SalidaX)
title('Graficas de movimiento')
xlabel('Tiempo (segundos)')
ylabel('Desplazamiento en X (metros)')
```

```
yyaxis right
plot(SalidaY)
title('Graficas de movimiento')
ylabel('Desplazamiento en Y (metros)')
xlabel('Tiempo (segundos)')
xlim([0 91.655])


subplot(2,1,2)
plot(SalidaYaw)
title('Rotación del dron')
xlabel('Tiempo (segundos)')
ylabel('Rotación (radianes)')
xlim([0 91.655])

%plot(SalidaX);hold on;
%plot(SalidaY);hold off;

%%Grafica del trayecto recorrido
figure('Color','w');mapshow(SalidaX1,SalidaY1*-
1,'DisplayType','line');shg;

%Representación grafica del movimiento realizado por el dron
figure(6)
scatter3(SalidaX1,SalidaY1,SalidaZ1*-1)
```