ESCUELA POLITÉCNICA NACIONAL

FACULTAD DE INGENIERÍA ELÉCTRICA Y ELECTRÓNICA

ESTUDIO, DESARROLLO E INTEGRACIÓN DE EQUIPOS AUXILIARES PARA EL CONTROL Y/O MONITOREO DE PLATAFORMAS ROBÓTICAS

DISEÑO E IMPLEMENTACIÓN DE UN PROTOTIPO AUXILIAR QUE ENVIÉ INFORMACIÓN A UN AUTOPILOTO PIXHAWK ENFOCADO A LA APLICACIÓN DE EVASIÓN DE OBSTÁCULOS

TRABAJO DE INTEGRACIÓN CURRICULAR PRESENTADO COMO REQUISITO PARA LA OBTENCIÓN DEL TÍTULO DE INGENIERO EN ELECTRÓNICA Y AUTOMATIZACIÓN

ERICK STEVEN LOYAGA CARRANZA

erick.loyaga@epn.edu.ec

DIRECTOR: ING. PATRICIO JAVIER CRUZ DÁVALOS, PhD.

patricio.cruz@epn.edu.ec

DMQ, octubre 2022

CERTIFICACIONES

Yo, Erick Steven Loyaga Carranza declaro que el trabajo de integración curricular aquí descrito es de mi autoría; que no ha sido previamente presentado para ningún grado o calificación profesional; y, que he consultado las referencias bibliográficas que se incluyen en este documento.

ERICK STEVEN LOYAGA CARRANZA

Certifico que el presente trabajo de integración curricular fue desarrollado por Erick Steven Loyaga Carranza, bajo mi supervisión.

Ing. PATRICIO JAVIER CRUZ DÁVALOS, PhD.

DIRECTOR

DECLARACIÓN DE AUTORÍA

A través de la presente declaración, afirmamos que el trabajo de integración curricular aquí descrito, así como el (los) producto(s) resultante(s) del mismo, son públicos y estarán a disposición de la comunidad a través del repositorio institucional de la Escuela Politécnica Nacional; sin embargo, la titularidad de los derechos patrimoniales nos corresponde a los autores que hemos contribuido en el desarrollo del presente trabajo; observando para el efecto las disposiciones establecidas por el órgano competente en propiedad intelectual, la normativa interna y demás normas.

Erick Steven Loyaga Carranza

Ing. Patricio Javier Cruz Dávalos, PhD.

Ing. Henry Paul Lema Ordóñez

Edwin Orlando Amaguaña Tabango

Luis David Ortega Toaquiza

AGRADECIMIENTO

Durante el trayecto de esto a lo que llamamos vida, he tenido la oportunidad de conocer a bastantes personas maravillosas, no obstante, quiero agradecer primeramente a Dios, porque he podido ver como él siempre ha estado conmigo en todo este trayecto, poniendo a todas estas buenas personas en mi camino, sé que sin eso no hubiera logrado nada. Quiero agradecer a mi madre porque fue ella quien me inculco la fe en Dios desde que era pequeño, quien me enseñó a nunca rendirme y quien durante gran parte de mi vida estuvo a mi lado dándome ánimos y su apoyo incondicional, definitivamente es una gran mujer.

Hoy en día, estoy a un paso de conseguir uno de los objetivos que me propuse obtener desde que era un niño, pero estoy cociente que esto no lo habría logrado sin la ayuda de aquellos que siempre confiaron en mí, principalmente sin la confianza y apoyo incondicional de Héctor y Wilma, a quienes les debo en gran parte, el haber conseguido este título universitario. Durante mi estadía con ellos he aprendido mucho, por eso mi agradecimiento infinito a ellos, a quienes considero como otros padres que me dio la vida. Pues bien, esta sección no estaría completa sino agradezco a todos aquellos que formaron parte de mi dentro de la EPN. A todos los Ingenieros que, con paciencia, me trasmitieron sus conocimientos, mi agradecimiento eterno al Doc. Patricio Cruz quien fue mi profesor en algunas materias y también quien me apoyo en el desarrollo de este trabajo de integración curricular, al grupo de investigación ATA de igual manera muchas gracias por el apoyo en este proyecto, sin duda alguna su ayuda fue vital para culminarlo. En esta parte también agradezco a los amigos, los hermanos que me dio la EPN, Andrés, Adrián, el grupo de relajoso “Los Brows”, Johita, Karlita, etc. Muchas gracias, chicos, hicieron que la estadía en la Poli sea más llevadera con sus ocurrencias, los llevare siempre en mi corazón a todos y espero que sigamos siendo amigos por toda la vida, los quiero mucho.

Hay aun una persona más que llego a mi vida casi al final de mi carrera universitaria, alguien que ha estado conmigo en todo momento dándome ánimos, apoyándome, siempre confiando en mí, esta persona me enseño que la vida es más bonita si se vive con amor, Kerly Maza, muchas gracias por hacer que mi vida sea más bonita cada día, sin duda alguna hoy en día eres una de las personas más valiosas para mí, te amo mucho y espero estes en mi vida siempre.

ÍNDICE DE CONTENIDO

RESUMEN

Un problema muy común al momento de realizar misiones con Vehículos Aéreos no tripulados (UAV) es que estos pueden sufrir colisiones con objetos presentes alrededor de su entorno, lo que puede provocar daños irreparables. Es por esto, que en este trabajo de integración curricular se diseña e implementa un sistema auxiliar de evasión de obstáculos para un UAV. Para ello, primero se presenta un estudio bibliográfico de diferentes sistemas de evasión comerciales y de los sensores que estos usan para la detección de objetos. Adicional, se presenta una breve descripción del controlador Pixhawk Cube, el protocolo MAVLink y de la técnica de visión artificial basada en operaciones morfológicas.

Este sistema auxiliar cuenta con una parte enfocada a detectar la posición y distancia a la que se encuentra el objeto a evadir, lo cual se hace a través del uso de la visión artificial y de un sensor LIDAR Lite. La segunda parte es en sí la maniobra de evasión guiada que lleva a cabo el UAV para impedir la colisión, para lo cual se usan diferentes mensajes de tipo comando que MAVLink ofrece, a través de la librería pymavlink. Para comprobar el funcionamiento del sistema auxiliar se diseñaron e imprimieron piezas 3D, sobre las cuales se montan los elementos que conforman el prototipo. Las pruebas realizadas demuestran que la evasión guiada resulto ser una maniobra estable para evitar colisiones.

PALABRAS CLAVE: Detección de Obstáculos, Evasión de Obstáculos, MAVLink, Pixhawk, Visión Artificial

ABSTRACT

A very common problem when performing missions with Unmanned Aerial Vehicles (UAV) is that they can suffer collisions with objects present around their environment, which can cause irreparable damage. Therefore, an obstacle avoidance auxiliary system for a UAV is designed and implemented in this work. For this purpose, first, a bibliographic study of different commercial avoidance systems and their sensors for object detection is presented. Additionally, we present a brief description of the Pixhawk Cube controller, the MAVLink protocol, and the artificial vision technique based on morphological operations.

This auxiliary system has a part focused on detecting the position and distance of the object to evade, which is done using artificial vision and a LIDAR Lite sensor. The second part is the guided evasion maneuver that is carried out by the UAV to avoid the collision. Different command messages provided by MAVLink through the pymavlink library are used to perform this maneuver. To test the operation of the auxiliary system it was necessary to design and print 3D parts, on which the prototype is mounted. The tests show the guided obstacle avoidance maneuver is stable and prevent potential collisions.

KEYWORDS: Obstacle Detection, Obstacle Avoidance, MAVLink, Pixhawk, Machine Vision

1 INTRODUCCIÓN

Debido a los avances tecnológicos en hardware de cómputo, y software de navegación, el uso de vehículos aéreos no tripulados (UAVs) ha crecido de manera considerable, especialmente en el desarrollo de misiones en la agricultura, medio ambiente, seguridad, entre otros. Es así que para el año 2025 se proyecta que su crecimiento alcanzará los 42.8 billones de dólares, según el reporte de Dronell [1]. Actualmente, y con la finalidad de optimizar el desempeño de los drones en misiones, es posible conseguir en el mercado una gran variedad de controladores de navegación. Un ejemplo, es la tarjeta de navegación autónoma Pixhawk que tuvo sus inicios en el año 2008 como un proyecto estudiantil en ETH Zurich. Este dispositivo hoy en día cuenta con un conjunto de sensores de posición, orientación, y aceleración, que facilitan el manejo y control de un dron mediante el uso de un software de piloto automático (ArduPilot o PX4) de manera satisfactoria [2].

Desde su aparición la tarjeta Pixhawk ha presentado diferentes actualizaciones, dando lugar así a la existencia de varios prototipos presentes en el mercado. En este trabajo se hace uso de la tarjeta de navegación Pixhawk Cube, la cual, debido a su tamaño, pocos cables de conexión y a que cuenta con un piloto automático de licencia abierta, es la más usada para fines comerciales; sin embargo, su sistema de navegación autónoma no cuenta con un sistema de evasión de obstáculos, lo que puede provocar que el dron sufra de un accidente que le ocasione daños irreparables en su hardware [3].

Para solucionar este inconveniente, el presente trabajo busca mejorar el sistema de pilotaje automático actual, implementando un equipo auxiliar de evasión de obstáculos autónomo. Actualmente existen equipos comerciales como Pensar [5] que suplen esta necesidad; sin embargo, su costo es muy elevado. No obstante, se puede cumplir este propósito usando un sistema de visión artificial conformado por una cámara mononuclear o estereoscópica, un sensor laser, y una tarjeta de alto procesamiento Jet Nano NVIDIA o una Raspberry Pi 4 [6][4]. En este proyecto se usa una Raspberry Pi 4, principalmente debido a que su costo en el mercado no resulta tan elevado como el de la tarjeta Jet Nano NVIDIA.

De modo que el componente resultante de este trabajo de integración curricular, hace uso de los sensores mencionados anteriormente (cámara y sensor laser) con el fin de detectar objetos, una etapa de procesamiento de señal que se llevará a cabo dentro de la Raspberry Pi 4, una etapa de control encargada de determinar los datos que serán enviados al autopiloto por medio del protocolo de comunicación Mavlink [7], y una etapa que

corresponde en sí a la maniobra de evasión de obstáculos que se llevara a cabo según los datos que le lleguen al Pixhawk Cube.

1.1 OBJETIVO GENERAL

Diseñar e implementar un prototipo auxiliar para un dron que envié información a un autopiloto Pixhawk Cube enfocado a la aplicación de evasión de obstáculos.

1.2 OBJETIVOS ESPECÍFICOS

- Realizar una revisión bibliográfica sobre los métodos de evasión de obstáculos en drones, sensores y sistema de procesado de señal que usan estos métodos y de la tarjeta Pixhawk Cube con su respectivo protocolo de comunicación MAVLink.
- Seleccionar los sensores que realizarán la detección de obstáculos y diseñar el hardware que conformará el equipo auxiliar de evasión de obstáculos, teniendo en cuenta que se usará como SBC una Raspberry Pi 4.
- Implementar e integrar a través de piezas creadas con impresión 3D, todos los elementos necesarios para el funcionamiento correcto del equipo sistema para la evasión de obstáculos, esto incluye la tarjeta Pixhawk Cube, sensores, Raspberry Pi 4, sistema de alimentación energética y cableado.
- Programar los algoritmos necesarios para la detección y evasión de obstáculos, y establecer la comunicación entre la Raspberry Pi 4 y el autopiloto Pixhawk Cube, a través del protocolo MAVLink.
- Analizar y comprobar el funcionamiento del sistema de detección y evasión de obstáculos implementado en el dron, tomando como referencia una misión creada desde el Mission Planner.

1.3 ALCANCE

- Se realiza una revisión bibliográfica sobre los diferentes métodos que se emplean para evadir obstáculos con drones, principalmente aquellos que usan la tarjeta Pixhawk Cube como autopiloto. Esto se realiza para conocer el funcionamiento de los sensores que se usan para llevar a cabo dicha maniobra, métodos del procesamiento de la señal adquirida por el sensor o sensores y comunicación entre el Pixhawk Cube y los sensores.

- Se estudia el funcionamiento de la tarjeta Pixhawk Cube, especialmente lo referente al protocolo de comunicación Mavlink y la estación a tierra Mission Planner.
- Una vez estudiados los métodos que se emplean para evadir obstáculos con drones, se selecciona el método más adecuado a implementar. Para ello se tiene en cuenta los sensores a usar, demanda de procesamiento, carga útil del dron, demanda de almacenamiento y costo económico.
- Se diseña el hardware electrónico para la implementación del sistema de evasión de obstáculos. El diseño se conforma de la tarjeta Pixhawk cube, sensor o sensores para detectar objetos, una tarjeta de alto procesamiento (Raspberry Pi 4), sistema de alimentación del circuito y cableado.
- Se diseñarán las piezas para el montaje de los elementos sobre el dron haciendo uso del software CAD, y se usa impresión 3D para su elaboración física.
- Se diseña e implementa el algoritmo de control para la evasión de obstáculos, así como también el algoritmo referente a la detección del obstáculo presentes en la trayectoria pre-programada desde la estación a tierra Mission Planner.
- Una vez terminada la etapa de diseño, se montan en el dron todos los elementos que componen al sistema de evasión de obstáculos haciendo uso de las piezas creadas con la ayuda de la impresión 3D.
- Se realizan las correspondientes conexiones de todos los elementos que componen el sistema de evasión de obstáculos, para posteriormente realizar la calibración de los sensores y el autopiloto que componen este sistema.
- Se comprueba el funcionamiento del sistema de evasión de obstáculos implementado y se analiza sus resultados.

1.4 MARCO TEÓRICO

1.4.1 SISTEMAS DE DETECCIÓN OBSTÁCULOS

Actualmente, los vehículos aéreos no tripulados (UAVs) o drones, como comúnmente se los conoce, cuentan con una gran oferta y comercialización que ha obligado a las empresas encargadas de fabricar estas aeronaves a desarrollar modelos más portátiles, sencillos de usar y de coste más económico. Esto a su vez a aumenta su campo de uso, haciendo que actualmente se los emplee en eventos como partidos de fútbol, para acceder a lugares peligrosos, en agricultura y en investigaciones biológicas.

Con el fin de conseguir que el uso de los UAVs sea más tecnificado, se ha empezado a desarrollar sistemas de piloto automático que permitan al dron realizar sus misiones sin necesidad de la interacción humana; sin embargo, hasta el momento no existe un autopiloto que cumpla al 100% con este objetivo. Empresas como DJI y Parrot han desarrollado su propio sistema de pilotaje que permite un vuelo casi autónomo del dron, no obstante, su uso es privado [9]. Por otro lado, existen proyectos de código abierto como lo es ArduPilot y PX4 que, si bien no son usados para desarrollar actividades de alta dificultad, permiten al fabricante adaptarlo a sus necesidades [2].

Por esto en el mercado existen sistemas de detección de objetos que pueden ser adaptados de manera sencilla, ya sea autopilos de código abierto como de uso privando en los cuales las mismas empresas propietarias de estos sistemas desarrollan sistemas de detección de objetos compatibles con sus autopilotos. El uso de estos sistemas comerciales es muy factible, sin embargo, su costo es muy elevado lo que limita su uso para la población en general.

1.4.1.1 Sistemas comerciales

Los sistemas comerciales de detección y evasión de obstáculos pueden sensar objetos de un tamaño considerable, por lo general están constituidos por más de un sensor, por lo que hacen uso de la fusión sensorial para combinar estas señales y así obtener resultados más precisos y confiables [4]. A continuación, se presentan dos de estos sistemas que se pueden encontrar a la venta y que son muy utilizados.

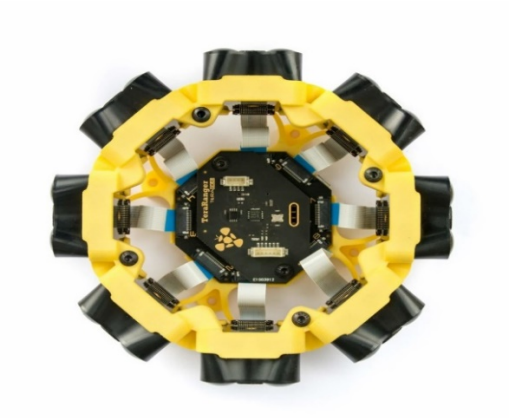


Figura 1.1. Sistema de detección y evasión de obstáculos TeraRanger Tower Evo [10]

TeraRanger Tower Evo: Es un sistema anticolisiones LIDAR de estado sólido de 360°, que gracias a que no cuenta con partes móviles tiene mayor robustez y funcionamiento silencioso. Este dispositivo es compatible con controladores de vuelo Pixhawk y se lo puede encontrar en dos configuraciones de hasta 8 módulos de sensores de detección tal

y como se observa en la Figura 1.1. [10]. Lo que destaca a este dispositivo es su sistema de anticolisiones interno que cuenta con localización y mapeo SLAM, por lo que no necesita de un sistema GPS para su correcto funcionamiento. Además, su tecnología infrarroja Time of Flight (TOF) permite su uso tanto para el día como para la noche y su costo no es tan elevado con respecto a la competencia, este bordea los 1000 dólares [10].

Pensar: Este sistema comercial anticolisiones está conformado de una cámara Sony 30x zoom HD visión, una cámara FLIR Boson que integra sensores infrarrojos y una GPU Jetson NVIDIA que le permita realizar un procesamiento de video en tiempo real basado en visión artificial. Con estos elementos y haciendo uso de las redes de aprendizaje profundo, Pensar es capaz de detectar, reconocer e incluso enmascarar objetos dependiendo las necesidades del usuario. Por esto puede ser usado para maniobras de evasión, inspección e identificación de aéreas de interés, y como sistema de búsqueda y rescate. En la Figura 1.3 se puede observar su presentación [11].

Figura 1.2. Sistema de visión por ordenador Pensar [11]

1.4.1.2 Sensores para detectar objetos

Existe una gran diversidad de sensores que pueden ser usados para detectar la presencia de objetos, sin embargo, la mayoría de ellos no brindan un resultado confiable al trabajar por si solos. Es por eso, que es recomendable usarlos en compañía de otros que complementen sus desventajas o para obtener redundancia de datos. A esta ciencia se le conoce como fusión sensorial y en Sección 1.4.1.4 se la explica a más detalle.

Sensor ultrasónico

Como su nombre lo indica estos tipos de sensores son capaces de medir la distancia que existe a un objeto a partir de ondas sonoras. Estos sensores están formados de un emisor que genera una señal ultrasónica, la cual, al chocar con un objeto, es reflejada y percibida por el receptor. La distancia entre el sensor y el objeto se mide teniendo en cuenta el tiempo que le tomo a la onda en ser receptada desde su emisión [11]. La principal ventaja que presentan estos sensores es la alta precisión con la que detectan obstáculos incluso en

condiciones adversas, ya que su funcionamiento no depende de la forma del objeto o de la luz solar. Sus precios no son tan elevados y son sencillos de usar, sin embargo, sólo son capaces de detectar correctamente objetos que están enfrente del sensor, por lo que su rango de medición en el plano es limitado [11].

Sensores Infrarrojos

Su principio de funcionamiento se basa en la medición de la radiación infrarroja emitida por los cuerpos que se encuentran dentro de su rango de visión. Pueden detectar objetos ubicados hasta 80 metros de distancia, lo que los hace muy llamativos para ser usado en UAVs que vuelan a grandes alturas. Por otro lado, al momento de usarlo hay que tener en cuenta que como el sensor ultrasónico solo es capaz de detectar objetos que se encuentran frente a él, y que como la luz del sol también emite señales infrarrojas son sensibles a la misma, por lo que la precisión de su medida depende del color y forma del objeto [11].

Como se puede observar tanto el sensor ultrasónico como el infrarrojo presentan muchos inconvenientes en su medida por lo que no son buenas opciones. No obstante, al trabajar conjuntamente estos sensores se vuelven en una buena tecnología para la detección de obstáculos, ya que el uno compensa las carencias el otro, reduciendo así de modo considerable el error de medida. Esta tecnología se suele aplicar utilizando un filtro de Kalman Extendido [11].

Laser Imaging Detection and Ranging (LIDAR)

Este sistema determina la distancia a una superficie u objeto, enviando un pequeño has laser pulsado que al chocar con un objeto es reflejado. Esta señal reflejada es capturada por una lente receptora para posteriormente ser enviada a una matriz de fotodetectores. El arreglo de fotodetectores segmenta cada señal en varias mediciones conformadas por hasta 20000 puntos por segundo, por lo que para el correcto funcionamiento del LIDAR se necesita de un equipo de alto procesamiento que funcione con gran rapidez [2], [11]. A diferencia de los sensores laser comunes, el LIDAR es capaz de medir el punto exacto en coordenadas XYZ en el que reboto el haz laser. Por lo tanto, este sensor es capaz de generar un mapeo 3D de toda la superficie que se encuentre dentro de su campo de visión, lo que lo convierte en una solución ideal para cumplir con el objetivo que tiene este trabajo de integración curricular [11].

Cámara

Este sensor es usado haciendo uso de la tecnología conocida como visión artificial. En el mercado se puede encontrar una gran diversidad de cámaras, pero generalmente estas se

dividen en dos grupos grandes, las de visión mononuclear que, como su nombre lo indica, cuentan con un único lente con el que captan la información del exterior, y las de visión estéreo que tienen dos lentes con el fin de simular con mayor precisión la visión humana [2]. La principal ventaja que brinda una cámara es el gran campo de visión con el que cuenta, por lo que pueden proporcionar una información excelente de lo que se encuentra en el entorno. No obstante, la técnica con la que se usa la cámara para detectar objetos, la visión artificial, necesita un alto procesado de señal que la convierte en una implementación costosa [11].

1.4.1.3 Visión artificial

La visión artificial es una parte de la inteligencia artificial, que tiene como finalidad simular la visión humana a través del uso de cámaras y un sistema de alto procesamiento [1]. Para esto, es necesario que la información adquirida del mundo real a través de la cámara sea procesada con la finalidad de obtener características como la forma, color, geometría, que ayuden al sistema a determinar de qué elementos está constituido el entorno captado y así se puede realizar la acción de control que se tenga como objetivo [13].

Procesamiento de imagen

Cuando se obtiene una imagen a través de una cámara, esta información entra a un computador en forma de una matriz de tamaño nxm pixeles, los cuales toman su valor en función de las intensidades del espectro de colores. Es por esto que es necesario usar herramientas matemáticas que permitan modificar estos números hasta obtener la información deseada. A esta manipulación de datos se le conoce como procesamiento de imagen [14].

El procesamiento de imagen va desde cosas sencillas como cambiar el tamaño de la imagen (resize), hasta cosas más complejas como identificar objetos o regiones de interés [13]. En este trabajo de integración curricular se pretende detectar la presencia de objetos dentro del campo de visión del UAV y de todos ellos escoger el más cercano, por lo que se necesita realizar operaciones que permitan separar este objeto del fondo de la imagen. Las operaciones que más se usan para cumplir con este propósito son las determinadas operaciones morfológicas [14].

Las operaciones morfológicas ayudan a obtener información como bordes, color, geometría, zonas convexas, entre otras, que facilitan posteriormente resaltar o eliminar características de una imagen. Estas operaciones actúan sobre un pequeño arreglo binario

que permite determinar la forma y el tamaño de la vecindad del pixel al cual se desea modificar su valor, un ejemplo de esto es la dilatación o llenado de huecos [14].

Como se puede apreciar la visión artificial es una buena técnica para detectar objetos, sin embargo, tiene una gran desventaja y es la pérdida de una de las dimensiones del mundo real, la profundidad. Por este motivo, para poder realizar un sistema de evasión de obstáculos, es necesario determinar este valor a través de triangular las medidas de dos cámaras o apoyarse en un sensor que pueda medir la distancia existente entre el dron y el objeto [1]. En este trabajo se ha optado por la segunda opción debido a que resulta más precisa y sencilla de programar.

1.4.1.4 Fusión sensorial

Como se dijo anteriormente la fusión sensorial es el método mediante el cual se combina señales provenientes de múltiples sensores con la finalidad de obtener resultados a los cuales un solo sensor no puede llegar o para obtener datos más confiables y precisos que los que se obtienen si se utilizan sensores por separado. Dentro de las diferentes formas en las que se puede catalogar una fusión sensorial existen dos que resaltan por ser las más usadas. Estas son:

- Complementaria: utilizada cuando se requiere obtener datos de un entorno en los cuales un solo sensor no puede adquirir la información necesaria para realizar la maniobra de control. De esta forma, los datos de sensores que toman la misma o diferente medida se combinan con la finalidad de cumplir con la meta propuesta.
- Cooperativa: combina la medida de varios sensores con la finalidad de obtener una nueva magnitud que es imposible conseguir usando los sensores por separado. Un ejemplo de esto seria las cámaras con visión estéreo, en las que se triangula la visión de ambos lentes con el fin de obtener la medida de profundidad.

Así como existe una forma de catalogar una red de fusión sensorial según su uso, también se lo puede hacer mediante su estructura, en base a cómo está conformada cada celda de fusión.

Estructura de redes de fusión sensorial

Toda red de fusión sensorial cuenta con celdas de fusión, las cuales son la parte más sencilla y pueden estar dentro de la red o después de los sensores adquiriendo la información que se usara en el resto del sistema. En la Figura 1.3. se puede observar que la información usada por la celda de fusión por lo general viene de tres fuentes, la primera es la información aportada por el sensor, la segunda conocida como información auxiliar,

que es aquella que se obtiene de procesa las medidas obtenidas con el sensor, y por último se tiene la llamada conocimiento de exterior la cual se conforma de los datos que el usuario ingresa y que son necesarios para obtener los resultados esperados [15].

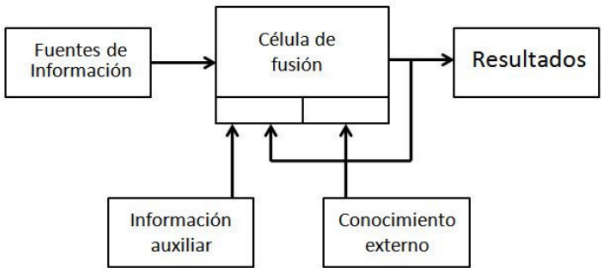


Figura 1.3. Estructura de una celda de fusión sensorial [15]

Las celdas de fusión pueden ser agrupadas en cuatro formas diferentes dando una estructura particular para cada red de fusión sensorial según el fin que se quiera obtener con ellas. En este trabajo solo se explica las estructuras que se usan para realizar el sistema de detección de obstáculos, pero en [15] se puede encontrar una explicación más detallada de estas cuatro estructuras.

- Redes paralelas: Cada sensor cuenta con una celda de fusión sensorial en la que se procesa toda la información y su resultado es enviado a una celda final encargada de dar los resultados finales [15].

- Redes iterativas: Son las más usadas ya que cuentan con un sistema de retroalimentación de la observación por lo que si se va a trabajar en entornos dinámicos es la más recomendable. Su uso sobresale en el seguimiento de objetos y la robótica móvil [15].

Como parte final del proceso de fusión, la observación obtenida por cada celda de fusión sensorial ingresa a una única celda que se encarga de unir y enviar todos los datos obtenidos al controlador. Por eso, debido a que el UAV con el que se cuenta para realizar este trabajo tiene como controlador de vuelo la tarjeta Pixhawk Cube 2.1 es necesario realizar un estudio de sus características principales, componentes y forma de comunicación con otros dispositivos externos.

1.4.2 PIXHAW CUBE 2.1

El Pixhawk Cube 2.1 conocido actualmente como piloto automático Cube, es una modificación del controlador Pixhawk FMUv3. Su diseño esta direccionado al uso comercial y de investigación, por eso cuenta con una conexión sencilla, de poco cableado y con un

tamaño bastante reducido. Como se pude observar en la Figura 1.4, la tarjeta cuenta de dos partes: la primera es un cubo metálico donde se encuentran tres sensores IMU para una mejor estabilidad, dos de ellos están aislados mecánicamente, lo que disminuye el efecto de vibraciones de alta frecuencia. Además, todas las unidades de gestión inercial (IMU) cuentan con control de temperatura mediante la integración de unas resistencias térmicas, lo que permite un óptimo funcionamiento de los sensores IMU. La segunda parte es una base rectangular en donde se encuentra la conexión a todas las interfaces con las que cuenta el Cube, y de un puerto DF17 de 80 pines, por donde pasan todas las entradas y salidas a las que se puede acceder en esta tarjeta [2][3].

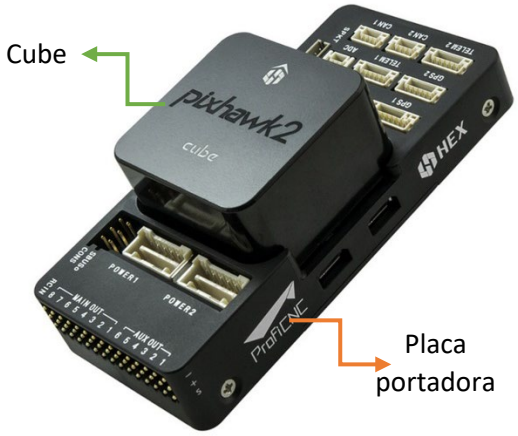


Figura 1.4. Estructura del controlador Cube [Fuente propia].

A continuación, se describen las especificaciones que son necesarias conocer para el uso adecuado del controlador de vuelo Cube.

- El controlador Cube viene con un módulo de energía power brick mini, el cual puede trabajar con una batería de hasta 8 celdas y una corriente máxima de 30 [A].
- Adicionalmente, el controlador cuenta con un GPS (here 2), que está constituido de un sensor IMU 1CM20948 y un barómetro MS561.
- Entre las interfaces con las que cuenta se disponen de 14 salidas servo/PWM, 3 entradas analógicas de 3.3 y 6.6 V, puertos CAN, I2C, telemetría, GPS y USB.
- Procesador ARM Cortex M4 STM32F427 de 32 bits, por lo que si se desea realizar procesos que demandan alto procesamiento es recomendable el uso de una tarjeta externa
- Cuenta con varias opciones de conectividad como: I2C, CAN, UART, telemetría [3].

Componentes del Pixhawk Cube 2.1

Al momento de adquirir el Pixhawk Cube 2.1 este viene incluido con algunos componentes externos que le permiten realizar tareas como controlar los motores, comunicación con la estación terrestre, ubicación global y alimentación. En la Tabla 1.1 se presenta estos elementos.

Tabla 1.1. Componentes externos que complementan al Pixhawk Cube 2.1

Componente	Figura	Función
Módulo Here 2		Se trata de un módulo GPS compuesto por un sensor IMU ICM20948, un barómetro MS5611 y de un procesador STM32F302 [2].
Power brick mini		Este elemento es el encargado de regular el voltaje y corriente que ingresa a la placa. Soporta baterías de 8 celdas y una corriente de hasta 30 A [2].
RFD900		Es un transceptor el cual como su nombre lo indica, permite entablar comunicación entre el UAV y la estación terrestre. Como se visualiza a pesar de tener un tamaño reducido cuenta con un alcance de comunicación mayor a 40km, por lo que su uso es muy amplio para fines comerciales [19].
Flysky FS-IA10		Se trata del receptor de 10 canales, cuya misión es la de recibir las señales enviadas desde el radio control [20].

1.4.2.1 Autopiloto

El autopiloto, también conocido como pila de navegación, es el software que se encarga de controlar un UAV. Actualmente, el PX4 y el ArduPilot son los dos pilotos automáticos más utilizados a nivel comercial, esto debido a que al ser de código abierto pueden ser adaptados a las necesidades del usuario, y a la diversidad de sistemas operativos con los que son compatibles como Windows, Linux, MacOS, NuttX [2]. El controlador de vuelo Cube permite la ejecución tanto del PX4 como de ArduPilot [3].

En el presente trabajo de integración curricular se usa el autopiloto ArduPilot, debido a que la estación a tierra con la que se trabaja es Mission Planner y esta se encuentra adaptada para trabajar con ArduPilot [16].

Autopiloto ArduPilot

Es un software de código abierto compatible con diversos tipos de vehículos como rovers, aeronaves de ala fija, submarinos, helicópteros y rastreadores de antenas. Cuenta con un código fuente desarrollado en C++ y Python para cada tipo de vehículo con el que es compatible, lo que facilita significativamente su uso. En la Figura 1.5 se observa la estructura interna de ArduPilot, la cual, para facilitar su entendimiento, se encuentra dividida en tres etapas: el firmware del vehículo, las bibliotecas y en la parte inferior la capa de conectividad del hardware externo [2].

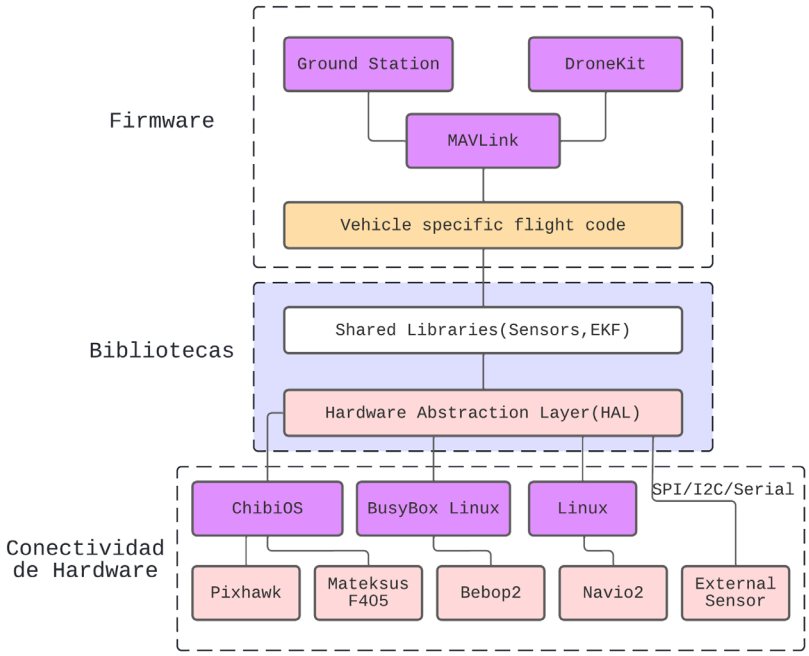


Figura 1.5. Estructura del piloto automático ArduPilot [Fuente propia].

Como se observa en la Figura 1.5., en la parte superior de la estructura interna del autopiloto se encuentra el código del vehículo, el cual es un directorio de alto nivel que define el firmware dependiendo del vehículo a emplear. En la segunda sección están las bibliotecas, las cuales incluyen controladores, estimadores de altitud y posición y el código de control, además de la capa AP_HAL, la cual se encarga de hacer que ArduPilot sea portátil y compatible con diferentes plataformas. Por último, en la parte inferior se tiene la capa de atracción del hardware, la cual permite cargar al autopiloto de funciones adicionales externas [17].

1.4.2.2 MAVLink

MAVLink es un protocolo de mensajería comúnmente usado para entablar comunicación entre un vehículo no tripulado y la estación terrestre. Este protocolo permite enviar y recibir datos, configurar parámetros y controlar el vehículo no tripulado desde la estación a tierra. Todo esto a partir de la definición de archivos XML, los cuales definen una serie de mensajes y microservicios admitidos por el sistema MAVLink [7].

En la Figura 1.6. se observa como todos los mensajes recogidos en los ficheros XML son estructurados y empaquetados en un formato de mensaje de 8 a 263 bytes. Los generadores de código usan los archivos XML para crear bibliotecas de software en diferentes lenguajes de programación, por lo que MAVLink es compatible con un gran número de lenguajes de programación. En la Tabla 1.2 se describe a más detalle la estructura del formato de mensajes que usa MAVLink [8].

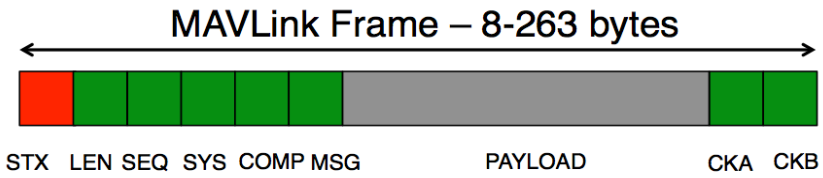


Figura 1.6. Estructura del paquete MAVLink [8].

Tabla 1.2. Estructura del formato de mensajería de MAVLink.

Contenido	Byte	Explicación
Start-of-frame	0	Indica el inicio de la trasmisión (v1.0: 0xFE)
Payload length	1	Longitud de la información o datos que contiene cada paquete
Packet sequence	2	Cada componente cuenta con su secuencia de envió. Permite detectar la pérdida de un mensaje
System ID	3	El identificador del sistema de envíos permite diferenciar diferentes mensajes en la misma red
Component ID	4	Permite diferenciar entre varios componentes en el mismo sistema
Message ID	5	Decodifica correctamente la información obtenida
Data	6 to (n+6)	Los datos del mensaje
CRC	(n+7) to (n+8)	Checksum del paquete. Evita errores y corrupción de datos

1.4.2.3 Mission Planner

Como se mencionó en la Sección 1.4.2.1, Mission Planner es una plataforma de estación a tierra desarrollada explícitamente para trabajar con el autopiloto ArduPilot [16]. Esta plataforma compatible únicamente con el sistema operativo Windows se utiliza como

planificador de vuelo, ya que permite tener un control dinámico más robusto del autopiloto que controla el vuelo del UAV. Entre las actividades que se pueden realizar con Mission Planner están:

- Cargar el software de control del vehículo (firmware) sobre la placa de piloto automático [16].
- Planificar y ejecutar la misión que se desea ejecute el UAV de manera autónoma. Esto se realiza mediante el ingreso de puntos de ruta (waypoints) sobre el mapa con el que cuenta la estación terrestre [16].
- Configurar características como velocidad de vuelo, ángulos de dirección, altura de vuelo y modo de vuelo [16].
- Visualizar el estado de vuelo en tiempo real mediante una interfaz y el sistema de posicionamiento global (GPS).

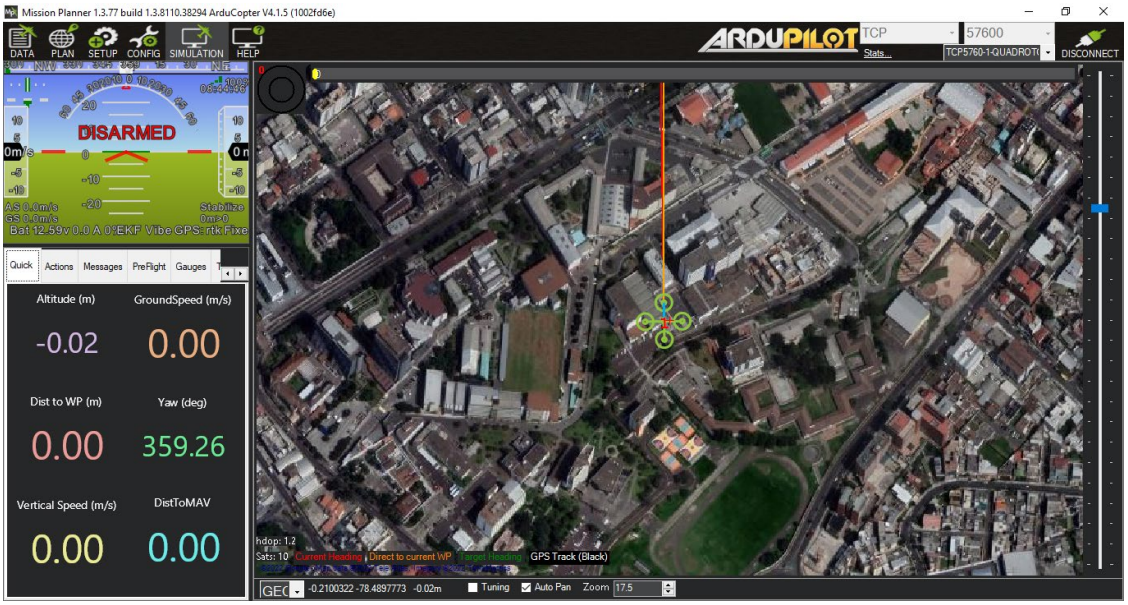


Figura 1.7. Interfaz de control de vuelo del Mission Planner [Fuente Propia]

Por todo lo mencionado, Mission Planner es una plataforma de control muy útil para el desarrollo de este trabajo, principalmente al momento de realizar la maniobra de evasión, para lo cual es necesario modificar el modo de vuelo y ciertas características de vuelo. Adicionalmente, como podemos ver en la Figura 1.7. Mission Planner cuenta con una interfaz de control en la cual se puede visualizar el estado actual de todas las características de vuelo, permitiendo así verificar de manera sencilla que el estado de vuelo del UAV sea el que se desea.

1.4.2.4 Modos de vuelo

El estudio de esta característica de vuelo es de gran importancia, ya que a través de modificar su estado general (modo automático) a otro modo de vuelo, se puede modificar en tiempo real la ruta de la misión preprogramada. Actualmente existen más de 5 modos de vuelo, sin embargo, solo 3 de ellos suelen ser usados con frecuencia [16]. Estos son:

- Modo automático: Este es el modo general con el que inicia el vuelo el dron, una vez enviada la orden de iniciar la misión. Mientras el modo automático este activo, no se puede modificar ninguna característica de vuelo previamente establecida en la misión preprogramada.
- Modo guiado: Este modo permite un control de la misión en tiempo real, es decir, mientras se permanezca en este modo de vuelo, se pueden dar órdenes al dron sin que estas hayan sido programadas con anterioridad [16].
- Modo Loiter: Como su nombre lo indica, este modo permite mantener al UAV volando en una misma posición a pesar del viento.

2 METODOLOGÍA

Este capítulo tiene como objetivo describir los procesos llevados a cabo para diseñar e implementar este trabajo de integración curricular (TIC), para ello, se hace uso de la teoría expuesta en el Capítulo 1 de este documento.

2.1 DESCRIPCIÓN DEL UAV DE PRUEBA

Para la implementación del presente trabajo de integración curricular se cuenta con un dron tipo cuadricóptero con un marco modelo F450. Este marco cuenta con una placa construida con fibra de vidrio, la cual incluye circuitería que permite soldar directamente los motores y la fuente de alimentación, y perforaciones que facilitan el montaje de cualquier hardware adicional. Sus brazos están construidos de un plástico resistente a los golpes y como se puede observar en la Figura 2.1 su longitud es de 215 mm, formando una distancia diagonal entre ejes de 455 mm [21].

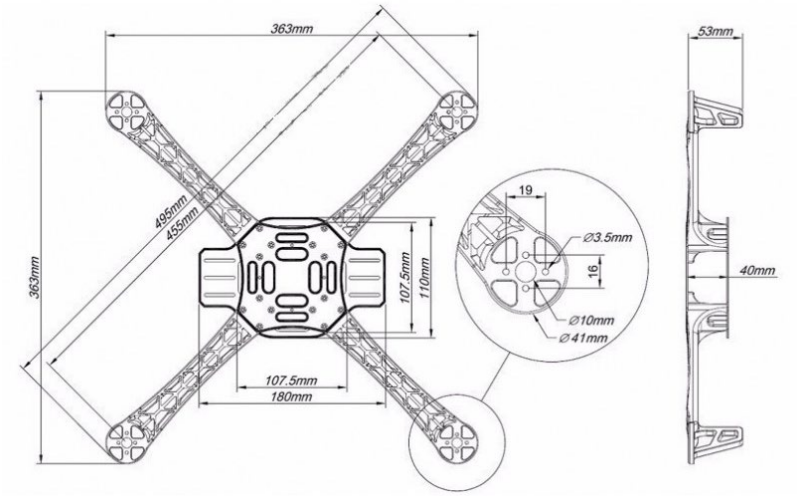


Figura 2.1 Estructura del Marco F450 [21].

Para generar la fuerza propulsora que eleva el UAV, cuenta con cuatro motores Brushless modelo A2212/13T de 1000KV, de aproximadamente 680 gramos cada uno, los cuales según su hoja de datos [22] pueden ser energizados por baterías de hasta 3 celdas. Para estos motores es posible obtener una eficiencia de empuje para propelas APC 10x4.5 E cuando son alimentados con un voltaje de 10.9 V. Teniendo en cuenta este dato el UAV puede elevarse hasta con una carga útil máxima de 2720 gramos, por lo que es importante que el peso del prototipo auxiliar de evasión de obstáculos sumado con todos los elementos que necesita para su correcto funcionamiento no supere este valor. En la Tabla 2.1 se presenta de forma más detallada el estudio de la carga útil para el presente trabajo.

Tabla 2.1. Carga del UAV de prueba.

Elemento	Peso/u (g)	Unidad	Peso Total (g)
Pixhawk Cube 2.1	73	1	73
Here 2 (GPS)	49	1	49
Transceptor RFD900+	41	1	41
Receptor FS – IA10B	19	1	17
Motor A2212/13T	48	4	192
Propelas 1045	7.5	4	30
Batería	675	1	675
Marco	282	1	300
Power Brick mini	16	1	16
Cableado extra	10	1	1
TOTAL:			1403

Como se observa en la Tala 2.3, el sistema tiene una carga fija de aproximadamente 1403 gramos, por lo que se puede obtener el peso máximo del sistema auxiliar de evasión de obstáculos, restando este valor de su carga útil máxima, dando como resultado 1317 gramos. Teniendo presente este valor y que se usa como SBC (Single Board Computer) una Raspberry Pi 4 modelo B, proporcionada por el grupo de investigación ATA de la Escuela Politécnica Nacional, se deben seleccionar los sensores que formaran parte del sistema de evasión de obstáculos.

2.2 SINGLE BOARD COMPUTER – RASPBERRY PI 4

Se trata de una computadora de placa reducida (SBC), que resalta de sus predecesores, principalmente por su procesador que resulta hasta tres veces más eficiente que el de su versión anterior, amplificando de esta manera, el campo de aplicaciones en las que se la puede usar. En la Figura 2.2 se puede observar el hardware de esta SBC y descrita en ella los principales cambios que la hacen sobresalir de la Raspberry Pi 3 modelo A+ [23].

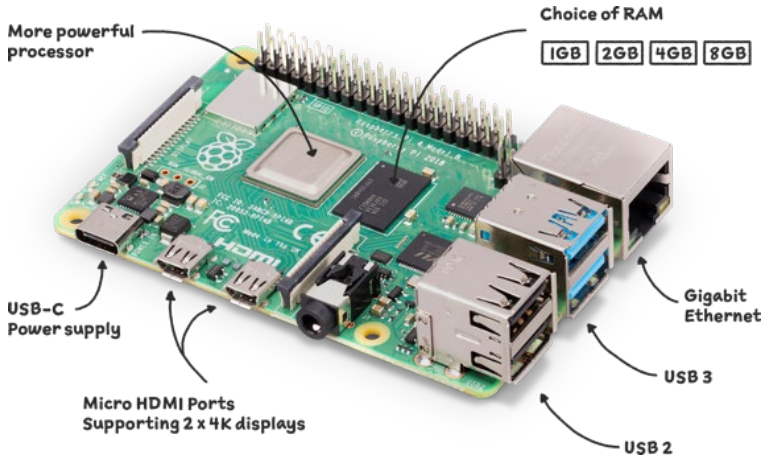


Figura 2.2 Hardware de la Raspberry pi 4 modelo B [23].

A continuación, se describen las especificaciones que son necesarias conocer para el uso adecuado de la Raspberry Pi 4 modelo B.

- Cuenta con un procesador Broadcom BCM2711B0, Quad – Core Cortex A72 de cuatro núcleos de 64 bits a 1.5 GHz [24].
- Memoria SDRAM LPDDR4 en valores de 1GB, 2 GB, 4 GB y 8 GB [24].
- Tiene alimentación vía USB-C y vía GPIO ambas de 5V/3A [24].

- Conectividad inalámbrica Wi-Fi de 2,4GHz / 5GHz IEEE 802.11ac, Bluetooth 5.0 y conectividad de red Gigabit Ethernet permitiendo llegar a tasas de 1000 Mbps [24].
- Cuenta con una unidad grafica integrada (GPU) VideoCore VI de 500 MHz que le permite tener salidas 4K/30fps a través de sus dos puertos micho HDMI [24].

La Raspberry pi 4 será la encargada de procesar las señales obtenidas del medio exterior a través de los sensores a emplear. Como para poder realizar una maniobra de evasión funcional se necesita obtener información del exterior, se optó por utilizar una cámara apoyada en la técnica de visión artificial. Si bien su implementación resulta compleja y demanda un procesamiento alto, también brinda mucha más información que la mayoría de los sensores existentes, por esto actualmente es muy utilizada en la detección de objetos

2.3 SELECCIÓN DE SENSORES

Como se analizó en la Sección 1.4.1.2, existe una gran variedad de sensores con principios de funcionamiento diferentes, que pueden ser usados para detectar obstáculos en un medio. No obstante, la selección de estos depende mucho de la acción de control que se desea realizar, del medio ambiente, del tiempo de respuesta y del rango de medida. Por ejemplo, un sensor infrarrojo tiene poca precisión en medios con mucha intensidad de luz, mientras que un sensor ultrasónico no se ve afectado por este problema. De esta manera, dado a que se necesita obtener información del medio en el que actúa un UAV, la mayoría de los sistemas comerciales de evasión de obstáculos cuentan como mínimo con dos sensores distintos, enfocados por lo general cada uno en medir una variable diferente.

Para realizar el prototipo auxiliar de evasión de obstáculos implementado en este trabajo, se tomó como base la estructura con la que está conformada el sistema comercial PENSAR [11], el cual cuenta de un equipo de alto procesamiento (Jetson Nano NVIDIA), una cámara que integra sensores infrarrojos y una cámara Sony 30x zoom HD visión.

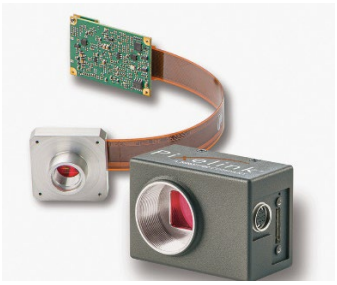
Para el presente proyecto, como se indicó anteriormente, se cuenta con una Raspberry Pi 4 modelo B como tarjeta de procesamiento, por lo que una característica de la cámara que se seleccione es que debe ser compatible con esta SBC.

2.3.1 CÁMARA

Como se describió en la Sección 1.4.1.2, existe una gran variedad de cámaras, sin embargo, se optó por estudiar aquellas que tengan una conectividad no tan compleja con la Raspberry PI 4 y que estén diseñadas para realizar aplicaciones de visión artificial. En

la Tabla 2.2. se puede observar una breve descripción y características más relevantes de algunas cámaras que cumplen con estas dos condiciones.

Tabla 2.2. Cámaras con uso en visión artificial

Nombre	Figura	Descripción
Cámara HD Runcam2		Se trata de una cámara ligera y robusta que gracias a su conectividad Wi-fi y precio no tan elevado, es muy usada para filmación en drones de carrera [25]. Resolución: 4 megapíxeles Dimensiones: 66x38x21 mm Video: 1080/60fps Peso: 49 gr Consumo de corriente: 210 mA Rango de medida: 120° Precio: 160 $
Cámara PL-D7718		Esta cámara enfocada principalmente para aplicaciones de visión artificial cuenta con conectividad USB 3.0 y tecnología CMOS, que le brinda una alta velocidad de fotogramas [26]. Resolución: 18 megapíxeles Alimentación: 5 V (USB 3.0) Peso: 35.8 gr Video: 1080/60fps Consumo de corriente: 15 mA Dimensiones: 55x38x30 mm Precio: 220 $
Cámara Raspberry Pi v1 OV5647		Esta cámara es adecuada para usarse con cualquier versión de Raspberry pi. Cuenta con 2 puntos infrarrojos que le permiten tener una grabación de alta calidad incluso en la noche [27]. Resolución: 5 megapíxeles Dimensiones: 25x25x9 mm Peso: 20 gr Video: 720/60fps Consumo de corriente: 300 mA Precio: 35 $
Camera Module v2		Esta cámara es la siguiente versión del módulo v1, tiene un sensor Sony IMX219 que ofrece una mayor resolución de video que el sensor OV5647 [27]. Resolución: 8 megapíxeles Dimensiones: 25x23x9 mm Peso: 3 gr Video: 720/60fps Consumo de corriente: 300 mA Precio: 45 $

HQ Camera		La HQ cámara es compatible con todas las tarjetas Raspberry pi. Cuenta con un sensor IMX477 que brinda una resolución de 1.55 micras de pixel [27]. Resolución: 12,3 megapíxeles Dimensiones: 38x38x18 mm Peso: 13 gr Video: 720/60fps Consumo de corriente: 300 mA Precio: 80 $

Debido a su velocidad de grabación de 1080/60fps y que cuenta con conexión USB 3.0 que la hace compatible con la Raspberry pi 4, la cámara PL-D7718 resulta una opción muy viable para ser usada en el desarrollo de este prototipo auxiliar, sin embargo, su precio un poco elevado limita su uso en proyectos de estudio. Por otro lado, los tres prototipos que ofrece la marca Raspberry no se diferencian mucho entre sí y dado que para este trabajo no es necesario usar una cámara de alta resolución, se optó por usar el módulo OV5647, la cual a más que cuenta con las características necesarias para realizar el prototipo auxiliar de evasión de obstáculos, permite hacerlo incluso con poca iluminación gracias a sus dos sensores infrarrojos integrados.

2.3.2 SENSOR DE DISTANCIA

Debido a que el prototipo auxiliar de evasión de obstáculos tiene como fin ser usado en exteriores, es importante tener en cuenta que el sensor de distancia no debe ser sensible a la luz. Además, debe brindar un rango de detección mínimo de 2m, ya que es necesario detectar un obstáculo presente en la trayectoria del UAV mínimo a una distancia dos veces su velocidad de vuelo, esto con el fin de que el UAV reaccione a tiempo para realizar la maniobra de evasión y así no colisione. En la Tabla 2.3 se presenta algunos sensores de distancia que cumplen con estos requisitos básicos.

Tabla 2.3. Sensores de distancia comerciales

Nombre	Figura	Descripción
Ultrasónico SRF10		Alcance: 6m Dimensiones: 32x15x10 mm Precisión: 3cm Peso: 10 gr Consumo de corriente: 15 mA Angulo de visión: 72° Precio: 45 $. Información obtenida de [28]

LIDAR Lite v3		Compatible con Arduino, Raspberry, Pixhawk Alcance: 40m Dimensiones: 40x48x20 mm Resolución: 1cm Peso: 22 gr Consumo de corriente: 130 mA Comunicación: I2C o PWM Precio: 135 $. Información obtenida de [29]
TeraRanger Evo 60m		Compatible con Arduino, Raspberry, Pixhawk Alcance: 60m Peso: 12 gr Conexión: I2C, USB y UART Procesado: hasta 240 señales por segundo Consumo de corriente: 330 mA Precio: 120 $. Información obtenida de [30]
RPLIDAR A1		Dimensiones: 98,5X70X60 mm Alcance: 12m Rango angular: 360° Resolución angular: <1° Consumo de corriente: 100 mA Peso: 170 gr Precio: 183 $. Información obtenida de [31]

El RPLIDAR A1, al tratarse de un sensor can capacidad de medir en un rango de 360°, es capaz de generar un mapeo 3D del entorno que lo rodea, resultando así una solución ideal para implementar un sistema de detección de obstáculos, sin embargo, debido a su peso y dimensiones resulta complicado montarlo en el dron de prueba que se tiene para realizar este trabajo, por lo que queda descartado. Generalmente los sensores con capacidad de realizar un mapeo 3D del entorno que los rodea tienen precios que superan los 500 dólares, por lo que su implementación en prototipos de bajo costo se ve limitada. Por esto, se optó por utilizar el LIDAR Lite v3 debido al gran alcance de medición que tiene, tiempo de respuesta de medida, y a su fácil conectividad con la Raspberry Pi 4. No obstante, este sensor presenta una gran desventaja y es que su medida es de un solo punto, por lo que solo se puede detectar objetos que estén frente al sensor.

Teniendo en cuenta los elementos seleccionados para conformar el sistema auxiliar de evasión de obstáculos el peso adicional que se suma a la carga del UAV se describe en la Tabla 2.4.

Tabla 2.4. Carga del UAV de prueba más prototipo auxiliar.

Elemento	Peso/u (g)	Unidad	Peso Total (g)
Carga del UAV	1403	1	1403
Raspberry Pi 4	45	1	45
Cámara v1 OV5647	17	1	17
LIDAR Lite v3	19	1	17
TOTAL:			1487

2.4 IMPLEMENTACIÓN DEL SISTEMA AUXILIAR

Para implementar el sistema auxiliar de evasión de obstáculos es necesario tener claro de los elementos que componen este sistema y cómo se comunican y conectan físicamente entre sí, esto se observa de manera detallada en la Figura 2.3. Adicional a esto, dado a que el prototipo de UAV empleado no cuenta con un espacio de soporte para el sistema auxiliar, es necesario diseñar e imprimir piezas de soporte que permitan montar todos los elementos necesarios en el dron.

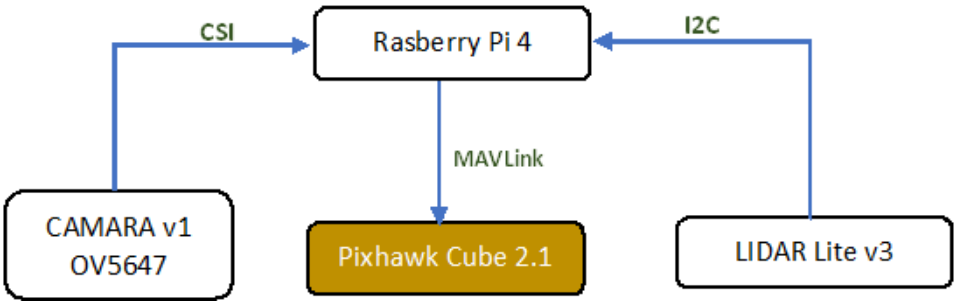


Figura 2.3 Diagrama de conectividad del sistema auxiliar de evasión de objetos [Fuente Propia].

Como se observa en la Figura 2.3, tanto la cámara como el LIDAR Lite van conectados a la Raspberry a través de diferentes interfaces de comunicación. La cámara v1 OV5647 se conecta a la Raspberry a través del Puerto CSI, el cual se configura desde el puerto terminal de la Raspberry haciendo uso del comando `sudo raspi-config`. Este comando abre el menú de interfaces, donde es necesario dirigirse a la pestaña Opciones de interfaces y habilitar la opción Legacy Camera para que el puerto CSI sea habilitado. Por otro lado, El LIDAR Lite v3 se comunica con la Raspberry a través de la interfaz I2C,

por lo que es necesario habilitar el bus I2C accediendo de igual manera al menú de interfaces y habilitar la opción bus I2C en la pestaña Opciones de interfaces.

Para entablar comunicación entre el controlador de vuelo Pixhawk Cube y la Raspberry Pi se usa el protocolo MAVLink, por lo que es necesario instalar la librería Pymavlink y mavutil que son bibliotecas que procesan mensajes MAVLink escritos desde Python. En la siguiente sección se describe a más detalle estos mensajes.

2.4.1 ESTABLECIMIENTO DE COMUNICACIÓN

Para establecer comunicación entre las Raspberry Pi y el sistema de vuelo se usó el comando `mavlink.conection(protocol:address:baudrate)` que ofrece el módulo mavutil de la librería Pymavlink. De esta forma, haciendo uso del protocolo TCP para entablar comunicación entre la Raspberry y la estación terrestre e igualando el valor de baudrate de ambas unidades, su implementación en Python queda de la siguiente manera:

```
# INICIO DE CONEXIÓN
the_connection= mavutil.mavlink_connection('tcp:127.0.0.1:5762')
the_connection.wait_heartbeat()
```

Una buena estrategia de programación es utilizar la función `wait_heartbeat()` que permite determinar el estado de la conexión, ya que al ser ejecutada entra a un estado de espera hasta que el UAV responda a la conexión.

Una vez ya establecida la comunicación del controlador de vuelo con la Raspberry, se puede enviar y recibir datos entre estos dos dispositivos. Para ello, existen dos clases de mensajes MAVLink que se pueden usar a través de la librería pymavlink, estos son:

- Mensajes de estado: Estos mensajes son aquellos que se envían desde el robot no tripulado hacia la estación a tierra y entregan información del estado del vuelo del dron, como la velocidad, posición, altura, entre otros [39].
- Mensajes de comando: A diferencia de los mensajes de estado, estos son enviados desde la estación a tierra (Mission Planner) hacia el robot no tripulado y se usan para ejecutar acciones o misiones en modo de vuelo automático [39]. Un ejemplo es el comando COMMAND_LONG_SEND el cual es un mensaje multipropósito que permite enviar diferentes tipos de mensaje, dependiendo el tipo de comando del mensaje que se defina en el campo command. En la Figura 2.4 se observa su estructura.

target system unit8_t	target component unit8_t	command unit16_t	confirmation unit8_t	param1 float	param2 float	param3 float	param4 float	param5 float	param6 float	param7 float

Figura 2.4 Estructura de un tipo COMMAND_LONG_SEND [39].

En la Figura 2.4 el campo command se refiere al tipo de comando a ejecutar, y se define en las enumeraciones de comandos MAV_CMD. A continuación, se presenta en la Tabla 2.5 los comandos MAV_CMD que se usan en este trabajo de integración curricular. Se sugiere revisar la referencia [7], para obtener más información respecto a todos los mensajes con los que cuenta MAVLink.

Tabla 2.5. Comandos COMMAND_LONG_SEND

Comando	ID	Descripción
DO_SET_MODE	176	Este comando se usa para establecer el modo de vuelo
ARM_DISARM	400	Arma y desarma los motores del UAV
MISSION_START	300	Comienza A ejecutar una misión en modo auto
CONDITION_YAW	115	Modifica el valor de posicion del yaw a un valor deseado

2.4.2 INTEGRACIÓN DEL SISTEMA DE EVASIÓN

El diseño e impresión de las piezas de montaje para los elementos que conforman el sistema auxiliar de evasión fue desarrollado con ayuda de un pasante de la Facultad de Ingeniería Mecánica, que está ofreciendo sus servicios al grupo de investigación ATA de la EPN. El diseño de todas las piezas se desarrolló en el software SolidWorks, el cual cuenta con un simulador que permite verificar la resistencia de los elementos a deformaciones por carga. Sobredimensionando el valor de carga del UAV a 3Kg y teniendo en cuenta que como material de impresión se usa ABS (polímero termoplástico), la simulación de una de las piezas diseñadas brinda estos resultados. Un ejemplo de este se observa en la Figura 2.5.

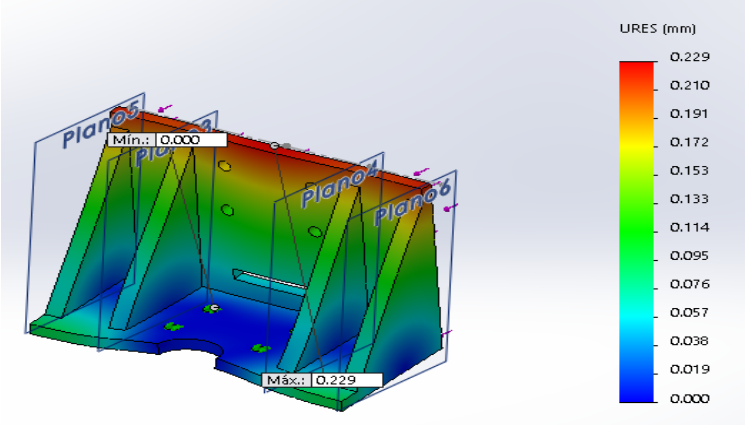


Figura 2.5 Resultado de la simulación de una pieza ante deformaciones.

De la Figura 2.5, al aplicar una carga de 3Kg sobre la pieza existe un desplazamiento de su centro de masa máximo de 0.229 mm, lo cual no representa ningún problema de deformación en la pieza.

Entre las piezas diseñadas para el montaje del prototipo auxiliar se tiene una plataforma de carga en donde se montará la Raspberry pi, y dos piezas rectangulares con sus respectivas perforaciones, para adaptar la cámara y el sensor LIDAR Lite v3. En la Figura 2.6 se puede observar el diseño de la plataforma de carga.

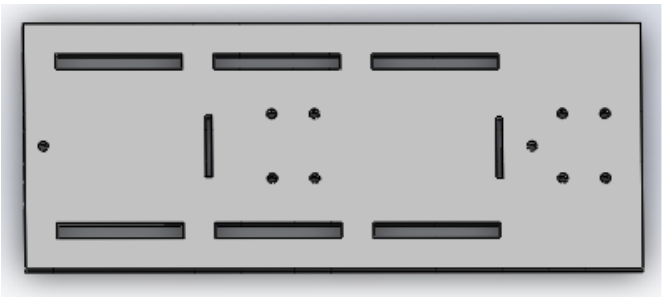


Figura 2.6 Plataforma de carga

La plataforma de carga que se observa en la Figura 2.6, se diseñó de tal manera que sirva para el montaje no solo del prototipo auxiliar de evasión de obstáculos diseñado en este TIC, sino también para otros prototipos auxiliares como puede ser uno de aterrizaje. Por ello la plataforma tiene algunas perforaciones en donde se puede adaptar el sistema de aterrizaje y de algunos orificios rectangulares por donde se puede pasar cables de conexión y amarraderas para sujetar los dispositivos.

Las piezas para el montaje de la cámara y el sensor LIDAR Lite son muy similares, la única diferencia que presentan es la posición y separación entre perforaciones, ya que estas se adaptan a las longitudes de cada sensor, estas se observan en la Figura 2.7.

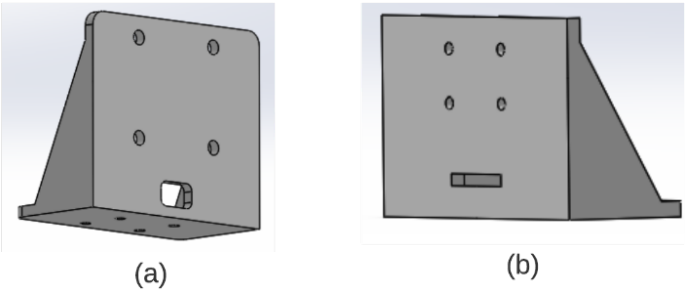


Figura 2.7 (a) Pieza para la cámara (b) Pieza para el LIDAR Lite [Fuente propia]

Una vez montado todos los elementos que conforman el sistema auxiliar de evasión en la aeronave, se realizó la conexión de todos los dispositivos. El esquema de conexión completo del prototipo auxiliar se puede ver a detalle en el Anexo I de este documento.

2.5 DETECCIÓN DE OBJETOS

El sistema auxiliar de evasión de obstáculos que se implementa en este trabajo de integración curricular cuenta con una etapa encarga de detectar objetos presentes en la trayectoria que lleva el UAV. Para ello, se hace uso de la cámara a través de una técnica de visión artificial que ayuda a determinar la posición del objeto con respecto a la aeronave, y del sensor de distancia LIDAR Lite v3 para determinar la distancia existente entre el UAV y el objeto. En la Figura 2.8. se puede observar a más detalle todo el proceso que se lleva a cabo para desarrollar la etapa de detección de objetos.

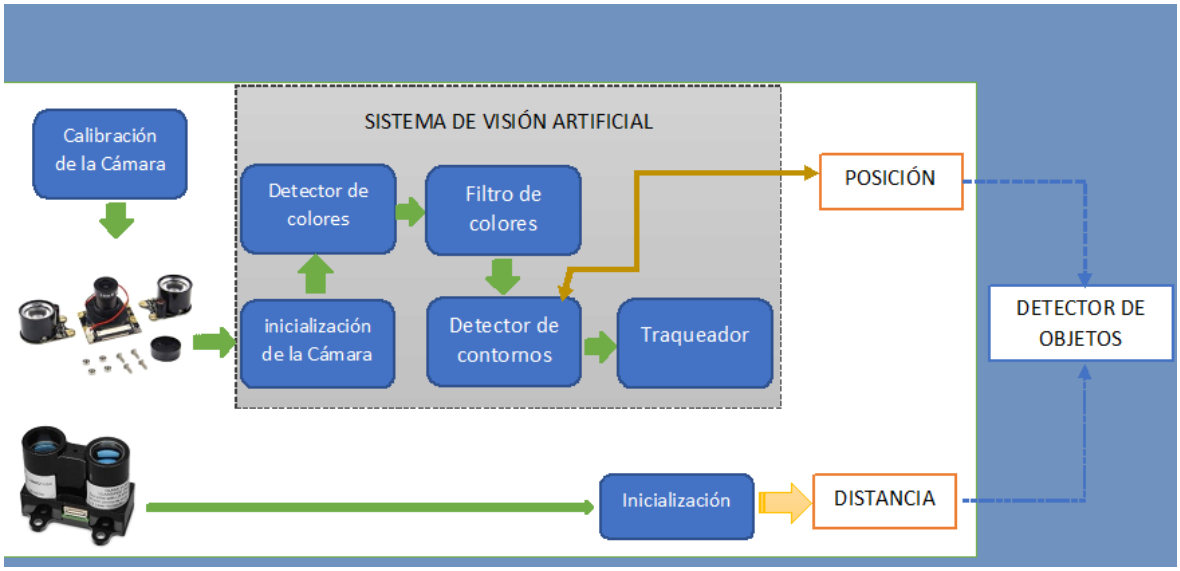


Figura 2.8 Estructura de la etapa de detección de objetos [Fuente Propia].

Como se puede observar en la Figura 2.8. el detector de objetos necesita información de dos variables, la primera es la posición del objeto, que se obtiene de procesar la señal

obtenida por la cámara, y la segunda la distancia que existe entre el UAV y el objeto, que es aportada por el sensor de distancia. El LIDAR Lite regresa directamente la variable de interés, por lo que el proceso para determinar la distancia no es complejo y solo hace falta inicializar el LIDAR Lite, como se describe más adelante en la Sección 2.5.2.

Para obtener la posición del objeto es necesario realizar una serie de procesos, a través de la visión artificial, con el fin, por ejemplo, de separar del fondo de la imagen el objeto de interés y así determinar su posición.

2.5.1 VISIÓN ARTIFICIAL CON OPENCV

OpenCV (Open Source Computer Vision Library) es una biblioteca de uso libre que permite desarrollar el procesamiento de imágenes y videos captados por una cámara, de forma fácil y rápida. Esta biblioteca es compatible con lenguajes de programación como C, C++, Python 3.0, por lo que se su aplicación es compatible con la SBC Raspberry Pi 4 [5]. Cuenta con muchas librerías que permiten realizar actividades como procesamiento de imagen, detector de características (color, tamaño), aceleradores gráficos, seguidores de objetos, calibración de cámaras y muchas aplicaciones más. En [32] se puede encontrar información más detallada de todo lo que contiene esta biblioteca y del uso apropiado de sus librerías.

Para que una imagen captada por una cámara sea procesada de manera apropiada esta no debe tener distorsiones, por eso antes de proceder a la etapa de procesamiento de imagen es necesario siempre realizar una etapa de calibración de la cámara en la que se elimine posibles distorsiones existentes en la imagen captada. OpenCV cuenta con librerías que hacen más fácil el proceso de calibración, por lo que se escogió usar esta biblioteca para desarrollar esta etapa.

2.5.1.1 Calibración de la Cámara

Existen diferentes métodos de calibrar una cámara usando OpenCV, sin embargo, el más usado es el método de Zhang o método del tablero de ajedrez como generalmente se lo conoce. Para llevar a cabo este método es necesario contar con una plantilla de un tablero de ajedrez asimétrico, tal y como se observa en la Figura 2.9, que sirva como patrón de referencia de la imagen ideal que se desea obtener [8].

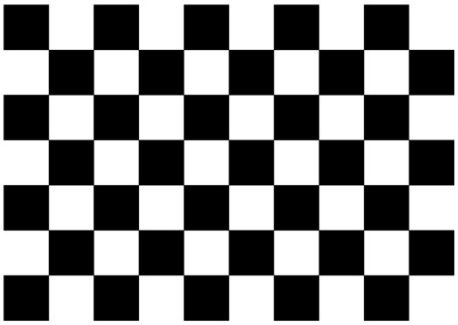


Figura 2.9 Patrón de referencia, tablero de ajedrez [34].

Como primer paso para calibrar la cámara, se deben calcular los vectores de rotación y traslación que se necesitan, para llevar el sistema de referencia de los puntos al plano de la cámara. Entonces se empieza rotando el sistema de coordenadas alrededor del eje z, el eje x y el eje y, con la ayuda de las siguientes funciones matriciales [8].

$$R_x(\theta) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\theta & \sin\theta \\ 0 & -\sin\theta & \cos\theta \end{bmatrix} \tag{2.1}$$

$$R_y(\varphi) = \begin{bmatrix} \cos\varphi & 0 & -\sin\varphi \\ 0 & 1 & 0 \\ \sin\varphi & 0 & \cos\varphi \end{bmatrix}$$

$$R_x(\theta) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\theta & \sin\theta \\ 0 & -\sin\theta & \cos\theta \end{bmatrix}$$

Una vez rotado el sistema de referencia, se debe trasladar hasta el plano de la cámara. Esto se representa como el vector de desplazamiento que se visualiza a continuación.

$$T = (x, y, z)_{objeto} - (x, y, z)_{camara} \tag{2.2}$$

Como siguiente paso para calibrar la cámara, se debe eliminar todo tipo de distorsión presente en la imagen. Existen dos tipos de deformaciones que distorsionan una imagen, la primera se conoce como distorsión radial y ocurre debido a la forma del lente de la cámara. Este fenómeno hace que la imagen se vea curvada hacia dentro o fuera de su plano, por lo que para realizar procesos de detección es necesario corregirlo [33]. Para resolver el problema de distorsión radial se hace uso de las siguientes ecuaciones:

$$X_{corrected} = x(1 + k_1 r^2 + k_2 r^4 + k_3 r^6) \tag{2.3}$$

$$Y_{corrected} = y(1 + k_1 r^2 + k_2 r^4 + k_3 r^6)$$

Donde x, y son los pares distorsionados, r la distancia entre el par distorsionado y el eje óptico y k_1, k_2, k_3 los coeficientes de la distorsión radial.

La segunda deformación que se presenta se llama distorsión radial y ocurre debido a que el plano de imagen captada no está alineada paralelamente al plano de la cámara. Este fenómeno hace que algunas áreas de la imagen captada con la cámara se vean más cerca de lo que realmente son [33]. Para resolver este problema se puede usar las siguientes expresiones:

$$X_{corrected} = x + [2p_1xy + p_2(r^2 + 2x^2)] \tag{2.4}$$

$$Y_{corrected} = y + [2p_2xy + p_1(r^2 + 2y^2)]$$

En esta última ecuación, p_1, p_2 los coeficientes de la distorsión tangencial.

Como resultado de solucionar los problemas de distorsión presentes en la imagen captada, se obtiene un vector con cinco parámetros al cual se le conoce como vector de coeficientes de distorsión. Con el sistema de referencias rotado y trasladado, el vector de coeficientes de distorsión y los parámetros intrínsecos de la cámara se obtiene la calibración que se busca [8]. Entre los parámetros intrínsecos que se necesitan conocer de la cámara esta la distancia focal (f_x, f_y) y los centros ópticos (c_x, c_y), los cuales se agrupan formando una matriz de 3x3 conocida como matriz de la cámara [33].

Para obtener los parámetros intrínsecos y los coeficientes de distorsión de la cámara Raspberry Pi OV5647 se usa la plantilla presentada en la Figura 2.9, a la que se le aplica 34 fotogramas obteniendo así los siguientes resultados:

$$Camera_{matriz} = \begin{bmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} 724.94 & 0 & 294.07 \\ 0 & 726.57 & 224.90 \\ 0 & 0 & 1 \end{bmatrix} \tag{2.5}$$

$$Coeff_{distortion} = [k_1 \quad k_2 \quad p_1 \quad p_2 \quad k_3]$$
$$= [-0.4904 \quad 0.5734 \quad -0.0012 \quad -0.0004 \quad -0.9917]$$

Tanto los coeficientes de distorsión como los parámetros intrínsecos dependen únicamente de la cámara, por lo que estos valores se pueden almacenar en un archivo para posteriormente ser usados en aplicaciones futuras. Una vez que se tenga los valores de la matriz de cámara y vector de coeficientes de distorsión, se usa el comando `cv2.undistort` como se presenta a continuación, para obtener la imagen calibrada que se observa en la Figura 2.10.

```
# Undistort
dst=cv2.undistort(img, mtx, dist, None, newcameramtx)
```

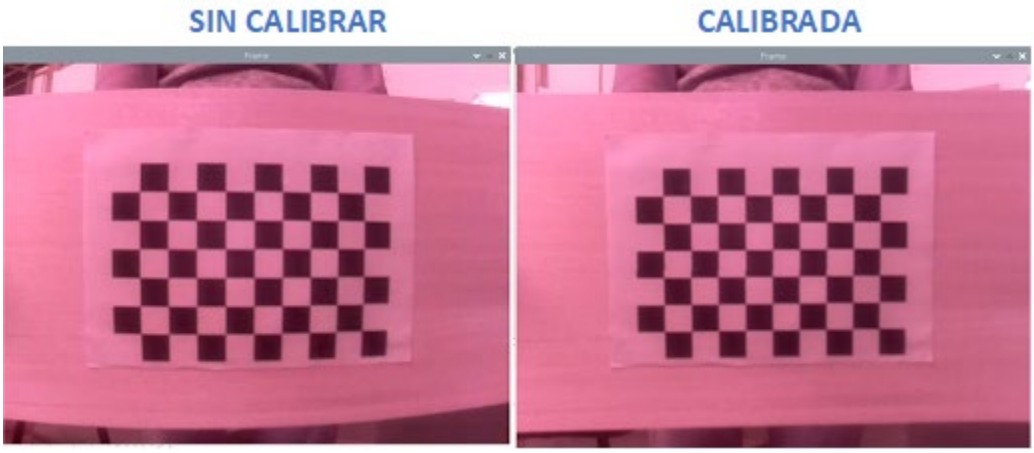


Figura 2.10 Resultados de la calibración (vale aclarar que el color rojizo que presenta la imagen resultante se debe a los sensores infrarrojos con los que cuenta la cámara) [Fuente propia].

Como se observa en la Figura 2.8, luego de haber realizado el proceso de calibración se procede a hacer un procesamiento de imagen usando operaciones morfológicas, que como se dijo en la Sección 1.4.1.3. ayudan a obtener de manera sencilla características de la imagen como color, zonas convexas, bordes, entre otros; mismas que se explican a continuación.

2.5.1.2 Detector de colores con OpenCV

En esta etapa se tiene como objetivo determinar el color del objeto más cercano al centro de la imagen captada con la cámara OV5647, esto con la finalidad de separarlo del fondo de la imagen, para posteriormente evadirlo. Por lo tanto, se describe a continuación de manera ordenada, todos los pasos que se realizaron para determinar el color del objeto más cercano al centro de la imagen.

Ajustar el tamaño de la imagen a uno constante

Dado a que solo interesa el objeto más cercano al centro de la imagen es necesario que el tamaño de la imagen sea contaste en todo momento, es decir que siempre tenga el mismo valor de pixeles de ancho y alto. De esta manera, es fácil determinar el centro de referencia desde el cual se procederá a realizar el detector de color. Justamente, para redimensionar una imagen con OpenCV se usa el comando `cv2.resize()` de la siguiente manera.

```
#redimensionamiento de imagen
image=cv2.resize(frame, (640,480))
```

Donde `image` es la imagen redimensionada, `frame` es la imagen original y los valores de `640, 480` son el nuevo ancho y alto que tendrá la imagen respectivamente. Una vez

redimensionada la imagen se obtiene su centro, el cual será usado como pixel central de la máscara NxN, que se crea más adelante, y que se usa para separar del fondo de la imagen del objeto de interés. Esto se consigue haciendo uso del comando `shape()`, el cual ayuda a determinar las dimensiones que tiene un array, en este caso la imagen. A continuación, se observa su implementación en Python.

```
# Determinar el centro de la imagen
W1 = image.shape[1]
W2= W1*0.5
H1 = image.shape[0]
H2 = H1*0.5
```

Donde:

- `Image`: Es la imagen redimensionada en el paso anterior
- `W1`: El ancho en pixeles de la imagen redimensionada
- `H1`: El alto de la imagen redimensionada
- `W2`: El valor central del ancho de la imagen
- `H2`: El valor central del alto de la imagen

Convertir la imagen a escala de HSV

Cuando se obtiene una foto a través de una cámara esta ingresa al computador en valores del espacio de colores RGB (R de Red, G de Green y B de Blue), sin embargo, es más conveniente usar el espacio de colores HSV (Hue, Saturation, Value / Matiz, Saturación, Brillo), cuando lo que se quiere hacer es determinar un rango de colores definido por valores límites [34]. El esquema de cómo se compone el espacio de colores HSV se puede visualizar en la Figura 2.11, en donde se ve claramente que el color de un objeto dentro de este espacio de colores se define únicamente por el valor del Hue, el cual indica en si el color del tono, siendo 0 blanco y 255 negro.

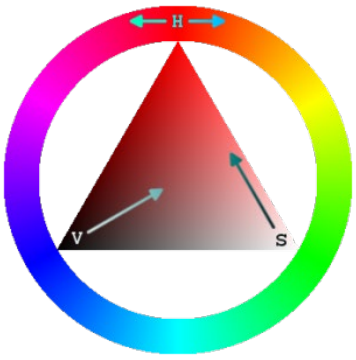


Figura 2.11 Espacio de colores HSV [34].

OpenCV ofrece un método rápido para convertir una imagen entre espacios de color. En este caso se necesita cambiar del espacio de colores RGB al espacio HSV, y para hacerlo se hace uso del comando `cv2.cvtColor`, tal y como se observa a continuación.

```
#Convertimos en escala de grises
gris = cv2.cvtColor(image, cv2.COLOR_BGR2HSV)
```

De esta manera se guarda en `gris`, el resultado obtenido de convertir la imagen original `image` al espacio HSV.

Filtrar la imagen

Toda señal digital o analógica está expuesta al ruido, las imágenes digitales no son la excepción. El objetivo de utilizar un filtro es reducir el ruido presente en la imagen, con la finalidad de suavizarla y eliminar detalles que están demás en la imagen. En el procesamiento de imágenes existen diferentes métodos para eliminar el ruido de una imagen digital (Gaussiano, aritmético, mediana), sin embargo, todos basan su funcionamiento en la operación matemática convolución, la cual, consiste en recorrer píxel a píxel la imagen de entrada con una máscara de tamaño NxN donde N indicara el número de pixeles con los que se va a trabajar [35]. Teniendo en cuenta esto, se programó un filtro de color con histéresis basado en [38], el cual determina el umbral máximo y mínimo que debe tener un pixel dentro del espacio de colores HSV, para separarlo del fondo de la imagen.

Filtro de color por histéresis

Este filtro cuenta de dos componentes, el primero son sus umbrales máximo y mínimo que son dos vectores creados a partir de limitar los tres componentes del espacio de colores HSV y guardar estos valores en forma de vectores con la ayuda del comando `np.array`(), y el segundo componente es la máscara con la que se convoluciona pixel a pixel la imagen de entrada con los valores de los umbrales máximo y mínimo. Para realizar lo descrito al final se usa el comando `cv2.inRange` el cual devuelve como resultado una imagen binaria, tal y como se observa en la Figura 2.12, en donde el color blanco presenta los lugares que se encuentran dentro del rango de tono que se define en el filtro de color. A continuación, se presenta la implementación del filtro de color en Python.

```
#Filtro de color por histéresis
Bajo = np.array([hc-5, sc-15, vc-25], np.uint8)
Alto = np.array([hc+5, sc+15, vc+25], np.uint8)
mask = cv2.inRange(frameHSV, Bajo, Alto)
```

Donde:

- `Bajo`: Es el vector que determina el umbral mínimo del filtro de color
- `Alto`: Es el vector que determina el umbral máximo del filtro de color
- `frameHSV`: La imagen original convertida al espacio de colores HSV
- `mask`: Es la máscara de convolución del filtro de color

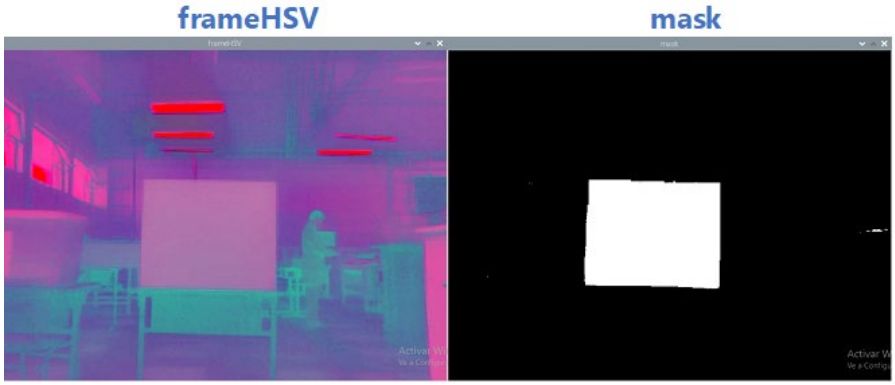


Figura 2.12 Resultado del detector de colores [Fuente propia].

2.5.1.3 Detector de contornos

Siguiendo la estructura descrita en la Figura 2.8, como siguiente proceso a desarrollar para diseñar el detector de objetos es un detector de contornos. Para desarrollar un detector de contornos fiable usando OpenCV, es recomendable diseñar primero un detector de bordes que entregue una imagen de referencia, para dibujar el contorno que encierran el objeto de interés. OpenCV brinda algunas funciones que permiten detectar bordes de una imagen, sin embargo, la función Canny es la que más sobresale de todas ella. La función Canny basa su funcionamiento en aplica un umbral por histéresis, en donde se establece el valor de un umbral máximo y un umbral mínimo [35]. De esta manera, se determina si un píxel pertenece a un borde teniendo en cuenta estas tres condiciones:

- Si el píxel tiene un valor mayor al del umbral máximo, se considere que pertenece a un borde
- Si el píxel tiene un valor menor al del umbral mínimo, se considera que no es parte de un borde
- Si el valor del píxel esta entre el umbral mínimo y máximo, se considera que está conectado a un píxel que forma parte de un borde por lo que se también se lo toma en cuenta.

Todo este proceso que puede resultar complejo implementarlo desde cero, se lo puede hacer de manera sencilla gracias a OpenCV a través del comando `cv2.Canny()`, el cual se lo implementa en lenguaje Python de la siguiente manera.

```
#Detector de bordes
canny = cv2.Canny(mask, umbral mínimo, umbral máximo)
```

Donde mask es la imagen binaria obtenida en el paso anterior, canny es la imagen resultante donde se visualiza los bordes calculados, y umbral mínimo y máximo son los vectores calculados en el proceso anterior, que determinan el rango del filtro de color. En la Figura 2.13. se puede observar los resultados que se obtenidos tras a ver implementado el detector de bordes Canny.

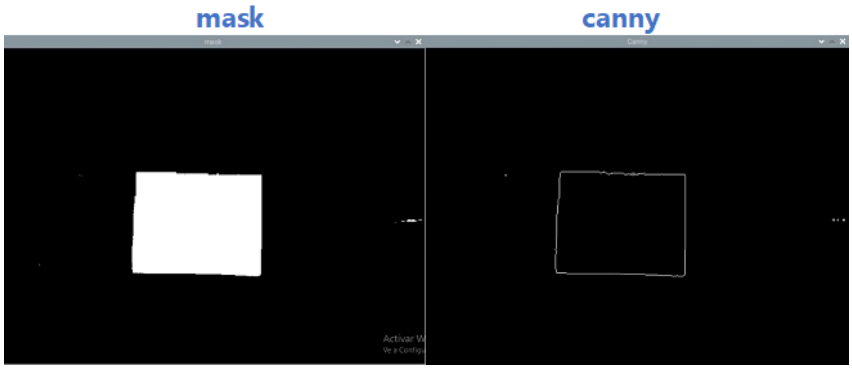


Figura 2.13 Resultado del detector de bordes canny [Fuente propia].

Antes de implementar el detector de contornos se realizó un proceso de dilatación de la imagen binaria obtenida con el detector de bordes Canny. La dilatación es una operación morfológica básica que se usa para expandir, ampliar una o unir partes rotas de un objeto [14]. En este caso se usa la dilatación para amplificar y unir los bordes obtenidos con el detector canny, haciendo que la imagen que entra al detector de contornos tenga los bordes más pronunciados. En la Figura 2.14 podemos observar los resultados de aplicar la operación dilatación a la imagen obtenida del detector de bordes.

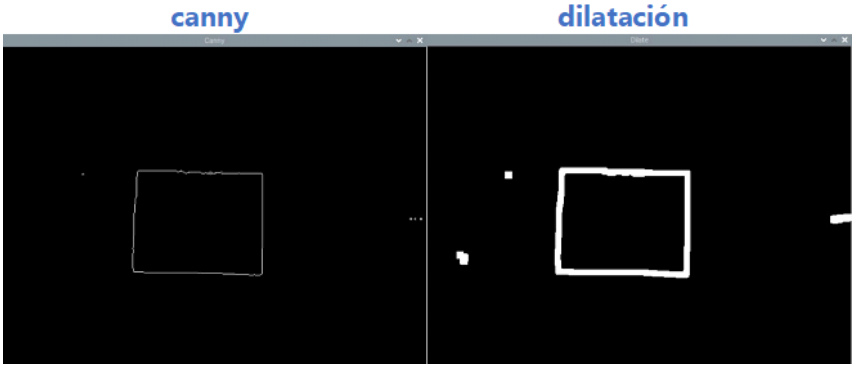


Figura 2.14 Dilatación de bordes [Fuente propia].

Con los bordes dilatados del objeto de interés, se prosigue a detectar todos los contornos presentes en la imagen, teniendo en cuenta que un contorno es una curva cerrada, que encierra un espacio sin huecos ni saltos en toda su longitud. En OpenCV para detectar si un borde es un contorno se usa el comando `cv2.findContours` el cual implementado en lenguaje de programación Python tiene la siguiente estructura.

```
#Detector de CONTORNOS
(contornos, jerarquia) = cv2.findContours(mask.copy(), modo_cont, método)
cv2.drawContours(image,contornos,num_contornos,color,grosor)
```

Donde:

- `contornos`: Es la variable donde se guara la lista de contornos detectado
- `jerarquia`: Es la jerarquía del contorno, es decir si es un contorno que encierra a otros contornos, o solo un contorno separado.
- `mask.copy`: Es una copia de la imagen dilatada obtenida en el proceso anterior.
- `modo_cont`: En este espacio se coloca el modo de contorno que queremos obtener. Estos pueden ser todos los contornos (`cv2. RETR_LIST`), solo contornos externos (`cv2.RETR_EXTERNAL`), o todos los contornos agrupados en orden jerárquico (`cv2.RET_TREE`).
- `Metodo`: Aquí se define la forma en cómo se aproxima los contornos, es decir cuántos puntos se toma para dibujar el contorno. Es recomendable usar el método `cv2.CHAIN_APPROX_SIMPLE`, ya que este elimina todos los puntos redundantes presentes en el contorno lo que disminuye el tiempo de procesado.
- `color`: El color con el que se requiere que se dibuje los contornos.
- `image`: Es la imagen donde se dibujan los contornos obtenidos. Este resultado se puede observar en la Figura 2.15
- `num_contornos`: El número de contornos que se quiere dibujar, si son todos se pone el valor de -1.

Figura 2.15 Detector de contornos con filtro de color [Fuente propia].

Con el fin de afinar el detector de contornos del proceso anterior, se emplea un filtro de contornos el cual elimina todos los contornos de la imagen con excepción del que representa al objeto que se encuentra más cercano al centro de la imagen. Para ello, es necesario calcular la distancia que existe desde el centro de la imagen hacia cada contorno presente en ella y tomar únicamente el que tenga la menor distancia. A continuación, se presenta como se realizó este filtro de contornos.

```
#CALCULO DEL CENTRO DE CADA CONTORNO
M = cv2.moments(cnt)
cx = int(M["m10"]/M["m00"])
cy = int(M["m01"]/M["m00"])
#DISTANCIA DE CADA CONTORNO CON RESPECTO AL CENTRO
distancias.append(math.dist([W2,H2],[cx,cy]))
index.append(c) #crea una lista con el índice de cada contorno
#Minima distancia
mínimo=min(distancias)
ind=distancias.index(mínimo)#se toma el índice del contorno más
cercano al centro
cnt=contornos[index[ind]]
```

Primero se calcula el centro de masa de cada contorno con la ayuda del comando `cv2.moments`, el cual devuelve todos los momentos de cada contorno. Esos momentos guardan información como el centro de masa, área, perímetro, entre otros. De esta manera se puede acceder a estas características del contorno a través de seleccionar el momento adecuado, un ejemplo es el momento `M["m00"]` que devuelve el área del contorno.

Con el centro de masa calculado de cada contorno, se puede determinar la distancia que existe entre ellos y el centro de la imagen usando el comando `math.dist` el cual devuelve el valor de la distancia que existe entre dos puntos. Estos valores se guardan en forma de lista en la variable `distancia` y con ayuda del comando `min(Variable)` se escoge solo el valor mínimo del vector distancia. Por último, para guardar únicamente el contorno más

cercano al centro de la imagen, se determina el índice que pertenece al contorno más cercano al centro y se apunta con el comando `contornos[index[ind]]`, que dice de todos los contornos, y de la lista de contornos index, selecciona solo aquel cuyo índice es igual a `ind`, donde `ind` indica el índice del contorno más cercano al centro.

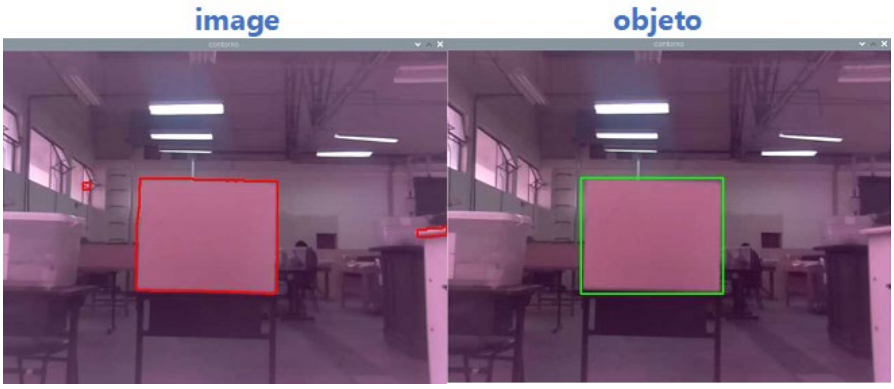


Figura 2.16 Resultado del detector de objetos [Fuente propia].

Por último, como se observa en la Figura 2.16, se encierra en un delimitador rectangular usando el comando cv2.rectangle, el objeto que pertenece al contorno definido en el filtro de contornos. Con este último paso se determina la posición del objeto que se desea evadir para impedir la colisión.

Una vez detectado el objeto de interés en el plano de la imagen, se implementa un seguidor de objetos con el fin de robustecer el sistema implementado. Con esto, se logra asegurar que no se pierda de vista el objeto detectado mientras siga en el campo de visión del UAV, y que una vez detectado el objeto no se vuelva a detectar otro objeto mientras el primero no haya sido evadido.

2.5.1.4 Traqueador

Un tracker o seguidor de objetos, como también se lo conoce, se usa para dar seguimiento a una región de la imagen previamente delimitada en un bounding box. El proceso de seguimiento empieza una vez obtenido el bounding box o cuadro delimitador que encierra el objeto que se desea seguir. La ubicación actual de los pixeles encerrados dentro del cuadro delimitador es comparada en los siguientes fotogramas, para de esta manera determinar la nueva posición del objeto y actualizar los valores que limitan el bounding box [35]. Este seguidor deja de funcionar una vez que el objeto seguido sale del campo de visión o cuando el seguidor recibe un falso positivo.

OpenCV ofrece una gran cantidad de trackers que pueden ser implementados para desarrollar este trabajo de integración curricular, sin embargo, se ha optado por usar el algoritmo KCF debido a que es el que menor recursos de procesamiento necesita en comparación a los demás, esto considerando que se está empleando una SBC como sistema de procesamiento. Un estudio más profundo de los diversos tracker que ofrece OpenCV se lo encuentra en [36].

Funcionamiento del seguidor KCF

Como se observa en la Figura 2.17, prácticamente el seguidor entrará a funcionar una vez que se obtenga el cuadro delimitador que encierra al objeto que toca evadir. Este cuadro delimitador se obtiene del proceso detección de objetos descrito en la Figura 2.8. Una vez determinado el bounding box se inicializa el tracker con el comando `tracker.init()`, el cual tiene como datos de entrada el fotograma en el que se detectó el objeto y el bounding box que lo encierra. Ya inicializado el seguidor lo único que queda es dar seguimiento al objeto con la ayuda del comando `tracker.update()`, el cual es el encargado de actualizar el valor del bounding box en función a la nueva posición de los pixeles que se encuentran en su interior. El algoritmo finaliza una vez el objeto sale del campo de visión o si el algoritmo recibe un falso positivo.

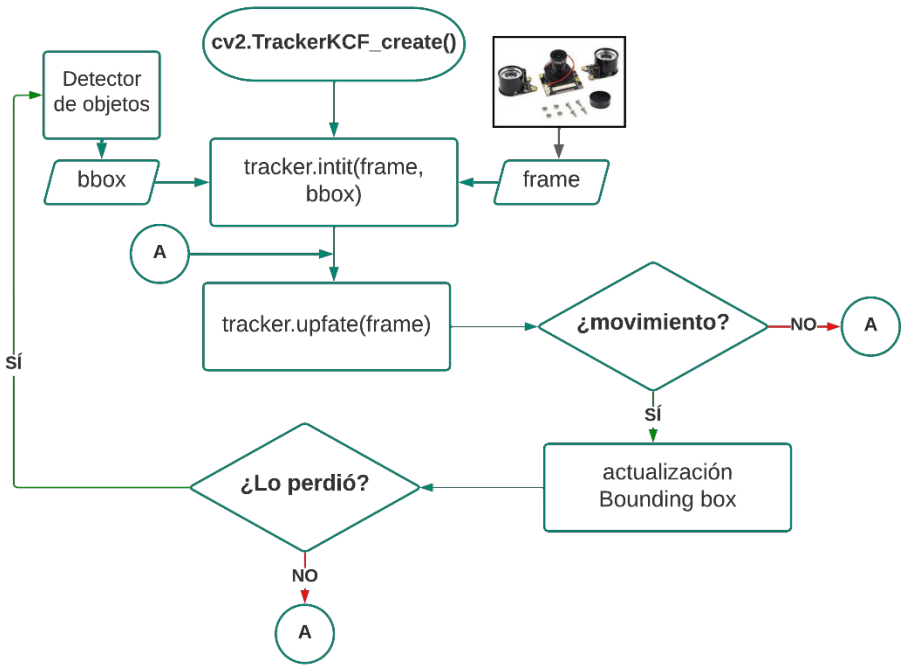


Figura 2.17 Diagrama de flujo del funcionamiento del tracker [Fuente propia].

Existen diversos factores por los que el seguidor puede regresar un falso positivos mientras el UAV está evadiendo el obstáculo, entre las más comunes se tienen: movimiento brusco

del objeto seguido, obstrucción del campo de visión, velocidad de movimiento muy alta, etc. Es por esto que se requiere que el sistema implementado, cuente con posibilidad de poder volver a detectar el objeto que está obstruyendo el camino del dron y volver a encerrarlo en un cuadro delimitador. Es aquí donde se hace uso del LIDAR Lite v3 como indicador de que existe un objeto dentro del rango de peligro de colisión, para volver a correr la parte de visión artificial del algoritmo detector de objetos descrito en la Figura 2.8.

2.5.2 LIDAR LITE V3

El sensor de distancia Lidar Lite v3 es el encargado de determinar la distancia que existen entre el objeto detectado y el UAV. Este valor de distancia se utiliza para activar una bandera que inicializa la cámara, la cual a su vez da inicio al sistema de detección de objetos. Entonces se define un umbral de distancia mínima, tomando como referencia la velocidad a la que se mueve el UAV, y se nombra zona de peligro toda aquella que se encuentra por abajo del umbral definido.

Para utilizar el LIDAR Lite como medidor de distancia no es necesario calibrarlo o realizar cálculos complejos, únicamente se necesita instalar la librería `adafruit_lidarlite`, y entablar comunicación I2C entre la Raspberry pi4 y el sensor. Esto último descrito se lo realiza usando el comando `busio.I2C` de la siguiente manera:

```
#Comunicacion I2C
i2c = busio.I2C(board.SCL, board.SDA)
```

Se recomienda revisar la referencia [37] para descargar la librería `adafruit_lidarlite` y el código completo de prueba para medir distancias con el LIDAR Lite v3.

Una vez que el UAV se encuentra en la zona de peligro y de que haya detectado el objeto presente en ese rango, se comienza a realizar la maniobra de evasión hasta que el objeto detectado salga del campo de visión y de que el LIDAR Lite detecte que el UAV tiene una distancia superior al umbral mínimo con respecto a cualquier objeto presente.

2.6 EVASIÓN DE OBJETOS

La maniobra de evasión de obstáculos tiene inicio una vez que se ha medido, gracias a los sensores colocados, la posición y distancia que tiene el objeto que se desea evadir con respecto al robot aéreo. Para ello, y teniendo en cuenta que como controlador de vuelo se está usando el Pixhawk Cube 2.1, es necesario tomar decisiones sobre el comportamiento del dron a través del uso de los comandos y mensajes que proporciona el protocolo de comunicación MAVLink, que fueron descritos en la Sección 2.4.1.

Teniendo en cuenta lo descrito en el párrafo anterior se pueden implementar varias estrategias de evasión, ya sea actuando sobre el modo de vuelo del dron o modificando la trayectoria definida al inicio de la misión. Esto último se realiza modificando los waypoints.

2.6.1 MANIOBRAS DE EVASIÓN DE OBJETOS

La forma más sencilla de evadir un obstáculo presente en una trayectoria previamente definida a través del Mission Planner, es modificando el valor del ángulo yaw en 45 grados, al momento que se detecta la presencia de un objeto [40]. Sin embargo, este método resulta ser limitado ya que solo serviría para evadir obstáculos de los cuales ya se conoce previamente sus dimensiones, y así calcular la distancia respecto al objeto, a la cual el dron debe comenzar a modificar en 45 grados el yaw, es decir rotar en 45 grados con respecto al eje z de la aeronave. No obstante, esta técnica se puede usar como referencia para crear una maniobra de evasión más dinámica basada en cambiar el modo de vuelo del UAV a modo de vuelo guiado o a modo de vuelo Loiter, los cuales permite tener control sobre el yaw a través de los mensajes de tipo comando CMD que ofrece la librería pymavlink. A continuación, se describe cómo funcionan estas dos opciones.

Evasión cambiando el modo de vuelo a Loiter

Esta maniobra de evasión solo funciona en caso de que el obstáculo detectado no sea estático. Su funcionamiento prácticamente se basa en cambiar el modo de vuelo del dron, a modo de vuelo Loiter (descrito en la Sección 1.4.2.4), una vez que se haya detectado un obstáculo a una distancia menor a la definida. Una vez que ya no se detecte un obstáculo dentro del rango de peligro, el dron retomara su misión. En la Figura 2.18 se puede observar su funcionamiento.

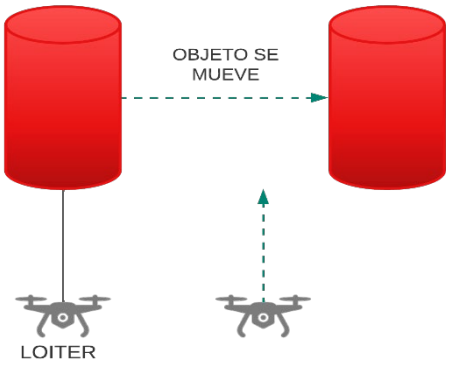


Figura 2.18 Evasión en modo Loiter [Fuente propia].

Evasión guiada

Otra forma de realizar una maniobra de evasión de obstáculos es cambiar el modo de vuelo a modo guiado (descrito en la Sección 1.4.2.4), . De esta manera una vez que se tenga al UAV en modo guiado, se pueden cambiar sus características de vuelo a voluntad haciendo uso de los mensajes de tipo comando descritos en la Sección 2.4.1. La ventaja que presenta la maniobra de evasión guiada es que se puede obtener un comportamiento más dinámico en la evasión de obstáculos, a través de hacer que este se mueva en todas las direcciones, o cambiar su velocidad de vuelo. En la Figura 2.19 se puede observar su funcionamiento.

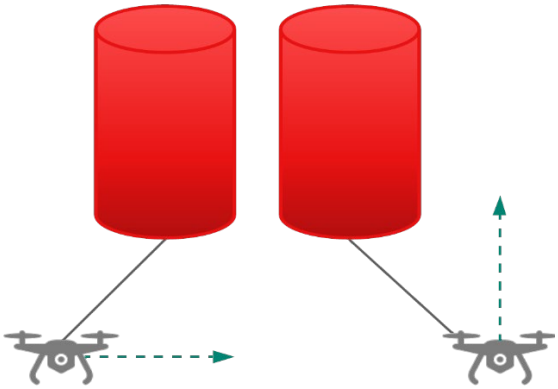


Figura 2.19 Evasión en modo guiado [Fuente propia].

2.6.2 SISTEMA DE EVASIÓN

Para este trabajo se realizó una maniobra de evasión de obstáculos, implementando el modo guiado y una maniobra basada en el principio de funcionamiento del modo Loiter, el cual mantiene la posicion y velocidad de vuelo del UAV constantes en un punto indicado. Esta última maniobra se hará manteniendo el modo de vuelo en guiado por eso, de a hora en adelante a esto se le conocerá como Loiter guiado. Para ello primero se crea una misión a través del Mission Planner y se carga al autopiloto Pixhawk Cube, posterior a eso cuando el LIDAR Lite v3 detecta que un objeto se encuentra a una distancia por encima del umbral mínimo y que el detector de objeto ya haya determinado la posición de este respecto al dron, se da paso a la maniobra de evasión.

El cálculo del umbral mínimo de distancia se hace a través de multiplicar la velocidad de vuelo del dron con el tiempo que demora en procesar la señal, el código implementado. En la Tabla 2.6 se observa algunos ejemplos de la respuesta de detección en función a la velocidad de vuelo y umbral mínimo de distancia.

Tabla 2.6. Respuesta de detección en función de un umbral de distancia

Velocidad de vuelo	Tiempo de respuesta	Umbral mínimo escogido	Resultado
3 [m/s]	1 s	4 [m]	El UAV se detiene a 1 metro del obstáculo, dado que el umbral mínimo escogido está por encima del que se necesita
5 [m/s]	2 s	5 [m]	Dado a que el umbral mínimo debería estar por encima de 10 m, en este caso el dron sufrirá una colisión
2 [m/s]	1 s	4 [m]	El umbral mínimo escogido está por encima del calculado de 2 m, entonces el UAV impide la colisión

De esta manera, la maniobra implementada consiste en mover a la aeronave a la derecha o a la izquierda hasta que el objeto que está impidiendo su avance salga del campo de visión de la cámara. La dirección del movimiento depende del valor de la posición del objeto con respecto al centro de la imagen, definiendo así que el movimiento sea a la derecha cuando la diferencia entre el centro de la imagen y la posición del objeto sea negativa, y a la izquierda en el caso que esta diferencia sea positiva. Mientras se lleva a cabo la maniobra de evasión el tracker se encuentra funcionando y es este mismo el que indicara que el objeto se encuentra fuera del campo de visión. En la Figura 2.20 se observa el algoritmo de esta lógica.

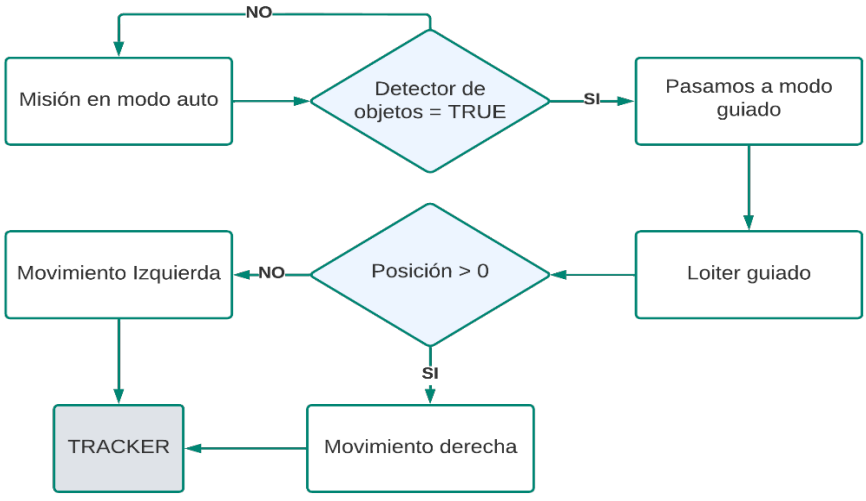


Figura 2.20 Diagrama del sistema de evasión de obstáculos [Fuente propia].

Como se observa en la Figura 2.20 el sistema de vuelo comienza en modo automático, una vez que se detecta la presencia de un objeto el modo de vuelo pasa a guiado, a través de usar el comando `MAV_CMD_DO_SET_MODE` de la siguiente manera:

```
#PASAR A MODO GUIADO
mode_id=the_connection.mode_mapping()['GUIDED']

the_connection.mav.command_long_send(the_connection.target_system,
        the_connection.target_component,mavutil.mavlink.MAV_CMD_DO_SET_MODE,
        0, 0, mode_id, 0, 0, 0, 0, 0)
```

Donde mode_id guarda el valor ID del modo de vuelo que se establece, en este caso guiado. El comando `MAV_CMD_DO_SET_MODE` toma en consideración el tercer parámetro para identificar el tipo de vuelo al que se desea acceder. Antes de hacer que el dron se mueva en alguna dirección se realiza un Loiter guiado, esto con el fin de hacer que la maniobra de evasión no sea tan brusca y así el tracker no vaya a recibir un falso positivo. Esto se realiza manteniendo su posición constante a través del comando de posición `SET_POSITION_TARGET_GLOBAL_INT` [18], el cual se configura a través de dos parámetros. El primero es conocido como MAV_FRAME y es el encargado de indicarme las coordenadas con las que se envía el mensaje, para este trabajo se usó el comando `MAV_FRAME_BODY_OFFSET_NED`, dado a que este comando configura las coordenadas de posición respecto a la dirección a donde a punta el GPS del UAV [18]. El segundo parámetro es una trama de 12 bits conocido como TYPE_MASK, la cual indica que dimensiones de movimiento deben ser ignoradas. En la Tabla 2.7. se describen las configuraciones más comunes del TYPE_MASK.

Tabla 2.7. Valores comunes del TYPE_MASK para posición [18].

Valor en decimal	Descripción	Valor en binario
1	Ignora la posición en x	0b000000000001
8	Ignora la velocidad en x	0b000000001000
64	Ignora la aceleración en x	0b000001000000
1024	Ignora el yaw	0b010000000000
3576	Usa posición	0b110111111000
3527	Usa velocidad	0b110111000111
3128	Usa aceleración	0b11000011100
3520	Usa posición + velocidad	0b110111000000
2559	Toma en cuenta la posición del yaw	0b110000000000
1535	Usa la velocidad en yaw	0b010111111111

De esta manera teniendo en cuenta las diferentes configuraciones del TYPE_MASK descritos en la Tabla 2.7, se usa la máscara de numero decimal 3576, ya que permite controlar la posición del dron, en este caso que se quede en la misma posición durante un tiempo establecido. A continuación, se presenta la implementación de lo descrito en este párrafo en Python.

```
# LOITER GUIADO
the_connection.mav.send(mavutil.mavlink.MAVLink_set_position_target_local_ned
        _message(10,the_connection.target_system,the_connection.target_compon
        ent, mavutil.mavlink.MAV_FRAME_BODY_OFFSET_NED, 3527, 0, 0, 0, 0, 5,
        0, 0, 0, 0, 0, 0))
```

Como se observa en la Figura 2.20, una vez que ya culmino el Loiter guiado, se procede a realizar la maniobra de evasión. En este caso debido a que solo se tiene visión en la vista frontal del UAV, se realiza un movimiento a la derecha o izquierda de la aeronave, manteniendo el ángulo yaw de posición. Si el dron se mueve a la derecha o izquierda depende del valor de posición que se obtiene la etapa de detección de objetos. Para mover el dron a la derecha o izquierda se hace uso nuevamente del comando `MAV_FRAME_BODY_OFFSET_NED` modificado en modo posición + velocidad, ya que en este caso es necesario también modificar la velocidad con la que realiza la maniobra a una velocidad inferior con la que vuela en modo automático, esto con el fin de impedir que el tracker pierda de vista el objeto detectado y envié un falso positivo.

Para mantener la posición del yaw igual a la que tiene en el momento que inicia a realizar la maniobra de evasión el dron, se usa el comando `MAV_CMD_CONDITION_YAW` configurado de la siguiente manera.

```
# MATENEMOS EL YAW
the_connection.mav.command_long_send(the_connection.target_system,
        the_connection.target_component,mavutil.mavlink.MAV_CMD_CONDITION_YAW
        ,0, YAW, 0, 0, 0, 0, 0, 0)
```

Donde Yaw es una variable entera que guarda el valor de posición del ángulo yaw que tiene la aeronave antes de empezar la maniobra de evasión.

La lógica de movimiento lateral del UAV (izquierda, derecha) depende del valor de la posición obtenido en el sistema de detección de objetos, de tal manera que si este valor es negativo quiere decir que para evadir el objeto la ruta más corta a seguir es moviéndose a la derecha y de igual manera si este valor es positivo significa lo contrario. Lo descrito se ve con más detalle a continuación.

```
# MOVIMIENTO IZQUIERDA
if posicion >0:
the_connection.mav.send(mavutil.mavlink.MAVLink_set_position_target_local_ned
        _message(10,the_connection.target_system,the_connection.target_compon
        ent, mavutil.mavlink.MAV_FRAME_BODY_OFFSET_NED, 3527, 0, -2, 0, 0,
-0.5, 0, 0, 0, 0, 0, 0))
#MOVIMIENTO DERECHA
if posicion <=0:
the_connection.mav.send(mavutil.mavlink.MAVLink_set_position_target_local_ned
        _message(10,the_connection.target_system,the_connection.target_compon
        ent, mavutil.mavlink.MAV_FRAME_BODY_OFFSET_NED, 3527, 0, 2, 0, 0,
0.5,    0, 0, 0, 0, 0, 0))
```

Una vez que el objeto sale del campo de visión del UAV, este sale del modo guiado y retoma la misión programada con la ayuda del Mission Planner que se le indicio al inicio debía realizar.

En la actualidad, existe una maniobra de evasión de obstáculos muy usada en el mundo comercial, su funcionamiento se basa en modificar la trayectoria del dron a través de crear nuevos waypoints mientras este realiza la misión. Los resultados que brinda esta maniobra son muy precisos y eficientes, sin embargo, se ha descartado su implementación para este trabajo debido a la alta dificultad que presenta su desarrollo y a la alta capacidad de procesamiento que demanda. Por esto, a continuación, se hace una breve descripción de ella simplemente como una posible referencia a tener en cuenta para trabajos futuros.

Evasión modificando la trayectoria

Esta maniobra consiste en crear nuevos waypoints al momento que se detecta la presencia de un obstáculo dentro del rango de peligro. Para ello es necesario tener un conocimiento muy amplio del escenario donde se está llevando a cabo la misión, para con ello generar la ruta más óptima que debe tomar el dron para impedir la colisión. Esto se puede realizar a través de técnicas SLAM (Simultaneous Localization And Mapping) [8], que permiten que un robot móvil autónomo pueda operar en un escenario desconocido, utilizando únicamente sensores como una cámara o de distancia, que le permiten realizar una reconstrucción incremental del escenario en donde opera, este a su vez utiliza este mismo mapa reconstruido para determinar su localización.

2.7 FUSIÓN DE SEÑALES

Los datos obtenidos de la cámara y el sensor de distancia LIDAR Lite v3 se usan de manera complementaria, con el fin de obtener información suficiente del exterior para poder realizar una maniobra de evasión segura. En la Sección 1.4.1.4 se describe que a esta técnica se le conoce como Fusión Sensorial y que puede ser implementada de diferentes maneras,

dependiendo de lo que el usuario necesite. Para este trabajo de integración curricular se optó por implementar una red sensorial con una estructura híbrida entre una red paralela y una red iterativa, de tal forma que se pueda obtener los beneficios que ambas estructuras brindan. En la Figura 2.21 se observa esta red de fusión sensorial híbrida.

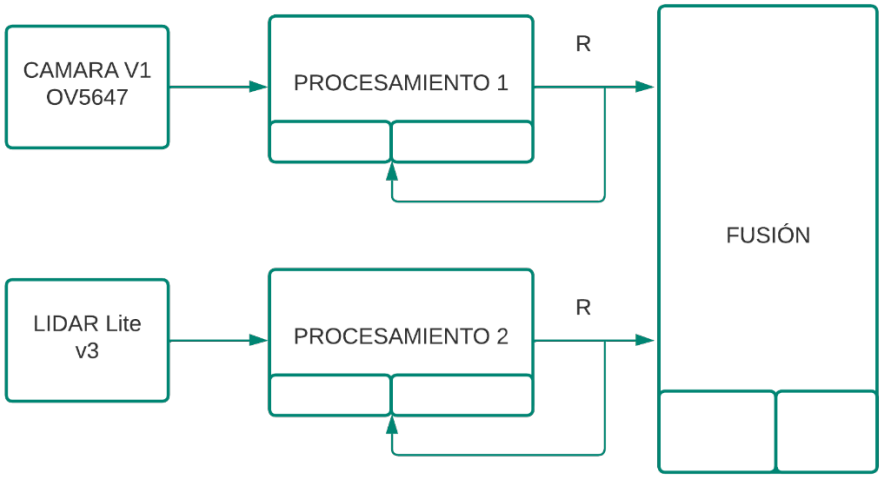


Figura 2.21 Estructura de la red sensorial implementada en este TIC [Fuente propia].

Una estructura de red paralela permite realizar el procesamiento de cada señal de manera separada por lo que facilita su desarrollo. Por otro lado, dado a que es necesario que el valor de la distancia y posición del objeto con respecto al dron este actualizándose de manera constante, para realizar operaciones como es el caso del tracker, se usó celdas de fusión en estructura iterativa, que cuentan con retroalimentación de la señal de salida.

Teniendo clara la estructura de la red de fusión sensorial a usar en este proyecto, se unió los datos obtenidos del detector de objetos, con el valor de distancia que da el LIDAR Lite v3, obteniendo así los datos necesarios para llevar a cabo la maniobra de evasión descrita en la sección anterior. El resultado de esta unión es el algoritmo completo del sistema auxiliar de evasión de objetos que se adjunta en el Anexo II.

3 RESULTADOS, CONCLUSIONES Y RECOMENDACIONES

3.1 PRUEBAS Y RESULTADOS

Para comprobar el correcto funcionamiento del prototipo auxiliar de evasión de obstáculos, se realizaron pruebas del sistema dividido en diferentes procesos. Primero, se comprueba el funcionamiento del detector de objetos, luego del seguidor y por último del sistema completo. Todas las pruebas se realizaron bajo un entorno controlado y sus resultados son respaldados por archivos CSV (Valores Separados por Comas) generados desde la

Raspberry y que registran datos como la posición, velocidad, distancia y el valor del yaw , a medida que se lleva a cabo la maniobra. También se obtienen resultados de cada prueba al analizar los Data Logs de vuelo que genera el controlador Pixhawk Cube, en estos archivos se almacena información de todas las variables de vuelo y los cambios que presentan, durante una misión de vuelo.

3.1.1 PRUEBAS DEL DETECTOR DE OBJETOS

La finalidad de esta prueba es verificar el funcionamiento del detector de obstáculos y la factibilidad del cambio de modo de vuelo a guiado, para posteriormente realizar el Loiter guiado explicado en la Sección 2.6.2 de este trabajo. Para ello, se inicia el vuelo del UAV de manera manual y se lo va acercando manualmente a un obstáculo presente en su entorno, cuando el sensor de distancia detecte un objeto a una distancia menor a 4 metros el modo de vuelo cambia a modo guiado y posteriormente hace un Loiter guiado, es decir mantiene la posición del último punto alcanzado antes de detectar el obstáculo.

El funcionamiento del filtro de color del detector se vio afectado debido a la variación de brillo del ambiente, en este caso las pruebas se realizaron en un día soleado y debido a eso los resultados obtenidos no fueron los deseados. Sin embargo, la librería PiCamera que es compatible con todas las cámaras de marca Raspberry brinda una forma de establecer el balance de blancos del fotograma a través de modificar su variable awb_mode con el siguiente comando:

```
# Modificar el balance de blancos.
Camera = (PiCamera.AWB_MODES ='mod')
# -----mod-----#
#'off','auto','sunlight','cloudy','shade','tungsten','fluorescent'
#'incandescent','flash','horizon'
```

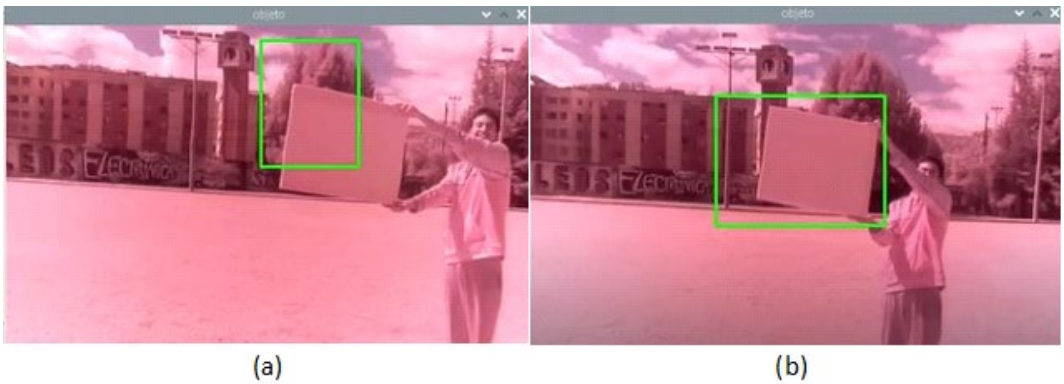


Figura 3.1 a) Resultado del detector de objetos sin modificar el balance de blancos b) Resultado del detector de objetos modificando el balance de blancos [Fuente propia].

En la Figura 3.1 se pueden ver las mejoras que se obtiene en el detector de objetos, al adaptar la cámara, a las condiciones de brillo del día. Esto también se lo puede hacer jugando con valores de los umbrales máximos y mínimos del filtro de color descrito en la Sección 2.5.1.2, pero resulta ser tedioso y poco práctico.

Para verificar el cambio de modo de vuelo a guiado y que, una vez detectado un objeto a menos de 4 metros, se proceda a hacer un Loiter guiado, se hizo uso de los datos obtenidos en los archivos CSV y en los Data logs de vuelo. Otro dato importante que es necesario tener en cuenta es la distancia a la que el sistema detecta la presencia de un objeto y a la distancia que el dron responde a los comandos de control. En la Figura 3.2 se puede apreciar encerrado en un cuadro azul, el valor de distancia a la que se detecta el objeto y resaltado de color amarillo el valor de distancia al que se detiene el UAV.

Figura 3.2 Respuesta del UAV al cambio de modo de vuelo.

Como se observa en la Figura 3.2 el obstáculo fue detectado a una distancia de 3.89 metros con respecto al UAV, mientras que este se detuvo, es decir paso a modo guiado e hizo el Loiter guiado, a una distancia de 3.62 metros del obstáculo. Para tener una idea más clara de cuanto le toma al UAV detenerse se repitió esta prueba bajo las mismas condiciones de vuelo (velocidad de vuelo del dron, altura, brillo de día) obteniendo los resultados que se presentan en la Tabla 3.1. En el Anexo III, se encuentran las gráficas de resultados de estas pruebas de donde se sacan los valores presentados en esta tabla.

Tabla 3.1. Respuesta de UAV al detector de obstáculos con una velocidad de vuelo de 10 [m/s].

Valor ideal [m]	Distancia detección [m]	Error detección [m]	Distancia de paro del UAV [m]	Error paro del UAV [m]
4	3.89	0.11	3.65	0.35
4	3.96	0.04	3.92	0.08
4	3.40	0.6	3.61	0.39

Con los datos obtenidos en las pruebas se nota que el detector de obstáculos tiene un error aceptable, ya que el dron responde a los comandos a una distancia segura de una posible colisión; sin embargo, hay que tener en cuenta que este error aumentara a medida que se aumente la velocidad de vuelo del UAV. Por ello, es recomendable tener en cuenta lo descrito en la Tabla 2.6 del capítulo anterior.

Por último, se presentan los gráficos obtenidos del archivo CSV y de los Data logs de vuelo. De estos archivos se corrobora que se realizó el cambio de modo de vuelo a guiado y que mientras este en este modo, el UAV haya mantenido su posición.

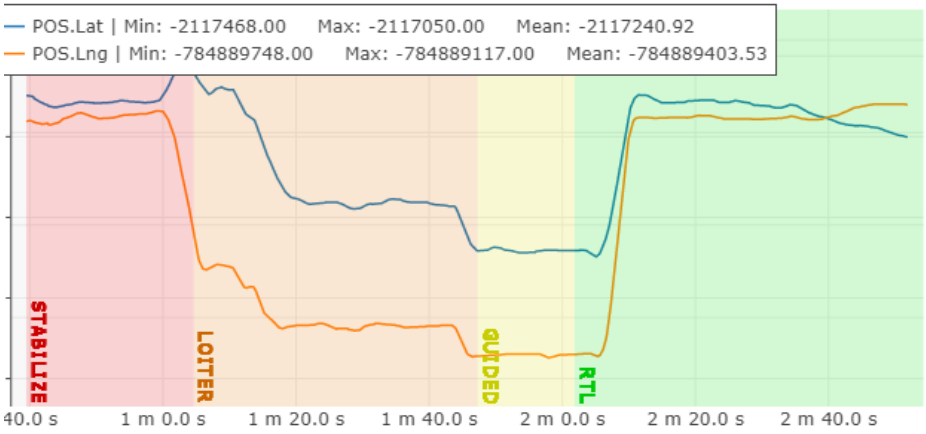


Figura 3.3 Latitud y Longitud del UAV obtenidos del Data log de vuelo

En la Figura 3.3 se observa los cambios de modo de vuelo y los valores de latitud y longitud que tuvo el UAV durante la prueba de detección. Se puede apreciar encerrado en color amarrillo el momento en que se realizó el cambio de modo de vuelo a guiado y que mientras el UAV estuvo en modo guiado tanto el valor de longitud y latitud tratan de mantenerse constantes. Por ello, se puede asegurar que el cambio de modo y la realización de un Loiter en modo guiado a través de código, son factibles y pueden ser usados para realizar posteriormente una maniobra de evasión.

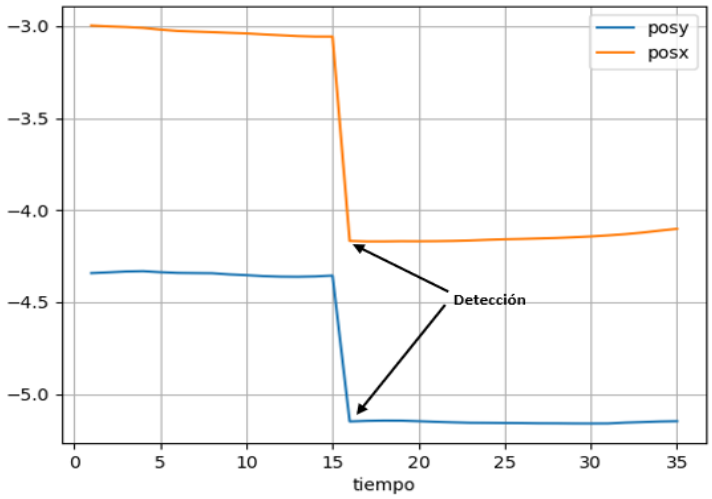


Figura 3.4 Posición X,Y vs tiempo

De igual manera en la Figura 3.4 se presenta como la posición en las coordenadas X como Y a lo largo del tiempo, trata de mantenerse constante una vez detectado el obstáculo

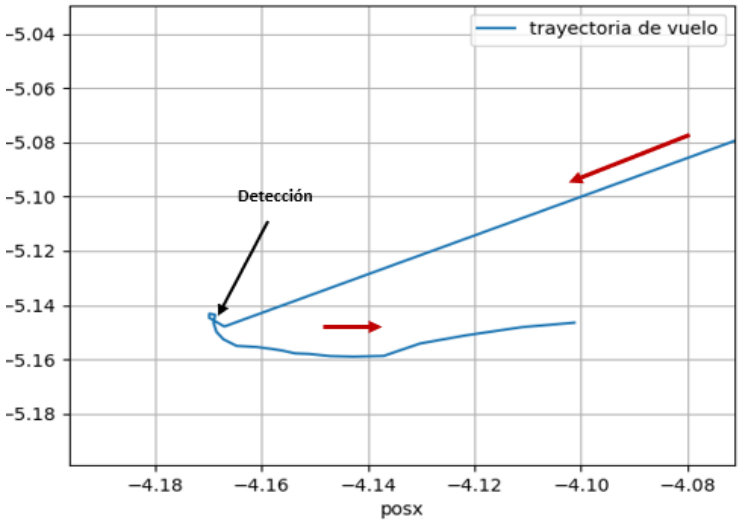


Figura 3.5 Posición X vs Posición Y. La línea roja indica la dirección de movimiento.

Por, último se presenta en la Figura 3.5 la gráfica de la posicion en X en función de la posición en Y. En esa imagen se observa como luego de la detección, la posición en X presenta una variación ligera en su valor, lo cual pudo darse debido a una corriente de viento que movió al UAV hacia al frente.

3.1.2 PRUEBAS DEL SEGUIDOR

Esta prueba se realizó con la finalidad de verificar el funcionamiento del seguidor o tracker y la factibilidad de que el dron realice un movimiento a la derecha o izquierda manteniendo

su valor de yaw constante mientras realiza este movimiento. Para ello, teniendo en cuenta que en la prueba anterior se verificó que el dron podía responder con facilidad a un objeto presente a 4 metros y que el cambio a modo guiado seguido de un Loiter funciona sin problemas, se llevó a cabo un plan de vuelo parecido al descrito en la prueba anterior, con la excepción de que ahora se reduce el umbral de detección a 3 metros y que luego del Loiter guiado el dron se moverá a la derecha hasta que pierda de vista al objeto presente en su trayectoria. Posterior a esto el dron se quedará quieto y esperará que el usuario le dé instrucciones de volver a su punto de partida.

Debido a que la visualización a través del comando `cv2.imshow` demanda un procesado alto y que esto afecta el tiempo de respuesta del UAV, para estas pruebas se decidió no utilizar el visualizador de cámara, sino más bien verificar el funcionamiento del tracker a través de imprimir los valores del Bounding box y como estos se actualizan a medida que el dron realiza el movimiento a la derecha. En la Figura 3.5 se pueden observar estos resultados.

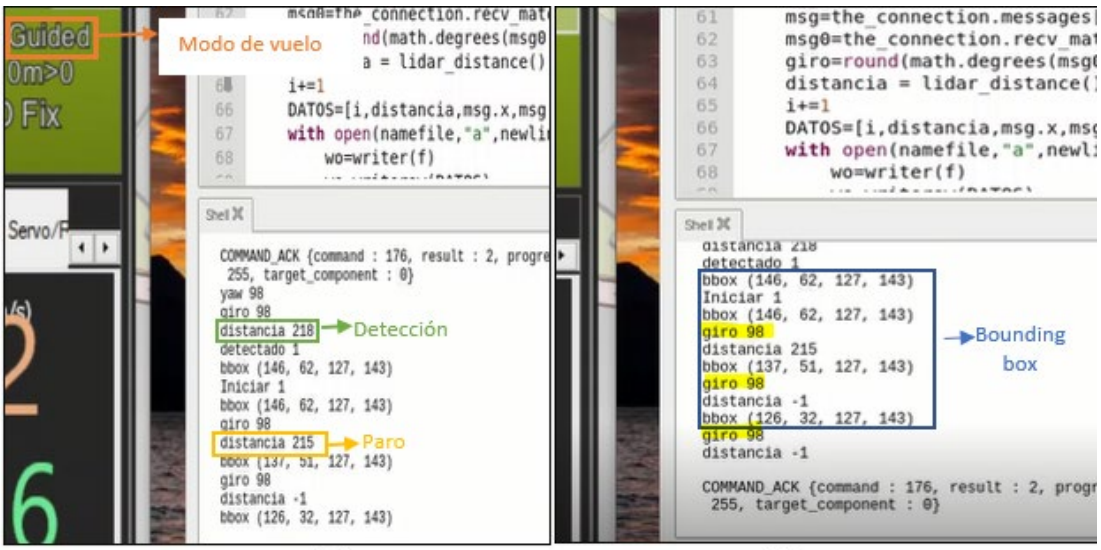


Figura 3.6 a) Valores de distancia y cambio de modo b) Actualización del Bounding box a medida que el dron se mueve a su derecha

En la Figura 3.6 (a) se puede observar encerrado en un cuadro tomate cómo una vez detectado un objeto a una distancia menor a 3 metros, el modo de vuelo pasa a guiado, en esta figura también se encierra en un cuadro de color amarillo la distancia a la que el dron se detiene, este dato se usa para analizar la velocidad de respuesta que tiene el prototipo una vez implementado el tracker. Posterior a eso, en la Figura 3.6 (b), se aprecia encerrado en un cuadro azul, como el bounding box se va actualizando a medida que el dron se mueve a la derecha .con el fin de que el objeto salga de su campo de visión. Por último,

para corroborar que durante la maniobra de evasión el valor del yaw se mantuvo constante se mandó a imprimir este valor, esto último se encuentra resaltado de color amarillo en la Figura 3.6 (b).

Para poder analizar cuanto varía la velocidad de respuesta del prototipo una vez implementado el tracker, se repitió esta prueba bajo las mismas condiciones de vuelo, obteniendo los resultados que se presentan en la Tabla 3.2. En el Anexo IV, se encuentran los resultados obtenidos de estas repeticiones de manera más detallada.

Tabla 3.2. Velocidad de respuesta de UAV una vez implementado el tracker y con una velocidad de vuelo de 10 [m/s].

Valor ideal [m]	Distancia a la que el dron se detiene [m]	Error de distancia [m]
3	2.15	0.85
3	1.87	1.13
3	2.08	0.92

Al analizar los errores obtenidos en la Tabla 3.2, los cuales resultan más altos que los calculados en la prueba anterior, se puede concluir que la implementación del tracker si redujo la velocidad de respuesta del prototipo auxiliar lo cual es malo, ya que se traduce en perdida de autonomía de vuelo.

Del archivo CSV obtenido durante la prueba se puede graficar como varia el valor de la posición en Y del dron, cuando se detecta un objeto a una distancia menor a 3 metros. Esta variación del valor de la posición en Y representan el movimiento a la derecha que realiza el dron para sacar de su campo de visión al obstáculo detectado.

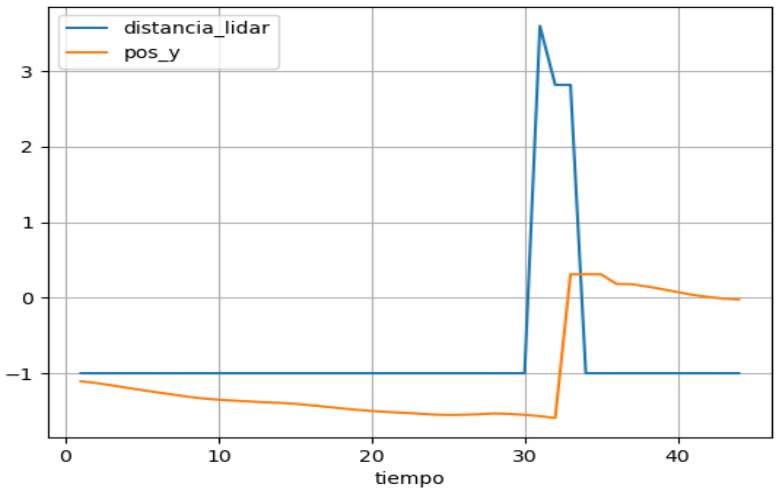


Figura 3.7 Distancia y posicion en Y del dron vs tiempo

La Figura 3.7 representa los valores de distancia y posición en Y obtenidos del archivo CSV, se puede observar como el Lidar toma valores de -1 en gran parte de su tiempo, esto es debido a que este valor representa una lectura fallida o una lectura de distancia por encima de los 40 metros. Por otro lado, se observa que la posición en Y trata de mantenerse constante, hasta que comienza a variar cuando se detecta un obstáculo a una distancia menor a 3 metros. Luego de eso, una vez finalizado el tracker, la posición en Y vuelve a tratar de mantenerse constante.

Del Data log de vuelo se puede obtener una imagen en 3D de la trayectoria de vuelo que llevo a cabo el UAV durante la prueba. En ella se puede visualizar de color amarillo como el UAV al estar en modo guiado, hace un movimiento a la derecha como se describió que haría con el fin de perder al obstáculo de vista, esto se presenta en la Figura 3.8.

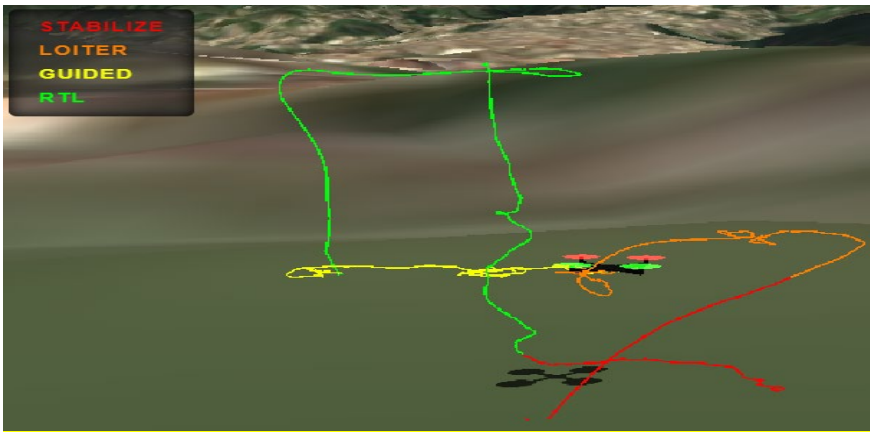


Figura 3.8 Trayectoria de vuelo del UAV durante la prueba del seguidor

Para observar lo descrito en el párrafo anterior con mayor facilidad, en la Figura 3.9 se presenta esta misma imagen, haciendo énfasis en la parte que se realiza la maniobra del movimiento a la derecha, vista desde dos perspectivas diferentes. La flecha azul indica la dirección de movimiento de la aeronave.

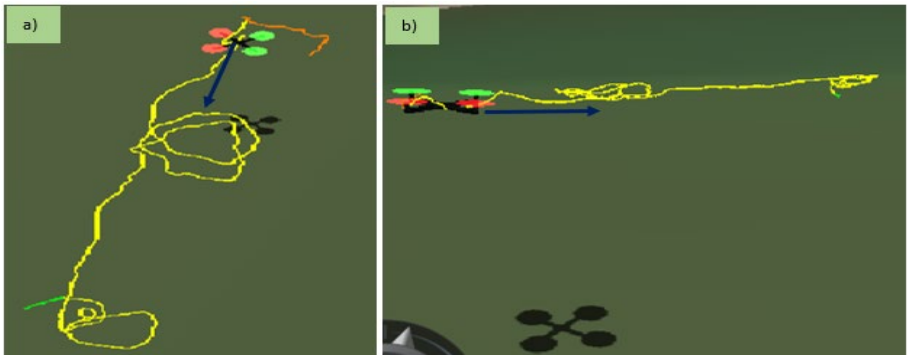


Figura 3.9 a)Vista superior de la maniobra b) Vista frontal de la maniobra

3.1.3 PRUEBAS DEL PROTOTIPO AUXILIAR DE EVASIÓN

Con esta prueba se corrobora el funcionamiento del sistema auxiliar de evasión de obstáculos diseñado e implementado en este trabajo de integración curricular. Para ello, se programa una misión de vuelo a través del Mission Planner y se lo carga al controlador de vuelo Pixhawk Cube haciendo uso del protocolo MAVLink.

El plan por llevar a cabo en esta prueba es que el dron comience y finalice la misión de vuelo preprograma en el Misión Planner de manera completamente automática, es decir sin que un usuario lo esté controlando a través de un mando. Además, solo cuando se haya detectado un objeto a una distancia menor a 3 metros, el modo de vuelo cambie de automático a guiado, para posteriormente hacer el Loiter guiado y moverse a la derecha o izquierda, tal y como se describió en la Sección 2.6.2, hasta que el objeto salga del campo de visión de la cámara. Una vez realizado esto, el dron retomara la misión de vuelo y permanecerá en ella hasta finalizarla o detectar otro obstáculo que pueda provocar una colisión.

Teniendo claro lo que se llevara a cabo, se inicia la misión planificada, lo que arma los motores del dron, por código, y luego inicia la misión. En esta parte fue necesario implementar un tiempo de espera de 4 segundos entre el armado de los motores del dron y el inicio de la misión, con el objetivo de que los motores tengan tiempo suficiente de alcanzar la velocidad que necesitan para estabilizarse. Esto es algo muy importante de tener en cuenta ya que si no se hace el dron perderá estabilidad y orientación durante el arranque.

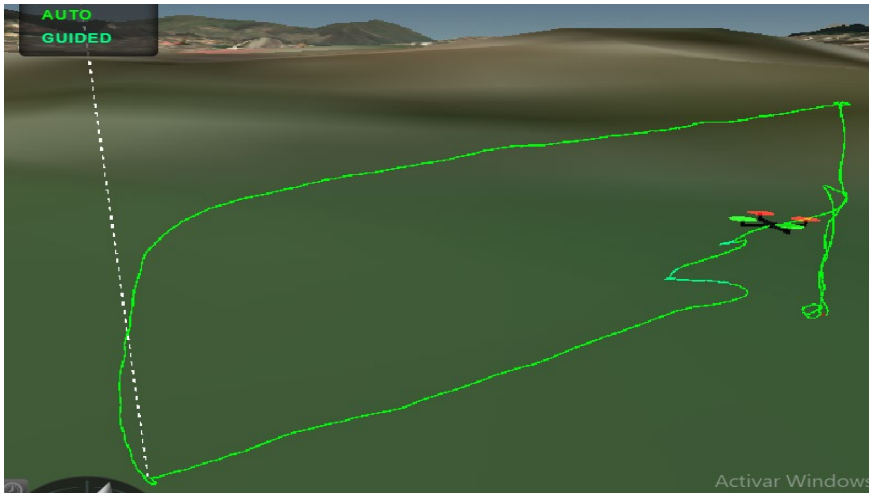


Figura 3.10 Trayectoria llevada a cabo por el UAV durante la prueba del prototipo

En la Figura 3.10 se observa como el dron lleva toda la misión en modo automático (parte verde) y que solo cuando detecta un obstáculo presente a valor menor al definido pasa a

guiado (parte turquesa) y realiza la maniobra de evasión. En este caso se le coloco dos veces un obstáculo a una distancia inferior a la descrita, por lo que el UAV entro dos veces a modo guiado, realizo la maniobra de evasión y posterior a eso vuelve a retomar la misión en modo automático. Esto se observa con mayor facilidad en la Figura 3.11.

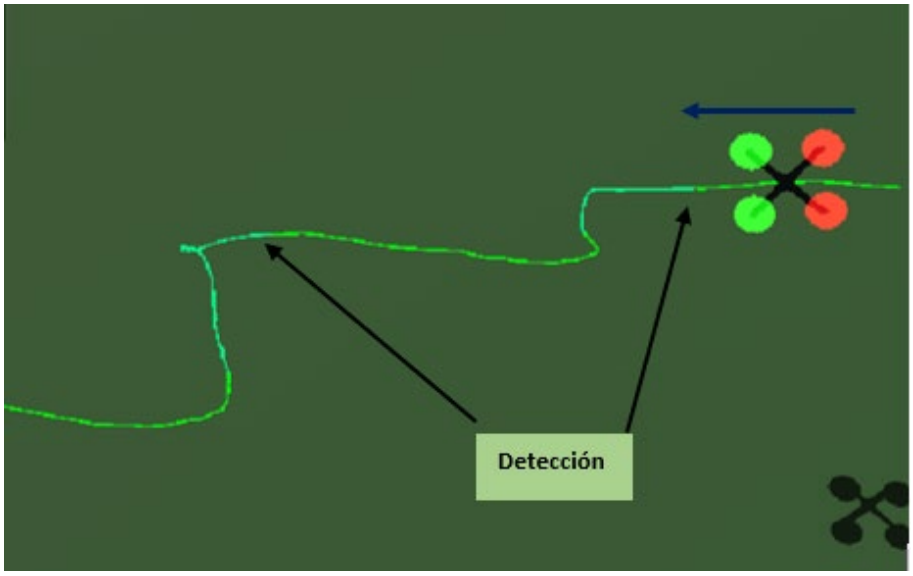


Figura 3.11 Vista superior de la maniobra de evasión de obstáculos, la flecha azul indica la dirección de movimiento de la aeronave

Una forma de apreciar de manera fácil como cambiaron los modos de vuelo durante la misión es visualizando los valores registrados de alguna variable de posicion. La variable que se escoja no tiene relevancia, ya que lo que interesa es revisar que los cambios de modo de vuelo se hayan realizado de manera adecuada.

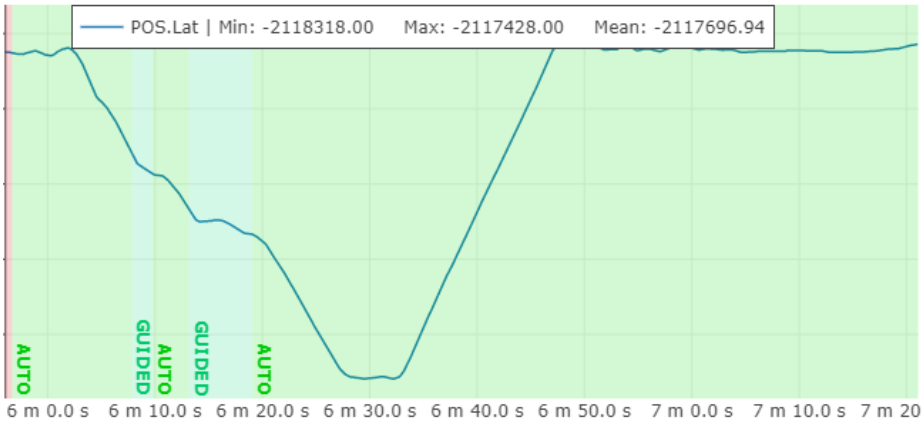


Figura 3.12 Grafica de los datos de latitud que registra el controlador durante la misión

Efectivamente en la Figura 3.12 podemos visualizar en color turquesa los momentos que el UAV paso a modo guiado y que posterior a eso retoma la misión en modo auto, esto último representado en color verde.

Del archivo CSV que se obtiene al registrar los datos de cada vuelo, también se puede obtener una gráfica en 3D de la trayectoria que llevo a cabo el dron durante la misión, al graficar los valores de posición en las coordenadas XYZ. Esto se observa en la Figura 3.13.

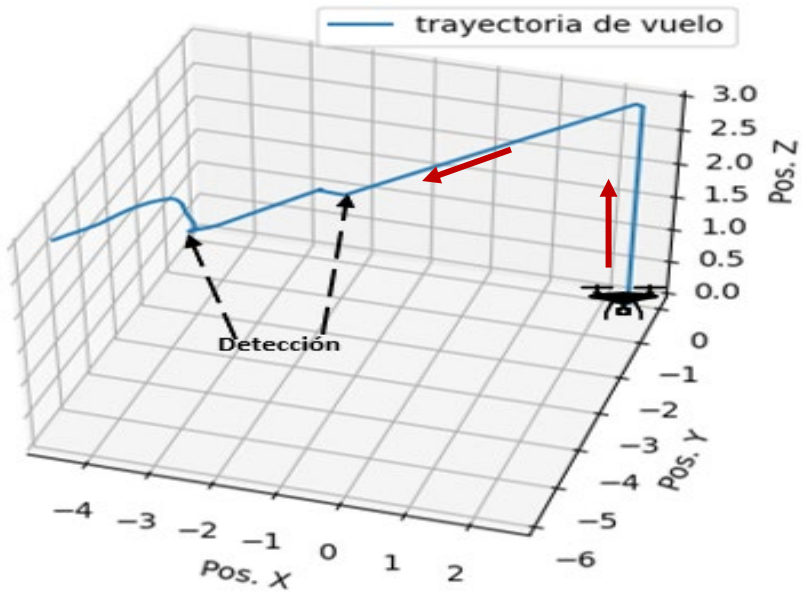


Figura 3.13 Trayectoria de vuelo obtenida de los datos del archivo CSV, las flechas rojas indican la dirección de movimiento de la aeronave

Al igual que se analizó la velocidad de respuesta en los procesos anteriores, se hace para esta prueba, ya que esta abarca la unión de las dos anteriores más la parte de control de vuelo completamente autónomo. Para ello se hace uso del archivo CSV, donde se registra el valor de distancia que mide el Lidar Lite v3. En la Tabla 3.3 se observan estos resultados para tres pruebas distintas en iguales condiciones de vuelo. Al igual que los anteriores casos, se puede encontrar en el Anexo V los resultados obtenidos de estas pruebas de manera más detallada.

Tabla 3.3. Velocidad de respuesta de UAV una vez implementado el prototipo completo y con una velocidad de vuelo de 10 [m/s].

Valor ideal [m]	Distancia a la que el dron se detiene [m]	Error de distancia [m]
3	2.18	0.82
3	1.98	1.02
3	1.12	1,88

Al analizar los datos obtenidos de las diferentes pruebas, se observa que la diferencia en la velocidad de respuesta del dron no varía significativamente con respecto a la prueba anterior, esto, con excepción de la última prueba en donde sí se ve un error casi del doble.

Esto se pudo dar debido a una fuerte corriente de viento que movió al dron en ese momento, provocando así ese incremento de error en posición.

3.2 CONCLUSIONES

El implementar un sistema auxiliar de evasión de obstáculos en un UAV, impide que esta sufra colisiones que le ocasionen daños irreparables a su estructura. En el mercado existen varios prototipos muy eficientes que realizan esta misión, sin embargo, sus precios suelen ser altos, por lo que no resulta rentable su uso para proyecto de investigación.

Al momento de seleccionar los elementos que formaran parte de un prototipo auxiliar que se desea adaptar en un UAV, existen características como la carga útil y la autonomía del dron que deben ser tomadas en cuenta, esto debido a que, si el prototipo auxiliar resulta tener un peso mayor a la carga útil, su estructura puede deformarse o quebrarse mientras el UAV este en vuelo o al momento del despegue.

La forma más adecuada para realizar una maniobra de evasión de obstáculos, al estar usando un controlador de vuelo Pixhawk, es a través del modo de vuelo guiado, ya que de esta forma se puede tener mayor control de la maniobra que se desea realice el UAV para impedir la colisión. La complejidad de esta dependerá del nivel de autonomía que se desee obtener, sin embargo, esto también demanda más capacidad de procesamiento, por lo que el costo de su implementación resulta más alto.

La visión artificial gracias al desarrollo del aprendizaje profundo cada vez brinda mejores resultados en la detección de objetos, sin embargo, para su implementación se necesita de un alto sistema de procesamiento. No obstante, una forma de bajo costo de procesado de detectar objetos a través de la visión artificial es el uso de operaciones morfológicas con las cuales se puede determinar características importantes de un objeto, como el color, bordes, forma, entre otros, y con ellas separarlo del fondo con el fin de poder determinar su posición en el fotograma.

Al momento de implementar un sistema auxiliar de evasión de obstáculos, es necesario tener en cuenta el tiempo que le toma en procesar toda la información a la tarjeta de procesamiento que se esté utilizando, ya que esto será útil para determinar la velocidad adecuada de vuelo que debe llevar el UAV, con el fin de que actúe a tiempo a la presencia de un objeto y así no colisione con el mismo.

Al realizar las pruebas se pudo concluir que factores ambientales, como una fuerte briza, o un el brillo del día puede afectar los resultados que se espera obtener del prototipo auxiliar

implementado. Sin embargo, estos errores se pueden reducir teniendo en consideración estos inconvenientes desde un inicio, como por ejemplo adaptar la cámara al brillo del día para que el detector de mejores resultados, o aumentando el umbral de distancia mínimo si se observa que es un día con fuertes ventiscas.

Al implementar el tracker al sistema de detección de objetos, se aumenta la robustez de este. No obstante, esto a su vez demanda mayor procesamiento de código, lo que disminuye de forma considerable, la velocidad de respuesta del prototipo auxiliar, obligando a que se considere reducir la velocidad de vuelo del dron o a su vez buscar una tarjeta de procesamiento con mayor capacidad.

3.3 RECOMENDACIONES

Se recomienda como un posible trabajo de maestría realizar un sistema de evasión de objetos, basado en técnicas SLAM, las cuales brindan un sistema más robusto, autónomo y preciso, al momento de evadir objetos. Para ello se debería remplazar la Raspberry Pi 4 usada en este TIC, por una GPU Jetson Nano NVIDIA.

Al momento de realizar una evasión en modo guiado, se recomienda tener claro los conceptos de los diferentes modos de vuelo y de cómo estos actúan sobre el controlador Pixhawk, ya que de esta forma se impide que durante el vuelo se pierda el control del UAV por errores de código.

Se recomienda que, al momento de realizar una misión completamente automática por código, se verifique el modo de vuelo en el que se encuentra el dron, ya que, si este no está en modo estabilizado o guiado, los motores no se armaran. Esta es una condición de seguridad de fábrica que viene activa en los controladores

4 REFERENCIAS BIBLIOGRÁFICAS

[1] L. Schroth, H. Bödecker y M. Radovic, “The Drone Market Report 2020-2025”, Hamburgo, Abril de 2020.

[2] A. M. Mera Chamorro and D. A. Ruano González, “Implementación de un robot de navegación autónoma terrestre para evasión de obstáculos usando el dispositivo Pixhawk,” Tesis de pregrado, Universidad de la Américas, Quito, 2019.

[3] “The Cube Overview — Copter documentation,” Accessed: Nov. 27, 2021. *ardupilot.org*. Available: https://ardupilot.org/copter/docs/common-thecube-overview.html.

[4] A. J. Carrasco Paredes, “Propuesta de un sistema de evasión de obstáculos para un dron aplicado a la inspección en redes de distribución y transmisión para la Empresa Eléctrica Riobamba S.A.,” Tesis de pregrado, Escuela Superior Politécnica de Chimborazo, Riobamba, 2018.

[5] S. O. Cayo Guamushig and I. D. Changoluisa Caiza, “Integración y automatización de un sistema de seguimiento de un UAV para establecer un enlace de comunicación con una estación de monitoreo en tierra,” Tesis de pregrado, Escuela Politécnica Nacional, Quito, 2018.

[6] Á. Bustillo de la Cruz, “Gestión autónoma de misiones en UAV basada en visión artificial,” Tesis de pregrado, Universidad Carlos III de Madrid., 2021.

[7] "Introduction · MAVLink Developer Guide", *Mavlink.io*, 2022. [Online]. Available:https://mavlink.io/en/?fbclid=IwAR2DyxLoUXhvYTi5eOoAOTYyKYdvl_q0RmrN1b42vh43TTBhWAPp3k-N. [Accessed: 04- May- 2022].

[8] J. García Merino, “Sistema avanzado de detección de obstáculos y navegación autónoma para vigilancia y protección basado en flota de vehículos aéreos no tripulados,” Tesis Doctoral, Universidad de Málaga, 2016.

[9] "DJI GS Pro - DJI", *DJI Official*, 2022. [Online]. Available: https://www.dji.com/ground-station-pro. [Accessed: 04- May- 2022].

[10] "Long Range TOF Sensor | Distance Sensor | Level Sensing", *Terabee*, 2022. [Online]. Available: https://www.terabee.com/shop/lidar-tof-range-finders/teraranger-evo-60m/. [Accessed: 09- May- 2022].

[11] "Pensar - Dual spectrum computer vision platform", *Aerialtronics Pensar*, 2022. [Online]. Available: https://aerialtronics.com/en/products/pensar. [Accessed: 09- May- 2022].

[12] L. Morcillo Martínez, "Sistema de detección de obstáculos para drones basado en sensor láser", Tesis de pregrado, Universidad Politécnica de Valencia, 2018.

[13] B. G. Castelo Pichucho and B. G. Mosquera López, “Diseño e implementación de un sistema para la detección y seguimiento de una persona a través de un cuadricóptero de tamaño reducido empleando visión artificial,” Tesis de pregrado, Escuela Politécnica Nacional, Quito,2022.

[14] A. J. Oña Oña, “Sistema de clasificación de granos de cacao frescos basados en visión computacional,” Tesis de pregrado, Escuela Politécnica Nacional, Quito, 2020.

[15] J. I. Córdova Sánchez, “Identificación y evasión de obstáculos mediante fusión sensorial para robots móviles,” Tesis de pregrado, Universidad del Valle, Cali, 2015.

[16] "Mission Planner Home — Mission Planner documentation", *Ardupilot.org*, 2022. [Online]. Available: https://ardupilot.org/planner/. [Accessed: 15- May- 2022].

[17] "Learning ArduPilot — Introduction — Dev documentation", *Ardupilot.org*, 2022. [Online]. Available: https://ardupilot.org/dev/docs/learning-ardupilot-introduction.html. [Accessed: 15- May- 2022].

[18] "MAVLink Interface — Dev documentation", *Ardupilot.org*, 2022. [Online]. Available: https://ardupilot.org/dev/docs/mavlink-commands.html. [Accessed: 15- May- 2022].

[19] "RFD900 Radio Modem — Copter documentation", *Ardupilot.org*, 2022. [Online]. Available: https://ardupilot.org/copter/docs/common-rfd900.html. [Accessed: 15- May- 2022].

[20] "FlySky FS-iA10B 2.4Ghz AFHDS 10CH Receiver, PPM, Telemetry", *Flying Tech*, 2022. [Online]. Available: https://www.flyingtech.co.uk/electronics/flysky-fs-ia10b-24ghz-afhds-10ch-receiver-ppm-telemetry. [Accessed: 15- May- 2022].

[21] "Chasis Para Cuadricóptero F450", *VISTRONICA S.A.S*, 2022. [Online]. Available: https://www.vistronica.com/robotica/chasis-para-cuadricoptero-f450-detail.html. [Accessed: 24- May- 2022].

[22] “A2212/13T TECHNICAL DATA.” Accessed: May 24, *24AD*. [Online]. Available: https://components101.com/sites/default/files/component_datasheet/A2212%20Brushless%20Motor%20Datasheet.pdf

[23] R. Ltd, "Buy a Raspberry Pi 4 Model B – Raspberry Pi", *Raspberry Pi*, 2022. [Online]. Available: https://www.raspberrypi.com/products/raspberry-pi-4-model-b/. [Accessed: 24- May- 2022].

[24] R. Ltd, "Raspberry Pi 4 Model B specifications – Raspberry Pi", *Raspberry Pi*, 2022. [Online]. Available: https://www.raspberrypi.com/products/raspberry-pi-4-model-b/specifications/. [Accessed: 24- May- 2022].

[25] "RunCam 2", *RunCam Store*, 2022. [Online]. Available: https://shop.runcam.com/runcam2/. [Accessed: 02- Jun- 2022].

[26] "PL-D7718 | Pixelink", *Pixelink*, 2022. [Online]. Available: https://pixelink.com/products/industrial-cameras/usb-30/123-sensors/pl-d7718/. [Accessed: 02- Jun- 2022].

[27] "Raspberry Pi Documentation - Camera", *Raspberrypi.com*, 2022. [Online]. Available: https://www.raspberrypi.com/documentation/accessories/camera.html. [Accessed: 02- Jun- 2022].

[28] "SRF10 MINI SENSOR DISTANCIAS ULTRASONIDOS I2C", *Superrobotica.com*, 2022. [Online]. Available: http://www.superrobotica.com/s320114.htm. [Accessed: 02- Jun- 2022].

[29] G. subsidiaries, "Garmin LIDAR-Lite v3 | GPS Sensors", *Garmin*, 2022. [Online]. Available: https://www.garmin.com/en-US/p/557294. [Accessed: 02- Jun- 2022].

[30] "Long Range TOF Sensor | Distance Sensor | Level Sensing", *Terabee*, 2022. [Online]. Available: https://www.terabee.com/shop/lidar-tof-range-finders/teraranger-evo-60m/. [Accessed: 02- Jun- 2022].

[31] T. Huang, "RPLIDAR-A1 Laser Range Scanner Parameters|SLAMTEC", *Slamtec.com*, 2022. [Online]. Available: https://www.slamtec.com/en/Lidar/A1Spec. [Accessed: 02- Jun- 2022].

[32] "OpenCV: OpenCV modules", *Docs.opencv.org*, 2022. [Online]. Available: https://docs.opencv.org/4.5.5/. [Accessed: 10- Jun- 2022].

[33] "Calibración de la cámara Opencv - ▷ Cursos de Programación de 0 a Experto © Garantizados", *▷ Cursos de Programación de 0 a Experto © Garantizados*, 2022. [Online]. Available: https://unipython.com/calibracion-la-camara-opencv/. [Accessed: 10- Jun- 2022].

[34] "Detección de colores en OpenCV - Python (En 4 pasos) » omes-va.com", *OMES*, 2022. [Online]. Available: https://omes-va.com/deteccion-de-colores/. [Accessed: 10- Jun- 2022].

[35] "Detector de bordes Canny, cómo contar objetos con OpenCV y Python", *Programar fácil con Arduino*, 2022. [Online]. Available: https://programarfacil.com/blog/vision-artificial/detector-de-bordes-canny-opencv/. [Accessed: 12- Jun- 2022].

[36] Z. Kalal, K. Mikolajczyk, y J. Matas, “Tracking-Learning-Detection”, *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 34, num. 7, pp. 1409-1422, Jul. 2012, doi: 10.1109/TPAMI.2011.239.

[37] "GitHub - adafruit/ Adafruit_CircuitPython_LIDARLite: Circuit Python library for Garmin LIDAR Lite sensor", *GitHub*, 2022. [Online]. Available: https://github.com/adafruit/Adafruit_CircuitPython_LIDARLite. [Accessed: 18- Jun- 2022].

[38] "DETECCIÓN DE COLORES Y Tracking en OpenCV – Parte2 » omes-va.com", *OMES*, 2022. [Online]. Available: https://omes-va.com/deteccion-de-colores2/. [Accessed: 18- Jun- 2022].

[39] A. Koubâa *et al.,* " Micro Air Vehicle Link (MAVLink) in a Nutshell: A Survey", en *CISTER-TR-190701,* vol. 4, 2019. DOI: 10.1109/ACCESS.2022.2924410

[40] Diebel, «Representing attitude: Euler angles, unit quaternions, and rotation vectors,» Matrix, vol. 58, pp. 15-16, 2006.

5 ANEXOS

ANEXO I. Esquema de conexión del prototipo

ANEXO II. Diagrama de flujo del sistema auxiliar de evasión de obstáculos

ANEXO III. Pruebas del detector de obstáculos

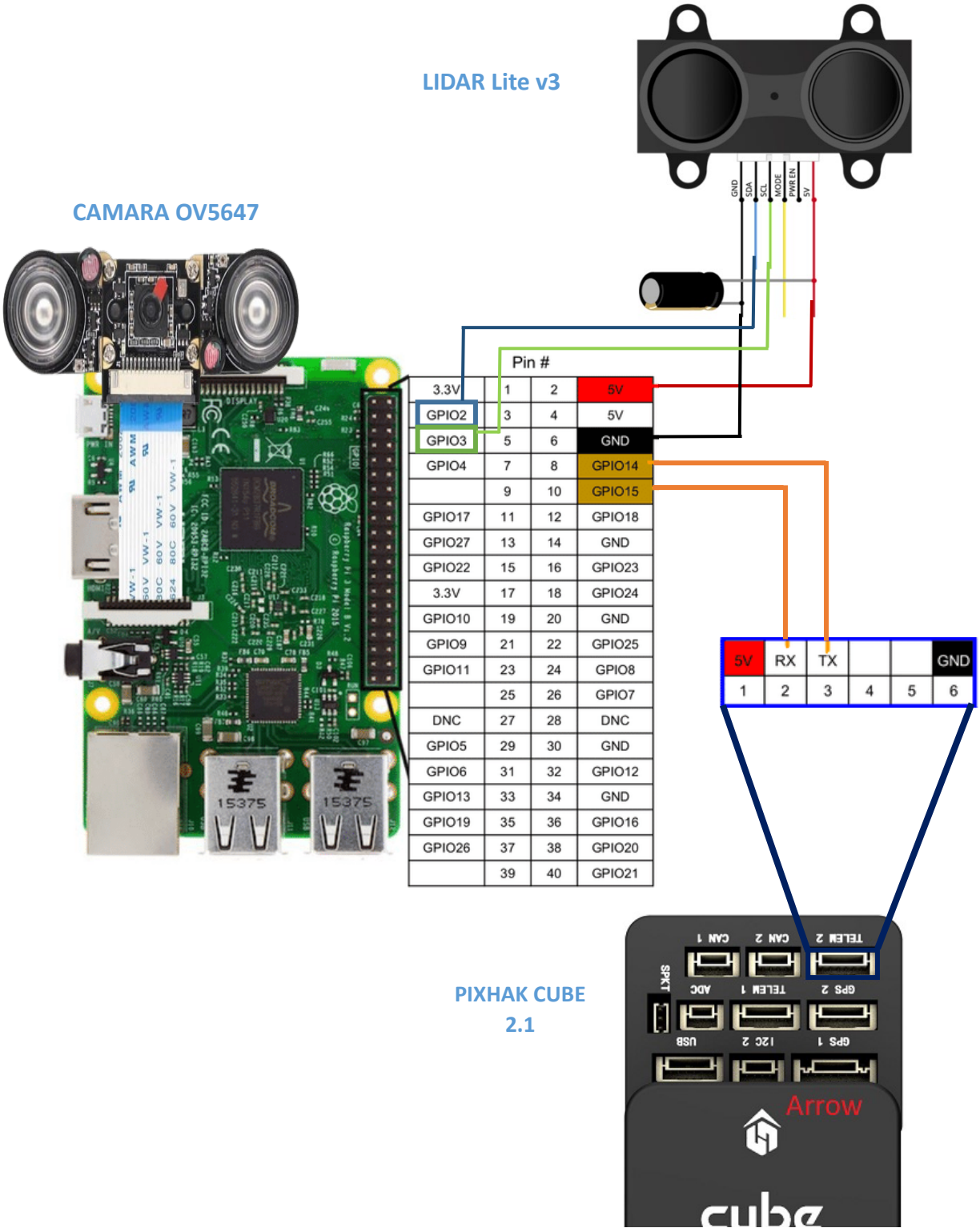
ANEXO IV. Pruebas del Tracker

ANEXO V. Pruebas del Prototipo auxiliar de evasión

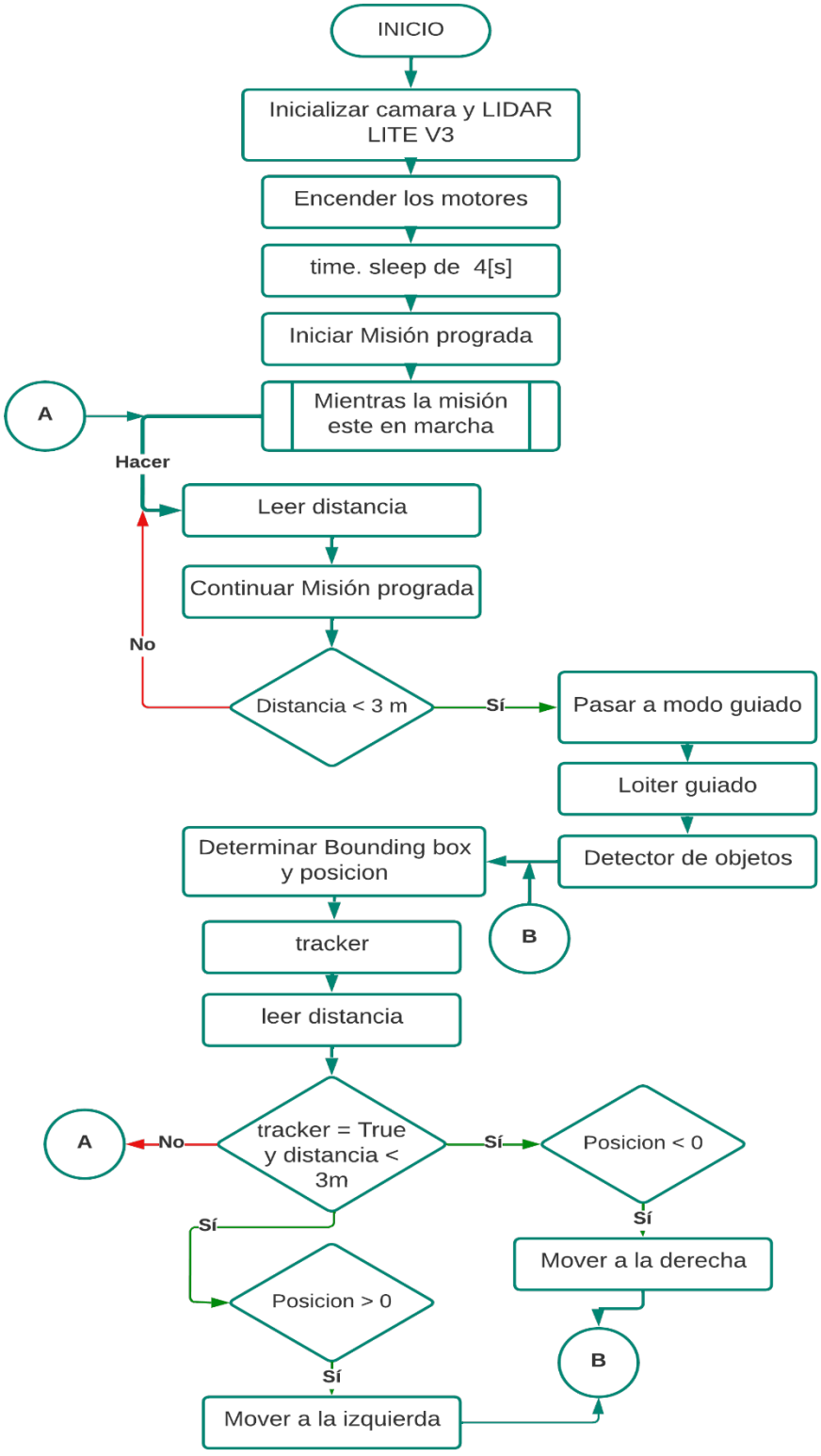
ANEXO VI. Enlace a videos demostrativos

ANEXO VII. Manual de usuario

ANEXO I. ESQUEMA DE CONEXIÓN DEL PROTOTIPO



ANEXO II. DIAGRAMA DE FLUJO DEL SISTEMA AUXILIAR DE EVASIÓN DE OBSTÁCULOS



ANEXO III. PRUEBAS DEL DETECTOR DE OBSTÁCULOS

En este anexo se presentan los resultados obtenidos al haber realizado pruebas del detector de obstáculos, vale aclarar que para estas pruebas ya se adaptó la cámara al brillo del entorno y que igual como se describió en la Sección 3.1.1, estos datos fueron obtenidos de los Data logs de vuelo y de los archivos CSV que se crean durante la misión.

PRUEBA 1:

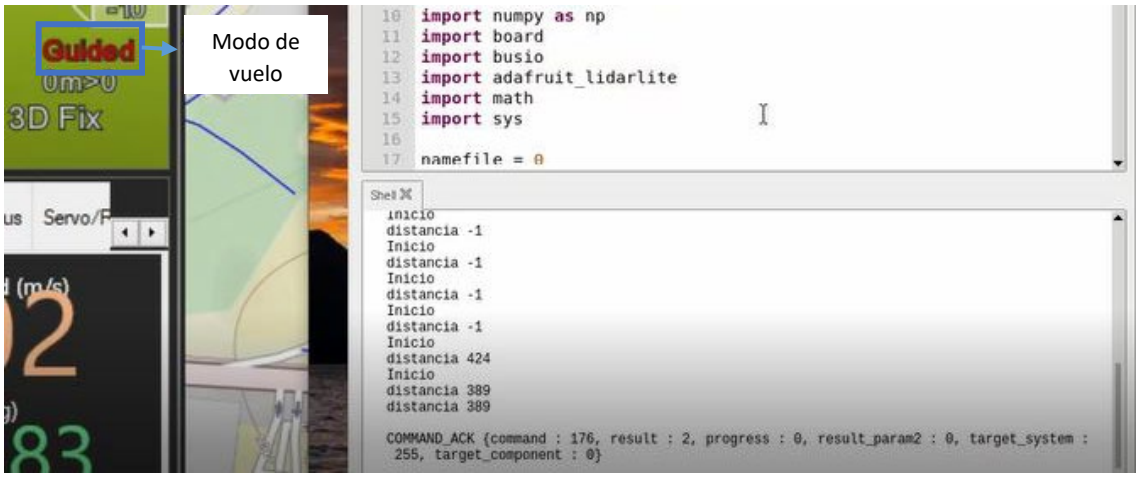


Figura III.1 Registro del cambio de modo a guiado y distancia de detección del obstáculo, prueba 1

Figura III.2 Respuesta del detector de obstáculos para la prueba 1

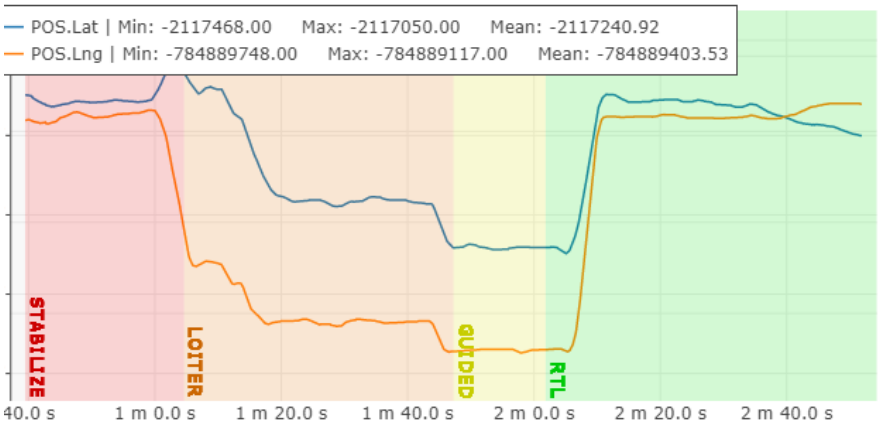


Figura III.3 Latitud y Longitud registrada durante la prueba 1 del detector

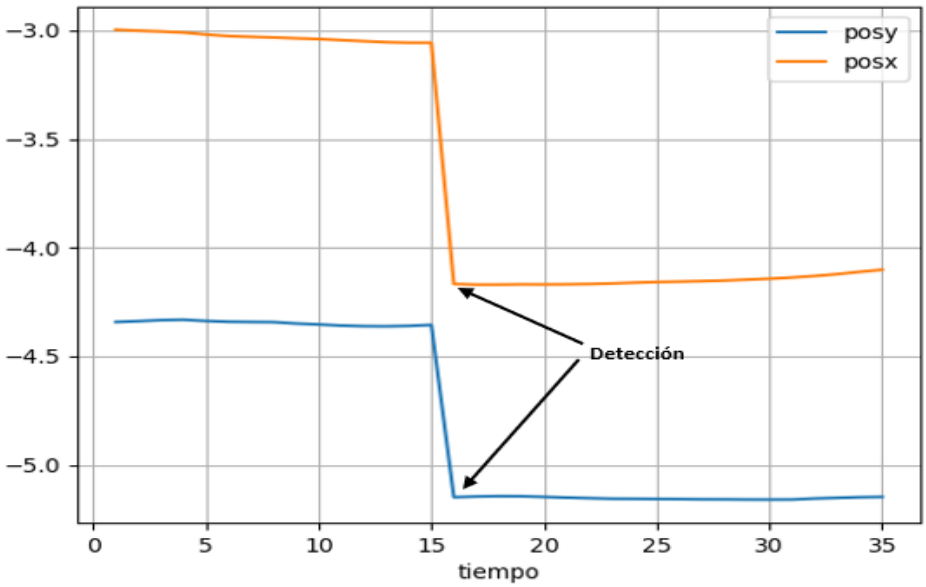


Figura III.4 Posicion XY vs el tiempo, prueba 1

PRUEBA 2:

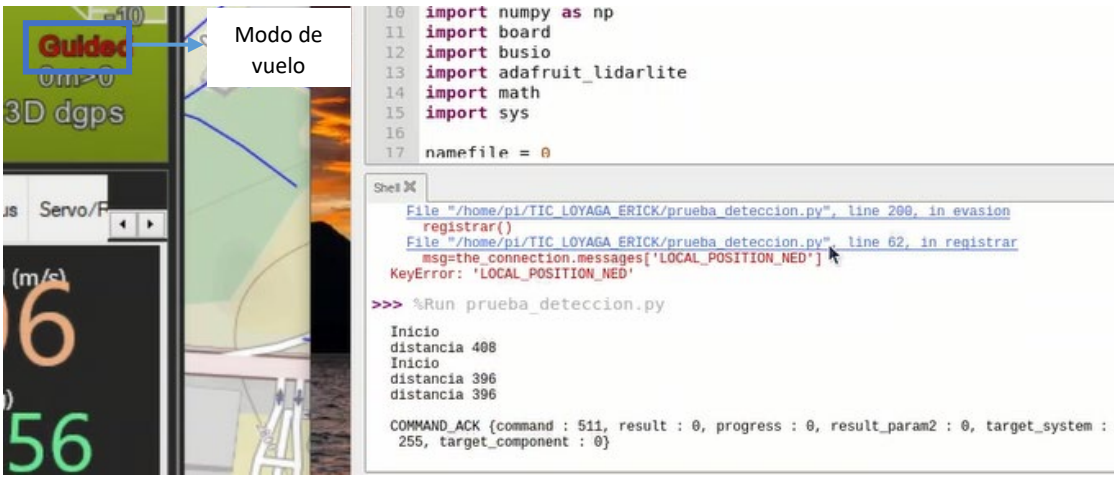


Figura III.5 Registro del cambio de modo a guiado y distancia de detección en la prueba 2

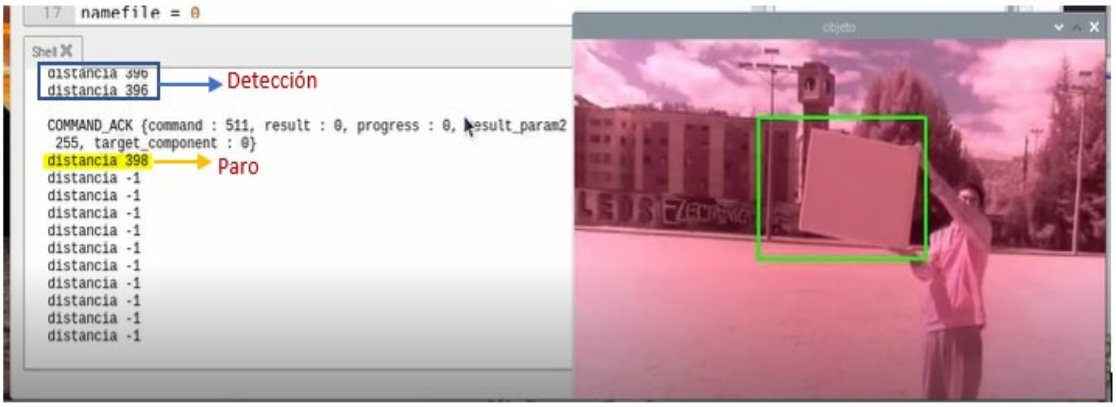


Figura III.6 Respuesta del detector de obstáculos para la prueba 2

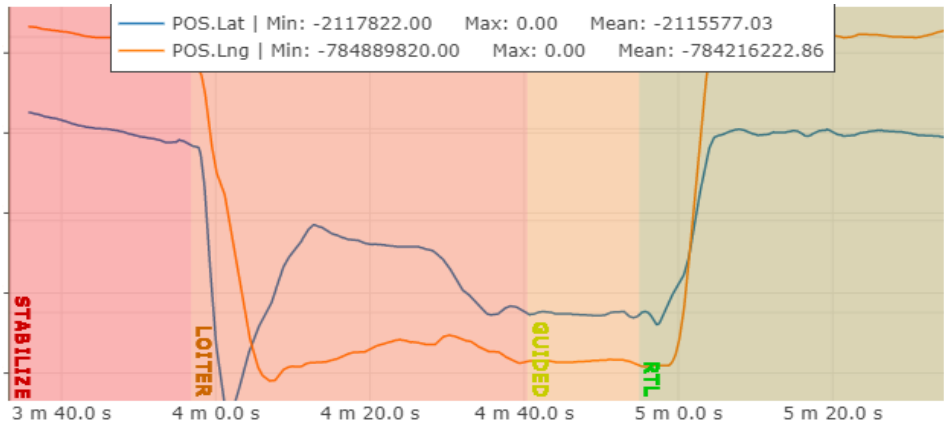


Figura III.7 Latitud y longitud registradas en la prueba 2

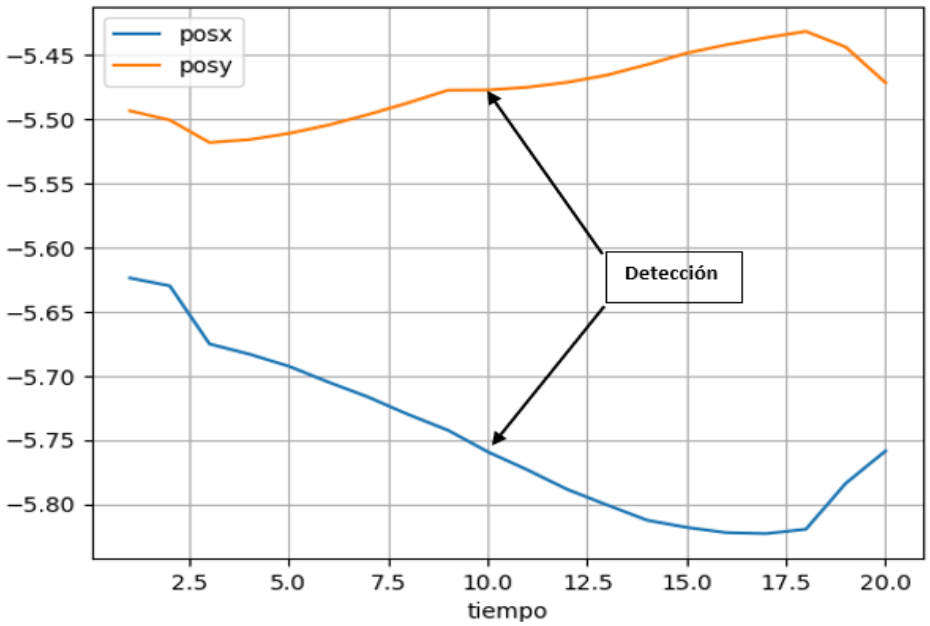


Figura III.8 Posicion XY vs el tiempo para la prueba 2

En la Figura III.8 observamos que luego de la detección el dron tiene un movimiento como curvado hacia adelante, esto se pudo dar debido a una fuerte corriente de viento que movió

al UAV de su posicion de reposo, sin embargo, vemos como luego intenta volver a colocarse en el punto donde se le ordeno que se quede.

PRUEBA 3:

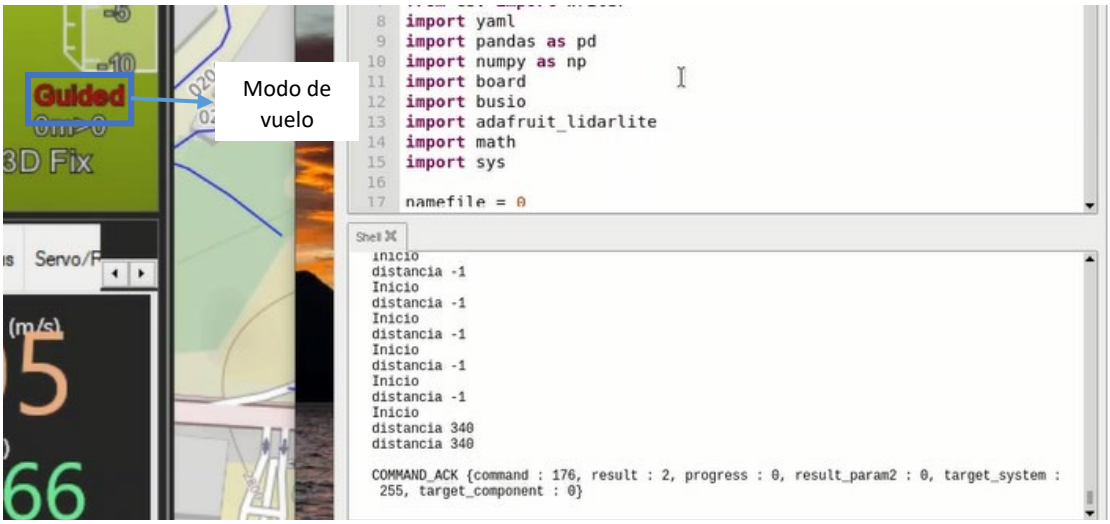


Figura III.9 Registro del cambio de modo a guiado y distancia de detección en la prueba 3

Figura III.10 Respuesta del detector de obstáculos para la prueba 3

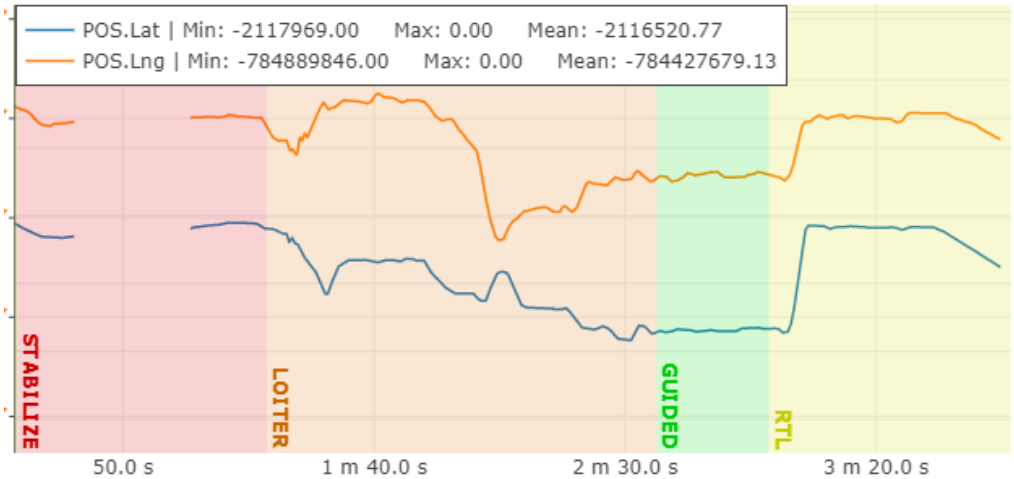


Figura III.11 Latitud y longitud registradas en la prueba 3

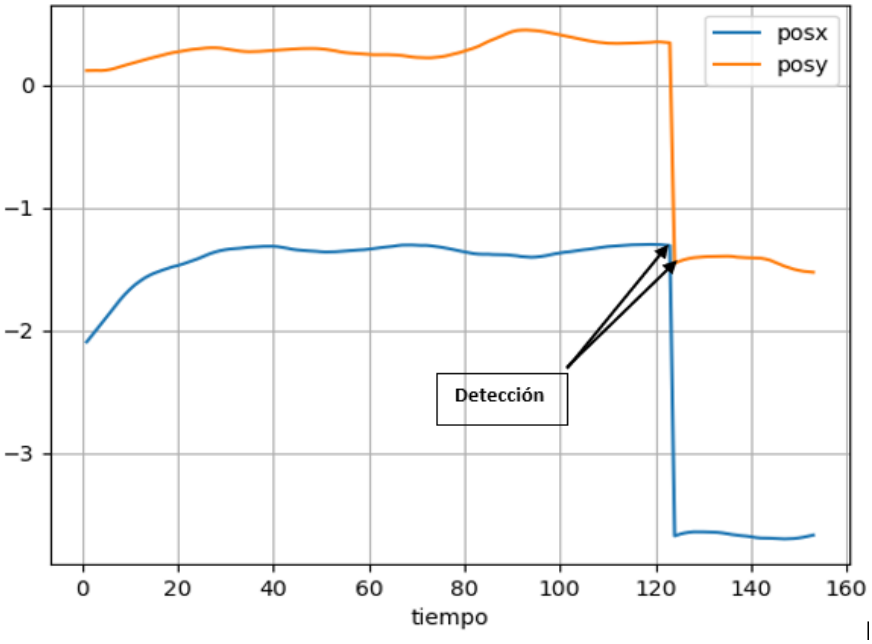


Figura III.12 Posicion XY vs el Tiempo para la prueba 3

ANEXO IV. PRUEBAS DEL TRACKER

En este anexo se presentan los resultados obtenidos al haber realizado pruebas del detector de obstáculos, vale aclarar que para estas pruebas ya se adaptó la cámara al brillo del entorno y que igual como se describió en la Sección 3.1.2, estos datos fueron obtenidos de los Data logs de vuelo y de los archivos CSV que se crean durante la misión.

PRUEBA 1:

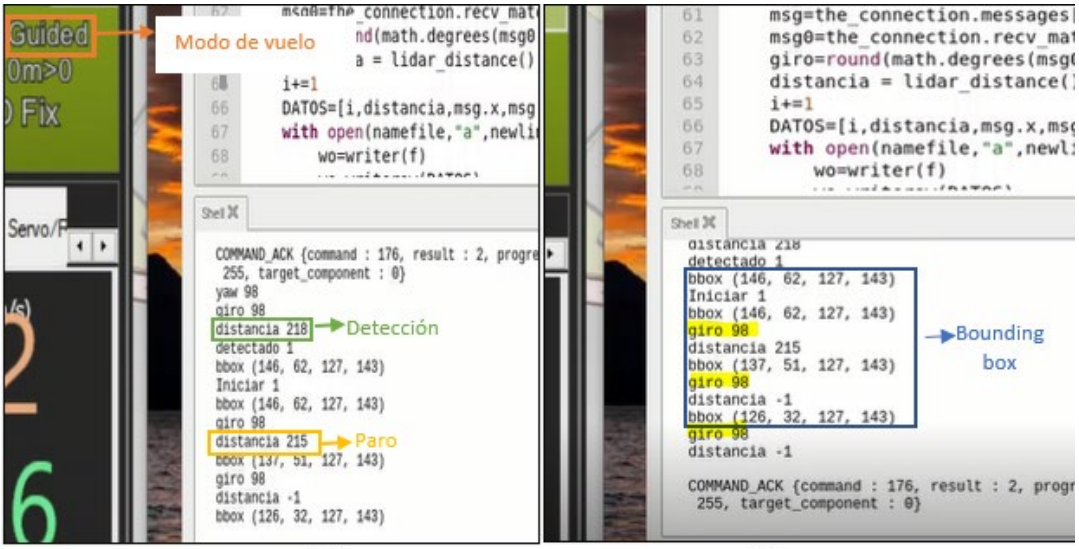


Figura IV.1 Actualización de los datos del Bounding Box para la prueba 1 del tracker

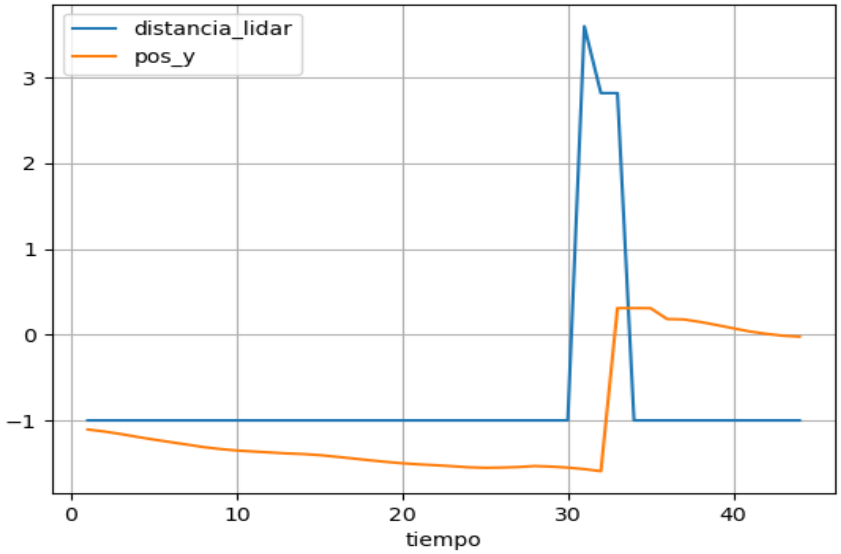


Figura IV.2 Registro de la distancia y posicion Y vs el tiempo para la prueba 1 del tracker

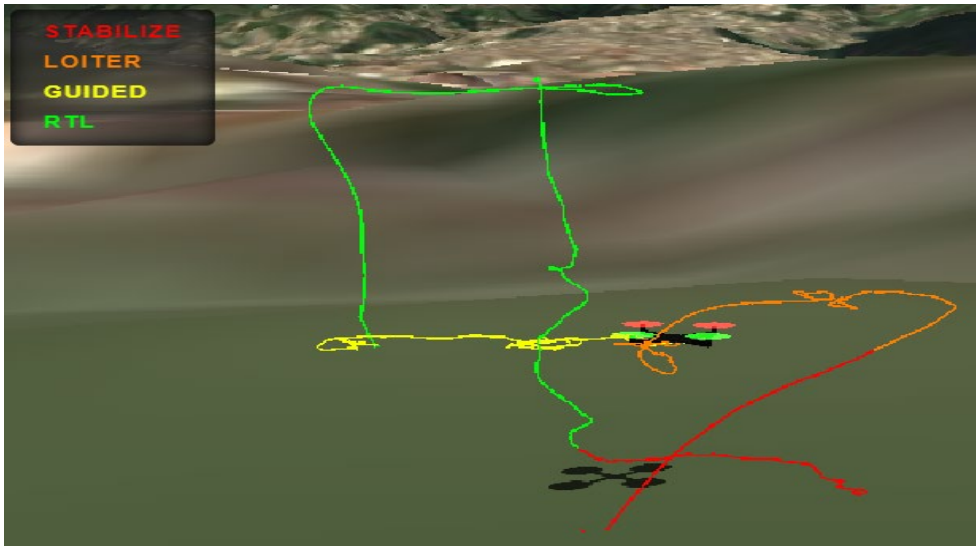


Figura IV.3 Trayectoria llevada a cabo por el dron durante la prueba 1 del tracker

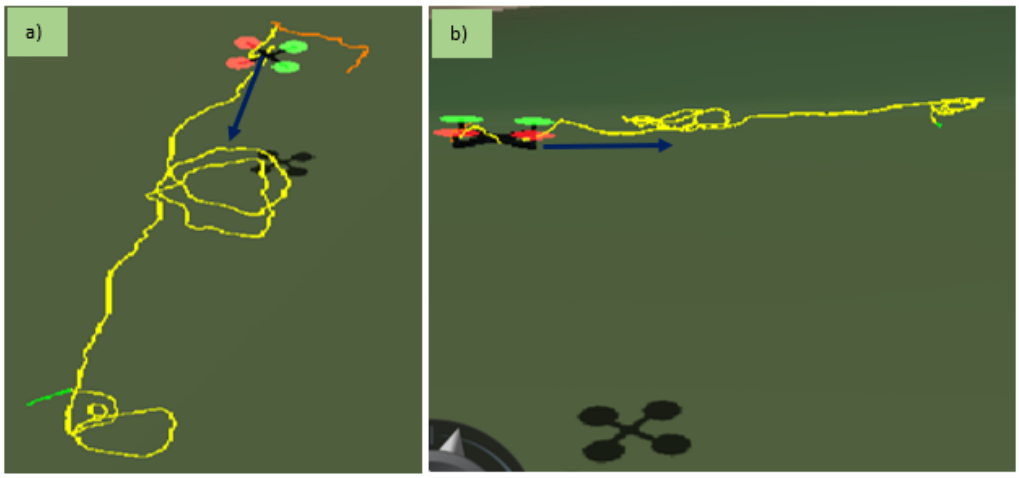


Figura IV.4 a)Vista superior de la maniobra b) Vista frontal de la maniobra prueba 1

PRUEBA 2:

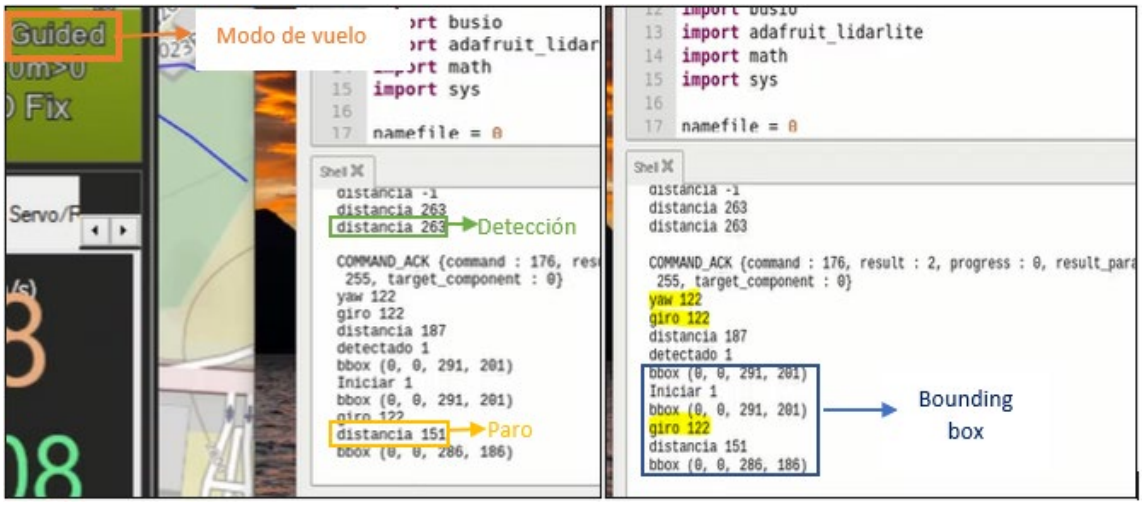


Figura IV.5 Actualización de los datos del Bounding Box para la prueba 2 del tracker

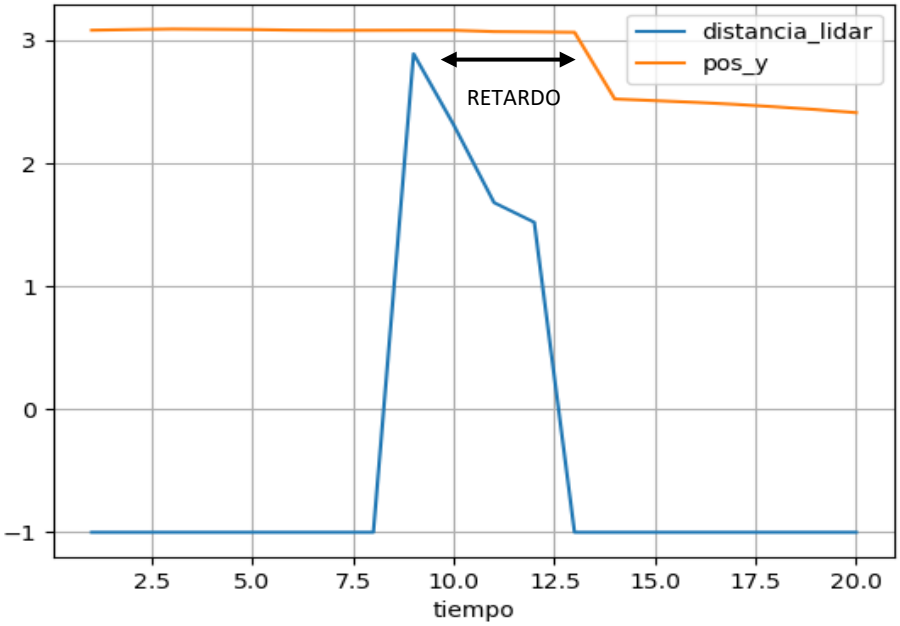


Figura IV.6 Registro de la distancia y posicion Y vs el tiempo para la prueba 2 del tracker

En la Figura IV.6 se observa un retardo de aproximadamente 2.5 segundos entre la detección de una distancia inferior a los 3 metros y la activación del tracker. Esto se da debido a que la implementación del tracker exige un nivel más alto de procesamiento, por lo que la comunicación entre la Raspberry y el Pixhawk Cube también se ve afectado.

Figura IV.7 Trayectoria llevada a cabo por el dron durante la prueba 2 del tracker

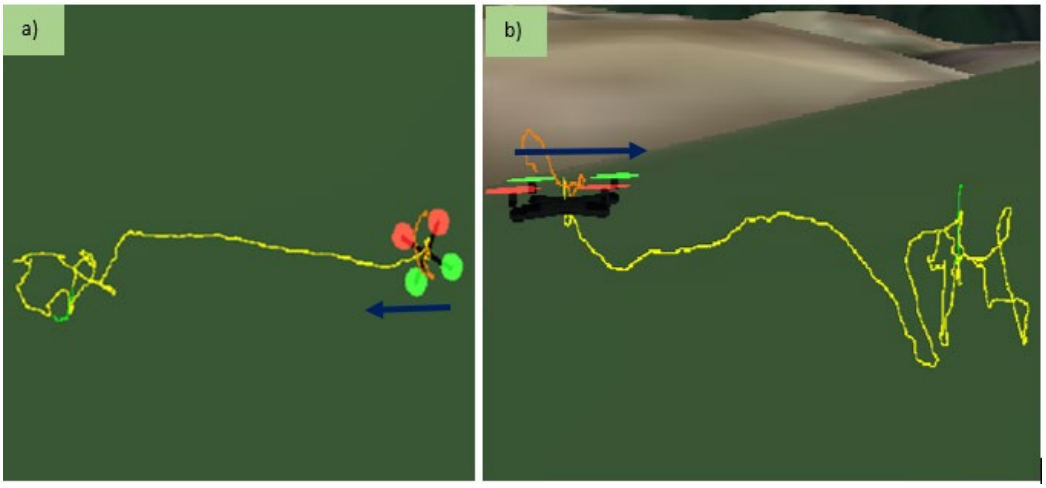


Figura IV.8 a)Vista superior de la maniobra b) Vista frontal de la maniobra prueba 2

PRUEBA 3:

Para esta prueba se cambió un poco la dinámica y en vez de que el obstáculo se quede quieto, se le mueve en función al movimiento del dron de tal manera que el objeto nunca sale del campo de visión del dron hasta que se lo permita.

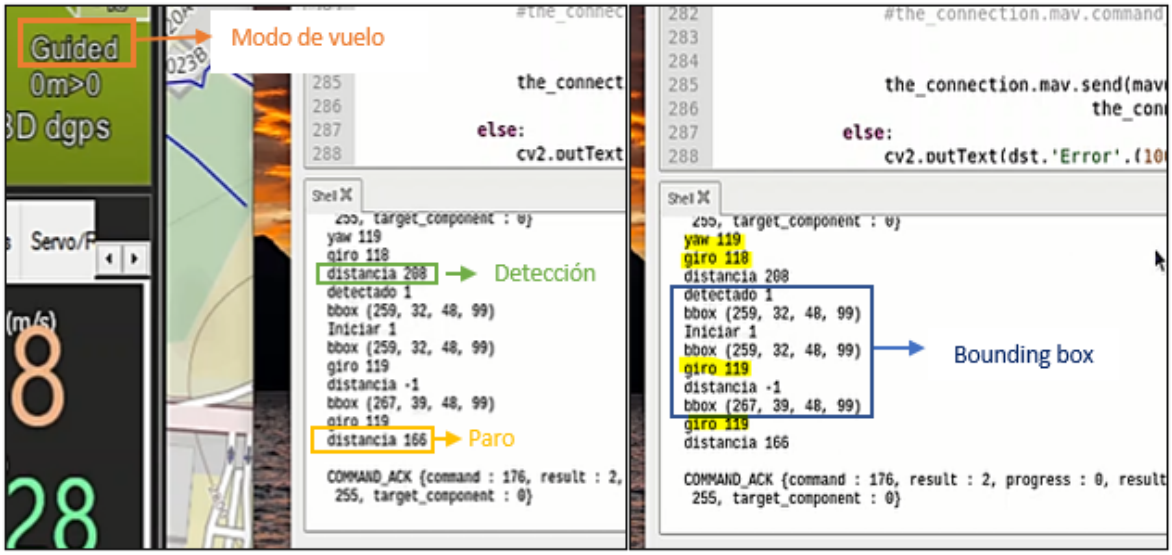


Figura IV.9 Actualización de los datos del Bounding Box para la prueba 2 del tracker

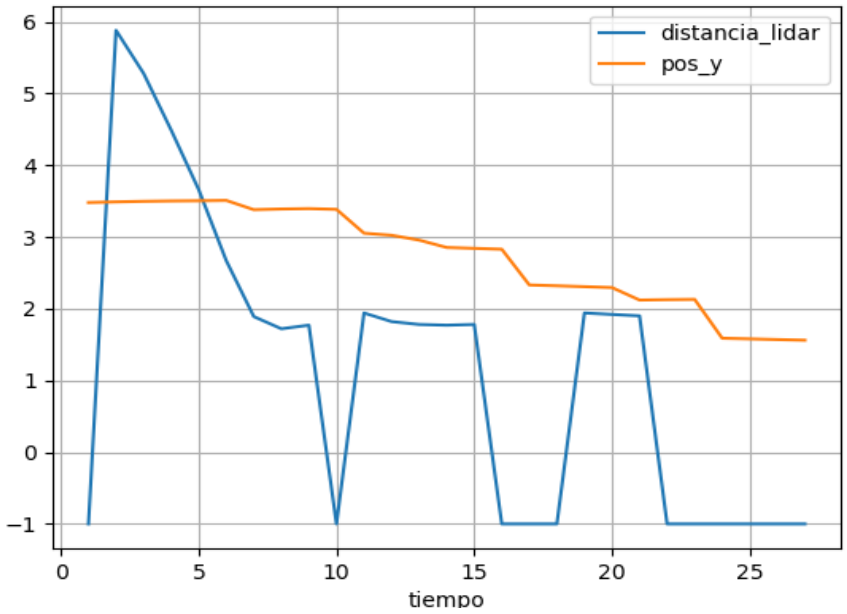


Figura IV.10 Registro de la distancia y posicion Y vs el tiempo para la prueba 2 del tracker

En la Figura IV.10 se observa que existe un retardo entre la distancia y la posicion Y, tal y como paso en la prueba 2, pero aparte de eso, también se observa como la posicion en Y varia sus valores a la derecha de manera continua sin parar, tal y como se esperaba debido a que el obstáculo se está moviendo a la misma dirección a la que se mueve el UAV.

Figura IV.11 Trayectoria llevada a cabo por el dron durante la prueba 3 del tracker

La Figura IV.11 resulta ser la misma de la Figura IV.7, esto debido a que los Data Logs de vuelo se guardan por cada vez que prendemos y apagamos el Pixhawk Cube, es por ello que en esta misma imagen se presentan los dos vuelos realizados para ambas pruebas.

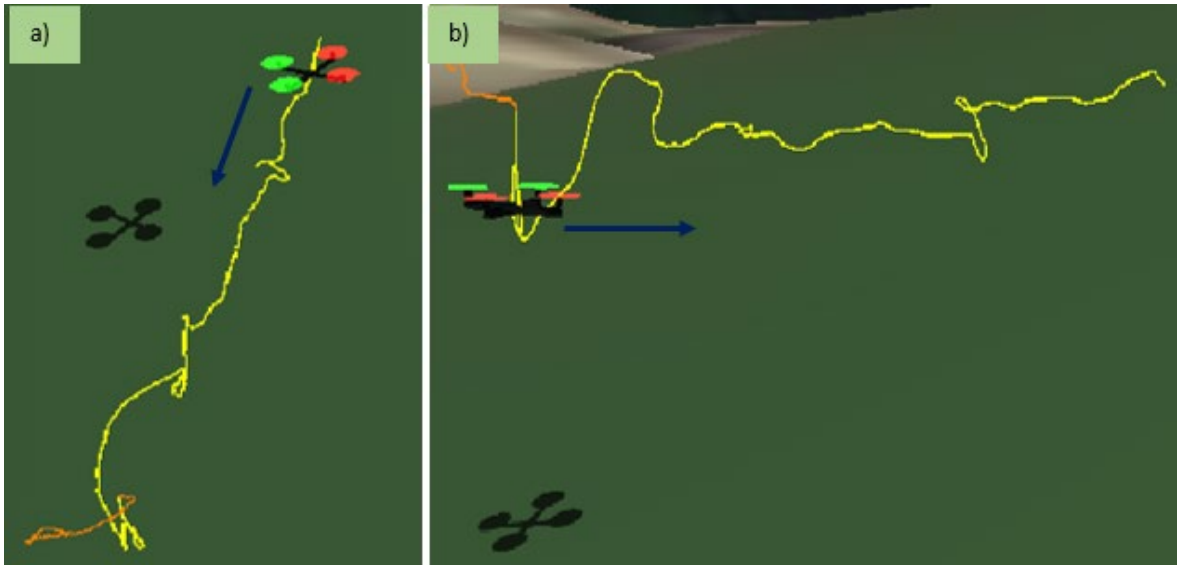


Figura IV.12 a)Vista superior de la maniobra b) Vista frontal de la maniobra prueba 3

ANEXO V. PRUEBAS DEL PROTOTIPO AUXILIAR DE EVASIÓN

En este anexo se presentan los resultados obtenidos al haber realizado pruebas del detector de obstáculos, vale aclarar que para estas pruebas ya se adaptó la cámara al brillo del entorno y que igual como se describió en la Sección 3.1.3, estos datos fueron obtenidos de los Data logs de vuelo y de los archivos CSV que se crean durante la misión.

PRUEBA 1:

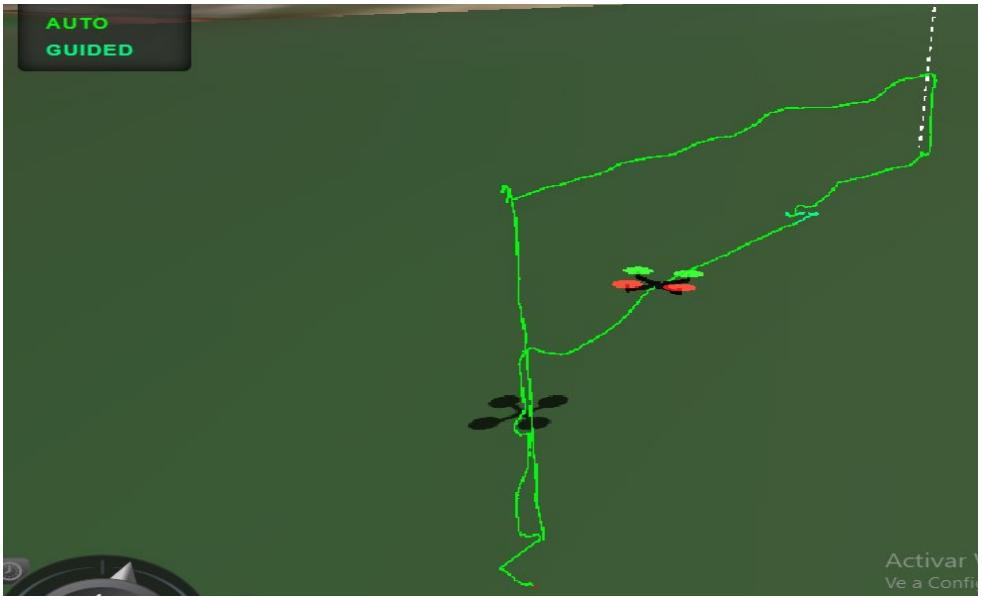


Figura V.1 Trayectoria llevada a cabo por el UAV durante la prueba 1 del prototipo

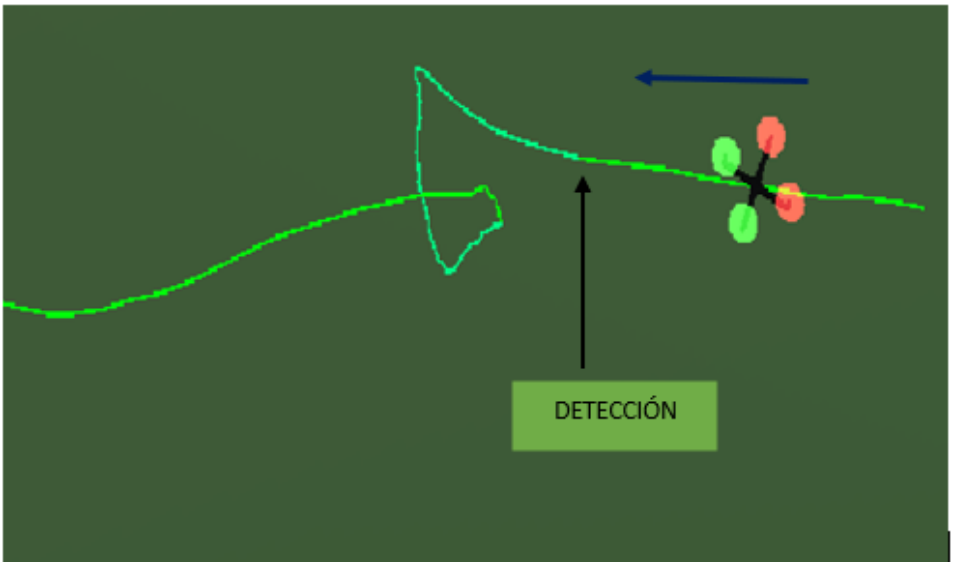


Figura V.2 Vista superior de la maniobra de evasión de obstáculos, la flecha azul indica la dirección de movimiento de la aeronave

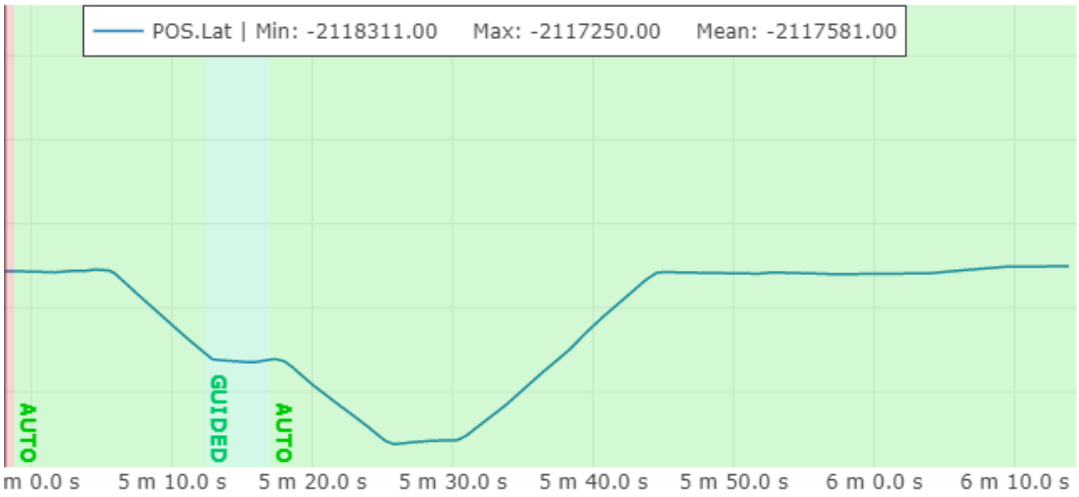


Figura V.3 Registro de la latitud del UAV durante la prueba 1 del prototipo

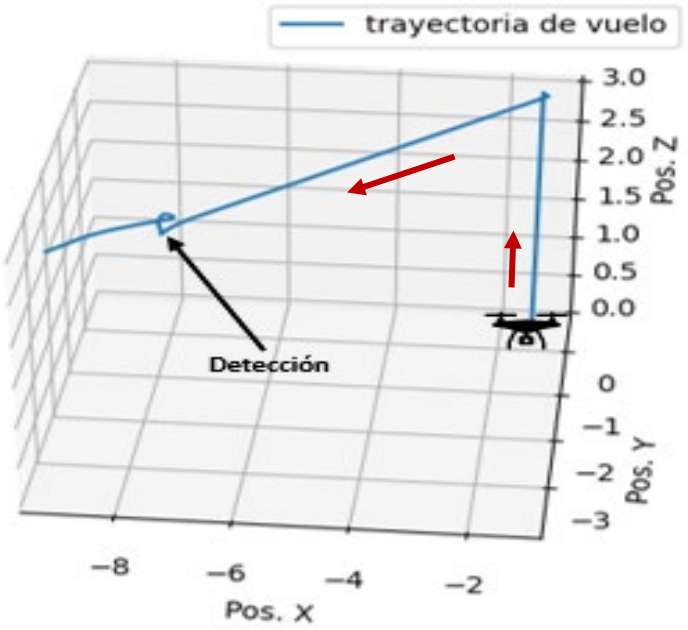


Figura V.4 Trayectoria de vuelo obtenida de los datos del archivo CSV para la prueba 1 del prototipo

PRUEBA 2:

En esta prueba se deicidio comprobar si mientras el dron se encuentra volando en modo auto, es factible cambiar su modo de vuelo de manera manual, esto como sistema de seguridad en caso de que el dron se salga de control en medio de una misión

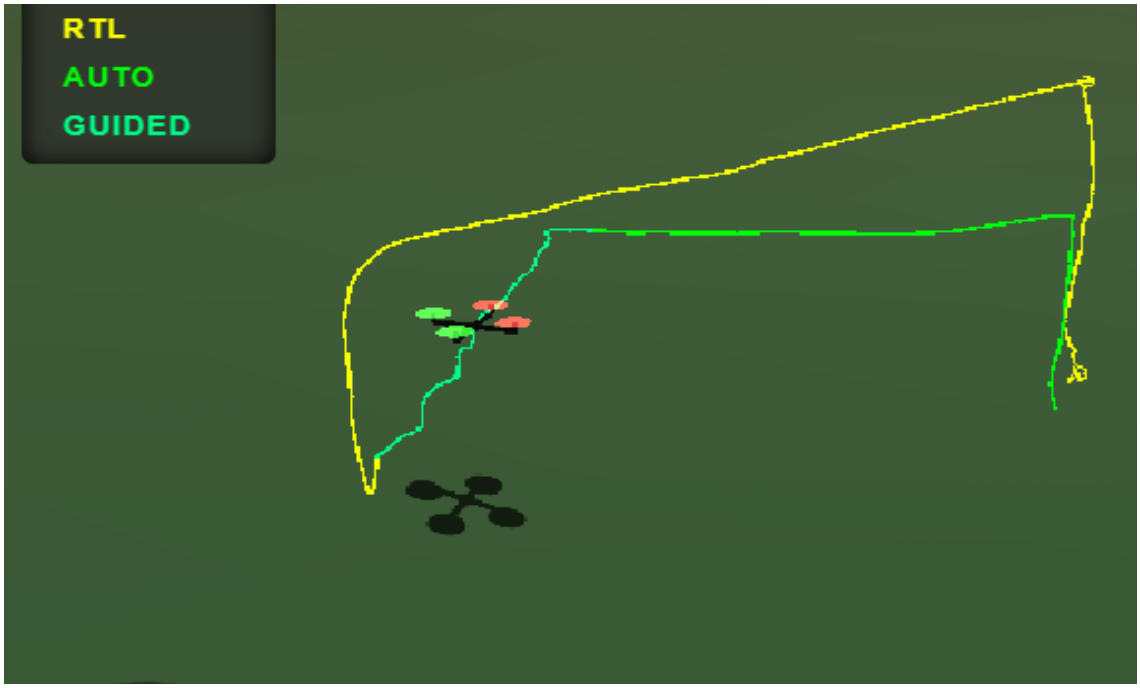


Figura V.5 Trayectoria llevada a cabo por el UAV durante la prueba 2 del prototipo

En la Figura V.5 se observa cómo una vez que se realiza la maniobra de evasión para evadir un obstáculo presentante en la trayectoria de vuelo del UAV, se le ordena de manera manual que realice un Return to lanch, esto nos indica que si vale rescatar al dron en caso de energía mientras este está realizando una misión en modo manual.

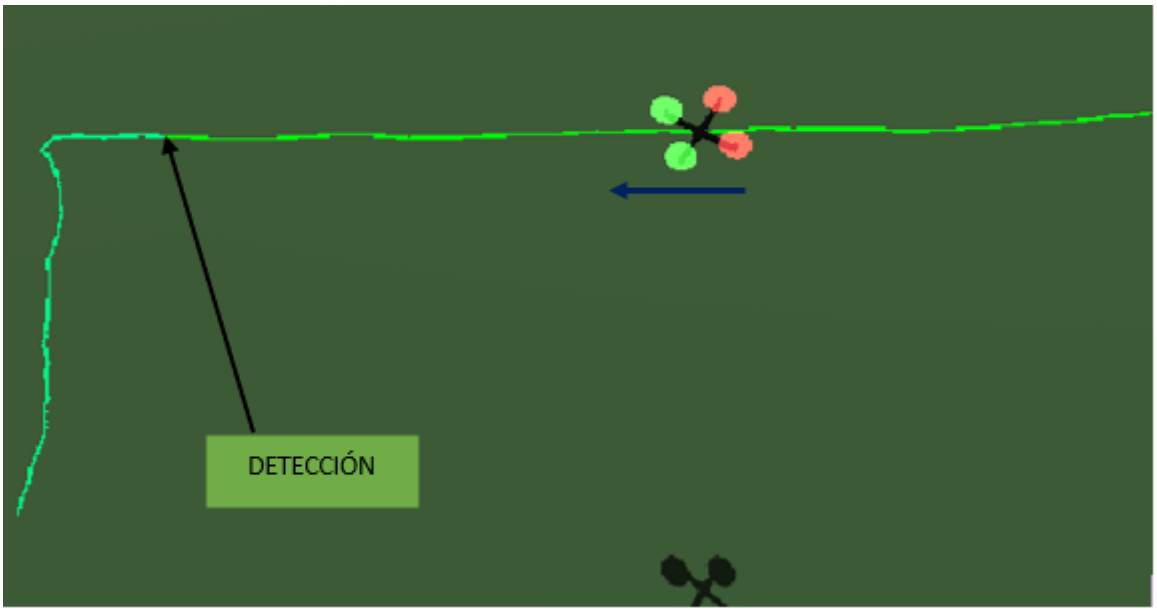


Figura V.6 Vista superior de la maniobra de evasión de obstáculos, la flecha azul indica la dirección de movimiento de la aeronave, prueba 2

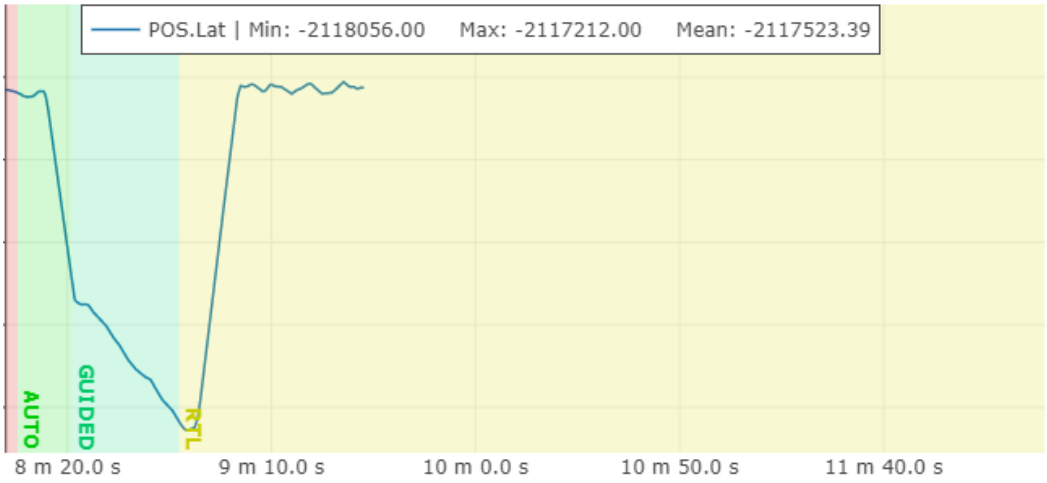


Figura V.7 Registro de la latitud del UAV durante la prueba 2 del prototipo

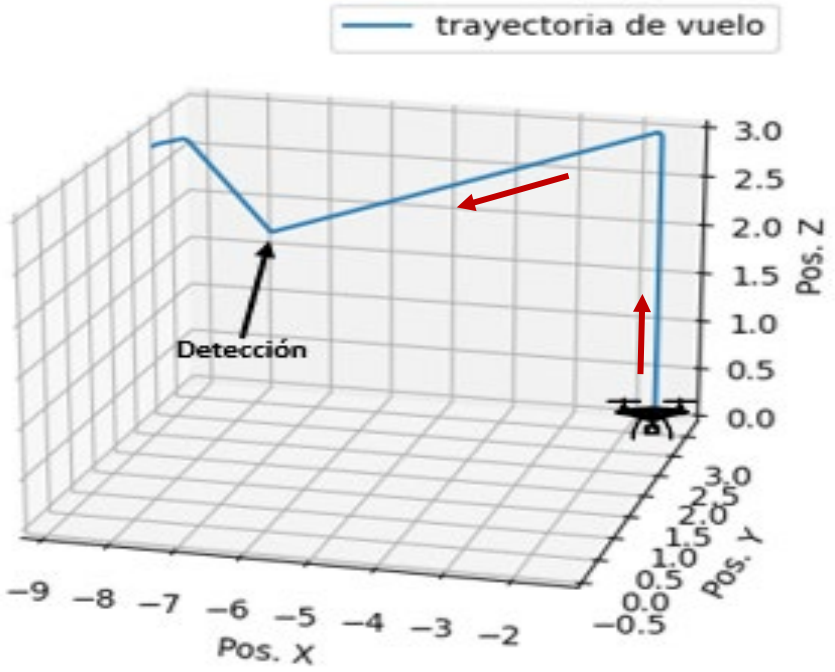


Figura V.8 Trayectoria de vuelo obtenida de los datos del archivo CSV para la prueba 2 del prototipo

PRUEBA 3:

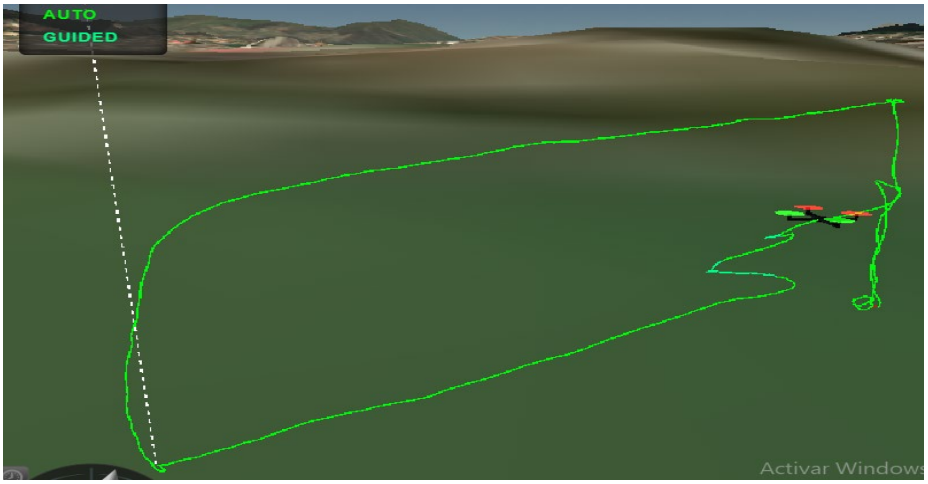


Figura V.9 Trayectoria llevada a cabo por el UAV durante la prueba 3 del prototipo

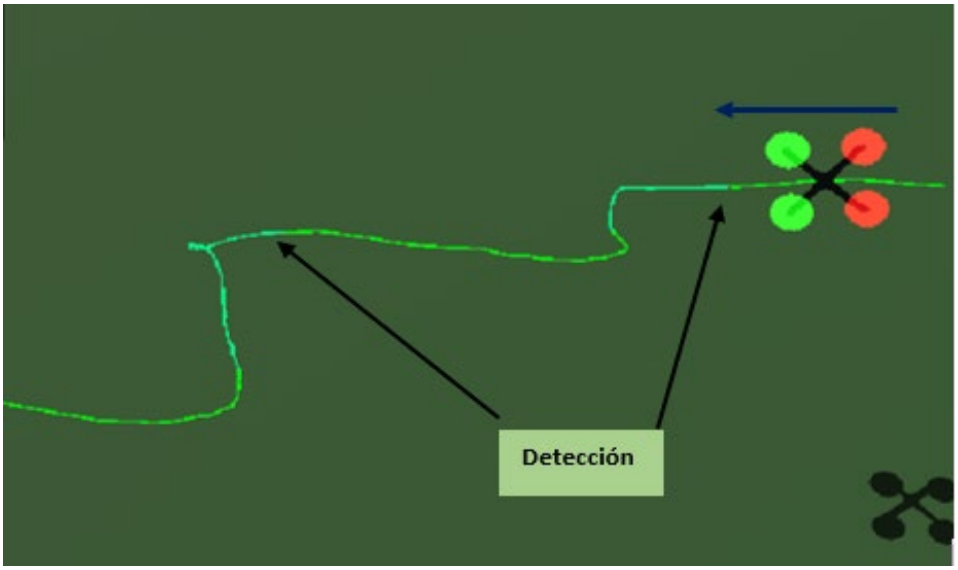


Figura V.10 Vista superior de la maniobra de evasión de obstáculos, la flecha azul indica la dirección de movimiento de la aeronave, prueba 3

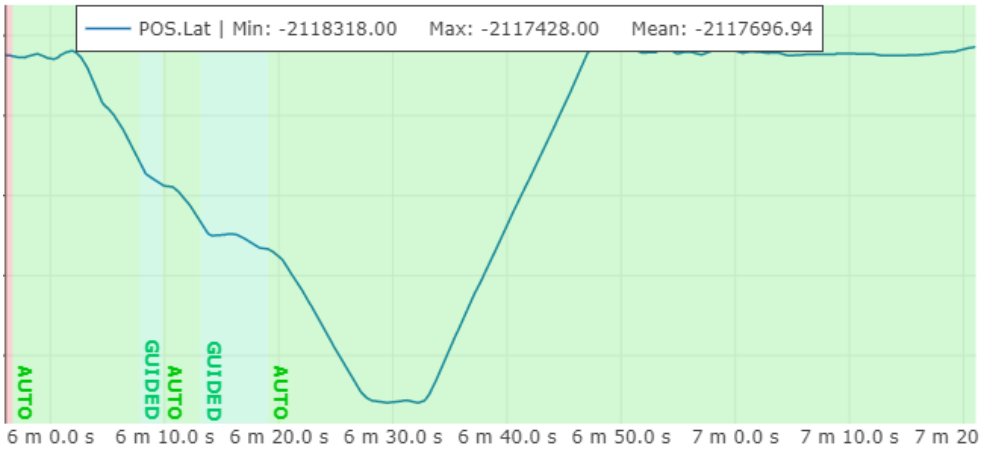


Figura V.11 Registro de la latitud del UAV durante la prueba 3 del prototipo

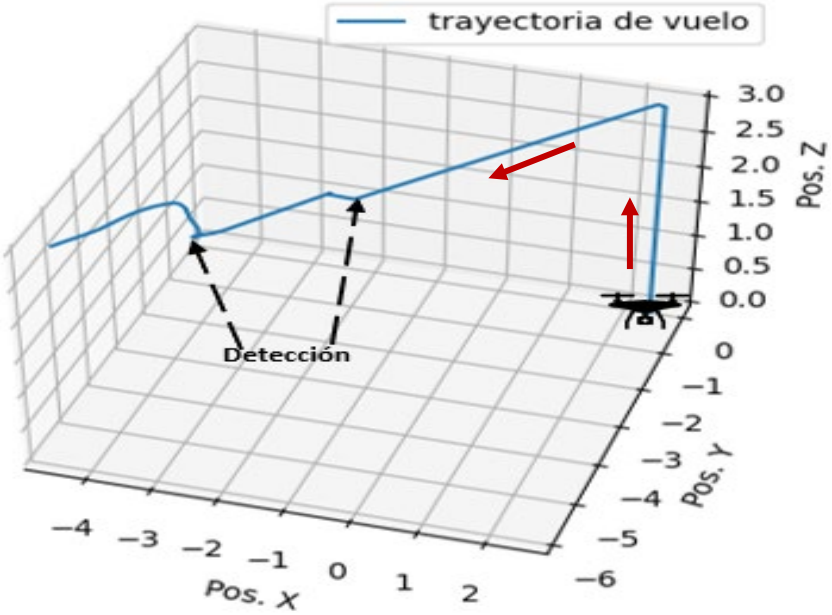


Figura V.12 Trayectoria de vuelo obtenida de los datos del archivo CSV para la prueba 3 del prototipo

ANEXO VI. ENLACE A VIDEOS DEMOSTRATIVOS

CARPETA COMPARTIDA:

ERICK LOYAGA - TIC - TITD201_GR9

En la carpeta compartida se encuentran todos los archivos que validan la realización de este trabajo de integración curricular. Se recomienda primero leer los archivos Readme.txt, ya que en ellos se encuentra la descripción de lo que contiene cada carpeta.

VIDEOS DEMOSTRATIVOS:

https://www.youtube.com/playlist?list=PLU-nFrzi9h-zEwnSHQWpo__7_AsiAouY9

ANEXO VII. MANUAL DE USUARIO

Este manual de usuario contiene todos los pasos a seguir para el correcto funcionamiento del prototipo auxiliar de evasión de obstáculos desarrollado en el trabajo de integración curricular “DISEÑO E IMPLEMENTACIÓN DE UN PROTOTIPO AUXILIAR QUE ENVIÉ INFORMACIÓN A UN AUTOPILOTO PIXHAWK ENFOCADO A LA APLICACIÓN DE EVASIÓN DE OBSTÁCULOS” . Se presenta desde las consideraciones previa a tomar en cuenta antes de poner en marcha este prototipo, hasta la obtención de los datos registrados en la Raspberry y controlador de vuelo, una vez culminada la misión.

Montaje:

El montaje del prototipo se realiza en un cuadricóptero modelo F450, el cual pertenece al laboratorio de UAVs del grupo de investigación ATA EPN. Para ello es necesario usar la plataforma de cargas y las piezas para el montaje de los sensores adaptadas en el cuadricóptero de pruebas. En la Figura VII.1 se presenta como deben estar colocados los elementos.

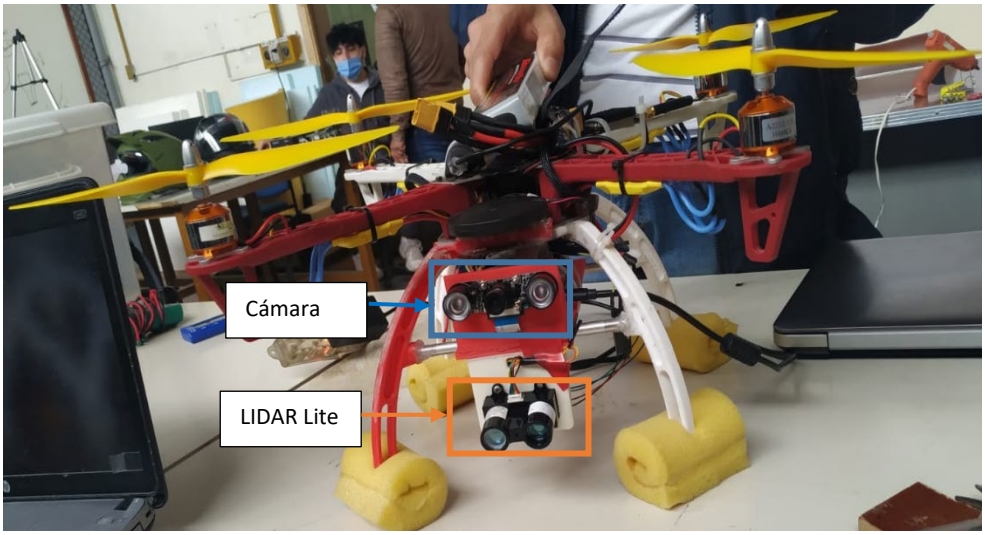


Figura VII.1 Montaje del prototipo auxiliar en el UAV de pruebas

Une vez montados todos los elementos, se debe verificar que los mismos estén conectados de manera correcta. La conexión apropiada de los diferentes elementes que componen el prototipo se puede visualizar en la Figura VII.2.

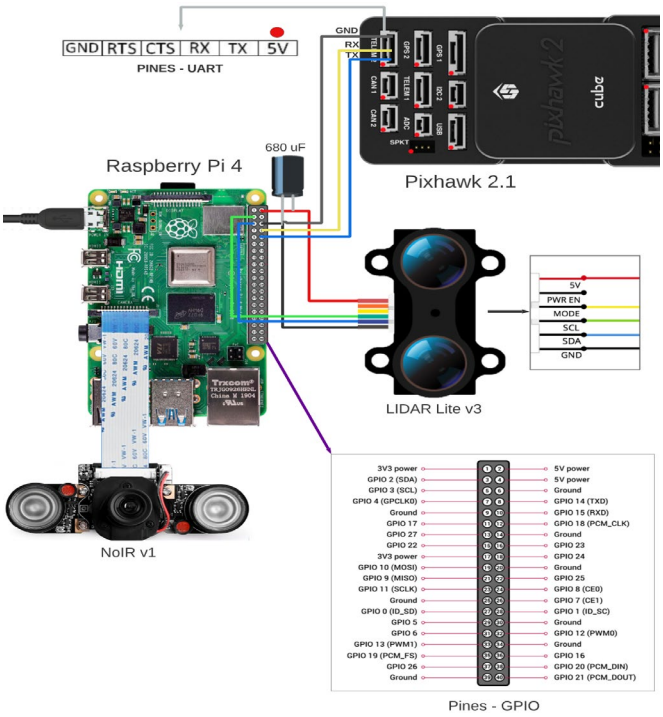


Figura VII.2 Esquema de conexión del prototipo auxiliar

En el caso que se desee probar el funcionamiento de este trabajo en un cuadricóptero de otro modelo se debe tener en cuenta las siguientes características.

- El modelo debe tener una carga útil que le permita añadir 84 gr de peso adicionales
- Tener espacio para colocar los elementos, teniendo en cuenta que la cámara y el Lidar Lite se deben colocar en forma que siempre estén indicando para el frente.
- Solo funciona con controladores de vuelo que permitan el protocolo de comunicación MAVLink

Nota: Al momento de montar los elementos se debe tener cuidado con la distribución de peso en el dron, ya que este debe estar lo más balanceado posible con respecto a su centro de masa.

Conexión inalámbrica entre la Raspberry y una Computadora:

Para acceder a la Raspberry a través de una computadora, es necesario que ambos dispositivos se encuentren conectados a una misma red. Por ello, para usar el sistema en exteriores se puede usar un router inalámbrico que genere una red a la cual se puedan conectar la Raspberry y la computadora con la cual se llevara el monitoreo del estado del vuelo.

Una vez que ambos dispositivos se encuentren conectados a la misma red, ya se puede acceder a la Raspberry con el uso de una computadora, a través del terminal de comandos siguiendo los siguientes pasos.

- Se digita el comando `ssh user@ip_address`, Donde user es el nombre de usuario de la Raspberry, para este caso pi, y el ip_address es la dirección IP de la Raspberry en la red generada por el router. En la Figura VII.3 se observa a más detalle lo descrito.

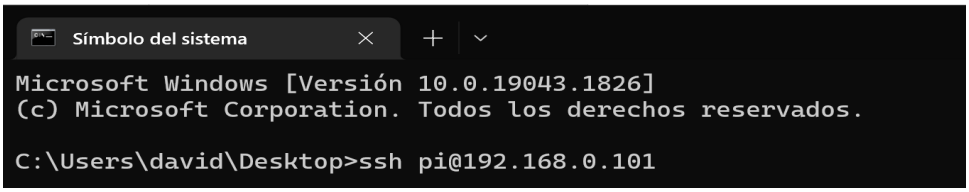


Figura VII.3 Acceso a la Raspberry a través del terminal de comandos

Luego, se coloca la contraseña de la Raspberry y se verifica que el acceso haya sido exitoso. Si se logeo correctamente con la Raspberry aparecerá un cuadro parecido al que se presenta en la Figura VII.4.

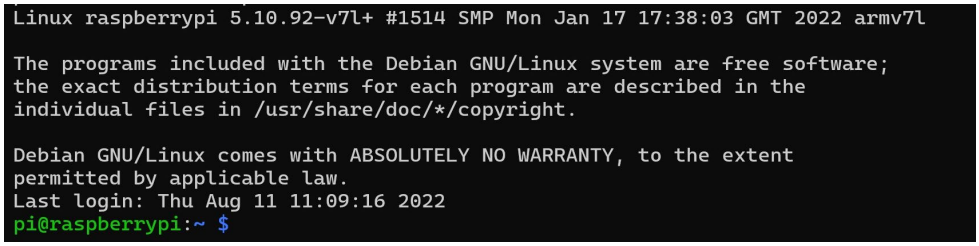


Figura VII.4 Acceso a la Raspberry exitosa

Conexión del Pixhawk Cube con el Mission Planner:

El UAV de pruebas utilizado en el desarrollo de este prototipo, cuenta con un receptor y un transmisor de telemetría. El receptor se debe montar en el dron de pruebas, mientras que el trasmisor se debe conectar a una entra USB de la computadora con la que se va a monitorear el vuelo. Posterior a esto, en la pestaña de configuraciones del Mission Planner se debe verificar que tanto el receptor como el transmisor tengan la misma configuración local y remota, principalmente el mismo valor de Net ID. En la Figura VII.5 se presenta la pestaña donde se debe verificar lo descrito en este párrafo.

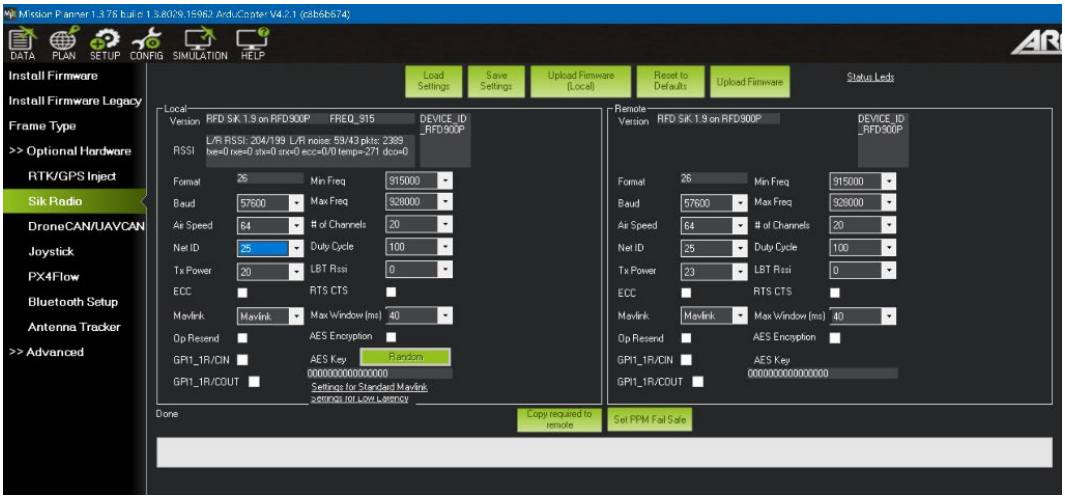


Figura VII.5 Pestaña de configuración del Net ID.

Una vez verificado la configuración del receptor y transmisor, se da clic en el botón CONNECT y a través del menú DATA se verifica el estado de las variables de vuelo de la aeronave.

Figura VII.6 Localización de las variables de vuelo de interés

Antes de mandar a ejecutar el código de este prototipo es necesario verificar ciertas variables como el nivel de la batería, el cual debe estar por encima de los 9V, que el número de satélites sea mayor a 15 y el modo de vuelo del dron, el cual, si o si debe estar en modo stabilize, el cual equilibra al dron con el eje horizontal, esto debido a que los motores se mandan a encender por medio de código y como medida de seguridad el controlador solo permite esto cuando el modo de vuelo es stabilize. En la Figura VII.6 se presenta la ubicación de estos variables.

Nota: Durante el vuelo del dron es necesario siempre verificar el valor de la batería, teniendo en cuenta que esta nunca debe bajar de 9V, en caso de que esto suceda se debe inmediatamente ordenar a la aeronave que aterrice o que regrese a su punto de partida.

Ejecución de la evasión guiada:

Para ejecutar el algoritmo de evasión guiada, se debe acceder al programa a través del terminal de comandos, haciendo uso del servidor ssh. Para ello, primero es necesario dirigirse al directorio donde esta guardado el programa a través del siguiente comando `cd/home/pi/<directorio>`. Una vez ya en la carpeta se usa el comando `ls` para visualizar los archivos que se encuentran dentro de esa carpeta. En la Figura VII.7 se visualiza lo que se acaba de describir.

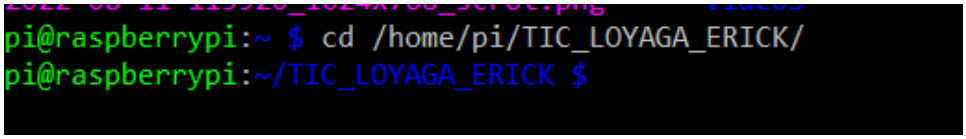


Figura VII.7 Ingreso a la carpeta que contiene el programa

Por último, tal y como se presenta en la Figura VII.8, se ejecuta el programa de apoyo para la evasión guiada a través del comando `python3 prueba_tesis_completa.py`.

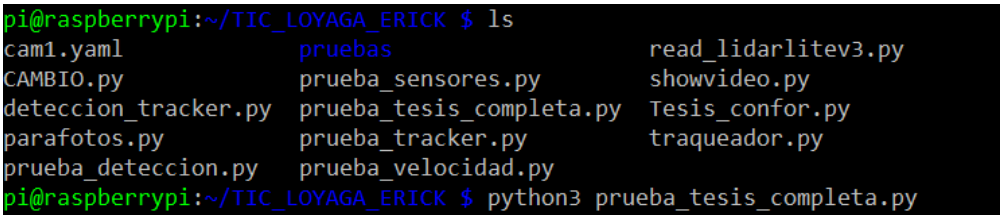


Figura VII.8 Ejecución del programa auxiliar de evasión

Datos registrados en la Raspberry

Para descargar los archivos .csv creados durante el desarrollo de la misión de vuelo, se debe usar el terminal de comandos y el protocolo de transferencia de datos Secure File Transfer Protocol (sftp) con los siguientes pasos:

- Acceder a la Raspberry a través del comando `sftp.pi@ip_address` y luego escribir la contraseña de esta.
- Dirigirse al directorio donde se encuentra la carpeta que guarda los documentos .csv, a través del comando `cd /home/pi/TIC_LOYAGA_ERICK/DATOS`.
- Revisar la lista de documentos que contiene la carpeta escogida en el paso anterior

- Usar el comando `get` seguido del nombre del archivo que desea descargar. Esto último se presenta en la Figura VII.9.

```
sftp> get datos1_08_08_11h03m.csv
Fetching /home/pi/TIC_LOYAGA_ERICK/pruebas/datos1_08_08_11h03m.csv to dato
s1_08_08_11h03m.csv
/home/pi/TIC_LOYAGA_ERICK/pruebas/datos1_08_08_11h03m.csv
                      100% 4442    154.9KB/s   00:00
```

Figura VII.9 Descarga de un archivo .csv

Datos registrados en el controlador de vuelo

Para poder descargar los datos de vuelo, es decir los Data logs generados por el controlador de vuelo, se debe conectar el controlador a través de un puerto USB a un computador. Luego siga los siguientes pasos:

- Dirigirse a la pestaña DataFlash Logs Y SELECCIONE LA OPCION Dowload DataFlash Log vía MAVLink.
- Una vez dentro se escoge el Data Log de vuelo de interés y se da clic en la opción Dowload Selected Logs.
- Estos pasos descritos se observan con mayor detalle en la Figura VII.10.

Figura VII.10 Pasos a seguir para descargar los Data Logs de vuelo

Nota: Los Data logs de vuelo se guardan dentro de la carpeta logs que se encuentra dentro de la carpeta Mission Planner. Además, se debe acceder a través de cualquier buscador a la herramienta UAV Log Viewer que ofrece la compañía ArduPilot, ya que ella decodifica todos los datos que tiene el Data log de vuelo. A continuación, se presenta un ejemplo de su uso.

Primero es necesario acceder a la herramienta UAV Log Viewer y arrastrar el Data log de vuelo a revisar. En la Figura VII.11 se visualiza la pantalla principal de esta herramienta.

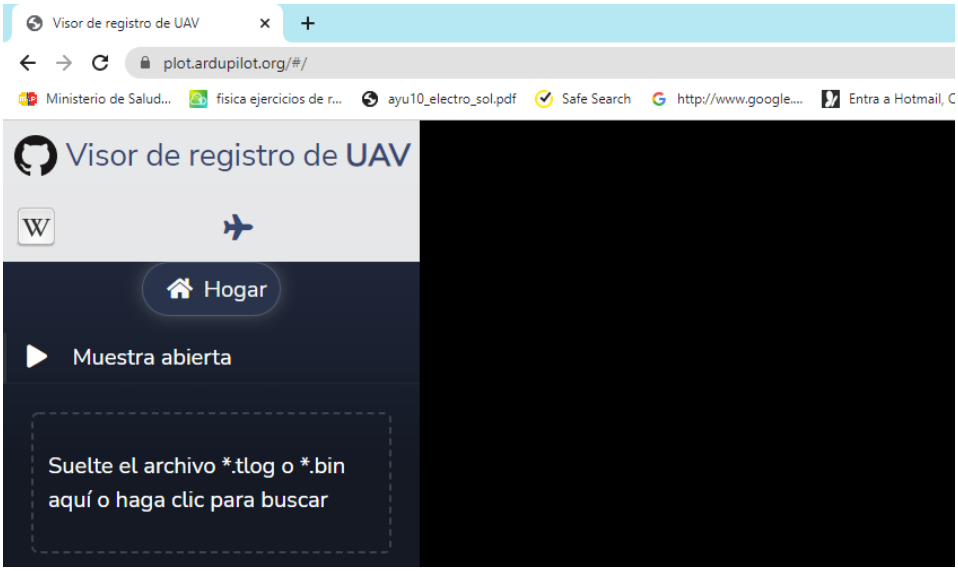


Figura VII.11 Pantalla principal del UAV Log Viewer

Luego, se abre una ventana en donde se observa la trayectoria de vuelo en 3D llevada por la aeronave durante la misión de vuelo. En ella al dar clic en la pestaña plot se accede a la información de todas las variables de vuelo. En la Figura VII.12 se presenta lo descrito.

Figura VII.12 Lectura de un Data log de vuelo usando la herramienta UAV Log Viewer

Para visualizar una variable de vuelo en específico se debe extender la pestaña Plot Individual Field y luego dar clic sobre la variable que se desea revisar. Por ejemplo, si se desea revisar la posicion que tuvo la aeronave durante la misión se debe dar clic sobre la subpestaña POS y luego escoger las variables de interés. En este caso se escoge la Latitud y Longitud. En la Figura VII.13 se visualiza el resultado de este ejemplo.

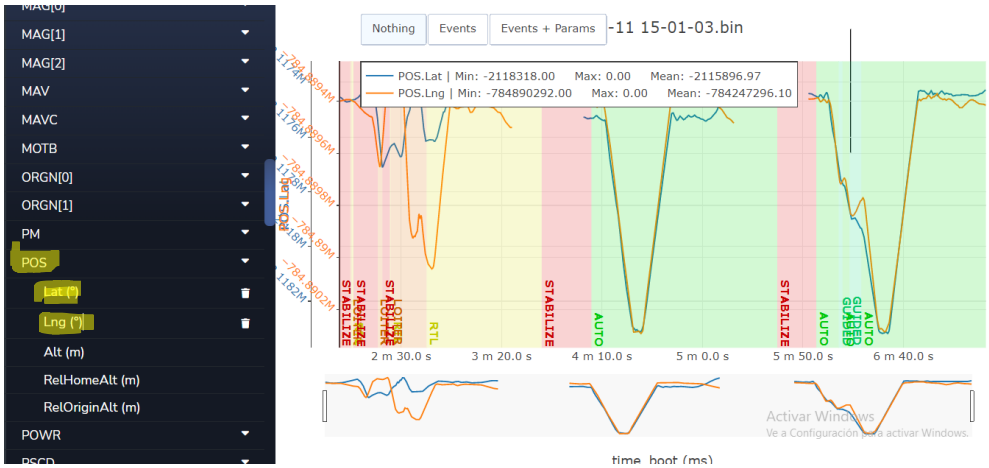


Figura VII.13 Latitud y Longitud del resultado de una misión de vuelo