https://repository.uaeh.edu.mx/revistas/index.php/investigium/issue/archive

Boletín Científico INVESTIGIUM de la Escuela Superior de Tizayuca

Publicación semestral, Vol. 11, No. 21 (2025) 77-88

InvESTigium

ISSN: 2448-4830

Sistema de control para vuelo coordinado de Vehículos Aéreos no Tripulados basado en el drone DJI Ryze Tello® y Rasberry Pi®

Control system for coordinated flight of Unmanned Aerial Vehicles based on the DJI Ryze Tello® drone and Raspberry Pi®

Gerardo Loreto Gómez [a]

Abstract:

The development of unmanned aerial vehicles (UAVs) will allow them to play a key role in the future, supporting various services. Today, their application is found in areas such as research, agriculture, photography and cinematography, archaeology, package delivery, among others. The UAVs available on the market are a viable solution for different tasks because they have a reliable control system for flight stabilization, on-board cameras and sensors that allow controlling the movement to be performed. However, an undesirable characteristic is their limited flight time due to the energy restriction inherent to the batteries they have, reducing the ability to perform complex or lengthy tasks. Swarm robotics is presented as a promising solution to address these limitations by improving the robustness and adaptability of the system by distributing the workload among multiple drones, increasing operational efficiency because each one will be able to perform a part of the total task, allowing the mission to continue even if one or more were to fail. This article describes the results obtained by implementing algorithms for the control of a swarm of drones using an experimental platform developed for this purpose. The platform architecture is divided into two modules, the low-level one, dedicated to the Wi-Fi communication protocol and sending commands between the Raspberry Pi® control card and the DJI Ryze Tello® commercial drone. The second high-level module is integrated by each low-level module of the drones that will make up the swarm and a master Raspberry Pi® card, whose main function is mission programming. Communication is carried out through a wired network implemented in a commercial router. The results obtained allow validating that the control algorithms and the architecture of the developed platform are an efficient proposal to control the coordinated flight of drones.

Keywords:

Swarm Robotics, Unmanned Aerial Vehicles, DJI Ryze Tello

Resumen:

El desarrollo de vehículos aéreos no tripulados (UAV) permitirá que en un futuro desempeñen un papel clave, brindando apoyo a diversos servicios. Hoy en día, su aplicación se encuentra en áreas como investigación, agricultura, fotografía y cinematografía, arqueología, entrega de paquetes, entre otros. Los UAV disponibles en el mercado resultan una solución viable para diferentes tareas debido a que disponen de un sistema de control confiable para la estabilización de vuelo, cámaras a bordo y sensores que permiten controlar el movimiento a realizar, sin embargo, una característica no deseada es su tiempo de vuelo limitado debido a la restricción energética inherente a las baterías que poseen, reduciendo la capacidad de realizar tareas complejas o prolongadas. La robótica de enjambre se presenta como una solución prometedora para abordar estas limitaciones al mejorar la robustez y adaptabilidad del sistema distribuyendo la carga de trabajo entre múltiples drones, aumentando la eficiencia operativa debido a que cada uno podrá realizar una parte de la tarea total, permitiendo que la misión continúe incluso si uno o más fallarán. En este artículo se describe los resultados obtenidos al implementar algoritmos para el control de un enjambre de drones utilizando una plataforma experimental desarrollada para tal fin. La arquitectura de la plataforma está dividida en dos módulos, el de bajo nivel, dedicado al protocolo de comunicación Wi-Fi y envió de comandos entre la tarjeta Rasberry Pi® de control y el drone comercial DJI Ryze Tello®. El segundo módulo de alto nivel, se integra por cada módulo de bajo nivel de los drones que conformaran el enjambre y una tarjeta Rasberry Pi® maestra, cuya función principal es la programación de la misión, la comunicación se realiza a través de una red cableada implementada en un router comercial. Los resultados obtenidos permiten validar que los algoritmos de control y la arquitectura de la plataforma desarrollada son una propuesta eficiente para controlar el vuelo coordinado de drones.

Palabras Clave:

Robótica de enjambre, Vehículo Aéreo No Tripulado, DJI Ryze Tello

[a] Tecnológico Nacional de México | Instituto Tecnológico Superior de Uruapan | Uruapan – Michoacán | México, https://orcid.org/0000-0002-3960-8801, Email: gerardo.lg@uruapan.tecnm.mx

Fecha de recepción: 19/08/2024, Fecha de aceptación: 05/12/2024, Fecha de publicación: 05/07/2025

DOI: https://doi.org/10.29057/est.v11i21.13535

Introducción

Los micro vehículos aéreos (MAV), hacen referencia a drones pequeños y livianos que pesan pocos gramos [1]. Estos drones están diseñados para ser altamente maniobrables y pueden ser utilizados en una variedad de aplicaciones, incluyendo vigilancia, reconocimiento, investigación científica, y en algunos casos, misiones militares [2]. Lamentablemente, un tamaño menor implica capacidades más limitadas con respecto al tiempo de vuelo, sistemas de detección a bordo y potencia necesaria para realizar tareas más complejas por sí solo, entre otros. Esto ha creado un gran interés en investigar como ampliar sus capacidades a través de utilizar un grupo de MAV para realizar tareas de forma colectiva [3].
El desarrollo de aplicaciones con vuelo coordinado de grupos de MAV es un desafío multidisciplinario que se puede dividir en dos aspectos principales. Por un lado, el diseño del control individual de cada MAV tomando en cuenta sus capacidades locales, y por el otro, el control de vuelo en grupo mediante el cual se necesita planificar formaciones de vuelo y establecer la arquitectura de control que permitan lograr de manera eficiente el objetivo propuesto [4].

Las características individuales a tener en cuenta para el control de un MAV incluyen el tiempo de vuelo, peso, sensores a bordo, procesadores y actuadores. Además, de los sistemas para el control de orientación y posición, así como de velocidad [5]. El MAV debe ser capaz de navegar de forma segura en su entorno con la capacidad para detectar y evitar posibles obstáculos [6]. Por otro lado, las características de grupo incluyen la comunicación entre los MAV, de modo que puedan compartir información de la ubicación de los vecinos cercanos que permita asegurar el comportamiento deseado del grupo y que garantice que los MAV no colisionaran entre sí en el aire [7]. Para cumplir este último punto, es requerido definir el tipo de formación que adoptara el grupo, esto es, como se organizaran en una estructura global modificando sus ubicaciones para completar una tarea definida mientras considera sus restricciones dinámicas [8]. En general, el vuelo coordinado enfrenta tres problemas principales, la generación de la formación, mantener su forma y la reconfiguración [9]. En este contexto, la formación puede ser rígida (los MAV se mueven para mantener las posiciones relativas entre ellos), flexible (reconfiguración o división por obstáculos detectados), o ambas dependiendo de la tarea.

En la literatura, los métodos que abordan el problema del tipo de formación generalmente se agrupan en las categorías de Líder-seguidor, estructura virtual, basada en el comportamiento y basada en el consenso [10]. En el enfoque de líder-seguidor se designa a uno o más MAV de la formación como líderes y a los demás como seguidores; en estos casos, la trayectoria del líder es la referencia para los seguidores. La principal ventaja de esta estrategia es su simplicidad y escalabilidad; sin embargo, esta configuración depende en gran medida del líder [11]. En la técnica de estructura virtual, los MAV deben mantener una relación geométrica rígida entre ellos; su principal desafío es mantener la formación para tareas de prevención de colisiones [12]. En el método basado en el comportamiento, la formación se mantiene considerando reglas basadas en una combinación ponderada de diferentes comportamientos, que puede incluir el evitar colisiones, moverse hacia un objetivo, mantener la formación, entre otros [13]. Finalmente, la estrategia de consenso propone un control cooperativo entre integrantes del grupo, se utiliza comúnmente en sistemas descentralizados [14].

Una vez establecido el tipo de formación se requiere considerar el tipo de control para el vuelo coordinado de múltiples MAV, los enfoques utilizados son centralizado o descentralizado. En el primero, todos los MAV están coordinados por un único dispositivo externo que planifica las acciones con base en el conocimiento de los estados de todos los MAV. La planificación puede realizarse a priori o en línea, su principal ventaja es la simplicidad de los integrantes del grupo; sin embargo, su excesiva dependencia del sistema central puede ser un problema [15]. En un control descentralizado, los MAV toman sus decisiones localmente, siguiendo reglas simples de comportamiento en función de la información que obtienen de su entorno inmediato, esto es, no existe un dispositivo externo que controle el comportamiento del grupo, su principal desventaja es la complejidad computacional del hardware y software de cada integrante del grupo. Este principio de descentralización se ha convertido en la piedra angular de la denominada robótica de enjambre [16].

Un aspecto importante dentro de la arquitectura de control es la forma en la cual se realiza la detección del entorno, por un lado, es posible utilizar dispositivos externos como sistemas de navegación global o sistemas de captura de movimiento, dependiendo de si los MAV vuelan al aire libre o en interiores, y a través de sensores que están a bordo del MAV. El uso de una planificación centralizada y un posicionamiento externo hace que el sistema de control, generalmente, no sea escalable. En cambio, una estrategia distribuida en la que cada MAV toma una decisión de forma independiente permite obtener las ventajas de redundancia, escalabilidad y adaptabilidad a los cambios [17]. Sin sistemas de posicionamiento

externos o controles centralizados, el problema de volar varios MAV a la vez se vuelve más desafiante, debido a que los MAV tienen que confiar solo en la percepción a bordo, a medida que aumenta el tamaño del grupo, disminuye el conocimiento relativo que cada MAV tendrá de su entorno global [18].

Por todo lo anterior, es requerido el diseño de plataformas flexibles y robustas que permitan tomar todas estas características para realizar pruebas experimentales para un vuelo coordinado de MAV. En este artículo se describe los resultados obtenidos al implementar algoritmos para el control de vuelo coordinado utilizando una plataforma experimental basada en el drone DJI Ryze Tello® y la tarjeta Rasberry Pi®. La arquitectura de control distribuido propuesta permite realizar el procesamiento de la tarea en dos niveles, el dedicado al protocolo de comunicación y control individual de cada Tello y el segundo nivel dedicado a la planificación de la tarea y coordinación grupal de los MAV.

El contenido del artículo se encuentra organizado de la siguiente forma. En la sección 2 se realiza un análisis profundo del Micro UAV comercial DJI Tello®, se cubre las características de su hardware y protocolos de comunicación, así como el software disponible para poder realizar su programación. En la sección 3 se describe la arquitectura de la plataforma desarrollada. En la sección 4 se muestran los resultados de las pruebas realizadas. Finalmente, en la sección 5 se presentan las conclusiones del trabajo.

2. Mini Drone DJI Ryze Tello®

El DJI Ryze Tello® es un mini drone fabricado por *Shenzhen Ryze Technology*, se encuentra disponible en el mercado en tres modelos, ver figura 1. El modelo Tello EDU presenta las siguientes características, utiliza el último kit de desarrollo de software (SDK 3.0), es posible controlar varios drones desde un único dispositivo externo pero presenta la limitante de no poder enviar el video de la cámara hacia el dispositivo en este modo de control, se tiene acceso al video de la cámara colocada en la parte inferior del drone, presenta el apagado de los motores cuando se encuentra en modo estacionario para evitar que se sobrecaliente y permite planificar misiones utilizando patrones especiales llamados plataformas de misión.

Por otro lado, la edición más avanzada del drone es Robomaster Tello Talent® que contiene un módulo ESP32, una pantalla LED de matriz de puntos programable de 8x8 y un indicador LED, un sensor ToF de un solo punto, un módulo Wi-Fi de 5,8 GHz y un adaptador de extensión especial para conectar sensores adicionales. Existe un kit de expansión ofrecido por el fabricante denominado RMTT, que permite actualizar el Tello Edu para tener características similares que el módulo del Robomaster Tello Talent®. Debido a que la disponibilidad de los modelos Tello EDU y Robomaster TT es limitada en el momento de realizar el presente proyecto, la plataforma propuesta utiliza el modelo básico del Tello.

Figura 1. Mini Drones DJI Ryzen Tello

El DJI Tello básico es propulsado por cuatro motores sin escobillas y hélices de 3 pulgadas capaces de proporcionar un vuelo estable, es compacto (98 × 92.5 × 41 mm) y ligero (peso aproximado de 80 gramos), fácil de manejar y transportar, haciéndolo adecuado para entornos cerrados y abiertos. Equipado con un procesador Intel Movidius Myriad 2 capaz de realiza tareas avanzadas de procesamiento de imagen y control de vuelo, incluyendo estabilización electrónica y visión computarizada en tiempo real. Su cámara frontal es de 5 megapíxeles con un campo de visión de 82.6º, puede capturar imágenes con una resolución de 2592 × 1936 píxeles y videos en HD a 1280x720 a 30p, lo que permite la recolección de datos visuales precisos para aplicaciones de monitoreo y mapeo. Tiene un alcance operativo de hasta 100 metros con una altitud máxima de 30 metros.

El Tello utiliza un sistema de posicionamiento visual (VPS) que consta de un par de sensores infrarrojos (IR) y una cámara que apunta hacia abajo para garantizar un vuelo estable y evitar ir a la deriva en vuelo estacionario. Sin embargo, puede verse afectado cuando vuela sobre superficies monocromáticas, altamente reflectantes y transparentes sin patrones o cuando vuela en entornos oscuros o con poca iluminación. Si el VPS falla, Tello pasa a un modo de actitud, donde mantiene su orientación (inclinación, cabeceo y guiñada) pero sin estabilización completa pudiendo ya no ser estable, lo cual pudiera resultar inseguro al volar en interiores.

Por otro lado, el Tello tienen múltiples funciones de seguridad incorporadas, un sistema de detección de colisiones que apaga los motores en caso de colisión con algún obstáculo, la capacidad de aterrizaje automático cuando no se han recibido comandos en los últimos 15 segundos, no permite el despegue o realizar el aterrizaje cuando el nivel de batería es demasiado bajo o cuando la conexión con el dispositivo de gestión es demasiado débil o se pierde. Estos comportamientos nativos permiten su manejo seguro en caso de problemas de vuelo o fallas en el sistema.

El Tello cuenta con una batería Li-Po de 3,8 V (1100 mAh) que le permite una autonomía de vuelo de aproximadamente 13 minutos en condiciones normales de funcionamiento (sin viento y a una velocidad nominal de 15 km/h). Se pueden seleccionar dos velocidades de vuelo, lenta (con un ángulo máximo de posición de vuelo de 9° y velocidad máxima de 14.4 km/h), y rápida (con un ángulo máximo de posición de vuelo de 25° y una velocidad máxima de 28.8 km/h). Además, está equipado con un barómetro que mide la presión atmosférica, utilizado para estimar la altitud relativa, este sensor trabaja en conjunto con el VPS para mantener una altura constante durante el vuelo, además, tiene integrado un sensor IMU (Unidad de Medición Inercial) que incluye un acelerómetro y giroscopio que permiten obtener la aceleración y rotación, respectivamente. Estos datos se utilizan para la estabilización del vuelo, permitiendo mantener su orientación y compensar movimientos involuntarios. Tello no tiene un sistema de refrigeración integrado y solo se enfría con el flujo de aire que se genera durante el vuelo.

La estructura general del hardware de control se divide en dos partes, ver figura 2. En primer lugar, se tiene un controlador de vuelo a bordo que proporciona funciones básicas como estabilización de vuelo y regulación de altitud, control de velocidad de traslación y ejecución de trayectoria simple. Esta parte del sistema es una arquitectura de hardware cerrado, operado con un protocolo de comandos dedicados. En segundo lugar, se puede lograr un mayor nivel de autonomía utilizando el protocolo de control basado en comandos UDP (*User Datagram Protocol*) para la comunicación entre el Tello y un dispositivo externo de control, como un smartphone, tableta o computadora. Este enfoque de arquitectura dividida y distribuida permite superar las limitaciones de tamaño y peso del hardware en los mini drones, para aprovechar estaciones de control externas con mayor capacidad de procesamiento para gestionar una planificación de trayectorias y control de vuelo de alto nivel. El Tello se encuentra configurado de fábrica como un punto de acceso inalámbrico para que se conecte dispositivo externo. La comunicación se realiza a través de una conexión Wi-Fi 802.11n de 2,4 GHz.

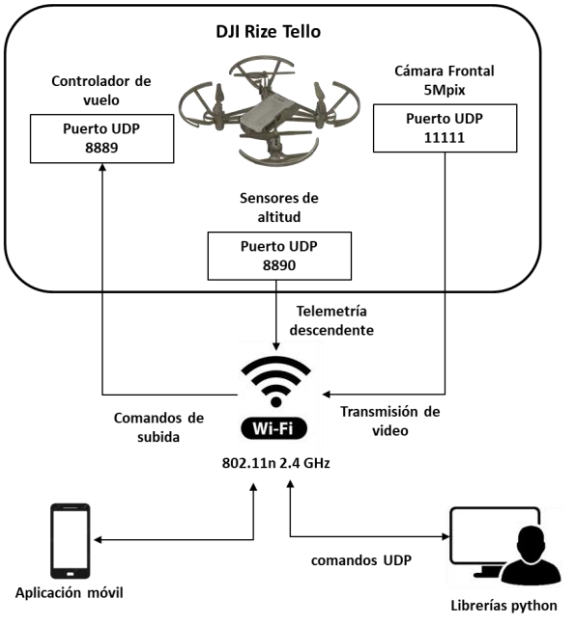

Figura 2. Arquitectura del sistema de comunicación

Por otro lado, con el kit de desarrollo de software (SDK 2.0) oficial del Tello se dispone de una amplia variedad de comandos posibles para su programación, desde simples movimientos de traslación y rotación hasta la realización de trayectorias de vuelo más sofisticadas. El protocolo proporciona un mecanismo de confirmación de recibo de comandos de subida y un conjunto de códigos de error devueltos por el controlador de vuelo. La transmisión de vídeo se realiza con la cámara integrada a la que se puede acceder de forma independiente a través de una comunicación IP estándar. La telemetría descendente proporcionar al usuario los datos de estado del Tello, como la orientación (balanceo, cabeceo y guiñada), velocidades y aceleraciones de traslación, nivel de energía de la batería y altitud actual obtenida tanto por la presión barométrica como por el sensor de tiempo de vuelo (ToF). El flujo de información se realiza a través de puertos UDP diferentes, lo que permite una comunicación *full-duplex*.

El Tello puede ser programado utilizando varias plataformas y lenguajes de programación, convirtiéndolo en una herramienta versátil tanto para la educación como para la investigación y para aplicaciones más avanzadas. Entre las opciones más accesibles para usuarios con un nivel básico se encuentra Scratch, un lenguaje de programación visual que permite desarrollar programas mediante bloques de código predefinidos, facilitando el aprendizaje de conceptos básicos de programación, robótica y control sin la necesidad de conocimientos

previos. Además, el drone Tello es compatible con aplicaciones móviles mediante el uso de la API Tello SDK, que permite la creación de aplicaciones personalizadas para el control desde dispositivos móviles, potenciando su capacidad para aplicaciones comerciales y recreativas.

Para el desarrollo de aplicaciones de nivel intermedio, existen diversas librerías disponibles para operar el Tello utilizando el lenguaje de programación de Python, aunque varían en sus aplicaciones, las librerías se basan en el SDK oficial del Tello y están estructuradas de acuerdo con el protocolo de comandos proporcionado por el fabricante. Uno de los paquetes más populares es la librería Tello-Python, desarrollada por la propia compañía DJI [19]. Esta librería incluye una amplia gama de funciones que permiten a los usuarios controlar los movimientos del Tello de manera sencilla, mediante el uso de comandos predefinidos. Además, incluye múltiples aplicaciones de ejemplo, que van desde el control mediante uso del teclado hasta la utilización de la cámara para crear un sistema de visión que realiza el reconocimiento de la pose humana y a partir de ella realiza el control de vuelo.

Dada la versatilidad del Tello, existe una comunidad de entusiastas que han desarrollado diferentes librerías basadas en Python que permiten ampliar sus capacidades. Entre ellas, tal vez la más destaca, es el paquete DJITelloPy realizado por Damià Fuentes Escoté que amplía las capacidades de la librería oficial proporcionando una interfaz simplificada para el control de movimientos, captura de video en tiempo real y recepción de datos de telemetría [20]. Este paquete también incluye características avanzadas que permiten el control en configuraciones de enjambre para el modelo DJI Tello Edu, lo que es particularmente útil en estudios de robótica colaborativa. La librería TelloPy ofrece una interfaz más avanzada, permitiendo no solo la operación simultánea de múltiples drones, sino también el acceso a los sensores integrados, lo que facilita su aplicación en proyectos de monitoreo y sistemas de navegación autónoma [21]. Finalmente, el paquete easyTello desarrollado por Ezra Fielding se destaca por su enfoque en la simplicidad, proporcionando una manera accesible para que los usuarios envíen comandos [22]. Aunque es más básica que las anteriores, esta librería ha sido desarrollada para facilitar el aprendizaje de la teoría de control, que la convierte en una herramienta ideal para la educación en robótica y automatización.

Para aplicaciones más complejas, es posible utilizar plataformas de desarrollo como ROS (*Robot Operating System*) existiendo también una gran variedad de librerías diseñadas para el control del Tello. Entre las librerías más destacadas se encuentra tello_driver, que proporciona nodos ROS para el control básico, incluyendo despegue, aterrizaje y comandos de vuelo, así como la transmisión de video en tiempo real [23]. La librería tello_ros, permite el control de movimiento y facilita la captura de imágenes y video, el acceso a datos de telemetría, y la ejecución de tareas autónomas mediante la programación de trayectorias predefinidas [24]. Además, la comunidad de ROS ha desarrollado aplicaciones adicionales como la librería Tello_Drone_In_Gazebo, que permite la simulación del drone en entornos virtuales a través de Gazebo, un simulador físico 3D, proporcionando un entorno seguro para el desarrollo y prueba de algoritmos antes de su implementación en el drone real [25].

3. Plataforma experimental

La figura 3 muestra el diagrama esquemático de la arquitectura propuesta para la plataforma experimental, así como sus componentes e interacciones, está dividida en dos módulos, el de bajo nivel o nodo de control, integrado por una tarjeta de control Rasberry Pi 4 emparejadas con el drone DJI Tello básico mediante Wi-Fi, la tarjeta tiene instalado el sistema operativo Ubuntu Server 20.04, esto se justifica debido a que la plataforma ha sido diseñada como banco de pruebas enfocado al control de drones, robótica de enjambre y robótica cooperativa para fines tanto de investigación como de enseñanza, por lo tanto, es posible utilizar el lenguaje de programación *python* o el *middleware* robótico de ROS *Noetic* que es compatible con el sistema operativo instalado en la Rasberry Pi.

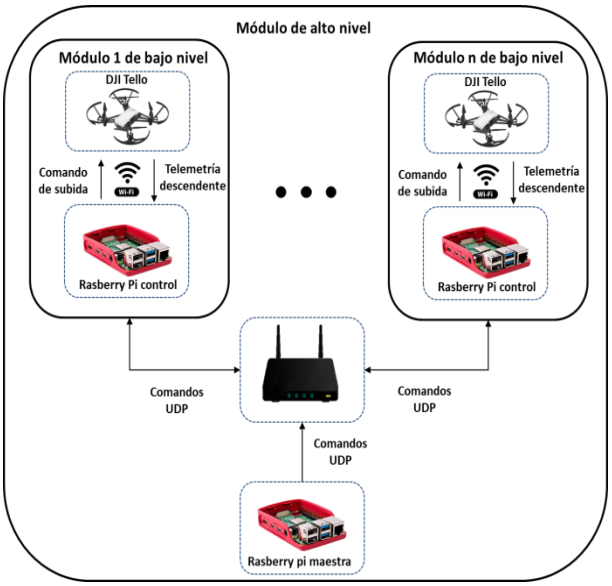

Figura 3. Arquitectura de la plataforma experimental

En la figura 4 se muestra los nodos de control físico, donde se tiene además un monitor con la tarjeta Rasberry Pi montada en su parte posterior, además de sus periféricos y el Tello.

El uso de una tarjeta de control por cada Tello se justifica debido a que el modelo básico no presenta el modo de enjambre de fábrica, se conecta cada drone mediante el mismo puerto UDP 8890, debido a esto no se puede realizar la conexión de varios drones a un único dispositivo externo.

Figura 4. Nodos de control de la plataforma propuesta

El envió de comandos para el control de cada Tello como despegar, aterrizar, activar y desactivar la transmisión de video, moverse hacia arriba, moverse hacia abajo, girar a la izquierda, girar a la derecha, avanzar, retroceder, dar vueltas, entre otros, ha sido probada utilizando las librerías DJITelloPy y la librería socket de Python mediante la transmisión de comandos UDP (*User Datagram Protocol*) de acuerdo con las especificaciones del kit de desarrollo de software (SDK 2.0). Los comandos se envían desde la Rasberry Pi al Tello a través del puerto UDP 8889. Por otro lado, las respuestas y la transmisión de video se envían de regreso desde el Tello a la tarjeta a través de los puertos UDP 11111 y 8890, respectivamente.

El proceso para llevar a cabo la comunicación UDP entre la Rasberry Pi y el Tello para cada nodo de control es mostrado en la figura 5. Inicialmente, el cliente y servidor deben generar una instancia de socket con la dirección IP y puerto correspondiente. A través de los métodos connect() y bind() se realiza el enlace de conexión cliente – servidor, respectivamente. El método Send() del cliente permite enviar la solicitud al drone, que utiliza un método recvfrom() para realizar la recepción y procesar la solicitud, el drone luego utiliza un método Send() para enviar una respuesta que el cliente procesa con su método recvfrom(), cuando el cliente ha procesado y finalizado todas las transferencias y se encuentra listo. Para finalizar los procesos anteriores se debe utilizar un método close() que permita cerrar todos los enlaces de socket.

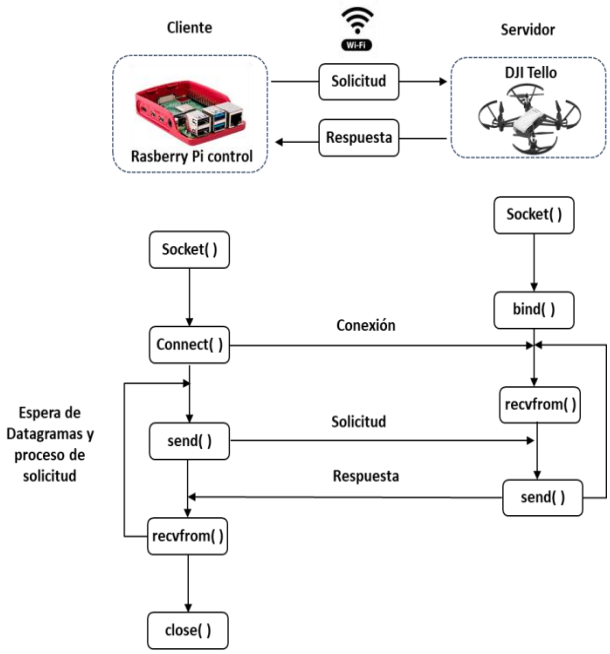

Figura 5. Arquitectura de comunicación.

La comunicación entre el nodo de control y un dispositivo externo se implementa mediante una red LAN utilizando un *router* comercial marca T-Link, ver figura 6. Esta arquitectura permite establecer un esquema de control centralizado al integrar como dispositivo externo una Rasberry Pi maestra, aunque pudiera utilizarse otro dispositivo como una computadora. La tarjeta maestra tiene la función de recibir y controlar la información del estado de cada nodo de control. Además, la configuración presenta la característica de poder escalar la cantidad de drones debido a que solo es requerido conectar un nuevo nodo de control al *router*.

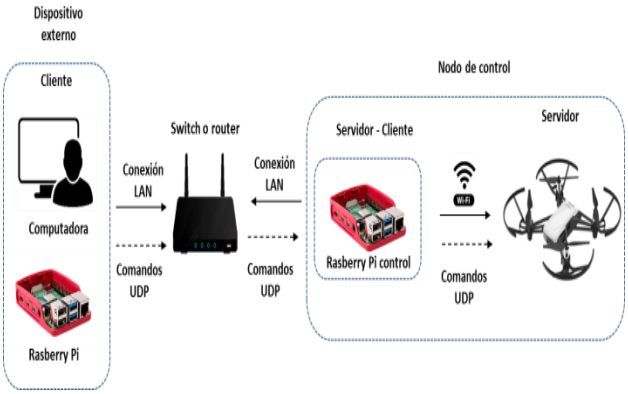

Figura 6. Arquitectura de comunicación entre nodo de control y dispositivo externo.

Para configurar una estructura de control descentralizado para el enjambre en la plataforma, cada nodo debe comunicarse con el nodo vecino a través de la red LAN, esto es, cuando una Raspberry Pi de control recibe un mensaje, automáticamente cambia a modo de envío para

responder, mientras que la otra Raspberry Pi alterna a modo de recepción. De esta forma, las tarjetas pueden comunicarse de manera alternada, garantizando que ambas puedan enviar y recibir mensajes, esta estructura es mostrada en la figura 7.

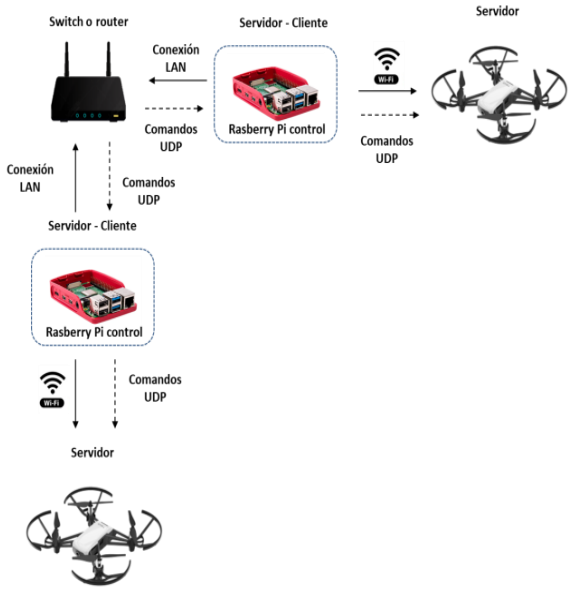

Figura 7. Arquitectura de comunicación descentralizada entre nodos de control.

En la figura 8 se muestra el diagrama a bloques con las funciones realizadas para implementar el control centralizado. El código en el lenguaje de programación en Python se inicia realizando la comunicación entre las Rasberry Pi conectadas a la red a través de la función *sockets()* utilizando la IP asignada en el *router* y configurando la tarjeta maestra como cliente y los nodos de control como servidores.

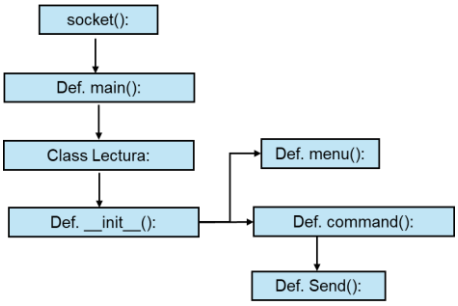

Figura 8. Diagrama de bloques de las funciones para control centralizado.

La función *main()* genera un ciclo continuo de llamada a la clase *Lectura* que ejecuta el método *__init__*, en éste, se llama a dos diferentes funciones *menu()* y *command()*, la función *menu()* despliega las instrucciones de comandos disponibles que pueden enviarse al Tello, ver figura 9, y genera una interfaz gráfica. La función *command()* identifica el comando ingresado, es importante destacar que el diseño para las aplicaciones de control de vuelo se realizó basado en el concepto de un programa de control idéntico que se ejecute en todas las Rasberry Pi en el que se pueda seleccionar la configuración de control (autónomo, remoto o enjambre), con el objetivo de lograr una configuración rápida y lograr la escalabilidad de la plataforma para la integración de nuevos drones en el enjambre.

```
CONTROL POR TECLADO DRON TELLO CON TECLADO
ATERIZAR            -BARRA ESPACIADORA
DESPEGAR            -q
ADELANTE            -w
ATRAS               -s
IZQUIERDA           -a
DERECHA             -d
BATERIA             -b
GIRAR A LA DERECHA -FLECHA DERECHA
GIRAR A LA IZQUIER -FLECHA IZQUIERDA
ASCENDER            -FLECHA ARRIBA
DESCENDER           -FLECHA ABAJO

INICIAR STREAM      -v
RECONOCIMIENTO R    -f
PID SEGUIMIENTO     -p
EMERGENCIA          -e
DETENER             -z
MAPEO               -m
SEGUIMIENTO COLOR   -c
```

Figura 9. Comandos disponibles de control

La figura 10 muestra el diagrama de bloques con las funciones implementadas para el funcionamiento de los nodos de control. El código inicia realizando la comunicación entre las Rasberry Pi y el drone Tello a través de la función *socket()* con la IP asignada de fábrica para conexión Wi-Fi. La función *main()* genera un ciclo continuo de llamada a la clase *Lectura()*, posteriormente, se ejecuta el método __init__, donde se generan tres hilos de ejecución para las funciones *menu()*, *servidor()* y *command()*, la función *servidor()* realiza la comunicación de la Rasberry Pi a través de sockets con la IP asignada en el *router* y la configura como servidor, a su vez ejecuta el hilo de la función *handle_commands()* que tiene la tarea de recibir el dato de la tarjeta maestra y realizar la función solicitada.

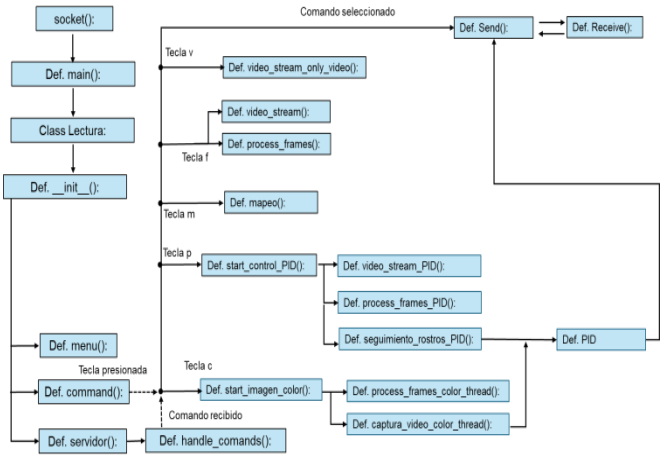

Figura 10. Diagrama de bloques de las funciones programadas en los nodos de control.

Por otro lado, La función *command()* es utilizada para poder identificar el comando ingresado directamente desde el teclado conectado a la tarjeta de control y poder realizar la tarea solicitada mostradas por la función *menu()*. Al presionar o recibir la información del teclado, se realizará el envío de comando UDP para el movimiento del Tello asignado a cada una a través de la función *send()* y el drone responderá si fue realizado o se produjo algún error, dicho mensaje se recibe mediante la función *receive()*.

Para algunas teclas particulares se inician en paralelo algunos hilos que se describen a continuación: La función *video_stream_only_video()*, se encarga de obtener y mostrar los *frames* en una ventana gráfica en tiempo real que proporciona la cámara del Tello, utilizando la librería de opencv. La función *video_stream()* obtiene el video de la cámara del Tello y trabaja en paralelo con la función de *process_frames()* que realiza el procesamiento para el reconocimiento de rostros, la muestra en una ventana gráfica y realiza el cálculo del área identificada y las coordenadas de su centroide. La función *mapeo()* permite dibujar en una ventana gráfica la trayectoria que sigue cada drone con base en los comandos de movimiento recibido. La función *star_control_PID()* ejecuta los hilos encargados de realizar la captura, procesamiento para el reconocimiento de rostros y llamada al controlador PID para que el Tello realice el seguimiento del rostro detectado. La función *star_image_color()* ejecuta los hilos encargados de realizar la captura, procesamiento para el reconocimiento del color y coordenadas de un objeto y llamada al controlador PID para que el Tello realice el seguimiento del objeto detectado.

4. Resultados

Las pruebas para evaluar el comportamiento de la plataforma se llevaron a cabo en un espacio interior para asegurar un entorno controlado y sin variables externas que pudieran afectar el rendimiento. Al realizar las pruebas bajo techo, fue posible evaluar la estabilidad y respuesta de los drones sin la influencia de factores como el viento o controlar los cambios en la iluminación, lo que garantizó una evaluación de su desempeño bajo condiciones ideales.

La prueba 1 se realizó para el control autónomo a una altura deseada de un solo drone, en la figura 11 se muestra la interfaz gráfica de la aplicación desarrollada por los autores.

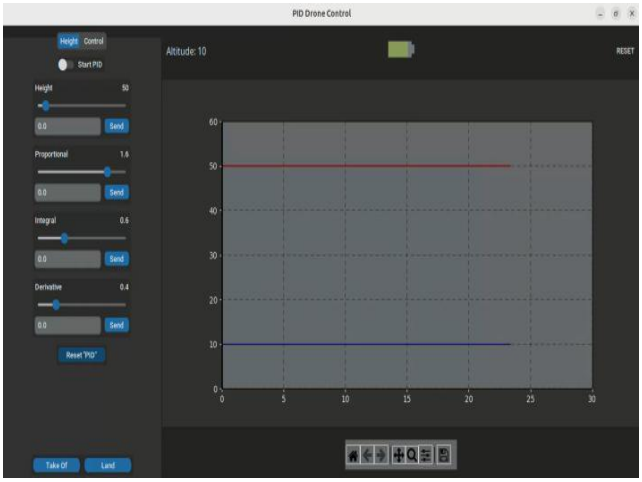

Figura 11. Interfaz gráfica para control del Tello.

En la interfaz se puede seleccionar el modo de vuelo manual o autónomo, además de poder visualizar gráficamente la posición actual y el error entre la posición deseada y la actual. Si el botón “start PID” se encuentra en la posición izquierda, se habilita el control remoto, con lo cual es posible utilizar el teclado para mover el drone con los comandos mostrados en la figura 12.

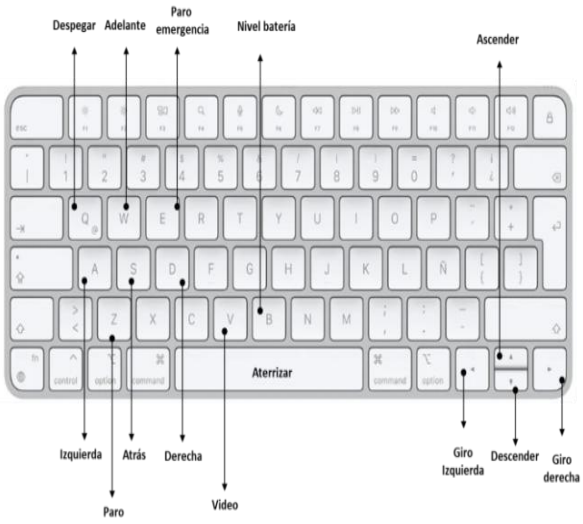

Figura 12. Teclas para el control de vuelo del drone.

Si el botón “start PID” se encuentra en la posición derecha, se habilita el control autónomo, con lo cual es posible indicar una altura deseada. La figura 13 muestra el comportamiento del drone dada una altura de referencia deseada, en la parte superior se encuentra la altura de referencia en color rojo y en color azul la posición actual del Tello, en la parte inferior en color amarillo se muestra el error entre la altura deseada y la real del drone.

Figura 13. Gráfica de control de altura del Tello.

La prueba 2 correspondió al control para el seguimiento de rostros o de objetos considerando su color. El principio de funcionamiento se muestra en la figura 14 para el seguimiento de rostros y que es el mismo concepto utilizado para el seguimiento de objetos, se basa en definir un tamaño de área deseado para el rostro u objeto detectado y con base en la diferencia que se presente el controlador PID ajusta el movimiento del drone para mantener en el centro la imagen y al tamaño de área especificado.

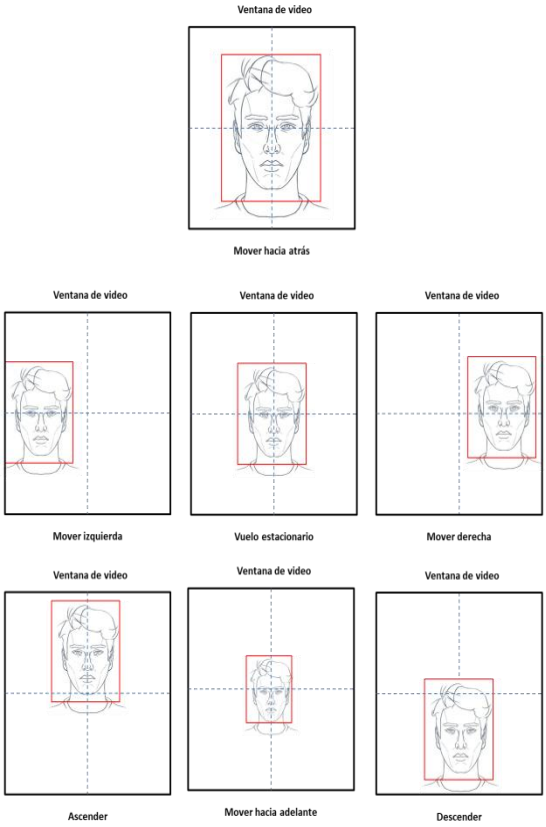

Figura 14. Control para detección de rostros.

La figura 15 muestra la gráfica del error en posicionamiento considerando únicamente un desplazamiento del rostro hacia la derecha.

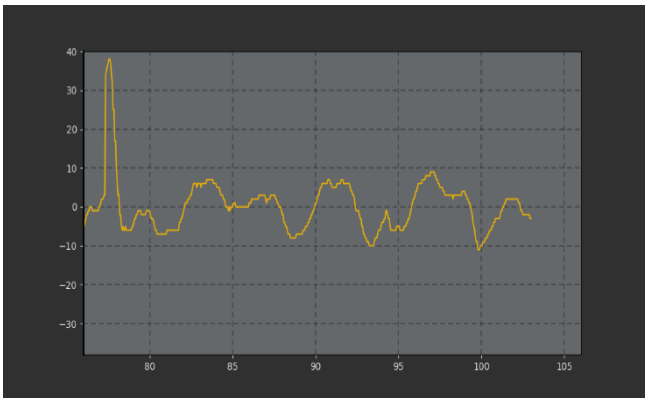

Figura 15. Error de seguimiento de rostros.

Por otro lado, las pruebas realizadas para evaluar el comportamiento de la plataforma en modo enjambre se implementaron utilizando dos drones. La prueba 3 corresponde a un control descentralizado, los drones alternaban él envió del comando a una altura deseada. Se utilizó la misma interfaz gráfica mostrada en la figura 11, donde debe estar seleccionada la opción “control” en la parte superior para trabajar el modo enjambre. La figura 16 muestra la gráfica de la señal del sensor de altura de cada drone.

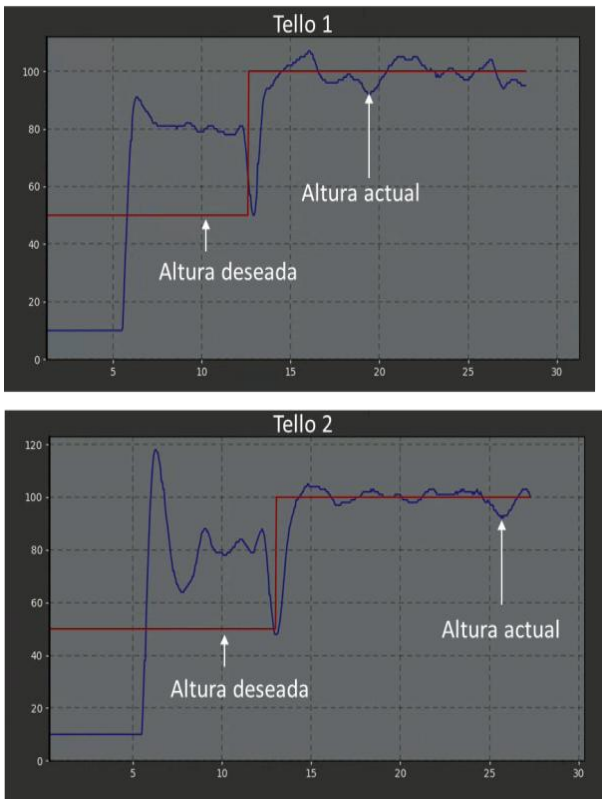

Figura 16. Graficas de respuesta para altura

Inicialmente durante el despegue de los drones, el nodo de control del Tello 2 envía el comando, ambos drones se estabilizan a una altura de 80 cm que viene programada de fábrica, posteriormente, se cambia a un control

autónomo que tiene predefinido una referencia de altura de 50 cm, inmediatamente después se envía una referencia de 100 cm, como se observa en la gráfica los drones responden para alcanzar la altura de 50 cm al momento de realizar el cambio al control PID y posteriormente suben hasta alcanzar la altura de 100 cm, en esta etapa el nodo de control del Tello 1 envió el comando de la altura deseada.

Para las tres pruebas presentadas se realizó un análisis estadístico considerando una muestra de 100 repeticiones por algoritmo para cuantificar el margen de error que presentan con respecto a la posición deseada, en cada prueba debido a que los drones no se mantienen en una posición constante se obtuvo el promedio del error generado durante 1 minuto una vez alcanzada la posición de referencia. En la figura 17 se muestra la gráfica de caja y bigotes de los resultados obtenidos para cada prueba realizada. En la prueba 1 se tiene una mediana de 8 y una desviación estándar de 6.9, en la prueba 2 se obtuvo una mediana de 13 y una desviación estándar de 7.7, finalmente en la prueba 3 para el drone 1 se obtuvo una mediana de 11 y una desviación estándar de 7.6 y para el drone 2 se obtuvo una mediana de 9 y una desviación estándar de 7.1.

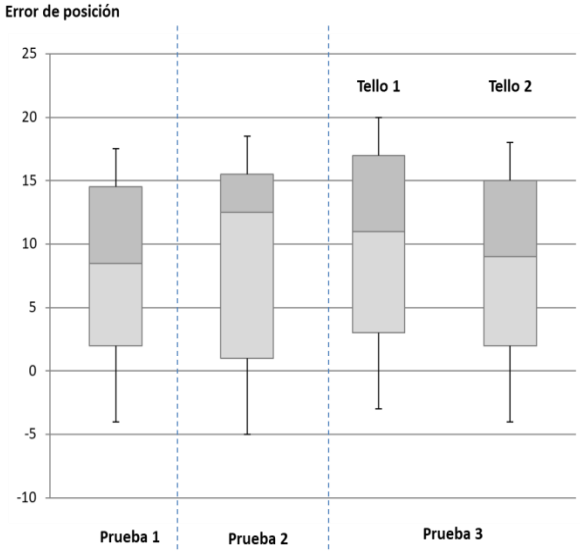

Figura 17. Error de posición de altura.

La última prueba realizada correspondió a un control centralizado, donde la tarjeta maestra permite enviar comandos de vuelo mediante el teclado a ambos drones, la evaluación realizada fue cualitativa en base a la observación verificando si en los drones se genera un desfasamiento en la formación, esto es, debido a que no se tiene un sensor abordo y un sistema de control PID que permita controlar la formación del grupo para los movimientos laterales de manera similar al control de altura. Los resultados observados permitieron concluir que los movimientos hacia adelante y hacia atrás generan un menor desfasamiento de la formación comparados con los movimientos hacia la derecha y hacia la izquierda. Una solución a futuro para controlar de forma más precisa la formación durante estos movimientos será utilizar la cámara de los drones y el controlador PID para su alineamiento utilizando el enfoque realizado en las aplicaciones de detección de rostros y objetos. La figura 18 muestra imágenes de la secuencia de movimiento de ambos drones de la prueba realizada para el enjambre mediante un control centralizado.

Como resultado de la serie de pruebas realizadas para el grupo de vehículos aéreos no tripulados, se encontró que la iluminación, el nivel de energía disponible en la batería y la superficie debajo de los Tellos son características que deben controlarse ya que pueden afectan su comportamiento en el vuelo en interiores.

Por otro lado, a pesar de tener solo dos nodos configurados actualmente, la solución desarrollada nos permite la escalabilidad deseada con solo configurar en la red LAN los nuevos nodos de control que se van integrando al enjambre, esto es, en gran medida debido al enfoque de diseño de un programa de control idéntico que se ejecuta en todas las Raspberry Pi de la plataforma. Además, el que cada drone este controlado con una Raspberry Pi permite ampliar las capacidades de procesamiento a bordo de este sistema, ya que la potencia que tiene cada dispositivo solo se dedica a administrar su drone emparejado, lo que permite que cada Rasberry Pi puedan programarse tareas más complejas.

Figura 18. Pruebas en interior

5. Conclusiones

En el presente trabajo se consideró el modelo del drone Tello básico para desarrollar una plataforma de control para un vuelo coordinado de vehículos aéreos no tripulados, debido a que presenta la ventaja de poder ser programado a través de lenguajes como *Python*, *Swift*, *Scratch* y ROS utilizando el kit de desarrollo de *software* del fabricante (Tello SDK 2.0) lo que permite implementar desde simples hasta complejas tareas para un Tello o un grupo de drones.

El desarrollo de la arquitectura de control distribuido propuesta a través de la conexión del Tello a una tarjeta *Rasberry Pi* y a su vez las tarjetas a una red LAN generada por un *router* comercial se justifican debido a que este modelo no presenta de fábrica la opción de controlar el vuelo coordinado de un conjunto de drones. Además, la configuración permite distribuir el procesamiento de la tarea en dos niveles, el dedicado al protocolo de comunicación Wi-Fi y del procesamiento de la tarea individual, y el nivel dedicado a la planificación de la tarea y coordinación grupal de los UAV. Más aún, esta configuración permite la implementación de una arquitectura de control tanto centralizada como descentralizada para el vuelo en formación de drones.

La evaluación inicial de su funcionamiento para el control de un enjambre se realizó utilizando solo dos drones, sin embargo, la arquitectura propuesta y el diseño de la aplicación de control basada en un programa idéntico en todas las *Rasberry Pi*, permitirá a futuro conectar una mayor cantidad de drones, realizando sencillas configuraciones como lo es la asignación de las IP de los nuevos drones conectados, lo que permite una fácil escalabilidad. Además, en esta primera etapa se desarrolló los programas basados en el SDK y el lenguaje de programación *Python* para el control de vuelo de un solo drone y para el control de vuelo de dos drones, sin embargo, está habilitada para que en un futuro puedan implementarse aplicaciones utilizando el *middleware* robótico de ROS Noetic.

La descripción de las funciones para generar los programas para las pruebas desarrolladas y el modelo de intercambio de información entre el Tello y las tarjetas de control, así como las aplicaciones desarrolladas para el control de vuelo de uno y dos drones permiten concluir el potencial que tiene la plataforma propuesta para su aplicación en temas de control, robótica cooperativa y robótica de enjambre.

Referencias

[1] Singhal, G.; Bansod, B.; Mathew, L. Unmanned Aerial Vehicle Classification, Applications and Challenges: A Review. Preprints 2018, 2018110601. https://doi.org/10.20944/preprints201811.0601.v1

[2] PS, R., & Jeyan, M. L. (2020). Mini Unmanned Aerial Systems (UAV)-A Review of the Parameters for Classification of a Mini UAV. International Journal of Aviation, Aeronautics, and Aerospace, 7(3), 5. https://doi.org/10.15394/ijaaa.2020.1503

[3] Hejase, M., Noura, H., & Drak, A. (2015). Formation flight of small scale unmanned aerial vehicles: a review. Control Theory: Perspectives Applications and Developments, 221-247. https://doi.org/10.1002/9781634827300ch10

[4] Coppola, M., McGuire, K. N., De Wagter, C., & De Croon, G. C. (2020). A survey on swarming with micro air vehicles: Fundamental challenges and constraints. Frontiers in Robotics and AI, 7, 18. https://doi.org/10.3389/frobt.2020.00018

[5] Liu, Y., & Bucknall, R. (2018). A survey of formation control and motion planning of multiple unmanned vehicles. Robotica, 36(7), 1019-1047. https://doi.org/10.1017/S0263574718000218

[6] Yasin, J. N., Mohamed, S. A., Haghbayan, M. H., Heikkonen, J., Tenhunen, H., & Plosila, J. (2020). Unmanned aerial vehicles (uavs): Collision avoidance systems and approaches. IEEE access, 8, 105139-105155. https://doi.org/10.1109/ACCESS.2020.3000064

[7] Campion, M., Ranganathan, P., & Faruque, S. (2018, May). A review and future directions of UAV swarm communication architectures. In 2018 IEEE international conference on electro/information technology (EIT) (pp. 0903-0908). IEEE. https://doi.org/10.1109/EIT.2018.8500274

[8] Dias, P.G.F., Silva, M.C., Rocha Filho, G.P., Vargas, P.A., Cota, L.P., Pessin, G., (2021), Swarm robotics: A perspective on the latest reviewed concepts and applications. Sensors 21(6), 2062. https://doi.org/10.3390/s21062062

[9] Do, H. T., Hua, H. T., Nguyen, M. T., Nguyen, C. V., Nguyen, H. T., Nguyen, H. T., & Nguyen, N. T. (2021). Formation control algorithms for multiple-uavs: a comprehensive survey. EAI Endorsed Transactions on Industrial Networks and Intelligent Systems, 8(27), e3-e3. https://doi.org/10.4108/eai.10-6-2021.170230

[10] Zhou, Y., Rao, B., & Wang, W. (2020). UAV swarm intelligence: Recent advances and future trends. IEEE Access, 8, 183856-183878. https://doi.org/10.1109/ACCESS.2020.3028865

[11] Rafifandi, R., Asri, D. L., Ekawati, E., & Budi, E. M. (2019). Leader–follower formation control of two quadrotor UAVs. SN Applied Sciences, 1, 1-12. https://doi.org/10.1007/s42452-019-0551-z

[12] Low, C. B., & San Ng, Q. (2011, December). A flexible virtual structure formation keeping control for fixed-wing UAVs. In 2011 9th IEEE international conference on control and automation (ICCA) (pp. 621-626). IEEE. https://doi.org/10.1109/ICCA.2011.6137876

[13] Wang, X., Chen, G., Gong, H., & Jiang, J. (2021). UAV swarm autonomous control based on Internet of Things and artificial intelligence algorithms. Journal of Intelligent & Fuzzy Systems, 40(4), 7121-7133. https://doi.org/10.3233/JIFS-189541

[14] Yan, M. D., Zhu, X., Zhang, X. X., & Qu, Y. H. (2017). Consensus-based three-dimensionalmulti-UAV formation control strategy with high precision. Frontiers of Information Technology & Electronic Engineering, 18(7), 968-977. https://doi.org/10.1631/FITEE.1600004

[15] Do, H. T., Hua, H. T., Nguyen, M. T., Nguyen, C. V., Nguyen, H. T., Nguyen, H. T., & Nguyen, N. T. (2021). Formation control algorithms for multiple-uavs: a comprehensive survey. EAI Endorsed Transactions on Industrial Networks and Intelligent Systems, 8(27), e3-e3. https://doi.org/10.4108/eai.10-6-2021.170230

[16] Arnold, R., Carey, K., Abruzzo, B., & Korpela, C. (2019). What is a Robot Swarm: A Definition for Swarming Robotics. In 2019 IEEE 10th Annual Ubiquitous Computing, Electronics & Mobile Communication Conference (UEMCON)(pp. 0074-0081). https://doi.org/10.1109/UEMCON47517.2019.8993024

[17] Lv, C., Yu, R., Cao, J., Gong, C., Wu, W., & Wang, X. (2022, August). A survey on unmanned aerial vehicle swarm communication and navigation. In International Conference on Guidance, Navigation and Control (pp. 2386-2393). Singapore: Springer Nature Singapore. https://doi.org/10.1007/978-981-19-6613-2_232

[18] Wickramasinghe, O. E., Rajan, N., Wanniarachchi, C., Karunanayaka, K., & Wimalaratne, P. (2024, June). Evolution of Formation Control Algorithms for Unmanned Aerial Vehicles. In 2024 IEEE 33rd International Symposium on Industrial Electronics (ISIE) (pp. 1-7). IEEE. https://doi.org/10.1109/ISIE54533.2024.10595718

[19] DJI-SDK, Tello-Python, (2018), GitHub repository, https://github.com/dji-sdk/Tello-Python.

[20] D.F.Escoté, djitellopy, (2024), GitHub repository, https://github.com/damiafuentes/DJITelloPy

[21] hanyazou, Tellopy, (2018), GitHub repository, https://github.com/hanyazou/TelloPy.

[22] Ezra Fielding, easyTello, (2019), GitHub repository, https://github.com/ezrafielding/easyTello

[23] Anqi Xu., (2004). tello_driver, Repositorio en GitHub. https://github.com/anqixu/tello_driver

[23] Clyde McQueen, (2019). tello_ros, Repositorio en GitHub. https://github.com/clydemcqueen/tello_ros

[24] YocoZuna, (2023). Tello_Drone_In_Gazebo, Repositorio en GitHub. https://github.com/YocoZuna/Tello_Drone_In_Gazebo.