Estructura de hardware y software para el hospedaje y ejecución de rutinas de comportamiento autónomo en vehículos aéreos no tripulados

Hardware and software structure for hosting and executing autonomous behavior routines in unmanned aerial vehicles

Pablo Andrés Araya Castillo[1]

Araya-Castillo, P. Estructura de hardware y software para el hospedaje y ejecución de rutinas de comportamiento autónomo en vehículos aéreos no tripulados. Tecnología en marcha. Edición especial Movilidad Estudiantil 7. Abril, 2020. Pág.54-64.

https://doi.org/10.18845/tm.v33i6.5167

1 Estudiante de Ingeniería Mecatrónica, Instituto Tecnológico de Costa Rica / Grupo Integrado de Ingeniería de la Universidad de la Coruña, España. Correo electrónico: pabloaraya446@gmail.com

Palabras clave

UAV; autónomo; Pixhawk; mini PC; CCT; puerta de enlace.

Resumen

El Grupo Integrado de Ingeniería (GII) es un grupo de I+D de la Universidad de La Coruña que posee una línea de investigación de vehículos aéreos no tripulados (UAV, del inglés *unmanned aerial vehicle*). Han trabajado en múltiples aplicaciones con UAVs usando el autopiloto Pixhawk. Los vehículos reciben instrucciones de un mando remoto a la vez que envían información del estado de vuelo a una computadora de control en tierra (CCT) para visualizar lo que está pasando durante la ejecución de una misión.

Sin embargo, los UAVs eran incapaces de ejecutar tareas que involucraran comportamiento autónomo, además de que la comunicación con la CCT no era suficientemente fluída para visualizar el vuelo correctamente.

Para solucionar el problema se implementó una estructura funcionalmente flexible de hardware y software en un UAV, que incluyó una mini PC a bordo en la cual se pueden programar rutinas de vuelo tanto autónomas como no autónomas, que le dan instrucciones al Pixhawk directamente. Además en una CCT se visualiza el estado de vuelo con una comunicación más fluída y las decisiones que está tomando el vehículo.

Se verificó el correcto funcionamiento de la estructura a través de simulaciones y se validó con un vuelo de prueba autónomo, en el cual el UAV decidió y ejecutó su propia ruta de acuerdo a las condiciones de la misión programada.

Keywords

UAV; autonomous; Pixhawk; mini PC; GCS; gateway.

Abstract

The Integrated Group for Engineering Research (GII) is an R & D group from the University of A Coruña that has a research line of unmanned aerial vehicles (UAV). They have worked on multiple applications with UAVs using the Pixhawk autopilot. The vehicles receive instructions from a remote control while sends flight status information to a ground control station (GCS), a computer to visualize what is happening during the execution of a mission.

However, the UAVs were unable to perform tasks that involved autonomous behavior, and the communication with the GCS was not fluid enough to visualize the flight correctly.

To solve the problem, a functionally flexible hardware and software structure was implemented in a UAV, which included a mini PC on board in which autonomous and non-autonomous flight routines can be programmed, which give the Pixhawk instructions directly. In addition, a GCS displays the flight status with more fluid communication and the decisions taken by the vehicle.

The correct performance of the structure was verified through simulations and validated with an autonomous test flight, in which the UAV decided and executed its own route according to the conditions of the programmed mission.

Introducción

Los vehículos aéreos no tripulados o UAVs (del inglés *unmanned aerial vehicles*), tienen una enorme cantidad de usos, por ejemplo militares, en metrología, inspección, captura de imágenes y video, seguridad, entre otros. Los hay de dos tipos según el tipo de sustentación que les permite volar: de ala fija y multicópteros o multirrotores. Estos últimos son de importancia en este trabajo, ya que el mismo se desarrolló sobre este segundo tipo de UAVs.

Los multicópteros poseen una serie de motores que hacen girar hélices que por su forma, al interaccionar con el aire, crean una fuerza de sustentación, lo cual permite el vuelo. Sin embargo, necesitan de una estructura de hardware-software que permita recibir instrucciones de movimiento y pueda gestionar los actuadores para cumplir la misión: esta es la función de lo que se conoce como un autopiloto, un dispositivo con múltiples entradas, que incluyen instrucciones de un usuario y sensores necesarios para que tome decisiones, así como salidas eléctricas para motores e informar sobre el estado del vuelo.

Ardupilot es un software de autopilotos de código abierto, lo cual favorece toda una comunidad pendiente de su progreso y mejora, con lo que se obtiene documentación fácilmente, razón por la que es ampliamente usado [1]. Uno de los autopilotos que respalda ardupilot es el Pixhawk, el cual es de hardware abierto y es muy usado para controlar varios tipos de vehículos, incluyendo UAVs. Dicho esto, se puede entender la estructura de control que suelen tener los UAVs, misma que se presenta en la figura 1.

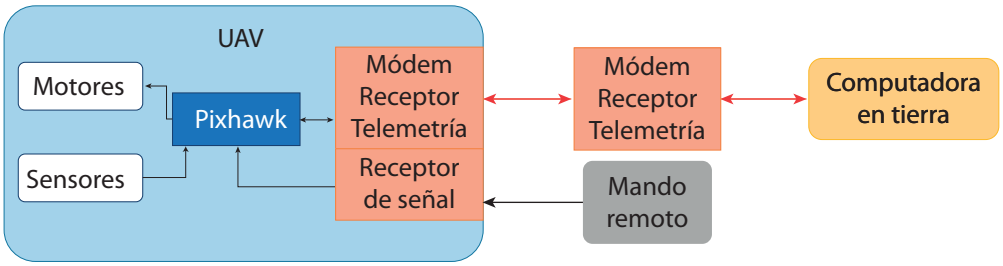

Figura 1. Estructura común de control de un UAV.

En la figura 1 se ve que a bordo del UAV se encuentra su autopiloto (en este caso el Pixhawk), al cual le llegan órdenes de movimiento desde un mando remoto, así como la información de vuelo que recibe de sensores como GPS, aceleración, velocidad, inclinación, etc, con lo cual genera salidas eléctricas para los motores (normalmente pasan por una interfaz de potencia) de manera que se ejecuten las instrucciones. Por su parte, el Pixhawk es capaz de enviar toda esa información de vuelo (comúnmente llamada telemetría) hasta una computadora de control en tierra (CCT) para visualizar lo que ocurre durante el vuelo por medio de dos módems.

En la CCT debe haber un software de vuelo, el cual permite la visualización del estado de vuelo en un entorno gráfico, así como simular rutinas de vuelo. Un ejemplo es Mission Planner, que además es compatible con los productos de Ardupilot (los cuales emplean el protocolo de comunicación denominado mavlink). En la figura 2 se muestra el estado de vuelo de la simulación de una misión con Mission Planner. A la izquierda se observa una serie de indicadores como la inclinación del UAV, la dirección, altura, rapidez con respecto a tierra, entre otros. Por su parte, a la derecha se muestra la ruta que el vehículo está haciendo, la dirección instantánea (línea negra) y la localización gps (en la parte inferior). Todo lo anterior se va actualizando conforme avanza el tiempo.

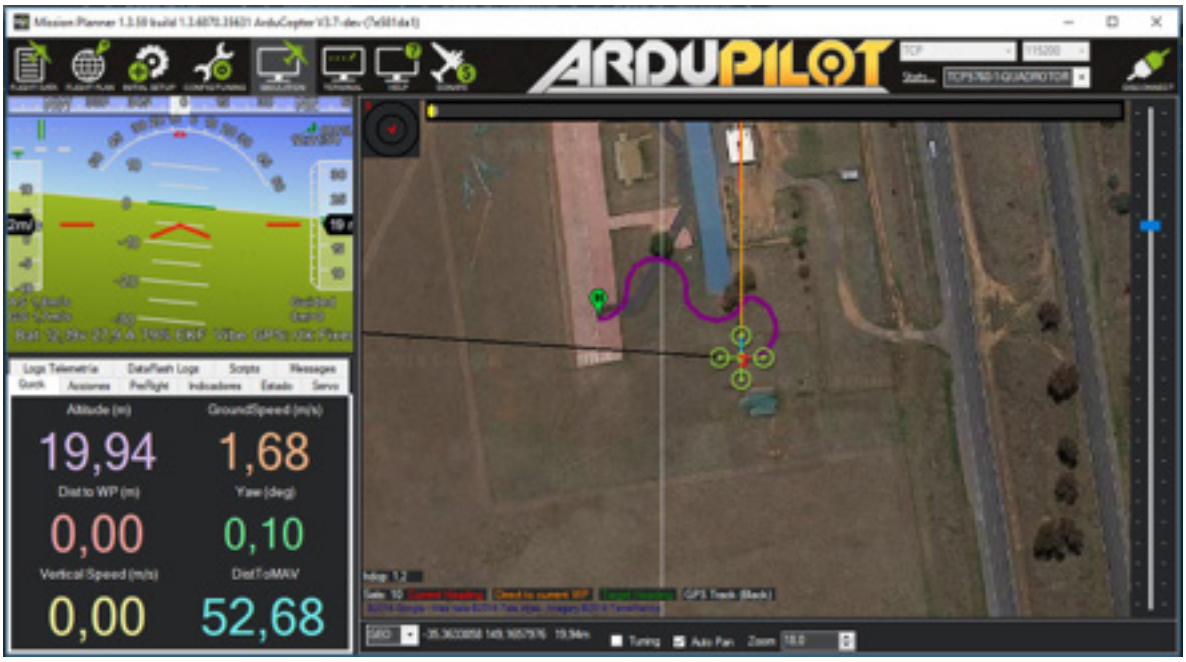

Figura 2. Simulación de un vuelo en Mission Planner *[2]*.

En infinidad de aplicaciones a los UAVs se les incorpora sensores y/o actuadores para llevar a cabo alguna misión en particular. Sin embargo, cuando la tarea a realizar requiere de un comportamiento autónomo en el que el UAV defina sus propias rutas y manera de emplear sus recursos, la estructura de control de la figura 1 se queda corta, ya que no hay ningún elemento en el UAV para que pueda definir sus propias instrucciones y ejecutarlas, dada una serie de requerimientos.

En el Grupo Integrado de Ingeniería de la Universidad de La Coruña, España, se identificó este problema y con la colaboración de este autor, se optó por tomar un octocóptero (multicóptero de ocho hélices) y dotarlo de una estructura de hardware y software para el hospedaje y ejecución de rutinas de comportamiento autónomo, donde destaca una mini computadora a bordo que alberga y ejecuta los programas de vuelo. Estos últimos se realizan con la biblioteca *dronekit* de Python, la cual puede interactuar directamente con un Pixhawk.

Materiales y métodos

Un esquema general de la estructura implementada en el UAV se presenta en la figura 3. En este se observa la mini PC que se le añadió a bordo (una Intel NUC), la cual le da instrucciones al Pixhawk, mientras que este le regresa el valor de todas las variables que describen el estado de vuelo. Esta misma información es enviada a la computadora de control de tierra (CCT) a través de un par de dispositivos emisor (a bordo del octocóptero) y receptor (con la PC de tierra) de telemetría. Este mismo canal de comunicaciones se usó para acceder al terminal de la NUC desde la CCT para ejecutar los programas de vuelo y visualizar las decisiones que está tomando el UAV, mientras que se sigue observando el estado de vuelo en Mission Planner y el usuario podría decidir tomar el control de la nave con el mando remoto en cualquier momento.

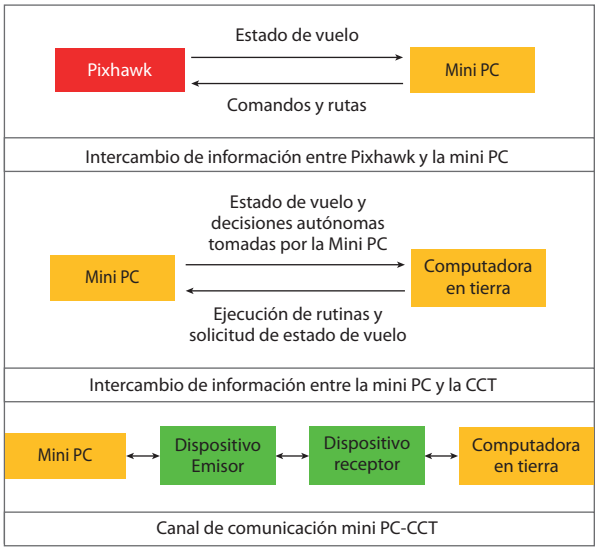

Figura 3. Esquema general de la estructura implementada.

Red inalámbrica

Los dispositivos emisor y receptor son dos puente Ethernet inalámbrico/puerta de enlace (pueden cumplir ambas funciones) de la serie *IPn20 de Microhard Systems INC* conectados en una topología punto a punto, donde el maestro es el terrestre y el esclavo el aéreo.

En la figura 4 se muestra la red operando (se hizo un *ping* desde la CCT a la mini PC); a la izquierda está la NUC y a la derecha la CCT con el terminal abierto mostrando la información del *ping*, cada una conectada por un cable Ethernet a sus respectivas puertas de enlace (negro en la mini PC y blanco en la CCT), las cuales tienen sus respectivas antenas (negra y pequeña en la puerta de enlace aérea, blanca y grande en la de tierra) y baterías (rectangulares y blancas).

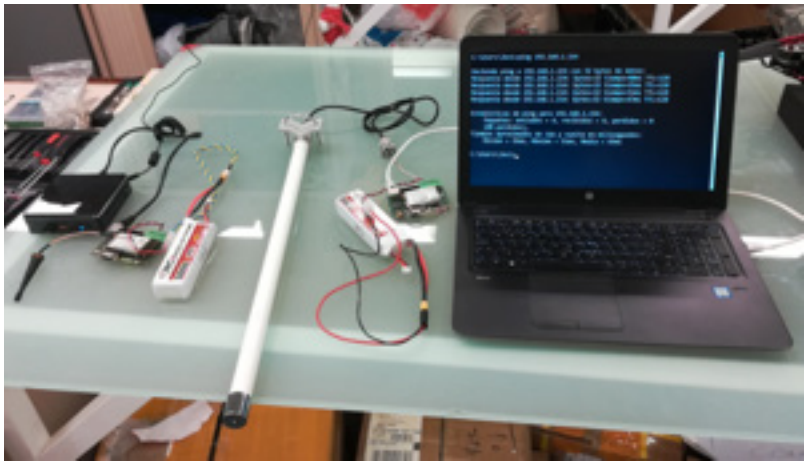

Figura 4. Red Inalámbrica.

En el cuadro 1 se muestran las direcciones IP asignadas a cada elemento de la red para que el lector comprenda las siguientes secciones del documento. Sin embargo, si se deseara replicar esta estructura, el desarrollador es libre de elegir otras direcciones, máscara de subred, así como otros elementos que realicen funciones similares.

Cuadro 1. Direcciones IP de los elementos de red.

Elemento	Dirección IP
CCT	192.168.1.155
Mini PC	192.168.1.154
Puente aéreo	192.168.1.253
Puente de tierra	192.168.1.254
Máscara de subred: 255.255.255.0	

Direccionamiento de información entre el Pixhawk, mini PC y CCT

Con la red funcionando ya se tenía el canal para el intercambio de información entre el UAV y la CCT, sin embargo, aún no se estaban transmitiendo datos. El Pixhawk requiere comunicarse tanto con los programas de vuelo en la mini PC como con la CCT en donde se visualiza el estado de vuelo. Por otro lado, desde la PC de tierra debe tenerse control de la mini PC para ejecutar programas que estén en ella cuando se desee, a la vez que deben visualizarse sus decisiones en la CCT.

Enrutador virtual

El Pixhawk se conecta desde su puerto mini USB a un puerto USB de la mini PC con un cable que tenga ambos puertos. Normalmente la PC le asignará a dicho puerto USB el nombre de COM4, aunque debe verificarse. Para direccionar la información de telemetría se creó un "enrutador virtual" en la mini PC empleando el software mavproxy (un software de vuelo con funcionalidades adicionales de direccionamiento de paquetes mavlink). En la figura 5 se muestran las líneas de código en el terminal de la mini PC que se requieren.

Figura 5 Enrutador virtual en la mini PC. Fuente: Elaboración propia.

Una vez que se accede a la carpeta donde se encuentra mavproxy la señal del Pixhawk (puerto COM4 en este caso) es replicada a dos puertos UDP, uno interno en la NUC para el procesamiento de las instrucciones autónomas con dirección IP 127.0.0.1:14551 y otro puerto interno de la CCT con dirección IP 192.168.1.155:14552. Conviene mencionar que la dirección 127.0.0.1 es la que usa una computadora para llamarse a sí misma y al añadir el ":14551" hace referencia a su puerto UDP interno 14551.

De esta manera, las rutinas de comportamiento autónomo en la mini PC no "llaman" al COM4 para conectarse con el Pixhawk, sino al puerto UDP 14551, mientras que el Mission Planner en la CCT se conecta al UDP 14552 para visualizar el estado de vuelo.

Es deseable que la mini PC ponga a funcionar el enrutador sin necesidad de que un usuario tenga que digitar esas 2 líneas en la terminal, ya que es impráctico, puesto que la PC irá a bordo del UAV y en un campo de vuelo no necesariamente se dispone de teclado, ratón ni monitor. Para esto, las dos líneas de código deben guardar en un archivo batch (archivo de texto con extensión .bat en el nombre) en la carpeta de inicio, que es donde se guardan todos los

programas que se quieren ejecutar automáticamente cada vez que se enciende la computadora. En Windows dicha carpeta se accede con la siguiente dirección: C:\Users\nombre_de_usuario\ AppData\Roaming\ Microsoft\Windows\StartMenu\Programs\Startup [3].

Es vital que previo a la ejecución del archivo las direcciones IP estén debidamente configuradas según lo establecido; en este caso, según el cuadro 1, porque de lo contrario el enrutador no sabrá dónde enviar las señales correspondientes y simplemente no lo hará. Esto implicará que la mini PC deba reiniciarse cuando las direcciones IP sean correctas, pero este cambio puede no ser nada práctico o fácil de hacer en un campo de vuelo.

Programas de vuelo

La biblioteca dronekit de python permite interactuar con el Pixhawk para controlar UAVs. Sus funciones permiten tareas como establecer conexión con el firmware cargado en el autopiloto, enviar instrucciones y/o solicitar información de vuelo. Se recomienda consultar la página de dronekit [4] para comprender todas las funcionalidades disponibles; acá se comentarán algunas con el fin de que el lector comprenda cómo usarlas en la estructura propuesta en este escrito.

Una función esencial es *connect* que toma como parámetro el puerto en el que está el autopiloto (en este proyecto se usó 127.0.0.1:14551 como se ve en la figura 5) y crea un objeto con sus características y funciones para enviarle instrucciones, tales como *simple_takeoff y simple_goto*. Estas últimas son fundamentales ya que permiten darle al UAV una elevación inicial y enviarlo a una posición gps determinada, respectivamente, Sin embargo, usar únicamente estas dos hace impráctica la creación de programas de vuelo, ya que si por ejemplo, en una línea de programa se da *simple_takeoff* y en la siguiente un *simple_goto*, esta última se sobreescribe a la primera, por lo que no terminará de elevarse, lo mismo que si se dan dos *simple_goto* seguidos.

Por otro lado, se puede solicitar información de vuelo como la posición del UAV, altura, velocidad, entre otras, mientras que hay sistemas de seguridad como el atributo *armed* del hardware y los modos de vuelo que se pueden estudiar en [4]. Estos últimos son de especial importancia ya que permiten que el vehículo opere con condiciones de vuelo particulares; por ejemplo, el modo *stabilize* hace que el UAV se mantenga horizontal, el modo *alt hold* permite hacer desplazamientos en únicamente en un plano horizontal y el modo guiado *guided* hace que el vehículo reciba y ejecute órdenes de una PC e ignore al mando remoto.

Teniendo en cuenta estas funcionalidades se propone crear una biblioteca adicional (la manera de crear una biblioteca de Python se indica en [5]) en la que se tengan funciones que después de dar la instrucción *simple_takeoff* o *simple_goto* soliciten la posición del UAV periódicamente por medio de un ciclo, cuya condición de salida sea haber alcanzado la posición deseada, de esta manera no se sobreescribirían las instrucciones. Así mismo en [4] se proponen más funciones que pueden implementarse. Con dicha biblioteca se pueden escribir programas de python que contengan rutinas de vuelo, pudiendo usar incluso algún enfoque de inteligencia artificial, dependiendo de la misión que se desee llevar a cabo.

Por otro lado, el usuario debe tomar en cuenta la verificación continua del modo de vuelo en que está el vehículo -que debe ser *guided* para recibir instrucciones de la mini PC-, lo cual se puede lograr combinando las funciones *add_attribute_listener* y *mode_callback* en [4].

Lo anterior es necesario porque durante un vuelo puede darse alguna circunstancia que desestabilice el UAV mientras está recibiendo instrucciones de la PC a bordo; ante esto, lo más lógico es tomar el control del vehículo con el mando remoto (basta con accionar un botón en el mando que cambie de modo *guided* a cualquier otro) para estabilizarlo. Una vez que esto sucede, se puede volver a establecer el modo *guided* para que la PC siga dando instrucciones, pero si a esta no se le notificó el cambio de modo, pudo estar enviando instrucciones cuando

el Pixhawk realmente no las iba a ejecutar, con la posibilidad de haber quedado en un ciclo esperando a que se cumpliera alguna condición, dejando la misión incompleta.

Si no se gestiona adecuadamente la notificación del cambio de modo (tanto cuando el modo deja de ser *guided* como cuando vuelve a serlo), la funcionalidad de la biblioteca es incompleta y al tomar el control de la nave, habría que llevarla hasta tierra para reiniciar el programa en modo *guided* y esperar a que no haya otra complicación durante el vuelo.

Por último, es posible que se necesite ejecutar un programa varias veces y en cada uno se requiera cambiar alguna variable, como la altura de vuelo, velocidad, etc. Es impráctico tener que acceder directamente al archivo donde se encuentra el programa y cambiar estas variables previo a la ejecución de la rutina de vuelo por lo que se suele aprovechar la herramienta *argparse* de Python que permite indicar el valor de estas variables al ejecutar el programa desde el terminal, lo cual es muy conveniente como se verá a continuación.

Control de la CCT sobre la mini PC

Hasta el momento se sabe que la información que entra y sale del Pixhawk tiene dos posibles caminos según la figura 5; hacia la mini PC, a través de su puerto interno 127.0.0.1: 14551 usando programas de vuelo basados en la biblioteca dronekit de Python y el otro camino es hacia la CCT por medio de puerto UDP 192.168.1.155:14552, cuya forma de usarse se discutirá en esta sección.

Antes de hacer un vuelo real, es altamente recomendado simular el respectivo programa de vuelo. Para ello, se abre Mission Planner en la misma PC que se tiene Python y el programa a usar, se selecciona la opción *simulation* y se elige el tipo de vehículo a usar, lo cual llevará a una pantalla como la de la figura 2; el simulador crea un vehículo virtual que intercambia información con Mission Planner a través del puerto TCP 5760. Al ejecutar el programa de Python se hace *connect* con el puerto tcp:127.0.0.1:5760 y en Mission Planner se verá todo lo que el UAV haría.

Cuando se ha depurado por completo el programa se coloca en la mini PC para ser llevado a un vuelo real. Para visualizar el estado de vuelo en la CCT se debe abrir el Mission Planner y en vez de abrir el simulador y conectarse al puerto TCP 5760, simplemente debe cambiarse la opción de TCP a UDP, conectar y seleccionar el puerto 14552. Para completar la descripción del flujo de información en la estructura, a continuación se expondrá cómo acceder desde la CCT a la mini PC del UAV a ejecutar los programas de vuelo.

Se habilitó un servidor SSH abierto en la mini PC que inicia automáticamente al encender la computadora, mientras que en una CCT se habilita un cliente SSH. Desde este último se entra al terminal de la PC del UAV como se observa en la figura 6. Se accede a la carpeta donde está el programa de interés -que en este caso se llama prueba_conexion.py- y se ejecuta, sin olvidar indicar las variables empleadas con la herramienta *argparse* antes mencionada (en este caso la variable *connect*, que guarda el puerto en el que está el Pixhawk).

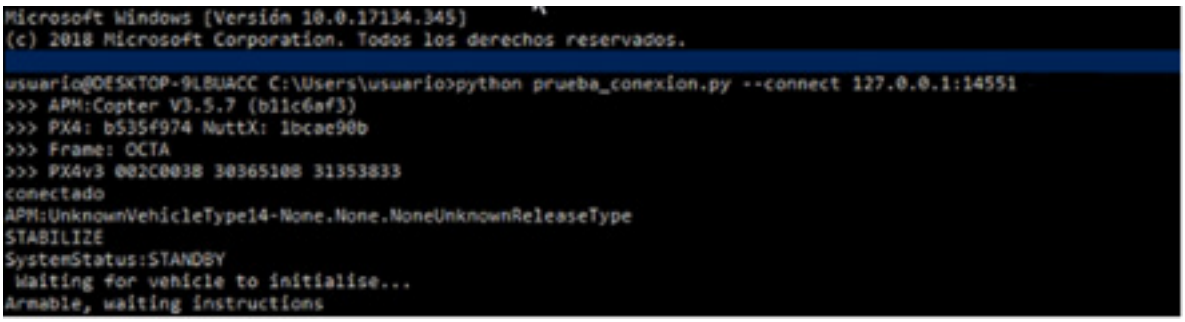

Figura 6 Acceso al terminal de la mini PC desde la CCT.

Montaje Final

Para alimentar eléctricamente la mini PC y puente Ethernet a bordo del UAV se recomienda usar baterías y en caso de que la tensión no sea suficiente se pueden usar convertidores DC-DC; el Pixhawk por su parte obtiene su energía de la mini PC al conectarse USB. Por otro lado, se debe crear una estructura mecánica para sujetar todos los componentes al UAV.

En la figura 7 se muestra el UAV sobre el que se implementó la estructura; en la parte inferior se muestra la mini PC en un soporte mecánico naranja hecho con una impresión 3D, el cual también contiene el puente Ethernet (no se observa en la figura 7). A los costados están las baterías (en color azul), mientras que el Pixhawk está dentro de la carcasa negra en la parte superior de la nave.

Figura 7 Montaje de la estructura del UAV autónomo.

Resultados

Para validar la estructura implementada se creó y programó una aplicación que simula un UAV repartidor de paquetes al cual le dan un conjunto de puntos donde debe entregar los mismos, pero sólo puede llevar un máximo de 4 a la vez, así que por medio de un algoritmo genético genera aleatoriamente posibles rutas y las muta, buscando hacer la más corta; una vez escogida, la ejecuta de manera autónoma. Al llegar a cada punto, desciende 5 m, como si estuviera entregando un paquete y vuelve a elevarse para continuar con el vuelo hasta ir a todos los puntos de la ruta.

La estructura se llevó a un vuelo de prueba real con la aplicación programada. Tanto la elevación como el descenso se hicieron manualmente con el mando remoto por seguridad, sin embargo, mientras estuvo en el aire, se mantuvo en modo guiado obedeciendo a la mini PC, haciendo el recorrido de la figura 8.

En la imagen de la izquierda obtenida con Mission Planner se observan numerados los 6 puntos que el UAV podía escoger (el H es la posición de despegue y aterrizaje), así como la ruta elegida y ejecutada (6,3,2,5). Por su parte, en la imagen en la derecha, obtenida con Google Earth, se aprecia la altura del trayecto y los descensos en cada punto de entrega, así como que las trayectorias no son completamente rectas, debido a que a la altura a la que se voló (50 m sobre el nivel del suelo) los efectos del viento eran inevitables.

Figura 8. Resultados de la prueba de vuelo.

Se puede apreciar que la ruta elegida no fue la más corta, ya que para lograrlo, el orden correcto debió ser 6,3,5,2 y aunque se evidenció que la aplicación debe mejorarse, el UAV eligió y ejecutó una ruta autónomamente -a excepción de la elevación y descenso por tratarse de la primer prueba y querer reducir los riesgos de que el UAV se estrellara-, permitiendo monitorear su trayectoria y estado de vuelo en cada instante, lo cual corresponde al propósito del proyecto.

Conclusiones y recomendaciones

Se presentó una propuesta de estructura flexible de hardware y software que le permite a un UAV ejecutar rutinas de vuelo, ya sea con comportamiento autónomo, totalmente preprogramado, así como manejado manualmente con un control remoto o una combinación de las anteriores.

Cuando se tiene un dispositivo en el aire que al caer accidentalmente podría ocasionar un daño a una persona, es muy importante estar monitoreando todo lo que está pasando en el vuelo para tomar las acciones necesarias en caso de una eventualidad. Para esto se recomienda el uso de herramientas como Mission Planner y programar que la rutina esté imprimiendo avisos del progreso de la misión que se está llevando a cabo, lo cual, según este proyecto, se vería en el terminal de la mini PC, a través de SSH.

Se recomienda complementar esta estructura con sensores y actuadores que permitan incrementar las funcionalidades de un UAV, así como intentar elevar y descender el vehículo por software y no manualmente, ya que esto aumentaría su autonomía.

Referencias

[1] Ardupilot, «Ardupilot,» 2016. [En línea]. Available: http://ardupilot.org/about. [Último acceso: 30 Septiembre 2018].

[2] M. Oborne, «Mission Planner (1.3.59) [Software],» 2018. [En línea]. Available: http://ardupilot.org/planner/docs/mission-planner-installation.html.

[3] Microsoft, «Microsoft,» 8 noviembre 2015. [En línea]. Available: https://answers.microsoft.com/es-es/windows/forum/windows_10-start/iniciar-un-programa-al-encenderse-windows-10/1704b332-b34d-472d-838f-a621ce-4a58b7. [Último acceso: 1 octubre 2018].

[4] 3D Robotics, «Dronekit,» 2016. [En línea]. Available: https://dronekit.netlify.com/. [Último acceso: 20 mayo 2019].

[5] WEBO, «01 Crear un módulo para Python,» 6 Mayo 2016. [En línea]. Available: https://www.youtube.com/watch?v=dxCPuyfAvD0. [Último acceso: 19 Mayo 2019].

[6] Geekland, «Blog de Tecnología,» 3 junio 2018. [En línea]. Available: https://geekland.eu/instalar-cliente-servidor-ssh-en-windows/. [Último acceso: 2 noviembre 2018].