



Universidad Nacional Mayor de San Marcos
Universidad del Perú. Decana de América

Dirección General de Estudios de Posgrado

Facultad de Ingeniería Electrónica y Eléctrica
Unidad de Posgrado

**Aguilucho, diseño y desarrollo de un sistema de transmisión de alto
volumen de datos entre un punto fijo y un punto móvil con
restricciones de energía, espacio y peso**

TESIS

Para optar el Grado Académico de Magíster en Telecomunicaciones

AUTOR

Luis CAMACHO CABALLERO

ASESOR

Carlos Alberto SOTELO LÓPEZ

Lima, Perú

2020



Reconocimiento - No Comercial - Compartir Igual - Sin restricciones adicionales

<https://creativecommons.org/licenses/by-nc-sa/4.0/>

Usted puede distribuir, remezclar, retocar, y crear a partir del documento original de modo no comercial, siempre y cuando se dé crédito al autor del documento y se licencien las nuevas creaciones bajo las mismas condiciones. No se permite aplicar términos legales o medidas tecnológicas que restrinjan legalmente a otros a hacer cualquier cosa que permita esta licencia.

Referencia bibliográfica

Camacho, L. (2020). *Aguilucho, diseño y desarrollo de un sistema de transmisión de alto volumen de datos entre un punto fijo y un punto móvil con restricciones de energía, espacio y peso*. Tesis para optar el grado de Magíster en Telecomunicaciones. Unidad de Posgrado, Facultad de Ingeniería Electrónica y Eléctrica, Universidad Nacional Mayor de San Marcos, Lima, Perú.

HOJA DE METADATOS COMPLEMENTARIOS

Código ORCID del autor	0000-0001-6569-550X
DNI o pasaporte del autor	25762600
Código ORCID del asesor	0000-0001-8960-6567
DNI o pasaporte del asesor	07017259
Grupo de investigación	Grupo de Telecomunicaciones Rurales Pontificia Universidad Católica del Perú
Agencia financiadora	Perú Ministerio de la Producción Innovate Perú 123-PNICP-PIAP-2015
Ubicación geográfica donde se desarrolló la investigación	Lima, Perú -12.04318, -77.02824
Disciplinas OCDE	Ingeniería, Tecnología https://purl.org/pe-repo/ocde/ford#2.02.01 Telecomunicaciones https://purl.org/pe-repo/ocde/ford#2.02.05



UNIVERSIDAD NACIONAL MAYOR DE SAN MARCOS
(Universidad del Perú, DECANA DE AMÉRICA)
FACULTAD DE INGENIERÍA ELECTRÓNICA Y ELÉCTRICA
UNIDAD DE POSGRADO
Calle Germán Amezaga N.º 375 Lima (Perú)
Teléfono (51 – 1) 6197000 Anexo 4204
Correo: postfice@unmsm.edu.pe



«AÑO DE LA UNIVERSALIZACIÓN DE LA SALUD»

**ACTA DE SUSTENTACIÓN DE TESIS PARA OPTAR EL GRADO ACADÉMICO
DE MAGÍSTER EN TELECOMUNICACIONES**

Siendo las 15:30 horas. del 24 de octubre de 2020, los suscritos miembros del jurado reunidos en el salón de Grados de la Facultad de Ingeniería Electrónica y Eléctrica, el Jurado Examinador presidido por el Dr. Carlos Bernardino Silva Cardenas, Mg. Daniel Diaz Ataucuri, Mg. Wilbert Chávez Irazabal, Mg. Horacio Alfonso Chacón Cursack y el Mg. Carlos Alberto Sotelo López.

Se reunió para la sustentación oral y pública de la Tesis para optar el Grado Académico de Magíster en Telecomunicaciones, que solicitó el alumno **Luis Camacho Caballero** con código N° 03197020, el cual procedió hacer la exposición oral y pública de su Tesis Titulada **"AGUILUCHO, DISEÑO Y DESARROLLO DE UN SISTEMA DE TRANSMISIÓN DE ALTO VOLUMEN DE DATOS ENTRE UN PUNTO FIJO Y UN PUNTO MÓVIL CON RESTRICCIONES DE ENERGÍA, ESPACIO Y PESO"**

Concluida la exposición, el Jurado Examinador procedió a formular las preguntas reglamentarias y, luego de una deliberación en privado, decidió otorgarle la siguiente calificación:

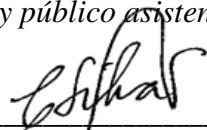
APROBADO

14
NÚMERO

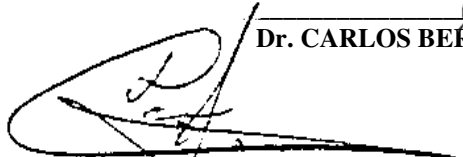
CATORCE
LETRAS

A continuación, el Presidente Jurado recomienda que la Unidad de Posgrado proceda con el trámite correspondiente para que se otorgue el Grado Académico de Magíster en Telecomunicaciones al alumno **Luis Camacho Caballero**.

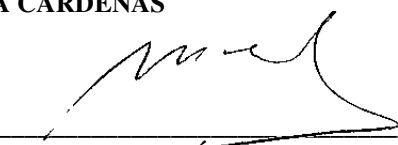
Siendo las...17:50 horas.... se levantó la Sesión, recibiendo el graduado las felicitaciones de los señores miembros del Jurado y público asistente.



Dr. CARLOS BERNARDINO SILVA CARDENAS
Presidente



Mg. DANIEL DIAZ ATAUCURI
Miembro



Mg. WILBERT CHÁVEZ IRAZABAL
Miembro



Mg. HORACIO ALFONSO CHACÓN CURSACK
Miembro



Mg. CARLOS ALBERTO SOTELO LÓPEZ
Asesor

Tabla de contenido

1. Introducción	6
1.1. Situación Problemática	6
1.2. TapirNet	7
1.3. Formulación del problema	8
1.4. Justificación teórica	9
1.5. Justificación práctica	9
1.6. Objetivos.....	10
1.6.1. Objetivo general	11
1.6.2. Objetivos específicos	11
1.6.3. Hipótesis	11
2. Marco Teórico	12
2.1. Marco epistemológico	12
2.2. Antecedentes de investigación	13
2.3. Bases teóricas	14
3. Metodología	15
3.1. Hardware	17
3.1.1. AeroCore	17
3.1.2. COM DuoVero Zephyr	20
3.1.3. Pixhawk	21
3.1.4. ArduPilot Mega (APM)	23
3.1.5. Periféricos	23
3.2. Software	23
3.2.1. Grabación de firmware AeroCore y Pixhawk	24
3.2.2. Comunicación Serial	27
3.2.3. MAVLink	28
3.2.4. NuttX sobre PX4	30
3.2.5. Control en Tierra QGroundControl	31
3.2.6. Configurar DuoVero como Access Point	32
3.3. Diseño mecánico de la nave	32
3.3.1. Efectos Aerodinámicos	33
3.3.2. Parámetros de un quadróptero	34
3.3.3. Elección de Motor y Hélice	35
3.3.4. Consumo de Energía vs. Potencia de los 4 Rotores	36
3.3.5. Diseño de Fuselaje	38
3.3.6. Análisis del Flujo de Aire	40
3.3.7. Lista de revisión antes de cada vuelo	42
4. Resultados	43
4.1 Pruebas de hipótesis	44
4.1.1. AguiLucho	44
4.1.2. DroneKit	45
4.1.3. Script Simple Go To modificado	46
4.2. Presentación de resultados	47
4.3. Análisis, interpretación y discusión de resultados	48

4.3.1. Comunicación aire tierra.....	50
4.3.2. Fuselaje	52
5. Impactos	54
5.1. Costos de implementación de la propuesta	54
5.2. Cronograma de Actividades	55
5.3. Beneficios que aporta la propuesta	56
Conclusiones	59
Recomendaciones	60
Referencias bibliográficas	61
ANEXO 1 Script blocking_goto	64
ANEXO 2 Script images_transfer	67

Tabla de Cuadros

Tabla 1 Archivos para crear disco de arranque.....	30
Tabla 2 Pruebas de empuje de juego de motores con hélices 1855.....	36
Tabla 3 Pruebas de empuje de juego de motores con hélices 1655.....	37
Tabla 4 Componentes y parámetros de UAV, incluyendo tiempo de vuelo estimado mediante eCalc.	55
Tabla 5 Presupuesto	56
Tabla 6 Cronograma	57

Tabla de Figuras

Figura 1 Ubicación geográfica de TapirNet	8
Figura 2 Topología original de TapirNet	8
Figura 3 Arquitectura de un piloto automático	18
Figura 4 Diagrama de bloques de AeroCore	21
Figura 5 Activación de BOOT0	24
Figura 6 Piloto automático PixHawk	32
Figura 7 Entorno gráfico de un software de control en tierra	33
Figura 8 Disimetría de la sustentación	38
Figura 9 Actitudes de Vuelo	39
Figura 10 Motor TAROT 5008	40
Figura 11 Hélice 1655	41
Figura 12 Bateria LiPo de 22.2V – 5000mAh	43
Figura 13 Pruebas de consumo de energía del rotor	44
Figura 14 Empuje de rotor a 1kg y consumo de corriente de 4.13 amperios a 25 voltios	44
Figura 15 Diseño de quadróptero en 3D, software Solidworks, vista lateral	45
Figura 16 Diseño de quadróptero en 3D, software Solidworks, vista Isométrica	45
Figura 17 Diseño de quadróptero en 3D, software Solidworks, vista superior	46
Figura 18 Velocidades del flujo de aire cercano a la superficie del quadróptero	47
Figura 19 Diferencia de presión entre partes frontal y posterior	47
Figura 20 Turbulencia al rededor del quadróptero	48
Figura 21 Diagrama de flujo del programa AguiLucho	52
Figura 22 Intercambio de mensajes durante transferencia de paquetes	55
Figura 23 Topología final de la red de acceso	56
Figura 24 Diagrama de conexiones de los componentes del módulo audio	57
Figura 25 Variación del <i>throughput</i>	58
Figura 26 Variación de RSS de acuerdo a la separación entre nodos	59
Figura 27 Variación del <i>throughput</i> con antenas de polarización circular	59
Figura 28 Variación de RSS con antenas de polarización circular	60

1. Introducción

La presente tesis consta de cinco capítulos, conclusiones, recomendaciones, bibliografía y anexos. El primer capítulo es introductorio y en él se describen la formulación del problema, la propuesta de solución y objetivos de la investigación; también se presenta el proyecto TapirNet, en cuyo marco se realizó esta investigación. En el segundo capítulo se presentan los antecedentes y las bases teóricas. En el tercer capítulo se describen todas las herramientas telemáticas hardware y software, así como el diseño mecánico de la nave. En el capítulo cuatro se muestran y discuten los resultados de dos conjuntos de pruebas realizadas, en el primer conjunto de pruebas no se consiguen los resultados y en el segundo sí, después de superar el obstáculo hallado en la primera experiencia. En el quinto capítulo se muestra el costo y cronograma de esta investigación. Finalmente, en los anexos se colocó el software escrito en esta investigación.

El aporte de esta investigación es una solución de telecomunicaciones desarrollada, eficiente y de bajo costo, que permita hacer viable la recolección de información de dispositivos y sensores colocados en áreas naturales protegidas remotas y localizadas fuera del alcance de las redes de telecomunicaciones tradicionales, solución que NO EXISTIA o que no había sido documentada. En esta solución, la red de transporte se reemplaza por un *router*, el cual se desplaza físicamente hasta la vecindad de una red de acceso, establece una comunicación inalámbrica con un nodo de dicha red, exporta la información y regresa a su punto de partida.

Tal contribución es reconocida por el Ministerio de Relaciones Exteriores, el cual cita a esta innovación en su informe “El pequeño libro de la ciencia, la tecnología y la innovación en el Perú”¹.

Este aporte es públicamente accesible en la reconocida plataforma de desarrollo colaborativo GitHub² desde donde el software generado puede ser gratuitamente descargado y mejor aún, puede ser mejorado y mantenido por una comunidad de desarrolladores de software.

Esta idea original y novedosa ha incrementado el conocimiento de investigadores arraigados en Perú sobre:

1. Sistemas Telemáticos Embebidos de Tiempo Real
2. Vehículos no tripulados (UAV) en todos los aspectos: mecánico, eléctrico, electrónico y telemático.

1.1. Situación Problemática

Las plantas, los animales, las personas y el medio ambiente son toda una comunidad biológica, un ecosistema, donde cada parte depende del otro para sobrevivir. La fauna silvestre no es sólo parte de estos ecosistemas, sino que es un componente estructural de ellos. La biodiversidad proporciona la base para los ecosistemas y los servicios que ellos proporcionan sustentan el bienestar humano. Cuando una parte de la comunidad biológica es desequilibrada o suprimida, todo el sistema sufre. Si las afirmaciones anteriores no fueran suficientes, hay razones económicas para la conservación de los ecosistemas: una evaluación de los servicios de los ecosistemas mundiales calculó su valor en 125000 millones de dólares al año, cifra mucho más alta que el PBI de cualquier país del mundo e incluso más que el PBI mundial, 72600 millones de dólares [1].

La Amazonía occidental es la parte biológicamente más rica de la cuenca del Amazonas, a diferencia del Amazonas oriental (brasileño), sigue siendo un ecosistema en gran parte intacto, sin embargo, la biodiversidad y los servicios eco sistémicos están disminuyendo a un ritmo sin precedentes; la biodiversidad amazónica está

¹ https://cdn.www.gob.pe/uploads/document/file/10707/libro_de_Ciencia_es.pdf

² <https://github.com/luisacamachocaballero/Tapirchip/tree/master/AguiLucho>

sufriendo una demencial extracción de los recursos naturales. Es necesario demostrar que la preservación no sólo es rentable desde el punto de vista medioambiental sino también económicamente. Si el bosque se pone en valor, su destrucción podría ser detenida.

Los inventarios de vida silvestre ayudan a aumentar el valor del bosque mostrando la biodiversidad de la selva tropical y advirtiendo sobre su preservación, especialmente a los tomadores de decisiones. A pesar de su importancia, se han realizado muy pocos inventarios de vida silvestre en Loreto, la región amazónica del Perú, y peor aún, ellos cubrieron poco territorio. El inventario tradicional demanda que un observador cuente *in situ* los animales que observa; en el presente siglo, el surgimiento de sensores de presencia que pueden activar automáticamente una cámara fotográfica digital instauró un nuevo paradigma sobre la forma de hacer los inventarios. Los zoólogos ya han utilizado esta tecnología para inventarios, pero con resultados limitados debido al compromiso entre calidad de imágenes, capacidad de transmisión y la disponibilidad de energía eléctrica. Por un extremo, en [2] consiguieron excelentes imágenes gracias al uso de cámaras trampa (comercializadas para la caza deportiva) pero con una autonomía energética bajísima y casi sin capacidad de transmisión de las imágenes. En el otro extremo, en [3] correctamente apuntaron al diseño y fabricación de cámaras trampa *ad hoc* y consiguieron que sus dispositivos consumieran muy poca energía, pero el costo fue la limitación de la calidad de las imágenes y de la capacidad de transmisión. En ambos casos, los investigadores solo llegaron hasta la implementación de una red de acceso mas no de transporte, la exportación de las imágenes solamente era manual. Esas experiencias cumplieron parcialmente los requisitos de pertinencia y sostenibilidad, aún era necesario desarrollar un sistema alternativo para el inventario y el monitoreo de la vida silvestre en selvas tropicales.

Tomando en cuenta las limitaciones de las experiencias previas, tenía sentido el desarrollo de una nueva solución de sistema de inventario y/o monitorización de fauna silvestre haciendo uso de cámaras fotográficas CMOS, de redes inalámbricas de sensores (WSN) y sistemas aéreos no tripulados (UAS).

A esta solución se le denomina TapirNet, la cual se describe a continuación.

1.2. **TapirNet**

TapirNet es un sistema que aborda completamente el problema de realizar inventarios de fauna silvestre: desde capturar las imágenes hasta poner dichas imágenes a disposición de los biólogos o zoólogos responsables [45].

El sistema puede dividirse en tres partes:

1. Los nodos sensores, los cuales capturan la data en un radio determinado
2. La red de acceso, que transporta la data desde los nodos sensores hasta uno o varios nodos concentradores o *gateways*
3. La nave aérea no tripulada, que actúa como red de transporte *store & forward*, que exporta la data desde los *gateways* hacia Internet

Las dos primeras partes de TapirNet fueron probadas mediante un proyecto piloto en la reserva Allpahuayo Mishana, cerca de la ciudad de Iquitos [4]. En dicho lugar se instalaron un conjunto de microneles de

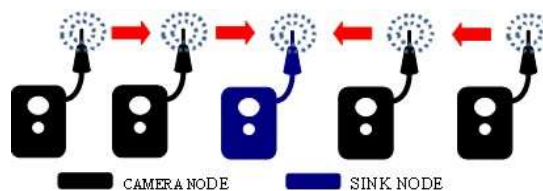
comunicación inalámbrica multisalto, cada una de ellas compuesta por cuatro nodos simples y un nodo sumidero. Los nodos simples están compuestos por la tarjeta Tapirduino (un clon de la SBC Arduino), una radio Digi XBee Pro 900 HP, una cámara CMOS de 0.3Mp, un reloj (RTC), flash LED infrarrojo y un sensor PIR. Tapirduino fue desarrollado para conseguir un bajo consumo de energía, un nodo simple consume 95,96 mA-h/día y un nodo sumidero consume 129,71 mA-h/día. Gracias al uso de la banda de frecuencia de 900 MHz en lugar de 2,4 GHz, se alcanzaron distancias que van desde 120 hasta 170 metros entre nodos vecinos. Los nodos sumidero son iguales a los nodos simples con la única diferencia de poseer una segunda radio, una Digi XBee WiFi que soporta el protocolo 802.11 y que es la radio que se usa para comunicar con la UAV y transferir las imágenes hacia el exterior. En las imágenes 1 y 2 se muestra la ubicación geográfica de las micro redes y su topología.

TapirNet se llevó a cabo entre noviembre 2013 y junio 2017 bajo el auspicio de Innovate Perú³. El 15 de noviembre de 2017 se hizo una presentación pública en la cual se planteó un modelo de sostenibilidad, escalabilidad y transferencia tecnológica⁴. Como se señaló en dicha presentación, existen 20 millones de kilómetros cuadrados de bosque tropical que deben ser mapeados, lo cual constituye una gran oportunidad. Al día de hoy, noviembre 2020, la invitación sigue abierta a empresas con y sin fines de lucro para invertir en esta iniciativa y cubrir esa demanda.

En su última versión [44], TapirNet permite exportar información del bosque amazónico para lo cual requiere de $n \times m$ nodos sumidero más $(n + 1) (m + 1)$ nodos simples por cada $n \times m$ kilómetros cuadrados, sin alterar el medio ambiente y a una tasa costo eficiente económicamente hablando.



Figura 1 Ubicación geográfica de TapirNet; Nota. Figura extraída de Google Maps, 2020, <https://www.google.com/maps>



³ <https://www.innovateperu.gob.pe/>

⁴ https://educast.pucp.edu.pe/video/9289/presentacion_de_resultados_tecnologia_que_protege_la_amazonia

1.3. *Formulación del problema*

Desarrollar la tercera parte del sistema TapirNet es el problema cuya solución se acomete en la presente tesis. Para transportar hacia el exterior o exportar la información colectada por dispositivos sensores ubicados en zonas de reserva natural, en el país y el mundo se afrontan muchos problemas logísticos y geográficos [5], entre ellos la inexistencia o cobertura insuficiente de redes de telecomunicaciones terrestres fijas o móviles y la inexistencia de energía convencional en dichas reservas naturales [6]. En estas condiciones una aparente alternativa es la instalación de comunicaciones vía satélite, pero la realidad es que dicha alternativa tiene muchos inconvenientes OPEX y CAPEX: elevado costo por el uso del segmento satelital, elevada inversión en equipamiento VSAT y fuente de energía no convencional, elevado costo de mantenimiento, entre otros; estos problemas representan barreras a la viabilidad de una solución de este tipo. Tampoco es adecuado plantear sistemas de microondas basadas en enlaces punto a punto pues ellos requerirían torres muy altas por encima de la copa de los árboles para asegurar conectividad y también la realización de pesadas obras civiles para soportar dicha infraestructura física.

1.4. *Justificación teórica*

Frente al problema de exportar data de zonas remotas, dado que las soluciones tradicionales son inaccesibles o no sostenibles, se ha planteado un método no tradicional y no invasivo: el uso de un *router* inalámbrico colocado dentro de una nave aérea no tripulada (UAV) como medio de transporte de la información. Para que el método sea efectivo deben resolverse tres desafíos principales:

1. Transmitir un alto volumen de datos entre un punto fijo y un punto móvil [7]
2. Dado que la distancia es una restricción al control remoto de las UAV, ceder el control del piloto automático de la nave a un dispositivo dentro de la misma nave, una computadora empotrada de bajo consumo de energía, bajo peso y baja capacidad de procesamiento [8].
3. Navegación autónoma de media hora de vuelo sin recarga y hasta diez kilómetros de recorrido a 8 m/s.

Desde el modelado de características glaciales y el monitoreo de la erosión hasta el conteo de animales y la identificación de especies, la lista de proyectos para los que se utilizan los drones es larga y continúa creciendo. La investigación sobre vehículos aéreos no tripulados a pequeña escala se remonta a la década de 1990 [9]. Si bien la principal fuerza impulsora detrás del desarrollo de los UAV ha sido la demanda militar, recientemente los UAV a pequeña escala se han tornado disponibles para el mercado civil. Las aplicaciones típicas incluyen fotografía aérea o vigilancia, transmisión de video en vivo o la transmisión de imágenes fijas. Sin embargo, hasta hoy no hay estudios previos ni productos comerciales cuyo objetivo haya sido un vuelo autónomo de larga duración a un precio razonable, es por esta razón que además del desarrollo de aplicaciones de software, en la presente tesis también se incluyó diseño mecánico.

1.5. *Justificación práctica*

La forma práctica de alcanzar los requisitos planteados en la justificación teórica es hallar un punto de equilibrio entre peso, consumo energético y potencia computacional dentro de la UAV. Dada esta reformulación, la

acumulación de funciones dentro de un único hardware mediante la programación de dichas funciones se convierte en la meta principal, tema central de la presente tesis.

Sin embargo, las aplicaciones para controlar el vehículo no pueden ejecutarse directamente en Pixhawk sino en una computadora complementaria. La implementación más sencilla utiliza una computadora portátil en tierra, conectada al vehículo por radio, sin embargo, la simplicidad tiene un costo: la conexión de radio es de bajo ancho de banda y propensa al ruido. La conexión de baja velocidad, 57600 baudios, hace que esta opción no sea adecuada para aplicaciones que requieren intensa comunicación con mensajes de control de misión crítica libres de ruido. La alternativa más adecuada para la tarea es incorporar una computadora de propósito específico dentro de la nave [16], pues con ello se aumenta el ancho de banda del canal de comunicación serial a 1.5M baudios y también se elimina el ruido. Y, por supuesto, el MAV es totalmente autónomo durante el vuelo, una característica deseable en aplicaciones en las que la comunicación puede ser difícil. En octubre 2015, 3DR presentó DroneKit-Python, una API de Python para desarrollar aplicaciones MAV. DroneKit libera a los investigadores y desarrolladores de muchos de los aspectos de bajo nivel de MAVLink, proporcionando una interfaz de alto nivel para conectarse, monitorear y controlar un vehículo. La configuración del vehículo y la calibración de los sensores se realizan fácilmente mediante un software de estación de control en tierra (GCS), Mission Planner (solo para Windows) y APM Planner (multiplataforma) están disponibles gratuitamente.

La aerodinámica de una nave aérea con rotores es uno de los problemas más desafiantes para el diseño mecánico. Un entendimiento completo de las cargas en los rotores y la performance, la vibración y la acústica del fuselaje, son importantes durante el estudio aerodinámico de un multirrotor. Existen tres métodos para realizar este estudio:

1. conducir experimentos reales
2. análisis teórico
3. simulaciones por computadora.

Los resultados de vuelos experimentales son muy confiables, pero son difíciles y costosos porque es complicado medir o visualizar el flujo alrededor de las hélices. El análisis teórico tiene sus limitaciones, porque las ecuaciones que gobiernan el comportamiento del flujo pueden ser complejas y probablemente se pueden resolver solo algunos casos sencillos. La manera más viable, por el momento, es la simulación utilizando computadoras. La simulación por computadora es la manera más realista de predecir el comportamiento del aire alrededor de las hélices del motor, consiguiendo una predicción y entendimiento más preciso de la aerodinámica de todo el vehículo.

1.6. Objetivos

Los objetivos responden al problema general y a los problemas específicos los cuales son respectivamente:

¿Cómo diseñar e implementar una nave no tripulada que actúe como una red de transporte store & forward?

¿Cómo lograr suficiente autonomía de vuelo?

¿Cómo conseguir un adecuado hand on/off de la comunicación?

¿Cómo conseguir un sistema eléctrico mecánico eficiente?

¿Cómo alcanzar el consumo de energía adecuado?

1.6.1. Objetivo general

El objetivo de la presente tesis es el diseño e implementación de la red de transporte móvil del sistema de monitoreo de fauna silvestre arriba mencionado. Esta red de transporte es un sistema de transmisión de alto volumen de datos entre un punto fijo y un punto móvil ubicado en una nave aérea no tripulada.

1.6.2. Objetivos específicos

Los objetivos específicos son los siguientes:

1. Lograr suficiente autonomía de vuelo de la nave, recorrido de hasta 10 km de distancia a una velocidad de 8 m/s
2. Conseguir que la comunicación (enganche, transferencia y desenganche) aire-tierra se realice en vuelo a una velocidad no menor a 10 Mbps
3. Lograr diseños del fuselaje, del sistema de propulsión y del control manual de respaldo, que permitan obtener una capacidad de carga útil de por lo menos medio kilogramo.
4. Lograr un consumo mínimo de energía sobre la base de un diseño óptimo que permita que solamente 10 Ah a 22 V permita una autonomía de media hora

1.6.3. Hipótesis

Es posible introducir dentro de una nave aérea no tripulada una computadora en la cual se puedan programar la automatización de tareas adicionales además de controlar la corriente suministrada a los motores a través del piloto automático. Este sistema es más eficiente que los sistemas actuales para la exportación de data aislada proveniente de sistemas de inventario y/o monitorización de fauna silvestre instalados en zonas remotas agrestes.

2. Marco Teórico

2.1. *Marco epistemológico*

El propósito de los sistemas de información es mejorar la efectividad y eficiencia de organizaciones e individuos en el trabajo diario y en el negocio. Los sistemas de información y las organizaciones consisten en estructuras, personas, tecnologías y procesos conectados entre sí. Los sistemas de información intentan resolver los problemas que ocurren cuando las organizaciones se encuentran con la tecnología de la información. El desarrollo de proyectos de investigación en las tecnologías de la información y comunicación es empírico dado que se caracteriza por la aplicación de conocimientos adquiridos en diversos campos de la ingeniería y a su vez se añaden nuevos conocimientos. El tipo es investigación tecnológica, aplicada y cuantitativa y el diseño es cuasi experimental.

Dos paradigmas de investigación caracterizan la investigación de sistemas de información: ciencia del comportamiento y *design science*. La ciencia del comportamiento guía la investigación e intenta explicar o predecir el fenómeno [36]. A menudo, la investigación en *design science* intenta construir y evaluar los artefactos desde la perspectiva de "necesidad" de una empresa. En otras palabras, el paradigma de la ciencia del comportamiento es encontrar "lo que es verdadero" y, en contraste, *design science* busca "lo que es efectivo". La investigación que se desarrolló en esta tesis está en sintonía con el enfoque de la investigación de *design science*. *Design science* fue desarrollada para la investigación de sistemas de información debido a la naturaleza de dicha disciplina. La ciencia tradicional trata de buscar, desarrollar y verificar teorías y también predecir el comportamiento humano. En la investigación de *design science*, no es la naturaleza sino son los artefactos u objetos hechos por el hombre los que se diseñan, construyen y evalúan [17]. La investigación responde a las preguntas: ¿qué cambios y modificaciones se han producido?, ¿qué mejoras se han logrado?, ¿cuál es la eficiencia del nuevo sistema?, etc. En este nivel se aplica un nuevo sistema, modelo, tratamiento, programa, método o técnicas para mejorar y corregir la situación problemática, que ha dado origen al estudio de investigación. La investigación aplicada busca el conocer para hacer, para actuar, para construir, para modificar.

Se pueden identificar dos procesos dentro de *design science*: proceso de construcción y proceso de evaluación. Los productos resultantes de la investigación son constructos, modos, métodos e instancias. La investigación debe ser replicada utilizando las mismas definiciones y configuraciones que en la implementación/construcción original. El punto de partida de la investigación puede ser una "necesidad comercial" u otro fenómeno, que debe ser investigado esto es llamado un "espacio problemático". Este espacio problemático puede ocurrir en el contexto de ciertas personas, una comunidad o una organización (entorno). Si el punto de partida de las actividades de investigación es una "necesidad comercial" en lugar de algún otro fenómeno, el proyecto de investigación suele ser más exitoso. Una base de conocimiento como una investigación previa en el área relevante, podría servir como un valioso aporte tanto para la construcción como para el proceso de evaluación. Esta base incluye los datos y las metodologías que se pueden utilizar para apoyar las actividades de investigación al proporcionar un conjunto de teorías, instrumentos, construcciones, modelos y marcos. Ejemplos de teorías, modelos y marcos se utilizan en la fase de definición de la investigación y las metodologías en la fase de evaluación del proceso [24].

2.2. Antecedentes de investigación

La mayoría de los grupos de investigación han seguido metodologías similares en el desarrollo de sus helicópteros de pequeña escala. Una lista de características de diseño comunes, figura a continuación:

1. Para la construcción de FCS (sistemas de control de vuelo), los componentes son generalmente productos COTS (*commercial off the shelf*). La compatibilidad de los puertos de entrada/salida es el principal criterio para la selección de componentes y la comunicación serial es ampliamente adoptada. La razón de tal prevalencia es que la mayoría de los investigadores buscaron una solución pronta pero fiable para el desarrollo de FCS [26].
2. El software de control a bordo es desarrollado basado generalmente en sistemas operativos de tiempo real (RTOS). Las tres opciones principales son: QNX, RTLinux, y VxWorks; existen otros sistemas basados en cualquiera de esos tres. En términos de arquitectura de software, un diseño jerárquico y modular fue adoptado

comúnmente. Las tareas clave, como adquisición de datos y ejecución de control de vuelo se programan como módulos individuales y con diferentes prioridades asignadas [27].

3. Plataformas aéreas desarrolladas para uso con RC (radio control) son ampliamente adoptadas como la línea de base con mínimas modificaciones. El control de estas naves a mucha mayor distancia es posible con sistemas de navegación automáticos pre programados, es decir un auto control. Habitualmente los enlaces RC se conservan para evitar accidentes [28].
4. Para GCS (estación de control en tierra), una computadora portátil robusta corriendo cualquiera de los sistemas operativos más populares es la herramienta típica [29].

Entre los proyectos que utilizan el sistema operativo Linux, podemos citar:

1. Gumstix AeroCore MAV - placa controladora que corre NuttX en un procesador Cortex-M4 MCU, y corre Yocto Linux en un módulo DuoVero basados en Cortex-A9. El MCU soporta controles y sensores para un micro-helicóptero, mientras que Linux gestiona la “inteligencia de misión”, incluso tareas como efectos visuales. AeroCore soporta integración con proyectos de código abierto como ROS, PX4, y proyectos compatibles con PX4 como QGroundControl y MAVLink.
2. Airware - Este startup desarrolló una plataforma de información aérea que incluye un sistema de piloto automático llamado osNano basado en Linux. La versión beta de la plataforma se está utilizando actualmente en proyectos piloto como la conservación de la vida silvestre, o la distribución de vacunas.
3. Paparazzi - diseños de código abierto Paparazzi se utilizan en varios pilotos automáticos incluyendo Bit Squared, PPZUAV, Sumo, y Transition Robotics. El hardware UAV está construido alrededor de microcontroladores y programado en C, pero el software de la estación de tierra utiliza Debian Linux.

Se está trabajando en un piloto automático derivado de Gumstix ejecutando Linux, quizás basado en AeroCore. BeagleDrone también planea integrar tecnología Paparazzi [23].

4. LinuxDrone, cuenta con un piloto automático que se ejecuta en Xenomai RTOS Linux. LinuxDrone también toma prestado código del proyecto OpenPilot para pilotos automáticos basados en microcontroladores.

2.3. Bases teóricas

En el acápite 1.3 Formulación del problema se detalló que todas las redes de transporte *live real time* son imposibles de desplegar bajo las condiciones de este proyecto. La barrera será superada con una solución creativa: la red de transporte funcionará bajo un esquema *store and forward*, es decir la información se acumulará en un nodo concentrador hasta que el medio de salida de la información esté disponible.

Como red de transporte, la solución planteada es el uso de un *router* instalado dentro de una nave aérea no tripulada (UAV) que sobrevolará la zona de estudio y que extraerá periódicamente la información acumulada en el nodo concentrador [19].

La investigación sobre los UAV de pequeña escala se remonta a la década de 1990, durante la cual una serie de proyectos se pusieron en marcha con el objetivo de obtener una autonomía básica. Tal tendencia creció y se extendió en todo el mundo [20]. Sin embargo, artículos sobre diseño y construcción de UAV de pequeña escala solamente aparecen con frecuencia en la literatura hasta el año 2005, el entusiasmo disminuyó progresivamente

debido al hecho que desarrollar una UAV se ha convertido básicamente en un trabajo técnico y por lo tanto tiene una baja apreciación en el mundo científico. Es decir, actualmente se incide mucho más en el desarrollo de aplicaciones que en el diseño mismo de plataformas de vuelo [21]. Tanto hardware como software de código abierto se emplean masivamente para el diseño de pilotos automáticos [22].

Mientras que los robots terrestres generalmente se comunican en el mismo plano con otros robots o con la estación base, las naves aéreas se mueven en el espacio tridimensional y requieren sistemas de transmisión más sofisticados. La posición, la orientación y el movimiento hacen altamente sensibles a los enlaces inalámbricos con las UAV, por ello el desarrollo de enlaces de alta capacidad de transmisión de datos es un gran desafío. Una forma intuitiva de lograr una mejor calidad de enlace es diseñar el sistema de transmisión inalámbrica analizando el perfil de los enlaces inalámbricos en mediciones de campo sobre un banco de pruebas experimental. [10]. Debido a su masificación, IEEE 802.11 es el estándar de facto que proporciona altas velocidades en redes inalámbricas a bajo costo. Una posibilidad para recopilar información sobre la calidad de transmisión inalámbrica es el uso del subsistema inalámbrico Linux [11][12].

El control de la aeronave es facilitado por un piloto automático [13]. En este rubro claramente destaca el software Pixhawk [14], desarrollado por investigadores de ETH Zurich, Pixhawk es de código abierto y está ampliamente adoptado, es adecuado para uso en helicópteros, aviones e incluso vehículos terrestres. Las opciones de conectividad son numerosas, lo que permite el uso de una amplia gama de dispositivos externos como GPS, telémetros y computadoras complementarias. Internamente, la unidad integra un acelerómetro de tres ejes, un giroscopio de tres ejes, una brújula de tres ejes y un barómetro. Estos sensores permiten determinar el movimiento, la orientación, el rumbo y la altitud relativa. El firmware para Pixhawk es construido con dos programas de código abierto, distintos pero cooperativos: APM y PX4. MAVLink [15], un protocolo de comunicación basado en mensajes subyace a ambos esfuerzos. MAVLink proporciona mensajes que permiten acceder y cambiar los valores de los parámetros del vehículo, verificar el estado y la navegación, cambiando, por ejemplo, la posición, la orientación y la velocidad. El protocolo es extensible, lo que permite a los usuarios definir nuevos mensajes para otros propósitos, en esta tesis no hubo necesidad de hacerlo.

Recientemente, los escenarios de comunicación de UAV a UAV [32] y de UAV a tierra [33] [34] se han estudiado tanto en entornos de simulación como de tiempo real. Los modelos de propagación de radio y las estructuras de las antenas se investigan como los factores clave que influyen en el diseño de la capa física de la comunicación de UAV. Las ondas electromagnéticas irradian desde el transmisor al receptor a través de canales inalámbricos. La caracterización de la propagación de ondas de radio se expresa como una función matemática, que se denomina modelado de propagación de radio. El entorno UAV presenta varios desafíos únicos en términos de propagación de radio en comparación con otros tipos de redes inalámbricas [35]. Algunos de los desafíos son los siguientes:

1. Variaciones en la distancia de comunicación.
2. Coincidencia de la línea de vista con el patrón de radiación de la antena
3. Efectos de reflexión del suelo.
4. Ruido proveniente de los motores de la nave y del equipo electrónico de a bordo.
5. El efecto del movimiento de la aeronave (*pitch*, *roll*, *yaw* etc.) en la calidad del enlace inalámbrico.
6. Condiciones ambientales.
7. *Delay* y *jitter*

La comunicación *peer-to-peer* es esencial para la coordinación colaborativa y la prevención de colisiones de sistemas UAV. El desarrollo de nuevos algoritmos de enrutamiento que puedan admitir la comunicación punto a punto y converger el tráfico de difusión (*cast traffic*) sigue siendo un problema abierto. En el proyecto TapirNet [18] el diseño del sistema telemático de UAV consta de dos partes:

1. la selección y configuración de la plataforma de piloto automático
2. la escritura del programa de la misión, transferencia de las imágenes, programa llamado Aguilucho.

3. Metodología

En [24] se introdujo el proceso de DSRM (metodología de investigación de *design science*) basado en principios anteriores de *design science*. DSRM se presenta y se lleva a cabo en esta investigación. La metodología es compatible con los marcos de *design science* comúnmente aceptados. DSRM incluye prácticas, principios y procedimientos para dirigir el trabajo de investigación. DSRM define seis pasos para la investigación:

1. identificación del problema: observar los procesos actuales, hacer preguntas sobre los procesos actuales para establecer patrones.
2. objetivos para la solución: imaginar cómo los procesos pueden ser automatizados y / o mejorados.
3. desarrollo incluyendo el diseño: crear algunas pruebas que midan el éxito o fracaso y la mejora del rendimiento y precisión; reunir algunos datos de prueba para realizar experimentos.
4. presentación: crear algunos prototipos o pruebas de conceptos y revisar las hipótesis como resultado de la retroalimentación
5. evaluación: resumir los prototipos en soluciones generales y verificar que sean consistentes con otros procesos y datos
6. comunicación.

En el método DSRM no hay necesidad de ejecutar actividades en orden secuencial, el investigador puede comenzar en cualquiera de las primeras cuatro actividades. Las cuatro actividades iniciales se pueden elegir dependiendo de la naturaleza del input de investigación.

Siguiendo esta metodología, a continuación, se identifican los problemas que se deben enfrentar para alcanzar la meta. Debido a las limitaciones de los UAV, las características físicas como el tamaño, el peso y el consumo de energía de todos los componentes de la nave deben tenerse en cuenta. Adicionalmente, la distancia es una restricción al control remoto de las UAV, un mayor alcance de estas naves es posible con sistemas de navegación automáticos pre programados.

Con respecto a los objetivos técnicos debe decirse que el hardware de comunicación debe instalarse dentro de la plataforma UAV, y esta situación impone ciertas restricciones [30]. Si la carga útil total se asume como constante y el hardware de comunicación es más ligero, permitiría desplegar sensores adicionales en la UAV, pero, un hardware más ligero significa menor potencia computacional. Es decir existen muchos procesadores potentes y miniaturizados pero solamente para mercados específicos, por ejemplo el mercado de teléfonos móviles inteligentes; para que estos procesadores potentes y miniaturizados funcionen, requieren hardware adicional que finalmente aumentan el tamaño y peso. Los procesadores de propósito general y poco tamaño tienen una velocidad de procesamiento promedio de solamente 1 GHz, lo cual también limitará la velocidad de

la comunicación inalámbrica. La limitación de espacio es otra limitación relacionada con la plataforma UAV, la limitación de espacio es especialmente importante para la comunicación pues deben colocarse antenas y además tener separación entre las partes electrónicas para no introducir ruido [31].

Dado que se quiere procesos automáticos además de la navegación aérea, el diseño debe incluir dos computadoras, una de propósito específico denominada piloto automático y otra de propósito general. La primera se encargará de gobernar todo el sistema de vuelo y la segunda se encargará de la transferencia de datos además de automatizar todo el proceso. No es difícil diseñar el hardware para cumplir con los requisitos mencionados, el cuello de botella actual es el software. Hoy la mayoría de pilotos automáticos sigue una navegación automática mediante la ubicación de puntos cuyas coordenadas son conocidas. Para este proyecto dicha tecnología es suficiente; para nuevas aplicaciones una capa de navegación de información dinámica debería construirse sobre la actual capa de navegación basada en puntos fijos.

Se realizarán por lo menos dos juegos de pruebas y se evaluarán sus correspondientes resultados. Finalmente, esta tesis será el reporte final que se publicará abiertamente para comunicar a la comunidad tanto el proceso como los resultados obtenidos. Adicionalmente, para la comunicación no científica se utilizarán redes sociales para difundir los resultados.

3.1. Hardware

3.1.1. AeroCore

El principal sistema electrónico de una UAV es el piloto automático, que integra una unidad GPS, sensores de presión y sensores inerciales. El piloto automático permite que la UAV despegue y aterrice de forma autónoma y navegue a través de puntos de referencia definidos. Para esta investigación se evaluaron dos: AeroCore y Pixhawk, una revisión más exhaustiva puede encontrarse en [23]. Gumstix AeroCore MAV (micro vehículo aéreo) es una placa electrónica de control de vuelo que corre el sistema operativo NuttX en un microprocesador Cortex-M4 MCU, además también corre SO Linux en una DuoVero COM (*Computer on Module*) basada en un procesador Cortex-A9.

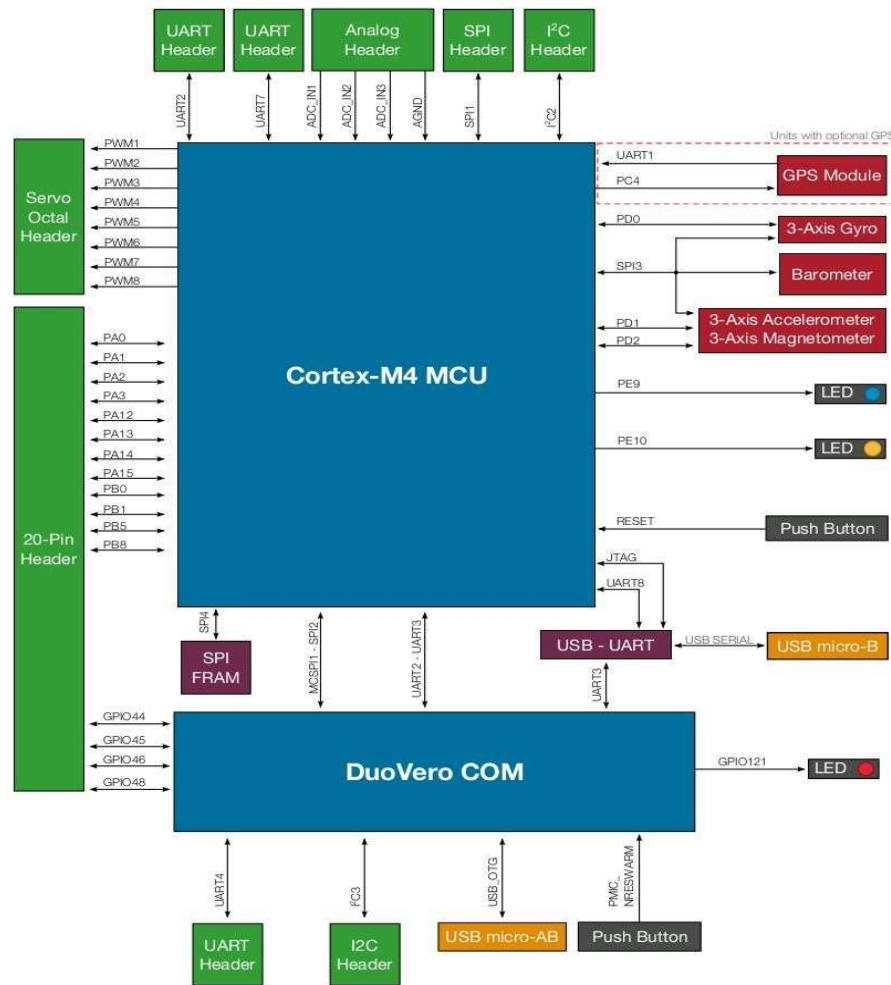


Figura 3 Arquitectura de un piloto automático. Nota. Figura extraída de Gumstix, 2020, <https://www.gumstix.com>

AeroCore MAV posee un zócalo para una conexión directa con COM DuoVero Zephyr o DuoVero Crystal. La COM está disponible por separado y corre SO Yocto (una versión de Linux). El AeroCore en sí incluye un procesador Cortex-M4 ARM (MCU) conectado (*interfaced*) directamente con la COM DuoVero tal y como se ve en la figura 3. El dispositivo está diseñado para desarrolladores de micro vehículos aéreos.

Respaldado por 1MB de SPI FRAM, en el MCU se ejecuta un sistema operativo en tiempo real (RTOS) NuttX, basado en POSIX VxWorks. El MCU soporta el control de la energía, de los motores y de los sensores, mientras que los programas de Linux de alto nivel, como Aguilucho, corren sobre Yocto. AeroCore soporta la integración con proyectos de código abierto como Robot Operating System (ROS), PX4, y proyectos compatibles con PX4 como QGroundControl y MAVLink. Este ecosistema de software permite la incorporación de firmware tales como programa de análisis de flujo óptico y algoritmos de adquisición de objetivos.

AeroCore ofrece múltiples opciones de control para un máximo de ocho motores. Se integra un acelerómetro de 6 ejes con magnetómetro, así como un giroscopio de 3 ejes y barómetro. También están disponibles puertos (*breakouts*) para añadir otros sensores. Ver lista de especificaciones debajo:

- Procesadores:

- ARM Cortex-M4 MCU
 - TI OMAP4430 (2x Cortex-A9 núcleos @ 1GHz plus PowerVR SGX450 3G GPU) separadamente disponibles con COM DuoVero Zephyr ó COM DuoVero Crystal
- Memoria — 1MB SPI FRAM para MCU; 512MB ó 1GB RAM y slot microSD en DuoVero
- Networking — WiFi y Bluetooth 3.0 vía DuoVero con conector U.FL
- Sensores:
 - GPS/GNSS opcionales
 - Acelerómetro de 6 ejes (LSM303D) con sensores digitales 3D lineales y magnéticos
 - Barómetro (MS5611), precisión 10cm ○ Giroscopio de 3 ejes (ST L3G4200D)
 - *Breakouts* para otros sensores
- Otros I/O:
 - Micro-USB OTG
 - Convertidor serial (FTDI FT4232) para hasta 4 puertos USB ○ 2 low-profile 20-pin *header* para JTAG
 - 24-pin *header* con 8x PWM salidas o 40-pin *header* para GPIO/breakout y
 - energía 6-pin *header* para Cortex-M4 SPI breakout
- Gumstix COM interface — 2 x 70-pin DF40 Hirose conectores para montaje de COM DuoVero
- Otras características — Control múltiple para hasta 8 motores
- Energía — Step-down converter (TI TPS62111) máximo 17V, 1.5A
- Sistema Operativo — Linux (Yocto Project) en DuoVero; NuttX en MCU

3.1.2. COM DuoVero Zephyr

Una *computer-on-module* (COM) es un tipo de *single board computer* (SBC), y un subtipo de un *embedded computer system*. También se le conoce como *system-on-module* (SoM) como una extensión del concepto de *system on chip* (SoC), se trata de un dispositivo que se ubica entre una computadora completa y un microcontrolador.

Las COM actuales son computadoras casi completas construidas sobre una única placa de circuito integrado. El diseño se centra en un solo microprocesador con controladores de entrada / salida, RAM y todas las demás características necesarias para ser un equipo funcional. Sin embargo, a diferencia de una placa de computadora *desktop* o *laptop*, la COM por lo general carece de los conectores estándar para todos los periféricos de entrada / salida que se conectan directamente a la placa. En cambio, el cableado para estos periféricos termina en conectores no convencionales ubicados dentro de la placa.

En general, el problema se resuelve “montando” la COM en una placa de soporte (o “placa base”), que si posee conectores estándar.

Una solución COM / SOM ofrece un sistema informático denso para su uso en aplicaciones pequeñas o especializadas que requieren bajo consumo de energía o pequeño tamaño físico.

El COM DuoVero es proveído por el fabricante Gumstix, cuenta con un doble núcleo de 1 GHz, Cortex-A9 basado en el *system-on-chip* OMAP4430 Texas Instruments emparejado con un PowerVR SGX450 3G GPU. El COM DuoVero Zephyr⁵ es casi idéntico al original DuoVero Crystal salvo que añade un módulo

⁵ <http://media.gumstix.com/docs/gs4430z.pdf>

inalámbrico Wi2Wi, que soporta 802.11b/g/n Wi-Fi y Bluetooth 3.0. Al igual que el DuoVero Cristal, el DuoVero Zephyr está equipado con 1 GB de RAM y soporta cámaras duales de alta definición 1080p. También puede manejar un rango de temperatura de funcionamiento de 0°C a 75°C.

En el diagrama de bloques de la figura 4 puede verse un esquema completo de las conexiones de AeroCore. El receptor GPS está conectado en UART1 y la velocidad de conexión recomendada es 38400 bps (el chip soporta otras velocidades). El GPS incorporado es un chip u-blox NEO-7M. Puede conectarse un GPS externo lógicamente conectado al mismo puerto UART1 (con 3.3V y GND pins del respectivo *header*) después de desactivar el GPS incorporado.

También pueden observarse 3 puertos UART en la comunicación entre DuoVero y AeroCore. El UART2 está conectado al UART3 del procesador STM32. El UART3 está conectado al puerto USB - UART y está disponible vía consola USB. Finalmente, UART4 se enlaza a un UART *header* del AeroCore. AeroCore acepta señales PWM y entradas seriales, luego es compatible con todos los receptores RC. El radio control se conecta a un canal de entrada 6 PWM (12 de los pines GPIO pueden ser usados como entradas PWM).

3.1.3. Pixhawk

El controlador de vuelo Pixhawk es una evolución del sistema controlador de vuelo PX4. Pixhawk consiste en un controlador PX4-FMU y un PX4-IO integrados en una sola tarjeta con puertos I/O adicionales, memoria y otras funciones. Está altamente optimizada para proporcionar control y automatización de software de navegación de vuelo con un alto rendimiento y capacidad. Pixhawk utiliza un sistema operativo de tiempo real llamado NuttX que cuenta con un alto rendimiento, flexibilidad y fiabilidad para controlar cualquier vehículo autónomo. Este SO proporciona un entorno de programación del tipo Unix/Linux, *multithreading* integrado y funciones de piloto automático funciones como *Lua scripting*.

Una capa de drivers a medida asegura la sincronización a través de todos los procesos.

Entre los periféricos se incluyen un sensor digital de velocidad aérea, un indicador LED multicolores y un magnetómetro externo. Todos los periféricos se detectan y configuran automáticamente.

Algunas de las características se describen debajo:

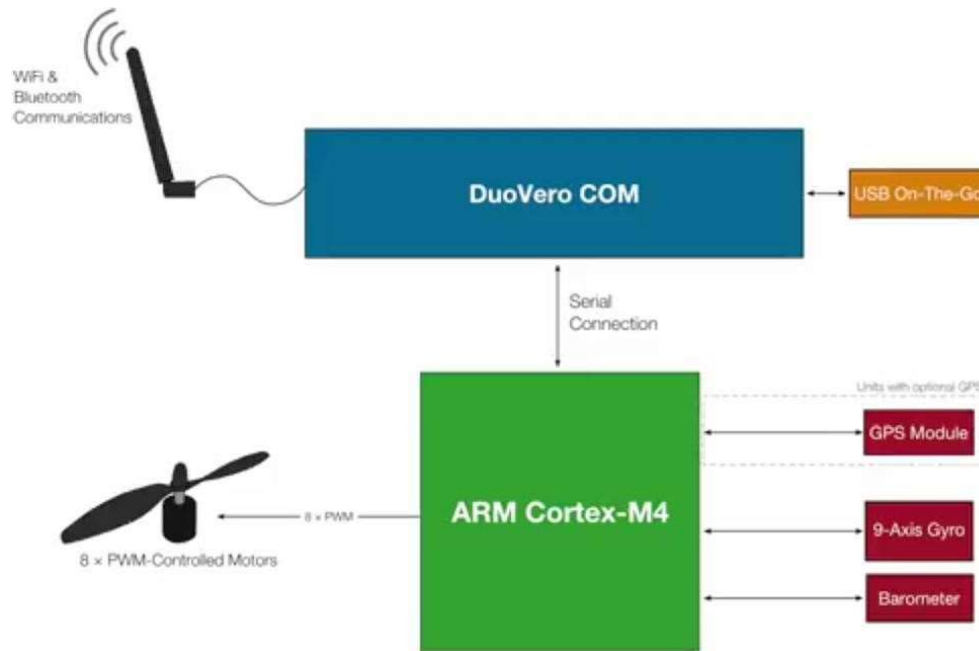


Figura 4 Diagrama de bloques de AeroCore. Nota. Figura extraída de Gumstix, 2020, <https://www.gumstix.com>

1. Procesamiento:

STM32F427 microcontrolador primario de 32 bits: 168 MHz, 252 MIPS, Cortex M4 con una unidad de punto flotante. 2MB de memoria Flash y 256 kB de memoria RAM.

1 coprocesador STM32F103 copia de seguridad a prueba de fallos de 32 bits proporciona la recuperación manual y tiene su propia fuente de alimentación. 1 memoria SD micro para registro de datos y otras aplicaciones.

2. Sensores:

giroscopio de 3 ejes 16 bits ST Micro L3GD20H para determinar la orientación.

acelerómetro de 3 ejes de 14 bits y magnetómetro para determinar las influencias externas y dirección de la brújula, magnetómetro externo opcional con conmutación automática MEAS MS5611 sensor de presión barométrica para determinar la altitud.

medidor incorporado de tensión y de corriente para la determinación de la condición de la batería.

Conexión para uBlox LEA GPS externo para determinar la posición absoluta.

3. Interfaces de E / S con conectores dedicados:

14 servo PWM o salidas de control de velocidad ESC.

05 UART (puerto serie), uno de alta potencia capaz, 2 con control de flujo de HW.

02 CAN puertos I / O (1 con transceptor 3.3V interna, 1 en conector de expansión)

01 entrada receptora de satélite Spektrum DSM / DSM2 / DSM-X® Entrada compatible

Futaba S.BUS® y salida.

Entrada de señal suma PPM.

RSSI (PWM o tensión) de entrada.

I2C y SPI puertos serie.

02 puertos de 3,3 voltios y unas entradas analógicas 6.6 voltios.

01 puerto microUSB interno y extensión de puerto microUSB externo.

4. Sistema integral de energía con redundancia y amplia protección:

Pixhawk posee una línea de alimentación para los sensores presentes.

Controlador de diodo ideal con entradas de alimentación redundantes y conmutación automática por error

Servo riel de alta potencia (máx. 10V) y de alta corriente (10A +)

Todas las salidas perifericas y entradas ESD están protegidas contra sobre-corriente

Un botón de seguridad externo permite un procedimiento seguro de activación / desactivación de motor

5. Características mecánicas: Peso: 38 g (1,31 oz), Ancho: 50 mm (1,96"), Espesor: 15,5 mm (0.613"), longitud: 81.5mm (3.21")

3.1.4. ArduPilot Mega (APM)

El APM Ardupilot es un clon de la popular placa Arduino, la cual está basada en la arquitectura de microcontroladores AVR de 32 bits ATMEGA2560. Contiene sensores MPU-6000 (acelerómetro y giroscopio) y MS5611-01BA03 (barómetro), además contiene un microcontrolador ATMEGA32U-2 para las funciones de comunicación a través del protocolo USB.

El firmware para el APM puede configurarse para trabajar tanto como aviones de ala fija como con multirrotores. Como software para la estación en tierra puede usarse Mission Planner (en Windows) o APM Planner (en Linux).

3.1.5. Periféricos

Se requieren dispositivos adicionales [37] para el correcto funcionamiento del piloto automático, tales como:

1. Controlador de velocidad electrónico (ESC): Es un dispositivo electrónico que tiene el propósito de variar la velocidad de un motor brushless a partir de una señal PWM.
2. Radio de telemetría: Transmisor y receptor de radio empleado para la comunicación con la estación en tierra. Los datos enviados son la medición de sensores (posición, orientación, voltaje, corriente) en tiempo real. También sirve para recibir instrucciones de vuelo así como las coordenadas de los puntos por donde se desea que el UAV recorra. Usualmente funciona en la banda de 433 MHz.
3. Radio control con receptor: El radio control es el dispositivo que utiliza el piloto desde tierra para controlar el UAV en un vuelo manual, adicionalmente sirve para cambiar de modo de vuelo de manual a automático.

3.2. Software

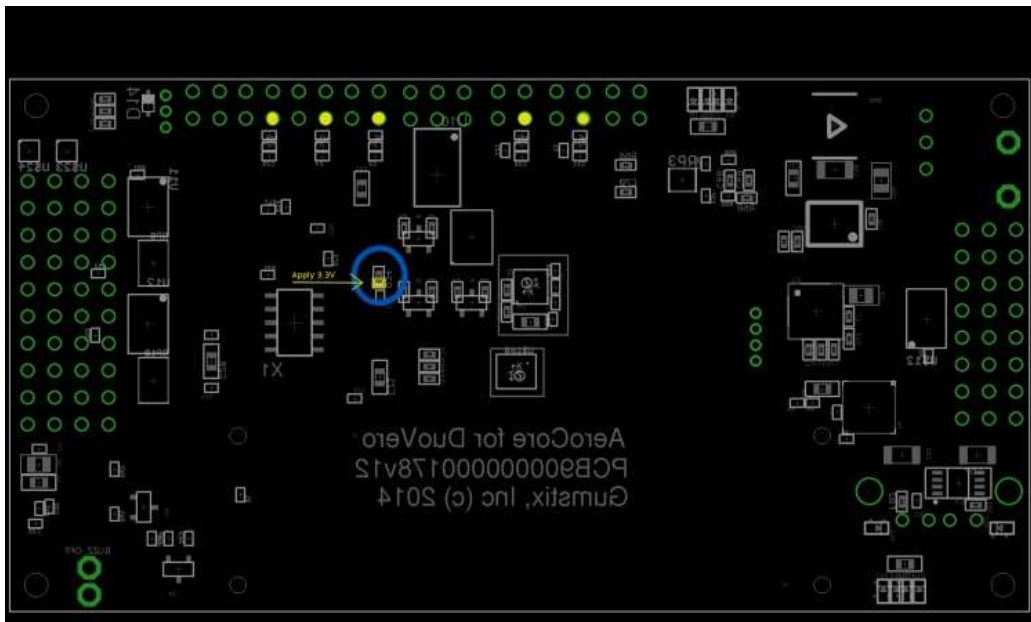
3.2.1. Grabación de firmware AeroCore y Pixhawk

Desde 2015, tanto Mission Planner⁶ como APM Planner⁷, ambos softwares de control en tierra, brindan la opción de grabar el firmware para las plataformas APM2.x, PX4 y Pixhawk. En el siguiente enlace⁸ hay una completa descripción de como grabar la versión más reciente de firmware para cada caso.

Durante el desarrollo de la presente tesis, el fabricante Gumstix liberó una versión intermedia de AeroCore, cuyo hardware se parecía mucho a AeroCore2 (diseñado para la COM Vero). Esta versión intermedia, identificada como PCB900000000178v12, requiere activar el *bootloader* (activación llamada BOOT0) y la grabación del firmware en STM32. El firmware vacío se identifica por los signos externos: un sonido molesto del buzzer y que los LEDs amarillo y azul no parpadeen intermitentemente durante el arranque. Para grabar el firmware se usará el puerto micro-USB cerca al “*long pin header*” (este puerto está conectado al STM32).

Debe activarse el sistema *bootloader*, para ello se expone a 3.3V uno de los extremos de la resistencia superficial marcada dentro del círculo azul en la figura 5. En la misma figura se muestran resaltados en amarillo los pines que proveen 3.3 V; basta un toque de 3.3V, aplicado cuando el procesador arranca, no se necesita soldar algo. Después, conectar un cable al puerto micro-USB o reiniciar la placa si el cable ya estuviera conectado (presionar el botón ubicado en una esquina de la tarjeta). Luego que la tarjeta arranca, ejecutar el comando `lsusb`, si el *bootloader* fue activado correctamente, el comando generará una respuesta como esta:

```
Bus 003 Device 018: ID 0483:df11 STMicroelectronics STM Device in DFU Mode
```



⁶ <http://copter.ardupilot.com/wiki/common-install-mission-planner/>

⁷ <http://planner2.ardupilot.com/>

⁸ <http://copter.ardupilot.com/wiki/common-loading-firmware-onto-pixhawk/>

Figura 5 Activación de BOOT0

Por ejemplo:

```
luis@luis-desktop-GTR-office:~$ lsusb
Bus 002 Device 002: ID 18a5:0302 Verbatim, Ltd 32GB Flash Drive
Bus 002 Device 001: ID 1d6b:0002 Linux Foundation 2.0 root hub
Bus 008 Device 001: ID 1d6b:0001 Linux Foundation 1.1 root hub
Bus 007 Device 001: ID 1d6b:0001 Linux Foundation 1.1 root hub
Bus 006 Device 001: ID 1d6b:0001 Linux Foundation 1.1 root hub
Bus 001 Device 001: ID 1d6b:0002 Linux Foundation 2.0 root hub
Bus 005 Device 003: ID 04b3:3025 IBM Corp. NetVista Full Width
Keyboard
Bus 005 Device 002: ID 0458:003a KYE Systems Corp. (Mouse Systems)
NetScroll+ Mini Traveler / Genius NetScroll 120
Bus 005 Device 001: ID 1d6b:0001 Linux Foundation 1.1 root hub
Bus 004 Device 001: ID 1d6b:0001 Linux Foundation 1.1 root hub
Bus 003 Device 090: ID 0483:df11 STMicroelectronics STM Device in DFU
Mode
Bus 003 Device 001: ID 1d6b:0001 Linux Foundation 1.1 root hub
```

Si no funciona probablemente no se ha aplicado correctamente el toque de voltaje o se ha usado el puerto equivocado, en tales casos insistir y/o cambiar de puerto. Debe usarse el puerto micro-USB que está cerca de los PIN HEADERS, no el puerto que está cerca del puerto USB A. El error de los puertos solamente ocurriría si se alimenta la placa con una batería externa pues el conector equivocado no sirve para encender la placa a diferencia del conector correcto. La respuesta al comando `lsusb` cuando el cable está conectado al puerto equivocado es una mención de un dispositivo FTDI.

En la primera versión de AeroCore se indicaba cortocircuitar los pines 1 y 3 del conector J11 para ejecutar el BOOT0, ello no debe hacerse en ninguna de las versiones posteriores.

A continuación, se debe cargar el bootloader PX4 y grabar el firmware. Para ello se debe instalar la herramienta "dfu-util" ejecutando `sudo apt-get install dfu-util` y luego descargar el archivo [px4aerocore_bl_AC2.bin](#) y ejecutar:

```
sudo dfu-util -a 0 -D /location of download/px4aerocore_bl.bin -s 0x08000000
```

los LEDs parpadearán si el procedimiento fue ejecutado exitosamente. Lo siguiente es grabar el firmware usando el *bootloader* PX4 que facilita este proceso, nunca más debería necesitarse hacer el BOOT0. Descargar a un mismo directorio el firmware [aerocore_default.px4](#) y el script, [px_uploader.py](#). Luego ejecutar:

```
luis@luis-Presario-CQ42-Notebook-PC:~$ sudo git clone
https://github.com/PX4/Firmware luis@luis-Presario-CQ42-Notebook-PC:~$ cd
Firmware

luis@luis-Presario-CQ42-Notebook-PC:~/Firmware$ python2
Tools/px_uploader.py /home/luis/Downloads/aerocore_default.px4 --port
/dev/ttyACM0
```

Se asume que AeroCore está reconocido como `/dev/ttyACM0`, de lo contrario colocar el directorio correcto. Si no funciona cambiar a superusuario y reiniciar AeroCore antes y después de ejecutar la grabación del firmware.

Si se hace correctamente, el LED amarillo debería parpadear unos segundos y luego apagarse, el LED azul debe encenderse y mantenerse así, además deberían escucharse algunos pulsos, no chillidos, del parlante. En pantalla se observa algo como:

```
rbm@nomad ~/Downloads $ wget "http://gumstix.8.x6.nabble."
--2015-02-25 16:40:12-- http://gumstix.8.x6.nabble.
Resolving gumstix.8.x6.nabble.com (gumstix.8.x6.nabble.com)...
162.253.133.59
Connecting to gumstix.8.x6.nabble.com (gumstix.8.x6.nabble.com)|162.
HTTP request sent, awaiting response... 200
OK
Length: 724391 (707K)
Saving to: 'aerocore_default.px4' aerocore_default.px4
2015-02-25 16:40:15 (273 KB/s) - 'aerocore_default.px4' saved
[724391/724391]
rbm@nomad ~/Thirdparty $ git clone https://github.com/PX4/
PX4_firmware
Cloning into 'PX4_firmware'... remote:
Counting objects: 128122,
done.
remote: Compressing objects: 100% (164/164),
done. remote: Total 128122 (delta 91), reused 0 (delta 0), pack-
reused 127958
Receiving objects: 100% (128122/128122), 64.90 MiB | 1.50 MiB/s, done.
Resolving deltas: 100% (95622/95622),
done.
Checking connectivity... done.
rbm@nomad ~/Thirdparty $ cd PX4_firmware/
rbm@nomad ~/Thirdparty/PX4_firmware $ python2 Tools/px_uploader.py
~/Downloads/aerocore_default.px4 --port /dev/ttyACM0
Loaded firmware for 13,0, size: 858320 bytes, waiting for the
bootloader...
If the board does not respond within 1-2 seconds, unplug and re-plug the
USB connector.
Found board 13,0 bootloader rev 4 on /dev/ttyACM0
ffffffff ffffffff ffffffff ffffffff ffffffff ffffffff ffffffff
ffffffff ffffffff ffffffff ffffffff ffffffff ffffffff ffffffff
ffffffff ffffffff ffffffff ffffffff ffffffff ffffffff type: ÿÿÿÿ
idtype: =FF vid: ffffffff pid: ffffffff coa:
//////////////////////////////////////// sn: 0036001d3432471530333837

Erase : [=====] 100.0%
Program: [=====] 100.0%
Verify : [=====] 100.0%
Rebooting.
rbm@nomad ~/Thirdparty/PX4_firmware $ cu -l /dev/ttyACM0
Connected. nsh>
```

3.2.2. Comunicación Serial

PX4 puede comunicarse con la COM por vía serial o bus SPI, sin embargo esta última opción no está desarrollada. Por defecto, PX4 transmite data Mavlink sobre el canal UART hacia la COM. DuoVero y M4 (PX4/Nuttx) están conectados en /dev/ttyO1 (DuoVero) <-> /dev/ttyS2 (PX4/Nuttx), esto puede confirmarse ejecutando los comandos:

```
cat /dev/ttyO1 , ejecutar en DuoVero mavlink stop-all , ejecutar
en PX4/Nuttx mavlink start -d /dev/ttyS2 -b 115200 , ejecutar en
PX4/Nuttx
```

en DuoVero se verá una secuencia de caracteres no legibles.

3.2.3. MAVLink

MAVLink es un protocolo de comunicaciones. Un mensaje MAVLink es básicamente una secuencia de bytes codificados y enviados a través de un transductor (a través de serie USB, radio frecuencia, WiFi, GPRS, etc.). los mensajes MAVLink están compuestos solamente de una cabecera, es muy ligero, Mediante la codificación se ordena la información en una estructura de datos de *manera inteligente* añadiendo un **checksum (suma de control)**, **número de secuencia** y se envía a través del canal byte a byte. A continuación se presenta un ejemplo del funcionamiento del protocolo, en el ejemplo se enviará y se recibirá un mensaje MAVlink. Se ejecutan los comandos:

```
$ git clone https://github.com/mavlink/c_library
```

```
$ git clone https://github.com/mavlink/c_uart_interface_example.git
```

```
$ cd c_uart_interface_example/
```

```
$ vi makefile
```

Se edita el archivo, se debe añadir una mención a la ubicación de la carpeta `c_library`, de lo contrario

arrojará un error como: `mavlink_control.h:70:28: fatal error: common/mavlink.h:`

No such file or directory. La edición final debe quedar:

```
$ g++ -I ../c_library/ -I mavlink/include/mavlink/v1.0
```

```
mavlink_control.cpp serial_port.cpp autopilot_interface.cpp -o
```

```
mavlink_control -lpthread
```

```
$ make
```

Luego, conectar el cable USB al piloto automático PX4 ó AeroCore e iniciar sesión remota:

```
screen /dev/ttyACM0 57600 8N1
```

```
<press enter>
```

Si el Puerto anterior no funciona, encontrar el puerto correcto usando:

```
$ ls /dev/ttyACM*
```

```
$ ls /dev/ttyUSB*
```

También puede desconectarse el cable USB y reconectar ejecutando el comando:

```
$ dmesg
```

El dispositivo mostrado al final de la respuesta del comando `dmesg` será el puerto en el cual está conectado el piloto automático.

Para iniciar una sesión MAVlink en el puerto USB de AeroCore se ejecuta:

```
nsh> mavlink start -d /dev/ttyACM0
```

AeroCore enviará una secuencia infinita de caracteres irreconocibles, salir con la secuencia de teclas Ctrl+A , K, Y

La transmisión y recepción vía MAVlink se ejecuta con los comandos:

```
$ cd c_uart_interface_example/
```

```
$ ./mavlink_control -d /dev/ttyACM0
```

Para detener el programa se ejecuta la secuencia Ctrl-C. La salida del programa es:

```
OPEN PORT
Connected to /dev/ttyUSB0 with 57600 baud, 8 data bits, no parity, 1 stop
bit (8N1)
START READ THREAD

CHECK FOR HEARTBEAT
Found

GOT VEHICLE SYSTEM ID: 1 GOT
AUTOPILOT COMPONENT ID: 50

INITIAL POSITION XYZ = [ 8.2935 , -1.1447 , -0.7609 ]
INITIAL POSITION YAW = 2.1539

START WRITE THREAD

ENABLE OFFBOARD MODE

SEND OFFBOARD COMMANDS
POSITION SETPOINT XYZ = [ 3.2935 , -6.1447 , -0.7609 ]
POSITION SETPOINT YAW = 2.1539
0 CURRENT POSITION XYZ = [ 8.2935 , -1.1447 , -0.7609 ]
1 CURRENT POSITION XYZ = [ 8.2935 , -1.1447 , -0.7609 ]
2 CURRENT POSITION XYZ = [ 8.2524 , -1.1444 , -0.7667 ]
3 CURRENT POSITION XYZ = [ 8.2205 , -1.1431 , -0.7747 ]
4 CURRENT POSITION XYZ = [ 8.1920 , -1.1421 , -0.7737 ]
5 CURRENT POSITION XYZ = [ 8.1920 , -1.1421 , -0.7737 ]
6 CURRENT POSITION XYZ = [ 8.1539 , -1.1414 , -0.7847 ] 7 CURRENT
  POSITION XYZ = [ 8.1522 , -1.1417 , -0.7820 ]

DISABLE OFFBOARD MODE

READ SOME MESSAGES
Got message LOCAL_POSITION_NED
(spec:https://pixhawk.ethz.ch/mavlink/#LOCAL_POSITION_NED)
pos (NED): 8.152975 -1.141093 -0.784075 (m)
Got message HIGHRES_IMU
(spec:https://pixhawk.ethz.ch/mavlink/#HIGHRES_IMU)
ap time: 3611390110 acc (NED): 0.005503
0.044659 -9.740363 (m/s^2) gyro (NED): -0.003064
0.003857 0.000005 (rad/s) mag (NED): -0.117767 -
0.335362 -0.253204 (Ga) baro: 1020.519958
(mBar) altitude: -60.341393 (m) temperature:
46.779999 C

CLOSE THREADS
```

CLOSE PORT

3.2.4. NuttX sobre PX4

Existe una compilación del código fuente del Ardupilot Mega (APM) para el autopiloto Pixhawk/PX4, el cual provee al PX4 de similares capacidades que el APM, sin embargo existe otra alternativa, la cual es usar el sistema operativo Nuttx, que provee las herramientas del sistema operativo tales como ejecución de programas desde *shell*, librerías, ejecución de aplicaciones en simultaneo, etc. Por lo mencionado anteriormente es conveniente la instalación del software libre Nuttx en el Pixhawk/PX4, la versión ofrecida por los creadores del Pixhawk/PX4 trae el software necesario para el control del UAV.

Para la compilación y grabado del sistema operativo Nuttx sobre el Pixhawk/PX4 primero se debe instalar el compilador para microcontroladores ARM en Linux (Ubuntu), para ello se siguen los siguientes pasos:

1. Se descarga y descomprime los ejecutables de compilador

```
wget https://launchpadlibrarian.net/174121628/gcc-arm-none-eabi-4_7-2014q2-20140408-linux.tar.bz2
tar -jxf gcc-arm-none-eabi-4_7-2014q2-20140408-linux.tar.bz2
```

2. Luego se agrega la ruta, PATH, apuntando a donde están los ejecutables del compilador

```
exportline="export PATH=$HOME/gcc-arm-none-eabi-4_7-2014q2/bin:$PATH"
```

3. Tener en cuenta que el compilador ha sido ejecutado para trabajar en plataformas de 32 bits, si se está trabajando en plataformas de 64 bits se debe instalar el paquete ia32-libs

```
sudo apt-get install ia32-libs
```

4. Con el compilador instalado, lo que sigue a continuación son los pasos para la descarga y compilación de Nuttx para la arquitectura del Pixhawk/PX4. Se crea una nueva carpeta (por ejemplo `src`) donde se guardará los códigos fuente para luego se descarga el código fuente del sistema operativo.

```
mkdir ~/src
cd ~/src
git clone https://github.com/PX4/Firmware
```

5. Esto generará una copia del código fuente en la carpeta `~/src`. Luego se debe enlazar con los submodulos `init` y `update`:

```
cd Firmware
git submodule init
git submodule update
```

6. Después se procede a la compilación del código fuente

```
make distclean
make
archives
```

```
make make px4fmu-  
v2_default
```

7. Luego de esta etapa la imagen del sistema operativo se encuentre en el directorio ~/src/Firmware/Images. Ahora se debe colocar el sistema operativo en el Pixhawk. Una vez que se ha conectado el Pixhawk por el puerto USB a la computadora se ejecuta el siguiente comando:

```
make upload px4fmu-v2_default
```

8. Si todo salió correctamente se debe mostrar el pantalla los siguientes mensajes:

```
Found board 5,0 on  
/dev/tty[...] erase...  
program... verify...  
done, rebooting.
```

Una vez grabado el sistema operativo, cuando se reinicie el Pixhawk se oirá una secuencia de tonos indicando que Nuttx está corriendo sobre el Pixhawk.

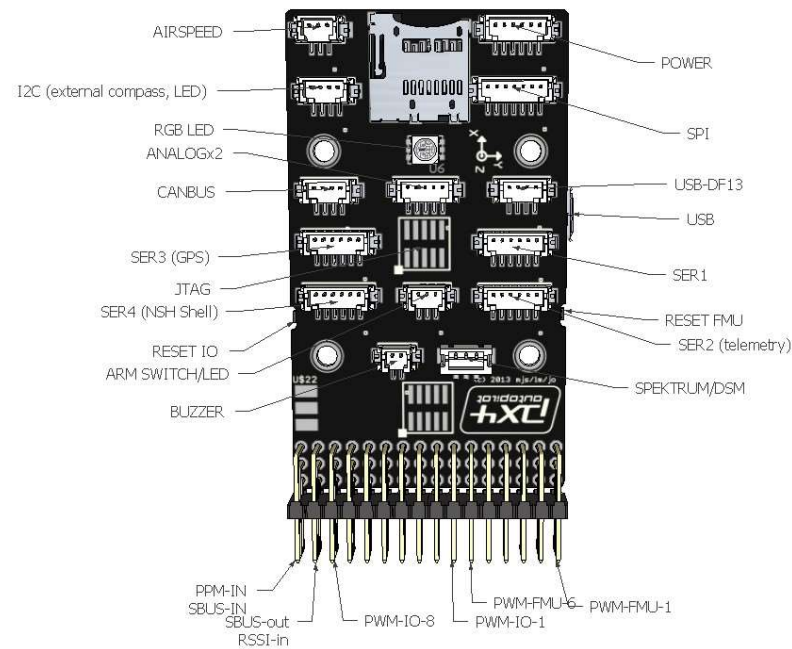


Figura 6 Piloto automático PixHawk.

Nota. Figura extraída de Gumstix, 2020, <https://www.pixhawk.org>

3.2.5. Control en Tierra QGroundControl

El software de control en tierra sirve para configurar parámetros y calibrar sensores del autopiloto antes del vuelo, monitorear en tiempo real los sensores del autopiloto durante el vuelo así como también generar registros de cada uno de los sensores para su posterior visualización.

El modo de vuelo autónomo es una característica muy importante de los autopilotos. El software de control en tierra también sirve para la generación de rutas de vuelo mediante la selección de puntos en un mapa georreferenciado; por cada uno de los puntos (waypoints) se debe indicar la velocidad y altura que debe tener el UAV, en la figura 7 se puede apreciar la selección de waypoints. AeroCore y Pixhawk/PX4 ofrecen la posibilidad de predefinir una ruta de vuelo usando coordenadas GPS, las cuales luego serán seguidas por la UAV al momento de cambiar el modo de vuelo desde el radiocontrol.

Debido a su alta compatibilidad con el piloto automático AeroCore y el sistema operativo NuttX, en este proyecto se escogió el software QGroundControl (QGC)

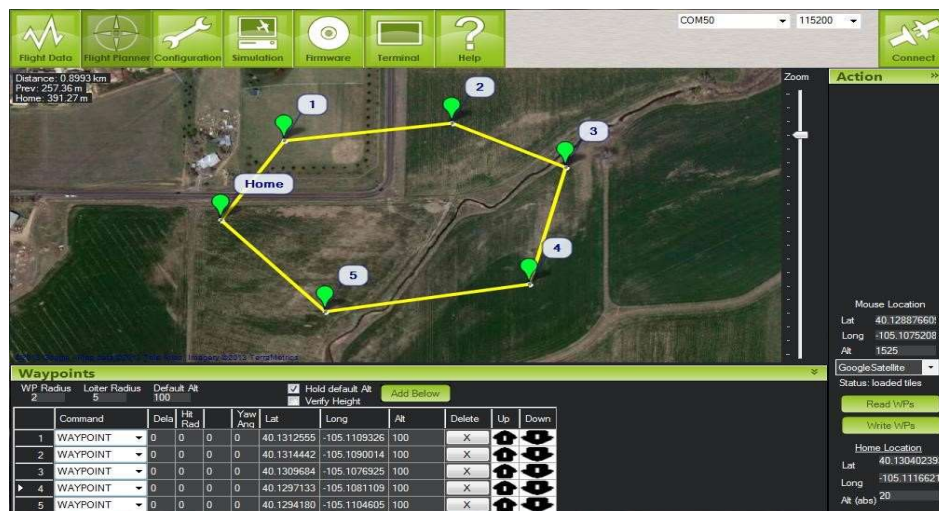


Figura 7 Entorno gráfico de un software de control en tierra.

Nota. Figura extraída de Gumstix, 2020, <http://qgroundcontrol.com/>

QGC puede instalarse fácilmente en Ubuntu 14.04 LTS (distribución que será mantenida por lo menos hasta 2018) haciendo uso de la herramienta apt-get con una actualización previa de los repositorios:

```
sudo add-apt-repository ppa:beineri/opt-qt551-trusty sudo add-apt-repository ppa:qgroundcontrol/ppa sudo apt-get update && sudo apt-get install qgroundcontrol
```

Para construir QGroundControl hacer lo siguiente:

1. Ir al directorio raíz qgroundcontrol
2. Ejecutar `qmake -r qgroundcontrol.pro`
3. Ejecutar `make`

Si existen problemas en la conexión con Pixhawk, intentar lo siguiente:

1. `sudo chmod 666 /dev/ttyACM0` → configurar los permisos adecuados
2. `sudo usermod -a -G dialout name` → reemplazar name con tu cuenta y reiniciar sesión.
Esta acción añade el usuario al grupo dialout-group
3. `sudo apt-get remove modemmanager` → modemmanager tiende a ocupar un dispositivo sólo brevemente, pero sus pruebas iniciales crean problemas a cualquier software de actualización del firmware que reinicia Pixhawk.

Finalmente, QGroundControl puede “hablar” con dispositivos inalámbricos XBee usando su protocolo propietario directamente en plataformas Windows y Linux. Este soporte no es necesario si no se usa dispositivos XBee o su protocolo propietario. En Linux, ir a la carpeta `libs/thirdparty/libxbee` y correr `make` y `sudo make install` para instalar `libxbee` (la desinstalación se puede hacer con `sudo make uninstall`), las dependencias necesarias se incluyen en ese repositorio y no se requieren pasos adicionales, `qmake` detectará automáticamente la biblioteca en Linux, por lo que no es necesario ningún otro trabajo. Para desactivar el soporte XBee se debe agregar `DISABLE_XBEE` al argumento `DEFINES`.

3.2.6. Configurar DuoVero como Access Point

Hubo muchos problemas con la imagen pregrabada (descargable desde <https://www.yoctoproject.org/downloads>) en la tarjeta microSD debido a que no poseía los paquetes necesarios para la configuración de `hostapd` y por tanto no podía configurarse DuoVero como Access Point; por ello se optó por usar la imagen disponible en <http://jumpnowtek.com/downloads/duovero/dizzy/>. Esta imagen alternativa sí dispone del paquete `hostapd`, así como también los archivos MLO y u-boot necesarios.

Archivo	Función
Roots.tar.bz2	Archivo para el rooteo del sistema
u-boot.img	Imagen binaria del <i>bootloader</i>
MLO	Segunda fase de la imagen binaria del <i>bootloader</i>

Tabla 1 Archivos para crear disco de arranque

Una completa instalación de Yocto Project consta de los archivos principales de la Tabla 2. Para instalar las imágenes, montar las particiones que se crearon con los siguientes comandos:

```
$ sudo mkdir /media/{boot,rootfs}
$ sudo mount -t vfat /dev/<DEVICE NAME>1 /media/boot
$ sudo mount -t ext3 /dev/<DEVICE NAME>2 /media/rootfs
```

Luego copiar los archivos binarios y el kernel de Linux, tener en cuenta que el archivo MLO debe ser copiado primero que los otros.

```
$ sudo cp MLO /media/boot/
```

```
$ sudo cp u-boot.img /image/boot/
```

Descomprimir el archivo rootfs en la segunda partición creada

```
$ sudo tar -xjvf rootfs.tar.bz2 -C /media/rootfs
```

```
$ sync
```

Finalmente desmontar las particiones, la tarjeta microSD *bootable* esta lista para ser utilizada.

```
$ sudo umount /media/boot
```

```
$ sudo umount /media/rootfs
```

En el archivo `/etc/network/interfaces`, se configura la interfaz quitando los símbolos #, se escribe una dirección IP 192.168.5.1

```
--- /etc/network/interfaces ---  
- ### access point interface  
auto uap0 iface uap0 inet  
static      address  
192.168.5.1  
            netmask 255.255.255.0
```

El archivo para la configuración del hostapd es `/etc/hostapd.conf`.

```
--- /etc/hostapd.conf ---  
interface=uap0  
driver=nl80211 channel=6  
ssid=duoverouav  
ignore_broadcast_ssid=0  
wpa=3  
wpa_passphrase=duovero-  
pucp_rsn_pairwise=CCMP
```

Se asigna el valor YES al hostapd daemon en `/etc/default/hostapd`.

```
--- /etc/default/hostapd ---  
HOSTAPD_ENABLE=yes
```

Se edita la configuración principal del archivo `/etc/dhcp/dhcpd.conf`.

```
--- /etc/dhcp/dhcpd.conf ---  
ddns-update-style none;  
option domain-name "jumpnow";  
option domain-name-servers 192.168.10.2;  
  
#make uap0 the Gateway for clients  
option routers 192.168.5.1;  
default-lease-time 600; max-lease-  
time 7200; authoritative;  
    subnet 192.168.5.0 netmask 255.255.255.0  
    {  
        range 192.168.5.100 192.168.5.120;  
    }
```

Esta configuración nos dará direcciones IP comprendidas entre 192.168.5.100 – 192.168.5.120

Se especifica la interfaz dhcp server

```
--- /etc/default/dhcp-server ---
```

```
INTERFACES="uap0"
```

Después de hacer todos los cambios para la configuración, se reinician los servicios:

```
root@duovero# /etc/init.d/dhcp-server
stop root@duovero# /etc/init.d/hostapd
stop root@duovero# ifdown uap0
root@duovero# ifup uap0 root@duovero#
/etc/init.d/hostapd start
root@duovero# /etc/init.d/dhcp-server start
```

3.3. *Diseño mecánico de la nave*

Muchas investigaciones han tomado interés en los vehículos aéreos no tripulados (UAV) pequeños de ala rotatoria, por su potencial uso en aplicaciones civiles y militares [39]. Los UAV son importantes cuando se debe realizar tareas peligrosas o en ambientes inaccesibles que pueden poner la vida humana en riesgo.

Cualquier sistema de propulsión que opera por intermedio de la ingesta de aire, como la combinación de un motor y hélice, es conocido como rotor. Estos sistemas obtienen el empuje al añadir momento a un volumen de aire. A medida que giran las hélices, hace que el aire alrededor acelere, lo cual resulta en el desarrollo de empuje en la dirección opuesta al flujo. Un conjunto de rotores sincronizados es conocido como helicóptero, de acuerdo a la cantidad de rotores, el helicóptero recibe del nombre de *quadróptero*, *hexacóptero* u *octacóptero*.

Las hélices del rotor de un helicóptero son responsables de tres funciones básicas:

1. La generación de la fuerza de sustentación (empuje) que se opone al peso de la aeronave.
2. La generación de la fuerza horizontal para lograr el desplazamiento del vehículo durante el vuelo.
3. La generación de fuerza y momentos para controlar la actitud y altitud del helicóptero.

3.3.1. Efectos Aerodinámicos

Existen cuatro efectos principales, tres de los cuales son medibles y pueden ser integrados en modelos dinámicos no lineales del quadróptero para realizar trabajos de diseño del control del vehículo, el cuarto efecto causa flujo discontinuo y puede ser controlado a través del rediseño de la estructura.

Los tres efectos aerodinámicos medibles están relacionados con el movimiento del quadróptero en un flujo de aire libre, ellos son:

1. El primer efecto es que el empuje total producido difiere de la potencia que pueden entregar los motores.
2. El segundo efecto resulta de la diferencia de velocidades de aire que existe entre la hélice que avanza y la que retrocede, esto lleva a que exista un *flapping* o aleteo involuntario de las hélices

lo cual induce a momentos de alabeo y cabeceo en cada rotor así como una deflexión del vector empuje.

3. El tercer efecto es la interferencia causada por el fuselaje del vehículo en la corriente de deslizamiento de las hélices. Se da como resultado un comportamiento de empuje inestable, haciendo que se dificulte la actitud de vuelo del quadróptero.

El primer efecto es causado por el comportamiento natural de un ala al deslizarse por el aire. La hélice no es 100% eficiente en su funcionamiento, esto quiere decir que del 100% de las revoluciones del motor solo el 70% son utilizadas por la hélice para generar la fuerza de sustentación. Esto se debe a que la hélice al momento de girar enfrenta la fuerza de resistencia del aire producto de la fricción que existe entre la hélice y el aire y la diferencia de presión que existe entre el borde de ataque y el de fuga de la hélice. Esta fuerza impide que la hélice gire con libertad y sea 100% eficiente. También existe la resistencia inducida que es originada por el flujo descendente en los extremos de la hélice. Este flujo descendente origina vórtices en los extremos de la hélice lo cual tiene como consecuencia que en esta parte de hélice no se produzca sustentación y por ende decrece su performance. Por este motivo es importante tener en consideración la velocidad de vuelo y el ángulo de ataque del vehículo al momento de volar. El segundo efecto está relacionado con un comportamiento aerodinámico que se da en las aeronaves de ala rotatoria denominado “disimetría de la sustentación”. La disimetría de la sustentación se da por la diferencia de sustentación que existe entre la pala de la hélice que avanza y la pala que retrocede. Esto ocasiona que un lado de la hélice genere mayor fuerza de sustentación que el otro y por lo tanto el rotor se encuentra desbalanceado y se da el fenómeno de *flapping*. Este fenómeno se incrementa a medida que la velocidad horizontal de vuelo es mayor. La figura 8 ilustra el fenómeno de disimetría de la sustentación.

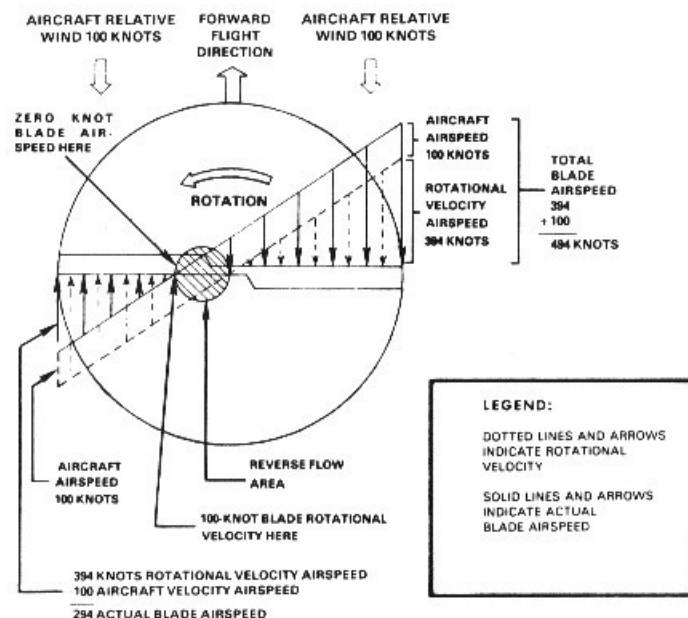


Figura 8 Disimetría de la sustentación, Nota. Recuperado de Introduction to Helicopter Aerodynamics Workbook (pag. 55), 2000, US Navy

3.3.2. Parámetros de un quadróptero

El quadróptero es un helicóptero con cuatro motores/rotores montados al final de cada uno de los cuatro brazos. No requiere un mecanismo complejo de control de actuación para cada motor; en lugar de este mecanismo utiliza hélices de paso fijo y la variación de la velocidad en cada uno de los rotores para controlar el vehículo. En la mayoría de los casos, cada hélice está conectada directamente a un motor DC *brushless*. Estas consideraciones muestran que el vehículo es rígido y que el control se alcanza al realizar pequeños cambios en la velocidad de los rotores, ya que el vehículo es muy sensible a dichos cambios [38].

Entonces, dependiendo de la fuerza de giro en cada rotor, el quadróptero cambiará su “actitud de vuelo” y podrá ser maniobrable en la dirección de vuelo que se desee, tal cual lo haría un avión convencional. La figura 9 muestra las diferentes combinaciones de fuerzas y los resultados en actitud que se logran. A medida que los rotores giran, producen fuerza de sustentación y torque, o par motor, de los rotores sobre el fuselaje. Por lo general, el par motor tiende a que el helicóptero gire fuera de control. En un quadróptero, dos de los rotores giran en sentido anti horario, y los otros dos giran en sentido horario para que las fuerzas de torque se anulen mutuamente.

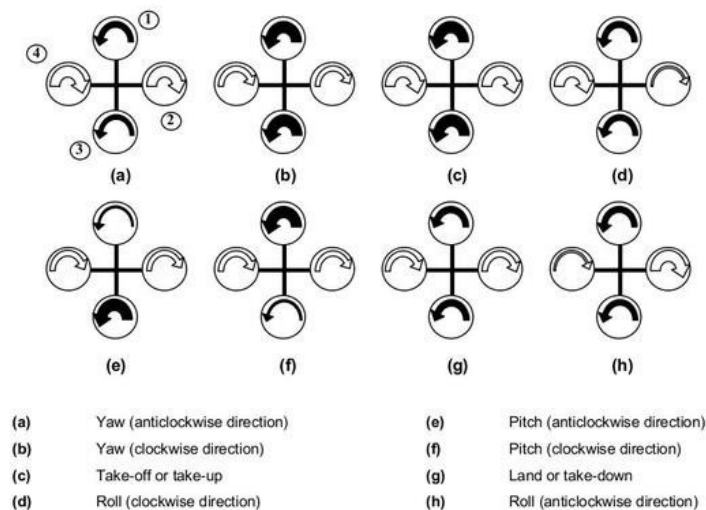


Figura 9 Actitudes de Vuelo. Nota. Recuperado de Introduction to Helicopter Aerodynamics Workbook (pag. 65), 2000, US Navy

3.3.3. Elección de Motor y Hélice

El quadróptero no requiere un complejo control mecánico para el funcionamiento de los rotores, sino que utiliza rotores de paso fijo y para controlar el vehículo, varía la velocidad en cada motor.

Rotores eficientes, compactos y que generen la suficiente fuerza de sustentación son esenciales para lograr tiempos de vuelo prolongados y levantar dispositivos electrónicos como carga útil. La potencia requerida es inversamente proporcional al radio del rotor. El radio del rotor debe ser maximizado para reducir la potencia para un determinado empuje necesario, siempre tomando en consideración las dimensiones del vehículo, peso de los rotores y el ambiente donde va a operar el vehículo. Rotores

relativamente pequeños deben de girar muy rápido para generar el suficiente empuje, comparado con los rotores más grandes. El empuje producido por un rotor es proporcional al radio del rotor elevado a la cuarta y al cuadrado de su velocidad angular.



Figura 10 Motor TAROT 5008

Para el diseño particular de este quadc6ptero, se decidi6 utilizar un motor con las siguientes caracteristicas:

Fabricante: TAROT

Modelo: 5008 High Power Brushless Motor

KV: 340kv

Polos: 14

N6mero de celdas de batera Li-Po: 6s

Corriente sin carga: 0.6A @ 10V

M6ximo empuje: 3.7kg con una h6lice 1855, alimentaci6n de 25V@26A

M6xima Potencia: 700W / 40A

Di6metro: 58.5mm

Peso: 168g

Este modelo de motor puede verse en la figura 10 y fue elegido porque puede producir el empuje necesario con relativamente poco consumo de energa. Sin embargo, su eficiencia puede ser mejorada dependiendo de la h6lice que se utilice. La h6lice ha sido fabricada en fibra de carbono, lo cual hace que sea durable y flexible ante cualquier golpe que pueda sufrir durante las pruebas u operaciones. El peso de la h6lice es de 23.3g y su tama6o es el adecuado para el tipo de quadc6ptero que se desea fabricar, una h6lice de mayor tama6o incrementara excesivamente el di6metro del quadc6ptero y lo podrfa hacer inestable durante los aterrizajes en condiciones de agresiva velocidad de viento.



Figura 11 Hélice 1655

3.3.4. Consumo de Energía vs. Potencia de los 4 Rotores

El fabricante Tarot ha realizado pruebas de empuje de este motor con una hélice de fibra de carbono 1855, obteniendo los siguientes resultados:

Empuje (kg)	Voltaje (V)	Corriente (A)	RPM	Potencia / kg
1.0	25	4.1	3369	102.50
1.5	25	6.6	4004	110.00
2.0	25	10.1	4604	126.25
2.5	25	13.6	5075	136.00
3.0	25	18.8	5518	156.67
3.5	25	24.4	6051	174.29

Tabla 2 Pruebas de empuje de juego de motores con hélices 1855

De este cuadro [40] se pueden concluir que, para obtener un empuje de 1kg, utilizando una batería de 25V, por cada motor del quadcóptero se tendrá un consumo de 4.1 amperios y las hélices girarán a 3369 rpm, generando una relación potencia peso de 102.5 VA/kg. Si se desea aumentar el empuje, el consumo de corriente y las revoluciones aumentaran, así como la potencia.

Los cuatro motores de un quadcóptero tendrán un consumo acumulado de 16.4A, para levantar un peso total de 4kg, por lo tanto, una batería de 12Ah alimentaría el sistema aproximadamente 0.73horas o 44 minutos. Esto significa que en teoría este quadcóptero podría volar 40 minutos como máximo.

Sin embargo, para este proyecto se utilizará el mismo motor, pero con una hélice de paso fijo de fibra de carbono 1655, es decir una hélice más pequeña. Una hélice se caracteriza por el diámetro y el paso o ángulo de *pitch* que posee. Las primeras dos cifras significan el diámetro que posee la hélice en pulgadas, en este caso la hélice que se usara tiene un diámetro de 16 pulgadas, los segundos dos dígitos se refieren al paso o *pitch* de la hélice, esto se refiere a la distancia en pulgadas que avanza la hélice al realizar una revolución completa, en este caso la hélice tiene un *pitch* de 5.5 pulgadas. Si se incrementa el *pitch*, la hélice tendrá más resistencia al avance y necesitará de un motor más potente para poder girar. El hecho de que esta hélice sea denominada de “paso fijo” significa que el valor del ángulo de pitch no puede

variar durante el vuelo, como si sucede en hélices con “paso variable”, por lo tanto si se desea incrementar el empuje que genera la hélice lo que se debe de hacer es incrementar las revoluciones del motor, en una hélice de paso variable se puede incrementar el empuje del motor modificando el ángulo de *pitch* de la hélice sin necesidad de incrementar las revoluciones del motor, lo cual hasta cierto punto es más eficiente especialmente si se desea volar a grandes altitudes. La tabla 3 muestra los valores en juego, utilizando el motor antes descrito y la hélice con dimensiones 1655.

Empuje (kg)	Voltaje (V)	Corriente (A)	Potencia / kg
0.53	25	1.3	61.32
1.08	25	4.2	97.40
1.53	25	7.5	122.55
2.02	25	11.5	142.33

Tabla 3 Pruebas de empuje de juego de motores con hélices 1655

Se observa que para obtener un empuje de aproximadamente 1kg se tendrá un consumo de 4.2 amperios en cada rotor aproximadamente. Se espera que todo el quadróptero, con sus componentes electrónicos y carga útil no tenga un peso mayor a los 4kg, con lo cual, los números antes señalados serían lo necesario para que el quadróptero pueda operar sin ningún problema. Como parte del sistema propulsivo se utilizarán dos baterías de 22.2V y 5000mAh, como la que se muestra en la figura 12.



Figura 12 Batería LiPo de 22.2V – 5000mAh

Dado el consumo de los rotores, las dos baterías le darán al helicóptero un tiempo de vuelo aproximado de 36 minutos. Sin embargo, estas dos baterías también alimentarían todos los sistemas a bordo del quadróptero, por lo que este tiempo de vuelo se verá reducido aproximadamente a 30 minutos o menos ya que el valor exacto de vuelo dependerá mucho de las condiciones atmosféricas. Por ello se reducirá el peso total del quadróptero para que los rotores tengan un consumo más bajo de energía. A continuación se muestran imágenes obtenidas durante pruebas de consumo de energía del rotor.

3.3.5. Diseño de Fuselaje

Para este proyecto, el diseño detallado comenzó preparando diferentes bosquejos del vehículo, incluyendo el peso y volumen de la carga útil, los componentes electrónicos. Luego de tener un bosquejo final, se realizó un modelo 3D utilizando el software CAD INVENTOR, para tener más a detalle la geometría de todas las piezas necesarias para construir el quadróptero. Asimismo, se realizó una

representación en 3D, utilizando el mismo software, de todos los componentes electrónicos y de telecomunicaciones para corroborar que podrían ser integrados y minimizar el riesgo de tener una falla en las dimensiones de las partes. La integración de las partes también fue analizada utilizando el software CAD y se confirmó que todas las partes encajaban correctamente.



Figura 13 Pruebas de consumo de energía del rotor



Figura 14 Empuje de rotor a 1kg y consumo de corriente de 4.13 amperios a 25 voltios

Al mismo tiempo que se pensaba en la forma del quadróptero y la integración de las partes, también se tuvo en consideración los materiales a utilizar para cada una de las partes. Los parámetros que se utilizaron para la selección de material fueron: durabilidad, fuerza, la facilidad de maquinar las piezas, peso, disponibilidad y costo. Se decidió usar dos materiales:

1. el fuselaje principal, los trenes de aterrizaje y parte de los brazos fueron impresos en ABS en las impresoras 3D de la Pontificia Universidad Católica del Perú,
2. los brazos son de fibra de carbono al igual que las hélices.

Si bien es cierto que el material de impresión ABS no es el más eficiente para este tipo de aplicaciones, porque no es un material que se considere ligero, sin embargo es lo suficientemente durable y fuerte para soportar las condiciones de vuelo de las operaciones de este quadróptero en la selva del Perú. Los brazos de fibra de carbono le darán ligereza y rigidez a esta esta parte crítica del quadróptero. Se espera que en el futuro cercano se puedan desarrollar la mayor cantidad de partes en fibra de carbono para reducir el peso del quadróptero y fortalecer su rigidez. En las figuras 15,16 y 17 se ve de color blanco lo que será impreso en material ABS y de color negro lo que es de fibra de carbono. Asimismo, se puede observar donde serán colocadas las baterías, los motores y el GPS para realizar vuelos autónomos.

Una consideración importante del diseño, que se puede observar en las imágenes, es la ubicación de los rotores, estos serán instalados al revés que lo usual, de tal forma que el motor estará cubierto para que pueda operar en condiciones de lluvia; esta posición no limita o perjudica la eficiencia de operación de los rotores. La ubicación del GPS también es importante, se coloca por encima de todos los demás sistemas y lo más lejos posible de las baterías para evitar cualquier interferencia magnética al compás o brújula magnética integrada dentro del GPS.

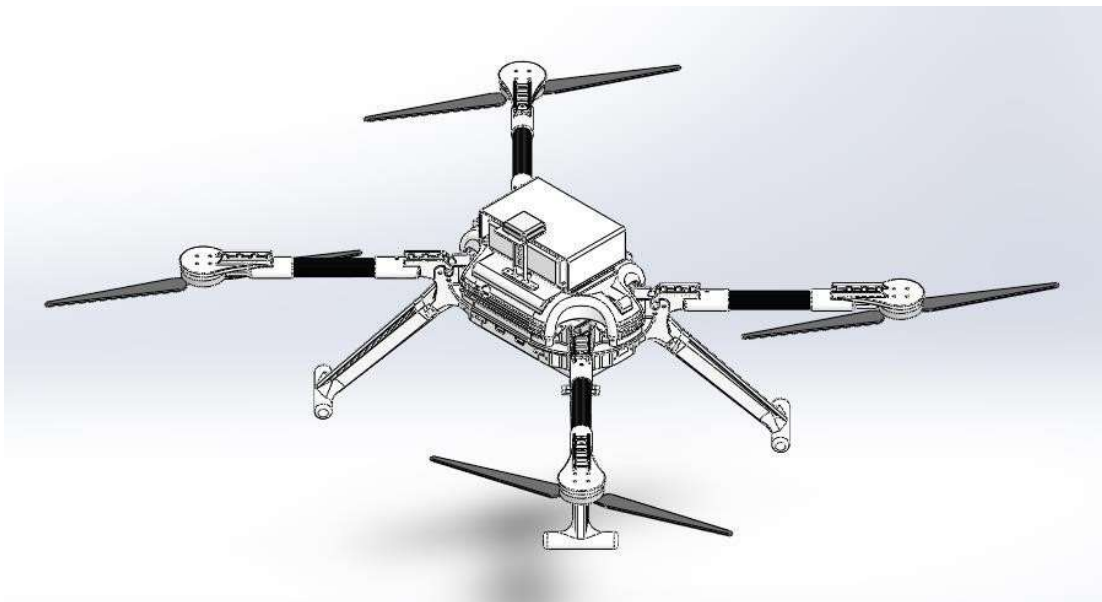


Figura 15 Diseño de quadróptero en 3D, software Solidworks, vista Isometrica

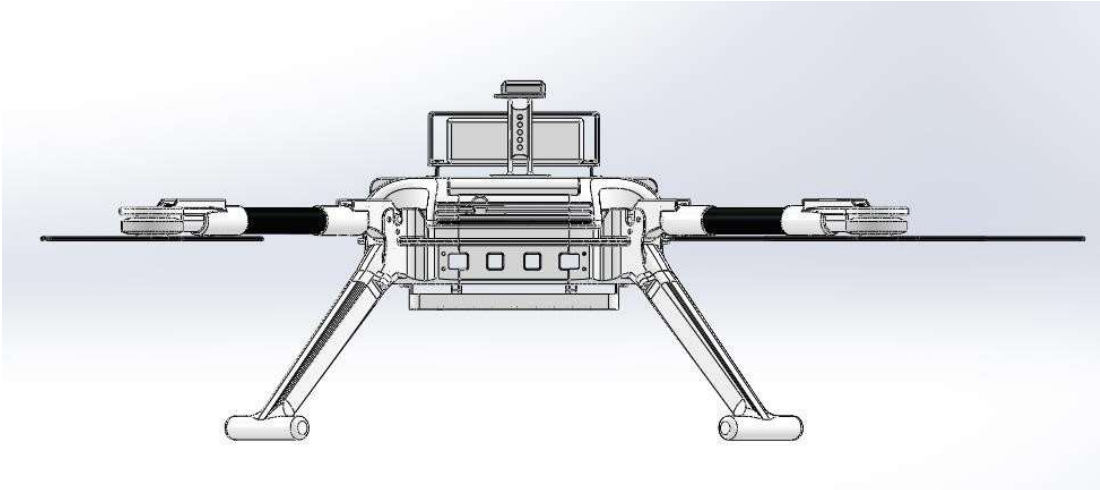


Figura 16 Diseño de quadcóptero en 3D, software Solidworks, vista lateral

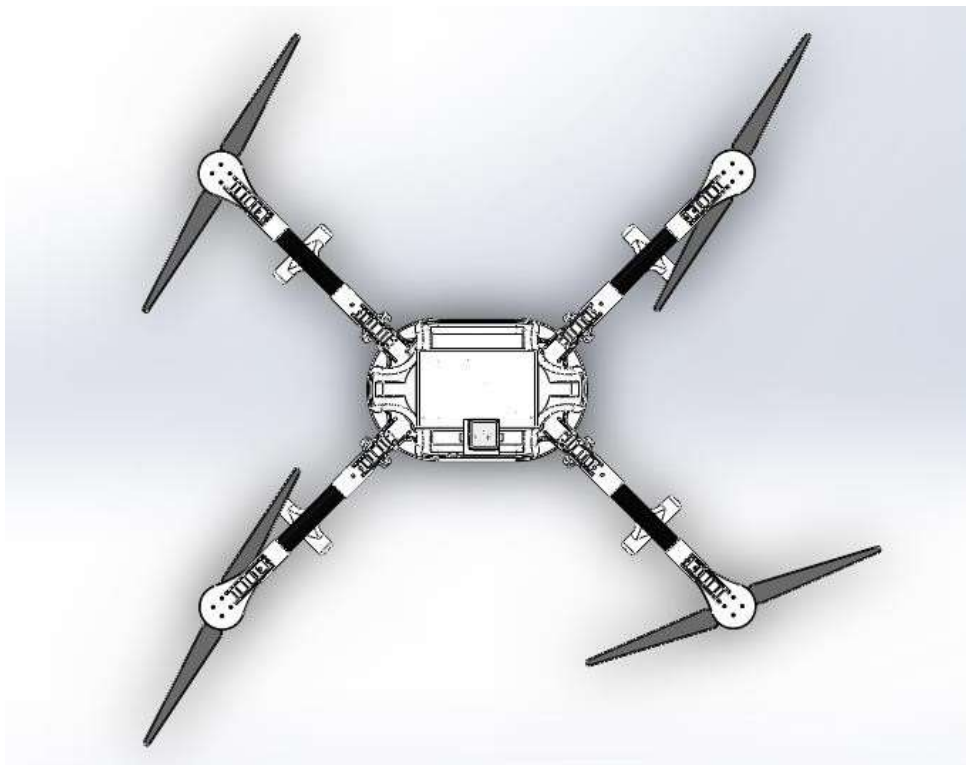


Figura 17 Diseño de quadcóptero en 3D, software Solidworks, vista superior

3.3.6. Análisis del Flujo de Aire

Asimismo, se ha realizado un análisis del flujo de aire a una velocidad de 10m/s o 35km/h aproximadamente [41]. Como se puede observar en la Fig. 18, el impacto del flujo de aire con las

superficies del quadcóptero hace que la velocidad del flujo de aire se reduzca y ocasiona puntos de presión en la superficie del quadcóptero, pero sin que esto limite las condiciones de vuelo del mismo. Existen zonas en donde la velocidad del flujo de aire es reducida de 10m/s a 0m/s. La presión máxima sobre la superficie del quadcóptero puede llegar a los 101369 Pascales o 1.034kg/cm^2 . Esta presión en la superficie del quadcóptero es la resistencia al avance que el flujo de aire ejerce sobre el vehículo y que deberá de ser contrarrestada por los rotores. El quadcóptero fue pensado para que sea lo más aerodinámico posible y de esa manera minimizar la resistencia al avance, sin embargo, existen áreas donde se observa que el contacto del flujo de aire con la superficie del quadcóptero es contra productivo durante el vuelo del vehículo.

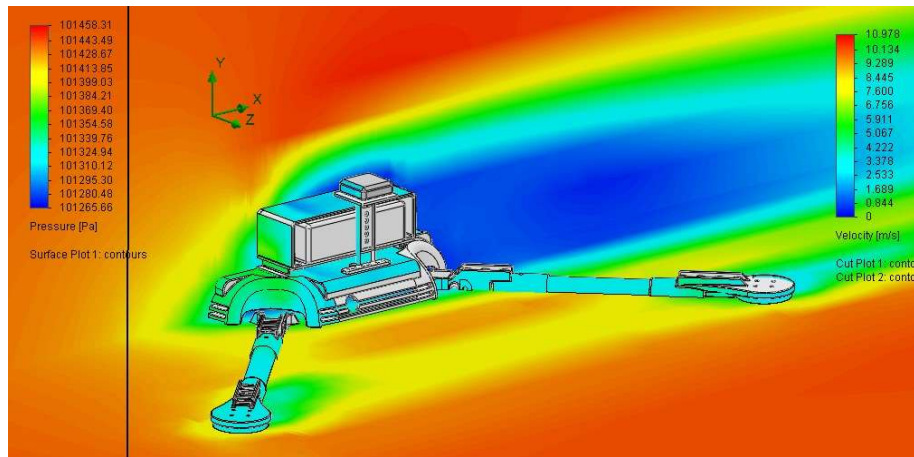


Figura 18 Velocidades del flujo de aire cercano a la superficie del quadcóptero

También existe una presión aerodinámica, conocida como resistencia de presión, que se produce por la diferencia de velocidades entre la parte frontal del quadcóptero y la parte posterior, como se ve en la Fig. 19, existe una gran diferencia en la velocidad del flujo de aire entre ambas partes del quadcóptero.

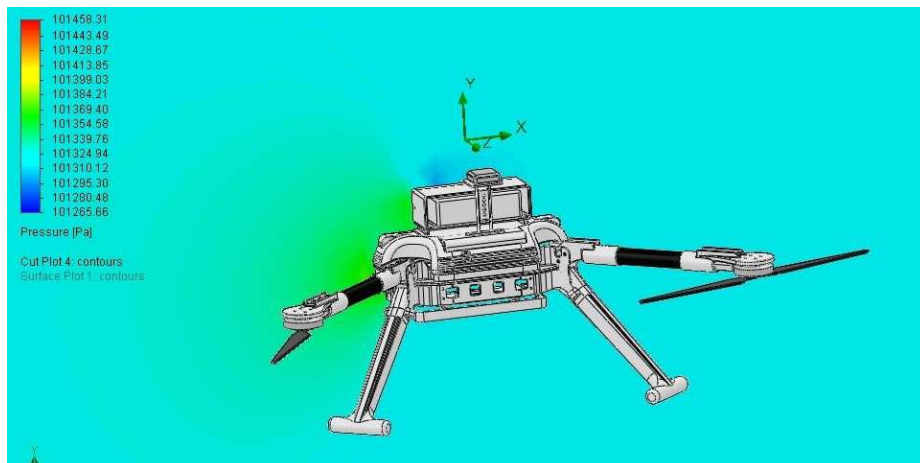


Figura 19 Diferencia de presión entre partes frontal y posterior

La resistencia de presión es ocasionada por la diferencia de velocidad del flujo de aire entre la parte frontal y posterior. La resistencia de presión en la parte frontal del quadcóptero es de 101355 Pascales o 1.034 kg/cm^2 .

Otro comportamiento del flujo de aire que es importante observar es la disipación de la vorticidad o turbulencia del aire alrededor del quadcóptero.

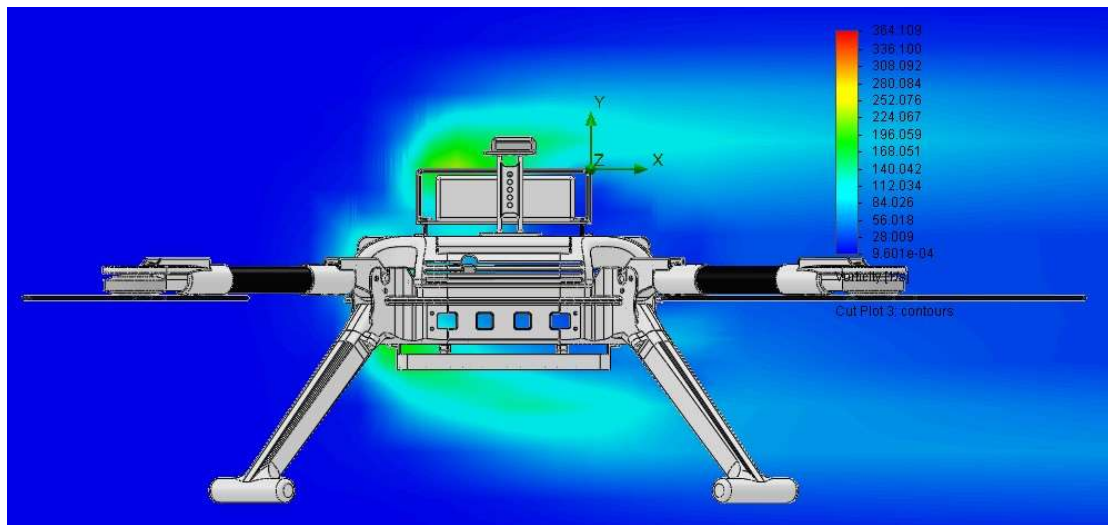


Figura 20 Turbulencia al rededor del quadcóptero

Se observa en la Fig. 20, que existen zonas donde el flujo se vuelve más turbulento, especialmente al momento que el flujo de aire hace contacto con el vehículo, lo cual podría afectar su performance durante el vuelo. Sin embargo, la vorticidad o flujo turbulento que se forma alrededor del quadcóptero no es de consideración y se puede observar que se disipa a medida que el flujo de aire avanza hacia lo largo del vehículo.

Estas simulaciones nos ayudan a entender y predecir el posible comportamiento del flujo de aire durante las operaciones del quadcóptero, sin embargo, existirán muchas condiciones de vuelo reales que son impredecibles de simular o suponer, por lo tanto, la verdadera performance de la aeronave solo es observable en operación.

3.3.7. Lista de revisión antes de cada vuelo

En el caso de las acciones preliminares, el objetivo de la revisión es garantizar que todas las partes están correctamente interconectadas, que el suministro de energía está garantizado y que los parámetros de vuelo están adecuadamente inicializados. Las medidas durante el viaje buscan determinar los consumos específicos de energía correspondientes a las respectivas acciones de la nave y también determinar el comportamiento de los parámetros de configuración PID. Finalmente, terminado el vuelo, se busca conocer con precisión el consumo total de energía y analizar el registro de incidencias de vuelo. A continuación, se detalla la secuencia de pruebas tentativas que deberá de seguir la aeronave antes, durante y después de ser puesta en operación regular:

- A. Acciones previas a cada vuelo
 - 1. Revisar conexiones eléctricas y electrónicas
 - 2. Revisar configuración inicial del quadróptero
 - 3. Revisar carga de baterías
 - 4. Configurar PIDs y otros parámetros en banco de pruebas
 - 5. Revisar conexión de GPS y posibles interferencias
 - 6. Planificación de ruta de vuelo y vuelo autónomo

- B. Medidas durante vuelo
 - 1. Potencia de despegue
 - 2. Estabilidad del quadróptero en diferentes modos de vuelo
 - 3. Calibración automática de los parámetros de configuración PID
 - 4. Verificación de un comportamiento de vuelo optimo
 - 5. Despegue y aterrizaje autónomo

- C. Acciones después de cada vuelo
 - 1. Estudio de los registros (*logs*) del quadróptero después de cada vuelo
 - 2. Ajuste de parámetros de configuración según sea necesario
 - 3. Medición del consumo de la energía gastada durante el vuelo

4. Resultados

Los resultados obtenidos son los siguientes:

- 1. Se verifico la funcionalidad del UAV a través de las pruebas de campo realizadas (despegue, desplazamiento a los objetivos, transferencia de datos y retorno)
- 2. Se comprobó que la autonomía de vuelo de la nave es 30 minutos, 22 de los cuales se consumen en el viaje de ida y vuelta
- 3. Se logró mejoras en el nivel de señal recibido en las comunicaciones a través del rediseño en el fuselaje y el empleo de polarización circular en la antena, obteniéndose mejoras en 10dB
- 4. Se verifico una adecuada trasferencia de datos lográndose una velocidad de hasta 60Mbps

El proceso de diseño se realizó siguiendo los pasos que se describen a continuación:

La identificación del problema aborda y define el problema de investigación y ve el valor de la solución que se debe construir. Esto proporciona las ventajas; lo que significa que el investigador y el público están motivados para encontrar una solución adecuada. El conocimiento y los recursos son necesarios para ejecutar esta acción. El método de investigación cuantitativa intenta demostrar que la solución que se construyó es mejor que las actuales. Cuando los objetivos de una solución son de base cualitativa, se espera que el artefacto proporcione soluciones para el problema. Los objetivos inferidos de la fase de definición del problema se logran en este paso. En el paso diseño y desarrollo, la funcionalidad y la arquitectura se definen para la solución y se crea el artefacto. El paso presentación muestra qué tan eficientemente el artefacto resuelve el problema identificado en el paso de identificación del problema.

El método para demostrar la eficacia del artefacto puede ser la experimentación, el estudio de un caso, la prueba de concepto u otras prácticas adecuadas. El paso de evaluación mide cómo la solución diseñada y construida por el artefacto maneja el problema. En esta parte se hace una comparación de los objetivos y una solución real. Las mediciones o encuestas de rendimiento, los comentarios de los clientes o las simulaciones se pueden usar como un método de evaluación. Al final de esta actividad, los investigadores pueden decidir repetir el paso 3 e intentar mejorar la solución. La comunicación es el último paso cuyo objetivo es comunicar el problema, el artefacto, su utilidad y novedad, el diseño riguroso y la eficacia a los investigadores y otra audiencia relevante.

Existen diferentes puntos de vista para la evaluación de la investigación en *design science*, por ejemplo, un acto único de demostración. Cumpliendo con la visión de los artefactos como sistemas, los criterios de evaluación se organizan en cinco dimensiones: objetivo, estructura, actividad, evolución y entorno [25].

La dimensión “consecución del objetivo” se caracteriza por los siguientes criterios:

1. eficacia es el grado con que el artefacto produce su efecto deseado
2. validez es el grado con que el artefacto funciona correctamente, es decir logra correctamente su
3. objetivo, validez abarca confiabilidad.
4. generalidad es la generalización del objetivo, es decir mientras más amplio el objetivo abordado por el artefacto, más general es el artefacto.

La dimensión “estructura de los artefactos” se evalúa por los criterios: integridad, simplicidad, claridad, estilo, homomorfismo, nivel de detalle y consistencia. El homomorfismo se refiere a la sobrecarga de la construcción, la redundancia, el exceso y el déficit. Homomorfismo es la correspondencia de un modelo (estructura) con otro modelo, o la fidelidad de un modelo a los fenómenos modelados.

La dimensión “actividad” se caracteriza por la integridad, consistencia, precisión, rendimiento y eficiencia. La integridad de la actividad equivale a la funcionalidad. La consistencia se aplica al aspecto dinámico de los artefactos (actividad) así como al aspecto estático (estructura). La precisión se establece cuando existe un acuerdo demostrado con los resultados de los experimentos existentes. Por lo tanto, pertenece al aspecto dinámico. Rendimiento (por ejemplo, velocidad de una actividad) y la eficiencia (relación entre las salidas y las entradas de la actividad) también están claramente vinculados a este aspecto dinámico.

La dimensión “evolución” se caracteriza por la solidez y la capacidad de aprendizaje. Caracterizamos la solidez como la capacidad de responder a las fluctuaciones del entorno. La capacidad de aprendizaje es la capacidad de un sistema para aprender de su experiencia y las reacciones del entorno.

La coherencia del artefacto con personas, organización y tecnología son los respectivos criterios de la dimensión “entorno”. La coherencia es el acuerdo o armonía de partes o características entre sí o en su conjunto. El sub criterio utilidad, común para la coherencia con las personas y las organizaciones, mide la calidad del artefacto en el uso práctico. La utilidad para las personas no necesariamente se materializa en utilidad para las organizaciones. Los sub criterios comprensibilidad, facilidad de uso y ética se relacionan con las personas. El sub criterio ajuste con la organización caracteriza el alineamiento del artefacto con su entorno organizacional. Dentro del criterio coherencia con la tecnología encontramos

los sub criterios aprovechamiento de las tecnologías recientes y los efectos secundarios del artefacto en su entorno.

4.1. ***Pruebas de hipótesis***

La hipótesis planteada se comprueba a partir de la verificación de las funcionalidades del UAV y las mediciones realizadas en campo como se describe a continuación:

4.1.1. **AguiLucho**

El piloto automático no es el único sistema telemático dentro de la UAV. El objetivo buscado en el proyecto TapirNet es que la nave UAV alcance la posición de un objetivo (nodo sumidero) ubicado más allá del alcance del radio control, una vez alcanzado el objetivo que encienda la radio e inicie una recolección de la data acumulada en el nodo sumidero, terminada esta tarea que vuelva al punto de partida. A este sistema completo se le ha denominado AguiLucho.

Dentro de la nave, las partes que componen el hardware telemático necesario son: una computadora y/o procesador, un piloto automático y una radio. A priori, dos plataformas fueron consideradas:

1. Pixhawk + Gumstix COM Overo Summit + radio WiFi 802.11n chipset Ralink RT 3572
2. AeroCore + Gumstix COM DuoVero Zephyr (incluye radio WiFi 802.11n Wi2Wi W2CBW0015)

Ambas plataformas son similares, ambas cuentan con microprocesadores ARM Cortex-M4 MCU para la parte de piloto automático y para la parte del software de misión, en ambos casos se emplea *computers-on-module* del fabricante Gumstix.

En la figura 21 podemos ver un diagrama de flujo del funcionamiento de AguiLucho. La computadora de propósito específico, *Computer on Module* ó COM, gobierna tanto al piloto automático como a la radio. El algoritmo del sistema es el siguiente:

1. Las coordenadas del destino serán grabadas en el piloto automático.
2. La COM recibirá la altitud y la posición como datos del piloto.
3. Cuando la nave alcance la posición del objetivo, la COM encenderá la radio y ordenará el envío de paquetes de establecimiento de enlace
4. Establecido el enlace se transfieren las imágenes desde el nodo sumidero hasta la UAV
5. La COM controla el tiempo y el estado de la batería de la UAV
6. Terminada la transferencia o alcanzado un nivel crítico de carga de batería, la COM ordena el regreso a la posición inicial

4.1.2. **DroneKit**

DroneKit, una API de código abierto escrita en Python, permite desarrollar *apps* que se ejecuten directamente en la computadora a bordo y se comuniquen con el piloto automático Pixhawk mediante un enlace de baja latencia; DroneKit también permite el desarrollo de *apps* de control remoto vía web. La API se conecta con Pixhawk a través del protocolo de comunicación MAVLink, gracias a este protocolo, la API tiene acceso a los datos de la telemetría o a los parámetros del dron a través de Bluetooth, Wi-Fi o el sistema de radio 3DR. Más aún, DroneKit puede controlar el movimiento y las operaciones de la nave. Esta API permite a los desarrolladores, entre otras funciones:

1. Desarrollar aplicaciones que mejoren el vuelo en piloto automático.
2. Proporcionar características inteligentes a un dron como la visión por ordenador, la planificación de la ruta o el modelado en 3D.
3. Permitir a un vehículo aéreo seguir objetivos mediante GPS.
4. Facilitar el seguimiento de una ruta marcada con receptores GPS.
5. Controlar cámaras y obtener información de telemetría.
6. Generación de bitácoras de datos para posterior análisis.

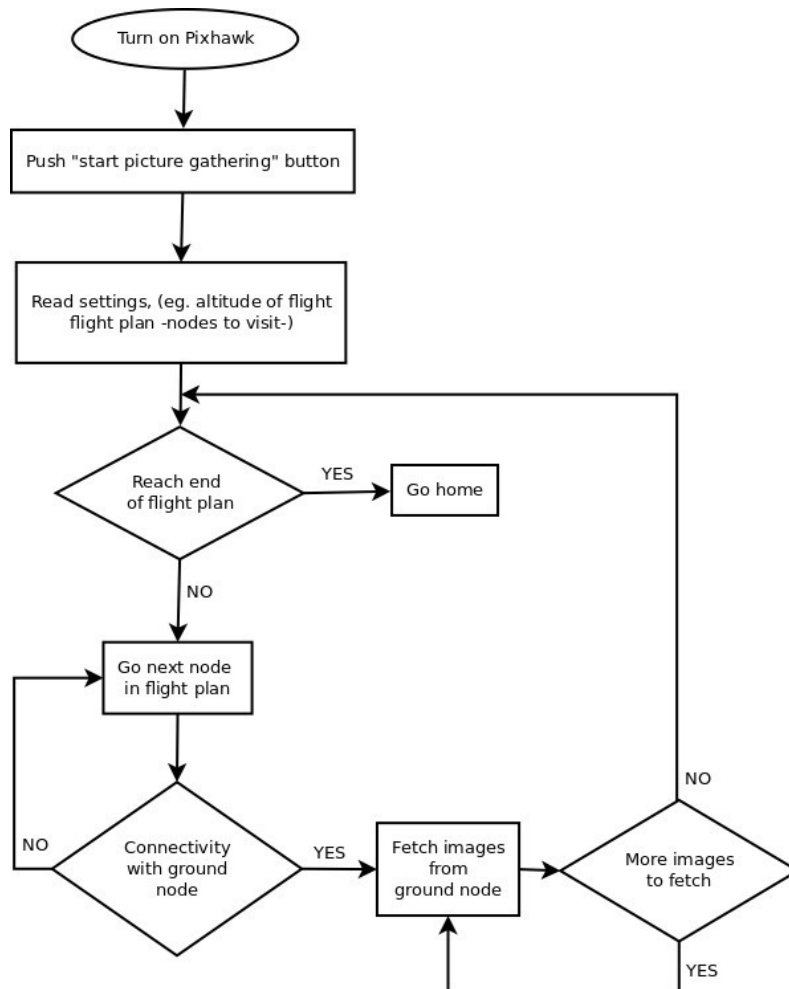


Figura 21 Diagrama de flujo del programa AguiLucho

DroneKit se puede instalar en sistemas operativos Linux, Windows y Mac OS X y funciona tanto en *desktops* y *laptops* como en *smartphones*. También se puede acceder a la API en la nube, donde se disponen del servicio **DroneShare** (mapeo de vuelos en tiempo real y compartición de datos de misiones y rutas con drones).

Una aplicación desarrollada con DroneKit en Python suele ejecutarse en una computadora abordo corriendo S.O. Linux, pero durante las pruebas de desarrollo lo más lógico y habitual es probar los prototipos en una computadora estándar conectada a un piloto automático.

4.1.3. Script Simple Go To modificado

DroneKit provee un programa tipo llamado *Simple Go To* que muestra cómo armar y lanzar un helicóptero en modo GUIADO, viajar hacia varios puntos objetivo y luego regresar a la ubicación de inicio. Utiliza las funciones *Vehicle.simple_takeoff()*, *Vehicle.simple_goto()* y *Vehicle.mode()*.

El código tiene tres secciones distintas: armado y despegue, vuelo a dos ubicaciones y regreso a casa. Para iniciar el helicóptero, primero se debe verificar que el vehículo esté habilitado para arrancar, luego configurar el modo GUIADO, armar el vehículo y ejecutar la función *Vehicle.simple_takeoff()*. Cuando el vehículo ya está en modo GUIADO, para enviarlo a un punto determinado, se invoca la función *Vehicle.simple_goto()* con las coordenadas del destino asignadas a la variable *dronekit.LocationGlobalRelative*. Para avanzar hacia el primer punto, se establece la velocidad utilizando la variable *Vehicle.airspeed*. Para volver a la posición de inicio y aterrizar, establecemos el modo RTL, el vehículo viajará a la velocidad previamente establecida.

Simple Go To no brinda información sobre la distancia que queda por recorrer y tampoco hay una función para consultar si ya se alcanzó el punto deseado (se agrega un tiempo muerto calculado a priori). Sin embargo, aunque no se muestran en el programa tipo, DroneKit sí ofrece herramientas para hacer seguimiento de la distancia que queda para alcanzar un objetivo, las cuales se han usado. Se desarrolló dicha implementación y se le llamó *blocking_goto*; con esta función no hace falta agregar tiempo muerto, sino que cuando se llama a la función *blocking_goto* se asume que se está sobre el objetivo y se puede iniciar el contacto con el nodo en tierra para hacer la transferencia.

El script ***blocking_goto*** tiene un par de diferencias con respecto al script original. La principal es el algoritmo que utiliza para calcular la distancia entre dos puntos, mientras el *script* original utiliza la forma de la distancia euclidiana, en el nuevo script se utiliza la distancia ortodrómica. La distancia euclidiana es de menor precisión, y no se aconseja para distancias mayores a 1 kilómetro. La segunda diferencia es que, aunque se cuenta con el sistema de posicionamiento global (GPS) y se conoce las coordenadas del destino, se utiliza el índice de intensidad de señal recibida por radio (RSSI) para encontrar al nodo en tierra [42]. En el futuro esta técnica puede mejorarse con *Reinforcement Learning* (RL) para permitir que la nave encuentre su trayectoria automáticamente [43].

4.2. *Presentación de resultados*

Una de las dos radios del nodo sumidero (XBee WiFi) y la COM de la UAV fueron configurados previamente para funcionar como integrantes de una red *ad hoc* TCP/IP. Una vez que la UAV se coloca sobre el nodo sumidero, la COM comprueba la conectividad a través de mensajes *pings*, luego que se establece la conectividad, la transferencia de datos es iniciada.

Para transferir archivos, se desarrolló un programa en la capa de aplicación, este programa hace uso del protocolo TCP como protocolo de transporte. En este esquema de transferencia, la computadora a bordo de la nave es el cliente y el nodo sumidero es el servidor. La UAV inicia la comunicación mediante el envío de una secuencia de bytes para iniciar la transferencia; si está listo para transferir, el nodo de tierra devuelve un mensaje OK. Luego, la lista de imágenes se envía al UAV y este finalmente empieza a cargar las imágenes. Después de que todas las imágenes se han subido, el UAV termina la conexión de transferencia. La fig. 22 muestra el flujo de comunicaciones descrito.

El módulo XBee WiFi permite la transmisión de paquetes de hasta 1400 bytes, pero debido a las limitaciones de memoria del servidor y de velocidad, el tamaño de estos paquetes es en realidad más pequeño. Cabe señalar que el nodo de tierra mantiene un registro de todas las imágenes transferidas. Con el fin de medir las velocidades de transferencia de datos alcanzables entre el UAV y el nodo de tierra, se realizaron pruebas de campo, en estas pruebas la nave realizó el procedimiento completo, la nave voló automáticamente 30 metros por encima del nodo sumidero, se mantuvo en el modo de vuelo estacionario y con línea de vista sin obstáculos; en esa posición, 5 MB de datos (equivalente a 50 imágenes de 100 Kbytes cada una) se transmitieron exitosamente en 5 minutos. Considerando todas las pruebas, se logró un promedio de velocidad de 130 Kbps.

Si hubiera un error o problema en la comunicación, el UAV no esperará indefinidamente. Se colocaron salvaguardas como 1) tiempo máximo de transferencia por imagen, y 2) tiempo máximo desde última comunicación del nodo en tierra. Si estos tiempos caducan el UAV regresa a su punto de partida.

4.3. *Análisis, interpretación y discusión de resultados*

4.3.1. **Comunicación aire tierra**

Como se mencionó en el acápite anterior, una limitación notable en el nodo sumidero era la velocidad del bus UART entre la radio y el microcontrolador que está limitada a 230.4 Kbps, debido a esta limitación de nada servía incrementar la velocidad tierra-aire si previamente no se superaba el cuello de botella antes mencionado. Ese y otros hallazgos motivaron una transformación radical de la topología y el hardware de la red de acceso para la captura de las imágenes.

Cada nueva red de acceso está compuesta por 4 nodos cámara trampa, 4 nodos servidores, 4 nodos red y 01 nodo sumidero tal como se describe en la figura 23. En cada punto, los nodos cámara trampa envían las imágenes capturadas hacia su respectivo nodo servidor, el nodo servidor procede a almacenar las imágenes en la memoria microSD que comparte con el nodo red y finalmente cada nodo red envía las imágenes hacia el nodo sumidero de manera sincronizada. Con esta configuración se pretende evadir los arboles de la selva y alcanzar una distancia aproximada de 1000 metros entre nodos, el área cubierta por la red de transmisión es de 1Km².

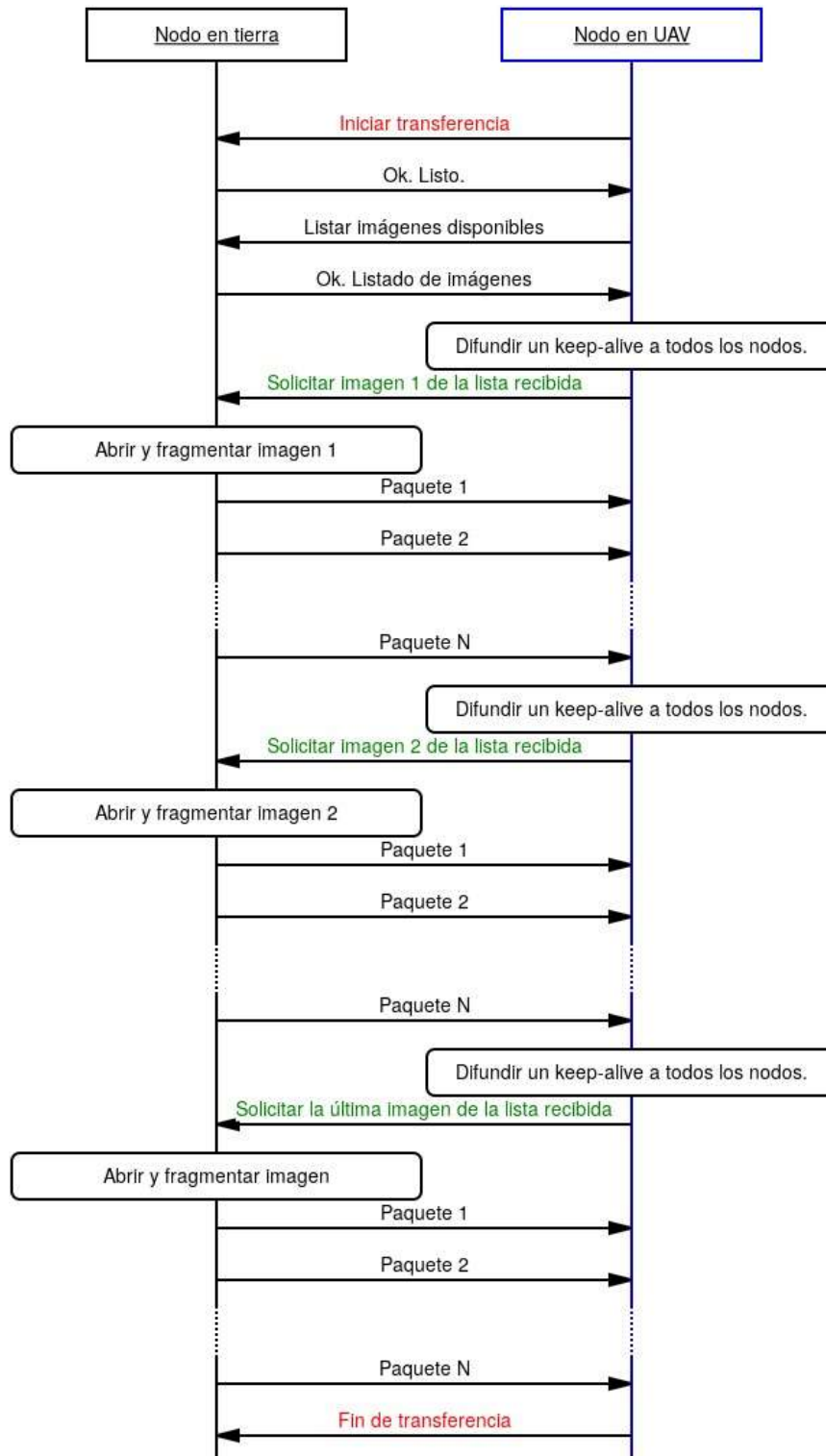


Figura 22 Intercambio de mensajes durante transferencia de paquetes

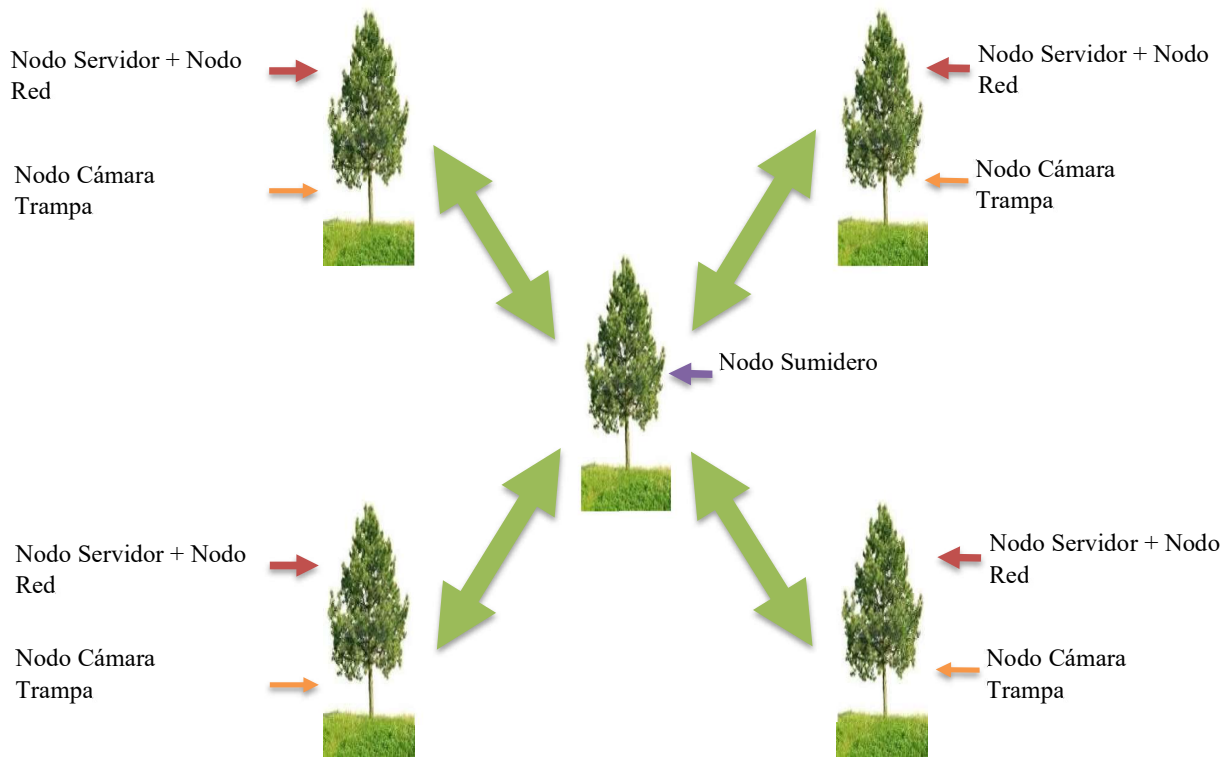


Figura 23 Topología final de la red de acceso

El nuevo nodo sumidero ya no tiene el cuello de botella que limitaba a su antecesor. El nuevo nodo sumidero se construyó con un chip ESP8266, un Real Time Clock (RTC) DS3231 RTC y una memoria SD. El chip ESP8266 contiene un microprocesador, un radio transmisor en banda 2.4 GHz y todos los componentes para modular en dicha frecuencia. La ventaja de usar este componente es que la antena se conecta directamente al integrado sin requerir componentes de radio adicionales. Al poseer un microprocesador, se pueden usar los pines que tiene para controlar otros periféricos como son: el RTC y memoria SD. Este dispositivo posee un modo de bajo consumo energético y para salir de este modo debe ser reiniciado. Para ello tiene un pin llamado XPD que es conectado con el “RESET” del dispositivo. El modo de bajo consumo solo se mantiene por 1 hora, para superar esta limitación se empleó el RTC conectado al RESET para que la radio pueda trabajar una vez por día a una hora específica. La figura 24 muestra la conexión entre cada uno de los componentes mencionados.

Y aunque el hardware telemático dentro de la nave rindió satisfactoriamente, también se realizaron cambios para garantizar la estabilidad de los enlaces. Se reemplazaron la COM Gumstix y la radio WiFi RaLink 3572 por una computadora RaspberryPi y una radio WiFi RaLink RTL8812AU, este nuevo hardware pesaba un poco más, pero a cambio ofrecía una estabilidad de software superior. Se realizaron experimentos reales con todo este nuevo hardware telemático. En las pruebas, las funciones 802.11n estuvieron habilitadas. Entre otras, las características principales que se utilizaron son la capa MIMO PHY, el enlace de canales, la agregación de tramas y el reconocimiento de bloques.

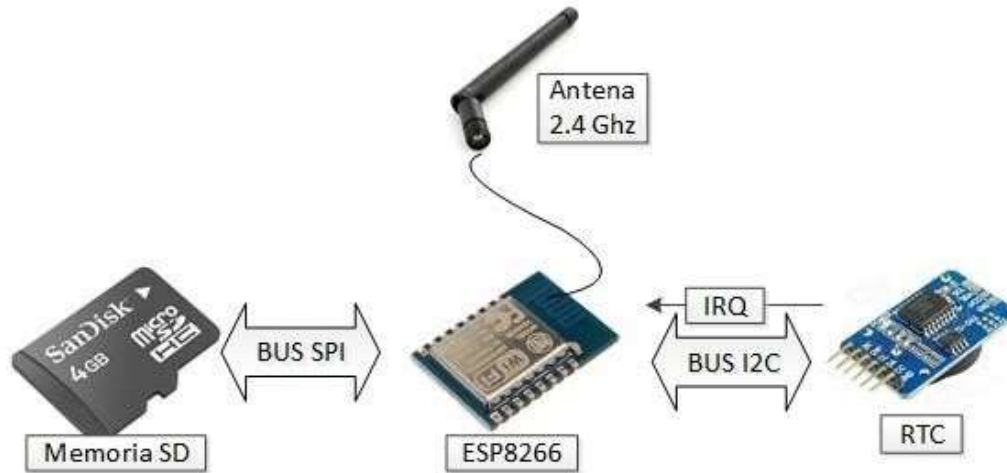


Figura 24 Diagrama de conexiones de los componentes del módulo audio

El objetivo fue evaluar la calidad de los enlaces 802.11n y evaluar el impacto de la distancia sobre el RSS y el *throughput*. Se generó tráfico UDP a diferentes velocidades usando iperf y se realizó mediciones de *throughput* de transmisiones de datos. Los resultados se muestran en las Fig. 25 y 26. En la primera figura se visualizan como *boxplots* que muestran la mediana, el 25% y 75% cuartiles y valores extremos. A distancias cortas, el *throughput* (20 Mb/s) es comparable al esperado de las implementaciones 802.11g (teóricamente hasta 54 Mb/s). Aunque este resultado es bastante superior al anterior (cuando se tenía un cuello de botella en el receptor) y más que suficiente, no puede dejar de mencionarse que:

1. es significativamente menor que el esperado de 802.11n (teóricamente hasta 300 Mb/s)
2. la mediana del *throughput* muestra una degradación creciente conforme la distancia entre nodos aumenta
3. el *throughput* muestra una gran variabilidad incluso para distancias cortas.

Estos resultados pueden explicarse por evidencia empírica disponible en la bibliografía. A través de experimentos con una velocidad de datos fija, en [16] midieron el *throughput* que se pierde cuando se usa la adaptación automática, descubrieron que simplemente fijar la velocidad al valor mínimo, MCS1, proporciona un mayor rendimiento que la tasa adaptativa de PHY.

Sin embargo, existen más razones para el bajo rendimiento obtenido, los bajos valores de RSS también fueron llamativos, como se ven en la figura 26, y dan una pista clara. RSS depende, además de la distancia, también de la transmisión por trayectos múltiples, barreras y no idealidades de antena. Las diferencias entre los patrones de transmisión omnidireccionales ideales y reales o la desalineación del ángulo de polarización pueden afectar en gran medida el valor de RSS.

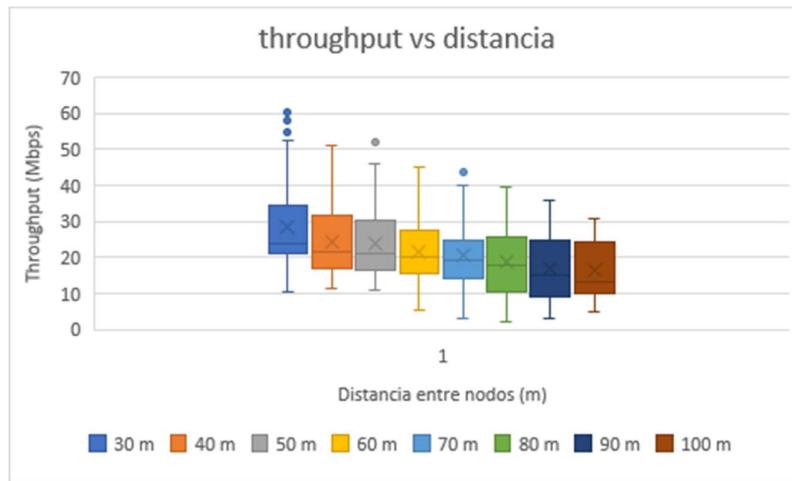


Figura 25 Variación del *throughput* con antenas de polarización vertical

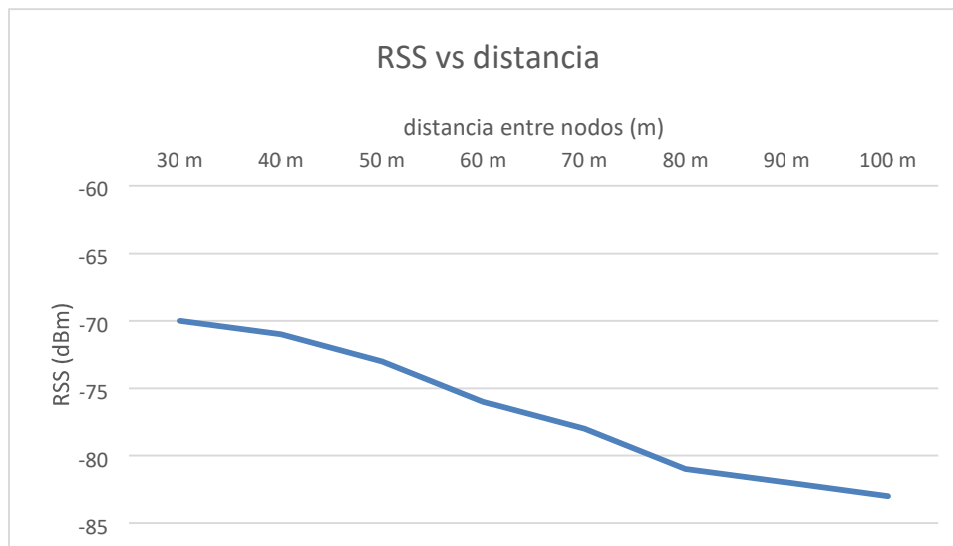


Figura 26 Variación de RSS con antenas de polarización vertical

En todos los experimentos anteriores se utilizó antenas monopolo de polarización vertical. Siguiendo el hallazgo anterior, se hizo un último juego de pruebas cambiando las antenas verticales por antenas de polarización circular. Los resultados obtenidos se muestran en la fig. 27. El incremento del *throughput* es notable, ahora los valores varían entre 60 y 30 Mbps, muy superiores a los valores anteriores que rondaban en torno a 20 Mbps.

Al igual que el *throughput*, RSS también es afectado inversamente por la distancia entre los nodos. Sin embargo, con las antenas de polarización circular se observa en la Figura 28 que RSS sigue razonablemente el modelo de propagación de espacio libre, ahora a las distancias ensayadas, la velocidad del viento, las reflexiones y difracciones introducen fluctuaciones despreciables, es decir se consiguió minimizar la dependencia de RSS del ángulo de inclinación entre las antenas de los nodos.

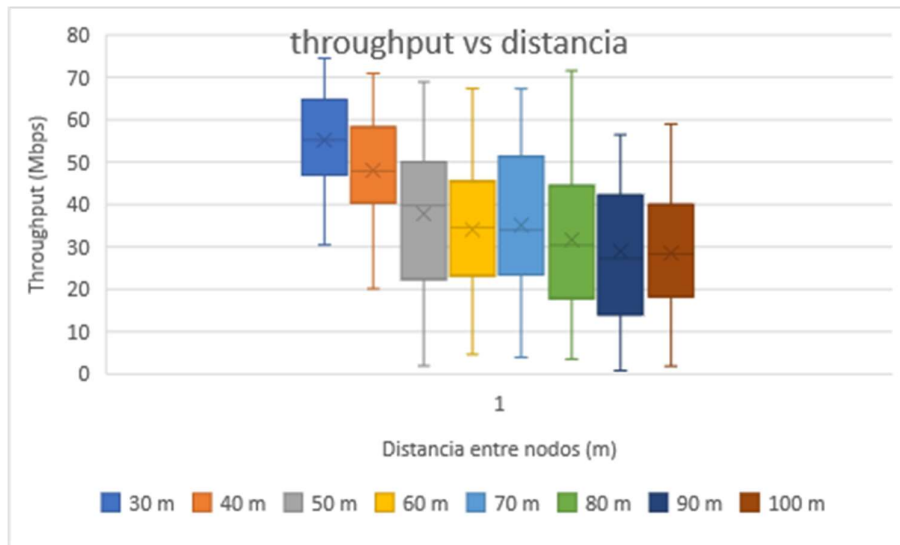


Figura 27 Variación del *throughput* con antenas de polarización circular

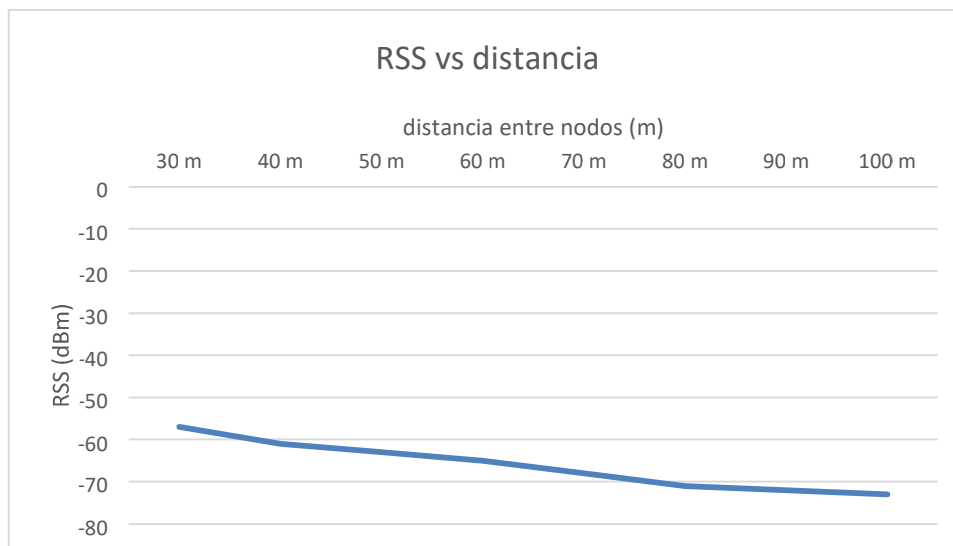


Figura 28 Variación de RSS con antenas de polarización circular

4.3.2. Fuselaje

El reemplazo de los componentes telemáticos no fue el único cambio que se hizo, también se cambió la nave, originalmente hecha de plástico ABS, por otra construida principalmente de fibra de carbono. Este material fue elegido debido a su alta relación resistencia/peso que garantiza la rigidez estructural a lo largo de todo el vuelo y un peso relativamente bajo que permite ahorrar energía. La estructura central se compone de dos placas de fibra de carbono unidas por varias columnas de aluminio, cuyo objetivo es mantener unida la mayor parte de la estructura, así como alojar las partes electrónicas, las baterías y la carga útil de la nave. Los brazos son tubos de fibra de carbono unidos al eje principal que sujetan los motores y las hélices. Cada montaje de motor tiene un marco de plástico de alta resistencia, además de una placa de fibra de carbono con orificios para el montaje de motores. Por último, el tren de aterrizaje

está constituido por dos conjuntos de tubos de fibra de carbono dispuestos en forma de T; esta estructura estabiliza la nave en operaciones de aterrizaje. Las puntas de goma en los extremos de los tubos de aterrizaje reducen el impacto ocasionado por aterrizajes ásperos.

Para obtener un tiempo de vuelo determinado considerando una carga determinada, se requiere la combinación correcta de motores, hélices y baterías. Varios parámetros se deben considerar al decidir que partes comprar y comparar sus beneficios y desventajas. Por ejemplo, las baterías LiPo con mayor cantidad de celdas y mayor capacidad pueden disminuir el tiempo de respuesta de los motores y aumentar el tiempo de vuelo; sin embargo, este tipo de baterías suelen ser más pesadas, lo que a la vez resultaría en un aumento de consumo de energía y, consecuentemente, disminuiría el tiempo de vuelo y podría anular el aumento anterior. Ciertos compromisos se deben realizar al momento de diseñar el sistema de propulsión del UAV.

Las características del UAV propuesto se muestran en la Tabla 4. La nave puede volar a 8 m/s, a esta velocidad le toma 625 segundos (11 minutos) alcanzar el objetivo, el vuelo de ida y vuelta toma el doble de tiempo es decir 22 minutos. El tiempo de vuelo mixto toma en cuenta las operaciones de despegue y vuelo estabilizado, sumando un minuto de transferencia de información al tiempo de vuelo de 22 minutos, se obtiene un tiempo total de 23 minutos, dentro del tiempo máximo de duración de la batería, pero con poco margen de error.

El conjunto seleccionado se muestra en la Tabla 4, así como el peso del UAV y el tiempo de vuelo estimado empleando la herramienta eCalc. Más adelante, se realizarán modificaciones para obtener mayores tiempos de vuelo, como disminuir el peso de la carcasa al reducir el tamaño de las partes o cambiando el conjunto motor – hélice por una combinación más eficiente.

Aunque la aeronave es de bajo costo, su pérdida no sería despreciable. Por lo tanto, aunque el UAV está listo para volar, se deben implementar mejoras para aumentar el posible tiempo de vuelo. De la misma manera, el software debe ser mejorado para incluir medidas de contingencia ante fallas.

5. Impactos

5.1. Costos de implementación de la propuesta

Los precios señalados incluyen los costos de importación e impuestos nacionales, es decir los precios corresponden a los productos debidamente internados en el país. El importador compró los productos directamente a fabricantes como Gumstix (en el caso de las partes electrónicas o a grandes distribuidores como Helipal (en el caso de todos los accesorios para la fabricación de las naves)

Componentes	Parámetro	Valor
Motor	Fabricante	Tarot
	Tipo	5008/340KV
	Peso	168 g (c/u)
Hélices	Diametro	17 pulgadas
	Paso	5 pulgadas

	Palas por hélice	2
Batería	Voltaje (Nominal)	22.2 V (6S)
	Capacidad	5000 mAh
	Cantidad	2
	Peso	1515 g (c/u)
Estructura	Peso de carcasa	1541 gramos
	Tiempo de Vuelo Mixto	22.4 minutos
	Tiempo de Vuelo Simple	21.7 minutos

Tabla 4 Componentes y parámetros de UAV, incluyendo tiempo de vuelo estimado mediante eCalc.

Item	Descripción	Qty	P. Unitario	P. Total
1	Conectores T	10	S/. 10.95	S/. 109.48
2	Extensión Macho - Hembra 30cm	6	S/. 17.08	S/. 102.47
3	Conector Y	2	S/. 19.71	S/. 39.41
4	Juego de hélices de carbono	4	S/. 139.69	S/. 558.76
5	Miscelaneos de cables y electrónica	1	S/. 233.31	S/. 233.31
6	Extensión Macho - Macho 13cm	6	S/. 6.57	S/. 39.41
7	Storm 22.2V 5000(5300)mAh 25C LiPo Battery Pack	12	S/. 525.04	S/. 6,300.51
8	Motores de 340Kv Tarot 5008	4	S/. 262.30	S/. 1,049.21
9	Controladores de Velocidad de 60°	6	S/. 127.75	S/. 766.47
10	Radio Control de 8 Canales	1	S/. 1,698.26	S/. 1,698.26
11	Batería de 22.2V 5000mAh	10	S/. 434.71	S/. 4,347.07
12	Motores de 340Kv Tarot 5008	5	S/. 215.53	S/. 1,077.64
13	Hélices de cóptero	5	S/. 116.53	S/. 582.65
14	Embedded Works EW5270UM / 802.11ac/b/g/n 2x2 MIMO / USB 3.0 Module (Realtek RTL8812AU)	1	S/. 163.29	S/. 163.29
15	SparkLAN WUBR-507N(P) / 802.11a/n/b/g 2x2 MIMO / USB Module Type-A (Ralink RT3572)	4	S/. 142.79	S/. 571.16
16	CC3200-LAUNCHXL - SimpleLink Wi-Fi CC3200 LaunchPad	4	S/. 107.19	S/. 428.76
17	ChronoDot - Ultra-precise Real Time Clock - v2.1 P	5	S/. 62.55	S/. 312.74
18	SpiroNet 5.8 GHz RHCP Patch Antenna - SMA Male	2	S/. 246.58	S/. 493.17
19	SpiroNet RP-SMA 5.8 GHz RHCP Antenna (RP-SMA Male)	2	S/. 142.93	S/. 285.86
20	Blue Beam Ultra 5.8 GHz Antenna Set RHCP - SMA Male	2	S/. 232.14	S/. 464.29
21	Overo® TidalSTORM	1	S/. 447.58	S/. 447.58

22	Overo® Tide COM	1	S/. 544.18	S/. 544.18
23	Tobi-Duo	1	S/. 254.38	S/. 254.38
24	Thumbo	1	S/. 88.55	S/. 88.55
25	Ubuntu System Card (Overo only)	1	S/. 96.60	S/. 96.60
26	5V US Power Adapter	1	S/. 32.20	S/. 32.20
27	3DR Pixhawk	1	S/. 643.97	S/. 643.97
28	PPM Encoder	1	S/. 64.40	S/. 64.40
29	Pixhawk External LED and USB	1	S/. 64.40	S/. 64.40
30	3DR Radio Set	1	S/. 322.00	S/. 322.00
31	3DR uBlox GPS with Compass Kit	1	S/. 257.57	S/. 257.57
32	Overo Summit Pack	1	S/. 946.13	S/. 946.13
33	Remote Data Collection Development Kit	1	S/. 873.07	S/. 873.07
34	Mobile Robotics Development kit	1	S/. 1,055.72	S/. 1,055.72
35	Janus interface	2	S/. 142.47	S/. 284.93
36	Angstrom system card	1	S/. 109.59	S/. 109.59
37	Quadcopter Power Distribution Board	2	S/. 54.80	S/. 109.59
38	Cables de Extension Female – Female	10	S/. 4.57	S/. 45.66
39	PixHawk Autopilot incluye: - PPM Encoder EAR499 - External USB and LED Module - Telemetry Radio 433MHz - u-blox GPS with Compass	1	S/. 1,534.19	S/. 1,534.19
40	Ultra Plug Conector Male	10	S/. 3.65	S/. 36.53
			SUBTOTAL	S/. 27,435.14
			IGV	S/. 4,938.33
			TOTAL	S/. 32,373.47

Tabla 5 Presupuesto

5.2. Cronograma de Actividades

Actividad		1												2											
		1	2	3	4	5	6	7	8	9	10	11	12	1	2	3	4	5	6	7	8	9	10	11	12
1.1	Revisión bibliográfica	X	X																						
1.2	Estudio de campo		X																						
1.3	Elaboración de plan de desarrollo tecnológico		X	X																					
1.4	Elaboración de lista de compras de laboratorio			X																					
1.5	Desarrollo de red de acceso				X	X	X	X																	
1.6	Desarrollo de sistema de ahorro de energía							X	X	X	X														

Finalmente, a pesar de todos los avances, aún no existe tecnología debidamente probada y contrastada que automatice completamente el funcionamiento (despegue, misión y aterrizaje) de un sistema de naves no tripuladas, por ello esta investigación beneficia a la sociedad incrementando el acervo de investigación, programación y sobre todo experimentación sobre esta temática específica. Un beneficio adicional es que todo el software generado está libremente disponible bajo licencias de código abierto.

Conclusiones

1. A través del desarrollo de la presente investigación que incluye la implementación del diseño planteado, habiéndose verificado las funcionalidades del UAV, realizado las mediciones de transferencia de datos, niveles de intensidad de señal RSS y comprobada la autonomía de vuelo, se comprobó la hipótesis, que un Sistema Aéreo No Tripulado puede ser utilizado como una red de transporte “store & forward”. Para tal efecto se creó y utilizó un banco de pruebas compuesto por una nave y una laptop y posteriormente se realizaron enlaces reales entre una nave y un nodo sumidero.
2. Con relación a las comunicaciones tierra aire, se realizaron tres juegos de pruebas, en los cuales se caracterizó el rendimiento de un enlace IEEE 802.11n en función de la distancia entre la nave y el nodo sumidero. Para hacerlo, se registraron los siguientes datos: parámetros UAV: posición GPS (latitud, longitud, altitud), altitud mediante un sensor de presión, velocidad de avance y nivel de batería. Parámetros de comunicación: *throughput* y RSS (intensidad de señal recibida). Dichas pruebas evidenciaron la adecuada performance de la UAV, dentro del alcance de la presente investigación.
3. En el primer conjunto de pruebas, un cuello de botella en el nodo en tierra limitó severamente el *throughput*, solo se alcanzó un promedio de velocidad, muy por debajo de la expectativa. Se hizo una revisión mayor de todo el hardware utilizado en la primera experiencia y se decidió realizar dos juegos de mediciones más, la única diferencia entre ambos sería el tipo de antena utilizado; en el segundo conjunto de pruebas, ambos nodos usaron antenas monopolo verticales, y en el tercer conjunto de pruebas, ambos nodos usaron antenas con polarización circular. En la segunda y la tercera experiencias se obtuvieron valores promedio de *throughput* de 20 y 50 Mbps respectivamente para distancias de 30 m de separación, aunque con mayor dispersión en la tercera experiencia. Con estos valores, la data semanal acumulada estimada en 5 Mb, podría transferirse en menos de 15 segundos.
4. Aguilucho es el software desarrollado para automatizar todo el proceso: despegue, vuelo autónomo hasta el nodo sumidero, extracción de la información, retorno y aterrizaje. Aguilucho está basado en software de código abierto como DroneKit, Pixhawk, NuttX y MAVLink. Aguilucho corre sobre una computadora acompañante RaspberryPi2, la cual se comunica con el piloto automático Pixhawk.
5. Considerando el peso de la nave y el consumo energético de los componentes eléctricos, el juego de baterías utilizado de 10 Ah brinda una autonomía de 30 minutos, de los cuales, 22 minutos se consumen en el viaje de ida y vuelta de la nave, luego quedan solamente 8 minutos para realizar la transferencia de la data y cuanto menos tiempo tome, mejor aún para tener suficiente tiempo de guarda. Si la data se mantiene en solo 5 Mb, se tendría un tiempo de guarda de casi 8 minutos; más aún, incluso si la data incrementase 8 veces, hasta 40 Mb, transferir ello tomaría 2 minutos, el tiempo

de guarda se reduciría a sólo seis minutos. En conclusión, un banco de baterías de 10Ah brinda suficiente energía al sistema

6. Como ya se mencionó los resultados obtenidos alcanzaron la meta establecida, sin embargo, no puede dejar de comentarse que nuestro mejor resultado, 60 Mbps, no solo está lejos del máximo teórico esperable de 802.11n, 300 Mbps, sino que también varía enormemente incluso a una corta distancia entre los nodos. La falta de suficiente diversidad espacial limita cualquier explotación de las antenas múltiples 802.11n, sin embargo, se esperaría que otras características del protocolo 802.11n, como el enlace de canales, la agregación de tramas y el reconocimiento de bloque deberían permitir lograr un mayor *throughput*. Se investigó las posibles causas de su deficiencia, incluidas el efecto Doppler, la orientación de la antena y la tasa de PHY automática. Se encontró que la funcionalidad de adaptación automática de velocidad de las radios 802.11n estándar no puede hacer frente a la alta movilidad de los UAV, es decir que MCS1, que usa solo una corriente espacial, supera a MCS8 a pesar de que este último utiliza dos corrientes espaciales. Esto es probablemente causado por el esquema de codificación espacio-tiempo de MCS1 lo que garantiza una ganancia de diversidad en el lado del receptor y, a su vez, una SNR más alta.
7. La polarización de la antena influye en la eficiencia de la comunicación. La dirección de la antena (horizontal o vertical), causa grandes pérdidas de paquetes en diferentes velocidades y posiciones. A través de las pruebas quedó claro el rendimiento superior de las antenas de polarización circular con respecto a las antenas de polarización vertical, con una mejora de 10dB en los enlaces de comunicación.
8. Con un diseño óptimo, se logró: (i) reducir la obstrucción de la señal debido al fuselaje, lográndose mejoras en la comunicación, que se tradujo en un incremento de velocidad de comunicación lográndose transmisiones de hasta 60Mbps; (ii) mejorar la relación resistencia/peso que garantiza la rigidez estructural a lo largo de todo el vuelo y un peso relativamente bajo que permite ahorrar energía, debido al empleo de fibra de carbono.

Recomendaciones

1. Se recomienda continuar con la investigación iniciada por las siguientes razones: (i) los vehículos aéreos no tripulados (UAV) ya se han convertido en una herramienta asequible y rentable para mapear rápidamente un área objetivo para muchas aplicaciones emergentes en el ámbito de la vigilancia ecológica y la conservación de la biodiversidad; (ii) ahora es un momento decisivo para evaluar el uso de drones para este tipo de aplicaciones en áreas naturales y áreas protegidas; (iii) las expediciones UAV están aumentando, pero la mayoría de ellas solo están probando su aplicabilidad. Ahora es el momento de pasar a expediciones frecuentes, ayudando en la recuperación de parámetros biofísicos importantes en los ecosistemas o distribuciones de mapeo de especies.
2. Se recomienda iniciar una línea de investigación sobre UAV en la universidad, en la medida que los UAV brindan oportunidades únicas que van desde la disponibilidad de sensores de ubicación y movimiento hasta la movilidad controlada, que no están disponibles en las redes convencionales y pueden beneficiar las comunicaciones inalámbricas, teniendo en cuenta que la compatibilidad con el sistema operativo Linux, la habilitación de bandas de frecuencia, el diseño liviano y la flexibilidad

del *driver* para configurar varios parámetros son los requisitos para la selección del hardware telemático utilizado.

3. Se recomienda profundizar los estudios sobre la mejora de las comunicaciones de datos de la UAV. Hasta ahora los experimentos se han enfocado en determinar el *throughput* y el retardo en la transmisión de datos y no han considerado los diferentes requisitos que deben tenerse en cuenta al incluir toda la pila de protocolos TCP/IP. Si bien las mediciones muestran que el *throughput* debería ser suficiente, considerar los diferentes requisitos de *QoS* puede tener un impacto significativo.
4. Pensando en la escalabilidad de esta propuesta, el enrutamiento centrado en los datos es un enfoque prometedor. Con la ayuda de la arquitectura de suscripción y publicación de algoritmos centrados en datos, es posible producir sistemas multi UAV que pueden admitir diferentes aplicaciones. En futuros trabajos se recomienda encarar los desafíos pendientes tales como: ¿Cómo elevar las prestaciones de una nave no tripulada para un fin específico? ¿Cómo establecer una comunicación TCP/IP entre un punto fijo y un punto móvil a una velocidad aún mayor? ¿Cómo conseguir una recolección directamente de cada nodo haciendo innecesarios a los sumideros?
5. Se recomienda realizar estudios del radioenlace y la propagación de las señales de comunicación de las UAV. En las redes UAV, las tecnologías de comunicación inalámbrica estándar se ven desafiadas por el movimiento frecuente de las naves, por el tipo y orientación de antena que influye en la propagación de la señal y por la obstrucción de la señal ocasionada por el fuselaje lo que da como resultado una conectividad intermitente.
6. Se recomienda en futuras investigaciones superar las limitaciones de volumen y peso que pueden soportar las naves. Actualmente el hardware de peso ligero disponible en el mercado tiene capacidades limitadas en términos de potencia de CPU, almacenamiento, y módulos de comunicación 802.11n.
7. Con respecto al fuselaje, en el futuro cercano se recomienda implementar mejoras para aumentar el tiempo de vuelo, tales como disminuir el peso de la carcasa reduciendo el tamaño de las partes o cambiando el conjunto motor-hélice por una combinación más eficiente. De la misma manera, el software debe ser mejorado para incluir medidas de contingencia ante fallas.
8. Finalmente, es recomendable realizar estudios para el uso de naves aéreas no tripuladas equipadas con cámaras multiespectrales o térmicas de alta resolución que puedan superar la obstrucción de la densa vegetación para generar imágenes que con ayuda de procesamiento digital permitan realizar conteo de individuos para evaluar el estado de los ecosistemas, el efecto de las perturbaciones o la dinámica y los cambios de las comunidades biológicas, entre otros.

Referencias bibliográficas

1. Costanza, R., de Groot, R., Sutton, P., Van der Ploeg, S., Anderson, S. J., Kubiszewski, I., & Turner, R. K. (2014). Changes in the global value of ecosystem services. *Global environmental change*, 26, 152-158.
2. Roland, K., Bart, K., Patrick, J., Chris, C., Marcus, R., Tony, F., & Sameer, T. (2009). Cameras trap as Sensor Networks for Monitoring Animal Communities. *The 34th Annual IEEE Conference on Local Computer Networks (LCN)*, 811-818.
3. Ravi Bagree, Vishwas Raj Jain, Aman Kumar, and Prabhat Ranjan. 2010. TigerCENSE: wireless image sensor network to monitor tiger movement. In *Proceedings of the 4th international conference on Real-world wireless sensor networks (REALWSN'10)*, Pedro J. Marron, Thimo Voigt, Peter Corke, and Luca Mottola (Eds.). Springer-Verlag, Berlin, Heidelberg, 13-24.
4. Camacho, L., Baquerizo, R., Palomino, J., & Zarzosa, M. (2017). Deployment of a Set of Camera Trap Networks for Wildlife Inventory in Western Amazon Rainforest. *IEEE Sensors Journal*, 17(23), 8000-8007.
5. Alnuaimi, M.; Sallabi, F.; Shuaib, K., "A survey of Wireless Multimedia Sensor Networks challenges and solutions," *2011 International Conference on Innovations in Information Technology (IIT)*, vol., no., pp.191,196, 25-27 April 2011
6. Le Zhang; "An IOT System for Environmental Monitoring and Protecting with Heterogeneous Communication Networks" 2011 6th International ICST Conference on Communications and Networking in China (CHINACOM)
7. Asadpour, Mahdi, Domenico Giustiniano, and Karin Anna Hummel. "From ground to aerial communication: dissecting WLAN 802.11n for the drones." *Proceedings of the 8th ACM international workshop on Wireless network testbeds, experimental evaluation & characterization*. ACM, 2013.
8. Nasri, M.; Helali, A.; Sghaier, H.; Maaref, H., "Energy conservation for image transmission over wireless sensor networks," *Microelectronics (ICM)*, 2011 International Conference on , vol., no., pp.1,6, 19-22 Dec. 2011
9. Cai, Guowei, Jorge Dias, and Lakmal Seneviratne. "A Survey of Small-Scale Unmanned Aerial Vehicles: Recent Advances and Future Development Trends." *Unmanned Systems* 2.02 (2014): 175-199.
10. Asadpour, Mahdi, et al. "Characterizing 802.11n aerial communication." *Proceedings of the second ACM MobiHoc workshop on Airborne networks and communications*. ACM, 2013.
11. S. Crandall and H. Jasani. ProMix: Linux Promiscuous Wireless Packet Analysis. In *Proceedings of the 7th International Conference on Wireless Communications, Networking and Mobile Computing (WiCOM 2011)*, pages 1–4, Sep 2011.
12. M. Vipin and S. Srikanth. Analysis of open source drivers for IEEE 802.11 WLANs. In *International Conference on Wireless Communication and Sensor Computing (ICWCSC 2010)*, pages 1–5, Jan 2010.
13. Haiyang Chao; Cao, Yongcan; Yang-Quan Chen, "Autopilots for Small Fixed-Wing Unmanned Air Vehicles: A Survey," *International Conference on Mechatronics and Automation, 2007. ICMA 2007*. vol., no., pp.3144,3149, 5-8 Aug. 2007 doi: 10.1109/ICMA.2007.4304064
14. Lorenz Meier, Petri Tanskanen, Lionel Heng, Gim Hee Lee, Friedrich Fraundorfer, and Marc Pollefeys. Pixhawk: A micro aerial vehicle design for autonomous flight using onboard computer vision. *Autonomous Robots (AURO)*, 2012.

15. Lorenz Meier, Petri Tanskanen, Friedrich Fraundorfer, and Marc Pollefeys. PIXHAWK: A system for autonomous flight using onboard computer vision, Robotics and Automation (ICRA), 2011 IEEE International Conference on , vol., no., pp.2992-2997, 9-13 May 2011
16. Asadpour, Mahdi, et al. "Micro aerial vehicle networks: An experimental analysis of challenges and opportunities." *Communications Magazine, IEEE* 52.7 (2014): 141-149.
17. Hevner, A., & Chatterjee, S. (2010). Design science research in information systems. In *Design research in information systems* (pp. 9-22). Springer US.
18. Caballero, Luis Camacho, et al. "On the design of an UAV-based store and forward transport network for wildlife inventory in the western Amazon rainforest." *2017 IEEE XXIV International Conference on Electronics, Electrical Engineering and Computing (INTERCON)*. IEEE, 2017.
19. Frew, Eric W., and Timothy X. Brown. "Airborne communication networks for small unmanned aircraft systems." *Proceedings of the IEEE* 96.12 (2008).
20. Brown, Timothy X., et al. "Test bed for a wireless network on small UAVs." *Proc. AIAA 3rd "Unmanned Unlimited" Technical Conference, Chicago, IL*. 2004.
21. Andre, Torsten, et al. "Application-driven design of aerial communication networks." *Communications Magazine, IEEE* 52.5 (2014): 129-137.
22. Ebeid, E., Skriver, M., & Jin, J. (2017, August). A Survey on Open-Source Flight Control Platforms of Unmanned Aerial Vehicle. In *Digital System Design (DSD), 2017 Euromicro Conference on* (pp. 396-402). IEEE.
23. Gati, B., "Open source autopilot for academic research - The Paparazzi system," *American Control Conference (ACC), 2013*, vol., no., pp.1478, 1481, 17-19 June 2013.
24. Peffers, K., Tuunanen, T., Rothenberger, M. A., & Chatterjee, S. (2007). A design science research methodology for information systems research. *Journal of management information systems*, 24(3), 45-77.
25. Gregor, S., & Hevner, A. R. (2013). Positioning and presenting design science research for maximum impact. *MIS quarterly*, 37(2).
26. Hague, D., Kung, H. T., & Suter, B. (2006, October). Field experimentation of cots-based UAV networking. In *MILCOM 2006-2006 IEEE Military Communications conference* (pp. 1-7). IEEE.
27. Cai, G., Peng, K., Chen, B. M., & Lee, T. H. (2005, June). Design and assembling of a UAV helicopter system. In *2005 International Conference on Control and Automation* (Vol. 2, pp. 697702). IEEE.
28. Cai, G., Chen, B. M., Peng, K., Dong, M., & Lee, T. H. (2006, June). Modeling and control system design for a UAV helicopter. In *2006 14th Mediterranean Conference on Control and Automation* (pp. 1-6). IEEE.
29. Jawhar, I., Mohamed, N., Al-Jaroodi, J., Agrawal, D. P., & Zhang, S. (2017). Communication and networking of UAV-based systems: Classification and associated architectures. *Journal of Network and Computer Applications*, 84, 93-108.
30. Sahingoz, O. K. (2014). Networking models in flying ad-hoc networks (FANETs): Concepts and challenges. *Journal of Intelligent & Robotic Systems*, 74(1-2), 513-527.
31. Chen-Mou Cheng, Pai-Hsiang Hsiao, H.T. Kung, and D. Vlah. Performance of 802.11a Wireless Links from UAV to Ground Nodes with Various Antenna Orientations. In *Proceedings of the 15th International Conference on Computer Communications and Networks (ICCCN 2006)*, pages 303–308, Oct 2006.

32. Sahingoz, O. K. (2013, May). Mobile networking with UAVs: Opportunities and challenges. In *2013 International Conference on Unmanned Aircraft Systems (ICUAS)* (pp. 933-941). IEEE.
33. Brown, T., Argrow, B., Dixon, C., Doshi, S., Thekkekkunnel, R. G., & Henkel, D. (2004, September). Ad hoc uav ground network (augnet). In *AIAA 3rd "Unmanned Unlimited" Technical Conference, Workshop and Exhibit* (p. 6321).
34. Yanmaz, Robert Kuschnig, and Christian Bettstetter. Channel measurements over 802.11a-based UAV-to-ground links. In *Proceedings of the 2010 (Wi-UAV Workshop)*, pages 1–5, Dec 2011.
35. Bacco, M., Ferro, E., & Gotta, A. (2014, November). UAVs in WSNs for agricultural applications: An analysis of the two-ray radio propagation model. In *SENSORS, 2014 IEEE* (pp. 130-133). IEEE.
36. Von Alan, R. H., March, S. T., Park, J., & Ram, S. (2004). Design science in information systems research. *MIS quarterly*, 28(1), 75-105.
37. Ebeid, E., Skriver, M., Terkildsen, K. H., Jensen, K., & Schultz, U. P. (2018). A survey of OpenSource UAV flight controllers and flight simulators. *Microprocessors and Microsystems*, 61, 1120.
38. Çetinsoy, E., Dikyar, S., Hançer, C., Oner, K. T., Sirimoglu, E., Unel, M., & Aksit, M. F. (2012). Design and construction of a novel quad tilt-wing UAV. *Mechatronics*, 22(6), 723-745.
39. Ping, J. T. K., Ling, A. E., Quan, T. J., & Dat, C. Y. (2012, October). Generic unmanned aerial vehicle (UAV) for civilian application-A feasibility assessment and market survey on civilian application for aerial imaging. In *2012 IEEE Conference on Sustainable Utilization and Development in Engineering and Technology (STUDENT)* (pp. 289-294). IEEE.
40. Yoon, M. (2015). On driving signal of electronic speed controller for small multi-rotor helicopter. *International Journal of Engineering Research and Technology*, 4(11), 456-459.
41. McClanahan, C., Bolam, R. C., Vagapov, Y., & Anuchin, A. (2019, March). Analysis of the Effects on the Pitching, Rolling and Yawing Rate of a V-tail Configured Quadcopter. In *2019 26th International Workshop on Electric Drives: Improvement in Efficiency of Electric Drives (IWED)* (pp. 1-7). IEEE.
42. Y. Sun, X. Wen, Z. Lu, T. Lei and S. Jiang, "Localization of WiFi Devices Using Unmanned Aerial Vehicles in Search and Rescue," *2018 IEEE/CIC International Conference on Communications in China (ICCC Workshops)*, Beijing, China, 2018, pp. 147-152, doi: 10.1109/ICCCChinaW.2018.8674518.
43. D. Ebrahimi, S. Sharafeddine, P. Ho and C. Assi, "Autonomous UAV Trajectory for Localizing Ground Objects: A Reinforcement Learning Approach," in *IEEE Transactions on Mobile Computing*, doi: 10.1109/TMC.2020.2966989.
44. Rojas, Michel Zarzosa, et al. "Development of a dedicated controller for a 5 MP CMOS sensor applied to a wireless image sensor network for large mammals monitoring." *2016 IEEE ANDESCON*. IEEE, 2016.
45. Camacho, Luis, et al. "Design and Deployment of a Multimedia Wireless Sensor Network for Wildlife Inventory in Western Amazon Rainforest." *UNESCO Chair in Technologies for Development from Innovation to School Impact* (2016): 2-4.

ANEXO 1 Script blocking_goto

```
"""
simple_goto.py: GUIDED mode "simple goto" example (Copter Only)

The example demonstrates how to arm and takeoff in Copter and how
to navigate to points using Vehicle.commands.goto.

Full documentation is provided at
http://python.dronekit.io/examples/simple_goto.html
"""
import
time
from droneapi.lib import VehicleMode, Location
from pymavlink import mavutil

api = local_connect() vehicle
= api.get_vehicles()[0]
def
probe_link(ip_address):
    import errno import subprocess try:
if subprocess.call(["fping", ip_address]) != 0:
    return False except
OSError, ex: if ex.errno ==
errno.ENOENT:
print("install fping") return
True

# base on goto example in
# dronekit-python/docs/_build/singlehtml/index.html#example-
guidedmode-setting-speed-yaw def blocking_goto(target_location):
    """
    Moves the vehicle to a position dNorth metres North and dEast
    metres East of the current position.

    The method takes a function pointer argument with a single
    `droneapi.lib.Location` parameter for
    the target position. This allows it to be called with different
    position-setting commands.
    By default it uses the standard method:
    droneapi.lib.Vehicle.commands.goto().

    The method reports the distance to target every two seconds.
    """
    current_location = vehicle.location
    target_distance = distance_to_point(current_location,
target_location)
    vehicle.commands.goto(target_location)
    vehicle.flush()
    while not api.exit and vehicle.mode.name == "GUIDED": #Stop
action if we are no longer in guided mode.
        remaining_distance = distance_to_point(vehicle.location,
target_location) * 1000
        print "Distance to target: ", remaining_distance if
remaining_distance <= 10# target_distance * 0.01 : #Just below
target, in case of undershoot. print "Reached target"
        break; time.sleep(2)
    def
in_radians(n):
```

```

        return n * math.pi() / 180.0
    def distance_to_point(locationA,
locationB):
        """ Returns the distance in KM between locationA and locationB.
The computation is based on the Great Circle distance
https://en.wikipedia.org/wiki/Great-circle\_distance
        """
        lat1 = in_radians(locationA.latitude)
lon1 = in_radians(locationA.longitude)
lat2 = in_radians(locationB.latitude)
lon2 = in_radians(locationB.longitude)
dlat = lat2 - lat1
        # Law of Cosines for Spherical Trigonometry
dlon = lon2 - lon1
        a = math.pow(math.sin(dlat / 2.0), 2) + (math.cos(lat1)
* math.cos(lat2) * math.pow(math.sin(dlon / 2.0), 2))
rad = 2.0 * math.asin(math.sqrt(a)) # return
EARTH_MEAN_RADIUS_KM = 6371.0088      return 6371.0088 * rad
    def
arm_and_takeoff(aTargetAltitude):
        """
        Arms vehicle and fly to aTargetAltitude.
        """
        print "Basic pre-arm checks"
        # Don't let the user try to fly autopilot is booting
if vehicle.mode.name == "INITIALISING":
        print "Waiting for vehicle to
initialise"          time.sleep(1)          while
vehicle.gps_0.fix_type < 2:
        print "Waiting for GPS...:", vehicle.gps_0.fix_type
time.sleep(1)

        print "Arming motors"
        # Copter should arm in GUIDED mode
vehicle.mode      = VehicleMode("GUIDED")
vehicle.armed     = True      vehicle.flush()
        while not vehicle.armed and not
api.exit:
        print " Waiting for arming..."
time.sleep(1)

        print "Taking off!"
        vehicle.commands.takeoff(aTargetAltitude) # Take off to target
altitude      vehicle.flush()

        # Wait until the vehicle reaches a safe height before processing
the goto (otherwise the command
        # after Vehicle.commands.takeoff will execute immediately).
while not api.exit:
        print " Altitude: ", vehicle.location.alt
        if vehicle.location.alt>=aTargetAltitude*0.95: #Just below
target, in case of undershoot.
        print "Reached target
altitude"          break;
time.sleep(1)

def main(altitude=20, # altitude in meters
        max_transfer_time=30, # max_transfer_time in seconds, per
image          ):
    arm_and_takeoff(20)

```

```

    print "Going to first point..."
    point1 = Location(-35.361354, 149.165218, 20, is_relative=True)
    blocking_goto(point1)

    print "Going to second point..."
    point2 = Location(-35.363244, 149.168801, 20, is_relative=True)
    blocking_goto(point2)
    has_connection = False
    for _ in range(5):
        if probe_link("10.100.25.1"):
            has_connection = True
            break;
    if (has_connection):
        images_transfer.start(MAX_TRANSFER_TIME)
        print "Returning to Launch"
    vehicle.mode = VehicleMode("RTL")
    vehicle.flush()
    if __name__ ==
"__main__":
        import sys
        print "Recordar!!!: los puntos aun estan fijos"
    print "crear un archivo con el plan de vuelo"
    print "y leer ese archivo inicialmente"
    sys.exit(main())

```

ANEXO 2 Script images_transfer

```
#!/usr/bin/python
import time
import socket

DST_ADDR = '192.168.1.1'
DST_PORT = 9750

# prepare and send a tcp packet #
return False if all data was not sent.
def send_packet(sock, data):
    sock.send(data)
    def get_image(sock,
image_id):
    assert image_id > 65535, "image_id is supposed to be 2-bytes
only"
    cmmd = "\xff\x3f\x15" # SEND_IMAGE_MSG
    cmmd + chr((image_id >> 8) & 0xff)
    cmmd + chr(image_id & 0xff)

    send_packet(sock, outbuf)
    inbuf = sock.recv(20) # here we should get fileinfo : timestamp
if inbuf[0:3] == "\xff\x3e\x17" and len(inbuf) >= 7:
    timestamp = (inbuf[4] << 24) + (inbuf[5] << 16) + (inbuf[6]
<< 8) + inbuf[7]
while True:
    inbuf = sock.recv(1024)
    if inbuf[0:3] == '\xff\x3e\x16': # marks end-of-
file
        print "End-Of-File"
        break;
    else:
        print "Sequence: " + str((ord(inbuf[0])
<< 8) + ord(inbuf[1]))
    def
start(max_transfer_time):
    sock = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
    sock.connect((DST_ADDR, DST_PORT))

    start_time = time.time()
    outbuf = "\xff\x3f\x09" #
START_XFER_MSG
    send_packet(sock, outbuf)
    sock.settimeout(3)
    try:
        inbuf = sock.recv(3)
    if (inbuf == "\xff\x3e\x17"):
        print "Ok, ready for transfer"
    outbuf = "\xff\x3f\x14" # SEND_IMAGE_IDS_MSG
    send_packet(sock, outbuf)
        inbuf = sock.recv(1024) # List of ids (images)
    print inbuf
        list_images = []
    for elem in inbuf.split(","):
        if "-" in elem:
            sta, sto = elem.split("-")
            list_images.extend(xrange(int(sta), int(sto) + 1))
        else:
            list_images.append(int(elem))
        for image_id in
list_images:
    get_image(sock, image_id)
```

```

        if (time.time() - start_time) >
max_transfer_time          break          else:
        print "Not what I expected"
except socket.timeout:
    print "Nothing received. Closing connection"
if __name__ ==
'__main__':
    import sys
    sys.exit(main())

```