Proyecto Fin de Grado GIA

Aplicación para la toma de datos de contexto para un UAV basado en Raspberry y ArduPilot

Autor: José Ramos Gálvez
Tutor: Daniel Gutiérrez

Dept. de Ingeniería Electrónica
Escuela Técnica Superior de Ingeniería
Universidad de Sevilla

Sevilla, 2017

Proyecto Fin de Grado
GIA

Aplicación para la toma de datos de contexto para un UAV basado en Raspberry y ArduPilot

Autor:
José Ramos Gálvez

Tutor:
Daniel Gutiérrez
Profesor titular

Dept. de Ingeniería Electrónica
Escuela Técnica Superior de Ingeniería
Universidad de Sevilla

Sevilla, 2017

Proyecto Fin de Carrera: Aplicación para la toma de datos de contexto para un UAV basado en Raspberry y ArduPilot

Autor: José Ramos Gálvez

Tutor: Daniel Gutiérrez

El tribunal nombrado para juzgar el Proyecto arriba indicado, compuesto por los siguientes miembros:

Presidente:

Vocales:

Secretario:

Acuerdan otorgarle la calificación de:

Sevilla, 2017

El Secretario del Tribunal

A mis profesores y compañeros

A mis amigos

A mis padres

Agradecimientos

En primer lugar, me gustaría agradecer a Sergio Vigorra y a Jesús Sánchez, por iniciarme en el fantástico y apasionante mundo de los UAVs y por dejarme formar parte de su grupo de talleres y así poder llevar a cabo este trabajo.

Gracias a todos aquellos que, en algún momento de estos años de carrera, se han prestado de forma desinteresada a echarme una mano para resolver cualquier duda, para acudir a tutorías, noches de estudio sin dormir… Entre todos los compañeros de la Universidad que he conocido y que me han acompañado en estos años, me gustaría destacar a los amigos que me llevo, pero no creo que sea necesario plasmarlo aquí, ellos sabrán sentirse identificados, ya que más que compañeros de pupitre no hay muchos.

Me gustaría hacer una especial mención a Daniel Gutierrez, tutor de este trabajo, ya que, sin su inestimable ayuda y paciencia, no se podría haber llevado a cabo la finalización del mismo.

Por último, me gustaría agradecer a mi familia y a una amiga muy especial, todos los ánimos que me han profesado y en la estima que me tienen, que hacen que tenga la motivación suficiente para enfrentarme ante cualquier problema y seguir progresando como ingeniero y como persona.

Gracias.

José Ramos Gálvez

Sevilla, 2017

Resumen

En esta memoria se recogen una serie de instrucciones detalladas, acerca de la creación de una aplicación, en el lenguaje *Python*, que tiene como principal objetivo gobernar un Quadrotor para que, durante un pilotaje manual, realice de forma simultánea los siguientes procesos:

-Captura de Imágenes.

-Toma de datos GPS.

Gracias a que se realizan dichos procesos al mismo tiempo, a cada fotografía se le puede asignar una coordenada GPS y por tanto esta aplicación posibilita la creación de una base de datos a tiempo real.

Por otro lado, también se llevó a cabo el montaje y la puesta a punto de un Quadrotor para poder realizar las pruebas en vuelo de la aplicación.

Una parte fundamental del proyecto fue el ensamblado y conexionado de los distintos elementos hardware adicionales al Quadrotor, y de esta forma poder llevar a cabo el trabajo y cumplimentar todos los objetivos propuestos.

Abstract

In this memory, we collect a series of detailed instructions about the creation of an application in Python language, whose main objective is to govern a Quadrotor so that, during manual piloting, it simultaneously performs the following processes:

-Capture of Images.

- GPS data socket.

Thanks to these processes, being carried out at the same time, each photograph can be assigned a GPS coordinate and therefore this application enables the creation of a database in real time.

On the other hand, the assembly and set-up of a Quadrotor was also done, in order to carry out in-flight tests of the application.

A fundamental part of the project was the assembly and connection of the various additional hardware elements to the Quadrotor, in order to develop the work and fulfill all the proposed objectives.

Índice

1 OBJETIVOS Y MOTIVACIÓN DEL PROYECTO

"Scientists discover the world that exists, engineers create the world that never was"

- Theodore Von Karman (1947)-

Los matemáticos o físicos se caracterizan por estudiar todos los temas de la ciencia, creando nuevas teorías y/o ecuaciones que explican el comportamiento de la naturaleza, es decir algo que ya estaba delante de nosotros, únicamente que no teníamos una ley por la cual demostrásemos dicho comportamiento. En cambio, un ingeniero, gracias a una serie de conocimientos aprendidos, crea soluciones a raíz de tener que solventar un problema o a causa de querer satisfacer una necesidad.

Como futuro ingeniero, sé que todo tipo de idea que llegue a mi mente puede, con esfuerzo, hacerse realidad. Esto provoca en mí una gran motivación ya que al realizar un trabajo de desarrollo e investigación como es un trabajo de fin de grado, puede tener una utilidad futura y puede contribuir al desarrollo científico y al avance tecnológico.

A la hora de desarrollar este proyecto, se ha querido conjugar la eficacia y eficiencia de los resultados con la seguridad del hombre, siendo estas las motivaciones primordiales que impulsan este trabajo.

Una de los usos más comunes de los UAVs, tal y como se comentará mas adelante, es la fotografía. Este campo es muy amplio y tiene una gran variedad de aplicaciones, puede abarcar desde el análisis y observación del estado de una plantación para actuar ante posibles plagas, hasta el espionaje de una base enemiga con objeto de preparar una estrategia de ataque. Podemos ver como en dichas aplicaciones, está implícito lo citado antes, la eficacia, ya que desplegando un quadrotor de alta tecnología se ahorra en tiempo y dinero al no tener a un operario manejando maquinaria, y la seguridad del hombre, porque en el segundo caso, si el UAV fuera derribado, solo sería un gasto a correr y no una vida perdida.

A continuación, se describen los objetivos fundamentales del trabajo:

- Ensamblaje y puesta a punto de un Quadrotor con placa controladora tipo APM (ArduPilotMega 2.7).
- Creación de una aplicación para que el Quadrotor realizase las siguientes labores:
    - Toma de fotografías mientras que estuviese bajo ciertos requisitos.
    - Recogida de datos GPS del propio Quadrotor.
    - Volcado de dicha información en una base de datos a tiempo real.
- Control remoto de Quadrotor a través de una Raspberry Pi.
- Prueba en vuelo del Quadrotor y de la aplicación, en una zona alejada del núcleo urbano.

2 INTRODUCCIÓN

En este capítulo se va a llevar a cabo una descripción del contexto en el que se encuentra enmarcado el proyecto, así como la metodología empleada para poder lograr los objetivos que se describieron en el capítulo anterior.

2.1 Contexto

Es notable el enorme avance tecnológico que se está produciendo en los últimos años en las diferentes ramas de la ingeniería, entre los que se encuentra el "boom" en los vehículos aéreos no tripulados o *Unmanned Aerial Vehicles (UAV)*. Debido a este gran desarrollo, es de nuestra competencia investigar acerca de nuevas posibles aplicaciones o acerca de cómo mejorar el estado presente del arte para así poder aprovecharnos de una forma óptima del nicho de mercado emergente ya que tal y como se dice en [1] "Se estima que en 5 años el mercado de los drones supondrá el 10% de la facturación del sector aeronáutico en Europa".

Es cierto que se nos presentan limitaciones a la hora de llevar a cabo este tipo de proyectos debido a la normativa actual vigente, pero el avance y desarrollo técnico requiere una adaptación legislativa, por lo que una necesidad genera un desarrollo, un desarrollo genera un producto y un producto novedoso precisa de una regulación adaptada (Ilustración 2-1) pero de todo esto se hablará en detalle en capítulo 7.

Ilustración 2-1 Evolución de la Regulación

2.2 Metodología

A continuación, se describe la metodología empleada para realizar el proyecto:

- La documentación técnica, que consistió en una descripción empezando por el estado del arte de los *UAVs* y de sus aplicaciones más usuales para acabar detallando los materiales y del software empleado.
- El montaje del *quadrotor*, el conexionado entre todos los dispositivos empleados y la puesta a punto de los mismos.
- El desarrollo de una aplicación para la toma de fotos y recogida de datos para a continuación almacenarlas en una base de datos.
- Por último, se describen los resultados obtenidos, nombrando las limitaciones actuales y proponiendo mejoras para un desarrollo futuro.

3 ESTADO DEL ARTE

En este capítulo se va a proceder en primer lugar a realizar una descripción detallada de la evolución histórica que han sufrido los vehículos aéreos no tripulados, su clasificación, así como las aplicaciones más comúnmente empleadas. El final del capítulo está centrado en el objetivo del trabajo, una descripción acerca del estado actual del uso de los drones en la fotografía y la toma de datos.

3.1 Evolución Histórica

La idea de llevar a cabo el cumplimiento de un objetivo sin la exposición directa del ser humano, ha sido desde la antigüedad un hito a conseguir. La motivación que ha conseguido un desarrollo tan abrumador ha sido principalmente, como se verá a continuación, la bélica.

El primer precedente que encontramos al vuelo no tripulado se remonta a 1849 cuando Austria, que dominaba gran parte de Italia, quiso bombardear la ciudad de Venecia con globos aerostáticos no tripulados montados con bombas, los cuales tenían implementados temporizadores (Ilustración 3-1)

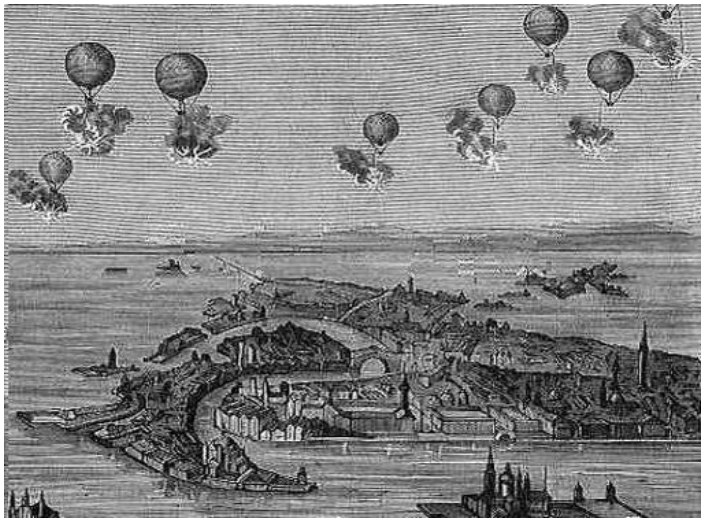

Ilustración 3-1 Bombardeo sobre Venecia con globos aerostáticos no tripulados

Para poder seguir con la evolución que ha sufrido este campo hay que mencionar al hombre gracias al que el control remoto fue posible, Nikola Tesla.

Para poder demostrar la teoría de que era posible una comunicación mediante ondas de radio (1894), inventó en 1898 lo que denominó "*Telautomaton"* [2] un engendro mecánico del tipo naval, capaz de avanzar, girar a ambos lados, pestañear las luces embarcadas, todo ello mediante distintas frecuencias de radio.

En la siguiente (Ilustración 3-2) se observa un dibujo de la demostración que llevó a cabo, así como de una foto de la invención:

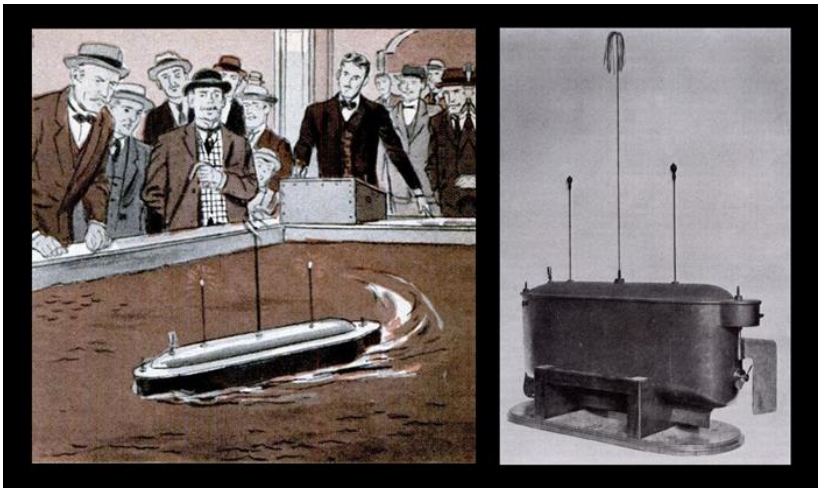

Ilustración 3-2 Telautomaton de Nikola Tesla

La época principal de desarrollo de las aeronaves no tripuladas comienza durante la primera guerra mundial, atravesando su auge durante la segunda guerra mundial y la guerra fría.

Primera guerra mundial:

Durante la primera guerra mundial, la aviación comercial avanzaba con gran rapidez mientras que la aviación no tripulada se vio frenada debido a la falta de desarrollo tecnológico enfocado a este campo. Los principales impedimentos ante los que se encontraron fueron:

- Dificultad de la estabilización automática.
- Control remoto.
- Navegación autónoma.

Peter Cooper Hewitt, un inventor de sistemas eléctricos tenía en mente fusionar las ideas de Tesla con un giróscopo para poder guiar de forma satisfactoria, automáticamente un avión convencional. En 1918 consiguió terminar el invento, un bombardero biplano no tripulado de madera el cual podía albergar hasta 136 kg de carga explosiva, impulsado mediante un motor *Ford* de 40 caballos de potencia, el cual seguía la siguiente metodología de actuación:

- Conocido el viento y la distancia al objetivo, calculaba las revoluciones del motor necesarias para alcanzar el blanco.
- Una vez alcanzado el objetivo, se desprendían las alas del fuselaje cayendo en picado sobre el objetivo.
- El avión era controlado mediante un giróscopo simple y un barómetro.

De forma paralela, se desarrollaron otros sistemas similares que fueron al igual predecesores de los actuales misiles crucero, tales como el Liberty Eagle de Estados Unidos en 1918 (Ilustración 3-3) o como el AT (Aerial Target) de Inglaterra en 1917 (Ilustración 3-4).

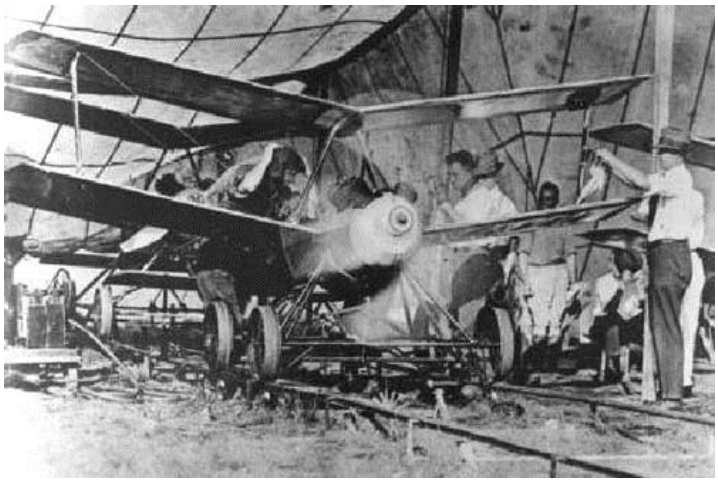

Ilustración 3-3 Liberty Eagle "Bug" 1918

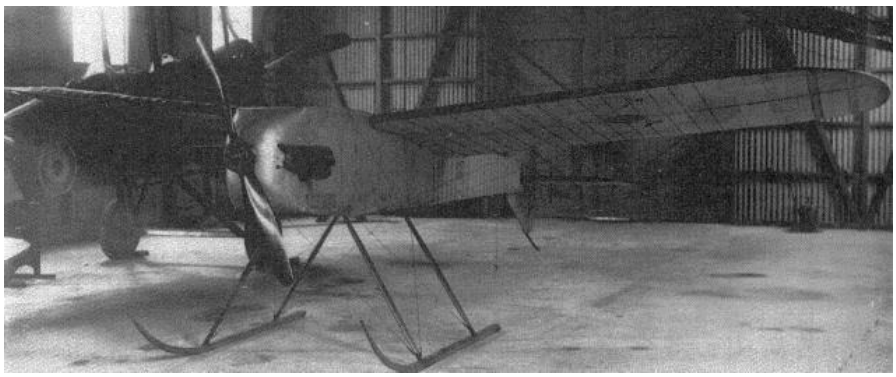

Ilustración 3-4 AT (Aerial Target) 1917

Periodo entre guerras:

Durante esta etapa, se recuperó el interés por este tipo de sistemas. La Royal Navy de Gran Bretaña en 1927 desarrolló un monoplano denominado LARNYX ("*long-range gun with lynx engine*") que podía recorrer una distancia de 480 km y llevaba implementado un sistema que permitía el radio-control para el principio de la misión para después ser pilotado con un plan de vuelo definido a priori.

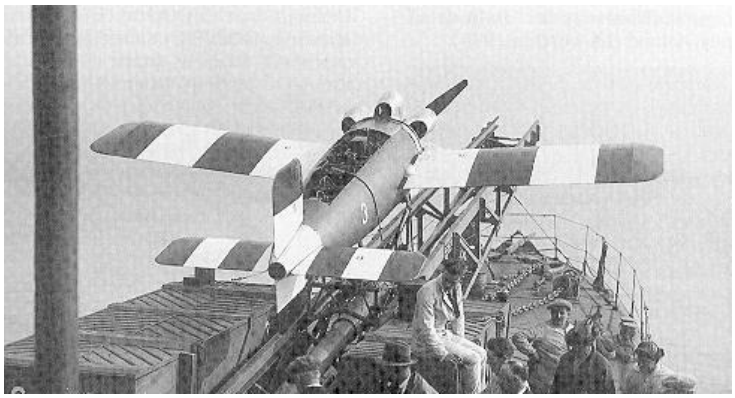

Ilustración 3-5 Larnyx 1927

Por otro, lado la Armada de los Estados Unidos empezó a experimentar con el control de las aeronaves mediante radiocontrol, para dar finalmente como resultado en 1938 el *Curtiss N2C-2*, esta aeronave se controlaba de forma remota desde otra aeronave y era usado como "diana aérea".

Ilustración 3-6 Curtiss N2C-2

Segunda guerra mundial – Guerra fría:

Como en todos los periodos anteriores, se estaban produciendo varios desarrollos en paralelo acerca del mismo campo. Por un lado, Estados Unidos siguió progresando con su idea de los blancos aéreos con control remoto e Inglaterra se pasó a este también a este campo, abandonando su desarrollo de misiles crucero, mientras que la Alemania nazi desarrolló el primer el misil crucero propulsado con un motor de reacción.

Las innovaciones llevadas a cabo por Inglaterra (Ilustración 3-7) y Estados Unidos (Ilustración 3-8) acabaron sirviendo de ayuda a la formación y entrenamiento, mientras que el desarrollo de Alemania (Ilustración 3-9) fue empleado para el bombardeo estratégico a países enemigos, fundamentalmente el sur de Inglaterra.

Ilustración 3-7 Queen Bee Drone 1933-1943

Ilustración 3-8 Rp4 Drone 1939

Ilustración 3-9 V1 Vengueance Weapon 1943

Postguerra:

Durante la postguerra, se produjo un punto de inflexión con respecto a la producción, hasta ahora había sido de escala pequeña-media, sin embargo, gracias a la compañía Northtrop se originó una producción masiva en lo que se denominaba blancos aéreos.

Su principal producto llamado *Shelduck* acabó recibiendo el nombre de BTT ("Basic Training Target"), los cuales embarcaban sistemas controlados por radio-control y siguieron en desarrollo hasta los años 80.

Ilustración 3-10 Shelduck (BTT)

Las fuerzas armadas de los Estados Unidos seguían con el propósito de continuar evolucionando un avión no tripulado con el propósito de servir de prácticas de tiro-tierra, así fue como la empresa Teledyne –Ryan ganó un contrato en 1948 con su UAV Firebee.

Este emblemático avión no tripulado, podía ser lanzado desde tierra (con motores cohete (Ilustración 3-11)) o desde otra aeronave tripulada como el C130 Hércules y era recuperable gracias al uso de paracaídas (Ilustración 3-12). Lo más significante de este UAV, fue que sufrió numerosos desarrollos, pudiendo alcanzar la velocidad supersónica y acabó cambiando su labor, pasando a realizar tareas de reconocimiento y espionaje.

Ilustración 3-11 Despegue Firebee

Ilustración 3-12 Aterrizaje del Firebee

En 1960 el gobierno de Estados Unidos, invirtió una gran suma de dinero en la empresa Ryan, y obtuvo los fondos suficientes para poder llevar a cabo el programa "Big Safari" por el cual se desplegaba una flota de Firebees desde un Hércules (Ilustración 3-13), que actuaba como nave nodriza mediante la cual se podía tomar el control por operadores las distintas naves "insecto", las cuales después de realizar su misión de vigilancia desplegaban paracaídas y eran recogidos por helicópteros.

Ilustración 3-13 Hércules portando 3 Firebees

En 1985 uno de los ingenieros más influyentes en el mundo de las naves no tripuladas Abraham E. Karem, firmó un contrato para el desarrollo de un UAV que poseía gran autonomía denominado Ámbar, éste voló con éxito, pero debido a falta de presupuesto tuvo que simplificar el modelo por lo que nació Gnat.

Ya en 1994, a raíz del Gnat al que le incorporaron comunicaciones vía satélite, surgió el conocido Predator, empleado por la Fuerza Aérea Estadounidense en primera instancia como unidad de reconocimiento y espionaje, pero pasó a ser un UAV de ataque aire-tierra.

En la Ilustración 3-14 se muestra la evolución que sufrieron los distintos prototipos hasta convertirse en el Predator [5]:

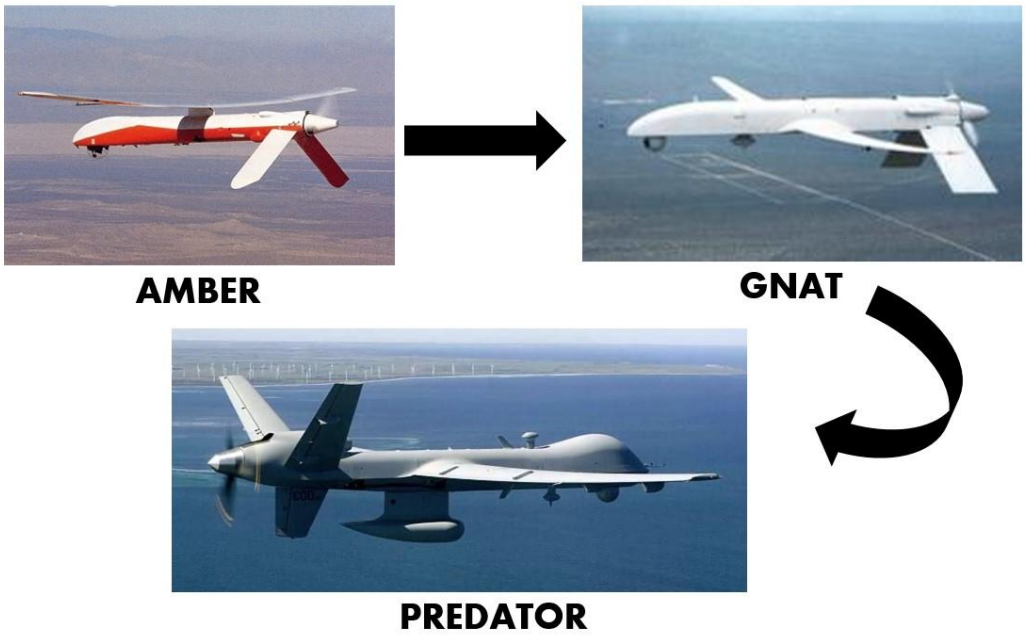

Ilustración 3-14 Esquema de la Evolución del Predator

De aquí en adelante se fueron desarrollando nuevos prototipos, intentando mejorar la eficacia y eficiencia de los anteriores, de donde destacan los modelos de drones VTOL (*vertical take off and landing*) que eludían uno de los problemas principales, el lanzamiento y la recogida. Uno de los más famosos es el Yamaha Rmax que podemos ver en la Ilustración 3-15:

Ilustración 3-15 Operador manipulando un Yamaha Rmax

A partir de este punto, gracias al gran desarrollo tecnológico que está sucediendo, el mundo de los drones se ha metido de lleno en el panorama comercial, sirviendo para infinidades de labores como se explicará a continuación.

3.2 Clasificación de los UAVs

Debido a la gran variedad de UAVs que existen y las innumerables posibilidades que poseen se va a proceder a la clasificación según los siguientes criterios [6] y [7]:

- Máximo peso en despegue (MTOW).
- Alcance.
- Altura media de vuelo.
- Aplicaciones.
- Nivel de Autonomía.
- Morfología.

Según el MTOW:

Nombre	Rango de Peso
Micro	$W \leq 5Kg$
Ligero	$5Kg \leq W \leq 50Kg$
Mediano	$50Kg \leq W \leq 200Kg$
Pesado	$200Kg \leq W \leq 2000Kg$
Muy Pesado	$W \geq 2000Kg$

Tabla 1 Clasificación según su MTOW

Según su alcance:

	Km	horas
Muy Corto Alcance	$10 \leq R \leq 30$	$2 \leq h \leq 4$
Corto Alcance	$30 \leq R \leq 70$	$3 \leq h \leq 6$
Medio Alcance	$70 \leq R \leq 200$	$6 \leq h \leq 10$
Largo Alcance	$R \geq 200$	$h \geq 10$

Tabla 2 Clasificación según su alcance

Según su altura media de vuelo:

	m
Low Altitude Endurance	h ≤ 2.500
Medium Altitude Endurance	2.500 ≤ h ≤ 8.000
High Altitude Endurance	8.000 ≤ h ≤ 20.000

Tabla 3 Clasificación según la altura media de vuelo

Según su aplicación:

- Militar
- Civil
    - Comercial
    - Aficionado

Según su nivel de autonomía en la toma de decisiones:

Nivel	Descripción
0	Vehículo controlado de forma remota
1	Capaz de ejecutar una misión pre-programada
2	Con posibilidad de cambiar de misión
3	Respuesta tiempo-real a distintos eventos
4	Vehículo que se adapta a fallos/eventos
5	Coordinación a tiempo-real multi-vehiculos
6	Cooperación a tiempo-real multi-vehiculos
7	Con conocimiento del espacio de batalla
8	Con reconocimiento del espacio de batalla
9	Conocimiento del espacio de batalla de forma cooperativa
10	Totalmente Autónomo

Tabla 4 Clasificación según el nivel de autonomía

Por último, se puede llevar a cabo una clasificación según la morfología que se puede resumir en la siguiente ilustración [8]:

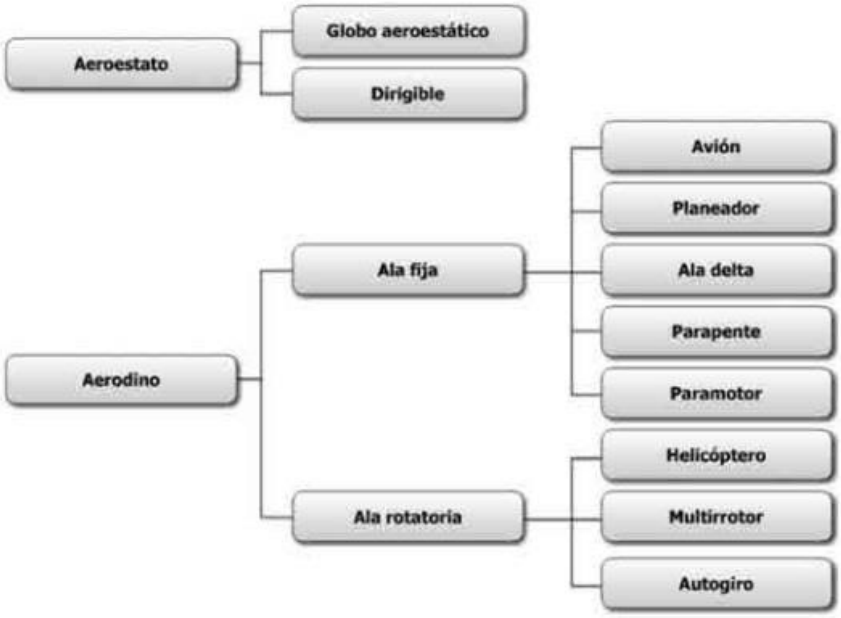

Ilustración 3-16 Clasificación según el tipo de morfología

Para evitar este caos que surge a la hora de querer clasificar estos novedosos engendros mecánicos, la OTAN (Organización del Tratado Atlántico Norte) llevó a cabo su propia categorización que se recoge en la siguiente tabla [8]:

Clase	Categoría	Altitud Media (ft)	Radio medio de misión	Ejemplos
Clase I MTOW ≤ 150 Kg	Pequeño ≥ 20 Kg	h ≤5.000	50 km	Luna, Hermes 90
	Mini 2≤ Kg ≤ 20	h ≤3.000	25 km	Strix
	Micro ≤ 2 Kg	h ≤2.000	5 km	Black Widow
Clase II 150≤ MTOW ≤ 600 Kg	Táctico	h ≤10.000	10 km	Hermes Aerostar Ranger
Clase III MTOW ≥ 600 Kg	Strike Combat	h ≤65.000	Ilimitado	
	HALE (high altitude long endurance)	h ≤65.000	Ilimitado	Global Hawk
	MALE (medium altitude long endurance)	h ≤45.000	Ilimitado	Predator A,B

Tabla 5 Clasificación realizada por la OTAN

3.3 Aplicaciones

La razón fundamental del rápido progreso que han sufrido las aeronaves de este tipo, son las innumerables aplicaciones que tienen y las que pueden llegar a tener en un futuro.

En primer lugar, realizaremos una distinción entre dos grandes grupos, las aplicaciones de tipo militar y las del tipo civil [9]:

3.2.1 Aplicaciones del tipo militar

A continuación, se enumeran las aplicaciones más habituales de este tipo:

- Seguridad
    - Reconocimiento Aéreo
    - Gestión del campo de batalla
    - Guerra biológica
- Búsqueda y Rescate
    - Señalización
    - Supervisión
    - Despliegue de unidades de salvamento
- Comunicaciones
    - Aporte de cobertura
    - Asegurar la seguridad de la misma
- Misiones de ataque
    - Aire-Tierra
    - Aire-Aire

3.2.2 Aplicaciones del tipo civil

Como se ha dicho anteriormente, el gran auge que está viviendo este campo hace que constantemente se produzcan nuevos avances y por tanto surjan nuevas posibles aplicaciones, a continuación, se enumeran las aplicaciones más habituales del tipo civil:

- Cinematográfico.
- Monitorización y gestión de los cultivos (Ilustración 3-17).
- Creación de mapas más precisos Monitorización de la contaminación atmosférica.

- Envío de mercancía (Ilustración 3-18).
- Monitorización de la contaminación atmosférica.

Ilustración 3-17 Dron gestionando un cultivo

Ilustración 3-18 Dron de Correos

3.4 Estado actual de la fotografía y la toma de datos mediante UAVs

Dado que este trabajo trata de desarrollar una aplicación basada en la fotografía y la toma de datos haciendo uso de vehículos aéreos no tripulados, por eso es de relevancia realizar un estudio acerca del estado actual de estas aplicaciones.

Un campo para los que este avance tecnológico ha sido de gran ayuda es el de la topografía, el dron eBee de la compañía SenseFly según [11] es capaz de mapear en 3D una zona de 100 hectáreas en tal solo 40 minutos, mientras que ese era el trabajo de 3-4 semanas realizándose de manera convencional.

Ilustración 3-19 Dron eBee de la compañía SenseFly

Otro uso muy extendido es el del control de los cultivos, el cual se centra en la búsqueda de plagas, localización prematura de enfermedades y el análisis del nivel hidratación y temperatura.

Las cámaras del tipo NDVI (Normalized Difference Vegetation Index) cuantifican a través de la reflectancia de la radiación infrarroja cercana la mayor o menos nutrición que poseen los cultivos, de modo que como se observa en la Ilustración 3-20 el color verde representa un correcto estado del cultivo mientras que el rojo denota falta de nutrientes.

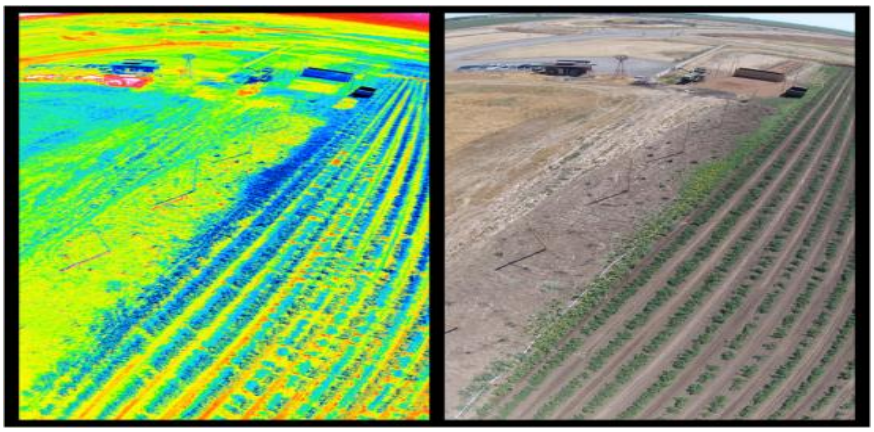

Ilustración 3-20 Índice de Vegetación de un cultivo

4 COMPONENTES DEL SISTEMA

En el capítulo actual se pueden diferenciar tres grandes bloques. Por un lado, el primer bloque engloba el montaje y la puesta a punto del Quadrotor, así como de una explicación detallada de todos los elementos que incorpora el mismo (diferenciando los elementos para llevar a cabo un control básico y los elementos adicionales para cumplimentar los objetivos propuestos.

Por otro lado, en el segundo bloque se habla del software empleado en el proyecto, así como del uso que se ha hecho del mismo.

Para finalizar, el último bloque, explica en detalle el conexionado que se llevó a cabo entre los distintos dispositivos que forman parte del proyecto, así como de la configuración realizada para optimizar la comunicación.

4.1 Hardware Empleado

En este primer bloque, se va a describir de forma detallada, en primer lugar, los elementos que lleva embarcado nuestro quadrotor, así como del proceso que se siguió para el montaje y la puesta a punto del mismo. Esta parte del proyecto fue realizada en conjunto con otros 3 compañeros del grupo de investigación A.C.E.T.I, con los que se llevó a cabo el ensamblaje de una flota de 4 quadrotores.

4.2.1 Elementos Embarcados

A continuación se va a pasar a describir los elementos que se emplearon para poder construir el quadrotor básico, así como del hardware extra que se le implementó para poder lograr los objetivos propuestos, por eso haremos distinción entre ambos:

4.1.1.1 Componentes para la construcción de un quadrotor básico

Para poder llevar a cabo el montaje de un quadrotor que cumpla requisitos sencillos como el de realizar un vuelo controlada de forma remota o el de seguir una serie de *waypoints* (vuelo autómatico), es necesario una serie de componentes fundamentales, los cuales se describen a continuación.

La descripción técnica que se muestra en cada uno de los siguientes elementos, ha sido tomada directamente de las especificaciones proporcionadas por las distintas páginas web donde se realizó la compra [12].

- Chasis:

Comúnmente denominado *frame*, es la estructura principal del quadrotor sobre el que irán montado el resto de elementos. Es importante que tenga una relación alta de resistencia-peso ya que al optimizar al máximo este parámetro nos proporcionará una mayor autonomía, parametro crítico en este tipo de UAV.

Como se ve en la Ilustración 4-2, el chasis consta de cuatro brazos, dos pisos en la parte central y unas patas para poder realizar aterrizajes y despegues suaves y seguros. En los extremos de los brazos se colocan los motores, los cuales iran conectados directamente a los reguladores que se colocan bien sujetos con bridas en la mitad de los brazos. Los distintos pisos se usaran para disponer los demas elementos de la forma mas conveniente posible para que no colisionen o interfieran.

Distancia entre los extremos diagonales del chasis	**700 mm**
Altura con las Patas	360 mm
Peso	825 g

Tabla 6 Especificaciones del frame [14]

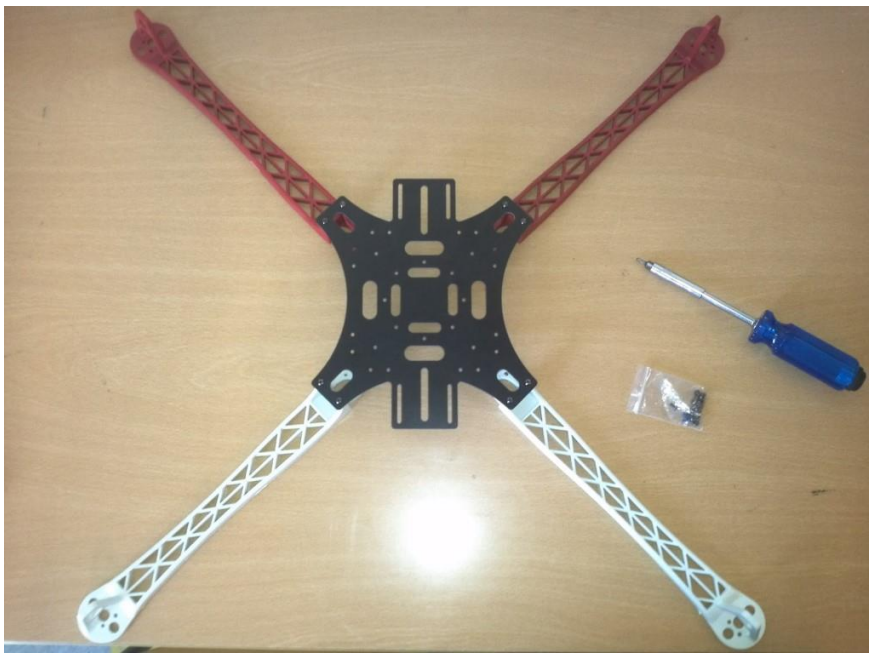

Ilustración 4-1 Parte inferior del *frame* sin patas

Ilustración 4-2 *Frame* ensamblado con las patas

- Motores:

Son los elementos que transforma la energía eléctrica proveniente de la fuente de alimentación en energía mecánica mediante la acción de campos magnéticos generados por sus bobinas. En el caso que nos atañe, convierte la energia eléctrica proveniente de los reguladores en movimiento rotatorio de las palas.

En este proyecto se eligió los motores *MultiStar 4822*, que se pueden ver en la Ilustración 4-3, los cuales tienen las siguientes características:

Tipo de Alimentación	**Trifásica**
Corriente de Trabajo	16 Amperios
Nº de Polos	22
Resistencia Interna	0.052 Ohmios
Diámetro	48 mm
Máximo Voltaje	15 Voltios
Peso por unidad	95 g

Tabla 7 Especificaciones de los Motores [13]

Ilustración 4-3 Motor MultiStar 4822

- Hélices :

Es el elemento que se conecta directamente al eje del motor, al que se le transmitirá la energía mecánica para que al girar genere la sustentación necesaria para poder elevarse. A más revoluciones realice el motor, más sustentación generarán las hélices y por tanto mayor velocidad de elevación obtendrá el quadrotor.

Las especificaciones de las hélices son las siguientes:

Material	**Fibra de Carbono**
Peso	20 g
Longitud	14 pulgadas
Paso	4.7 pulgadas

Tabla 8 Especificaciones de las Hélices

En la Ilustración 4-4 podemos ver las hélices que se escogieron para el montaje:

Ilustración 4-4 Hélices sin ensamblar en el motor

- ESCs:

Los ESCs (*electronic speed controller*), o habitualmente conocidos como reguladores, son unos componentes que realizan las siguientes tareas fundamentales:

a) Hace de BEC (*Battery Eliminator Circuit)*, esto significa que elimina la necesidad que se tenía de llevar distintas baterías para los motores y para el resto de componentes.

b) Ya que la alimentación, como se verá a continuacion, es del tipo continua y los motores deben ser alimentados de forma trifásica, los ESCs realizan la conversión continua-alterna..

c) Hacen uso de su nombre, regulando la velocidad de giro de los motores en función de la demanda emitida por el piloto a través de la conexión con la placa controladora.

Los reguladores que se han cogido para este proyecto en concreto son los *Turnigy Plush 30A* (Ilustración 4-5), los cuales tienen las siguientes especificaciones:

Peso	**25 g**
Tamaño	45x24x11
Corriente Máxima	10 A
Burst Actual	40 A
Modo BEC	Lineal
BEC	5V/2ª

Tabla 9 Especificaciones de los ESCs [15]

Ilustración 4-5 ESC *Turnigy Plush 30A*

Como se puede apreciar en la ilustración anterior, de cada regulador salen por lado el izquierdo 3 cables de corriente que irán conectados a un motor, mientras que por el derecho salen 3 cables, 2 de corriente que irán conectados al PDB y otro irá conectado a la placa controladora, de modo que se posibilita el cumplimiento de las funciones anteriormente descritas.

- Placa de Control:

La placa de control elegida para este proyecto es del tipo ArduPilotMega (APM) y en concreto se ha elegido la HKPilot Mega 2.5.

Las funciones principales de la controladora APM son por un lado, permitir el control remoto y que a su vez sea estable, y por otro lado, que lleve acabo unas misiones preasignadas de manera autónoma y precisa. Todo esto es posible gracias a su programación como software libre (del cual hablaremos en profundidad en el siguiente capítulo).

A continuación se enumeran algunos de los elementos que lleva integrado la placa:

-Puerto micro-USB.

-Puerto de Telemetría.

-Puerto GPS.

-Magnetómetro encargado de dar información a la controladora acerca de su orientación.

-Regulador de 3.3 V.

-Dataflash (memoria interna).

-Barómetro que mide la presión atmosférica respecto a una de referencia..

-IMU (Es la placa de control inercial, da información acerca de la actitud del quadrotor, es decir sus aceleraciones).

-Microprocesador.

-Puerto de la brújula magnética externa.

A continuación se muestra la placa controladora elegida sin la carcasa protectora:

Ilustración 4-6 APM HkMega 2.7 sin carcasa protectora

- Receptor y Emisora de Radio

El receptor de radio es el elemento encargado de recibir la demanda ejercida por el piloto (mediante el emisor), el receptor esta conectado a la controladora para que pueda asimilar la información y mandar órdenes a los motores para que el quadrotor siga las instrucciones marcadas.

Se ha elegido el modelo *Turnigy 9X8C-V2* que tiene las siguientes especificaciones [18]

Dimensiones	**52x35x15 mm**
Peso	18 g
Canales	8
Frecuencia	2.4 Ghz

Tabla 10 Especificaciones Receptor Turnigy 9X8C-V2

Ilustración 4-7 Receptor Turnigy 9X8C-V2

La emisora de radio es el elemento por el cual, podremos controlar de forma remota nuestro quadrotor. Habrá que enlazarla con el receptor que se encontrará en el dron para que se establezca la comunicación de forma satisfactoria. El modelo elegido es el Turnigy 2.4G 9x .

Ilustración 4-8 Emisora Turnigy 9x

- Batería

Es la encargada de suministrar alimentación a todos los elementos funcionales del quadrotor, desde la controladora hasta los motores pasando por los componentes adicionales que se describirán más adelante.

El modelo de bateria escogido es el *Zippy FlightMax 8000mAh 30C* con las siguientes especificaciones:

Dimensiones	**184x60x90 mm**
Peso	845 g
Capacidad	8000 mAh
Celdas x Voltaje	4x14.8V

Tabla 11 Especificaciones Batería

Ilustración 4-9 Batería Zippy 8000 mAh

- Power Distribution Board (PDB)

Es el componente encargado de dividir la corriente proveniente de la bateria y enviarla a los distintos motores a través de su conexión con los reguladores.

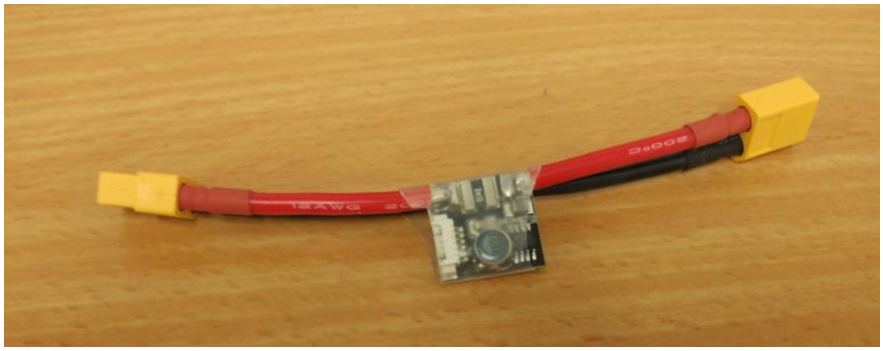

Ilustración 4-10 Power Distribution Board

4.1.1.2 Componentes adicionales

Con los elementos descritos anteriormente se podrían conseguir las funciones básicas de un quadrotor, sin embargo, ya que se definieron unos objetivos a cumplir (Capítulo 1), es necesario implementar otros componentes adicionales para poder lograr dichos objetivos, los cuales se describen a continuación:

- Raspberry Pi 3

Es uno de los componentes clave de este proyecto, se trata de un computador de placa simple o SBC (*single board computer*). La Raspberry Pi es famosa por ser un ordenador de tamaño reducido con bajo coste, creada por la Universidad de Cambdridge para fomentar el estudio y desarrollo de la programación en las escuelas, siendo actualmente la más vendida entre los de su tipo.

La razón de que se haya escogido esta placa es su versatilidad con la cual se pueden llevar a cabo infinidades de proyectos, desde fabricar una emisora FM hasta controlar la domótica de un domicilio.

Gracias a este ordenador que podremos llevar embarcado en el quadrotor, podremos realizar las tareas necesarias para poder cumplir los objetivos preestablecidos, como por ejemplo la conexión remota con el dron desde otro ordenador a traves de una conexión wifi o como por ejemplo el ensamblaje de una cámara para poder realizar varias capturas mientras nuestro dron realiza una misión.

A continuación se muestra las especificaciones del modelo de SBC empleado,la Raspberry Pi 3B [16]:

1. Procesador:
    - Chipset Broadcom BCM2387.
    - Procesador ARM Cortex A53 de 4 nucleos de 64 bits a 1.2 GHz sobre el que correremos el sistema operativo Raspbian (Linux).
2. RAM: 1GB.
3. Conectividad:
    - Puerto Ethernet.
    - 4 Puertos USB y 1 Puerto Micro-USB.
    - Bluetooth 4.1.
    - LAN inalámbrica.
    - Salida Video/Audio.
        i. HDMI 1.4.
        ii. RCA (PAL y NTSC).
        iii. Jack de 3.5 mm.
    - Conector GPIO 40 clavijas (20x2) de 2.54 mm.
4. Ranura para tarjeta MicroSD.
5. Ranura para PiCamera.

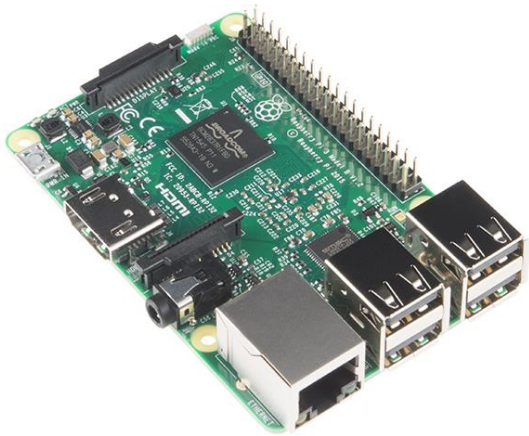

Ilustración 4-11 Raspberry Pi 3B

- Cámara

Gracias a este componente podremos realizar las capturas que se requieran para, al igual que antes, completar los objetivos marcados.

En este caso se ha elegido el modelo PiCamera V2 (Ilustración 4-13) ya que teniendo un coste bajo, nos proporciona unas imágenes y videos con resolucion HD y para más índole está preparado para su implementación directa a la placa Raspberry Pi, teniendo su propia ranura, por lo cual nos facilita el conexionado (Capitulo 4.3).

Ilustración 4-12 Ranura para la Conexión de la Picamera V2

A continuación se enumeran sus especificaciones [17]:

- Enfoque fijo de 8 Megapíxeles.

- Compatible con 1080, 760p60 y VGA90.

- Sensor Sony IMX219PQ.

- Conexión cable plano de 15 contactos.

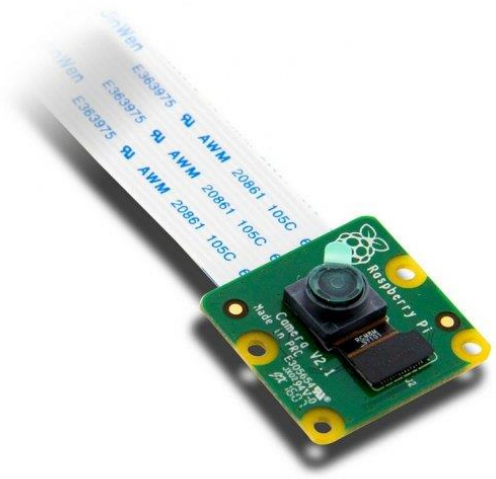

Ilustración 4-13 PiCamera V2

4.2.2 Montaje del Quadrotor

En este apartado del capítulo se va a proceder a explicar el procedimiento que se siguió para poder llevar a cabo el montaje del quadrotor y de todos los demas elementos que integran este proyecto. Recalcar que cuando se mencione el conexionado entre dos componentes, se encuentra explicado en detalle dicho proceso en el Capítulo 4.3.

4.1.2.1 Instalación de los Motores en las patas

Se empezó el montaje del dron con la instalación de los motores en cada una de las patas, por lo que previamente se ensamblo el chasis con el piso inferior (Ilustración 4-14). Mediante tornillos y arandelas se fijan los motores a las patas como se ve en la figura Ilustración 4-15, se ha empleado bridas para que los cablen queden sujetos a la estructura.

Ilustración 4-14 Ensamblaje del piso inferior del Chasis

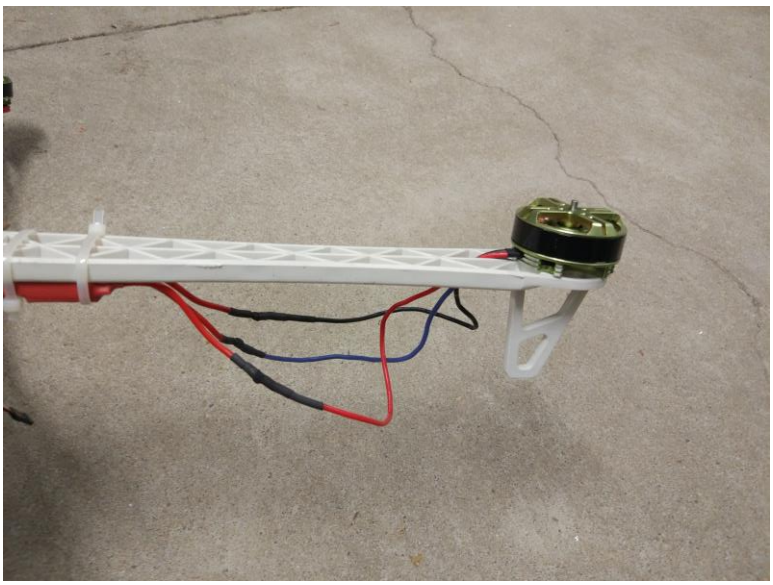

Ilustración 4-15 Colocación de los motores y sus cables

4.1.2.2 Conexionado de los ESCs

Como se explicó antes, era necesario el uso de unos reguladores que fuesen conectados a los motores para que controlasen su velocidad de giro, por lo cual en este apartado se lleva a cabo dicho proceso. Empezamos dándonos cuenta de que al soldar los cables de ambos componentes obteniamos una conexión demasiado larga, por lo optamos por reducir el cableado por parte del motor. Lo siguiente que se hizo fue agregarle conectores macho-hembra a los cables de la siguiente manera:

-Conectores macho a los motores.

-Hembra a la parte de los reguladores que van al motor.

-Macho a la parte de los reguladores que se conectarán a posteriori al PDB.

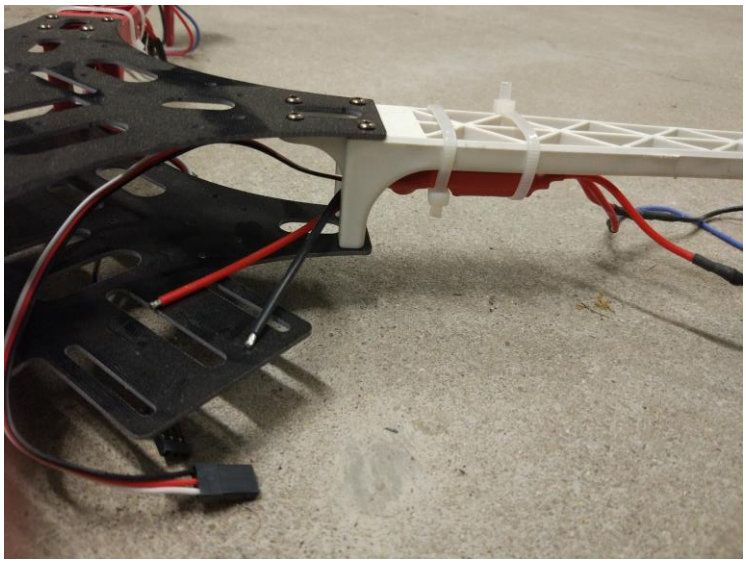

Ilustración 4-16 Regulador instalado con bridas

Ilustración 4-17 Dron con Motores y Reguladores

Después de este proceso largo y tedioso de soldadura se procedió al conexionado de los motores con los ESCs, como sabemos, los quadrotores tienen 4 motores los cuales 2 giran a derechas y los otros 2 a izquierdas como se muestra en la Ilustración 4-18. Dichos motores, como ya explicamos eran del tipo trifasico, así que en el caso en que no girasen cuando se pusiese en marcha como se especifia en dicha ilustracion , bastaría con intercambiar dos de los tres cables de la conexión que salen de los reguladores y van a los motores.

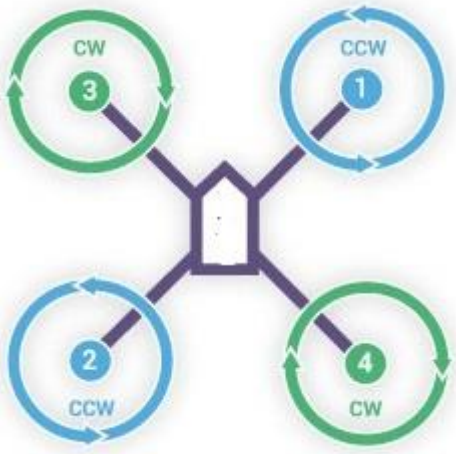

Ilustración 4-18 Giros de los motores en un quadrotor básico

Por otro lado, los ESCs estan conectados tambien al PDB y a la controladora como se dijo anteriormente, por lo que conectamos los cables de corriente al PDB y el otro cable a la ranura de *outputs* que posee la controlador, ya que las revoluciones de cada motor es una respuesta que procesa la APM ante una demanda del piloto.

Como se explicará más adelante en la parte de configuración llevada a cabo, se ha elegido la configuración de quadrotor X, por lo que las salidas de los reguladores que van a la placa controladora deberán conectarse de la siguiente manera siguiendo la ilustracion anterior:

Output1-Motor1.

Output2-Motor2.

Output3-Motor3.

Output4-Motor4.

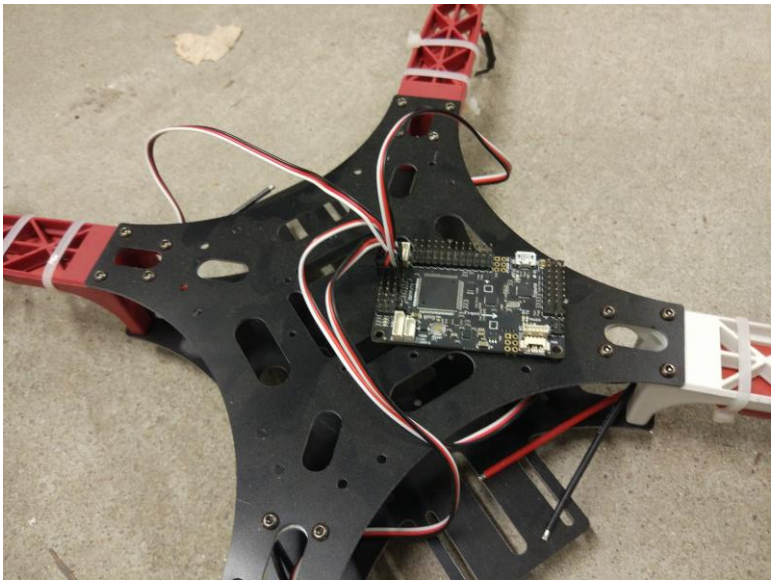

Ilustración 4-19 Conexión Reguladores-APM

4.1.2.3 Instalación de la controladora APM

Ahora pasamos a explicar una de las elecciones fundamentales del montaje. Ya que todos los elementos integrantes de este proyecto están directamente o indirectamente conectados a la controladora, la elección de su posición ha sido por tanto crucial, se decidió mantenerla fijada a la bandeja inferior mediante cinta adhesiva de doble cara con el objetivo de tener la batería (la cual ocupa un volumen importante) en la bandeja superior ya que al necesitar ser cargada con bastante frecuencia, de este modo sería más accesible.

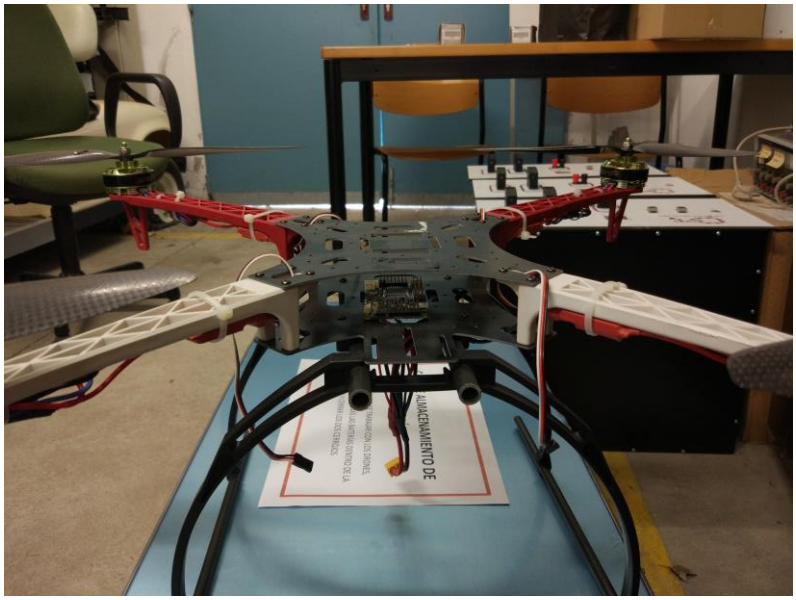

Ilustración 4-20 Instalación de la placa controladora (APM)

4.1.2.4 Conexión de la controladora con el receptor

Una vez decidida la posición de la controladora, es momento de empezar a conectar los distintos elementos anteriormente explicados, empezaremos por el receptor el cual como se explicó tenia ocho canales, de los cuales hemos usado 5, cuatro para poder controlar la posicion y actitud del quadrotor y el quinto para poder poder cambiar entre modos de vuelos con la emisora.

El tipo de cable (tipo servo) es similar al de la conexión regulador-controladora, tal y como se muestra en la Ilustración 4-21:

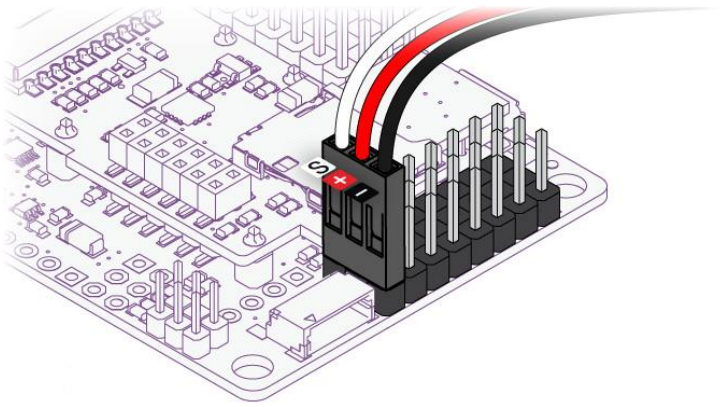

Ilustración 4-21 Cable tipo servo

Para el correcto conexionado se ha seguido la siguiente ilustración:

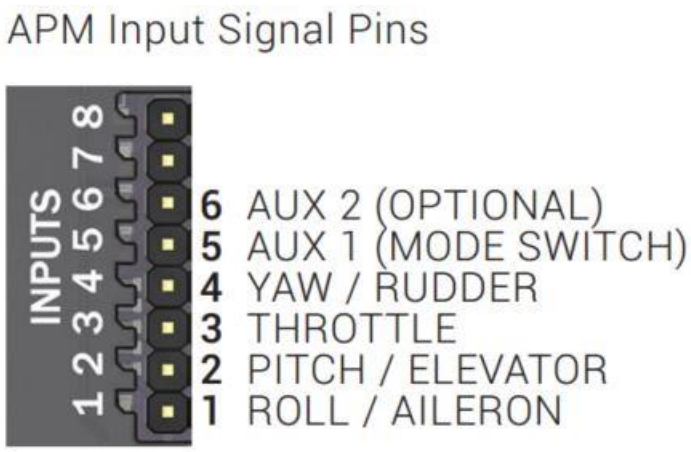

Ilustración 4-22 Conexionado Imputs del receptor

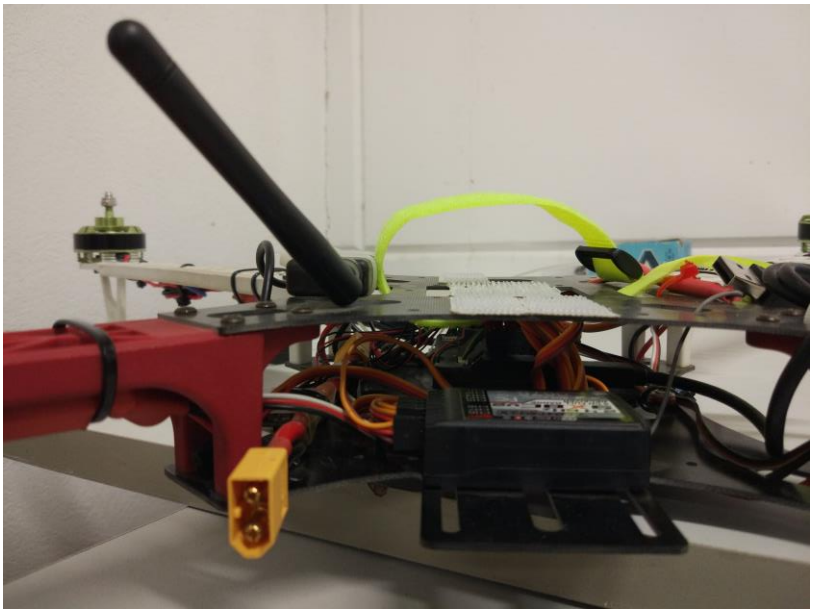

Ilustración 4-23 Conexión Receptor-APM

4.1.2.5 Conexión del módulo GPS y el Magnetómetro

Para la conexión del GPS y el magnetómetro se emplearon las ranuras correspondientes como se puede observar en la siguiente figura (superior de GPS y la inferior de magnetómetro).

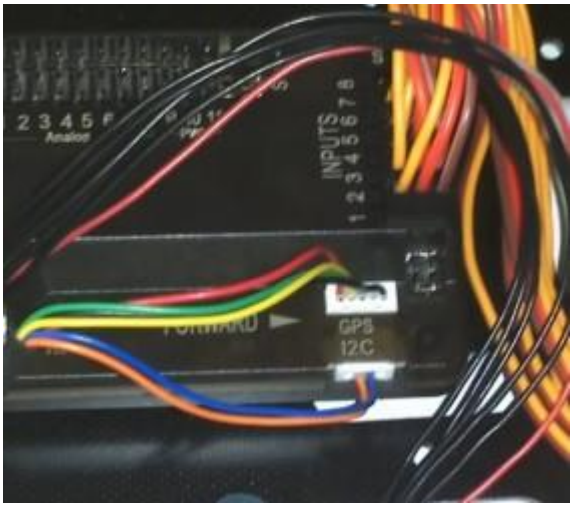

Ilustración 4-24 Conexion GPS y Magnetómetro

Finalmente se colocó de la siguiente manera el modulo GPS para que tuviese menor interferencia con el resto de componentes:

Ilustración 4-25 Quadrotor con el GPS instalado

4.1.2.6 Alimentación del Quadrotor

Ahora pasamos a la colocación de la bateria y al conexionado de la misma.

Como se puede observar en la Ilustración 4-26 y tal y como se dijo anteriormente, se decidió colocar la bateria en la bandeja superior, por ser mas accesible.

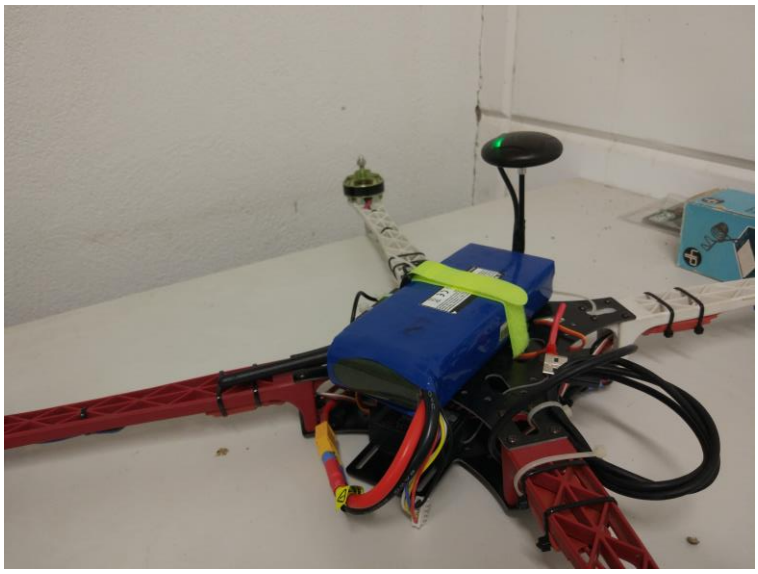

Ilustración 4-26 Quadrotor con la Batería instalada en la bandeja superior

A continuación, en la Ilustración 4-27 se muestra el conexionado explicado hasta este momento[19]

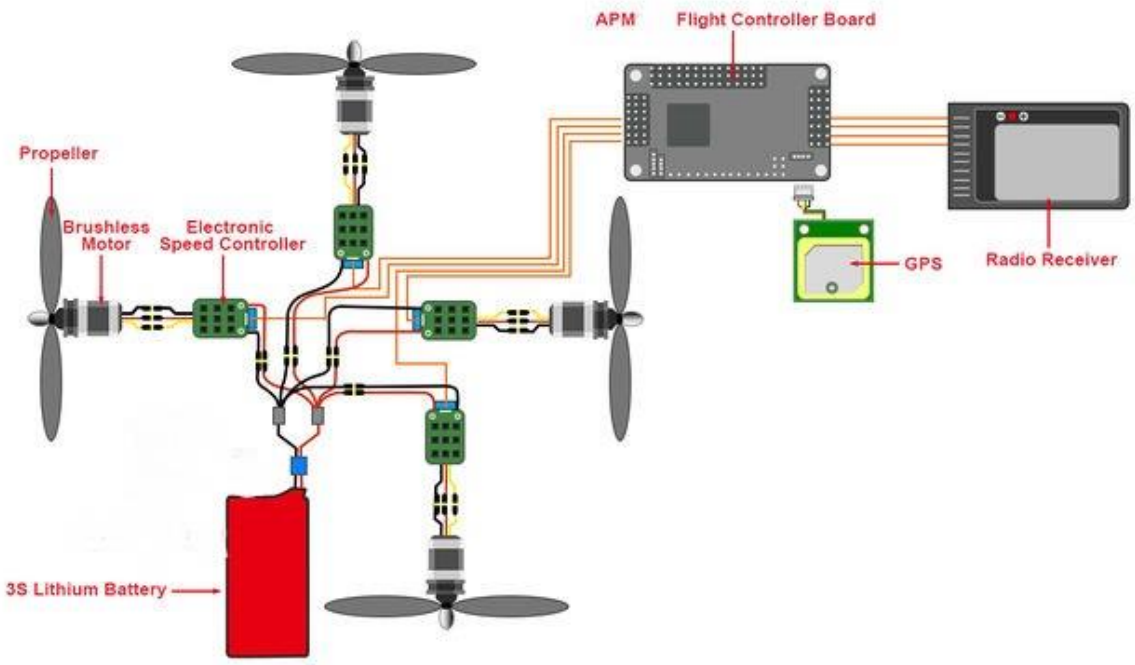

Ilustración 4-27 Conexionado del Quadrotor

4.1.2.7 Montaje de Raspberry Pi y la Camara

Tal y como se observa en la Ilustración 4-28, la cámara se fijó mediante bridas, de forma que apuntase a la parte frontal del quadrotor, el resto del conexionado se explica en detalle en el Capitulo 4.3.

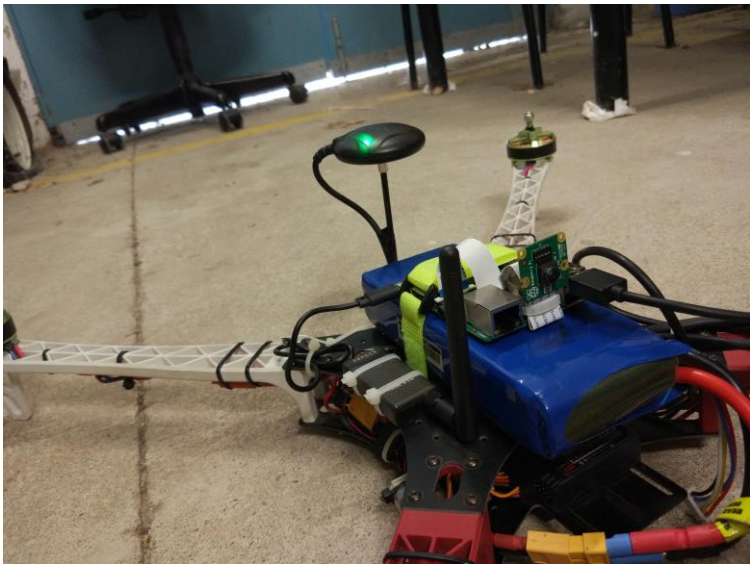

Ilustración 4-28 Raspberry Pi y Cámara instalados en el Quadrotor

A continuación se muestra una foto del quadrotor terminado:

Ilustración 4-29 Montaje final del Quadrotor

4.2.3 Configuración (MissionPlanner)

Una vez se llevo a cabo todo el montaje físico, era necesaria una configuración del hardware embarcado. Dicha configuración se realizó conectando la controladora a nuestro portatil mediante un cable MicroUSB-USB y haciendo uso del programa *Mission Planner* (Ilustración 4-30) tal y como se explica a continuación:

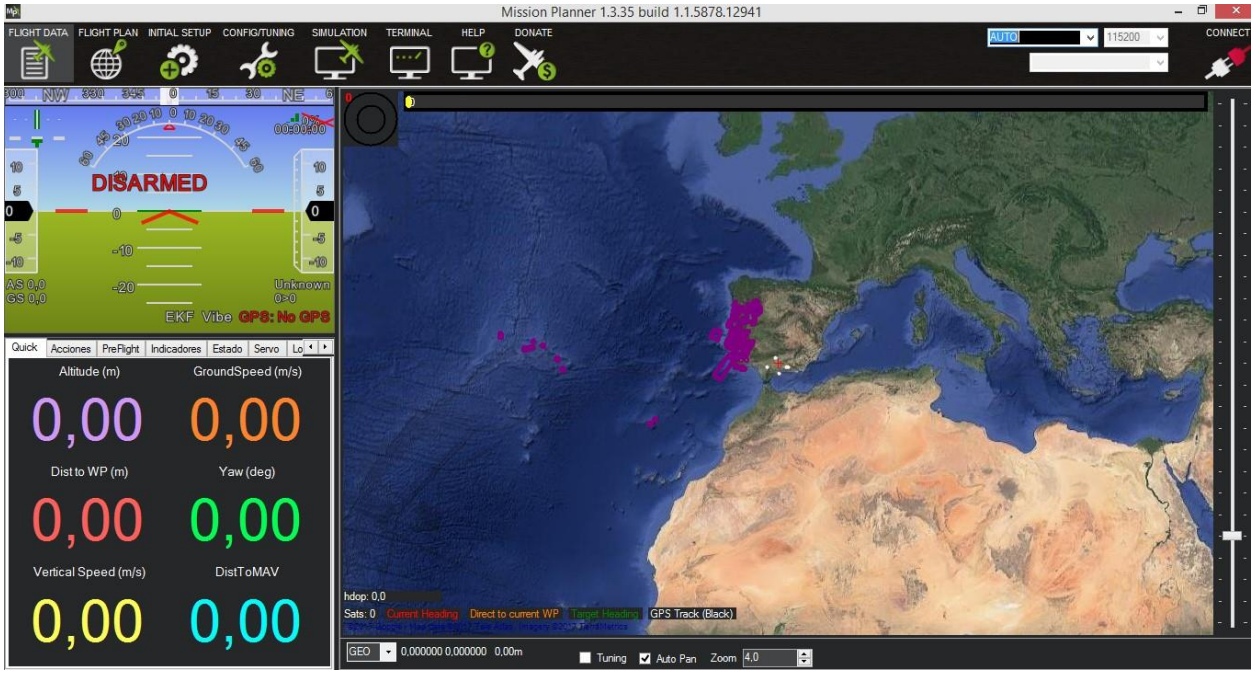

Ilustración 4-30 Pantalla principal del Mission Planner

- Instalación del Firmware

El primer paso a realizar fue cargar el firmware a nuestro quadricoptero para ello seleccionamos el puerto correspondiente y un baudrate de 115200, tal y como se muestra en la siguiente ilustración:

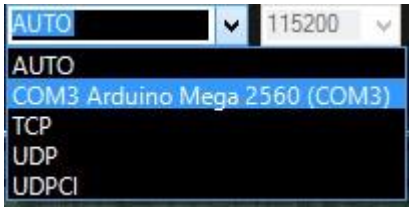

Ilustración 4-31 Selección de Puerto y Baudrate

Ahora clickamos en *Install Firmware*, seleccionando nuestro modelo de quadrotor:

Ilustración 4-32 Instalación de Firmware

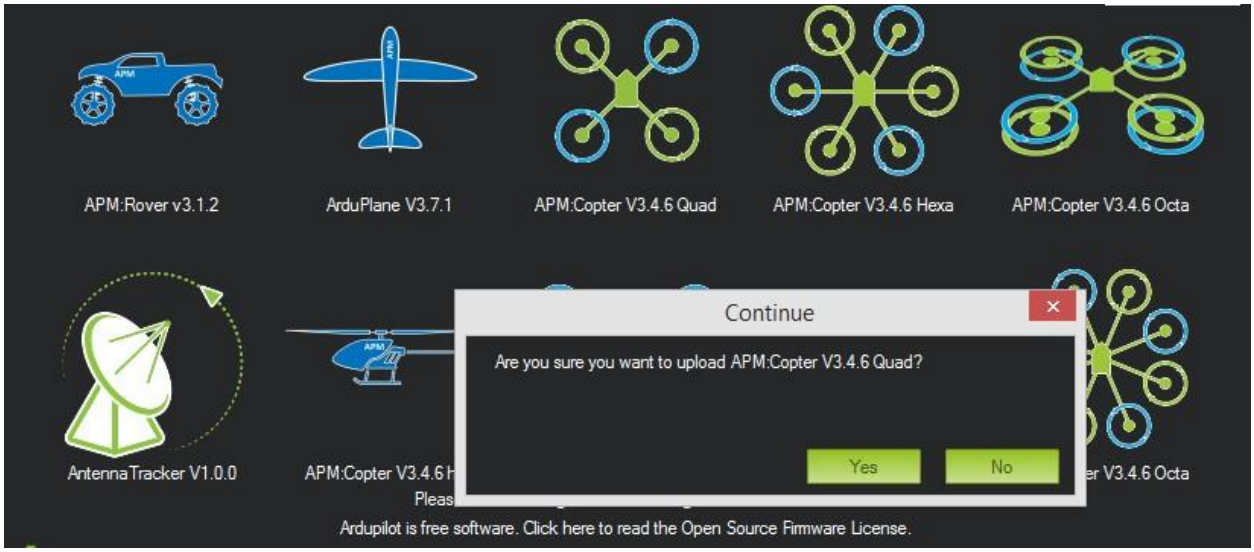

Ilustración 4-33 Instalación de Firmware 2

- Configuración para poder llevar a cabo el primer vuelo:

Clicando justo debajo de *Install Firmware* encontramos la opcion de *Wizard* con la que podremos configurar mediante una serie de pasos nuestro quadrotor para que este listo para realizar el primer vuelo de forma estable y controlada:

1 Selección de nuestro tipo de vehículo.

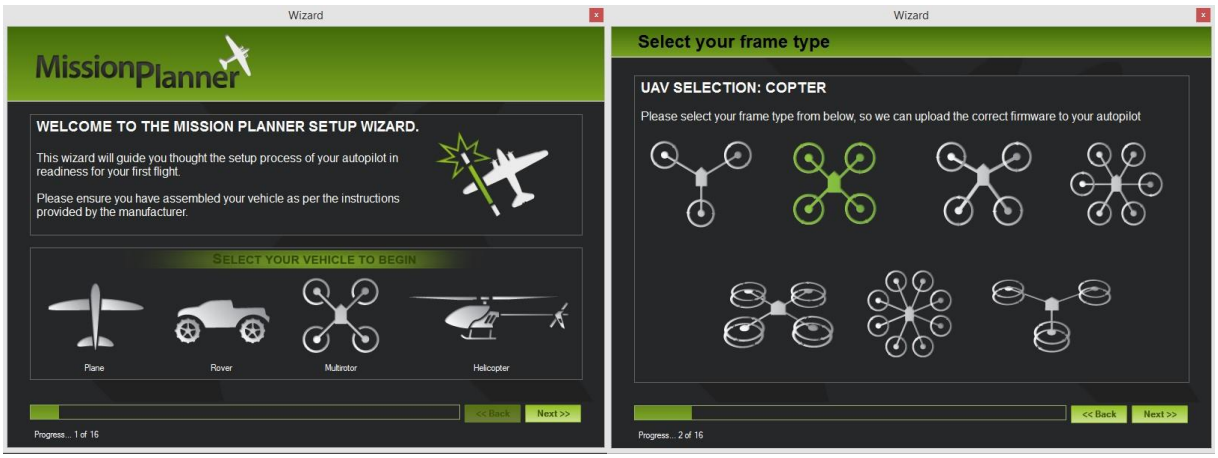

Ilustración 4-34 Selección del tipo de Vehículo y de Chasis

2 Calibración del acelerómetro.

Para la correcta configuración del acelerómetro, colocaremos el dron en las posiciones que nos indica el programa, pulsando cualquier botón para pasar a la siguiente pestaña:

Ilustración 4-35 Posiciones para configurar el acelerómetro

Ilustración 4-36 Ejemplo de una de las posiciones en la que disponer el dron

3 Calibración de la brujula.

Siguiendo las indicaciones, debemos rotar el dron en todos los ejes 360 grados, como se puede observar en la siguiente figura:

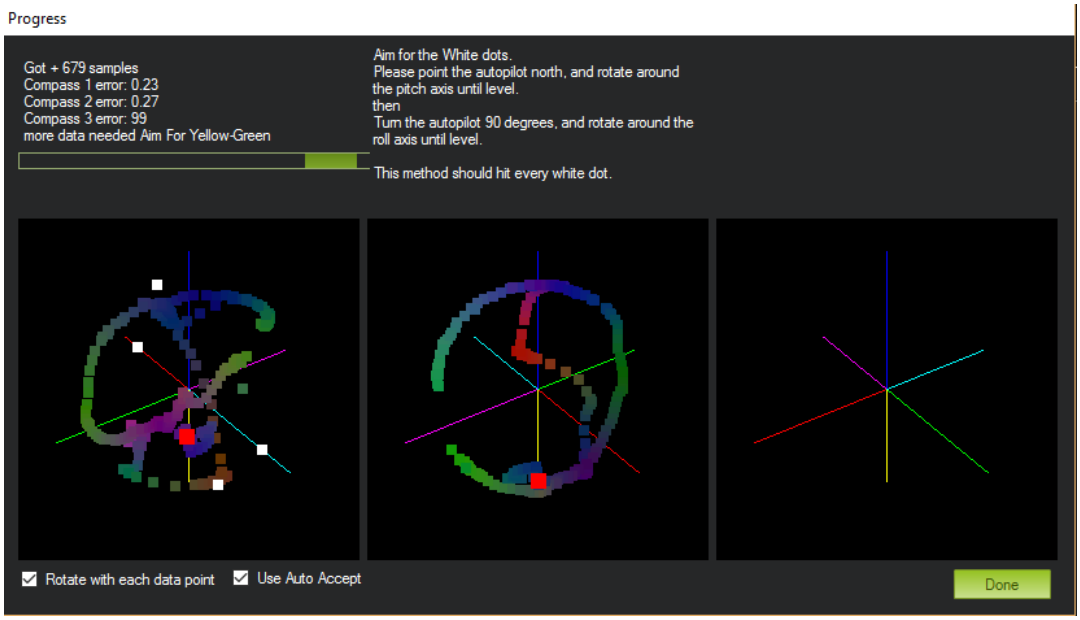

Ilustración 4-37 Calibración de la brújula

4 Batería.

Para la correcta configuración de la batería, rellenamos los datos de la siguiente imagen:

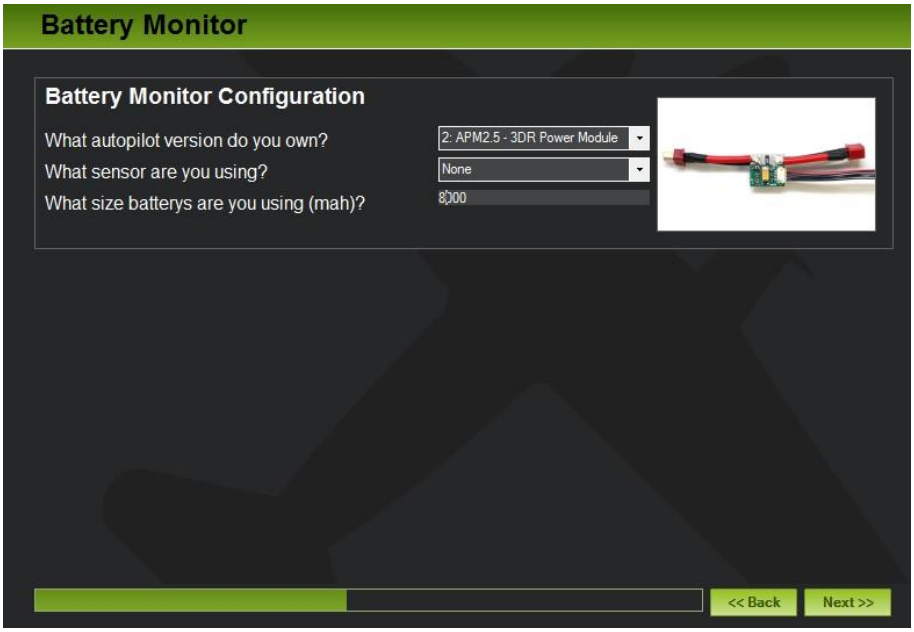

Ilustración 4-38 Configuración de la batería

5 Calibración de la Radio.

Para llevar a cabo una correcta calibración es necesario tener encendida la radio y tener conectado el receptor a la controladora. Esta parte de la configuración es crucial para que se pueda llevar a cabo un correcto control remoto.

Unicamente tendremos que siguiendo [18] asegurarnos que los sticks estan centrados antes de proceder a la calibracion, mover ambos sticks a su maxima deflexion y obtendremos lo mostrado en la siguiente ilustracion:

Ilustración 4-39 Calibración de la Radio

Como podemos observar, el valor verde es la posición actual de la palanca, mientras que las líneas rojas delimitan los valores máximos y mínimos que pueden tomar las diferentes variables.

6 Configuración de los Parámetros.

Ahora una vez esta instalado, ahora hay que configurar ciertos parámetros clicando en *configuration-FullParametresList* podemos acceder a ellos.

Ilustración 4-40 Configuración de los parámetros

Si bien hay numerosos parámetros que pueden ser variados y/u optimizados, a continuación se describen los de mayor importancia:

- *MOT_SPIN_ARMED:* Por defecto viene con el valor de 1 y habrá que cambiarlo a 0 para que cuando se arme el dron con la emisora no giren los motores hasta que no empleemos la palanca de gases, lo que si no fuese asi podria provocar accidentes.

- *ANGLE_MAX*: Por defecto viene en 4500 (45º) y lo tornaremos a 3500 (35º) ya que el anterior era muy elevado y podria ocasionar un comportamiento irreversible.

- *BATT_CAPACITY:* Cambiarlo a nuestra capacidad de bateria, en nuestro caso es de 8000mAh.

4.2 Software Empleado

A continuación, se va proceder a la descripción de los distintos programas empleados en este proyecto, así como de extensiones y protocolos de comunicación de los que se hizo uso para posibilitar el cumplimento de los objetivos.

4.2.1 Mission Planner

Para poder llevar a cabo una correcta puesta a punto es necesario hacer uso de algún tipo de aplicación o programa que nos permita configurar nuestro quadrotor para poder realizar nuestros objetivos específicos, ésta es la razón por la que usamos este programa.

MissionPlanner es una estación de tierra para el configurado y el control de dispositivos autónomos o de

control remoto, compatibles con el proyecto de ArduPilot. Gracias a este programa podremos tanto ajustar ciertos parámetros a nuestra conveniencia como diseñar y ejecutar misiones.

4.2.2 Python

Es el lenguaje de programación del que se ha hecho uso fundamentalmente en este trabajo. Se ha usado a la hora de llevar a cabo la configuración del resto de software, para habilitar el conexionado de los distintos elementos y para llevar a cabo la creación de aplicaciones.

Tiene las siguientes características como lenguaje de programación [21]:

- Propósito general (Se pueden crear todo tipo de programas).
- Multiplataforma (Es compatible con numerosos sistemas operativos).
- Es un lenguaje de programación interpretado (No hay que compilar el código antes de su ejecución).
- Interactivo (Cada sentencia se ejecuta y produce un resultado visible).
- Tiene numerosas librerías con las que se pueden realizar infinidades de labores.

Ventajas con respecto a otros lenguajes de programación:

- Portabilidad (Se puede transcribir fácilmente los scripts a otros lenguajes).
- Es gratuito.
- Hay una gran comunidad, ahí que haya esa gran cantidad de librerías.
- Simplicidad.
- Facilidad para su uso en dispositivos (Hay numerosos elementos que están basados en este lenguaje de programación por lo que la implementación de códigos es directa, un ejemplo es la Raspberry Pi).

Ahora se van a describir las dos desventajas fundamentales que se tiene si se emplea este lenguaje de programación:

- Poca flexibilidad o rapidez (Es la contrapartida que tiene que sea un lenguaje simple y fácil de usar).
- Seguridad (Tiene algunos problemas de seguridad, cosa muy a tener en cuenta cuando de internet se trata.

4.2.2.1 Librerías y Módulos empleados

- NumPy:

Es una librería de código abierto que da soporte a vectores y arrays para Python.

- Pip:

Gestor de paquetes de Python.

- Matplotlib:

Es una librería que se sirve para poder graficar de manera rápida y sencilla a partir de vectores y arrays.

- PiCamera:

Es la librería necesaria para poder controlar la cameraV2 (explicada en el Capítulo 6).

- PyMAVlink:

Dado que es necesaria una comunicación entre el ordenador y la unidad de vuelo (en nuestro caso el UAV), gracias a esta librería se habilita dicha conexión median el protocolo de comunicación Mavlink que se más adelante se explicará.

4.2.3 MAVlink

Es el protocolo de comunicación que se ha empleado ya que está especializado para el intercambio de información entre un UAV y una estación tierra.

Esta información se encuentra contenida en archivos *.xml* lo que facilita ser estructurada y empaquetada.

La base de esta comunicación son los paquetes MAVlink que siguen la siguiente estructura [21]:

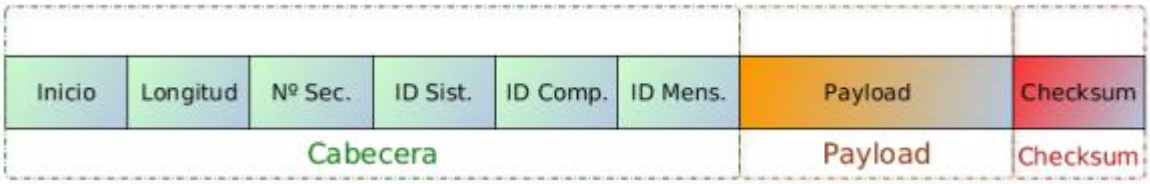

Ilustración 4-41 Paquete MAVlink

Byte	Descripción
Inicio	Indica el principio del paquete
Longitud	Indica el número de bytes que lleva el paquete (0-255)
Número de secuencia	Muestra el número de paquete enviado (Para detectar errores)
Identificación del sistema	Identifica al sistema diferenciando otros de la misma red
Identificación del Componente del Sistema	Diferencia que componente del sistema se está comunicando
Identificación del Mensaje	Indica el tipo de mensaje
Payload	Carga útil del paquete, donde se encuentra la información útil que se desea transmitir
CheckSum	Controla y declara errores producidos

Tabla 12 Paquete MAVlink

4.2.4 DroneKit

Es la librería en la que se basa en su mayoría todo el desarrollo de la aplicación (Capítulo 7), gracias a la cual podemos realizar una gran variedad de funciones [24] como por ejemplo crear una misión, inicializar una misión, cambiar de modos de vuelo, extraer datos del GPS…

4.2.5 Raspbian Jessie

Es una distribución del sistema operativo *Linux* basado en *Debian Wheezy* optimizado para la SBC Raspberry Pi, es decir el sistema operativo que está siendo ejecutado por nuestra SBC.

4.3 Conexionado

Una parte fundamental del proyecto es llevar a cabo el correcto conexionado entre los elementos hardware, explicados con detalle en primer bloque, así como de asegurarnos de que se está realizando de forma precisa la transferencia de información mediante los protocolos de comunicación definidos en el segundo bloque.

En primera instancia se pensó que la mejor manera de llevar a cabo el conexionado fue la siguiente:

1 Conexión en serie de la Raspberry Pi con la placa Ardupilot HkPilot Mega 2.7 (APM).

2 Conexión de la Raspberry Pi con una cámara modelo *Camera V2*.

3 Conexión de la Raspberry Pi a una red Wi-Fi.

4 Conexión de un ordenador a una red Wi-Fi.

5 Conexión remota a la Raspberry Pi con dicho ordenador a través de dicha red Wi-Fi.

En la siguiente imagen podemos observar el esquema básico del conexionado del proyecto pensado en primera instancia:

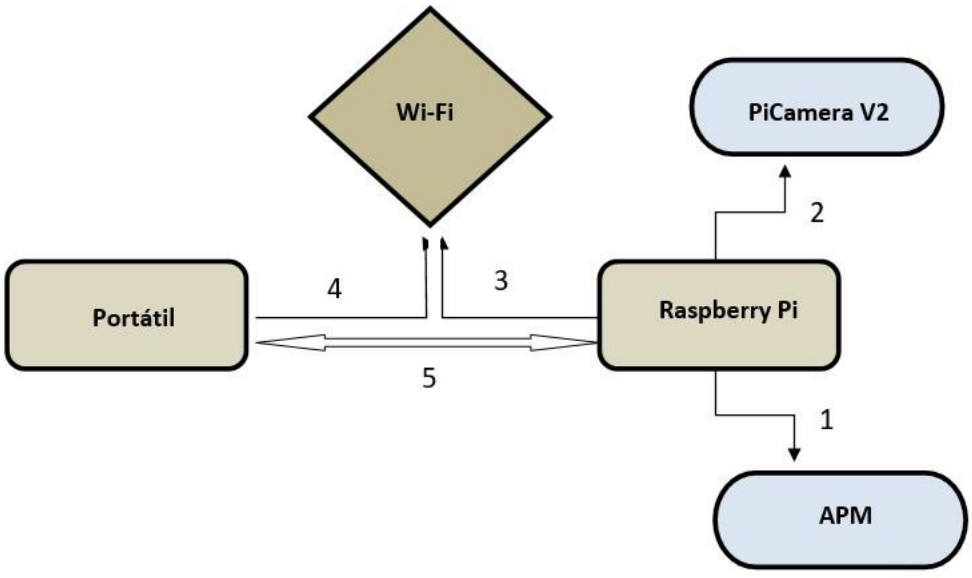

Ilustración 4-42 Esquema Básico del Conexionado 1

Después de realizar una serie de pruebas, se pensó que era más eficiente realizarla de la siguiente manera:

1 Conexión en serie de la Raspberry Pi con la placa Ardupilot HkPilot Mega 2.7 (APM)

2 Conexión de la Raspberry Pi con una cámara modelo *Camera V2*

3 Raspberry actuando como *hot-spot*

4 Conexión de un ordenador a la red Wi-Fi que genera la Raspberry Pi para poder controlarla de forma remota

En la siguiente figura se muestra un esquema básico del conexionado final:

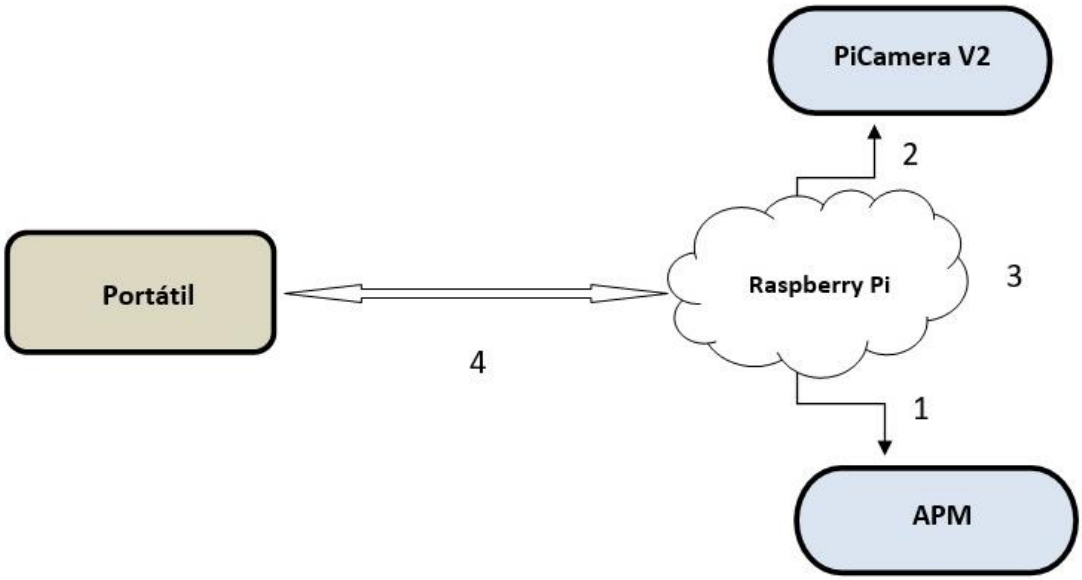

Ilustración 4-43 Esquema Básico del Conexionado 2

A continuación, se va a explicar en detalle cada una de las conexiones del proyecto, así como de las configuraciones que se han llevado a cabo para el logro de las mismas:

4.2.1 Conexión en Serie de la Raspberry Pi con la Placa Ardupilot Hkpilot mega 2.7 (APM)

Para poder llevar a cabo el objetivo del trabajo, era necesaria una conexión en serie de la placa APM con el *companion computer* (en este caso la Raspberry Pi 3).

En primera instancia se decidió realizarla por medio de los puertos GPIO desde la Raspberry Pi ("General Purpuse Input/Output"), que tal y como su nombre indica, es un sistema de pines que puede usarse como entrada y/o como salida, al puerto UART0 de la placa APM.

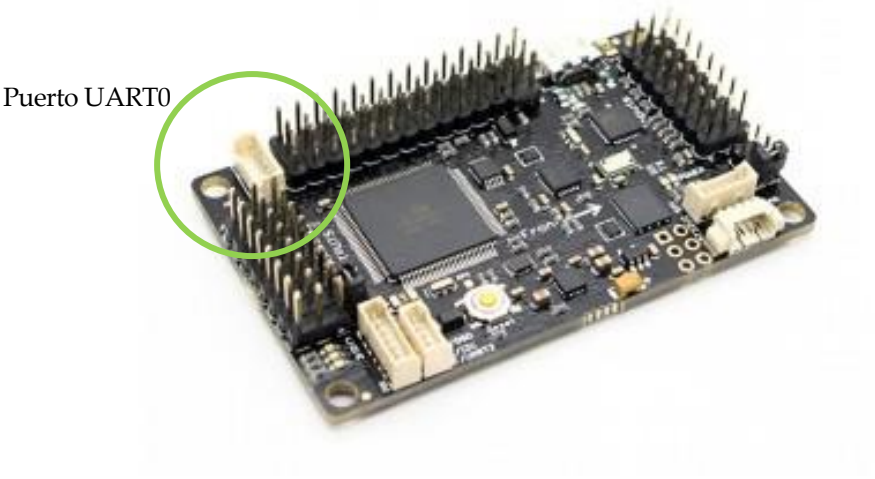

Ilustración 4-44 Puerto UART0 de la Rpi3 Modelo B

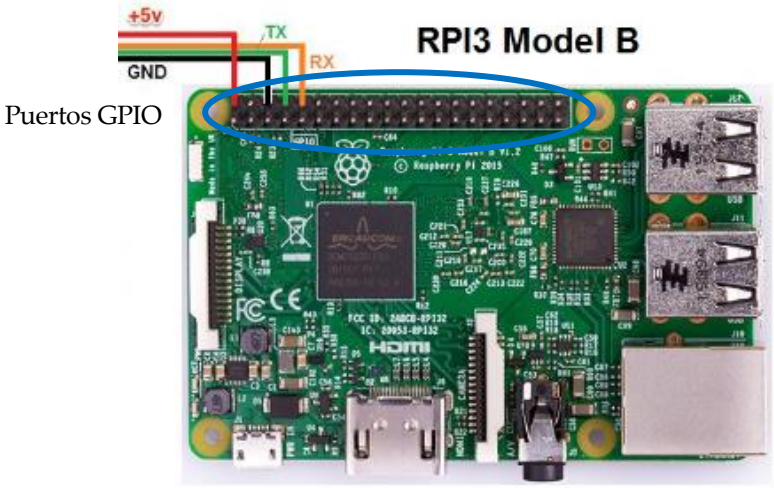

Ilustración 4-45 Puertos GPIO de la Rpi3 Modelo B

Para el proceso de configuración de esta conexión en serie se siguieron una serie de pasos, los cuales siguen:

-Conectar la Raspberry Pi al ordenador en serie

El principal objetivo de conectar en serie el ordenador a la Raspberry Pi es únicamente para que sea más cómodo y menos aparatoso, es decir el poder prescindir en cualquier momento de un monitor, un teclado y un ratón a la hora de trabajar con nuestra Raspberry, de modo que la controlaremos con el teclado de nuestro portátil y con el ratón del mismo

Este proceso se llevó a cabo mediante el uso de un cable Ethernet y el acceso a una red Wi-Fi local conocida.

- Descarga e Instalación de los paquetes adecuados de los paquetes adecuados para el conexionado

Introduciendo los siguientes comandos en la terminal instalamos todo lo necesario para que se cree la conexión:

```
sudo apt-get update    #Update the list of packages in the software center
sudo apt-get install screen python-wxgtk2.8 python-matplotlib python-opencv python-pip python-
numpy python-dev libxml2-dev libxslt-dev
sudo pip install pymavlink
sudo pip install mavproxy
```

Gracias a esto, podremos establecer una conexión entre ambas placas mediante el protocolo de comunicación MAVlink, explicado en el capítulo 5.

- Testeo de la Conexión

Una vez realizada la conexión y la instalación de los paquetes convenientes para nuestro propósito, hay que probar que todo está correctamente enlazado y la comunicación bidireccional entre ambas placas está sucediendo de forma correcta:

A continuación, se muestra el conexionado a través de los pines GPIO-UART0

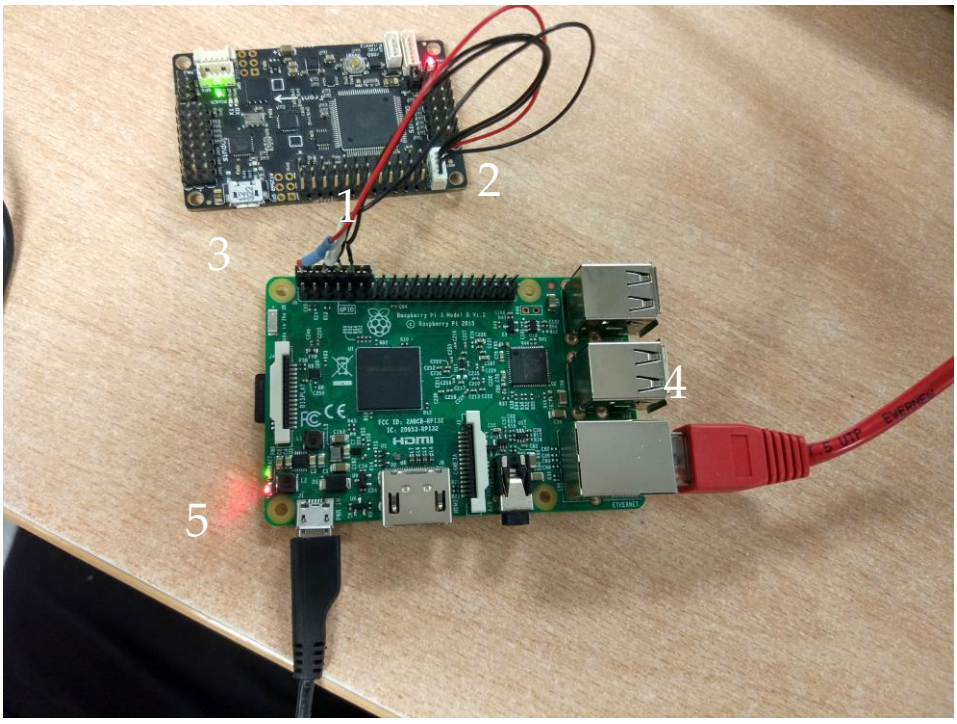

Ilustración 4-46 Conexionado V1 Rpi-APM

1 Cable que conecta en serie 2 con 3 por el que se transmite y recibe de manera bidireccional la información y se le da alimentación a la APM.

2 Puerto UART0 de la Raspberry Pi.

3 Pines GPIO de la placa APM 2.7.

4 Puerto Ethernet para poder controlar la Raspberry Pi a través del ordenador.

5 Puerto mini-USB para darle alimentación a la Raspberry Pi.

Para realizar dicha comprobación, introducimos el siguiente comando en la Raspberry Pi:

```
sudo -s
mavproxy.py --master=/dev/ttyAMA0 --baudrate 57600 --aircraft prueba1
```

Ahora deberíamos de obtener una respuesta de la APM reflejada en la Raspberry, pero en cambio se obtiene lo siguiente:

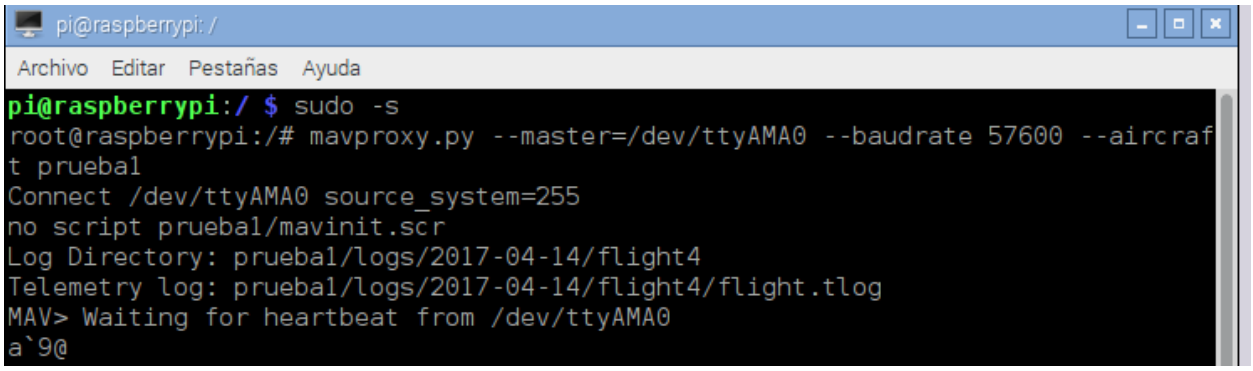

Ilustración 4-47 Resultados Conexión V1

Como podemos observar, se recibe por pantalla el siguiente mensaje:

"Waiting for the hearbeat from /dev/tttyAMAO"

Esto quiere decir, que la APM no está enviando ningún dato, por lo cual no se ha establecido de forma correcta la conexión en serie de dichas placas.

Después de la investigación acerca del posible fallo que se podría estar cometiendo: Cambiando algunos parámetros, probando otro tipo de configuraciones, pero siempre haciendo uso de los pines GPIO. A través de la documentación encontrada en diferentes foros, se explicaba que dependiendo del modelo de la Raspberry Pi y del modelo de la placa Ardupilot que se quisiese conectar en serie a través de los pines GPIO-UART0 se producían una gran variedad de *bugs*, por lo que se decidió solventar dicho problema realizando la conexión a través de USB era la solución más conveniente.

A continuación, se muestra el conexionado a través del puerto USB – mini-USB:

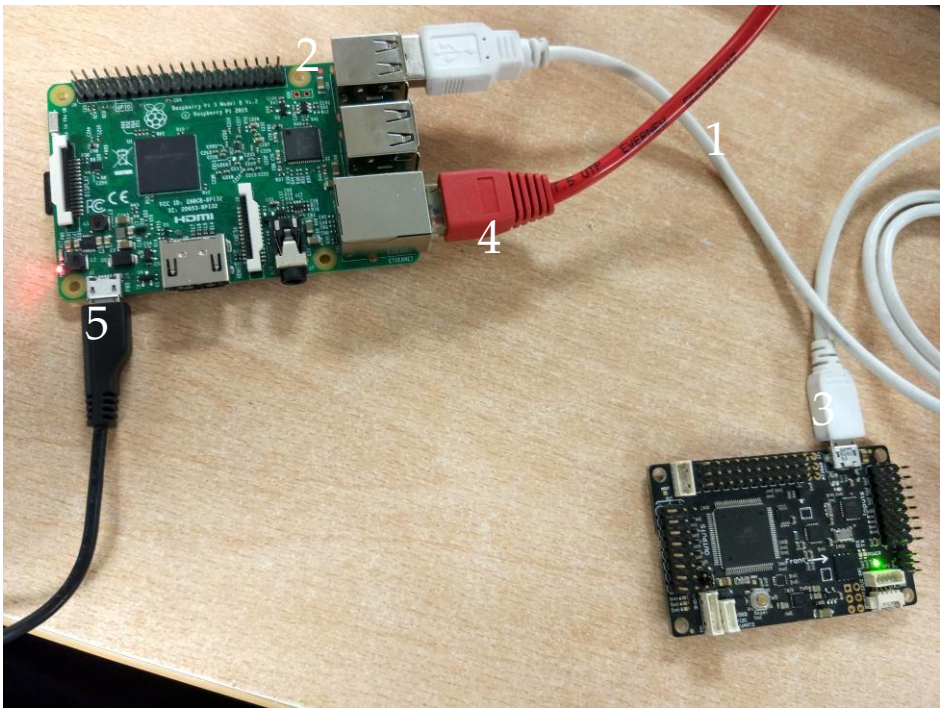

Ilustración 4-48 Conexionado V2 Rpi-APM

1) Cable que conecta en serie 2 con 3 por el que se transmite y recibe de manera bidireccional y se le da alimentación a la APM.

2) Puerto USB de la Raspberry Pi.

3) Puerto mini-USB de la placa APM 2.7.

4) Puerto Ethernet para poder controlar la Raspberry Pi a través del ordenador.

5) Puerto mini-USB para darle alimentación a la Raspberry Pi.

El único cambio que se debía hacer para realizar dicha conexión con respecto a lo descrito anteriormente es lo siguiente:

`=/dev/ttyAMA0 → =/dev/ttyACM0`

`--baudrate 57600 → --baudrate 115200`

Al realizar estos cambios en el código obtenemos lo siguiente:

```
sudo -s
mavproxy.py --master=/dev/ttyACM0--baudrate 115200 --aircraft DroneUSB
```

Al ejecutar este comando obtenemos:

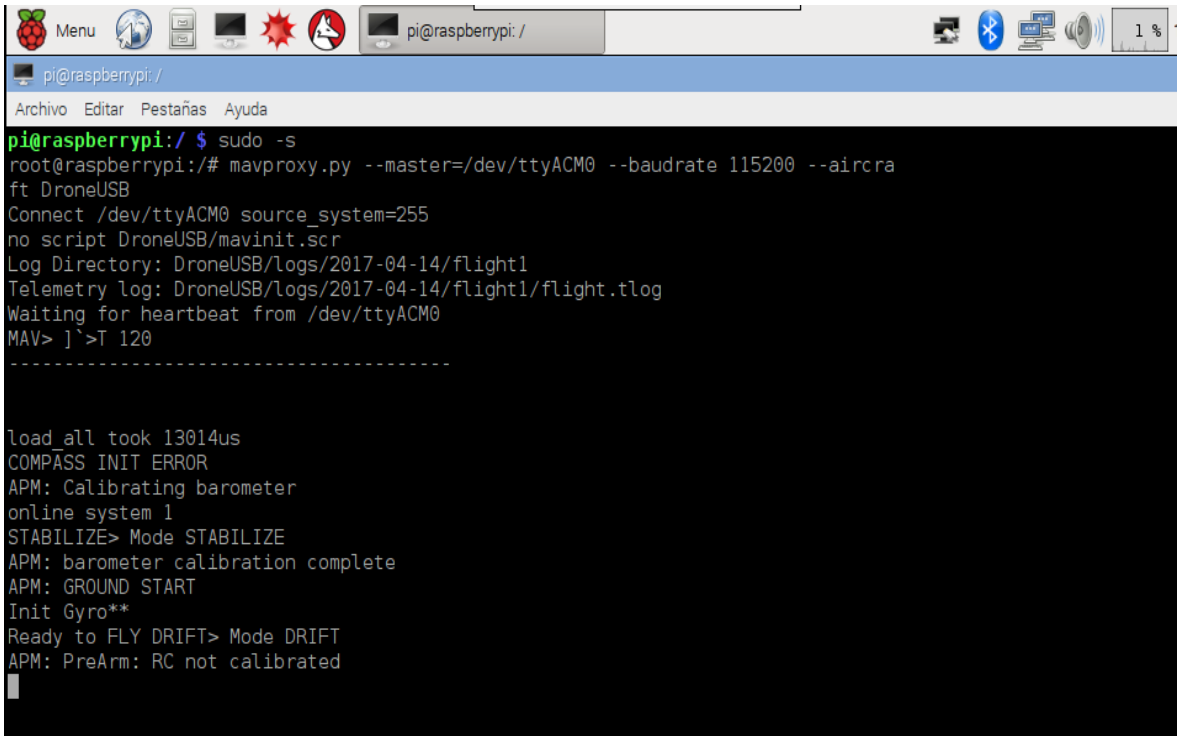

Ilustración 4-49 Resultados Conexión V2

Podemos observar como ahora obtenemos por pantalla entre otras cosas:

"Online System 1"

"APM: Ready to Fly > Mode DRIFT"

Ahora si se está produciendo la emisión y recepción de información entre ambas placas y podemos enviar comandos y recibir respuesta, por lo que podemos proceder con el siguiente paso.

4.2.2 Conexionado de la Raspberry Pi con una cámara modelo *Camera V2*

Una vez se dispone del módulo cámara para Raspberry (Ilustración 4-50), hay que realizar una serie de configuraciones para poder hacer uso del mismo.

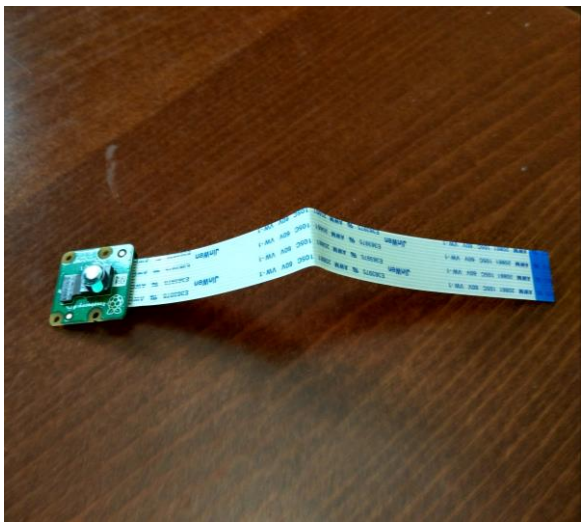

Ilustración 4-50 Camera V2

El primer paso es conectar el módulo en la Raspberry Pi, esta conexión se realiza a través de un bus de cinta que va a un conector especializado en nuestra placa como se observa en la siguiente ilustración:

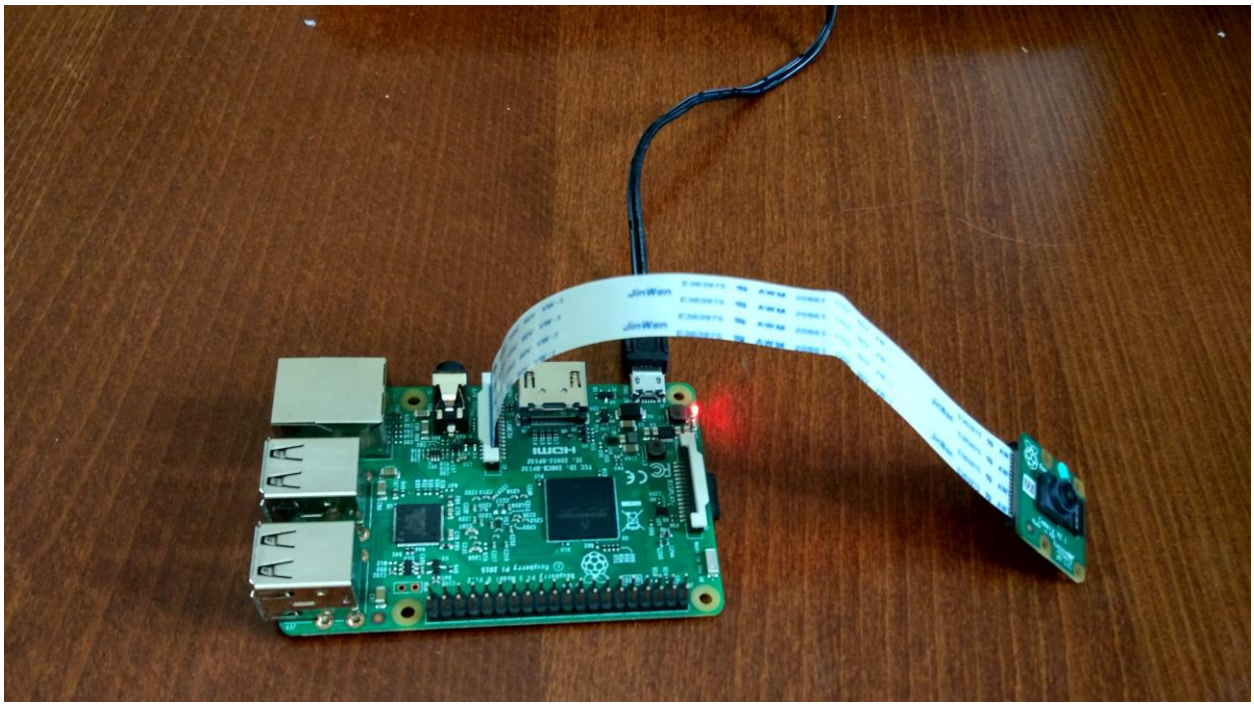

Ilustración 4-51 Conexionado Rpi-Camera V2

Una vez se ha conectado, se procede a habilitar el modulo cámara en la configuración de la Raspberry Pi, introducimos el siguiente comando para acceder al menú de configuración de la misma:

```
sudo raspi-config
```

Nos aparecerá la siguiente ventana y tendremos que seleccionar la opción interfaces (Ilustración 4-52), y más

tarde la de cámara (Ilustración 4-53):

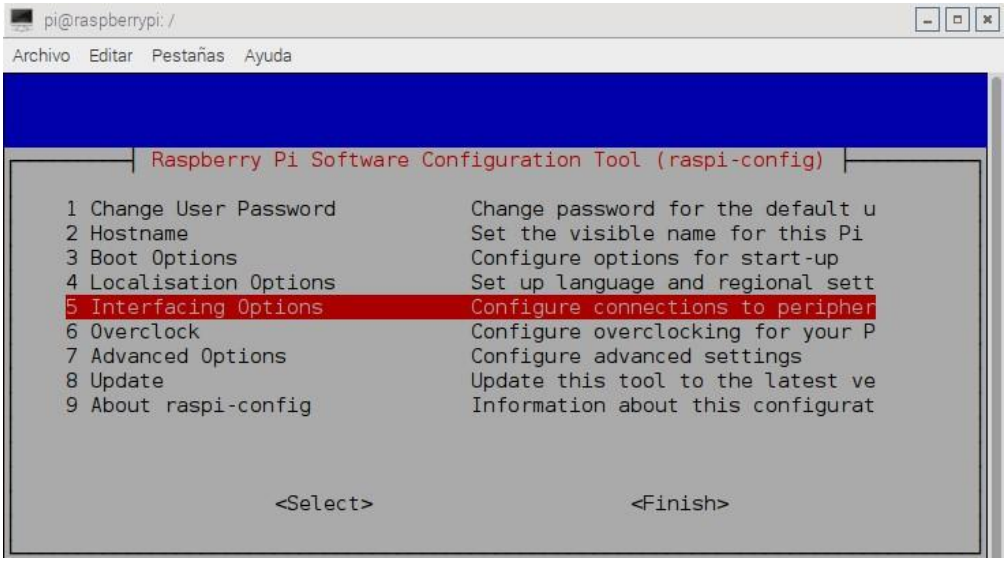

Ilustración 4-52 Interfaces_1

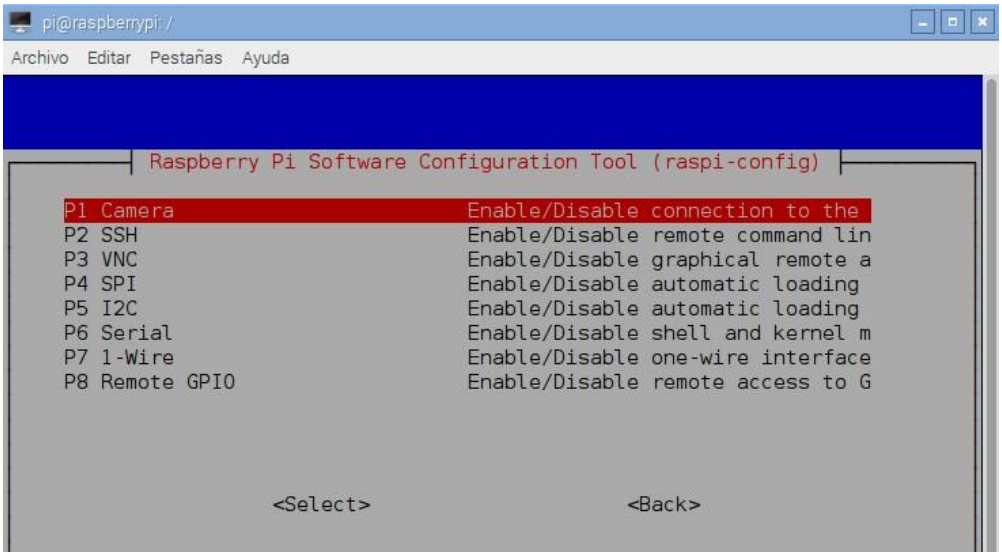

Ilustración 4-53 Interfaces_2

Y ahora la habilitamos:

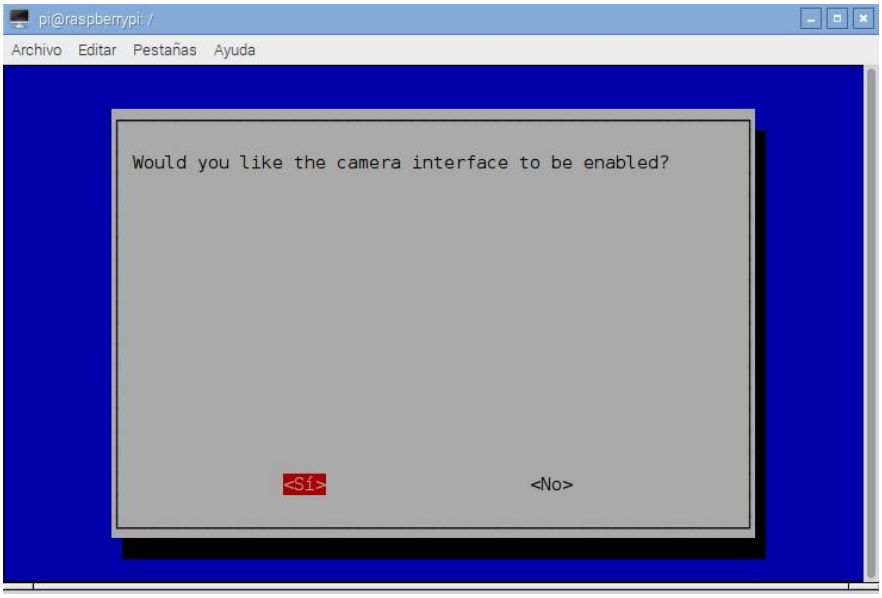

Ilustración 4-54 Interfaces_3

Y por último la reseteamos la Rpi3 para que el cambio en la configuración sea efectivo con el siguiente comando:

```
sudo reboot
```

4.2.3 Conexión Remota a la Raspberry Pi con un ordenador a través de una red Wi-fi

Para poder llevar a cabo esta parte del conexionado, es necesario que ambos elementos (ordenador y Raspberry Pi) estén conectador a la misma red Wi-fi.

Una vez se hace esto, se procede con los siguientes pasos:

Para realizar este paso se ha hecho uso del programa *TightVNC*, para el cual se ha necesitado descargar e instalar tanto en la Raspberry Pi como en el ordenador dicho programa.

Una vez descargado el programa, ejecutamos el siguiente comando en el terminal de la Raspberry Pi para correr el servidor:

```
Tightvncserver
```

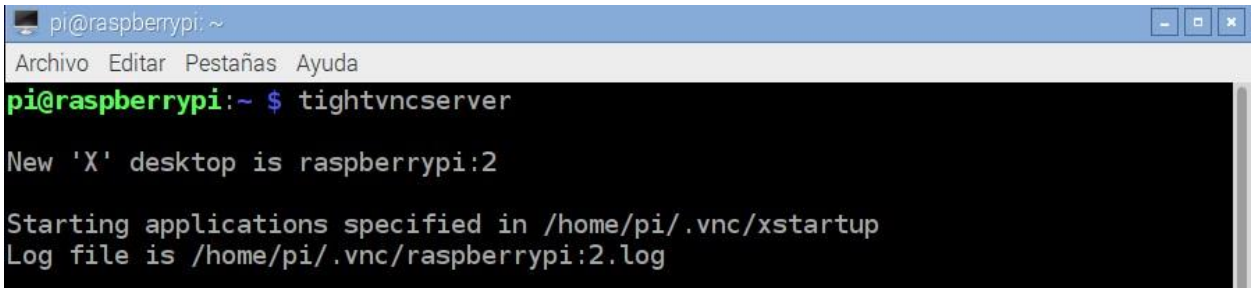

Ilustración 4-55 TightVncServer

Ahora abrimos el *TightVNC* del ordenador e introducimos la IP de la red en común y el puerto que nos ha salido por pantalla en la ilustración anterior, en mi caso 2:

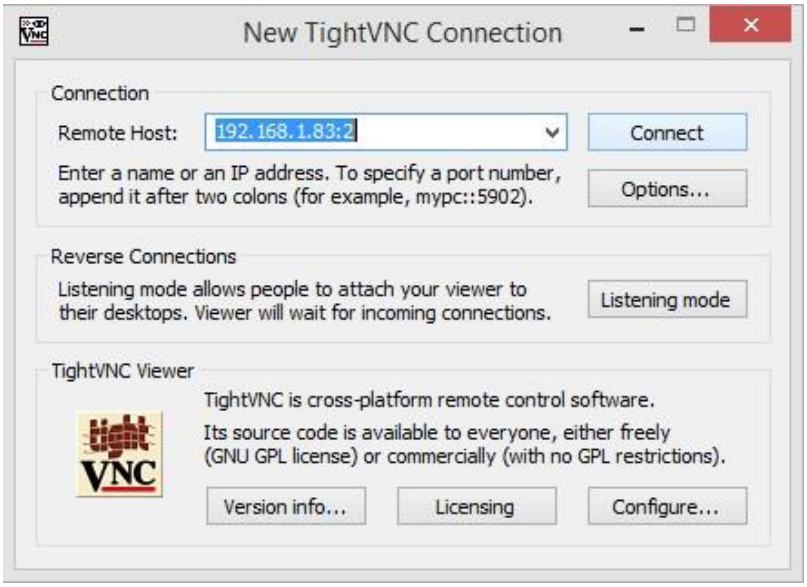

Ilustración 4-56 TightVNC connection

Introducimos la contraseña previamente elegida en la descarga:

Ilustración 4-57 Contraseña de TightVNC

Ahora aparecerá una ventana llamada *TightVNC Viewer* en la cual ya podremos controlar la Raspberry Pi desde nuestro ordenador sin necesidad de una conexión en serie como un cable Ethernet.

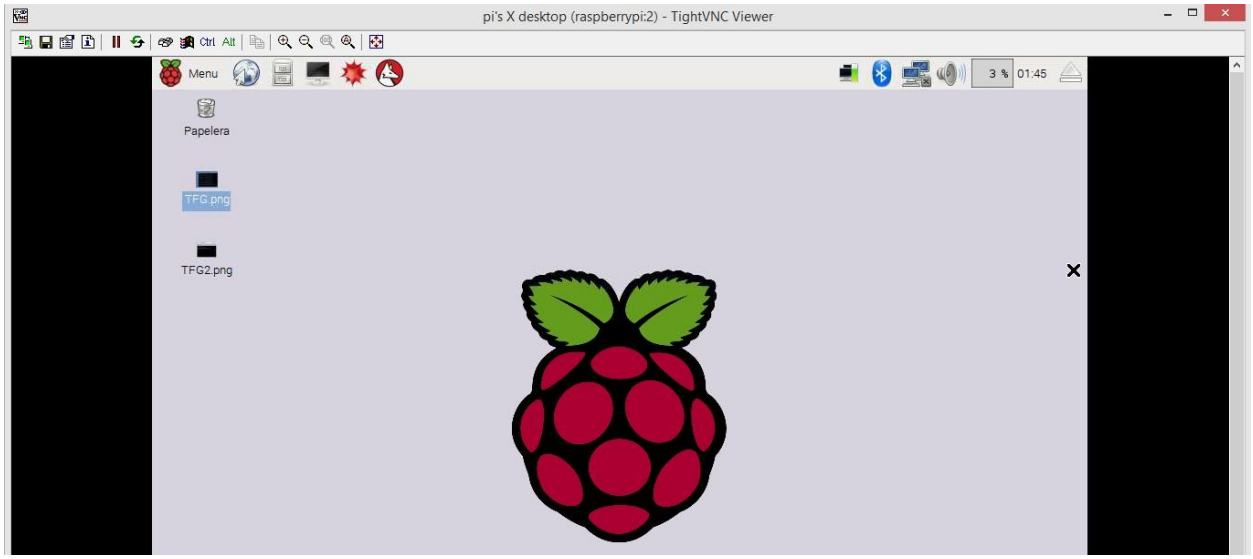

Ilustración 4-58 Resultado satisfactorio de la conexión

4.2.4 Configuración de la Raspberry como punto *Hot-Spot*

El que la raspberry generase su propia red Wi-fi y que con nuestro ordenador nos conectásemos a ella, era una idea de la que éste proyecto podría hacer un buen uso, por lo que se llevó a cabo la configuración de la siguiente manera:

-La Raspberry Pi al recibir alimentación, si está en el rango de alguna red Wi-Fi conocida, se conectará a ella, mientras que, en caso contrario, de no conocer ninguna red Wi-Fi, actuará como punto de acceso, y creará su propia red Wi-Fi a la que nos conectaremos a posteriori.

Para realizar dicha configuración se siguieron los siguientes pasos [23]:

-Descargar los siguientes paquetes:

```
sudo apt-get install dnsmasq hostapd
```

-Configurar las *interfaces*:

```
sudo nano /etc/network/interfaces
```

En la parte de *wlan0* pegar los siguientes comandos:

```
allow-hotplug wlan0
iface wlan0 inet static
    address 172.24.1.1
    netmask 255.255.255.0
    network 172.24.1.0
    broadcast 172.24.1.255
#    wpa-conf /etc/wpa_supplicant/wpa_supplicant.conf
```

-Ejecutar los siguientes comandos en la terminal:

```
sudo service dhcpcd restart
sudo ifdown wlan0; sudo ifup wlan0
```

-Configurar *HostApd:*

```
sudo nano /etc/hostapd/hostapd.conf
```

```
# This is the name of the WiFi interface we configured above
interface=wlan0
# Use the nl80211 driver with the brcmfmac driver
driver=nl80211
# This is the name of the network
ssid=Pi3-AP
# Use the 2.4GHz band
hw_mode=g
```

```
# Use channel 6
channel=6
# Enable 802.11n
ieee80211n=1
# Enable WMM
wmm_enabled=1
# Enable 40MHz channels with 20ns guard interval
ht_capab=[HT40][SHORT-GI-20][DSSS_CCK-40]

# Accept all MAC addresses
macaddr_acl=0
# Use WPA authentication
auth_algs=1
# Require clients to know the network name
ignore_broadcast_ssid=0
# Use WPA2
wpa=2
# Use a pre-shared key
wpa_key_mgmt=WPA-PSK
# The network passphrase
wpa_passphrase=raspberry
# Use AES, instead of TKIP
rsn_pairwise=CCMP
```

Ahora ya podemos crear dicha conexión wifi, pero como queremos que se cree nada mas recibir la alimentación sin ejecutar ningún código manualmente hay que realizar los siguientes pasos:

```
sudo nano /etc/default/hostapd
```

Reemplazar: `#DAEMON_CONF="" ← DAEMON_CONF="/etc/hostapd/hostapd.conf"`

-Configurar DNSmasq:

```
sudo mv /etc/dnsmasq.conf /etc/dnsmasq.conf.orig
sudo nano /etc/dnsmasq.conf
```

-Pegar el siguiente código:

```
interface=wlan0      # Use interface wlan0
listen-address=172.24.1.1 # Explicitly specify the address to listen on
bind-interfaces      # Bind to the interface to make sure we aren't sending things elsewhere
server=8.8.8.8       # Forward DNS requests to Google DNS
```

```
domain-needed        # Don't forward short names
bogus-priv           # Never forward addresses in the non-routed address spaces.
dhcp-range=172.24.1.50,172.24.1.150,12h # Assign IP addresses between 172.24.1.50 and 172.24.1.150
with a 12 hour lease time
```

- Configuracion del Protocolo IPV4:

```
sudo nano /etc/sysctl.conf
net.ipv4.ip_forward=0  net.ipv4.ip_forward=1
```

-Pasos finales:

```
sudo sh -c "echo 1 > /proc/sys/net/ipv4/ip_forward"
sudo nano /etc/rc.local
```

Adjuntar encima de *exit0:*

```
iptables-restore < /etc/iptables.ipv4.nat
sudo service hostapd start
sudo service dnsmasq start
```

Solo queda reiniciar la Rpi y comprobar que todo es correcto con el siguiente comando:

```
sudo reboot
```

Como podemos observar en la siguiente ilustración, hemos conseguido crear dicha red Wi-Fi

Ilustración 4-59 Conexión Wi-Fi Rpi3

4.2.5 Conexión Remota Ordenador-Raspberry Pi

Una vez se ha llevado a cabo de forma satisfactoria los apartados anteriores, se puede proceder a realizar este.

Para empezar se le conecta a la Raspberry una fuente de alimentación, como no estará en rango ninguna red Wi-Fi conocida (ya que el dron lo volaremos en sitios alejados del núcleo urbano), actuará como punto de acceso y generará su propia red wifi a la que nos conectaremos con nuestro ordenador, tal y como se ve en la Ilustración 4-59.

A través de la aplicación *Putty* podremos acceder de forma remota de la siguiente manera:

Ya que la IP que hemos configurado para la Rpi (la cual es estática) es 172.24.1.1, la introducimos en dicho programa tal y como se ve en la Ilustración 4-60

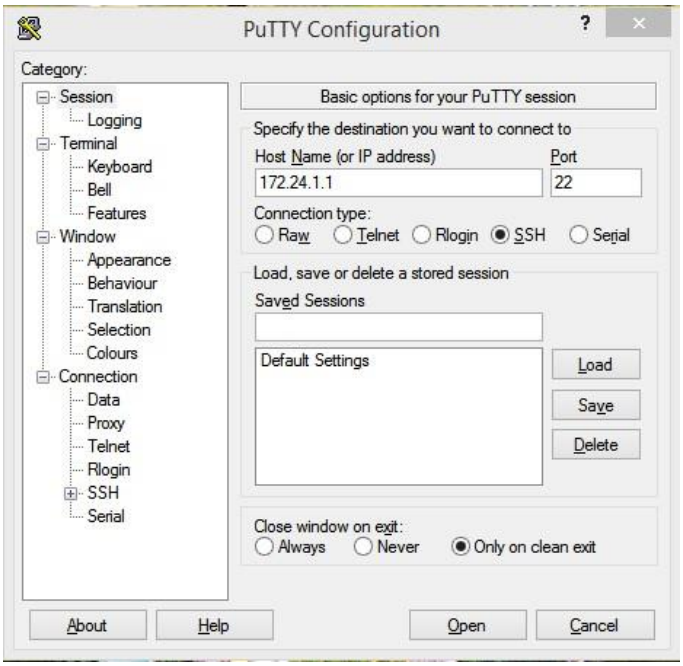

Ilustración 4-60 IP en Putty

Ahora introducimos el usuario y contraseña (los cuales no hemos cambiado, así que serán los predeterminados):

Usuario: pi

Contraseña: raspberry

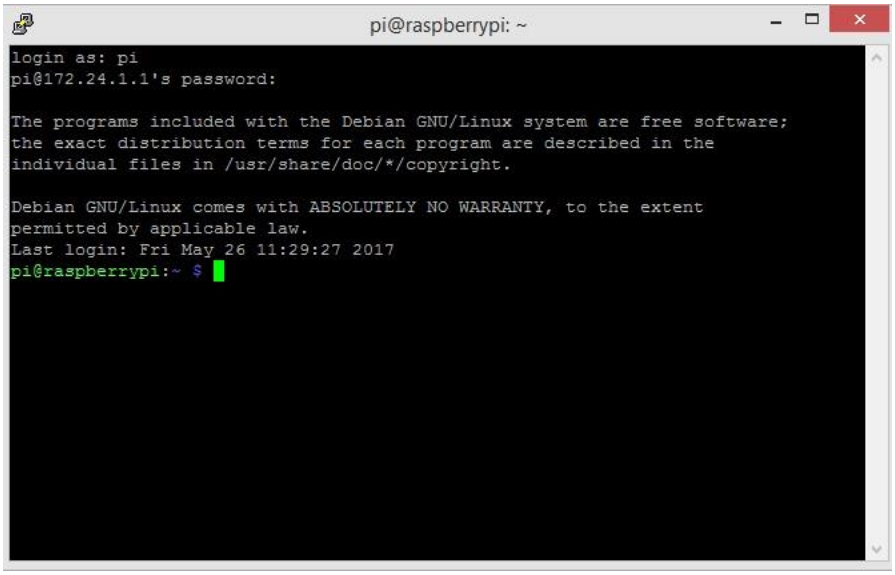

Ilustración 4-61 Usuario y Contraseña en Putty

Gracias a *Putty* únicamente podemos ejecutar comandos y hacer uso de la terminal del sistema, sin embargo, si hacemos uso del *TighVNCserver* podremos controlar y visionar desde nuestro portátil como si estuviese conectada a un monitor (es decir seguir la parte final del apartado 4.2.1).

Para resumir, una vez llegado a este punto, somos capaces de que la Raspberry Pi cree una red local Wi-Fi al encenderse, a la cual nos conectaremos con un portátil para poder controlarla de forma remota sin tener que hacer uso de cables, monitor, teclado, ratón ni red Wi-Fi disponible.

5 APLICACIÓN

Este capítulo versa sobre los desarrollos que se han realizado para poder llevar a cabo el cumplimiento de los objetivos. Por un lado, se incluyen la descripción y utilidad de los diferentes scripts que se hicieron para poder satisfacer el problema descompuesto y al final del capítulo se realiza una descripción del script final, el cual engloba y cumple los requisitos de todas las partes en las que se descompuso el problema.

Recalcar que en este capítulo se encuentran explicaciones acerca de los distintos scripts realizados, mientras que dichos scripts se encuentran al final del documento en el *Anexo de Scripts*.

5.1 Metodologia Empleada

Para poder llevar a cabo una aplicación que cumpliese los objetivos establecidos fue necesaria una descomposición del problema en distintas partes para que nuestro procedimiento fuera más eficiente. La descomposición que se llevó a cabo fue la siguiente:

- Llevar a cabo la comunicación bidireccional entre la controladora APM y la Raspberry.

- Envío por parte de la APM y recepción de la Raspberry de distintos datos (longitud, latitud, altura…).

- Creación de un documento .txt que almacene dichos datos durante el curso de una misión.

- Toma de fotografías por parte de la Raspberry y almacenamiento de las mismas en una carpeta destino.

- Unificación de todo lo anterior en un único código.

Este apartado del proyecto se llevó a cabo haciendo uso de una librería llamada *DroneKIT* [24], especializada en interactuar con las controladoras de los drones.

5.2 Comunicación Bidireccional

Esta fue la parte más crítica de la aplicación, la cual sufrió varias modificaciones, ya que el conexionado entre APM-Rpi (Capitulo 4.3) empezó siendo mediante los puertos GPIO y acabo realizándose mediante los puertos USB-microUSB.

Seleccionando de entre las opciones posibles la que se adecua a nuestra configuración, se completará esta parte del problema.

Connection type	Connection string
Linux computer connected to the vehicle via USB	`/dev/ttyUSB0`
Linux computer connected to the vehicle via Serial port (RaspberryPi example)	`/dev/ttyAMA0` (also set `baud=57600`)
SITL connected to the vehicle via UDP	`127.0.0.1:14550`
SITL connected to the vehicle via TCP	`tcp:127.0.0.1:5760`
OSX computer connected to the vehicle via USB	`dev/cu.usbmodem1`
Windows computer connected to the vehicle via USB (in this case on COM14)	`com14`
Windows computer connected to the vehicle using a 3DR Telemetry Radio on COM14	`com14` (also set `baud=57600`)

Ilustración 5-1 Conexionado del Script

Aunque esta conexión fue satisfactoria, se decidió escribir otra ruta dado que se detectó que al conectar la APM a la Rpi se creaba siempre una nueva carpeta de la siguiente forma:

Donde estaba escrito en el código `'/dev/ttyUSB0'` se sustituyó por lo siguiente:

```
'/dev/serial/by-id/usb-Arduino__www.arduino.cc__Arduino_Mega_2560_7403931363235170A162-if00'
```

El código que hace la comunicación posible se recoge en el script denominado:

Conecction_and_Information_Transfer.py

5.3 Transmisión de información situacional del Dron

Para esta parte del proceso fue necesario establecer una correcta conexión para que fuese posible el trasvase de información entre ambos componentes, es decir, que era indispensable haber realizado a priori lo explicado en el punto 5.2.

Este apartado se basó en la solicitud de ciertos parámetros por parte de la Rpi y la transmisión de los mismos por parte de la APM, para ver si eran efectivos se programó para que mostrase dichos parámetros por pantalla.

El código del que se hace referencia, es el mismo que en el apartado anterior por lo explicado anteriormente.

5.4 Creación de un documento .txt durante la misión

Para crear la base de datos, sobre la que irá escribiendo la Raspberry Pi según reciba los datos demandados, se eligió que el método más visual y sobre todo más eficiente, era mediante la creación de un documento .txt.

Ya que la tarea a realizar es la escritura sobre dicho archivo, nuestra siguiente decisión fue la de ser elegir cuando queríamos que ésto se llevase a cabo, es decir bajo qué circunstancias queremos que se produzca la escritura y cada cuanto tiempo. La decisión tomada fue que se escribiera en el archivo de texto mientras el dron estuviese armado, con 5 segundos entre cada escritura y con un máximo de 6 entradas (ya que se estimó que los vuelos iban a ser de corto periodo). Para la realización de este script se hizo uso de [25].

El código al que se hace referencia se ha denominado *Writing_on_a_LOG_Streamly.py*

5.5 Toma de fotografías

Dado que el tipo de cámara que empleamos (PiCamera V2 Ilustración 4-50), está preparada para trabajar con la Raspberry Pi la creación del código resultó bastante sencilla.

Dado que queríamos que realizase fotos a la vez que se realizaba la toma de datos (Capitulo 5.4) para que cada foto tuviese asignada una coordenada GPS, se eligió el mismo criterio para determinar cuándo se produciría la toma de fotografías. Dichas fotos se almacenarán en una carpeta especificada en el código, de modo que cada foto llevará su nombre acorde a su posición en la base de datos (Image***nº***).

El código al que se hace referencia se ha denominado *Taking_and_Saving_Pictures.py*

5.6 Unificación de Códigos y el Código Final

Una vez cumplidos todas las partes en las que se descompuso la aplicación, hubo que realizar una unificación de todos los programas con lógica y de manera que cuando se realizase la captura fotográfica, se realizase la escritura en el archivo.

El resultado final se encuentra en el código denominado *Photo_and_DataBaseRecorder.py* y funciona de la siguiente manera:

-Se importan todos los módulos necesarios.

-Se establece la conexión entre la APM y la Raspberry Pi

-Se espera a que el dron se arme (se realizará de forma manual)

-Una vez que el dron está armado se crea el archivo .txt sobre el que se escribirá a posteriori.

-A partir de este punto, se realizará una fotografía y se escribirá en el mismo momento en el archivo .txt las coordenadas GPS hasta el momento de que se realicen 6 entradas o que el dron se desarme, lo que suceda antes.

-Se cierra la conexión y se da por terminada la misión.

6 RESULTADOS

En este capítulo se van a presentar por un lado los resultados obtenidos al ejecutar la aplicación, así como del procedimiento que se siguió para realizar las pruebas en vuelo. Al final del capítulo se muestra la misión que se llevó a cabo gracias a un post-procesado con la información de la base de datos.

Para poder verificar que la aplicación creada funcionaba tal y como se había programado, hubo que realizar una serie de pruebas en vuelo haciendo uso del dron que se ensambló (Capitulo 4.1). Para poder realizar dichas pruebas nos trasladándonos a una zona lo bastante alejada del núcleo urbano.

El procedimiento seguido fue el siguiente:

-Conectarle la batería al quadrotor (y por tanto a la Raspberry) por lo que ésta generara una red Wi-Fi a la que nos conectaremos desde el portátil.

-Conexión remota PC-Rpi (tal y como se explica al final del Capítulo 4.3).

-Una vez que ya visionamos la Raspberry y la podemos controlar desde nuestro teclado y pantalla, ejecutamos el script *Photo_and_DataBaseRecorder.py*.

```
>>> APM:Copter V3.3 (d6053245)
>>> Frame: QUAD
>>> Calibrating barometer
>>> Initialising APM...
>>> barometer calibration complete
>>> GROUND START
Connected
 Waiting for arming...
Dron Armed
LOG file open
Making the Picture...
Saving the Picture...
Taking Data
We are writing on the LOG
Making the Picture...
Saving the Picture...
Taking Data
```

Ilustración 6-1 Resultados de la Aplicación en la terminal_1

```
We are writing on the LOG
Making the Picture...
Saving the Picture...
Taking Data
We are writing on the LOG
Making the Picture...
Saving the Picture...
Taking Data
We are writing on the LOG
Making the Picture...
Saving the Picture...
Taking Data
We are writing on the LOG
The mission has concluded
```

Ilustración 6-2 Resultados de la Aplicación en la terminal_ 2

Como se puede apreciar, una vez que está armado el dron empieza a hacer fotos y a tomar datos hasta el momento que cumple el número de fotos establecido.

-Comprobación de que se han creado correctamente las fotos y el archivo .txt

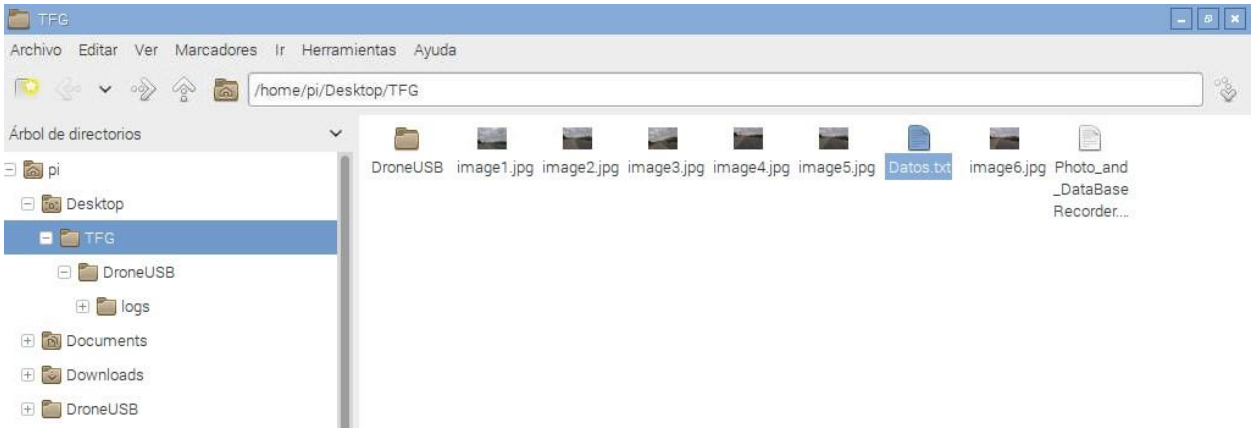

Ilustración 6-3 Comprobación de la Creación de Archivos

A continuación, se muestran fotografías tomadas durante la prueba de vuelo:

Ilustración 6-4 Puesta a punto Pre-Vuelo

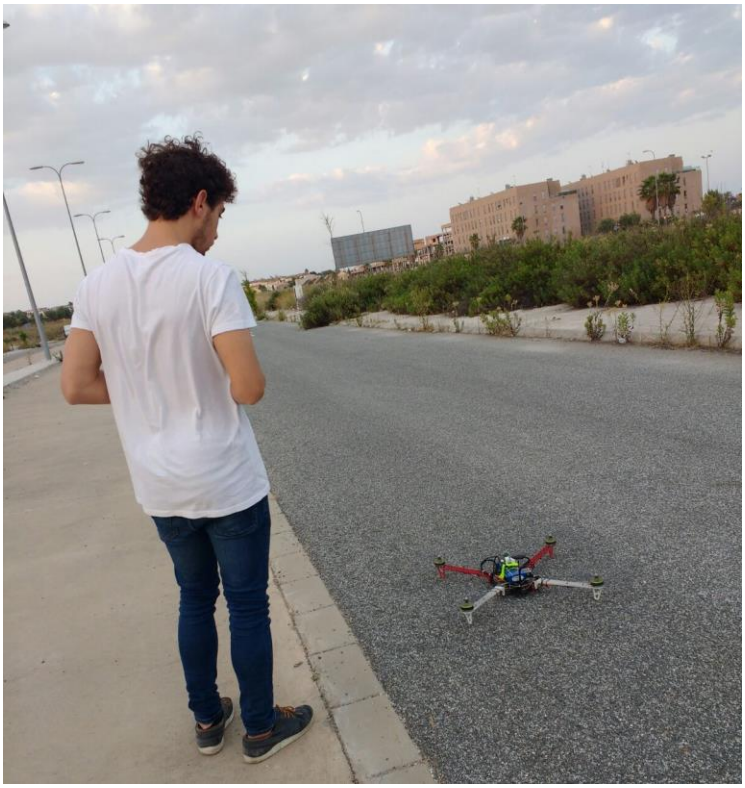

Ilustración 6-5 Armado del Dron

Ilustración 6-6 Prueba en Vuelo

Tal y como se explicó antes, al terminar la prueba en vuelo se comprobó que las fotografías se habían realizado con éxito. A continuación, se muestran algunas de las fotos realizadas por el dron:

Ilustración 6-7 Captura del Dron al ser armado

Ilustración 6-8 Captura del Dron durante la misión

Una vez comprobadas las fotografías, se pasó a analizar el documento .txt creado, el cual se muestra a continuación:

DATA BASE:

Quadrotor Position:	LocationGlobal: lat=37.347423, lon=-6.071893, alt=84.25	Photo number 1
Quadrotor Position:	LocationGlobal: lat=37.347407, lon=-6.071910, alt=85.13	Photo number 2
Quadrotor Position:	LocationGlobal: lat=37.347353, lon=-6.071925, alt=89.34	Photo number 3
Quadrotor Position:	LocationGlobal: lat=37.347382, lon=-6.071987, alt=88.81	Photo number 4
Quadrotor Position:	LocationGlobal: lat=37.347414, lon=-6.071985, alt=86.85	Photo number 5
Quadrotor Position:	LocationGlobal: lat=37.347409, lon=-6.071943, alt=88.45	Photo number 6

Como podemos observar, a cada fotografía se le ha asignado una coordenada GPS, por lo que se ha podido realizar un postprocesado para poder representar la misión que el quadrotor siguió:

Ilustración 6-9 Representación de la Mision_1

Ilustración 6-10 Representación de la Misión_2

7 CONCLUSIÓN Y DESARROLLOS FUTUROS

Una vez se ha terminado de exponer los resultados de la aplicación, se va a proceder, por un lado a explicar las conclusiones a las que se ha llegado al llevar a cabo el proyecto, además de los posibles desarrollos futuros, es decir aplicaciones que se podrían realizar a partir de este trabajo. En la parte final del capítulo se definen las limitaciones que se han encontrado a la hora de llevar a cabo el proyecto.

7.1 Conclusiones

Al principio del documento se establecieron una serie de objetivos a cumplir, y en base a dichos objetivos se realizó una planificación para poder llegar a satisfacerlos. Para ello, se ha llevado a cabo, por un lado, la creación de una aplicación, que realiza una base de datos a tiempo de real con fotografías y coordenadas GPS, y por otro lado, el montaje y la puesta a punto de un quadrotor para hacer las pruebas en vuelo de dicha aplicación.

Una vez llevado a cabo dichos procesos y la prueba final en vuelo, tal y como se puede observar en los resultados (Capitulo 6), se han podido satisfacer todos los objetivos del Trabajo Fin de Grado y da pie a que este trabajo pueda ser utilizado en siguientes proyectos. A continuación, se explican posibles desarrollos que podrían llevarse a cabo a raíz de este.

7.2 Desarrollos Futuros

Como se explicó al principio del documento, el empleo de los drones en el entorno fotográfico es considerablemente amplio, siendo por tanto la fotografía el objetivo de la gran mayoría de las aplicaciones actualmente.

Por un lado, la prueba en vuelo se realizó de forma manual, es decir fue una persona la que controló el dron de forma remota mediante una emisora lo cual tiene las siguientes desventajas:

- Se necesita de una persona controlando el dron constantemente, lo cual reduciría la eficiencia y la productividad en cualquier sector, ya que esa persona podría estar realizando otra labor.
- Siendo una persona la que controla dicho dron, puede incurrir en numerosos errores debido a la imprecisión humana.

En cambio, si la misión que se llevó a cabo de forma manual se realizase de forma automática tendría las siguientes ventajas:

- No se tendría que hacer uso de un ordenador para ejecutar el programa, ni de la emisora para realizar el control, por lo cual el proceso seria menos aparatoso.
- Sería más eficiente y eficaz el proceso ya que la persona que antes controlaba el dron podría estar, por ejemplo, monitorizando el vuelo a la vez que trabaja en otros asuntos.

Por todo esto, un posible desarrollo futuro sería el de realizar una programación para que el vuelo se hiciese de forma automática (siguiendo una serie de waypoints) a la vez que realiza fotos y escribe en una base de datos.

De la forma en que se ha realizado el proyecto, la única forma de acceder a las fotos y a la base de datos es una vez la misión ha concluido, tomando la Raspberry pi y extrayendo los datos de ella. Por lo que un posible desarrollo podría ser el de que la Raspberry a la vez que almacena la información que recibe, que mandase dicha información a un servidor web al que se podría acceder de forma remota a tiempo real.

7.3 Limitaciones del Proyecto

Tal y como se empieza introduciendo en el Capítulo 2, una de las grandes limitaciones que sufren los UAVs, y por tanto nuestro proyecto es la legislativa. Por culpa de estas limitaciones, aun siendo la diversidad de aplicaciones bastante amplia, provoca que numerosas buenas ideas para aplicaciones no se lleven a cabo dado que su empleo sería ilegal o no sería rentable debido a las restricciones existentes.

A continuación, se resumen algunos de los requisitos para poder realizar el vuelo de un dron según un Real Decreto-Ley publicado en el BOE con fecha del 5 de Julio de 2014 [26] :

- Requisitos del Piloto:

  -Seguro y/o garantía que cubra la responsabilidad civil sobre posibles terceros.

- Certificado avanzado para el pilotaje de aeronaves civiles pilotadas por control remoto emitido conforme al anexo VII del Reglamento (UE) nº. 1178/2011.

- Certificado médico de habilitación para control de drones según el apartado MED.B.095 del anexo IV, Parte MED, del Reglamento (UE) n.º 1178/2011.

-Acreditar que posee los conocimientos teóricos necesarios por medio de un certificado básico o avanzado emitido por una organización de formación aprobada (ATO) tras superar un curso al efecto, de acuerdo al Ministerio de Fomento.

- Requisitos para el Vuelo:

  - Se debe volar durante el día y bajo las Condiciones Meteorológicas Visuales.

  - Prohibido volar sobre núcleos urbanos ni sobre reuniones de personas.

  -Permiso de vuelo de la AESA. Para obtenerlo es necesario realizar la solicitud con 5 días de antelación y debe incluir las aeronaves en cuestión, datos del piloto, especificaciones de las aeronaves, objetivo de la operación y plan de seguridad.

Con respecto a estas limitaciones, tal y como se dijo en el capítulo 2, hay unas expectativas muy positivas ya que debido a que los avances producidos en este sector, próximamente se tendrá que realizar un proyecto de cambio de ley para poder así regular de manera correcta el uso de vehículo aéreos no tripulados en los tiempos actuales.

Con respecto a las limitaciones del tipo hardware, tenemos por un lado que los motores no eran demasiado potentes y además el chasis empleado no era muy rígido, provocando un pandeo en el centro del chasis (debido a toda la electrónica que llevaba embarcada) lo que provocaba muchas vibraciones y por tanto inestabilidades. Por culpa de esta inestabilidad se descartó realizar el vuelo de forma automática, ya que se necesitaban numerosas correcciones para cualquier tipo de colisión.

Por otro lado, si en vez de una PiCameraV2 pudiésemos disponer de una cámara de alta resolución con tecnología NDVI (Normalized Difference Vegetation Index, explicado con detalle en el Capítulo 3), se podría realizar una infinidad de aplicaciones como, por ejemplo:

- Crear una base de datos de las zonas de la plantación que se encuentren en peor estado.
- Mapeo de una zona en 3D para soporte cartográfico.

ANEXO DE SCRIPTS

A continuación, se muestran los distintos códigos de los que se hace referencia en el Capítulo 7:

Conecction_and_Information_Transfer

```
"""
@author: José Ramos Gálvez


"""

# Import DroneKit-Python
from dronekit import connect, VehicleMode

import time


# Connection with the APM
serial_port_apm = '/dev/serial/by-id/usb-
Arduino__www.arduino.cc__Arduino_Mega_2560_74039313632351 70A162-if00'
print 'Connecting to vehicle on: %s' % serial_port_apm
vehicle = connect(serial_port_apm, wait_ready=True)
time.sleep(5)

print 'Connected'


print " Global Location (relative altitude): %s" %
vehicle.location.global_relative_frame

print "Closing the connection.."
vehicle.close()
sys.exit()
```

Writing_on_a_LOG_Streamly

```
"""
@author: José Ramos Galvez
"""
#Once the connection is made and the dron is armed

#Drone Armed, opening the LOG file LOG = open('Data_Base.txt','w')
LOG.close()
print 'LOG file open'
LOG = open('Data_Base.txt','a')
LOG.write('DATA BASE:\n\n')
LOG.close()


i=0
while(str(vehicle.armed) == 'True'):
    while i<= 6:
        i=i+1

        print 'Taking Data'
        Local_Position = str(vehicle.location.global_frame)
        time.sleep(1)
        print 'We are writing on the LOG'
        LOG = open('Data_Base.txt','a')
        LOG.write('Quadrotor position: ')
        LOG.write(Local_Position)
        LOG.write('    ')

        LOG.write('\n')
        LOG.close()

        time.sleep(4)


print "Closing the connection.."
vehicle.close()
```

Taking_and_Saving_Pictures

```
"""
@author: José Ramos Galvez
"""

# Import Camera
from picamera import PiCamera

camera = PiCamera()

#Once the connection is made and the dron is armed

#Run the Camera
camera.start_preview()

i=0
while(str(vehicle.armed) == 'True'):
    while i<= 6:
        i=i+1
        print 'Making the Picture...'

        camera.capture('/home/pi/Desktop/TFG/DroneUSB/image%s.jpg' % i)
        time.sleep(1)
        print 'Saving the Picture...'

        time.sleep(4)


#Swich off the Camera
camera.stop_preview()

print "Closing the connection.."
vehicle.close()
```

Photo_and_DataBaseRecorder

```
"""
@author: José Ramos Gálvez
"""

# Import DroneKit-Python
from dronekit import connect, VehicleMode
import sys
import time

# Import Camera
from picamera import PiCamera

camera = PiCamera()

# Connection with the APM
serial_port_apm = '/dev/serial/by-id/usb-
Arduino__www.arduino.cc__Arduino_Mega_2560_7403931363235170A162-if00'
print 'Connecting to vehicle on: %s' % serial_port_apm
vehicle = connect(serial_port_apm, wait_ready=True)
time.sleep(5)

print 'Connected'


# Waiting for arming the drone to start the LOG
while (str(vehicle.armed) == 'False'):
    a = 1


print 'Dron Armed'

#Drone Armed, opening the LOG file
LOG = open('Data_Base.txt','w')
LOG.close()
print 'LOG file open'
LOG = open('Data_Base.txt','a')
LOG.write('DATA BASE:\n\n')
LOG.close()



#Run the Camera
camera.start_preview()

#Inicialising Variables
i=0


while(str(vehicle.armed) == 'True'):
    while i<= 6:
        i=i+1
        print 'Making the Picture...'
        camera.capture('/home/pi/Desktop/TFG/DroneUSB/image%s.jpg' % i)
        time.sleep(1)
        print 'Saving the Picture...'

        print 'Taking Data'
        Local_Position = str(vehicle.location.global_frame)

        print 'We are writing on the LOG'
        LOG = open('Data_Base.txt','a')
        LOG.write('Quadrotor position: ')
        LOG.write(Local_Position)
```

```
        LOG.write('    ')
        LOG.write('Photo number %s' % i)
        LOG.write('\n')
        LOG.close()

        time.sleep(4)


#Swich off the Camera
camera.stop_preview()

print "The mission has concluded"
vehicle.close()
sys.exit()
```

REFERENCIAS

[1] Think Up LKS. (2017). *Los drones en el ámbito empresarial, un mercado por explotar - Think Up LKS*. [online] Available at: http://www.thinkuplks.com/los-drones-en-el-ambito-empresarial-un-mercado-por-explotar/ [Accessed 18 Mach 2017].

[2] Šarboh, S. (2010). The patents of Nikola Tesla. *World Patent Information*, 32(4), pp.335-339.

[3] Aplicaciones y Operación con Drones-RPAS. (2017). *Origen y desarrollo de los drones*. [online] Available at: http://drones.uv.es/origen-y-desarrollo-de-los-drones/ [Accessed 18 March 2017].

[4] El Drone. (2017). *Historia de los drones - El Drone*. [online] Available at: http://eldrone.es/ [Accessed 19 March 2017].

[5] Richard Whittle (2013) , The Man Who Invented the Predator, Air & Space magazine

[6] P. van Blyenburgh (2006), UAV systems: global review. Presented at the Avionics'06 conference, Amsterdam,

[7] Hassanalian, M. and Abdelkefi, A. (2017). Classifications, applications, and design challenges of drones: A review. *Progress in Aerospace Sciences*.

[8] NATO. (2017). *Speeches & transcripts*. [online] Available at: http://www.nato.int/cps/en/natohq/opinions.htm [Accessed 5 May 2017].

[9] El mundo de los drones en tus manos. (2017). *Tipos de drones - Conoce todos los tipos de drones que existen*. [online] Available at: http://www.xdrones.es/tipos-de-drones-clasificacion-de-drones-categorias-de-drones/ [Accessed 5 May 2017].

[10] Development and operation of UAVs for military and civil applications =. (2000). *1st ed. Neuilly-sur-Seine Cedex, France: North Atlantic Treaty Organization, Research and Technology Organization.*

[11] Roze, A., Zufferey, J. C., Beyeler, A., & McClellan, A. (2014). *eBee RTK accuracy assessment*. White Paper Sense Fly.

[12] Hobbyking. (2017). Radio Control Planes, Drones, Cars, FPV, Quadcopters and more - Hobbyking. [online] Available at: https://hobbyking.com/ [Accessed 8 May 2017].

[13] Hobbyking. (2017). Turnigy Multistar 4822-690Kv 22Pole Multi-Rotor Outrunner. [online] Available at: https://hobbyking.com/es_es/turnigy-multistar-4822-690kv-22pole-multi-rotor-outrunner.html [Accessed 8 May 2017].

[14] Hobbyking. (2017). Z700-V2 Quadcopter Frame White/Red With Crab Landing Gear (700mm) V2. [online] Available at: https://hobbyking.com/en_us/z700-v2-quadcopter-frame-white-red-with-crab-landing-gear-700mm-v2.html?___store=en_us [Accessed 8 May 2017].

[15] Hobbyking. (2017). Controlador de velocidad de 30amp felpa TURNIGY w / BEC. [online] Available at: https://hobbyking.com/es_es/turnigy-plush-30amp-speed-controller.html [Accessed 8 May 2017].

[16] B, R. (2017). Raspberry Pi 3 Modelo B. [online] Pccomponentes.com. Available at: https://www.pccomponentes.com/raspberry-pi-3-modelo-b [Accessed 9 May 2017].

[17] V2, R. (2017). Raspberry Pi Cámara V2 |PcComponentes. [online] Pccomponentes.com. Available at: https://www.pccomponentes.com/raspberry-pi-camara-v2 [Accessed 9 May 2017].

[18] Hobbyking. (2017). Turnigy 9X 2,4 GHz 8Ch receptor (V2). [online] Available at: https://hobbyking.com/es_es/turnigy-9x-2-4ghz-8ch-receiver-v2.html [Accessed 11 May 2017]

[19] Arduino based Arducopter UAV, the open source multi-rotor. (2017). Connecting everything forArducopter. [online] Available at: http://www.arducopter.co.uk/all-arducopter-guides/2connectingeverything-for-arducopter [Accessed 11 May 2017].

[20] Arduino based Arducopter UAV, the open source multi-rotor. (2017). Iris Advanced Manual. [online] Available at: http://www.arducopter.co.uk/advanced.html [Accessed 12 May 2017].

[21] Mészárosová, E. (2015). Is Python an Appropriate Programming Language for Teaching Programming in Secondary Schools?. *International Journal of Information and Communication Technologies in Education*, 4(2).

[22] El mundo de los drones en tus manos. (2017). *MAVLink: protocolo de comunicación para drones*. [online] Available at: http://www.xdrones.es/mavlink/ [Accessed 13 May 2017].

[23] Frillip's Blog. (2017). *Using your new Raspberry Pi 3 as a WiFi access point with hostapd*. [online] Available at: https://frillip.com/using-your-raspberry-pi-3-as-a-wifi-access-point-with-hostapd/ [Accessed 6 Jun. 2017].

[24] DroneKit. (2017). *DroneKit by 3D Robotics*. [online] Available at: http://dronekit.io/ [Accessed 7 Jun. 2017].

[25] Diseño, montaje y puesta en marcha de vehículos aéreos multirrotor para su uso en redes de comunicaciones aéreas. (2016). 1st ed. Sevilla: Sergio Vigorra Treviño.

[26]Real Decreto regulacion civil de aeronaves tripuladas por control remoto). [online] Available at: https://www.fomento.gob.es/NR/rdonlyres/63ECAE3A-B29E-45A7-A885-D314153883EE/139826/RDRPAS27102016.pdf [Accessed 8 Jun. 2017].

ÍNDICE DE ILUSTRACIONES

ÍNDICE DE TABLAS