UNIVERSIDAD DE CASTILLA-LA MANCHA
ESCUELA SUPERIOR DE INFORMÁTICA

GRADO EN INGENIERÍA INFORMÁTICA

TRABAJO FIN DE GRADO

Transporte aéreo de mercancías de corta distancia basado en UAV's

Daniel Hurtado Cárdenas

Julio, 2015

TRANSPORTE AÉREO DE MERCANCÍAS DE CORTA DISTANCIA BASADO EN UAV'S

UNIVERSIDAD DE CASTILLA-LA MANCHA

ESCUELA SUPERIOR DE INFORMÁTICA

Tecnologías y Sistemas de Información

TECNOLOGÍA ESPECÍFICA DE
INGENIERÍA DE COMPUTADORES

TRABAJO FIN DE GRADO

Transporte aéreo de mercancías de corta distancia basado en UAV's

Autor: Daniel Hurtado Cárdenas

Director: Dr. David Villa Alises

Julio, 2015

Daniel Hurtado Cárdenas

La Mancha – Spain

E-mail: Daniel.Hurtado@alu.uclm.es
Teléfono: 600 382 692

TRIBUNAL:

Presidente:

Vocal:

Secretario:

FECHA DE DEFENSA:

CALIFICACIÓN:

PRESIDENTE	VOCAL	SECRETARIO
Fdo.:	Fdo.:	Fdo.:

Resumen

Este documento es un trabajo fin de grado de ingenieria informática de la especialidad ingenieria de computadores, que versa sobre el transporte aéreo de mercancias de corta distancia basado en UAV. Este proyecto está dividido en 3 partes, como son la fabricación de un UAV desde cero adaptandose a nuestras necesidades de transporte. El desarrollo de una estación terrena, la cual nos permite establecer una conexión con la aeronave, monitorizar su estado y controlar todos sus parametros. Y por último, el desarrollo de una aplicación web para la realización de los pedidos por el cliente.

Aspectos reseñables son el tiempo empleado en la investigación de los antecedentes para poder tomar la mejor decisión a la hora de elegir los diferentes componentes del sistema. El uso de herramientas libres, librerias y paquetes de desarrollo con licencias que facilitan el acceso al código y su adaptación a otros proyectos. Su modularidad, que nos permite reutilizar ciertas partes. Facilidad de uso. Adaptación a diferentes entornos, escala muy facil y eficiente. En lo que se refiere a hardware, se han tenido los siguientes requisitos como son un bajo consumo, alto rendimiento y bajo coste. Para su diseño y desarrollo.

Abstract

This document is a work order degree of computer engineering specialty Computer Engineering, which deals with the carriage of goods based on short-haul UAV. This project is divided into 3 parts, such as the manufacture of a UAV from scratch adapting to our transportation needs. The development of an earth station, which allows us to connect to aircraft, monitor their status and control all its parameters. And finally, the development of a web application for the realization of customer orders.

Noteworthy aspects are the time spent in the vetting to make the best decision when choosing the various components system. The use of free tools, libraries and development packages with licenses facilitating access to the code and its adaptation to other projects. Its modularity, that allows us to reuse parts. Easy to use. Adaptation to different environments, very easy and efficient scale. As regards hardware, they have been the following requirements such as low power consumption, high performance and low cost. For design and development.

Agradecimientos

Después de más de 8000 km recorridos en bici + 3000 km en coche en 5 años, 240 créditos superados, más 12 de inglés, 38 asignaturas y un TFG, con unos aumentos desproporcionados de la matrícula, puedo decir estar a punto de completar el Grado en Ingeniería Informática por la Universidad de Castilla La Mancha. Han sido muchos los esfuerzos realizados, pero que sin la ayuda de mi familia no hubiese sido posible. Gracias a mi Ivi y mi hijo, que en los momentos más difíciles a punto de desvanecer han estado ahí para apoyarme y darme fuerzas. ¿Y que decir de mis pequeños dioses? mi padre y mi madre, Don Primitivo y Emilia, mis maestros. Tampoco quiero olvidarme de mis hermanas y compañeros. Y un profesor especial, el cual ha hecho que sea más llevadero este infierno. No ha habido día que le diera el coñazo y no me echara una mano, me refiero al doctor David Villa Alises. No quiero olvidarme de otros grandes profesores como Eduardo Dominguez, María José Santofimia, Luis Jimenez y Macario Polo.

Daniel Hurtado Cárdenas

A mi gran maestro Primitivo Hurtado

Índice general

Índice de cuadros

Índice de figuras

Índice de listados

Listado de acrónimos

APM	Placa base de ArduPilot
ARM	Maquina avanzada con Conjunto reducido de instrucciones (RISC)
BEC	Circuito eliminador de bateria
BSON	Notación para Objetos JavaScript Binarios
CEB	Circuito Eliminador de Batería
CPU	Unidad central de proceso
DIY	«Hazlo Tu Mismo»
ESC	Control Electrónico de Velocidad
ESI	Escuela Superior de Informática
FPV	Vista en Primera Persona
GCS	Estación de Control Terrena
GND	Tierra
GPL	Licencia pública general
GPS	Sistema de posicionamiento global
GPU	Unidad de Procesamiento Gráfico
GSM	Sistema global para comunicaciones moviles
I2C	Inter-Circuitos Integrados, bus de comunicación en serie
ICAO	Organización Internacional de Aviación Civil
IMU	Unidad de Medición Inercial
JSON	Notación para Objetos JavaScript
HDL	Lenguaje de Descripción Hardware
HTML	Lenguaje de marcas de hipertexto
HTTP	Protocolo de Transferencia de Hipertexto
LED	Diodo Emisor de Luz
MHz	Mega Hercios
MVC	Modelo Vista Controlador

NTSC	Comisión Nacional de Sistemas de Televión
OSD	Visualización en Pantalla
OSI	Modelo de Interconexión de Sistemas Abiertos
PAL	Linea de fase alterna
PC	Computador personal
PCB	Placa de circuito impreso
PPM	Modulación por posición de pulso
PWM	Modulación por ancho de pulsos
RAM	Memoria de acceso aleatoria
RC	Radio Control
RISC	Conjunto reducido de instrucciones
RSSI	Indicador de fuerza de la señal recibida
SGBD	Sistema de gestión de base de datos
SD	Secure Digital - formato de tarjeta de memoria
SITL	Loop Software de simulación de vehiculos
SO	Sistema Operativo
SPI	Interfaz de comunicación de Periféricos Serie
SQL	Lenguaje de consulta estructurado
SUM	Suma
UART	Transmisor Receptor Asíncrono Universal
UAV	Vehículo aéreo no tripulado (VANT) o dron
UDP	Protocolo de comunicación de datagramas de usuario
UHF	Ultra-High Frequency
URL	Localizador de recursos uniforme
USB	Bus universal serie
V	Voltios
XML	Lenguaje extensible de marcas

Capítulo 1

Introducción

El transporte de mercancías ligeras a corta distancia ofrece muchas posibilidades de mejora y revolución. Si observamos actualmente cómo se realizan estos transportes, vemos que el 95 % de estos trayectos, son vehículos a explosión y terrestres. La posibilidad que nos ofrece la tecnología de realizar estos transportes mediante vehículos aéreos comúnmente llamados UAV es innumerable. Las ventajas son evidentes, ahorro de combustible, personal, infraestructuras, reducción del tiempo de entrega, accesibilidad a cualquier zona. Los inconvenientes actualmente: la legislación española impide su uso en centros urbanos, los propios de aeronaves voladoras. [BOE14]

1.1 Transporte aéreo de mercancías de corta distancia basado en UAV

En un principio este proyecto se basa en el transporte de radio fármacos a cortas distancias. Aunque su utilidad está mucho más halla del transporte de fármacos. La elección de los radio fármacos se debe a dos razones, su corta caducidad, llegando a ser de horas, y su peso ligero; estas dos razones hacen que su transporte a través de UAV sea ideal. Solo cabe destacar el posible riesgo ante un accidente del UAV, hecho tratado en el apartado de seguridad. [Bla14]

1.1.1 Radio Fármacos

Los radio fármacos son medicamentos que contienen materiales radioactivos, llamados radioisótopos. Se pueden administrar por via venosa u oral, o se pueden colocar en una cavidad del cuerpo. Dependiendo del medicamento y de su forma de administración, estos materiales viajan a varias partes del cuerpo para tratar el cáncer o aliviar sus síntomas. Emiten radiación, principalmente en forma de partículas alfa y beta, que se dirige a las áreas afectadas. Se emplean con más frecuencia en pequeñas cantidades para estudios por imágenes, aunque se pueden usar mayores dosis para administrar radiación.

1.1.2 Componentes del sistema

Hardware

- Raspberry Pi es un ordenador de placa reducida o (placa única) de bajo coste desarro-

llado en Reino Unido por la Fundación Raspberry Pi. El diseño incluye un System-on-a-chip Broadcom BCM2835, que contiene un procesador central (CPU) ARM1176JZF-S a 700 MHz, un procesador gráfico (GPU) VideoCore IV, y 512 MB de memoria RAM. Con tarjeta SD de 8 GB y adaptador WIFI.

- Tarjeta Controladora de Vuelo. Microcontrolador CRIUS AIO PRO v2, AtMega 2560, Salida para 8 motores más 3 servos para cardan, 8 canales de entrada para estándar receptor y canal de entrada suma PPM, 4 puertos serie para Depuración/Bluetooth/OSD/GPS/telemetria, 8 puertos analógicos para Sensor velocidad viento/Sensor Tensión Intensidad/LED control, puerto I2C 5V para dispositivo externo, FT232RQ USB-UART, 16 Mbit Memoria Flash de datos para logging, MPU6050 giroscopio y acelerómetro de 6 ejes con unidad de procesamiento de movimiento, HMC5883L brújula digital de 3 ejes y MS5611-01BA03 altímetro de alta precisión.
- Sistema de Telemetría. HM-TRP Módulos transceptores de 100 mW, modulación FSK, comunicación 2-vias half-duplex, banda ISM 433 MHz, potencia máxima de salida 100 mW(20dBm) ajustable entre 1-20 dBm, Sensibilidad -117 dBm, Transmisor 100 mW y Receptor 25 mW, Modo Sleep baja intensidad 1 uA, Interface Estándar TTL UART extensible a RS232 y otros.
- Sistema de Radio Control para UAV. – Turnigy 9x de 2.4 GHz.
- Controlador de Velocidad de los motores. – Controlador de velocidad de motor brushless RC TIMER ESC 30A, entrada de 6-16.8 V, Intensidad de trabajo 30 A continuo y 40 A sobre 10 seg, salida de 5 V.
- Sistema GPS. – GPS NEO-6M, Interface UART, USB, SPI y DDC, actualización 5Hz, 32 Kbit I2C EEPROM para grabar configuración.
- Sensor Voltaje e Intensidad – Voltaje max. 51.8 V, Intensidad max. 89.4 A, se auto alimenta.
- Motores. – HP2212 Motor Brushless, 1000 KV, 6-16.8 V, empuje 780 gr.
- Tarjeta para la distribución y alimentación de energia a los diferentes componentes del UAV.

Software

- Servidor Web Gunicorn que permite websocket.
- Framework Web Django.
- Websocket Wise.
- Firmware ArduCopter APM 3.02.
- Raspbian sistema operativo.

- MAVProxy estación terrena.
- Protocolo de comunicación Mavlink.

Capítulo 2

Objetivos

El objetivo principal del proyecto es desarrollar un sistema de transporte basado en UAV's que sea integral, que incluya las aeronaves, sistemas de comunicación, estación terrena y los servidores y dispositivos necesarios para su correcto uso y despliegue.

El objetivo principal del proyecto implica varios objetivos secundarios que se deben resolver. El sistema deberá gestionar múltiples vehículos, que deben estar coordinados especialmente en las operaciones de salida y llegada. Los vehículos informarán (y serán informados) de cualquier eventualidad que surja durante su operación, tales como eventos meteorológicos, averías o accidentes. El sistema debe proporcionar rutas seguras a través de las zonas pobladas y monitorizar que los vehículos las siguen de forma eficiente y consistente. Otros aspectos importantes son el alcance de las comunicaciones, el consumo y la seguridad de la población.

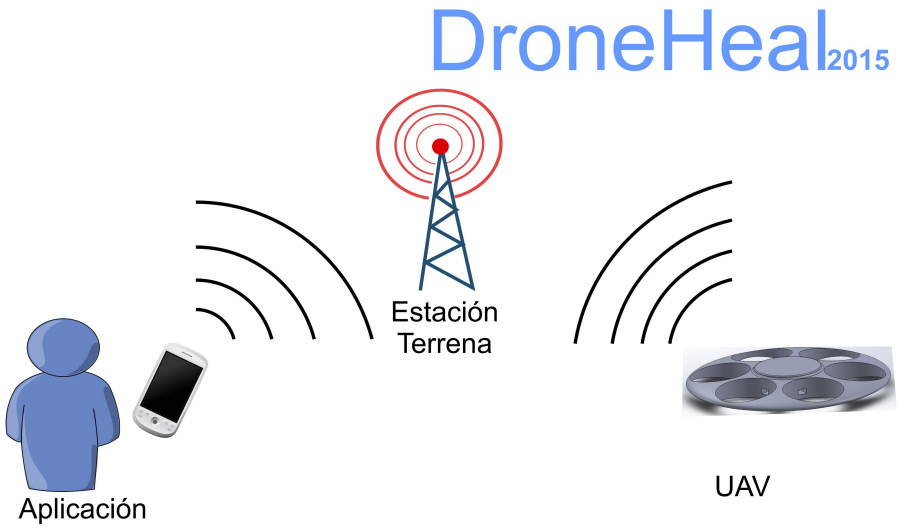

Figura 2.1: Dibujo del sistema

Las restricciones hardware y software a utilizar condicionan notablemente las decisiones de a tomar, como se especifica en el capítulo 5 de este documento.

Respecto al sistema, éste debe cumplir las siguientes premisas de calidad:

- Alto rendimiento.
- Bajo consumo.
- Bajo coste.

2.1 Objetivos específicos

Dado el alcance del proyecto y la complejidad añadida por el hardware y el software a utilizar, se desglosa el objetivo principal en varios subobjetivos que nos proporcionaran una visión más detallada del desarrollo del sistema.

La manera de funcionar entre ellos es la siguiente:

El UAV se comunica con la estación terrena gracias al protocolo MAVLink, y con el cliente web gracias a wise, REST. [MAV14]

El UAV una vez configurado y encendido, se conectará a la estación terrena que tambien estará encendida y lista para usarse. En la estación terrena estaran los servidores escuchando en sus respectivos puertos. Una vez hecho esto solo falta acceder desde cualquier dispositivo con navegador web gráfico. El cual, primero deberá conectar a su red, y a continuación acceder a la dirección facilitada por el administrador, una vez hecho esto se abrirá la aplicación lista para comunicarnos con el UAV.

A continuación se listan todos los subobjetivos, más adelante se describirá cada uno detalladamente.

- Fabricación del UAV.
- Aplicación web.
- Servidores de base de datos, web y GPS.

Cada uno de estos tres sub-objetivos se divide en otros sub-objetivos que se detallarán a continuación.

2.1.1 Fabricación del UAV

Se debe analizar la plataforma hardware elegida de modo que se conozcan los distintos componentes que la forman y la iteración de los mismos. [Are14]

La arquitectura utilizada está compuesta por muchas partes que trabajan conjuntamente y controladas por la controladora de vuelo. La arquitectura de todos los microcontroladores, sensores y actuadores está basada en Arduino. Se adjunta diagrama de bloques en el anexo 4 para facilitar su comprensión.

Este sub-objetivo se divide en los siguientes objetivos.

- Estudio de los sistemas de control, protocolos y vehículos autónomos.
- Montaje del vehículo y análisis de sus características.
- Diseño y codificación de los prototipos.
- Evaluación de los resultados.

2.1.2 Aplicación

La aplicación es en realidad la estación de control terrena, desde la cual se gestionarán los usuarios, clientes y drones. Y por otro lado se controlará el envío de los drones para realizar los pedidos.

Se deben analizar las distintas posibilidades existentes a la hora de desarrollar una aplicación para el control del sistema, con los criterios de calidad antes detallados.

Intentando cumplir en todo momento con los criterios de eficiencia, modularidad y versatilidad. Por eso será una aplicación web. Lo que nos permite acceder desde cualquier tipo de dispositivo que disponga de navegador web gráfico.

2.1.3 Servidores

Aquí podemos encontrar 3 componentes y un dispositivo como es la Raspberry Pi que se utilizará de unidad central de computo. Los 3 componentes son, el servidor de la base de datos, el servidor web y el servidor de GPS.

Analizar, diseñar y configurar distintos tipos de servidores para la base de datos, la aplicación web, la estación terrena y la comunicación.

Capítulo 3

Antecedentes

En este capítulo se van a discutir las diferentes alternativas que existen para el desarrollo de este tipo de sistemas, y porqué se ha decidido cada una de las opciones. Como dijimos anteriormente el sistema está dividido en 3 componentes clave, el UAV o UAV, la estación terrena y los servidores.

Antes de comenzar con el estudio de las diferentes alternativas, debemos establecer los requisitos de nuestro sistema para su mejor adecuación. Estos requisitos ya se discutieron en otro apartado, con lo cual este apartado solo se discutirán las diferentes alternativas de diseño encontradas.

Cada uno de los componentes clave está dividido a su vez por muchos otros componentes. A continuación se detallan cada uno de los componentes necesarios para nuestro sistema.

Drone o UAV	**Estación Terrena**	**Servidores**
Controladora de vuelo	Interface Web	Base de Datos
Control Electrónico de Velocidad	Antena Base	Web
Motores	Raspberry Pi	GPS
Hélices	S.O. Raspbian	
Baterías		
Estructura		
GPS		
Comunicaciones		
Firmware		

Cuadro 3.1: Componentes del Sistema

El primer elemento que tenemos que elegir es el controlador de vuelo, ya que por un lado hemos decidido realizar un desarrollo top-down, y por otro, es el elemento que más va a influir en todos los demás.

Antes de decidir qué controlador usaremos, vamos a detallar todas las alternativas de controladores encontradas. Los requisitos que valoraremos en cada uno de los controladores serán, si es privativo o código abierto, esto en un principio parece que es algo ético y no funcional, y en realidad es un factor práctico, los controladores de código abierto nos permiten

en casi todos los casos modificar el firmware, y la documentación existente es mucho mas extensa en la mayoría de los casos en proyectos de código abierto. El protocolo de comunicación también es muy importante, ya que nos permitirá la interacción con otros sistemas que utilicen el mismo protocolo. Otra característica que vamos a valorar es su precio, en relación a su funcionalidad. También el tamaño de las operaciones a nivel de microcontrolador. Y por último todas sus características y funciones.

Una vez que hemos discutido los diferentes sistemas de control de vuelo, debemos discutir las características que necesitamos en la aeronave y que diferentes alternativas nos podemos encontrar.

El peso máximo de carga útil del UAV será de 1,5 kg aproximadamente. El peso del propio UAV que debe estar sobre 1 kg contando las baterías. Dejaremos un margen de 1,5 kg, con lo cual el peso total que debe poder empujar los motores será de unos 4 - 4,5 kg.

Si nuestro UAV tiene 4 motores, cada uno debe tener una fuerza de empuje de 1 kg. Y en el caso de que sea de 6 motores cada tendrá una fuerza de empuje de 750 gr. Si estudiamos las características de cada motor, podemos ver que su empuje no es siempre igual, y que depende del voltaje de las baterías y del tamaño de las hélices. A mayor voltaje, más fuerza de empuje. Y si las hélices son más largas también más fuerza de empuje.

3.1 UAV

Un UAV es un vehículo aéreo no tripulado, conocido como UAV, y hace referencia a vehículos no tripulados y pilotados remotamente, según la Organización Internacional de Aviación Civil (ICAO). La ICAO diferencia dos tipos de aeronaves:

- Las aeronaves autónomas.
- Las aeronaves controladas remotamente.

El método de lanzamiento y recuperación típica es por la función de un sistema automático o un operador externo en el suelo. El número de aplicaciones civiles es muy extenso, tales como vigilancia y extinción de incendios, seguridad privada, transporte de mercancías, rescates, inspecciones en altura, grabación de imágenes, cartografía, etc. [D'A15]

3.1.1 Controladora de vuelo y Firmware

En general suelen tener todas una estructura similar y unos componentes más o menos sofisticados, pero cuentan con:

- Acelerómetro para poder medir la propia "inercia" de los movimientos.
- Giróscopo para poder medir la velocidad angular de los cambios de posición.
- Magnetómetro utilizado como una brújula que permite saber en todo momento la dirección a la que apunta el UAV.

- Sensor barométrico empleado para conocer con una precisión extrema la altura real de vuelo.
- GPS para conocer las coordenadas exactas en el espacio del UAV (incluida la altura) y poder desplazarse de forma autónoma.
- Un procesador lo suficientemente potente como para realizar las máximas lecturas y operaciones por segundo en base a todos los datos que recibe (que no son pocos).

Con la combinación de todos esos componentes electrónicos, se consigue suficiente información del medio para tomar las decisiones correctas sobre los actuadores que harán posible el vuelo.

En el mercado existen multitud de controladoras de vuelo, debido a nuestros requisitos funcionales como son bajo coste, bajo consumo y alto rendimiento, nos centraremos en los siguientes:

- ArduPilot.
- OpenPilot.
- Acro Naze32/Flip32.
- Brain FPV.
- KK Multicontroller.
- MultiWii.

Tipo de Controladora	CPU	Waypoint	Tipos de multicopter	Licencia
ArduPilot	8 / 32 bits	Si	2, 3, 4, 6 y 8 rotores	GPL v3
OpenPilot	32 bits	No	2, 3, 4 y 6 rotores	GPL v3
Acro Naze32/Flip32	32	No	2, 3, 4, 6 y 8 rotores	GPL v3
Brain FPV	32	Si	2, 3, 4, 6 y 8	GPL v3
KK Multicontroller	8 bits	No	2, 3, 4 y 6	GPL/LGPL
MultiWii	8 bits	No	2, 3, 4, 6 y 8	GPL v3

Cuadro 3.2: Controladoras de vuelo

APM - Ardu Copter

Es una plataforma de código abierto para vehículos aéreos no tripulado, capaz de controlar multicopter autónomos, aeronaves de ala fija, helicópteros tradicionales y vehículos de tierra. Ardupilot fue una plataforma ganadora en 2012 y 2014 de la competición UAV Outback Challenge. Fue creado en 2007 por la comunidad «Hazlo Tu Mismo» (DIY) UAVs. Esta basado en la plataforma de código abierto de prototipado electrónico Arduino. La primera versión de ArduPilot se basó en una termo pila, que se basa en la determinación de la ubicación del horizonte con respecto al avión por la medición de la diferencia de temperatura entre el cielo y el suelo. Más tarde el sistema se ha mejorado para remplazar termo pilas

con una Unidad de Medición Inercial (IMU) utilizando una combinación de acelerómetros, giroscopios y magnetómetros. Bajo una licencia GPL v3.

Hoy en día, el proyecto ArduPilot ha evolucionado a una gama de productos hardware y software muy amplia. Los productos hardware que nos podemos encontrar son:

- APM es una controladora de 8 bits, basada en el chip Atmel ATMEGA 2560.
- Pixhawk/PX4 es una controladora de 32 bits, basada en el chip STM32F427 Cortex M4 core con FPU.

Y los productos software son:

- ArduCopter es el sistema de control de vuelo para multicopter y helicópteros tradicionales.
- ArduPlane es el sistema de control de vuelo para aviones de ala fija y planeadores.
- ArduRover es el sistema de control para vehículos de tierra.
- Mission Planner es la estación terrena de control para todos los tipos de vehículo y arquitectura.
- AntennaTraker es el sistema de control para establecer el ángulo de inclinación de la antena, para obtener una mejor señal, usando la posición GPS del vehículo.

También podemos encontrar otras versiones de los productos hardware en el mercado chino, que permite abaratar el coste del dispositivo en 3 veces. En nuestro caso utilizaremos una versión china de la placa APM, llamada Crius All in one Pro v2.0. [Ard15]

Open Pilot

Es un proyecto de software libre para vehículos aéreos no tripulados, como multicopter, aviones de ala fija y vehículos de tierra. Inicialmente fundado por David Ankers, Angus Peart y Vassilis Varveropoulos a finales de 2009. Fue concebido como una herramienta de aprendizaje y para hacer frente a diferentes áreas percibidas en otras plataformas de UAV.

El software OpenPilot de código abierto con piloto automático puede ser combinado con diferente hardware como un sistema de navegación inercial, una tarjeta controladora general, un receptor GPS, sistema de telemetría y un enlace 2.4 GHz serie de comunicación con la estación terrena. Se distribuye bajo una licencia GPL v3.

El proyecto OpenPilot consiste en dos componentes, por un lado el firmware alojado en el UAV y por otro la Estación de Control Terrena (GCS). El firmware está escrito en C, mientras que el GCS esta escrito en C++ utilizando Qt.

Existen dos tipos de hardware, el Revolution y el CC3D. La diferencia entre ellos es que el Revolution permite vuelo automático e incorpora un modem UHF. Aparte del coste, que es mucho más reducido el del CC3D. El CC3D cuenta con 6 grados de libertad, o lo que es

lo mismo, con 6 giroscopios que permiten saber en todo momento su posición. Aunque simple, es un verdadero controlador de vuelo que no compromete el rendimiento del vuelo. En esencia, estas tarjetas controladoras de vuelo disponen del mismo firmware de estabilización. [Ope15]

Acro Naze32/Flip32

Diseñado en un principio para multirotores aéreos de pequeño y mediano tamaño para interior y exterior, o como un estabilizador independiente de cámara, el controlador de vuelo Naze32 es simple su configuración, con configuración software basado en la familia "MultiWii". [QGr15]

- 36 x 36 mm.
- 6 gramos (8 con los conectores).
- Procesador ARM de 32 bits corriendo a 3.3V/72MHz.
- Giroscopios, acelerómetros, brújula y sensor de presión
- Soporte para Quad, Tri, Hex, Octo, varias configuraciones para coaxial, todo tipo de vehículos aéreos y configuraciones personalizadas.
- Entrada RC flexible, estándar, CPPM(PPM SUM), satélite Spektrum.
- Monitorización de la batería del voltaje y alarma de bajo voltaje.
- Convertidor de telemetría FrSky.
- USB para configurar y controlar.
- GPS bloqueo de posición / vuelta a casa. [1]

Brain FPV

Se trata de una controladora de vuelo de última generación, diseñada específicamente para la vista en primera persona (FPV) del vuelo e incluye una Visualización en Pantalla (OSD). Esta controladora es pequeña y ligera (36 mm x 36 mm, 8 gramos), por lo que es ideal para mini-quads de FPV, pero se puede volar cualquier cosa, desde mini quads a grandes opto-cópteros y aviones de ala fija.

El firmware que se ejecuta en esta controladora es de código abierto (GPL V3) y basada en Tau Labs, lo que le ofrece muchas características interesantes y también una gran plataforma que funciona para todos los sistemas operativos (Linux, Windows, OS-X) con el Software GCS, para configurar la controladora de vuelo utilizando un USB.

Es importante destacar que el firmware “open source” ofrece total libertad para realizar cualquier modificación; tanto para modificar la forma en que la dirección de retorno se

[1] En proceso

muestra en la OSD, como para mejorar el código y contribuir para que el máximo número de personas se beneficie. Naturalmente, esto no significa que se tenga que ser un programador, o saber programar, para utilizar el cerebro, pero eres libre de modificar el código si lo deseas.

Características Especificas de FPV :

- OSD Full-graphic (360x266 de PAL).
- Niveles de blanco y negro ajustables por Software.
- Detección automática de PAL/NTSC.
- 4 páginas OSD totalmente configurables por el usuario, seleccionable con el transmisor "swicht on"
- Salida de audio (todavía no soportado por el software).
- 3 entradas analógicas para tensión, corriente y medición RSSI.
- Medición RSSI utilizando PWM, PPM, o entrada analógica.

Otras características:

- CPU: STM32F405RG (32bit, 168MHz, 192kB RAM, 1MB Flash).
- Flash de 64Mbit para la configuración, puntos de paso y logging.
- InvenSense MPU-9250 de última generación 3-axis gyro/accel/mag.
- Barómetro: MeasSpec MS5611.
- Compatibilidad del receptor: PWM, PPM, S.Bus, DSM2, DSMX, HoTT SUMD/SUMH.
- Hasta salidas de 10 PWM (hasta 400Hz velocidad de actualización).
- Hasta 3 puertos serie para telemetría, GPS, receptor RC, etc.
- Puerto I2C externa,que puede, por ejemplo, ser utilizado con una brújula HMC5883 externa.
- Puerto micro USB para configuración via PC.

KK Multicontroller 2.0

El KKmulticontroller es una placa controladora de vuelo para control remoto de multicopter con 2,3,4 y 6 rotores. Su propósito es estabilizar la aeronave durante el vuelo. Para ello se necesita la señal de los 3 giroscopios en la placa(equilibrio, cabeceo y estabilidad) y alimenta la información en el circuito integrado (ATMega). Éste procesa la información según el software KK y envía una señal de control a los controladores de velocidad electrónicos (ESC), que están conectados a la placa y también conectados a los motores. Dependiendo de la señal del microcontrolador, el ESC, o bien acelera, o bien reduce la velocidad de los motores (e inclinar el rotor trasero en el caso de un tricopter) a fin de establecer el nivel de vuelo. La

placa también toma una señal de control desde el mando a distancia del receptor y alimenta a esto en el microcontrolador a través de los pines de alerones, elevador, acelerador y timón en la placa. Después de procesar esta información, el microcontrolador envía una señal a los motores (a través de los conectores M1 a M6 de la placa) para acelerar o reducir la velocidad para alcanzar el vuelo controlado (arriba, abajo, hacia atrás, hacia delante izquierda, derecha, giro en su propio eje) en el mandato que el piloto de RC envía a través de su transmisor. En el caso de un tricopter, uno de los conectores M4 controlará un servo para lograr el movimiento de giro.

Está basado en el microcontrolador Atmel ATMega 168 de 8 bits, con 16 Kbytes de memoria flash, frecuencia de operación 20 MHz, sin interface USB y sin puntos de control. [Aut15]

MultiWii

MultiWii es un software de propósito general para controlar un modelo de Radio Control (RC) multirotor. Puede usar varios sensores pero fue inicialmente desarrollada para soportar el mando de la consola Nintendo Wii gracias a sus giroscopios y acelerómetros. Podemos encontrar estos sensores en una extensión de la Nintendo WiiNote: Wii Motion Plus y Wii Nunchuk. Este proyecto está desarrollado en una plataforma Arduino. La estabilidad lograda es excelente para FPV y permite cualquier tipo de acrobacias. El software es, por el momento capaz de controlar un tricopter, un cuatricopter o un hexacopter.

Las características de este controlador es muy parecido a APM, está basado en el mismo microcontrolador el Atmel ATMega2560 de 8 bits, pero no soporta puntos de control.

3.1.2 Control Electrónico de Velocidad (ESC)

Los modelos controlados por radio funcionan con pequeños motores electrónicos. Estos motores deben utilizar un dispositivo para controlar la velocidad y el frenado. Un ESC, o Control Electrónico de Velocidad, se utiliza para este propósito.

Definición

Un control electrónico de velocidad es un circuito utilizado para manipular la dirección electrónica del motor y el rango de velocidad. Estos controles también pueden ser utilizados como un freno dinámico, o como un método para bajar la velocidad de un motor devolviéndole su energía o potencia a la línea de suministro.

Proceso

El control electrónico de velocidad recibe información del control y la utiliza para manipular la conmutación en su red de transistores que controla el motor a través de la electricidad. Los dispositivos electrónicos de velocidad utilizan un CEB, o circuito eliminador de batería

para regular el voltaje para el receptor, un circuito electrónico que recibe su entrada a través de una antena de señal de radio.

3.1.3 Motores sin escobillas o brushless

Un motor eléctrico sin escobillas o motor brushless es un motor eléctrico que no emplea escobillas para realizar el cambio de polaridad en el rotor.

Los motores eléctricos solían tener un colector de delgas o un par de anillos rozantes. Estos sistemas, que producen rozamiento, disminuyen el rendimiento, desprenden calor y ruido, requieren mucho mantenimiento y pueden producir partículas de carbón que manchan el motor de un polvo que, además, puede ser conductor.

Los primeros motores sin escobillas fueron los motores de corriente alterna asíncronos. Hoy en día, gracias a la electrónica, se muestran muy ventajosos, ya que son más baratos de fabricar, pesan menos y requieren menos mantenimiento, pero su control era mucho más complejo. Esta complejidad prácticamente se ha eliminado con los controles electrónicos.

El inversor debe convertir la corriente alterna en corriente continua, y otra vez en alterna de otra frecuencia. Otras veces se puede alimentar directamente con corriente continua, eliminado el primer paso. Por este motivo, estos motores de corriente alterna se pueden usar en aplicaciones de corriente continua, con un rendimiento mucho mayor que un motor de corriente continua con escobillas. Algunas aplicaciones serían los coches y aviones con radio control, que funcionan con pilas.

Otros motores sin escobillas, que sólo funcionan con corriente continua son los que se usan en pequeños aparatos eléctricos de baja potencia, como lectores de CD-ROM, ventiladores de ordenador, etc. Su mecanismo se basa en sustituir la conmutación (cambio de polaridad) mecánica por otra electrónica sin contacto. En este caso, la espira sólo es impulsada cuando el polo es el correcto, y cuando no lo es, el sistema electrónico corta el suministro de corriente. Para detectar la posición de la espira del rotor se utiliza la detección de un campo magnético. Este sistema electrónico, además, puede informar de la velocidad de giro, o si está parado, e incluso cortar la corriente si se bloquea para que no se queme. Tienen la desventaja de que no giran al revés al cambiarles la polaridad (+ y -). Para hacer el cambio se deberían cruzar dos conductores del sistema electrónico.

Existen infinidad de tipos de motores, como ya discutimos anteriormente las características de los motores son muy concretas, un empuje aprox. de 4.5 Kgr. y que no supere el 1,5 Kg de peso de toda el aeronave.

3.1.4 Hélices

La hélice es un dispositivo mecánico formado por un conjunto de elementos denominados palas o álabes, montados de forma concéntrica y solidarias de un eje que, al girar, trazan un movimiento rotativo en un plano. Las palas no son placas planas, sino que tienen una

forma curva, sobresaliendo del plano en el que giran, y obteniendo así en cada lado una diferencia de distancias entre el principio y el fin de la pala. Provocando una diferencia de velocidades entre el fluido de una cara y de la otra. Según el principio de Bernoulli esta diferencia de velocidades conlleva una diferencia de presiones, y por lo tanto aparece una fuerza perpendicular al plano de rotación de las palas hacia la zona de menos presión. Esta fuerza es la que se conoce como fuerza propulsora de un buque o aeronave.

La primeras aplicaciones de las hélices, hace miles de años, fueron los molinos de viento y agua. Hoy en día, también bajo los nombres de rotor, turbina y ventilador, las hélices y los dispositivos derivados de ellas se emplean para multitud de propósitos: refrigeración, compresión de fluidos, generación de electricidad, propulsión de vehículos e incluso para la generación de efectos visuales (estroboscópico).

El inventor de la primera hélice operativa para propulsar un buque fue el checo-germano Josef Ressel, quien solicitó la patente austriaca el 28 de noviembre de 1826. Astilleros Españoles, S.A. propuso varias formas de mejorar el rendimiento de las hélices marinas, con ganancias, bien en velocidad punta del buque, o en consumo de combustible.

Como evidencia la variedad de denominaciones y campos de aplicación, existe una gran variedad de hélices, en tamaños, pesos, número de palas, velocidad de rotación, ángulo de ataque, paso, etc.

3.1.5 Baterías

El tipo de batería a utilizar es muy importante, ya que de ello depende la autonomía de la aeronave y su peso. Según el material con el que estén hechás, las baterías pueden ser, de litio y de níquel para este tipo de aeronaves.

Comencemos identificando los tipos de baterías que existen, sus respectivos nombres, abreviaturas y especialmente las utilizadas en radio control. Básicamente existen seis tipos de baterías:

- Nickel Cadmiun (NiCd).
- Nickel-Metal Hydride (NiMH).
- Lead Acid.
- Lithium Ion (Li-ion).
- Lithium Ion Polymer (Li-ion Polymer).
- Baterías Alcalinas.

Antes de comenzar a explicar los detalles de cada batería, se mostrará una pequeña tabla que resume información importante de comparación entre todas ellas. Esta información fue obtenida de los Laboratorios CADEX Electronics Inc. [Cad]

Figura 3.1: Tipos de hélices.

Figura 3.2: Tipos de hélices.

Niquel Cadmiun (NiCd).

Las baterías Nickel Cadmiun fueron inventadas en 1899 por Waldmar Jungner, de allí han obtenido un gran desarrollo hasta nuestros días, son las mas utilizadas en la actualidad y se podría decir que están en mas de 70 % de los equipos electrónicos a nivel mundial. Las bate-

rías NiCd son utilizadas donde se requiere larga durabilidad, alta capacidad de proporcionar energía y lo mas importante, precios económicos. Estas baterías son utilizadas en su mayor parte en radios transmisores, equipos médicos, cámaras de vídeo profesionales, radio control y herramientas de trabajo. Las baterías NiCd contienen metales tóxicos y no es ambientalmente amistosa. Podemos resumir algunas ventajas y limitaciones de las baterías NiCd:

Ventajas:

- Acepta fácilmente carga rápida y carga lenta.
- Con apropiado mantenimiento, la batería puede llegar a 1000 ciclos de carga.
- Proporcionan buena eficiencia y se pueden cargar a bajas temperaturas.
- Larga vida en cualquier método de carga.
- De fácil transporte y almacenaje.
- Baja temperatura de trabajo.
- Son las mas económicas.
- Disponibles en muchos tamaños.

Limitaciones:

- Relativamente de baja densidad de energía comparada con otras baterías.
- Por el efecto memoria, tienen que ser periódicamente recicladas.
- Contiene materiales tóxicos, algunos países tienen limitación para su uso.
- Moderado consumo de su propia energía en el almacenaje.
- Necesita ser recargada después del almacenaje.

Efecto Memoria

En el mundo del radio control se puede encontrar que la mayoría de las personas tienen sus propias versiones del efecto memoria en las baterías y muchas de ellas resultan ser incorrectas, otras versiones son exageradas y encaminadas en temas que no están relacionados; sin embargo, las siguientes líneas estarán destinadas a informarle de todo lo relacionado al efecto memoria, como se combate y como se previene. La información fue obtenida de la empresa CADEX ELECTRONIX y las fotografías de soporte fueron proporcionadas por United States Army Electronics Command.

El efecto memoria es un problema que afecta directamente a las baterías Nickel Cadmiun (NiCd) en mayor proporción. Cuando las baterías NiMH fueron introducidas al mercado, estas fueron promocionadas como baterías libres de memoria, aunque esta información no es

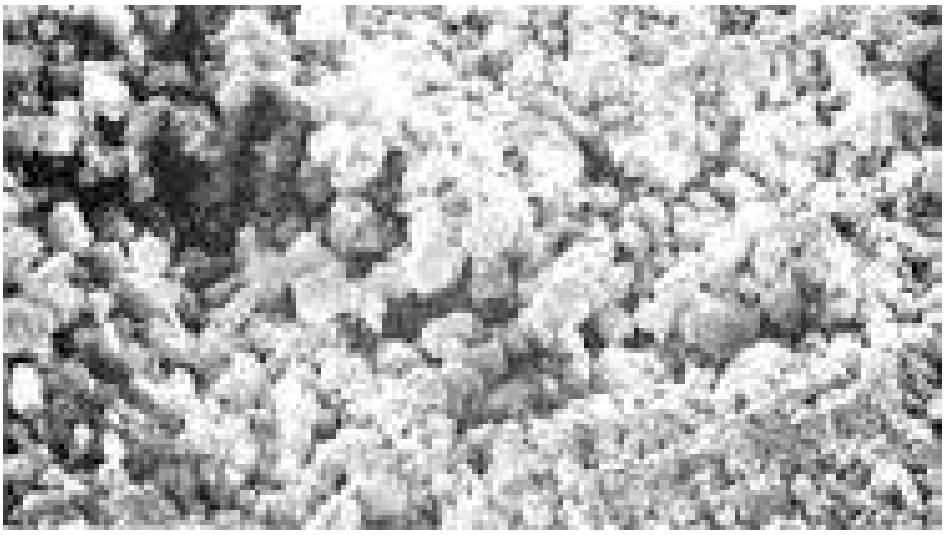

Figura 3.3: Efecto Memoria I.

cierta, las baterías NiMH sufren de memoria pero en mucho menor cantidad que las baterías NiCd.

Las baterías Nickel Cadmiun utilizan material activo denominado cadmio. Este material esta dividido en pequeños cristales que tienen un longitud de 1 micrón. Esta pequeña medida en los cristales le da la particularidad de que mayor cantidad de ellos tienen contacto directo con el electrodo proporcionando así mayor eficiencia o energía. En la imagen de abajo, figura 3.3 se puede observar el material en forma de cristales en condiciones normales, es decir sin el efecto memoria presente. Cuando el efecto memoria ocurre, estos cristales aumentan considerablemente de tamaño entre 10 y 100 micrones implicando que menor cantidad de ellos tendrán contacto directo con el electrodo precisamente por su tamaño. Observe la imagen de abajo, figura 3.4, en donde se señala un cristal que apenas tiene 10 Micrón.

¿Cuando ocurre el efecto memoria? El efecto memoria expresado por Duracell dice: «El voltaje cae porque solo una porción de los materiales activos de la batería son descargados y recargados casi de igual forma por un cierto tiempo, por ejemplo, si se recarga una batería de 1000 mAh y se utilizan tan solo 200 mAh y este proceso es repetido varias veces, el efecto memoria aparecerá en la batería.» Los materiales activos cambian sus características físicas e incrementan su resistencia. El efecto memoria tendrá un tamaño dependiendo del usuario, por eso no existirá ninguna regla general. Si usted tan solo utiliza un 10 % de la batería cada vez que la recarga, tendrá un 90 % de memoria en su batería. En el mundo del radio control, específicamente en el helimodelismo y aeromodelismo, el efecto memoria es un factor importante, por que la batería es prácticamente el corazón del aeromodelo. Una falla de la batería por el efecto memoria, podría llegar a destruir totalmente el aeromodelo.

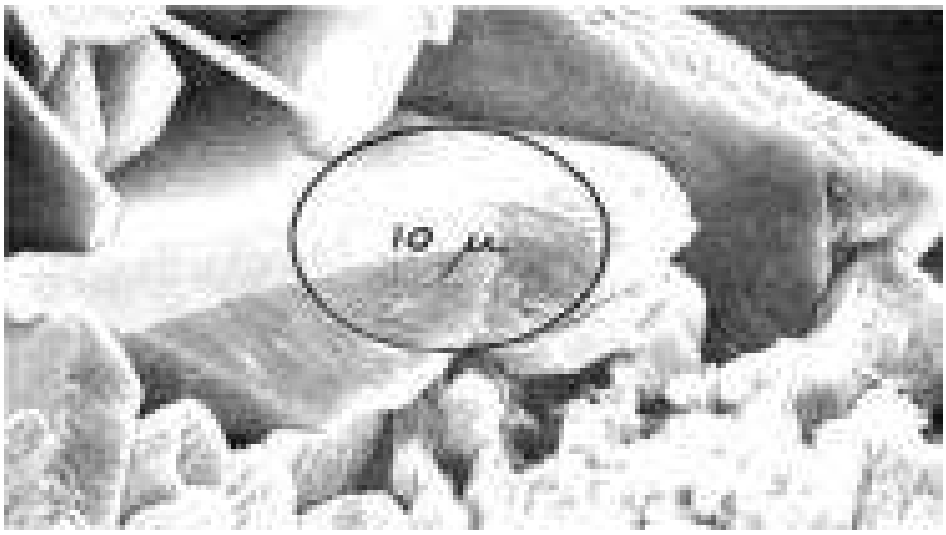

Figura 3.4: Efecto Memoria II.

Polímero de Litio

La Batería de Li-Polymer se diferencia básicamente de otras baterías por el tipo de electrolito usado. El diseño original data de 1970 en la cual utilizaba un electrolito Polímero sólido seco. Este electrolito fue reinstalado como una lámina plástica que no tiene la propiedad de no conducir electricidad pero si tiene la propiedad de intercambiar iones (grupos de átomos o átomos cargados de electricidad). El electrolito reemplaza el tradicional separador poroso con un elemento impregnado de electrolito.

El diseño de polímero seco ofrece sencillez con respecto a la fabricación: seguridad y baterías extremadamente planas. No existe peligro de incendio ya que no tienen electrolito en forma de liquido o gel.

Con el espesor de la celda la cual mide como un milímetro (0.039 pulgadas), los diseñadores podrán utilizar su imaginación con respecto a la forma y tamaño de la batería. Es posible crear formas de baterías que conforman parte del encapsulado del equipo o prácticamente cualquier configuración conveniente que se adapte a las necesidades del diseñador.

Lamentablemente el polímero sólido seco sufre de mala conductividad, la resistencia interna es demasiado alta y no entrega la corriente que requieren por ejemplo los equipos de comunicación. Pero si la celda es calentada a 60 grados centígrados o mas incrementa la conductividad en un nivel aceptable. Las investigaciones continúan desarrollando un polímero sólido seco que trabaje bien a temperatura ambiente. La versión de baterías de polímero seco aparecerá comercialmente en nuestros días.

Actualmente algunas baterías de Li-Polymer no trabajan bien en climas calientes. Un fabricante agrego elementos químicos (gel electrolítico) en la cual la batería permanece conductiva en todos los rangos de temperatura. Este tipo de batería es muy utilizada actualmente

en teléfonos celulares. El nombre correcto de esta batería es LITHIUM ION POLYMER; pero por razones comerciales los fabricantes suelen escribir un nombre simplificado: Li-polymer.

Ahora con el gel electrolítico agregado a esta batería, ¿cual es la diferencia entre la batería de Li-ion y la batería Li-ion-polymer? Analizando las características y la eficiencia de ambas tecnologías nos encontramos que son muy similares; la batería Li-ion polímero utiliza un electrolito sólido que remplaza el separador poroso y la batería con electrolito de gel simplemente es agregado para que los iones se vuelvan conductivos.

Existe un pequeño retraso en la producción masiva de las baterías de Li-ion-polymer debido a que los fabricantes invirtieron en equipos y en investigación referidas a las baterías de Li-ion, y actualmente se encuentra en un periodo de retraso hasta que retorne las inversiones estimadas por ellos. Con la producción masiva de las baterías Li-ion-polymer implicaría que estas bajarían de precio considerablemente.

Ventajas:

- Pueden obtenerse versiones de tamaños como tarjetas de crédito.
- Muy livianas.
- Mas resistentes a la sobrecarga.
- Menos propensas a que se rompa el electrolito.

Limitaciones:

- Nivel de energía de densidad bajo. Es la cantidad de energia que es capaz de suministrar por tiempo.
- Tiempo menor de vida comparada con las baterías de Li-ion.
- Alto coste de fabricación.

Baterías de Litio usadas en UAV's

Las personas que han visto el progreso de las baterías en el mundo de los UAV's, han podido notar sin lugar a dudas un avance tecnológico; por ejemplo en las baterías de NiCd se ha podido observar celdas de tamaño AA que tan solo podían suministrar hasta 500 mAh [2]; pero ahora existen celdas del mismo tamaño que fácilmente pueden llegar a 1700 mAh. Los paquetes de baterías de NiCd estaban típicamente conformados por 4 celdas al igual ha sucedido con las baterías de Lithium pero con una gran diferencia tanto en el volumen de la batería como el peso de la misma.

Tomemos como ejemplo un paquete de baterías de Sanyo conformados por 4 celdas de 2200 mA. Se puede observar como tiene un peso de 278 gramos (más de 1/4 de kg) y si

[2] miliamperios por hora-Intensidad por unidad de tiempo

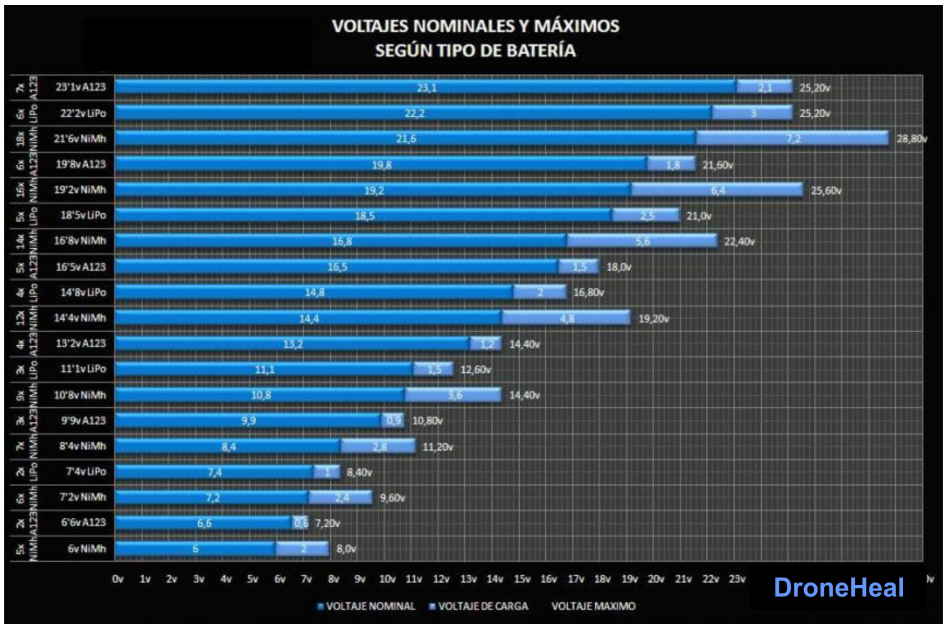

Figura 3.5: Voltajes nominales y máximos.

se agrega una batería de Lithium de 4 celdas configuradas como si fueran dos celdas para proporcionar 7.4 voltios, se puede observar que pesa 218 gramos y con la gran diferencia que tenía una capacidad para proporcionar 4000 mAh. Resumamos la comparación en una pequeña tabla para observarlo mejor:

Comparación entre baterías de litio y de níquel

Descripción	Batería de Litio	Batería de NiCd
Capacidad de Corriente	4000	2000
Número de celdas	4	4
Peso del paquete	218 gramos	278 gramos
Volumen del paquete	3 X 3 X 6.5 cms = 58 cm3	11.5 X 6 X 3 cms = 2017 cm3

Cuadro 3.3: Comparación entre los tipos de baterías para UAV (CADEX ELECTRONIC INC [Cad])

Se puede observar claramente que todas las variables que se analizaron en la tabla, la batería de Lithium siempre va a la vanguardia. Esta diferencia tan grande ha implicado un avance tecnológico en los modelos radio controlados, específicamente los modelos eléctricos. Inicialmente cuando se compró un Dragon Fire IV, esté venía con un paquete de batería de 10 celdas y la capacidad era de 600 mA, el peso de este paquete de baterías es de 178 gramos. Posteriormente fue remplazado por un paquete de batería de Lithion de 11.1 Voltios, 3 celdas, 1200 mAh y tan solo pesa 76 Gramos. ¿Que diferencias existen entre los dos tipos de baterías? Despegaba con mayor facilidad por el peso eliminado, segundo con la batería NiCd volaba escasamente 4-6 minutos y con la batería de Lithium vuela fácilmente 11-14 minutos. Note que el modelo tiene cuatro motores eléctricos.... Los modelos de mini heli-

cópteros que trabajan con 3 celdas de batería de Lithium pueden fácilmente volar 20 minutos continuos.

En el radio control (aeromodelismo y helimodelismo) los paquetes de baterías de NiCd de 4 celdas proporcionan un voltaje nominal de 4.8 voltios en la cual se utilizan para suministrar energía al receptor y los servos del modelo. Actualmente el receptor y los servos trabajar sin ningún problema hasta los 6 voltios, después de ese voltaje los elementos electrónicos podían quemarse o dañarse. Ahora como una batería de Lithium de 2 celdas tiene 7.4 voltios nominal, ésta no se podrá conectar directamente al receptor o los servos y se requiere un regulador de voltaje.

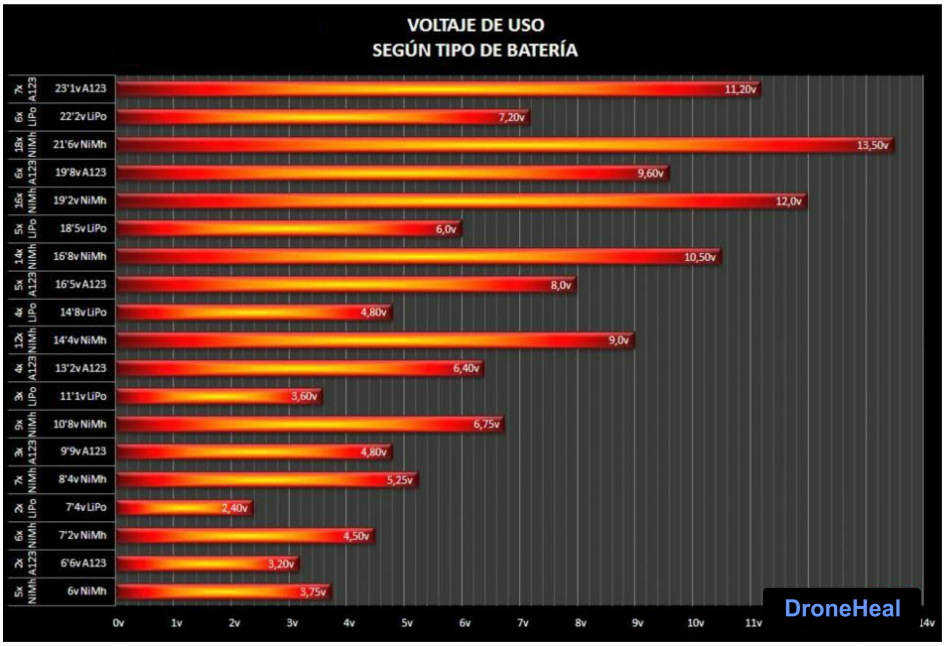

Figura 3.6: Voltaje de uso.

Para el modelismo a radio control, se están utilizando dos reguladores de voltaje diferentes: el primero diseñado para regular 5.1 voltios y el segundo diseñado para regular 6.0 voltios. Un regulador de voltaje es un pequeño dispositivo electrónico que en su salida mantiene un voltaje constante de 5.1 o 6.0 voltios y en la entrada puede recibir un voltaje mucho mayor. Así que fácilmente le podemos conectar una batería de Lithion de 7.4 voltios a un regulador de voltaje y obtener 5.1 o 6.0 Voltios para no dañar o quemar los componentes electrónicos como servos y receptores.

3.1.6 GPS

El sistema de posicionamiento global (GPS) es un sistema que permite determinar en todo el mundo la posición de un objeto (una persona, un vehículo) con una precisión de centímetros (si se utiliza GPS diferencial), aunque lo habitual son unos pocos metros. El sistema fue desarrollado, instalado y empleado por el Departamento de Defensa de los Estados Unidos,

para determinar las posiciones en el globo, el sistema GPS está constituido por 24 satélites y utiliza la trilateración.

El GPS funciona mediante una red de 24 satélites en órbita sobre el planeta tierra, entre 20 a 200 km de altura, con trayectorias sincronizadas para cubrir toda la superficie de la Tierra. Cuando se desea determinar la posición, el receptor que se utiliza para ello localiza automáticamente como mínimo cuatro satélites de la red, de los que recibe unas señales indicando la identificación y la hora del reloj de cada uno de ellos. Con base en estas señales, el aparato sincroniza el reloj del GPS y calcula el tiempo que tardan en llegar las señales al equipo, y de tal modo mide la distancia al satélite mediante el método de trilateración inversa, la cual se basa en determinar la distancia de cada satélite respecto al punto de medición. Conocidas las distancias, se determina fácilmente la propia posición relativa respecto a los satélites. Conociendo además las coordenadas o posición de cada uno de ellos por la señal que emiten, se obtiene la posición absoluta o coordenadas reales del punto de medición. También se consigue una exactitud extrema en el reloj del GPS, similar a la de los relojes atómicos que llevan a bordo cada uno de los satélites.

La antigua Unión Soviética construyó un sistema similar llamado GLONASS, ahora gestionado por la Federación Rusa.

Actualmente la Unión Europea está desarrollando su propio sistema de posicionamiento por satélite, denominado Galileo.

A su vez, la República Popular China está implementando su propio sistema de navegación, el denominado Beidou, prevén que cuente con 12 y 14 satélites entre 2011 y 2015. Para 2020, ya plenamente operativo deberá contar con 30 satélites. En abril de 2011 había ocho en órbita.

Señal GPS

Cada satélite GPS emite continuamente un mensaje de navegación a 50 bits por segundo en la frecuencia portadora de microondas de aproximadamente 1.600 MHz. La radio FM, en comparación, se emite a entre 87,5 y 108,0 MHz y el Wi-Fi funciona alrededor de 5000 MHz y 2400 MHz. Más concretamente, todos los satélites emiten a 1575,42 MHz (ésta es la señal L1) y 1227,6 MHz (la señal L2).

La señal GPS proporciona la “hora de la semana” precisa de acuerdo con el reloj atómico a bordo del satélite, el número de semana GPS y un informe de estado para el satélite de manera que pueda reducirse si es defectuoso. Cada transmisión dura 30 segundos y lleva 1500 bits de datos codificados. Esta pequeña cantidad de datos está codificada con una secuencia pseudoaleatoria (PRN) de alta velocidad que es diferente para cada satélite. Los receptores GPS conocen los códigos PRN de cada satélite y por ello no sólo pueden decodificar la señal sino que la pueden distinguir entre diferentes satélites.

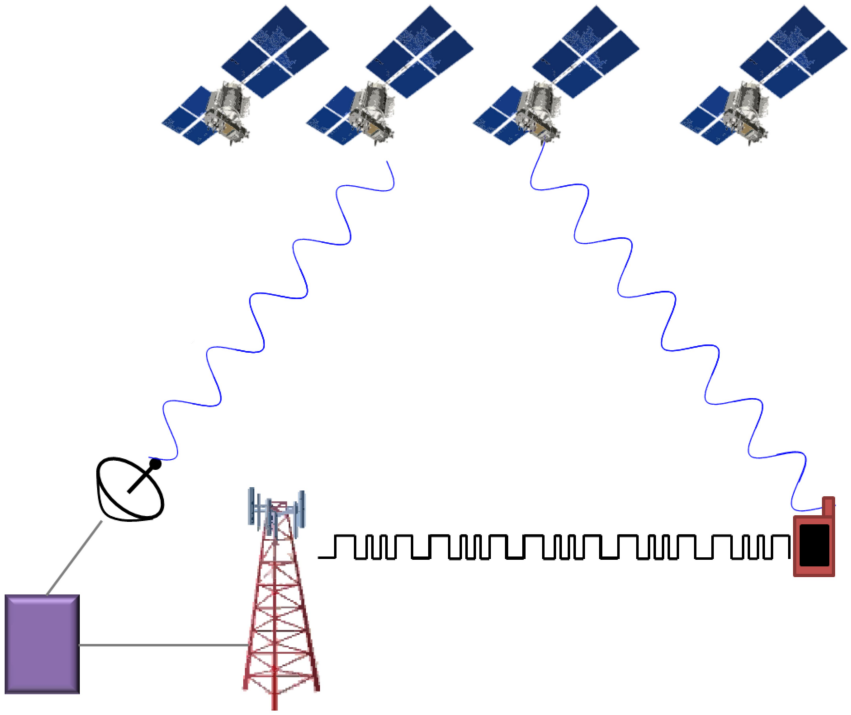

Figura 3.7: Diagrama de funcionamiento del sistema GPS.

Las transmisiones son cronometradas para empezar de forma precisa en el minuto y en el medio minuto tal como indique el reloj atómico del satélite. La primera parte de la señal GPS indica al receptor la relación entre el reloj del satélite y la hora GPS. La siguiente serie de datos proporciona al receptor información de órbita precisa del satélite.

Funcionamiento

La información que es útil al receptor GPS para determinar su posición se llama *efeméride*. En este caso cada satélite emite sus propias efemérides, en la que se incluye la salud del satélite (si debe o no ser considerado para la toma de la posición), su posición en el espacio, su hora atómica, información doppler, etc.

Mediante la trilateración se determina la posición del receptor:

- Cada satélite indica que el receptor se encuentra en un punto en la superficie de la esfera, con centro en el propio satélite y de radio la distancia total hasta el receptor.
- Obteniendo información de dos satélites queda determinada una circunferencia que resulta cuando se interceptan las dos esferas en algún punto de la cual se encuentra el receptor.
- Teniendo información de un tercer satélite, se elimina el inconveniente de la falta de

sincronización entre los relojes de los receptores GPS y los relojes de los satélites. Y es en este momento cuando el receptor GPS puede determinar una posición 3D exacta (latitud, longitud y altitud).

En el mercado existen multitud de GPS, en nuestro caso nos limitaremos a los receptores con conexión serie UART. Muchos de estos GPS llevan incorporado una brújula, en este caso no será necesario ya que este controlador de vuelo lo lleva incorporado. Es mejor que la brújula sea externa para evitar interferencias, ya que es muy sensible a los cambios magnéticos.

3.1.7 Comunicación

El protocolo de comunicación entre los UAV y la estación terrena es MavLink, que es una librería muy ligera de *marshalling* [3] de cabeceras de mensaje para micro vehículos aéreos.

Puede empaquetar estructuras C a través de canales serie con alta eficiencia y enviar estos paquetes a la estación de control de tierra. Está probado extensivamente en la PX4, PIXHAWK, APM y Parrot AR.UAV y sirve como columna vertebral de la comunicación, entre las aeronaves y nuestro sistema operativo en la estación terrena Raspbian.

3.2 Aplicación

Un servidor es una aplicación en ejecución (software) capaz de atender las peticiones de un cliente y devolverle una respuesta en concordancia. Los servidores se pueden ejecutar en cualquier tipo de computadora, incluso en computadoras dedicadas a las cuales se les conoce individualmente como «el servidor». En la mayoría de los casos una misma computadora puede proveer múltiples servicios y tener varios servidores en funcionamiento. La ventaja de montar un servidor en computadoras dedicadas es la seguridad. Por esta razón la mayoría de los servidores son procesos demonio, diseñados de forma que puedan funcionar en computadoras de propósito específico.

3.2.1 Sistema de gestión de base de datos

Un sistema de gestión de bases de datos (SGBD) es un conjunto de programas que permiten el almacenamiento, modificación y extracción de la información en una base de datos, además de proporcionar herramientas para añadir, borrar, modificar y analizar los datos. Los usuarios pueden acceder a la información usando herramientas específicas de consulta y de generación de informes, o bien mediante aplicaciones al efecto.

Estos sistemas también proporcionan métodos para mantener la integridad de los datos, para administrar el acceso de usuarios a los datos y para recuperar la información si el sistema se corrompe. Permiten presentar la información de la base de datos en variados formatos. La

[3] consiste en un proceso de codificación de un objeto en un medio de almacenamiento con el fin de transmitirlo a través de una conexión en red como una serie de bytes o en un formato humanamente más legible como XML o JSON, entre otros

mayoría incluyen un generador de informes. También pueden incluir un módulo gráfico que permita presentar la información con gráficos y tablas.

Hay muchos tipos distintos según cómo manejen los datos y muchos tamaños distintos de acuerdo a si operan en computadoras personales y con poca memoria o grandes sistemas que funcionan en mainframes con sistemas de almacenamiento especiales.

Generalmente se accede a los datos mediante lenguajes de consulta, lenguajes de alto nivel que simplifican la tarea de construir las aplicaciones. También simplifican la consulta y la presentación de la información. Un Sistema de gestión de base de datos (SGBD) permite controlar el acceso a los datos, asegurar su integridad, gestionar el acceso concurrente a ellos, recuperar los datos tras un fallo del sistema y hacer copias de seguridad. Las bases de datos y los sistemas para su gestión son esenciales para cualquier área de negocio, y deben ser gestionados con esmero. Vamos a analizar diferentes opciones:

Sqlite

SQLite es una biblioteca que está en proceso, que implementa una forma autocontenida, sin servidor, sin configuración, motor de base de datos transaccional de SQL autónomo. El código para SQLite es de dominio público y por lo tanto libre para el uso para cualquier propósito, comercial o privado. SQLite es la base de datos más utilizada en el mundo con más aplicaciones que podemos contar.

MySQL

MySQL es un sistema de gestión de bases de datos relacional, multihilo y multiusuario con más de seis millones de instalaciones. MySQL AB —desde enero de 2008 una subsidiaria de Sun Microsystems y ésta a su vez de Oracle Corporation desde abril de 2009— desarrolla MySQL como software libre en un esquema de licenciamiento dual.

Por un lado se ofrece bajo la GNU GPL para cualquier uso compatible con esta licencia, pero para aquellas empresas que quieran incorporarlo en productos privativos deben comprar a la empresa una licencia específica que les permita este uso. Está desarrollado en su mayor parte en ANSI C.

Al contrario de proyectos como Apache, donde el software es desarrollado por una comunidad pública y los derechos de autor del código están en poder del autor individual, MySQL es patrocinado por una empresa privada, que posee el copyright de la mayor parte del código. Esto es lo que posibilita el esquema de licenciamiento anteriormente mencionado. Además de la venta de licencias privativas, la compañía ofrece soporte y servicios. Para sus operaciones contratan trabajadores alrededor del mundo que colaboran vía Internet. MySQL AB fue fundado por David Axmark, Allan Larsson y Michael Widenius.

PostgreSQL

PostgreSQL es un sistema de gestión de bases de datos objeto-relacional, distribuido bajo licencia BSD y con su código fuente disponible libremente. Es el sistema de gestión de bases de datos de código abierto más potente del mercado y en sus últimas versiones no tiene nada que envidiarle a otras bases de datos comerciales.

PostgreSQL utiliza un modelo cliente/servidor y usa multiprocesos en vez de multihilos para garantizar la estabilidad del sistema. Un fallo en uno de los procesos no afectará el resto y el sistema continuará funcionando.

MongoDB

MongoDB es una base de datos documental de código abierto que proporciona un alto rendimiento, alta disponibilidad y escalabilidad automática. MongoDB es una base de datos orientada a documentos. Esto quiere decir que en lugar de guardar los datos en registros, guarda los datos en documentos. Estos documentos son almacenados en Notación para Objetos JavaScript Binarios (BSON), que es una representación binaria de Notación para Objetos JavaScript (JSON). Una de las diferencias más importantes con respecto a las bases de datos relacionales, es que no es necesario seguir un esquema. Los documentos de una misma colección –concepto similar a una tabla de una base de datos relacional-, pueden tener esquemas diferentes.

3.2.2 Wise

Es un framework web de comunicaciones orientado a objetos. Es una implementación de los websocket para Django y otras muchas plataformas de desarrollo. Este framework ha sido desarrollado por Oscar Aceña en el grupo de investigación ARCO, perteneciente a la ESI de la Universidad de Castilla La Mancha.

3.2.3 Servidor Web

Un servidor web o servidor Protocolo de Transferencia de Hipertexto (HTTP) es un programa informático que procesa una aplicación del lado del servidor, realizando conexiones bidireccionales y/o unidireccionales y síncronas o asíncronas con el cliente y generando o cediendo una respuesta en cualquier lenguaje o Aplicación del lado del cliente. El código recibido por el cliente suele ser compilado y ejecutado por un navegador web. Para la transmisión de todos estos datos suele utilizarse algún protocolo. Generalmente se usa el protocolo HTTP para estas comunicaciones, perteneciente a la capa de aplicación del modelo Modelo de Interconexión de Sistemas Abiertos (OSI). El término también se emplea para referirse al ordenador que ejecuta el programa.

En el mercado existen muchas opciones, nos centraremos en servidores web para entornos GNU/Linux, ya que se instalará sobre una distribución Raspbian. También necesitaremos

que se ha muy ligero. Analizaremos los siguientes:

Gunicorn

Gunicorn es un WSGI HTTP servidor Python para sistemas basados en Unix. Tiene un modo de worker re-fork. El servidor Gunicorn es ampliamente compatible con varios framework web, implementación simple, ligero en los recursos del servidor y bastante rápido.

Tornado

Tornado es un framework web Python y una librería asíncrona para comunicaciones, originalmente desarrollada por `www.FriendFeed.com`. Tornado puede escalar de 10 a cientos de conexiones abiertas, haciéndolo ideal para long polling[4], WebSockets y otras aplicaciones que requieran largas conexiones en vivo para cada usuario.

Lighttpd

Lighttpd es posiblemente el servidor web más liviano que hay disponible en la actualidad. Esto significa que consume muy pocos recursos de la máquina en la que se instala. Y por este motivo muchos usuarios lo eligen para montar un servidor web en su Raspberry Pi.

Nginx

Aunque Apache es el servidor web más usado en Internet, tiene el inconveniente de que está pensado para máquinas con un hardware potente. Nginx, por el contrario, es un servidor web que consume pocos recursos, por lo que es más adecuado para instalarlo en una pequeña placa como la Raspberry Pi.

LAMP

LAMP es el acrónimo que corresponde a Linux, Apache, MySQL y PHP. Se trata del conjunto de elementos necesario para montar un completo servidor web. Una vez instalado, cualquier cliente, mediante un navegador web, podrá conectar con nuestra Raspberry Pi y ver las páginas que deseemos mostrar desde el servidor. Apache es probablemente el servidor web más utilizado del mundo.

[4]Long polling es una variación de la técnica tradicional de polling y permite emular información colocada desde un servidor a un cliente en forma similar al polling normal. Sin embargo, si el servidor no tiene información disponible para el cliente, en vez de enviar una respuesta vacía, el servidor guarda la petición y espera a que alguna información esté disponible. Una vez la información está disponible (o después de un tiempo establecido), se envía una respuesta completa al cliente. Entonces el cliente normalmente realizará un re-pedido de información al servidor, para que éste siempre tenga un pedido en espera, que puede ser usado para responder a un evento.

3.3 Estación Terrena

La función principal de la estación terrena es la adecuación de las señales de radio control para su transmisión a las antenas, desde donde se realiza la radiofusión de las mismas. Dependiendo del tipo de estación, ésta se puede encargar de transmitir y/o recibir información, controlar el estado del UAV y su posición global.

3.3.1 Interface de Usuario

La interfaz de usuario es el medio con que el usuario puede comunicarse con una máquina, un equipo o una computadora, y comprende todos los puntos de contacto entre el usuario y el equipo. Normalmente suelen ser fáciles de entender y fáciles de accionar (aunque en el ámbito de la informática se les denomina como «amigables e intuitivos» pues es muy complejo y subjetivo decir que algo es «fácil»).

Las interfaces básicas de usuario son aquellas que incluyen elementos como menús, ventanas, teclado, ratón, los beeps y algunos otros sonidos que la computadora hace, y en general, todos aquellos canales por los cuales se permite la comunicación entre el ser humano y la computadora. La mejor interacción humano-máquina a través de una interfaz adecuada (Interfaz de Usuario), que le brinde tanto comodidad, como eficiencia.

Sus principales funciones son las siguientes:

- Puesta en marcha y apagado.
- Control de las funciones manipulables del equipo.
- Manipulación de archivos y directorios.
- Herramientas de desarrollo de aplicaciones.
- Comunicación con otros sistemas.
- Información de estado.
- Configuración de la propia interfaz y entorno.
- Intercambio de datos entre aplicaciones.
- Control de acceso.
- Sistema de ayuda interactivo.

Existen muchos tipos de interfaces, pero en nuestro caso solo vamos a hablar de los siguientes tipos:

- Escritorio
- Móvil
- Web

Escritorio

Este tipo de interfaces se realiza para un sistema operativo concreto, como pueda ser Windows, Linux, Mac, Unix, etc. Esto hace que su rendimiento pueda ser más alto que en los otros tipos, pero esto también hace que su versatilidad se reduzca a solo ese tipo de sistema operativo.

Móvil

Las interfaces para móviles tienen varias peculiaridades pero son muy parecidas a las de escritorio. Una de esas peculiaridades es que la resolución varía enormemente de un dispositivo a otro, como es el caso de un móvil y una tablet.

Web

Sin duda, la más versátil de todas. Permite ejecutarse en cualquier dispositivo que disponga de navegador gráfico.

3.3.2 Antena Base

Para la estación terrena se necesitará una antena para recibir los datos de todos los UAVs y poder controlarlos y realizar los pedidos.

Características de las antenas

Una antena es un dispositivo hecho para transmitir (radiar) y recibir ondas de radio (electromagnéticas). Existen varias características importantes de una antena que deben de ser consideradas al momento de elegir una específica para su aplicación:

- Patrón de radiación
- Ganancia
- Directividad
- Polarización

Patrones de Radiación

El patrón de radiación de una antena se puede representar como una gráfica tridimensional de la energía radiada vista desde fuera. Los patrones de radiación usualmente se representan de dos formas, el patrón de elevación y el patrón de azimut. El patrón de elevación es una gráfica de la energía radiada por la antena vista de perfil. El patrón de azimut es una gráfica de la energía radiada vista directamente desde arriba. Al combinar ambas gráficas se tiene una representación tridimensional de como es realmente radiada la energía desde la antena.

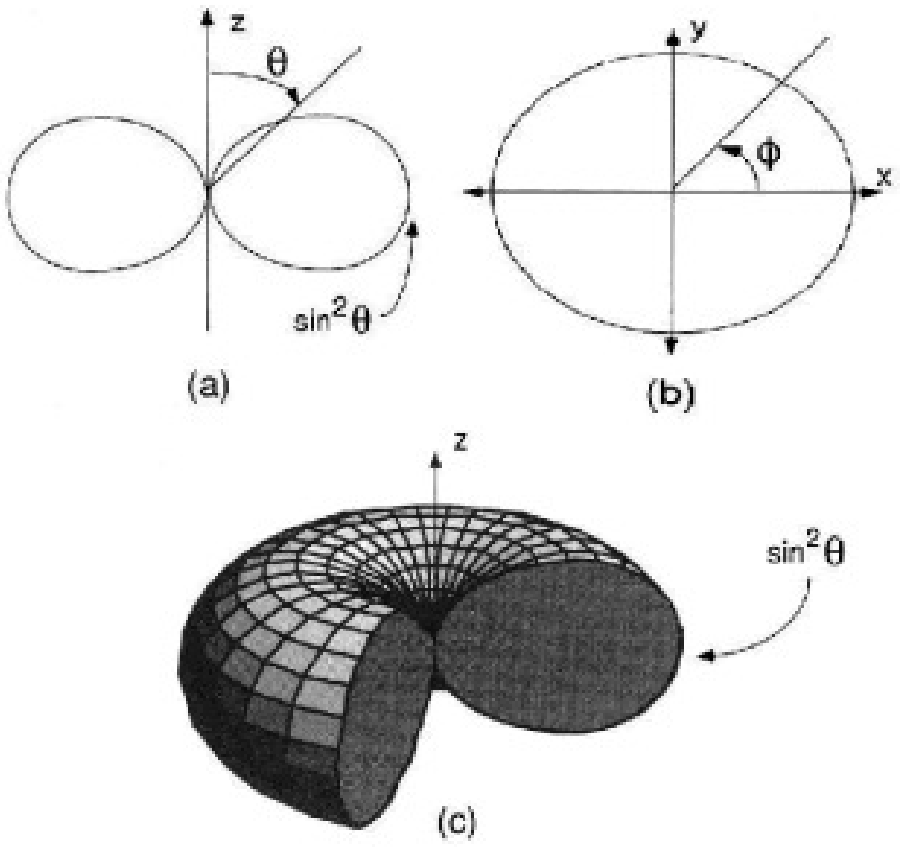

Figura 3.8: Patrones de radiación. a) Patrón de elevación de un dipolo genérico b) Patrón de azimut de un dipolo genérico c) Patrón de radiación 3D

Ganancia:

La ganancia de una antena es la relación entre la potencia que entra en una antena y la potencia que sale de ésta. Esta ganancia es comúnmente referida en dBi, y se refiere a la comparación de cuanta energía sale de la antena en cuestión, comparada con la que saldría de una antena isotrópica. Una antena isotrópica es aquella que cuenta con un patrón de radiación esférico perfecto y una ganancia lineal unitaria.

Directividad:

La directividad de la antena es una medida de la concentración de la potencia de la señal radiada en una dirección particular. Se puede entender también como la habilidad de la antena para direccionar la energía radiada en una dirección especifica. Es usualmente una relación de intensidad de radiación en una dirección particular en comparación a la intensidad promedio isotrópica.

Polarización:

Es la orientación de las ondas electromagnéticas al salir de la antena. Hay dos tipos básicos de polarización que aplican a las antenas, como son: lineal (incluye vertical, horizontal y

oblicua) y circular (que incluye circular derecha, circular izquierda, elíptica derecha, y elíptica izquierda). No se podrá olvidar que tomar en cuenta la polaridad de la antena es muy importante si se quiere obtener el máximo rendimiento de ésta. La antena transmisora debe de tener la misma polaridad de la antena receptora para máximo rendimiento.

Tipos de antenas

Hay varios tipos de antenas. Los más relevantes para aplicaciones en bandas libres son:

- Antenas Dipolo
- Antenas Dipolo multi-elemento
- Antenas Yagi
- Antenas Panel Plano (flat panel)
- Antenas parabólicas (plato parabólico)

Antenas dipolo

Todas las antenas de dipolo tienen un patrón de radiación generalizado. Primero el patrón de elevación muestra que una antena de dipolo es mejor utilizada para transmitir y recibir desde el lado amplio de la antena. Es sensible a cualquier movimiento fuera de la posición perfectamente vertical. Se puede mover alrededor de 45 ^{o} de la verticalidad antes que el desempeño de la antena se degrade más de la mitad. Otras antenas de dipolo pueden tener diferentes cantidades de variación vertical antes que sea notable la degradación.

Un ejemplo de patrón de elevación puede verse en la figura 3.8. A partir del patrón de azimut se ve que las antenas operan igualmente bien en 360 grados alrededor de la antena. Físicamente las antenas dipolo son cilíndricas por naturaleza, y pueden ser ahusadas o con formas específicas en el exterior para cumplir con especificaciones de medidas. Estas antenas son usualmente alimentadas a través de una entrada en la parte inferior, pero también pueden tener el conector en el centro de la misma.

Antenas dipolo multi-elemento

Las antenas multi-elemento tipo dipolo cuentan con algunas de las características generales del dipolo simple. Cuentan con un patrón de elevación y azimut similar al de la antena dipolo simple. La diferencia más clara entre ambas es la direccionalidad de la antena en el plano de elevación, y el incremento en ganancia debido a la utilización de múltiples elementos. Con el uso de múltiples elementos en la construcción de la antena, esta puede ser configurada para diferentes ganancias, lo cual permite diseños con características físicas similares. Tal como se puede ver en el patrón de elevación de la fig. 3.9, múltiples antenas de dipolo son muy direccionales en el plano vertical. Debido a que la antena de dipolo radia igualmente bien en todas las direcciones del plano horizontal, es capaz de operar igualmente

bien en configuración horizontal.

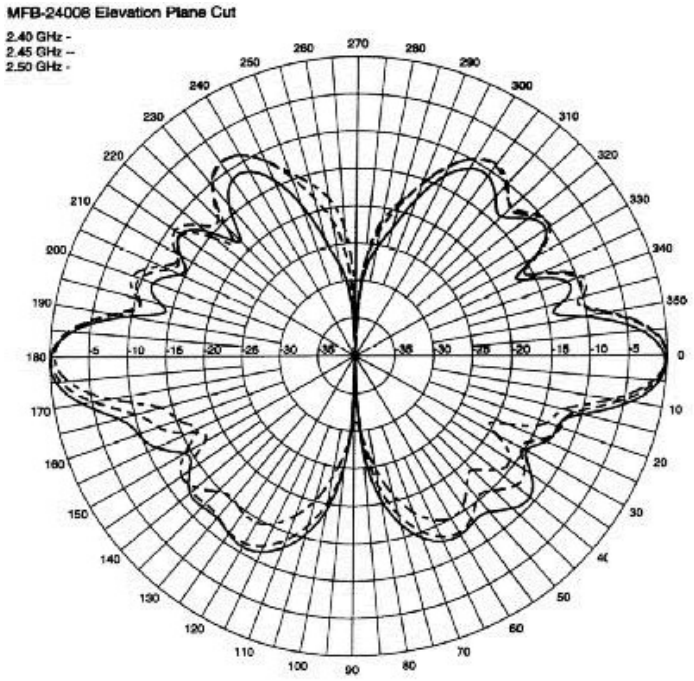

Figura 3.9: Patrón de Elevación multi-dipolo. Patrón de Elevación de una antena multi-dipolo

Antenas Yagi

Se componen de un arreglo de elementos independientes de antena, donde solo uno de ellos transmite las ondas de radio. El número de elementos (específicamente, el número de elementos directores) determina la ganancia y directividad. Las antenas Yagi no son tan direccionales como las antenas parabólicas, pero son más directivas que las antenas panel.

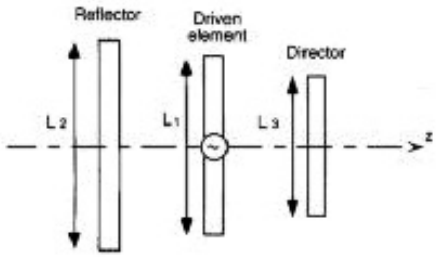

Figura 3.10: Antena Yagi. Construcción de una antena Yagi

Antenas panel plano (flat panel)

Las antenas de panel plano como su nombre indica son un panel con forma cuadrada o rectangular. y están configuradas en un formato tipo patch. Las antenas tipo flat panel son muy direccionales ya que la mayoría de su potencia radiada es una sola dirección ya sea en

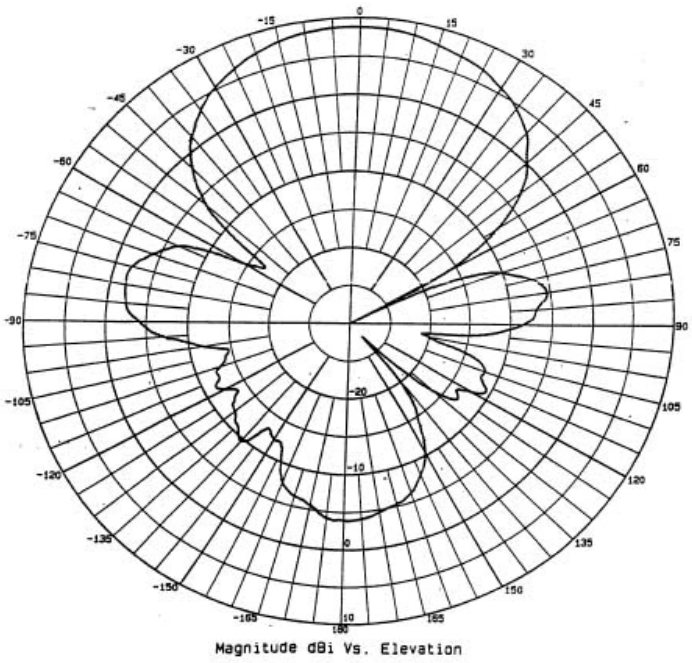

Figura 3.11: Patrón de elevación Yagi. Patrón de radiación en elevación Yagi

el plano horizontal o vertical. En el patrón de elevación (Fig. 3.12) y en el patrón de azimut (Fig. 3.13) se puede ver la directividad de la antena flat panel. Las antenas flat panel pueden ser fabricadas en diferentes valores de ganancia de acuerdo a su construcción. Esto puede proveer excelente directividad y considerable ganancia.

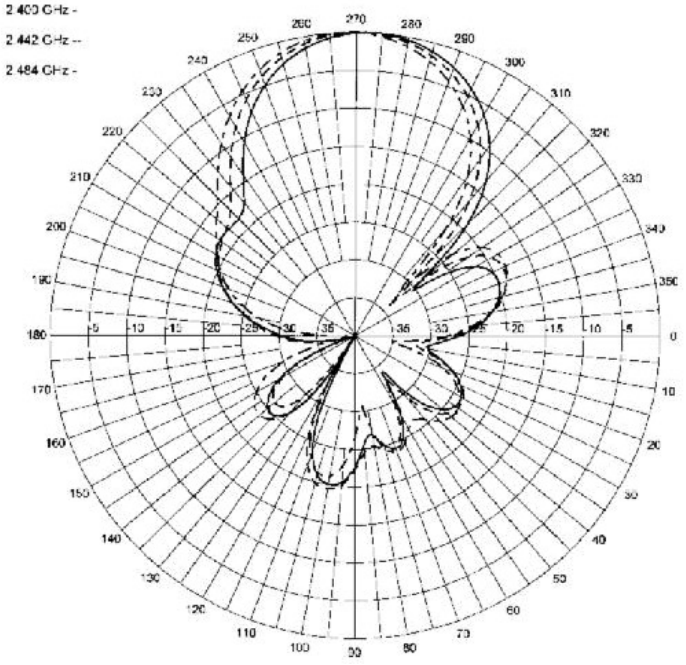

Figura 3.12: Patrón de elevación flat panel. Patrón de elevación flat panel de alta ganancia

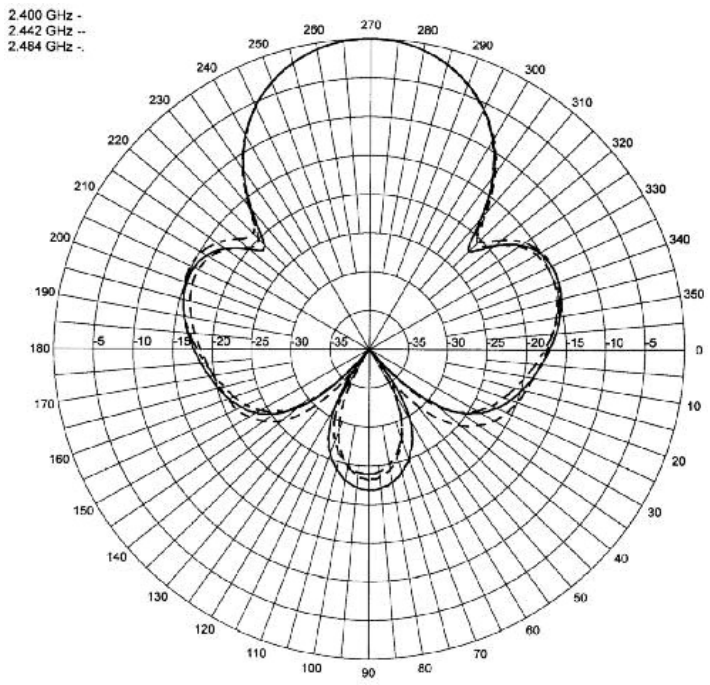

Figura 3.13: Patrón de Azimut flat panel. Patrón de Azimut flat panel de alta ganancia

Antenas parabólicas

Las antenas parabólicas usan características físicas así como antenas de elementos múltiples para alcanzar muy alta ganancia y direccionalidad. Estas antenas usan un plato reflector con la forma de una parábola para enfocar las ondas de radio recibidas por la antena a un punto focal. La parábola también funciona para capturar la energía radiada por la antena y enfocarla en un haz estrecho al transmitir. Como puede verse en la Figura 3.13, la antena parabólica es muy direccional. Al concentrar toda la potencia que llega a la antena y enfocarla en una sola dirección, este tipo de antena es capaz de proveer muy alta ganancia.

Antena de ranura

Las antenas de ranura cuentan con características de radiación muy similares a las de los dipolos, tales como los patrones de elevación y azimut, pero su construcción consiste solo de una ranura estrecha en un plano. Así como las antenas microstrip mencionadas abajo, las antenas de ranura proveen poca ganancia, y no cuentan con alta direccionalidad, como evidencian su patrones de radiación y su similiridad al de los dipolos. Su característica más atractiva es la facilidad de construcción e integración en diseños existentes, así como su bajo costo. Estos factores compensan por su desempeño poco eficiente.

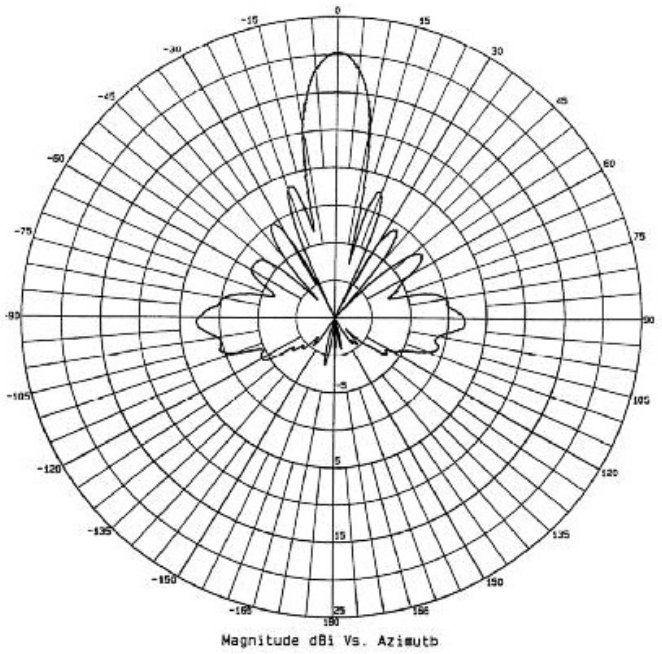

Figura 3.14: Patrón de elevación parabólica, patrón de elevación de plato parabólico

Antenas microstrip

Estas antenas pueden ser hechas para emular cualquiera de los diferentes tipos de antenas antes mencionados. Las antenas microstrip ofrecen varios detalles que deben de ser considerados. Debido a que son manufacturadas con pistas en circuito impreso, pueden ser muy pequeñas y livianas. Esto tiene como costo no poder manejar mucha potencia como es el caso de otras antenas, además están hechas para rangos de frecuencia muy específicos. En muchos casos, esta limitación de frecuencia de operación puede ser benéfico para el desempeño del radio. Debido a sus características las antenas microstrip no son muy adecuadas para equipos de comunicación de banda amplia.

Conclusión

De esta introducción básica a las antenas, podemos obtener una comprensión simple de los tipos de antenas y aplicaciones de estas. Por ejemplo, las antenas dipolo aún cuando no proveen mucha ganancia ofrecen la mejor flexibilidad en cuanto a orientación de la antena. Las antenas flat panel ofrecen mayor direccionalidad y son buena opción para instalaciones fijas. La antena parabólica con su alta ganancia y gran direccionalidad son muy adecuadas para proveer enlaces punto a punto en largas distancias, con antenas instaladas permanentemente. Finalmente las antenas de ranura y las de microstrip son correctas para aplicaciones de desempeño moderado que necesitan integrar la antena dentro del radio y aplicaciones OEM. Adicionalmente es posible usar diferentes tipos de antena en el mismo sistema. Por ejemplo, se puede montar una antena flat panel en una pared cerca de un acceso punto Cuando una

pieza de equipo con antena dipolo cerca del punto de acceso, el sistema podría actualizar estadísticas inmediatamente en el equipo.

Para ayudar en la elección de la antena correcta para su aplicación, la tabla 3.4 se provee como un medio de comparación entre los diferentes tipos:

	Patrón de Radiación	**Ganancia**	**Directividad**	**Polarización**
Dipolo	Amplio	Baja	Baja	Lineal
Dipolo Multi-Elemento	Amplio	Baja/Media	Baja	Lineal
Panel Plano (Flat Panel)	Amplio	Media	Media/Alta	Lineal/Circular
Plato Parabólico	Amplio	Alta	Alta	Lineal/Circular
Yagi	Amplio	Media/Alta	Media/Alta	Lineal
Ranura	Amplio	Baja/Media	Baja/Media	Lineal
MicroStrip	Amplio	Media	Media	Lineal

Cuadro 3.4: Comparación entre los distintos tipos de antenas (TIPOS DE ANTENAS)

3.3.3 Raspberry Pi

La raspberry Pi es un ordenador de bajo costo, del tamaño de una tarjeta de crédito que se conecta a un monitor de ordenador o televisor, y utiliza un teclado y un ratón estándar. Es un pequeño dispositivo que permite a personas de distintas edades explorar el mundo de la computación, y para aprender a programar en lenguajes como Scratch y Python. Es capaz de hacer todo lo que se espera de un ordenador de escritorio, desde navegar por Internet y reproducir vídeo de alta definición a realizar hojas de cálculo, procesador de texto y jugar a juegos.

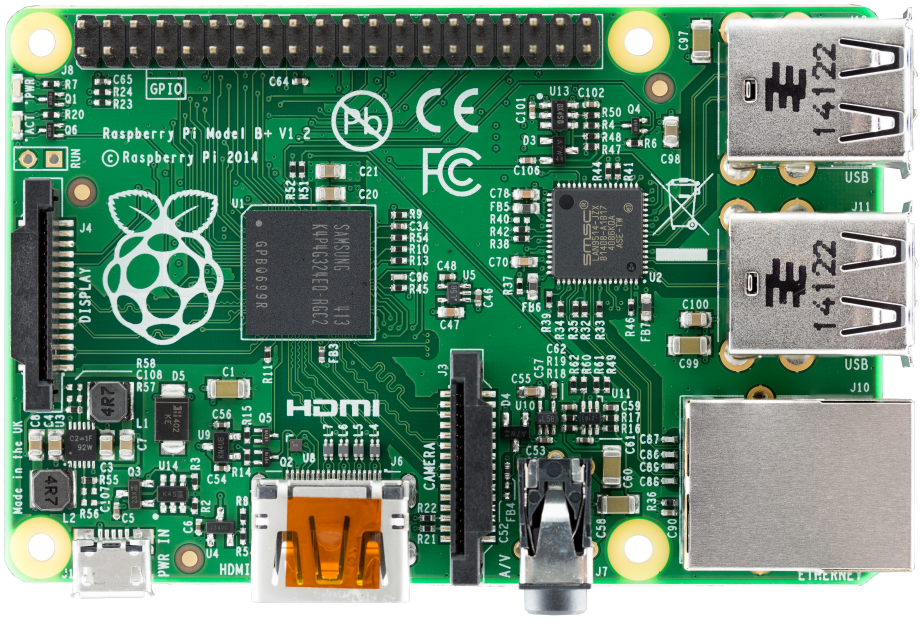

Figura 3.15: Raspberry Pi

Lo que es más, la Raspberry Pi tiene la capacidad de interactuar con el mundo exterior, y se ha utilizado para fabricar una amplia gama de proyectos digitales, desde máquinas de música, detectores de padres, a estaciones meteorológicas y pajareras que tuitean con cámaras de infrarrojos.

3.3.4 S.O. Raspbian

Raspbian es un sistema operativo libre basado en Debian optimizado para la Rapberry Pi. Un sistema operativo es un conjunto básico de programas y utilidades que hacen funcionar la Raspberry Pi. Sin embargo, Raspbian proporciona más que un puro Sistema Operativo, viene con más de 35000 paquetes, paquetes software pre-compilado en un formato que hace más fácil la instalación en la Raspberry Pi.

3.4 Seguridad

Para asegurar la seguridad debemos alcanzar los siguienetes objetivos de integridad, disponibilidad y confidencialidad en todos los componentes de nuestro sistema, como son el hardware, software, comunicaciones, y firmware. [INF14]

Como ya habrá podido comprobar el lector, el sistema abarca multitud de componentes y elementos, que debemos asegurar y garantizar su seguridad. A continuación se detallan todos los elementos y características utilizadas. Otros aspectos importantes que debemos garantizar son la autenticidad y responsabilidad.

Comenzaremos hablando de la seguridad fisica de los componentes hardware, para ello debemos garantizar que el usuario y personas no autorizadas esten seguras.

3.4.1 Hélices

Este es un elemento muy peligroso debido a su forma de cuchillas y su gran velocidad de giro, que puede ocasionar heridas y cortes en extremidades. Para evitar esto se ha diseñado la carcasa externa de tal forma que ningun elemento extraño pueda afectar o tocar las helices. No solo la carcasa externa protegen las helices, sino también unas mallas metalizas con forma de red, evitan su contacto.

3.4.2 Peso del UAV

El propio peso de la aeronave es un factor importante de seguridad, ya que a mayor peso, mayor será el impacto. Esto también se ha tenido en cuenta en el diseño, y gracias a su forma y su peso ligero, hace que en el caso de que entre en caida libre, su caida sea muy despacio y no cause grandes daños.

3.4.3 Paracaidas

Se dispone opcionalmente de un paracaidas en el UAV, que garantiza aun más que su caida sea muy suave y sin daño alguno. Este elemento está conectado a un watch-dog de la placa

controladora de vuelo, que en el caso de que no reciba señal el paracaidas del UAV saltará y se habrirá el paracaidas, esto nos asegura que si el UAV se quedase sin batería (este es el peor de los casos posibles) se abra el paracaidas. También se dispone de un sistema automático de emergencias, si detecta que entra en caida libre, el propio sistema de la controladora de vuelo enviará una señal para que el paracaidas se abra.

3.4.4 Alarma Emergencia

Se dispone opcionalmente también de un sistema de sónido, con unos mensajes de emergencia adaptados al tipo de transporte, para que en el caso de accidente el UAV emita estos mensajes para evitar que personal no autorizado se acerque y entre en contacto con la aeronave. Evitando así daño no deseados.

3.4.5 UAV

Se disponen de 2 elementos de seguridad como son los rally y los fence. Los rally's son puntos de aterrizaje de emergencia, donde un UAV puede aterrizar en casa de emergencia. Estos puntos estaran configurados cerca las trayectorias de los pedidos, para en el caso de que el UAV tenga algún problema, como quedarse sin batería, pueda realizar un aterrizaje seguro donde el operario pueda recogerlo sin problemas y riesgos. También se disponen de fence, que son zonas de exclusión donde el UAV no podrá volar. Como por ejemplo, si hemos configurado una zona fence como pueda ser un colegio, podemos evitar que sobre vuele su espacio aereo. Si dentro de la ruta de envío, el UAV detecta una zona fence, realizará los cambios necesarios para no sobre volar ese area.

3.4.6 Hardware

La seguridad en los elementos hardware es algo que se debería realizar en trabajos futuros, aunque no es algo prioritario, ya que para acceder al UAV debes tener acceso físico. Aunque a traves de la telemetría podríamos acceder a él.

3.4.7 Comunicaciones

Cuando hablamos de comunicaciones nos referimos al protocolo MavLink utilizado entre los UAV's y la estación terrena. Para garantizar la seguridad se pueden utilizar dos opciones. Una, utilizar una contraseña cada vez que se inicia el sistema. Y otra, cifrar todos los mensajes para evitar su monitorización y posible captura.

3.4.8 Servidores

En los sevidores se ha utilizado multiples medidas de seguridad. Como son el control de acceso mediante usuario y contraseña al acceder e iniciar el sistema, también para conectarnos a la red wifi con nuestro disposito, debemos introducir otra contraseña de nivel WPA 2. Para que los clientes puedan acceder se utilizan par de claves, pública y privada.

3.4.9 Web

Para el acceso a la aplicación web se utiliza también un control de acceso, pero solo para la parte de gestión, ya que para el area de operaciones, por hacer más comodo y accesible se ha suprimido su control de acceso.

Capítulo 4

Método de trabajo

PARA el desarrollo de todo el proyecto se han utilizado diferentes metodologías de trabajo, ya que por un lado se necesita diseñar un UAV que es un dispositivo hardware, y por otro lado se necesita desarrollar el software necesario para su control y una aplicación web para la gestión de los pedidos.

No se ha seguido ninguna metodología estricta, debido a la gran variedad de componentes y sistemas necesarios. Para el desarrollo hardware se ha utilizado una metodología top-down.

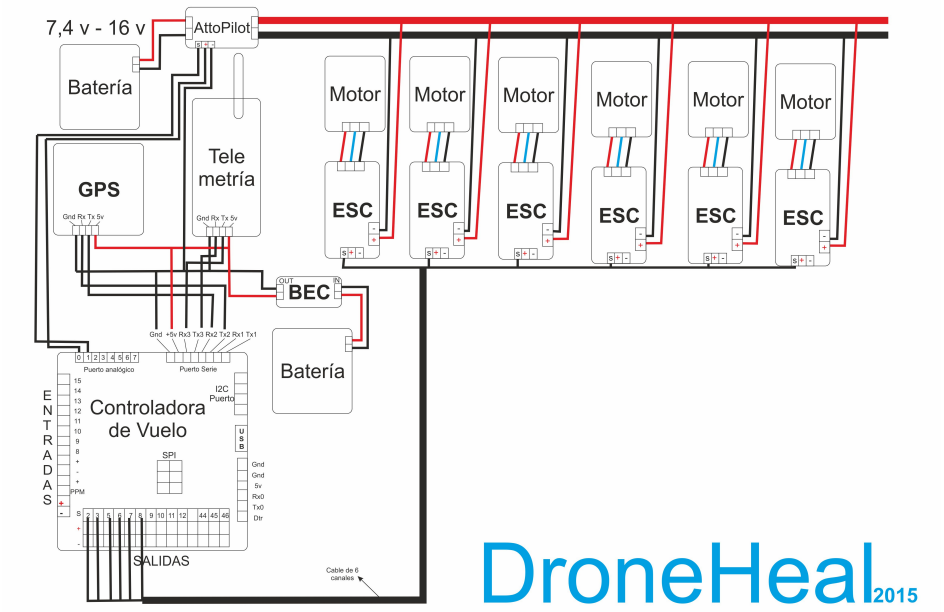

Figura 4.1: Diagrama de todos los componentes del sistema

4.1 Hardware

Para el desarrollo de sistemas hardware se utilizan lenguajes de descripción como Lenguaje de Descripción Hardware (HDL), que es un lenguaje de programación especializado que se utiliza para definir la estructura, diseño y operación de circuitos electrónicos, y más comúnmente, de circuitos electrónicos digitales. Así, los lenguajes de descripción de hardware

hacen posible una descripción formal de un circuito electrónico, y posibilitan su análisis automático y su simulación.

4.1.1 Top-Down

Se comienza a nivel de sistema con un módelo de computación y se genera a partir de aquí una plataforma o estructura del sistema donde cada componente tiene sus parámetros y métricas definidas.

Se deja el proceso de ubicación y enrutado para el último paso evitando hacer una implementación a niveles superiores de abstracción. Las métricas de los componentes y del sistema se conocen al final haciendo muy difícil optimizar el diseño completo. Se repite el proceso de descomposición y de síntesis.

Para evitar demasiadas iteraciones se necesitan métricas de niveles inferiores de abstracción para crear una estimación de métricas a niveles superiores que será usado en las iteraciones siguientes.[Don15]

4.2 Software

Un modelo para el desarrollo de software es una representación abstracta de un proceso. Cada modelo representa un proceso desde una perspectiva particular y así proporciona información parcial sobre el proceso. Estos modelos generales no son descripciones definitivas de los procesos del software, más bien son abstracciones de los procesos que se pueden utilizar para el desarrollo del software. Puede pensarse en ellos como marcos de trabajo del proceso y que pueden ser adaptados para crear procesos más específicos.

Se han utilizado varios modelos, a nivel global de todo el software se ha utilizado el modelo en cascada, siguiendo casi todas sus fases. A este modelo le hemos integrado el modelo de desarrollo por componentes, ya que disponíamos de varios componentes ya desarrollados.

4.2.1 Modelo en cascada

Considera las actividades fundamentales del proceso especificación, desarrollo, validación y evolución. Los representa como fases separadas del proceso, tales como la especificación de requerimientos, el diseño del software, la implementación, las pruebas.

- Análisis y definición de requerimientos. Los servicios restricciones y metas del sistema se definen a partir de las consultas con los usuarios. Entonces, se definen en detalle y sirven de manera específica al sistema.
- Diseño del sistema y del software. El proceso de diseño del sistema divide los requerimientos en sistemas ya sea hardware o software. Establece una arquitectura completa del sistema, el diseño del software identifique describe los elementos abstractos que son fundamentales para el software y sus relaciones.

- Implementaciones de prueba unitarias. Durante esta etapa el diseño del software se lleva a cabo como un conjunto de programas, la prueba unitaria implica verificar que cada una cumpla con su función específica.
- Integración y prueba del sistema. Los programas o las unidades individuales de programas se integran y se prueban como un sistema completo para así asegurar que se cumplan los requerimientos del software, después se entrega al cliente.
- Funcionamiento y mantenimiento. En esta fase el sistema se instala y se pone en funcionamiento práctico el mantenimiento implica corregir errores no descubiertos en las etapas anteriores del ciclo de vida, mejorar la implementación de las unidades del sistema y resaltar los servicios del sistema una vez que se descubren en nuevos requerimientos.

4.2.2 Modelo de desarrollo basado en componentes

En la mayoría de los proyectos de desarrollo de software existe la reutilización. Por lo general esto sucede informalmente cuando las personas conocen diseños o códigos similares al requerido. Los buscan, los modifican según lo creen necesario y los incorporan en un nuevo sistema. En el enfoque evolutivo, la reutilización es indispensable para el desarrollo más ágil de un sistema. Esta reutilización es independiente del proceso de desarrollo que se utilice. Sin embargo, en los últimos años ha surgido un enfoque de desarrollo de software denominado «ingeniería de software basada en componentes», el cual se basa en la reutilización. Este enfoque se basa en la reutilización y se compone de una gran base de componentes de software que son reutilizables.

Aunque la etapa de especificación de requerimientos y la revalidación son comparables con otros procesos, las etapas intermedias en el proceso orientado a la reutilización son diferentes. Estas etapas son:

- Análisis de componentes. En esta se buscan los componentes para implementar los con base en su especificación. Por lo general, no existe una concordancia exacta y los componentes que se utilizan sólo proporcionan parte de la funcionalidad requerida.
- Modificación de requerimientos. En esta etapa los requerimientos se analizan utilizando información acerca de los componentes que se han descubierto. Entonces dichos componentes se modifican para reflejar los componentes disponibles, la actividad de análisis de componentes se puede llevar a cabo para buscar soluciones alternativas.
- Diseño del sistema con reutilización. En esta fase los diseñadores tienen en cuenta los componentes que se reutiliza y que se organizan el marco de trabajo para que los satisfaga. Si dichos componentes no están disponibles se puede diseñar nuevos software.
- Desarrollo e integración. El software que no se puede adquirir externamente se desa-

rrolla y se integra a los componentes. En este modelo, la integración del sistema es parte del proceso de desarrollo, más que una actividad separada.

4.3 Lenguajes de programación

- Python - Para los servidores web, para el backend de la aplicación web, Estación terrena, Wise y DroneApi.
- Javascript - Para el frontend de la aplicación web.
- C++ - Para el firmware del UAV
- HTML - Para la web.
- Shell Script - Para lanzarlo todo.

4.4 Programas SW y herramientas HW utilizadas en el proyecto

A continuación se detallan cada uno de los programas externos que forman parte del sistema.

- Gunicorn - Servidor web
- Django - Framework de desarrollo web
- Wise - Framework de comunicación web orientado a objetos
- OpenLayers - Biblioteca de funciones para todas tus necesidades de mapeo, con alto rendimiento.
- ArduCopter - Firmware UAV.
- MegaPirateNG - Firmware UAV.
- Bootstrap - Framework para desarrollo sensible a las diferentes resoluciones de pantalla.
- MavProxy - Estación terrena para UAV.
- DroneApi - Módulo para MavProxy.
- Sqlite - Base de datos.

También se han utilizado herramientas como Corel Draw que tienen una licencia privativa. Para la edición de imagenes y su maquetación. El resto de programas y herramientas han sido abiertas. Como son Wireshark para el análisis de las comunicaciones entre los elementos del sistema. Se ha utilizado Chromium como navegador para la ejecución de pruebas como para el propio desarrollo del proyecto y para la busqueda de información.

Capítulo 5

Desarrollo

En este apartado en discutir los aspectos referentes a análisis de la solución elegida, su diseño, implementación, pruebas y su posterior mantenimiento si lo hubiese.

Diferenciaremos 3 tipos de desarrollo, por un lado, el desarrollo de componentes software estrictamente, como puede ser el servidor de datos GPS, por otro, el desarrollo de componentes hardware, aunque más bien sea una configuración y modificación de su firmware. Y por último el desarrollo de estructuras físicas, como el frame del multicopter, su carcasa externa, etc.

5.1 Análisis de Requisitos

Debe poder transportar paquetes de pequeño volumen (20 cm x 20 cm x 20 cm) y peso ligero de entre 1 kg y 1,5 kg. A una distancia de unos 10 km max. Esto en principio que supone, que el peso util de la aeronave más su carga no debe superar la mitad del peso total de empuje. Con lo cual si nuestra aeronave pesa 1 kg aprox. y necesitamos una carga de 1-1.5 kg, necesitaremos una aeronave con unos 4.5 kg de empuje. Para poder cumplir con el requisito de distancia necesitamos saber la velocidad de crucero de la aeronave y el tiempo de duración de las baterias. En aeronautica se establece la siguiente regla en cuestión de autonomía. La capacidad de energía ya sea fosil o de baterias, debe ser suficiente como realizar 3 veces la distancia de la ida. Esto quiere decir que si queremos tener una autonomia de 10 km, la aeronave debe tener por lo menos autonomia para 15 km. Con las baterias actuales tenemos una autonomia de 30 minutos. Debemos conocer la velocidad real para concretar su autonomia.

Una vez que tenemos resueltos estos requisitos, debemos ocuparnos de subobjetivos secundarios originados por estos.

- Por un lado debemos fabricar una aeronave que cumpla con estos requisitos.
- Por otro, encontrar o diseñar un sistema de control para aeronaves.
- Por último, diseñar todos los sistemas necesarios para disponer de toda la información, como son servidores de base de datos, web, GPS.

5.2 Desarrollo Software

Para el desarrollo software se ha seguido una metodología agil, dentro de las propias limitaciones del proyecto. Esto hace que muchas características e instrumentaciones sean dificiles de aplicar debido a la diversidad de disciplinas y componentes, y también que para todo el desarrollo solo ha sido realizado por una persona. Para conseguir esta metodología de desarrollo de software se ha tenido que hacer una adaptación más orientada a la organización del trabajo y la manera de enfrentarse a los problemas intentando seguir los principios del manifiesto ágil.

- La mayor prioridad es satisfacer al cliente mediante la entrega temprana y continua de software con valor.
- Se acepta que los requisitos cambien, incluso en etapas tardías del desarrollo. Los procesos Ágiles aprovechan el cambio para proporcionar ventaja competitiva al cliente.
- Se entrega software funcional frecuentemente, entre dos semanas y dos meses, con preferencia al periodo de tiempo más corto posible.
- Los responsables de negocio y los desarrolladores trabajan juntos de forma cotidiana durante todo el proyecto.
- Los proyectos se desarrollan en torno a individuos motivados. Hay que darles el entorno y el apoyo que necesitan, y confiarles la ejecución del trabajo.
- El método más eficiente y efectivo de comunicar información al equipo de desarrollo y entre sus miembros es la conversación cara a cara.
- El software funcionando es la medida principal de progreso.
- Los procesos Ágiles promueven el desarrollo sostenible. Los promotores, desarrolladores y usuarios debemos ser capaces de mantener un ritmo constante de forma indefinida.
- La atención continua a la excelencia técnica y al buen diseño mejora la Agilidad.
- La simplicidad, o el arte de maximizar la cantidad de trabajo no realizado, es esencial.
- Las mejores arquitecturas, requisitos y diseños emergen de equipos auto-organizados.
- A intervalos regulares el equipo reflexiona sobre cómo ser más efectivo para a continuación ajustar y perfeccionar su comportamiento en consecuencia.

5.2.1 Iteración 0

Esta primera iteración consiste en integrar tres tipos de componentes, como son el servidor web, en nuestro caso Gunicorn, el framework de desarrollo web, en nuestro caso Django, y por último Wise, que es un framework de comunicación web.

La primera tarea a cumplir es comunicar el servidor web con el cliente web. Para ello se deberá invocar un método del cliente, en el servidor.

Se deberá lanzar el servidor Gunicorn mediante un script, dispone de multitud de opciones de configuración que estan detalladas en su manual, como el tipo de worker, la aplicación que lanzará. Una vez lanzado Gunicorn, estará sirviendo páginas de Django, en la IP y puerto que se le asigne.

Solo falta unirlo con Wise. Para esto solo hay que crear un archivo llamado servants.py en el directorio de nuestra aplicación. Django es uno de los framework web más utilizado del momento. Por un lado se crean proyectos, los cuales disponen de todas las aplicaciones posibles, que a su vez se pueden intercambiar de un proyecto a otro, porque en Django todo es modular y reutilizable. Una vez que se ha creado este archivo se debe crear un método llamado inizialize(broker) al cual se le pasará un broker de la comunicación. Aquí es donde se inicia la comunicación con Wise, y aquí es donde debemos definir que tipo de comunicación se quiere o se necesita. Si se quiere que sea del servidor al cliente o al contrario.

Hay que tener claro varios conceptos, como son el cliente y el servidor. El cliente en una comunicación es siempre quien solicita la información y el servidor quien la sirve. Sin embargo, cuando se hable de cliente web se referirá al navegador, y cuando se hable de servidor web se referirá al propio servidor web o Gunicorn.

Llegados a este punto, ya tenemos la columna vertebral de nuestro sistema y podemos continuar con los siguientes pasos.

5.2.2 Iteración 1

Integración con Mavproxy

MavProxy es una estación terrena por línea de comandos que utiliza el protocolo de comunicación MavLink entre los UAV y la aplicación. En el capítulo 3 se detalló su funcionamiento. Para adaptarlo a nuestro proyecto y poderlo integrar necesitamos que no se quede a la espera de ningún comando para utilizarlo como una librería de Python. Para esto hemos eliminado el siguiente código.

```
1       # while not mpstate.input_queue.empty():
2       #     line = mpstate.input_queue.get()
3       #     mpstate.input_count += 1
4       #     cmds = line.split(';')
5       #     if len(cmds) == 1 and cmds[0] == "":
6       #           mpstate.empty_input_count += 1
7       #     for c in cmds:
8       #         process_stdin(c)
```

Este bucle lo que hace es estar a la espera de la linea de comando, mientras la cola de entrada de comandos no este vacía. También se ha cambiado el método main(), eliminando todo su contenido y se ha convertido en global la variable parser.

Conexión con UAV

Para conectar el UAV a la estación terrena se pueden utilizar diferentes sistemas como radio telemetría, bluetooth, radio control, GSM y satélite. En este caso se ha utilizado radio telemetría que funciona a 433 MHz.

Cuando se ha seleccionado el tipo de conexión entre el UAV y la estación terrena solo falta indicárlo al sistema. Para ello se debe hacer lo siguiente. Según el tipo de conexión se usará un comando:

Tipo de Conexión	Comando MAVProxy
Linux PC a vehiculo USB	–master=/dev/ttyUSB0
Linux PC a vehiculo Serial	–master=/dev/ttyAMA0 –baudrate 57600
SITL a vehiculo UDP	–master=127.0.0.1:14550
OSX PC a vehiculo USB	–master=/dev/cu.usbmodem1
Windows PC a vehiculo USB	–master=/dev/cu.usbmodem1

Cuadro 5.1: Tipos de conexión

El comando se introducirá en la linea número 838:

```
(opts, args) = parser.parse_args(args="--master=/dev/ttyUSB0".split())
```

Protocolo MAVLink

Como ya se explico en el capitulo 3, por eso solo se explicará como se construye un mensaje MAVLink.

Los mensajes MAVLink están definidos en Lenguaje extensible de marcas (XML) y se convierten a código C/C++, C# o Python. El proceso de añadir un mensaje se explica a continuación.

Se debe tener en cuenta que el mensaje heartbeat es el único mensaje obligatorio, todos los demás son opcionales.

A continuación se muestra la definición de un mensaje en XML, parte del archivo mavlink/message_definitions/common.xml.

```
<message id="0" name="HEARTBEAT">
    <description>El mensaje heartbeat muestra que un sistema esta presente y respondiendo. El
        tipo de hardware MAV y Autopilot permite al sistema receptor tratar mensajes de otros
        sistemas.</description>
    <field type="uint8_t" name="type">Tipo de MAV (quadrotor, helicopter, etc., mas de 15 tipos,
         definido en MAV_TYPE ENUM)</field>
    <field type="uint8_t" name="autopilot">Autopilot tipo / clase. Definido en MAV_CLASS ENUM</
        field>
    <field type="uint8_t" name="base_mode">Modo sistema bitfield, mira MAV_MODE_FLAGS ENUM en
        mavlink/include/mavlink_types.h</field>
```

```
6       <field type="uint32_t" name="custom_mode">Modo navegacion bitfield, mira
            MAV_AUTOPILOT_CUSTOM_MODE ENUM para algunos ejemplos. Este campo es especificado por
            autopilot.</field>
7       <field type="uint8_t" name="system_status">Bandera de estado del sistema, mira MAV_STATUS
            ENUM</field>
8       <field type="uint8_t_mavlink_version" name="mavlink_version">MAVLink version</field>
9   </message>
```

Las diferentes partes del XML:

- Cada mensaje se delimita con <message></message>.
- id=0 Esto significa que este mensaje tiene el id/index número 0. El rango válido de números es del 0 al 255, los id's del 150 al 240 están reservados para extensiones (serán necesarios para realizar mensajes personalizados).
- name=HEARTBEAT Codifica el nombre legible por humanos. El nombre se utiliza solamente en el código y no se transmite. Es la manera en la que el sistema lo identifica.
- <description></description>Es muy importante, pero es opcional. Esta información es mostrada en la interfaz de usuario y en los comentarios del código. Debe contener toda la información (e hipervínculos) para comprender plenamente el mensaje.
- <field></field>Codifica un campo del mensaje. Es similar a una variable de una estructura C. Los campos pueden ser números enteros de 8, 16, 32 y 64 bits de longitud (con y sin signo) y número de punto flotante de doble y simple precisión IEEE754.
- type=uint8_t Define este campo como un entero sin signo de 8 bits de. Arrays pueden ser definidos como type=uint8_t[5] para un array de tamaño 5. El type *uint8_t_mavlink_version* es un tipo especial. Codifica un número de 8 bits sin signo que sostiene la version del protocolo actual. Este campo es de solo lectura y se fija automáticamente por MAVLink durante la transmisión. Esto permite al receptor decodificar la versión del protocolo.

Definición de mensajes personalizados

Para definir mensajes personalizados dentro del piloto automático, se debe organizar en un archivo de definición de mensaje único. A continuación se muestra un ejemplo de archivo XML de definición de mensaje, es un esqueleto. Si este archivo está en el mismo directorio que el archivo common.xml, el contenido del archivo será incluido en el final del código generado por MAVLink durante la transmisión. Esto permite al receptor decodificar la version del protocolo.

```
1   <?xml version="1.0"?>
2   <mavlink>
3       <include>common.xml</include>
4       <!-- NOTE: If the included file already contains a version tag, remove the version tag here,
                else uncomment to enable. -->
```

```
5       <!--<version>3</version>-->
6       <enums>
7       </enums>
8       <messages>
9               <message id="150" name="RUDDER_RAW">
10                      <description>This message encodes all of the raw rudder sensor data from the
                             USV.</description>
11                      <field type="uint16_t" name="position">The raw data from the position sensor
                            , generally a potentiometer.</field>
12                      <field type="uint8_t" name="port_limit">Status of the rudder limit sensor,
                            port side. 0 indicates off and 1 indicates that the limit is hit. If
                            this sensor is inactive set to 0xFF.</field>
13                      <field type="uint8_t" name="center_limit">Status of the rudder limit sensor,
                             port side. 0 indicates off and 1 indicates that the limit is hit. If
                            this sensor is inactive set to 0xFF.</field>
14                      <field type="uint8_t" name="starboard_limit">Status of the rudder limit
                            sensor, starboard side. 0 indicates off and 1 indicates that the limit
                            is hit. If this sensor is inactive set to 0xFF.</field>
15              </message>
16      </messages>
17</mavlink>
```

Compilar XML a C/C++ o Python

Después de guardar este archivo de definición de mensajes, puede ser compilado en código C. Este proceso se describe a continuación.

```
$ git clone https://github.com/mavlink/mavlink mavlink-generator
$ cd mavlink-generator
$ python generate.py
```

Se abrirá una interfaz gráfica de usuario de Python que le permite seleccionar los archivos de entrada y salida de los directorios apropiados. Después de compilar, el código, la estructura C resultante es la siguiente:

```
#define MAVLINK_MSG_ID_HEARTBEAT 0

3typedef struct __mavlink_heartbeat_t
4{
5 uint32_t custom_mode; ///< Navigation mode bitfield, see MAV_AUTOPILOT_CUSTOM_MODE ENUM for some
      examples. This field is autopilot-specific.
6 uint8_t type; ///< Type of the MAV (quadrotor, helicopter, etc., up to 15 types, defined in
      MAV_TYPE ENUM)
7 uint8_t autopilot; ///< Autopilot type / class. defined in MAV_CLASS ENUM
8 uint8_t base_mode; ///< System mode bitfield, see MAV_MODE_FLAGS ENUM in mavlink/include/
      mavlink_types.h
9 uint8_t system_status; ///< System status flag, see MAV_STATUS ENUM
10 uint8_t mavlink_version; ///< MAVLink version
11} mavlink_heartbeat_t;
```

Además MAVLink genera funciones para serializar (empaquetado) y deserializar (desempaquetado) mensajes.

```
static inline uint16_t mavlink_msg_heartbeat_pack(uint8_t system_id,
       uint8_t component_id, mavlink_message_t* msg, uint8_t type, uint8_t autopilot,
       uint8_t base_mode, uint32_t custom_mode, uint8_t system_status)

/**
 * @brief Encode a heartbeat struct into a message
 *
 * @param system_id ID of this system
 * @param component_id ID of this component (e.g. 200 for IMU)
 * @param msg The MAVLink message to compress the data into
 * @param heartbeat C-struct to read the message contents from
 */
static inline uint16_t mavlink_msg_heartbeat_encode(uint8_t system_id,
       uint8_t component_id, mavlink_message_t* msg, const mavlink_heartbeat_t* heartbeat)
```

Y por supuesto, también para decodificar el contenido del mensaje.

```
/**
 * @brief Decode a heartbeat message into a struct
 *
 * @param msg The message to decode
 * @param heartbeat C-struct to decode the message contents into
 */
static inline void mavlink_msg_heartbeat_decode(const mavlink_message_t* msg, mavlink_heartbeat_t*
     heartbeat)
```

5.2.3 Iteración 2

Desarrollo Web

Como ya se dijo anteriormente, para el desarrollo de la aplicación se decidió realizar con el framework de desarrollo web Django. Se ha usado este framework por ser modular, reutilizable y estar escrito en python. Django permite construir en profundidad, de forma dinámica, sitios interesantes en un tiempo extremadamente corto. Django está diseñado para hacer foco en la diversión, en las partes interesantes de tu trabajo, al mismo tiempo que alivia el dolor de las partes repetitivas.

Django utiliza el patrón Modelo Vista Controlador (MVC), con algunas peculiaridades como son: Los controladores son las vistas en Django, o sea en el fichero view.py se escriben los controladores de tu aplicación. Las vistas son las plantillas en Django, el cual dispone de un lenguaje especifico para su construcción. Los modelos son en realidad la base de datos.

Para comenzar el desarrollo de la aplicación lo primero que se debe hacer es crear un proyecto:

```
$ django-admin.py startproject nombre_de_tu_proyecto
```

Este comando creara los siguientes archivos y directorios:

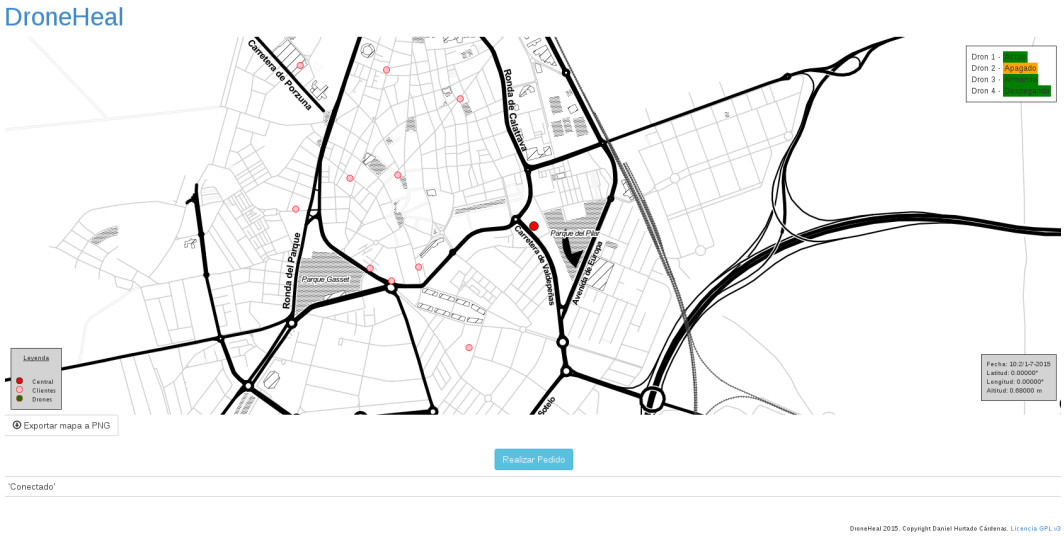

Figura 5.1: Captura de pantalla de la aplicación.

```
nombre_de_tu_proyecto/
        __init__.py
        manage.py
        settings.py
        urls.py
```

Estos archivos son los siguientes:

- __init__.py: Un archivo requerido para que Python trate a este directorio como un paquete (i.e. un grupo de módulos).
- manage.py: Una utilidad de línea de comandos que te deja interactuar con este proyecto de Django de varias formas.
- settings.py: Opciones/configuraciones para este proyecto de Django.
- urls.py: La declaración de las URL para este proyecto de Django; una "tabla de contenidos" de tu sitio hecho con Django.

¿Dónde debería estar este directorio? Si el lector viene de PHP, probablemente pondrá el código debajo de la carpeta raíz del servidor web (en lugares como /var/www). Con Django, no se tiene que hacer esto. No es una buena idea poner cualquier código Python en la carpeta raíz del servidor web, porque al hacerlo se arriesga a que la gente sea capaz de ver el código en la web. Esto no es bueno para la seguridad.

Para crear nuestra aplicación, se asegurará de estar en el directorio de nombre_de_tu_-proyecto e introducir el siguiente comando:

```
$ python manage.py startapp mi_aplicacion
```

Esto creará un directorio mi_aplicacion, que contienen la siguiente estructura:

```
1       polls/
2           __init__.py
3           models.py
4           tests.py
5           views.py
```

- models.py: En este archivo es donde crearemos nuestra base de datos.
- view.py: Aquí es donde se localizan nuestros controladores.

Activando los modelos

Se debe editar nuevamente el archivo settings.py y cambiar el parámetro INSTALLED_-APPS para incluir la cadena 'nombre_de_tu_proyecto.mi_aplicacion'

```
1       INSTALLED_APPS = (
2           'django.contrib.auth',
3           'django.contrib.contenttypes',
4           'django.contrib.sessions',
5           'django.contrib.sites',
6           'nombre_de_tu_proyecto.mi_aplicacion'
7       )
```

Ahora se ejecutará otro comando para crear las tablas SQL.

```
$ python manage.py sql mi_aplicacion
```

Es posible ejecutar también los siguientes comandos:

- python manage.py validate polls – Busca errores en la construcción de tus modelos.
- python manage.py sqlcustom polls – Muestra las sentencias SQL definidas manualmente para la aplicación (por ejemplo, modificaciones de tablas y restricciones).
- python manage.py sqlclear polls – Muestra las sentencias DROP TABLE necesarias para esta aplicación, de acuerdo a las tablas que ya existen en la base de datos, si corresponde.
- python manage.py sqlindexes polls – Muestra la salida de las sentencias CREATE INDEX para esta aplicación.
- python manage.py sqlall polls – Una combinación de todas las sentencias SQL de los comandos 'sql', 'sqlcustom', y 'sqlindexes'.

```
$ python manage.py syncdb
```

El comando syncdb ejecuta el sql de 'sqlall' en la base de datos para todas las aplicaciones en INSTALLED_APPS que no existían ya en la base de datos. Esto crea todas las tablas, los datos iniciales y los índices para cada aplicación que se haya agregado al proyecto desde la última vez que se ejecutó syncdb. syncdb puede ser ejecutado tan frecuentemente como se desee, y sólo creará las tablas que no existan.

Desarrollo base de datos

Ahora, se debe editar settings.py. Es un módulo Python normal con variables a nivel de módulo que representan configuraciones de Django. Modifica los siguientes valores en el elemento 'default' de DATABASES para que coincida con la configuración de tu base de datos.

- ENGINE – Ya sea 'django.db.backends.postgresql_psycopg2', 'django.db.backends.mysql' o 'django.db.backends.sqlite3'. Otros backends también están disponibles.
- NAME – El nombre de tu base de datos. Si estás usando SQLite la base de datos será un archivo en tu ordenador; en tal caso, NAME debería ser la ruta absoluta, incluyendo nombre de archivo, de dicho archivo. Si el archivo no existe se creará automáticamente cuando sincronices la base de datos por primera vez (ver abajo). Cuando especifiques la ruta, siempre utiliza barras hacia la derecha / incluso en Windows (por ejemplo: C:/homes/user/mysite/sqlite3.db).
- USER – El usuario de la base de datos (no usado en SQLite).
- PASSWORD – La contraseña de la base de datos (no usado en SQLite).
- HOST – La máquina donde está ubicada la base de datos. Déjalo como una cadena vacía si la base de datos está en la misma máquina física (no usado en SQLite).

Si las bases de datos son algo nuevo para el lector, se recomienda usar simplemente SQLite (poniendo en ENGINE 'django.db.backends.sqlite3'). SQLite viene incluido como parte de Python 2.5 y superior, así que no tendrá que instalar nada.

Vistas

Una función de vista, o vista en pocas palabras, es una simple función de Python que toma como argumento una petición Web y retorna una respuesta Web. Esta respuesta puede ser el contenido HTML de la página web, una redirección, o un error 404, o un documento XML, o una imagen... o en realidad, cualquier cosa. La vista en sí contiene toda la lógica necesaria para retornar esa respuesta. El código puede encontrarse donde quieras, mientras que se encuentre dentro de tu Python path. No hay otro requerimiento – no hay "magia", por así decirlo. Por poner el código en algún lugar, creemos un archivo llamado views.py en el directorio mysite, el cual creamos en el capítulo anterior.

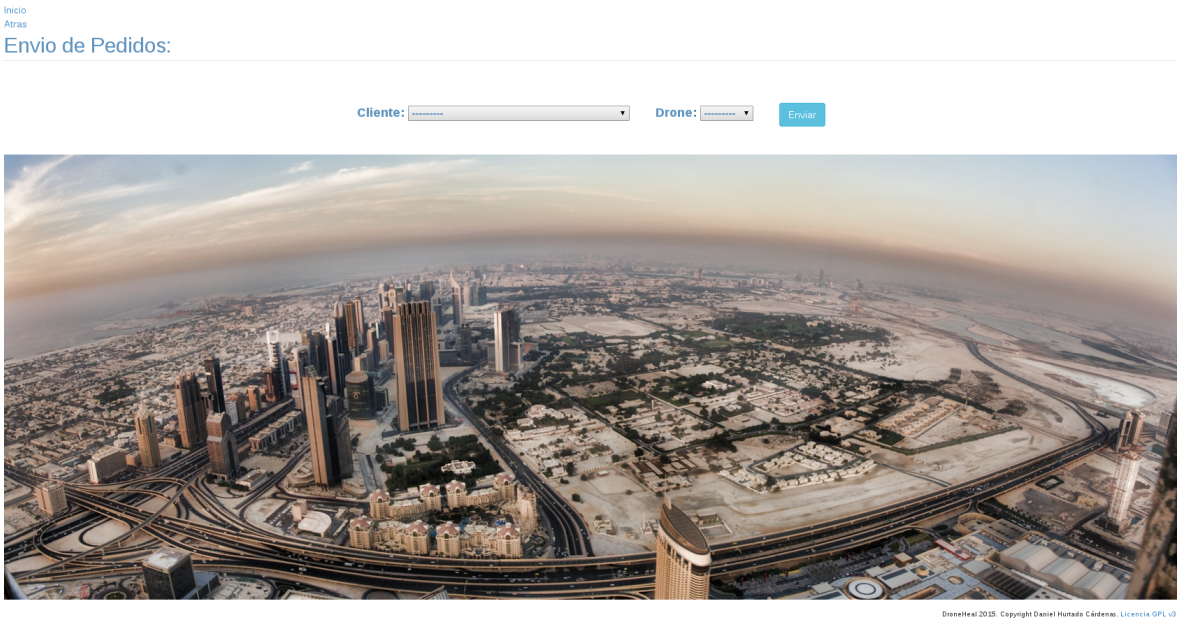

Figura 5.2: Captura de pantalla de la aplicación.

URLconfs

Ahora es el momento de resaltar una parte clave de filosofía detrás de las URLconf y detrás de Django en general: el principio de acoplamiento débil (loose coupling). Para explicarlo simplemente, el acoplamiento débil es una manera de diseñar software aprovechando el valor de la importancia de que se puedan cambiar las piezas. Si dos piezas de código están débilmente acopladas (loosely coupled) los cambios realizados sobre una de dichas piezas va a tener poco o ningún efecto sobre la otra.

Las URLconfs de Django son un claro ejemplo de este principio en la práctica. En una aplicación Web de Django, la definición de la URL y la función de vista que se llamará están débilmente acopladas; de esta manera, la decisión de cuál debe ser la URL para una función, y la implementación de la función misma, residen en dos lugares separados. Esto permite el desarrollo de una pieza sin afectar a la otra.

Plantillas

Una plantilla de Django es una cadena de texto que pretende separar la presentación de un documento de sus datos. Una plantilla define rellenos y diversos bits de lógica básica (esto es, etiquetas de plantillas) que regulan como debe ser mostrado el documento. Normalmente, las plantillas son usadas para producir HTML, pero las plantillas de Django son igualmente capaces de generar cualquier formato basado en texto.

Bootstrap

Bootstrap, es un framework originalmente creado por Twitter, que permite crear interfaces web con CSS y JavaScript, cuya particularidad es la de adaptar la interfaz del sitio web al tamaño del dispositivo en que se visualice. Es decir, el sitio web se adapta automáticamente al tamaño de una PC, una Tablet u otro dispositivo. Esta técnica de diseño y desarrollo se conoce como "responsive design" o diseño adaptativo.

El beneficio de usar responsive design en un sitio web, es principalmente que el sitio web se adapta automáticamente al dispositivo desde donde se acceda. Lo que se usa con más frecuencia, y que a mi opinión personal me gusta más, es el uso de media queries, que es un módulo de CSS3 que permite la representación de contenido para adaptarse a condiciones como la resolución de la pantalla y si trabajas las dimensiones de tu contenido en porcentajes, puedes tener una web muy fluida capaz de adaptarse a casi cualquier tamaño de forma automática.

5.2.4 Iteración 3

En esta iteración el objetivo es desarrollar los módulos para el cálculo de las rutas de envío. Para realizar esto se debe hacer en la estación terrena, la cual recibirá un mensaje del cliente web con los datos del UAV que realizará el envío y la localización del objetivo.

Las rutas se pueden realizar de dos formas fundamentalmente:

- Guiado - Este modo tiene algunas peculiaridades, se le envian ordenes al UAV, y él responde realizando las acciones. En todo momento se tiene que disponer de comunicación con el UAV para ir dando ordenes.
- Automático - Se introduce un fichero con las ordenes, y automáticamente realiza la ruta.

Como hemos dicho anteriormente se pueden realizar las rutas de dos formas, en nuestro caso hemos elegido la opción automática, aunque hemos implementado las dos formas. Ya que la opción de guiado nos permite realizar y analizar el trayecto en tiempo real y actuar en consecuencia.

Para cargar el archivo a nuestro UAV debemos crearlo con un formato concreto según se muestra en el anexo F.

5.3 Desarrollo Hardware

No se ha desarrollado ningún componente desde cero, ya que en el mercado existen gran variedad de alternativas. Lo que se ha hecho ha sido integrar cada componente y su configuración. Los componentes hardware que intervienen en nuestro sistema son los siguientes:

- Raspberry Pi

- Controlador de Vuelo Crius AIO Pro v2.0
- Receptor GPS u-Blox NEO-6M
- Módulo potencia
- Radio telemetría 433 MHz
- Receptor/Transmisor WiFi
- Controlador de velocidad 30A SimonK

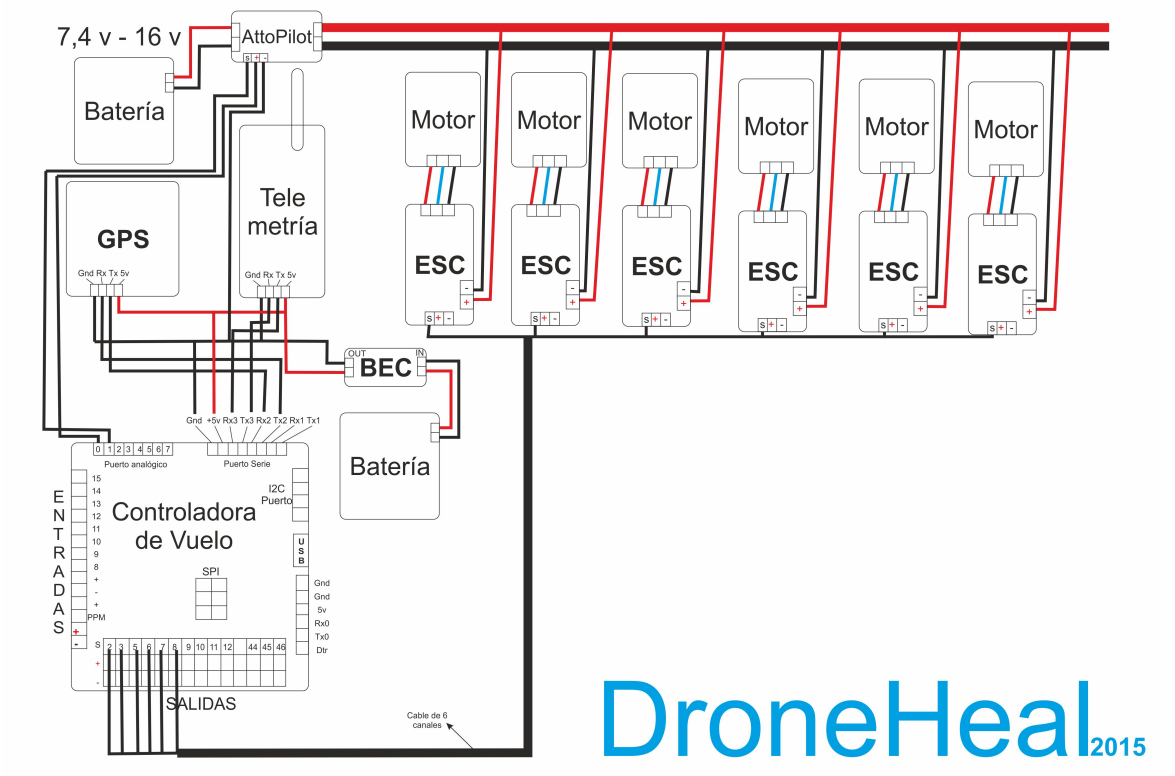

Figura 5.3: Diagrama del sistema hardware.

5.3.1 Raspberry Pi

En la Raspberry Pi se ha instalado el sistema operativo Raspbian, para ello se ha formateado una memoria micro SD de clase 10 para garantizar la velocidad de transferencia de 10 MB/s. Una vez formateada se grabo el SO Raspbian. Se ejecuta por primera vez y se realizan las configuraciones iniciales como poner la hora, la franja horaria, etc. Cuando se ha realizado las configuración se reinicia y los cambios surtirán efecto, momento en que se puede añadir nuevos componentes. Se conecta el adaptador WiFi, no se ha mostrado la marca del que se ha utilizado, ya que en el mercado existen multitud de ellos que funcionan correctamente para nuestro cometido, en las páginas de documentación de la Raspberry Pi existe una lista con algunos de los adaptadores que se han probado. En el anexo A: «Cómo hacer un

punto de acceso SW y HW libre» se muestra como configurar la Raspberry Pi para convertirla en un punto de acceso, el cual permite conectar con cualquier dispositivo mediante WiFi. También se desplegara la aplicación con todos sus servidores, como el de base de datos, web, GPS.

Paquetes necesarios:

Gunicorn	Wise	MAVProxy
Python	python-jsonpickle	python-opencv
python-gunicorn	python-brimstone	python-wxgtk2.8
python-gevent	python-commodity	python-numpy
	phantomjs	python-serial
	nodejs-legacy	python-pyparsing
	npm	python-droneapi
	karma	

Cuadro 5.2: Paquetes necesarios

5.3.2 Controlador de vuelo

Para configurar la controladora elegida Crius AIO Pro v2.0, se tiene primero que descargar una version de Arduino para poder cagar el firmware a la tarjeta. Se recomienda la versión 1.03, que es la que mejor se adapta y reconoce la tarjeta. También tenemos que descargar el firmware del proyecto MegaPirateNG, que es una versión del proyecto Arducopter, pero adaptado a la tarjeta Crius.

Una vez hecho esto ya se puede comenzar.

- Arduino es usado para compilar, configurar y cargar el código a la tarjeta Crius.
- Se crea un directorio en el PC (a ser posible en el escritorio).
- Dentro se crean dos directorios más, uno llamado Arduino y otro MegaPirate.
- Se descarga el firmware de MegaPirate y se guarda en su directorio.
- Se descarga el software de Arduino y se guarda también en su directorio.
- Ahora se abre el directorio de MegaPirate y se copian las librerías al directorio de Arduino.
- Ahora en el directorio de Arduino, se debe ejecutar.
- Se debe comprobar que herramientas (Tools) estan configuradas para la tarjeta elegida, se debe poner Mega 2560 or Mega ADK, es el tipo de microcontrolador. >Tools>Board>Arduino Mega 2560 or Mega ADK.
- Ahora se debe seleccionar el puerto serie del Crius, esto normalmente se auto detecta.
- Se configura Arduino para comenzar con MegaPirate.

- se debe usar >File>Open en Arduino y navegar hasta el directorio de MegaPirate que se creo, se abrira el subdirectorio llamado ArduCopter. Dentro de este directorio se encuentra un archivo llamado ArduCopter.pde. Se debe editar este archivo.
- Se abrirá una ventana con todos los archivos del proyecto abiertos. Se debe cerrar la ventana anterior, y solo se deben hacer cambios en un archivo llamado APM_Config.h.

Estas son las partes de código que se deben cambiar:

Se debe configurar el tipo de tarjeta de vuelo.

```
#define MPNG_BOARD_TYPE   RCTIMER_CRIUS_V2
        RCTIMER_CRIUS_V2                -- RCTimer CRIUS V2 board
        CRIUS_V1                        -- RCTimer CRIUS V1(1.1)
        HK_RED_MULTIWII_PRO             -- HobbyKing MultiWii
        BLACK_VORTEX
        MULTIWII_PRO_EZ3_BLACK          -- MultiWii PRO Ez3.0 Blacked MAG
        PARIS_V5_OSD                            -- Paris v5 Mega iOSD
```

También se tiene que configurar el tipo de barómetro.

```
#define CONFIG_BARO AP_BARO_MS5611_I2C
```

Y el tipo de GPS. Esto depende del tipo de gps que estés usando.

Para GPS de extensión de tarjeta (Si se esta usando Ublox o extend board)

```
#define GPS_PROTOCOL GPS_PROTOCOL_UBLOX
```

Para todos los demás GPS.

```
#define GPS_PROTOCOL GPS_PROTOCOL_AUTO
```

Lo siguiente que se debe configurar es el tipo de estructura del UAV.

```
#define FRAME_CONFIG HEXA_FRAME (elige tu tipo de estructura de la lista)
        QUAD_FRAME
        TRI_FRAME
        HEXA_FRAME
        Y6_FRAME
        OCTA_FRAME
        OCTA_QUAD_FRAME
        HELI_FRAME
```

Finalmente debes configurar la orientación de la estructura.

```
#define FRAME_ORIENTATION X_FRAME (elige tu tipo de estructura de la lista)
        PLUS_FRAME
```

```
3                   X_FRAME
4                   V_FRAME
```

Con esto se ha configurado las opciones básicas para poder realizar nuestro cometido. Se podría seguir configurando todos los componentes que se quieran integrar, como sonar, medidores de velocidad, etc.

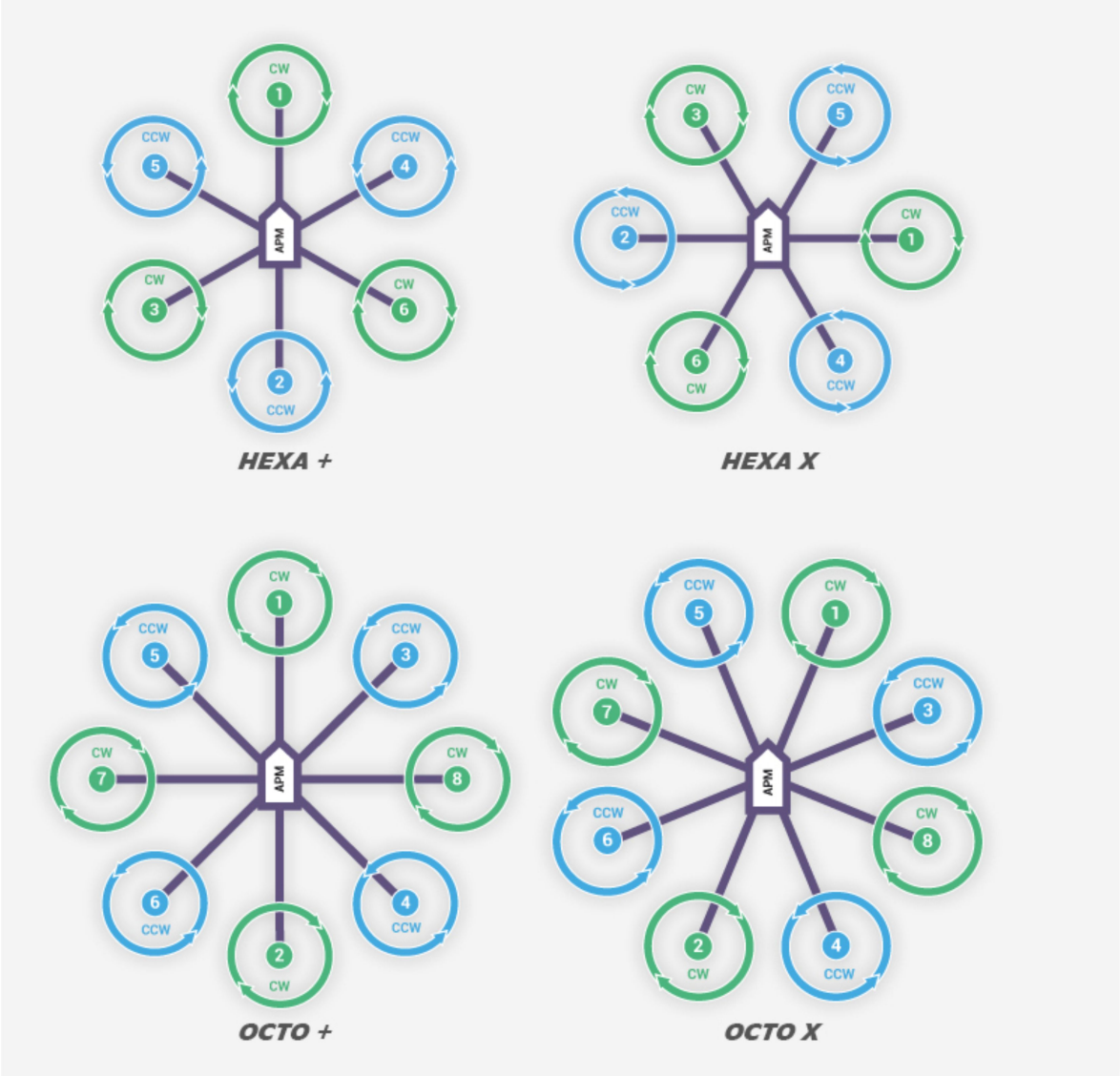

Figura 5.4: Configuración de hélices y motores.

Solo nos queda compilar y cargar el código dentro de la tarjeta. Si se encontrase algún error, se mostraría antes de cargarlo dentro de la tarjeta.

Una vez que se ha compilado y cargado el código MegaPirateNG correctamente en la tarjeta Crius, se puede ahora instalar la estación terrena, y comprobar que todo funciona correctamente.

5.3.3 Receptor GPS

Para conectar el GPS a la Crius se puede hacer en cualquiera de sus puertos serie S1-S3, aunque normalmente se usa el puerto TX2 y RX2. Está configurado por defecto a 38400 baud.(Se puede cambiar editando el archivo APM_Config.h).

Se debe asegurar la velocidad del GPS para configurarlo correctamente en el firmware.

El TX se debe conectar en el RX y el RX en el TX. El GPS necesita ser configurado en el archivo APM_Config.h, en Arduino, las siguientes líneas muestran el código que se debe modificar para seleccionar el GPS correcto.

```
1       #define GPS_PROTOCOL GPS_PROTOCOL_UBLOX
2               GPS_PROTOCOL_NONE without GPS
3               GPS_PROTOCOL_NMEA
4               GPS_PROTOCOL_SIRF
5               GPS_PROTOCOL_UBLOX <<< Selecciona este para UBLOX LEA-6 (CRIUS GPS tarjeta y otras)
6               GPS_PROTOCOL_MTK16
7               GPS_PROTOCOL_BLACKVORTEX
8               GPS_PROTOCOL_AUTO auto select GPS
```

También se puede configurar el GPS con la aplicación del fabricante, y modificar varias características.

5.3.4 Radio Telemetría

Gracias a este módulo podemos comunicar la estación terrena con el UAV. Actualmente existen 2 tipos de frecuencias disponibles 900 MHz y 433 MHz se debe elegir la que este permitida en el país de uso. 433 MHz es lo recomendado en Europa.

Se puede conectar al puerto FTDI/S0 o al S3 del puerto S1-S3. Se recomienda usar el RX3 y TX3 del puerto S1-S3, esto dejará el puerto FTDI libre para otros usos como el Bluetooth. Usando el puerto S3 no requiere ningún cambio de baud rate en el archivo APM_-Config.h.

La radio consiste en una unidad remota y una local, la unidad local es un USB y la remota es un módulo pequeño. Antes de poder usar el modulo, se necesitará configurarlo, para hacerlo se necesita conectar el USB a tu PC y el módulo remoto conectarlo al crius.

Es muy importante siempre conectar las antenas antes de realizar cualquier función.

El primer paso a seguir es configurar el baud rate de ambos dispositivos, si se esta usando S3 en el Crius como entrada, entonces la radio debe ser configurada a 57600 baudios, si se esta usando el puerto FTDI/S0 entonces la radio debe ser configurada a 115200 baudios.

Si se cambia la velocidad de los módulos de telemetría se debe garantizar hacerlo en los dos módulo al mismo tiempo. Ya que sino perderá la conexión entre ellos.

Figura 5.5: Radio Telemetría 433 MHz.

5.3.5 Módulo potencia

También llamado Atto Pilot, el Crius Aio Pro v2.0 es capaz de medir el voltaje y la intensidad usando el mini modulo Attopilot. MegaPirateNG está configurado correctamente para ser usado en los pines A0 y A1.

Esta información puede ser pasada mediante Mavlink y puede ser vista en nuestra estación terrena y mediante un OSD si lo configuramos.

Conexiones:

- V-Pin del Attopilot al Pin A0 de la tarjeta Crius.
- I-Pin del Attopilot al Pin A1 de la tarjeta Crius.
- Tierra (GND) del Attopilot al Ground de la tarjeta Crius.
- IN+ de la batería.
- Out+ al ESC, BEC, etc.
- Heavy Ground a la batería.

5.3.6 Controlador de velocidad

Son elementos pasivos que no necesitan configuración, aunque existe la posibilidad de cambiar su firmware mediante una interfaz SPI.

5.4 Desarrollo de la Estructura del UAV

Para el desarrollo de la estructura, se han tenido en cuenta multitud de características. Como el uso de materiales ligeros, dureza de los materiales, diseño y forma. Se ha utilizado SolidWorks como aplicación completa para el diseño 3D. Se adjuntan imágenes de los prototipos, los materiales utilizados para la estructura propiamente dicha ha sido fibra de vidrio para las PCB's con toda la circuitería, aluminio para los brazos, y fibra de carbono para las sujeciones de los motores a los brazos, ya que es el punto donde más fuerza se ejerce, y es la fibra de carbono uno de los materiales más resistentes. Para la carcasa exterior se ha utilizado poliespan o polietileno expandido.

Como se puede apreciar en la figura 5.6 la carcasa externa tiene apariencia de disco, su radio es de 45 cm, en los agujeros circulares grandes es donde se localizan las hélices con los motores. Estos agujeros van cubiertos con una malla metálica protectora, de un tamaño que impida que se introduzca un dedo.

En la figura 5.7 se pueden ver los distintos compartimentos, en la parte superior se encuentra el compartimento de las baterías con una tapa que permite fácilmente intercambiarlas y cargarlas, también como medida de seguridad por si se incendian evitar que afecte a la electrónica. En el medio se encuentra el compartimento donde se aloja toda la electrónica, protegida por las placas PCB, que están fabricadas en fibra de vidrio tanto en las parte superior como en la inferior. En la parte inferior se localiza el anclaje para la paquetería, que funciona mediante un pequeño servo con una pestaña, que se acciona y deja caer el paquete. También se pueden apreciar los huecos para cada uno de los brazos.

Para diseñar los brazos se ha tenido en cuenta el grosor del propio UAV, y se ha intentado que sea lo menor posible. El material adecuado para su fabricación es el aluminio, por su ligereza y resistencia. En las figuras 5.8, 5.9, 5.10 y 5.11 se puede ver cómo se han suavizado los contornos y se ha eliminado todo el material sobrante para aligerar la pieza, también se ha añadido la marca como se aprecia en la figura 5.8 y 5.10.

Se han realizado pruebas con distintos tipos de materiales, para comprobar su resistencia ante grandes fuerzas como son las fuerzas de torsión ejercidas por los motores sobre sus brazos. Hemos podido comprobar como materiales plasticos de alta resistencia y dureza no son adecuados, ya que el motor en momentos muy puntuales ejerce gran fuerza sobre los brazos, debido a su estabilización y control. Esto hace que el motor vibre y pueda entrar en caida.

También se ha tenido en cuenta que sea resistente al agua, se han realizado pruebas con distintos productos que impermeabilizan los circuitos electronicos ya que son la parte sensible al agua. Aunque las partes que podrian estar en contacto con agua son solo los motores y sus conexiones, ya que el resto se encuentra dentro de la carcasa y es muy dificil que entre agua dentro del compartimento debido a su sellado. Todos los componentes electronicos

Figura 5.6: Carcasa exterior.

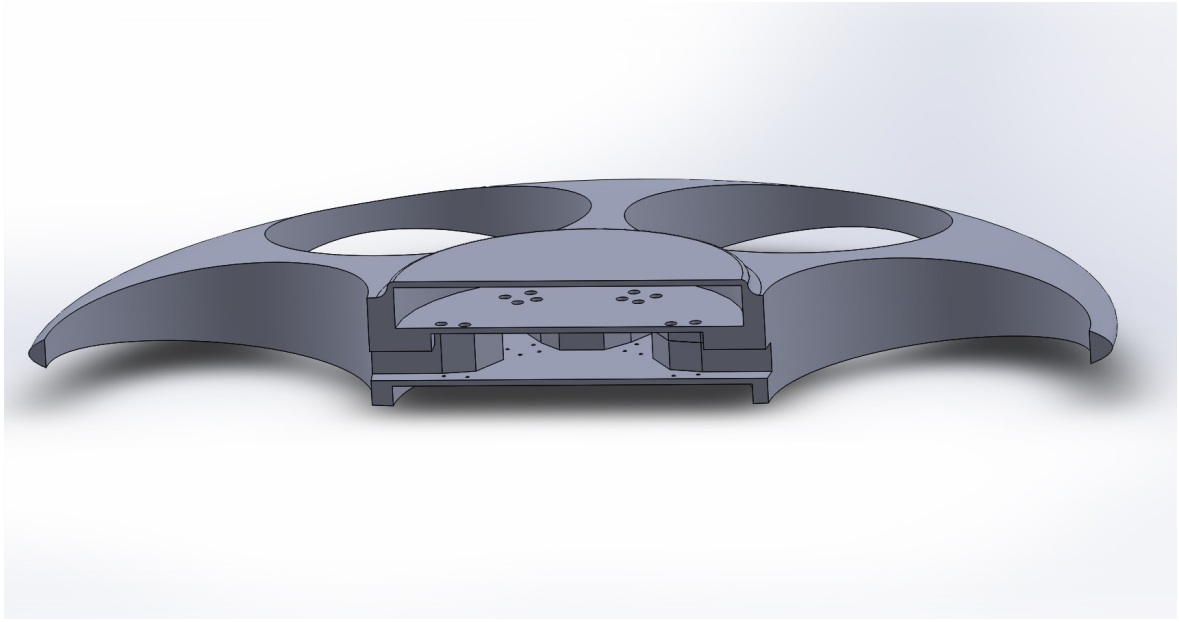

Figura 5.7: Carcasa exterior II.

como hemos dicho anteriormente estan protegidos contra el agua.

5.5 Pruebas

El problema fundamental respecto a la prueba de software es que no se puede probar completamente un sistema, por lo que, en el momento de realizar las pruebas, se deben tomar decisiones respecto a qué partes del sistema probar y el grado de profundidad de los casos de prueba.

Para probar completamente un sistema se deben ejercitar todos los caminos posibles del

programa a probar. Myers mostró en 1979 un programa que contenía un bucle y unas pocas instrucciones condicionales, y que tenía 100 trillones de caminos. La prueba exhaustiva de este programa podría requerir un billón de años.

5.5.1 Caso de Prueba

Un caso de prueba (test case) es un conjunto de valores de entrada, precondiciones de ejecución, resultados esperados y poscondiciones de ejecución, desarrollados con un objetivo particular o condición de prueba, tal como ejercitar un camino de un programa particular o para verificar que se cumple un requisito especifico.[IEE90]

5.5.2 Objetivos de la pruebas

La prueba debe ser vista como el proceso destructivo de encontrar errores (cuya presencia se asume) en un programa. Si el propósito del tester es verificar que el programa funciona correctamente, entonces el tester está fallando al conseguir su propósito cuando encuentra defectos en el programa.[G04]

Se han realizado 5 tipos de pruebas, Pruebas de Caja Negra, de Caja Blanca, Unitarias, de integración y de sistemas.

5.5.3 Pruebas de Caja Negra

Este tipo de prueba se han utilizado sobre los componentes del sistema externos, aunque en todos ellos teniamos acceso al código, ya que son proyectos de código abierto. La forma inicial de comenzar es haciendo este tratamiento de abtraernos de ese código.

En este tipo de pruebas, el elemento que se va a probar se entiende como una caja negra de la que sólo se conocen sus entradas y sus salidas. Así, al elemento bajo prueba se lo somete a una serie de datos de entrada, se observan las salidas que produce y se determina si éstas son conformes a las entradas introducidas.[G04]

5.5.4 Pruebas de Caja Blanca

El código probado con este tipo de pruebas, ha sido todo el código implementado a medida para el sistema. Desde el cliente web hasta los servidores GPS. Las pruebas de caja blanca realizan, de alguna manera, un seguimiento del zonas del sistema que van ejecutando los casos de prueba. Si el artefacto que se está probando es un programa, se determinan de manera concreta las instrucciones, bloques, etc. que han sido ejecutados por los casos de prueba. En este caso, este tipo de pruebas permiten conocer cuánto código se ha recorrido.

5.5.5 Pruebas Unitarias

En general En general, en orientación a objetos se asume que la unidad de prueba es la clase, por lo que se comprueba que el estado en el que queda la instancia de la clase que se está probando es correcto para los datos que se le pasan como entrada. En el caso de

las pruebas unitarias de caja negra para un sistema orientado a objetos, se entiende la clase como, en efecto, una caja cuyo interior no interesa: lo único que importa desde este punto de vista es el conjunto de entradas suministradas y las salidas obtenidas.

Las pruebas unitarias centran su aplicación en lo que se denomina la "unidad de prueba" que, dependiendo del contexto, puede ser una clase, un método o un subsistema. El estándar ANSI/IEEE 1008/1987 [IEE87], define la unidad de prueba de la siguiente forma:

Un conjunto de uno o más módulos de un programa, junto a los datos de control asociados (por ejemplo, tablas), procedimientos de uso y procedimientos de operación que satisfagan las siguientes condiciones:

- Todos los módulos pertenecen a un único programa.
- Al menos uno de los módulos nuevos o cambiados del conjunto no ha pasado las pruebas unitarias (puesto que una unidad de prueba puede contener uno o más módulos previamente probados).
- El conjunto de módulos junto con sus datos y procedimientos asociados son el único objetivo del proceso de pruebas.

5.5.6 Pruebas de Integración

Las pruebas de integración se emplean para comprobar que las unidades de prueba, que han superado sus pruebas de unidad, funcionan correctamente cuando se integran, de manera que lo que se tiende a ir probando es la arquitectura software. Durante la integración, las técnicas que más se utilizan son las de caja negra, aunque se pueden llevar a cabo algunas pruebas de caja blanca para asegurar que se cubren los principales flujos de comunicación entre las unidades [RS93].

5.5.7 Pruebas de Sistema

Las pruebas de sistema tienen por objetivo comprobar que el sistema, que ha superado las pruebas de integración, se comporta correctamente con su entorno (otras máquinas, otro hardware, redes, fuentes reales de información, etc.). Las pruebas funcionales forman parte de las pruebas del sistema, al igual que las siguientes [JA06]:

- Pruebas de recuperación. Consisten en forzar el fallo del software y comprobar que la recuperación se lleva a cabo de manera correcta, devolviendo al sistema a un estado coherente.
- Pruebas de seguridad. Intentan verificar que los mecanismos de protección incorporados al sistema lo protegerán, de hecho, de penetraciones inadecuadas.
- Pruebas de resistencia. Estas pruebas están diseñadas para que el sistema requiera recursos en cantidad, frecuencia o volumen anormales. La idea es intentar que el sistema se venga abajo por la excesiva tensión a la que se lo somete.

- Pruebas de rendimiento. En sistemas de tiempo real o sistemas empotrados, es inaceptable que el software proporcione las funciones requeridas fuera de las condiciones de rendimiento exigidas.

5.6 Presupuesto

Por un lado se ha aplicado el modelo constructivo de costes COCOMO para la parte software del proyecto. COCOMO es una jerarquía de modelos de estimación de costes software que incluye submodelos básico, intermedio y detallado. Para el apartado hardware se ha realizado un presupuesto común con cada unos de los componentes y su precio.

SLOC	Directorio	SLOC por lenguaje
1394	mapa	python=1394
71	dronheal	python=71
21	top_dir	sh=15,python=6
0	static	
0	templates	

Cuadro 5.3: Lineas de código

- Total por lenguaje agrupado (primer lenguaje dominante):
- python: 1471 (98.99 %)
- sh: 15 (1.01 %)

- Total Physical Source Lines of Code (SLOC) = 1,486
- Development Effort Estimate, Person-Years (Person-Months) = 0.30 (3.64)
- (Basic COCOMO model, Person-Months = 2.4 * (KSLOC**1.05))
- Schedule Estimate, Years (Months) = 0.34 (4.08)
- (Basic COCOMO model, Months = 2.5 * (person-months**0.38))
- Estimated Average Number of Developers (Effort/Schedule) = 0.89
- Total Estimated Cost to Develop = $ 40,951
- (average salary = $ 56,286/year, overhead = 2.40).

Componentes Hardware del sistema:

UAV

- Controlador de Vuelo 150 €
- Controlador de velocidad 6 x 15 €

- Estructura 50 €
- Helices 12 x 2 €
- Motores sin escobillas o brushless 6 x 25 €
- Baterías 2 x 35 €
- BEC 10 €
- GPS 40 €
- Telemetría 50 €

Subtotal 634 €

Estación Terrena

- Raspberry Pi 40 €
- SD 15 €
- Batería Raspberry Pi 15 €
- Adaptador Wifi 15 €
- Antenas 40 €

Subtotal 125 €

Total 759 €

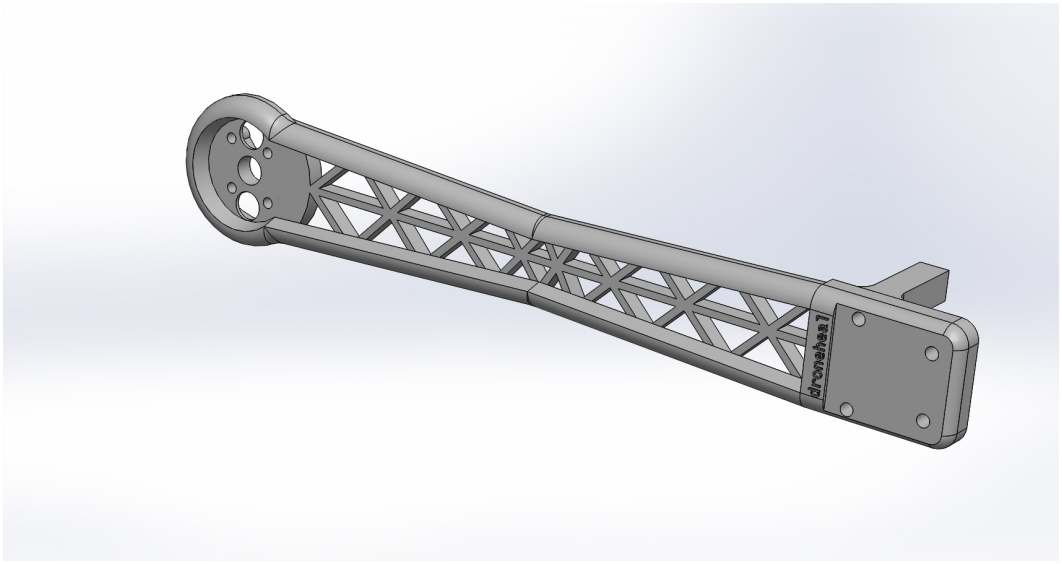

Figura 5.8: Brazo.

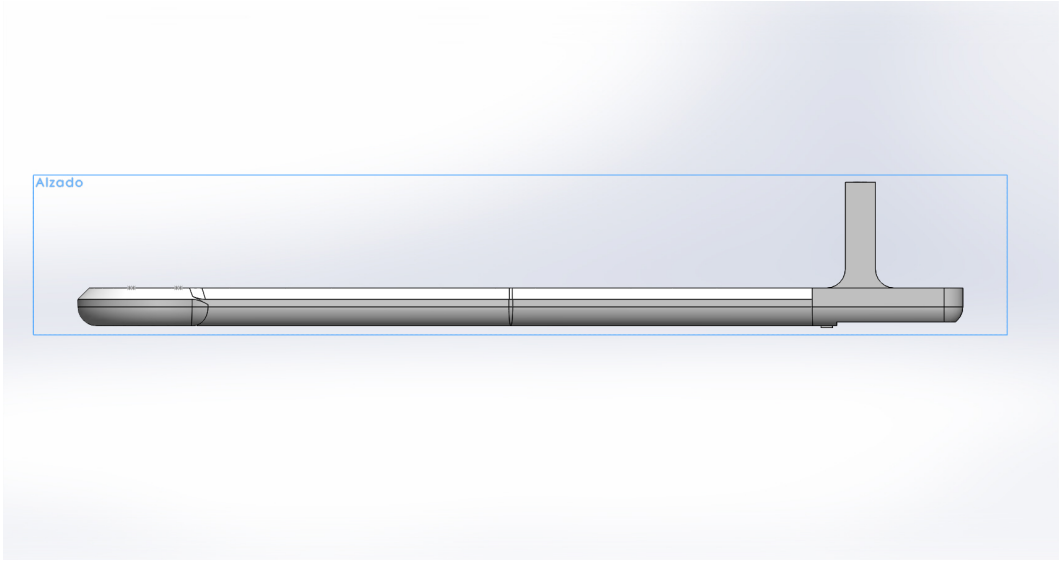

Figura 5.9: Brazo, vista lateral.

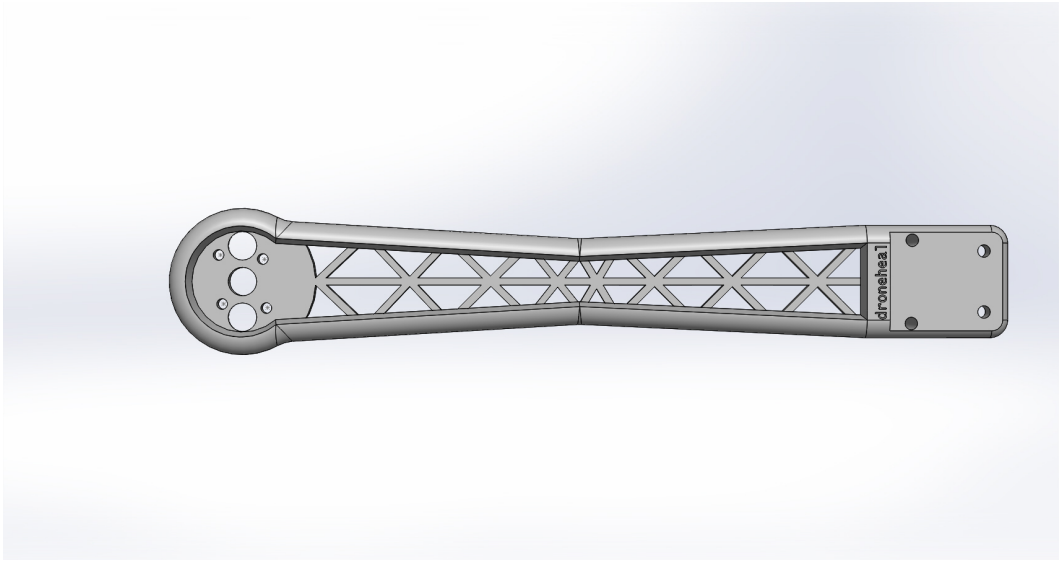

Figura 5.10: Brazo, vista alzado.

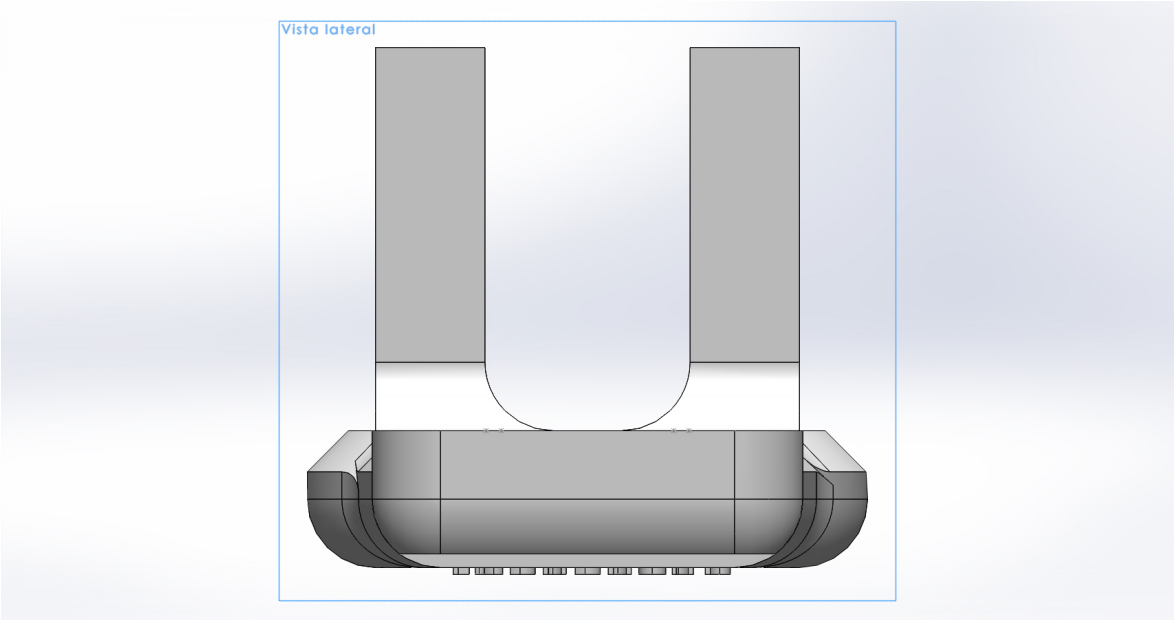

Figura 5.11: Brazo, vista sección interna.

Capítulo 6

Conclusiones y Resultados

LLEGADOS a este punto, se han podido comprobar los objetivos alcanzados y resultados obtenidos. El objetivo principal ha sido realizar un sistema de transporte aereo de mercancias de corta distancia basado en UAV, el cual nos permita su transporte, entrega y control de la aeronave. Este objetivo ha sido satisfecho correctamente, para poder alcanzar este, se han tenido que alcanzar otros objetivos secundarios como la fabricación de UAV, realizado correctamente gracias al asesoramiento del doctor ingeniero Antonio Gonzalez Rodriguez, profesor de la Escuela de Ingenieros Industriales de Ciudad Real de la Universidad de Castilla La Mancha. Tambien se ha alcanzado la realización de la estación terrena, la cual nos permite por un lado controlar el UAV y por otro dispone de todos los servidores necesarios para lanzar la aplicación web.

6.1 UAV

Se puede afirmar al termino del proyecto de ser capaz de fabricar cualquier tipo de UAV, dependiendo de los requerimientos y el presupuesto de la aeronave. Un multicopter es una aeronave considerada como una piedra volante, ya que su fuerte no es la aerodinámica. El uso de otro tipo de aeronaves, con forma de aviones necesitan un tratamiento especial en el estudio de su aerodinámica, ya que en estos aparatos es fundamental la resistencia al viento. Para esto se debe construir un tunel del viento y observar las turbulencias producidas por los contornos y formas de las aeronaves.

6.2 Aplicación

Para desarrollo de aplicaciones web existen actualmente una gran variedad de herramientas y muy diferentes entre ellas. Se debe realizar un estudio concienzudo de las necesidades, restricciones y requesitos de la aplicación web, para poder elegir la mejor opción. Muchas veces la mejor opción no es la que más rapido o mejor rendimiento tiene, sino la que más y mejor documnetación existe.

6.3 Estación Terrena

Se ha podido observar como una placa económica como es la Raspberry Pi puede cumplir con las exigencias de multitud de proyectos. Y lo más importante es que un proyecto debe ser consecuente con su entorno, exigencias y requisitos necesarios. A de ser acorde al cliente y su entorno.

6.4 Repositorio

Se ha utilizado un sistema de control de versiones como es Mercurial. Toda la documentación e implementación del proyecto se encuentra en la siguiente dirección `https://bitbucket.org/danicianuro/tfg.daniel.hurtado`.

Capítulo 7

Mejoras

LAS posibles mejoras y trabajos futuros las podemos dividir en varios grupos, debido a su complejidad y su alto grado de diferenciación debido a su arquitectura. Por un lado tenemos el UAV, la estación terrena y la aplicación.

7.1 UAV

Las posibles mejoras son inumerables. Por un lado a lo que se refiere a su estructura, se pueden realizar otros diseños buscan mayores grados de aerodinámica, fabricarlo con otro tipo de materiales como son la fibra de carbono, u otros materiales muy ligeros y absorventes de glopes. En lo que se refiere a la potencia y respuesta de los motores se pueden introducir motores de más potencia, aunque esto hace que aumentemos considerablemente el peso y puede ocasionar más riesgos y problemas burocraticos. Para terminar podemos mejorar la electrónica, incluyendo muchos más sensores, e incluso un ordenador de abordo para resolver las rutas dinamicamente ante posibles incidencias.

En lo que se refiere a sensores, existen multiples opciones y alternativas. Como por ejemplo un sonsar, para evitar obstaculos. Medidores de velocidad del viento, para ser más precisos en la mediciones, una brújula externa, para evitar interferencias debido a su gran precisión y sensibilidad.

7.2 Aplicación

La aplicación web se puede diseñar para dispositivos moviles, permitiendo una mejor adaptación, rendimiento y visualización de la aplicación en el dispositivo. Se pueden añadir opciones como poder hacer nuevos objetivos o puntos de destino, pulsando con el dedo sobre el mapa. Para facilitar al usuario su tarea.

7.3 Estación Terrena

La estación terrena se puede mejorar permitiendo controlar más de un UAV, ya que por el momento no se ha implementado esta opción. También se puede mejorar con una antena de seguimiento, que permite alcanzar mucha sensibilidad en la recepción de la señal de los UAV. Está antena necesita saber en todo momento la posición del UAV para orientarse y apuntar

en su dirección. El sistema de comunicación entre el UAV y la estación terrena se puede cambiar por un sistema GSM que permita conexión continua en toda el centro urbano, pero problemático para el uso en zonas rurales donde la covertura es deficiente.

ANEXOS

Anexo A

Cómo hacer un punto de acceso SW y HW libre

A.1 Necesitamos

- Raspberry Pi.
- Adaptador wifi.
- Cable ethernet.
- Cable usb (para alimentar la raspberry pi).

Para el hardware he utilizado la Rasberry Pi y un adaptador wifi con el driver rtl8192, podéis utilizar el que queráis, lo que pasa que no todos permiten modo access point. Aquí os dejo un enlace con los adaptares reconocidos por la raspberry pi.

`https://wiki.debian.org/WiFi#USB_Devices`

En la raspberry pi he utilizado Raspbian os dejo aquí el enlace para descargar e instalar el sistema en una SD.

`http://www.raspberrypi.org/downloads/`

En el primer arranque de la raspberry pi se ejecutara la utilidad de configuración de raspberry pi, configurando lo siguiente:

- Expandimos el sistema de archivos (Esto permite usar todo el espacio de la tarjeta SD).
- Configuramos el timezone.
- Nos aseguramos que la opción de arranque por consola este activado.
- En Opciones Avanzadas activamos SSH.

Nos aseguramos de reiniciar después de cambiar la configuración. Una vez hecho esto ya podemos conectarnos por SSH a la raspberry pi, deberemos tener acceso a la red y saber la IP de la raspberry pi.

```
$ ssh pi@192.168.1.10
```

Por defecto la password de la raspberry es raspberry.

Cuando nos hemos conectado por ssh a la raspberry ya podemos comenzar con el manual de configuración.

A.2 Instalamos hostapd y dhcp server

```
$ sudo apt-get update
$ sudo apt-get install hostapd isc-dhcp-server
```

Con esto actualizamos los repositorios para asegurarnos que se descargan los últimos paquetes requeridos por nuestro sistema. También instalamos el servidor dhcp (protocolo de configuración dinámica de host) y el hostapd (permite autenticar redes IEEE 802,11).

```
$ [FAIL] Starting ISC DHCP server: dhcpd
$ [....] check syslog for diagnostics. ... failed!
$ failed!
```

Esto no importa, no afecta a la funcionalidad.

A.3 Configuramos el Servidor DHCP

```
$ sudo nano /etc/dhcp/dhcpd.conf
```

Buscamos las siguientes lineas:

```
1       option domain-name "example.org";
2       option domain-name-servers ns1.example.org, ns2.example.org;
```

y las cambiamos por estas:

```
1       #option domain-name "example.org";
2       #option domain-name-servers ns1.example.org, ns2.example.org;
```

Buscamos también:

```
1       # If this DHCP server is the official DHCP server for the local
2       # network, the authoritative directive should be uncommented.
3       #authoritative;
```

y las cambiamos por:

```
1       # If this DHCP server is the official DHCP server for the local
2       # network, the authoritative directive should be uncommented.
3       authoritative;
```

Al final del fichero añadimos:

```
subnet 192.168.42.0 netmask 255.255.255.0 {
        range 192.168.42.10 192.168.42.50;
        option broadcast-address 192.168.42.255;
        option routers 192.168.42.1;
        default-lease-time 600;
        max-lease-time 7200;
        option domain-name "local";
        option domain-name-servers 8.8.8.8, 8.8.4.4;
        }
```

Estas lineas configuran la dirección de red del servidor DNS e información del gateway que la Rpi asignará a los clientes que se conecten. Si tu usas la Rpi como puente para acceder a otra red mediante conexión ethernet. Es importante que este rango de direcciones no entre en conflicto con la red a la que esta conectada la Rpi. En otras palabras, si tu red local usa este rango de direcciones 192.168.42.xx necesitaras seleccionar un rango diferente de direcciones. La dirección del routher 192.168.42.1 debe ser la dirección de la Rpi.

Para guardar los cambios realizados Control-X pulsamos Y después return. Ahora debemos editar el siguiente archivo /etc/default/isc-dhcp-server

```
$ sudo nano /etc/default/isc-dhcp-server
```

Buscamos la linea que pone:

```
interfaces=""
```

y la cambiamos por:

```
interfaces="wlan0"
```

Para guardar los cambios realizados Control-X pulsamos Y después return. Configuramos la interfaz wlan0 como una dirección estática

```
$ sudo ifdown wlan0
$ sudo nano /etc/network/interfaces
```

Cambiamos el archivo por lo siguiente:

```
auto lo
iface lo inet loopback
iface eth0 inet dhcp
allow-hotplug wlan0
iface wlan0 inet staticaddress 192.168.42.1
netmask 255.255.255.0
```

```
7       #wpa-roam /etc/wpa_supplicant/wpa_supplicant.conf
8       #iface default inet dhcp
```

Para guardar los cambios realizados Control-X pulsamos Y después return. Esto configura una dirección estática para la interfaz inalámbrica (wlan0). Esta configuración tendrá efecto una vez que se reinicie, también si quieres aplicarlo inmediatamente puedes asignar esta dirección escribiendo lo siguiente:

```
$ sudo ifconfig wlan0 192.168.42.1
```

A.4 Configurar los detalles del punto de acceso

Creamos el siguiente archivo:

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

añadimos la siguiente configuración:

```
1       interface=wlan0
2       driver=rtl871xdrv
3       ssid=NombrePuntoAcceso
4       hw_mode=g
5       channel=6
6       macaddr_acl=0
7       auth_algs=1
8       ignore_broadcast_ssid=0
9       wpa=2
10      wpa_passphrase=clavemasde8caracteres
11      wpa_key_mgmt=WPA-PSK
12      wpa_pairwise=TKIP
13      rsn_pairwise=CCMP
```

Para guardar los cambios realizados Control-X pulsamos Y después return. Ahora editamos el siguiente fichero:

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

Buscamos la siguiente linea:

```
1       DAEMON_CONF=""
```

y la cambiamos por esta:

```
1       DAEMON_CONF="/etc/hostapd/hostapd.conf"
```

Para guardar los cambios realizados Control-X pulsamos Y después return.

A.5 Configuración NAT

Configurar NAT permite a los clientes WIFI de la Rpi tener un túnel hacia la interfaz ethernet.

```
$ sudo nano /etc/sysctl.conf
```

Buscamos las siguientes lineas:

```
# Uncomment the next line to enable packet forwarding for Ipv4
# net.ipv4.ip_forward=1
```

y las dejamos así:

```
# Uncomment the next line to enable packet forwarding for Ipv4
net.ipv4.ip_forward=1
```

Este cambio no será aplicado hasta la próxima vez que se reinicie. Para aplicarlo inmediatamente ejecutamos lo siguiente:

```
$ sudo sh -c "echo 1 > /proc/sys/net/ipv4/ip_forward"
```

Ahora ejecutamos los siguientes comandos para configurar las tablas de rutas entre la interfaz inalámbrica y el puerto ethernet:

```
$ sudo iptables -t nat -A POSTROUTING -o eth0 -j MASQUERADE
$ sudo iptables -A FORWARD -i eth0 -o wlan0 -m state -state RELATED,ESTABLISHED -j ACCEPT
$ sudo iptables -A FORWARD -i wlan0 -o eth0 -j ACCEPT
```

Comprobamos que los cambios se han aplicado en las tablas:

```
$ sudo iptables -t nat -S
$ sudo iptables -S
```

Para restaurar estos cambios después de reiniciar necesitamos grabar la configuración en un archivo para que pueda usarse la próxima vez que se reinicie.

```
$ sudo sh -c "iptables-save > /etc/iptables.ipv4.nat"
```

Ahora actualizamos el archivo de interfaces otra vez:

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

y añadimos la siguiente linea al final del fichero:

```
1       up iptables-restore < /etc/iptables.ipv4.nat
```

El archivo completo debe ser así:

```
1       auto lo
2       iface lo inet loopback
3       iface eth0 inet dhcp
4       allow-hotplug wlan0
5       iface wlan0 inet static
6       address 192.168.42.1
7       netmask 255.255.255.0
8       #wpa-roam /etc/wpa_supplicant/wpa_supplicant.conf
9       #iface default inet dhcp
10      up iptables-restore < /etc/iptables.ipv4.nat
```

Para guardar los cambios realizados Control-X pulsamos Y después return. Ya tenemos listo el software para ejecutar el punto de acceso. Antes actualizaremos la versión que soportará nuestro adaptador.

A.6 Actualizamos hostapd

La versión de hostapd instalada por apt-get no tiene todos los adaptadores WIFI que usaremos, por eso actualizamos a la ultima versión:

```
$ wget http://adafruit-download.s3.amazonaws.com/adafruit_hostapd_14128.zip
```

Descargamos y cambiamos la vieja versión por la nueva, asignando permisos de ejecución:

```
$ unzip adafruit_hostapd_14128.zip
$ sudo mv /usr/sbin/hostapd /usr/sbin/hostapd.ORIG
$ sudo mv hostapd /usr/sbin
$ sudo chmod 755 /usr/sbin/hostapd
```

Configuramos el demonio

```
$ sudo service hostapd start
$ sudo service isc-dhcp-server start
```

Debemos ver el siguiente mensaje en el terminal:

```
$ pi@MavStation ~ $ sudo service hostapd start
$ [ ok ] Starting advanced IEEE 802.11 management: hostapd.
$ pi@MavStation ~ $ sudo service isc-dhcp-server start
$ [ ok ] Starting ISC DHCP server: dhcpd.
```

Ahora en orden activamos los servicios al inicio:

```
$ sudo update-rc.d hostapd enable
$ sudo update-rc.d isc-dhcp-server enable
```

El último paso es eliminar WPASupplicant para no interferir con el punto de acceso:

```
$ sudo mv /usr/share/dbus-1/system-services/fi.epitest.hostap.WPASupplicant.service ~/
```

Finalmente reinicia tu Rpi:

```
$ sudo reboot
```

A.7 Consideraciones importantes

- Asegurarnos que la Rpi recibe potencia suficiente de energía para el adaptador WIFI y la conexión Ethernet, 2A sería suficiente.
- Si consigues ver la red inalámbrica pero no puedes conectarte, puede que entre en conflicto con otras redes vecinas, prueba a cambiar el canal, del archivo /etc/hostapd/hostapd.conf
- Asegúrate que la clave de la red tiene 8 caracteres por lo menos.
- Verifica que tu adaptador de red es compatible y lo reconoce tu Rpi.

Anexo B

EL USO DE LOS UAV'S EN ESPAÑA

La Agencia Estatal de Seguridad Aérea (AESA) es la entidad responsable de controlar que el uso de aeronaves tripuladas por control remoto en España se realiza en el ámbito de la ley y la seguridad.

El uso de este tipo de aparatos es reciente y por ello, con el objetivo de evitar mal entendidos y posibles incidentes, AESA quiere aclarar en qué circunstancias y condiciones se pueden usar los UAVs y en cuales no, y qué consecuencias tiene hacerlo en este último caso.

B.1 ¿Qué es un dron?

Un UAV es una aeronave pilotada por control remoto. Así se llamaba tradicionalmente a algunas de estas aeronaves de uso militar y en la actualidad se ha extendido este nombre a todas las aeronaves pilotadas por control remoto, tanto militares como civiles.

Sin embargo, una aeronave pilotada por control remoto técnicamente se considera UAV cuando tienen un uso comercial o profesional. Cuando el uso de estas aeronaves tiene exclusivamente un fin deportivo o de recreo, son consideradas Aeromodelos, y se rigen bajo la normativa de éstos.

Hay que subrayar pues que los UAV SON AERONAVES. Como tales, están sujetas a la legislación aeronáutica general vigente en España, así como al resto de la normativa aeronáutica.

B.2 ¿Se pueden usar UAV en España?

En España no está permitido el uso de UAV para aplicaciones civiles (para uso militar existe una normativa que permite su operación exclusivamente en espacio aéreo segregado).

Es decir, no está permitido, y nunca lo ha estado, el uso de aeronaves pilotadas por control remoto con fines comerciales o profesionales, para realizar actividades consideradas trabajos aéreos, como la fotogrametría, agricultura inteligente (detectar en una finca aquellas plantas específicas que necesitarían de una intervención, como riego, fumigación, para optimizar el cultivo), reportajes gráficos de todo tipo, inspección de líneas de alta tensión, ferrovia-

rias, vigilancia de fronteras, detección de incendios forestales, reconocimiento de los lugares afectados por catástrofes naturales para dirigir las ayudas adecuadamente, etc.

B.3 El uso de UAV/aeromodelos por particulares para fines deportivos o de recreo

La actividad del Aeromodelismo la regula la Real Federación Aeronáutica de España y además, cada Comunidad Autónoma y cada Municipio puede tener su regulación sobre esta práctica deportiva o lúdica, aunque siempre deben respetar la legislación aeronáutica general. Los aeromodelos vuelan por debajo de los 100 metros de altura y no pueden volar sobre núcleos urbanos ni sobre grupos de población (playas, conciertos, las calles de cualquier ciudad, etc...). Deben volar en zonas habilitadas para ello. Lo contrario, puede suponer sanciones y se debe denunciar.

Por tanto los particulares que adquieran en una tienda generalista un equipo ligero y de fácil uso con sistema de radio control (R/C) y GPS, con o sin cámara incorporada, o compren un kit para montar un multirrotor con autopiloto, con una mini-cámara, o construyan ellos mismos un avión para FPV (vuelo con "visión en primera persona"), con cámara de visión frontal, piloto automático, transmisión de vídeo, sólo podrán usarlo en las zonas habilitadas para ello conforme a la normativa que regula las actividades de aeromodelismo. Deben consultar la normativa de su municipio o comunidad autónoma, además se recomienda que se pongan en contacto con algún club de aeromodelismo de su localidad para poder volar los aeromodelos con seguridad. En ningún caso podrán utilizarlos para una actividad profesional o con carácter comercial.

B.4 El uso profesional de los UAV/ trabajos aéreos

Como se ha indicado más arriba, en España no está permitido, y nunca lo ha estado, el uso de aeronaves pilotadas por control remoto con fines comerciales o profesionales, para realizar actividades consideradas trabajos aéreos, como la fotogrametría, agricultura inteligente (detectar en una finca aquellas plantas específicas que necesitarían de una intervención, como riego, fumigación, para optimizar el cultivo), reportajes gráficos de todo tipo, inspección de líneas de alta tensión, ferroviarias, vigilancia de fronteras, detección de incendios forestales, reconocimiento de los lugares afectados por catástrofes naturales para dirigir las ayudas adecuadamente, etc.

La realización de trabajos especializados (también llamados trabajos aéreos), como son las filmaciones aéreas, los de vigilancia, de detección y / o extinción de incendios, de cartografía, de inspección, etc., tal como indican los artículos 150 y 151 de la Ley 48/1960 sobre Navegación Aérea, requiere autorización por parte de AESA, y hasta que no esté aprobada la nueva normativa específica que regule el uso de este tipo de aparatos, AESA no puede emitir dichas autorizaciones porque carece de base legal para ello. Por tanto, utilizar UAV para la

realización de este tipo de trabajos con fines profesionales o comerciales sin autorización es ilegal y está sujeto a la imposición de las correspondientes sanciones. Lo anterior incluye tanto la realización de ese tipo de trabajos por cuenta de terceros como por cuenta propia con carácter privado.

La legislación aeronáutica general vigente contiene una serie de disposiciones que no hacen posible el vuelo de los UAV en la mayor parte de los casos. La regulación específica de estas aeronaves, en la que AESA está trabajando en colaboración con la industria, contendrá disposiciones particulares para ellas, que sustituyan o complementen a las generales y hagan posible su vuelo con determinadas condiciones y limitaciones.

La nueva normativa establecerá una clasificación de estas aeronaves, especificando qué categorías quedarán exentas de disponer de matrícula y certificado de aeronavegabilidad y estableciendo los requisitos para la certificación de las que lo requieran, así como para su fabricación, mantenimiento y operación, y para su acceso al espacio aéreo, determinando en particular en qué lugares y bajo qué condiciones podrán volar, y las medidas de seguridad específicas que puedan requerirse en cada uno de esos lugares.

Mientras no se publique, no se pueden utilizar ese tipo de aeronaves para realizar trabajos aéreos. La Agencia puede dar únicamente autorizaciones puntuales para vuelos de desarrollo o de demostración, así como para los vuelos requeridos para la certificación de estas aeronaves.

B.5 La denominada “capa de libre circulación”

Existe la creencia, errónea, de que en la capa de espacio aéreo que se extiende desde el suelo hasta 400 pies se puede volar con estos aparatos sin restricciones. Esta creencia puede tener su origen en que las aeronaves tripuladas deben permanecer normalmente por encima de los 500 pies sobre el terreno, salvo para el despegue y el aterrizaje. Sin embargo, la competencia de AESA sobre la seguridad del espacio aéreo se extiende hasta el suelo.

B.6 Vuelo de UAVs en recintos cerrados

Los recintos completamente cerrados (un pabellón industrial o deportivo, un centro de convenciones, un domicilio particular, etc.) no están sujetos a la jurisdicción de AESA, al no formar parte del espacio aéreo. Los titulares de esos recintos pueden decidir si autorizan el vuelo de UAVs en su interior y en qué condiciones. Un estadio de fútbol no tiene la consideración de recinto cerrado, a menos que su cubierta cubra la totalidad de su superficie, sin abertura ninguna.

Anexo C

Legislación y marco normativo

El Consejo de Ministros del pasado viernes 4 de julio aprobó el régimen temporal para las operaciones con aeronaves pilotadas por control remoto, los llamados drones, de peso inferior a los 150 kg al despegue, en el que se establecen las condiciones de explotación de estas aeronaves para la realización de trabajos técnicos y científicos.

Esta nueva regulación responde a la necesidad de establecer un marco jurídico que permita el desarrollo en condiciones de seguridad de un sector tecnológicamente puntero y emergente, y será desarrollada reglamentariamente en los próximos meses.

Este reglamento temporal contempla los distintos escenarios en los que se podrán realizar los distintos trabajos aéreos y en función del peso de la aeronave. Además, las condiciones ahora aprobadas se completan con el régimen general de la Ley 48/1960, de 21 de julio, sobre Navegación Aérea, y establecen las condiciones de operación con este tipo de aeronaves, además de otras obligaciones.

Las condiciones y requisitos se pueden consultar en el [BOE14] del sábado 5 de julio de 2014.

La Resolución de la Directora de la Agencia Estatal de Seguridad Aérea [AES14] por la que se adoptan medios aceptables de cumplimiento y material guía para la emisión de títulos habilitantes, se detallan a continuación:

- Apéndice A.1 Art. 50.3
- Apéndice A.2, art. 50.4
- Apéndice B.1, art. 50.3
- Apéndice B.2, art. 50.4
- Apéndice C. Normativa aplicable (art. 50.1)
- Apéndice D. Caracterización del sistema RPAS para las operaciones previstas en los puntos 3.a) y b) y 4 del artículo 50 con RPAS de menos de 25 Kg de MTOM (art. 50 3.d.1º, 50.4 y 50.6.d)
- Apéndice E. Contenido del Manual de operaciones (art. 50.3.d.1º)

- Apéndice F. Estudio aeronáutico de seguridad en la operación de aeronaves pilotadas por control remoto (art. 50.3.d.3º, 50.4 y 50.6)
- Apéndice G. Contenido de "los vuelos de prueba que resulten necesarios para demostrar que la operación pretendida puede realizarse con seguridad" (art. 50.3.d.4 y 50.6).
- Apéndice H. Revisiones y pruebas a incluir en el programa de mantenimiento de una aeronave pilotada por control remoto (art. 50, 3.d.5 y 50.6)
- Apéndice I, revisión 1 (31.07.2014). Medios aceptables para acreditar el cumplimiento de los requisitos para los pilotos para la operación de aeronaves pilotadas por control remoto (art. 50.5)

Anexo D

Diagramas de bloques de los distintos componentes

D.1 Diagrama completo de la aeronave

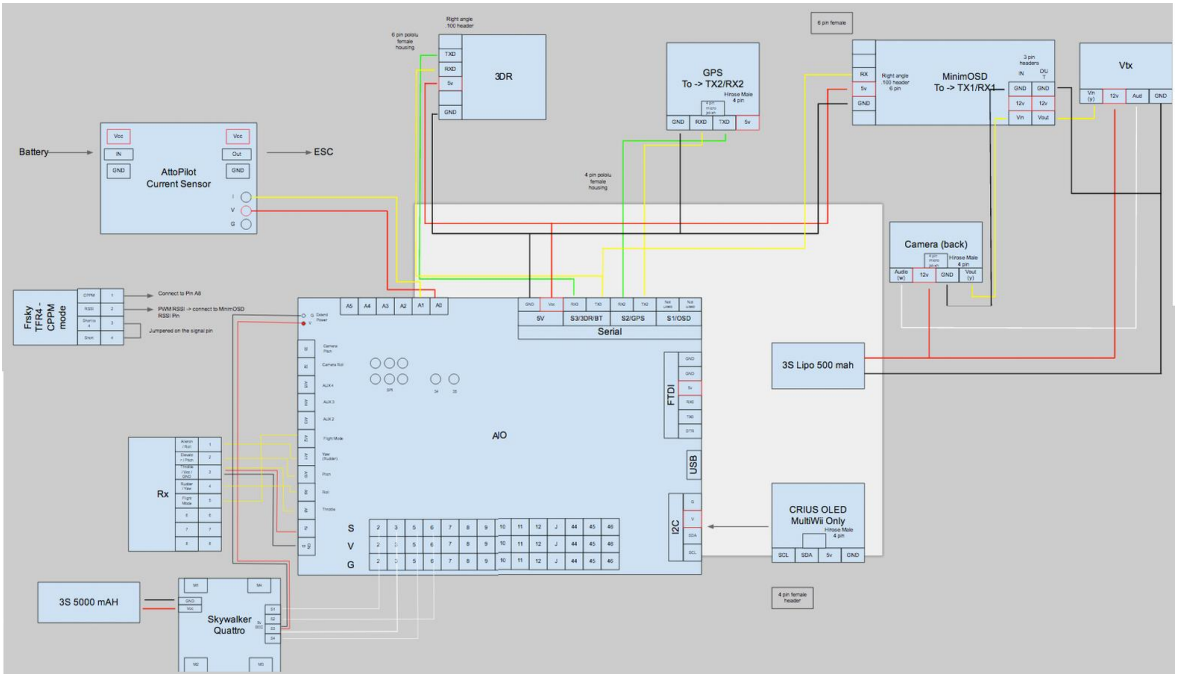

Figura D.1: Diagrama de bloques de la aeronave completa.

D.2 Controladora de vuelo

D.3 Telemetría

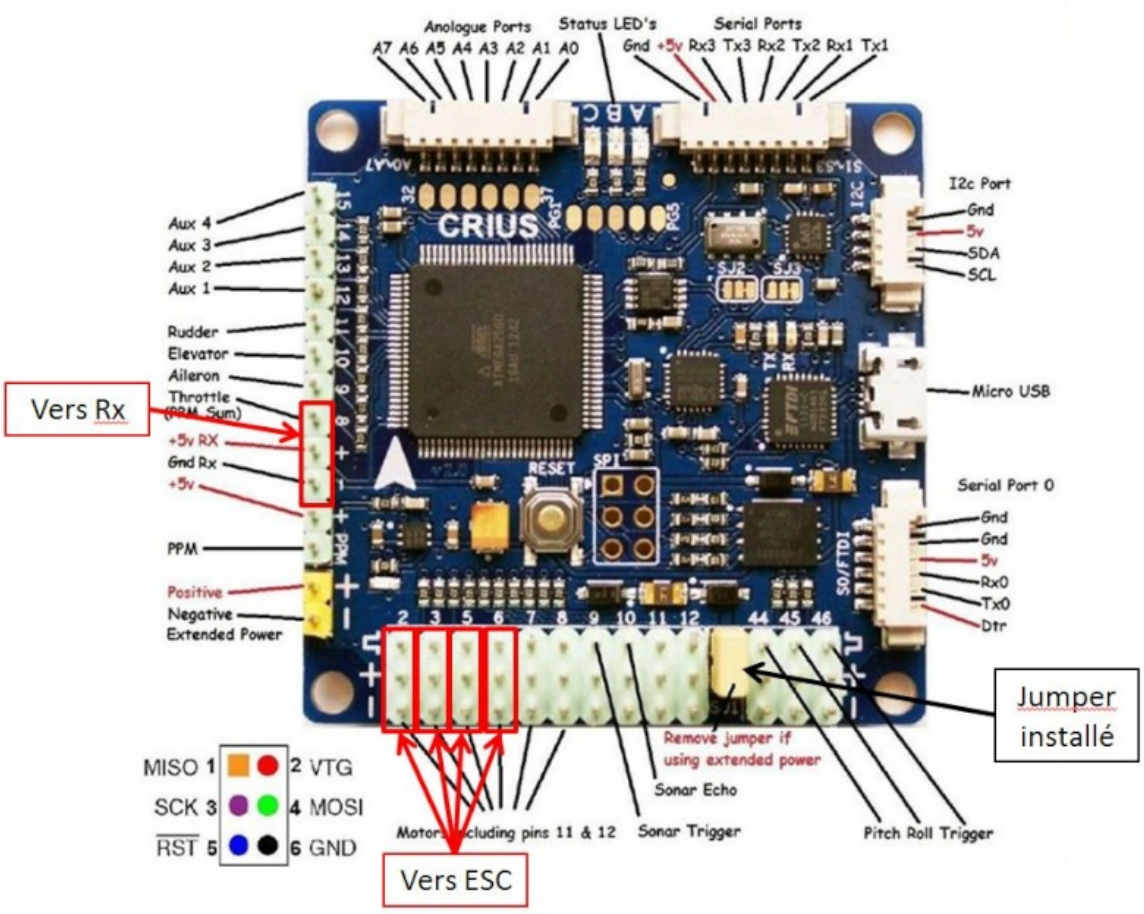

Figura D.2: Diagrama de bloques de la controladora de vuelo.

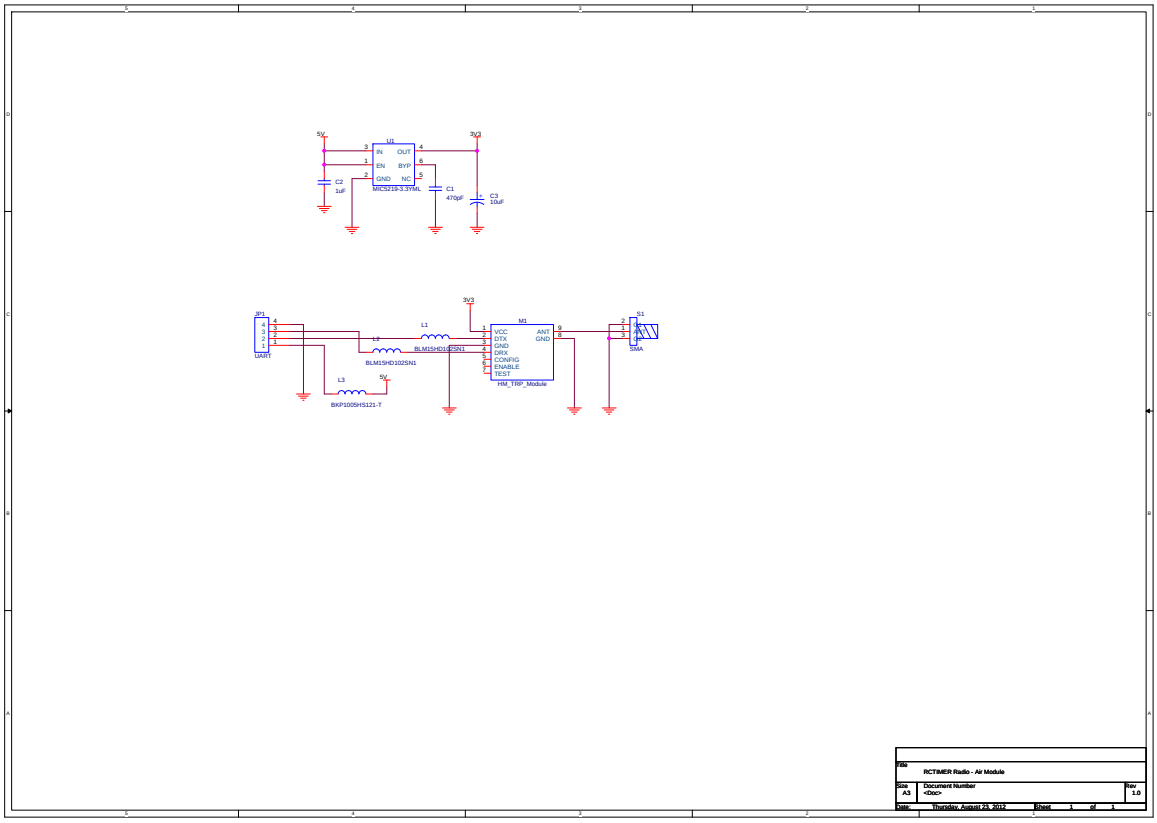

Figura D.3: Telemetría Modulo Aéreo.

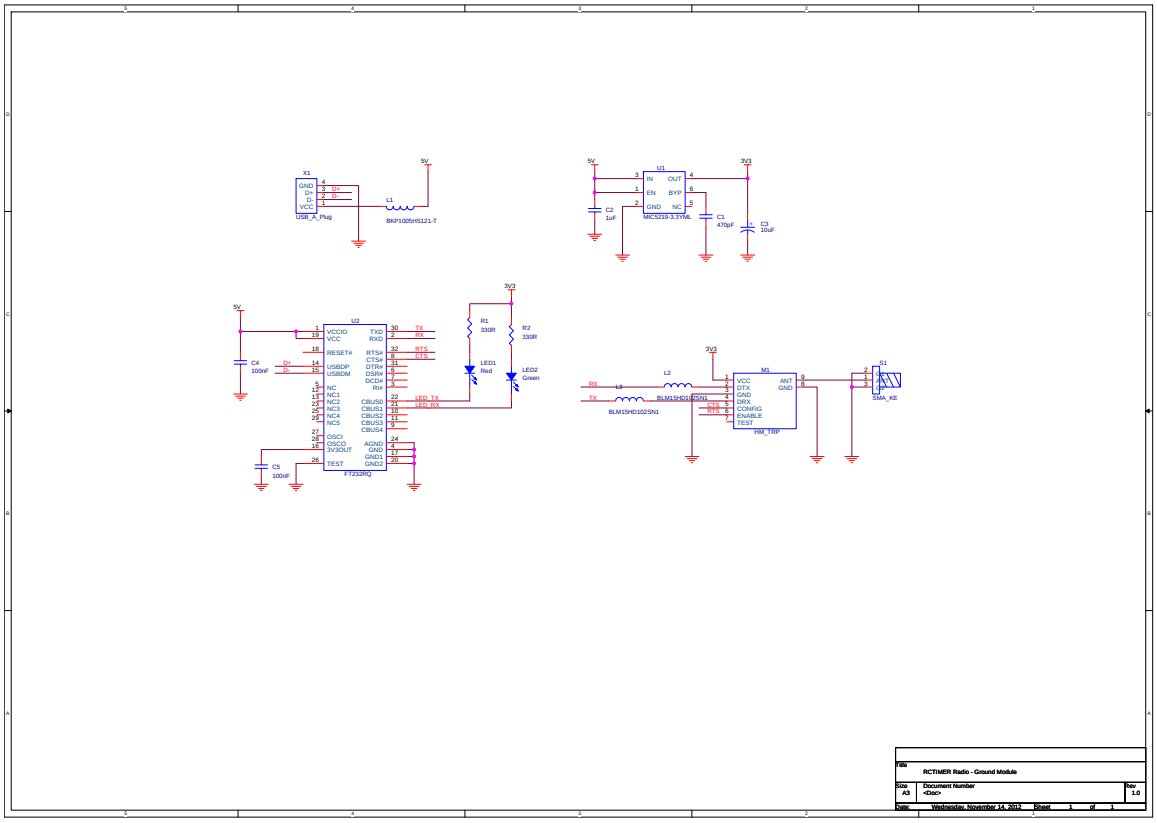

Figura D.4: Telemetría Modulo Terreno.

Anexo E

Modos de Vuelo

E.1 Estable

Como su propio nombre indica, hará que nuestro multicopter tenga un vuelo muy estable. Disponemos de diversos parámetros para configurar su comportamiento, como velocidad de giro, ángulo máximo de inclinación y volteo.

E.2 Alt Hold

Este modo de vuelo lo que hace es bloquear la altura. Podemos manejar a nuestro antojo el multicopter pero sin descender de la altura a la que lo hemos activado. Para activar este modo debemos estar en la altura deseada.

E.3 Loiter (sin rumbo)

Este modo lo que hace es dejar el multicopter en posición estacionaria lo más estable posible y en su posición de altura, longitud y latitud.

E.4 RTL

Este modo es volver a casa. En el momento de activarlo el multicopter ascenderá hasta una altura de 15 metros a no ser que lo configuremos para tener otra altura.

E.5 Auto

Este modo pasa a vuelo automático, deberemos tener la ruta a realizar con los diferentes puntos de control.

E.6 Acro

Este modo es vuelo acrobático. Es el más difícil de controlar, el acelerador actúa directamente sobre los motores.

E.7 Sport

Este modo de vuelo hace que la altitud y velocidad sean estables. Ideal para FPV y grabación de vídeo, nos facilita su manejo, haciendo que nos preocupemos de otras cosas.

E.8 Drift

Este modo de vuelo realiza un control más natural del multicopter, realizando unos giros controlando balanceo y cabeceo automáticamente.

E.9 Guided

Este modo de vuelo esta especialmente diseñado para usar la estación terrena y el sistema de telemetría para guiar al multicopter por una ruta establecida. Es interactivo con lo que en tiempo real podemos decirle donde queremos que se dirija.

E.10 Circle

Este modo de vuelo realiza un vuelo en circulo con el morro apuntando al centro del circulo con un radio de 10 m. pero configurable.

E.11 Position

Este modo de vuelo actúa igual que el loiter pero puedes controlar el acelerador.

E.12 Land

Modo aterrizaje, cuando lo activamos coge el mando del multicopter y desciende hasta llegar al suelo, apagando motores y desarmando.

E.13 Follow Me

Este modo de vuelo lo que hace que el multicopter te siga a donde tu valla, necesitas un sistema de telemetría, un receptor de GPS en el portátil y el portátil con el Mission Planner.

E.14 Simple and Super Simple

Este modo de vuelo se puede utilizar con otros modos con el Estable, Sport, Drift y Land. Lo que hace es que el control del multicopter varié dependiendo de su posición.

Anexo F

Formato del fichero con los puntos de control para realizar rutas

```
QGC WPL 110
0       1       0       16      0.000000        0.000000        0.000000        0.000000
    38.990383       -3.920056       0.000000        1
1       0       0       22      0.000000        0.000000        0.000000        0.000000
    38.970222       -3.950937       8.590000        1
2       0       0       21      0.000000        0.000000        0.000000        0.000000
    38.970203       -3.950919       8.590000        1
3       0       0       21      0.000000        0.000000        0.000000        0.000000
    38.970203       -3.950916       8.930000        1
4       0       0       21      0.000000        0.000000        0.000000        0.000000
    38.970203       -3.950917       9.110000        1
5       0       0       21      0.000000        0.000000        0.000000        0.000000
    0.000000        0.000000        1.910000        1
6       0       0       21      0.000000        0.000000        0.000000        0.000000
    0.000000        0.000000        2.210000        1
7       0       0       21      0.000000        0.000000        0.000000        0.000000
    0.000000        0.000000        2.670000        1
8       0       0       21      0.000000        0.000000        0.000000        0.000000
    0.000000        0.000000        2.670000        1
9       0       0       21      0.000000        0.000000        0.000000        0.000000
    0.000000        0.000000        2.860000        1
10      0       0       21      0.000000        0.000000        0.000000        0.000000
    0.000000        0.000000        2.840000        1
```

Anexo G

Manual de Usuario

A continuación se detallan los pasos a seguir para ejecutar correctamente la aplicación de transporte de mercancias de corta distancia basado en UAV. El sistema consta de 3 tres componentes principales como son el UAV, la estación terrena y un cliente web.

Lo primero que tenemos que hacer es encender nuestra estación terrena, para ello existen dos posibilidades:

- Con báteria de 5 V
- Con un adaptador enchufado a 220 V

Una vez encendida e iniciada nuestra estación terrena, iniciamos sesión:

```
$ nick:droneheal
$ passwork:droneheal
```

Debemos asegurarnos que la estación terrena ha sido descargada y se han instalado todos los paquetes necesarios. La estación terrena se puede descargar del repositorio del proyecto y los paquetes necesarios de sus repectivas fuentes.

Antes de ejecutar los servidores nos debemos asegurar que nuestro UAV esta listo para poder usarse, para ello debemos instalar las respectivas baterias y encenderlo. Nos aseguraremos que nuestra estación terrena tiene enchufado el adaptador de telemetría.

Ahora podemos lanzar los servidores. Debemos comprobar por seguridad la ip de nuestro dispositivo mediante el siguiente comando:

```
$ sudo ifconfig
```

También tenemos que asegurarnos donde se conecto nuestro adaptador de telemetria, para ello ejecutamos el siguiente comando:

```
$ dmesg
```

Una vez que sabemos nuestra ip, y donde se ha conectado nuestro adaptador de telemetria, ya podemos lanzar nuestros servidores, y lo haremos de la siguiente forma:

```
$ ./server 127.0.0.1 8000
```

Asegurate de estar en el directorio raiz del proyecto, y donde pone 127.0.0.1 debes poner tu ip que obtuviste del comando ifconfig y el puerto que tu quieras. Sino tienes permiso para usar los puertos serie debes ejecutarlo como administrador.

Ya tienes los servidores lanzados y esperando a funcionar. Ahora debes conectarte con cualquier cliente web que quieras y poner en el navegador la dirección que pusiste a la hora de lanzar los servidores y su respectivo puerto de la siguiente forma:

127.0.0.1:8000

Así es como se accede a la parte pública de la aplicación para que la puedan usar los operarios, para acceder a la parte privade debes poner lo siguiente en tu navegador:

127.0.0.1:8000/admin

Te pedirá una contraseña, debes hablar con tu administrador para que te facilite una.

Para navegar por la aplicación, dispones de dos pantallas, una donde visualizaras todos los trayectos en un mapa de tu zona. Y otra donde podras realizar los envios. No hace falta que se explique como funciona ya que es muy intuitiva y facil de usar.

Anexo H

Código relevante

H.1 servants.py

```
# -*- mode: python; coding: utf-8 -*-

from droneapi.lib import VehicleMode, Location
import time, sys, socket
import thread, threading
import mavproxyapi, mision
import wise

ALTITUDVUELO = 20
ALTITUDENTREGA = 2

modulo = mavproxyapi.mpstate.modules[-1][0]
conexion = modulo.get_connection()
vehiculo = conexion.get_vehicles()[0]
observable = wise.Observable()


def despegar(latorigen, lonorigen):
        try:
                if vehiculo.mode.name == "INITIALISING":
                        print "Vehiculo todavia esta arrancando, intentalo mas tarde"
                        return

                cmds = vehiculo.commands
                vehiculo.mode = VehicleMode("GUIDED")
                dest = Location(latorigen, lonorigen, ALTITUDVUELO, is_relative=True)
                cmds.goto(dest)
                vehiculo.flush()
                time.sleep(10)

        except socket.error:
                print "Error: no puede acceder a los datos del GPS"

def viajar(latdestino, londestino):
        try:
                if vehiculo.mode.name == "INITIALISING":
                        print "Vehiculo todavia esta arrancando, intentalo mas tarde"
                        return

                cmds = vehiculo.commands
                vehiculo.mode = VehicleMode("GUIDED")
                dest = Location(latdestino, londestino, ALTITUDVUELO, is_relative=True)
```

```
43              cmds.goto(dest)
44              vehiculo.flush()
45              time.sleep(10)

47      except socket.error:
48              print "Error: no puede acceder a los datos del GPS"

50def entregar(latdestino, londestino):
51      try:
52              if vehiculo.mode.name == "INITIALISING":
53                      print "Vehiculo todavia esta arrancando, intentalo mas tarde"
54                      return

56              cmds = vehiculo.commands
57              vehiculo.mode = VehicleMode("GUIDED")
58              dest = Location(latdestino, londestino, ALTITUDENTREGA, is_relative=True)
59              cmds.goto(dest)
60              vehiculo.flush()
61              time.sleep(10)

63      except socket.error:
64              print "Error: no puede acceder a los datos del GPS"

66def aterrizar(latdestino, londestino):
67      try:
68              if vehiculo.mode.name == "INITIALISING":
69                      print "Vehiculo todavia esta arrancando, intentalo mas tarde"
70                      return

72              cmds = vehiculo.commands
73              vehiculo.mode = VehicleMode("GUIDED")
74              dest = Location(latdestino, londestino, 0, is_relative=True)
75              cmds.goto(dest)
76              vehiculo.flush()
77              time.sleep(10)

79      except socket.error:
80              print "Error: no puede acceder a los datos del GPS"

82def Pedido(latorigen, lonorigen, latdestino, londestino):
83      despegar(latorigen, lonorigen)
84      _observable.notify("set_message", "Despegando...")
85      viajar(latdestino, londestino)
86      _observable.notify("set_message", "Viajando al destino...")
87      entregar(latdestino, londestino)
88      _observable.notify("set_message", "Bajando para la entrega...")
89      viajar(latorigen, lonorigen)
90      _observable.notify("set_message", "Volviendo...")
91      aterrizar(latorigen, lonorigen)
92      _observable.notify("set_message", "Aterrizando...")
93      _observable.notify("set_message", "Pedido realiazado correctamente")

95def crear_mision(latorigen, lonorigen, latdestino, londestino):
96      mision = Mision(latorigen, lonorigen, latdestino, londestino, ALTITUDVUELO, ALTITUDENTREGA)
97      mision.hacer_ruta()
98      mision.crear_archivo()

100     print "Generando %s waypoints..." % len(mision.wp)
```

```
101         cmds = vehiculo.commands
102         cmds.clear()
103         for i in xrange(0, len(mision.wp)):
104                 cmds.add(mision.wp[i])
105
106         vehiculo.flush()
107
108         if vehiculo.mode.name == "INITIALISING":
109                 print "Esperando a que el vehiculo arranque"
110                 time.sleep(1)
111
112         while vehiculo.gps_0.fix_type < 2:
113                 print "Esperando al GPS...:", vehiculo.gps_0.fix_type
114                 time.sleep(1)
115
116         vehiculo.mode    = VehicleMode("AUTO")
117         vehiculo.armed   = True
118         vehiculo.flush()
119
120 def send_position(observable):
121     vehiculo.wait_init()
122     observable.notify("set_message", "Conectando con el drone ...")
123     time.sleep(20)
124     observable.notify("set_message", "Conectado")
125     while True:
126         time.sleep(1)
127         lat = vehiculo.location.lat
128         lon = vehiculo.location.lon
129         alt = vehiculo.location.alt
130         observable.notify("set_coor", lon, lat)
131         observable.notify("set_locatorScreen", alt)
132
133 def initialize(broker):
134
135     adapter = broker.createObjectAdapter("Adapter", "-w ws1")
136     adapter.add(_observable, "Observable")
137     time.sleep(5)
138     thread.start_new_thread(send_position, (_observable,))
```

H.2 views.py

```
1 #!/usr/bin/env python
2 # -*- coding: utf-8 -*-
3
4 from django.http import HttpResponseRedirect
5 from django.shortcuts import render_to_response, render
6 from django.core.urlresolvers import reverse
7 from mapa.models import *
8 from django.views.generic import ListView
9 import servants, thread
10
11
12 # Create your views here.
13
14 class EnvioList(ListView):
```

```
16        model = Envio


18def index(request):
20        clientes=Cliente.objects.order_by('nombre')
21        drones=Drone.objects.order_by('nombre')
22        central=Central.objects.order_by('nombre')
23        form = EnvioForm()
24        return render_to_response('mapa.html', {'form':form, 'clientes': clientes, 'drones': drones,
              'central':central})

26def add_envio(request):
27        #Si el usuario esta mandando el formulario con los datos
28        if request.method == 'POST':
29                form = EnvioForm(request.POST)
30                #Comprobamos que el formulario es correcto

32                if form.is_valid():
33                        clientes = Cliente.objects.all()
34                        cliente = clientes[int(form.data['cliente'])-1]
35                        central = Central.objects.all()[0]
36                        thread.start_new_thread(servants.Pedido, (central.lat, central.lon, cliente.
                              lat, cliente.lon,))
37#                       thread.start_new_thread(servants.crear_mision, (central.lat, central.lon,
      cliente.lat, cliente.lon,))
38                        #Guardamos los datos en la base de datos
39                        new_envio = form.save()

41                        return HttpResponseRedirect(reverse('index'))

43        else:
44                form = EnvioForm()

46        return render(request, 'envio_form.html', {'form':form})
```

H.3 mapa.html

```
{% extends "admin/base_site.html" %}
{% load i18n admin_static %}
{% load staticfiles %}
{% block extrastyle %}{{ block.super }}{% endblock %}
{% block bodyclass %}{{ block.super }} login{% endblock %}
{% block nav-global %}{% endblock %}
{% block content_title %}{% endblock %}
{% block breadcrumbs %}{% endblock %}
{% block content %}
{% if form.errors and not form.non_field_errors %}
<p class="errornote">
{% if form.errors.items|length == 1 %}{% trans "Please correct the error below." %}{% else %}{%
      trans "Please correct the errors below." %}{% endif %}
</p>
{% endif %}
{% if form.non_field_errors %}
{% for error in form.non_field_errors %}
```

```
<p class="errornote">
   {{ error }}
</p>
{% endfor %}
{% endif %}
                <style>
                        .map {
                                height: 600px;
                                width: 100%;
                        }
                        #info {
                                position: absolute;
                                font-size: 0.7em;
                                top: 578px;
                                right: 10px;
                                background-color: lightgrey;
                                border-style: solid;
                                border-width: 1px;
                                padding: 8px;
                        }
                        #leyenda {
                                position: absolute;
                                font-size: 0.7em;
                                top: 568px;
                                left: 10px;
                                background-color: lightgrey;
                                border-style: solid;
                                border-width: 1px;
                                padding: 8px;
                        }
                        #drones {
                                position: absolute;
                                font-size: 0.9em;
                                top: 88px;
                                right: 10px;
                                background-color: white;
                                border-style: solid;
                                border-width: 1px;
                                padding: 8px;
                        }
                        p {
                                text-align:center;
                        }
                        p.leyenda {
                                text-decoration: underline;
                        }
                        fallo {
                                class: danger;
                                background-color: red;
                                padding: 2px;
                        }
                        activo {
                                class: success;
                                background-color: green;
                                padding: 2px;
                        }
                        apagado {
                                class: info;
```

```
                        background-color: orange;
                        padding: 2px;
                }
        </style>
        <script src="{% static 'js/ol.js' %}" type="text/javascript"></script>
        <script src="{% static 'js/wise.js' %}" type="text/javascript"></script>
        <div id="map" class="map" alt="Cargando.."></div>
<div id="info"></div>
<div id="leyenda"></div>
<div id="drones"></div>
        <script type="text/javascript">

                var dron = new ol.style.Circle({
                        radius: 3,
                        fill: new ol.style.Fill({color: 'green'}),
                        stroke: new ol.style.Stroke({color: 'red', width: 1})
                        });

                function prepend(id, text) {
                        div = document.getElementById(id);
                        div.innerHTML = text + div.innerHTML;
                        }

                var position;

                Servant = Class.extend({
                        set_coor: function(message) {
                                position = message;
                                var coordenadasDrones = [];
                                coordenadasDrones.push(ol.proj.transform(position, 'EPSG
                                    :4326', 'EPSG:3857'));
                                map.on('postcompose', function(event) {
                                        var vectorContext = event.vectorContext;
                                        vectorContext.setImageStyle(dron);
                                        vectorContext.drawMultiPointGeometry(new ol.geom.
                                            MultiPoint(coordenadasDrones), null);
                                        map.render();
                                        });

                                map.render();
                                },
                        set_locatorScreen: function(message) {
                                var alt = parseFloat(message) || 0;
                                var fecha = new Date();
                                var hora = fecha.getHours();
                                var minutos = fecha.getMinutes();
                                var meses = fecha.getMonth()+1;
                                var day = fecha.getDate();
                                var year = fecha.getFullYear();
                                var coor = position;
                                var html = ['Fecha: ' + hora + ':' + minutos + '/' + day + '
                                    -' + meses + '-' + year,
                                        'Latitud: ' + coor[0].toFixed(5) + '&deg;',
                                        'Longitud: ' + coor[1].toFixed(5) + '&deg;',
                                        'Altitud: ' + alt.toFixed(5) + ' m',
                                        ].join('<br />');
                                document.getElementById('info').innerHTML = html;
                                },
```

```
130                 set_message: function(message) {
131                         prepend('messages', "'{0}'<br/>".format(message));
132                         },
133                 });

135         function wise_application(ws) {
136                 ws.createObjectAdapter("WebAdapter").then(on_adapter_ready);

138                 function on_adapter_ready(adapter) {
139                         var observer = adapter.addWithUUID(new Servant());
140                         ws.stringToProxy("Observable -w ws1")
141                         .then(on_stringToProxy_response);
142                         function on_stringToProxy_response(proxy) {
143                                 proxy.attach(observer).then(ws.ready);
144                                 }
145                         }
146                 }

148         var cliente = new ol.style.Circle({
149                 radius: 5,
150                 name:'Cliente',
151                 fill: new ol.style.Fill({color: 'pink'}),
152                 stroke: new ol.style.Stroke({color: 'red', width: 1})
153                 });

155         var base = new ol.style.Circle({
156                 radius: 7,
157                 fill: new ol.style.Fill({color: 'red'}),
158                 stroke: new ol.style.Stroke({color: 'black', width: 1})
159                 });

161         var vector = new ol.layer.Vector({
162                 source: new ol.source.KML({
163                         projection: 'EPSG:3857'})
164                 });

166         var raster = new ol.layer.Tile({
167                 source: new ol.source.Stamen({
168                         layer: 'toner'})
169                 });

171         var coordenadaBase = [];
172         var central = ol.proj.transform([parseFloat("{{ central.0.lat }}".toString()
                .replace(/\,/g,'.')), parseFloat("{{ central.0.lon }}".toString().
                replace(/\,/g,'.'))], 'EPSG:4326', 'EPSG:3857');
173         coordenadaBase.push(central);

175         var view = new ol.View({
176                 center: central,
177                 zoom: 15
178                 });

180         var map = new ol.Map({
181                 layers: [raster, vector],
182                 target: document.getElementById('map'),
183                 view: view
184                 });
```

```
186                     var coordenadasCli = [];
187                     {% for cliente in clientes %}
188                             coordenadasCli.push(ol.proj.transform([parseFloat("{{ cliente.lat }}
                                    ".toString().replace(/\,/g,'.')), parseFloat("{{ cliente.lon }}"
                                    .toString().replace(/\,/g,'.'))], 'EPSG:4326', 'EPSG:3857'));
189                     {% endfor %}

191                     console.log(coordenadasCli[0]);

193                     map.on('postcompose', function(event) {
194                             var vectorContext = event.vectorContext;
195                             vectorContext.setImageStyle(base);
196                             vectorContext.drawMultiPointGeometry(new ol.geom.MultiPoint(
                                    coordenadaBase), null);
197                             vectorContext.setImageStyle(cliente);
198                             vectorContext.drawMultiPointGeometry(new ol.geom.MultiPoint(
                                    coordenadasCli), null);
199                             map.render();
200                             });

202                     map.render();

204                     var leyenda = ['<p class="leyenda">Leyenda</p>',
205                             '<svg width="10" height="10"><circle cx="5" cy="5" r="5" fill="red"
                                    stroke="black" stroke-width="1"/></svg>    
                                    Central',
206                             '<svg width="10" height="10"><circle cx="5" cy="5" r="5" fill="pink"
                                     stroke="red" stroke-width="1"/></svg>    
                                    Clientes',
207                             '<svg width="10" height="10"><circle cx="5" cy="5" r="5" fill="green
                                    " stroke="red"/></svg>     Drones'].join('<
                                    br />');

209                     document.getElementById('leyenda').innerHTML = leyenda;

211                     var estado_drones = new ol.Geolocation({
212                             projection: view.getProjection(),
213                             tracking: true
214                             });

216                     estado_drones.on('change', function(estado) {
217                             var info = [
218                                     {% for dron in drones %}
219                                             '{{ dron.nombre }}' + ' - ' +
220                                             {% if dron.estado == 0 %}
221                                                     '<fallo>FALLO</fallo>',
222                                             {% endif %}
223                                             {% if dron.estado == 1 %}
224                                                     '<activo>Activo</activo>',
225                                             {% endif %}
226                                             {% if dron.estado == 2 %}
227                                                     '<apagado>Apagado</apagado>',
228                                             {% endif %}
229                                             {% if dron.estado == 3 %}
230                                                     '<activo>Armando</activo>',
231                                             {% endif %}
232                                             {% if dron.estado == 4 %}
233                                                     '<activo>Despegando</activo>',
```

```
                                        {% endif %}
                                        {% if dron.estado == 5 %}
                                                '<activo>De camino</activo>',
                                        {% endif %}
                                        {% if dron.estado == 6 %}
                                                '<activo>De vuelta</activo>',
                                        {% endif %}
                                        {% if dron.estado == 7 %}
                                                '<activo>Aterrizando</activo>',
                                        {% endif %}
                                {% endfor %}
                                ].join('<br />');
                        document.getElementById('drones').innerHTML = info;
                        });
        </script>
        <div class="row-fluid">
                <div class="span12">
                        <div id="no-download" class="alert alert-error" style="display: none
                             ">
                                Estas utilizando un navegador que no soporta este tipos de
                                     descarga. Para saber que navegadores lo permiten visita
                                <a href="http://caniuse.com/#feat=download">esta pagina.</a>
                        </div>
                        <a id="export-png" class="btn btn-default btn-md" download="mapa.png
                             "><i class="glyphicon glyphicon-download"></i> Exportar mapa a
                             PNG</a>
                </div>
        </div>
        <br>
        <p>
                <a class="btn btn-info btn-md" href="{% url 'padd' %}">
                        Realizar Pedido
                </a>
                <div id="messages" style="border: 1px solid #CCC; padding: 5px"></div>
        </p>
        <script type="text/javascript">
                var exportPNGElement = document.getElementById('export-png');

                if ('download' in exportPNGElement){
                        exportPNGElement.addEventListener('click', function(e) {
                                map.once('postcompose', function(event) {
                                        var canvas = event.context.canvas;
                                        exportPNGElement.href = canvas.toDataURL('image/png'
                                             );});
                                map.renderSync();},
                        false);}
                else {
                        var info = document.getElementById('no-download');
                        /**
                        * display error message
                        */
                        info.style.display = '';
                        }
        </script>
    </body>
</html>
{% endblock %}
```

H.4 views.py

Anexo I

GNU Free Documentation License

Version 1.3, 3 November 2008

0. PREAMBLE

The purpose of this License is to make a manual, textbook, or other functional and useful document "free" in the sense of freedom: to assure everyone the effective freedom to copy and redistribute it, with or without modifying it, either commercially or noncommercially. Secondarily, this License preserves for the author and publisher a way to get credit for their work, while not being considered responsible for modifications made by others.

This License is a kind of "copyleft", which means that derivative works of the document must themselves be free in the same sense. It complements the GNU General Public License, which is a copyleft license designed for free software.

We have designed this License in order to use it for manuals for free software, because free software needs free documentation: a free program should come with manuals providing the same freedoms that the software does. But this License is not limited to software manuals; it can be used for any textual work, regardless of subject matter or whether it is published as a printed book. We recommend this License principally for works whose purpose is instruction or reference.

1. APPLICABILITY AND DEFINITIONS

This License applies to any manual or other work, in any medium, that contains a notice placed by the copyright holder saying it can be distributed under the terms of this License. Such a notice grants a world-wide, royalty-free license, unlimited in duration, to use that work under the conditions stated herein. The "Document", below, refers to any such manual or work. Any member of the public is a licensee, and is addressed as "you". You accept the license if you copy, modify or distribute the work in a way requiring permission under copyright law.

A "Modified Version" of the Document means any work containing the Document or a portion of it, either copied verbatim, or with modifications and/or translated into another language.

A "Secondary Section" is a named appendix or a front-matter section of the Document that deals exclusively with the relationship of the publishers or authors of the Document to the Document's overall subject (or to related matters) and contains nothing that could fall directly within that overall subject. (Thus, if the Document is in part a textbook of mathematics, a Secondary Section may not explain any mathematics.) The relationship could be a matter of historical connection with the subject or with related matters, or of legal, commercial, philosophical, ethical or political position regarding them.

The "Invariant Sections" are certain Secondary Sections whose titles are designated, as being those of Invariant Sections, in the notice that says that the Document is released under this License. If a section does not fit the above definition of Secondary then it is not allowed to be designated as Invariant. The Document may contain zero Invariant Sections. If the Document does not identify any Invariant Sections then there are none.

The "Cover Texts" are certain short passages of text that are listed, as Front-Cover Texts or Back-Cover Texts, in the notice that says that the Document is released under this License. A Front-Cover Text may be at most 5 words, and a Back-Cover Text may be at most 25 words.

A “Transparent” copy of the Document means a machine-readable copy, represented in a format whose specification is available to the general public, that is suitable for revising the document straightforwardly with generic text editors or (for images composed of pixels) generic paint programs or (for drawings) some widely available drawing editor, and that is suitable for input to text formatters or for automatic translation to a variety of formats suitable for input to text formatters. A copy made in an otherwise Transparent file format whose markup, or absence of markup, has been arranged to thwart or discourage subsequent modification by readers is not Transparent. An image format is not Transparent if used for any substantial amount of text. A copy that is not “Transparent” is called “Opaque”.

Examples of suitable formats for Transparent copies include plain ASCII without markup, Texinfo input format, LaTeX input format, SGML or XML using a publicly available DTD, and standard-conforming simple HTML, PostScript or PDF designed for human modification. Examples of transparent image formats include PNG, XCF and JPG. Opaque formats include proprietary formats that can be read and edited only by proprietary word processors, SGML or XML for which the DTD and/or processing tools are not generally available, and the machine-generated HTML, PostScript or PDF produced by some word processors for output purposes only.

The “Title Page” means, for a printed book, the title page itself, plus such following pages as are needed to hold, legibly, the material this License requires to appear in the title page. For works in formats which do not have any title page as such, “Title Page” means the text near the most prominent appearance of the work’s title, preceding the beginning of the body of the text.

The “publisher” means any person or entity that distributes copies of the Document to the public.

A section “Entitled XYZ” means a named subunit of the Document whose title either is precisely XYZ or contains XYZ in parentheses following text that translates XYZ in another language. (Here XYZ stands for a specific section name mentioned below, such as “Acknowledgements”, “Dedications”, “Endorsements”, or “History”.) To “Preserve the Title” of such a section when you modify the Document means that it remains a section “Entitled XYZ” according to this definition.

The Document may include Warranty Disclaimers next to the notice which states that this License applies to the Document. These Warranty Disclaimers are considered to be included by reference in this License, but only as regards disclaiming warranties: any other implication that these Warranty Disclaimers may have is void and has no effect on the meaning of this License.

2. VERBATIM COPYING

You may copy and distribute the Document in any medium, either commercially or noncommercially, provided that this License, the copyright notices, and the license notice saying this License applies to the Document are reproduced in all copies, and that you add no other conditions whatsoever to those of this License. You may not use technical measures to obstruct or control the reading or further copying of the copies you make or distribute. However, you may accept compensation in exchange for copies. If you distribute a large enough number of copies you must also follow the conditions in section 3.

You may also lend copies, under the same conditions stated above, and you may publicly display copies.

3. COPYING IN QUANTITY

If you publish printed copies (or copies in media that commonly have printed covers) of the Document, numbering more than 100, and the Document’s license notice requires Cover Texts, you must enclose the copies in covers that carry, clearly and legibly, all these Cover Texts: Front-Cover Texts on the front cover, and Back-Cover Texts on the back cover. Both covers must also clearly and legibly identify you as the publisher of these copies. The front cover must present the full title with all words of the title equally prominent and visible. You may add other material on the covers in addition. Copying with changes limited to the covers, as long as they preserve the title of the Document and satisfy these conditions, can be treated as verbatim copying in other respects.

If the required texts for either cover are too voluminous to fit legibly, you should put the first ones listed (as many as fit reasonably) on the actual cover, and continue the rest onto adjacent pages.

If you publish or distribute Opaque copies of the Document numbering more than 100, you must either include a machine-readable Transparent copy along with each Opaque copy, or state in or with each Opaque copy a computer-network location from which the general network-using public has access to download using public-standard network protocols a complete Transparent copy of the Document, free of added material. If you use the latter option, you must take reasonably prudent steps, when you begin distribution of Opaque copies in quantity, to ensure that this Transparent copy will remain thus accessible at the stated location until at least one year after the last time you distribute an Opaque copy (directly or through your agents or retailers) of that edition to the public.

It is requested, but not required, that you contact the authors of the Document well before redistributing any large number of copies, to give them a chance to provide you with an updated version of the Document.

4. MODIFICATIONS

You may copy and distribute a Modified Version of the Document under the conditions of sections 2 and 3 above, provided that you release the Modified Version under precisely this License, with the Modified Version filling the role of the Document, thus licensing distribution and modification of the Modified Version to whoever possesses a copy of it. In addition, you must do these things in the Modified Version:

- A. Use in the Title Page (and on the covers, if any) a title distinct from that of the Document, and from those of previous versions (which should, if there were any, be listed in the History section of the Document). You may use the same title as a previous version if the original publisher of that version gives permission.
- B. List on the Title Page, as authors, one or more persons or entities responsible for authorship of the modifications in the Modified Version, together with at least five of the principal authors of the Document (all of its principal authors, if it has fewer than five), unless they release you from this requirement.
- C. State on the Title page the name of the publisher of the Modified Version, as the publisher.
- D. Preserve all the copyright notices of the Document.
- E. Add an appropriate copyright notice for your modifications adjacent to the other copyright notices.
- F. Include, immediately after the copyright notices, a license notice giving the public permission to use the Modified Version under the terms of this License, in the form shown in the Addendum below.
- G. Preserve in that license notice the full lists of Invariant Sections and required Cover Texts given in the Document's license notice.
- H. Include an unaltered copy of this License.
- I. Preserve the section Entitled "History", Preserve its Title, and add to it an item stating at least the title, year, new authors, and publisher of the Modified Version as given on the Title Page. If there is no section Entitled "History" in the Document, create one stating the title, year, authors, and publisher of the Document as given on its Title Page, then add an item describing the Modified Version as stated in the previous sentence.
- J. Preserve the network location, if any, given in the Document for public access to a Transparent copy of the Document, and likewise the network locations given in the Document for previous versions it was based on. These may be placed in the "History" section. You may omit a network location for a work that was published at least four years before the Document itself, or if the original publisher of the version it refers to gives permission.
- K. For any section Entitled "Acknowledgements" or "Dedications", Preserve the Title of the section, and preserve in the section all the substance and tone of each of the contributor acknowledgements and/or dedications given therein.
- L. Preserve all the Invariant Sections of the Document, unaltered in their text and in their titles. Section numbers or the equivalent are not considered part of the section titles.
- M. Delete any section Entitled "Endorsements". Such a section may not be included in the Modified Version.
- N. Do not retitle any existing section to be Entitled "Endorsements" or to conflict in title with any Invariant Section.
- O. Preserve any Warranty Disclaimers.

If the Modified Version includes new front-matter sections or appendices that qualify as Secondary Sections and contain no material copied from the Document, you may at your option designate some or all of these sections as invariant. To do this, add their titles to the list of Invariant Sections in the Modified Version's license notice. These titles must be distinct from any other section titles.

You may add a section Entitled "Endorsements", provided it contains nothing but endorsements of your Modified Version by various parties—for example, statements of peer review or that the text has been approved by an organization as the authoritative definition of a standard.

You may add a passage of up to five words as a Front-Cover Text, and a passage of up to 25 words as a Back-Cover Text, to the end of the list of Cover Texts in the Modified Version. Only one passage of Front-Cover Text and one of Back-Cover Text may be added by (or through arrangements made by) any one entity. If the Document already includes a cover text for the same cover, previously added by you or by arrangement made by the same entity you are acting on behalf of, you may not add another; but you may replace the old one, on explicit permission from the previous publisher that added the old one.

The author(s) and publisher(s) of the Document do not by this License give permission to use their names for publicity for or to assert or imply endorsement of any Modified Version. 5. COMBINING DOCUMENTS

You may combine the Document with other documents released under this License, under the terms defined in section 4 above for modified versions, provided that you include in the combination all of the Invariant Sections of all of the original documents, unmodified, and list them all as Invariant Sections of your combined work in its license notice, and that you preserve all their Warranty Disclaimers.

The combined work need only contain one copy of this License, and multiple identical Invariant Sections may be replaced with a single copy. If there are multiple Invariant Sections with the same name but different contents, make the title of each such section unique by adding at the end of it, in parentheses, the name of the original author or publisher of that section if known, or else a unique number. Make the same adjustment to the section titles in the list of Invariant Sections in the license notice of the combined work.

In the combination, you must combine any sections Entitled “History” in the various original documents, forming one section Entitled “History”; likewise combine any sections Entitled “Acknowledgements”, and any sections Entitled “Dedications”. You must delete all sections Entitled “Endorsements”.

5. COLLECTIONS OF DOCUMENTS

You may make a collection consisting of the Document and other documents released under this License, and replace the individual copies of this License in the various documents with a single copy that is included in the collection, provided that you follow the rules of this License for verbatim copying of each of the documents in all other respects.

You may extract a single document from such a collection, and distribute it individually under this License, provided you insert a copy of this License into the extracted document, and follow this License in all other respects regarding verbatim copying of that document.

6. AGGREGATION WITH INDEPENDENT WORKS

A compilation of the Document or its derivatives with other separate and independent documents or works, in or on a volume of a storage or distribution medium, is called an “aggregate” if the copyright resulting from the compilation is not used to limit the legal rights of the compilation’s users beyond what the individual works permit. When the Document is included in an aggregate, this License does not apply to the other works in the aggregate which are not themselves derivative works of the Document.

If the Cover Text requirement of section 3 is applicable to these copies of the Document, then if the Document is less than one half of the entire aggregate, the Document’s Cover Texts may be placed on covers that bracket the Document within the aggregate, or the electronic equivalent of covers if the Document is in electronic form. Otherwise they must appear on printed covers that bracket the whole aggregate.

7. TRANSLATION

Translation is considered a kind of modification, so you may distribute translations of the Document under the terms of section 4. Replacing Invariant Sections with translations requires special permission from their copyright holders, but you may include translations of some or all Invariant Sections in addition to the original versions of these Invariant Sections. You may include a translation of this License, and all the license notices in the Document, and any Warranty Disclaimers, provided that you also include the original English version of this License and the original versions of those notices and disclaimers. In case of a disagreement between the translation and the original version of this License or a notice or disclaimer, the original version will prevail.

If a section in the Document is Entitled “Acknowledgements”, “Dedications”, or “History”, the requirement (section 4) to Preserve its Title (section 1) will typically require changing the actual title.

8. TERMINATION

You may not copy, modify, sublicense, or distribute the Document except as expressly provided under this License. Any attempt otherwise to copy, modify, sublicense, or distribute it is void, and will automatically terminate your rights under this License.

However, if you cease all violation of this License, then your license from a particular copyright holder is reinstated (a) provisionally, unless and until the copyright holder explicitly and finally terminates your license, and (b) permanently, if the copyright holder fails to notify you of the violation by some reasonable means prior to 60 days after the cessation.

Moreover, your license from a particular copyright holder is reinstated permanently if the copyright holder notifies you of the violation by some reasonable means, this is the first time you have received notice of violation of this License (for any work) from that copyright holder, and you cure the violation prior to 30 days after your receipt of the notice.

Termination of your rights under this section does not terminate the licenses of parties who have received copies or rights from you under this License. If your rights have been terminated and not permanently reinstated, receipt of a copy of some or all of the same material does not give you any rights to use it.

9. FUTURE REVISIONS OF THIS LICENSE

The Free Software Foundation may publish new, revised versions of the GNU Free Documentation License from time to time. Such new versions will be similar in spirit to the present version, but may differ in detail to address new problems or concerns. See http://www.gnu.org/copyleft/.

Each version of the License is given a distinguishing version number. If the Document specifies that a particular numbered version of this License "or any later version" applies to it, you have the option of following the terms and conditions either of that specified version or of any later version that has been published (not as a draft) by the Free Software Foundation. If the Document does not specify a version number of this License, you may choose any version ever published (not as a draft) by the Free Software Foundation. If the Document specifies that a proxy can decide which future versions of this License can be used, that proxy's public statement of acceptance of a version permanently authorizes you to choose that version for the Document.

10. RELICENSING

"Massive Multiauthor Collaboration Site" (or "MMC Site") means any World Wide Web server that publishes copyrightable works and also provides prominent facilities for anybody to edit those works. A public wiki that anybody can edit is an example of such a server. A "Massive Multiauthor Collaboration" (or "MMC") contained in the site means any set of copyrightable works thus published on the MMC site.

"CC-BY-SA" means the Creative Commons Attribution-Share Alike 3.0 license published by Creative Commons Corporation, a not-for-profit corporation with a principal place of business in San Francisco, California, as well as future copyleft versions of that license published by that same organization.

"Incorporate" means to publish or republish a Document, in whole or in part, as part of another Document.

An MMC is "eligible for relicensing" if it is licensed under this License, and if all works that were first published under this License somewhere other than this MMC, and subsequently incorporated in whole or in part into the MMC, (1) had no cover texts or invariant sections, and (2) were thus incorporated prior to November 1, 2008.

The operator of an MMC Site may republish an MMC contained in the site under CC-BY-SA on the same site at any time before August 1, 2009, provided the MMC is eligible for relicensing.

Referencias

[AES14] Nuevo marco regulatorio temporal para las operaciones con drones, 2014. url: http://www.seguridadaerea.gob.es/lang_castellano/cias_empresas/trabajos/marco_drones/resolucion_directora_drones/default.aspx.

[Ard15] ArduPilot. «This is the full-featured, open-source multicopter UAV controller», 2015. url: http://copter.ardupilot.com/.

[Are14] The Flying Machine Arena. «The Flying Machine Arena offers a safe, controlled sandbox environment allowing the testing and validation of mobile robots», 2014. url: http://flyingmachinearena.org/research/.

[Aut15] «AutoQuad is an Open Source firmware project», 2015. url: http://autoquad.org/.

[Bla14] Susana Blazquez. «El negocio de los drones gana espacio», 2014. url: http://goo.gl/0GBF29.

[BOE14] Boletin Oficial del Estado - Normativa referente a drones, 2014. url: http://www.seguridadaerea.gob.es/media/4243006/rdl_8_2014_4julio.pdf.

[Cad] Pagina oficial de Cadex Electronic Inc. url: http://www.cadex.com/en.

[D'A15] Raffaello D'Andrea. Control of Distributed, Autonomous Systems, 2015. url: http://www.idsc.ethz.ch/research-dandrea.html.

[Don15] J.D. Dondo Gazzano. *Metodologías de Diseño de Sistemas*. Asignatura Sistemas Empotrados, Universidad de Castilla La Mancha, España, 2014/15.

[G04] Myers G. *The art of sotfware testing, 2nd edition, ISBN:0-471-46912-2*. Jonh Wiley and Sons Inc., 2004.

[IEE87] IEEE Standard for Software Unit Testing, 1987. url: https://www.ieee.org/.

[IEE90] IEEE Standard Glossary of Software Engineering Terminology Institute of Electrical and Electronics Engineers, ISBN: 155937067X, 1990. url: https://www.ieee.org/.

[INF14] INFODEFENSA. «El negocio de los vehículos aéreos sin tripulación será imparable en la próxima década», 2014. url: http://goo.gl/g5Eek0.

[JA06] Offutt J, Gallagher L y Cincotta A. Integration testing of object-oriented components using finite state machines. *Software Testing, Verification and Reliability*, (16), 2006.

[MAV14] MAVLink. «MAVLink is a very lightweight, header-only message marshalling library for micro air vehicles», 2014. url: http://qgroundcontrol.org/mavlink/start.

[Ope15] «OpenPilot is a next-generation Open Source UAV autopilot», 2015. url: http://www.openpilot.org/.

[QGr15] QGroundControl. «This is the home of the Open Source Micro Air Vehicle Ground Control Station/Operator Control Unit.», 2015. url: http://qgroundcontrol.org/.

[RS93] Pressman RS. *Ingeniería del Software, un enfoque práctico(3ª Edición).* McGraw-Hill, 1993.

Este documento fue editado y tipografiado con LaTeX empleando
la clase **esi-tfg** (versión 0.20150701) que se puede encontrar en:
`https://bitbucket.org/arco_group/esi-tfg`