_ TELECOM ESCUELA TÉCNICA VLC SUPERIOR DE UPV INGENIEROS DE TELECOMUNICACIÓN

UNIVERSITAT POLITÈCNICA DE VALÈNCIA

Escola Tècnica Superior d'Enginyeria Telecomunicación
Universitat Politècnica de València

DronePi: Construcción de un dron basado en Raspberry Pi

Autor: Diego Puertas Gayoso

Director: Ricardo José Colom Palero

Trabajo Fin de Grado presentado en la Escuela Técnica Superior de Ingenieros de Telecomunicación de la Universitat Politècnica de València, para la obtención del Título de Graduado en Ingeniería de Tecnologías y Servicios de Telecomunicación
Curso 2015-16
Valencia, 1 de julio de 2016

Agradecimientos

A mis superhéroes, Berta y Marcial. Los mejores maestros que uno puede tener, apoyándome desde el primer día, en las buenas y en las malas, enseñándome unos valores de los cuales no podría sentirme más orgulloso. A Ricardo, por su paciencia y sabios consejos a la hora de realizar este proyecto. A Paco, por su inestimable ayuda y porque me ha enseñado que el oficio de profesor va mucho más allá que enseñar unos determinados contenidos.

La educación es el camino a la superación personal.

Resumen

0.1. Resumen

En los últimos años se está produciendo un incremento considerable del uso de drones, tanto en aplicaciones profesionales para la mejora y seguridad de muchas labores profesionales, como en ambientes lúdicos para el disfrute personal. Un dron se puede adquirir comercialmente con unas especificaciones y dentro de un abanico económico en consonancia con la calidad y prestaciones o existe la posibilidad de construírselo uno mismo, a partir de la gran variedad de piezas que ofrece el mercado. Este segundo aspecto es el más interesante para la finalidad del presente trabajo.

Hoy en día la gran mayoría de drones son realizados únicamente con un controlador basado en Arduino y un sistema de radio control, esto limita las posibilidades ya que los microcontroladores Arduino solo son capaces de gestionar los motores.

Por el contrario, los nuevos System-On-a-Chip (SOC) de bajo coste, como pueden ser las Raspberry Pi, considerados como microordenadores de bolsillo, disponen de una mayor versatilidad debido a su gran capacidad de procesado, pudiendo realizar tanto el control del dron en tiempo real desde un dispositivo móvil, como la grabación y envío de imágenes.

Sin embargo, la Raspberry no puede controlar los motores en tiempo real, por lo que el uso de una controladora se antoja indispensable. Por lo que este proyecto versa sobre el diseño y la construcción de un cuadricoptero controlado mediante una Raspberry pi y una placa controladora Crius.

La Crius se encargará del control de los motores, a partir de los comandos suministrados por la Raspberry, que a su veces provendrán de un teléfono móvil conectado mediante Wi-Fi.

En este proyecto también se diseñara la aplicación móvil en Android para el manejo del dron. Esta aplicación constara de varias actividades, entre las cuales habrá una actividad de control, un mapa con capacidad de geolocalizar y una actividad que nos muestre el tiempo con los datos del viento como parámetro importante.

0.2. Resum

En els últims anys s'està produint un increment considerable de l'ús de drons, tant en aplicacions professionals per a la millora i seguretat de moltes labors professionals, com en ambients lúdics per al gaudi personal. Un dron es pot adquirir comercialment amb unes especificacions i dins d'un palmito econòmic d'acord amb la qualitat i prestacions o hi ha la possibilitat de construir-se'l u mateix, a partir de la gran varietat de peces que ofereix el mercat. Este segon aspecte és el més interessant per a la finalitat del present treball.

Hui en dia la gran majoria de drons estan realitzats únicament amb un controlador basat en Arduino i un sistema de ràdio control, açò limita les possibilitats ja que els microcontroladors Arduino només són capaços de gestionar els motors.

Al contrari, els nous System-On-a-Chip (SOC) de baix cost, com poden ser les Raspberry Pi, considerats com a microordinadors de butxaca, disposen d'una major versatilitat degut a la seua gran capacitat de processat, podent realitzar tant el control del dron en temps real des d'un dispositiu mòbil, com la gravació i enviament d'imatges.

No obstant això, la Raspberry no pot controlar els motors en temps real, per la qual cosa l'ús d'una controladora s'antulla indispensable. Pel que este projecte versa sobre el disseny i la construcció d'un cuadricopter controlat per mitjà d'una Raspberry Pi i una placa controladora Crius.

La Crius s'encarregarà del control dels motors, a partir dels comandos subministrats per la Raspberry, que a la seua vegada provindran d'un telèfon mòbil connectat per mitjà de Wi- Fi.

En este projecte també es dissenyarà l'aplicació mòbil en Android per al maneig del dron. Esta aplicació constarà de diverses activitats, entre les quals hi haurà una activitat de control, un mapa amb capacitat de geolocalització i una activitat que ens mostre el temps amb les dades del vent com a paràmetre important.

0.3. Abstract

In recent years, is taking a considerable increase in the use of drones, both in professional applications for the improvement and safety of many professional tasks, and recreational environments for personal enjoyment. A drone is commercially available with a specification and within an economic range consistent with the quality and performance or there is a possibility of building it yourself, from the wide variety of parts available on the market. This second aspect is the most interesting for the purpose of this work.

Today the majority of drones are performed solely based on Arduino and radio control system controller, this limits the possibilities since Arduino

microcontrollers are only able to manage the engine.

By contrast, the new System-On-a-Chip (SOC) low cost, such as the Raspberry Pi, considered microcomputers pocket offer greater versatility due to its high processing capacity and can perform both drone control in real time from a mobile device, such as recording and sending images.

However, Raspberry engines cannot control real-time, so the use of a controller seems indispensable. So this project is about the design and construction of a controlled Quadcopter by Raspberry Pi and Crius controller board.

The Crius be responsible for monitoring of engines, from the commands supplied by the Raspberry, which in turn come from a mobile phone connected via Wi-Fi.

In this project the Android mobile application for managing drone is also designed. This application will consist of several activities, among which there will be a control activity, a map with geolocation capability and an activity to show us the time with the wind data as an important parameter.

Índice general

Índice de figuras

Capítulo 1

Introducción

1.1. Contexto

En los últimos años el interés, tanto comercial como lúdico, del desarrollo de vehículos no tripulados (del inglés, UAV (Unmanned Aerial Vehicle)) ha sufrido un aumento muy considerable. Las aplicaciones de este tipo de vehículos son muy diversas, desde acceso a zonas con entornos de difícil acceso o que representen un cierto riesgo hasta seguir a una persona realizando deporte.
Dentro del marco de los UAV, podemos distinguir los mini-UAV, en los cuales está centrado este proyecto, los cuales son sistemas no tripulados de dimensiones reducidas. Este tipo de desarrollo ha sido posible gracias a la evolución que han sufrido recientemente el mundo de los microcontroladores, los cuales a día de hoy nos dan la posibilidad de realizar una serie de cálculos complejos en un espacio muy reducido y a un coste muy reducido. También cabe destacar la evolución que han sufrido los sensores y las baterías, los cuales han mejorado de manera exponencial su rendimiento en los últimos años.
Este proyecto versa en la construcción de un cuadricoptero pilotado desde un dispositivo móvil Android. El dron contará con una cámara que pueda captar imágenes o video, el cual será capaz de enviar al terminal móvil. Además sera capaz de realizar procesamiento de imágenes en el propio dron gracias a una Raspberry Pi, la cual también se encargará de recibir las ordenes del móvil y transmitirlas a la controladora que será la que maneje los motores. La idea de este proyecto es desarrollar una base la cual pueda ser ampliada y modificada para satisfacer las necesidades de cada usuario.
La motivación principal de este proyecto es que, a pesar de su complejidad y dificultad técnica, puede suponer un avance en este tipo de vehículos y añadir funcionalidades a los ya existentes. Además como proyecto final de grado de telecomunicaciones, abarca funcionalidades de las cuatro ramas del grado.

1.2. Objetivos del proyecto

El alcance de nuestro proyecto incluye todo el desarrollo necesario para tener el cuadricoptero volando y controlado mediante un dispositivo móvil Android. Consta de:

- Diseño y construcción de la estructura.
- Diseño de los algoritmos de control.
- Selección de la controladora.
- Integración de los motores.
- Integración de la Raspberry.
- Diseño de la comunicación.
- Diseño de la aplicación.
- Integración de la aplicación.

El desarrollo de los sistemas de control para este tipo de dispositivos no es trivial, debido a la dinámica tan compleja inherente a los sistemas aerodinámicos. Este tipo de sistemas son multivariables además de presentar características no lineales lo cual requeriría mucho más tiempo del que tenemos disponible por lo que no se diseñara este algoritmo desde cero si no que nos basaremos en uno ya hecho y de software libre. En un futuro la intención seria la de mejorar dicho algoritmo o incluso crear uno desde cero.

La aplicación si se diseñará desde cero, con características adicionales como puede ser la geolocalización o datos de la climatología actual.

1.3. Metodología

Con motivo del tiempo disponible para el desarrollo de este proyecto se intentará reutilizar todo el código posible, adaptado a nuestras necesidades, además de guiarse por proyectos que se pueden localizar por internet de código abierto, lo cual nos ayudará sustancialmente a la hora de escoger materiales.

Primeramente se realizará un exhaustivo análisis de mercado para ver las opciones que ya están hechas o en desarrollo y ver las diferentes opciones que tenemos disponibles. Ayudándonos esto a realizar una primera planificación la cual seguramente varíe a lo largo de la realización de este proyecto. Además de ayudarnos a elaborar un presupuesto en base a las piezas requeridas.

Posteriormente mientras llegan las piezas necesarias, procederemos a empezar el diseño de la aplicación Android, la cual iremos completando en función de nuestras necesidades y del tiempo.

Una vez recibidas las piezas, nos pondremos con el montaje y las modificaciones necesarias en los algoritmos de control para adaptarlos a nuestras necesidades. Además de proceder con la impresión del chasis con una impresora 3D.

Por ultimo, se implementaran todos los códigos necesarios y realizaran las pruebas necesarias para comprobar su correcto funcionamiento.

1.4. Análisis DAFO:

En el momento de planificar este proyecto se ha decidido realizar un análisis DAFO, debido a que es uno de los análisis más conocidos y sirve de ayuda a la hora de realizar la planificación y la toma de decisiones. Este tipo de análisis se basa en una matriz 2x2 donde cada una de las celdas se analizan los siguientes puntos para conocer su viabilidad presente y futura.

1.4.1. Debilidades

Este punto trata sobre los puntos débiles de nuestro proyecto:

- La falta de experiencia en un proyecto de este tipo, lo que ocasionará una necesidad mayor de tiempo en el análisis y estudio de todo lo relacionado con el mundo de los UAVs.
- La mayoría de proyectos existentes no poseen este método de control, y los que lo tienen son de carácter comercial por lo que sus esquemas y diseños no están disponibles.
- Conocimiento limitado del reglamento legal sobre el vuelo de UAVs, ya que está en constante cambio debido a su masificación en los últimos años.

1.4.2. Amenazas

En el momento de planificar este proyecto se pueden prever ciertos riesgos que pueden afectar a su correcto desarrollo:

- Una mala planificación inicial: Conlleva una gran dificultad aproximar la duración de las diversas tareas del proyecto debido a la falta de experiencia en la realización de estas, con lo que la planificación inicial puede variar durante el desarrollo del proyecto.
- Retraso en la recepción del material: Debido a la intención de realizar este proyecto comprando todas las piezas por separado y de código

abierto es posible que haya retrasos en la recepción por diversos motivos, como pueden ser falta de existencias, perdidas en los envíos, envíos demasiado lejanos..

- Problemas con el hardware: Debido a la complejidad de este proyecto podemos tener problemas de hardware debido a una mala implementación o por impactos no esperados.

1.4.3. Fortalezas

- Sistema de control innovador, a día de hoy no hay casi ningún proyecto de software libre que implemente un sistema de control mediante el móvil.
- Bajo coste de los materiales, una de las directrices de nuestro proyecto es trabajar con los productos de mejor calidad-precio.
- Motivación personal en este tipo de proyectos, y en aprender lo máximo posible de este tipo de proyectos.

1.4.4. Oportunidades

Al planificar este proyecto también nos gustaría analizar una serie de oportunidades, las cuales nos pueden influir positivamente, y podrían darnos ventaja en el futuro si las sabemos aprovechar.

- Posibilidad de crear un sistema que ahorre los costes de controles por radiofrecuencia. Lo que puede facilitar que mucha gente se anime a fabricar su propia aeronave.
- Aprendizaje de nuevos lenguajes de programación, además de entender el funcionamiento de estas aeronaves, que debido al auge que están teniendo, está creciendo el interés de grandes empresas en este tipo de proyectos.

Figura 1.1: Análisis DAFO

Capítulo 2

Estado del Arte

2.1. Introducción

En este capitulo se va a abordar la fase de estudio de mercado, la cual pone en perspectiva los productos que existen actualmente. Con esto podremos determinar si ya existen productos comerciales parecidos o si por el contrario es necesario desarrollar este proyecto desde cero.

Existen una cantidad destacable de proyectos que se basan en la construcción de un dron, independientemente del número de hélices. Según el enfoque que hagamos de estos proyectos, se pueden dividir en dos categorías:

Software libre vs privativo:

Antes de proseguir, cabe diferenciar estos términos para lo cual podemos usar una metáfora. Imaginemos la elaboración de una tarta. Los ingredientes de esta tarta si fuese industrial (Software privativo) no podrían ser revelados porque cualquier otra empresa podría imitar la misma receta y restarles beneficios. Por el contrario, cuando cualquiera de nosotros realiza la tarta, podemos aprender la receta de diferentes fuentes (Software libre)y no hay ningún inconveniente en compartir la receta.
En resumen, el software libre o abierto, es accesible por todo el mundo y pueden ser modificados. Por el contrario, el software privativo solo defiende los intereses de las empresas que proporcionan el software y este no puede ser modificado para responder a necesidades particulares.
Cabe destacar que existen una multitud de comunidades que desarrollan software de código abierto orientadas a determinadas plataformas. Sin embargo no todas publican sus códigos de manera gratuita. Uno de los mejores ejemplos de hardware de código abierto es Arduino [6].

2.2. Análisis de los productos del mercado

La multitud de proyectos sobre los drones actualmente es enorme, con lo que nos basaremos en los mas parecidos a nuestro proyecto, o los que al menos realicen unas funciones similares. Escogeremos tres proyectos, uno de software libre y dos de software privativo.

2.2.1. Phantom 3 Standard

El cuadricoptero Phantom 3 es un producto de la empresa DJI [9]. Dentro de este modelo podemos encontrar varias gamas, desde la standard, que es la mas básica, hasta una profesional y avanzada en las cuales lo que varía es la calidad de la cámara. En este trabajo vamos a mencionar la versión estándar, la cual se puede observar en la Figura 2.1, debido a que es la que más se puede asemejar a nuestro proyecto.

Figura 2.1: Phantom 3 Standard

Este modelo en concreto dispone de una cámara capaz de grabar video a 2.7K a una velocidad de 30 fotogramas por minuto. También es capaz de realizar fotografías de 12 Megapixeles. Consta de un GPS integrado capaz de memorizar su posición inicial a la cual vuelve solo presionando un botón. Es capaz de volar hasta 1 km de distancia con una durabilidad aproximada de vuelo de unos 25 minutos. Se puede controlar mediante un mando de control remoto y usar una aplicación móvil disponible tanto en IOS como en Android para visualizar lo que graba la cámara. Sin embargo con esta aplicación no podríamos controlar el vuelo.

Su precio a fecha 1 de Junio de 2016 es de 599$.

2.2.2. Parrot Ar Drone

El Parrot Ar Drone [5] es un proyecto con cierta antigüedad, pero merece la pena nombrarlo y analizarlo debido a lo que supuso para el uso y la

construcción de drones ya que fue el primero que se pudo controlar mediante dispositivos móviles. Este proyecto fue presentado en el año 2010. Este cuadricoptero, el cual podemos observar en la Figura 2.2, consta de una carcasa de plástico de aproximadamente 30 centímetros de largo

Figura 2.2: Parrot Ar Drone 2.0

Este modelo dispone de dos microcámaras que permiten grabar video a 720p. Se puede controlar mediante una aplicación, la cual está disponible tanto para iOS como Android, mediante una conexión Wi-Fi.
El software que usa este cuadricoptero corre sobre un sistema operativo Linux. Aunque es un producto completamente comercial y privativo, es decir, mantienen el código como propietario, si que dispone de una API con la que nos permite modificar, dentro de unos limites, el comportamiento de dicho dron. Lo más destacable de este modelo es que es capaz de reconocer objetos 3D y es compatible con juegos de realidad aumentada que tanto están en auge hoy en día. Sus especificaciones técnicas constan de un microcontrolador ARM9 a 468Mhz con 128 Megabytes de RAM. Dispone de conexión Wi-Fi y USB. Además tiene un acelerómetro de 3 ejes, dos giroscopios y un altímetro ultrasónico. Posee 4 motores Brushless de 15 W y una batería recargable de 1Ah de litio capaz de proporcionar 11.1 voltios nominales. Su peso ronda los 400g y es capaz de volar durante 11 minutos a unos 18 km/h.

Su precio a fecha de 1 de junio de 2016 es de 249$

2.2.3. AeroQuad

En este caso, no nos vamos a centrar en un modelo en concreto, si no que vamos a hablar del proyecto AeroQuad [3]. Este proyecto no abarca un solo modelo sino que al ser software libre da unas bases de los componentes hardware y software, lo que permite que haya muchas versiones, dependiendo de

lo que busque la persona que lo quiera realizar. Los modelos constan de una placa controladora de vuelo y un chasis AeroQuad con diversos sensores, como acelerómetro, giroscopio, magnetómetro, barómetro, sensores ultrasónicos y GPS. La configuración de este tipo de drones puede variar múltiples características, como por ejemplo puede ser el chasis, que puede ser desde tres ejes hasta ocho. No hay un precio fijo para este tipo de drones debido a que depende esencialmente de las piezas utilizadas y varían en función de las necesidades de cada uno.

En la Figura 2.3 podemos observar uno de los últimos modelos denominado AeroQuad32 [14].

Figura 2.3: AeroQuad32

2.3. Conclusión

Actualmente con el uso masivo de smartphones se podría reducir costes debido al ahorro de los dispositivos de control remoto, ya que gracias a los dispositivos móviles sería innecesario el uso de otros sistemas de radiocontrol. Aunque, como se ha mencionado antes, existen modelos de cuadricopteros controlados por dispositivos móviles, como puede ser el Parrot AR Drone, no se ofrecen ni esquemas de hardware ni los códigos empleados debido a que es software privativo. Esto nos obliga a analizar y ver cual es la manera idónea de llevar a cabo este proyecto. En definitiva, como en el mercado no hay disponibles productos de código abierto que tengan características similares a nuestro proyecto, se desarrollara desde cero basándonos en proyectos de código abierto y tratando de minimizar los costes los máximo posible.

Capítulo 3

Estructura general del dron

3.1. Antecedentes

En este capítulo se van a plantear las diferentes alternativas que existen para el desarrollo de este tipo de proyectos, y el motivo por el cual nos hemos decidido por cada una de las opciones. Este proyecto se puede dividir en dos grandes bloques, el Drone o UAV y la aplicación móvil.

Cada uno de los bloques puede estar formado por diversos componentes. El primer componente que debemos elegir es la controladora de vuelo, partiendo de un desarrollo bottom-up, la cual nos influirá en el resto de componentes.

Antes de decidir que controladora usaremos, vamos a analizar algunas de las alternativas mas usadas de los controladores actuales. En los cuales valoraremos, si es privativo o de código abierto, lo cual en un principio puede parecer más ético que funcional, pero en realidad se trata de un factor práctico, ya que los controladores de código abierto se adaptan mejor a este tipo de proyectos debido a que nos permiten, en la mayoría de casos, modificar el software a nuestro gusto y necesidad. Además la documentación de este tipo de códigos es mucho mas extensa y la información es mas fácil de localizar. El protocolo de comunicaciones también es un factor muy importante, ya que nos permitirá interactuar con los demás componentes. Otra de las características a tener muy en cuenta será su coste, el cual trataremos de minimizar en la medida de lo posible.

Una vez analizados los distintos sistemas de control de vuelo, procederemos a enumerar las características que necesitamos en la aeronave y las diferentes alternativas que podemos encontrar en el mercado actual.

El peso máximo del dron será de aproximadamente unos 1,5kg. Se intentara que el peso del propio dron no pase de 1kg en total. Por lo que si nuestro dron tiene 4 motores, cada uno debe tener una fuerza de empuje de aproximadamente 250gr. Si estudiamos las características de cada motor, podemos observar que su empuje no siempre es igual, y que depende del

voltaje de las baterías y del tamaño de las hélices. A mayor voltaje y mayor tamaño de las hélices, mayor fuerza de empuje tendrá.

3.2. Componentes Drone

En la figura 3.1 podemos observar un diagrama de bloques de los componentes que son necesarios para este proyecto, los cuales detallaremos la continuación:

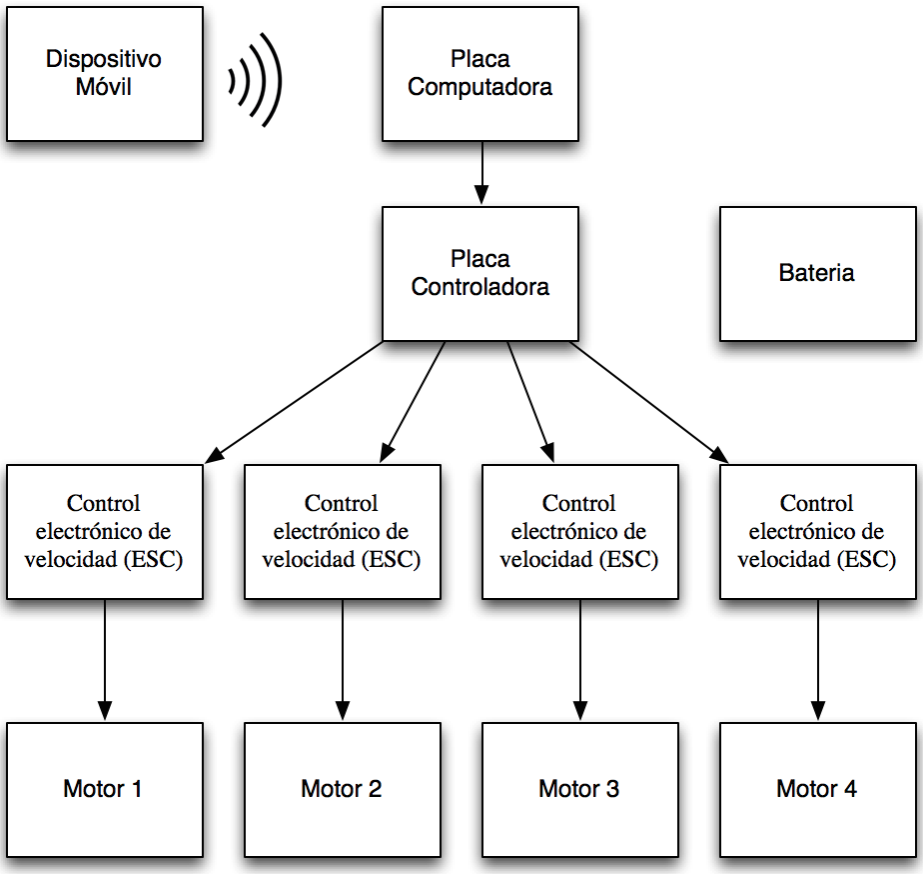

Figura 3.1: Diagrama de bloques de los componentes utilizados

Nótese que la batería no tiene ninguna conexión porque alimenta a todo el sistema, hay diferentes formas de conectarla por lo que en el diagrama simplemente se pone el bloque para que se sobreentienda que alimenta a todo el sistema, independientemente de las conexiones necesarias.

3.2.1. Controladora de vuelo

En general, la mayoría suelen tener un diseño muy parecido, un estructura muy similar, y unos componentes de mayor o menor calidad. Suelen contar con:

- Acelerómetro para poder medir los diferentes movimientos.
- Giróscopio para poder medir la velocidad angular de los cambios de posición.
- Magnetómetro utilizado como una brújula que permite saber la dirección a la que apunta el Dron
- Sensor barométrico empleado para conocer con una gran precisión la altura real del vuelo.
- GPS para conocer las coordenadas exactas del dron y poder desplazarse de forma autónoma.
- Procesador que sea capaz de realizar las lecturas y las operaciones por segundo necesarias en base a todos los datos que recibe.

Con la combinación de todos o alguno de los componentes que acabamos de mencionar, se consigue suficiente información del medio para poder tomar las decisiones apropiadas sobre los actuadores que harán posible el vuelo.

En el mercado actual existen multitud de controladores de vuelo, en base a nuestros requisitos funcionales y de coste, nos centraremos en las siguientes que podemos observar en la Figura 3.2 :

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

Figura 3.2: Tabla comparativa de controladoras

- **ArduPilot**
  Esta controladora pertenece a una plataforma de código abierto para los UAV´s, es capaz de controlar una gran variedad de vehículos aéreos no tripulados, tanto de forma autónoma como de ala fija, o helicópteros tradicionales o incluso vehículos de tierra. Esta plataforma fue

creada en 2007 por 3DRobotics [1]. Esta plataforma esta basada en código abierto de prototipo electrónico Arduino [6]. La primera versión de esta controladora se basó en un termo pila, el cual 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. Poco a poco el sistema ha ido mejorando para reemplazar una combinación de acelerómetros, giroscopios y magnetómetros. Todo bajo licencia GPL v3 [21]. Esta plataforma fue la vencedora en los años 2012 y 2014 de la competición UAV Outback Challenge.

Actualmente, la plataforma ha evolucionado a una gama de productos hardware y software con mucha diversidad. En cuanto al hardware, podemos destacar:

- APM es una controladora de 8 bits, basada en un chip Atmel ATMEGA2560 [7]
- Pixhawk/PX es una controladora de 32 bits, basada en el chip STM32F427 Cortex M4 core con FPU [18].

Y en cuanto a los productos software:

- 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 el sistema de control de la estación terrena para todo tipo de vehículos 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.

  Esta es una de las plataformas mas usadas en la actualidad, gracias a su popularidad, podemos encontrar copias en el mercado chino, lo que nos permite abaratar el coste de forma sustancial. Una de las replicas más usadas y que mejor relación rendimiento/coste ofrecen es la réplica china de la placa APM, llamada Crius All In One Pro v2.1 [10] la cual podemos observar en la Figura 3.3

- **OpenPilot**
  Este proyecto también se basa en el software libre para vehículos aéreos no tripulados, desde multicopteros, aviones de ala fija o vehículos de tierra. Este proyecto se fundó en el año 2009 por David Ankers, Angus Peart y Vassilis Varveropoulos. Fue concebido como una herramienta

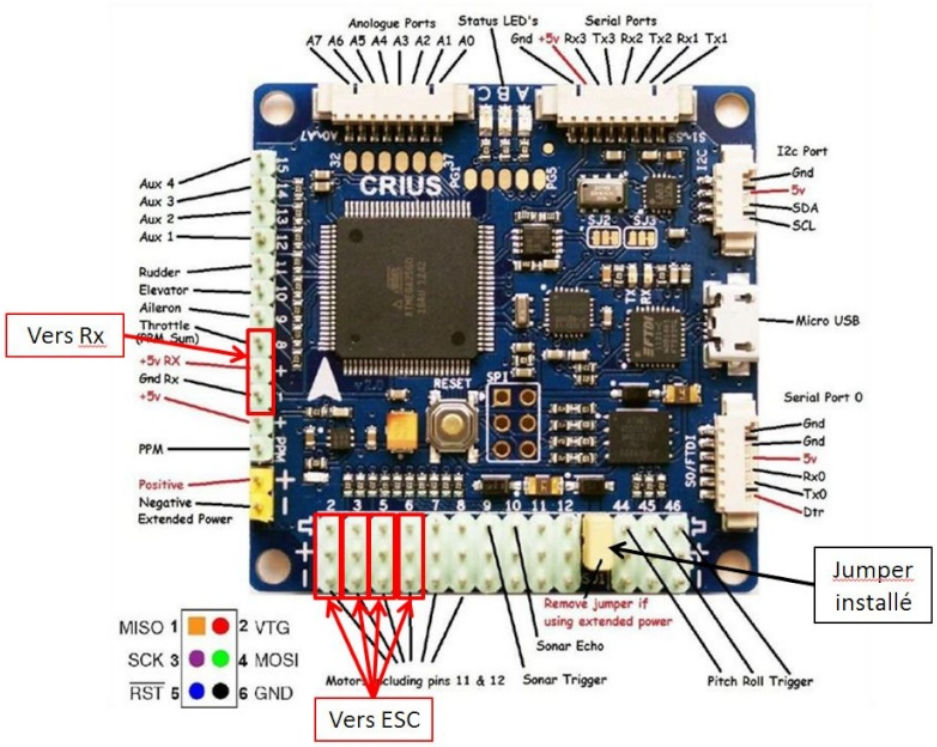

Figura 3.3: Crius All in One Pro v2.1

de aprendizaje para hacer frente a distintas áreas percibidas en otras plataformas de los UAV.

El software de esta plataforma,de código abierto, con piloto automático puede combinarse con diferente hardware, como puede ser un sistema de navegación inercial, una tarjeta controladora, un receptor GPS, un sistema de telemetría o un enlace de 2,4GHz sería para la comunicación con la estación base. Se distribuye bajo licencia GPL v3.

La parte software de este proyecto consta de dos códigos, el firmware implementado en el dron y el firmware de la estación base. Están escritos en C y en C++ usando Qt respectivamente.

En la parte hardware, existen dos tipos, el Revolution y el CC3D. El primero de ellos permite vuelo automático, además de incorporar un módem UHF. El segundo cuenta con 6 giroscopios que permiten saber en todo momento su posición. Este tiene un coste mucho más reducido que el primero. Aunque en esencia, este tipo de controladoras disponen del mismo firmware de estabilización.

- Procesador STM32 de 32 bits.
- Giroscopios y acelerómetros de 3 ejes.
- Permite varias configuraciones, Tri, Quad y Hexa.
- USB de alta velocidad sin necesidad de controladores específicos.
- Soporte para Windows, Mac y Linux

- **Acro Naze32/Flip32**
  Este proyecto esta orientado a multirotores aéreos de pequeño y mediano tamaño, tanto para exteriores como para interiores. También como un estabilizador independiente de una cámara. Su configuración es en cierto modo simple, esta configuración esta basada en la familia "Multiwii", la cual veremos más adelante. Las características a destacar de esta controladora son:

  - Procesador ARM de 32 bits.
  - Giroscopios, acelerómetros, brújulas y sensores de presion.
  - Varias configuraciones, Quad, Tri, Hex o Octo.
  - Monitorización de la batería.
  - Convertidor de telemetría FrSky.
  - GPS con bloqueo de posición y vuelta a casa.
  - USB para configurar y controlar.

- **Brain FPV**
  Esta controladora forma parte de las denominadas 'de ultima generación' ya que esta diseñada específicamente para la vista en primera persona (FPV) del vuelo e incluye una visualización en pantalla (OSD). La controladora en si tienen un tamaño y un peso reducido, por lo que es ideal para los mini-UAV´s, aunque también es capaz de manejar grandes opto-copteros y aviones de ala fija.

  En cuanto al software que utiliza esta controladora es de código abierto, como en los casos anteriores, bajo una licencia GPL v3. Está basada en Tau Labs, lo que ofrece un variedad de opciones y características destacables. Esta plataforma también esta disponible en los tres grandes sistemas operativos de hoy en día, como son Windows, Mac y Linux. Usa el software GCS, el cual nos ayuda a configurar la controladora mediante USB.

  Lo destacable de esta plataforma es que, al ser código abierto, ofrece el mayor grado de libertad posible, con lo que se puede realizar cualquier modificación posible en el código para adaptarla a las necesidades de cada uno. Las características a destacar de esta controladora son:

- Procesador Stm32 F4 de 32 bits.
- Giroscopios y acelerómetros de Bosch a 1,6KHz usando SPI, Barómetro, brújulas.
- OSD con autodetección de PAL/NTSC
- Soporte de telemetría para FrSKY
- Emisor de infrarrojos
- Puerto microUSB

- **KK Multicontroller**

  Esta placa controladora para control remoto de multicopter es capaz de soportar multicopteros de 2, 3,4 y 6 rotores. Su propósito es la de estabilizar el UAV durante el vuelo. para ello usa la señal de 3 giroscopios, los cuales están integrados en la placa (equilibrio, cabeceo y estabilidad). Toda la información esta controlada por un circuito integrado de ATMega, el cual procesa toda la información y envía señales de control a los controladores de velocidad electrónicos (ESC), los cuales van conectados a la placa y a los motores. En función de esta señal de control, el ESC, aumenta o reduce el voltaje suministrado, lo que equivale a un aumento o disminución de la velocidad de rotación de dichos motores con el fin de establecer el nivel de vuelo. Esta señal de control varía en función de la señal recibida por el mando radio control. Las características a destacar de esta controladora son:

  - Microcontrolador ATMega 168 de 8 bits
  - Giroscopios y acelerómetros piezoeléctricos
  - No dispone de USB, solo de ISP
  - Dispone de 6 cabezales ESC/servo

- **Multiwii**

  En este caso, este proyecto no es una controladora en si, si no que se basa en un software de propósito general que es capaz de controlar un multirotor por control remoto. Actualmente es capaz de usar una gran variedad de sensores, pero inicialmente fue desarrollado para usarse con los sensores que lleva el mando de la consola Nintendo WII,concretamente en los mandos WiiNote: Wii Motion Plus y Wii Nunchuck. Ya que estos mandos contienen giroscopios y acelerómetros. Este proyecto está desarrollado sobre la plataforma Arduino, gracias a este software la estabilidad que se puede conseguir es realmente buena para cualquier UAV e incluso es capaz de permitir realizar acrobacias. Este software, por el momento, es capaz de controlar 3, 4 o 6 rotores.

  Las características de este controlador son muy parecidas a las vistas anteriormente en APM, de hecho, esta basada en el mismo controlador,

un Atmel ATMega 2560 de 8 bits. Aunque esta no es capaz de soportar puntos de control.

3.2.2. Control electrónico de velocidad (ESC´s)

Los modelos controlados remotamente, bien sea por radio o mediante otro método de comunicación funcionan con pequeños motores electrónicos. Estos motores deben utilizar un dispositivo que controle el nivel de voltaje que le llegue al motor, para así poder controlar su velocidad, tanto para aumentarla como para disminuirla. Estos dispositivos se denominan ESC, es decir, Control Electrónico de Velocidad.

Los ESC constan de un circuito electrónico que esta controlado por un microcontrolador, habitualmente un Atmel del tipo AVG. Dentro de los ESC's podemos distinguir dos tipos, para controlar los motores con escobillas (brushed), y los motores sin escobillas (brushless). Cada tipo de ESC solo se puede usar con el tipo de motor correspondiente. En este proyecto solo nos centraremos en los motores sin escobillas, los cuales veremos un poco mas adelante.

El funcionamiento de estos dispositivos se basa en un pulso de energía que puede enviar al motor. Este pulso funciona en base a una frecuencia, la cual podemos encontrar en las especificaciones de cada ESC, que generalmente va desde los 2KHz hasta los 12Khz. Por lo tanto estos dispositivos no funcionan como una resistencia variable que ajusta el voltaje entregado al motor, sino que es un interruptor ultra rápido que enciende y apaga la energía entregada al motor, es decir, cuando el ESC está 'encendido' este entrega casi todo el amperaje requerido y todo el voltaje disponible en nuestras baterías. Por lo que, cuando nosotros mandemos una señal de cambio de velocidad, lo que hará el ESC será modificar la frecuencia de los ciclos de encendido y apagado, aumentando o disminuyendo así las revoluciones, o lo que es lo mismo, gana o pierde velocidad.

Para escoger el ESC adecuado, hay que tener en cuenta que el tamaño viene dado por unos cuantos amperios que puede controlar y cuantos voltios puede soportar. También hay que tener muy en cuenta el motor-hélices y batería que vayas a usar.

En la Figura 3.4 podemos observar el modelo escogido para nuestro proyecto. Se trata de un ESC de la marca Hobby King, el cual está diseñado para motores sin escobillas y nos proporciona una limitación de corriente de 30 A.

3.2.3. Motores

Como se ha mencionado antes, podemos distinguir dos tipos de motores eléctricos para los UAV, con escobillas y sin ellas. Antiguamente, los motores

Figura 3.4: Hobby King 30A ESC 3A UBEC

solían tener un colector de delgas o un par de anillos rozantes. Estos sistemas, los cuales producen rozamiento, disminuyen de manera significativa el rendimiento, desprenden calor y ruido, requieren de un elevado mantenimiento y producen partículas de carbón las cuales manchan el motor de un polvo que, además puede ser conductor. Estos eran los denominados motores eléctricos con escobillas.

Los primeros motores sin escobillas fueron los motores de corriente alterna asíncronos. Hoy en día, gracias en su mayor parte al avance de la electrónica, se muestran muy ventajosos ya que son más baratos de fabricar, pesan considerablemente menos (característica notable en este tipo de proyectos) y requieren mucho menos mantenimiento. El inconveniente de este tipo de motores es su control, el cual se vuelve mucho más complejo. Afortunadamente, gracias al avance de la electrónica, esta complejidad se ha ido reduciendo drásticamente con los controles electrónicos.

El funcionamiento se basa en que el inversor debe convertir la corriente alterna en corriente continua, y otra vez en corriente alterna pero con otra frecuencia. Otras veces, se puede alimentar directamente con corriente continua, de esta manera se elimina el primer paso. Debido a esto, estos motores de corriente alterna se pueden usar en aplicaciones de corriente continua, con un rendimiento muy superior a un motor de corriente continua con escobillas.

Otros motores sin escobillas que se usan, son los que funcionan con corriente continua para los pequeños aparatos electrónicos de baja potencia como pueden ser, los lectores de CD-ROM o los ventiladores de un ordenador. Su mecanismo radica en sustituir la conmutación (cambio de polaridad) mecánica por otra electrónica sin contacto. Con lo que la espira solo se impulsa cuando el polo es el correcto, y en caso contrario se corta el suministro de corriente. Para detectar la posición de la espira del rotor se utiliza la detección mediante un campo magnético. Gracias a todo esto, este tipo de sistemas pueden informar de la velocidad de giro, si esta en reposo o incluso cortar la corriente como mecanismo de protección. Como desventaja, es que este tipo de motores no cambia el sentido de giro según su polaridad, para

cambiarlo se deberían cruzar dos conductores del sistema electrónico.

La velocidad en este tipo de motores 'brushless' viene definida por un parámetro, denominado kV. Este indica el número aproximado de revoluciones del motor por cada voltio de electricidad aplicado. Es decir, si cogemos un motor de 1100kV girará aproximadamente a 11000 revoluciones si le aplicas 10V.

Para nuestro proyecto hemos escogido los motores que podemos ver en la Figura 3.5 el cual tiene las especificaciones que se pueden ver en la Figura 3.6. La elección de estos motores se debe a que tienen una excelente relación calidad-precio, además de que tienen unas prestaciones óptimas para nuestro proyecto y nuestros requisitos, de los cuales ya hablamos antes, de peso que puedan levantar. Además de que son unos motores muy ligeros. Estos motores tienen un amperaje máximo de 22A lo cual nos vale para el ESC escogido, del cual hablamos en el punto anterior.

Figura 3.5: Turnigy D3530/14 1100KV Brushless Outrunner Motor

3.2.4. Hélices

Las hélices son un dispositivo mecánico formado por un conjunto de elementos denominados palas o álabes, están montados de forma concéntrica y solidarias de un eje que, al girar, realizan un movimiento rotativo en un plano. Estas palas no son placas planas, sino que tienen una forma curva, de modo que sobresalen del plano en el que giran, de modo que obtienen una diferencia de distancias entre el inicio y fin de la pala. Esto provoca 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. A esta fuerza se le denomina fuera propulsora de una aeronave.

Kv(rpm/v)	1100
Weight (g)	73
Max Current(A)	22
Resistance(mh)	0
Max Voltage(V)	15
Power(W)	315
Shaft A (mm)	5
Length B (mm)	30
Diameter C (mm)	35
Can Length (mm)	12
Total Length E (mm)	46

Figura 3.6: Especificaciones Turnigy D3530/14 1100KV Brushless Outrunner Motor

Antiguamente, las primeras hélices que se usaron, fueron las de los molinos de viento y agua. Hoy en día, también bajo los nombres de rotor,turbina o ventilador, se emplean para una multitud de propósitos, como pueden ser la refrigeración, la compresión de fluidos, propulsión de vehículos o generación de electricidad.

El inventor de la primera hélice operativa fue Josef Ressel en 1826, para propulsar un buque. Posteriormente, se han mejorado sustancialmente tanto el rendimiento como el consumo de las hélices.

Como se puede observar existen una gran variedad de denominaciones y campos de aplicación, por lo que también existe una gran variedad de hélices, en tamaño, peso, numero de palas, velocidades de rotación, etc.

Para nuestro proyecto nos vamos a centrar en un modelo concreto, el cual podemos observar en la Figura 3.7, tienen como medidas 10x4,5cm y una rotación como podemos observar en la Figura 3.8.

Figura 3.7: Hobbyking Slowfly Propeller 10x4.5 Blue

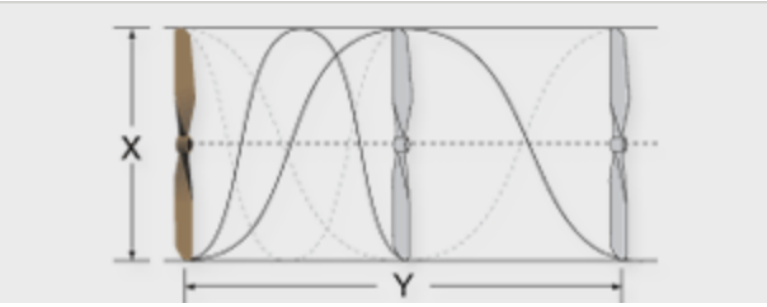

Figura 3.8: Explicación de la rotación de las hélices

Este tipo de hélices, tienen la particularidad de que no sirve colocarlas de cualquier forma ya que de ello depende como expulsan el flujo de aire y por tanto que el vuelo sea correcto e incluso que despegue o no. Para colocarlas correctamente primero debemos distinguir dos tipos de hélices, las cuales podemos observar en la Figura 3.9, nótese la torsión de la hélice en cada modelo presentado. Una vez tenemos ambos tipos de hélices, deben quedar dispuestas en el drone de la forma que se observa en la Figura 3.10, es decir, en forma de cruz, la frontal izquierda y la trasera derecha deben ser iguales, por ejemplo, modelo A. La frontal derecha y la trasera izquierda deben ser el modelo B.

Figura 3.9: Modelos de hélices

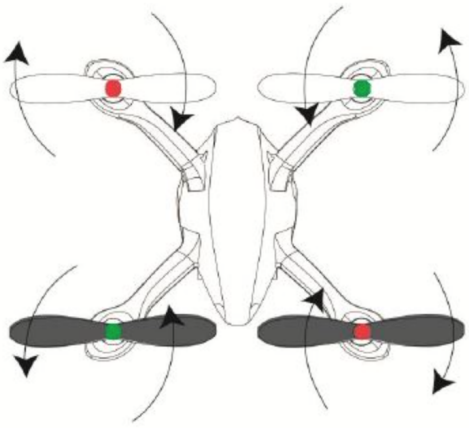

Figura 3.10: Disposición de las hélices

3.2.5. Baterias

El tipo de batería a utilizar es muy importante, de ello depende la autonomía del dron y su peso. Según el material del que están hechas, las baterías pueden ser, de litio y de níquel para este tipo de aeronaves. Para empezar vamos a identificar los tipos de baterías que existen, sus respectivos nombres, abreviaturas y especialmente las utilizadas en este tipo de proyectos. Básicamente existen seis tipos de baterías:

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

Niquel Cadmiun (NiCd).
Este tipo de baterías fueron inventadas en 1899 por Waldmar Jungner. En la actualidad son las más utilizadas y se podría afirmar que están en mas de un 70 % de los equipos a nivel mundial. Las baterías de NiCd son utilizadas donde se requiere una gran durabilidad, alta capacidad de proporcionar energía y un coste reducido. Como por ejemplo, en radios transmisores, equipos medicos, cámaras de video profesionales, etc. El inconveniente más claro de este tipo de baterías radica en que contienen materiales tóxicos y son nocivas para el medioambiente. En resumen, podemos ver una serie de ventajas y desventajas:

- *Ventajas:*
  - Capacidad de carga rápida o lenta.
  - Gran número de ciclos de carga, aproximadamente 1000 con un buen mantenimiento.
  - Buena eficiencia y amplio rango de temperaturas.
  - Facilidad de transporte y almacenaje.
  - Baja temperatura de trabajo.
  - Disponibilidad en gran variedad de tamaño y coste.
- *Inconvenientes:*
  - Baja densidad de energía en comparación con otras baterías
  - Deben ser periódicamente recicladas a causa del efecto memoria[1]
  - Contienen materiales tóxicos, lo cual limita su uso en ciertas areas.
  - Consumo moderado de energía durante el almacenaje, lo que provoca recargarla después.

Polímero de litio

Este tipo de baterías se diferencia de otras por el tipo de electrolito usado. El diseño original data de 1970 en la cual se utilizaba un electrolito de Polímero solido seco. Este electrolito fue reinstalado como una lamina de plástico que no tiene la propiedad de conducir la electricidad pero si tiene la propiedad de intercambiar iones (grupo 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. Con este tipo de baterías no existe peligro de incendio ya que no contiene electrolitos en forma de liquido o gel. Gracias al espesor de la celda, la cual mide como un milímetro, es posible crear formas de baterías que conformen parte del encapsulado del dispositivo o prácticamente cualquier configuración conveniente que se adapte a las necesidades.

[1]Es un problema que afecta directamente a este tipo de baterías, este tipo usa un material activo denominado cadmio. Este material esta dividido en pequeños cristales que tienen una longitud de 1 micrón, esta pequeña medida otorga una mayor cantidad de ellos que tienen contacto directo con el electrodo proporcionando así mayor eficiencia o energía. Por lo tanto, cuando el efecto memoria ocurre, estos cristales aumentan considerablemente su tamaño implicando que una menor cantidad de ellos tengan contacto directo con el electrodo. Esto sucede porque solo una porción de los materiales activos de la batería se descargan y se recargan en un determinado tiempo

Lamentablemente, al usar el polímero solido seco, este tipo de baterías sufren de mala conductividad a temperatura ambiente, debido a que la resistencia interna es demasiado alta y no entrega la corriente necesaria que requieren por ejemplo los equipos de comunicación. Si conseguimos aumentar la temperatura de la celda por encima de los 60º se incrementa hasta niveles aceptables la conductividad.

Actualmente, algunas baterías de Li-Polymer no trabajan bien en climas cálidos. Un fabricante agregó elementos químicos (gel electrolítico) con el cual la batería permanece con un alto grado de conductividad en todos los rangos de temperatura. Este tipo de batería es muy usado actualmente en los teléfonos móviles.

- *Ventajas*
  - Variedad de tamaños.
  - Muy ligeras.
  - Mayor resistencias a las sobrecargas.
  - Menos propensas a que se rompa el electrolito.
- *Inconvenientes*
  - Nivel de energía capaz de suministrar por unidad de tiempo.
  - Menor tiempo de vida en comparación con las baterías de Li-on
  - Alto coste de fabricación

Baterias de litio usadas en UAV's

Desde que se empezó a desarrollar este tipo de vehículos hemos podido observar un avance tecnológico bastante importante. Por ejemplo, en las baterías de NiCd, al principio poseían celdas de tamaño AA con una capacidad de 500mAh, mientras que actualmente pueden llegar a los 2000mAh. Los paquetes de baterías de NiCd estaban típicamente formadas por 4 celdas al igual que las baterías de Lithium pero con la gran diferencia respecto al volumen como al peso de la batería. En la siguiente Figura 3.11 podemos observar una comparativa entre ambas baterías. La diferencia es notable entre ambas. Esta diferencia tan grande a implicado un avance tecnológico en los modelos eléctricos.

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

Figura 3.11: Comparativa entre los tipos de baterías para los UAV

3.2.6. Estructura

Antes de analizar la estructura, hay que tener una idea de los materiales que vamos a poder usar y así poder diseñar una estructura siendo conscientes de las características o propiedades de los materiales disponibles. Por lo que se ha realizado una búsqueda intensiva de materiales para poder construir el dron, centrándose en lo que nos proporcionara una alto ratio de resistencia, peso y coste. Los principales materiales encontrados para este fin son:

- **Aluminio:**
  Es un metal no ferromagnético, el cual tiene unas propiedades mecánicas que lo hacen muy útil en la ingeniería, como una baja densidad ($2,7 g/cm^3$) y una alta resistencia mecánica, según que aleación disponga el aluminio se puede llegar hasta los 690 MPa. Además es un material de coste reducido y se mecaniza con facilidad. En el mercado se puede encontrar en una variedad de formas y tipos como pueden ser planchas, barras, varillas, etc.

  Por otro lado, es resistente a la corrosión, es un buen conductor de la electricidad y el calor, aunque estas últimas propiedades no se aprovechen en este proyecto.

- **Fibra de carbono:**
  Es una fibra constituida por finos filamentos de 5-10 μm de diámetro compuesto principalmente por carbono. Cada filamento de carbono es la unión de miles de fibras de carbono. Tiene propiedades similares al acero, su resistencia longitudinal está sobre los 1100 MPa, pero la transversal solo esta sobre los 50MPa y además tienen una baja densidad ($1,6 g/cm^3$). Pero su principal inconveniente es su elevado coste.

  Por otro lado, es un gran aislante térmico, conduce la electricidad y es resistente a los cambios de temperatura y a los agentes externos.

- **Poliestireno Extrusionado:**
  Es una espuma rígida que resulta de la extrusión del Poliestireno en presencia de un gas espumante. Su principal propiedad es su bajísima densidad de $0,33 g/cm^3$ pero también su baja resistencia (250KPa), es un gran aislante térmico. Su principal uso, aparte de aislante, es para choques utilizando un grosor elevado ya que es capaz de absorber grandes golpes añadiendo muy poco peso.

- **Fibra de vidrio:**
  Es un material que consta de numerosas y extremadamente finas fibras de vidrio. Tiene como propiedades un densidad alta ($2,58 g/cm^3$) y una resistencia similar a la fibra de carbono, siendo su resistencia longitudinal de 1080MPa y la transversal de 50MPa. Aunque este material es mucho más económico que la fibra de carbono.

- **PLA:**
  Es un plástico biodegradable derivado de recursos renovables como el almidón de maíz o la caña de azúcar. Sus principales propiedades son una densidad baja ($1,25 g/cm^3$) y una resistencia mecánica de 65 MPa. Se usa principalmente en las impresoras 3D, en las cuales permite crear piezas altamente complejas usando un software de CAD como puede ser Solidworks. Además, es un buen aislante del calor y la electricidad. Aunque las propiedades de este material no sean las que más destacan, si que ofrece una gran relación calidad-precio, además, se dispone de una impresora 3D para la realización de la estructura, por lo que la mayor parte del coste esta solventada. Por ese motivo, este material es el elegido para este proyecto.

3.2.7. Placa computadora

Necesitamos una placa capaz de hacer de servidor para recibir mensajes desde un dispositivo móvil y después reenviarlos a la placa controladora. Además, en lineas futuras nos aprovecharemos de la capacidad computacional de esta placa para poder realizar procesado de video en tiempo real. Existe una gran variedad de este tipo de placas de bajo coste, analizaremos algunas de ellas para decidir cual es la que mejor se adapta a nuestro proyecto.

- **Raspberry Pi:**
  La Raspberry Pi es una placa computadora (SBC) de bajo coste, se desarrollo en el Reino Unido por la fundación Raspberry Pi, con el objetivo de estimular la enseñanza de las ciencias de la computación en las escuelas.

  En realidad se trata de una diminuta placa base de 85 x 54 mm(todos los modelos, salvo la Zero). Para que funcione, necesitamos de un medio de almacenamiento (SD o microSD en función del modelo) y conectarlo a una fuente de alimentación. En función del modelo que escojamos, los cuales analizaremos a continuación, dispondremos de más o menos opciones de conexión, aunque siempre dispondremos al menos de un puerto de salida de video HDMI y otro de tipo RCA, minijack de audio y un puerto USB 2.0 al que poder conectar un teclado y un ratón.

  A continuación vamos a analizar los diferentes modelos que nos pueden ser útiles en este proyecto:

  - **Raspberry Pi Model A+:**
    Este modelo fue lanzando al mercado en Noviembre del 2014, es la evolución del modelo A. Con este diseño han conseguido reducir el peso al eliminar el puerto Ethernet y tener solo un puerto

USB, gracias a esto también han reducido considerablemente el consumo (alrededor de un 30 %) que el modelo B+. Como inconveniente es que cuenta solo con 256MB de memoria RAM, la mitad que el modelo B+. También cuenta con un puerto microSD y un GPIO de 40 pines.

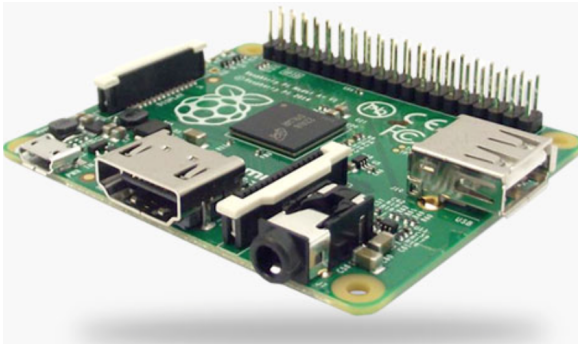

Figura 3.12: RASPBERRY PI MODEL A+

- **Raspberry Pi 2 Model B**
  Este modelo llegó al mercado en Febrero de 2015 y ha sido la Raspberry Pi más utilizada hasta la fecha. Sus prestaciones aumentaron considerablemente ya que disponíamos de una placa con un 1GB de memoria RAM, un procesador de la familia ARMv7, el cual es 6 veces más rápido que su antecesor, y además compatible con Windows 10 y Ubuntu.

  Esta placa ofrece una excelente flexibilidad de aprendizaje y es una opción a tener muy en cuenta para posibles proyectos. Además es totalmente compatible con los dispositivos de la Raspberry Pi 1. Esta placa lleva un procesador ARM Cortex-A7 de cuatro núcleos y 1GB de memoria RAM, una GPU VideoCore IV con soporte OpenGL y aceleración por hardware OpenVG capaz de reproducir a 1080p 30 frames por segundo. Además incorpora 4 puertos USB, HDMI, puerto Ethernet, jack de 3,5mm y un GPIO de 40 pines. La fuente de alimentación necesaria es de 5v 1A como mínimo.

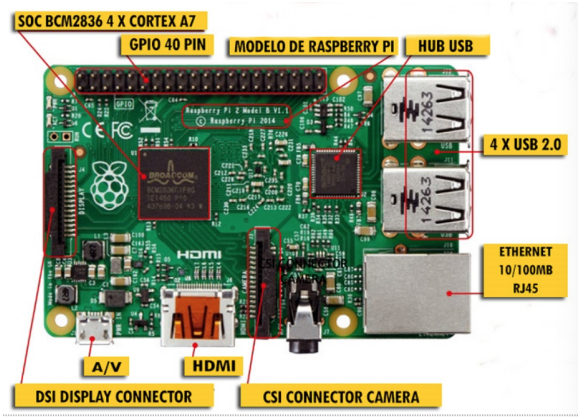

Figura 3.13: RASPBERRY PI 2 MODEL B

- **Raspberry Pi 3:**
  Este modelo llegó al mercado un año después que la Raspberry Pi 2. Esta es la evolución más importante a nivel de potencia y posibilidades, ya que incorpora conectividad Wi-Fi y Bluetooth. Además lleva un nuevo procesador ARM Cortex-A53 de 64 bits con cuatro núcleos que funciona a 1,2 GHz, el cual puede llegar a ser 10 veces más potente que el original.

  A simple vista, es igual que el modelo anterior. Pero al incorporar los nuevos componentes para la conectividad, también ha cambiado la fuente de alimentación necesaria. Ahora es necesario una fuente de alimentación de como mínimo, 5.1V y 2.5A para poder sacar todo el rendimiento.

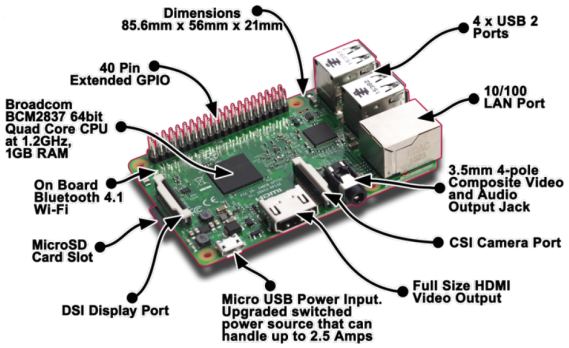

Figura 3.14: RASPBERRY PI 3 MODEL B

- **Raspberry Pi ZERO:**
  Este modelo ha sido el ultimo en salir, es la versión mas reducida, tanto en tamaño como en precio. Cuenta con las funcionalidades de sus predecesoras e incluso puede llegar a ser un 50 % más rápida que la original. La potencia de esta placa viene dada por un procesador ARM11 de 1 GHz, cuenta con 512MB de memoria RAM. En cuanto a los puertos disponibles, esta placa cuenta con un Mini-HDMI de 1080p y con dos MicroUSB que se encargan de la alimentación y de la transferencia de datos. Lo más destacable de esta tarjeta es que solo pesa 9 gramos.

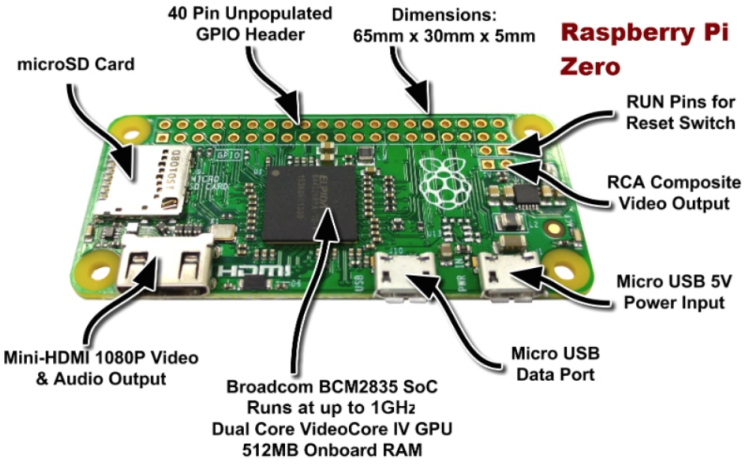

Figura 3.15: RASPBERRY PI ZERO

- **HummingBoard:**
  Esta placa, la cual podemos ver en la Figura 3.16, se vende en el mismo formato de ordenador diminuto, basado en una plataforma ARM. Existen tres modelos, los cuales son de igual tamaño pero se diferencian en las características para adaptarse a los diferentes niveles de exigencia. En comparación con la Raspberry, se puede decir que es más potente ya que su modelo más avanzado cuenta con un i.MX6 Dual con dos núcleos de CPU de 64 bits, 1 GB de RAM con salidas de HDMI, Ethernet, audio estéreo y dos puertos USB. También es superior en precio, el modelo más básico es 10$ más caro. Por lo que este modelo no mejoraría la relación calidad-precio de la Raspberry.

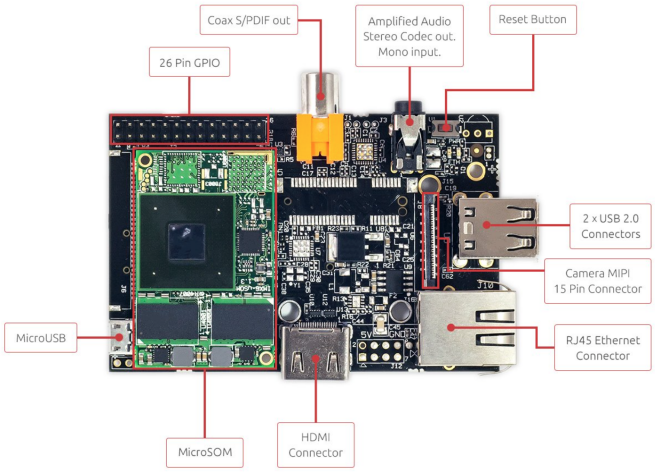

Figura 3.16: HUMMINGBOARD-I2EX

- **BeagleBone Black**
  Esta placa, es para muchos la gran rival de Raspberry. Mantiene el mismo formato y es más potente que las Raspberrys. Es capaz de ofrecer una CPU Ti Sitara ARM Cortex-A8 a 1 GHz, 512MB de memoria y unos gráficos SGX530. Lo que más destaca de esta placa es que es muy escalable ya que ofrece una multitud de módulos compatibles, con los cuales se amplían considerablemente las posibilidades de la placa. Lo malo de esta placa es que también tiene un coste superior a la Raspberry y es más difícil conseguirla ya que no es tan popular como la Raspberry. Cabe destacar que esta placa sería la ideal si no tuviésemos una controladora que maneje los motores y quisiéramos controlarlos con esta placa, ya que las prioridades de los pines están mucho mejor gestionadas que en la Raspberry.

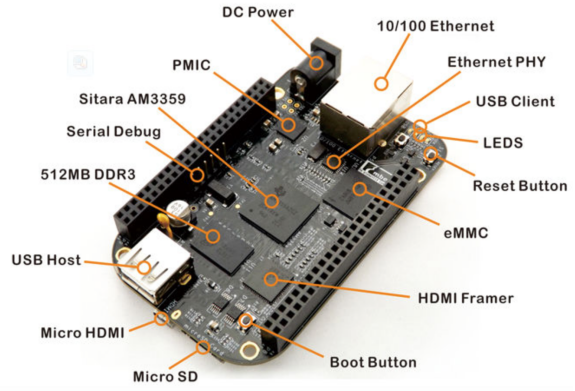

Figura 3.17: BeagleBone Black

- **Odroid**
  Otra de las placas a tener en cuenta, son las hechas por hardkernel [11]. Hay una multitud de modelos en base a las necesidades. El modelo que podría adaptarse a nuestro proyecto es la placa C1, la cual podemos observar en la Figura 3.18. Esta placa se basa en un Amlogic S805 SoC con cuatro núcleos[2] cuenta además con un GB de RAM, tiene 4 USB y uno con capacidad OTG, para usar desde terminales Android. El precio de esta placa es muy similar al de la Raspberry Pi, pero por experiencia propia, al ser aún una placa en desarrollo vienen con ciertos bugs y además la comunidad que hay detrás no es tan grande como la de Raspberry, lo que dificulta encontrar determinada información.

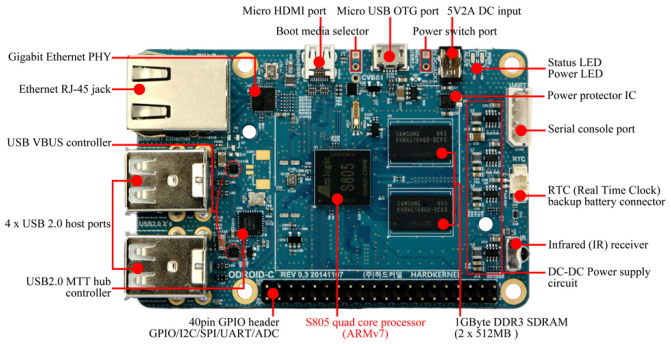

Figura 3.18: Odroid-C1

- **Conclusión:**
  Una vez analizadas las placas que más se adaptan a nuestro proyecto, nos hemos decidido por las placas de Raspberry, ya que consideramos que son las placas que, aunque no son las que más destacan por su potencia, si destacan por su relación calidad-precio y por la gran comunidad de desarrolladores que hay detrás de esta plataforma. Además nos interesa mucho la capacidad de poder cambiar de placa,manteniendo nuestra configuración, según nuestras necesidades ya que como dijimos antes, en lineas futuras queremos añadir la capacidad de procesado de imágenes y video.

[2]Es el mismo procesador que montan los Samsung Galaxy S4

3.2.8. Comunicaciones

La base de este proyecto radica es ser capaces de comunicar los diferentes componentes que formarán parte de el. Para eso, analizaremos algunas de las alternativas posibles que se pueden implementar. Para ello, dividiremos en dos partes las comunicaciones que deberemos implementar. Una que comunicará el teléfono móvil con la Raspberry Pi y la otra que comunicará la controladora con la propia Raspberry. Podemos observar en la Figura 3.1, un esquema de como serán las conexiones.

- *Móvil-Raspberry Pi* Para la conexión entre el móvil y la Raspberry Pi se pueden usar distintos tipos de comunicaciones:

  - **Bluetooth:**
    Este tipo de comunicación se basa en una especificación industrial para redes inalámbricas de area personal (WPAN) la cual posibilita la transmisión de voz y datos entre diferentes dispositivos mediante un enlace por radiofrecuencia en la banda ISM de los 2,4 GHz. Las ventajas de usar esta norma serían las de facilitar las comunicaciones entre equipos móviles, eliminar cables y conectores y tener la posibilidad de crear pequeñas redes inalámbricas.

    Este protocolo esta diseñado para dispositivos de bajo consumo, que requieren corto alcance de emisión y este basado en transceptores de bajo coste. Los dispositivos que incorporan este protocolo pueden comunicarse entre sí cuando se encuentran dentro de su alcance. Las comunicaciones se realizan por radiofrecuencia de forma que los dispositivos no tienen que estar alineados. Según su potencia de transmisión se pueden clasificar en clases, como podemos observar en la Figura 3.19

Clase	Potencia máxima permitida (mW)	Potencia máxima permitida (dBm)	Alcance (aproximado)
Clase 1	100 mW	20 dBm	~100 metros
Clase 2	2.5 mW	4 dBm	~5-10 metros
Clase 3	1 mW	0 dBm	~1 metro

Figura 3.19: Distintas clases de los dispositivos Bluetooth

  - **Wi-Fi:**
    Cuando hablamos de Wi-Fi nos referimos a una de las tecnologías de comunicación inalámbrica mediante ondas más utilizada hoy en día. También llamada WLAN o estándar IEEE 802.11. Esta tecnología permite la conexión de dispositivos electrónicos de

forma inalámbrica. Los dispositivos habilitados con Wi-Fi pueden conectarse a internet a través de un punto de acceso de red inalámbrica. Wi-Fi es una marca de la Wi-Fi Alliance, una organización comercial que adopta, prueba y certifica que los equipos cumplen con los estándares 802.11 relacionados con las redes inalámbricas de área local. Existen diversos tipos de Wi-Fi, basado cada uno en una estándar IEEE 802.11 aprobada, los cuales podemos observar en la Figura 3.19. Las ventajas de este tipo de comunicación son, eliminación de cables, acceso de múltiples dispositivos simultáneamente y compatibilidad absoluta entre dispositivos con tecnología Wi-Fi.

Normas (capa física y de acceso al medio)	Velocidad transmisión máxima (Mbps)	*Throughput* máximo típico (Mbps)	Numero máximo de redes colocalizadas	Banda de frecuencia	Radio de cobertura típico (interior)	Radio de cobertura típico (exterior)
IEEE 802.11a/h	54 Mbps	22 Mbps	14 (5.7 GHz)	5 GHz	85 m	185 m
IEEE 802.11b	11 Mbps	6 Mbps	3	2.4 GHz	50 m	140 m
IEEE 802.11g	54 Mbps	22 Mbps	3	2.4 GHz	65 m	150 m
IEEE 802.11n (40 MHz)*	>300 Mbps	>100 Mbps	1 (2.4 GHz) 7 (5.7 GHz)	5 GHz	120 m	300 m
IEEE 802.11n (20 MHz)*	144 Mbps	74 Mbps	3 (2.4 GHz) 14 (5.7 GHz)	2.4 GHz y 5 GHz	120 m	300 m

Figura 3.20: Distintas estándares Wi-Fi

- **Wi-Fi vs Bluetooth:**
  Vamos a realizar una comparativa para su utilización en nuestro proyecto. El estándar Wi-fi es similar a la red Ethernet tradicional y como tal necesita una configuración previa. Aunque ambos estándares utilizan el mismo espectro de frecuencia, la Wi-Fi tiene una potencia de salida mayor que lleva a conexiones más sólidas. Además, permite conexiones más rápidas, con un rango de distancias mayor y mejores mecanismos de seguridad. Todas estas diferencias resultan de gran relevancia en nuestro proyecto, con lo que la comunicación entre el dispositivo móvil y la Raspberry Pi se realizara mediante Wi-Fi.

- *Raspberry Pi-Controladora*

  - **USB**:
    Es un bus estándar industrial que define los cables, conectores y protocolos usados en un bus para conectar, comunicar o proveer de alimentación eléctrica entre periféricos y dispositivos electrónicos. Su uso se extiende en la actualidad a cualquier dispositivo electrónico o con componentes, desde los automóviles a los reproductores multimedia, pasando por todo tipo de juguetes modernos. Aunque donde más se nota su uso es en los teléfonos

inteligentes donde a día de hoy, ha reemplazado a conectores propietarios casi por completo.

- **GPIO**:
  Es un pin genérico en un chip, en nuestro caso en la Raspberry Pi, cuyo comportamiento se puede controlar por el usuario en tiempo de ejecución. Estos pines no tienen ningún propósito especial definido y por lo tanto no se usan de forma predeterminada. La idea es que a veces, para el diseño de un sistema completo que utiliza dicho chip podría ser útil contar con un puñado de lineas digitales de control adicionales, y tener a su disposición el ahorro de tener que organizarlos en circuitos adicionales para proporcionarlos. Este pin puede observarse en cualquiera de las figuras de las Raspberry Pi que se mostraron anteriormente..

- **Xbee**:
  Estos módulos son soluciones integradas que brindan un medio inalámbrico para la interconexión y comunicación entre dispositivos. Estos módulos utilizan el protocolo de red llamado IEEE 802.15.4 para crear redes FAST POINT-TO-MULTIPOINT o para redes punto a punto. Este pequeño chip, el cual podemos observar en la Figura 3.21, es capaz de comunicarse de forma inalámbrica con otros Xbee [19]. Existen distintos modelos de Xbee, pero una de las ventajas de este chip, es que todos usan los mismos pines[3] que además se encuentran en los mismo lugares, independientemente del modelo utilizado.

- **Comparativa:**
  Para nuestro proyecto se ha decidido el uso del USB. Los principales motivos son que el Xbee necesitaría de dos módulos, uno para la Raspberry Pi y otro para la controladora, lo que nos encarecería el coste final del proyecto. Y con respecto al GPIO, su programación es más compleja que la del USB. Por lo que en este proyecto la comunicación entre estos dos periféricos sera mediante USB y un software que veremos a continuación.

[3] alimentación, tierra y TX/RX

Figura 3.21: Dispositivo Xbee

3.2.9. Firmware

En esta sección se explicarán los lenguajes utilizados y porque se han elegido para este proyecto.

- **Android:**
  Android es un sistema operativo móvil basado en Linux, el cual tiene la mayoría de código bajo la licencia Apache el cual es libre y de código abierto,que junto con aplicaciones middleware[4], está enfocado para ser utilizado en dispositivos móviles, como pueden ser los teléfonos inteligentes, tabletas o otros dispositivos. Está desarrollado por Open Handset Alliance, la cual es liderada por Google.

  Las unidades vendidas de teléfonos inteligentes con Android lideran las cuotas de mercado año tras año a nivel mundial. Lo que más destaca de este sistema es que tiene un gran comunidad de desarrolladores programando todo tipo de aplicaciones para extender las funcionalidades de los dispositivos. La cantidad de aplicaciones disponibles hoy en día en la tienda oficial de Android[5] a sobrepasado el millón, destacando que casi dos tercios de ellas sean gratuitas. Las aplicaciones están programadas en el lenguaje de programación *Java*

  La estructura de este sistema operativo se compone de aplicaciones que se ejecutan en un framework Java de aplicaciones orientadas a objetos sobre el núcleo de las bibliotecas de Java en una maquina virtual Dalvik con compilación en tiempo de ejecución. Las bibliotecas están escritas en lenguaje C, incluyen un administrador de interfaz gráfica, un framework OpenCore, una base de datos relacional SQLite, una interfaz de programación de API gráfica OPENGL, un motor de renderizado Webkit, un motor gráfico SGL, SSL y una biblioteca estándar de C, Bionic. El sistema operativo está compuesto por 12 millones de líneas de código, incluyendo XML,C,Java y C++.

[4] intercambio de información entre aplicaciones
[5] Google Play

En nuestro proyecto usaremos este sistema operativo, entre otras opciones disponibles como pueden ser, iOS,Windows Phone.. por varias razones. En primer lugar por tratarse de una plataforma de código abierto y libre.

En segundo lugar, por ser una de las plataformas que mayor expansión está sufriendo en los últimos años. En tercer lugar, a la par que este proyecto, he cursado una asignatura en la que se trabajaba con Android y con Java. Además de haber usado Java durante toda la carrera, lo que me ha ayudado al desarrollo de este proyecto.

Figura 3.22: Logo Android

- **Arduino:**
  Arduino es un proyecto de hardware libre, que ideó y desarrollo una plataforma completa de hardware y software compuesta por unas placas de desarrollo que integran un microcontrolador y un entorno de desarrollo (IDE), diseñado para facilitar el uso de la electrónica en proyectos multidisciplinarios. Toda esta plataforma, tanto su parte hardware como su parte software, son open-source, es decir, permite total libertad de acceso a los mismos.

  El hardware consiste en un placa de un circuito impreso con un microcontrolador, habitualmente de la familia Atmel, y puertos digitales y analógicos de entrada/salida, a los cuales se les pueden conectar placas de expansión las cuales amplían las funcionalidades de la placa.

  El software consiste en un entorno de desarrollo (IDE) basado en el entorno de Processing y el lenguaje de programación basado en Wiring, así como en el cargador de arranque que es ejecutado en dicha placa. El microcontrolador se programa a través de un ordenador, gracias a la comunicación serial mediante un convertidor de niveles RS-232 a TTL serial. En nuestro proyecto usaremos esta plataforma, cabe destacar, que la mayoría de controladoras de vuelo de código abierto trabajan con Arduino. En nuestro caso, implementaremos el código Multiwii en la controladora CRIUS AIO PRO mediante el IDE de Arduino.

Figura 3.23: Logo Arduino

- **Python:**
  Es un lenguaje de programación interpretado cuya filosofía hace hincapié en una sintaxis que favorezca un código legible. Se trata de un lenguaje de programación multiparadigma, ya que es capaz de soportar orientación a objetos, programación imperativa y , en bastante menor medida, programación funcional. Usa tipado dinámico y conteo de referencias para la administración de memoria. Además, es multiplataforma.

  En nuestro proyecto vamos a usar este lenguaje por una serie de razones, la primera es por que este lenguaje es capaz de realizar la tarea de la transmisión serial de una manera mucho más cómoda que Java, de esto hablaremos un poco mas adelante cuando expliquemos el desarrollo del software y las limitaciones que tuvimos con Java. En segundo lugar, la cantidad de librerías que tiene disponibles nos van a venir muy bien a la hora de desarrollar el proyecto, de hecho, la gran mayoría de proyectos de este estilo usan este lenguaje por su simplicidad y cantidad de librerías que facilitan enormemente el trabajo.

Figura 3.24: Logo Python

3.2.10. S.O Raspberry

En cuanto a los sistemas operativos que se pueden implementar en una Raspberry Pi, podemos afirmar que hay una gran variedad y se pueden clasificar en función de las necesidades de cada uno, que pueden ir desde un ordenador personal o servidor hasta un Media Center. A continuación vamos a enumerar los más relevantes, para eso vamos a clasificarlos en dos

grupos, los oficiales[6] o los no oficiales[7].

- *Oficiales:*
  - Raspbian o Raspbian Lite
  - Pidora
  - OSMC
  - OPENELEC
  - RISC OS
- *No oficiales:*
  - ARCH Linux
  - Occidentalis
  - OpenSUSE
  - Ubuntu 14.04
  - Windows 10
  - Android[8]

Existen muchos más sistemas operativos que no vamos a nombrar por no ser interesantes para la resolución de este proyecto. Para este proyecto hemos decidió usar Raspbian. Este sistema operativo es libre basado en Debian, es el que está más optimizado para el hardware de la Raspberry Pi. Este sistema operativo viene con más de 35000 paquetes que proporcionan programas básicos y utilidades.

Este sistema operativo se completó en Junio de 2012, sin embargo su desarrollo sigue activo con especial énfasis en la mejora de la estabilidad y la actualización de los paquetes de los que ya se dispone.

Figura 3.25: Logo de Raspbian

[6]Descargables desde la propia web oficial de Raspberry
[7]Descargables desde la web particular de cada uno
[8]No es completamente funcional por el momento

Capítulo 4

Implementación del Dron

4.1. Diagrama de Gantt

En este apartado, vamos a analizar el tiempo de dedicación a este proyecto con un diagrama de Gantt, esta es una gran herramienta gráfica, la cual nos permite exponer el tiempo de dedicación previsto para diferentes tareas a lo largo de un tiempo total determinado.

Como podemos observar en la Figura 4.1 dada la posición de cada tarea, podemos observar las relaciones de interdependencia que hay en nuestro proyecto.

Cabe destacar de esta planificación, que aunque no se pueda observar en la Figura 4.1, en la mayoría de tareas tenemos holgura cero, es decir, no solo tenemos un camino crítico. Un retraso en alguna de las tareas implicaría un retraso general del proyecto. También es destacable que debido a la inexperiencia en este tipo de proyectos, dicha planificación se basa en la analogía de otros proyectos de similares características por lo que es posible que las fechas varíen durante la realización del proyecto, además, en esta planificación no se ha tenido en cuenta el aprovisionamiento de materiales, el cual es muy difícil de medir y predecir.

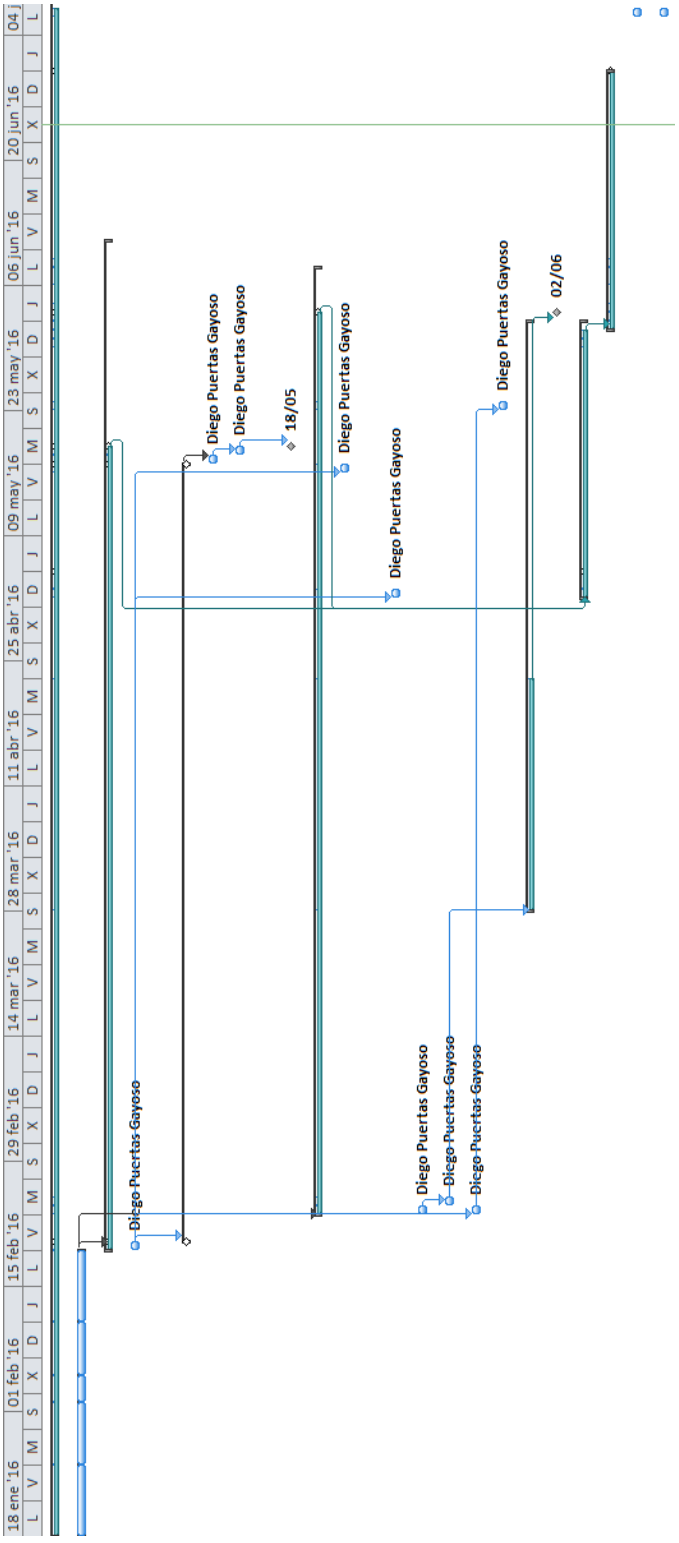

Figura 4.1: Diagrama de Gantt del proyecto

Para entender un poco mejor el diagrama anterior, en la Figura 4.2 vemos el desglose mediante la herramienta fundamental en la gestión de proyectos, la EDT, la cual es una descomposición jerárquica, del trabajo a ser ejecutado en este proyecto para poder cumplir los objetivos y crear los entregables requeridos. En la figuras mencionadas anteriormente solo vemos las tareas mas destacables del proyecto y no las subtareas, ya que se nos extendería demasiado.

	EDT	Nombre de tarea	Trabajo	Comienzo	Fin	Predecesoras
1	**1**	**Proyecto drone**	**536 días**	**lun 18/01/16**	**jue 07/07/16**	
2	**1.1**	**Requisitos del Sistema**	**51 días**	**lun 18/01/16**	**jue 18/02/16**	
9	**1.2**	**Requisitos hardware**	**147 días**	**vie 19/02/16**	**vie 10/06/16**	**2**
18	**1.3**	**Requisitos software**	**152 días**	**mar 23/02/16**	**mar 07/06/16**	**2**
28	**1.4**	**Montaje**	**30 días**	**lun 02/05/16**	**mié 01/06/16**	**9;18**
36	**1.5**	**Pruebas**	**30 días**	**mié 01/06/16**	**mié 29/06/16**	**28**
47	1.6	Producto final	1 día	mié 06/07/16	mié 06/07/16	
48	2	Presentación al tribunal	1 día	mié 06/07/16	mié 06/07/16	

Figura 4.2: EDT del proyecto

4.2. APP: DronePi

En este apartado vamos a describir las distintas actividades de las que dispone nuestra aplicación. Todas las acciones que el usuario puede realizar conllevan la interacción con la interfaz gráfica del usuario y estas peticiones actualizan dicha interfaz para mostrar los datos de interés.

Esta aplicación ha sido pensada para el control de drones que lleven una Raspberry Pi a mayores del controlador. La aplicación consta de varias actividades. La principal, el control del dron, se divide en dos actividades. Una para controlarlo con una serie de botones y otra mediante el uso del acelerómetro de nuestro dispositivo. Estas actividades utilizan una conexión UDP entre el dispositivo y la Raspberry para mandar los distintos controles o valores de los sensores, para que el controlador pueda variar la corriente suministrada a los motores. Además de poder controlarlo se busca que el usuario disponga de información importante siempre a mano. Para esto se ha implementado:

- Una actividad que nos muestra el tiempo, con los datos del viento como parámetro importante.
- Una actividad que nos permite geolocalizarnos además de poder trazar una ruta de un punto A a un punto B.
- Una actividad que nos permite comprobar la conexión UDP.
- Una actividad que nos permite refrescar la actividad en la que estamos.
- Un actividad que nos muestra información acerca del proyecto y nos permite mandar un e-mail con posibles sugerencias.

- En todas las actividades hay un ActionBar que nos permite movernos por las demás actividades, además de mostrar información del tiempo actualizada.

Antes de desarrollar la aplicación, explicaremos un poco el funcionamiento del framework que android nos ofrece y las actividades.

Las actividades son la capa de presentación de las aplicaciones. Las interfaces gráficas se construyen extendiendo de la clase AppCompatActivity. Uno de los aspectos más importantes de las aplicaciones es su ciclo de vida. Las aplicaciones Android no controlan sus propios ciclos de vida, para eso está el RunTime. El estado de una actividad ayuda a determinar cual es la prioridad de esa aplicación. Estas actividades se crean y se destruyen o son movidas a la pila de actividades. Como podemos observar en la Figura 4.3 podemos encontrar las actividades en cinco estados diferentes y los métodos que se utilizan para los cambios de estado:

- Activa:
  Cuando la actividad está en primer plano, es decir, es visible y el usuario puede interaccionar con ella. Android tratará de mantener esta actividad funcionando a casi cualquier coste, bloqueando a otras actividades que estén por debajo.

- Pausada o visible:
  La actividad es visible pero no tiene el foco. Este estado se alcanza cuando otra actividad con alguna parte transparente o que no ocupa toda la pantalla. Si la actividad está tapada por completo pasa a estar parada.

- Parada:
  Cuando la actividad no es visible. El programador debería guardar el estado de la interfaz de usuario, preferencias, etc.

- Inactiva:
  Antes de destruir una actividad y antes de que se muestre, su estado es inactivo. Las actividades en este estado se han eliminado de la pila y se tienen que volver a iniciar antes de que se puedan mostrar al usuario.

- Destruida:
  Cuando la actividad termina al invocarse al método finish(), o es destruida por el sistema.

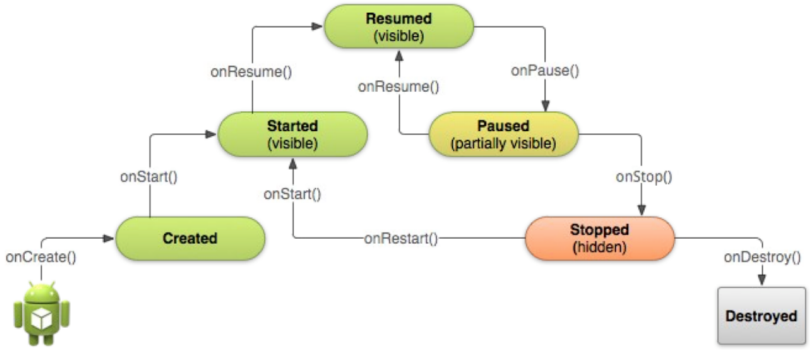

Figura 4.3: Ciclo de vida de las actividades

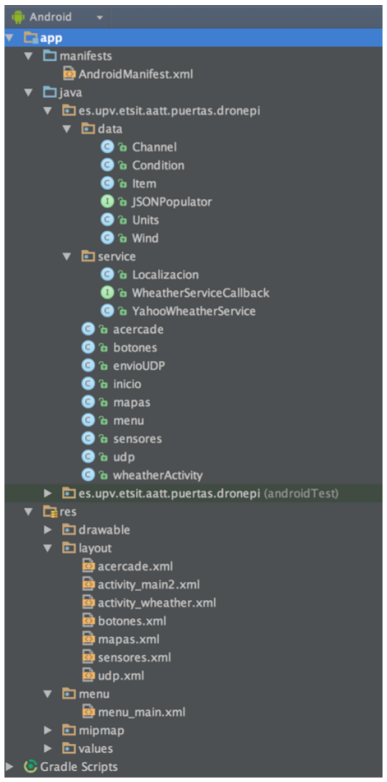

Figura 4.4: Esquema de los ficheros de la aplicación

Ahora procederemos a la explicación de las diferentes actividades de las que consta la aplicación y a la explicación del software que hay detrás de ellas.

En la Figura 4.4 podemos ver el esquema de la aplicación al completo,

con las diferentes clases y las diferentes interfaces que hemos creado.

En la Figura 4.5 Observamos una pantalla de inicio con el logo de Raspberry, esta actividad simplemente consta de una ProgressBar la cual tiene una duración de 4 segundos mientras iniciamos la aplicación. Esta actividad da paso a la siguiente actividad automáticamente. Su finalidad es simplemente estética.

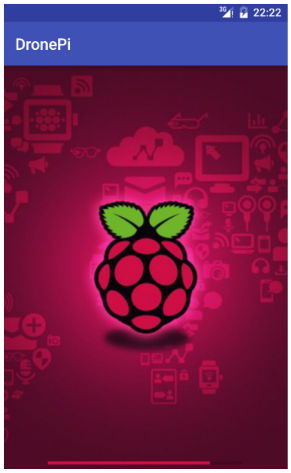

Figura 4.5: Actividad de inicio de la aplicación

En la Figura 4.6 Observamos una actividad que consta de 5 buttons, una seekbar y un textview con los cuales podemos manejar esta actividad. Los 4 botones que observamos en la parte superior de la actividad sirven para elegir la dirección que tomara nuestro dron. Con la seekbar podremos elegir la aceleración de los motores, la cual podremos observar en todo momento en el textview, aunque esta solo se enviara una vez hayamos soltado la barra en un numero fijo. Hemos estipulado cuatro velocidades disponibles, con un máximo de 160, se han dividido en rangos de 40, teniendo cuatro estados, LOW, MID, HIGH o SUPER. Por último, el botón restante, con forma de casa, nos llevará de vuelta al punto de origen[1]. En cuanto a la parte no visible de esta actividad, se encarga de enviar mediante paquetes UDP toda la información necesaria para el control de los motores al servidor ubicado en la Raspberry que más adelante comentaremos.

[1] aún por implementar en la controladora

Figura 4.6: Actividad de control mediante botones de la aplicación

La Figura 4.7 consiste de una serie de layouts sencillos donde se encuentran componentes tales como TextView, un Button y un WebView. En los TextView tenemos la latitud y longitud[2] correspondiente a la localización del usuario para que luego con esas coordenadas pueda introducirlas en el WebView. Se decidió usar un WebView por simplicidad de código para de este modo poder geo localizarnos con las coordenadas obtenidas y poder trazar rutas con las opciones que proporciona la propia página web. Lo ideal sería que hubiéramos generado una página web sencilla para este propósito pero dada la escasez de tiempo de la que disponíamos no pudo ser y tuvimos que apoyarnos en páginas ya existentes. Adicionalmente se proporciona un botón para que el usuario pueda recalcular su posición y re-introducir esos datos en el WebView.

[2]En la figura 4.7 se observa en los textview que no aparecen como no disponibles, esto es debido a que la captura se realizo en el emulador, el cual por defecto no proporciona estas coordenadas

Figura 4.7: Actividad de mapas de la aplicación

En la Figura 4.8 podemos observar tres TextView en los cuales obtenemos la información que nos proporciona el acelerómetro de nuestro dispositivo. También dispone de una serie de avisos cuando las coordenadas varían por encima de un determinado mínimo. El funcionamiento de esta actividad es parecido a la actividad de botones explicada antes, se trata de que mediante el envío de paquetes UDP al servidor alojado en la Raspberry, los procese y los reenvíe a la placa controladora. La función de esta actividad es poder controlar el dron mediante el movimiento de nuestro dispositivo móvil, el cual analizaremos mediante los sensores ubicados en el, y trataremos de emular esos movimientos en el dron[3].

Figura 4.8: Actividad de control mediante sensores de la aplicación

[3]Aún por implementar en la controladora

En la Figura 4.9 podemos observar una serie de TextViews con un ImageView en la parte superior de la actividad. Esta actividad tiene como funcionalidad mostrarnos un icono en función del tiempo que haga, estos iconos son personalizados ya que los que nos ofrecía la API de Yahoo no disponían de una buena calidad y no se veían correctamente. Además, nos da información de la temperatura, la sensación térmica, y lo que resulta más importante en relación a nuestro proyecto, la dirección del viento y su velocidad. Por último el TextView del final de la imagen es un requerimiento de Yahoo por el uso de su API. En cuanto a la parte no visible de esta actividad, lo que hace es conectarse a la API de Yahoo mediante un JSON para obtener todos los datos actualizados en tiempo real. Cabe destacar que se podrían aumentar los datos mostrados en función de nuestro interés, dentro de la limitación que tenga de datos la API mencionada anteriormente.

Figura 4.9: Actividad de visualización del tiempo de la aplicación

En la Figura 4.10 tenemos una serie de EditText en los cuales podemos poner los datos de la ip del servidor y del puerto en el que está a la escucha de nuevos paquetes UDP y enviar un mensaje. La funcionalidad de esta actividad es comprobar que la conexión está establecida correctamente y que el servidor es capaz de procesar los paquetes. La parte no visible de esta actividad se encarga, de almacenar tanto la ip, el puerto y el mensaje escrito para después enviarlos en un paquete UDP al servidor.

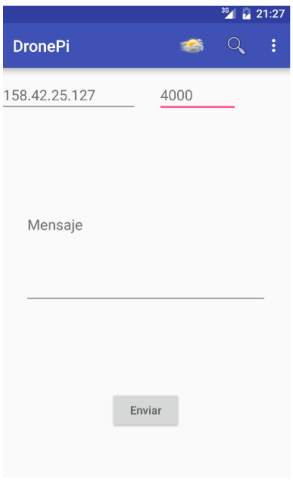

Figura 4.10: Actividad configuración del envío UDP de la aplicación

Por ultimo, en la Figura 4.11 podemos observar dos imágenes, la de la izquierda es la actividad en si, donde tenemos una imagen con un mensaje y un botón el cual nos permite abrir nuestro gestor de correo,imagen de la derecha, y enviar un e-mail de sugerencias. Podemos fijarnos como en el gestor de correo ya nos aparece puesto tanto nuestro correo, como el correo de envío, como el asunto y una breve descripción del correo a enviar. Por lo que la parte no visible de esta actividad realiza un intent en el cual llamamos al gestor de correos pasándole la información antes mencionada.

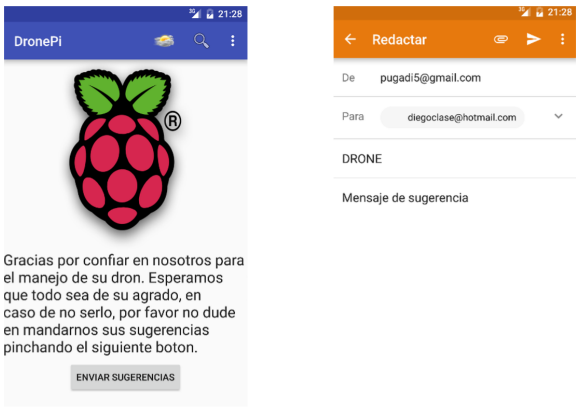

Figura 4.11: Actividad acerca de de la aplicación

Nótese que en todas las actividades disponemos de una ActionBar en la cual podemos observar siempre el icono del tiempo. el cual se actualiza automáticamente con nuestra ubicación. También disponemos un botón buscar y un action overflow, el cual nos permite navegar entre las distintas

actividades de nuestra aplicación y recargar la propia actividad en la que estemos. Esta aplicación ha sido diseñada para una total compatibilidad con la ultima version de Android, Marsmallow. Como requerimiento mínimo de version será necesario disponer de la version Ice Cream Sandwich, o lo que equivaldría a la versión 15 de la API.

4.3. Configuración Raspberry

En este apartado explicaremos las distintas configuraciones que debemos realizar en nuestra Raspberry, desde instalar el sistema operativo a la configuración como punto de acceso y la realización y puesta en marcha de nuestro servidor.

- ***Configuración inicial***
  Lo primero que debemos hacer con la Raspberry es preparar un tarjeta microSD con uno de los sistemas operativos que comentamos en uno de los apartados anteriores. Como dijimos, el sistema elegido ha sido Raspbian por ser uno de los más estables actualmente, además de ser uno de los que más paquetes nos proporciona, entre ellos cabe destacar los IDEs de programación de Python o Bluej. Para la preparación de la tarjeta existen diferentes métodos, se puede realizar mediante terminal o mediante alguna de las aplicaciones que hay para este propósito que nos facilitan la tarea. Lo primero que debemos hacer, independientemente del método, es descargar la imagen del sistema operativo, aquí también existen diferentes metodologías, una que merece destacar para los que están empezando, es la distribución NOOBS [24], la cual ya trae por defecto varios sistemas operativos y facilita su instalación. Una vez obtenida la imagen,es decir, descomprimida del archivo que hemos descargado, se puede copiar a la tarjeta de varias manera, mediante un terminal con la instrucción[4] :

  sudo dd if=Raspbian.img of=/dev/NOMBRE bs=2m

  Donde Raspbian.img es nuestro nombre de archivo y NOMBRE es el nombre de nuestra tarjeta. La otra alternativa es mediante alguna aplicación externa como puede ser ApplePi Baker [27] la cual nos facilitara el trabajo y además es bastante más rápida de lo que sería hacerlo mediante el terminal.

- ***Configuración como punto de acceso***
  En este apartado explicaremos como configuramos nuestra raspberry

[4]Instrucción válida para el terminal de MAC

para crear una red Wi-Fi, apoyándonos en el diagrama de bloques que aparece en la Figura 4.12, para que el dispositivo móvil pueda conectarse a ella y por lo tanto poder enviar los paquetes UDP al servidor que estará alojado en ella. Para poder realizar esta configuración es necesario matizar los materiales necesarios en función del modelo de Raspberry que usemos, ya que dependiendo de este, tendrá incorporado o no el adaptador wifi. Como tenemos la ventaja que la tarjeta que preparemos, y las configuraciones que realicemos quedaran guardadas en la propia tarjeta, luego podremos usarla en cualquier modelo de Raspberry.[5]

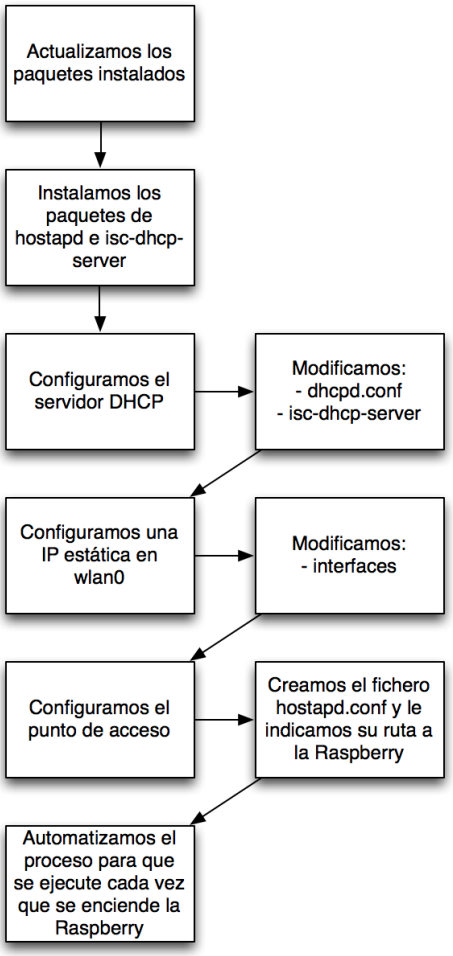

Figura 4.12: Diagrama de bloques de la configuración del punto de acceso

[5]Nótese que para la configuración del punto de acceso seria necesario, en caso de usar un adaptador, especificar los drivers de este, ya que, salvo el modelo Raspberry Pi 3, el resto necesita de un adaptador adicional

Lo primero que debemos hacer es actualizar los paquetes de la distribución Raspbian, instalada previamente. Una vez actualizado tendremos que descargarnos los paquetes que nos permitirán crear el punto de acceso WiFi, hostapd e isc-dhcp-server. Después procederemos a configurar el servidor DHCP[6] mediante la modificación de sus archivos de configuración, los cuales están ubicados en */etc/dhcp* y */etc/default* y se llaman dhcpd.conf e isc-dhcp-server. Una vez modificados con los parámetros adecuados procedemos a configurar una IP estática para la interfaz de wlan0, de esta manera siempre será la misma, lo que nos facilita la implementación del servidor. Para configurarla primero deberemos desactivar la interfaz de wlan0 y después modificar el archivo que esta ubicado en */etc/network* y que se llama interfaces, una vez modificado volveremos a habilitar la interfaz. Ahora es el turno de configurar nuestro punto de acceso con las características que nos interesen, como pueden ser, el SSID,contraseña, el tipo de seguridad.. Para esto debemos crear el archivo hostapd.conf[7] en la ruta */etc/hostapd*. Una vez modificado es necesario especificarle a la Raspberry donde puede encontrar este fichero, para ello, debemos modificar el archivo ubicado en */etc/default* y con nombre hostapd. Por ultimo, debemos crear un servicio o daemon que se ejecute cada vez que la Raspberry se encienda.

- ***Implementación del servidor UDP***
  En la Figura 4.13 podemos ver el diagrama de bloques en el que se basa nuestro servidor UDP. Este servidor tiene que ser capaz de recibir paquetes UDP desde el dispositivo móvil, para ello debe estar escuchando siempre el puerto 4000 en nuestro caso, recibir el paquete, comprobar su contenido y luego ser capaz de reenviarlo por puerto USB a la controladora, la cual controlará los motores en función del comando que haya recibido por dicho puerto. En la planificación inicial, se pensó en la realización de este servidor mediante el lenguaje de programación Java, ya que es el lenguaje que más hemos usado durante la carrera, se pensó que seria el lenguaje ideal para el propósito de este servidor. Una vez programado, a la hora de la implementación y comprobación de su funcionamiento, nos dimos cuenta que al migrar el servidor a la Raspberry este no funcionaba correctamente, es decir, recibía el paquete UDP pero no era capaz de reenviarlo por el puerto USB. Esto es debido a que inicialmente se utilizo una librería hecha por unos chicos de PanamaHitek [17] que facilitaba el uso de la librería RXTX [13] pe-

[6] Protocolo que nos permitirá asignar automáticamente una dirección IP al conectarnos a la red

[7] Es en este archivo donde debemos fijarnos en el parámetro driver, el cual es el que tendremos que modificar según el tipo de Raspberry utilizado y su adaptador WiFi, en caso de usar la RPI3 no haría falta ponerlo.

ro esta solo funcionaba en el sistema operativo Windows. Después de programarlo solo con la librería RXTX y de analizar ambas librerías se descubrió que la maquina virtual que usa Java no es capaz de llegar tan abajo, ya que este lenguaje es de alto nivel y acceder a los puertos serie se puede convertir en una gran desventaja de este lenguaje. Una de las grandes ventajas de este lenguaje es que es multiplataforma, y su maquina virtual interpreta pero no compila, lo que en nuestro proyecto nos penaliza ya que como la maquina virtual no es capaz de llegar a controlar los puertos serie, se basa en drivers genéricos[8], los cuales no funcionan para la Raspberry Pi.

Este contratiempo nos obligo a buscar una solución en otro lenguaje que si que fuese capaz de controlar a bajo nivel estos puertos o que al menos tuviese librerías que si que fuesen capaces. La mejor solución, fue aprender lo básico de python y programarlo en este lenguaje, ya que gracias a la librería pyserial, es posible realizar la comunicación por el puerto USB. El cambio de lenguaje no afecto a la estructura del servidor, el cual sigue siendo la misma que podemos ver en la Figura 4.13.

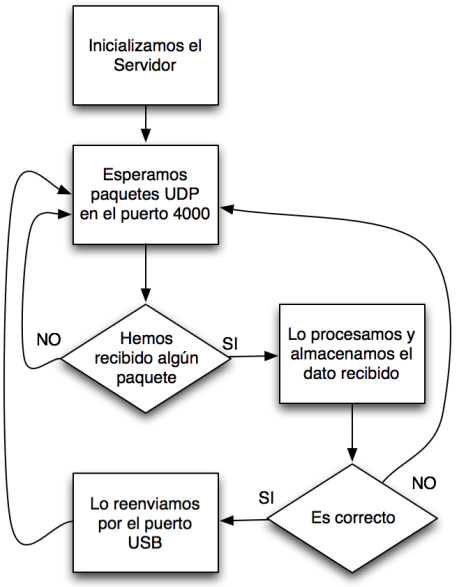

Figura 4.13: Diagrama de bloques del servidor UDP

[8]Están disponibles para Mac, Linux, BSD, Windows y Toybox

Los comandos que tiene que enviar el servidor a la controladora programada con el software de Multiwii, se basan en la dirección y en la aceleración que le hayamos ordenado con el dispositivo móvil, pero el software Multiwii no los entiende de esta manera que hemos comentado. El software de la Multiwii se basa en varios grados de libertad, en nuestro caso, como tenemos cuatro motores, parece razonable que sean cuatro los grados de libertad que son controlables[9]. Los grados que son controlables los podemos ver en la Figura4.14. Una breve explicación de cada grado del que dispondremos es la que sigue:

- **Roll:**
  Es la inclinación que hay hacia los costados o balanceo.
- **Pitch:**
  Es la inclinación que hay hacia delante/atrás o cabeceo.
- **Yaw:**
  Es la rotación sobre si mismo.
- **Throttle o Altitud:**
  Es la aceleración de los motores.

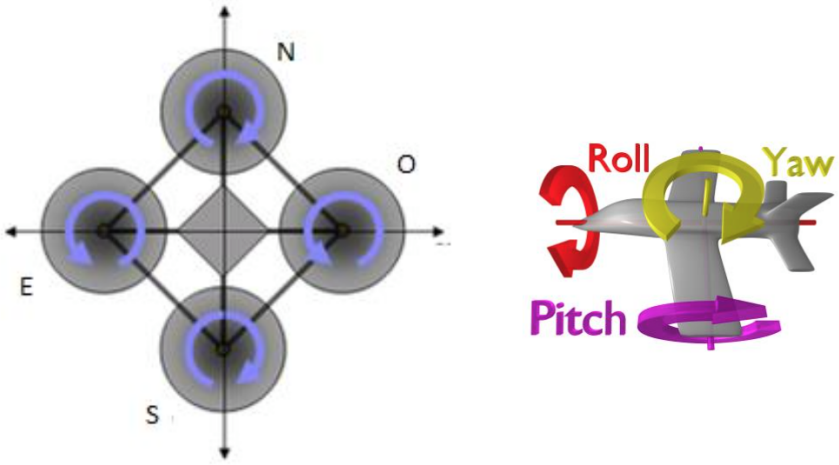

Figura 4.14: Grados del dron

Para entender un poco más como deberíamos controlar los motores para que tomen las direcciones que les hemos ordenado, nos basamos en la Figura 4.15 donde podemos ver, según la dirección a la que queramos ir, como debemos reaccionar en los cuatro grados de libertad antes mencionados.

[9]El software Multiwii permite más grados de libertad

	Norte	Sur	Este	Oeste
Throttle	↑	↑	↑	↑
Pitch	↓	0	↑	0
Roll	0	↓	0	↑
Yaw	↓	↑	↓	↑

Figura 4.15: Matriz de comandos

Una vez analizados los grados de libertad, ¿Cómo podemos enviarlos a la controladora?

El software Multiwii utiliza un protocolo binario con CRC. Este protocolo se denomina MSP (Multiwii Serial Protocol) [28]. La estructura de este protocolo es la que se puede ver en la Figura 4.16

$M>[data length][data][checksum]

Figura 4.16: Formato del mensaje MSP

Donde:

- $M>son tres bytes que marcan el comienzo del frame. Depende de si es >o <significa si es de entrada o de salida, si se desconoce no se pone ninguno.
- data length: es un byte que representa el número de bytes que componen el mensaje
- code: es un byte que identifica el comando o request
- data: son los parámetros del comando o del request
- checksum : es el byte de control

Podemos ver un poco más concretamente en la Figura 4.17 el formato que acabamos de mencionar tanto para enviar como para recibir los mensajes.

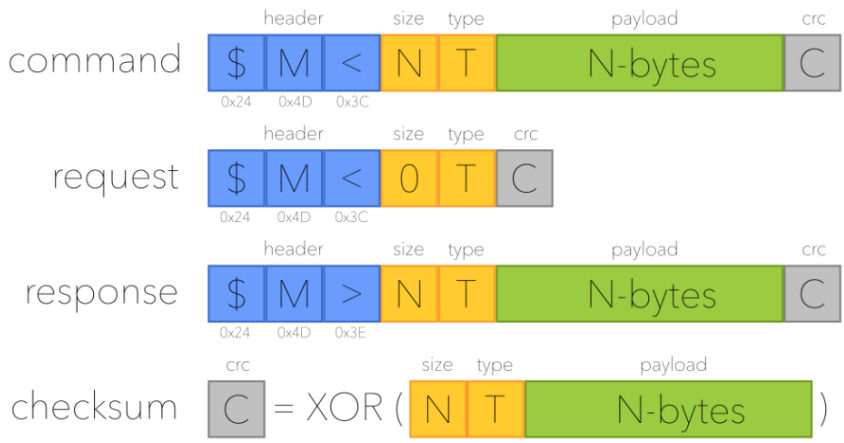

Figura 4.17: Formato del mensaje MSP para enviar y recibir los mensajes

Dentro de la variedad de mensajes que podemos intercambiar nos interesan principalmente dos:

- MSP_ATTITUDE: Este request hace que la controladora nos responda con un mensaje en el cual están los ángulos de Euler para roll, pitch y yaw.
- MSP_SET_RAW_RC: Este comando nos permite enviarle a la controladora los valores de cada canal.[10] Estos comandos sobreescriben a los de la radio.[11]

Una vez implementado el servidor, lo configuramos para que se inicie siempre cuando se arranca la Raspberry.

4.4. Configuración Crius AIO Pro

En nuestro caso, como usamos el software de Multiwii, para la implementación del software controlador es necesario el IDE de Arduino, el cual nos facilitará la tarea de implementación y la configuración de ciertos parámetros importantes. Existen distintos tipos de software que podemos implementar en nuestra placa controladora, las opciones disponibles son:

[10] estos valores han de estar entre 1000 y 2000.

[11] sólo sobreescriben los de la radio si se encuentran en el buffer del puerto serie, luego la radio retoma el control, lo que dificulta el control, ya que deberíamos enviar los comandos a más de 50Hz para mantener un control estable.

- MegaPirateNG
  Este software fue creado originalmente como un repositorio hijo del original Ardupilot, en base a esto, este software permite utilizar un código prácticamente idéntico en funcionalidades.[12] Para cargar este software sería necesario un IDE propio de MegaPirateNG, el cual podemos ver en la Figura 4.18

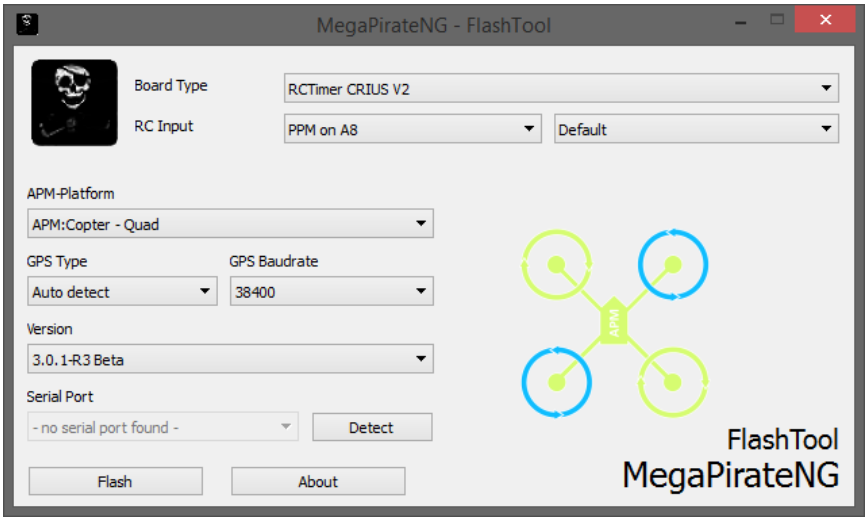

Figura 4.18: IDE de MegaPirateNG

- Multiwii
  De este software ya hablamos en el apartado de las controladoras. Es el elegido para nuestro proyecto por su sencillez a la hora de configurar los distintos parámetros y porque consigue un estabilidad muy notable.

  En la Figura 4.19 podemos observar el propio IDE de Arduino con el software cargado de la Multiwii. Podemos ver una serie de pestañas a la derecha, estas pestañas son de las que esta compuesto el código completo. Como hemos explicado antes, al final no modificaremos este código para cambiar la forma de coger los datos, si no que el propio servidor de la Raspberry los enviará con el formato adecuado para que el software los entienda. Por lo que la única modificación que debemos realizar es el archivo *config.h* en cual deberemos especificar el tipo de frame que usaremos, el numero de motores, y el tipo de placa controladora que estamos usando.

[12] Pero siempre más atrasado, ya que hay que esperar a que salgan las nuevas versiones de Ardupilot

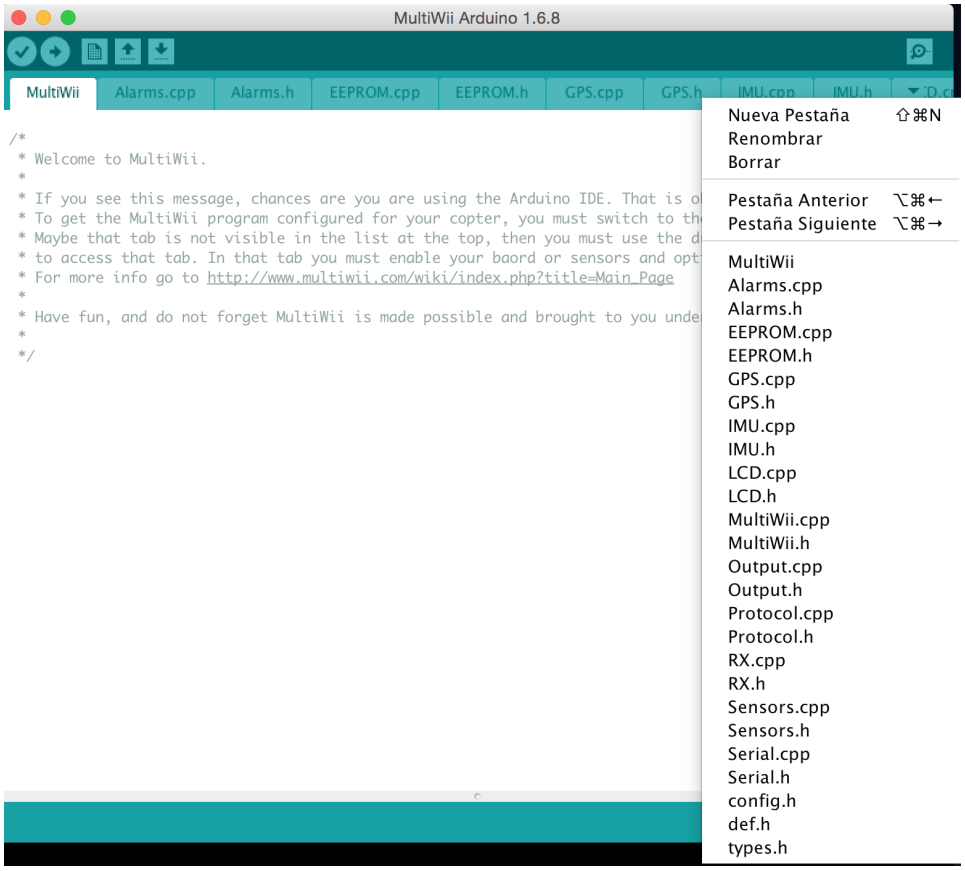

Figura 4.19: IDE de Arduino con el software Multiwii cargado

Capítulo 5

Conclusiones

5.1. Valoración

Como valoración final de este proyecto, hay que destacar que este proyecto ha sido muy ambicioso y con una complejidad técnica elevada. La idea principal era poder desarrollar un proyecto multidisciplinar donde interviniesen tanto el diseño del hardware como el diseño del software, incluyendo la comunicación entre todas las partes implicadas.

El valor de lo que hemos podido aprender en este proyecto es incalculable, hemos tenido la experiencia de desarrollar este sistema desde cero sin ningún tipo de experiencia previa. Esto nos ha servido para poner en práctica una infinidad de cosas que hemos ido aprendiendo a lo largo de esta carrera, las cuales nos han dado la capacidad para resolver casi cualquier tipo de problema, siempre intentado mantener una relación calidad/coste elevada en un plazo muy corto de tiempo.

Dentro de este proyecto podríamos diferenciar cuatro grandes bloques, los cuales han supuesto grandes y emocionantes retos para solucionar. Primeramente, el bloque al cual le he dedicado el mayor número de horas ha sido a documentarme y buscar toda la información posible sobre el mundo de los UAV. Tanto a la parte hardware como a la parte software para poder llevar a cabo este proyecto. Por otro lado, el uso de la Raspberry Pi ha supuesto un gran descubrimiento, ya que he tenido que documentarme sobre la propia placa y sobre su uso con diferentes sensores, shields y conexiones. Esta investigación me ha permitido conocer una gran variedad de dispositivos que pueden complementar a esta placa y que las posibilidades sólo están limitadas por la imaginación. En lo referente a la comunicación entre todos los dispositivos mediante diferentes protocolos, me ha servido para afianzar los conocimientos que había adquirido gracias a la carrera y además poder aumentarlos y llevarlos a la práctica mediante la realización de un servidor que trabaja con datagramas UDP y con los puertos serie como son los USB. Para la implementación de dicho servidor, he tenido que aprender el

lenguaje de programación Python, ya que con Java no fue posible su implementación en nuestra placa. Por último, la parte de Android ha sido otra de las partes que ha requerido muchas horas de investigación y búsqueda de documentación. En cuanto a la planificación inicial, se ha visto modificada sustancialmente debido al retraso ocasionado por la recepción de la placa controladora, sin la cual nos ha sido imposible llegar a montar el dron por completo y a realizar todas las pruebas del software en el propio dron. En cuanto al diseño del software y del hardware si han cumplido los plazos estimados inicialmente, hemos podido comprobar su funcionamiento con otros recursos, aunque lamentablemente no hayamos podido implementarlos con el montaje del dron. El diseño de la aplicación y del servidor UDP ha sido todo un éxito y ha superado las expectativas iniciales.

5.2. Objetivos marcados

Por la naturaleza del proyecto y de que hemos trabajado con tecnologías de las que hemos ido aprendiendo durante la realización de este, nos hemos encontrado con problemas inesperados que han ralentizado el progreso. El más destacable ha sido el retraso en la recepción de las piezas, al usar piezas complicadas de encontrar o que hay que realizar el pedido a zonas bastante lejanas nos ha supuesto no poder montar el dron en el plazo de tiempo establecido. A pesar de esto se han cumplido las funcionalidades necesarias para la comunicación mediante el software desarrollado y se ha diseñado el dron con todos sus componentes hardware.

Vamos a analizarlo mediante los objetivos expuestos al principio de este documento.

- Diseño y construcción de la estructura.
  Se buscó un diseño acorde a nuestras necesidades por internet, una vez localizado el diseño se guardo para su posterior impresión, pero desgraciadamente no nos ha sido posible imprimirlo debido al tiempo.

- Diseño de los algoritmos de control.
  Se han diseñado correctamente los algoritmos de envío para el software de control, sin necesidad de cambiar el software de control.

- Selección de la controladora.
  La selección de la controladora se realizó correctamente, aunque este apartado nos ha consumido mucho tiempo por culpa del envío, el cual tardó mucho más de lo esperado, lo que ocasionó retrasos importantes en todo el proyecto.

- Integración de los motores.
  La selección de los motores se realizó mediante unos cálculos teóricos para que fuesen capaces de llevar un cierto peso. La selección se realizó

correctamente aunque su implementación no por motivo del retraso de la controladora.

- Integración de la Raspberry.
  La integración de la placa ha sido todo un éxito.

- Diseño de la comunicación.
  El diseño de la comunicación ha sido de los procesos que más tiempo ha requerido, ya que primeramente se intentó realizar dicho diseño en Java, el cual al final no fue posible su implementación y hubo que buscar una solución. Esa solución fue Python, la cual nos sirvió para que este objetivo tuviese éxito.

- Diseño de la aplicación.
  El diseño de la aplicación ha sido todo un éxito.

- Integración de la aplicación.
  La aplicación ha sido probada en varios terminales Android con un resultado muy satisfactorio, la integración en el dron no ha sido posible por motivo de no tener el dron montado.

5.3. Lineas futuras

El diseño ha sido muy satisfactorio, tanto a nivel hardware como a nivel software. Pero se han encontrado una serie de puntos que deberemos realizar en un futuro:

- Montaje:
  Lamentablemente, por motivos de retraso en la recepción de ciertas piezas, no nos ha sido posible llegar a montar el dron. Lo cual con más tiempo sería el primer paso a seguir.

- Implementación:
  Implementar todo el código desarrollado y desarrollar todas las pruebas necesarias.

- Procesado de imagen y video:
  Implementar una cámara capaz de tomar imágenes y grabar video, además de procesarlo para diferentes aplicaciones, tales como videovigilancia, detección de situaciones de riesgo..

- Aumentar la capacidad de tiempo de vuelo:
  Al usar la Raspberry la capacidad de la batería se ve reducida drásticamente, por lo que deberíamos optimizar tanto la placa como las baterías utilizadas para conseguir el máximo tiempo de vuelo posible.

- Aumento de los dispositivos móviles compatibles:
  Desarrollo de la aplicación para el resto de plataformas actuales, como puede ser iOS o Windows Phone
- Control de la Raspberry Pi mediante el terminal Android:
  Implementar en la aplicación un serie de comandos mediante ssh para poder tener un mayor control de la placa en todo momento.

Bibliografía

[1] 3d robotics inc. https://3dr.com [Online].

[2] Adafruit. http://www.adafruit.com [Online].

[3] Aeroquad. http://aeroquad.com [Online].

[4] Android para desarrolladores. http://developer.android.com/index.html [Online].

[5] Ardrone. http://www.parrot.com/es/productos/ardrone-2/ [Online].

[6] Arduino. https://www.arduino.cc [Online].

[7] Atmega2560. http://www.atmel.com/devices/atmega2560.aspx [Online].

[8] Diy drones. comunidad para los uavs personales. http://diydrones.com [Online].

[9] Dji. http://www.dji.com/es [Online].

[10] Foro de multicopters. tema crius all in one pro. http://www.multicopters.es/foro/vbulletin/showthread.php?1068-Todo-sobre-la-Crius-AIO-Pro [Online].

[11] Hardkernel. http://www.hardkernel.com/main/main.php [Online].

[12] Instructables. http://www.instructables.com [Online].

[13] Librería rxtx de java. http://rxtx.qbang.org/wiki/index.php/Main_Page [Online].

[14] Modelo de aeroquad32. http://aeroquad.com/showwiki.php?title=AeroQuad32-Flight-Control-Board-v2 [Online].

[15] The multiwii serial protocol. http://www.stefanocottafavi.com/msp-the-multiwii-serial-protocol/ [Online].

[16] Open pilot. http://www.openpilot.org [Online].

[17] Panamahitek. http://panamahitek.com [Online].

[18] Pixhawk. https://pixhawk.org/choice [Online].

[19] Qué es xbee. http://xbee.cl/que-es-xbee/ [Online].

[20] Raspberry. http://www.raspberrypi.org/ [Online].

[21] Tipos de licencias. http://www.gnu.org/licenses/licenses.es.html [Online].

[22] Wikipedia. http://en.wikipedia.org [Online].

[23] Python Software Foundation. Tutorial sobre python. http://docs.python.org.ar/tutorial/3/index.html [Online].

[24] Raspberry Pi Foundation. Instalación noobs. https://www.raspberrypi.org/help/noobs-setup/ [Online].

[25] Chris Liechti. Libreria de python pyserial. https://pypi.python.org/pypi/pyserial [Online].

[26] Mario. Motores sin escobillas. http://www.neoteo.com/motores-brushless-bldc [Online].

[27] tweaking4all. Apple pi baker. http://www.tweaking4all.com/hardware/raspberry-pi/macosx-apple-pi-baker/ [Online].

[28] Waltr. Protocolo serie multiwii. http://www.multiwii.com/wiki/index.php?title=Multiwii_ [Online].