UNIVERSITAT POLITÈCNICA DE VALÈNCIA

etsinf

Escola Tècnica Superior d'Enginyeria Informàtica

Escola Tècnica Superior d'Enginyeria Informàtica
Universitat Politècnica de València

Diseño y Construcción de un Vehículo Aéreo no Tripulado Autónomo

TRABAJO FIN DE GRADO

Grado en Ingeniería Informática

Autor: Francisco Javier Giner Bellot

Tutor: Pascual Pérez Blasco

Curso 2017-2018

Resumen

Los Vehículos Aéreos No Tripulados (VANT) están tomando gran importancia en los últimos tiempos, ya que cada vez más se usan en un mayor número de aplicaciones. Debido a esto, este trabajo consiste en la construcción de un dron que permite realizar vuelos autónomos de forma que no es necesario que sea pilotado, pues se pueden planificar rutas y misiones de forma que se realizan sin la necesidad de controlarlo manualmente. Además, puesto que el rango de control de estos aparatos está limitado a unos pocos kilómetros, el objetivo también es la creación de otro método de comunicación con el dron que nos permita controlarlo a grandes distancias: la red móvil.

Para lograr esto, es necesaria la modificación de la aplicación elegida para Android con la que controlaremos el dron, Tower; así como la creación de un dispositivo que irá conectado al dron que envía y recibe los datos mediante una tarjeta SIM. Para conectar ambas partes se utiliza un servidor que hace de intermediario, a modo de puente.

Como resultado, el dron construido es capaz de realizar los vuelos de forma autónoma con éxito, aunque la comunicación vía red móvil sufre algunos problemas: el cuadricóptero y la aplicación comienzan a intercambiar información pero la conexión se pierde.

Palabras clave: dron, construcción, autónomo, ruta, Android, red móvil

Resum

Els Vehicles Aeris No Tripulats (VANT) estan agafant gran importància en els útims gemps, ja que cada vegada es fan servir en un major nombre d'aplciacions. Degut a açó, aquest treball consisteix en la construcció d'un dron que permet realitzar vols autònoms de manera que no cal ser pilotat, doncs es poden planificar rutes i missions de forma que es realitzen sense necessitat de controlar-lo manualment. A més a més, ja que el rang de control d'aquestos aparells es troba limitat a uns pocs kilòmetres, l'objectiu també es la creació d'un altre mètode de comunicació amb el dron que ens permeta controlar-lo a grans distàncies: la xarxa mòbil.

Per a conseguir açò, és necessària la modificació de l'aplicació triada per a Android amb la que controlarem el dron, Tower; així com la creació d'un dispositiu que anirà connectat al dron que envia i rep les dades mitjançant una targeta SIM. Per connectar dues parts s'utilitza un servidor que fa d'intermediari, com si fora un pont.

Com a resultat, el dron construït es capaç de realitzar vols de manera autònoma amb éxit, encara que la comunicació vía xarxa mòbil patix alguns problemes: el quadricòpter y l'aplicació comencen a intercanviar informació però la connexió es perd.

Paraules clau: dron, construcció, autònom, ruta, Android, xarxa mòbil

Abstract

Unmanned Aerial Vehicles (UAV) are taking great importance in recent times, because they are being used in a greater number of scenes. For this reason, this project consists in the building of a drone that allows autonomous flights, as it allows to schedule routes and missions without the need to control it manually. Furthermore, since the control range of these vehicles is limited to few kilometers, the goal of this project is also the creation of a new method of communication with the drone that allows us to control it over long distances.

In order to achieve that, it is necessary to modify the Android application chosen to control the drone, Tower; as well as the creation of a device connected to the vehicle that sends and receives the data through a SIM card. To connect both parts a server is used as an intermediary, as a bridge.

As a result, the drone is capable to perform autonomous flights successfully, although the communication through the mobile network has some problems: the information exchange starts, but then the connection is lost.

Key words: dron, building, autonomous, route, Android, mobile network

Índice general

Índice general v
Índice de figuras vii

1 Introducción 1
1.1 Motivación . . . 1
1.2 Objetivos . . . 1
1.3 Estructura de la memoria . . . 1

2 Estado del arte 3
2.1 Proyectos similares . . . 3
2.1.1 MultiWii . . . 3
2.1.2 Baseflight y Cleanflight . . . 4
2.1.3 YMFC-32 . . . 5
2.1.4 ArduPilot . . . 5
2.1.5 Raspberry Pi 3 + Navio2 + Mando XBOX (conexión vía 4G) . . . 6

3 Tipos de dron y aplicaciones 7
3.1 Dron de ala fija . . . 7
3.2 Helicóptero . . . 8
3.3 Multirotor . . . 8

4 Partes de un dron 11
4.1 Chasis . . . 11
4.1.1 Material . . . 11
4.1.2 Configuración (multirotor) . . . 12
4.2 Batería . . . 13
4.3 Hélices . . . 13
4.4 Motores . . . 14
4.5 Variadores . . . 15
4.5.1 Tipos de ESC . . . 16
4.5.2 Firmware y protocolo de comunicación . . . 17
4.6 Sensores . . . 18
4.6.1 Acelerómetro . . . 18
4.6.2 Giroscopio . . . 19
4.6.3 Magnetómetro . . . 20
4.6.4 Barómetro . . . 20
4.6.5 IMU . . . 21
4.7 Controladora de vuelo o Unidad de Control . . . 22
4.7.1 Control PID . . . 22
4.7.2 Comunicación Serie . . . 23
4.7.3 I^2C . . . 23
4.7.4 Protocolo MAVLink . . . 24

5 Construcción del dron 27
5.1 Elección de las piezas . . . 27
5.2 Montaje del dron . . . 28

5.3 Pasos iniciales . . . 30
5.3.1 Configuración de la emisora HK-T6A V2 . . . 30
5.3.2 Configuración de la controladora de vuelo . . . 31
5.3.3 Primeros vuelos . . . 34
6 Comunicación vía red móvil **35**
6.1 Programa en el servidor . . . 35
6.1.1 Configuración previa del servidor . . . 35
6.1.2 Programa en el servidor: primera versión . . . 37
6.1.3 Programa en el servidor: segunda versión . . . 38
6.2 Envío de la telemetría: Arduino . . . 40
6.2.1 Construcción . . . 40
6.2.2 Probando el módulo SIM808 . . . 42
6.2.3 Programando Arduino: la librería TinyGsm . . . 43
6.3 Modificación de la GCS . . . 45
6.4 Resultados . . . 46
7 Presupuesto **49**
7.1 Coste material . . . 49
7.2 Coste humano . . . 50
8 Conclusiones **51**
Bibliografía **53**

Apéndice
A Glosario **61**
Siglas . . . 61

Índice de figuras

2.1 Interfaz MultiWiiConf . . . 4
2.2 Configurador de Baseflight . . . 4
2.3 Configurador de Cleanflight . . . 5
2.4 Proyecto YMFC-32 finalizado . . . 5
2.5 Logo de ArduPilot . . . 6
2.6 Navio2 montada en una Raspberry Pi 2 . . . 6

3.1 Dron de ala fija usado en fotogrametría . . . 7
3.2 Helicóptero . . . 8
3.3 Hexacóptero utilizado en fotografía aérea . . . 8

4.1 Configuraciones de multirotores más comunes . . . 13
4.2 Diagrama de funcionamiento de un motor con escobillas y uno sin escobillas 14
4.3 Fases a activar en un motor brushless según la posición del rotor detectada 15
4.4 ESC de 60A con BEC . . . 15
4.5 Pulso Pulse Width Modulation (PWM) para controlar un servo con un ciclo de trabajo del 0 %, 50 % y 100 % . . . 18
4.6 Esquema del funcionamiento de un acelerómetro capacitivo . . . 19
4.7 Una rotación del disco suficientemente rápida hace mantener la dirección de giro, cambiando la posición del marco . . . 19
4.8 Disco rotatorio de un giroscopio MEMS . . . 20
4.9 Brújula común . . . 20
4.10 Presión en función de la altura sobre el nivel del mar . . . 21
4.11 IMU GY-80 . . . 21
4.12 Diagrama de bloques de un controlador PID . . . 22
4.13 Ejemplo de conexión de un bus I^2C con tres esclavos y dos maestros . . . 24
4.14 Diferencias en la estructura de un paquete en MAVLink 1 y MAVLink 2 . 25
4.15 Mensaje *REQUEST_DATA_STREAM* definido en el fichero *common.xml* . . 26

5.1 Pack que incluye un APM2.6, GPS y soportes . . . 27
5.2 Pack que incluye un APM2.6, GPS y soportes . . . 27
5.3 Arduino DUE . . . 28
5.4 Módulo SIM808 . . . 28
5.5 ESC con los conectores soldados . . . 29
5.6 Estado de montaje del dron con los motores y ESC instalados . . . 29
5.7 . . . 30
5.8 Emisora FlySky i6 . . . 30
5.9 T6config, la utilidad para configurar nuestra emisora . . . 31
5.10 Modelo que configuraremos . . . 31
5.11 Mission Planner: Selección del tipo de vehículo . . . 32
5.12 Mission Planner: Instalación del firmware en la placa . . . 32
5.13 Mission Planner: Calibración de acelerómetro y magnetómetro . . . 33

6.1 Lista de plantillas a escoger . . . 35

6.2 Versión inicial del desarrollo de las comunicaciones 37
6.3 Código IMEI en un módulo SIM808 38
6.4 Segunda versión del transcurso de las comunicaciones 39
6.5 Segunda versión del transcurso de las comunicaciones 40
6.6 Ampliación del puerto de telemetría 41
6.7 Conector 41
6.8 Conexiones del dispositivo que nos permitirá la comunicación via red móvil 42
6.9 Realización de una llamada de voz mediante comandos AT 43
6.10 Bucle que comunica el puerto serial del APM con nuestro servidor. 44
6.11 Salida que genera nuestro programa de Arduino. 44
6.12 Ejemplo de conexión Transmission Control Protocol (TCP), donde son necesarios varios dispositivos 45
6.13 A la izquierda, opciones de conexión por defecto. A la derecha, la nueva opción de conexión añadida. 46
6.14 Intento de conexión de la GCS con el dron desconectado 47
6.15 Únicamente los primeros datos son mostrados correctamente 47

CAPÍTULO 1
Introducción

1.1 Motivación

Es un hecho indiscutible que, en los últimos años, los Vehículos Aéreos No Tripulados (VANT) o en inglés Unmanned Aerial Vehicle (UAV) han experimentado un gran crecimiento en cuanto a su popularidad, puesto que no solo se les da un uso recreativo sino que, cada día más, comienzan a ser usados en todo tipo de tareas.

Ya es posible contemplar drones tomando planos con cámaras en conciertos y espectáculos, en la gestión agrícola, lanzando salvavidas a personas que necesitan ayuda en el agua, ayudando a localizar personas perdidas o incluso la Dirección General de Tráfico (DGT) los está comenzando a usar en la vigilancia de carreteras y conductores.

Este aumento en su uso obedece al hecho de que ya es muy barato construir estos artilugios y a la mayor permisividad existente respecto a la posibilidad de operar en distintas zonas y alturas con gran facilidad. También son muy útiles gracias a su característica principal: no son tripulados, con lo cual son perfectos en zonas o ambientes peligrosos (radiactivos, tóxicos, etc.) manejados a distancia, e incluso pueden realizar viajes de manera autónoma.

1.2 Objetivos

Los objetivos de este trabajo son, por orden de relevancia:

1. Construcción de un VANT
2. Que el vehículo pueda realizar vuelos autónomos, siguiendo una ruta definida por el usuario.
3. Hacer posible la comunicación usando sólo la red móvil, permitiéndonos usarlo a grandes distancias.

1.3 Estructura de la memoria

La estructura seguida por esta memoria comienza con el estado del arte, repasando los actuales proyectos semejantes que sirven de base para alcanzar los objetivos. Posteriormente, se detallan los distintos tipos de dron que existen así como las partes comunes a estos que hacen posible su funcionamiento. En el siguiente capítulo se trata la construcción del dron base: la elección de las piezas, su montaje y la configuración inicial.

Después se aborda la comunicación vía red móvil: el despliegue del servidor, la aplicación que controla el dron así como el dispositivo que se encarga de enviar y recibir los datos a través de la red móvil. En los dos capítulos siguientes se encuentran el presupuesto y las conclusiones obtenidas, donde se explican los resultados del proyecto. Seguidamente se encuentran las fuentes más representativas de la bibliografía utilizada, así como un apéndice con varios códigos fuente. Por último, se puede encontrar un glosario con definiciones de conceptos y siglas (marcadas en rojo) que pueden ser consultadas en cualquier momento durante la lectura de este documento.

CAPÍTULO 2
Estado del arte

Actualmente hay numerosos proyectos que nos permiten construir vehículos aéreos no tripulados, algo que puede resultar confuso al principio por no saber qué hardware o software escoger. A continuación, veremos algunos proyectos similares de los que podremos partir para realizar nuestro trabajo, de forma que no se pierda el tiempo en partes que ya están desarrolladas y con una gran cantidad de desarrolladores detrás.

2.1 Proyectos similares

2.1.1. MultiWii

MultiWii nació como un proyecto hecho con un Arduino Pro mini y un *Wii Nun chuck* junto con un *Wii Motion Plus*, un accesorio para mejorar la precisión en la detección de los movimientos del *Wii Mote*, el control remoto para la consola Wii. De esta forma, se creó un controlador de vuelo sencillo y barato, además de original.

El proyecto, que en la actualidad es de código abierto bajo licencia GPL, ha madurado mucho, soportando distintas configuraciones de multicóptero (bicóptero, tricóptero, cuadricóptero, hexacóptero, y octacóptero, con distintos modos de volar), diversos sensores (acelerómetros, barómetros, giroscopios, GPS, sonar, etc.), ESCs, así como otros microcontroladores o MCU (Microcontroller Unit) y funciones para usar Firs Person View (FPV) o gimbal. Se puede usar comprando una placa con todos los sensores, conexiones y software incluidos, o subiendo el código al microcontrolador que deseemos y haciendo las conexiones necesarias con el hardware que deseemos, siempre que sea compatible.

La ventaja de este proyecto es que, entre otras placas, soporta el Arduino UNO, Arduino Mega y también el Arduino DUE, que aunque no dispone de una memoria EEPROM en la que poder guardar datos no volátiles de configuración, el código de este proyecto hace uso de unas librerías que emulan esa función con la memoria flash de esta placa.

Como vemos en la figura 2.1, tiene un software para ajustar diversos parámetros de vuelo del dron y comprobar los datos obtenidos por los sensores.

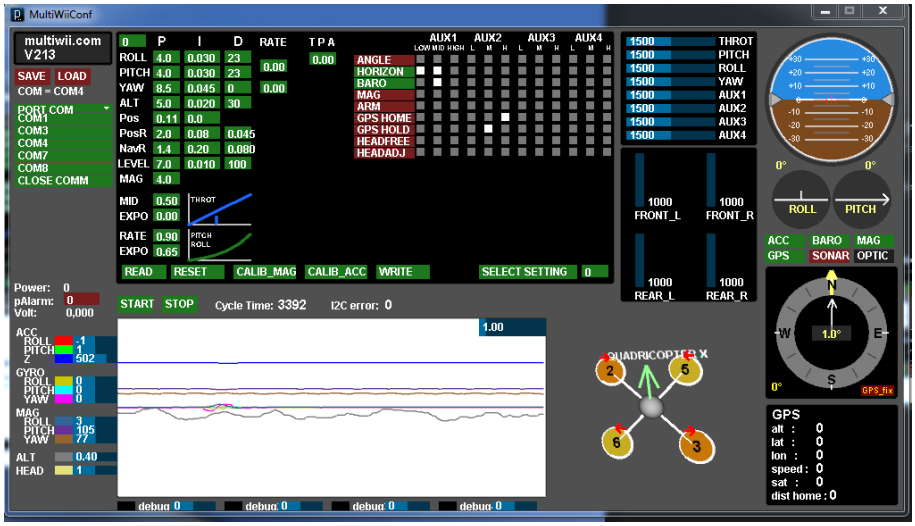


Figura 2.1: Interfaz MultiWiiConf

2.1.2. Baseflight y Cleanflight

Baseflighy nace como un fork de MultiWii bajo licencia GPLv3 (al igual que MultiWii). El fork se hace con la intención de dar soporte a placas con procesadores de 32 bits, dejando atrás las de 8 bits. El problema que existía era que no soportaba muchas placas controladoras de vuelo. Además, el código no era claro y por tanto era más difícil de mantener.

Así pues, Cleanflight se creó partiendo del proyecto Baseflight, con un código mejor estructurado, corrección de muchos bugs, soporte para más placas, controladores PID adicionales, soporte para tiras de led multicolor, pantallas OLED, o la corrección del PID durante el vuelo entre otras características.

La historia detrás de estos dos proyectos es larga, ya que ha habido acusaciones de plagiar y posteriormente registrar los derechos de la extensión de Chrome para configurar la placa (figuras 2.2 y 2.3) y se ha creado una especie de rivalidad entre los creadores, aunque hay gente que afirma que esto ayudará a que los dos proyectos avancen mejor.

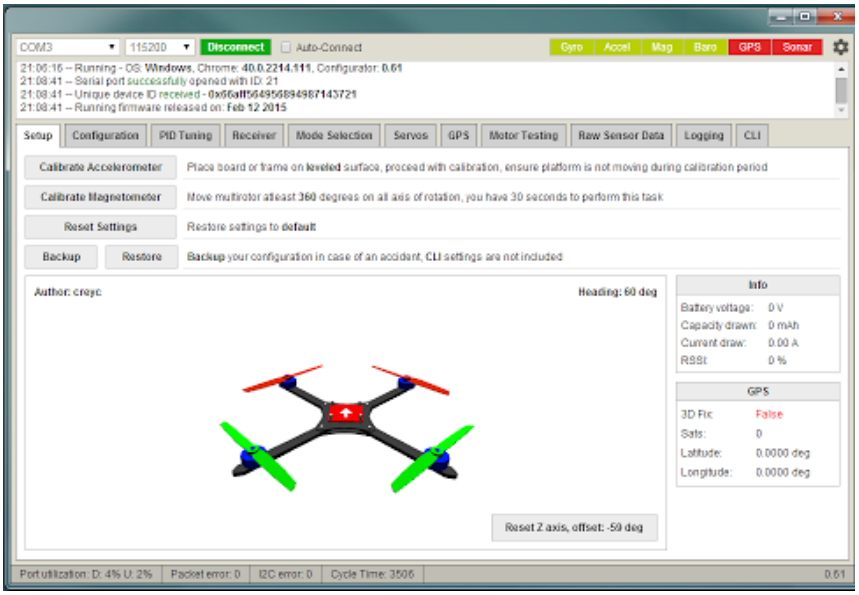


Figura 2.2: Configurador de Baseflight

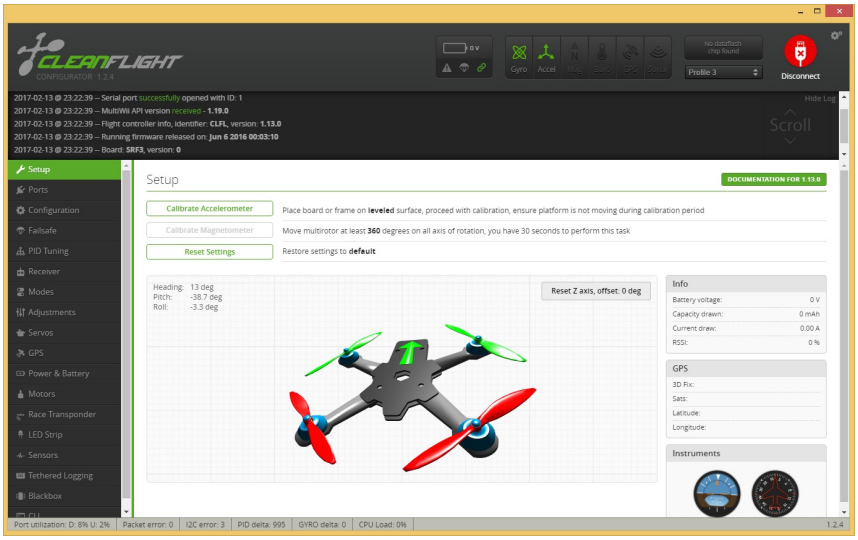


Figura 2.3: Configurador de Cleanflight

A su vez, el proyecto Cleanflight ha dado lugar a otros forks como son Betaflight o iNav.

2.1.3. YMFC-32

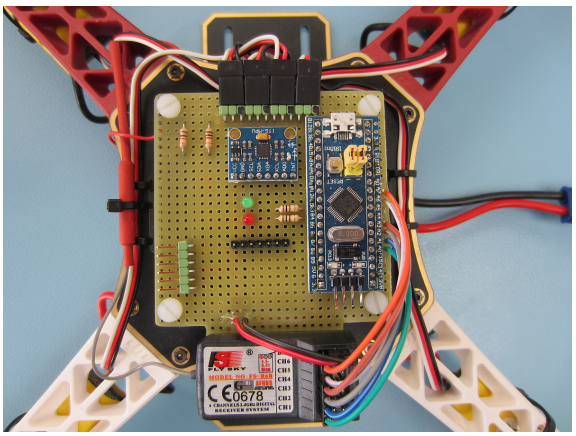


Figura 2.4: Proyecto YMFC-32 finalizado

El proyecto YMFC-32 está basado en la placa microcontroladora STM32 y es de muy temprana edad, ya que la primera versión 0.1 se publicó en enero de este mismo 2018. La placa microcontroladora se puede programar desde el IDE de Arduino (junto a un add-on) y, aunque hay algunas diferencias en cuanto a programación, estas son mínimas.

Con este proyecto podremos construir fácilmente un cuadricóptero barato (su coste rondaría los $=125) y, como en su misma página web informan, es un proyecto básico cuya finalidad es proporcionar el código mínimo y entendible para construir un controlador de vuelo para cuadricóptero. Debido a la temprana edad del proyecto y su tamaño, el hardware que soporta es bastante limitado, aunque nos puede ayudar para introducirnos en el proceso de crear un cuadricóptero propio.

Hay otra versión de este proyecto, llamado YMFC-AL que implementa la función Auto Leveling, es decir: cuando se sueltan los sticks, el cuadricóptero se nivelará automáticamente.

2.1.4. ArduPilot

Es un sistema de piloto automático *Open Source* creado y mantenido por una gran comunidad de desarrolladores. Tiene varios firmwares para distintos tipos de vehículos, como lo es APM Copter, que es compatible con helicópteros tradicionales y multicópteros; APM Plane, compatible con aviones de ala fija; APM Sub, vehículos submarinos;

APM Rover, para vehículos terrestres y barcos.
Además, dispone de varios software de estación de control para pc y dispositivos móviles y tabletas.

La ventaja que tiene es que al ser una comunidad tan grande, varias empresas comercializan hardware ya preparado y cargado con el software para conectar los componentes y comenzar a volar, como son la placa PixHawk, HKPilot Mega o ArduPilot Mega (APM) basada en un Arduino Mega, aunque las nuevas versiones de ArduPilot ya no son compatibles desde la versión Copter 3.3-rc1 del 11 de Abril de 2015 (así como otras placas con procesadores de 8 bits) cuando actualmente van por la versión Copter 3.5.5-rc1 del 24 de Enero de 2018.

Figura 2.5: Logo de ArduPilot

2.1.5. Raspberry Pi 3 + Navio2 + Mando XBOX (conexión vía 4G)

En este proyecto se usa el código de ArduCopter 3.4, en una Raspberry Pi 3 complementada con la placa controladora de vuelo Navio2 (sucesora de la Navio+). Navio2 es un Hardware Attached on Top (HAT), es decir, es un hardware que se conecta en la Raspberry para extender o aumentar las capacidades de la Raspberry (en Arduino reciben el nombre de *Shields*). Se usa en proyectos que necesiten ser pilotados, pues incluye todos los sensores necesarios para ello y es compatible con la versión de ArduPilot para Linux. Su precio es algo elevado, ya que solamente la placa sin incluir la Raspberry, ronda los 200 €.

Como en el vídeo[1] nos muestra el autor, el portátil se encarga de recibir los datos enviados vía Internet por el dron, y de enviarle a éste los datos del mando de Xox.

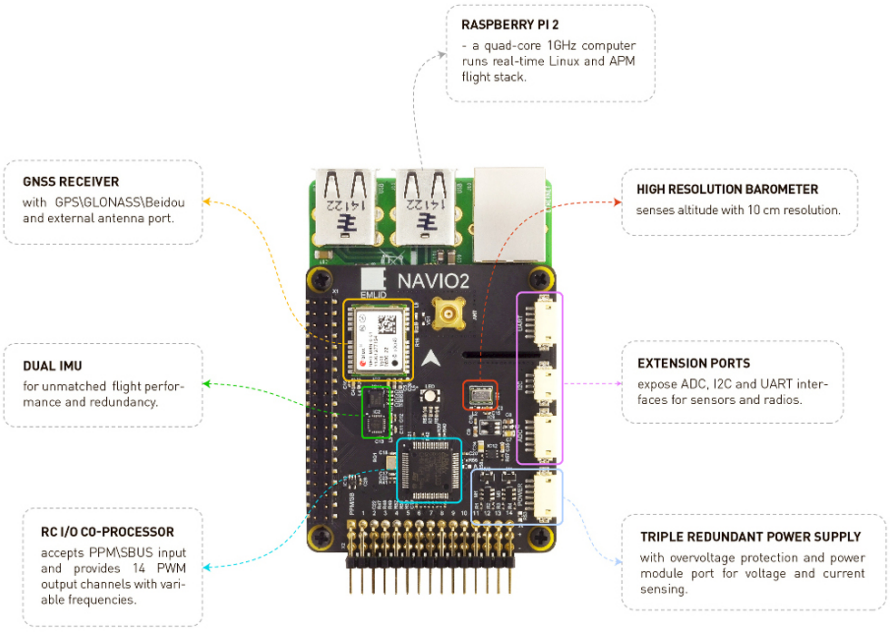


Figura 2.6: Navio2 montada en una Raspberry Pi 2

[1] https://www.youtube.com/watch?v=MmcBcOFGWd0

CAPÍTULO 3
Tipos de dron y aplicaciones

No todos los drones siguen el mismo patrón. Hay tres familias o tipos que definirán la estructura, forma de volar y los usos que le podemos dar. Así pues, se puede clasificar un dron como: de ala fija, o de ala rotatoria, donde a su vez se distinguen los helicópteros y los multirotores. ¿Pero qué diferencias hay entre ellos?

3.1 Dron de ala fija

Los drones de ala fija tienen aspecto de dron de avión de aeromodelismo. Son los más eficientes, ya que solo utilizan un motor con una hélice junto a dos servos que ajustan la posición de las aletas que nos darán el giro. Además, tienen forma aerodinámica, lo que les permite planear sin consumir apenas energía y dando opciones de seguir maniobrando si el motor principal falla.

Su gran desventaja es que necesitan un largo espacio plano y sin obstáculos para poder despegar y aterrizar, ya que no pueden ascender verticalmente. Además, no pueden quedarse fijos en un punto de forma estacionaria, por lo que son descartados para algunos trabajos.

Son muy recomendados cuando se necesita recorrer mucho terreno y disponer autonomía, como en labores de fotogrametría[1] y agricultura de precisión[2].

Figura 3.1: Dron de ala fija usado en fotogrametría

[1] La fotogrametría es la técnica para obtener las dimensiones y forma de un objeto o terreno mediante fotografías. Los drones de ala fija son muy útiles para generar mapas y planos de terrenos mediante fotografías aéreas.

[2] La agricultura de precisión permite optimizar la gestión de una parcela agrícola, ayudar a la toma de decisiones y mejorar la calidad del producto, ya que podemos conocer o predecir las necesidades de la planta o la densidad óptima de siembra.

3.2 Helicóptero

Un dron de tipo helicóptero, es como un helicóptero tradicional. Posee un motor principal con una hélice grande y otro rotor en la cola. También son más eficientes que los multirotores, ya que no varían las revoluciones del motor para moverse, sino el paso de las hélices por lo que usan menos energía que un multirotor.

Figura 3.2: Helicóptero

Este tipo de drones sí pueden mantener la posición del punto en el que se encuentran y, ante el efecto de rachas de aire, es más estable que un multirotor. Estos factores, junto a la gran capacidad de carga de la que disponen, los hacen una herramienta perfecta en trabajos audiovisuales.

Tradicionalmente, era más difícil de manejar que los multirotores, aunque puesto que los helicópteros ya incorporan placas controladoras, sensores y GPS, por lo que no difiere mucho del control de un multirotor. En cambio, su mecánica es algo compleja, por lo que necesita revisiones y mantenimiento regular.

3.3 Multirotor

Un multirotor es el primer vehículo en el que piensa la mayoría de gente cuando escucha la palabra *dron*, debido a su reciente gran aumento de popularidad, ya que según los aficionados que pilotan los tres tipos de drones, es el tipo de dron que antes podemos aprender a pilotar a nivel básico.

Figura 3.3: Hexacóptero utilizado en fotografía aérea

Generalmente, consta de una parte central de la cual salen los brazos y, en los extremos de los brazos, se encuentran los motores con las hélices, de forma que todos están a la misma distancia del centro de gravedad del dron para hacer que sea estable. Con más brazos ganaremos más estabilidad, aunque necesitaremos más motores y, por tanto, el consumo crecerá.

Esta capacidad que tienen los multirotores de mantener la posición de manera estacionaria, al igual que los helicópteros, los hace excelentes en producciones audiovisuales. Por contra, al usar

más motores generalmente el consumo será más elevado que en los otros tipos de dron y por tanto poseen menos autonomía.

CAPÍTULO 4
Partes de un dron

Son muchas las posibles configuraciones y elementos que podemos tener en nuestro dron, aunque hay elementos básicos que son indispensables en cualquier dron. Se tratarán en las siguientes secciones.

4.1 Chasis

El chasis (*frame* en inglés) es la estructura en la que se montarán el resto de componentes, la que soportará el peso. Definirá en gran medida la forma de nuestro vehículo. Hay algunas partes importantes a tener en cuenta, según la aplicación que vaya a tener.

4.1.1. Material

Como ya hemos dicho, el frame debe estar pensado para soportar una determinada carga sin sufrir deformaciones ni vibraciones, ya que, si las hay, la estabilidad del dron se verá afectada y algún componente puede sufrir daños. Además si se utiliza algún sistema de obtención de imágenes, éstas no serán de calidad. También conviene que sea ligero para reducir el consumo y que sea resistente a golpes y caídas.

Así pues, dependiendo del uso final del dron, los principales materiales que en los chasis suelen predominar son:

- POLIPROPILENO EXPANDIDO (PPE). Es común escucharlo simplemente como corcho. Es un material muy ligero y barato. Puede verse en drones de ala fija baratos, así como en algún multirotor casero.
- MADERA. Aunque no muy estética, para un dron es un material muy barato y fácil de reponer. No debe ser una madera muy pesada aunque, como hemos dicho anteriormente, no ha de tener vibraciones y soportar bien las fuerzas de torsión. Es ideal para quienes necesitan un multirotor barato.
- PLÁSTICO Y FIBRA DE VIDRIO. Son baratos y resisten algunos golpes, aunque para drones de gran tamaño no se usan debido a que no soportan grandes fuerzas de flexión y torsión.
- ALUMINIO. Su peso es un poco más elevado, pero no produce casi vibraciones y es bastante resistente. Aunque no suele quebrarse o partirse, sí puede deformarse.
- FIBRA DE CARBONO. Se considera el material más acertado ya que es muy ligero y a la vez resistente, aunque su precio es más elevado. Su desventaja es que al ser conductiva puede interferir con las ondas de radiofrecuencia (RF).

- G10, es una variación de la fibra de vidrio. Se forma con láminas de fibra de vidrio apiladas y después se sumerge en resinas epoxi, quedando posteriormente endurecidas al aplicarle calor. Este material se usa como una alternativa más barata de la fibra de carbono, y que en cambio no produce interferencias con las señales de RF.

En los casos en los que el dron necesite ocupar el espacio mínimo cuando no está en uso, también se ha de considerar la portabilidad del vehículo, es decir, si éste puede plegarse para ocupar menos espacio, doblando o recogiendo los brazos, alas o cola (en el caso de multirotores, drones de ala fija o helicópteros respectivamente) ya sea de forma manual o automática.

4.1.2. Configuración (multirotor)

En los multirotores, otro punto a tener en cuenta es la configuración de los motores. Número de brazos, de motores, cómo están dispuestos... Podemos distinguir dos grupos de multirotores: los clásicos, con un motor por cada brazo o eje; y los coaxiales, que tienen dos motores por brazo.

El prefijo de la configuración (a menudo se usa la forma en inglés) nos indica la cantidad de motores que el multirotor dispondrá, por ejemplo, un hexacóptero dispondrá de seis motores, uno en cada uno de los seis brazos, o dos en cada uno de los tres brazos si el hexacóptero es coaxial. Hay gran variedad de configuraciones, desde alguna más simple como un bicóptero hasta alguna más compleja como octacóptero. Además, para cada configuración se podría volar con alguna variación en cuanto a la orientación de vuelo.

La configuración más común es de cuadricóptero ya que es la más barata y menos difícil de construir. En este caso se volar en forma de «X», es decir dos motores delante y dos detrás; en forma de «+», un motor en la parte delantera, dos a cada lado de parte central, y otro en la parte trasera; o en forma de «H» girada.

Cuantos más motores hayan, el dron dispondrá de mayor potencia y velocidad, así como de mayor carga útil y estabilidad (permitiendo seguir funcionando incluso si algún motor sufre algún daño y deja de funcionar), pero a cambio de un mayor consumo y un precio bastante más elevado.

La forma de los brazos o la estructura que los forma también son de gran importancia, por ejemplo si se han de soportar cargas muy grandes son preferibles brazos formados por secciones tubulares (cuadradas o redondas) a diferencia de los brazos con refuerzos triangulares.

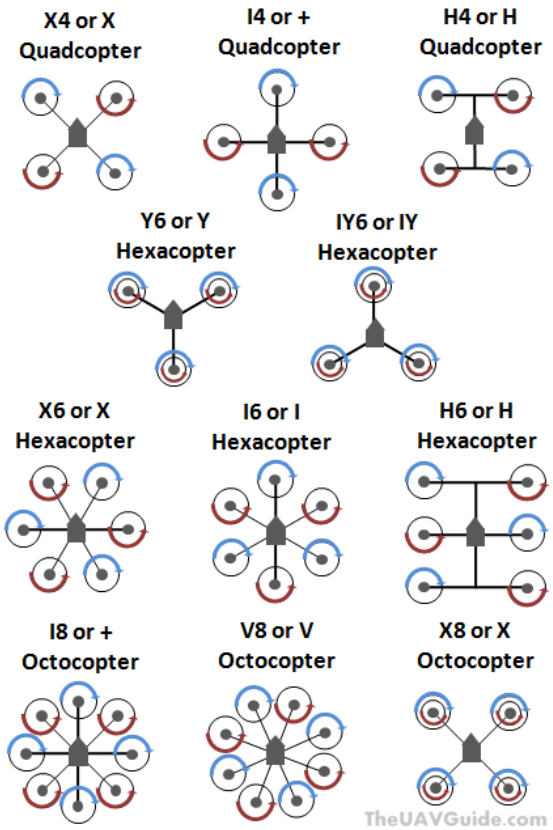


Figura 4.1: Configuraciones de multirotores más comunes

4.2 Batería

El elemento que nos proporcionará la energía para alimentar los circuitos de nuestro dron es la batería.

Las baterías más comunes son las LiPo (Polímero de iones de Litio), aunque también existen baterías NiMh o NiCd. Las más usadas en este tipo de proyectos son las baterías de polímero de Litio, ya que tienen una mayor densidad de energía, es decir, se puede almacenar más energía con un menor espacio y peso. Además, también tienen una alta tasa de descarga (C), esto es, la velocidad con la que se puede descargar (o intensidad máxima que puede dar en un momento puntual) sin que haya riesgos de dañar la batería.

4.3 Hélices

Son los elementos encargados de generar el empuje mediante la rotación generada por los motores, que permitirá al dron despegar y desplazarse. Según el sentido de giro del motor, horario o antihorario, habrá que poner la hélice adecuada hecha para que al girar genere el empuje hacia abajo.

Suelen estar fabricadas con materiales como el plástico o nylon, aunque también hay de mayor calidad como la fibra de carbono.

Las características básicas que diferencian una hélice de otra son: el tamaño de la hélice de punta a punta, o diámetro de la circunferencia al girar (suelen expresarse en milímetros o pulgadas); el número de aspas, que generalmente son dos; el ángulo de las aspas, que determinarán el paso de la hélice, es decir, la distancia recorrida por la

hélice en una vuelta completa; y por último, la forma de las aspas, que hacen cambiar la superficie de contacto con el aire.

El consumo, el empuje generado y la forma de volar variará según las características de la hélice, por ejemplo las aspas acabadas en punta tienen una mejor relación consumo-empuje; cuanto más tamaño o más paso, más cantidad de aire mueve la hélice y más empuje genera (por tanto mayor velocidad), pero los motores necesitan más trabajo y tendremos un mayor consumo. Como podemos ver, hay muchas configuraciones según las necesidades y el uso que vaya a tener el dron.

4.4 Motores

Los motores transforman la energía eléctrica en movimiento circular que se transmitirá a las hélices. En multirotores se recomienda el uso de motores sin escobillas o *brushless*(trifásicos) ya que ofrecen ventajas importantes respecto a los motores con escobillas o *brushed*.

Los motores con escobillas (o bifásicos) disponen de un imán fijo en el estator. Después se le transmite una corriente que llega a las escobillas, la parte que está en contacto con el conmutador que está fijo en el rotor, transfiriéndole la corriente y creando un campo magnético en las bobinas del armazón del rotor.

Debido a este constante contacto de las escobillas con el conmutador que está girando, las escobillas van degradándose y pudiendo soltar restos que conviene limpiar regularmente ya que podrían dañar o reducir la eficiencia del motor.

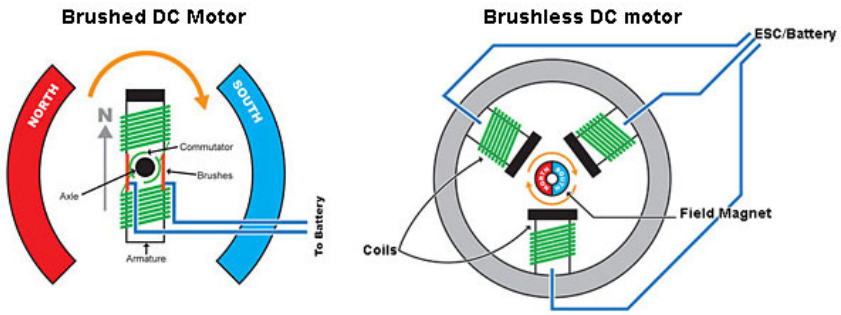
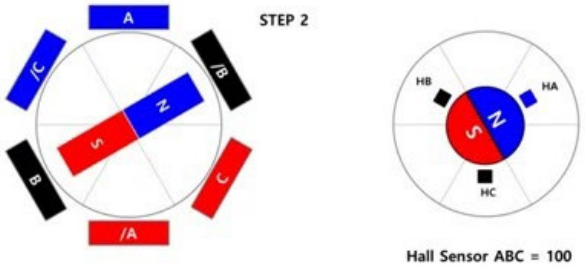


Figura 4.2: Diagrama de funcionamiento de un motor con escobillas y uno sin escobillas

En cambio, en los motores brushless (trifásicos) no se emplean escobillas para realizar el cambio de polaridad que le permite rotar, librándose así de generar tanto rozamiento debido a las escobillas, y por tanto, son más eficientes. La corriente pasa directamente por las bobinas que se encuentran estator, se generando un campo magnético que interaccionará con los imanes fijos del rotor. Esto también permitirá que se disipe mejor el calor, ya que las bobinas se encuentran en la parte exterior. Así pues, si se hace pasar la corriente por las bobinas adecuadas y lo hacemos en el momento adecuado lograremos que el motor gire a la velocidad deseada.

Algunos de estos motores disponen de sensores de efecto Hall, unos sensores que miden o detectan campos magnéticos, para así poder determinar en qué orientación se encuentra el rotor.



Step	Hall Sensor			Motor Phase		
	HA	HB	HC	A	B	C
1	1	1	0	OFF	+	-
2	1	0	0	+	OFF	-
3	1	0	1	+	-	OFF
4	0	0	1	OFF	-	+
5	0	1	1	-	OFF	+
6	0	1	0	-	+	OFF

Figura 4.3: Fases a activar en un motor brushless según la posición del rotor detectada

Una gran desventaja que tienen los motores sin escobillas es que necesitan un elemento externo para controlar estas tres fases de motor, un variador o Electronic Speed Controller (ESC) (sección 4.5), superando en algunas ocasiones el precio del motor.

En la siguiente tabla podemos ver una comparativa de algunas propiedades entre los dos tipos de motores.

Propiedad \ Tipo de motor	Sin escobillas	Con escobillas
Potencia de salida / Tamaño	Menor	Mayor
Peso	Menor	Mayor
Calor desprendido	Menor	Mayor
Ruido producido	Menor	Mayor
Mantenimiento	Poco	Mucho
Precio	Mayor	Menor
Control	Mayor complejidad	Menor complejidad
Vida útil	Mayor	Menor
Rango de velocidad	Mayor	Menor
Relación velocidad/par motor	Mejor	Peor (a mayor velocidad, mayor fricción)
Inercia	Baja	Alta

Tabla 4.1: Comparativa ente un motor brushless y brushed

4.5 Variadores

Un variador, controlador electrónico de velocidad, o en inglés ESC es el componente electrónico que sirve para controlar la velocidad de giro de un motor trifásico o burshless.

Se alimenta con dos cables provenientes de la batería, uno para el polo positivo y otro para el negativo. Como salida dispondremos de tres cables a los que deberemos conectar el motor. Si deseamos cambiar el sentido de giro sólo

Figura 4.4: ESC de 60A con

debemos intercambiar las conexiones de dos cables, no importa cuáles. Así lograremos cambiar el orden de dos de las fases, con lo cual el sentido se invierte.

Además, el ESC también incluirá un conector de servo, normalmente con los colores blanco, negro y rojo. El cable blanco se usa para recibir la información de control, el cable negro se deberá conectar a tierra y el cable rojo se usará para la alimentación +5V aunque dependiendo del tipo de ESC encontraremos algunas diferencias, es por ello que algunos incluso no lo incorporan.

La propiedad del ESC más importante que debemos tener en cuenta es el amperaje soportado por éste. Debemos consultar el consumo de los motores en su máxima potencia según el voltaje suministrado y las hélices instaladas (recordemos que cuanto mayor sea el diámetro de las hélices mayor será el consumo). Esta información suele encontrarse en el manual proporcionado por el vendedor de los motores, y aunque a veces son muy poco claras pueden orientarnos. El amperaje soportado por el ESC debe ser como mínimo igual a este consumo máximo, aunque se recomienda dejar un poco de margen.
No hay problema en usar un ESC que soporte un amperaje mayor, pero es un gasto innecesario de tamaño, peso y dinero.

4.5.1. Tipos de ESC

Cuando elegimos un variador, a parte del amperaje que soporta, también es importante conocer de qué tipo es, ya que no funcionan del mismo modo.

ESC con BEC lineal

La gran mayoría de variadores incorporan un dispositivo llamado Battery Elimination Circuit (BEC) que le permitirá proporcionar 5V a través del cable rojo del conector de servo. Un BEC es básicamente un regulador de tensión, de forma que nos proporciona un flujo de corriente estable para alimentar por ejemplo la placa controladora, el receptor RC o los servos.

Los BECs lineales reducen el exceso de voltaje de entrada hasta los 5V en forma de calor mediante resistencias. Es por ello que es recomendable para baterías LiPo de hasta tres celdas (3S, 11.1V), cuantas menos celdas son más eficientes debido a que hay menos diferencia de voltaje. Para baterías de más de tres celdas no son recomendados, pues debido a que tienen mayor voltaje se desperdicia mucha energía en forma de calor. En su lugar deberemos usar un ESC con BEC conmutado.

ESC con BEC conmutado

Este BEC mantiene la eficiencia constante sin importar el voltaje. Esto se logra debido a que funcionan con un regulador de voltaje conmutado.
El conmutador (que básicamente puede ser un transistor, aunque pueden usar otros elementos más eficaces que no adentraremos debido a que no incumbe a este trabajo) va conmutando entre corte y saturación, zonas de trabajo donde se disipa muy poca potencia, por eso son tan eficientes. Esto se hace mediante un pulso PWM. La anchura del ciclo de trabajo definirá el voltaje final de salida: cuanto más ancho sea, mayor voltaje obtendremos.

De esta forma, integrado en el ESC obtendremos 5V, pero podremos trabajar con voltajes mayores (por ejemplo con una batería de 6 celdas) y proporcionar más corriente,

todo esto desperdiciando menos energía en forma de calor por lo que el ESC no se calentará tanto.

ESC Optoacoplado

Los variadores optoacoplado, o simplemente variador opto, separa el circuito de control del circuito de potencia. Así pues, la señal eléctrica se transmite mediante un emisor de luz que detectará el receptor. De esta forma se consigue separar eléctricamente los dos circuitos, evitando que cada circuito genere ruido en el otro.

Evidentemente no incorporan BEC, ya que esto comunicaría los dos circuitos y no tendría sentido usarlo en este tipo de ESC, ya que se desea que ambos circuitos estén separados. Es por ello que el cable rojo del conector de servo no saca +5V. Así pues, al no disponer de un BEC deberemos usar uno externo, reduciendo el tamaño y peso del variador.
Cuando el BEC es externo también puede recibir el nombre de Universal Battery Elimination Circuit (UBEC). Los UBEC son usualmente más eficientes que los BEC integrados en un variador, se calientan menos y pueden proporcionar mayor corriente. Esto es debido a que la gran mayoría usan el BEC en modo de conmutación.

4.5.2. Firmware y protocolo de comunicación

Hoy en día la gran mayoría de variadores tienen instalado un firmware, que ha de ser compatible con el ESC, el cual encarga de controlar su funcionamiento pudiendo modificar algunas configuraciones y parámetros.

Los firmwares más conocidos son SimonK, BLHeli junto a sus siguientes generaciones BLHeli_S y BLHeli_32 (creado para procesadores de 32 bits) o KISS.

Estos firmwares suelen venir preinstalados en el ESC, aunque también pueden recibir actualizaciones que deben ser instaladas en el ESC mediante una conexión a unos pines para realizar la comunicación con el PC.
El software instalado debe ser compatible con el ESC, pues hay fabricantes que no permiten la instalación de firmwares que no sean los suyos propios.

Pero ¿cómo nos comunicamos con el firmware? Esta comunicación se realiza a través del cable blanco del conector de servo.
Hay varios protocolos para controlar un ESC, pero el más extendido y sencillo es mediante Pulse Width Modulation (PWM) o modulación de ancho de pulso. Este protocolo de comunicación mediante pulsos es utilizado por servos, receptores RC y variadores entre otros.

La comunicación mediante pulsos PWM, como su nombre indica, funciona modulando la anchura del pulso. Cada 20ms, al inicio la señal se lleva a nivel lógico alto durante un tiempo entre 1ms y 2 ms representando el valor transmitido, con lo que podremos transmitir valores de 1000 a 2000. Después la señal volverá a estado lógico bajo hasta que acaben los 20 ms. El periodo de comunicación puede ser menor o mayor de 20 ms, pero 50Hz (cada 20ms) es la frecuencia recomendada en la gran mayoría de situaciones.

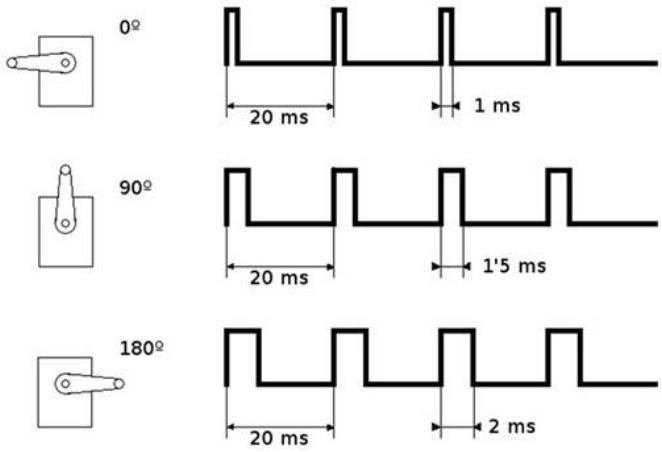


Figura 4.5: Pulso PWM para controlar un servo con un ciclo de trabajo del 0 %, 50 % y 100 %

Si se divide el tiempo que está en estado alto restándole el primer milisegundo, entre el tiempo máximo que se puede transmitir, obtendremos el ciclo de trabajo o *duty cycle*, que indica el porcentaje de tiempo que un pulsos está activo durante un ciclo. Para calcularlo aquí solo tendremos en cuenta los dos primeros ms, que es donde se transmite el valor, con lo que si el pulso es de 2ms, tendremos un ciclo de trabajo del 100 %. Si es de 1.75ms el ciclo de trabajo es del 75 %. Así pues, cuando el ciclo de trabajo sea del 0 % el ESC no hará girar el motor y al 100 % hará que gire a su máxima potencia.

Actualmente, hay otros protocolos más actuales y eficientes que los pulsos PWM. Algunos son señales analógicas parecidas al PWM como Oneshot 125, Oneshot 42 o Multishot, pero son más rápidos y reducen latencias.
Protocolos más modernos que no son soportados por variadores algo antiguos ya que usan señales digitales son Dshot150, Dshot300, Dshot600 o el más novedoso Dshot1200. En estos Dshot, el número indica la velocidad en Kbps. A diferencia con los protocolos con señal analógica, estos son más rápidos, tienen una mayor resolución de valores, y pueden comprobar si ha habido un error en la transmisión. En su contra, necesitan de un hardware más específico necesitando algunos de ESC con procesadores de 32 bits.

4.6 Sensores

Según la Wikipedia, un sensor es *"un objeto capaz de detectar magnitudes físicas o químicas, llamadas variables de instrumentación, y transformarlas en variables eléctricas. [...] Una magnitud eléctrica puede ser una resistencia eléctrica (como en una RTD), una capacidad eléctrica (como en un sensor de humedad), una tensión eléctrica (como en un termopar), una corriente eléctrica , etc."*[1].
Para que un dron pueda funcionar correctamente necesita incorporar obligatoriamente varios sensores, aunque a menudo podemos encontrar otros sensores para ampliar o mejorar el funcionamiento del vehículo. A continuación se detallan algunos de los más utilizados:

4.6.1. Acelerómetro

El acelerómetro, como su nombre indica, mide la aceleración (variación de velocidad por unidad de tiempo) en cada uno de sus ejes. Por tanto, podemos medir cambios de velocidad en sus ejes y por ende podremos determinar la posición de un objeto estacionario gracias a una aceleración muy conocida: la gravedad. Cabe destacar que el acelerómetro

sólo puede medir las aceleraciones, pero sin embargo no puede medir la velocidad del cuerpo o el tiempo que está en movimiento.

Por poner un ejemplo, con el sensor en reposo en el eje vertical obtendríamos una aceleración de 9.8 m/s^2. Pero, supongamos que el objeto no fuera estacionario y estuviera en caída libre: en este caso se generaría otra aceleración en sentido opuesto al de la gravedad y por tanto obtendríamos una medición de 0 m/s^2 en el eje vertical.

Podemos encontrar diversos tipos de acelerómetros dependiendo de su modo de funcionamiento, algunos de ellos son: mecánicos, piezoeléctricos, piezoresistivos, capacitivos, térmicos o microelectricomecánicos (MEMS).

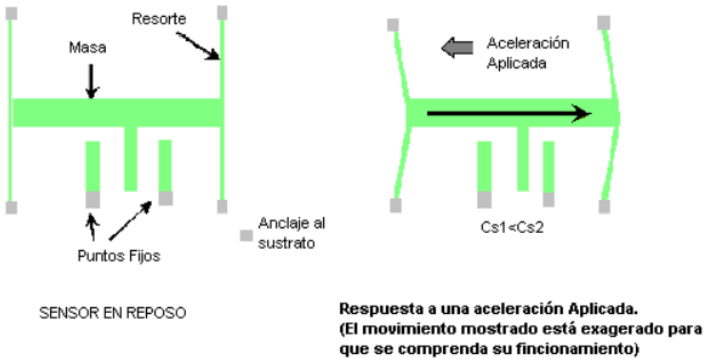


Figura 4.6: Esquema del funcionamiento de un acelerómetro capacitivo

4.6.2. Giroscopio

El giroscopio (o giróscopo) fue inventado por Foucault en 1852 basándose en la *máquina de Bohnenberger*[1] (1852), y en él, se pone de manifiesto el efecto giroscópico, el mismo que hace que una peonza girando no caiga o que una motocicleta inclinada y con determinada velocidad se mantenga en equilibrio. Este mecanismo consiste en un disco girando a velocidad constante sobre un eje, de forma que si se aplica una fuerza que hace cambiar la inclinación del eje, se generará otra que lo contrarresta. Así pues, si éste se coloca sobre un soporte Cardano[2] sin rozamiento (o casi nulo), la orientación del eje de rotación del giróscopo permanecerá idéntica gracias a la conservación del momento angular.

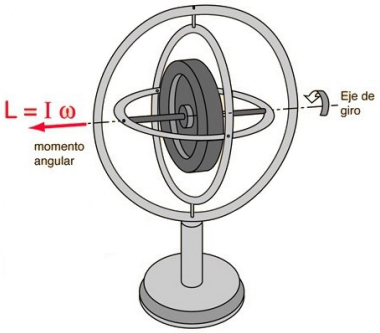


Figura 4.7: Una rotación del disco suficientemente rápida hace mantener la dirección de giro, cambiando la posición del marco

[1] https://es.wikipedia.org/wiki/Johann_Gottlieb_Friedrich_von_Bohnenberger

[2] Un soporte Cardano, según la Wikipedia, es *un mecanismo consistente en dos o tres aros concéntricos cuyos ejes forman un ángulo recto, lo cual permite mantener la orientación de un eje de rotación aunque su soporte se mueva*

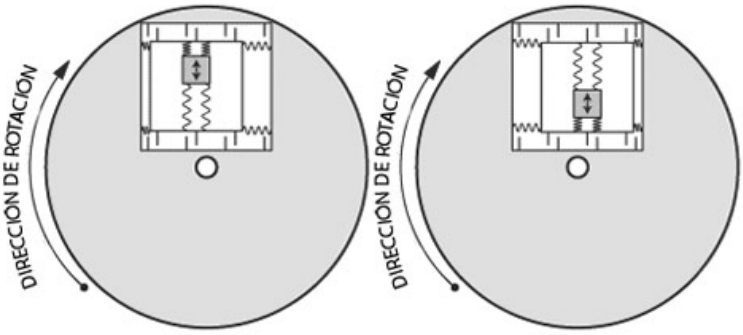


Figura 4.8: Disco rotatorio de un giroscopio MEMS

Este comportamiento nos permite medir la orientación de un cuerpo: un sensor giroscopio incorpora una pequeña masa en el disco que está rotando que se se desplaza dentro de un alojamiento con sensores, de forma que mide la velocidad de rotación o velocidad angular en sus ejes, que posteriormente servirá para obtener el desplazamiento que ha realizado el objeto y el ángulo respecto a cada uno de los ejes para determinar la orientación del objeto. Este comportamiento descrito corresponde a los giroscopios de estructura vibrante, aunque también existen de otros tipos como los ópticos que no funcionan en base a partes mecánicas o la velocidad angular sino de láseres y la velocidad de la luz.

4.6.3. Magnetómetro

Un magnetómetro mide campos gravitatorios, ya sea la dirección y/o la fuerza de estos. El tipo o ejemplo de magnetómetro que nos interesa, en cuanto a nuestro vehículo se refiere, es el tipo más conocido: la brújula.

La brújula es un instrumento que sirve para orientarnos en los puntos cardinales gracias al campo magnético terrestre. La brújula clásica consiste en una aguja imantada que señala el norte geográfico (sur magnético). Este instrumento permitirá al vehículo conocer en qué dirección apunta, permitiéndonos medir el ángulo formado entre ésta y el norte geográfico. Esto es esencial para mantener el rumbo en vuelos programados para seguir automáticamente una ruta.

Figura 4.9: Brújula común

Su defecto es que su medición se ve afectada por la presencia de otros campos magnéticos (pueden ser generados por corrientes eléctricas de circuitos cercanos) y metales, por lo que se debe ser cuidadoso a la hora de elegir el lugar donde colocarlo, y posteriormente calibrarlo para reducir el posible error generado por alguno de los casos anteriores.

4.6.4. Barómetro

El barómetro es el encargado de determinar la presión atmosférica. Fue un invento de Torricelli en 1643 y antiguamente consistía en un fino tubo de vidrio de 850mm lleno de mercurio y boca abajo sumergido en un recipiente con más mercurio. La presión ejercida por el aire en la superficie del mercurio del recipiente hace subir o bajar el nivel del mercurio en el tubo de vidrio, indicando la presión ejercida por el aire.

En la actualidad se suele usar barómetros que no hacen uso del mercurio debido a su alta toxicidad, sino que se usan barómetros que en su interior alberga un cilindro o

cápsula con pared elástica (aneroide) donde se ha hecho un vacío parcial, que se deforma con la presión atmosférica. Esta deformación produce un movimiento en la aguja que nos permite realizar la medición.

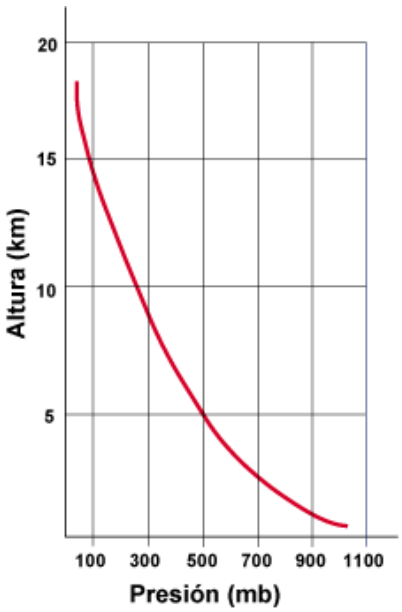


Figura 4.10: Presión en función de la altura sobre el nivel del mar

Estos sensores pueden ser usados para realizar predicciones meteorológicas locales mediante la presión atmosférica obtenida. Por otro lado, también es usado en drones ya que son útiles para estimar la altura, pues la presión atmosférica disminuye conforme la altura sobre el nivel del mar (760 mmHg o 1013 mbar) aumenta. Sin embargo esto no ocurre de forma lineal, sino que en los primeros kilómetros de ascenso disminuye más rápidamente que en alturas mayores.

4.6.5. IMU

Una Inertial Measurement Unit (IMU) o unidad de medición inercial es usado en distintos tipos de aeronaves ya sean tripuladas o no, así como en naves. Es un dispositivo que combina sensores como los anteriormente descritos para obtener información precisa acerca del movimiento o posición del vehículo. La explicación a que este dispositivo pueda tener varios sensores redundantes, es para minimizar el error que puedan afectar a estos. Son dispositivos muy cómodos ya que disponen varios sensores en la misma placa minimizando el espacio.

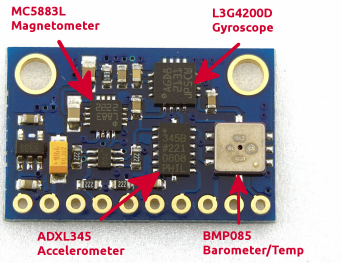


Figura 4.11: IMU GY-80

Por ejemplo, la IMU GY-80 incluye los siguientes sensores:

- L3G4200D, un giroscopio de 3 ejes
- ADXL345, un acelerómetro de 3 ejes
- MC5883L, un magnetómetro de 3 ejes
- BMP085, un sensor que es barómetro y termómetro

Cada sensor tiene asignada una dirección I_2C que sirve para identificarlos y podremos encontrar en el *datasheet* de este dispositivo. Además, para hacer funcionar la placa solo necesitamos cuatro de los pines que posee: los pines SCL y SDA para la comunicación I_2C, el pin GND y el de alimentación que es compatible con 5V y 3.3V, así que puede funcionar fácilmente con un Arduino.

4.7 Controladora de vuelo o Unidad de Control

Se podría decir que es el cerebro del dron, se encarga de recibir todos los datos de sensores, datos del receptor RC (si lo hubiera) procesar la información y responder a esos estímulos.

4.7.1. Control PID

El control PID es un mecanismo de control automático por realimentación, el cual obtiene la desviación o error entre la salida deseada para el proceso (*setpoint*) y la salida la medida, con el fin de minimizar este error. Para que este control pueda llevarse a cabo, son necesarias al menos las siguientes partes:

- Un **sensor**, que realizara alguna medición en el proceso o sistema
- El **controlador**, el cual recibirá el valor que se desea alcanzar así como el valor medido por el sensor. Con estos datos y unas constantes hará el cálculo matemático para corregir el error haciendo un ajuste con el actuador.
- El **actuador** es la parte que nos permitirá modificar el estado del sistema para corregir el error.

Términos de la ecuación

El cálculo PID se compone de tres partes: la proporcional, la integral y la derivativa, donde cada parte tiene una constante llamada de igual forma con el fin de asignarle un peso.

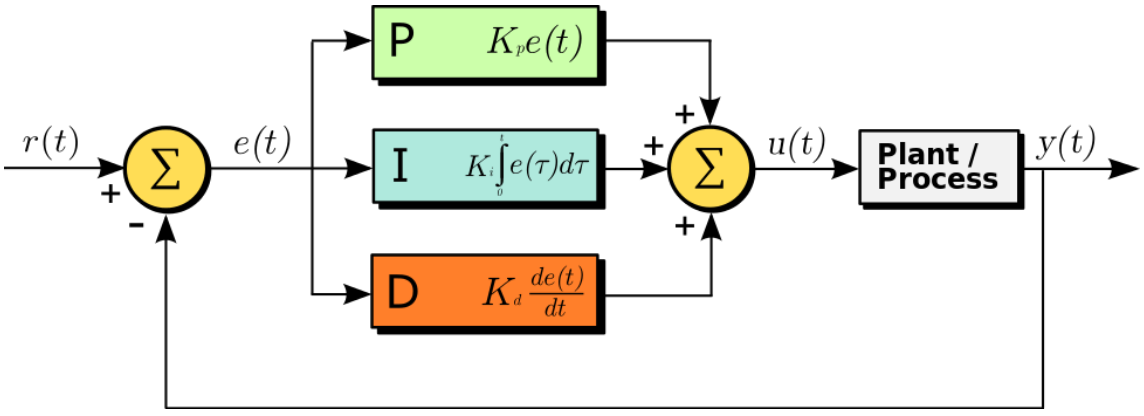


Figura 4.12: Diagrama de bloques de un controlador PID

Tal y como se puede ver en la figura 4.12, se resta el valor de la medición realizada por el sensor, $y(t)$, al valor que deseamos obtener, $r(t)$, obteniendo como resultado el error $e(t)$. Este error será utilizado para calcular tres términos de la ecuación, los pertenecientes a la parte proporcional, integral y derivativa), que serán sumados para finalmente obtener $u(t)$, la señal de salida del controlador con la que se controlarán los actuadores.
Vistos todos los términos, la ecuación para calcular la salida del controlador sería la siguiente:

$$u(t) = K_p e(t) + K_i \int_0^t e(\tau)d\tau + K_d \frac{de(t)}{dt} \tag{4.1}$$

En el caso de los drones es imprescindible que el software incluya el algoritmo PID, ya que es necesario para mantener la estabilidad del vehículo en el aire. En este ejemplo de control PID, el acelerómetro y giroscopio miden la velocidad y posición del dron en cada uno de los ejes para calcular la potencia con la que debe girar cada motor. La mayoría de tarjetas o placas de vuelo disponen de distintos PID para poder elegir la configuración que mejor se adapte a nuestras necesidades. También permiten modificar los valores de cada componente.

4.7.2. Comunicación Serie

Un puerto es una interfaz, ya sea virtual o física, a través de la cual se pueden enviar o recibir datos.
Un puerto serie, a veces referidos como Universally Asinchronous Receiver Transmitter (UART) (en realidad una UART el hardware encargado de construir los paquetes para enviar y recibir los datos convirtiendo la secuencia de bits en serie a paralelo y viceversa, gestionar las interrupciones, así como la velocidad a la que se transmiten los datos (llamada *baud rate* y medida baudios o bits por segundo (bps))) es aquel que hace uso de la comunicación serial, es decir, que transmite la información bit a bit, a diferencia de la comunicación en paralelo que envía todos los bits de cada símbolo enviado, con lo cual necesita más líneas de transmisión. Para ello, son necesarias al menos dos conexiones: RX, para la recepción y TX para la transmisión. En principio, la comunicación serial es más lenta pero puede solventarse aumentando la frecuencia de comunicación.

La ventaja de la comunicación serial sobre la paralela es que el método de comunicación es más sencillo, y además se permiten distancias mayores de comunicación, con un máximo de 1200m frente a los 20m respectivamente. Esto es debido a que pueden ocurrir problemas de interferencias o desincronización entre el tiempo de llegada de los bits enviados.

Algunos ejemplos de comunicación serial pueden ser el famoso código morse, o los puertos Ethernet, FireWire, Serial ATA, el bus I^2C, o el USB entre otros.

4.7.3. I^2C

Un circuito interintegrado, en inglés Inter-Integrated Circuit (I^2C), es un bus de comunicación usado ampliamente en distintos dispositivos, por ejemplo sensores. En este bus la comunicación es síncrona, y hay dos papeles que intervienen: el maestro, y los esclavos. Todos los dispositivos conectados tienen una dirección (tiene un direccionamiento de 7 bits, esto es hasta 112 nodos, ya que 16 son reservadas) para hacer posible saber con qué nodo queremos comunicarnos. La comunicación siempre es iniciada el maestro y no es posible hacerla directamente entre esclavos.

Existe un modo multimaestro en el que dos dispositivos pueden ser maestros, pero solo uno lo puede ser a la vez, actuando el otro como esclavo. El maestro que quiera enviar datos debe comprobar si el bus está libre, y si lo está se convertirá en el nuevo maestro del bus, que liberará después de realizar la comunicación. El cambio de maestro es bastante complejo de realizar, así que no es algo usual.

La gran ventaja es que sólo son necesarios dos cables o líneas de señal: CLK, la frecuencia de reloj que determinará la velocidad de comunicación; y SDA, del inglés *Serial Data*, por donde se transmitirá la información. Son necesarias resistencias pull-up, pero gran parte de dispositivos ya las incluyen.

Otro punto a favor es que la conexión es muy sencilla, ya que todos los esclavos están conectados directamente a las dos líneas.

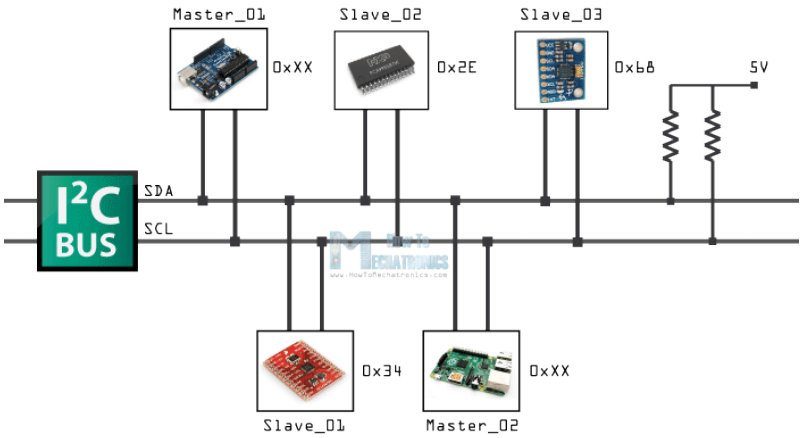


Figura 4.13: Ejemplo de conexión de un bus I^2C con tres esclavos y dos maestros

En su contra, cambe mencionar que no hay verificación de que el contenido del mensaje sea correcto, además de no disponer de una velocidad muy grande.
Otra desventaja es que la comunicación es semidúplex o *half-duplex*, es decir, hay comunicación en ambos sentidos pero ésta no es simultánea.

4.7.4. Protocolo MAVLink

El protocolo MAVLink, del inglés *Micro Air Vehicle Link* es un protocolo de comunicación creado en 2009 y diseñado para vehículos no tripulados, especialmente para la transmisión de información entre el vehículo no tripulado y la estación de tierra.

MAVLink es una librería escrita en C muy ligera y sólo con cabeceras, por lo que es muy simple incluirla en muchos proyectos ya que sólo se necesita copiar las cabeceras e incluirlas en el programa. Además, está pensada y optimizada para casos en los que se dispone poco ancho de banda, así como sistemas con memoria RAM y flash limitadas.

Actualmente va por la versión 2.3 y es usado en muchos proyectos y sistemas como placas controladoras o estaciones de tierra (APM Planner, QGroundControl, iDroneCtrl, etc.).

Estructura de un paquete MAVLink

Un paquete[3] MAVLink se compone de tres partes: la cabecera, la zona de datos o *payload* (carga útil).

[3]En redes y comunicaciones, un paquete es el conjunto de datos o bloque fundamental necesario para el transporte de la información. La información se divide en paquetes. Normalmente están compuestos por una cabecera donde a menudo están indicados la información relativa al protocolo, una parte de datos, y una cola que nos puede ayudar a detectar errores en la transmisión.

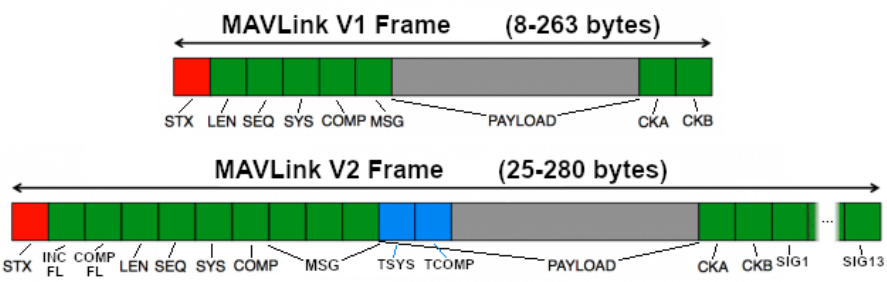


Figura 4.14: Diferencias en la estructura de un paquete en MAVLink 1 y MAVLink 2

En la figura 4.14 cada sección cuadrada se corresponde con un byte.

En la primera versión del protocolo, la cabecera de un paquete comienza siempre con el byte de inicio, o como en la documentación es referido *magic marker* ya que es el byte con el que siempre se inicia un paquete y siempre tiene el valor 0xFE (por eso está marcado en rojo). El segundo byte nos indica el tamaño del payload (*payload length*). Los dos siguientes bytes indican el ID del sistema que emite el mensaje, y el ID del componente o parte del sistema que lo envía, con lo cual podemos identificar diferentes sistemas en la misma red, así como diferentes componentes del mismo sistema (IMU, OSD[4], cámara). Posteriormente se encuentra el bloque de datos o payload, es el bloque que incluye el mensaje y puede tener una longitud de 0 a 255 bytes.

A continuación del bloque de datos se incluyen dos bytes para el Cyclic Redundance Check (CRC), una verificación cíclica. Es un mecanismo para detectar la integridad de los datos, ya que pueden producirse errores en la transmisión de la información debido a ruidos durante ésta.
Un CRC hace repetidamente una operación matemática, que luego vuelve a aplicar a su resultado (cíclica). Generalmente esta operación matemática se suele ser una división de polinomios, con uno de ellos conocido por la parte emisora y receptora, llamado polinomio generador. Para hacer los cálculos se usa la forma binaria de los polinomios, y uno de los métodos usados para realizar la división es mediante la operación lógica XOR.
Así pues, un paquete será correcto si el CRC indicado en el paquete es igual al calculado.

El método CRC usado por MAVLink es el X.25, un tipo de CRC usado para la transmisión de paquetes vía radio.

La segunda versión de la librería incluye más bytes en cada paquete ya que hay más campos, como son dos bytes para indicar flags o características del paquete compatibles o incompatibles; ahora el ID del mensaje ocupa 3 bytes debido a la inserción de nuevos tipos de paquetes con nuevas funcionalidades (esto nos permite diferenciar más de 16 millones de mensajes, frente a los 256 de la versión 1); los dos bytes iniciales del payload son usados en la comunicación punto a punto para indicar el sistema y componente de destino; y finalmente después de los dos bytes de CRC encontraremos 13 bytes para la firma, que nos ayudará a detectar que el paquete no está manipulado.
Además, en esta versión podemos extender opcionalmente algunos mensajes con más campos o parámetros, sin la necesidad de crear otro tipo nuevo de mensaje.

Mensajes

En la carga útil de los paquetes se incluirá el contenido del mensaje enviado. Los mensajes que podrá utilizar un sistema vendrán determinados por un archivo .xml, donde también vendrán especificado el tipo de los parámetros, o las posibles opciones para

[4] *On Screen Display*, sirve para mostrar información sobre la imagen obtenida de la cámara

algunos parámetros y flags, todo ello junto a sus correspondientes descripciones. Este subconjunto de mensajes y nombres usados recibe el nombre de dialecto. El conjunto de mensajes más utilizado es el definido en el archivo *common.xml* disponible con las librerías y sobre el que se suele partir si se desea crear un conjunto de mensajes personalizado pudiendo crear nuevos mensajes o eliminando los que no sean necesarios.

Hay multitud de mensajes definidos en el fichero *common.xml* (v2.3), pero algunos que cabe destacar son:

- El mensaje de *heartbeat* o latido (ID=0) que envía la estación de tierra para comprobar que el sistema sigue respondiendo y no se ha perdido la señal, con lo que si se pierde el bit de vida (la señal se ha perdido) el vehículo puede aterrizar inmediatamente o que volver a la posición de despegue.
- *Request data stream* (ID=66, obsoleto), el cual nos sirve para solicitar información a algún componente, y que nos proporcione esta información a una determinada frecuencia. Por ejemplo a la placa controladora podemos pedirle nos envíe información sobre la posición GPS cada segundo, la información de los sensores o los canales RC.
  Actualmente este mensaje está obsoleto y se recomienda usar el mensaje *SET_MESSAGE_INTERVAL*:

```
09      <message id="66" name="REQUEST_DATA_STREAM">
10        <description>THIS INTERFACE IS DEPRECATED. USE SET_MESSAGE_INTERVAL INSTEAD.</description>
11        <field type="uint8_t" name="target_system">The target requested to send the message stream.</field>
12        <field type="uint8_t" name="target_component">The target requested to send the message stream.</field>
13        <field type="uint8_t" name="req_stream_id">The ID of the requested data stream</field>
14        <field type="uint16_t" name="req_message_rate" units="Hz">The requested message rate</field>
15        <field type="uint8_t" name="start_stop">1 to start sending, 0 to stop sending.</field>
16      </message>
```

Figura 4.15: Mensaje *REQUEST_DATA_STREAM* definido en el fichero *common.xml*

- *SET_MESSAGE_INTERVAL* (ID=511), reemplaza al mensaje anterior, y sirve para lo mismo. Los parámetros que recibe son el id del mensaje que queremos que se nos envíe, y la frecuencia con la que debe hacerlo.

CAPÍTULO 5
Construcción del dron

5.1 Elección de las piezas

Para empezar a construir el dron tenemos que elegir los componentes con los cuales daremos vida al dron.
Después de analizar las distintas posibilidades y proyectos, podemos enfocarnos ya en elegir las piezas con las cuales construiremos el dron. Hay que elegir las piezas que más se adapte a nuestras necesidades.

Figura 5.1: Pack que incluye un APM2.6, GPS y soportes

La placa controladora que montaremos será la placa APM 2.6, ya que viene respaldada por una comunidad muy grande como la de ArduPilot y es muy sencillo montarla. Además, incluye todos los sensores necesarios además de GPS.
Con esto dispondremos de una gran diversidad de funciones para el vuelo ya programadas y evitando errores en algoritmos como el de control PID que no conciernen a nuestro estudio.

Después se ha elegido una estructura o *frame* imitando la del DJI F450, de plástico y con dos colores diferentes de brazos para así poder distinguir la parte delantera de la trasera y no perder la orientación durante el vuelo.

Figura 5.2: Pack que incluye un APM2.6, GPS y soportes

Puesto que hay muchos factores a tener en cuenta a la hora de elegir los motores, ESC's y hélices, se ha optado por comprar un kit económico para el quadcopter DJI F450. El kit incluye cuatro motores brushless A2212 de 1000KV, junto con cuatro ESC de 30A más cuatro pares de hélices 1045 (para los dos sentidos de giro). La numeración de estas hélices indican que tienen 10 pulgadas de diámetro, y 4.5 pulgadas de paso. Es decir, que con una revolución al 100 % de rendimiento la hélice avanza 4.5 pulgadas.

Con el fin de proteger las hélices, se han adquirido cuatro protectores que van atornillados al *frame*, y un tren de aterrizaje compuesto por cuatro patas para poder aterrizar.

Como suministro de energía tendremos una batería LiPo de la marca Multistar de 4000mAh, tres celdas, 11.1V y 10C. Se ha elegido esta ya que tiene muy buenas opiniones y en las especificaciones aseguran que *proporciona un aumento*

de los tiempos de vuelo de hasta el 20 % o más en comparación con las baterías Lipoly estándar del mismo peso

Para comunicar la placa APM con la estación de tierra, se usará un Arduino DUE junto con un módulo SIM808.
El módulo SIM808 está basado en el chip de SIMCOM SIM808, y nos ofrece la posibilidad de conectarnos a redes GSM/GPRS (tecnología móvil 2G), aunque también incorpora la tecnología para usar GPS. Se ha elegido este módulo ya que había mucha información online y tutoriales relacionados con este chip. El motivo de no usar un módulo con tecnología móvil 3G es que sus precios son bastante más elevados, sobrepasando la cifra de los 100 €.

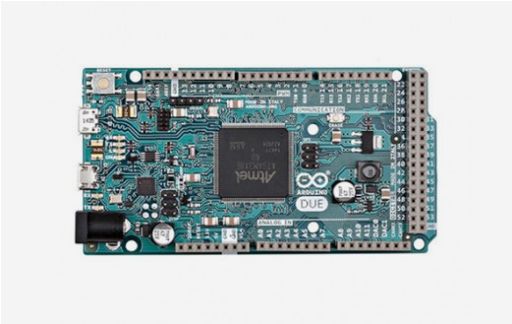


Figura 5.3: Arduino DUE

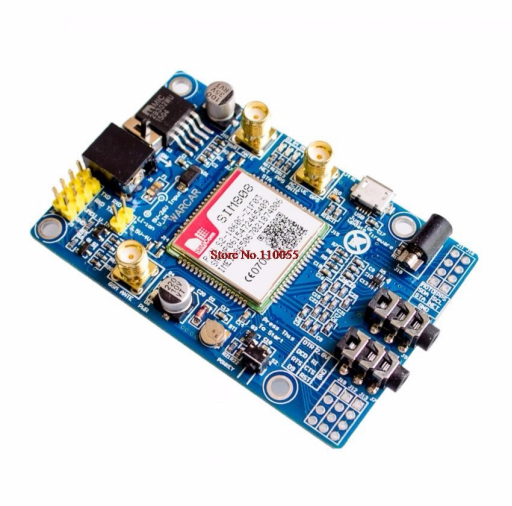


Figura 5.4: Módulo SIM808

5.2 Montaje del dron

El primer paso es construir un dron con las funciones básicas que nos ofrece la placa APM. Es decir, será controlado con una emisora RC y los datos de telemetría serán enviados a través de un módulo de radio 433MHz. Se usará primero este método de comunicación con el fin de comprobar que el vehículo no tiene problemas para despegar y mantener un vuelo aceptable, antes de abordar el tema de realizar las comunicaciones a través de la red móvil (capítulo 6).

Una vez hemos recibido todos los paquetes y tenemos los componentes listos para comenzar a dar vida a nuestro dron.

Empezamos soldando unos conectores a los ESC's y motores para facilitarnos la sustitución de estos en posibles futuras reparaciones. El conector elegido es el de tipo banana (de 3.5mm de diámetro), que disponen de un terminal macho y otro hembra. Después de soldar los conectores, éstos se han protegido con tubo termoretráctil.

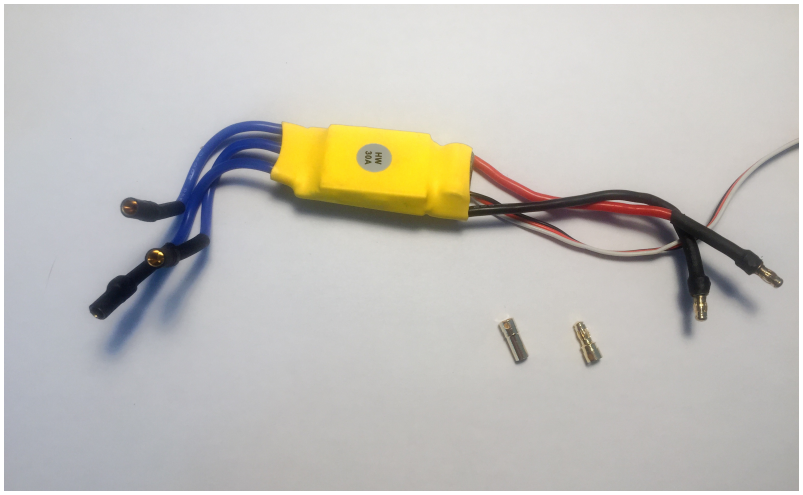


Figura 5.5: ESC con los conectores soldados

Hecho esto pasamos a ensamblar las partes del *frame*, así como del tren de aterrizaje, protectores de hélices, motores y ESC's. También se ha instalado el cable de la alimentación donde irá conectada la batería y distribuirá la energía. En vez de usar una placa para la distribución de energía o PDB (*Power Distribution Board*), se usará un cable que para cada polo se separa en 6 conectores, con lo que podremos hacer 6 conexiones en paralelo, de las que cuatro serán para los ESC.

Figura 5.6: Estado de montaje del dron con los motores y ESC instalados

El siguiente paso es montar el soporte anti vibraciones para la APM, junto con la placa, GPS y el receptor radiocontrol y hacemos todas las conexiones según indica la documentación de ArduPilot.

Para la emisora al menos 4 canales para controlar el *pitch* o cabeceo, el *roll* o alabeo, el *throttle* o aceleración, y el *yaw* o guiñada; los otros dos son canales auxiliares que pueden ser usados para otros fines, como controlar el movimiento de un gimbal para una cámara o para cambiar el modo de vuelo. Podemos encontrar emisoras con más canales, incluso 15, aunque lo usual son de 4, 6 u 8 canales, aunque por el momento no necesitaremos más de 6. Utilizaremos la emisora de HobbyKing HK-T6A V2 junto con su receptor trabajan-

do a 2.4GHz. Esta emisora dispone de 6 canales, así como dos conmutadores (*Switch A, Switch B*) y dos piezas que podemos girar a modo de rueda (*VR(A), VR(B)*).

Para la telemetría usaremos un módulo radio de 3DR que dispone de su emisor (con su conector para el puerto de telemetría) y su receptor por USB, trabajando en los 433MHz.

Figura 5.7

5.3 Pasos iniciales

5.3.1. Configuración de la emisora HK-T6A V2

Una vez tenemos todas las piezas montadas y conectadas, y antes de abordar la configuración de nuestra placa de vuelo, es necesario configurar nuestra emisora RC.

Existen emisoras que incorporan pantallas y botones para modificar distintos parámetros y guardar varias configuraciones (muy útil si disponemos de varios vehículos), aunque nuestra emisora no dispone de esta funcionalidad: la configuración ha de realizarse mediante un cable USB usando una herramienta en el pc llamada *T6config*

Figura 5.8: Emisora FlySky i6

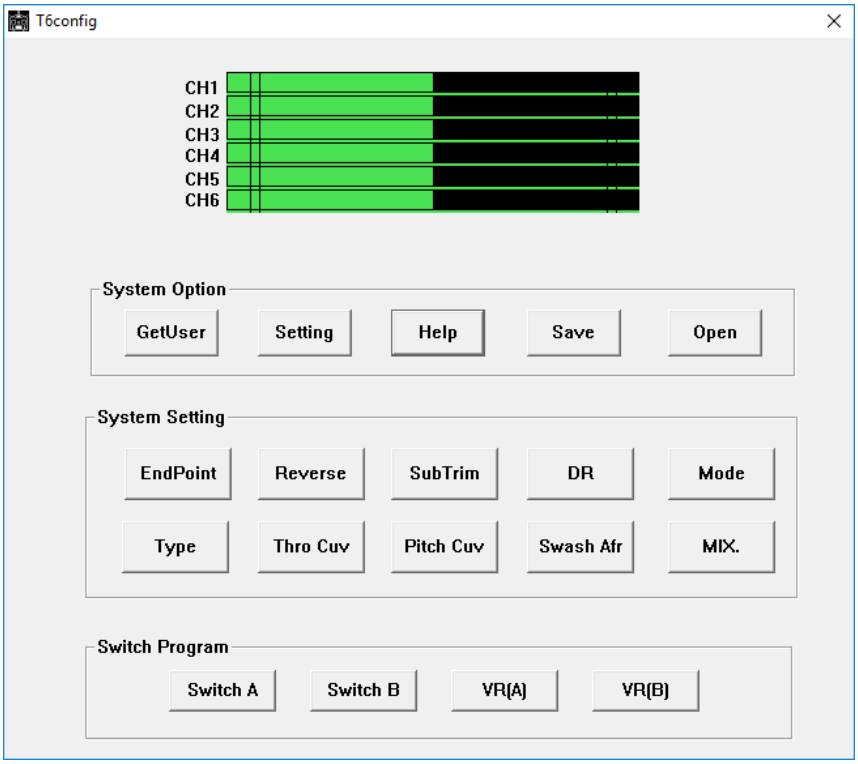


Figura 5.9: T6config, la utilidad para configurar nuestra emisora

Al ejecutar la aplicación, salta a la vista que la herramienta es algo antigua y no está muy cuidada visualmente. Vemos que podremos configurar multitud de opciones como qué canales controla cada *stick*, la curva de aceleración, el comportamiento que tendrán los conmutadores y ruedas sobre los canales, invertir los canales, etc.

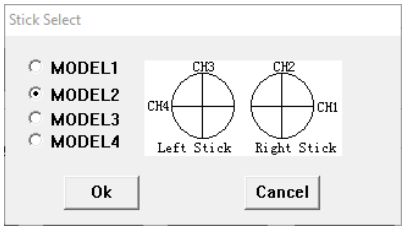


Figura 5.10: Modelo que configuraremos

Los puntos importantes a configurar son: poner los canales que controlan el movimiento del dron en el stick adecuado, y se ha elegido el modelo 2 (aunque esto puede cambiarse según preferencias personales, el más usado en drones es éste); y poder cambiar la salida del canal 5 para posteriormente cambiar el modo de vuelo, por lo que se ha configurado el switch B para ello. Además, también se ha configurado el switch A en *ThroCut*, de forma que cuando lo activemos el canal de aceleración se mantendrá a cero aunque movamos la palanca correspondiente, a modo de seguridad.

Una vez tengamos la emisora configurada según nuestros gustos y necesidades, podremos exportar la configuración y guardarla en un archivo.

5.3.2. Configuración de la controladora de vuelo

Una vez tenemos las piezas del dron ensambladas, es el momento de instalar el firmware de ArduCopter a través de Mission Planner, la estación de tierra de ArduPilot para PC. Después se debe seguir un proceso de configuración guiado, el cual establecerá los distintos parámetros y configuraciones de vuelo y calibrará los sensores que se incorporen.

Cuando Mission Planner esté instalado, debemos ejecutarlo y seleccionar en el menú superior la opción *INITIAL SETUP*, después la opción *Wizard*. Esta opción nos permitirá realizar la configuración inicial de nuestro dron siguiendo los pasos indicados.

El primer paso es seleccionar el tipo de vehículo, que en nuestro caso será un multirotor, y posteriormente el tipo de frame. Con esta información Mission Planner sabrá qué fork de ArduPilot ha de instalar, ArduCopter. Para poder instalar (o actualizar) el firmware adecuado, debemos conectar la placa mediante USB y seleccionar el puerto correspondiente en la lista, de forma que el programa detectará qué placa tenemos. En nuestro caso detecta que nuestra placa ya está obsoleta, y nos informa que instalará el último firmware compatible con ella (ArduCopter 3.2.1)

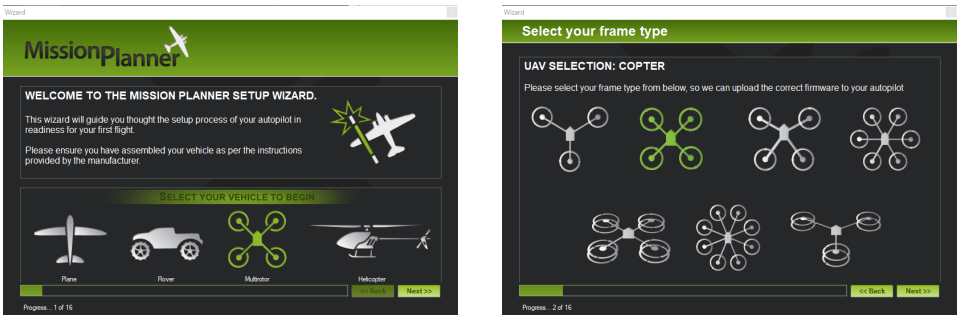


Figura 5.11: Mission Planner: Selección del tipo de vehículo

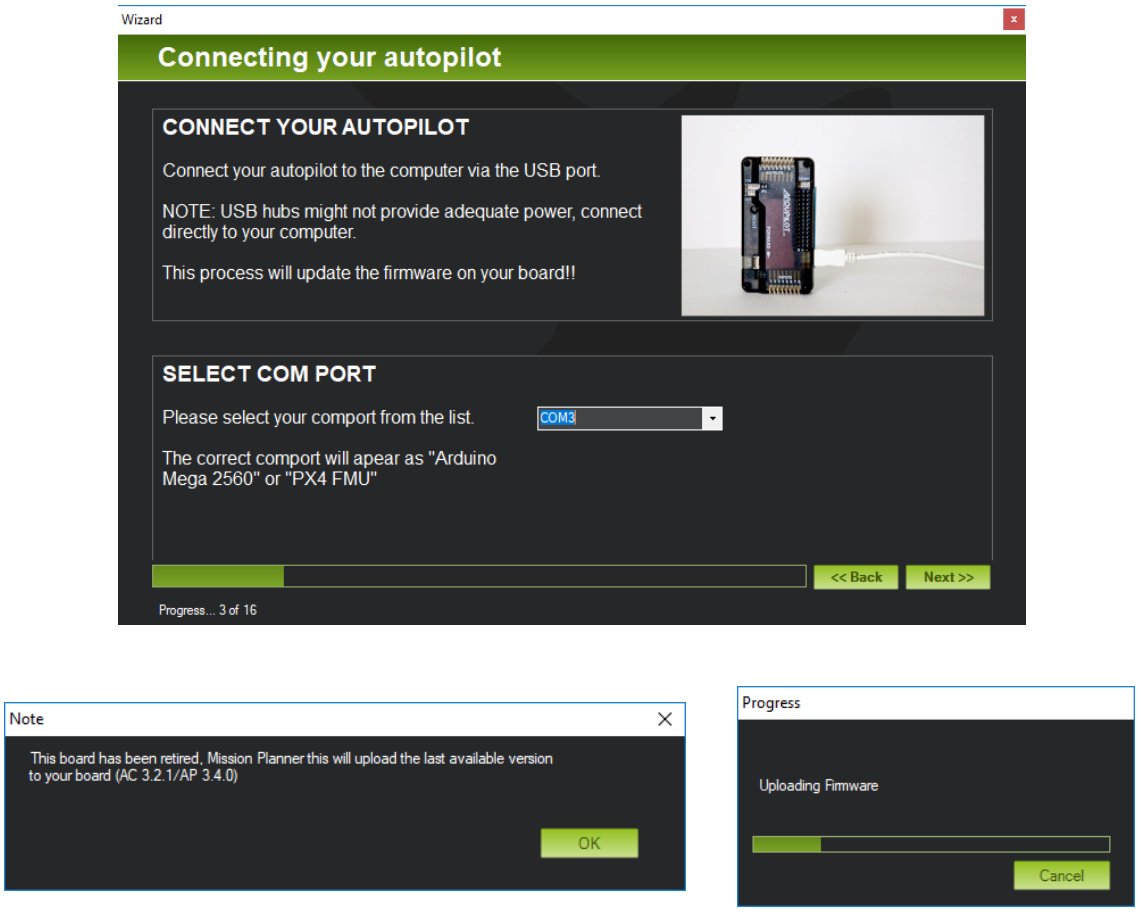


Figura 5.12: Mission Planner: Instalación del firmware en la placa

Cuando el firmware acabe de subirse a la placa y el programa verifique que se ha hecho correctamente, se han de calibrar los sensores. Esto ya puede hacerse mediante un módulo de radio, ya que es incómodo realizarlo con cable, aunque es posible hacerlo. En primer lugar se calibra el acelerómetro, poniendo el vehículo en las distintas posiciones que nos indica. Después es el turno del *compass* o magnetómetro, un proceso que al principio puede resultar algo confuso: debemos girar el vehículo alrededor de sus ejes, y veremos que en el eje de coordenadas (hay tres, uno para cada uno de los 3 compases que podemos instalar) de la ventana se van trazando unos puntos que deberán pasar por los puntos blancos haciéndolos desaparecer; cuando todos los puntos desaparezcan el proceso finalizará, y obtendremos los *offsets* o desplazamientos en cada eje para obtener buenas mediciones, compensando posibles interferencias generadas por los cables, ESC o motores. Hay que tener en cuenta que *offsets* altos indican que hay interferencias, con lo que cuanto menores sean mejor. Si hay algún *offset* demasiado alto (mayor a 600) se coloreará de rojo, y el vehículo no obtendrá buenas mediciones.

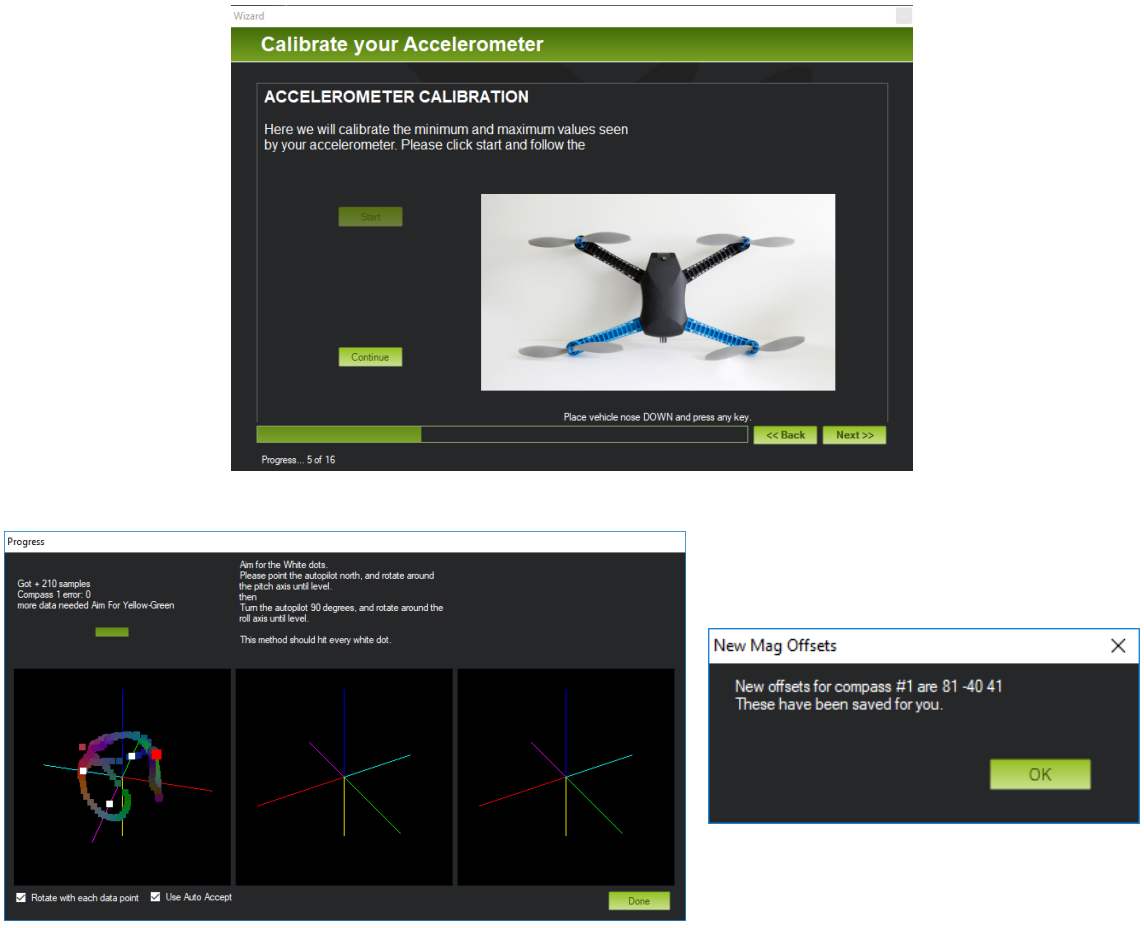


Figura 5.13: Mission Planner: Calibración de acelerómetro y magnetómetro

En el siguiente paso se configurarán las opciones de alimentación: si disponemos de algún módulo de alimentación (*Power Module*, nos ayuda a determinar el nivel de batería o el consumo de corriente), así como la capacidad de la batería.

Seguidamente se calibrará la emisora RC, moviendo todos los *sticks* a sus posiciones extremas, con el fin de determinar el rango de valores que enviará a la placa. También se han de configurar los modos de vuelo, que vendrá determinado por el PWM del canal 5. Podremos configurar teóricamente hasta seis modos de vuelo diferentes (seis secciones de todo el rango posible de PWM). El comportamiento de la emisora debe configurarse

para que podamos modificar el PWM del canal 5 ya sea con algún *switch* o palanca, o con algún control que podemos variar girándolo (algo así como un potenciómetro variable). En nuestro caso sólo usaremos dos modos de vuelo ya que es algo complejo, y la emisora no dispone de ninguna pantalla así que la configuración se ha de realizar conectándola al pc. Configuraremos un modo de vuelo en *Auto* para realizar vuelos autónomos y el resto en modo *Loiter* para controlar el dron manualmente pero con ayudas como por ejemplo mantener la altura y la posición.

Para finalizar podremos configurar acciones en el caso de que se pierda la señal RC o la batería esté a punto de agotarse, como por ejemplo volver al punto de despegue (*RTL, Return To Launch*) o aterrizar *Land*. También podremos definir un espacio a partir del punto de despegue, de el cual el vehículo no saldrá. Estas opciones no las activaremos por el momento ya que no son estrictamente necesarias.

5.3.3. Primeros vuelos

La primera parte del proyecto ya está finalizada. Es el momento de buscar un lugar espacioso y poco o nada transitado para poder probar su correcto funcionamiento.

La primera vez que se intentó volar el vehículo, una hélice no debía estar bien sujeta ya que salió disparada cuando el vehículo ascendió a un metro de altura. El segundo vuelo fue satisfactorio, a pesar del viento existente, y sirvió para comprobar que el vehículo disponía de la suficiente potencia y que no había ningún problema.

A estas alturas ya podemos seguir con la siguiente fase: la comunicación a través de la red móvil.

CAPÍTULO 6
Comunicación vía red móvil

Este capítulo describe todo el proceso necesario para que la comunicación entre dron y Ground Control Station (GCS) se realice a través de Internet vía red móvil. Cabe destacar que no se ha desarrollado cada parte en serie, sino que se han ido desarrollando las tres partes en paralelo para poder así detectar errores prematuramente y ir siguiendo gradualmente el esquema pensado para el intercambio de datos.

6.1 Programa en el servidor

6.1.1. Configuración previa del servidor

Inicialización

Como se ha explicado anteriormente, el servidor nos ayudará a que la conexión sea posible dando a conocer la dirección de nuestro dron a la estación de tierra.

Para ayudarnos a llevar a cabo esto, y puesto que no disponemos de ninguna dirección IP estática, se ha alquilado un Cloud VPS[1] económico (en este caso el precio es de 1 € al mes) de la empresa italiana Aruba Cloud. De momento no habrán numerosas conexiones, por lo que no se necesitará gran cantidad de recursos.

El servidor escogido se encuentra alojado en Francia, y dispone de una dirección IP estática, 1 vCPU (*Virtual CPU*), 1GB de RAM, 20GB de almacenamiento en discos duros SSD (no se necesitará una gran cantidad de almacenamiento), y podremos transferir 2TB de datos al mes. Es perfecto para empezar a hacer las pruebas aunque si el número de conexiones y hilos aumenta necesitaremos alquilar otro plan.

Posteriormente, debemos inicializar el servidor. Primero deberemos establecer un usuario y una contraseña. Podemos elegir entre multitud de plantillas (*templates*) ya configuradas, dependiendo del plan contratado. Por ejemplo en nuestro caso podemos elegir entre plantillas con CentOS, Ubuntu Server o Debian, entre otros. También podemos encontrar varias versiones de éstos (Windows Server 2008 o 2012), o con distintas arquitecturas

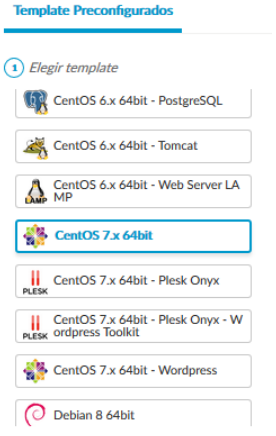


Figura 6.1: Lista de plantillas a escoger

[1] En realidad es un VPS. Un servidor VPS o *Virtual Private Server*, es un servidor en una máquina virtual, compartiendo con otras máquinas virtuales una máquina física. Es decir, la máquina física es dividida en varias máquinas virtuales. En cambio, en los servidores Cloud se abstrae el servidor del hardware físico, y por ello es más tolerable a fallos y ofrece opciones de fácil escalado, pero por contra tienen un precio más elevado.

de procesador (32 o 64 bits), junto con un gran abanico de variantes de estos con diverso software preinstalado y configurado para empezar a trabajar, por ejemplo como servidor web (CentOS 6.x 64bit - Web Server LAMP), servidor de correo electrónico (CentOS 6.x 64bit - Mail Server), o servidor de bases de datos (CentOS 6.x 64bit - PostgreSQL).
En nuestro caso se ha optado por usar una plantilla de CentOS 7 de 64 bits limpia, sin ningún software preinstalado.

Seguridad: cortafuegos

Tras esperar a que la máquina acabe de inicializarse, el siguiente paso es configurar el firewall para abrir únicamente los puertos que necesitemos y cerrar el resto para evitar posibles ataques. Durante la realización de las primeras pruebas, este paso se omitió, y al dejar la máquina conectada sin usar durante varios días, la empresa contactó para comunicar que la máquina iba a ser cerrada, puesto que había estado atacando a otras máquinas realizando numerosas conexiones: la máquina había sido infectada y usada para atacar a otra(s). Tras contactar con la empresa y clarificar los hechos, se devolvió el crédito perteneciente a ese mes, ya que se había perdido.

El kernel de Linux dispone de un poderoso firewall basado en tablas, basado en reglas (*rules*) y objetivos (*targets*). Mediante estas reglas podremos configurar multitud de opciones pudiendo filtrar el tráfico por multitud de opciones como por ejemplo protocolo (Transmission Control Protocol (TCP), udp, etc.) o puerto, entre otros.
La tabla que nos permitirá permitir o bloquear parte del tráfico es la tabla *filter*, la cual contiene tres cadenas por defecto: *INPUT*, para el tráfico con destino la máquina local; *OUTPUT*, para el tráfico generado desde la máquina local hacia otra; y la cadena *FORWARD* para el tráfico en el que la máquina local es un intermediario.

Un objetivo puede ser una cadena definida por el usuario, uno de los objetivos ya definidos (*ACCEPT, DROP, QUEUE, LOG*) o extensiones de éstos. Así pues, la política asignada para cada cadena por defecto es la siguiente, la cual se aplicará cuando ninguna regla de la cadena haga matching:

- *INPUT*, para el tráfico entrantes, la política por defecto será *DROP*. El paquete es rechazado sin generar ninguna respuesta hacia la máquina que lo envía (a diferencia de REJECT, que envía un paquete ICMP indicando que el puerto es inalcanzable)

- *OUTPUT*, para el tráfico saliente, la política por defecto es *ACCEPT*. Todo el tráfico saliente se permitirá.

- *FORWARD*, para el tráfico reenviado, la política por defecto será *DROP*.

En este estado, todo el tráfico entrante sería bloqueado, por lo que se insertan las reglas correspondientes para permitirnos conectar al servidor mediante escritorio remoto (puerto 3389), permitir la comunicación por Secure Shell (SSH) (*Secure SHell*, puerto 22). También permitiremos el tráfico entrante de conexiones ya abiertas o establecidas.

Posteriormente, en el servidor se ha cambiado el número de puerto por defecto para SSH, así como las reglas en iptables correspondientes, ya que un posible atacante conocería ya el número de puerto SSH para realizar un ataque. Además se ha instalado *fail2ban*, un programa que nos ayudará a bloquear conexiones (modificando iptables) que realizan demasiados intentos de conexión fallidos.

6.1.2. Programa en el servidor: primera versión

Una vez el servidor está operativo, el siguiente paso es desarrollar el programa que se encargará de realizar las comunicaciones. Se ha optado por usar Java, ya que dispone de muchas herramientas y será fácil poner el programa a funcionar en nuestro servidor, lanzándolo desde la consola. El IDE que usaremos será Eclipse ya que se ha usado con anterioridad y no nos retrasará a la hora de comenzar nuestra labor de programación.

El curso de la comunicación ideado en primer lugar seguiría el esquema de la figura 6.2.

Se desea que sea la estación de tierra o
quien se conecte al dron, y puesto que la dirección IP proporcionada por el proveedor no es estática, a priori no la conocemos, así que necesitaremos la existencia de un servidor. Cuando el dron se conecte, éste se registrará en el servidor indicando que es el dron, el servidor guardará su dirección IP y cerrará la conexión.

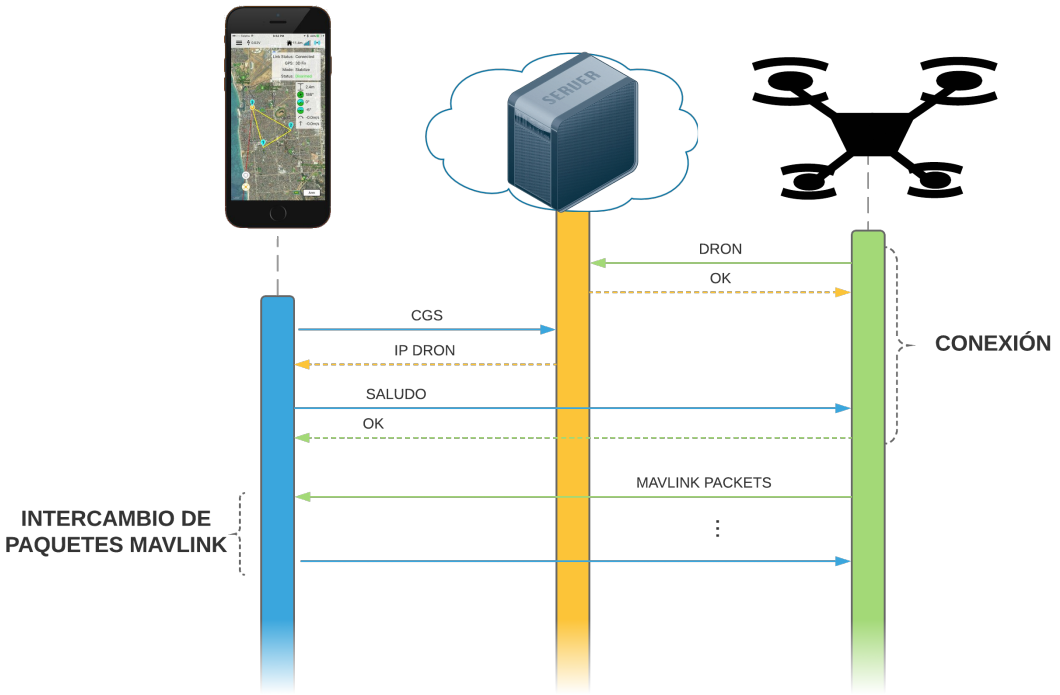


Figura 6.2: Versión inicial del desarrollo de las comunicaciones

El programa creará un hilo que estará a la espera de nuevas conexiones (habrá que identificar qué parte es: el dron o la estación de tierra) en un puerto recibido como argumento, de forma que al aceptar la conexión se creará un hilo donde se esperará recibir una cadena que indica si se trata de un dron (la cadena será: "*dron*") o una estación de tierra ("*gcs*"). Si el cliente envía cualquier otra cadena, la conexión se cerrará. En el caso de que la cadena recibida sea correcta se guardará cada dirección IP en la variable correspondiente (*ipDron* o *ipGCS*. Cuando la estación de tierra intente conectarse, deberá haberlo hecho primero el dron y en caso contrario se le notificará que debe intentarlo cuando el dron esté conectado: si se le contesta con un "0"se indicará que el dron ya está conectado y se iniciará la comunicación; un "1"significa que la dirección IP del dron todavía no se conoce (porque todavía no se ha conectado), y se cerrará la conexión. Cuando la conexión de la

estación de tierra sea satisfactoria, se le comunicará al dron con la cadena "gcsok", y éste abandonará la conexión con el servidor y esperará a que la GCS se conecte a él, y cuando ocurra la comunicación entre ambos, mediante el protocolo MAVLink, comenzará.

El programa en el servidor podrá ser detenido en cualquier momento escribiendo en la consola la palabra *quit*, gracias a un *Scanner* en otro hilo que comprobará lo que introducimos por teclado.

Resultados

Los resultados obtenidos en esta versión no son los esperados: nuestro servidor recibe las conexiones de ambas partes, las identifica y obtiene sus direcciones IP. También logra comunicar satisfactoriamente la dirección IP del dron a la GCS, pero la conexión de la estación de tierra al dron nunca se realiza.

Tras investigar detenidamente[2] se descubre que no es ningún error de programación, ni problema del módulo SIM808. El problema viene dado por la compañía proveedora del servicio de Internet ya que no permiten conexiones entrantes directas, algo usual en todas las compañías. Esto es debido a que usan direcciones IP públicas compartidas por varios dispositivos.

Una posible solución a nuestro problema sería utilizar una SIM que nos ofrezca el proveedor con una IP fija o estática, de esta forma nos aseguraríamos de que nuestra IP no está siendo compartida con ningún otro dispositivo y podremos abrir los puertos deseados. La gran desventaja de esta solución es que su precio es bastante elevado ya que deberemos abonar una cuota de alta y otra cuota mensual, además del precio mensual de la tarifa que hayamos escogido.

Esto hace que necesitemos cambiar el esquema de comunicación, puesto que no podemos realizar tal esfuerzo económico, aunque ésta sería la opción ideal. En la siguiente sección se propone una alternativa a este esquema de comunicación.

6.1.3. Programa en el servidor: segunda versión

Puesto que la primera opción para realizar la comunicación ha quedado descartada, debemos cambiar la forma en la que se realiza.

En esta versión introduciremos un par de cambios importantes:

- Se desea que el servidor pueda ser usado por varios drones y estaciones de tierra, de forma que pueda manejar distintas comunicaciones de forma simultánea. Esto se logrará registrando cada parte en el servidor identificándolas con IMEI (un identificador único a nivel mundial de 15 dígitos que distingue a cada dispositivo dentro de la red móvil) del módulo SIM808 y una contraseña, de forma que si el IMEI y la contraseña enviada por la GCS es igual a la establecida por el dron se permitirá la comunicación.
  El IMEI de nuestro módulo SIM808 es el que viene indicado en la pegatina del chip. La distinción de usuarios también podría llevarse a cabo usando en lugar del IMEI el número telefónico que tenga la SIM o un nombre de usuario, por ejemplo.

Figura 6.3: Código IMEI en un módulo SIM808

[2] www.raviyp.com/embedded/202-using-sim900-as-tcp-server-read-this-first

- La GCS no se conectará al dron directamente, sino que la comunicación se realizará en todo momento a través del servidor, a modo de puente intermedio. Esto hará que la carga en el servidor aumente ya que el servidor será fundamental durante toda la conversación.
- Se creará un archivo a modo de *Log* donde se volcarán los sucesos que ocurran en el servidor y en qué momento: cuándo se inicia el servidor, conexiones satisfactorias, qué parte conecta, cuándo se detiene el servidor, etc.

En esta segunda versión, añadiremos más datos a la cadena que envía cada parte para registrarse en el servidor. Además de la palabra que identifica si es un dron o una GCS, se le añadirá una cadena de texto (podría ser IMEI, número de teléfono, correo, etc., pero usaremos el IMEI) y una contraseña. Sin la contraseña, una GCS que no es la nuestra podría tomar el control de nuestro dron si introduce nuestro IMEI. Para enviar todos estos datos en una cadena necesitamos definir un separador para los distintos campos, función para la que escogeremos un carácter de la tabla ASCII que no se pueda introducir mediante teclado. El carácter se ha escogido aleatoriamente y es el carácter cuyo valor decimal en la tabla ASCII es 30.

Cuando el dron se conecte se guardarán los objetos necesarios en una tabla hash, siendo la clave el IMEI. En esta versión, cuando la GCS se conecta puede obtener tres respuestas: dron no conectado (indicado con un "2"), contraseña incorrecta ("1") o dron conectado y contraseña correcta ("0"). Si la respuesta obtenida es ésta última, se envía al dron la cadena "gcsok".

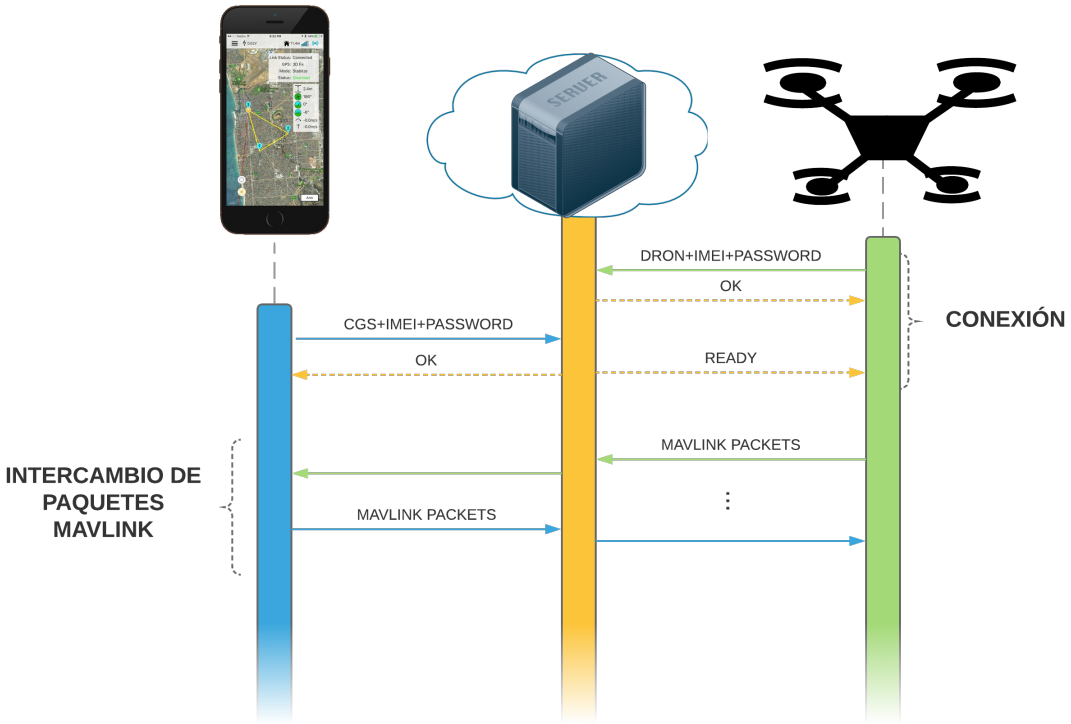


Figura 6.4: Segunda versión del transcurso de las comunicaciones

Una vez las dos partes están listas, comenzará la comunicación. Pero aquí la comunicación no se realizará de forma directa entre las dos partes: el servidor hará de interme-

diario.
Para hacer posible la comunicación en los dos sentidos, se ha creado la clase *NonBlockingWay* (véase anexo) que extiende la clase *Thread* y recibe como parámetro un objeto *InputStream* y un *OutputStream*, de forma que lee bytes del objeto *InputStream* y los escribe en el *OutputStream*; puesto que la operación de lectura es bloqueante (el programa quedará en espera hasta que hayan datos que leer) esto se ejecuta en el método *run()*. Para que la comunicación se pueda llevar a cabo, necesitaremos dos objetos de la clase *NonBlockingWay*: uno para cada sentido de la comunicación.

El archivo .jar generado por Eclipse se subirá al servidor mediante SCP (que hace uso de SSH) con el programa *WinSCP*, con el que podremos tener sincronizado un directorio en la máquina local con otro directorio del servidor, enviando automáticamente al servidor archivos creados o modificados. Para ejecutar nuestro programa no usaremos el escritorio remoto, ya que es algo engorroso debido a la latencia, por lo que usaremos *PuTTy*: un cliente SSH (entre otros) con el que podremos tener acceso a la consola del servidor y ejecutar nuestro programa.

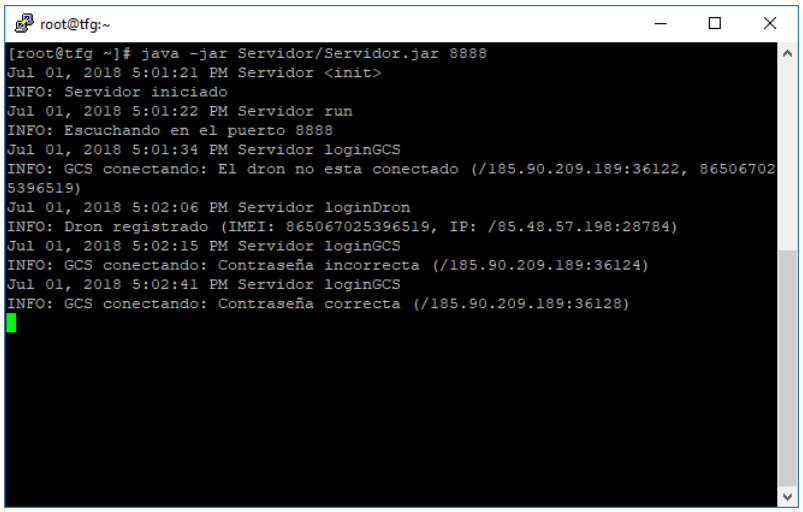


Figura 6.5: Segunda versión del transcurso de las comunicaciones

6.2 Envío de la telemetría: Arduino

6.2.1. Construcción

Para poder que el dron pueda comunicarse con la estación de tierra via red móvil, necesitamos algún tipo de dispositivo que envíe toda la información que pasa través del puerto de telemetría, en ambos sentidos. Para el dron debe ser algo transparente, como si se conectara un sistema de telemetría radio.

Como anteriormente se ha dicho, usaremos un módulo SIM808 (GSM/GPRS) junto a un Arduino DUE, ya que gracias a su procesador ARM, tiene una mayor velocidad que por ejemplo el UNO, y además es de 32 bits. El otro punto por el que ha sido elegido es que dispone de cuatro UART (aunque una es usada por el puerto de programación), es decir, podemos usar a la vez varios puertos seriales sin necesidad de emularlos vía software (mediante la librería *SoftwareSerial*) donde la velocidad se reduce mucho.

El puerto de telemetría es un puerto serial, que está formado por un conjunto de 5 pines: un pin con +5V (pin 1, de abajo hacia arriba), dos pines correspondientes a TX y RX (pines 2 y 3, conectados a otro RX y TX respectivamente), un pin CTS (*Clear To Send*,

pin 4) que según el estándar RS-232 se conecta a otro pin del dispositivo usado llamado RTS (*Request To Send*) para indicar que está listo para enviar; y por último el pin GND (pin 5) que deberá ir a masa compartida con el Arduino DUE.

Conectaremos los pines RX y TX a los pines TX3 y RX3, de forma que en el código será accesible mediante Serial3. Los dos pines GND serán conectados, y con el pin +5V alimentaremos el Arduino DUE.

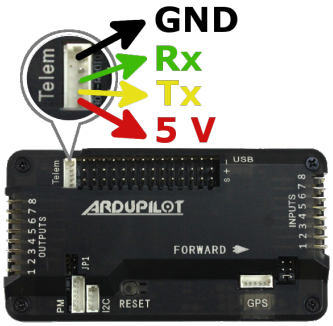


Figura 6.6: Ampliación del puerto de telemetría

La segunda parte importante a conectar es el módulo GSM, lo que nos permitirá acceder a Internet gracias a la red móvil y una tarjeta SIM.

En primer lugar se ha adquirido una tarjeta SIM de prepago, con el fin de controlar el consumo y poder recargarla según nuestras necesidades.
En primer lugar se ha buscado la documentación para nuestro módulo, con el significado de los pines y cómo iniciarlo.

El módulo puede ser alimentado a través de tres interfaces: una para una batería de litio de 2.5V a 4.2V, y el pin V_{in} o el conector DC044. Las últimas dos admiten de 5V a 26V. Hay que tener en cuenta que viene indicado que la corriente máxima es de 2A, por lo que hay que asegurarse de que la fuente sea capaz de proporcionarlos.
Según la documentación del Arduino DUE, éste no podrá alimentar el módulo ya que la corriente máxima que pueden proporcionar sus pines es de 130mA, pudiendo ocasionar daños en la placa si se excede este límite.

Figura 6.7: Conector

Así pues, para hacer las pruebas de comunicación el módulo SIM808 será alimentado con un transformador, pero a la hora de integrarlo en el dron, la fuente de energía será la misma que para los ESC's y motores: la batería principal.
Para ello se ha comprado un conector (figura 6.7) al cual conectaremos los cables provenientes de la batería.

Una vez sabemos cómo alimentar el módulo, ¿Cómo comunicarse con él? Mediante la interfaz serial TTL[3].

Para usar esta interfaz, como anteriormente, haremos uso de los pines RX, TX y GND, aunque en este caso también debemos conectar el pin VMCU al voltaje con el que se represente el nivel lógico alto. En el caso del Arduino DUE, se trabaja a 3.3V, así que deberemos conectar VMCU al pin de 3.3V para que el módulo SIM808 nos "entienda".

Veamos una imagen donde quedan reflejadas todas las conexiones mencionadas anteriormente:

[3] *Transistor-Transistor Logic*, es decir, lógica transistor a transistor. Es una familia lógica creada con transistores bipolares

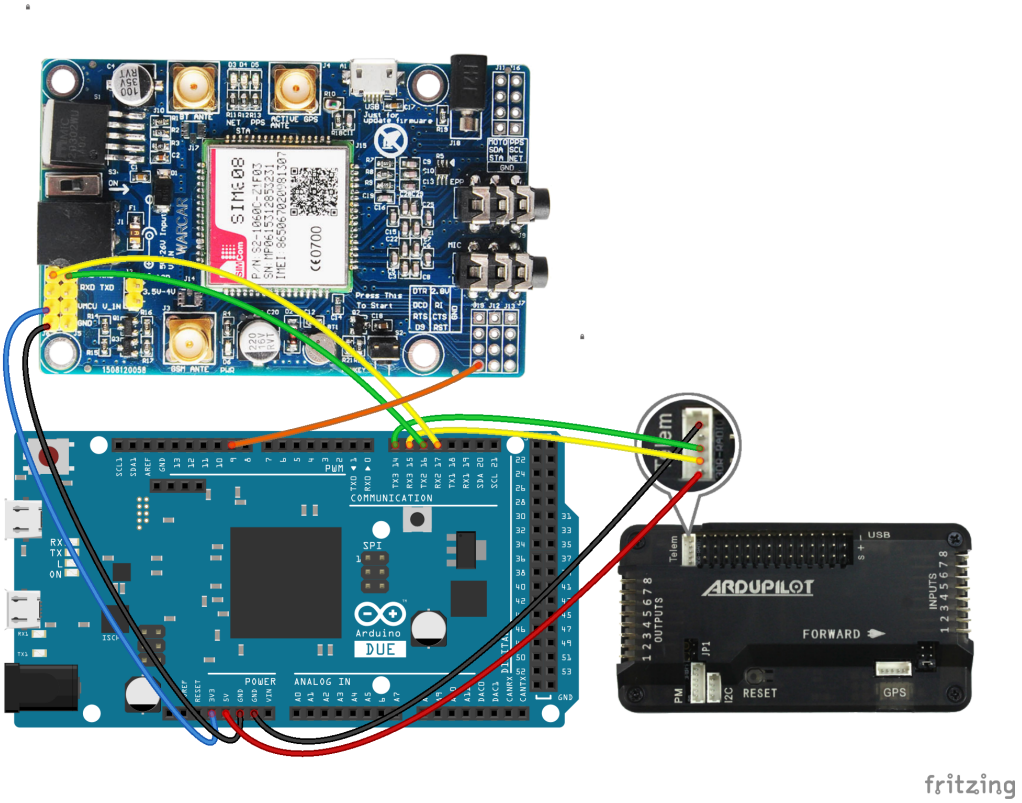


Figura 6.8: Conexiones del dispositivo que nos permitirá la comunicación via red móvil

La utilización del pin D9 (cable naranja) es para encender el módulo sin la necesidad de presionar el botón físico, se explicará más adelante.

Por último conectamos la antena para que pueda encontrar la señal en la parte inferior del módulo, donde se encuentra etiqueta *GSM ANTE*. Ya podemos pasar a comprobar su correcto funcionamiento.

6.2.2. Probando el módulo SIM808

Una vez están todas las conexiones hechas, debemos saber cómo manejar el módulo. Primero hay que comprobar que la comunicación no falla. Subiremos un *sketch*[4] muy simple al Arduino, para poder comunicarnos vía comandos AT[5] con el modem, configurarlo, y probar algunos comandos.

Conectamos el módulo a la corriente y encendemos el switch para alimentar la placa. Una vez la placa esté alimentada hay que mantener un botón por dos segundos para iniciarla. Ya encendida podremos ver parpadear algunos leds indicando que la placa está operativa.

[4]Se le llama *sketch* al programa o proyecto para Arduino. Es obligatorio que contenga las dos funciones principales *setup()*, que se ejecuta sólo una vez al principio del programa; y *loop()*, que se ejecuta en bucle continuamente.

[5]El juego original de comandos AT (de la abreviatura *attention*) fue desarrollado en 1977 por Dennis Hayes, para facilitar la comunicación entre un hombre y un módem. Así por ejemplo podemos enviarle instrucciones para realizar o recibir una llamada o SMS, desbloquear la tarjeta SIM... Más tarde algunas compañías siguieron con su desarrollo hasta universalizarlo.

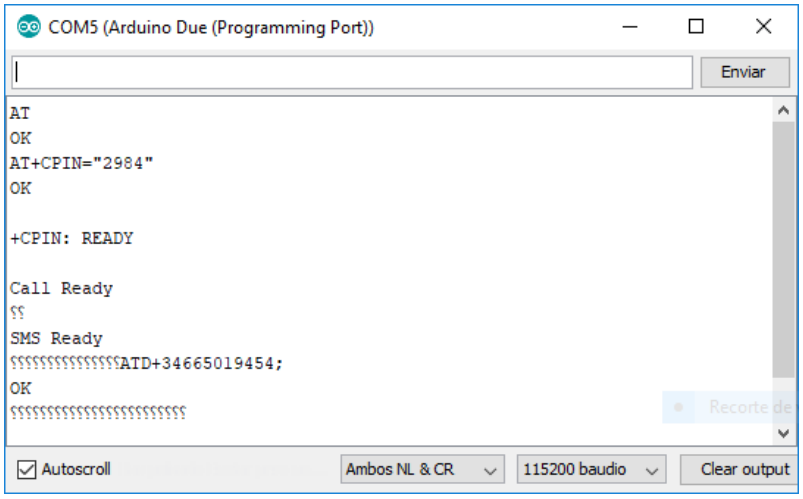


Figura 6.9: Realización de una llamada de voz mediante comandos AT

Tras probar unos pocos comandos AT indicados en la documentación a través del monitor Serial del IDE de Arduino, vemos que no hay ningún problema con la comunicación. También se ha probado a desbloquear la SIM y realizar una llamada con éxito.

Antes de seguir, vemos que para encender el módulo debemos hacerlo mediante un switch que lo alimenta y después presionar un botón durante dos segundos para que se encienda/apague. Este puede ser un comportamiento que no deseamos, ya que queremos que cuando lo alimentemos éste ya se encienda sin necesidad de presionar ningún botón. Este comportamiento lo conseguiremos mediante software, poniendo el pin D9 del módulo en nivel bajo durante un segundo y después lo devolviéndolo a estado alto.

6.2.3. Programando Arduino: la librería TinyGsm

Para ayudarnos a manejar los comandos AT enviados y las respuestas recibidas por el módulo, se ha usado la librería TinyGsm disponible en GitHub, que incluye métodos muy útiles para controlar los comandos enviados al módulo así como las respuestas por varios módulos GSM, entre ellos el SIM808, permitiendo hacer llamadas, enviar SMS, crear conexiones TCP o usar el GPS que incluye el módulo, de una forma más sencilla.

El IDE que se ha utilizado es *Atom*, que es un editor de texto que permite la instalación de paquetes que amplían sus funcionalidades. Para poder compilar y subir el programa a distintas placas de Arduino, se ha utilizado *PlatformIO*, en concreto los paquetes *platformio-ide* y *platformio-terminal*, que además soporta numerosos tipos de microcontroladores.

Lo primero que tendremos que hacer es crear un nuevo proyecto a través de la pestaña de inicio de PlatformIO, y se creará la estructura para el proyecto. Las partes importantes que necesitamos conocer son:

- El fichero *platformio.ini* que encontraremos en el directorio raíz de nuestro proyecto, y donde podremos configurar distintas opciones de nuestro proyecto, como la placa que usaremos (Arduino DUE), configurar distintas opciones para generar el programa o para subirlo a la placa.
- La carpeta *lib*, en la que situaremos las librerías que usemos; cada librería deberá ir a su vez dentro de otra carpeta, de forma que dentro del directorio *lib* encontraremos una carpeta para cada librería, y dentro de esta encontraremos el código fuente de la librería en cuestión.

- La carpeta *src*, donde se encontrará nuestro código principal en un archivo llamado *main.cpp*.

El fichero principal, *main.cpp* es donde escribiremos nuestro programa. Vemos que por defecto incluye la librería *Arduino.h* para que podamos usar los tipos y constantes estándar del lenguaje Arduino, así como las funciones *void setup()*, donde configuraremos todo antes de comenzar la comunicación; y *void loop()*, el bucle principal donde realizaremos la conexión al servidor y la comunicación, de forma que si hay algún error vuelva a iniciarse todo el proceso.

La zona superior del archivo la usaremos para definir parámetros de configuración de nuestro código, como el código PIN de la tarjeta SIM, los datos de APN (*Access Point Name*) de nuestro operador para tener acceso a la red de datos GPRS, la dirección IP y puerto del servidor.

En el método *setup* se inicia la comunicación serial con el APM (en el código, Serial2) y con el módulo SIM808 (Serial3), así como la comunicación USB con el monitor serial de nuestro IDE (Serial) para poder enviar información acerca del estado en el que se encuentra nuestro programa. Después se enciende el módulo a través del pin D9 como se ha explicado anteriormente y por último se desbloquea la tarjeta SIM con el código PIN especificado.

Después, en el método *loop* esperaremos hasta que haya red y nos conectaremos a la red de datos con los parámetros de APN introducidos (APN, usuario y contraseña) y seguidamente estableceremos la conexión con nuestro programa en el servidor, y nos registraremos como un dron (visto en la sección 6.1). Una vez recibida la cadena que nos indicará que la estación de tierra ya está conectada, el bucle que comunicará el puerto serie de la APM (Serial2) con nuestro servidor: leyendo desde uno y escribiendo en el otro, y viceversa.

```
151     while(true){
152       while(APM.available()){
153         client.write(APM.read());
154       }
155       client.flush();
156
157       while(client.available()){
158         APM.write(client.read());
159       }
160       APM.flush();
161     }
```

Figura 6.10: Bucle que comunica el puerto serial del APM con nuestro servidor.

```
PS C:\Users\Javier\OneDrive\TFG\Arduino\dronTelemetry3G> pio device monitor --port COM5 --baud 115200
--- Miniterm on COM5  115200,8,N,1 ---
--- Quit: Ctrl+C | Menu: Ctrl+T | Help: Ctrl+T followed by Ctrl+H ---
Encendiendo modem...
Inicializando modem...
Modem: SIM808 R14.18
Código PIN correcto
Buscando red... OK
Conectando a la APN orangeworld OK
Conectando a 89.40.112.154 OK
Registrándose en el servidor...  OK
Esperando a la GCS...  OK
Comunicación en curso
```

Figura 6.11: Salida que genera nuestro programa de Arduino.

6.3 Modificación de la GCS

La segunda parte de la comunicación es la GCS. No partiremos desde cero, sino que usaremos un proyecto ya desarrollado que en este caso no será *Mission Planner*, sino que será Tower: un proyecto *Open source* para dispositivos Android (smartphones y tablets) desarrollada para vehículos que usan ArduPilot.

Si se instala la aplicación oficial, se puede ver en la ventana principal una barra donde están listadas las distintas opciones para conectarse al dron. Actualmente ya hay disponible una opción de conexión mediante TCP, pero está pensada para conectarse a un dispositivo externo al vehículo en la misma red que hace de puente entre el dron y GCS, y que finalmente transmite los datos al dron mediante radiofrecuencia. Esta opción se descarta ya que se necesitan más dispositivos aumentando el coste y disminuyendo la portabilidad o movilidad. Además, el alcance se vería limitado según la potencia de la antena RF, a diferencia de la comunicación vía red móvil que nos permitirá cuyo alcance viene determinado por la cobertura que dispongamos.

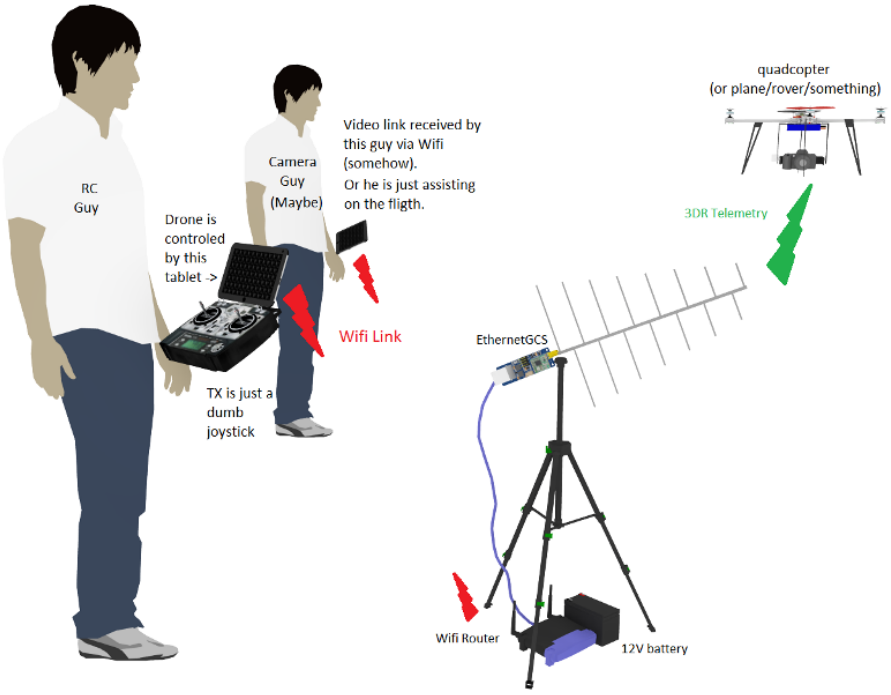


Figura 6.12: Ejemplo de conexión TCP, donde son necesarios varios dispositivos

Dicho esto, se desea modificar la aplicación de tal forma que se disponga de otra opción de conexión, además de las existentes, ampliando las funcionalidades de la app haciéndola compatible con nuestro sistema de comunicación.

Puesto que se desconoce la estructura del proyecto, los cambios no son obvios, así que el procedimiento que se ha seguido ha sido ir buscando las clases y variables relacionadas con las opciones de conexión ya implementadas, y se han creado o modificado las pertinentes partiendo de la base de la opción TCP ya implementada. También se ha modificado el tiempo para determinar si el mensaje de Heartbeat llega fuera de tiempo, aumentándolo ya que la latencia se espera que sea mayor.

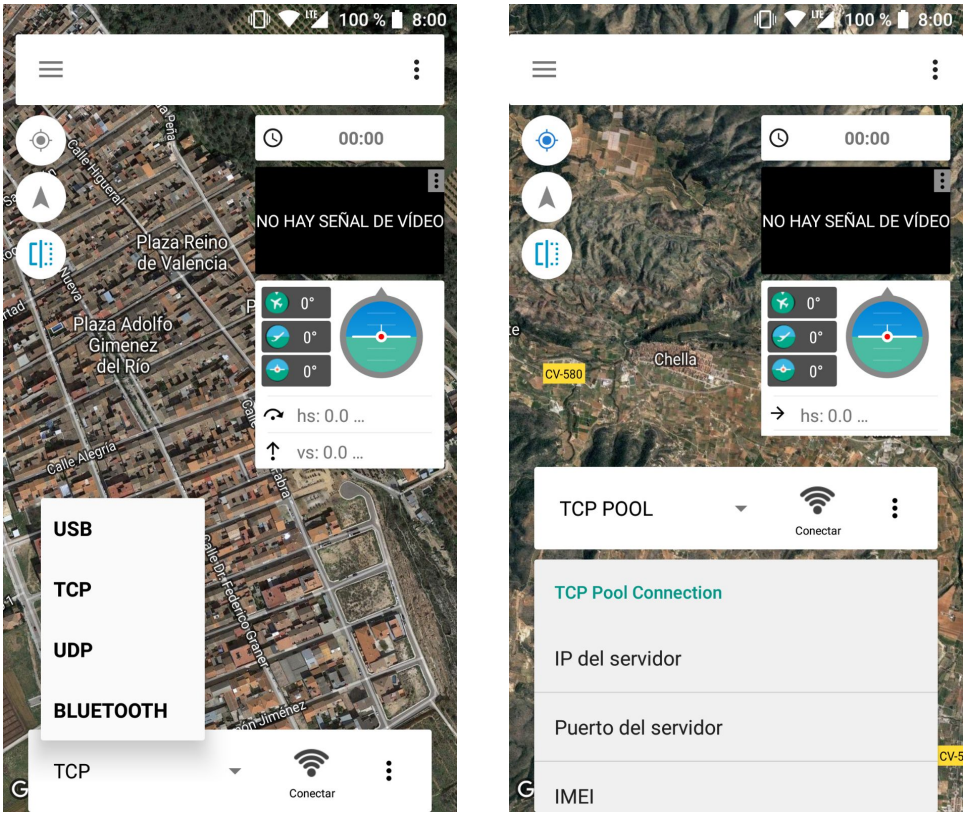


Figura 6.13: A la izquierda, opciones de conexión por defecto. A la derecha, la nueva opción de conexión añadida.

Cuando se introduzcan los datos de conexión (dirección IP, puerto, IMEI y contraseña) y se presione el botón de conectar, la aplicación se conectará al servidor y comenzará la comunicación con el dron.

6.4 Resultados

Con esta segunda versión del programa para el servidor, los paquetes MAVLink tardarán más tiempo en llegar hasta la GCS, puesto que tienen que pasar primero por el servidor. El registro de ambas partes en el servidor se realiza de forma correcta (usando la tabla hash), y notifica a la GCS si el dron está conectado o no y si la contraseña introducida es correcta. En el momento que empieza el intercambio de paquetes MAVLink, obtenemos por un instante información en la GCS pero seguidamente ésta informa que la conexión se pierde y se reanuda, no volviendo a mostrar más información acerca del dron a pesar de que siguen llegando datos en ambos sentidos. Se ha llegado a la conclusión de que la latencia es muy elevada o que hay paquetes que se corrompen, probablemente debido a errores en la transmisión de los datos.

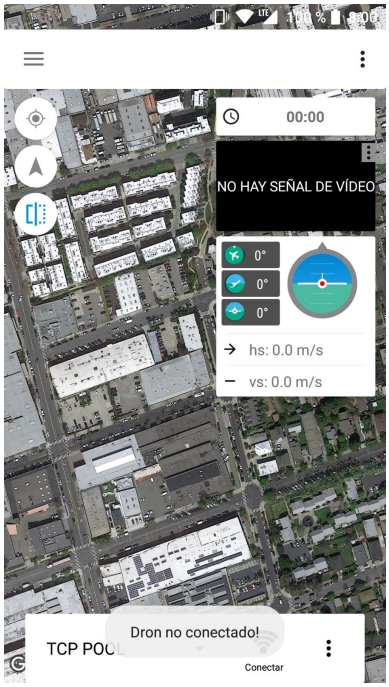


Figura 6.14: Intento de conexión de la GCS con el dron desconectado

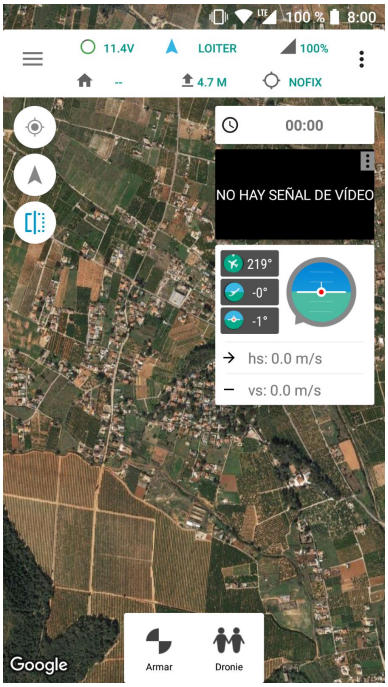


Figura 6.15: Únicamente los primeros datos son mostrados correctamente

CAPÍTULO 7
Presupuesto

En este capítulo se verán los distintos costes del proyecto, tanto de materiales adquiridos para su montaje como costes humanos si se hubiera realizado de forma profesional.

7.1 Coste material

	Descripción	Precio (€)
	Frame Quadcopter F450 White Red	17,99
	Tren de Aterrizaje Universal	6,99
	4 Protectores de hélices (para 10″ o 12″) para DJI F450	8,47
	4Pcs A2212 1000KV Brushless Motor 30A ESC+1045 Propeller para DJI F450 550 6LW0	39,99
	Conector Banana cobre 10 pares 3.5mm	2,29
	Tubo termoretráctil 2m x 5mm	2,11
	APM 2.6 + GPS Ublox M8N + soportes	36,51
	Módulo 3dr Radio Telemetry 430 MHz emisor y receptor + OTG	14,24
	Módulo SIM808 GSM GPRS GPS con Antenas	16,12

	Batería LiPo Multistar 3S 4000mAh 11.1V Multi-Rotor	16,85
	Módulo de alimentación para APM	10,22
	XT60 to 6 X 3.5mm bullet Multistar ESC Power Breakout Cable	3,70
	HobbyKing Lipo Voltage Checker (2S 8S)	1,68
	Cargador-Balanceador IMAX B6AC 80W (con fuente incl.)	29,99
	HK-T6A V2 Emisora + receptor (6 canales)	26,21
	TOTAL:	217,24

7.2 Coste humano

Suponiendo que el sueldo de un programador junior es de 15€, y para el resto de actividades realizadas de 10€\hora, se ha estimado que los costes humanos son los siguientes según las horas trabajadas:

Descripción	**Horas**	**€\hora**	**Subtotal (€)**
Búsqueda de información	200	10	2000
Construcción del dron	6	10	60
Desarrollo e implementación de los programas	120	15	1800
Realización de la memoria	200	10	2000
		TOTAL:	5840

CAPÍTULO 8
Conclusiones

A lo largo de este trabajo, han intentado alcanzarse los objetivos marcados al principio del documento de forma incremental. Inicialmente no se tenía casi ninguna competencia sobre esta área, así que el proceso de adaptación y recolección de información ha sido bastante extenso.

Una vez se adquirieron los conocimientos necesarios, se comenzaron a abordar los objetivos, empezando por el primero: la construcción de la parte básica del dron. Una vez se reunieron todas las piezas, el montaje de éste no fue muy complicado puesto que existe mucha documentación en Internet que se puede consultar ante cualquier duda que surja. Puesto que no se tenía ninguna experiencia en el pilotaje de drones, las primeras pruebas se realizaron con extrema precaución puesto que el tamaño que posee el dron lo hace más peligroso. Aun así surgió un imprevisto: una hélice se soltó (no estaba bien sujeta) y el protector correspondiente a esta hélice se partió al perder fuerza en ese brazo y precipitarse, no provocando más daños afortunadamente. En los siguientes vuelos no hubo ningún contratiempo pese a las condiciones atmosféricas, pues el viento era bastante fuerte.

Puesto que la placa controladora elegida posee ya todas las funcionalidades para realizar vuelos autónomos, las pruebas del segundo objetivo se realizaron con éxito.

En cuanto al tercer objetivo, el más extenso (ya que se desarrollaban las tres partes en paralelo, con el fin de ir testeando cada paso), no se ha logrado cumplir con éxito total. En un principio, se había planificado de manera que la GCS se conectara directamente al dron, aunque posteriormente se descubrió que esto no era posible sin una tarjeta SIM con dirección IP fija, cuyo costo y cuotas son bastante altas por lo que quedó descartada esta opción.
Así pues se tuvo que plantear otra solución, en la que todas las comunicaciones pasaban a través del servidor, algo que añade bastante más latencia, debido a que el servidor dispone de un procesador con un sólo núcleo, por lo que realmente los dos sentidos de la comunicación no se realizarán de forma simultánea (esto también sucede en el Arduino, que solo realiza un sentido de comunicación a la vez). En esta solución ambas partes se registran con éxito en nuestro servidor, y la comunicación empieza. Se recibe información en la GCS pero inmediatamente se corta, pese a que los datos siguen fluyendo en ambos sentidos. Debido a la falta de tiempo, no se ha podido determinar exactamente qué origina este comportamiento, tras varias semanas en busca de la causa. La GCS nos informa que la conexión se pierde y se restaura (los sockets en ningún momento se cierran), por lo que se cree que es debido a la latencia generada por las partes del servidor y con más influencia el Arduino DUE y el módulo SIM808, ya que no se puede realizar la comunicación de éstos de forma simultánea.

Como alternativa para su correcto funcionamiento, la conexión directa dron - GCS podría funcionar mejor quitando el servidor como intermediario del intercambio de paquetes: sólo sería necesario para conocer las IP. Esto supondría una disminución de la latencia de los mensajes, así como de la carga y tráfico en el servidor. No obstante, debido a las cuotas que se deben costear, no ha sido posible probar esta alternativa (en un primer momento la opción escogida).

Otra posible alternativa que podría funcionar sería usar un smartphone conectado por USB al puerto de telemetría, junto a una aplicación que realice la comunicación del puerto al servidor, y viceversa. Esta opción eliminaría la necesidad de usar un módulo GSM+GPRS, pues un smartphone ya posee todo lo necesario para ello. La programación de la aplicación sería en principio más sencilla que la del Arduino junto con el módulo SIM808, que usa comandos AT cuyo manejo y respuestas son más difíciles de procesar. Además, la persona que quisiera usarlo sólo necesitaría instalar la aplicación en el smartphone que irá en el dron y conectarlo al puerto de telemetría.

Bibliografía

[1] Wikipedia, "Sensor," Nov. 2017, page Version ID: 103763318. [Online]. Available: https://es.wikipedia.org/w/index.php?title=Sensor&oldid=103763318

[2] "Piezas necesarias para construir un drone de carreras," Mar. 2016. [Online]. Available: http://mundodron.net/piezas-necesarias-para-construir-un-drone/

[3] "Partes de un Drone." [Online]. Available: http://droneymas.es/partes-de-un-drone/

[4] esenziale, "Todas las Partes de los Drones. Explicadas al Detalle," Sep. 2017. [Online]. Available: https://esenziale.com/tecnologia/partes-drone/

[5] M. Gortolev, "Quadcopter vs Hexacopter vs Octocopter: Pros & Cons," Nov. 2014. [Online]. Available: http://dronebly.com/quadcopter-vs-hexacopter-vs-octocopter-the-pros-and-cons

[6] "Como elegir un variador o ESC correcto para nuestro drone o avión," Apr. 2017. [Online]. Available: http://www.loslocostambientienenblog.com/como-elegir-un-variador-o-esc-correcto-para-nuestro-drone-o-avion/

[7] "QAV210 Quadcopter Drone Beginners Build Guide," Dec. 2016. [Online]. Available: http://blog.dronetrest.com/qav210-assembly-build-guide/

[8] "Variadores: ¿Qué es un ESC? y el BEC ??" [Online]. Available: http://www.multicopters.es/foro/vbulletin/showthread.php?399-Variadores-%BFQu%E9-es-un-ESC-y-el-BEC-(Explicaci%F3n)

[9] "Bec | Drones De Carreras." [Online]. Available: http://dronesdecarreras.com/mini-cuadricoptero/bec/

[10] "Feeding power to Arduino: the ultimate guide." [Online]. Available: https://www.open-electronics.org/the-power-of-arduino-this-unknown/

[11] "Calculo para Motores y baterias - Espacio Drone." [Online]. Available: https://www.espaciodrone.com/foros/hilo/calculo-para-motores-y-baterias/

[12] "Baterías LiPo: conceptos básicos y consejos de uso," Apr. 2015. [Online]. Available: http://fulchis.es/baterias-lipo/

[13] alex, "RasPiO Duino as a Lipo Monitor," Mar. 2015. [Online]. Available: http://raspi.tv/2015/raspio-duino-as-a-lipo-monitor

[14] "BATERÍAS LIPO: Ventajas, inconvenientes y cuidados en el proceso de carga y almacenaje." [Online]. Available: http://www.cochesrc.com/baterias-lipo-ventajas-inconvenientes-y-cuidados-proceso-de-carga-y-almacenaje-a1819.html

[15] "Baterías LiPo - Guía de uso y cuidados para evitar accidentes y dar mayor durabilidad." May 2017. [Online]. Available: http://blog.teslabem.com/como-usar-y-cuidar-las-baterias-lipo/

[16] "Hélices para Drones: Tipos y Tamaños," Feb. 2017. [Online]. Available: http://fpvmax.com/2017/02/10/helices-drones-tipos-tamanos/

[17] D. D. C. staff, "Helices para mini drones. ¿Cual elijo?" Jan. 2015. [Online]. Available: http://dronesdecarreras.com/gemfan-5030-6030-5045-diferencias-entre-las-diferentes-helices-para-mini-quads/

[18] "Ventajas de los motores Brushless - patineteelectrico.net," Jun. 2017. [Online]. Available: http://www.patineteelectrico.net/ventajas-motores-brushless/

[19] T. Samsung, "Diferencia entre giroscopio y acelerometro," Aug. 2016. [Online]. Available: https://www.trucosgalaxy.net/diferencia-entre-giroscopio-y-acelerometro/

[20] A. Sabán, "Cómo funcionan el acelerómetro y el giroscopio de los móviles," Aug. 2016. [Online]. Available: https://hipertextual.com/2016/08/acelerometro-giroscopio

[21] "Giróscopo," Oct. 2017, page Version ID: 102840165. [Online]. Available: https://es.wikipedia.org/w/index.php?title=Gir%C3%B3scopo&oldid=102840165

[22] C. Escura, "La electrónica de vuelo en un drone." [Online]. Available: https://vueloartificial.com/introduccion/primeros-pasos/la-electronica-de-vuelo/

[23] "MultiWii." [Online]. Available: http://www.multiwii.com/wiki/index.php?title=Main_Page

[24] D. D. C. staff, "FPV ¿Y eso que es? 1era parte," Dec. 2014. [Online]. Available: http://dronesdecarreras.com/fpv-y-eso-que-es-1era-parte/

[25] "Suspensión Cardán," Aug. 2017, page Version ID: 101351200. [Online]. Available: https://es.wikipedia.org/w/index.php?title=Suspensi%C3%B3n_Card%C3%A1n&oldid=101351200

[26] "Comparing common flight controllers - Hobby News - Radio Control Planes, Drones, Cars, FPV, Quadcopters Articles." [Online]. Available: https://www.hobbynews.com/articles/camera-drones/2016-08/comparing-common-flight-controllers

[27] "Drones | Aprendiendo Arduino." [Online]. Available: https://aprendiendoarduino.wordpress.com/tag/drones/

[28] "ArduPilot Open Source Autopilot." [Online]. Available: http://ardupilot.org/

[29] "Brokking.net - Project YMFC-32 - The STM32 quadcopter - Home." [Online]. Available: http://www.brokking.net/ymfc-32_main.html

[30] "Cleanflight ·." [Online]. Available: http://cleanflight.com/

[31] "Baseflight VS Cleanflight - Which is better?" Jan. 2015. [Online]. Available: https://oscarliang.com/baseflight-cleanflight-comparison/

[32] "baseflight: 32 bit fork of the MultiWii RC flight controller firmware," Jan. 2018, original-date: 2012-09-05T15:27:39Z. [Online]. Available: https://github.com/multiwii/baseflight

[33] "cleanflight: Clean-code version of the baseflight flight controller firmware," Feb. 2018, original-date: 2013-03-27T23:49:19Z. [Online]. Available: https://github.com/cleanflight/cleanflight

[34] "Falcon 2 Flight Controller – Flitetronix." [Online]. Available: https://flitetronix.com/falcon-2-flight-controller/

[35] "Introduction · MAVLink Developer Guide." [Online]. Available: https://mavlink.io/en/

[36] "Welcome to the ArduPilot Development Site — Dev documentation." [Online]. Available: http://ardupilot.org/dev/

[37] "Navio2 or The Raspberry Pi Flight Controller," Apr. 2016. [Online]. Available: https://www.robotshop.com/blog/en/navio2-or-the-raspberry-pi-flight-controller-18362

[38] "Comparativa de HATs Raspberry Pi," Oct. 2015. [Online]. Available: https://comohacer.eu/comparativa-hats-raspberry-pi/

[39] R. Pujar, "Using SIM900 as TCP server ? - Read this first !!" [Online]. Available: http://www.raviyp.com/embedded/202-using-sim900-as-tcp-server-read-this-first

[40] "Controlador PID," Feb. 2018, page Version ID: 105610072. [Online]. Available: https://es.wikipedia.org/w/index.php?title=Controlador_PID&oldid=105610072

[41] D. D. C. staff, "PID para Cuadricópteros ¿Que es?" Feb. 2015. [Online]. Available: http://dronesdecarreras.com/pid-para-cuadricopteros-que-es/

[42] "Control PID." [Online]. Available: http://control-pid.wikispaces.com/

[43] "Configurar el PID de tu drone," Sep. 2017. [Online]. Available: http://joyplanes.com/configurar-pid-drone/

[44] designthemes, "SIM808: GSM/GPRS + GPS | Tienda y Tutoriales Arduino." [Online]. Available: https://www.prometec.net/sim808/

[45] "Comunicación de Arduino con puerto serie." [Online]. Available: https://www.luisllamas.es/arduino-puerto-serie/

[46] "Comunicación Serial: Conceptos Generales - National Instruments." [Online]. Available: http://digital.ni.com/public.nsf/allkb/039001258CEF8FB686256E0F005888D1#hs

[47] "Puerto (informática)," Apr. 2018, page Version ID: 107322237. [Online]. Available: https://es.wikipedia.org/w/index.php?title=Puerto_(inform%C3%A1tica)&oldid=107322237

[48] "El bus SPI en Arduino." [Online]. Available: https://www.luisllamas.es/arduino-spi/

[49] "I2c," Apr. 2018, page Version ID: 106954230. [Online]. Available: https://es.wikipedia.org/w/index.php?title=I%C2%B2C&oldid=106954230

[50] "Comunicación serie - Wikipedia, la enciclopedia libre." [Online]. Available: https://es.wikipedia.org/wiki/Comunicaci%C3%B3n_serie

[51] E. Gómez, "Cómo funciona el Puerto Serie y la UART," Oct. 2017. [Online]. Available: https://www.rinconingenieril.es/funciona-puerto-serie-la-uart/

[52] W. b. d. Waard, "I2c For Arduino – SuperHouse Automation." [Online]. Available: https://www.superhouse.tv/i2c-for-arduino/

[53] "How does multimaster I2c work and its significance? - Electrical Engineering Stack Exchange." [Online]. Available: https://electronics.stackexchange.com/questions/115577/how-does-multimaster-i2c-work-and-its-significance?utm_medium=organic&utm_source=google_rich_qa&utm_campaign=google_rich_qa

[54] "El bus I2c en Arduino." [Online]. Available: https://www.luisllamas.es/arduino-i2c/

[55] "RS-232," Jul. 2017, page Version ID: 100427172. [Online]. Available: https://es.wikipedia.org/w/index.php?title=RS-232&oldid=100427172

[56] "RS-232 vs. TTL Serial Communication - SparkFun Electronics." [Online]. Available: https://www.sparkfun.com/tutorials/215

[57] "Bluehack - Proyectos / Comandos AT." [Online]. Available: http://bluehack.elhacker.net/proyectos/comandosat/comandosat.html

[58] "6 diferencias entre Servidores Cloud y VPS," Oct. 2013. [Online]. Available: https://blog.guebs.com/2013/10/02/servidor-cloud-versus-vps/

[59] LaInformacion, "Así son los nuevos drones de la DGT para 'freír' a multas a los malos conductores." [Online]. Available: https://www.lainformacion.com/espana/asi-son-los-nuevos-drones-de-la-dgt-para-freir-a-multas-a-los-malos-conductores/6342688

[60] Oscar, "Types of Multirotor," Nov. 2016. [Online]. Available: https://oscarliang.com/types-of-multicopter/

[61] "How To Pick The Best Multirotor Frame: Creating Your Own Drone." [Online]. Available: http://mydronelab.com/buyers-guide/how-to-pick-the-best-multirotor-frame.html

[62] "Wiki drone: elegir el frame correcto para el drone," May 2017. [Online]. Available: http://kit-drone.com/wiki-drone-elegir-el-frame-correcto-para-el-drone/

[63] "How to Make a Drone / UAV - Lesson 2: The Frame," Jan. 2015. [Online]. Available: https://www.robotshop.com/blog/en/make-uav-lesson-2-platform-14448

[64] "Tipos de drones aéreos," May 2016. [Online]. Available: http://dronespain.pro/tipos-de-drones-aereos/

[65] "Fotogrametría," Mar. 2018, page Version ID: 106616349. [Online]. Available: https://es.wikipedia.org/w/index.php?title=Fotogrametr%C3%ADa&oldid=106616349

[66] "APLICACIÓN DE LA FOTOGRAMETRÍA EN LA ING. AMBIENTAL." [Online]. Available: http://fotogrametriaaplicacioningamb.blogspot.com/2015/09/fotogrametria-y-su-relacion-con-la.html

[67] "Agricultura de precisión," Apr. 2018, page Version ID: 106704327. [Online]. Available: https://es.wikipedia.org/w/index.php?title=Agricultura_de_precisi%C3%B3n&oldid=106704327

[68] "Dron helicóptero: ¿desconocido y útil?" Dec. 2016. [Online]. Available: http://www.acgdrone.com/dron-helicoptero-desconocido-y-util/

[69] "Drones de ala fija: Parrot Disco," Mar. 2018. [Online]. Available: https://www.inventosygadgets.com/index.php/drones-de-ala-fija-parrot-disco-2/

[70] "G10 (material)," Feb. 2018, page Version ID: 826447967. [Online]. Available: https://en.wikipedia.org/w/index.php?title=G10_(material)&oldid=826447967

[71] Gago, "ESC para drones, ¿para qué sirven?" Apr. 2016. [Online]. Available: https://mobus.es/blog/esc-para-drones-para-que-sirven/

[72] iulian207, "How a brushless motor works (animation)." [Online]. Available: https://www.youtube.com/watch?v=bCEiOnuODac

[73] Anthony, "Diferencias entre motores con escobillas y brushless," Nov. 2016. [Online]. Available: https://clr.es/blog/es/diferencias-motores-con-escobillas-brushless/

[74] T. Moran, "brushless versus brushed electricmotors." [Online]. Available: http://ilikepie47.blogspot.com/2015/11/brushless-and-brushed-motors-are-use.html

[75] Oscar, "PWM and PPM Difference and Conversion," Nov. 2013. [Online]. Available: https://oscarliang.com/pwm-ppm-difference-conversion/

[76] "Firmware Variadores (ESC): Tipos Y Evolución," Aug. 2017. [Online]. Available: http://fpvmax.com/2017/08/24/firmware-variadores-esc/

[77] "¿Que es un ESC opto?" [Online]. Available: http://www.multicopters.es/foro/vbulletin/archive/index.php/t-3966.html

[78] Oscar, "What are ESC, UBEC and BEC," Jun. 2015. [Online]. Available: https://oscarliang.com/what-is-esc-ubec-bec-quadcopter/

[79] D. D. C. staff, "Firmware BLheli o SimonK," Jan. 2015. [Online]. Available: http://dronesdecarreras.com/firmware-blheli-o-simonk/

[80] "Variador electrónico (ESC): Qué es y cómo funciona," Dec. 2016. [Online]. Available: http://fpvmax.com/2016/12/21/variador-electronico-esc-funciona/

[81] "What is a switching regulator, switching voltage regulators - Future Electronics." [Online]. Available: http://www.futureelectronics.com/en/regulators-references/switching-regulators.aspx

[82] "Reguladores de voltaje conmutados - Convertidores DC-DC," Oct. 2015. [Online]. Available: https://unicrom.com/reguladores-de-conmutacion-convertidores-dc-d/

[83] "Ciclo de trabajo," Mar. 2014, page Version ID: 73049037. [Online]. Available: https://es.wikipedia.org/w/index.php?title=Ciclo_de_trabajo&oldid=73049037

[84] "Control de Servos con Java y Raspberry PI mediante I2c," Aug. 2013. [Online]. Available: https://unpocodejava.com/2013/08/19/control-de-servos-con-java-y-raspberry-pi-mediante-i2c/

[85] Oscar, "DShot1200 ESC Protocol," May 2017. [Online]. Available: https://oscarliang.com/dshot1200-esc-protocol/

[86] "MAVLink," Apr. 2018, page Version ID: 834512840. [Online]. Available: https://en.wikipedia.org/w/index.php?title=MAVLink&oldid=834512840

[87] "GPRSBee," Nov. 2014. [Online]. Available: http://xbee.cl/gprsbee/

[88] "MAVLink · Erle Robotics Gitbook." [Online]. Available: https://erlerobotics.gitbooks.io/erlerobot/es/mavlink/mavlink.html

[89] E. Santana, "MAVLink: protocolo de comunicación para drones." [Online]. Available: http://www.xdrones.es/mavlink/

[90] "how to read mavlink package .. is it ok to do it on my way?" [Online]. Available: https://stackoverflow.com/questions/40232083/how-to-read-mavlink-package-is-it-ok-to-do-it-on-my-way?utm_medium=organic&utm_source=google_rich_qa&utm_campaign=google_rich_qa

[91] "Protocols · MAVLink Developer Guide." [Online]. Available: https://mavlink.io/en/protocol/overview.html

[92] "Cyclic redundancy check," May 2018, page Version ID: 841473145. [Online]. Available: https://en.wikipedia.org/w/index.php?title=Cyclic_redundancy_check&oldid=841473145

[93] "The Cyclic Redundancy Check (CRC) for AX.25." [Online]. Available: http://practicingelectronics.com/articles/article-100003/article.php

[94] Othon Batista, "Cálculo de CRC - Parte 1." [Online]. Available: https://www.youtube.com/watch?v=D3wRqOtVWWY

[95] "mavlink en acción · Erle Robotics Gitbook." [Online]. Available: https://erlerobotics.gitbooks.io/erlerobot/es/mavlink/mavlinkaction.html

[96] "Mavlink 1 – Primeros pasos en la comunicación – | No Es Como Lo Cuentan..." [Online]. Available: https://noescomolocuentan.wordpress.com/2014/11/14/mavlink-1-primeros-pasos-en-la-comunicacion-2/

[97] "Giroscopio: qué es y qué aplicaciones tiene el efecto giroscópico," Sep. 2016. [Online]. Available: https://www.factoriadeingenieros.com/giroscopio/

[98] "El fundamento físico del acelerómetro | Átomos y bits," May 2014. [Online]. Available: http://atomosybits.com/la-fisica-tras-el-acelerometro/

[99] "¿Cuáles son las diferencias entre un giroscopio, acelerómetro y magnetómetro?" [Online]. Available: https://www.i-ciencias.com/pregunta/2053/cuales-son-las-diferencias-entre-un-giroscopio-acelerometro-y-magnetometro

[100] "Conoce la Diferencia entre el Giroscopio y el Acelerómetro," Jan. 2017. [Online]. Available: https://miracomohacerlo.com/diferencia-entre-el-giroscopio-y-el-acelerometro/

[101] "Giroscopio." [Online]. Available: http://cursos.mcielectronics.cl/giroscopio/

[102] "Electrónica del Automóvil. El sensor de guiñada." [Online]. Available: http://www.sapiensman.com/tecnoficio/electricidad/electricidad_del_automotor19.php

[103] Snorlax, "Tu telefono tiene giroscopio el sensor mas necesario para poder jugar Pokemon GO? Descubrelo aqui," Jun. 2016. [Online]. Available: http://www.pokenoticias.com/noticias/telefono-giroscopio-sensor-mas-necesario-poder-jugar-pokemon-go-descubrelo-aqui

[104] S. Reif-Acherman, "Toys as teaching tools in engineering: the cases of Newton's cradle and the," *Ingeniería y competitividad*, vol. 16, no. 2, pp. 189–198, Dec. 2014. [Online]. Available: http://www.scielo.org.co/scielo.php?script=sci_abstract&pid=S0123-30332014000200017&lng=en&nrm=iso&tlng=es

[105] "Brújula," Jun. 2018, page Version ID: 108509344. [Online]. Available: https://es.wikipedia.org/w/index.php?title=Br%C3%BAjula&oldid=108509344

[106] PatagoniaTec, "BMP085 Sensor De Tempreatura y Presion Atmosferica |." [Online]. Available: https://saber.patagoniatec.com/2014/06/bmp085-sensor-de-tempreatura-y-presion-atmosferica-arduino-argentina-ptec/

[107] Movistar, "¿Cuáles son y para qué sirven los sensores de los smartphone?" Mar. 2017. [Online]. Available: https://comunidad.movistar.es/t5/Blog-Movisfera/Cu%C3%A1les-son-y-para-qu%C3%A9-sirven-los-sensores-de-los-smartphone/ba-p/3159878

APÉNDICE A
Glosario

Siglas

BEC Battery Elimination Circuit. 16, 17

CRC Cyclic Redundance Check. 25

DGT Dirección General de Tráfico. 1

EEPROM Electrically Erasable Programmable Read-Only Memory. 3

ESC Electronic Speed Controller. 15, 16, 17

GCS Ground Control Station. 35, 38, 39, 45, 46, 51

HAT Hardware Attached on Top. 6

I²C Inter-Integrated Circuit. 23

IMU Inertial Measurement Unit. 21

PWM Pulse Width Modulation. VII, 17, 18

SSH Secure Shell. 36, 40

TCP Transmission Control Protocol. VIII, 36, 43, 45

UART Universally Asinchronous Receiver Transmitter. 23

UAV Unmanned Aerial Vehicles. IV

UAV Unmanned Aerial Vehicle. 1

UBEC Universal Battery Elimination Circuit. 17

VANT Vehículos Aéreos No Tripulados. III, 1

VANT Vehicles Aeris No Tripulats. III