

# Prototipo de Controlador de RPA vía redes móviles: Sky 3Guardian

Vianello Luis, Sergio Ignacio

Curso 2017-2018

Tutores:

Toni Adame, Albert Bel

Trabajo Final de Grado



Universitat  
Pompeu Fabra  
Barcelona

Escola  
Superior Politècnica



Yeah, I can fly.

*Anthony Edward Stark.*

*(Robert Downey Jr, Ironman, 30 de Abril de 2008)*



# Índice

0. Glosario de Terminología.....	11
0.1. Redes e Internet.....	11
0.2. Aeronáutica .....	13
0.3. Electrónica.....	14
0.4. Informática .....	15
0.5. Otros - Propios del Proyecto .....	16
1. Introducción .....	17
1.1. Resumen.....	17
1.2. Summary .....	18
1.3. Motivación .....	19
1.4. Objetivo del proyecto .....	20
2. Estado del Arte .....	21
2.1. Importancia tecnológica del proyecto .....	21
2.1.1. La tecnología de drones.....	21
2.1.2. The Internet of Things .....	21
2.1.3. Necesidad y propuesta de resolución .....	22
2.1.4. Selección de la herramienta de conectividad.....	22
2.1.5. Selección de la tecnología de comunicación .....	23
2.2. Identificación de la literatura específica.....	25
2.3. Competencia.....	26
2.3.1. Data Logging in Aerial Platform .....	26
2.3.2. UAV remote control system using mobile Internet.....	27
2.3.3. UAV supervision cloud platform based on Beidou GPS .....	28
2.3.4. Nodecopter: AR.Drone WebFlight.....	29
2.3.5. Flytbase .....	30
2.3.6. Matriz de Prototipos .....	31
2.3.7. Estudios públicos.....	31
3. Desarrollo del Prototipo .....	33
3.1. Concepto .....	33
3.2. Metodología .....	34
3.3. Descripción de las fases de desarrollo.....	36
3.3.1. Fase 1 .....	36
3.3.2. Fase 2 .....	40
3.3.3. Fase 3 .....	45

3.3.4. Fase 4 .....	55
3.4. Integración de fases.....	61
3.4.1. Subintegración Fase 2 y Fase 3 .....	61
3.4.2. Subintegración Fase 1 y Subfase 23 .....	62
3.4.3. Subintegración Fase 3 y Fase 4 .....	63
3.4.4. Subintegración Final .....	64
4. Evaluación del Prototipo .....	65
4.1. Resultados .....	65
4.1.1. Umbral mínimo de potencia.....	65
4.1.2. Delays.....	66
4.1.3. Peso del front-end.....	67
4.1.4. Presupuesto.....	68
4.1.5. Demostración de Funcionalidad .....	69
4.2. Dificultades y limitaciones encontradas.....	71
4.2.1. Dificultades de la Fase 1 .....	71
4.2.2. Dificultades de la Fase 2 .....	74
4.2.3. Dificultades de la Fase 3 .....	75
4.2.4. Dificultades en la fase 4 .....	78
5. Conclusiones .....	81
5.1. Conclusiones a partir de resultados.....	81
5.1.1. Objetivo Principal .....	81
5.1.2. Objetivos específicos.....	81
5.2. Propuestas de mejoras .....	83
5.2.1. Sistema de Streaming FPV.....	83
5.2.2. Control de Altitud y Aterrizaje de Emergencia.....	84
5.2.3. Global Positioning System (GPS) .....	85
5.2.4. Prototipo: Sky 4Guardian.....	86
5.3. Future Work .....	87
Anexos.....	89
Anexo 1. Material.....	89
Anexo 2. Vídeos.....	98
Anexo 3. Datasheets y Documentación.....	99
Referencias.....	101

# Índice de Figuras

Figura 1: Eje de la aeronave. Fuente: <a href="http://inforepuesto.com/">http://inforepuesto.com/</a> .....	13
Figura 2: Cobertura 3G de Vodafone. Fuente: <a href="http://www.vodafone.es">www.vodafone.es</a> .....	23
Figura 3: Cobertura 4G de Vodafone. Fuente: <a href="http://www.vodafone.es">www.vodafone.es</a> .....	23
Figura 4: Cobertura 5G de Vodafone. Fuente: <a href="http://www.vodafone.es">www.vodafone.es</a> .....	24
Figura 5: Esquema de prototipo US2017309088. Fuente: Base de Datos de Patentes .....	26
Figura 6: Esquema de prototipo CN107291098A. Fuente: Base de datos de Patentes .....	27
Figura 7: Esquema de prototipo CN107132852A. Fuente: Base de Datos de Patentes .....	28
Figura 8: Logo de Nodecopter. Fuente: <a href="http://www.nodecopter.com/">http://www.nodecopter.com/</a> .....	29
Figura 9: Visual de drone Flytbase. Fuente; <a href="https://flytbase.com/">https://flytbase.com/</a> .....	30
Figura 10: Circuito del proyecto. Fuente: Elaboración Propia.....	33
Figura 11: Esquema de metodología. Fuente: Elaboración Propia .....	35
Figura 12: Fase 1 de Integración. Fuente: Elaboración Propia .....	36
Figura 13: Fritzing de Fase 1. Fuente: Elaboración Propia.....	37
Figura 14: Circuito Físico de Fase 1. Fuente: Elaboración Propia.....	37
Figura 15: Método RIFE. Fuente: Elaboración Propia.....	38
Figura 16: Fase 2 de Integración. Fuente: Elaboración Propia .....	40
Figura 17: Fritzing de Fase 2. Fuente: Elaboración Propia .....	41
Figura 18: Circuito físico de Fase 2. Fuente: Elaboración Propia.....	41
Figura 19: Comunicación Serial con Modulo GPRS. Fuente: Elaboración Propia.....	44
Figura 20: Fase 3 de Integración. Fuente: Elaboración Propia .....	45
Figura 21: Conectividad Google Drive. Fuente: <a href="https://www.google.com/intl/es_ALL/drive/">https://www.google.com/intl/es_ALL/drive/</a> ..	46
Figura 22: Lobby ThingSpeak. Fuente: <a href="https://thingspeak.com/">https://thingspeak.com/</a> .....	47
Figura 23: Extracto código ThingSpeak. Fuente: Elaboración Propia .....	47
Figura 24: Lectura de variable exponencial en ThingSpeak. Fuente: Elaboración Propia.....	48
Figura 25: Servidor Web Sergio-Test. Fuente: <a href="https://dweet.io/">https://dweet.io/</a> .....	49
Figura 26: Servidor Web RIFE. Fuente: <a href="https://dweet.io/">https://dweet.io/</a> .....	49
Figura 27: Lectura Serial del Servidor RIFE. Fuente: Elaboración Propia .....	53
Figura 28: Extracto de Servidor RIFE. Fuente: Elaboración Propia .....	54
Figura 29: Fase 4 de Integración. Fuente: Elaboración Propia .....	55
Figura 30: Fritzing de Fase 4. Fuente: Elaboración Propia .....	56
Figura 31: Circuito Físico de Fase 4. Fuente: Elaboración Propia.....	56
Figura 32: Placa Ethernet en Arduino. Fuente: Elaboración Propia .....	57
Figura 33: Extracto de código de Fase 4. Fuente: Elaboración Propia .....	58
Figura 34: Método RIFE en Código. Fuente: Elaboración Propia .....	59
Figura 35: Envío de datos vía Ethernet. Fuente: Elaboración Propia.....	60
Figura 36: Delays mínimos. Comunicación Exitosa .....	61
Figura 37: Delays mínimos. Comunicación por debajo del delay mínimo: Colisión .....	61
Figura 38: Método RIFE Inverso en Código. Fuente: Elaboración Propia .....	62
Figura 39: Respuesta de Servidor RIFE: Estabilización. Fuente: Elaboración Propia.....	63
Figura 40: Respuesta de Servidor RIFE: Potencia++. Fuente: Elaboración Propia .....	63
Figura 41: : Respuesta de Servidor RIFE: Guiñada--. Fuente: Elaboración Propia .....	63
Figura 42: Respuesta de Servidor RIFE: Cabeceo--. Fuente: Elaboración Propia .....	63
Figura 43: Respuesta de Servidor RIFE: Alabeo++. Fuente: Elaboración Propia .....	63
Figura 44: Respuesta de Servidor RIFE: Potencia-- & Alabeo++. Fuente: Elaboración Propia.	63
Figura 45: Fritzing de Circuito completo. Fuente: Elaboración Propia .....	64
Figura 46: Demostración, Guiñada en Joystick. Fuente: Elaboración Propia.....	69
Figura 47: Demostración, Guiñada en Servidor RIFE. Fuente: Elaboración Propia.....	69

Figura 48: Demostración, Guiñada en Serial. Fuente: Elaboración Propia.....	70
Figura 49: Demostración, Guiñada en Simulador. Fuente: Elaboración Propia .....	70
Figura 50: Pines Analógicos de Arduino UNO. Fuente: Elaboración Propia .....	71
Figura 51: CPPM en una transmisora de Radio. Fuente: Elaboración Propia.....	72
Figura 52: Señales en Terminales. Fuente: Elaboración Propia.....	72
Figura 53: Integridad de señalización múltiple. Fuente: Elaboración Propia .....	73
Figura 54: Batería insuficiente. Fuente: Elaboración Propia .....	74
Figura 55: Modelos de Usuarios ThingSpeak. Fuente: <a href="https://thingspeak.com/prices">https://thingspeak.com/prices</a> .....	75
Figura 56: Twitter de Dweet.io. Fuente: <a href="https://twitter.com/dweet_io">https://twitter.com/dweet_io</a> .....	76
Figura 57: Esquema de Ataque DDOS. Fuente: <a href="https://www.hd-tecnologia.com">https://www.hd-tecnologia.com</a> .....	76
Figura 58: Esquema de Insuficiencia de Datos Móviles. Fuente: Elaboración Propia.....	77
Figura 59: Conexión bridge para Arduino. Fuente: Elaboración Propia.....	78
Figura 60: Cámara HD 700tvl. Fuente: <a href="http://www.tri-ed.com/">http://www.tri-ed.com/</a> .....	83
Figura 61: Sensor de distancia LIDAR. Fuente: <a href="http://www.xdrones.es/">http://www.xdrones.es/</a> .....	84
Figura 62: Módulo GPS M8N 8N. Fuente: <a href="https://www.dzduino.com/">https://www.dzduino.com/</a> .....	85
Figura 63: Esquema de Prototipo Sky 4Guardian. Fuente: Elaboración Propia .....	86

# Índice de tablas

Tabla 1: 3G vs 4G vs 5G en España. Fuente: elaboración propia.....	24
Tabla 2: Prototipo Sky 3Guardian VS La Competencia. Fuente: Elaboración Propia.....	31
Tabla 3: Valores de Data_numbers[]. Fuente: Elaboración Propia.....	62
Tabla 4: UMP de Potencia. Fuente: Elaboración Propia.....	65
Tabla 5: UMP de Alabeo/Cabeceo/Guiñada. Fuente: Elaboración Propia.....	65
Tabla 6: Delays en Setup, Front-end. Fuente: Elaboración Propia .....	66
Tabla 7: Delays en Setup, Back-end. Fuente: Elaboración Propia.....	66
Tabla 8: Delays en Loop, Front-end. Fuente: Elaboración Propia .....	66
Tabla 9: Peso del front-end. Fuente: Elaboración Propia.....	67
Tabla 10: Presupuesto del Proyecto. Fuente: Elaboración Propia.....	68
Tabla 11: Presupuesto del Future Work. Fuente: Elaboración Propia .....	68

Página en blanco intencional

# 0. Glosario de Terminología

## 0.1. Redes e Internet

**3G:** Es la abreviación de tercera generación de transmisión de voz y datos a través de telefonía móvil mediante UMTS (Universal Mobile Telecommunications System o servicio universal de telecomunicaciones móviles).

**4G:** En telecomunicaciones, 4G es la sigla utilizada para referirse a la cuarta generación de tecnologías de telefonía móvil. La 4G está basada completamente en el protocolo IP, siendo un sistema y una red, que se alcanza gracias a la convergencia entre las redes de cable e inalámbricas.

**5G:** En telecomunicaciones, 5G son las siglas utilizadas para referirse a la quinta generación de tecnologías de telefonía móvil. Es la sucesora de la tecnología 4G. Actualmente se encuentra sin estandarizar y las empresas de telecomunicación están desarrollando sus prototipos. Está previsto que su uso común sea en 2020.

**API:** Las siglas API vienen del inglés Application Programming Interface. Una API es un conjunto de funciones y procedimientos que cumplen una o muchas funciones con el fin de ser utilizadas por otro software.

**Delay:** El retardo (en inglés delay o lag) es una demora que se produce en una telecomunicación desde que se envía información desde un origen hasta que llega a su destino. Aunque este retardo puede deberse a una alta latencia de la red, también puede producirse debido a que no exista suficiente potencia de procesamiento en el servidor o cliente destino con el que se establece la comunicación.

**Ethernet:** Tecnología que define las características del cableado y señalización de nivel físico y los formatos de trama del nivel de enlace de datos del modelo OSI.

**GET:** Método de Petición HTTP. Solicita una representación de un recurso específico. Las peticiones que usan el método GET sólo deben recuperar datos

**HTTP:** El http son las siglas de “Hypertext Transfer Protocol” es un protocolo de transferencia donde se utiliza un sistema mediante el cual se permite la transferencia de información entre diferentes servicios y los clientes que utilizan páginas web.

**Internet of Things:** La internet de las cosas (IoT, por sus siglas en inglés) es un sistema de dispositivos de computación interrelacionados, máquinas mecánicas y digitales, objetos, animales o personas que tienen identificadores únicos y la capacidad de transferir datos a través de una red, sin requerir de interacciones humano a humano o humano a computadora.

**IP:** El protocolo de internet (en inglés Internet Protocol o IP) es un protocolo de comunicación de datos digitales clasificado funcionalmente en la capa de red según el modelo internacional OSI.

**Libre acceso:** Referente a un Servidor o Web. Que no requiera de una suscripción.

**POST:** Método de Petición HTTP. Se utiliza para enviar una entidad a un recurso en específico, causando a menudo un cambio en el estado o efectos secundarios en el servidor.

**Proxy:** Un proxy es un ordenador intermedio que se usa en la comunicación de otros dos. La información (generalmente en Internet) va directamente entre un ordenador y otro.

**End-to-end:** Se refiere a todo un conjunto desde el inicio hasta el final.

**Readkey:** Es un código de identificación que permite al usuario consultar cierta información.

**Ruido:** En comunicación, se denomina ruido a toda señal no deseada que se mezcla con la señal útil que se quiere transmitir.

**Servidor Web:** Los servidores web sirven para almacenar contenidos de Internet y facilitar su disponibilidad de forma constante y segura. Cuando visitas una página web desde tu navegador, es en

realidad un servidor web el que envía los componentes individuales de dicha página directamente a tu ordenador.

**Standby:** En modo de espera.

**Timeout:** Son eventos que se disparan al superar cierto periodo de tiempo preestablecido.

**URL:** Uniform Resource Locator (Localizador Uniforme de Recursos). Se trata de la secuencia de caracteres que sigue un estándar y que permite denominar recursos dentro del entorno de Internet para que puedan ser localizados.

**WriteKey:** Es un código de identificación que permite al usuario realizar ciertas modificaciones de información.

## 0.2. Aeronáutica

**Alabeo:** En inglés Roll es la rotación del drone sobre el eje X en un sistema tridimensional (agujas del reloj). Ver imagen de Glosario: Eje de la aeronave.

**Cabeceo:** En inglés Pitch es la rotación del drone sobre el eje Z en un sistema tridimensional (levantar y bajar la cabeza). Ver imagen de Glosario: Eje de la aeronave.

**Control a simple vista:** Se refiere a la distancia prudencial máxima en la que se puede operar con seguridad la aeronave usando solo la vista como método de control visual.

**Drones / RPA / UAV:** Se domina Drone/Drones a los vehículos aéreos no tripulados (UAV), la mayoría de uso militar. Actualmente, en Europa, también se les denomina como RPA (Remoted Piloted Aircraft).

**Entrada en pérdidas:** Es el fenómeno aerodinámico que consiste en la disminución súbita de la fuerza de sustentación. En multicópteros la entrada en perdida no depende del ángulo de ataque de los alerones (como es el caso de aviones convencionales) sino en la velocidad de caída y falta de recuperación del horizonte artificial.

**FPV:** Del inglés First Person View, se refiere a la capacidad de ver lo que la aeronave en primera persona.

**Guiñada:** En inglés Yaw es la rotación del drone sobre el eje Y en un sistema tridimensional (girar la cabeza hacia la derecha o izquierda). Ver imagen de Glosario: Eje de la aeronave.

**Hélices:** Son las aspas de los multicópteros.

**Multirrotores:** Un multirrotor o multicóptero es un helicóptero con más de dos rotores. Los multirrotores de cuatro motores se denominan Cuadracopteros mientras que los de seis se denominan Hexacopteros.

**Operador / Piloto:** Es quien controla la aeronave ya sea local o remotamente.

**Potencia:** En inglés Throttle, es la velocidad de rotación de las hélices en drones manuales. En otro tipo de drones puede determinar la altura o diferencia de altitud.

**Ráfagas de potencia:** Se trata de una técnica en aviación de multicópteros que consiste en dejar de dar potencia a la aeronave para que experimente una caída libre durante varios metros y suministrar súbitamente potencia para reiniciar la velocidad de caída. Esta técnica se utiliza en situaciones de emergencia donde el drone, que tiene pocos niveles de batería disponible, tiene la necesidad inmediata de aterrizar.

**Rango de control:** Rango desde el operador hasta el drone. Al llevar al límite de este rango, se le denomina como Rango Máximo de Control.

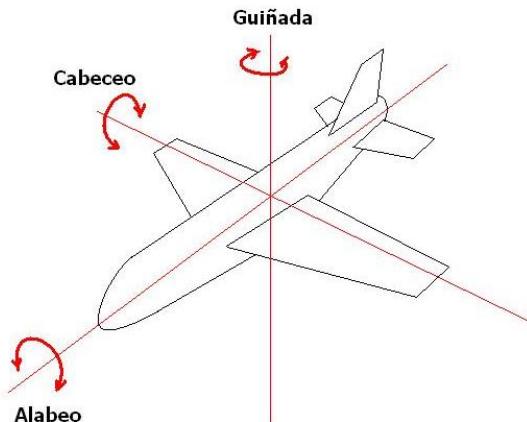


Figura 1: Eje de la aeronave. Fuente: <http://inforepuesto.com/>

## 0.3. Electrónica

**Arduino:** Es una plataforma de hardware libre, basada en una placa con un microcontrolador y un entorno de desarrollo, diseñada para facilitar el uso de la electrónica en proyectos multidisciplinares.

**Baudios:** El baudio (en inglés baud) es una unidad de medida utilizada en telecomunicaciones, que representa el número de símbolos por segundo en un medio de transmisión digital. Se establecen en el SETUP del código y solo se puede mantener una conversación entre dispositivos configurados para el mismo nivel de baudios.

**Botón de RESET:** Es un botón localizado en una de las esquinas del Arduino que ejecuta un reinicio del microprocesador.

**Comandos AT:** Los comandos AT, a diferencia de los lenguajes de programación convencionales, son instrucciones codificadas que conforman la comunicación entre un usuario y un terminal modem o telefonía móvil GSM. Todos los teléfonos móviles GSM poseen un juego de comandos AT específico que sirve de interfaz para configurar y proporcionar instrucciones.

**GPRS/GSM:** El GPRS (General Packet Radio Service) es una extensión de la tecnología de comunicaciones móviles GSM. En ella la información es dividida en pequeños bloques, los que posteriormente se reagrupan al llegar a destino.

**GPS:** Se conoce como GPS a las siglas “Global Positioning System” que en español significa “sistema de posicionamiento global”. El GPS es un sistema de navegación basado en 24 satélites (21 operativos y 3 de respaldo), en órbita sobre el planeta tierra que envía información sobre la posición de una persona u objeto en cualquier horario y condiciones climáticas.

**Joystick:** Palanca de control que permite desplazar manualmente, y con gran rapidez, el cursor en una pantalla de computadora, videojuego o simulador.

**LOOP:** Es una sección de código de Arduino IDE donde ocurre la mayor parte de iteraciones del código. A menos que se especifique, es un ciclo infinito de iteraciones.

**PIN:** Son salidas y entradas de señales digitales y/o analógicas. Pueden recibir o enviar voltaje para activar o ser parte de un circuito.

**Prototipo:** Se refiere a cualquier tipo de máquina en pruebas, o un objeto diseñado para una demostración de cualquier tipo.

**Señal Analógica:** Una señal analógica es un voltaje o corriente que varía suave y continuamente. Una onda sinusoidal es una señal analógica de una sola frecuencia. Los voltajes de la voz y del video son señales analógicas que varían de acuerdo con el sonido o variaciones de la luz que corresponden a la información que se está transmitiendo.

**Señal Digital:** Las señales digitales, en contraste con las señales analógicas, no varían en forma continua, sino que cambian en pasos o en incrementos discretos. La mayoría de las señales digitales utilizan códigos binarios o de dos estados.

**Serial:** Se refiere a la salida de comunicación del Arduino. Si está conectado vía USB a un ordenador, la comunicación Serial da prioridad al USB. En caso contrario se utiliza para recibir y transmitir información por los pines 0 y 1.

**SETUP:** Es una sección de código de Arduino IDE donde se establecen valores o Seriales. Solo se entra a esta sección cuando se enciende el Arduino.

**SIM900:** Es una tarjeta ultra compacta de comunicación inalámbrica que soporta comunicaciones GPRS. Permite al microprocesador hacer conectividad con redes móviles.

**SoftwareSerial:** Es una librería disponible en Arduino IDE que permite convertir dos PINES cualquiera en PINES de recepción (TX) y envío de datos (RX) normalmente reservados para los pines 0 y 1.

**SP\_F3:** Es el nombre de la controladora de vuelo usada. Más información en Anexos 1. Materiales.

**Tarjeta SIM:** Una tarjeta SIM (acrónimo en inglés de Subscriber Identity Module, en español Módulo de Identificación de Suscripción) es una tarjeta inteligente desmontable usada en teléfonos móviles

**Voltaje:** Es la presión que una fuente de suministro de energía eléctrica o fuerza electromotriz ejerce sobre las cargas eléctricas o electrones en un circuito eléctrico cerrado.

## 0.4. Informática

**Archivo.txt:** Es un tipo de archivo que contiene un formato mínimo, pero cumple con las definiciones de formato aceptadas por la terminal del sistema y los editores de texto simple.

**Array:** Es un tipo de dato estructurado que permite almacenar un conjunto de datos homogéneo, es decir, todos ellos del mismo tipo y relacionados. Cada uno de los elementos que componen un vector puede ser de tipo simple como caracteres, entero o real, o de tipo compuesto o estructurado como son vectores, estructuras, listas...

**Byte:** El byte es la unidad fundamental de datos en los ordenadores personales, un byte son ocho bits contiguos. Es también la unidad de medida básica para memoria, almacenando el equivalente a un carácter.

**C/C++:** C es un lenguaje de programación de propósito general que ofrece economía sintáctica, control de flujo y estructuras sencillas y un buen conjunto de operadores. C++ es un lenguaje imperativo orientado a objetos derivado del C.

**Consola Virtual:** Se refiere a la forma de controlar un dispositivo mediante una interfaz informática.

**Debugging:** Proviene del inglés y significa Depuración. Es la corrección de errores en la programación.

**Flush:** Proceso en el cual se libera la memoria dinámica previamente reservada.

**GitHub:** GitHub es una plataforma de desarrollo colaborativo de software para alojar proyectos utilizando el sistema de control de versiones Git. El código se almacena de forma pública, aunque también se puede hacer de forma privada, creando una cuenta de pago.

**int:** Es un tipo de dato que puede representar un subconjunto finito de los números enteros entre -32.768 y 32768.

**Librerías:** Es un conjunto de implementaciones funcionales, codificadas en un lenguaje de programación, que ofrece una interfaz bien definida para la funcionalidad que se invoca.

**Simulador:** Un simulador es un aparato, por lo general informático, que permite la reproducción de un sistema. Los simuladores reproducen sensaciones y experiencias que en la realidad pueden llegar a suceder.

**Trigger:** Los Triggers o Disparadores son objetos que se asocian con tablas y se almacenan en la base de datos. Su nombre se deriva por el comportamiento que presentan en su funcionamiento, ya que se ejecutan cuando sucede algún evento sobre las tablas a las que se encuentra asociado.

**Variable:** En programación, una variable está formada por un espacio en el sistema de almacenaje (memoria principal de un ordenador) y un nombre simbólico (un identificador) que está asociado a dicho espacio. Ese espacio contiene una cantidad de información conocida o desconocida, es decir un valor.

**Loop:** Traducido como bucle de programación se refiere a una instrucción escrita en el lenguaje de programación que dicta la repetición constante de la misma acción.

## 0.5. Otros - Propios del Proyecto

**Front-end:** En el proyecto, se refiere a la parte del prototipo que actúa como receptor de la información. Es decir, el sistema de la controladora de vuelo y por extensión el drone.

**Back-end:** En el proyecto, se refiere a la parte del prototipo que actúa como emisor de la información. Es decir, el sistema de joystick y por extensión el operador.

**Señales básicas:** En el proyecto, se refiere a las 4 señales de control de la aeronave: Potencia, Alabeo, Cabeceo y Guiñada.

**Estabilización en vuelo:** Cuando las señales del drone recuperan sus valores de estabilización promedio (ir a la sección 4.1.1 Umbral mínimo de potencia para más información)

# 1. Introducción

## 1.1. Resumen

Desde el año 2012, la palabra **drone** ha experimentado una creciente tendencia en los intereses de la sociedad. En concreto los protagonistas de esta tendencia son los RPA (Remoted Piloted Aircraft), que a día de hoy están siendo utilizados en diversos sectores como fotografía, agricultura, climatología, entre otros. “Docenas de compañías, que fabrican aeronaves pequeñas, controladas remotamente y con cámaras acopladas, han surgido en los últimos años” [1], colocando a estos dispositivos al alcance de consumidores finales y sectores comerciales.

Se han desarrollado un gran número de aplicaciones para los RPA que han puesto a prueba la utilidad y versatilidad de esta herramienta. Desde tareas relativamente simples como fotografía, envío de paquetes y vigilancia hasta tareas de gran complejidad en los ámbitos de la construcción, agricultura, minería, búsqueda y rescate e investigaciones científicas [2].

No obstante, una de las principales limitaciones de esta tecnología es el rango de control que puede haber entre el RPA y la consola de control manual, el cual limita una distancia máxima promedio de 6.000 metros de radio entre el RPA y el operador [3].

El prototipo desarrollado en este proyecto logra eliminar esta limitación al utilizar un sistema basado en comunicaciones móviles de 3G/4G. Dicho sistema garantiza un enlace constante entre el RPA y el operador. Por su parte el operador, emplea una consola virtual para el control del RPA vía Internet.

Este prototipo busca eliminar el factor “rango máximo” del proceso de comunicación entre el RPA y operador por lo que potencialmente se puede controlar el RPA **desde cualquier parte del mundo** siempre que ambas entidades tengan acceso constante a Internet.

El nombre asignado a este prototipo (el drone diseñado bajo el sistema de comunicación propuesto) y con el que se le hará referencia durante el proyecto es el de **Sky 3Guardian**, nombre inspirado por la unión de los nombres de Sky Guardian (maquinas voladoras en el videojuego The Legend of Zelda: Breath of the Wild) y 3G.

## 1.2. Summary

Since the year 2012, the word **drone** has experienced a growing trend in society's interest. In concrete the protagonists of this trend have been the RPA (Remoted Piloted Aircraft), which nowadays are being used in several working sectors like photography, agriculture, climatology, among others. "Dozens of companies, that manufacture small aircrafts, remotely controlled and with an attached camera, have emerged in recent years" [1], placing these devices within reach of final consumers and business sectors.

A great number of applications have been developed for the RPA that have put to the test the utility and versatility of this tool. From relatively simple tasks such as photography, shipment of packets and vigilance to very complex ones on the fields of construction, agriculture, mining, search and rescue and scientific studies [2].

Nonetheless, one of the main limitations of this technology is the control range between the RPA and the manual controller, which limits a maximum distance radio of approximately 6,000 meters between the RPA and the operator [3].

The developed prototype in this project manages to eliminate this limitation as it makes use of a system based on mobile communications 3G/4G. This system guarantees a constant link between the RPA and the operator. For his part the operator, makes use of a virtual console for the RPA control through the Internet.

This prototype has the objective of removing the "máximo range" factor from the communication process between the RPA and the operator which means that the RPA could potentially be controlled **from anywhere in the world** as long as both entities have constant Internet Access.

The name assigned to this prototype (the drone designed under the proposed communication system) and with which it will be referred during the project is **Sky 3Guardian**, a name inspired by the unión of the names Sky Guardian (flying enemies in the video game The Legend of Zelda: Breath of the Wild) and 3G.

## **1.3. Motivación**

Este proyecto significa la culminación de mis estudios de Grado. Representa una unión entre los conceptos de Ingeniería Telemática en los que he presentado mayor interés y talento, Redes y Sistemas, junto con mi pasatiempo favorito que es el pilotaje de aeronaves remotas.

En 2014 realicé cursos intensivos de pilotaje donde aprendí los conceptos relacionados con el manejo de aeronaves, climatología, fundamentos de aerodinámica, leyes y restricciones legales. Siempre he sentido una pasión por ver el mundo desde el cielo y este impulso fue el que me llevó a certificarme como piloto oficial de aeronaves remotas.

Fue a partir de aquí cuando pensé en desarrollar un proyecto que resolviera necesidades o problemas del campo del pilotaje de aeronaves remotas aplicando conceptos y habilidades adquiridas durante las distintas asignaturas y prácticas de la carrera. De esta forma presenté una propuesta de proyecto de Grado que engloba los principales aspectos de ambas facetas.

## 1.4. Objetivo del proyecto

Este proyecto tiene como objetivo desarrollar un nuevo sistema de control de vuelo de aeronaves multirotor pilotadas remotamente, comúnmente denominadas como drones, mediante el uso de redes móviles que se conectan a través de Internet. Los drones son controlados gracias a diferentes peticiones HTTP generadas desde el sistema de control. Como resultado del proyecto, se presenta un prototipo que muestra la factibilidad de operar bajo esta nueva modalidad.

Objetivos específicos:

- El delay desde que se ejecuta una acción en el back-end hasta que se ejecuta en el front-end ha de ser menor a 2 segundos.
- La comunicación desde el back-end al front-end no debe ser verse interrumpida durante no menos de 3 minutos.
- Se debe desarrollar una funcionalidad de estabilización aérea en caso de algún tipo de fallo en la comunicación del back-end al front-end.
- El montaje completo del front-end debe pesar menos de 1.5 kilogramos (peso promedio de seguridad para elevar la aeronave).
- El presupuesto del proyecto debe mantenerse por debajo de los 150 euros.

Aunque, inicialmente el objetivo del proyecto apuntaba a conseguir un prototipo real y físico de un drone para realizar demostraciones de vuelo en vivo, se ha determinado que por motivos de seguridad tanto del drone como de los espectadores cercanos **el proyecto se limitará a realizar las demostraciones de vuelo en un simulador de vuelo virtual**.

El no tener el control de un drone (especialmente manual) durante un intervalo de tiempo mayor a 2 segundos en cada iteración podría producir accidentes.

Además, también hay que cumplir con el objetivo de presupuesto mencionado anteriormente. Habría que añadir el coste de compra de piezas, como también el ensamblado del mismo lo cual elevaría considerablemente los costes por encima de lo calculado.

## 2. Estado del Arte

### 2.1. Importancia tecnológica del proyecto

#### 2.1.1. La tecnología de drones

Como se mencionó anteriormente, es innegable que la tecnología se ha vuelto una parte crucial de nuestro día a día. Continuamente buscamos nuevas e innovadoras ideas que resuelvan todo tipo de trabajos, problemas y situaciones. La tecnología de drones ha estado presente como una herramienta militar desde relativamente mucho tiempo [4], pero en la actualidad hay nuevas y distintas formas en que los drones pueden ayudarnos en nuestro día a día.

La accesibilidad que ha ganado esta tecnología en la última década ha despertado un gran interés. El poder explorar los cielos siempre ha sido una de las más grandes ambiciones del ser humano, y solo basta con reflexionar que hace apenas 115 años aconteció el primer vuelo exitoso de una aeronave tripulada gracias a los hermanos Orville Wright y Wilbur Wright [5].

A pesar de que aún existen leyes que prohíben el uso de drones en la mayoría de circunstancias, hoy en día la popularidad de estos aparatos ha experimentado un aumento considerable.

Hasta hace 10 años los usos de los drones ha sido de carácter exclusivamente militar, sin embargo, desde aproximadamente el año 2009, se han explorado multitudes de aplicaciones civiles:

- Fotografía y cine
- Gestión de Transito.
- Uso de bomberos.
- Envío de paquetes.
- Estudios climatológicos.
- Actividades agrícolas.
- Revisiones arquitectónicas

Todos estos usos abren las puertas a funciones que creímos imposibles o muy costosas de resolver.

#### 2.1.2. The Internet of Things

A la par del uso de Drones, otra tendencia que ha crecido exponencialmente durante los últimos diez años es la interconexión de dispositivos de todo tipo utilizando Internet como plataforma de interconexión. Esto es lo que se conoce como The Internet Things (IoT).

IoT se refiere a la interconexión en vivo de los objetos cotidianos, que a menudo están equipados con inteligencia propia. “Internet of Things (IoT) es un concepto que se basa en la interconexión de cualquier producto con cualquier otro de su alrededor. Desde un libro hasta el frigorífico de tu propia casa. El objetivo es hacer que todos estos dispositivos se comuniquen entre sí y, por consiguiente, sean más inteligentes e independientes” [6].

Así pues, retomando la sección anterior donde se plantea el sistema actual de control remoto de drones y entendiendo el alcance de IoT, surgió la inquietud de proponer una mejora.

#### ¿Es posible aplicar los conceptos básicos de IoT al pilotaje de drones?

Esta fue la pregunta que despertó el interés en llevar a cabo este proyecto y desarrollar un prototipo.

### 2.1.3. Necesidad y propuesta de resolución

Para comenzar a plantear el proyecto, fue necesario determinar una debilidad, un problema, una limitación actual del concepto del pilotaje de drones del que hubiera un interés general por parte de los interesados en este tema, en ser resuelta.

Se pensó entonces en algunos de los problemas que existen actualmente en el pilotaje de drones, que se listan a continuación:

- Corta vida de batería (promedio de 10 minutos).
- Fragilidad ante adversos fenómenos meteorológicos.
- Alto coste total de pilotaje (entre 1.000 y 2.000 euros para FPV).
- Invasión de la privacidad o de zonas aéreas restringidas.
- **Limitación del alcance de control por Radio (máximo 6.000 metros).**

Centrándonos en esta última, se pensó en el IoT como una solución plausible. De entre estas propuestas, se determinó que la extensión del rango de control mediante la herramienta de Internet era una idea interesante a aplicar. Por otra parte, por tratarse este proyecto de un Trabajo Final de Grado, se aplicarían los conocimientos sobre redes obtenidos a través del grado de Ingeniería Telemática.

Por lo tanto, se estableció en que se usarían las redes (Internet) como tecnología seleccionada para resolver los problemas de rango de control actuales.

### 2.1.4. Selección de la herramienta de conectividad

El siguiente paso consistió en determinar las herramientas de conectividad sobre las que se plantearía el proyecto. Se evaluaron las siguientes tecnologías:

- La primera opción evaluada fue la utilización de redes inalámbricas. A pesar de ser un estándar, el alcance de las señales Wi-Fi tienen un rango promedio de 30 metros que está 200 veces por debajo del máximo ya existente. Por lo tanto esta opción fue descartada.
- La segunda opción pensada fue la popular tecnología de conectividad Bluetooth. Esta tecnología presenta el mismo problema de limitación de distancia que Wi-Fi, aproximadamente 30 metros. Esta opción, también fue descartada.
- La tercera tecnología evaluada fue la de las comunicaciones móviles. Dado que los dispositivos móviles se han vuelto una parte tan esencial de nuestro día a día, la cobertura de redes móviles es una de las mayores prioridades de las operadoras. A primera vista, la tecnología móvil fue la opción más viable que se propuso y se planteó para ser aplicada en un sistema de control remoto de drones.

Una vez seleccionada la tecnología de comunicación, apareció una nueva limitación: **solo se tendrá control del drone mientras hubiese cobertura de señal móvil.** A pesar de esta limitación, se tomó la decisión de desarrollar el proyecto utilizando la tecnología móvil.

## 2.1.5. Selección de la tecnología de comunicación

Dado que las redes móviles se clasifican actualmente en 3 tecnologías 3G, 4G y 5G, se estudiaron cada una de ellas para determinar cuál fue la óptima.

La mayor diferencia entre estas tecnologías es el alcance y la velocidad de transmisión que ofrecen. No obstante es importante recordar que el tipo de comunicación propuesta en la sección de Objetivo Principal, peticiones HTTP, es un tipo de transmisión que requiere de un ancho de banda relativamente bajo de aproximadamente 2Mbps<sup>1</sup>. Simplemente se trata de una señalización de la acción que se desea que ejecute el drone a continuación. Esto sugirió que, dado que la solución propuesta es ampliar el rango de control del drone, la velocidad de conexión pasa a ser un parámetro de relevancia interior al de la cobertura de señal.

Para evaluar las tecnologías de redes móviles existentes actualmente en España, se seleccionó a la empresa Vodafone por tener actualmente accesibilidad a la red mediante una tarjeta SIM ya contratada y preparada para realizar pruebas.

Estos fueron los rangos de cobertura de cada una de las tecnologías 3G/4G/5G en España para la operadora Vodafone:



Figura 2: Cobertura 3G de Vodafone. Fuente: [www.vodafone.es](http://www.vodafone.es)



Figura 3: Cobertura 4G de Vodafone. Fuente: [www.vodafone.es](http://www.vodafone.es)

<sup>1</sup> Estimado de Ancho de Banda necesario – Basic Web Use: <https://www.homesc.com/estimating-bandwidth-needs>



Figura 4: Cobertura 5G de Vodafone. Fuente: [www.vodafone.es](http://www.vodafone.es)

La tabla 1 resume las tecnologías, alcance y velocidad de transmisión ofrecidas por Vodafone en el territorio español:

Tabla 1: 3G vs 4G vs 5G en España. Fuente: elaboración propia

Tecnología	Alcance	Velocidad de transmisión
<b>3G</b>	~ 90% de la península	2 Mbps
<b>4G</b>	~ 60% de la península	200 Mbps
<b>5G</b>	~ 10% de la península	1.000 Mbps

Con esta información se pudo concluir que de las tecnologías actuales, la que más se adecúa a las necesidades del proyecto son la **3G** y **4G** por su velocidad de transmisión y cobertura.

Hoy en día prácticamente todas las tarjetas SIM se comunican con su operadora mediante conexiones 3G o 4G y próximamente 5G. Esto garantiza que cualquier tarjeta SIM utilizada en el prototipo deberá tener suficiente velocidad de conexión para cumplir con las necesidades de peticiones HTTP.

No obstante durante el desarrollo del prototipo físico, debido a limitaciones de material, fue necesario el uso de tecnología GPRS (2G) la cual tiene velocidades de transmisión de aproximadamente 14.4 Kbps. Esto presentó importantes limitaciones en términos de delay que se explicarán más detalladamente en la sección de Resultados.

## 2.2. Identificación de la literatura específica

Con el fin de determinar qué tan innovadora es la propuesta del proyecto, se hicieron distintas búsquedas en varias bases d

e datos. El objetivo de dichas búsquedas fue determinar si a día de hoy, existe en España y Europa, algún producto o idea **patentada** lo suficientemente similar al prototipo Sky 3Guardian.

Para esto se consultaron las siguientes bases de datos:

- **Base de datos española:** Oficina Española de Patentes y Marcas<sup>2</sup>. La OEPM es el Organismo Público responsable del registro y la concesión de las distintas modalidades de Propiedad Industrial.
- **Base de datos Internacional:** Oficina Europea de Patentes Espacenet<sup>3</sup>. Es un servicio gratuito en línea para buscar patentes y solicitudes de patentes. Fue desarrollado por la Oficina Europea de Patentes (EPO) junto con los estados miembros de la Organización Europea de Patentes.
- **Base de datos Privada:** Thomson Innovation y Derwent World Patents Index<sup>4</sup>. Es una base de datos que contiene solicitudes de patentes y subvenciones de 44 de las autoridades de emisión de patentes del mundo.

Para realizar las búsquedas en las diversas bases de datos documentales, se utilizaron las siguientes categorías de la CIP<sup>5</sup> (Clasificación Internacional de Patentes):

- **B64C39/024:** Realización de Operaciones; Transporte; Vehículos en General, Aeronaves; Aviación; Cosmonautas, Aeroplanos; Helicópteros, Componentes de aviación; Multirrotores;
- **B64C2201/024:** Helicópteros, o autogiros.
- **B64C2201/146:** Control remoto.
- **H04L67/12:** Adaptaciones en entornos de red para aplicaciones personales o especiales, por ejemplo redes médicas, redes de sensores, redes en coches o control remoto de datos.
- **H04L69/16:** Protocolo de control de Transmisión/Protocolo de Internet [TCP/IP] o Protocolo de User Datagram [UDP].
- **H04N7/185:** Desde un dispositivo móvil, por ejemplo control remoto.
- **H04L43/0811:** Conectividad.

En el caso de las bases de datos de Espacenet y Thomson Innovation, las palabras claves de búsqueda fueron:

- Unmanned aerial vehicle/system
- Remote
- Internet, 3G, 4G, GSM, mobile/cellular network, wireless communication

La búsqueda realizada dio como resultado varios elementos que describiremos en la próxima sección y que hemos considerado como tecnologías comparables a la de este proyecto.

---

<sup>2</sup> Oficina Española de Patentes y Marcas : <https://www.oepm.es/es/index.html>

<sup>3</sup> ESPACENET: <https://ip.espacenet.com/>

<sup>4</sup> Thomson Innovation y Derwent World Patents Index : [http://stn-international.com/stn\\_home.html](http://stn-international.com/stn_home.html)

<sup>5</sup> CIP: <http://cip.oepm.es/>

También se realizó una búsqueda de algunas tecnologías militares similar, sin embargo, los casos encontrados de control remoto de drones se centran en comunicaciones satelitales y no vía Internet.

## 2.3. Competencia

La búsqueda en las bases de datos señaladas anteriormente arrojó resultados de interés que presentan ciertos elementos similares al del prototipo Sky 3Guardian. Los resultados se explican a continuación:

### 2.3.1. Data Logging in Aerial Platform

Nombre	Data Logging in Aerial Platform
Código de patente	US2017309088
Origen	Estados Unidos

#### Descripción:

El US2017309088 propone el uso de un RPA como interfaz de comunicación entre dispositivos.

En otras palabras, el objetivo principal es usar el RPA como un punto intermedio entre la conversación entre un dispositivo A y un dispositivo B a modo de extensión entre la comunicación vía Wi-Fi o comunicaciones móviles 3G/4G/5G.

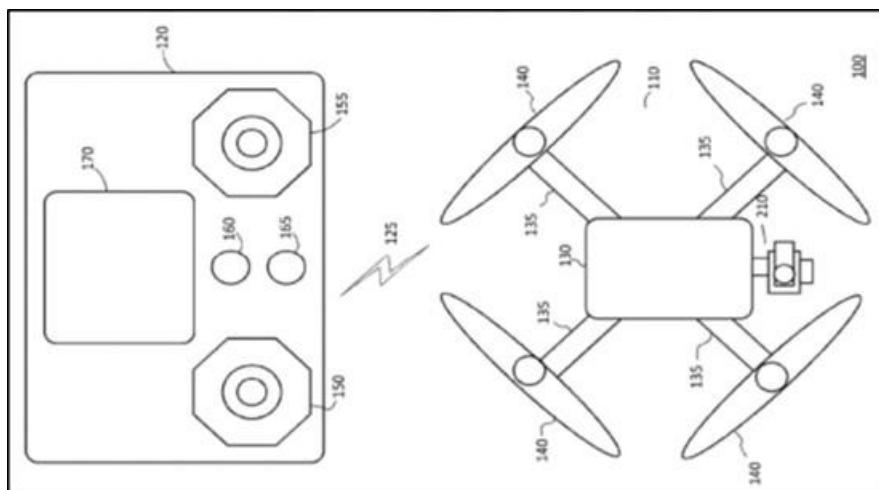


Figura 5: Esquema de prototipo US2017309088. Fuente: Base de Datos de Patentes

#### Comparación con el prototipo Sky 3Guardian:

La idea planteada por US2017309088 no se asemeja al prototipo Sky 3Guardian ya que lo que se propone es usar el RPA como proxy entre las comunicaciones de múltiples dispositivos, mientras que el objetivo del prototipo Sky 3Guardian es el control de vuelo de la nave RPA.

### 2.3.2. UAV remote control system using mobile Internet

Nombre completo	Unmanned aerial vehicle remote control system using mobile Internet
Código de patente	CN107291098A
Origen	China

#### Descripción:

El CN107291098A propone un sistema de obtención de información del vuelo del RPA vía conexión a la red por comunicaciones móviles. El sistema se comprende de 6 elementos base:

1. La aeronave RPA.
2. Módulo de comunicación móvil.
3. Antena receptora.
4. Internet.
5. Red de telefonía móvil.
6. Usuario final

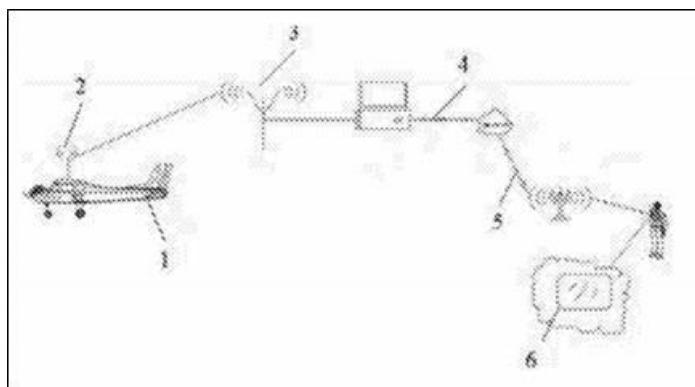


Figura 6: Esquema de prototipo CN107291098A. Fuente: Base de datos de Patentes

#### Comparación con el prototipo Sky 3Guardian:

Según la información obtenida del CN107291098A, el esquema de comunicación es considerablemente similar al del prototipo Sky 3Guardian. No obstante, no se hace mención en las especificaciones de que el usuario final tenga el control de la aeronave solo de que reciba información del vuelo: Posicionamiento, velocidad del viento, temperatura, presión, etc. Esto sugiere que el control de la aeronave se realiza de forma tradicional (comunicaciones de radio).

En este aspecto el prototipo Sky 3Guardian da un paso más allá al ser capaz de realizar las acciones propuestas por el CN107291098A mediante sensores de Arduino pero además propone un sistema de control de la aeronave.

### 2.3.3. UAV supervision cloud platform based on Beidou GPS

<b>Nombre</b>	UAV supervision cloud platform based on GPS
<b>Código de patente</b>	CN107132852A
<b>Origen</b>	China

#### Descripción:

El CN107132852A propone una de las mejores alternativas al control remoto de RPA mediante el uso de GPS.

La idea fundamental de la propuesta es usar la red satelital GPS Beidou [7] como punto de conexión a Internet y posteriormente a servidores cloud. A partir de entonces cualquier usuario con acceso a Internet puede tener el control total de la aeronave.

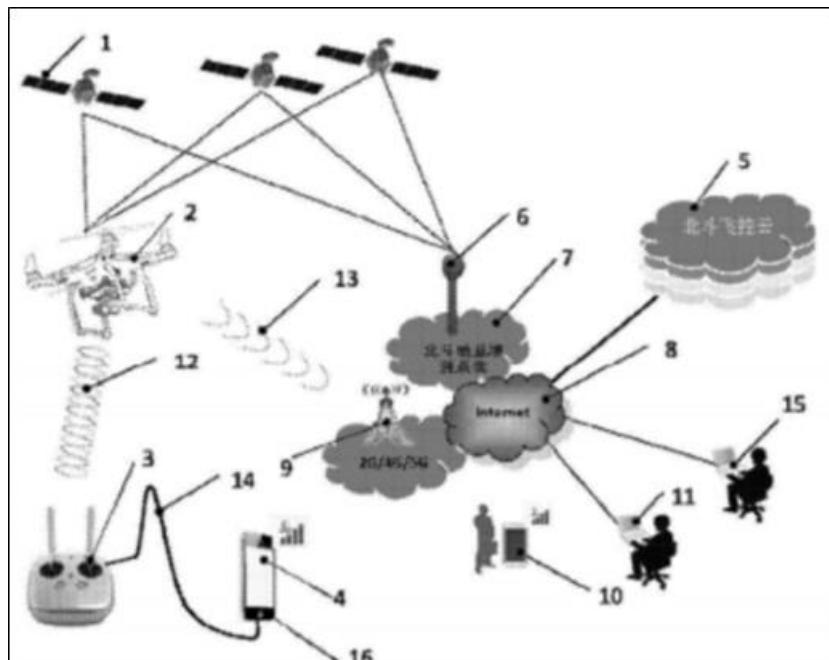


Figura 7: Esquema de prototipo CN107132852A. Fuente: Base de Datos de Patentes

#### Comparación con el prototipo Sky 3Guardian:

La propuesta del CN107132852A plantea una pregunta interesante. ¿Por qué no realizar el desarrollo en base GPS? La respuesta es simple, los sistemas de GPS son sistemas de navegación que solo funcionan como “observadores” y no pueden usarse de forma activa para realizar peticiones, es decir, son capaces de recibir cierto tipo de información de posicionamiento (si se han cumplido las bases legales para ello), pero no pueden/deben enviar ordenes de ningún tipo.

Beidou es el equivalente en China a la red satelital GPS/Galileo y es posible que opere con otras leyes exclusivas para ese país.

En cualquier caso, respecto al prototipo Sky 3Guardian, no se realiza la comunicación con ningún satélite sino que toda la información se realiza por redes móviles de antenas.

### 2.3.4. Nodecopter: AR.Drone WebFlight

Nombre	Nodecopter-remote: AR.Drone WebFlight
Código de patente	-
Origen	-

#### Descripción:

Nodecopter ofrece una avanzada interfaz de usuario para poder controlar RPA vía Internet. Se puede añadir funcionalidades de video-streaming y de integración de mandos de control con Arduino.



Figura 8: Logo de Nodecopter. Fuente: <http://www.nodecopter.com/>

#### Comparación con el prototipo Sky 3Guardian:

La propuesta de Nodecopter recopila el esfuerzo de muchos usuarios, usando diferentes tipos de lenguaje de programación para acceder a todo tipo de control del RPA vía Internet.

Fundamentalmente la propuesta es bastante similar a la del prototipo Sky 3Guardian, no obstante, la diferencia entre ambos proyectos es la forma de comunicar el RPA a través de Internet.

Nodecopter usa tecnología Wi-Fi para la conectividad, mientras que el prototipo Sky 3Guardian usa las redes móviles. Esto significa que el delay experimentado en Nodecopter será considerablemente menor al del prototipo Sky 3Guardian, pero al mismo tiempo el prototipo Sky 3Guardian tiene un alcance de control ampliamente superior al Nodecopter al estar basado en redes móviles.

### 2.3.5. Flytbase

Nombre	Flytbase: IoD (Internet of Drones)
Código de patente	-
Origen	Estados Unidos

#### Descripción:

El producto que ofrece Flytbase es sumamente amplio. Como dice en su nombre, Internet of Drones, Flytbase se basa en una plataforma de control para RPA.

Ofrece soluciones desde envío de paquetes, inspecciones remotas, construcción, etc. La comunicación se basa entre el RPA y un computador previamente preparado por el equipo de Flytbase para el control vía Wi-Fi y Bluetooth. También se menciona la opción de usar un adaptador 4G LTE *dongle* para poder recibir datos de telemetría y de vuelo en cualquier dispositivo móvil.



Figura 9: Visual de drone Flytbase. Fuente; <https://flytbase.com/>

#### Comparación con el prototipo Sky 3Guardian:

Flytbase cumple una gran cantidad de demandas relacionadas con drones. Sin embargo, particularmente hablando sobre el control remoto de los RPA, el sistema que utiliza es similar al Nodecopter. Conectividad mediante Wi-Fi con el dispositivo, en comparación con el prototipo Sky 3Guardian que pone, como base de la conectividad, redes móviles.

### 2.3.6. Matriz de Prototipos

La siguiente matriz recopila los competidores previamente mencionados (junto con algunos no mencionados) y los compara en términos de efectividad de control de drones:

Tabla 2: Prototipo Sky 3Guardian VS La Competencia. Fuente: Elaboración Propia

Prototipos:	Control	Patentado en:	Máximo Rango Potencial
US2017309088	Radio	Estados Unidos	6.000 metros
CN107291098A	Radio	China	6.000 metros
CN107132852A	Beidou (GPS)	China	Ilimitado
Nodecopter	Wi-Fi + Bluetooth	Sin patente	30 metros
Flytbase	Wi-Fi + Bluetooth	Estados Unidos	30 metros
CA2984021	5G + Radio	Canada	500 metros
KR2017103451	Wi-Fi + Bluetooth	Korea	30 metros
Gapter	3G/4G (Wireless) + Wi-Fi	Hong Kong	30 metros
<b>Sky 3Guardian</b>	<b>3G/4G (Operadora)</b>	<b>España</b>	<b>Ilimitado</b>

Como puede observarse, el prototipo Sky 3Guardian posee funcionalidad y prestaciones que no presentan ninguno de los principales prototipos encontrados en el mercado español ni europeo, lo que lo sitúa por encima del estado del arte.

### 2.3.7. Estudios públicos

Además de los prototipos en producción y/o patentados, se encontraron algunas investigaciones teóricas públicas sobre el control remoto de drones vía Internet usando al base de datos de Google Scholar como base de datos pública.

#### 1. Cellular Controlled Drone Experiment: Evaluation of Network Requirements

Este estudio, realizado por el estudiante Lassi Sundqvist de la Universidad Aalto (Finlandia) en 2015, evalúa distintos escenarios de comunicación entre un drone y antenas de comunicaciones móviles y determina.

Una de las principales ideas de este estudio es la siguiente: “En general multicópteros comerciales conectados a redes de comunicación móviles no existen. Es comprensible ya que hasta hace poco tecnologías como GSM (Global System for Mobile Communications) y UMST (Universal Mobile Telcommunications System), no tenían un delay lo suficientemente bajo”

#### 2. Estudio sobre la factibilidad de utilizar señales 4G LTE en combinación con RPA con el propósito de encontrar y rescatar víctimas de avalanchas.

Este estudio, realizado por varios estudiantes de la Universidad de Colorado (Estados Unidos) en 2014, pregunta por la viabilidad de usar un dispositivo 4G LTE acoplado a un RPA para realizar resonancias en las montañas nevadas de colorado y encontrar víctimas de avalanchas.

Similar al CN107291098A, este estudio propone utilizar el RPA como punto medio de control para un dispositivo de recopilación de información.

Página en blanco intencional

# 3. Desarrollo del Prototipo

## 3.1. Concepto

La tecnología desarrollada en este proyecto permite acceder a la controladora de vuelo de un drone, mediante Internet vía comunicaciones móviles. Los comandos enviados a la controladora son indicados y calibrados gracias a un mando de vuelo al otro lado del circuito:

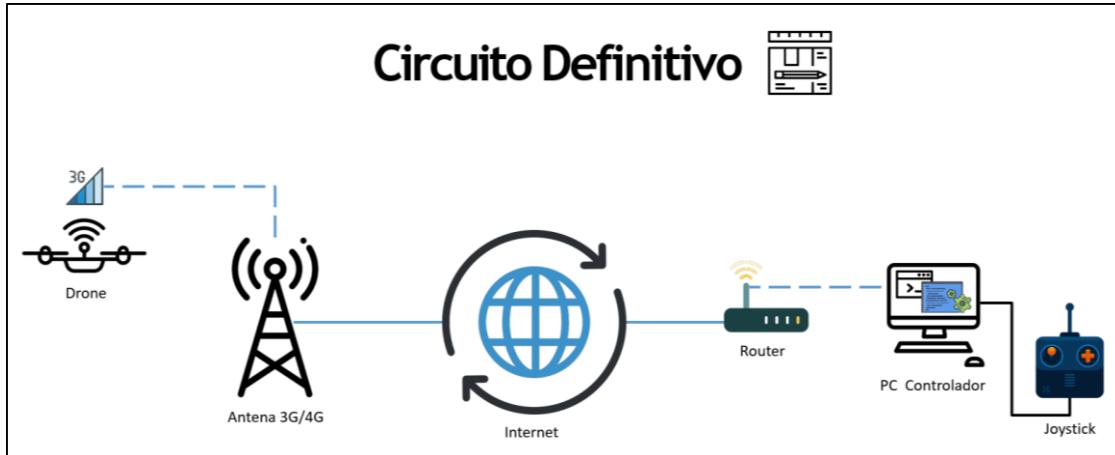


Figura 10: Circuito del proyecto. Fuente: Elaboración Propia

Este es un proyecto con un gran alcance por lo que el desarrollo se estructuró en 4 fases de integración, independientes entre sí pero integradas mediante el diagrama del circuito definitivo, además, durante el desarrollo de cada una de las fases siempre se tiene en cuenta este esquema general del circuito definitivo:

**Fase 1:** Comunicación entre el Arduino 1 y la controladora de vuelo.

**Fase 2:** Comunicación entre el Arduino 1 y el módulo GPRS.

**Fase 3:** Establecer un servidor web de libre acceso.

**Fase 4:** Establecer un sistema de recepción de datos vía el Arduino 2.

**Subfase 2 & 3:** Descargar información del servidor web mediante el módulo GPRS.

**Subfase 1 & 23:** Transmitir la información descargada a la controladora de vuelo.

**Subfase 3 & 4:** Subir información al servidor web desde el Arduino 2.

**Subfase final:** Circuito de la figura 10.

En cada fase se utilizaron materiales específicos que serán mencionados de forma general a lo largo de la explicación y de manera más detallada en el Anexo 1. Material.

## 3.2. Metodología

Para la realización de este proyecto primero se investigó la situación actual de las tecnologías relacionadas con el control remoto de aeronaves y se verificó si actualmente existe en el entorno tecnológico un sistema o prototipo que incorpore internet y redes móviles como herramienta de control a distancia de aeronaves.

El estudio realizado en el apartado de **Estado del Arte** del documento indicó que a pesar de existir algunas iniciativas de tecnologías comparables, no se considera aún como una idea consolidada y sigue abierta a muchas variaciones e interpretaciones, surgiendo de allí la oportunidad para desarrollar este proyecto.

El siguiente paso fue el de estructurar el proyecto de desarrollo del prototipo Sky 3Guardian y esto se consiguió mediante cuatro fases de integración. Cada fase contempló una parte aislada del proyecto que fue posteriormente integrada para construir el prototipo. Las fases fueron las siguientes:

FASE 1: En esta fase se diseñó el sistema de comunicación entre el microprocesador Arduino y la controladora de vuelo. Los pasos que se llevaron a cabo en esta fase fueron:

Paso 1: Diseño del circuito teórico.

Paso 2: Montaje del circuito definitivo.

Paso 3: Desarrollo del código de control.

FASE 2: En esta fase se estableció el protocolo de comunicación entre el microprocesador Arduino y el módulo GPRS para las comunicaciones móviles. Los pasos que se llevaron a cabo en esta fase fueron:

Paso 1: Comunicación el módulo GPRS y el microprocesador Arduino.

Paso 2: Especificación de los comandos AT para la inicialización del módulo GPRS.

Paso 3: Recepción de información de control vía comandos AT.

FASE 3: En esta fase se determinó un servidor web como mediador de comunicaciones entre front-end y el back-end del proyecto:

Paso 1: Especificación del servidor web ideal para el proyecto.

Paso 2: Identificación de los comandos AT de carácter HTTP transaccionales.

Paso 3: Realización de peticiones de subida y descarga de información al servidor web.

FASE 4: En esta fase se desarrolló el sistema de control remoto del RPA mediante un controlador electrónico diseñado en Arduino:

Paso 1: Desarrollo de una estructura de recepción de datos vía Joysticks.

Paso 2: Establecimiento de una comunicación vía Internet con otros dispositivos.

Paso 3: Envío de los datos recibidos vía Internet al servidor web.

En el siguiente diagrama se puede observar la metodología utilizada a lo largo del proyecto en cada fase:

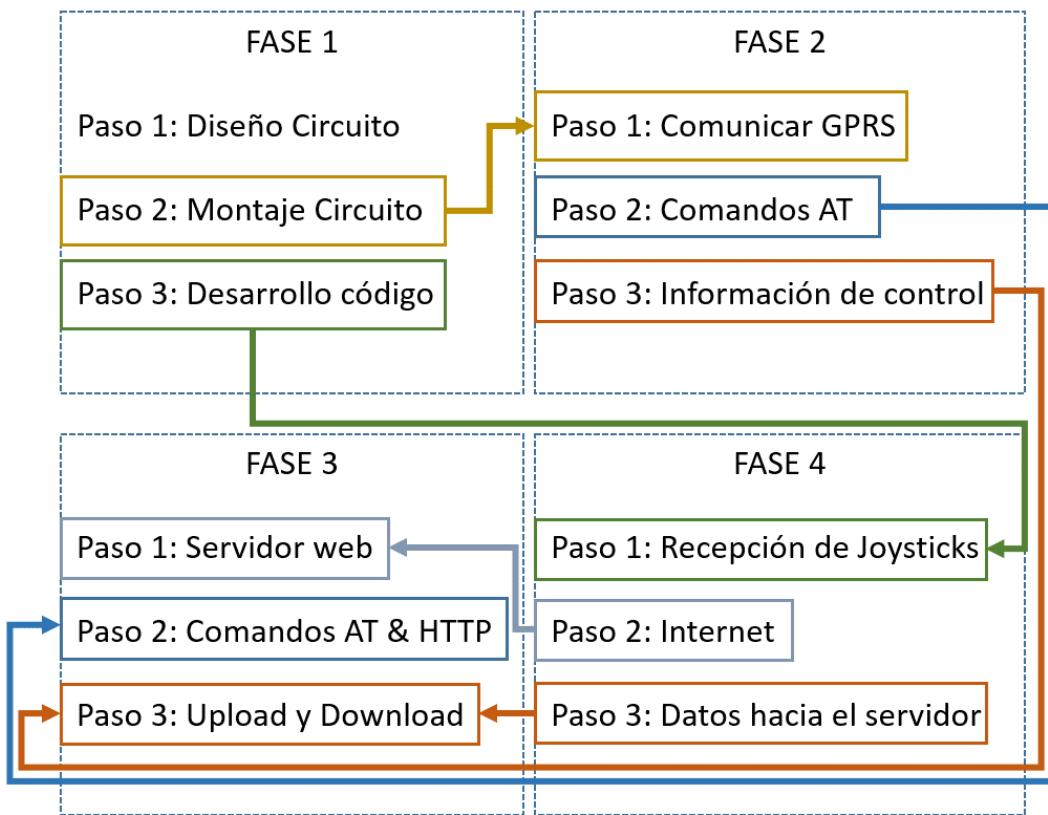


Figura 11: Esquema de metodología. Fuente: Elaboración Propia

Como se puede observar, aunque cada fase se ha desarrollado de forma aislada, siempre se ha tenido en cuenta un esquema general que pueda comunicar las fases posteriormente como se muestra a continuación:

INTEGRACIÓN DE LAS 4 FASES
<b>INTEGRACIÓN FASE 2 &amp; 3</b>
Paso 1: Identificar los comandos AT & HTTP para la descarga de información del servidor web.
Paso 2: Recibir correctamente la información en el front-end.
Paso 3: Almacenar dicha información en una variable.
<b>INTEGRACIÓN FASE 1 &amp; 2</b>
Paso 1: Interpretar la información almacenada en la variable.
Paso 2: Enviar la información a la controladora de vuelo.
<b>INTEGRACIÓN FASE 3 &amp; 4</b>
Paso 1: Almacenar los datos de lectura de los Joysticks en una variable.
Paso 2: Enviar los datos al servidor web mediante la comunicación HTTP.
<b>INTEGRACIÓN FINAL</b>
Paso 1: Enviar señales de los Joysticks a la red desde el back-end.
Paso 2: Descargar estas señales en el front-end y almacenarlas en una variable.
Paso 3: Reenviar la variable de ejecución de órdenes a la controladora de vuelo.
Paso 4: Visualizar en el simulador los resultados.

### 3.3. Descripción de las fases de desarrollo

#### 3.3.1. Fase 1

La primera fase de desarrollo del prototipo se basa en la comunicación entre el dispositivo Arduino y la controladora de vuelo SP\_F3:

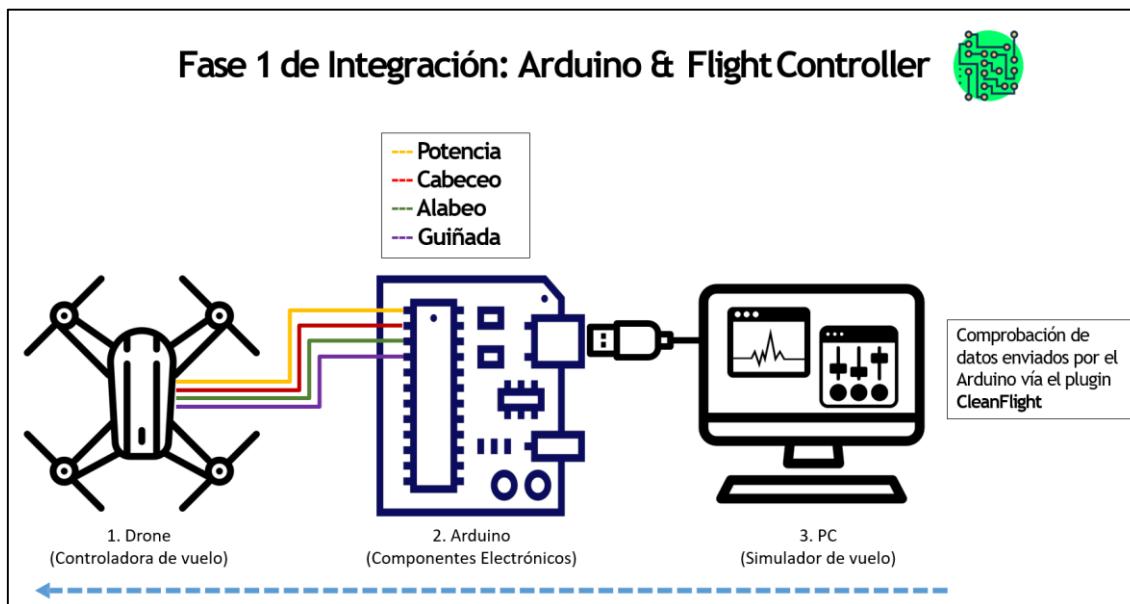


Figura 12: Fase 1 de Integración. Fuente: Elaboración Propia

Materiales:

1. Controladora de vuelo SP\_F3.
2. Microprocesador Arduino.
2. Cable de conexión USB B-M.
2. Protoboard.
2. Joystick SATKIT.
2. Cableado de pines.

Programas:

3. Arduino IDE.
3. Fritzing.
3. Cleanflight (Simulador de vuelo).

Para completar esta parte del proyecto, se realizaron previamente las siguientes tareas:

1. **Diseño del circuito teórico.** Mediante la herramienta Fritzing, se realizó el diseño preliminar con los detalles a grandes rasgos del circuito: que canales se conectan por cuales pines, que voltaje alimenta el circuito, entradas y salidas analógicas, etc.
2. **Montaje del circuito definitivo.** Incluye un joystick SATKIT que permite realizar pruebas de control enviadas desde el Arduino a la controladora de vuelo en lugar del mando vía señales de radio convencional. Después de varias iteraciones y mejoras, el circuito físico final resultó siendo el siguiente:

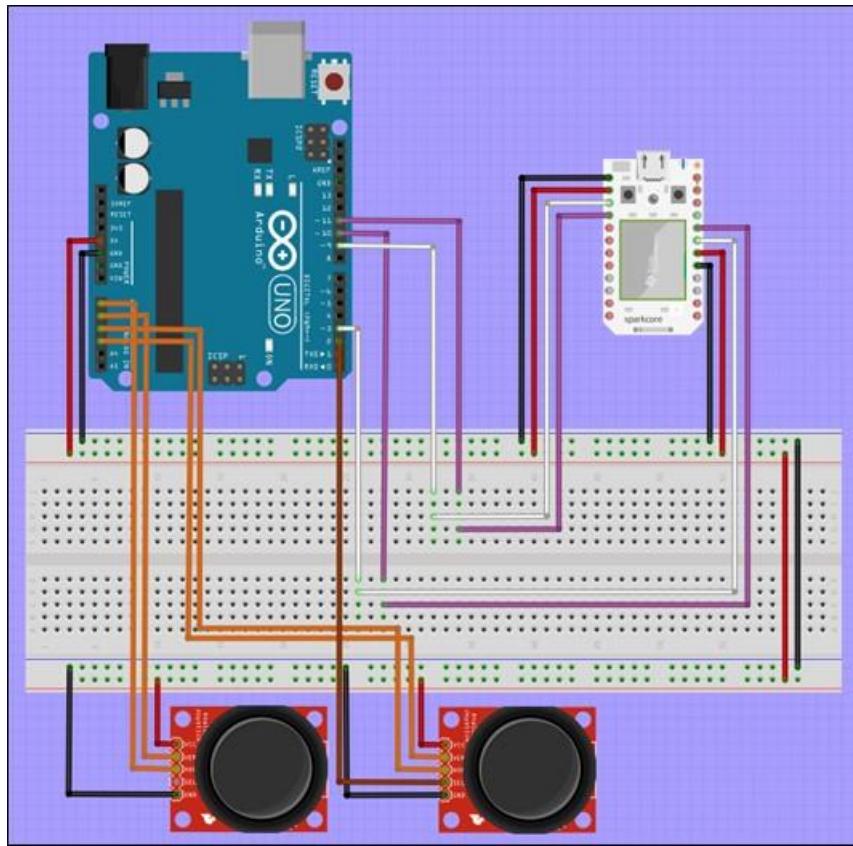


Figura 13: Fritzing de Fase 1. Fuente: Elaboración Propia.

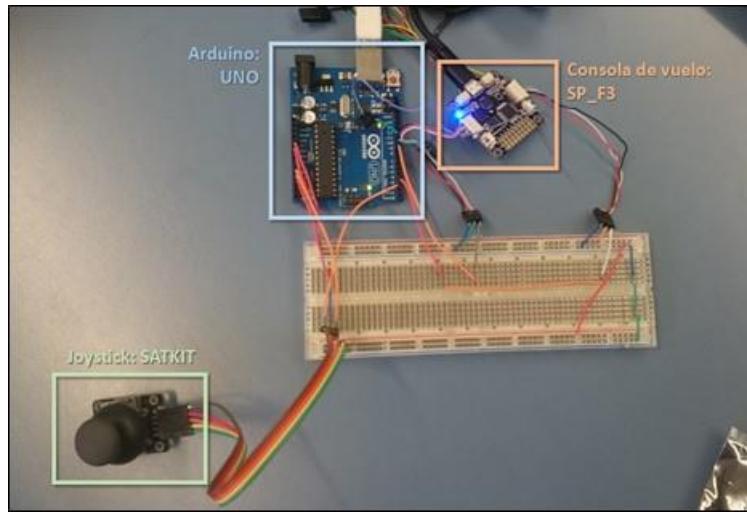


Figura 14: Circuito Físico de Fase 1. Fuente: Elaboración Propia.

**3. Desarrollo del código de control.** El código, que define las órdenes que enviará el Arduino hacia la controladora de vuelo, consta de más de 20 funciones diseñadas para calibrar y controlar el drone durante todo el vuelo; desde el despegue hasta el aterrizaje, considerando incluso momentos de inactividad. El código se escribió en lenguaje Arduino IDE que se basa en C y C++.

Entre estas funciones, las que constituyen el esqueleto de la lógica computacional son:

- **StableFunction()**: Función diseñada para estabilizar el drone en vuelo. Las 3 señales direccionales (Alabeo, Cabeceo y Guiñada) se neutralizan mientras que la potencia se ajusta al valor medio que mantiene el dispositivo a la misma altura. Se ha considerado

añadir próximamente un sensor de distancia en dirección al suelo para una mayor precisión.

- ***Increasing/DecreasingFunction()***: Estas funciones tienen como objetivo que una vez se ejecute la estabilización del dispositivo, en términos de potencia, se realice dicha estabilización gradualmente para evitar que el drone experimente una *entrada en pérdidas*.
- ***Throttle/Yaw/Pitch/RollFunction()***: Funciones encargadas de realizar lecturas constantes de dichas señales y realizar acciones de forma acorde.

Además de las funciones básicas de control, con el fin de ajustar el código a las demandas del proyecto, se diseñó un método clave donde una única variable contiene toda la información de los 4 parámetros básicos de control junto con un parámetro de control adicional, el delay. Este método se definirá a partir de ahora como método RIFE (Referente Identificador de Fuente Emisora).

RIFE tiene como objetivo unificar todas las órdenes enviadas por el circuito bajo un mismo comando que, posteriormente, será descompuesto en el Arduino en las 4 señales básicas de control junto al delay. La necesidad de RIFE se basa en los primeros pasos de la Fase de 2 donde cierta información de prueba será enviada a un módulo GPRS vía un único pin receptor, por lo tanto, los 5 valores han de ser recibidos al mismo tiempo por medio de un mismo pin, a diferencia de los 4 canales independientes en la controladora de vuelo.

El siguiente esquema explica el concepto de **RIFE v1.0**:

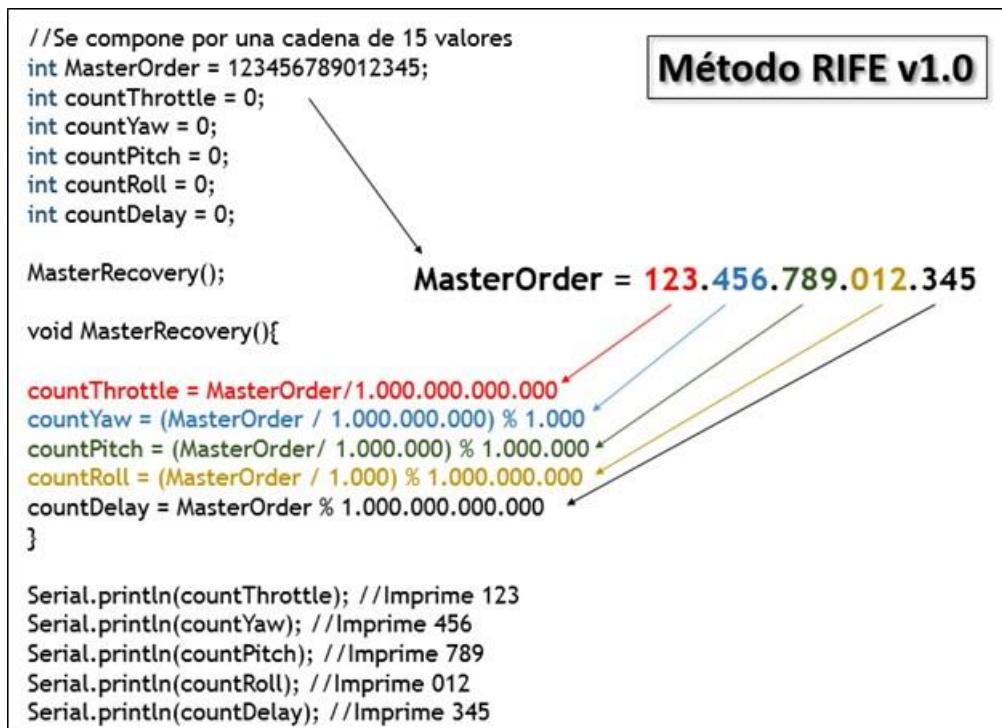


Figura 15: Método RIFE. Fuente: Elaboración Propia

Como se aprecia en el esquema, la variable *MasterOrder* contiene toda la información sobre el siguiente instante del vuelo. Los primeros 3 dígitos corresponden a la potencia, los siguientes 3 a la guinada, los siguientes 3 al cabeceo, los siguientes 3 al alabeo y los últimos 3 al delay.

Aunque es teóricamente plausible, en la práctica definir una variable de tipo int con 15 dígitos no es posible debido a las limitaciones de memoria asignada al mismo tipo de variable (de -32.767 a 32.767) por lo que el método RIFE debió ser modificado.

## RIFE v2.0: Solución por Vectores.

Una de las soluciones propuestas, fue la de convertir la variable int Rife en una tabla unidimensional con 5 posiciones, int *Rife[4]*, donde cada posición representa uno de los parámetros base de control:

*Rife[0] = Throttle; Rife[1] = Yaw; Rife[2] = Pitch; Rife[3] = Roll; Rife[4] = Delay;*

No obstante, esta solución presentó problemas en las siguientes fases así que se desarrolló una tercera versión de RIFE.

## RIFE v3.0: Solución por unsigned long long int.

La limitación esencial de RIFE v1.0 es la limitación de memoria que internamente tiene asignada una variable de tipo INT, que es de 2 bytes (de 0 a 65.535).

La extensión *long long int* admite un rango de variables de 0 a 18.446.744.073.709.551.615, que es un valor lo suficientemente grande como para admitir los **15 dígitos** valores necesarios.

Al acabar el proyecto, se decidió reducir RIFE a **12 dígitos** obviando los valores correspondientes al delay ya que durante las transacciones ya hay un delay de seguridad suficiente.

## Videos de Resultados:

A continuación, como también en la sección de Anexo 2. Vídeos, se incluyen 3 vídeos que visualizan la progresión durante el desarrollo de la Fase 1:

1. [Primer contacto](#): Es el haber conseguido enviar señales aleatorias desde el Arduino hasta la Controladora de vuelo.
2. [Estabilización y Aceleración](#): Lograr que el drone se estabilice y poder aumentar la potencia (velocidad) de los motores.
3. [Fase 1 de Integración Completada](#): Control del drone vía Joysticks Arduino.

### 3.3.2. Fase 2

La segunda fase de integración se basa en la comunicación entre el dispositivo Arduino y el módulo GPRS para las comunicaciones móviles:

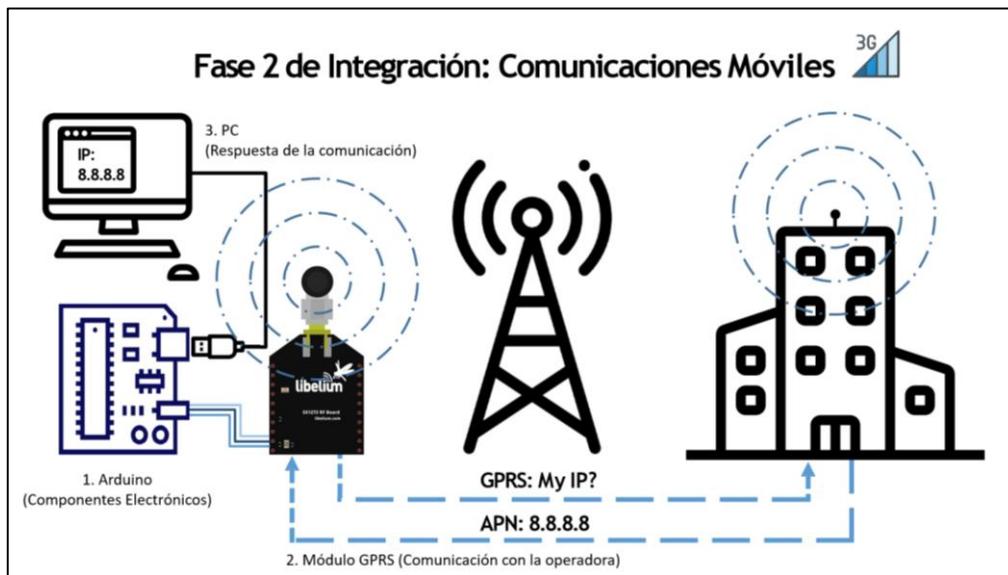


Figura 16: Fase 2 de Integración. Fuente: Elaboración Propia

Materiales:

1. Microprocesador Arduino.
1. Cable de conexión USB B-M.
1. Cableado de pines.
1. Cargador de Arduino Aukru.
2. Módulo SIM900 GPRS.
2. Tarjeta SIM.
2. Antena de comunicación móvil para módulo GPRS.

Programas:

3. Arduino IDE.
3. Fritzing.
3. Notepad++.

El objetivo principal de esta fase fue el de lograr una comunicación constante entre el módulo GPRS y el Arduino.

Para esto el ejercicio que se propuso hacer una petición del valor de IP asignado a la tarjeta SIM integrada en el módulo GPRS y mostrar dicha IP por pantalla.

El código de Arduino utilizado para realizar la comunicación Arduino-GPRS fue obtenido de la comunidad de código libre GitHub y tiene definidas una serie de funciones diseñadas para enviar comandos AT al GPRS para realizar la configuración base y posteriormente hacer peticiones más avanzadas como obtener el número IP o realizar un GET/POST a cierta URL.

Cabe mencionar que al principio del proyecto se contaba con un módulo capaz de realizar comunicaciones 3G/4G usando Libelium WaspMote como microcontroladora. No obstante por limitaciones técnicas (explicadas más detalladamente en la sección de Dificultades) el proyecto se vio forzado a usar un módulo GPRS (2G) para las comunicaciones.

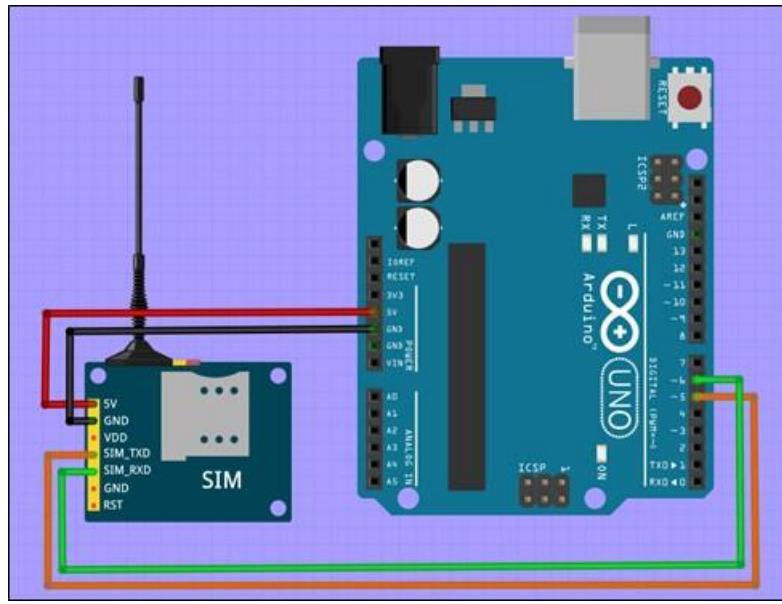


Figura 17: Fritzing de Fase 2. Fuente: Elaboración Propia

Para la realización de esta fase, fue necesaria la incorporación de las librerías SoftwareSerial que permiten establecer dos pines del microprocesador Arduino como comunicadores virtuales de envío de datos. Esto fue fundamental ya que para hacer debugging del programa fue necesario usar el Serial interno del Arduino para realizar impresiones por pantalla y controlar el funcionamiento del programa y la comunicación.

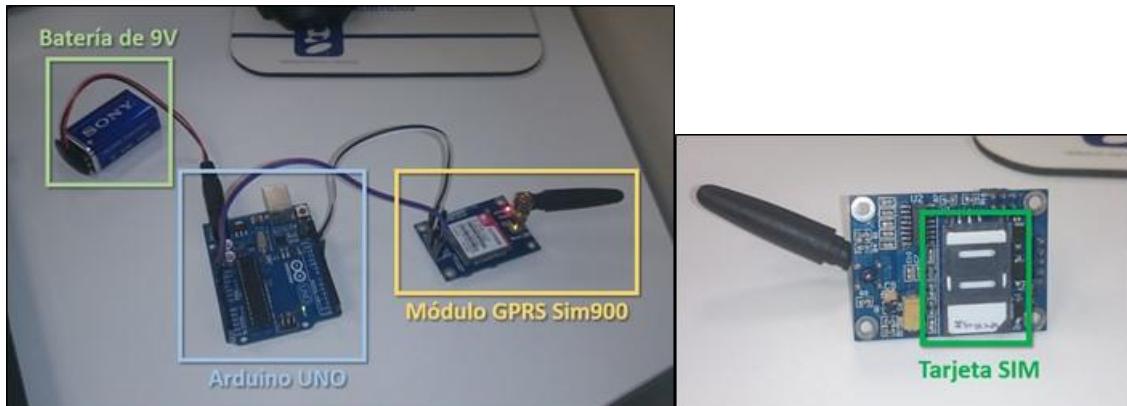


Figura 18: Circuito físico de Fase 2. Fuente: Elaboración Propia

Con el fin de comprender la rutina de inicialización, la comunicación inicial y obtener la IP asignada a la tarjeta SIM, se procede a continuación a la explicación de la estructura del código desarrollado:

0. Previamente se realizan las declaraciones y definiciones de variables/constants.
1. Se establece la comunicación Serial con el GPRS a 9600 baudios. Es fundamental que tanto el Arduino, como el módulo GPRS estén comunicándose en los mismos baudios
2. Liberación de memoria de los Serials involucrados en el proceso (Serial y SoftwareSerial) mediante la función flush().

A partir de ahora se realiza una rutina de inicialización que asegura una comunicación fluida y constante con el módulo GPRS:

<b>Comando AT</b>	<i>AT+CGATT?</i>
<b>Función</b>	Comprueba la modalidad de conexión del módulo GPRS SIM900.
<b>Respuesta esperada</b>	OK +CGATT: 1
<b>Respuesta de error</b>	+CGATT: (código de error de conexión)

<b>Comando AT</b>	<i>AT+CSQ</i>
<b>Función</b>	Comprueba la potencia de la señal móvil respecto a la antena de la operadora.
<b>Respuesta esperada</b>	OK +CSQ: (potencia de la señal, típicamente un valor entre 20 y 30)
<b>Respuesta de error</b>	+CME ERROR: (código de error de ejecución)

<b>Comando AT</b>	<i>AT+SAPBR=3,1 ; Contype ; GPRS</i>
<b>Función</b>	Establece el modo de uso GPRS de la tarjeta SIM. Si las tiene, habilita las funciones GPRS.
<b>Respuesta esperada</b>	OK
<b>Respuesta de error</b>	+CME ERROR: (código de error de ejecución)

A continuación, se han de presentar las credenciales de configuración de acceso de Internet móvil. Las credenciales durante el desarrollo del proyecto fueron los siguientes: [8]

- El nombre del APN: *ac.vodafone.es* (“em” durante las primeras pruebas)
- Usuario: *vodafone* ([void] durante las primeras pruebas)
- Contraseña: *vodafone* ([void] durante las primeras pruebas)

Existen dos formas de presentar dichas credenciales. Mediante el uso de tres veces el mismo comando *AT+SAPBR* explicado anteriormente:

<b>Comando AT</b>	<i>AT+SAPBR=3,1 ; APN/USER/PWD ; [credencial APN/USER/PWD]</i>
<b>Función</b>	Acredita las credenciales de la compañía para poder hacer peticiones vía Internet.
<b>Respuesta esperada</b>	OK
<b>Respuesta de error</b>	+CME ERROR: (código de error de ejecución) +SAPBR: (código de error de autenticación)

O bien mediante el siguiente comando que engloba las 3 credenciales en una sola ejecución:

<b>Comando AT</b>	<i>AT+CSTT= [credencial APN], [credencial USER], [credencial PWD]</i>
<b>Función</b>	Acredita las credenciales de la compañía para poder hacer peticiones vía Internet.
<b>Respuesta esperada</b>	OK
<b>Respuesta de error</b>	+CME ERROR: (código de error de ejecución)

Finalmente, dado que el objetivo de la fase 2 de integración es obtener la IP asignada a la tarjeta SIM en cuestión, ejecutamos el comando de petición de valor IP:

<b>Comando AT</b>	<i>AT+SAPBR=2,1</i>
<b>Función</b>	Comprueba si la conexión GPRS se ha establecido correctamente y devuelve el valor de IP de la tarjeta SIM.
<b>Respuesta esperada</b>	OK +SAPBR: [status de conexión], [valor IP]
<b>Respuesta de error</b>	+CME ERROR: (código de error de ejecución) +SAPBR: (código de error de autenticación)

Se puede apreciar visualmente la comunicación establecida mediante las respuestas del mismo Arduino como impresiones Serial:

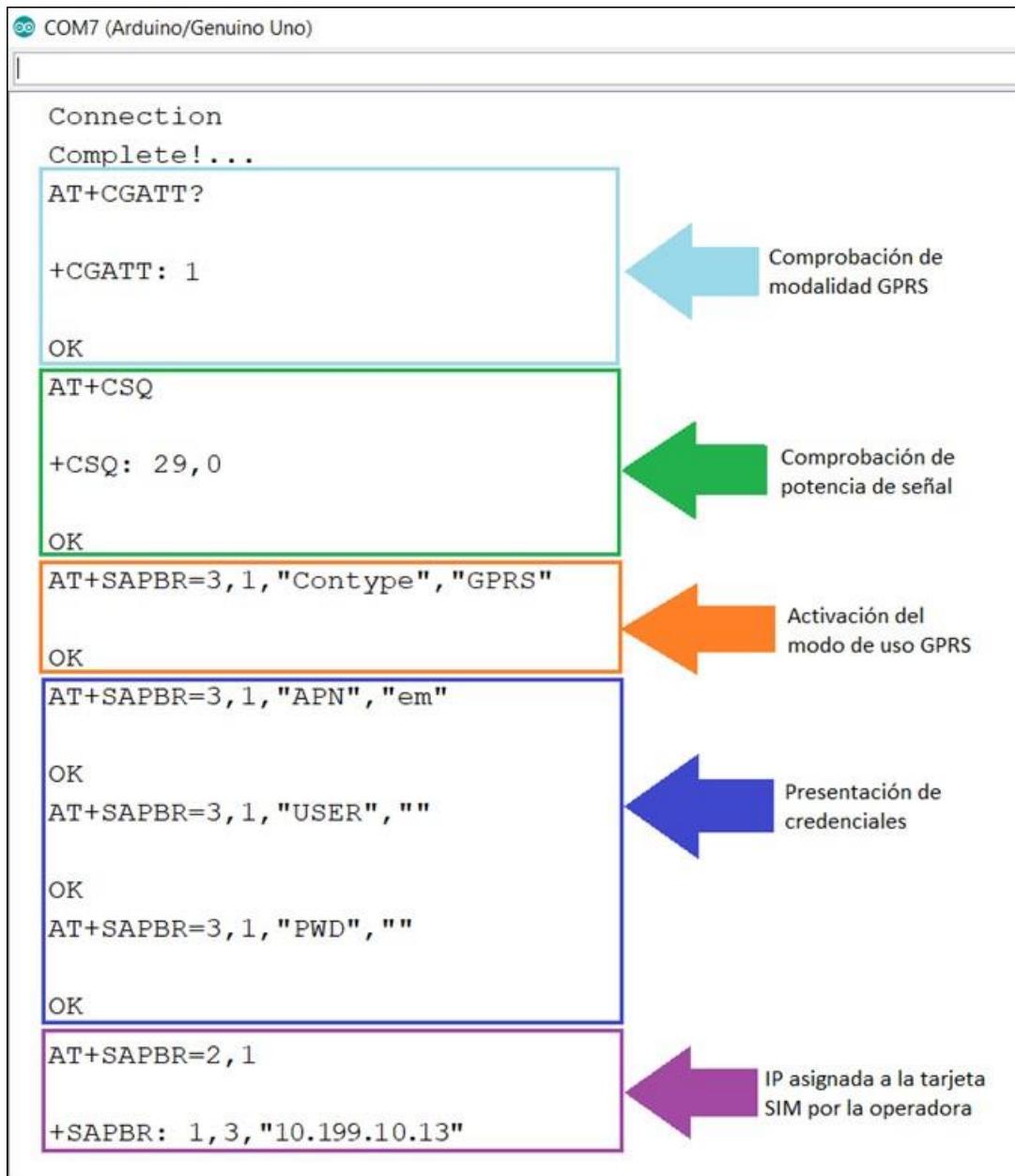


Figura 19: Comunicación Serial con Modulo GPRS. Fuente: Elaboración Propia

Habiendo obtenido este resultado, ahora el siguiente paso es hacer peticiones HTTP vía Internet hacia el servidor web preparado durante la fase 3 de desarrollo.

### 3.3.3. Fase 3

La tercera fase se basa en la creación o establecimiento de un servidor web al cual se le puedan hacer peticiones HTTP para almacenar y descargar información temporal:

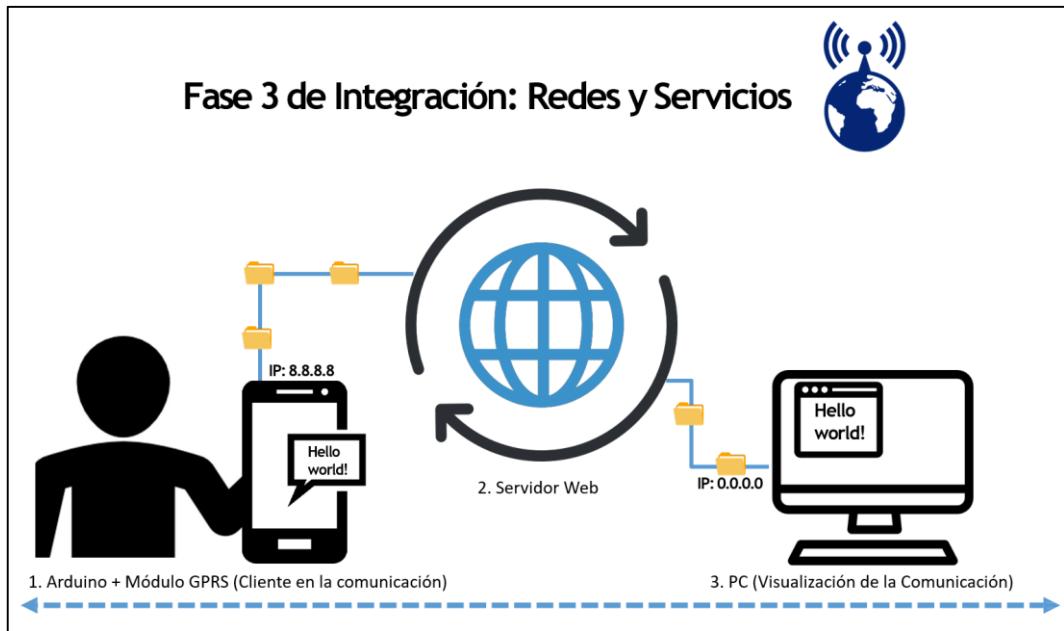


Figura 20: Fase 3 de Integración. Fuente: Elaboración Propia

Materiales:

1. Microprocesador Arduino.
1. Cable de conexión USB B-M.
1. Cableado de pines.
1. Módulo SIM900 GPRS.
1. Tarjeta SIM.
1. Antena de comunicación móvil para módulo GPRS.
1. Cargador de Arduino Aukru.

Programas y servidores:

2. Servidor web: Dweet.io
3. Arduino IDE.
3. Fritzing.

El objetivo principal de esta fase es el de establecer un servicio web que tenga la modalidad de almacenar y descargar datos de las señales de potencia, guiñada, alabeo y cabeceo.

Para alcanzar esta meta, es necesario que se cumplan los siguientes requisitos:

1. Identificar un servidor web de libre acceso que admita transacciones HTTP como GET y POST con el menor delay posible.
2. Diseñar, dentro de los códigos de front-end y back-end del circuito, las transacciones necesarias tanto para publicar datos como descargarlos.
3. Almacenar en una variable interna la información descargada del servidor web que, posteriormente, será descompuesta usando el método RIFE para la distribución de las señales de potencia, alabeo, cabeceo y guiñada.

### Fase 3 - Primer Requisito: Servidor Web

Durante el avance del proyecto se plantearon 3 servidores web distintos, de los cuales solo 2 son viables y solo uno cumple los requerimientos de delay del proyecto:

#### **Google Drive**

La primera opción que se planteó fue usar Google Drive a modo de servidor Web. El planteamiento era el de desarrollar una rutina en el microprocesador Arduino con el objetivo de crear un archivo.txt, subirlo vía internet a una cuenta privada de Google Drive y posteriormente descargarlo.

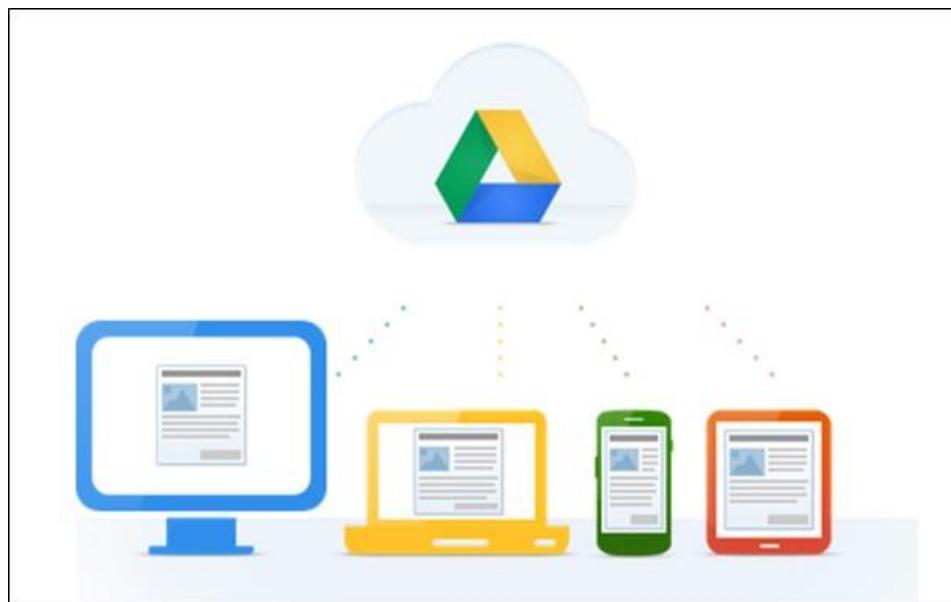


Figura 21: Conectividad Google Drive. Fuente: [https://www.google.com/intl/es\\_ALL/drive/](https://www.google.com/intl/es_ALL/drive/)

Al hacer pruebas se determinó que no solo los delays de subida y descarga estaban completamente fuera de los requerimientos del proyecto, sino que era necesaria una rutina de autorización de credenciales para cada iteración del ciclo de envío de señales. Es decir, para cada señal enviada desde el back-end al front-end era necesario presentar:

1. Credenciales módulo GPRS.
2. Credenciales de cuenta Google Drive (Gmail + contraseña).
3. Presentar las llaves de lectura y descarga para acceder al archivo.
4. Confirmación del tipo de transacción propuesta.

Todas estas transacciones de seguridad requeridas por Google Drive forzaron a plantear otra solución para el proyecto.

## ThingSpeak

La segunda opción consistió en usar el servidor web de ThingSpeak que, como se explicó anteriormente, es una solución para productos y servicios de Internet of Things. Similar a WordPress, permite a los usuarios crear blogs fácilmente y además permite a los desarrolladores interactuar con dispositivos que envían distintos tipos de señales vía Internet.



Figura 22: Lobby ThingSpeak. Fuente: <https://thingspeak.com/>

La gran ventaja que ThingSpeak ofrece es que es una API ya preparada para almacenar, descargar y gestionar datos desde distintos dispositivos vía Internet. Es tal la facilidad y accesibilidad de ThingSpeak que ya hay librerías en Arduino preparadas para hacer peticiones directamente al servidor web privado seleccionado:

```
ThingSpeak.setField(1,j);
ThingSpeak.writeFields(myChannelNumber, myWriteAPIKey);
Serial.print("valor de j: ");
Serial.println(j);
```

Figura 23: Extracto código ThingSpeak. Fuente: Elaboración Propia

Donde el valor “1” en ThingSpeak.setField representa el número del canal donde se va a subir la información y el valor “j” es el valor como tal que se desea subir.

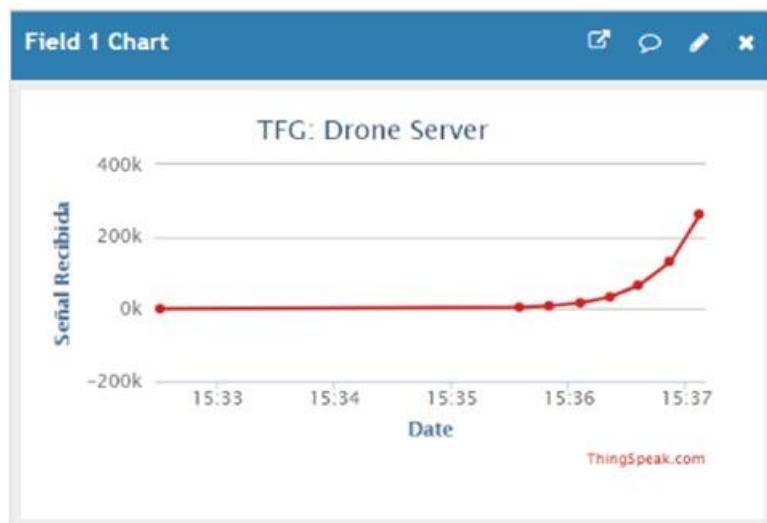


Figura 24: Lectura de variable exponencial en ThingSpeak. Fuente: Elaboración Propia

Los valores dentro de ThingSpeak.writeChannel son las credenciales confidenciales asignadas a la cuenta de ThingSpeak en cuestión. Es necesario presentarlas para tener la autorización de subir (WriteKey) o descargar (ReadKey) información.

Sin embargo, debido a las limitaciones de la licencia asignada a una cuenta gratuita, los delays experimentados eran demasiado elevados para seguir adelante (más información en el apartado de dificultades de la fase 3).

## Dweet.io

La última opción que se exploró durante la búsqueda del servidor web ideal fue la de la API de Dweet. La mejor forma de describir Dweet es como la hace su web:

“Fast, Free and ridiculously simple, it’s like twitter for social machines”.

Como lo han descrito, Dweet es una herramienta increíblemente fácil de usar. Hay que seguir estos simples pasos para la creación y modificación de tu propio servidor web:

- a. Definir el nombre y los valores que se desean incorporar en ese servidor web. Si, por ejemplo, se desea crear un servidor web llamado **Sergio-Test** y dentro de ese servidor web añadir una **fecha de una evaluación (el 26 de junio)**.

- b. Simplemente hay que escribir en cualquier navegador la siguiente URL:

<https://dweet.io/dweet/for/Sergio-Test?FechaExamen=26052018>

- c. Para visualizar el contenido de este servidor web basta con escribir la siguiente URL:

<https://dweet.io/get/latest/dweet/for/Sergio-Test>

La respuesta recibida para este ejemplo es la siguiente:



Figura 25: Servidor Web Sergio-Test. Fuente: <https://dweet.io/>

Es una cadena de caracteres en formato HTTP donde figuran tanto el nombre del servidor web como los valores almacenados posteriormente. Este formato, aunque simple, funciona perfectamente para el objetivo que planteamos, solo es necesario establecer los valores dentro del servidor web, mediante una petición POST de carácter HTTP, de las señales para el RPA.

Durante el proyecto, el nombre del servidor web usado es el de **ArduinoTest-RIFE-Project** y el valor asignado es un int de 12 dígitos (usando las extensiones de variable long long).



Figura 26: Servidor Web RIFE. Fuente: <https://dweet.io/>

Dado que el delay encontrado con la herramienta Dweet es lo suficientemente reducido (menos de 1 segundo), se decidió entonces trabajar con este servidor web durante el desarrollo del proyecto.

### Fase 3 - Segundo Requisito: Diseñar código

Una vez elegido y probado el servidor web, fue necesario hacer dos adaptaciones del código: una para el front-end y otra para el back-end del proyecto (se hablará en esta sección de la integración para el front-end mientras que se expandirá sobre la integración del back-end en el apartado de Fase 4).

Toda la información recopilada sobre la integración de peticiones Dweet a código Arduino IDE era mediante dispositivos conexiones Ethernet en lugar de conexiones móviles, así que fue necesario hacer un diseño de código lo bastante robusto como para mantener la conexión entre el servidor web ArduinoTest-RIFE-Project y el módulo GPRS, como al mismo tiempo eliminando todas las peticiones no fundamentales para reducir el tiempo de espera entre petición y petición.

Así pues la rutina desarrollada para el front-end del proyecto siguió la siguiente estructura:

1. Primero se realiza el SETUP, definido anteriormente en la fase 2, con la presentación de credenciales y comprobación de conectividad y funcionalidad GPRS:

<b>Comando AT</b>	<i>AT+CGATT?</i>
<b>Función</b>	Comprueba la modalidad de conexión del módulo GPRS SIM900.
<b>Respuesta esperada</b>	OK +CGATT: 1
<b>Respuesta de error</b>	+CGATT: (código de error de conexión)

<b>Comando AT</b>	<i>AT+CSQ</i>
<b>Función</b>	Comprueba la potencia de la señal móvil respecto a la antena de la operadora.
<b>Respuesta esperada</b>	OK +CSQ: (potencia de la señal, típicamente un valor entre 20 y 30)
<b>Respuesta de error</b>	+CME ERROR: (código de error de ejecución)

<b>Comando AT</b>	<i>AT+SAPBR=3,1 ; Contype ; GPRS</i>
<b>Función</b>	Establece el modo de uso GPRS de la tarjeta SIM. SI las tiene, habilita las funciones GPRS.
<b>Respuesta esperada</b>	OK
<b>Respuesta de error</b>	+CME ERROR: (código de error de ejecución)

<b>Comando AT</b>	<i>AT+SAPBR=3,1 ; APN/USER/PWD ; [credencial APN/USER/PWD]</i>
<b>Función</b>	Acredita las credenciales de la compañía para poder hacer peticiones vía Internet.
<b>Respuesta esperada</b>	OK
<b>Respuesta de error</b>	+CME ERROR: (código de error de ejecución) +SAPBR: (código de error de autenticación)

<b>Comando AT</b>	<i>AT+CSTT= [credencial APN], [credencial USER], [credencial PWD]</i>
<b>Función</b>	Acredita las credenciales de la compañía para poder hacer peticiones vía Internet.
<b>Respuesta esperada</b>	OK
<b>Respuesta de error</b>	+CME ERROR: (código de error de ejecución)

<b>Comando AT</b>	<i>AT+SAPBR=2,1</i>
<b>Función</b>	Comprueba si la conexión GPRS se ha establecido correctamente y devuelve el valor de IP de la tarjeta SIM.
<b>Respuesta esperada</b>	OK +SAPBR: [status de conexión], [valor IP]
<b>Respuesta de error</b>	+CME ERROR: (código de error de ejecución) +SAPBR: (código de error de autenticación)

Esta serie de comandos solo se ejecutará una vez, en el momento en que el microprocesador Arduino reciba energía vía USB, conector Aukru, batería, llamada de Serial desde un ordenador o físicamente presionando el botón de RESET.

2. Después de hacer el SETUP inicial, definimos el **loop** que repetirá el Arduino constantemente durante las transacciones de petición de información:

<b>Comando AT</b>	<i>AT+HTTPINIT</i>
<b>Función</b>	Inicializa una comunicación de tipo HTTP entre el módulo GPRS y la operadora.
<b>Respuesta esperada</b>	OK
<b>Respuesta de error</b>	+CME ERROR: (código de error de ejecución)

Definimos la petición HTTP del servidor web de donde queremos descargar información. Como se mencionó previamente, la URL a la que se le debe hacer la petición de recuperar los valores del servidor web en una cadena de caracteres es:

<https://dweet.io/get/latest/dweet/for/ArduinoTest-RIFE-Project>

<b>Comando AT</b>	<i>AT+HTTPPARA="URL", [URL del servidor web]</i>
<b>Función</b>	Establece los parámetros para realizar una inminente comunicación HTTP.
<b>Respuesta esperada</b>	OK
<b>Respuesta de error</b>	+CME ERROR: (código de error de ejecución)

<b>Comando AT</b>	<i>AT+HTTPACTION=[Valor de petición]</i>
<b>Función</b>	Establece la inminente comunicación HTTP como 0 (GET) de información de la URL previamente declarada.
<b>Respuesta esperada</b>	OK +HTTPREAD: 200
<b>Respuesta de error</b>	+CME ERROR: (código de error de ejecución) Errores comunes: 601 (error de red); 603 (URL incorrecta)

<b>Comando AT</b>	<i>AT+HTTPREAD=[Inicio],[Final]</i>
<b>Función</b>	De la petición HTTP realizada, solo se devolverá al Arduino los valores entre [Inicio] y [Final].
<b>Respuesta esperada</b>	OK [Respuesta del Servidor Web: "A0":188188188188]
<b>Respuesta de error</b>	+CME ERROR: (código de error de ejecución)

Finalmente es necesario cerrar la comunicación HTTP para evitar un Timeout y evitar la congestión del canal de comunicación con Dweet.

Comando AT	<i>AT+HTTPTERM</i>
<b>Función</b>	Finaliza la comunicación HTTP con la URL previamente declarada.
<b>Respuesta esperada</b>	OK
<b>Respuesta de error</b>	+CME ERROR: (código de error de ejecución)

Entre cada comando AT ejecutado se realiza una lectura Serial del módulo GPRS con el objetivo de mostrar por pantalla las respuestas HTTP a las peticiones a modo de debugging.

La comunicación definida dentro del loop se repite para cada ciclo de petición de datos. Es decir, cada vez que se envíe a la controladora de vuelo los valores de potencia, alabeo, cabeceo y guiñada, es necesario abrir la comunicación HTTP y volverla a cerrar.

```

← → ⌂ Es seguro | https://dweet.io/get/latest/d...
{
  "this": "succeeded",
  "by": "getting",
  "the": "dweets",
  "with": [
    {
      "thing": "arduinotest-rife-project",
      "created": "2018-06-05T14:09:22.395Z",
      "content": {"A0": 188188188188}
    }
  ]
}

COM7 (Arduino/Genuino Uno)

OK
AT+HTTPPARA="URL", "http://dweet.io/get/latest/dweet/for/arduino
AT+HTTPACTION=0

OK

+HTTPACTION: 0, 200, 163

AT+HTTPREAD=141, 152

+HTTPREAD: 22
{"A0": 188188188188} ] }

OK

Autoscroll Sin ajuste de linea 9600 baudio

```

Figura 27: Lectura Serial del Servidor RIFE. Fuente: Elaboración Propia

### Fase 3- Tercer Requisito: Almacenamiento en variable

Se ha conseguido establecer un servidor web, el cual experimenta delays mínimos, y se ha diseñado una rutina de comunicación HTTP que devuelve por Serial los valores deseados.

Este último requisito plantea el hecho de que, en lugar de imprimir por pantalla los valores recibidos del servidor web vía el Serial del microprocesador Arduino, se guarden dichos valores en una variable que se pueda controlar para hacer la descomposición de RIFE inverso y posteriormente enviar las señales a sus respectivas áreas en la controladora de vuelo.

Para completar este tercer requisito se desarrollaron dos funciones nuevas:

1. **PrintSerialData3()**: Es una versión modificada de la función PrintSerialData cuya única funcionalidad es la de imprimir por pantalla la respuesta del módulo GPRS a cada comando AT.

PrintSerialData3 además de mostrar por pantalla esta respuesta utiliza dos funciones integradas de la clase Serial:

- a. **Serial.find(carácter)**: Como su nombre indica, esta función busca dentro de todos los caracteres impresos por Serial el carácter en cuestión que se pase como argumento. Para poder aprovechar esta funcionalidad, durante el establecimiento de valores en lugar de marcar el comienzo de la lectura de señales mediante los caracteres “A0” a partir de ahora se usará el carácter de barra vertical “|“.

```
"content":{ "|":188188188188}}]}
```

Figura 28: Extracto de Servidor RIFE. Fuente: Elaboración Propia

- b. **Serial.readBytesUntil(carácter, array, tamaño de array)**: Esta función almacena en una array todos los caracteres recibidos de la petición HTTP a partir del carácter encontrado en Serial.find (barra vertical en el proyecto) y hasta otro carácter definido dentro de esta función. (Típicamente el carácter de llave cerrada “ } ”) Así pues, dentro de una array definida como **data** se almacenan los siguientes caracteres:

“	:	1	8	8	1	8	8	1	8	8	1	8	8	}	}
Ruido	Potencia	Guíñada	Cabeceo	Alabeo	Ruido										

2. **OnlyNumbers()**: Como se puede observar, dentro de la array data existen algunos caracteres no relevantes para la comunicación de las señales también denominados como “Ruido”. La función OnlyNumbers() nos permite guardar en una nueva array, data\_numbers, únicamente los valores numéricos que se encuentren dentro de data obviando los caracteres no relevantes que se consideran ruido durante la comunicación:

1	8	8	1	8	8	1	8	8	1	8	8
Potencia	Guíñada	Cabeceo	Alabeo								

Se logró entonces descargar desde un servidor web la información necesaria para poder dar órdenes a la controladora de vuelo sobre que niveles de voltaje se ha de suministrar en cada una de las señales básicas.

### 3.3.4. Fase 4

La cuarta y última fase de integración, el back-end, se basa en la creación de un subsistema del proyecto que permita la lectura, vía joysticks, de las señales que serán enviadas a la controladora de vuelo al otro lado del circuito en el front-end:

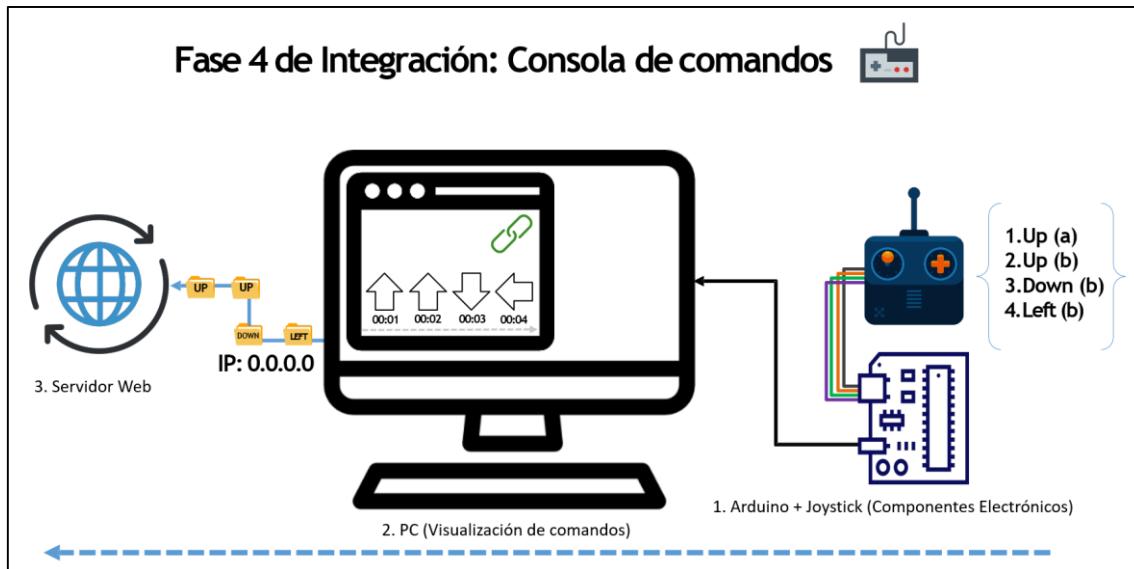


Figura 29: Fase 4 de Integración. Fuente: Elaboración Propia

Materiales:

1. Microprocesador Arduino.
1. Cable de conexión USB B-M.
1. Cableado de pines.
1. Cargador de Arduino Aukru.
1. Joystick SATKIT.
1. Placa Ethernet.
1. Cable de conexión Ethernet.

Programas y servidores:

2. Arduino IDE.
2. Fritzing.
2. Cleanflight.
3. Servidor web: Dweet.io

El objetivo principal de esta fase fue conseguir una codificación capaz de enviar vía Ethernet información recibida mediante un sistema de carga de datos para el servidor web Dweet.io. Este sistema se basa en la correcta lectura e interpretación de los INPUTS de los joysticks analógicos vía el microprocesador Arduino.

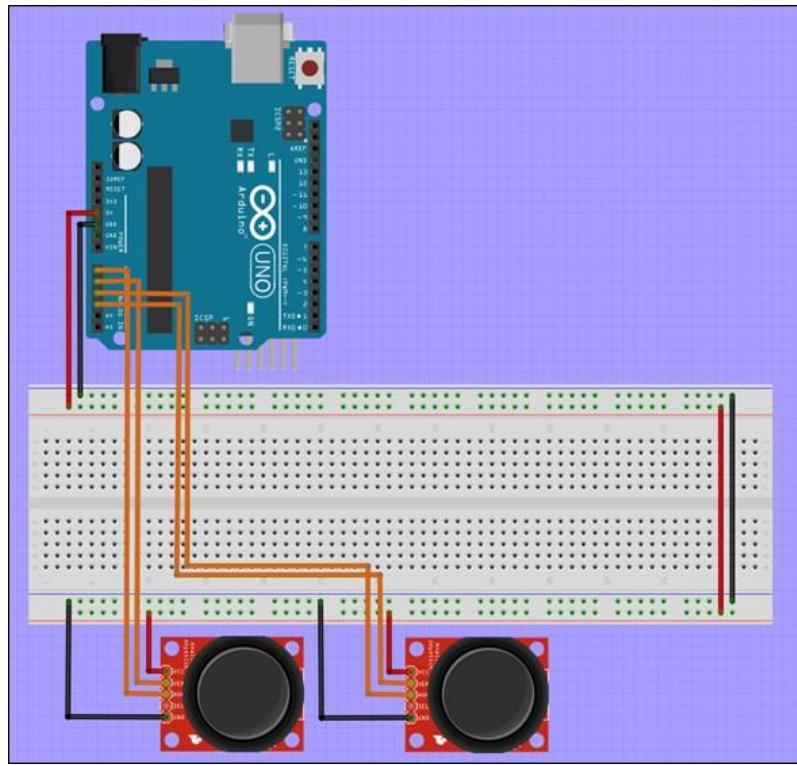


Figura 30: Fritzing de Fase 4. Fuente: Elaboración Propia

Para realizar la solución propuesta, fue necesaria la incorporación de una placa Ethernet (Ethernet Shield) compatible con el microprocesador Arduino al proyecto.

“El Arduino Ethernet Shield añade la capacidad de conectar un microprocesador Arduino a una red Ethernet. Es la parte física que implementa la pila de protocolos TCP/IP. Está basada en el chip Ethernet Wiznet W5100. El Wiznet W5100 provee de una pila de red IP capaz de soportar TCP y UDP. Soporta hasta cuatro conexiones de sockets simultáneas. Usa la librería Ethernet para leer y escribir los flujos de datos que pasan por el puerto Ethernet.” [9]

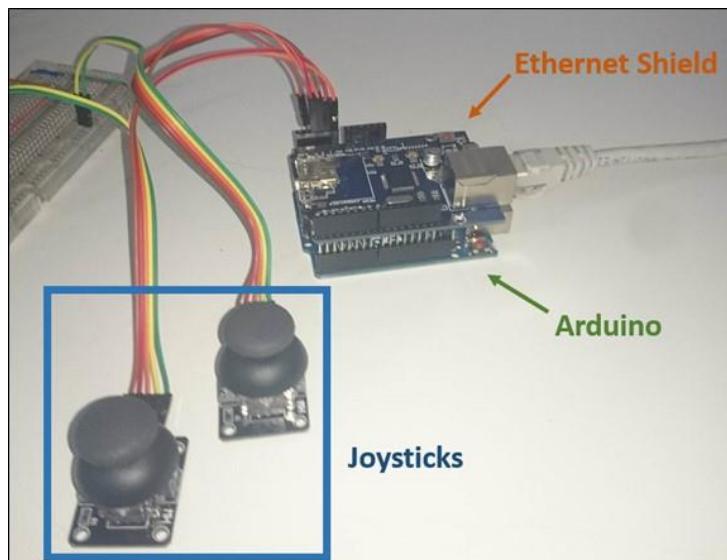


Figura 31: Circuito Físico de Fase 4. Fuente: Elaboración Propia

Es necesario mencionar que tanto el módulo GPRS como el Ethernet Shield realizan peticiones web que requieren de un voltaje superior al que puede proporcionar una fuente de alimentación vía USB.

Por lo tanto durante toda la Fase 4 fue necesario el uso de la misma fuente externa de energía como la de la Fase 2. O bien el cargador Aukru o bien el adaptador de vía alcalina de 9V.

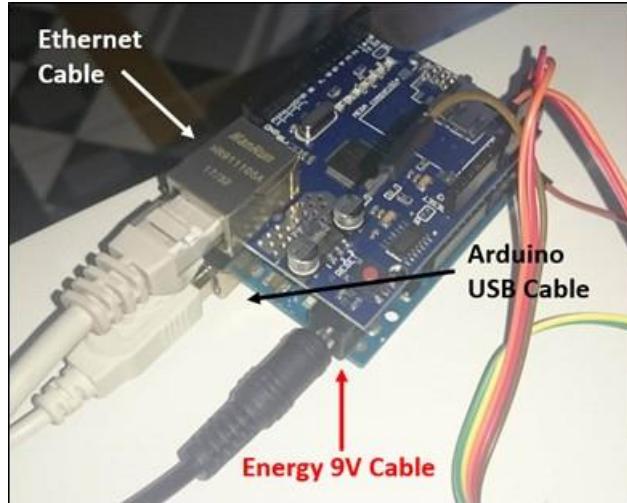


Figura 32: Placa Ethernet en Arduino. Fuente: Elaboración Propia

Una vez conseguida la placa Ethernet y habiendo podido proporcionarle la fuente de alimentación requerida, se desarrolló el código en cuestión.

Dado que el código desarrollado en la Fase 1 ya tenía establecida la estructura de recepción de datos e incluso tenía incorporada la funcionalidad de control vía Joystick, se desarrolló una variante que tuviese en cuenta las peticiones necesarias para una conexión de clase Ethernet al mismo tiempo que remplazara las funciones de envío de señales a la controladora de vuelo por funciones de envío de señales al servidor web Dweet.io.

El primer paso fue adaptar el código:

### 1. Declaración de variables Ethernet:

Para poder operar con las funciones Ethernet, fue necesario descargar la librería Ethernet desde el catálogo de librerías Arduino. Una vez hecho esto, se inicializaron tres variables clave:

```
byte mac[] = { 0xDE, 0xAD, 0xBE, 0xEF, 0xFE, 0xFE};  
byte ip[] = { 192, 168, 137, 24 };  
EthernetClient client;
```

Figura 33: Extracto de código de Fase 4. Fuente: Elaboración Propia

La primera variable `byte` `mac[ ]`, representa la dirección mac asignada al Ethernet Shield durante la conexión. “Una dirección MAC es el identificador único asignado por el fabricante a una pieza de hardware de red (como una tarjeta inalámbrica o una tarjeta Ethernet). «MAC» significa Media Access Control, y cada código tiene la intención de ser único para un dispositivo en particular.” [10]

La segunda variable `byte` `ip[ ]`, representa el valor IP asignado al puerto de conexión del Ethernet Shield en caso de que no se le asigne de forma externa. Para este proyecto lo que se realiza es una conexión *bridge* para retransmitir la conectividad recibida por Wi-Fi hacia el Ethernet Shield (se explicará que es una conexión *bridge* durante el apartado de dificultades en la Fase 4 de integración).

Por último, la tercera y última variable es el cliente Ethernet como tal que hará las peticiones web hacia el servidor web posteriormente. “La expresión cliente servidor se utiliza en el ámbito de la informática. En dicho contexto, se llama cliente al dispositivo que requiere ciertos servicios a un servidor. La idea de servidor, por su parte, alude al equipo que brinda servicios a las computadoras (ordenadores) que se hallan conectadas con él mediante una red.” [11]

## 2. Peticiones cliente-servidor / Ethernet Shield-Dweet.io:

Dentro del código desarrollado existe una función fundamental que se encarga de organizar las lecturas de los joysticks en una sola variable (RIFE) y después de realizar esta organización la transmite mediante una petición POST al servidor Dweet declarado durante la Fase 3.

Dicha función es **Rife\_MasterFunction()**, se estructura en 3 partes:

A. Declaración de función y llamada:

```
void Rife_MasterFunction(int Throttle, int Yaw, int Pitch, int Roll){
```

Toda llamada a Rife\_MasterFunction se ha de hacer enviando como argumentos el estado actual de cada una de las 4 señales: Potencia (Throttle), Yaw (Guiñada), Cabeceo (Pitch) y Roll (Alabeo).

La llamada a esta función se realiza después de haber hecho la lectura de joysticks y la interpretación de que significan dichas lecturas en términos de señales de vuelo.

B. Cálculos y método RIFE:

Después de recibir las señales como argumentos, se desarrolla el método RIFE dentro de una variable numérica como también en una cadena de caracteres:

```
rife = Throttle*10^12;
rife = (Yaw*10^9)%10^3;
rife = (Pitch*10^6)%10^6;
rife = (Roll*10^3)%10^9;
s_rife = String(Throttle)+String(Yaw)+String(Pitch)+String(Roll);
```

Figura 34: Método RIFE en Código. Fuente: Elaboración Propia

Dentro de la variable RIFE sencillamente se ubican los valores de Potencia en los dígitos 12, 11 y 10, los valores de Guiñada en los dígitos 9, 8 y 7, los valores de Cabeceo en los dígitos 6, 5 y 4 y finalmente los valores de Alabeo en los dígitos 3, 2 y 1.

Para el envío de datos vía Internet usamos otra nomenclatura en forma de cadena de caracteres **s\_rife**. Aquí se concadenan los valores cambiándolos a Strings para su envío.

### C. Envío de datos:

Por último, se realiza el envío de datos de señales vía Internet. Primero, se detiene toda conexión que el cliente pueda tener previamente activa, y se ejecuta la función **connect** que usa como argumentos el nombre del servidor ([www.dweet.io](http://www.dweet.io)) y el puerto de conexión (típicamente 80 para comunicaciones HTTP).

Si la conexión se realiza exitosamente, se concadenan tanto la petición POST como los valores guardados previamente en la variable *s\_rife* mediante la función **concat** y entonces se envía al cliente la cadena de caracteres que incluye la petición POST junto con la cadena de señales.

Para finalizar se cierra la conexión y se vuelve a repetir el proceso para la siguiente iteración de la comunicación.

```
client.stop();

int status_con = client.connect(server, 80);

// if there's a successful connection:
if (status_con == 1) {
    Serial.println("connected");
    // Make a HTTP request:
    String s = "POST /dweet/for/arduinotest-rife-project?|=";

    s.concat(s_rife);
    Serial.println(s);
    client.println(s);

    client.println("Host: www.dweet.io");
    client.println("Connection: close");
    client.println();
```

Figura 35: Envío de datos vía Ethernet. Fuente: Elaboración Propia

Habiendo conseguido este último objetivo, se ha completado todo el circuito necesario para el desarrollo de las fases finales del proyecto.

### Video de Resultado:

En el siguiente [video](#) se muestra el control con doble Joystick conseguido durante la Fase 4.

## 3.4. Integración de fases

El último paso para completar el proyecto es realizar la integración de las 4 fases. Esta integración entre fases se denominará Subfases de Integración o bien Subintegración de Fases.

### 3.4.1. Subintegración Fase 2 y Fase 3

La primera subintegración es de la Fase 2 con la Fase 3, es decir, realizar peticiones HTTP al servidor web Dweet.io mediante comandos AT. Gran parte de esta subintegración se realizó durante las pruebas con el servidor web para acortar el tiempo de pruebas.

El objetivo de esta subintegración es conseguir el menor delay posible en la comunicación.

Para esto se realizaron múltiples pruebas con distintos valores de delay en cada sección del intercambio transaccional de paquetes. Dichos resultados numéricos pueden apreciarse en la sección de **Resultados** del documento.

Los delays conseguidos son los mínimos disponibles para el hardware utilizado en el proyecto; el reducir aún más cualquiera de los valores del delay provoca un error de colisión de paquetes en la comunicación.

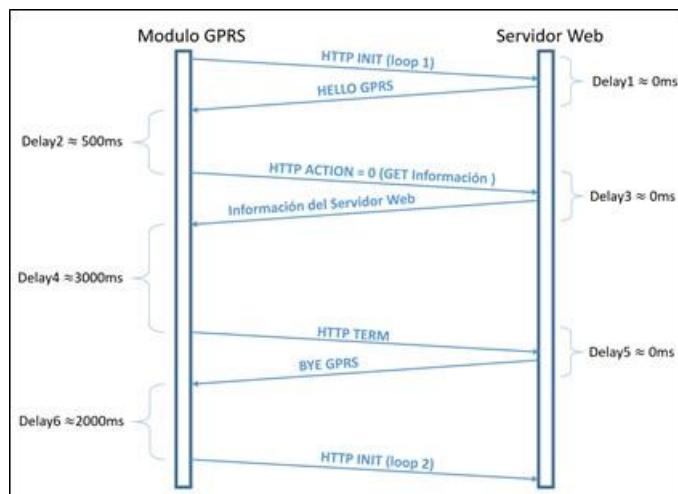


Figura 36: Delays mínimos. Comunicación Exitosa

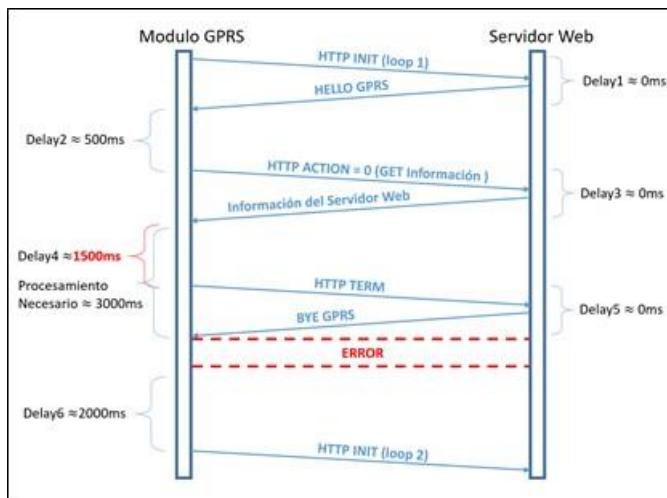


Figura 37: Delays mínimos. Comunicación por debajo del delay mínimo: Colisión

### 3.4.2. Subintegración Fase 1 y Subfase 23

La siguiente subintegración corresponde a la ampliación de la subintegración anterior para incluir funcionalidades de envío de información a la controladora de vuelo y constituye el **front-end** del proyecto.

El objetivo de esta subintegración es, usando RIFE inverso, conseguir descifrar la información descargada desde el servidor web para señalizar a la controladora de vuelo la siguiente acción del ciclo.

Para esto se realiza una conversión inversa de los valores de data\_number (variable donde está almacenada la información descargada) en 4 nuevas variables definidas como signalThrottle, signalYaw, signalPitch y signalRoll:

```
signalThrottle = (data_numbers[0]-48)*100+(data_numbers[1]-48)*10+(data_numbers[2]-48)+327;  
signalYaw = (data_numbers[3]-48)*100+(data_numbers[4]-48)*10+(data_numbers[5]-48)+312;  
signalPitch = (data_numbers[6]-48)*100+(data_numbers[7]-48)*10+(data_numbers[8]-48)+327;  
signalRoll = (data_numbers[9]-48)*100+(data_numbers[10]-48)*10+(data_numbers[11]-48)+312;
```

Figura 38: Método RIFE Inverso en Código. Fuente: Elaboración Propia

En esta operación se asigna cada bloque del array a su variable correspondiente:

Tabla 3: Valores de Data\_numbers[]. Fuente: Elaboración Propia

Data_numbers[i]: Señal:
0 signalThrottle
1 signalThrottle
2 signalThrottle
3 signalYaw
4 signalYaw
5 signalYaw
6 signalPitch
7 signalPitch
8 signalPitch
9 signalRoll
10 signalRoll
11 signalRoll

Es necesario también restar el valor 48, ya que al convertir un String a INT hay que tener en cuenta la numeración ASCII. Por ultimo cada digito se multiplica por su magnitud y se suma el factor de conversión de Joystick (Es necesario hacer esta suma ya que el código está preparado para leer lecturas de Joysticks, así que simplemente hemos de sumar +327 para movimiento vertical y +312 para movimiento horizontal del joystick).

### 3.4.3. Subintegración Fase 3 y Fase 4

La siguiente subintegración corresponde al envío de datos recibidos por Joysticks como inputs hacia el servidor web y constituye el **back-end** del proyecto.

El objetivo de esta subintegración es visualizar, vía el servidor web, distintas combinaciones de movimientos de joystick y comprobar que efectivamente dichos movimientos se reflejan en Dweet.io.

Primero se comprueba que, una vez inicializado el Arduino y el código de la fase 4 ya está en marcha con conectividad a Internet, los valores recibidos en el servidor web corresponden a la aeronave en modo estable, a la espera de recibir las siguientes instrucciones:

```
{"this": "succeeded", "by": "getting", "the": "dweets", "with": [{"thing": "arduinotest-rife-project", "created": "2018-06-07T18:49:10.998Z", "content": {"|": 188188188188}}]}
```

Figura 39: Respuesta de Servidor RIFE: Estabilización. Fuente: Elaboración Propia

Las siguientes pruebas consistieron en aumentar gradualmente y por individual las 4 señales:

```
{"this": "succeeded", "by": "getting", "the": "dweets", "with": [{"thing": "arduinotest-rife-project", "created": "2018-06-07T19:06:49.943Z", "content": {"|": 204188188188}}]}
```

Figura 40: Respuesta de Servidor RIFE: Potencia++. Fuente: Elaboración Propia

```
{"this": "succeeded", "by": "getting", "the": "dweets", "with": [{"thing": "arduinotest-rife-project", "created": "2018-06-07T18:54:55.859Z", "content": {"|": 188160188188}}]}
```

Figura 41: : Respuesta de Servidor RIFE: Guiñada--. Fuente: Elaboración Propia

```
{"this": "succeeded", "by": "getting", "the": "dweets", "with": [{"thing": "arduinotest-rife-project", "created": "2018-06-07T18:55:50.242Z", "content": {"|": 188188145188}}]}
```

Figura 42: Respuesta de Servidor RIFE: Cabeceo--. Fuente: Elaboración Propia

```
{"this": "succeeded", "by": "getting", "the": "dweets", "with": [{"thing": "arduinotest-rife-project", "created": "2018-06-07T18:57:06.452Z", "content": {"|": 188188188201}}]}
```

Figura 43: Respuesta de Servidor RIFE: Alabeo++. Fuente: Elaboración Propia

Por último se hace una combinación de dos señales de Joysticks distintos, Potencia y Alabeo:

```
{"this": "succeeded", "by": "getting", "the": "dweets", "with": [{"thing": "arduinotest-rife-project", "created": "2018-06-07T19:03:40.726Z", "content": {"|": 156188188207}}]}
```

Figura 44: Respuesta de Servidor RIFE: Potencia-- & Alabeo++. Fuente: Elaboración Propia

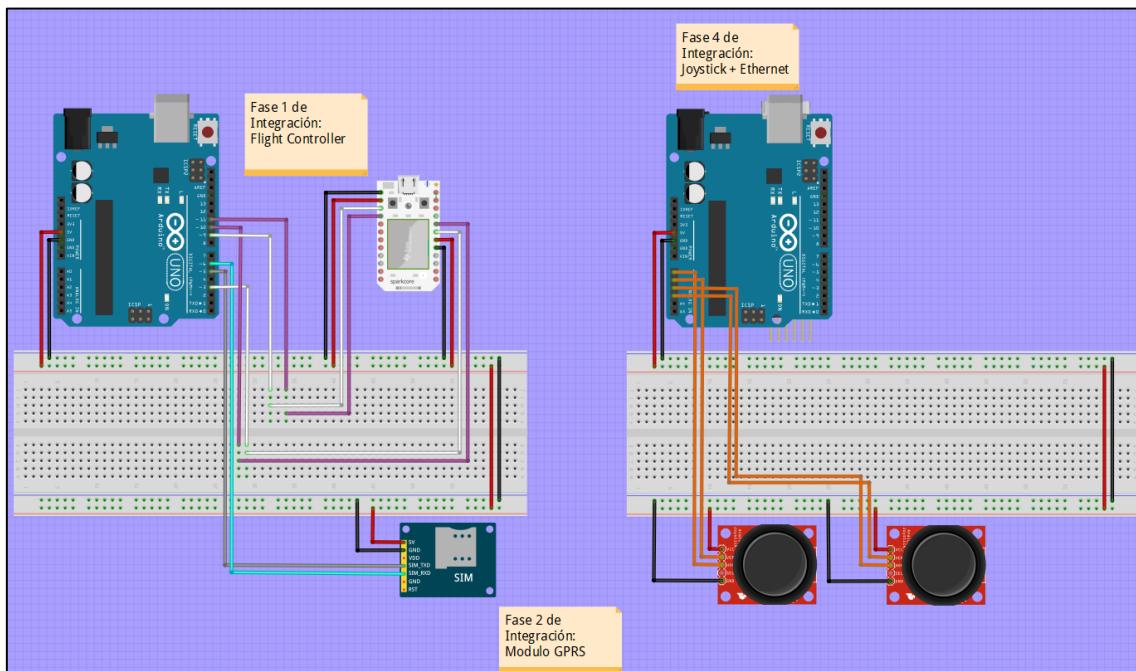
Se puede apreciar como ambas señales cambian a la vez sin afectarse entre ellas y otro punto que cabe destacar es que las señales de Cabeceo y Guiñada siguen estabilizadas.

### 3.4.4. Subintegración Final

Habiendo desarrollado tanto el front-end (fase 1, 2 y 3) como el back-end (fase 3 y 4) el proyecto se considera que ha sido completado. Se ha conseguido:

1. La lectura de datos de señal vía Joysticks SATKIT desde el back-end del proyecto.
2. La interpretación y envío de datos vía Internet al servidor web establecido en Dweet.io
3. La descarga de los datos en el front-end del proyecto mediante comandos AT y peticiones HTTP.
4. La descomposición de los datos recibidos en señales analógicas y conseguir enviarlos a la controladora de vuelo.
5. Evaluación de los resultados en el simulador de vuelo CleanFlight y realización de ajustes hasta conseguir el objetivo propuesto.

En la sección de Resultados del documento, se encuentra una descripción detallada de la funcionalidad del circuito completo, además, en la sección de Anexo 2. Vídeos se encuentra un link a un vídeo mostrando la integración en vivo.



# 4. Evaluación del Prototipo

## 4.1. Resultados

### 4.1.1. Umbral mínimo de potencia

El Umbral mínimo de Potencia (UMP) es una de las varias características particulares de la controladora de vuelo. Básicamente exige que el valor de voltaje recibido por la controladora de vuelo por cualquiera de los 4 canales básicos de control ha de ser igual o mayor a un valor en concreto. Después de varias pruebas, se pudo determinar que el valor que corresponde al UMP es de 1,93V. Es decir, si el voltaje transmitido está por debajo de ese del UMP la señal no será reconocida por la controladora de vuelo y, como consecuencia, se activará el *mecanismo de integridad de señalización múltiple* explicado próximamente en la sección de Dificultades de la Fase 1.

Las siguientes tablas tienen como objetivo ilustrar las distintas interpretaciones de los valores de Voltaje como señales analógicas de Arduino junto con las lecturas del simulador de vuelo y los respectivos resultado con el objetivo de identificar el UMP.

Tabla 4: UMP de Potencia. Fuente: Elaboración Propia

Voltaje (Voltios)	Arduino (int)	Potencia	Resultado
5	254	2200	Elevación máxima del drone
4	203	1760	Elevación ligera del drone
3,7	188	1628	Estabilización en vuelo
3	152	1320	Arranque promedio de hélices
2	115	880	Arranque mínimo de hélices
1,93	112	800	No signal
1	55	800	No signal
0	0	800	No signal

Tabla 5: UMP de Alabeo/Cabeceo/Guiñada. Fuente: Elaboración Propia

Voltaje (Voltios)	Arduino (int)	Alabeo/Cabeceo/Guiñada	Resultado
5	254	2100	Derecha / Arriba / Agujas del Reloj
4	203	1621	
3,7	188	1500	Estabilización
3	152	1216	Izquierda / Abajo / Contra Agujas del Reloj
2	115	810	
1,93	112	1500	No signal
1	55	1500	No signal
0	0	1500	No signal

## 4.1.2. Delays

Las siguientes tablas tienen como objetivo ilustrar los valores del delay experimentado durante la fase de integración final. Los delays se clasifican en dos secciones:

**Delay en Setup:** Es el tiempo necesario para inicializar el Arduino y el módulo GPRS (front-end) o el Ethernet Shield (back-end).

**Delay en Loop:** Es el tiempo entre peticiones HTTP realizadas por el módulo GPRS en el front-end (en el back-end el delay es virtualmente 0),

Como resultado final, los valores mínimos de delay que se pudieron establecer son los siguientes:

Tabla 6: Delays en Setup, Front-end. Fuente: Elaboración Propia

Evento en Setup (front-end)	Delay (ms)
Establecimiento de comunicación de serie (Serial)	2.000
Comprobación de funcionalidad GPRS	1.000
Comprobación señal de antena	1.000
Conectividad modalidad GPRS	2.000
Presentar credenciales	6.000
Iniciar GPRS	10.000
Mostrar IP actual	2.000
<b>Total</b>	<b>24.000</b>

Tabla 7: Delays en Setup, Back-end. Fuente: Elaboración Propia

Evento en Setup (back-end)	Delay (ms)
Establecimiento de Serials	2.000
Inicialización de Ethernet Shield	25.000
<b>Total</b>	<b>27.000</b>

Tabla 8: Delays en Loop, Front-end. Fuente: Elaboración Propia

Evento en Loop (front-end)	Delay (ms)
Inicializar servicio HTTP	500
Establecer URL de descarga	3.000
Establecer la acción HTTP como GET	2.000
Recibir la respuesta del servidor	500
Culminar servicio HTTP	500
Control entre comunicaciones	50
<b>Total</b>	<b>6.550</b>

En el front-end, se necesitan de aproximadamente 30 segundos para inicializar el módulo GPRS y recibir la primera señal. Una vez recibida, el delay entre señales es de 6,5 segundos.

En el back-end, se necesitan 27 segundos para inicializar el Ethernet Shield y enviar la primera señal. Una vez enviado, el delay entre señales es virtualmente 0.

### 4.1.3. Peso del front-end

A continuación se pesaron las piezas que componen el front-end. Hay que aclarar que solo se tomaron en cuenta los pesos del front-end ya que es éste el que corresponde al drone:

Tabla 9: Peso del front-end. Fuente: Elaboración Propia

Material	Peso
Arduino UNO	25 gramos
Controladora de Vuelo SP_F3	5 gramos
Protoboard	79 gramos
Batería de Arduino 9V	45 gramos
Módulo GPRS	10 gramos
Drone Hexacoptero (ya ensamblado)	1.000 gramos
Batería de Drone	360 gramos
<b>Total</b>	<b>1.524 gramos</b>

En el front-end, aproximadamente hay unos 1.524 gramos incluyendo todas las piezas.

Hay que tener en cuenta que solo el drone y su batería de alto voltaje constituyen el 90% del peso del front-end y hay piezas que fácilmente se pueden cambiar por otras más livianas como la Protoboard.

#### 4.1.4. Presupuesto

Las siguientes tablas tienen como objetivo presentar los gastos encontrados durante la realización del proyecto, se clasifican en presupuesto básico y presupuesto hipotético para la continuación del proyecto.

Estos costes básicos fueron asumidos por el autor del prototipo. El resto del material utilizado, que no figura en la tabla de presupuestos (2 Arduinos, 1 protoboard y el módulo GPRS SIM900), fue proporcionado por la Universidad Pompeu Fabra:

*Tabla 10: Presupuesto del Proyecto. Fuente: Elaboración Propia*

Concepto	Coste
Cableado de Arduino auxiliar F-M	4,99 €
Cableado de Arduino auxiliar M-M	3,89 €
Cargador de Arduino Aukru	8,99 €
Power Conector de Batería 9V (x2)	5,99 €
Placa Ethernet Arduino	12,09 €
Batería alcalina de 9V	4,99 €
Batería alcalina de 9V	4,99 €
Joystick SATKIT (x2)	2,80 €
Controladora de vuelo SP_F3	24,99 €
Caja de materiales Tayg M231213	3,85 €
Tarjeta SIM prepago vodafone	20,00€
Crédito de datos consumidos	40,00€
<b>Total</b>	<b>137,57 €</b>

A continuación se propone material extra de cara al crecimiento del proyecto propuesto en la sección del documento de **Propuestas de Mejora**:

*Tabla 11: Presupuesto del Future Work. Fuente: Elaboración Propia*

Concepto	Coste
Costes básicos (sin tarjeta SIM)	77,57 €
Camera FPV de 25mW	23,46 €
GPS M8 N8	25,00 €
Microprocesador Arduino Mega	35,88 €
Kit Hexacopter with GPS	120,99 €
Mando de control Wireless Arduino	64,00 €
Licencia de Streaming Vimeo.com (al mes)	70,00 €
Licencia Estándar ThingSpeak.com	500,00 €
Estudio de Patente del Prototipo	900,00 €
Realización de Patente del Prototipo	1500,00 €
<b>Total</b>	<b>3316,90 €</b>

#### 4.1.5. Demostración de Funcionalidad

En esta sección, se hará un ejemplo práctico de la funcionalidad del prototipo, es decir, del circuito completo desde el back-end hasta el front-end, desde que se hace el primer input en el joystick, hasta visualizar la respuesta de la controladora de vuelo en el simulador de CleanFlight:

1. Se realiza un aumento de guiñada en el joystick número 1 (movimiento horizontal derecho) en el back-end del circuito como muestra la imagen a continuación:



Figura 46: Demostración, Guiñada en Joystick. Fuente: Elaboración Propia

2. Una vez se hace el input del valor vía joystick y es procesado por el Arduino, se puede observar la señal recibida en el servidor web Dweet.io bajo el nombre de ArduinoTest-RIFE-Project:

```
{"this":"succeeded","by":"getting","the":"dweets","with":  
[{"thing":"arduinotest-rife-project","created":"2018-06-  
07T19:31:35.057Z","content":{"|":188196188188}}]}
```

Aumento de Guiñada:  
Giro a la derecha

Figura 47: Demostración, Guiñada en Servidor RIFE. Fuente: Elaboración Propia

3. Se realiza un debugging de la señal recibida en el front-end del circuito comprobando que los datos transmitidos desde el back-end se han recibido correctamente en el front-end:

```

COM7 (Arduino/Genuino Uno)

OK
AT+HTTPPARA="URL", "http://dweet.io/get/latest/dweet/for/arduino
AT+HTTPACTION=0

OK

+HTTPACTION: 0, 200, 163

AT+HTTPREAD=141, 152

+HTTPREAD: 22
{ " | " : 188196188188 } ] }

OK
Data stored value: 188196188188

Autoscroll Sin ajuste de línea 9600 baudio

```

Figura 48: Demostración, Guiñada en Serial. Fuente: Elaboración Propia

4. Como último paso, el Arduino interpreta la información almacenada en la variable y la retransmite a la controladora de vuelo y, a continuación, podemos ver la respuesta por pantalla de la señal recibida en el simulador CleanFlight:

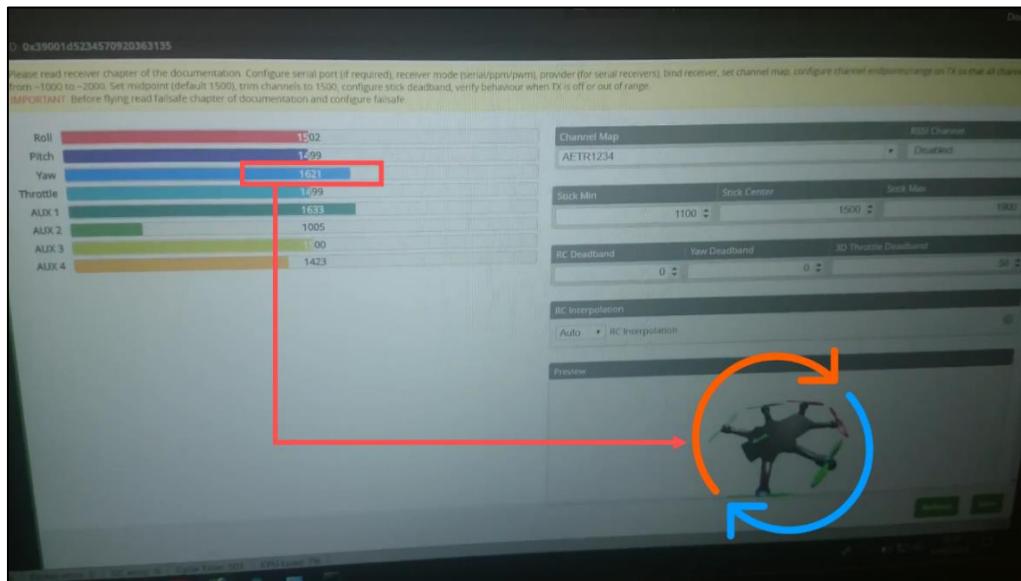


Figura 49: Demostración, Guiñada en Simulador. Fuente: Elaboración Propia

Por lo tanto, dado que la señal ha llegado satisfactoriamente al front-end desde el back-end, queda así demostrado que la integración del circuito ha sido exitosa y se da por concluido el proyecto.

### Video de Resultado:

En el siguiente [video](#) se muestra el proyecto finalizado con la comunicación desde el back-end hasta el front-end.

## 4.2. Dificultades y limitaciones encontradas

### 4.2.1. Dificultades de la Fase 1

**Limitación por Wasp mote:** Las primeras versiones de la fase 1 de desarrollo implicaron una importante inversión de tiempo. Se consideró la implementación por medio de la placa de desarrollo [Libelium Wasp mote](#) usada en proyectos similares. Sin embargo, el modelo de Libelium Wasp mote disponible por parte de la universidad qué, a pesar de tener puertos de recepción de señales analógicas, dichos puertos son incapaces de emisión del mismo tipo de señales por lo que la comunicación (Microprocesador -> Consola de vuelo) es imposible.

La solución fue cambiar a un microprocesador que sea capaz de enviar señales analógicas por al menos 4 pines. Por esta razón se por el uso de Arduino como microprocesador para el proyecto ya que dispone de hasta 6 pines para envío de este tipo de señales. Los pines en cuestión están marcados en la microprocesador por el símbolo “~” en frente del valor del pin.

En la siguiente imagen se puede apreciar como los canales 11, 10, 9, 6, 5 y 3 son capaces de transmisiones de carácter analógico:

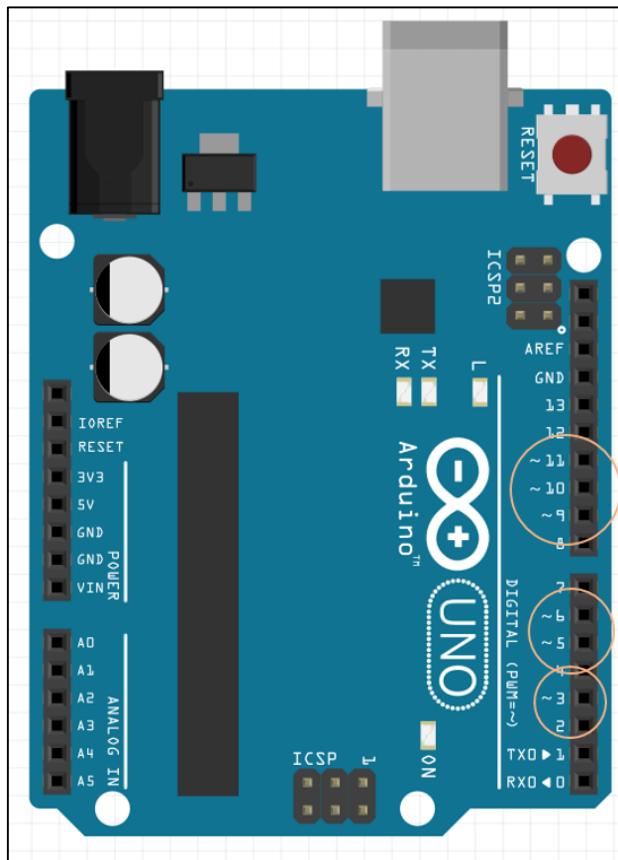


Figura 50: Pines Analógicos de Arduino UNO. Fuente: Elaboración Propia

**Controladora de vuelo rev6 10DOF:** Desafortunadamente la controladora de vuelo con la que se planeaba realizar el proyecto la rev6 10DOF (10 Degrees of Freedom) está fuera del mercado y fue reemplazada por la SP\_F3. Esto implicó un importante aprendizaje adicional ya que la nueva controladora usa un sistema distinto de control de señales al clásico PPM (Power Pulse Modulation). El nuevo sistema, llamado **CPPM** (Combined PPM), combina todas las señales de control de vuelo (Potencia, Alabeo, Cabeceo, Guiñada, Aux1, Aux2, Aux3 y Aux4) mediante algoritmos de codificación y cuantificación en una sola señal analógica como se muestra a continuación (Cableado Naranja):

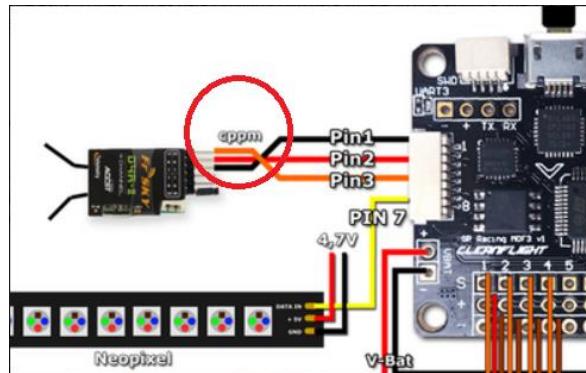


Figura 51: CPPM en una transmisora de Radio. Fuente: Elaboración Propia

El CPPM realiza una combinación de toda la información de control que viaja por una sola señal y esto representó un importante reto en el desarrollo del proyecto ya que el objetivo es el de controlar las 4 señales aéreas básicas individualmente.

Siguiendo el problema en cuestión, una solución que se pudo implementar para el control de señales individuales y no combinadas fue el de simular una transmisión PPM en lugar de una CPPM, lo cual implica el uso de ambos terminales de transmisión de la controladora de vuelo. Los terminales de transición son los puertos de conexión **IO\_1** y **IO\_2** como se puede apreciar en la Figura 51.

Cada terminal se compone por cinco canales. El primero asignado a tierra (**GND**), el segundo asignado a voltaje (**5V**), el tercero asignado a **Alabeo/Potencia** (dependiendo de si se trata del terminal izquierdo **IO\_1** o del terminal derecho **IO\_2**), el cuarto asignado a **Cabeceo/Guiñada**, y el quinto y sexto como canales auxiliares:

Pin	IO_1	IO_2
1	<b>GND</b>	<b>GND</b>
2	<b>5V</b>	<b>5V</b>
3	ALABEO	POTENCIA
4	CABECEO	GUIÑADA
5	AUX1	AUX3
6	AUX2	AUX4

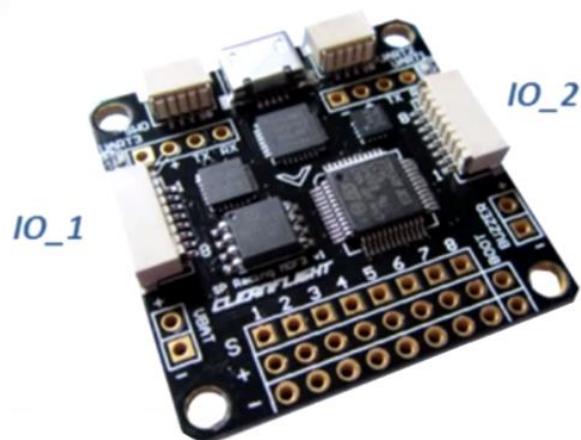


Figura 52: Señales en Terminales. Fuente: Elaboración Propia

**Integridad de señalización múltiple:** Otro de los múltiples obstáculos durante la primera fase de desarrollo se derivó de un mecanismo de seguridad oculto propio de la controladora de vuelo SP\_F3. El mecanismo en cuestión tiene un funcionamiento simple que se puede representar mediante el siguiente diagrama de puertas lógicas:

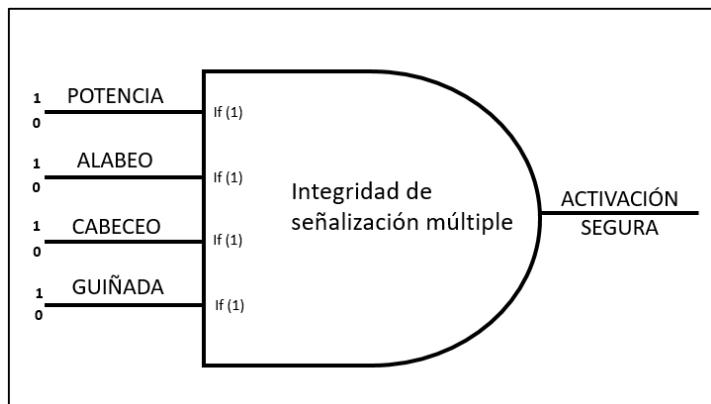


Figura 53: Integridad de señalización múltiple. Fuente: Elaboración Propia

El diagrama muestra como las 4 señales deben emitir un mínimo de voltaje para poder superar la limitación de seguridad establecida. Esta limitación existe para cumplir el objetivo de que, en caso de que una de las señales de control deje de recibir voltaje, o el voltaje que recibe caiga por debajo del **umbral mínimo de potencia**, el sistema se bloquee y no reciba ninguna de las otras señales.

#### 4.2.2. Dificultades de la Fase 2

**Adaptación de comunicación Wasp mote:** Durante las primeras etapas de desarrollo de la fase 2 de integración, se intentó adaptar un código de libre acceso de control y comunicación de módulos GPRS desarrollados para el entorno de los microprocesadores Libelium Wasp mote, que funciona de forma ligeramente distinto al de los microprocesadores Arduino.

Debido a la complejidad de adaptación de la librería de Libelium Wasp mote, se optó cambiar por un código más básico desarrollado para Arduino. Cuenta con considerablemente menos funciones pero con las necesarias para hacer funcionar el proyecto.

**Energía para la comunicación (batería/toma corriente):** Otro de los importantes contratiempos encontrados en la segunda fase de integración fue el de comprobar que los módulos GPRS requieren de una fuente de energía externa. El voltaje proporcionado por el puerto USB al cual está conectado el microprocesador Arduino es de entre 1,5V y 5V.

Ciertamente, la energía necesaria para la comunicación del módulo GPRS es de 5 voltios, no obstante, esta energía se recibe en forma de señal y virtualmente nunca se mantiene constantemente a 5V, por lo que fue necesario la incorporación del cargador Aukru que proporciona un voltaje de 9V, y posteriormente de una batería alcalina externa también de 9V.

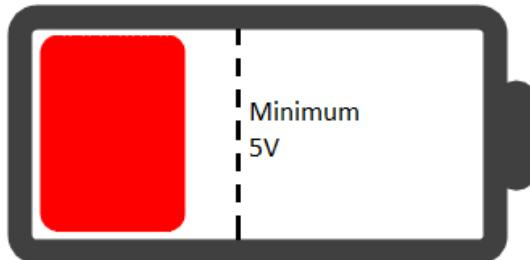


Figura 54: Batería insuficiente. Fuente: Elaboración Propia

### 4.2.3. Dificultades de la Fase 3

**ThingSpeak licencia básica:** La principal limitación que se encontró durante el desarrollo usando ThingSpeak como servidor web fue en los delays que se experimentaron durante las transmisiones.

En el mejor de los casos, el delay entre señal y señal era de unos 16 segundos mientras que el valor inicial del proceso tardaba aproximadamente unos 2 minutos en establecerse.

El delay de la transacción inicial se determinó que fue causado por la gestión de ThingSpeak. Cuando un servidor web deja de recibir señales, se categoriza como servidor web en standby y requiere de una serie de transacciones y de delays adicionales una vez se recupera la conectividad con el dispositivo en cuestión.

No obstante, una vez superado el delay de la transacción inicial, se experimentaba un delay de entre 15 y 20 segundos entre señalizaciones.

Después de varias pruebas y reducción de delays en las transacciones del microprocesador Arduino, se determinó la causa real del problema:

	FREE For time-limited commercial evaluation of the service	STANDARD For all commercial, government and revenue generating activities
Scalable for larger projects	✗ No. Annual usage is capped.	✓
Number of messages	3 million/year (~8,200/day) <sup>(4)</sup>	33 million/year per unit (~90,000/day per unit) <sup>(1)</sup>
Message update interval limit	Every 15 seconds	Every second
MATLAB Compute Timeout	20 seconds	60 seconds
Number of simultaneous MQTT subscriptions	Limited to 3	50 per unit
Private channel sharing	Limited to 3 shares	Unlimited
Technical Support	Forum	Standard MathWorks support

Figura 55: Modelos de Usuarios ThingSpeak. Fuente: <https://thingspeak.com/prices>

El modelo de uso de ThingSpeak no es de libre acceso sino es un modelo Freemium [12], lo que significa que es necesaria una inversión de 600 euros para una licencia estándar, y una inversión de tales magnitudes es completamente inviable para este proyecto.

**Ataque informático Dweet:** Durante las pruebas de la fase 3 de integración, el servidor web Dweet.io sufrió un ataque informático que inhabilitó los servidores durante varias horas:

The screenshot shows a tweet from the account @dweet\_io. The tweet content is: "We can confirm that our service is down. We are working now to bring dweet back online. Update shortly." The timestamp is 17:30 - 23 abr. 2018. The tweet has 1 like. The interface includes a blue circular profile picture, the handle @dweet\_io, a 'Seguir' (Follow) button, and a dropdown menu icon.

Figura 56: Twitter de Dweet.io. Fuente: [https://twitter.com/dweet\\_io](https://twitter.com/dweet_io)

El ataque en cuestión fue un ataque DDOS: “un ataque DDOS es cuando un grupo de personas o automatismos atacan a un servidor u ordenador desde muchos equipos a la vez. Este flujo masivo de datos hace que los recursos del servidor acaben no siendo suficientes, lo que provoca que colapse y deje de funcionar. Esto hace que si se trata de un equipo que mantiene una web, servicio o comunidad, esta caiga junto al servidor.” [13]

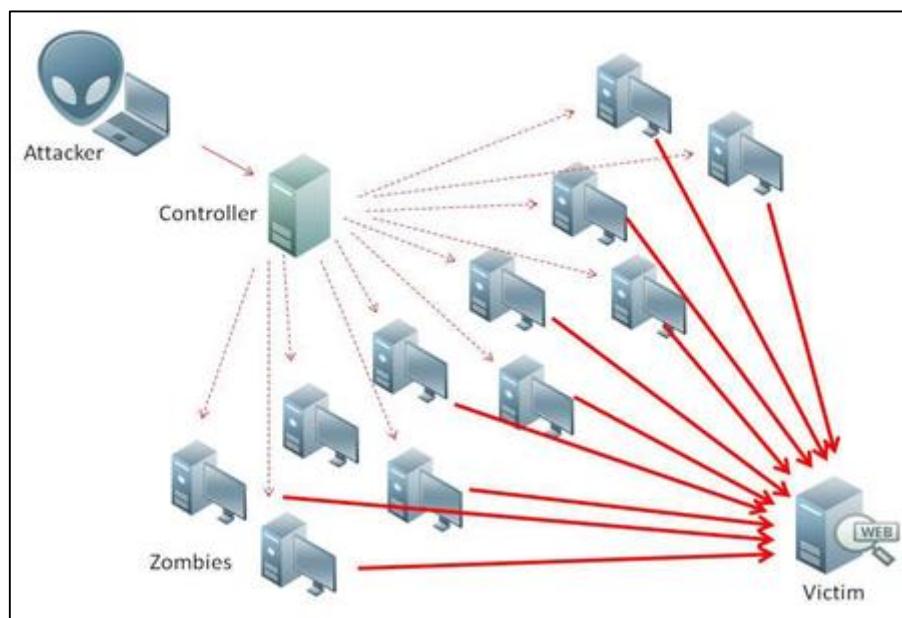


Figura 57: Esquema de Ataque DDOS. Fuente: <https://www.hd-tecnologia.com>

Este ataque, no solo provocó que los servidores de Dweet.io no estuvieran disponibles durante varias horas, sino también provocó un agotamiento de recursos indispensables para la comunicación entre el módulo GPRS y el servidor web.

**Insuficiencia de datos móviles:** A raíz del ataque DDOS explicado anteriormente, el módulo GPRS entró en un estado de repetición de peticiones continuo.

En concreto, el problema fue que el módulo GPRS inició la comunicación por internet mediante el comando *AT+HTTPINIT* pero fue incapaz de cerrar la comunicación mediante el comando *AT+HTTPTERM* lo que provocó una continua secuencia de peticiones de finalización de la comunicación. Esto saturó los datos móviles disponibles que tenía contratada la tarjeta SIM.

Varias horas después, cuando los servidores de Dweet.io volvieron a estar online, al intentar retomar las peticiones de información mediante el comando *AT+HTTPACTION=0* la respuesta recibida fue:

**+HTTPACTION: 0,601,0**

Dicho código de error significa que hay problemas en la red para gestionar la petición de GET que se ha solicitado.

Después de varios días de investigación, se determinó que este código de error pudo ser producido por los siguientes motivos:

1. Expiración del contrato de la tarjeta SIM.
2. No conectividad con la operadora.
3. Consumo de los datos móviles contratados.

Reemplazando la tarjeta SIM de pruebas con otra tarjeta SIM y cambiando las credenciales de APN, User y Password, el error se solucionó dando paso a la llegada de datos lo que quiere decir que el problema radicaba en los datos móviles insuficientes.

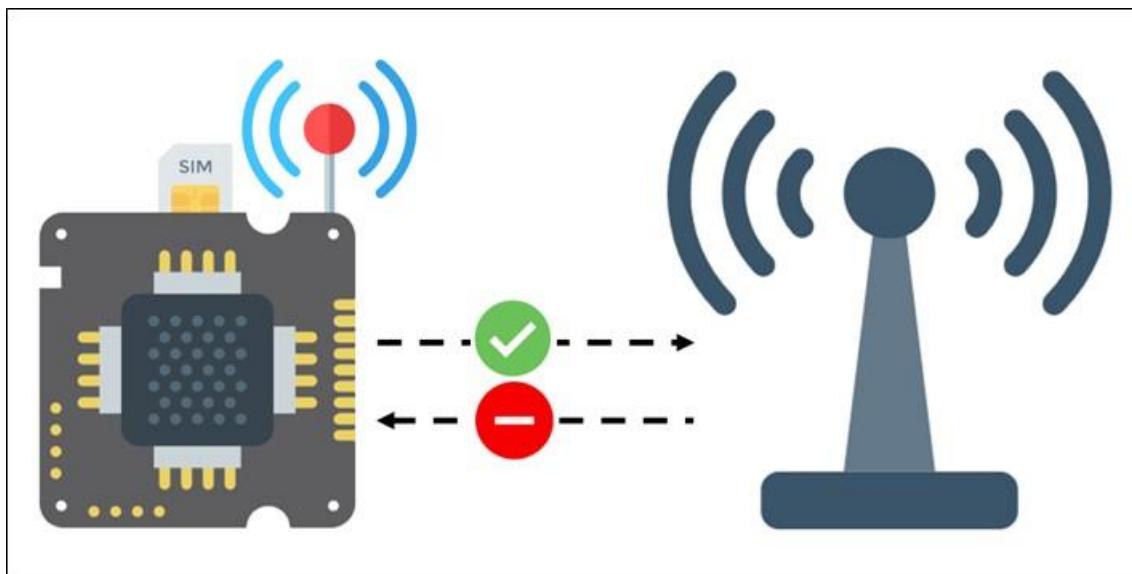


Figura 58: Esquema de Insuficiencia de Datos Móviles. Fuente: Elaboración Propia

#### 4.2.4. Dificultades en la fase 4

**Velocidad de peticiones RIFE:** Una de las principales dificultades encontradas en la fase 4 de integración fue el número de peticiones por segundo que se hacían de la función Rife\_MasterFunction.

Originalmente se seguía una estructura muy similar al código desarrollado en la fase 1 de integración, donde cada cambio puntual en una de las señales implica un envío de datos a la controladora de vuelo. El número de envío de señales por minuto en la fase 1 de integración es de aproximadamente 840 RPM (Requests Per Minute) o bien 14 envíos por segundo.

En el caso de la fase 4 de integración, inicialmente cada cambio puntual en una de las señales implica establecer una conexión, es decir, se realizan aproximadamente 14 conexiones con el servidor Dweet.io por segundo.

Esto implica una saturación de los recursos de la red y como resultado la placa Ethernet experimentaba un starving [14]. Pocos segundos de comenzar las pruebas se pierde la conexión y es necesario reiniciar el microprocesador.

La solución fue reducir el número de peticiones a 60 RPM o bien 1 petición por segundo, lo cual es tiempo suficiente para que el microprocesador pueda abrir las comunicaciones con el servidor web, hacer el envío y cerrar las comunicaciones. Sin embargo, esta solución tiene un coste para el proyecto en términos de delay, ya que aumenta el delay entre lecturas.

**Establecer conexión bridge Wi-Fi-Ethernet:** Una de las características propias de la fase 4 de integración es el de ¿Cómo conseguir conectar a Internet la placa Ethernet?

La respuesta rápida a este pregunta es conectándolo directamente a un router [15] con acceso a Internet (es una solución perfectamente válida y funcional). No obstante, todas las pruebas del proyecto se realizaron en el mismo ordenador incluyendo el debugging de la fase 4, por lo que es necesario que tanto la conexión a Internet como la visualización de los comandos de Joystick se realicen conjuntamente en un solo ordenador (figura 58).

La opción que se propuso es conectar la placa Ethernet del microprocesador Arduino al puerto Ethernet del ordenador en cuestión que se está usando para el control del circuito. Para esto es necesario establecer lo que se denomina una conexión bridge (puente) entre la entrada Ethernet y (en el caso de este proyecto) el Wi-Fi que proporciona acceso a la red.

“Una conexión bridge permite conectar 2 o más segmentos de red permitiendo dispositivos unirse a la red cuando no es posible conectarlos directamente al router o switch.”



Figura 59: Conexión bridge para Arduino. Fuente: Elaboración Propia

Esto habilita una conexión virtual entre ambos puntos de acceso proporcionándole al Arduino acceso directo a Internet.

**Perdida de paquetes HTTP:** Esta dificultad no es exclusivamente parte de la fase 4 de integración sino del sistema como conjunto.

Intentando reducir al máximo los delays para cumplir con los objetivos del proyecto planteados, la principal sección donde existe mayor delay es durante la descarga de datos del servidor web por parte del módulo GPRS (explicado en la fase 2 de integración).

Inicialmente, se emulaba un entorno similar al propuesto por ThingSpeak con limitación de datos por segundo. Es decir, un delay de aproximadamente 11,5 segundos entre descargas de información del servidor web Dweet.io.

Con el objetivo de reducir esta importante cantidad de delay, se redujeron el tiempo de espera del microprocesador en varios comandos:

**AT+HTTPINIT, AT+HTTPACTION=0 y AT+HTTPREAD.**

Dichas reducciones de delay pueden apreciarse en la sección de **Resultados** del documento.

Se logró reducir a aproximadamente a 6,5 segundos entre descargas de información, reduciendo los momentos de espera por parte del microprocesador Arduino. No obstante, al tratar de reducir aún más este delay empiezan a aparecer nuevos problemas de conectividad.

La velocidad de descarga e interpretación de los comandos no puede aumentarse más allá de los 6,5 segundos ya que el módulo GPRS requiere de un mínimo de tiempo para procesar la información y, al enviar con aún menos tiempo de espera el siguiente comando, los paquetes en cola se pierden lo que implica una pérdida de conexión y es necesario reiniciar el microprocesador. Este fenómeno se puede apreciar en las figuras 35 y 36 en la sección 3.4.1 Subintegración Fase 2 y 3.

Este tiempo mínimo de espera, 6,5 segundos, es varias veces superior al objetivo establecido de 2 segundos. No obstante, sabemos que muchas de las limitaciones del proyecto provienen del material proporcionado y no necesariamente de la estructura y ejecución del mismo.

Página en blanco intencional

# 5. Conclusiones

## 5.1. Conclusiones a partir de resultados

Una vez analizados los resultados obtenidos a continuación se estudiará la viabilidad del proyecto en cada uno de los objetivos establecidos inicialmente.

### 5.1.1. Objetivo Principal

El objetivo principal era el de construir un circuito que comunique el drone correspondiente al front-end vía redes móviles con su respectivo operador en el back-end.

No solo es hipotéticamente posible, sino que además se ha conseguido: plantear, diseñar, programar, montar y probar todo lo necesario para el funcionamiento del circuito completo en un plazo inferior a 8 meses. Por lo tanto la respuesta a la pregunta hipótesis planteada es:

“¿Es posible realizar la comunicación entre drone y operador a través de las redes móviles?”

**Si es posible** y se ha demostrado como tal.

### 5.1.2. Objetivos específicos

#### 1. Delay:

Es el más importante de los objetivos específicos ya que determina la viabilidad de, una vez establecida, que tan fluida y sensible es la comunicación entre el front-end y el back-end.

Desafortunadamente la meta era la de conseguir un delay end-to-end de menos de **2 segundos** pero **este objetivo no pudo ser cumplido** principalmente por las limitaciones físicas que plantea el módulo GPRS que se ha utilizado en el proyecto.

El delay promedio experimentado end-to-end es de aproximadamente **10 segundos**, 5 veces mayor al objetivo.

Este resultado se debe a la obligación de usar un módulo GPRS (2G) para el desarrollo del proyecto que ofrece una velocidad de conexión 200 veces más lenta que las de 3G (14.4Kbps en 2G VS 2Mbps en 3G).

#### 2. Comunicación ininterrumpida:

Este objetivo es el de comprobar que, una vez comiencen las comunicaciones, durante al menos **3 minutos** debe haber un flujo constante de datos entre el drone y el operador en cuestión.

**Este objetivo fue cumplido** durante el desarrollo final. Las comunicaciones entre el front-end y el back-end se mantuvieron constantes e hipotéticamente **se pueden mantener durante un tiempo indefinido** siempre y cuando:

- Se respeten los valores de delay mínimos de procesamiento.
- Se mantenga el drone dentro del rango de cobertura de la operadora.
- Haya suficiente batería para alimentar al módulo GPRS durante todo el trayecto.
- La tarjeta SIM tenga suficientes datos móviles para la comunicación.

### 3. Estabilización aérea:

Se indicó que es necesario desarrollar una **función de auto-estabilización aérea** en caso de que alguno de los 4 criterios mencionados en el objetivo de Comunicación Ininterrumpida sea transgredido y se pierda la comunicación entre el front-end y el back-end.

Este objetivo fue cumplido relativamente temprano en el desarrollo del proyecto específicamente durante la elaboración del código del front-end mediante la función `StableFunction()`;

### 4. Peso:

El peso objetivo que se propuso al iniciar el proyecto fue de mantener el front-end por debajo de los **1.500 gramos**.

Aunque técnicamente el peso final del front-end fue de **1.524 gramos** (1.360 del drone y 164 del circuito adicional), inicialmente no se tomó en cuenta que el 90% del peso correspondería al drone y a su batería. También se decidió cambiar de modelo de multirrotor, de un Cuadracoptero a un Hexacoptero, lo cual aumentó el peso total del drone. Además, fácilmente podemos reducir este peso cambiando algunas piezas por otras más livianas, por lo que se considera que este objetivo fue cumplido.

### 5. Presupuesto:

En todo tipo de proyecto es fundamental tener en cuenta el presupuesto y recursos disponibles. Para este proyecto el objetivo fue el de mantener los costes del proyecto por debajo de los **150 euros**.

Este objetivo fue cumplido al mantener los gastos personales en piezas, herramientas, licencias y fabricación de la versión base del proyecto por debajo de los **138 euros**.

No obstante, de haberse querido desarrollar la versión completa del prototipo físico hubiese sido necesario una inversión inicial de **3.300 euros** y luego, aproximadamente **1.725 euros** para la fabricación de cada unidad (restando los gastos de patentes).

## 5.2. Propuestas de mejoras

Durante el desarrollo del proyecto se contemplaron potenciales mejoras de las que podría beneficiarse el prototipo Sky 3Guardian que, por términos de presupuesto, se descartaron del desarrollo básico.

Estas potenciales mejoras son las siguientes:

### 5.2.1. Sistema de Streaming FPV

El objetivo del proyecto es el de tener una distancia de control muy superior a la convencional (entre 1.000 y 6.000 metros). Por lo tanto, dado que a partir de los 500 metros deja de ser legal (y físicamente posible) el control a simple vista [16] de un drone, es necesario diseñar un sistema de FPV ya que se busca controlar el drone a más de 6.000 metros de distancia.

Es necesario añadir al prototipo Sky 3Guardian una Cámara FPV. Para esto se recomienda una Cámara HD 700tvl para minimizar costes.

Este sistema deberá ser incorporado a la comunicación móvil entre el front-end y el back-end, por lo que la transmisión desde la Cámara FPV hasta el back-end se tiene que desarrollar vía Internet.

Una de las soluciones que se propone es hacer un Streaming [17] del visual recibido por la Cámara FPV. Esta solución representaría una importante integración al proyecto y se necesitaría una velocidad de conexión considerablemente superior a la usada en el proyecto (2 Mbps en 3G). A pesar de que 4G puede realizar streamings a 100 Mbps, idealmente se usaría 5G para disponer de una velocidad de 1.000 Mbps.

Por supuesto, el usar 4G o 5G conlleva importantes limitaciones de alcance de señal con la operadora contratada.



Figura 60: Cámara HD 700tvl. Fuente: <http://www.tri-ed.com/>

## 5.2.2. Control de Altitud y Aterrizaje de Emergencia

Otra de las mejoras planteadas durante el desarrollo del prototipo fue el de incluir un sensor de altitud.

El objetivo de dicho sensor es, al accionar cierto trigger en el back-end (i.e. un botón auxiliar en los joystick), aportar información a la función ***StableFunction()*** para mantener una altura determinada, ya sea la altura actual o ascender/descender a una altura de control (i.e. 5 metros).

Para completar dicho objetivo, hay que tener en cuenta cual es la pieza y como será incorporada en el prototipo.

El sensor en cuestión puede ser un sensor LIDAR (Light Detection And Ranging): “es un **sensor activo** que permite medir la distancia entre el dispositivo y un determinado objeto mediante la emisión de un haz láser pulsado”. [18]



Figura 61: Sensor de distancia LIDAR. Fuente: <http://www.xdrones.es/>

La integración en el prototipo puede realizarse en la base del drone, conectado al sistema energético de la controladora de vuelo (mismo sistema que da energía a las hélices) y también conectado al Arduino del front-end para decidir qué acciones tomar con la información recibida.

A pesar de que la función ***StableFunction()*** garantiza que el drone no caiga súbitamente del cielo por un problema en la conexión, no se ha desarrollado una funcionalidad para evitar una colisión a gran velocidad con el suelo por una eventual falta de batería.

Entonces se pensó en que en un futuro, se podría reutilizar el sensor de distancia LIDAR junto con un medidor de voltaje de la batería para desarrollar una función llamada ***EmergencyLanding()***.

Esta función lee dos parámetros, el nivel de batería del drone y la altura actual del mismo. En caso de que la batería del drone caiga **por debajo del 5%** y la altura de vuelo actual sea **superior a 10 metros**, esta función se activa haciendo descender al drone mediante ráfagas de potencia hasta aterrizar en cualquier superficie (de momento, no se contempla la posibilidad de aterrizar en cuerpos de agua o zonas restringidas).

### 5.2.3. Global Positioning System (GPS)

Por último, la tercera mejora que se planteó durante el desarrollo del prototipo Sky 3Guardian fue el de incluir un dispositivo GPS para mantener constancia de la ubicación exacta del drone en todo momento.

Si por algún error se pierde la conexión (i.e. pruebas de la operadora) se podrá rastrear la ubicación actual del drone y gracias a la función de *EmergencyLanding()* explicada anteriormente, solo sería necesario esperar a que se agote la batería del drone y eventualmente aterrice.

Se recomienda el GPS Module M8N 8N que ya viene preparado para ser integrado con la controladora de vuelo SP\_F3.



Figura 62: Módulo GPS M8N 8N. Fuente: <https://www.dzduino.com/>

La integración en el prototipo puede realizarse en la cubierta del drone para disponer de la máxima capacidad de comunicación satelital.

### 5.2.4. Prototipo: Sky 4Guardian

Teniendo en cuenta todas las integraciones mencionadas previamente, el prototipo que se presenta a continuación es una evolución del prototipo básico Sky 3Guardian.

El nombre del prototipo propuesto es **Sky 4Guardian**, llamado así porque requiere de una conexión de 4G.

Contará con una licencia Premium de ThingSpeak que permite el envío de datos a máxima velocidad y también permite registrar datos de importancia de vuelo como pueden ser velocidad de rotación de los motores, velocidad del viento, altura actual, temperatura, etc.

El microcontrolador que se comunica con la Consola de vuelo SP\_F3 será un [Arduino Mega](#) que cuenta con un mayor número de puertos de conexión para añadir sensores para que permitan medir los parámetros mencionados anteriormente. Además, el Arduino Mega cuenta con mayor espacio de memoria para la ejecución de códigos de programación más robustos y complejos.

Tendrá incorporado un sistema de visualización en primera persona FPV para poder ver en todo momento lo que ve el drone en el back-end.

Como sistemas de seguridad se añaden un Sensor de distancia LIDAR que mide la altitud con respecto al suelo, un sistema de GPS y funcionalidades de aterrizaje de emergencia en caso de algún tipo de fallo catastrófico.

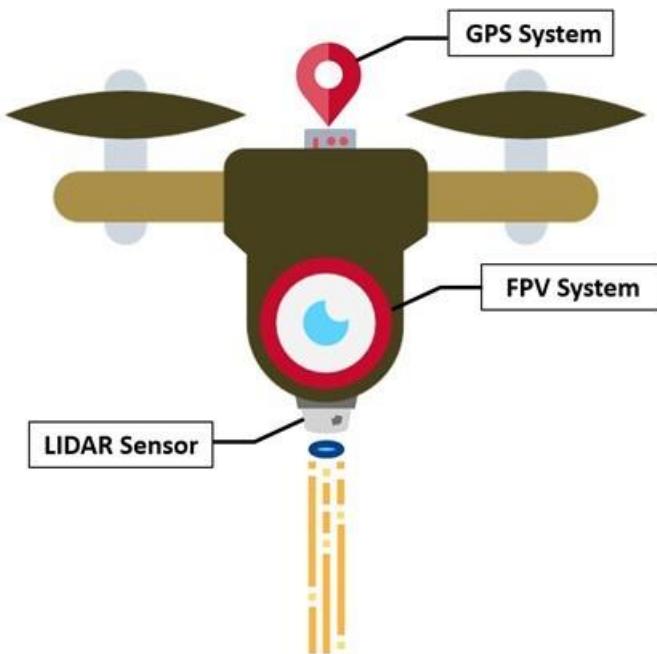


Figura 63: Esquema de Prototipo Sky 4Guardian. Fuente: Elaboración Propia

## **5.3. Future Work**

Debo finalizar agregando que la realización de este proyecto ha sido una experiencia muy especial para mí. No sólo me ha ayudado a aplicar los conocimientos adquiridos a lo largo de mi Grado de Ingeniería Telemática, sino que me ha hecho crecer como persona y como profesional.

Tal y como comenté al explicar mis motivaciones, el desarrollo de un prototipo real, me ha permitido materializar un producto que combina muchos factores técnicos y funcionales.

Tengo la certeza de que esta tecnología propuesta, puede ser aplicada a muchos campos científicos, gubernamentales y comerciales, en concreto en actividades como fotografía, agricultura, construcción, control de tránsito, operaciones de rescate, cobertura de eventos: deportivos, peligrosos, bélicos, climatológicos, entre muchos otros.

Luego del análisis realizado sobre el Estado del Arte, y comprobando que las patentes registradas en la Unión Europea se identificó una potencial oportunidad para continuar desarrollar el proyecto. Al plantearlo a diferentes inversores, la idea fue recibida con mucho entusiasme y contaré con su apoyo para registrar una patente del prototipo Sky 4Guardian en las próximas semanas.

Página en blanco intencional

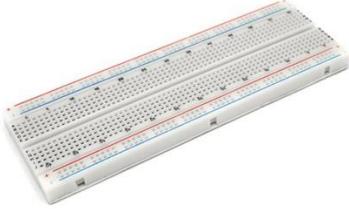
# Anexos

## Anexo 1. Material

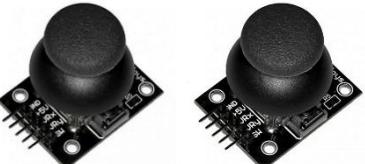
Durante el proyecto fue necesario el uso de los siguientes materiales físicos:

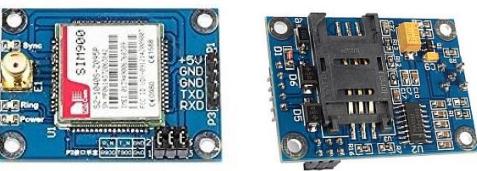
<b>Nombre:</b>	<b>1. Microprocesador Arduino UNO/YUN</b>
<b>Descripción:</b>	<p>“Es una plataforma de hardware libre, basada en una placa con un microcontrolador y un entorno de desarrollo, diseñada para facilitar el uso de la electrónica en proyectos multidisciplinares” [19].</p> <p>Funciona como interfaz de comunicación entre el código desarrollado y la controladora de vuelo.</p> <p>El modelo utilizado fue el modelo Arduino UNO y YUN, uno de los más básicos que ofrece la marca. Este fue el modelo utilizado debido a la disponibilidad de material en la universidad.</p>

A photograph of an Arduino Uno R3 microcontroller board. The board is blue and features the Arduino logo. It has a USB port on the left, a microcontroller chip in the center, and various pins, capacitors, and resistors. The word "UNO" is printed on the right side of the board.

<b>Nombre:</b>	<b>2. Protoboard (2 unidades)</b>
<b>Descripción:</b>	<p>“Una protoboard (breadboard en inglés) es una placa que posee unos orificios conectados eléctricamente entre sí siguiendo un patrón horizontal o vertical. Es empleada para realizar pruebas de circuitos electrónicos, insertando en ella componentes electrónicos y cables como puente.” [20]</p> 

<b>Nombre:</b>	<b>3. Controladora de vuelo SP_F3</b>
<b>Descripción:</b>	<p>Es el centro de mando del RPA (drone). Es el encargado de controlar la velocidad de rotación de las aspas, el grado de alabeo, cabeceo y guiñada. Se ha previamente de configurar mediante un programa externo usando un conector USB de móvil corriente. También dispone de opciones para conectar LEDs, baterías externas de alta potencia junto con GPS y sistemas de visualización en primera persona (First Person View FPV). Cabe destacar que esta controladora de vuelo es lo que se denomina una <b>controladora manual</b>, es decir, el voltaje transmitido por el canal de potencia corresponde a la velocidad de las aspas. En comparación, otras controladoras de carácter automático usan este canal para determinar la altura del RPA respecto a tierra.</p> <p>Se utilizó este modelo debido a la limitación de coste que suponen las controladoras automáticas más sofisticadas.</p> 

<b>Nombre:</b>	<b>4. Joystick SATKIT para Arduino (2 unidades)</b>
<b>Descripción:</b>	<p>Son componentes compatibles con Arduino que actuarán principalmente como el mando de control desarrollado durante la fase 4. También son usados durante la fase 1 para comprobar a modo de prueba para comprobar las lecturas analógicas. Se compone de 5 pines: Tierra (<b>GND</b>), Voltaje (<b>5V</b>), Control Horizontal (<b>VRx</b>), Control Vertical (<b>VRy</b>) y click del joystick (<b>SW</b>).</p> 

<b>Nombre:</b>	<b>5. Módulo Sim900 GPRS</b> (General Packet Radio Service)
<b>Descripción:</b>	<p>Es una placa externa que permite insertar una tarjeta SIM (típicamente una tarjeta de móvil) y realiza comunicaciones con el microprocesador Arduino. Estas comunicaciones se realizan mediante <i>comandos AT</i><sup>[6]</sup>. Además del slot para tarjeta SIM, cuenta con un conector de antenas y 5 pines de conexión: Tierra (<b>GND</b>) x2, Voltaje (<b>5V</b>), transmisión de información (TXD) y recepción de información (RXD).</p> <p>Requiere de una fuente externa de alimentación para realizar las peticiones móviles entre el modulo y la antena móvil a la que está conectada. Esta fuente de energía se proporciona al Arduino mediante una conexión directa a la red eléctrica de cualquier toma corriente o una batería alcalina de 9 voltios.</p> 

<b>Nombre:</b>	<b>6. Tarjeta SIM</b>
<b>Descripción:</b>	Comúnmente usadas en dispositivos móviles de nuestro día a día. Tiene contratado un plan específico para este proyecto centrado en datos móviles. La conexión realizada con el APN de la compañía (Access Point Network) puede ser 3G/4G. Mientras que 3G tiene cobertura virtualmente ilimitada en la unión europea, 4G ofrece una velocidad de conexión mucho mayor.



<b>Nombre:</b>	<b>7. Antena de comunicación móvil para módulo GPRS</b>
<b>Descripción:</b>	Se trata de una pequeña extensión conectada al módulo GPRS SIM 900 para ampliar el alcance de comunicación entre el modulo el APN de la compañía contratada.



<b>Nombre:</b>	<b>8. Cargador de Arduino Aukru (9V 1000mA)</b>
<b>Descripción:</b>	<p>Es un cargador de corriente que funciona como fuente de alimentación externa para el Arduino y consiguientemente para el módulo GPRS. Generalmente, un microprocesador Arduino no necesita de fuentes externas de alimentación mientras estén conectados vía USB a una fuente de energía (típicamente un ordenador). No obstante, el módulo GPRS y el modulo Ethernet usados en las fases 2 y 4 de integración requieren de una mayor fuente de energía que la proporcionada por la conexión USB.</p> <p>Es la fuente de energía utilizada durante el back-end del proyecto (Arduino YUN como mando de control vía Ethernet) para vitalizar la placa Ethernet de conexión red.</p> 

<b>Nombre:</b>	<b>9. Power Conector para batería alcalina de 9V</b>
<b>Descripción:</b>	<p>Similar al Cargador Aukru, proporciona energía externa para el Arduino. Su ventaja respecto al cargador es que puede conectarse a una batería alcalina de 9V por lo que es ideal para el montaje electrónico correspondiente al RPA que estará elevado a varios metros del suelo.</p> 

<b>Nombre:</b>	<b>10. Batería alcalina de 9V</b>
<b>Descripción:</b>	Es la fuente de energía utilizada durante el front-end del proyecto (RPA en vuelo).



<b>Nombre:</b>	<b>11. Placa Ethernet para Arduino UNO/YUN</b>
<b>Descripción:</b>	<p>Se trata de un adaptador de conexión Ethernet diseñado específicamente para microprocesadores Arduino UNO o YUN.</p> <p>Debido a que las placas de Arduino usadas en el proyecto solo cuentan con los elementos más básicos para realizar proyectos electrónicos, fue necesario instalar un adaptador que permita al microprocesador en cuestión hacer peticiones HTTP vía Internet.</p>



Placa Ethernet Desconectada



Arduino junto con Placa Ethernet incorporada

Así mismo también fue necesario el uso de programas informáticos que llevaron a cabo distintas funciones clave durante el proyecto:

<b>Nombre:</b>	12. Arduino IDE (Integrated Development Environment)
<b>Descripción:</b>	<p>Ofrece una simple interfaz de desarrollo de código. Usa el lenguaje de programación C y C++ como base de desarrollo.</p> <p>En este software se desarrollaron, compilaron y cargaron todas las versiones de código de las distintas 4 fases del proyecto. Ofrece librerías pre-instaladas para el desarrollo aunque fue necesario descargar las librerías SoftwareSerial y Ethernet para las fases 2 y 4 de desarrollo respectivamente.</p> <p>Cada archivo debe seguir la estructura de realizar un <b>Setup</b> inicial (típicamente para inicializar Serials y activar pines de entrada y salida) y un <b>Loop</b> que permite repetir la rutina programada.</p>

```

/*
Blink
Turns on an LED on for one second then off for one second, repeatedly.

This example code is in the public domain.
*/
int led = 13;

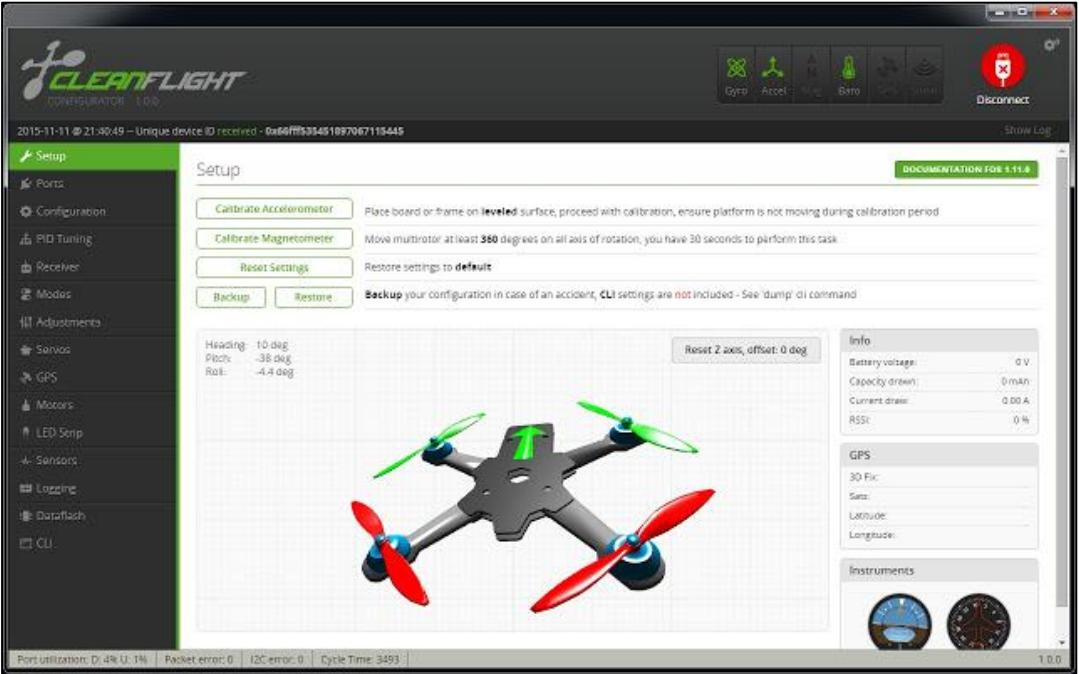
void setup() {
  // initialize digital pin 13 as an output.
  pinMode(led, OUTPUT);
}

// the loop function runs over and over again forever
void loop() {
  digitalWrite(13, HIGH);    // turn the LED on (HIGH is the voltage level)
  delay(1000);              // wait for a second
  digitalWrite(13, LOW);     // turn the LED off by making the voltage LOW
  delay(1000);              // wait for a second
}

```

Done compiling.

<b>Nombre:</b>	14. Fritzing
<b>Descripción:</b>	Este programa permite diseñar diagramas de circuitos. Contiene una gran base de datos de distintas partes electrónicas para personalizar todo tipo de circuitos electrónicos.

<b>Nombre:</b>	15. Cleanflight
<b>Descripción:</b>	<p>Es un programa desarrollado en código libre [21] que permite la configuración y simulación de varios tipos de RPA, desde dispositivos de un solo motor hasta todo tipo de multirrotores (Cuadracopteros, Hexacopteros, Octocopteros, etc.).</p> <p>Este programa es fundamental durante el desarrollo del proyecto ya que funciona como simulador de las acciones del RPA sin la necesidad de un RPA ya ensamblado. Esto evita los peligros derivados de las pruebas de aeronaves en un espacio aéreo real.</p> 

<b>Nombre:</b>	<b>16. Servidor web: <a href="#">ThingSpeak</a></b>
<b>Descripción:</b>	Es una solución para productos y servicios de Internet of Things. Similar a WordPress permite a los usuarios crear blogs fácilmente y además, permite a los desarrolladores interactuar con dispositivos que envían distintos tipos de señales vía Internet.

<b>Nombre:</b>	<b>17. Servidor web: <a href="#">Dweet</a></b>
<b>Descripción:</b>	Es una solución aún más simple que ThingSpeak. Permite crear una URL online a la cual se pueden realizar peticiones POST o GET para incorporar nuevos caracteres de hipertexto o recibir la información que fue previamente cargada.

## Anexo 2. Vídeos

Se incluyen los enlaces directos a vídeos privados en YouTube sobre la progresión del proyecto:

1. Primer contacto: <https://www.youtube.com/watch?v=x6knuPOz-Zs>
2. Estabilización y Aceleración: <https://www.youtube.com/watch?v=RQcXQeBLRMM>
3. Fase 1 de Integración Completada: <https://www.youtube.com/watch?v=X-JSXbe4LLI>
4. Fase 4 de Integración Completada: [https://www.youtube.com/watch?v=F5I5TR\\_v3ZE](https://www.youtube.com/watch?v=F5I5TR_v3ZE)
5. Proyecto Finalizado: <https://www.youtube.com/watch?v=eT6kba5WcF4>

## Anexo 3. Datasheets y Documentación

Documentos de utilidad que proveen información técnica sobre hardware o software.

1. Arduino UNO: <https://store.arduino.cc/usa/arduino-uno-rev3>
2. Controladora de Vuelo SP\_F3: <http://seriouslypro.com/files/SPRacingF3-Manual-latest.pdf>
3. Protoboard: <http://instrumentacion.qi.fcen.uba.ar/docs/protoboard.pdf>
4. Módulo GPRS SIM900: <https://www.sainsmart.com/products/sim900-gprs-gsm-board-quad-band-module-kit-for-arduino>
5. Joystick SATKIT: <https://www.prometec.net/joystick-servo/>
6. Ethernet Shield for Arduino:  
[https://www.mouser.com/catalog/specsheets/A000056\\_DATASHEET.pdf](https://www.mouser.com/catalog/specsheets/A000056_DATASHEET.pdf)
7. Comandos AT: <http://www.electronicaestudio.com/docs/ISTD-034.pdf>
8. GitHub: <https://github.com/>
9. Arduino Mega: <https://store.arduino.cc/usa/arduino-mega-2560-rev3>

Página en blanco intencional

# Referencias

- [1] CNN Tech. “Drones go mainstream”. 2013 | <http://money.cnn.com/technology>.
- [2] Microdrones. “Industry Experts”. 2017 | <http://microdrones.com>.
- [3] Lucia Silva. “Ventajas y desventajas de los drones” 09/01/2017 | <http://dronesweb.net>
- [4] Eldrone. “Historia de los drones” 01/06/2017 | <http://eldrone.es/historia-de-los-drones/>
- [5] Biografías y Vidas. “Hermanos Wright”. 2004 | [www.biografiasyvidas.com/biografia/w/wright\\_hermanos.htm](http://www.biografiasyvidas.com/biografia/w/wright_hermanos.htm)
- [6] Wortmann, F., & Flüchter, K. (2015). Internet of things. Business & Information Systems Engineering, 57(3), 221-224.
- [7] Jose Angel Zamora. “Beidou así es la tecnología china que sustituye al GPS”. 06/08/2016 | <https://elandroidelibre.elespanol.com/2016/08/beidou.html>
- [8] Wiki.Bandaancha. “APN de las operadoras”. 2017 | [https://wiki.bandaancha.st/APN\\_de\\_las\\_operadoras\\_para\\_configurar\\_el\\_m%C3%B3dem\\_de\\_Internet\\_m%C3%B3vil\\_3G/4G\\_LTE](https://wiki.bandaancha.st/APN_de_las_operadoras_para_configurar_el_m%C3%B3dem_de_Internet_m%C3%B3vil_3G/4G_LTE)
- [9] Aprendiendo Arduino. “Ethernet Shield”. 2017 | <https://aprendiendoarduino.wordpress.com/2016/07/04/ethernet-shield/>
- [10] Gnome Help. “¿Qué es una dirección MAC?”. 2017 | <https://help.gnome.org/users/gnome-help/stable/net-macaddress.html.es>
- [11] Julián Pérez Porto y Ana Gardey. “Definición de cliente servidor”. 2018 | <https://definicion.de/cliente-servidor/>
- [12] Diego Colagrosso. “¿Qué es el modelo Freemium?” 19/09/2011 | <https://winred.com/ideas-negocios/que-es-el-modelo-freemium/gmx-niv101-con23654.htm>
- [13] Yúbal FM. “¿Qué es un ataque DDoS y cómo puede afectarte?”. 06/04/2017 | <https://www.xataka.com/basicas/que-es-un-ataque-ddos-y-como-puede-afectarte>
- [14] Arno Formella. “Espera infinita o inanición de procesos”. 2005 | <https://www.formella.webs.uvigo.es/doc/cd04/node48.html>
- [15] Victoria Bembibre. “Router”. 06/01/2009 | <https://www.definicionabc.com/tecnologia/router.php>
- [16] GeoDrone. “Legalidad en la Operativa con drones en España”. 14/11/2015 | <http://www.geodrone.es/legalidad-en-la-operativa-con-drones-en-espana/>
- [17] ITE Educación. “¿Qué es el streaming?”. 2008 | <http://www.ite.educacion.es/formacion/materiales/107/cd/video/video0103.html>
- [18] Ernesto Santana. “Sistema LIDAR para drones”. 03/04/2017 | <http://www.xdrones.es/sistema-lidar-para-drones/>

[19] Aprendiendo Arduino. “¿Qué es Arduino?”. 2017 | <https://aprendiendoarduino.wordpress.com/2016/09/25/que-es-arduino/>.

[20] Tú Electrónica. “¿Qué es la Protoboard?”. 04/02/2016 | <https://tuelectronica.es/que-es-la-protoboard/>

[21] Augusto Blanco. “¿Qué es Código Abierto?”. 29/10/2009 | <https://www.informatica-hoy.com.ar/software-libre-gnu/Que-es-Codigo-Abierto.php>