Universidad del País Vasco Euskal Herriko Unibertsitatea
BILBOKO INGENIARITZA ESKOLA
ESCUELA DE INGENIERÍA DE BILBAO

GRADO EN INGENIERÍA EN TECNOLOGÍA DE TELECOMUNICACIÓN

TRABAJO FIN DE GRADO

MOVIMIENTO AUTONÓMO DE UN DRON ENTRE DOS PUNTOS EN ESPACIOS INTERIORES

Alumno/Alumna: Llaguno, Sánchez, Igor
Director/Directora: Espinosa, Acereda, Jon Koldobika

Curso: 2017-2018

Fecha: 23 de Julio de 2018

Resumen Trilingüe

Resumen

El proyecto consiste en la implementación de un dron que sea capaz de desplazarse de forma autónoma de un punto a otro en espacios interiores. Para ello, se desarrollará un software en una SBC capaz de gestionar los diferentes elementos que conforman la aeronave no tripulada.

El fin de este proyecto es conseguir un cuadricóptero autónomo, en el que el factor humano interviene solo a la hora de fijar una serie de parámetros. Al tratarse de maniobras en espacios interiores, puede darse el caso de existir obstáculos y limitaciones para llegar al punto final, por lo que la aeronave deberá superarlos y buscar alternativas para poder llegar a su meta.

Para la realización del dron se utilizará una placa con un microcontrolador programado en el lenguaje de código abierto Arduino, que, integrada junto a una gran variedad de elementos y componentes electrónicos, van a formar una aeronave de 4 motores, también llamado cuadricóptero.

Laburpena

Proiektuak espazio itxietan puntu batetik bestera mugitzeko gai den drone batean datza. Horretarako, gidaririk gabeko hegazkinaren osagaiak kudeatzeko gai den SBC batean garatuko da software bat.

Proiektu honen helburua quadrocopter autonomo bat lortzea da, eta gizakia soilik parametro batzuk ezartzen direnean hartzen du parte. Dronearen maniobrak barrualdeko espazioetan direnez, azken puntura iristeko oztopoak eta mugak egon daitezke, beraz, hegazkinak gainditu beharko ditu eta haren helburura heltzeko alternativa ezberdinak bilatu beharko ditu.

Dronea egiteko, Arduino kode irekian programatutako mikrokontrolagailu bat erabiliko da, non elementu eta osagai elektronikoekin batera integratuta, quadrocopter izena hartzen duen lau motoreko hegazkin bat osatuko du.

Abstract

The project consists in the implementation of a drone that is able to move autonomously from one point to another in interior spaces. To do this, a software will be developed in an SBC capable of managing the different elements that make up the unmanned aircraft.

The aim of this project is to achieve an autonomous quadcopter, in which the human factor intervenes only when setting a series of parameters. When dealing with maneuvers in interior spaces, there may be obstacles and limitations to reach the final point, so the aircraft must overcome them and look for alternatives to reach their goal.

A drone with a microcontroller programmed in the open-source Arduino language will be used to create the drone, in which, integrated with a variety of elements and electronic components, will form a 4-engine aircraft, also called quadrocopters.

Acrónimos

SBC, Single Board Computer

UAV, Unmanned Aerial Vehicle

RC, Radio Control

OS, Operative System

PWN, Pulse-Width Modulation

BEC, Battery Elimination Circuit

I2C, Inter-Integrated Circuit

BEC, Battery Elimination Circuit

GPS, Global Positioning System

ToF, Time of Flight

VCSEL, Vertical Cavity Surface-Emitting Laser

SPAD, Single Photon Avalanche Diode

EEPROM, Electrically Erasable Programmable Read-Only Memory

Índice

Índice de tablas

Índice de figuras

1. Introducción

Un vehículo aéreo no tripulado UAV (del inglés Unmanned Aerial Vehicle) o comúnmente conocido como dron, es una aeronave que vuela sin tripulación. En el presente documento se describe de forma detallada la realización de uno de estos aparatos, el cual debe ser capaz de desplazarse de un punto a otro previamente definido, de manera totalmente autónoma en un espacio interior. Este proyecto tiene como objetivo el desarrollo de un software propio, que integrándose con diferentes elementos Hardware, constituya un prototipo de aeronave exenta del factor humano.

Con el desarrollo de este proyecto se pretende obtener un software de control del dron totalmente fiable, y que, utilizando los datos obtenidos a través de distintos sensores integrados, sea capaz de modificar la conducta de los actuadores que posee la aeronave.

Figura 1. Dron cuadricóptero

Además de ser capaz de viajar de un punto a otro en espacios interiores, el cuadricóptero tendrá que esquivar los diferentes obstáculos y limitaciones que se encuentre durante su ruta hasta su destino. Por lo que, se debe saber en tiempo real cual es la posición del dron y buscar otro itinerario para poner rumbo al punto previamente fijado como final.

Durante el documento se irán viendo y explicando los beneficios y objetivos del proyecto. Por un lado, se realizará un estudio de las posibles alternativas existentes en este contexto. Por otro, se describirá en detalle la solución adoptada. Por último, se comentará el plan de trabajo seguido, los costes del proyecto desglosados y se obtendrán unas conclusiones.

2. Contexto

A diferencia de sus primeros usos como armamento militar, hoy en día el uso de los UAV se ha extendido a todo tipo de funciones. Hay constancia de más de 300 usos en diferentes ámbitos, lo que ha producido que su popularidad aumente en gran medida, denominando a los UAV comúnmente con el término dron, proveniente de la traducción de la palabra zángano (abeja macho) en inglés.

Aunque el uso de los drones destaca actualmente en campos como la agricultura de precisión [1], el medioambiente, infraestructuras, seguridad, emergencias, audiovisual, emergencias, logística y transporte entre otros, el ámbito que más cabe destacar y que aprecia la mayoría de las personas a primera vista es el ámbito comercial. Estas aeronaves se están comercializando como gadgets tecnológicos de bajo coste, las cuales están sustituyendo a los ya antiguos vehículos de radiocontrol utilizados con fines lúdicos.

Figura 2. Minidron utilizado de forma lúdica

Sin embargo, si se estudian con detenimiento las diferentes aplicaciones, no hay ninguna enfocada a espacios interiores [2]. Esto se debe en gran medida a la dificultad de orientación y localización que existe en lugares cerrados o que no están en el exterior, porque las tecnologías actuales se basan en diferentes satélites para realizar una triangulación y obtener la posición concreta

Debido a esto, se han empezado a ver algunas aplicaciones para localización en interiores, basándose en las señales de algunas tecnologías, tales como el Wifi o el 3G, pero sin llegar a obtenerse grandes resultados. El principal problema de estas aplicaciones es que lo que importa no es la señal que llega al dispositivo para guiarle, sino de qué forma le llega, ya que dependiendo de la potencia va a tener significados diferentes.

Finalmente, cabe destacar que los avances tecnológicos y la prospera investigación en el campo de los UAV, hacen posible que los drones estén en constante evolución llegando a límites insospechados. Además, poco a poco se está excluyendo el factor humano para el control de éstos, llegando a realizar trabajos de forma automatizada y precisa. En este proyecto se hace hincapié en este último aspecto, llevando al desarrollo de un software que implementará un dron con el objetivo de realizar una tarea sin necesidad de un piloto.

3. Alcance

El objetivo del proyecto es construir un dron y desarrollar un software que se ejecute en él. El software básicamente consiste en el manejo de la aeronave no tripulada de forma completamente autónoma. El movimiento del dron está pensado para que siempre se realice en espacios interiores cerrados, por lo que el usuario que utilice el dron deberá introducir previamente las dimensiones concretas del espacio interior, así como el punto de partida del dron y su destino final. Si durante el recorrido hacia su destino la aeronave encuentra algún obstáculo que le impida avanzar, se buscará una ruta alternativa por la que pueda llegar a su destino.

Debido a estas acciones que puede realizar el dron, el proyecto se divide en una serie de tareas o paquetes de trabajo a realizar. Cada paquete tiene unos objetivos parciales a cumplir. Las tareas del proyecto son los siguientes:

1. Construcción del dron y software de vuelo.
2. Generación del mapa y lógica de ruta.
3. Detección y evitación de obstáculos.

3.1. Construcción del dron y software de vuelo

Los objetivos parciales definidos para la tarea de construcción del dron y software de vuelo se muestran a continuación:

- Para empezar, se realizará un estudio de las partes y componentes electrónicos que puedan formar el dron. En este estudio se verán las alternativas disponibles, obteniendo las ventajas y desventajas de cada uno para elegir la más conveniente.
- Posteriormente, en función de los elementos seleccionados, se desarrollará el software que controle de forma conjunta todos los componentes para conseguir que el dron vuele de forma estable y controlada a través de un mando radiocontrol.
- Por último, cuando hardware y software se hayan integrado, se realizarán diferentes pruebas de vuelo en campo abierto, que una vez superadas, pasarán a realizarse en espacios cerrados.

3.2. Generación del mapa y lógica de ruta.

En la tarea de generación del mapa y la lógica de ruta del cuadricóptero, se han definido los siguientes objetivos parciales:

- Primeramente, se realizará un estudio de las diferentes formas existentes para realizar un mapeado del espacio interior en el que el dron deberá moverse. En él, se compararán las distintas alternativas y se elegirá la más conveniente.
- El siguiente objetivo es el desarrollo de una serie de funciones encargadas de generar un mapa en el que el dron sea capaz de saber la posición en la que se encuentra en tiempo real, además, de mandar las ordenes necesarias al software de vuelo del módulo anterior para que la aeronave se desplace por el aire.
- Finalmente, se realizarán pruebas en pantalla simulando el vuelo del dron entre un origen y un destino al azar en mapas de diferentes dimensiones. Cuando los resultados hayan sido satisfactorios, se realizarán pruebas con el dron en lugares cerrados.

3.3. Detección y evitación de obstáculos

En la última tarea, detección y evitación de obstáculos, los objetivos parciales son los siguientes:

- En primer lugar, realizar un estudio de los elementos disponibles en el mercado para poder utilizarlos sobre el hardware ya creado del dron. Ver las alternativas de cada uno y elegir el que mejor se acople con lo realizado anteriormente.
- En segundo lugar, realizar un estudio de las diferentes librerías software que se pueden utilizar con el hardware seleccionado. Tras realizar un análisis de las posibles alternativas se elige la que más se adapta a las necesidades del software anteriormente desarrollado.
- Después, realizar una función que sea capaz de tomar diferentes medidas de forma fiable, informando al dron de si un objeto está cerca de una distancia definida previamente.
- Posteriormente, si el dron detecta la existencia de algún obstáculo, se deberá de realizar una función que informe al módulo de mapeado y generación de ruta para que se genere una nueva ruta, ya que por esa hay algún impedimento.
- Finalmente, realizar diferentes pruebas en pantalla simulando la existencia de obstáculos en un mapa, consiguiendo que el dron llegue a su destino. Una vez se hayan superado las pruebas, realizar pruebas en un espacio interior cerrado con obstáculos entre el origen y el destino final.

4. Beneficios

Los drones abren un gran abanico de posibilidades, lo que genera una gran cantidad de beneficios, estos pueden ser técnicos, económicos y sociales.

4.1. Beneficios técnicos

Antiguamente, el control de Aeronaves No Tripuladas exigía tener unas nociones básicas sobre el manejo de las mismas. Nociones relacionadas con física y aerodinámica. Actualmente, gracias a las grandes mejoras en tecnología, se ha conseguido que cualquier persona sin conocimientos previos sobre aviación sea capaz de utilizarlos para sus propios fines. Todo esto, hace posible la utilización de los drones no solo como gadgets recreativos, sino como herramientas de trabajo que facilitan la realización de múltiples funciones y tareas.

Como se comenta en el contexto, los drones tienen más de 300 usos en diferentes ámbitos, destacando algunos como la agricultura, la industria, la seguridad y la vigilancia. Uno de los principales motivos por los que son utilizado para estos ámbitos, es que pueden llegar a sitios inaccesibles para cualquier persona, y gracias a la cámara que portan, se pueden obtener imágenes impresionantes a vista de pájaro. Además, los potentes motores y materiales utilizados en sus construcciones son capaces de soportar pesos de cierta consideración, lo que supone poder transportar objetos y poder añadir diferentes componentes que aumenten sus prestaciones.

Figura 3. Dron empleado en agricultura de precisión

Además, con este proyecto se pretende dar a conocer otra posible forma de localización y orientación en espacios interiores, ya que, no existe ninguna tecnología óptima para este tipo de aplicaciones.

4.2. Beneficios económicos

Como se ha visto en los beneficios técnicos, las aeronaves no tripuladas son herramientas utilizadas en múltiples trabajos. Esto repercute directamente en la parte económica de cualquier persona o empresa que utilice drones para cualquier tarea. En el caso concreto de una empresa

de seguridad y vigilancia, un dron es capaz de vigilar una zona concreta obteniendo imágenes desde una gran perspectiva en tiempo real, o en el caso de un uso destinado a agricultura, un dron es capaz de recorrer grandes distancias a gran velocidad portando un tanque de agua y unos difusores para regar una zona concreta, por ejemplo, un espacio interior como un invernadero. Un buen caso de uso en el que se empleen en espacios interiores seria en empresas relacionadas con la logística, utilizándose así para el transporte y distribución de enseres de diferentes tamaños, además de para realizar inventario utilizando cámaras.

Figura 4. Dron utilizado en logística

Desde el punto de vista económico, la utilización de aeronaves para estos trabajos concretos supone beneficios, ya que, si por el contrario estas tareas las realizarían personas, supondría un gran gasto en contratación de personal y en su correspondiente mano de obra. Por el contrario, contando con la ayuda de un dron, un solo empleado podría controlarlo y realizar el trabajo de una forma mucho más eficiente y menos costosa económicamente.

4.3. Beneficios sociales

Aparte del evidente beneficio económico que aportan los drones, también hay que tener en cuenta los beneficios sociales que se obtienen. Al desempeñar cualquier tarea utilizando un dron como herramienta, se están eliminando todos los riesgos asociados que se pueden sufrir si una persona la realizase. Es por esto, por la que una de las principales virtudes que tienen los drones, es la realización de trabajos de alto riesgo, como por ejemplo los que se realizan a gran altura o ponen en peligro la vida de las personas, tales como labores de salvamento y rescate.

Al utilizar vehículos aéreos para casi cualquier fin, la creación, distribución y venta de drones supondrá la generación de nuevas empresas relacionadas con este mundo. Lo que repercute en la generación de un gran número de puestos de trabajo, suponiendo un gran beneficio para la sociedad, en especial para los jóvenes.

5. Componentes del dron

Es importante conocer las partes que componen el dron, ya que más adelante van a tener que ser utilizadas en cierta medida con los módulos de software realizados. Por ello, se analizan cada una de ellas explicando cuál es su función principal en este proyecto [3]. Además, en la figura 5 se muestra un sencillo esquema en el que se ve claramente cómo se integran los diferentes componentes para poder comprender mejor cual es la ubicación de cada uno de ellos.

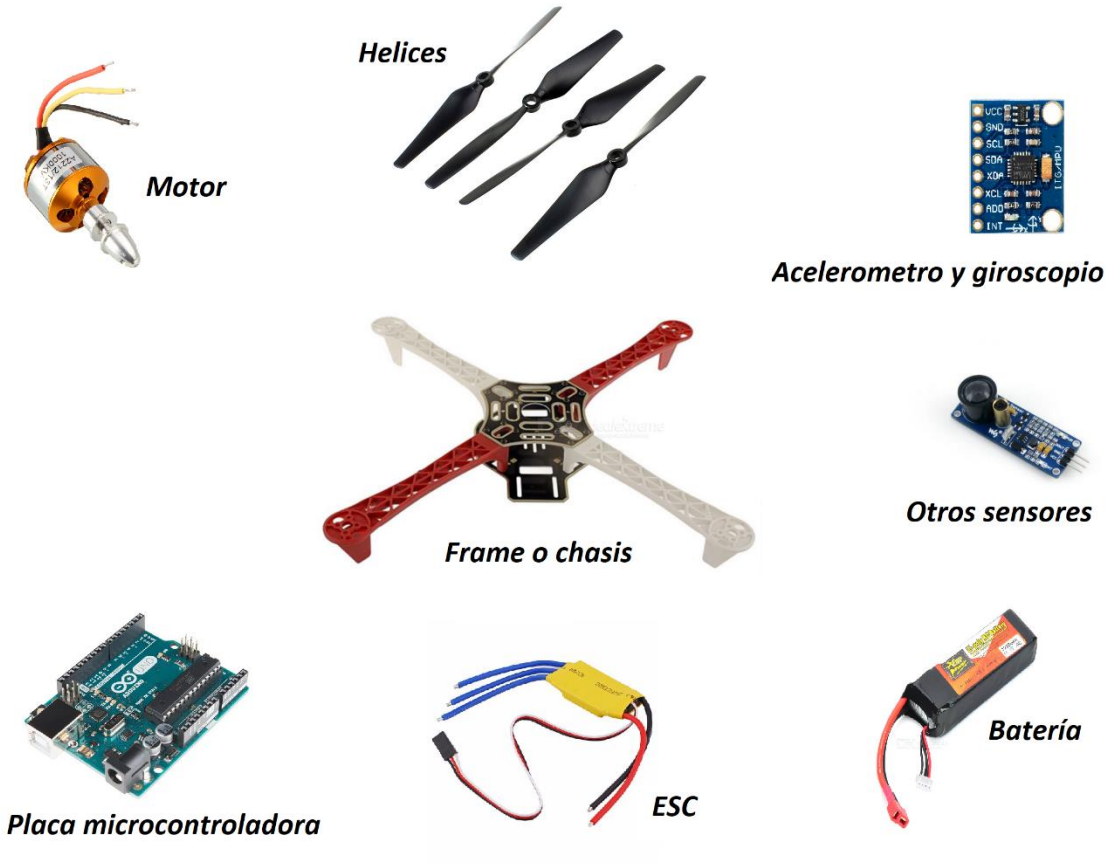


Figura 5. Esquema partes del dron

5.1.1. Chasis, estructura o frame

Es lo que soporta al resto de elementos, en aparatos profesionales será normalmente de carbono, muy ligero y resistente, combinando tubos y placas, con tornillería especial, y elementos anti-vibraciones, aunque se puede optar por materiales menos costosos y de peor calidad como pueden ser el plástico o la fibra de vidrio.

5.1.2. Placa microcontroladora

Es el cerebro del dron. Recoge datos de todo el sistema, velocidades de los motores, de los giróscopos y acelerómetros, los procesa y da las órdenes oportunas para mantener la estabilidad de la aeronave, además de transmitir convenientemente a cada motor las órdenes pertinentes. Estas placas están compuestas por una pequeña unidad central de procesamiento (CPU), una

memoria bastante escasa y puertos de entrada/salida a los que se puede agregar circuitería, sensores y módulos de comunicación externos a la placa original.

5.1.3. Motores

Los motores se encargan de hacer la misión principal del dron, girar las hélices para que pueda volar. Pueden ser de distintos tamaños, velocidades y potencias, así como trifásicos o bifásicos. Estos motores transforman la energía eléctrica en movimiento circular que pasa a transmitirse a las hélices de los drones, lo que causará un empuje que permitirá que la aeronave alce el vuelo.

5.1.4. ESCs o Variadores

Un variador, ESC (Electronic Speed Controller) o controlador de velocidad electrónico, es el circuito eléctrico que se encarga de variar la velocidad del motor, así mismo, variará su dirección y también podría actuar como un freno dinámico. Por lo que, convierte la energía de la batería de CC en trifásica CA, para impulsar a los motores y su objetivo es variar la velocidad de un motor eléctrico junto con el sentido de giro.

5.1.5. Hélices

Se colocan en los motores y su función es generar ráfagas de aire por la parte inferior del dron para conseguir elevarlo. Hay dos tipos de hélices en función del sentido de giro de éstas. En el caso concreto de un dron cuadricóptero, es decir, de cuatro hélices, habrá dos de cada tipo.

5.1.6. Acelerómetro y giroscopio

El acelerómetro es un sensor que va a medir la aceleración estática, y también la aceleración dinámica. La primera en el eje vertical, como la gravedad, y la segunda, en el eje horizontal, en el plano XY. Es utilizado para determinar la posición y la orientación del dron durante el vuelo. Por otro lado, está el giroscopio, encargado de medir los ángulos de ubicación del dron cuando este se encuentra en el aire. Generalmente este sensor, se ubica en la misma unidad, en la que se encuentra el acelerómetro de tres ejes, así trabajan en conjunto. Por una parte, el acelerómetro calculará la posición, por otra, el giroscopio calculará el ángulo en el que se encuentra. Esta unidad que integra los dos elementos se denomina Unidad de Medición Inercial o IMU (Inertial Measurement Unit).

5.1.7. Otros sensores

Aparte de los sensores mencionados, las aeronaves no tripuladas pueden incluir muchos otros. Algunos sensores típicos pueden ser módulos GPS para ubicar al dron en exteriores, módulos ultrasónicos para detectar y evitar obstáculos, barómetros para calcular presión, altura o temperatura, por último, magnetómetros que realizan las funciones de una brújula.

5.1.8. Baterías

Proporcionan la energía necesaria a todos los elementos que componen el dron. Las baterías, también llamadas Lipo, debido a que la componen polímeros de litio, pueden ser de varios tamaños y de varios voltajes por la diferente cantidad de celdas que utilizan al fabricarlas. Suelen ser uno de los puntos flacos de los drones debido a su gran peso, baja autonomía y largo tiempo

de carga, por lo que suele ser recomendable tener varios recambios. Actualmente, se está investigando bastante en esta materia, ya que aumentar la autonomía de los drones supondría eliminar muchos problemas.

6. Análisis de alternativas

El objetivo de este apartado es analizar las alternativas existentes a la hora de realizar el proyecto, escogiendo siempre la más adecuada en función de unos criterios específicos. Por ello, se van a ir analizando cada una de las alternativas disponibles para cada elemento que compone el dron.

6.1. Placa microcontroladora

Dentro del mercado de las placas microcontroladoras existe una gran variedad. Todas pensadas para unas funciones concretas, por lo que se van a analizar viendo cual es la más conveniente para este proyecto.

6.1.1. Arduino UNO R3

A diferencia de otras placas, Arduino no es una simple placa microcontroladora [4]. Como ellos mismo especifican, Arduino es una plataforma de prototipos electrónica de código abierto (open-source) basada en hardware y software flexibles y fáciles de usar. Está pensado para artistas, diseñadores, como hobby y para cualquiera interesado en crear objetos o entornos interactivos. Es decir, pone al alcance de cualquiera una potente herramienta que combina software y hardware.

Para programar una placa de Arduino se utiliza un IDE que está disponible de forma gratuita para múltiples plataformas. El lenguaje utilizado en la programación se llama Arduino también, y está basado en el famoso lenguaje de programación C++. El hardware de Arduino está basado en microcontroladores ATMEGA8 y ATMEGA168 de Atmel, fáciles de programar y de bajo coste, junto con un gran número de entradas/salidas digitales y analógicas.

El modelo Arduino UNO R3 es el buque insignia de Arduino, lo que la convierte en la más vendida de esta familia de placas. Por todo esto, es muy fácil y accesible encontrar infinitos proyectos de todo tipo para este modelo concreto, desde proyectos simples como encender un led, hasta sistemas electrónicos complejos con un gran número de periféricos.

6.1.2. STM32 ARM Coretex-M3

STM32 es una familia de circuitos integrados de microcontroladores de 32 bits de STMicroelectronics [5]. Los chips STM32 se agrupan en series relacionadas que se basan en el mismo procesador ARM de 32 bits, como Cortex-M7F, Cortex-M4F, Cortex-M3, Cortex-M0 + o Cortex-M0.

Al igual que las placa Arduino, las placas STM32 cuentan con una gran cantidad de entradas/ salidas analógicas y digitales, y pueden ser programadas con el mismo IDE creado por Arduino previa instalación de unas librerías de compatibilidad. Tienen un coste similar a las placas Arduino, con la ventaja de ser más rápidas.

6.1.3. Raspberry Pi 3B

Raspberry Pi es un ordenador de placa reducida, desarrollado en Reino Unido por la fundación con el mismo nombre [6]. Su objetivo principal es el de estimular la enseñanza de ciencias de computación en las escuelas, por lo que tiene un tamaño y un coste bastante reducidos.

A diferencia de las otras dos alternativas mencionadas antes, Raspberry no está pensando para mezclar software con hardware, sino que se podría decir que es más similar a un ordenador convencional, pero con unas menores especificaciones para realizar tareas que no requieran de grandes procesadores. Aun así, cuenta con varios pines para usar de entradas/salidas digitales.

Utiliza un procesador con arquitectura ARM capaz de trabajar a frecuencias mucho mayores que otras placas, es por ello por lo que se asemeja más a un PC que a una placa pensada para proyectos de electrónica. Dentro de ella es posible utilizar múltiples sistemas operativos, desde distribuciones basadas en UNIX como su OS oficial llamado Raspbian, hasta un sistema operativo Windows pensado para desarrolladores. La Raspberry Pi Foundation prioriza y promueve el uso del lenguaje de programación Python, aunque pueden usarse otros diferentes.

6.1.4. Criterios de selección y ponderación de las alternativas

Los criterios que se van a tener en cuenta para elegir la placa microcontroladora más adecuada para el dron serán los siguientes:

- Tamaño y peso. Priorizando que la placa sea lo más pequeña y ligera posible para ayudar al dron a ser más ligero y menos aparatoso. Ponderación del 10%.
- Implementación software y hardware. Teniendo en cuenta que para realizar el dron va a ser necesario utilizar una mezcla de ambos. Ponderación del 20%.
- Coste. Ponderación del 10%.
- Sencillez de programación. Priorizando lenguajes que se hayan estudiado en el grado como C o Java. Ponderación del 20%.
- Rapidez de la CPU. Ponderación del 20%.
- Implementación en la sociedad. Priorizando encontrar trabajos y proyectos que puedan ayudar y servir de referencia. Ponderación del 20%.

Tabla 1. Análisis de alternativas: placa microcontroladora

Placa	Tamaño y peso 10%	Implementación SW y HW 20%	Coste 10%	Sencillez de programación 20%	CPU 20%	Implementación en la sociedad 20%	Total
ARDUINO	7	10	8	9	7	9	8,5
STM32	8	8	10	8	8	5	7,6
RaspberryPi	6	7	7	7	9	9	7,7

En vista de los resultados, la placa microcontroladora que se va a utilizar para el montaje del dron va a ser la Arduino UNO R3, ya que es la que mejor se adapta al proyecto. Esta placa tiene las siguientes especificaciones:

- Microcontrolador ATmega328.
- Voltaje de entrada 7-12V.
- 14 pines digitales de I/O (6 salidas PWM).
- 6 entradas analógicas.
- 32k de memoria Flash.
- Reloj de 16MHz de velocidad.

6.2. Motores

Actualmente, casi todos los drones utilizan motores eléctricos, pero existen dos tipos claramente diferenciados de estos, los cuales aportan ventajas y desventajas en función de cual vaya a ser su uso.

6.2.1. Motores con escobillas o Brushed

Es el tipo de motor tradicional que se ha utilizado en RC [7]. Su funcionamiento se basa en un conjunto de bobinas giratorias que actúan como un electroimán con 2 polos. El inducido o rotor es un electroimán, y el imán de campo es un imán permanente. El conmutador es un dispositivo de anillo partido envuelto alrededor del eje que contacta físicamente con los cepillos, que están conectados a los polos opuestos de la fuente de alimentación. Los cepillos hacen que el conmutador de polaridad inversa al imán permanente, que a su vez provoca la armadura gire. La rotación de la dirección, en sentido horario y/o en sentido contrario, se puede invertir fácilmente mediante la inversión de la polaridad de los cepillos, es decir, la inversión de los cables de la batería.

Por lo que, algunas conclusiones que se obtienen son:

- Requiere más mantenimiento que el brushless.
- Precio económico.
- Problemas de disipación de calor dada su estructura.
- Más resistente a ambiente hostiles gracias a su diseño simple.

6.2.2. Motores sin escobillas o Brushless

Es la última tecnología desarrollada en motores eléctricos RC [8]. Utiliza un rotor externo de imán permanente, bobinas de tres fases de conducción, uno o más dispositivos de efecto Hall para detectar la posición del rotor, y la electrónica de accionamiento asociados. Las bobinas se activan, una fase después de la otra, por la electrónica de accionamiento como desencadenamiento por las señales de los sensores de efecto Hall, que actúan como motores síncronos trifásicos que contienen su propia electrónica de accionamiento de frecuencia variable.

Por todo esto, estos motores tienen las siguientes ventajas y desventajas:

- Alcanzar velocidades mucho mayores que un brushed.
- Control más complicado y complejo que un brushed.
- Precio más elevado.

6.2.3. Criterios de selección y ponderación de las alternativas

Se van a tener en cuenta los siguientes criterios para decantarse por uno de los dos tipos de motores eléctricos:

- Coste. Ponderación del 25%.
- Mantenimiento. Ponderación del 25%.
- Velocidad y potencia. Priorizando que es preferible unos motores y veloces, lo que proporcionara al dron mayor fuerza para transportar objetos y recorrer distancias en menos tiempo. Ponderación del 20%.
- Resistencia. Ponderación del 10%.
- Control y manejo. Ponderación del 20%.

Tabla 2. Análisis de alternativas: motores

Motor	Coste 25%	Mantenimiento 25%	Velocidad 20%	Resistencia 10%	Control y manejo 20%	Total
Brushed	8	6	5	9	8	7
Brushless	6	9	9	7	6	7,45

Como se puede comprobar, la mejor alternativa es el uso de motores Brushless, ya que aparte de ser los más adecuados en función de los parámetros seleccionados, son los más fáciles de encontrar y conseguir. Para el dron se van a utilizar 4 motores A2212 Brushless con las siguientes características:

- KV: 1000 rpm/V de máxima
- Eficiencia: 80%
- Corriente máxima (eficiencia): 4-10A (>75%)
- Capacidad actual: 12A/ 60s
- N.º de celdas: 2-3 Li-Po
- Dimensiones del motor: 27.5* 30mm
- Diámetro del eje: 3.17mm

6.3. Baterías

Las baterías de los drones son una de las partes más delicadas del mismo y requieren unos cuidados que pocas veces se tienen en cuenta.

Se utilizan baterías como sistema de aporte energético porque son fáciles de recargar, y mediante su corriente es posible alimentar todos los circuitos de drones de mediano y pequeño tamaño. El mercado está plagado de baterías LiPo, que son las que ofrecen una mejor relación entre la energía que son capaces de almacenar y el peso y volumen que ocupan, dentro de unos costes accesibles para el gran público.

6.3.1. Criterios de selección

Hay que tener en cuenta una serie de factores a la hora de elegir las baterías que se quieran utilizar en el dron. Son los siguientes:

- Capacidad, normalmente medida en mAh. Priorizando baterías con una capacidad entre 2000 y 3000, ya es la capacidad adecuada para drones de tamaño normal como el de este proyecto. Cuanta más capacidad, mayor tamaño.
- Celdas. Teniendo en cuenta que las baterías LiPo están compuestas por una o varias celdas, conectadas en serie, cada una de ellas con un voltaje de 3,7 voltios en condiciones de almacenamiento. Por norma general se conectan en serie, de forma que la tensión nominal total siempre será un múltiplo de 3,7. Por lo que hay que tener en cuenta la tensión que se va a tener en el circuito electrónico que compone el dron.
- Capacidad de descarga. Otro aspecto fundamental de estas baterías es su capacidad de entregar energía en momentos concretos y de forma segura, es lo que se conoce como tasa de descarga, y se identifica por un número, seguido de una C. Habitualmente 30C o 40C es lo utilizado. Por ejemplo, una batería de 6000 miliamperios, si fuese de tipo 1C, sería capaz de entregar esa energía de 6 amperios durante una hora. Sin embargo, si fuese de tipo 20C, entregaría 120 amperios, pero en 3 minutos.

En vista de los criterios descritos y del esquema electrónico visto en otros apartados, hay que decantarse por una batería LiPo de 3 celdas y 11.1 V. Además, deberá de ser de más de 20C de capacidad de descarga, y de entre 2200 mAh y 3800 mAh. Por todo esto, se va a seleccionar una batería LiPo de 11.1V, 2200mAh de capacidad y 30C de capacidad de descarga.

6.4. Variadores o ESCs

Los variadores en general son controladores de Modulación por Ancho de Puntos (PWM) para controlar motores eléctricos. El receptor del aeromodelo manda una señal PWM al ESC con variaciones de 1 a 2 milisegundos. En la primera el motor estar parado, a 1.5 milisegundos el motor funcionará a la mitad de su potencia y a 2ms el motor funcionará al 100%. Es importante mencionar que los ESC para motores con escobillas no son compatibles con motores brushless. Una manera fácil de identificarlos es que los variadores para motores con escobillas llevan dos cables, mientras que los otros llevan 3.

Los variadores para motores brushless crean una corriente alterna trifásica a partir de corriente continua proveniente de la batería. Es aquí donde entra en escena los 3 cables que llevan los variadores para motores sin escobillas. Uno de los polos genera un pequeño voltaje proporcional a la velocidad de giro del motor conocido como fuerza electromotriz. Este voltaje le sirve al ESC para determinar como de rápido y en qué dirección gira el motor en cualquier momento. Con

esta información el ESC es capad de averiguar cómo mandar la corriente a los electroimanes del motor para que este gire. [9]

6.4.1. Criterios de selección

Los variadores están categorizados en función de la máxima corriente que pueden soportar. A mayor corriente de aguante de un ESC, mayor será su precio. Lo recomendable a la hora de elegir un variador es optar por uno que se encuentre por encima de la demanda del motor. Una corriente alta podrá dañar el ESC en pleno vuelo y hacer que la aeronave caiga en picado. Así mismo, también hay que tener en cuenta la batería Lipo seleccionada. Además, estos variadores suelen incluir un sistema llamado BEC (Battery Elimination Circuit), que transforma el voltaje que tiene la batería y lo transforma al clásico voltaje de casi todos los componentes, 5V. La placa microcontroladora ya suministra un voltaje de 5V, por lo que, para este proyecto no va a ser necesario utilizar este sistema

Por todas estas razones, se va a seleccionar un ESC de 30 amperios con las siguientes características:

- Cont. corriente 30A
- Voltaje de entrada: 2-4 S LiPo/5-12 células de Ni-MH/Ni-cd
- BEC: 5 V
- Tensión de corte: 4 V

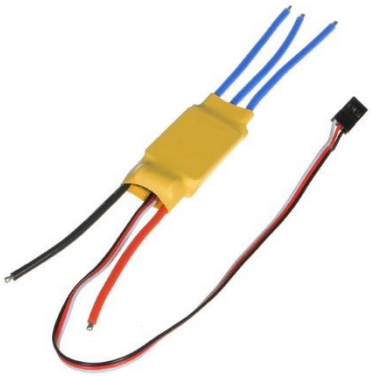


Figura 6. ESC de 30 A

6.5. IMU

Se trata de la unidad electrónica que se encarga de realizar las mediciones que determinan las actuaciones del dron y, por tanto, le permite conocer cuál es su posición relativa con respecto al estado inmediatamente anterior. Por medio de la IMU el aeromodelo conoce sus aceleraciones, sus desplazamientos y sus posiciones en cada momento, mediante dispositivos conocidos como acelerómetros, giróscopos o magnetómetros.

En el caso de un dron, esto es utilizado para saber la actual tasa de aceleración usando uno o más acelerómetros, y detecta los cambios en atributos rotacionales tales como cabeceo (pitch), alabeo (roll) y guiñada (yaw) usando uno o más giróscopos.

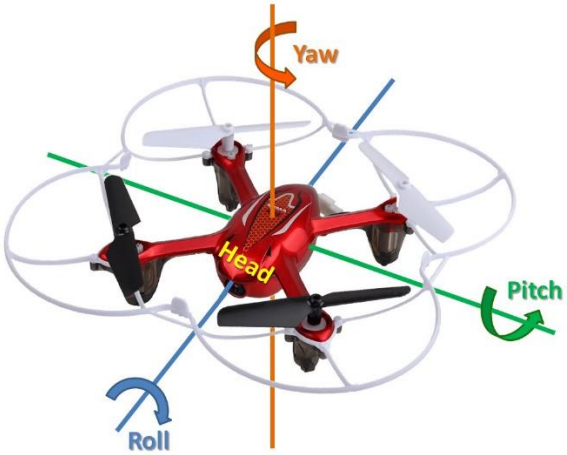


Figura 7. Ejes de movimiento de un dron

Las unidades de medición inercial son usadas en sistemas de guía inercial instaladas en vehículos. Actualmente casi todas las naves de superficie, comerciales o militares poseen una. La mayor parte de los aviones también están equipados con IMU. Además, también son usadas en naves aéreas y espaciales, con el propósito de informar de las medidas inerciales a un piloto.

La IMU se puede implementar en drones de dos formas, utilizando los elementos que la forman, tales como giroscopio, acelerómetro y magnetómetro, en una misma unidad que integra todos, o de la segunda forma, utilizando estos elementos por separado.

6.5.1. Criterios de selección

Como es obvio, es mucho más fácil y cómodo de utilizar una IMU que implemente todos los elementos en una misma unidad, de esta forma, además de ahorrar espacio en el dron, los datos de los elementos que se integren juntos viajarán juntos utilizando el mismo protocolo de comunicación.

Por esto, y por ser una de las IMUs por excelencia en los drones debido a su bajo coste y alta fiabilidad, se va a utilizar el MPU-6050. Se trata de una unidad de medida inercial de 6DOF (6 Degrees Of Freedom). Esto significa que lleva un acelerómetro y un giroscopio, ambos de 3 ejes (3+3 = 6DOF) [10]. Hay IMUs de 9DOF, en este caso también llevan un magnetómetro, pero para este proyecto en cuestión no va a ser necesario un magnetómetro que haga de brújula, ya que el dron se va a colocar en una posición determina al realizar su ruta.

El MPU-6050 opera con 3.3 voltios, aunque algunas versiones (como la que se va a utilizar) llevan un regulador que permite conectarla a 5V. Para comunicarse con la placa microcontroladora utiliza el protocolo de comunicación I2C. Este protocolo llamado Inter Integrated Circuits bus, o IIC, se convirtió en un estándar de facto en la industria desde su invención y se basa en los siguientes puntos:

- Protocolo de dos hilos de control, uno para transmitir los datos, SDA y otro, el reloj asíncrono que indica cuando leer los datos SCL. Mas GND y 5V (cuando se requiera).
- Cada dispositivo conectado al bus I2C y cada uno tiene su dirección exclusiva, de 7 bits,
- Uno de estos componentes, debe actuar como maestro, es decir controla el reloj.

- No se requiere una velocidad de reloj estricta, ya que es el maestro quien controla el reloj.

En este caso concreto, el MPU6050 será el esclavo y la placa microcontroladora será el maestro.

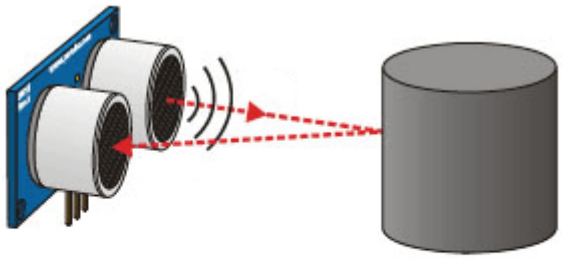
6.6. Sensores medidores de distancia

Para que el dron sea capaz de percibir los obstáculos que se encuentra durante su ruta, se pueden utilizar diferentes tipos de sensores. Para esta tarea, se suelen utilizar sensores de ultrasonidos, si no se utilizan estos, se pueden emplear también sensores de laser infrarrojos.

6.6.1. Sensor ultrasónico HC-SR04

Un sensor de ultrasonidos se basa simplemente en medir el tiempo entre el envío y la recepción de un pulso sonoro. Por tanto, es posible obtener la distancia a partir del tiempo entre la emisión y recepción de este pulso.

El sensor HC-SR04 es un sensor de distancia de bajo costo, su uso es muy frecuente en la robótica (lo que hace que se utilice mucho en proyectos de Arduino), utiliza transductores de ultrasonido para detectar objetos. Su funcionamiento consiste en emitir un sonido ultrasónico por uno de sus transductores, y esperar que el sonido rebote de algún objeto presente, el eco es captador por el segundo transductor. La distancia es proporcional al tiempo que tarda en llegar el eco.



Tiempo = 2 * (Distancia / Velocidad)

Distancia = Tiempo · Velocidad / 2

Figura 8. Funcionamiento de sensor ultrasónico

6.6.2. Sensor laser infrarrojo VL53L0X

El VL53L0X es un sensor de distancia infrarrojo láser de última generación, que puede emplearse junto con una placa como Arduino para medir distancias de 50 a 200 cm de forma precisa. Es capaz de operar incluso con elevada luz ambiental infrarroja, e incorpora un sistema de compensación de la medición que lo permite hacer funcionar incluso detrás de un cristal protector.

El VL53L0X es un sensor ToF (Time of flight). Su funcionamiento consiste en enviar un pulso láser de luz infrarroja y medir el tiempo necesario en el haz en volver al sensor. El integrado incorpora un emisor laser 940nm VCSEL (Vertical Cavity Surface-Emitting Laser), un detector SPAD (Single

Photon Avalanche Diodes) y la electrónica interna (denominada FlightSenseTM) que realiza los cálculos necesarios para obtener la distancia.

Es uno de los mejores sensores de distancia disponibles. Tiene una precisión superior que los sensores de ultrasonidos, y no se ve alterado por las condiciones del ambiente como los ecos o la reflectancia de los objetos.

Sin embargo, el ángulo de medición es relativamente estrecho. Esto es una ventaja en la mayoría de las circunstancias, donde se desea leer la distancia justo en frente del sensor. Aunque también puede ser una desventaja, por ejemplo, a la hora de detectar obstáculos en movimiento, sobre todo si no son planos y tienen formas variables. Además, su coste es bastante más elevado que el de un sensor de ultrasonidos.

6.6.3. Criterios de selección y ponderación de las alternativas

Para la elección del sensor utilizado para medir distancias se han tomado en cuenta los siguientes parámetros:

- Coste. Ponderación del 25%.
- Ángulo de medición. Ponderación del 25%.
- Distancia de detección. Ponderación del 20 %.
- Implementación con Arduino. Ponderación del 15%.
- Precisión. Ponderación del 10%.

Tabla 3. Análisis de alternativas: sensores medidores de distancia

Sensor	Coste 25%	Angulo de medición 25%	Distancia de detección 20%	Implementación con Arduino 15%	Precisión 10%	Total
	9	8	8	9	5	7,7
	7	6	7	8	9	6,75

En vista de los resultados, se va a utilizar el sensor ultrasónico HC-SR04, que tiene las siguientes características técnicas:

- Número de pines:
  - VCC: Alimentación +5V (4.5V min – 5.5V max)
  - TRIG: Trigger entrada (input) del sensor (TTL)

  - ECHO: Echo salida (output) del Sensor (TTL)
  - GND
- Corriente de reposo: < 2mA
- Corriente de trabajo: 15mA
- Ángulo de medición: 30º
- Ángulo de medición efectivo: < 15º
- Detección de 2cm a 400cm o 1" a 13 pies (Sirve a más de 4m, pero el fabricante no garantiza una buena medición).
- Precisión entre los 3mm o 0.3cm.
- Dimensiones: 45mm x 20mm x 15mm
- Frecuencia de trabajo: 40KHz

6.7. Medidor de altura

Si la aeronave no tripulada debe de volar de forma completamente autónoma, es necesaria la utilización de algún tipo de sensor conectado a esta que sea capaz de medir la altura a la que llega a elevarse. Esta función se puede realizar de diversas maneras como se verá a continuación.

6.7.1. Criterios de selección

Al igual que en el subapartado anterior, una de las alternativas a tener en cuenta es la de utilizar un sensor ultrasónico situado en la parte inferior del dron, dirigiendo las ondas que este emite contra el suelo y rebotando para calcular la distancia. La otra alternativa vista anteriormente, el sensor laser infrarrojo, es otra opción que podría servir.

Sin embargo, ya que las dos alternativas mencionadas no son del todo acertadas debido a su corto/medio alcance y ángulos de mediciones, para el cálculo de altura en vuelos de drones se utilizan principalmente dos sensores completamente diferentes.

El primero es un módulo GPS, que es capaz de calcular las coordenadas exactas del dron y su altura. Este sistema se sirve de 24 a 32 satélites y utiliza la trilateración para conseguir estos parámetros, pero para ello es necesario una buena cobertura que se consigue con una buena visión de estos satélites, por lo que su uso se reduce a aplicaciones al aire libre, quedando descartado como posible alternativa para su utilización en espacios interiores.

El segundo sensor, uno de los más utilizados debido a su buen funcionamiento y menor coste que un módulo GPS, es un barómetro. Como es bien sabido, un barómetro es utilizado para medir la presión. Con este parámetro se puede calcular la altura entre dos puntos midiendo la presión atmosférica en cada punto y utilizando una simple fórmula física. Además, también es capaz de calcular la temperatura.

Por estas razones, se empleará el módulo GY-63, que contiene el barómetro MS5611. Este módulo tiene las siguientes características y se puede apreciar en la figura 9:

Figura 9. Módulo GY-63

- Protocolo de comunicación: I2C
- Tensión de alimentación: 3-5 V (contiene un regulador)
- Rango de operación: de 100 a 1200 mbar y de -40 a +85 °C
- Corriente: 1mA
- Velocidad de conversión: 1 ms

7. Descripción de la solución

Como se ha observado en apartados anteriores, el proyecto cuenta con 3 paquetes claramente diferenciados:

1. Construcción del dron y software de vuelo
2. Generación del mapa y lógica de ruta
3. Detección y evitación de obstáculos

7.1. Construcción del dron y software de vuelo

Antes de empezar con el desarrollo del software que se ejecutará en el dron, es necesario revisar y comprobar que se tienen todos los elementos hardware que van a componer la parte física del cuadricóptero. Una vez se tenga clara cuál va a ser la estructura que se va a seguir, se realizará un esquema electrónico que muestre y especifique como han de ser las conexiones de todos los dispositivos electrónicos que se van a emplear e integrar entre sí. El esquema electrónico de referencia en este proyecto se encuentra en el primer anexo del documento, *Esquemático de Arduino UNO y del dron*.

Después de seguir este esquema, estarán todo los componentes y elementos perfectamente conectados para empezar a desarrollar un software que permita manejarlos y enviarles la información necesaria para realizar las acciones deseadas para conseguir un vuelo estable y controlado, pero antes hay que entender un poco como funciona Arduino.

Primeramente, cabe destacar que la programación de Arduino se define con dos funciones principales [4] al igual que por ejemplo la programación en C tiene una función principal denominada *main()* que es la que se ejecuta, Arduino consta de la función de configuración *setup()* y la función *loop().* La primera se usa para inicializar variables, modos de pin, comenzar a usar bibliotecas, etc. Esta función solo se ejecutará una vez, después de cada encendido o reinicio de la placa Arduino. La segunda, como su propio nombre indica, es una función que se ejecuta en bucle continuamente mientras la placa este encendida y después de haber terminado la función *setup().* El programa de formato *.ino* donde se encuentran mínimo estas dos funciones se denomina sketch, y tiene la estructura de la figura 10.

```
1  void setup() {
2    // put your setup code here, to run once:
3
4  }
5
6  void loop() {
7    // put your main code here, to run repeatedly:
8
9  }
```

Figura 10. Estructura básica de un sketch de Arduino

7.1.1. Programa de instalación y calibración

Una vez el dron se ha montado, hay que realizar algunos chequeos y calibraciones de los variados componentes para asegurarse de que posteriormente van a comportase de forma correcta. Para realizar este proceso, se va a tomar como referencia el proyecto del ingeniero

holandés Joob Brokking [11] en el que se carga un programa llamado YMFC-3D_V2_setup, el cual sirve para calibrar y memorizar una serie de datos en la memoria EEPROM de la placa microcontroladora Arduino UNO, los cuales posteriormente utilizará el software de vuelo como referencia. Cuando este programa corra en la placa a través del IDE de Arduino, hay que abrir el Serial (monitor serie) y seguir los siguientes pasos:

1. Comprobar que el reloj del protocolo I2C está configurado a 400 kHz.
2. Comprobar que existen señales de recepción de un mando para controlar el dron. Además, exigirá que todos los controles estén en su posición central para poder medir la señal en ese valor concreto.
3. A continuación, pedirá realizar diferentes movimientos con los controles para ir midiendo los valores extremos a los que pueden llegar los dos sticks del mando. Luego irá pidiendo que se simulen los movimientos de yaw, pitch y roll con los sticks para saber si los ejes están invertidos respecto a lo que suele ser la configuración por defecto de un mando de estas características.
4. Después, utilizando el protocolo de comunicación serie I2C, buscará entre unos cuantos modelos concretos, la IMU MPU-6050. Una vez la encuentre, pedirá realizar unos movimientos del dron (cogiéndolo con las manos) con unos grados concretos, simulando los futuros movimientos de yaw, pitch y roll.
5. Finalmente, teniendo en cuenta todos los datos obtenidos durante esta calibración, los guardará en la memoria EEPROM, la cual no es volátil, es decir, se mantiene a pesar de dejar de alimentar la placa Arduino.

Gracias a estos valores almacenados en memoria, el dron va a tener como referencia unos valores sobre cómo actúa el giroscopio y acelerómetro al realizar ciertos movimientos, lo que va a ser de gran ayuda después para conseguir un vuelo mucho más seguro y fluido. Además, al guardar los valores a los que el mando emisor puede enviar las señales al receptor conectado a la placa microcontroladora, este va a saber cómo traducirlos posteriormente y cuáles son esos valores máximos y mínimos.

7.1.2. PID

Lo primero que hay que tener en cuenta a la hora de usar cualquier sistema de control, que en el caso concreto de una aeronave son los motores, es el controlador PID [11]. Un controlador PID es un mecanismo de control que calcula la desviación o error entre un valor medido y el valor que se quiere obtener para aplicar una acción correctora. Básicamente, esta función les dice a los motores como de rápido tienen que girar, porque en realidad, es la única manera de conseguir estabilidad en un cuadricóptero. PID hace referencia a Proportional-Integral-Derivative (Proporcional-Integral-Derivativo). P depende del error presente, I de la acumulación de errores pasados y D es la que predice futuros errores basados en rato de cambio actual. En la práctica, cada uno de estos 3 parámetros afecta a las características de vuelo y estabilidad del dron.

Los valores de P, I y D aparecen de forma numérica en el software de la controladora de vuelo, los cuales, nos dan posibilidad de ajustarlos. Estos valores son realmente el coeficiente de los 3 algoritmos mencionados anteriormente. Cada uno de los coeficientes cambia la influencia de los algoritmos en la orden final para los motores. Como ya se ha explicado, el dron puede girar

en 3 ejes (yaw, pitch y roll), por lo que va a existir un conjunto de coeficientes PID para cada uno, es decir, 9 diferentes valores ajustables en total.

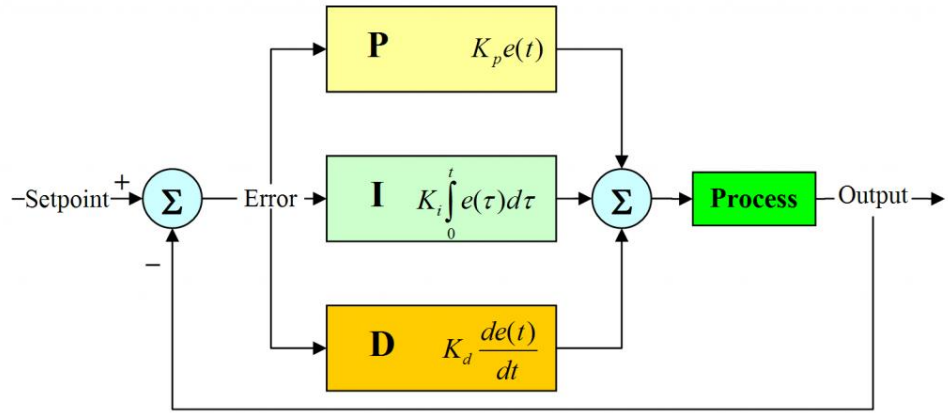


Figura 11. Diagrama de bloques de un controlador PID

Como se aprecia en la figura 11, los valores PID están correlacionados y el proceso en un bucle. Si cambia por ejemplo el valor de D seguramente haya que modificar también un poco el valor de P y lo mismo pasará con I. Para comprender que hace cada uno, se van a explicar más en detalle.

Proporcional

Es el valor más importante e indica el nivel de corrección necesario a aplicar, dado que es el encargado de la estabilidad y control del vuelo. Mientras más grande sea P, más intentará estabilizarse la placa microcontroladora, pero si el valor es excesivo, se pueden provocar oscilaciones debido a la gran sensibilidad que experimenta la placa.

Integral

La I indica la velocidad con la que se repite la acción proporcional, lo que influye directamente en la precisión de la posición angular del dron. Lo que hará I a grandes rasgos, es hacer más progresivo el movimiento de retorno a la estabilidad que dicta P. Si el valor de I es demasiado alto, los motores oscilarán lentamente, pareciendo que el dron se mueve de forma lenta y rígida. Si por el contrario es bajo, la aeronave no podrá mantener el ángulo de rotación y parecerá que derrapa en el aire.

Derivativo

La ganancia en D funciona como un amortiguador y reduce la posibles sobre correcciones de P. El objetivo de D es conseguir que el dron vuele de una manera más suave y minimice oscilaciones al cambiar la fuerza aplicada a corregir el error de posición. Un valor muy alto de D puede inducir a vibraciones debido a la amplificación del ruido. Sin embargo, un valor de D muy bajo hará el cuadricóptero perezoso y asemejarse a un comportamiento similar a tener la P a un nivel bajo.

Para realizar este sistema de control dentro del dron, se va a tomar como base otro programa de Joop Brokking [12], en este caso concreto en el sketch *YMFC-3D_V2_flight_controller.* Como se ha mencionado antes, cada eje de giro tiene su propio PID. Se va a detallar el cálculo del PID

del eje de giro roll (alabeo) como ejemplo, los otros dos ejes sería exactamente iguales creando las mismas variables terminadas en *_yaw* y *_pitch* en lugar de en *_roll.*

Para empezar, va a ser necesario declarar unas variables globales, con unos valores concretos para la ganancia de P, de la I, y de la D. Estos valores vienen definidos por Joob Brokking [11] pero pueden ir modificándose y ajustándose al ver que el comportamiento del dron en el aire va mejorando o empeorando. Después, se define otra variable global que define el PID máxima para el roll, ya que sino el valor que se puede obtener puede ser muy alto llevando al dron a perder el control de los motores. Teniendo estos valores se van a leer los valores recibidos del giroscopio y los de que el receptor recibe del mando, para después restarlos y calcular el error que puede haber entre ellos. Este error se utilizará para regular el valor de la I y compararlo con el valor máximo definido antes para el PID, de tal forma que la I puede tener ese valor como máximo. Finalmente, a través de una fórmula que simula el procedimiento de la figura 11, y teniendo en cuenta los valores de la P, la I y la D, se calcula el PID total de salida de roll.

```
//////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
//PID gain and limit settings
//////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
float pid_p_gain_roll = 1.2;              //Gain setting for the roll P-controller
float pid_i_gain_roll = 0.02;             //Gain setting for the roll I-controller
float pid_d_gain_roll = 17;               //Gain setting for the roll D-controller
int pid_max_roll = 400;                   //Maximum output of the PID-controller

//////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
//Declaring global variables
//////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
float pid_error_temp;
float pid_i_mem_roll, pid_roll_setpoint, gyro_roll_input, pid_output_roll, pid_last_roll_d_error;

void calculate_pid(){

//Roll calculations
  pid_error_temp = gyro_roll_input - pid_roll_setpoint;
  pid_i_mem_roll += pid_i_gain_roll * pid_error_temp;
  if(pid_i_mem_roll > pid_max_roll)
            pid_i_mem_roll = pid_max_roll;
  else if(pid_i_mem_roll < pid_max_roll * -1)
            pid_i_mem_roll = pid_max_roll * -1;

  pid_output_roll = pid_p_gain_roll * pid_error_temp + pid_i_mem_roll + pid_d_gain_roll * (pid_error_temp -
pid_last_roll_d_error);
  if(pid_output_roll > pid_max_roll)
            pid_output_roll = pid_max_roll;
  else if(pid_output_roll < pid_max_roll * -1)
            pid_output_roll = pid_max_roll * -1;

  pid_last_roll_d_error = pid_error_temp;
  }
```

7.2. Generación del mapa y lógica de ruta

La propuesta principal de este proyecto es el movimiento del dron en espacios interiores, lo que supone un gran reto. Esto se debe a que hoy en día no existe ninguna tecnología específica relacionada con el posicionamiento en lugares cerrados e interiores. Todos los sistemas de posicionamiento conocidos están diseñados para exteriores, incluido el más famoso, el GPS.

Debido a este principal problema, para desarrollar un sistema de bajo coste y alta eficiencia, la única alternativa es definir el mapa previamente, ya que ir descubriendo el mapa a medida que el dron se mueve por el espacio conllevaría mucho tiempo que se refleja en un gran consumo de batería, la mayor debilidad de estas aeronaves.

Por lo tanto, para generar el mapa se va a recurrir a dos conceptos básicos de las matemáticas, las matrices y los ejes de coordenadas cartesianas en un plano. El primero de los conceptos, las matrices, van a representar los diferentes espacios interiores por los que el cuadricóptero debe moverse. Cada una de las posiciones de las que consta la matriz va a representar un punto concreto de la habitación o espacio cerrado que se quiera representar. Para ello, ajustando la velocidad de los motores del dron, entre cada punto (representado por la fila y la columna a la que pertenecen) y sus respectivos contiguos, va a existir una distancia real de 1 metro de largo o ancho aproximadamente.

$$\mathbf{A} = \begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{m1} & a_{m2} & \cdots & a_{mn} \end{bmatrix}$$

Figura 12. Matriz A de dimensión MxN

Por ejemplo, en la figura 12 entre el elemento a_{11} y el a_{12} va a haber 1 metro de distancia. Lo mismo ocurrirá entre el elemento a_{11} y el a_{21}, por lo cual, las distancias se miden en metros con números enteros, y la distancia mínima que se puede recorrer es de un metro.

El segundo concepto, los ejes de coordenadas cartesianas en un plano, entran en juego a la hora de ayudar al usuario a visualizar el mapa del lugar interior donde se quiere realizar el vuelo. Ya que por medio de un plano cartesianos clásico, con unos ejes X e Y, es fácil establecer un origen y un destino de forma clara y bastante visual. Dado como ejemplo el espacio cerrado de la figura 13, que tiene unas medidas de 6x10 metros, si se coloca visto desde arriba sobre uno de estos ejes, se podría elegir de forma numérica el origen (1,4) y el destino del dron (8,1).

Figura 13. Espacio interior con vista superior sobre eje cartesiano

Combinando las dos ideas expuestas, se obtiene que cada punto (x,y) del eje de coordenadas, pertenece a una posición (fila,columna) de la matriz. El único detalle que hay que tener en cuenta, es que el dron se puede mover justo por el eje de abscisas (eje X) o por el de ordenadas (eje Y), por lo que ahí también van a existir elementos de la matriz que van a representar los puntos (x,0) o (0,y). Para entender mejor el concepto observar la figura 14:

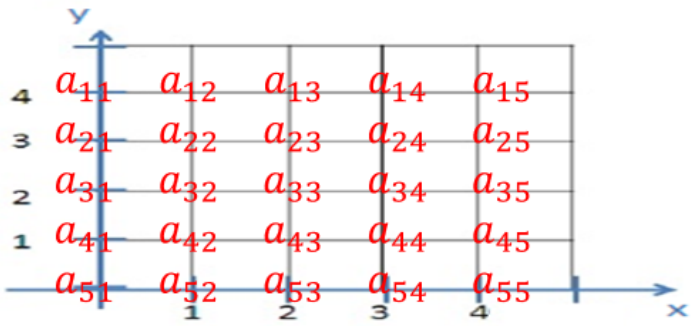


Figura 14. Matriz superpuesta sobre eje cartesiano

Las matrices, que representan al fin y al cabo los ejes, se forman a partir del ancho y alto del espacio interior. Estos datos son predefinidos por el usuario antes de realizar todo el mapeado. Junto con el alto y ancho del supuesto espacio por el que el dron debe moverse, también hay que definir un origen y un destino que el usuario puede definir a su gusto siempre y cuando sea un valor valido dentro de las dimensiones del mapa especificado. Todo esto se realiza en la función *crearMapa()*, a la que se llama desde el *setup()* para crear el mapa desde el primer momento de configuración.

```
////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
//DEFINICIÓN de filas y columnas
////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
#define filas 20
#define columnas 20
```

```
unsigned int mapa[filas][columnas];

////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
//DEFINICIÓN de origen sobre eje cartesiano (origenX,origenY)
////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
#define origenX 0
#define origenY 0

////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
//DEFINICIÓN de destino sobre eje cartesiano (destinoX,destinoY)
////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
#define destinoX 5
#define destinoY 5
void setup() {
  crearMapa();
}


void crearMapa() {

  for (unsigned int i = 0; i < filas; i++) {
    for (unsigned int j = 0; j < columnas; j++) {
      mapa[i][j] = ((filas - 1) - i) * columnas + j;
    }
  }
}
```

Como se aprecia en el código, se crea un array bidimensional con las medidas especificadas, de tipo *unsigned int* ya que una matriz no puede tener valores negativos. En cada posición del array se va colocando un número que va aumentando de izquierda a derecha y de abajo a arriba, para poder realizar después la comprobación de si el destino es el mismo que el origen. Por lo tanto, si se muestra el resultado por pantalla utilizando el Serial de Arduino por ejemplo para una matriz de 3x3, se obtendría el resultado de la figura 15:

```
6 7 8
3 4 5
0 1 2
```

Figura 15. Matriz 3x3 en sketch

Cuando el mapa está creado en la memoria de la placa microcontroladora, el dron debe colocarse en el punto real del espacio con la cabeza siempre mirando hacia el eje Y positivo, esta parte es fácilmente identificable debido a que es de color rojo, la parte trasera es de color blanco. Acto seguido, éste debe de trazar una ruta moviéndose de forma autónoma por el terreno hasta llegar a su destino, manteniendo constantemente la misma altura durante el vuelo. Es en este momento cuando entra en escena el barómetro, gracias a él, el software va a ser capaz de tomar varias medidas y calcular la altura a la que se está desplazando la aeronave, imprimiendo potencia a los motores cuando sea necesario para que siempre vuele a una altura preestablecida.

Para calcular la altura con el barómetro, se han creado dos funciones. La primera función es *calcularPresiónInicial(),* que calcula la presión en el punto donde se coloca el dron antes de iniciar el camino. Utilizando esta primera presión como referencia, se toma la presión en los diferentes puntos del aire donde el dron va a estar, y por la diferencia de ambas se puede

calcular la altura a la que el dron se desplaza, esto se realiza en la segunda función denominada *calcularAltura().* Se ha empleado la librería MS5611 [13] para utilizar la función que devuelve el valor de la presión en bares.

Figura 16. Throttle de un dron

Si la altura es menor de metro y medio, se sumará más valor a la variable *throttle*, la cual es la encargada de dar potencia a los motores en el eje vertical, es decir, que el dron suba o baje como se aprecia en la figura 16. Si el dron sube demasiado, se bajará el throttle para evitar una altura elevada. Esta función se empleará al principio para que el dron despegue y durante la ruta con la misión de que la altura del vuelo sea lo más estable posible.

```
#include <MS5611.h>
#define medidas 5

MS5611 baro;
int32_t pressure;
double presion, altura, suma, pini, ayuda;

void calcularPresionInicial() {
  baro = MS5611();
  baro.begin();
  suma = 0;
  for (int32_t i = 0; i < medidas; i++)
  {
    pressure = baro.getPressure();
    ayuda = (double)pressure;
    suma += ayuda;
  }
  pini = (suma / medidas);
}

void calcularAltura() {
  suma = 0;
  for (int32_t i = 0; i < medidas; i++)
  {
    pressure = baro.getPressure();
    ayuda = (double)pressure;
```

```
    suma += ayuda;
  }
  presion = (suma / medidas);

  altura = 44330.0 * (1 - pow(presion / pini, 1 / 5.255));
  if(altura < 1.5)
    throttle+=100;
  else if(altura > 2.5)
    throttle-=100;
}
```

Si el dron ha cogido la altura correspondiente, el programa a través de una función que implementa un algoritmo de ruta, se va moviendo por el aire (aunque él crea que se está moviendo por los diferentes puntos de la matriz), desde el punto de partida hasta el destino realizan una serie de movimientos. Estos movimientos realizados son siempre iguales y pueden ser de 4 formas, hacia delante, hacia atrás, hacia la izquierda y hacia la derecha. En definitiva, el dron estaría recibiendo una señal de throttle para mantener la altura, un movimiento de pitch (cabecero) positivo o negativo para moverse hacia delante y hacia atrás, por último, un movimiento de roll (alabeo) positivo o negativo para desplazarse hacia la izquierda o hacia la derecha.

Lo comentado se realiza en la función *calcularRuta(),* que lo que hace es ir moviendo el dron por las diferentes posiciones de la matriz. Para conseguir que el dron se mueva en línea recta en el eje X y en el eje Y hay que "engañar "a la IMU MPU-6050, esto quiere decir, hay que conseguir modificar en cierta forma los valores reales que el giroscopio toma, para que al calcular el PID se realice la corrección y el dron se mueva hacia el lado que necesite para estabilizar el dron. Esto se consigue modificando las variables gyro_roll_input y gyro_pitch_input, las cuales pueden ser entendidas y verse en el anexo MPU-6050. Al modificar estos dos valores en +30 o -30 según convenga, se le está diciendo al giroscopio que esta inclinado esos grados de más o menos lo que conlleva a desplazarse hacia un lado o hacia otro, consiguiendo el movimiento deseado. Este valor de 30 es debido a que al realizar la corrección el dron se desplaza un metro aproximadamente, lo que es una medida perfecta y pensada para que cuadre con la matriz que define el mapa, ya que esta separa sus elementos con un metro entre cada uno.

```
void calcularRuta() {

  unsigned int origen = mapa[filas - 1 - origenY][origenX];
  unsigned int destino = mapa[filas - 1 - destinoY][destinoX];
  unsigned int ox, oy, dx, dy;

  ox = origenX;
  oy = filas - 1 - origenY;
  dx = destinoX;
  dy = filas - 1 - destinoY;

  while (origen != destino) {

    calcularAltura();

    //Moverse hacia delante
    if (oy > dy) {
     gyro_pitch_input = gyro_pitch_input + 30;
```

```
      oy--;
    }
    //Moverse hacia atras
    if (oy < dy) {
      gyro_pitch_input = gyro_pitch_input - 30;
      oy++;
    }
    //Moverse hacia la derecha
    if (ox < dx) {
      gyro_roll_input = gyro_roll_input - 30;
      ox++;
    }
    //Moverse hacia la izquierda
    if (ox > ox) {
      gyro_roll_input = gyro_roll_input + 30;
      ox--;
    }

    origen = mapa[oy][ox];

  }
}
```

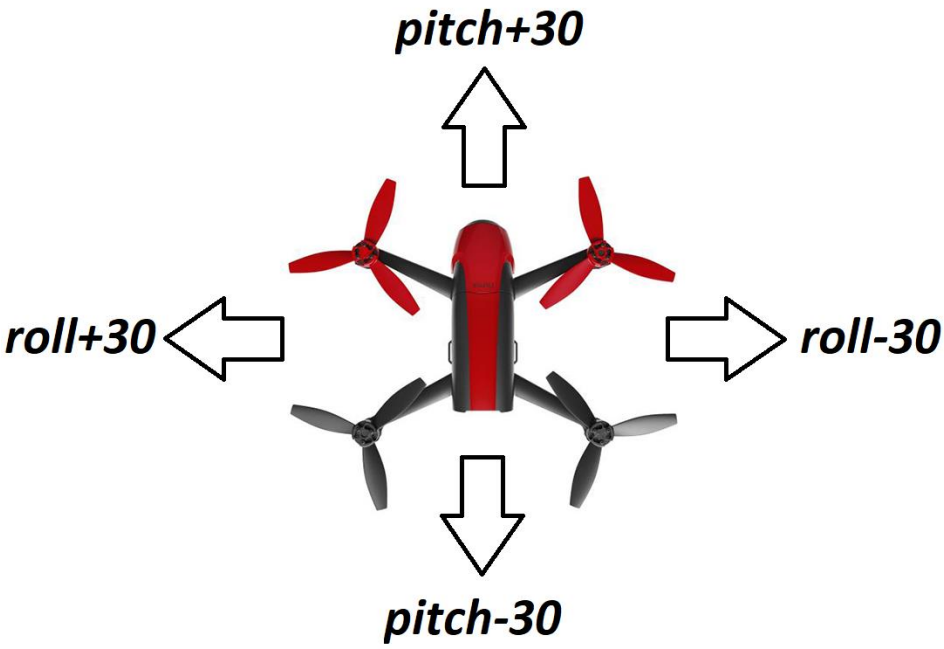


Figura 17. Lógica de movimiento sobre el mapa

Una vez se ha llegado al destino, el barómetro irá bajando poco a poco la altura predefinida mientras deja de dar potencia a los motores de forma progresiva para no realizar un aterrizaje brusco y dañar algún componente. Esto se realizará en el siguiente módulo.

7.3. Detección y evitación de obstáculos

Como era de esperar, el cuadricóptero no puede desplazarse por el aire siguiendo su trayectoria sin tener en cuenta lo que existe a su alrededor. Durante el viaje desde su origen hasta su destino por el aire, debe de percibir los posibles obstáculos que pueden existir en su ruta, ya que pueden suponer un peligro para el dron, pudiendo llegar a producirse una colisión. Es por ello por lo que se crea una tercera tarea pensado para detectar y evitar los obstáculos de todo tipo, tales como paredes, objetos, estructuras, etc.

Ahora, el principal protagonista es el sensor ultrasónico HC-SR04. Gracias a él, se va a poder mandar un pulso ultrasónico, denominado así porque tiene una frecuencia superior a 20 kHz (frecuencia que el oído humano puede percibir), que rebota sobre los objetos que el dron encuentre a su alrededor. Al recibir este pulso rebotado, la placa microcontroladora va a poder calcular el tiempo que ha tardado desde que lo ha mandado hasta que le ha llegado, por lo tanto, teniendo ese tiempo y sabiendo la velocidad a la que viaja el pulso, es posible calcular la distancia de forma simple.

Este sensor está muy implementando en múltiples aplicaciones de todo tipo, lo que hace que existan infinidad de librerías de programación que desarrollan algunas funciones de gran utilidad. Después de haber realizado un estudio de estas librerías, la que más funciones útiles tiene para este proyecto en concreto es la librería NewPing [14]. En concreto tiene una función llamada *ping_cm()* que devuelve la medida que tome el sensor en centímetros. Para usarla hay que definir previamente un objeto de la clase NewPing indicando los pines del trigger, del echo y la máxima distancia a medir (en caso de superar esta distancia el valor obtenido es cero).

Como se quiere tener el control total del espacio por el que el dron se desplaza, se van a utilizar 4 sensores, por lo que habrá que definir 4 objetos NewPing. Para ahorrar pines de la placa Arduino, todos los sensores van a compartir el pin de *trigger*, ya que es el encargado de disparar el pulso que choca con el objeto. Sin embargo, cada sensor debe de tener su propio pin de *echo*, que mide el pulso que vuelve después de impactar contra el objeto, porque si todos tienen el mismo, la placa no sabría interpretar a cuál de todos los pines pertenece la señal que recibe por ese pin. [15]

Tal y como se ha visto en el análisis de alternativas, el peor punto que tienen estos sensores es su precisión. Para solventar este problema se van a tomar varias medidas en cada sensor y se hará la media para cada uno con el fin de obtener un resultado certero. Por eso, se va a crear una estructura llamada *Sensores,* la cual va a constar de un array de enteros para guardar las distancias realizadas, también otro entero que guarde la media obtenida del array, y por último, un boolean que indique si el valor está cerca o no de un objeto en función de una distancia que se determine, en este caso esa distancia es de 1 metro. Este último valor que indica si el objeto está cerca o no es el más importante, ya que es el que va a determinar si el dron va a avanzar o no en su ruta.

Los campos de esta estructura se tienen que rellenar de alguna forma, por eso se han hecho 3 funciones diferentes. La función *addMedidas()* va ir rellenando el array de distancias para cada sensor, dejando un tiempo de 6ms entre cada medida realizada por sensor para que no haya

posibles interferencias entre los pulsos de echo. Una vez se tiene el array completado, se llama a la función *mediaDistancias()* que realiza la media de cada uno de los 4 arrays, siempre y cuando haya al menos más de 3 medidas correctas. Esto quiero decir que si hay dos medidas erróneas (con valor cero) o no hay objetos a la vista en dos o más de las medidas, la media total va a ser cero. Por último, la función *comprobarDistancias()* es la encargada de comprobar las medias obtenidas de la función anterior y compararla con el valor establecido en *DistanciaCerca,* que va a ser 1 metro. Si el valor de la media es distinto de cero (valor erróneo) y menor que 1 metro, significa que el dron está próximo a un objeto, por lo que pondrá la variable cerca de la estructura a true.

```
#include <NewPing.h>

/*Inicialización de los pines de los sensores. Librería "NewPing"
  NewPing NAME(Trigger, Echo, MAXDIST);
  El valor de MAXDIST es la distancia máxima que mide la librería.
  Si algún Echo devuelve un valor mayor de dicha distancia, se descarta automáticamente*/

NewPing sonar0(13, 2, 200); //Sensor delantero
NewPing sonar1(13, 3, 200); //Sensor trasero
NewPing sonar2(13, 15, 200); //Sensor derecho
NewPing sonar3(13, 16, 200); //Sensor izquiero

#define NDistancias     5    //Numero de distancias a medir para luego hacer la media
#define DistanciaCerca  100  //Distancia a la que empieza a actuar el control

struct Sensores {
  uint16_t Distancias[NDistancias]  = {0};
  uint16_t MediaDistancias          = 0;
  bool Cerca                        = false;
};

//Se inician las variables de cada sensor
#define NSensores 4

Sensores Sensor[NSensores];

void loop() {
  addMedidas();
  mediaDistancias();
  comprobarDistancias();
}

void addMedidas(){
  for (uint8_t i = 0; i < NDistancias; i++) {

 Sensor[0].Distancias[i] = sonar0.ping_cm();
  delay(6);

  Sensor[1].Distancias[i] = sonar1.ping_cm();
  delay(6);

  Sensor[2].Distancias[i] = sonar2.ping_cm();
  delay(6);

  Sensor[3].Distancias[i] = sonar3.ping_cm();
  delay(6);
 }
}

void mediaDistancias(){
    for (uint8_t i = 0; i <4; i++) {
      int Total   = 0;
```

```
    uint8_t Num = 0;
    for (uint8_t j = 0; j <NDistancias; j++){

      if (Sensor[i].Distancias[j] != 0  && Sensor[i].Distancias[j] < 300) {
       Total += Sensor[i].Distancias[j];
       Num += 1;
       }
       }
       if (Num > 3) {
         Sensor[i].MediaDistancias = Total / Num;
       } else {
         Sensor[i].MediaDistancias = 0;
       }
        }
}

void comprobarDistancias() {
  for (uint8_t i = 0; i < NSensores; i++) {
    if (Sensor[i].MediaDistancias != 0 && Sensor[i].MediaDistancias < DistanciaCerca) {
      Sensor[i].Cerca = true;
    } else {
      Sensor[i].Cerca = false;
    }
  }
}
```

Estas 3 funciones se realizan constantemente en bucle mediante la función *loop()* predefinida en Arduino, lo que supone tener un control total del espacio que rodea al dron. Pero ahora es necesario implementar esta función junto con la lógica de ruta, con el objetivo de conseguir que, si se percibe un objeto durante la ruta, el dron no continúe por ese camino y busque otra ruta alternativa. Para ellos se van a realizar algunos retoques en la función *calcularRuta()*. En cada una de las condiciones que chequean las coordenadas, se va a incluir otra condición que va a observar si la variable booleana de los sensores es falsa o verdadera, en caso de ser verdadera va a significar que tiene un objeto cerca del sensor, no avanzando por la zona que detecta y siguiendo por otro camino.

Aunque no es normal que ocurra, puede haber situaciones en las que el dron se quede sin salida porque encuentra caminos cerrados, por lo que se utilizada una variable que va almacenando la ruta que sigue, para en caso de quedarse encerrado en un punto y evitar que la función entre en un bucle, vuelva al origen inicial y aplique una lógica de ruta un poco diferente. Esta lógica de ruta es igual que la primera que se intenta, salvo que la inicial intenta avanza primero hacia delante o hacia atrás, es decir, desplazándose sobre el eje Y primero, aplicando un movimiento de pitch sobre el dron. Sin embargo, la segunda lógica de ruta intenta desplazarse primero hacia los lados, es decir, sobre el eje X, realizando un movimiento de roll sobre el dron. Al cambiar esta lógica el inicio es completamente diferente y el camino a seguir es otro desde el primer momento, lo que el bucle anterior se evitará.

```
void calcularRuta() {

 unsigned int origen = mapa[filas - 1 - origenY][origenX];
 unsigned int destino = mapa[filas - 1 - destinoY][destinoX];
 unsigned int ox, oy, dx, dy;
 int vueltas = 0;
 String s = "";
 ox = origenX;
 oy = filas - 1 - origenY;
```

```
  dx = destinoX;
  dy = filas - 1 - destinoY;

  while (origen != destino && vueltas < filas * columnas) {

    calcularAltura();
  addMedidas();
  mediaDistancias();
  comprobarDistancias();

    vueltas++;


    //Moverse hacia delante
    if (oy > dy && Sensor[0].Cerca == false) {
     gyro_pitch_input = gyro_pitch_input + 30;
      s += "a";
      oy--;
    }
    //Moverse hacia atras
    if (oy < dy && Sensor[1].Cerca == false) {
     gyro_pitch_input = gyro_pitch_input - 30;
      s += "r";
      oy++;
   }
    //Moverse hacia la derecha
    if (ox < dx && Sensor[2].Cerca == false) {
      gyro_roll_input = gyro_roll_input - 30;
      s += "d";
      ox++;
   }
    //Moverse hacia la izquierda
    if (ox > ox && Sensor[3].Cerca == false) {
      gyro_roll_input = gyro_roll_input + 30;
      s += "i";
      ox--;
   }
    origen = mapa[oy][ox];

  }

//Si el dron se queda sin salida
  if (vueltas == filas * columnas) {

    for (int i = s.length(); i >= 0; i--) {
      calcularAltura();
      if (s.charAt(i) == 'a') {
        //Retroceder
        gyro_pitch_input = gyro_pitch_input - 30;
        oy++;
      }
      if (s.charAt(i) == 'r') {
        //Avanzar
        gyro_pitch_input = gyro_pitch_input + 30;
        oy--;
      }
      if (s.charAt(i) == 'd') {
        //Izquierda
        gyro_roll_input = gyro_roll_input + 30;
        ox++;
      }
      if (s.charAt(i) == 'i') {
        //Derecha
        gyro_roll_input = gyro_roll_input - 30;
        ox--;
      }
    }
```

```
 origen = mapa[oy][ox];

 while (origen != destino ) {
  calcularAltura();
  addMedidas();
    mediaDistancias();
    comprobarDistancias();

   //Moverse hacia la derecha
   if (ox < dx && Sensor[2].Cerca == false) {
     gyro_roll_input = gyro_roll_input - 30;
     s += "d";
     ox++;
  }
   //Moverse hacia la izquierda
   if (ox > ox && Sensor[3].Cerca == false) {
     gyro_roll_input = gyro_roll_input + 30;
     s += "i";
     ox--;
   }
   //Moverse hacia delante
   if (oy > dy && Sensor[0].Cerca == false) {
    gyro_pitch_input = gyro_pitch_input + 30;
     s += "a";
     oy--;
  }
   //Moverse hacia atras
   if (oy < dy && Sensor[1].Cerca == false) {
    gyro_pitch_input = gyro_pitch_input - 30;
     s += "r";
     oy++;
  }

    origen = mapa[oy][ox];

  }
  }

 if(origen==destino)
   while(throttle>1000)
       throttle=throttle-100;
}
```

Como se aprecia al final del código, cuando el origen y el destino son iguales, el throttle se va disminuyendo de 100 en 100 hasta llegar a 1000 que es cuando los motores del dron están al ralentí, lo que quiere decir que giran, pero con poca potencia como para que el dron no despegue.

Para comprender mejor el funcionamiento de este módulo, se supone un ejemplo. Se define un mapa de 10 metros de alto y 10 metros de ancho. Después se establecen las coordenadas (2,2) como origen y (9,8) como destino. Se coloca la cabeza del dron (parte roja de la estructura) mirando hacia el eje Y positivo. Acto seguido, el dron empezara a realizar su ruta por el mapa, pero en el punto (5,5) detecta un obstáculo por lo que sigue hacia delante en vez de realizar el giro que debería de realizar si no llega a existir el obstáculo. En la coordenada (6,8) ocurre exactamente lo mismo. Al final, el dron llega a su destino.

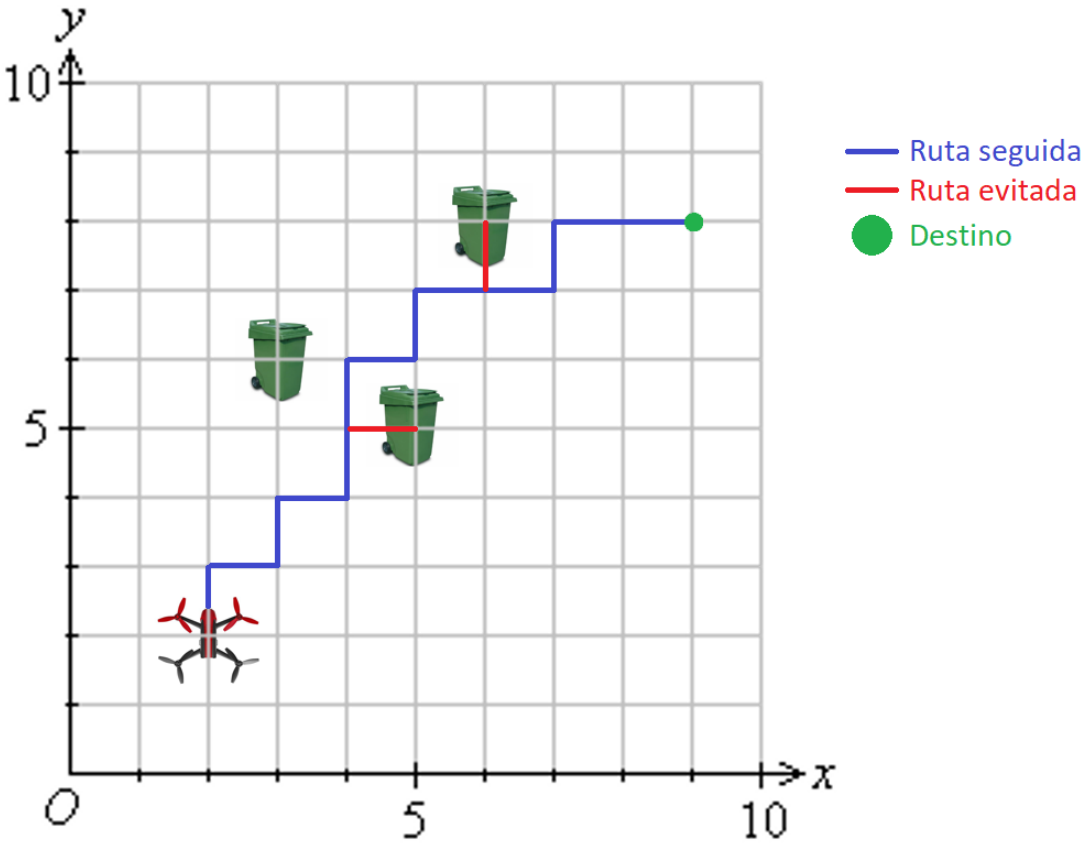


Figura 18. Ejemplo de ruta seguida por el dron

Este procedimiento queda reflejado en la figura 18, donde ser representa todo el trabajo realizado en las 3 partes explicadas anteriormente, resumiendo todo de una forma muy clara y visual.

8. Descripción de tareas

En este apartado se exponen las tareas realizadas a lo largo del proyecto, descomponiéndose en paquetes. Para finalizar este apartado, se muestra el diagrama de Gantt que ayuda a la visualización de lo comentado anteriormente.

8.1. Descomposición de las tareas

Tabla 4. Descomposición de las tareas con nombre y duraciones

EDT	Nombre de la tarea	Duración	F. Inicio	F. Final
PT-1	**Preparación del proyecto**	**50d**	**19/12/17**	**26/02/18**
T-1.1	**Elección del tema**	**4d**	**19/12/17**	**22/12/17**
T-1.2	Sondeo inicial sobre drones	10d	08/01/18	19/01/18
T-1.3	Investigación y análisis sobre proyectos existentes relacionados	5d	22/01/18	26/01/18
T-1.4	Análisis de componentes utilizados	8d	29/01/18	07/02/18
T-1.5	Definición del alcance del proyecto mediante objetivos generales y parciales	8d	08/02/18	19/02/18
T-1.6	Realizar planificación de tareas	4d	20/02/18	23/02/18
H-1	Inicio del proyecto	~0	26/02/18	26/02/18
PT-2	**Desarrollo del proyecto**	**88d**	**27/02/18**	**28/06/18**
T-2.1	**Construcción del dron y software de vuelo**	**22d**	**27/02/18**	**28/03/18**
T-2.1.1	Estudio de posibles partes y componentes del dron	4d	27/02/18	02/03/18
T-2.1.2	Desarrollo del software de vuelo con los componentes elegidos	10d	05/03/18	16/03/18
T-2.1.3	Pruebas de vuelo con mando en campo abierto	4d	19/03/18	22/03/18
T-2.1.4	Pruebas de vuelo con mando en espacios cerrados	4d	23/03/18	28/03/18
T-2.2	**Generación del mapa y lógica de ruta**	**28d**	**29/03/18**	**07/05/18**
T-2.2.1	Estudio para realizar un correcto mapeado de un espacio cerrado	3d	29/03/18	02/04/18
T-2.2.2	Desarrollo de función para generar mapa y posición del dron	5d	03/04/18	09/04/18
T-2.2.3	Desarrollo de función que gestione la lógica de ruta	6d	10/04/18	17/04/18
T-2.2.4	Integración de las nuevas funciones con el software de vuelo	7d	18/04/18	26/04/18
T-2.2.5	Pruebas en pantalla del mapeado y lógica de ruta	3d	27/04/18	01/05/18
T-2.2.6	Pruebas de vuelo del mapeado en espacios interiores	4d	02/05/18	07/05/18
T-2.3	**Detección y evitación de obstáculos**	**38d**	**08/05/18**	**28/06/18**
T-2.3.1	Estudio y elección de elementos hardware que complementen el dron	3d	08/05/18	10/05/18

T-2.3.2	Estudio y elección de librerías software que gestionen el hardware escogido	3d	11/05/18	15/05/18
T-2.3.3	Desarrollo de función que realice medidas contra diferentes objetos	9d	16/05/18	28/05/18
T-2.3.4	Integración con el mapeado y lógica de ruta	11d	29/05/18	12/06/18
T-2.3.5	Pruebas en pantalla simulando obstáculos en el mapa	5d	13/06/18	19/06/18
T-2.3.6	Pruebas de vuelo en un espacio interior con obstáculos	7d	20/06/18	28/06/18
PT-3	**Redacción de la memoria y documentación**	**17d**	**29/06/18**	**23/07/18**
T-3.1	**Elección de tipo de memoria adecuada**	**2d**	**29/06/18**	**02/07/18**
T-3.2	**Redacción de la memoria**	**15d**	**03/07/18**	**23/07/18**
H-2	Fin del proyecto	~0	24/07/18	24/07/18

8.2. Diagrama de Gantt

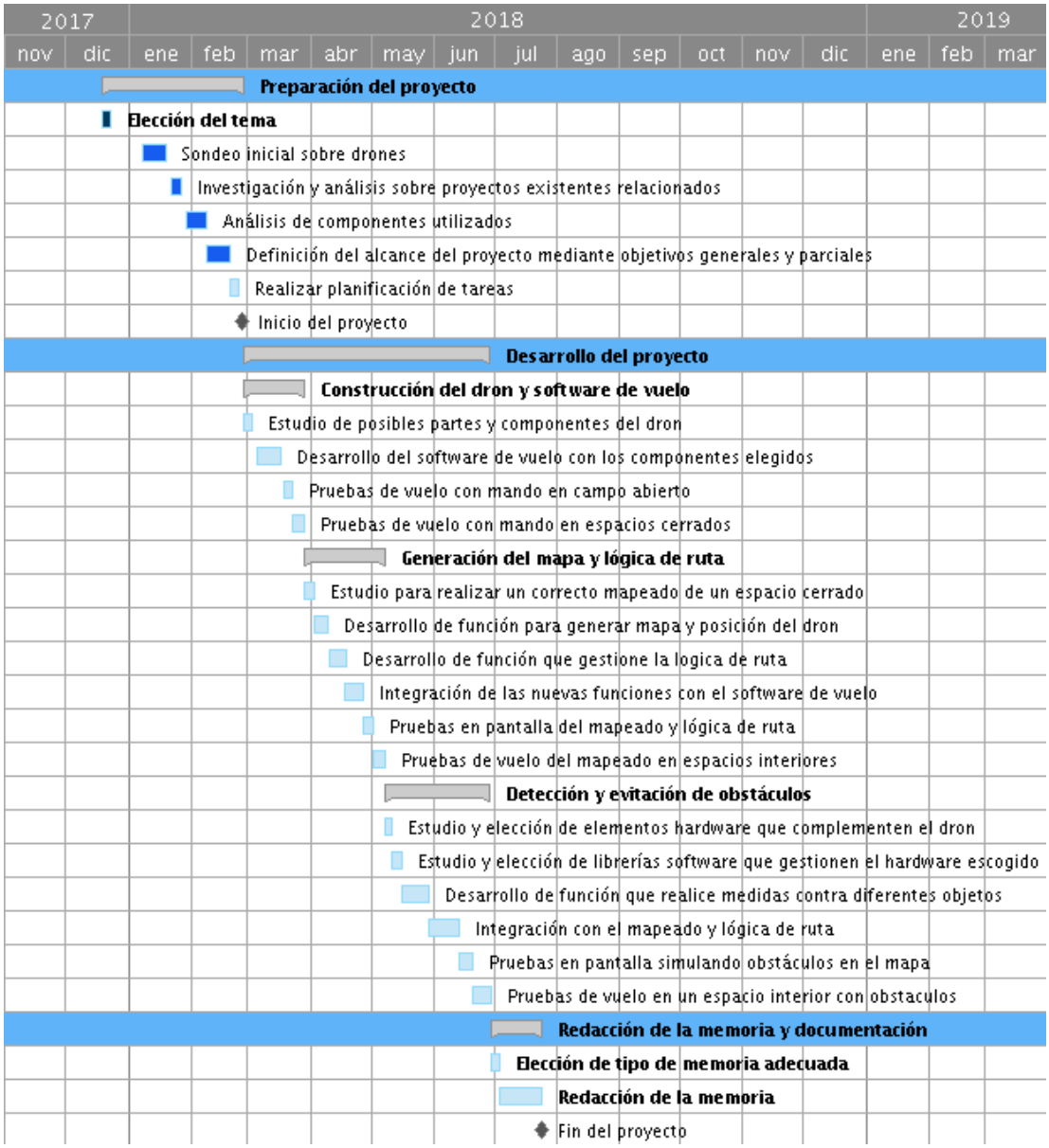


Figura 19. Diagrama de Gantt

9. Análisis de Costes

En este apartado se exponen los costes que supone la realización del proyecto. Se van a dividir los costes en dos clases de recursos, humanos y materiales.

9.1. Recursos humanos

En el desarrollo del proyecto han participado únicamente una persona, encargada de la total realización del proyecto, concretamente un ingeniero técnico en tecnología de telecomunicación.

En la siguiente tabla se muestra detalladamente las horas internas que se ha dedicado al proyecto:

Tabla 5. Horas internas

	Número	Coste horario	Horas	Coste
Ingeniero técnico	1	40 €/h	300 h	12.000 €
Total				12.000 €

9.2. Recursos materiales

Durante el proyecto ha habido unos gastos en determinados recursos materiales, los cuales se pueden dividir en gastos materiales y amortizaciones.

9.2.1. Gastos materiales

Tabla 6.Tabla de gastos materiales

	Precio unitario	Cantidad	Total
Estructura dron	13,31 €	1	13,31 €
Arduino Uno R3	20,00 €	1	20,00 €
Motor 1000kV + hélices 10x4.5 + ESC	9,29 €	4	37,16 €
Batería Lipo 11.1V 2200mAh 30C	14,00 €	1	14,00 €
Cargador de Batería	6,49 €	1	6,49 €
Transmisor Flysky FS-T6 6-CH TX + Receptor	45,87 €	1	45,87 €
Giroscopio MPU-6050	9,90 €	1	9,90 €
Sensor ultrasónico HC-SR04	1,75 €	4	7 €
Cableado	6 €	1	6 €
Modulo Gy-63	4,56 €	1	4,56 €

Componentes electrónicos	2 €	4	8 €
Subtotal			172,29 €

9.2.2. Amortizaciones

Tabla 7.Tabla de amortizaciones

	Número	Coste inicial	Vida útil	Tasa horaria	Uso	Coste
Ordenador	1	1000 €	10.000 h	0,1 €	250 h	25 €
Licencia Software Microsoft Office	1	100 €	700 h	0,14 €	75 h	10,50 €
Total						35,50 €

10. Conclusión

En el presente trabajo de fin de grado se ha desarrollado el montaje completo de una aeronave no tripulada o dron. Este montaje incluye tanto la construcción de los componentes hardware como el desarrollo del software interno que va a ser ejecutado por el dron.

Cabe mencionar que una de las principales motivaciones de este proyecto es la demostración de que, con los conocimientos adquiridos durante el grado en ingeniería en tecnología de telecomunicación, especialmente en las áreas relacionadas con la telemática y la electrónica, se puede diseñar, construir y desarrollar un dron totalmente autónomo. Una aeronave que como demuestra el análisis de costes, tiene unos recursos bastante básicos, pero que finalmente reportan unos grandes resultados.

Con todo esto, se pretende demostrar que los drones y sus infinitos usos están más cerca de lo que se puede pensar a priori de ser utilizados como herramientas de trabajo, sobre todo en tareas que desempeñan personas humanas día a día. Este proyecto pone en práctica una de ellas, el uso de drones excluyentes del factor humano.

Por último, la conclusión que se obtiene de este proyecto es que los beneficios que pueden aportar los drones a día de hoy son muchos, pero dentro de unos años van a ser utilizados para casi cualquier fin. Esto se va a deber al abaratamiento en los costes de fabricación y a los grandes avances en tecnología. Lo que conlleva a generar unos grandes beneficios sociales, facilitando las vidas de los seres humanos y de sus tareas.

11. Anexos

11.1. Normativa aplicable

En todos los lugares del mundo, pero concretamente en España, la venta indiscriminada de vehículos voladores supone un peligro para la Agencia Estatal de Seguridad Aérea (AESA). Por lo que el gobierno reguló su uso con la Ley 18/2014, "Aprobación de medidas urgentes para el crecimiento, la competitividad y la eficiencia", la cual no aplicaba una regulación muy exhaustiva, por lo que el 30 de diciembre del año 2017 entraba en vigor el nuevo marco normativo que regulará la utilización civil de las aeronaves pilotadas por control remoto en España. Se trata de una nueva normativa que introduce nuevos escenarios en los que hasta la fecha no era posible realizar: vuelos en ciudad, vuelos nocturnos, vuelos en espacio aéreo controlado, vuelos más allá del alcance visual para aeronaves de más de 2 kg y vuelos de alcance visual aumentado.

Después de esta regulación, el uso de los drones depende de la forma en que vayan a ser empleados, si se usan de forma profesional, entonces se debe cumplir los siguientes requisitos:

- Estar dado de alta como operador en la Agencia Estatal de Seguridad Aérea (AESA).
- Tener un seguro de responsabilidad civil.
- Tener el título de piloto de drones.
- Tener certificado médico en vigor.

Sin embargo, si se utilizan para fines recreativos:

- Volar a una distancia mínima de 8 km de cualquier aeropuerto o aeródromo.
- Volar fuera del espacio aéreo controlado.
- No sobrepasar los 120 metros de altura sobre el suelo, o sobre el obstáculo más alto situado dentro de una radio de 150 metros desde la aeronave.
- Volar de día y en buenas condiciones meteorológicas. Aquí hay que destacar que si la aeronave pesa menos de 2 kilogramos están permitidos los vuelos nocturnos siempre que no se superen los 50 metros de altura.
- Los vuelos siempre serán dentro del alcance visual del piloto.
- Las aeronaves de menos de 250 gramos podrán volar en ciudad y sobre aglomeraciones de personas y edificios siempre y cuando no se superen los 20 metros de altura.

Para cualquiera de los dos casos, cualquier aeronave debe de llevar una placa identificativa que contenga el nombre del fabricante, tipo, modelo, número de serie, nombre del operador y los datos de contacto del piloto. Aunque si su uso va a estar limitado a espacios interiores privados como es el caso, no hay una normativa especificada.

11.2. Esquemático de Arduino UNO y del dron

Para comprender como funciona internamente y las diferentes entradas/salidas de los pines que componen la placa Arduino UNO, se adjunta la figura 20.

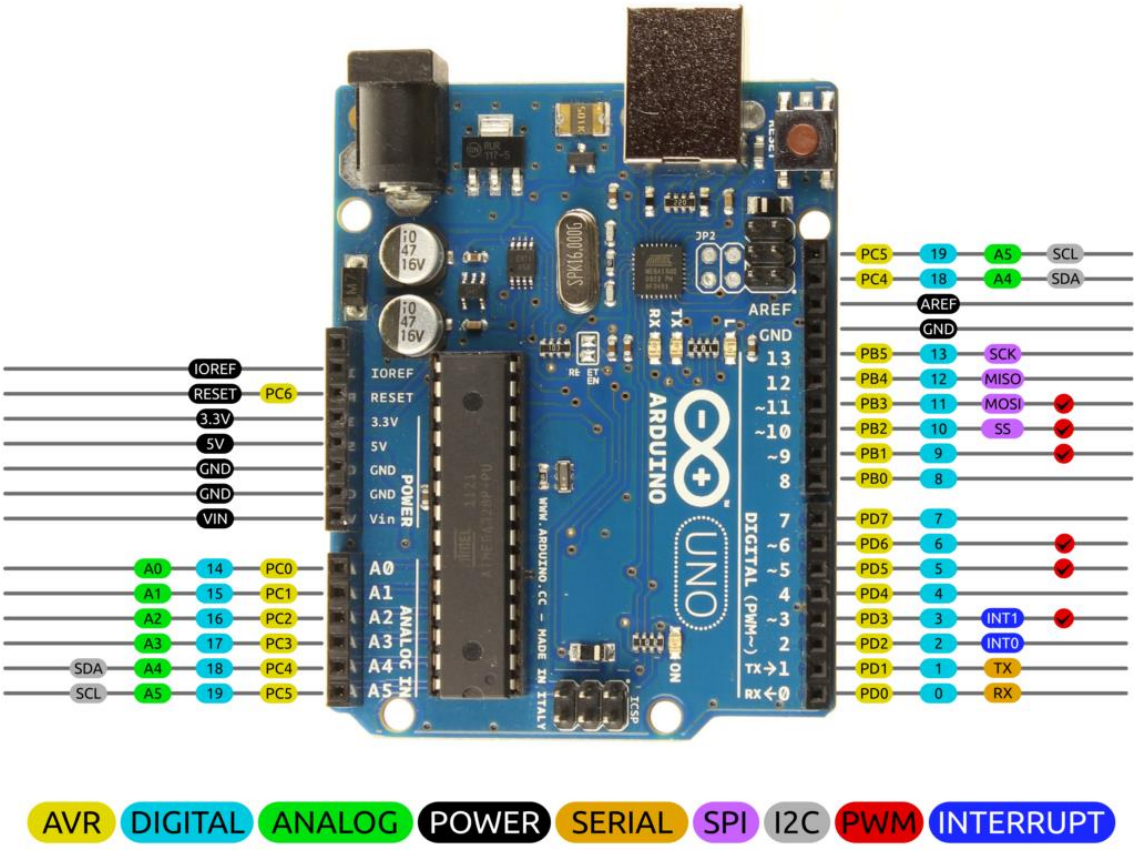


Figura 20. Esquemático Arduino UNO R3

Todos estos pines van conectados a una serie de elementos, es importante tener en cuenta la ubicación y número de pin al que se conectan ya que luego se va a ver reflejado en el código. Es por esto, que en la figura 21 se representa el esquemática que representa todas las conexiones.

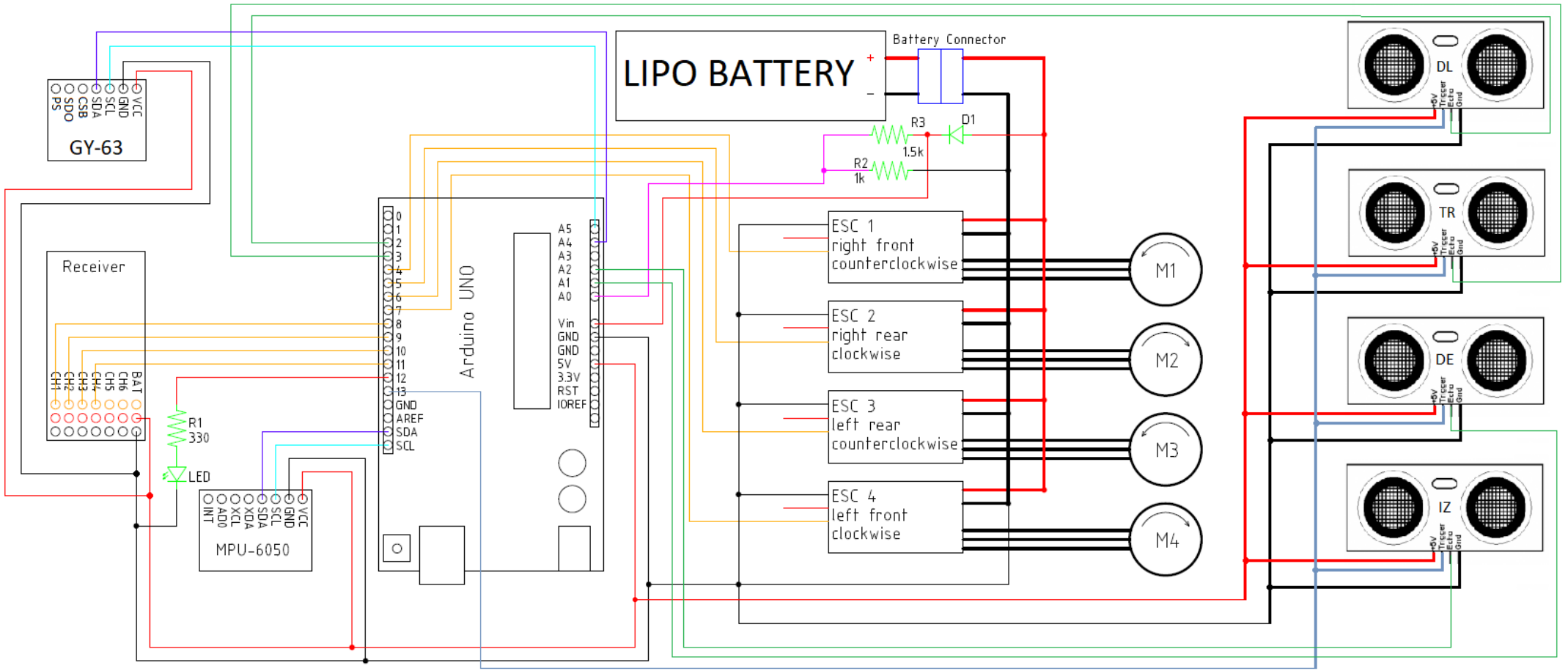


Figura 21. Esquemático del dron

11.3. MPU-6050

En este anexo se muestra el código de Joob Brokking, utilizado para leer los valores del giroscopio. [12]

```
//65.5 = 1 deg/sec (check the datasheet of the MPU-6050 for more information).
gyro_roll_input = (gyro_roll_input * 0.7) + ((gyro_roll / 65.5) * 0.3);   //Gyro pid input is deg/sec.
gyro_pitch_input = (gyro_pitch_input * 0.7) + ((gyro_pitch / 65.5) * 0.3);//Gyro pid input is deg/sec.
gyro_yaw_input = (gyro_yaw_input * 0.7) + ((gyro_yaw / 65.5) * 0.3);      //Gyro pid input is deg/sec.

void gyro_signalen(){
  //Read the MPU-6050
  if(eeprom_data[31] == 1){
    Wire.beginTransmission(gyro_address);                        //Start communication with the gyro.
    Wire.write(0x3B);                                            //Start reading @ register 43h and auto increment with every
read.
    Wire.endTransmission();                                      //End the transmission.
    Wire.requestFrom(gyro_address,14);                           //Request 14 bytes from the gyro.

    receiver_input_channel_1 = convert_receiver_channel(1);      //Convert the actual receiver signals for pitch
to the standard 1000 - 2000us.
    receiver_input_channel_2 = convert_receiver_channel(2);      //Convert the actual receiver signals for roll to
the standard 1000 - 2000us.
    receiver_input_channel_3 = convert_receiver_channel(3);      //Convert the actual receiver signals for
throttle to the standard 1000 - 2000us.
    receiver_input_channel_4 = convert_receiver_channel(4);      //Convert the actual receiver signals for yaw
to the standard 1000 - 2000us.

    while(Wire.available() < 14);                                //Wait until the 14 bytes are received.
    acc_axis[1] = Wire.read()<<8|Wire.read();                    //Add the low and high byte to the acc_x variable.
    acc_axis[2] = Wire.read()<<8|Wire.read();                    //Add the low and high byte to the acc_y variable.
    acc_axis[3] = Wire.read()<<8|Wire.read();                    //Add the low and high byte to the acc_z variable.
    temperature = Wire.read()<<8|Wire.read();                    //Add the low and high byte to the temperature
variable.
    gyro_axis[1] = Wire.read()<<8|Wire.read();                   //Read high and low part of the angular data.
    gyro_axis[2] = Wire.read()<<8|Wire.read();                   //Read high and low part of the angular data.
    gyro_axis[3] = Wire.read()<<8|Wire.read();                   //Read high and low part of the angular data.
  }

  if(cal_int == 2000){
    gyro_axis[1] -= gyro_axis_cal[1];                            //Only compensate after the calibration.
    gyro_axis[2] -= gyro_axis_cal[2];                            //Only compensate after the calibration.
    gyro_axis[3] -= gyro_axis_cal[3];                            //Only compensate after the calibration.
  }
  gyro_roll = gyro_axis[eeprom_data[28] & 0b00000011];           //Set gyro_roll to the correct axis that was
stored in the EEPROM.
  if(eeprom_data[28] & 0b10000000)gyro_roll *= -1;               //Invert gyro_roll if the MSB of EEPROM bit
28 is set.
  gyro_pitch = gyro_axis[eeprom_data[29] & 0b00000011];          //Set gyro_pitch to the correct axis that was
stored in the EEPROM.
  if(eeprom_data[29] & 0b10000000)gyro_pitch *= -1;              //Invert gyro_pitch if the MSB of EEPROM bit
29 is set.
  gyro_yaw = gyro_axis[eeprom_data[30] & 0b00000011];            //Set gyro_yaw to the correct axis that was
stored in the EEPROM.
  if(eeprom_data[30] & 0b10000000)gyro_yaw *= -1;                //Invert gyro_yaw if the MSB of EEPROM bit
30 is set.

  acc_x = acc_axis[eeprom_data[29] & 0b00000011];                //Set acc_x to the correct axis that was stored
in the EEPROM.
  if(eeprom_data[29] & 0b10000000)acc_x *= -1;                   //Invert acc_x if the MSB of EEPROM bit 29 is
set.
  acc_y = acc_axis[eeprom_data[28] & 0b00000011];                //Set acc_y to the correct axis that was stored
in the EEPROM.
  if(eeprom_data[28] & 0b10000000)acc_y *= -1;                   //Invert acc_y if the MSB of EEPROM bit 28 is
set.
```

```
  acc_z = acc_axis[eeprom_data[30] & 0b00000011];                    //Set acc_z to the correct axis that was stored in
the EEPROM.
  if(eeprom_data[30] & 0b10000000)acc_z *= -1;                       //Invert acc_z if the MSB of EEPROM bit 30 is
set.
}



void set_gyro_registers(){
  //Setup the MPU-6050
  if(eeprom_data[31] == 1){
    Wire.beginTransmission(gyro_address);                            //Start communication with the address found
during search.
    Wire.write(0x6B);                                               //We want to write to the PWR_MGMT_1 register (6B hex)
    Wire.write(0x00);                                               //Set the register bits as 00000000 to activate the gyro
    Wire.endTransmission();                                          //End the transmission with the gyro.

    Wire.beginTransmission(gyro_address);                            //Start communication with the address found
during search.
    Wire.write(0x1B);                                               //We want to write to the GYRO_CONFIG register (1B
hex)
    Wire.write(0x08);                                               //Set the register bits as 00001000 (500dps full scale)
    Wire.endTransmission();                                          //End the transmission with the gyro

    Wire.beginTransmission(gyro_address);                            //Start communication with the address found
during search.
    Wire.write(0x1C);                                               //We want to write to the ACCEL_CONFIG register (1A
hex)
    Wire.write(0x10);                                               //Set the register bits as 00010000 (+/- 8g full scale range)
    Wire.endTransmission();                                          //End the transmission with the gyro

    //Let's perform a random register check to see if the values are written correct
    Wire.beginTransmission(gyro_address);                            //Start communication with the address found
during search
    Wire.write(0x1B);                                               //Start reading @ register 0x1B
    Wire.endTransmission();                                          //End the transmission
    Wire.requestFrom(gyro_address, 1);                                    //Request 1 bytes from the gyro
    while(Wire.available() < 1);                                    //Wait until the 6 bytes are received
    if(Wire.read() != 0x08){                                        //Check if the value is 0x08
      digitalWrite(12,HIGH);                                        //Turn on the warning led
      while(1)delay(10);                                            //Stay in this loop for ever
    }

    Wire.beginTransmission(gyro_address);                            //Start communication with the address found
during search
    Wire.write(0x1A);                                               //We want to write to the CONFIG register (1A hex)
    Wire.write(0x03);                                               //Set the register bits as 00000011 (Set Digital Low Pass
Filter to ~43Hz)
    Wire.endTransmission();                                          //End the transmission with the gyro

  }
}
```

11. Referencias

[1] Usos de los drones en la sociedad: http://www.globalmediterranea.es/uso-drones-la-actualidad-futuro/

[2] Aplicaciones de orientación y localización pensadas para espacios interiores: https://www.techadvisor.co.uk/feature/software/best-apps-for-navigating-inside-buildings-3573460/

[3] Componentes de un dron: https://droningpage.wordpress.com/2014/10/19/que-partes-componen-un-drone-multirotor/

[4]Documentación sobre Arduino: https://www.arduino.cc/reference/en/

[5] Información sobre STM32: https://www.st.com/en/microcontrollers/stm32-32-bit-arm-cortex-mcus.html

[6] Información sobre Raspberry: https://opensource.com/resources/raspberry-pi

[7] Información sobre Motores brushed: http://www.cochesrc.com/motores-brushed-funcionamiento-y-caracteristicas-a3606.html

[8] Información sobre Motores brushless: http://www.quadruino.com/guia-2/materiales-necesarios-1/motores-brushless

[9] Información sobre ESCs: http://www.quadruino.com/guia-2/materiales-necesarios-1/esc

[10] Información sobre MPU-6050 en Arduino: https://www.prometec.net/usando-el-mpu6050/

[11]Información sobre el PID: https://www.hackster.io/ElvisWolcott/drone-basics-pid-83da06

[12] Proyectos de Joop Brokking:

- http://www.brokking.net/
- https://www.youtube.com/user/MacPuffdog/videos

[13] Información y librería MS5611: https://github.com/gronat/MS5611

[14] Información y librería NewPing: https://playground.arduino.cc/Code/NewPing

[15] Información sobre sensor ultrasónico en Arduino: https://www.prometec.net/sensor-distancia/