GRADO EN INGENIERÍA EN TECNOLOGÍA DE TELECOMUNICACIÓN

TRABAJO FIN DE GRADO

SEGUIMIENTO AUTÓNOMO DE UNA PERSONA CON MARCADOR VISUALMENTE IDENTIFICADA POR UN DRON

Alumno/Alumna: Rodríguez, Suárez-Llanos, Julio

Director/Directora: Espinosa, Acereda, Jon Koldobika

Curso: 2017-2018

Fecha: 17 de Julio de 2018

Resumen trilingüe

Resumen

En este proyecto se diseña y desarrolla un dron capaz de seguir a una persona con un marcador visual. El dron, también llamado cuadricóptero, incorpora una cámara en su parte frontal para poder seguir al objetivo.

Se va a diseñar un software que detecta un color especifico definido por el usuario y además realiza el seguimiento de dicho color en el tiempo. También se va a desarrollar el software que controla el vuelo del dron y que consiga mantenerse a una altitud específica sobre el suelo.

El dron es completamente autónomo por lo que no es necesaria la interacción con el usuario.

Abstract

In this project a drone that can follow a person with a visual marker is designed and developed. The drone, also called quadcopter, has a camera on its front so it can track the target.

It’s going to be designed a software that detects a specific colour predefined by the user, and it tracks that colour along the time also. Another software capable of controlling the drone and maintaining a specific altitude is going to be designed as well.

Since the drone is completely autonomous, it’s not necessary the interaction with the user.

Laburpena

Proiektu honetan diseinatuko eta egingo da jarraitzen duen dron bat kolorea detektatzen. Dron-a, “quadcopter” deitu ere, kamera bat aurrean dauka artezkaria jarraitzeko.

Diseinatuko den software-rra kolore bat detektatzen du eta jarraitzen du pertsona denboran. Bestalde, diseinatuko da dron-aren hegaldi kontrolatzailea eta hegan egin ahal duela altitude batean.

Ez du behar erabiltzailearen interakzioa, autonomoa balitelako.

Índice

Acrónimos

ESC: Electronic Speed Control

USB: Universal Serial Bus

UAV: Unmanned Aerial Vehicle

PWM: Pulse Width Modulation

BEC: Battery Eliminator Circuit

VDC: Volts Direct Current

RAM: Random Access Memory

EEPROM: Electrically Erasable Programmable Read-Only Memory

SRAM: Static Random Access Memory

IDE: Integrated Development Environment

IMU: Inertial Measurement Unit

I2C: Inter-Integrated Circuit

SPI: Serial Peripheral Interface

ISR: Interrupt Service Routine

NTSC: National Television System Committee

BGR: Blue Green Red

HSV: Hue Saturation Value

PID: Proportional Integral Derivative

Lista de figuras

Lista de tablas

1. Introducción

En este proyecto se realiza el diseño y desarrollo de un software que controla el vuelo de un dron de forma completamente autónoma en base a un marcador visual (un chaleco amarillo reflectante, por ejemplo) al cual sigue mediante una cámara conectada al dron. Además del desarrollo software de seguimiento, se realiza el montaje del dron por piezas y su respectivo programa de vuelo.

Los drones son auge en la actualidad ya que al humano siempre le ha apasionado volar, por lo que todos los aparatos que son capaces de elevarse del suelo son llamativos para las personas. El dron no solo se levanta del suelo, sino que tiene infinidad de posibilidades gracias a su posibilidad de mantenerse completamente quieto en un punto, manteniendo una estabilidad asombrosa.

El enfoque del proyecto al final es facilitar al usuario, tanto a nivel de ocio como a nivel profesional, el manejo y control de un dron ya que puede llegar a ser difícil y tedioso el manejo de un dron. De esta manera, el usuario solo tiene que caminar y el dron le seguirá a donde esa persona quiera.

Figura 1: DJI Mavic Pro

2. Contexto

Un dron capaz de realizar el seguimiento de una persona abre las puertas a el mundo de los selfis. Todo el mundo sabe que los selfis están a la orden del día, por ello un dron capaz de seguirte a donde tú quieras lleva los selfis al siguiente nivel.

En el mundo del cine o cortometrajes, el hecho de que puedas tener una cámara que te siga a donde vaya, sin restricciones de carreteras o caminos, da cabida a escenarios espectaculares sin la necesidad de un helicóptero o una grúa, cuyos costes pueden ser demasiado altos.

Figura 2: Dron siguiendo a una persona en una escena

Para seguridad en un festival o un evento del estilo, el dron puede seguir a una persona de seguridad de manera que tiene un rango de visión de lo que pasa a su alrededor mucho mayor y puede actuar con más rapidez.

En un asalto policial, un dron que sigue a un agente, da perspectiva a los coordinadores de lo que está ocurriendo desde un punto de vista diferente y puede guiar al equipo táctico.

3. Objetivo del proyecto

El objetivo de este proyecto es desarrollar el control completamente autónomo de un dron, de manera que no sea necesaria la interacción con el usuario, como se puede observar en el siguiente esquema:

Figura 3: Esquema objetivo proyecto

Se pretende diseñar un software de reconocimiento de colores para detectar un chaleco reflectante y otro que siga a la persona con el chaleco.

También se pretende integrar un barómetro en el dron de manera que pueda mantener una altitud constante.

4. Beneficios

4.1 Técnicos

Los drones hoy en día tienen muchas aplicaciones, así que el abanico que abarca de posibilidades es muy grande.

En este proyecto se va a trabajar con un dron equipado con una cámara y va a seguir a un objetivo. El desarrollo del software y posterior mejora del mismo, beneficia a la rama de electrónica de seguimiento ya que se puede aplicar en futuros coches autónomos o incluso aviones de pasajeros que viajan completamente automáticamente.

4.2 Económicos

Uno de sus usos más frecuentes es en el ámbito de la seguridad. Mientras que una cámara pegada a una pared tiene el inconveniente que solo tienes un campo de visión determinado, con el dron puedes realizar el seguimiento de una persona sospechosa, aprovechando así la altura que puede tomar el dron para no alertar a dicha persona. Esto reduce mucho los costos de instalación de cámaras de seguridad y, al tener un campo de visión tan amplio, no es necesario tanto personal de seguridad vigilando cada esquina.

Figura 4: Dron Prosegur

En la industria del cine empiezan a ser cada vez más utilizados estos drones, ya que te dan unos planos que no podría conseguir de otra forma. Para los planos aéreos hacían uso de helicópteros y los costos de un piloto y el combustible, así como el precio del helicóptero son muy elevados y no cualquiera puede permitírselo. Además, con el sistema de seguimiento del dron, el plano de la persona siempre es presente y no hay posibilidad de fallo humano. Esto abre muchas puertas al cine a gente con menos recursos económicos.

Otro ejemplo que está empezando a crecer es el servicio de correo mediante drones. Esto ahorra los atascos de las carreteras y la necesidad de carteros. El tiempo de envío se ve reducido extremadamente, así como el precio de envío, ya que se eliminan los costos de combustible y personal.

Figura 5: Dron Amazon

4.3 Sociales

En el sector de la vigilancia y prevención, los drones son auge. Tanto es así que en España se crearon los primeros drones que fueron diseñados para la prevención y control de los incendios forestales, lo cual es un problema en España últimamente, sobre todo en Galicia. Estos drones reúnen información recorriendo hectáreas de bosque y con ello se hacen unas estadísticas para una posible prevención o expansión de incendios. Si una persona se queda atrapada en un incendio forestal,

estos drones son capaces de hacer un seguimiento de dicha persona para poder mandar ayuda inmediatamente.

Figura 6: Dron vigilando el bosque

Otra de sus muchas posibilidades es en una obra. Muchas veces hay lugares de un edificio que son inaccesibles o de difícil acceso. Para ello se pueden utilizar los drones con una cámara conectada. Da una perspectiva al jefe de obra u obreros para analizar la situación de una forma segura.

5. Descripción elementos del dron

En la siguiente figura se muestra el dron desglosado en sus diferentes componentes:

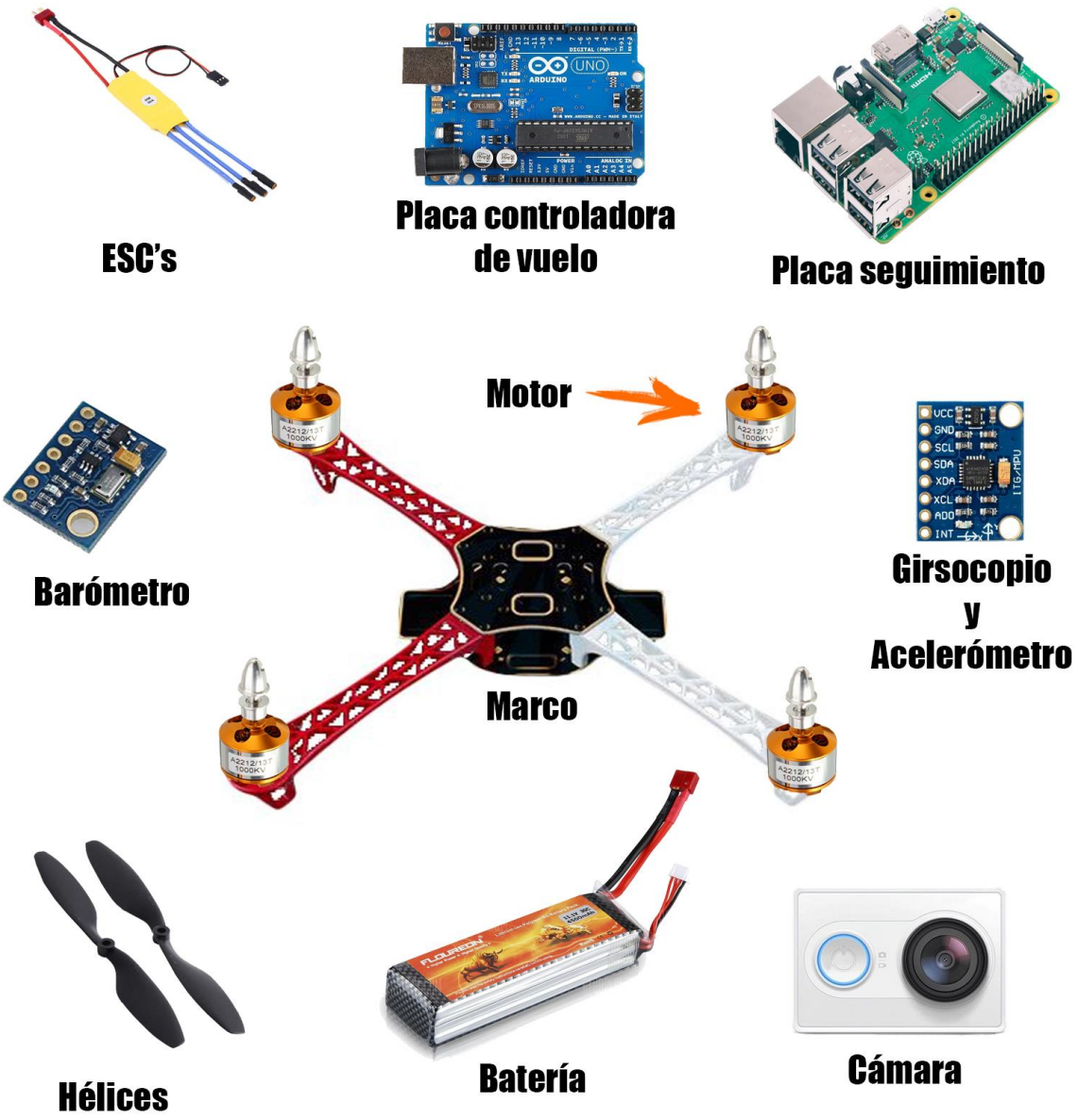

Figura 7: Esquema dron

Marco: El marco del dron es el soporte de todos los elementos. Tiene un centro metálico de dos pisos para poder instalar componentes, y 4 brazos largos donde irá 1 motor en la esquina de cada uno. El chasis dispone de diversos agujeros de tamaño universal para la compatibilidad de componentes.

Motor: El dron dispone de 4 motores eléctricos los cuales hacen rotar las hélices para hacer que el dron vuele.

Hélices: Son palas diseñadas para que, al girar a mucha velocidad, se produzca una fuerte corriente de aire hacia el suelo.

ESC's: Se encargan de transformar la señal que le envía el controlador de vuelo a una descarga de intensidad controlada para hacer rotar el motor a la velocidad especificada.

Batería: Parte fundamental del dron ya que entrega la energía que necesita el dron para su funcionamiento.

Placa controladora de vuelo: Es un microcontrolador que se encarga de mandar a los ESC's la potencia que necesita cada motor en cada instante para poder volar y dirigir el vuelo hacia donde se desee.

Placa seguimiento: Esta placa es encargada de procesar las imágenes recibidas mediante la cámara y manipular el controlador de vuelo para realizar el seguimiento.

Cámara: Captura lo que ve el dron en el momento y lo transmite a la placa de seguimiento mediante USB.

Giroscopio: Periférico que mide la velocidad angular del dron

Acelerómetro: Mide la aceleración estática (gravedad) y dinámica (en el eje horizontal XY) del dron.

Barómetro: Sensor de presión atmosférica que se utiliza para calcular la altitud del dron en cada instante.

5.1 Motores

Los motores eléctricos consumen mucha potencia y es por ello que es el elemento crítico del dron en cuanto a duración de vuelo. Hay dos tipos de motores eléctricos: Con escobillas (Brushed) y sin escobillas (Brushless) [BRSH15]. Los primeros son más baratos, pero menos eficientes y los segundos son más nuevos y, por tanto, más caros.

5.1.1 Brushed

Su uso en radiocontrol cada vez es menor ya que sobre todo en aparatos orientados al vuelo, se requiere mucha potencia y el menor peso posible, y estos son mucho más pesados.

Están compuestos por un rotor, un conmutador, un eje, un imán de campo y escobillas.

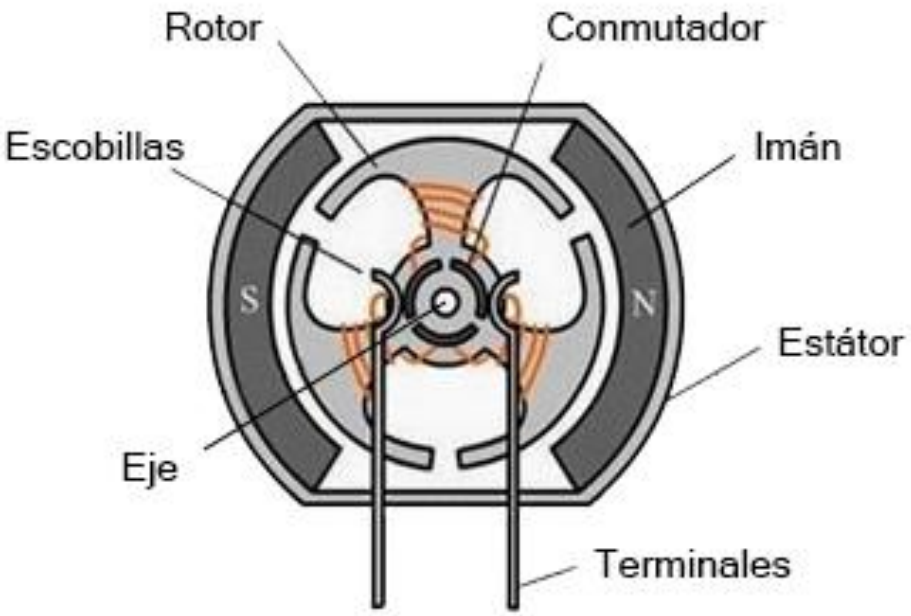

Figura 8: Motor Brushed partes

Las escobillas cargan el conmutador en la polaridad inversa respecto al imán, haciendo que el rotor gire. La dirección de rotación puede ser cambiada alimentando el motor del revés.

La principal ventaja de este tipo de motor es el precio y su robustez, dado que por su sencillez el motor es mucho más robusto y resistente. Por otro lado, la sencillez tiene sus problemas. Requiere más mantenimiento y tiene problemas de disipación del calor.

Figura 9: Motor Brushed

5.1.2 Brushless

Estos motores son mucho más eficientes y proporcionan mayores rpm. Al no tener escobillas como su propio nombre indica, no existe fricción internamente en el motor por lo que apenas requieren mantenimiento o sufren desgaste.

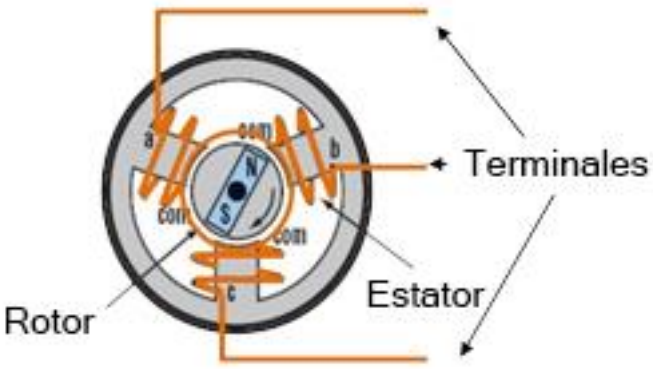

Figura 10: Motor Brushless partes

Es un motor síncrono, es decir, el rotor gira a la misma frecuencia que el estator y pueden ser monofásicos, bifásicos o trifásicos. No precisan de escobillas ni de conmutador, sencillamente tienen una circuitería de detección para saber en qué posición se encuentra el rotor en un momento preciso. Esto conlleva a una mayor precisión, pero mayor precio también, además de necesitar un controlador externo (ESC) que también encarece el uso de estos motores. A diferencia de los brushed, éstos hacen menos ruido, son más pequeños y tienen una mejor disipación del calor.

5.2 ESC's

Los ESC's [ESC17] son necesarios cuando elegimos un motor sin escobillas, ya que como se ha descrito anteriormente, éstos precisan de un control mucho más preciso. Este control se realiza desde un ordenador, que en éste caso será la placa microcontroladora encargada del control de vuelo. Aquí es donde entran en juego el uso de los ESC's.

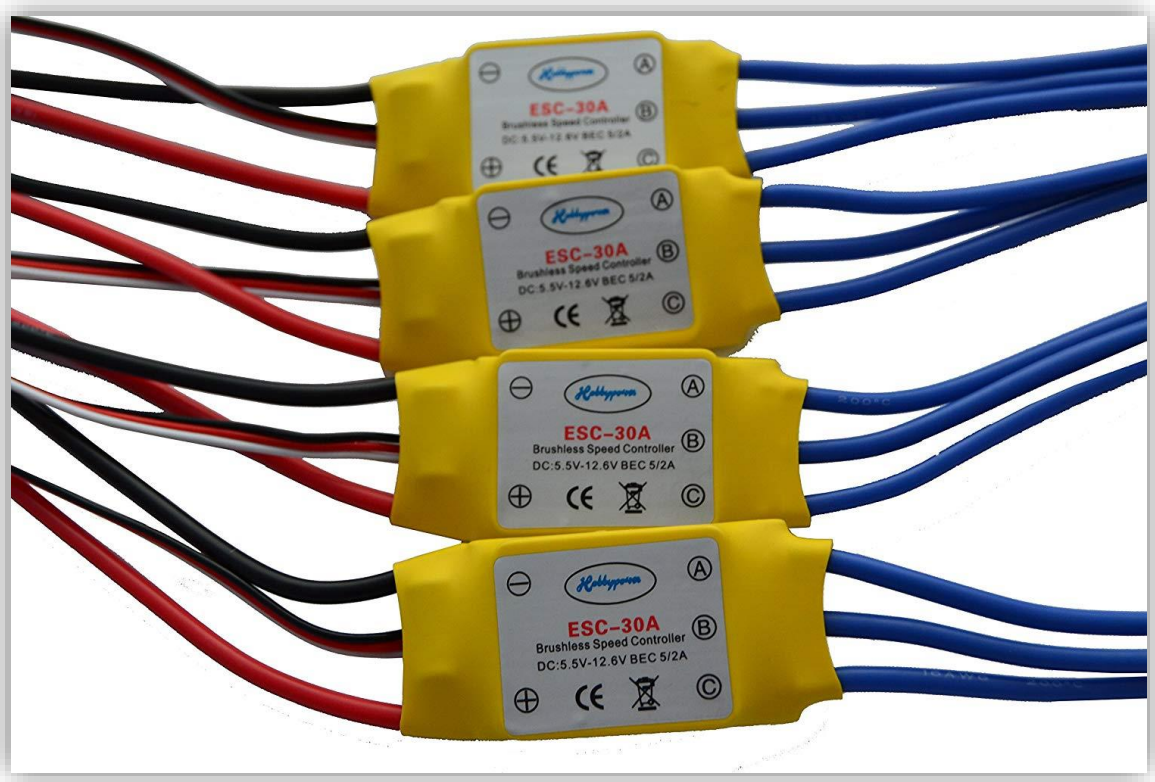

Figura 11: 4x ESC's

Los ESC's controlan la velocidad de rotación del motor eléctrico sin escobillas y para ello crean una corriente alterna trifásica a partir de una corriente continua, suministrada por la batería. Cada ESC tiene 3 cables que van conectados al motor. Uno de los polos, genera un pequeño voltaje proporcional a la velocidad de giro (fuerza electromotriz), y sirve para que el motor gire a una velocidad y un sentido específico. Con esta información, el ESC manda la corriente a los electroimanes del motor y éste empieza a girar.

Suelen incorporar un sistema de regulación de voltaje (BEC) transformando el voltaje de la batería (7.4V, 11.1V, 14.8V, ...) a 5V, que es el voltaje utilizado por la mayoría de los componentes del dron.

El control se hace desde un ordenador y la comunicación se realiza mediante PWM.

5.3 Placas microcontroladoras

El cerebro del dron es el controlador de vuelo ya que es el encargado de coordinar y manejar todos los periféricos del dron para mandar la información a los ESC's y que pueda volar. Para ello se necesita un ordenador, que en el caso de un dron deberá ser algo pequeño y de poco peso. Por ello se ve la necesidad de utilizar placas microcontroladoras.

Por otra parte, para el seguimiento autónomo del dron es necesaria una placa que procese las imágenes que se reciben desde la cámara para manipular el vuelo del dron de manera que siga al objetivo, por lo que será necesaria mucha velocidad de procesamiento.

5.3.1 Arduino

Arduino es un software de código abierto muy potente para proyectos de pequeña envergadura [ARD18]. La programación se realiza en código C, uno de los lenguajes de programación más comunes y utilizados.

Además, dispone de varias placas de pruebas (protoboard) como la Arduino UNO o la Arduino nano. Ambas comparten el mismo microprocesador ATmega328P con arquitectura AVR. Disponen de una velocidad de reloj de 16MHz y tienen un peso muy reducido.

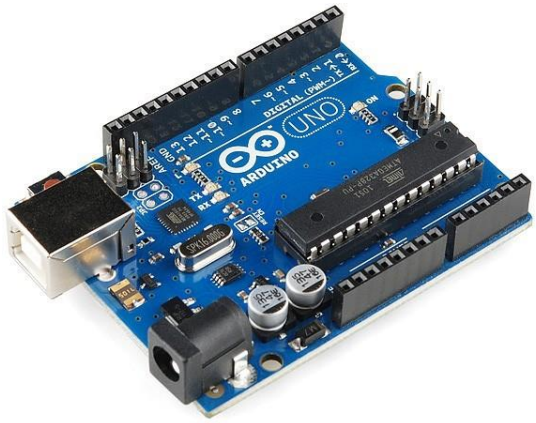

Figura 12: Arduino UNO R3

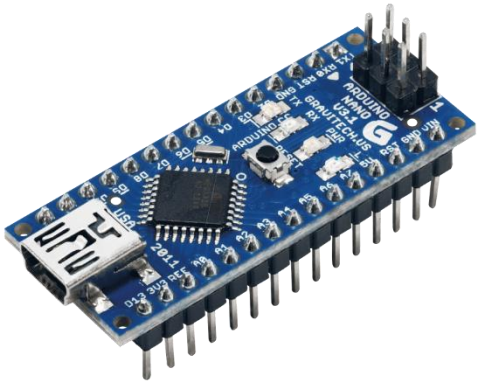

Figura 13: Arduino Nano

Tabla 1: Comparativa UNO y Nano

Arduino protoboards

	UNO	Nano	
Microcontrolador	ATmega328P	ATmega328P	-
Velocidad	16	16	MHz
Tensión DC	5	5	V
Voltaje entrada	6-20	7-12	V
Pines digitales	16	22	-
Entradas analógicas	8	6	-
Memoria Flash	32	32	KB
Memoria EEPROM	1	1	KB
Peso	25	7	g
Pines de prueba	SI	NO	-

Como se puede ver en la tabla, ambas versiones son relativamente parecidas.

5.3.2 Raspberry Pi

Esta placa es muy famosa debido a sus periféricos y capacidades en relación con su precio. Existen varios modelos, el más reciente, la Raspberry Pi 3B+ con un microcontrolador Broadcom BCM2837B0 basado en ARM que rinde a 1.4GHz (0.2GHz más que su predecesor) [PI18]. Incorpora además periféricos "well-known" como USB, Wi-Fi, HDMI y Bluetooth. Tiene una entrada CSI a la cual sería posible conectar una cámara PI.

Figura 14: Raspberry Pi 3B+

Tabla 2: Características Raspberry Pi 3B+

Raspberry Pi 3B+

Microcontrolador	BCM2837B0	-
Velocidad	1,4	GHz
Tensión DC	5	V
Voltaje entrada	3,3	V
Pines digitales	40	-
Entradas analógicas	0	-
Memoria	Externa SD	KB
RAM	1	GB
USB	4x 2.0	-
Wi-Fi	2,4 - 5	GHz
LAN	300	Mbps
Bluetooth	4.2	-
Peso	25	g

5.3.3 STM32

La familia de procesadores STM32 son conocidos por su potencia para aplicaciones de bajo consumo. El STM32F103C8 [STM18] tiene una velocidad de 72MHz basado en ARM 32-bit Cortex-M3. Es el competidor directo de la Arduino Nano o Uno, ya que cuadruplica su velocidad y tiene mayor memoria (128KB).

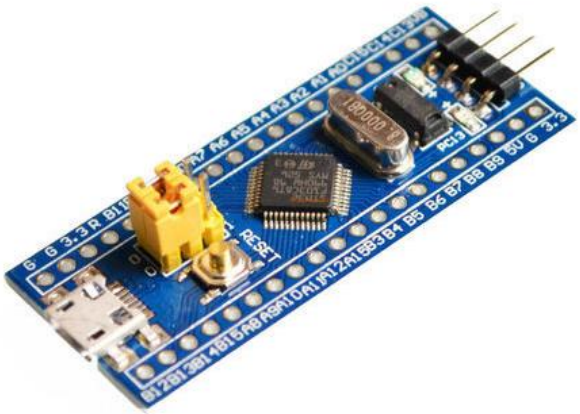

Figura 15: STM32 Protoboard

Tabla 3: Características STM32

STM32 Protoboard

Microcontrolador	STM32F103C8	-
Velocidad	72	MHz
Tensión DC	3.3-5	V
Voltaje entrada	6-20	V
Pines digitales	37	-
Entradas analógicas	16	-
Memoria Flash	128	KB
Memoria EEPROM	-	KB
Peso	5	g
Pines de prueba	SI	-

5.4 Giroscopio y acelerómetro (IMU)

Para que el dron pueda saber que orientación lleva, dónde está situado en el plano y que fuerzas están ejerciendo sobre él (aceleración), hacemos uso de una unidad de medición inercial (IMU) [IMU16]. Se trata de un dispositivo electrónico compuesto por un giroscopio y un acelerómetro que puede medir la velocidad, orientación y las fuerzas gravitacionales que actúan sobre él. Es algo imprescindible para el vuelo autónomo de un dron e incluso para aviones, buques o misiles guiados.

El giroscopio sirve para medir, mantener o cambiar la orientación que el dron tiene en el espacio. Esto es capaz de hacerlo ya que mide la velocidad angular, que es la velocidad con la que algo gira sobre un eje. El problema es que un giroscopio sería ideal en un entorno donde no existieran fuerzas externas de aceleración, como la gravedad. Por eso es necesario complementar el giroscopio con un acelerómetro. El acelerómetro mide la aceleración estática (la gravedad) y la aceleración dinámica (movimientos instantáneos). Gracias a la aceleración estática el dron puede saber cómo está orientado respecto a la tierra, que complementado con el giroscopio se consigue la estabilidad del dron.

5.4.1 L3G4200D

Es un giroscopio de 3 ejes diseñado por STMicroelectronics muy utilizado para drones de carreras dada su precisión y velocidad de acceso a memoria. Tarda tan solo 10ms en encenderse lo cual lo hace ideal ante posibles fallos instantáneos que pueda sufrir el dron en vuelo. Tiene una sensibilidad de hasta 2000dps y dispone de un sensor de temperatura, el cual es muy útil para poder corregir errores causados por cambios en la temperatura.

Figura 16: Giroscopio L3G4200D

5.4.2 LIS331

El acelerómetro LIS331 es uno de los mejores acelerómetros de tres ejes del mercado debido a su precio competente y su calidad excepcional. Utiliza modos de bajo consumo consiguiendo así un consumo de 250µA. Diseñado por STMicroelectronics este acelerómetro consigue una escala de hasta 24g, con la cual se consigue una precisión asombrosa.

Figura 17: Acelerómetro LIS331

5.4.3 MPU-6050

Fabricado por IvenSense, el MPU-6050 es un sensor de 6 ejes (acelerómetro y giroscopio) que combina ambos en un solo dispositivo [MPU18]. La sensibilidad del giroscopio es de 2000dps y el acelerómetro tiene un rango máximo de 16g. Incluye también sensor de temperatura para la corrección de errores.

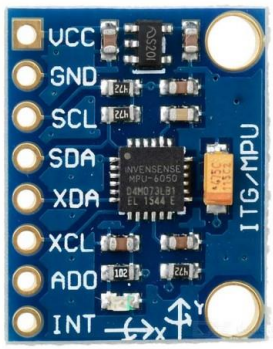

Figura 18: MPU-6050

5.5 Barómetro

El barómetro se utiliza para medir la presión del ambiente, y mediante la fórmula de presión atmosférica se puede calcular la altitud del dron. Estos componentes son muy importantes en cualquier vehículo aéreo ya que saber a qué altitud se encuentra cualquiera de ellos, es de vital importancia en condiciones de mínima visibilidad.

5.5.1 BMP180

El BMP180 [BMP18] ha sido diseñado por Bosch, para sustituir al BMP085. Ambos son completamente idénticos en cuanto a programación software por lo que existe mucha información respecto a este componente. La presión que puede medir es de 1100-300 mbar, equivalente a (-500)-9000m a nivel del mar, con una resolución de 0.03mbar / 0.25m.

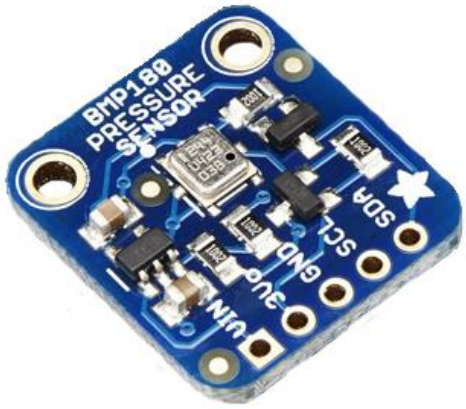

Figura 19: Barómetro BMP180

5.5.2 MS5611

TE Connectivity diseñó el barómetro MS5611 [MS18] con la capacidad de tener comunicación I2C y SPI. Es uno de los barómetros más utilizados para proyectos pequeños debido a su fiabilidad. Su rango de operación es 1200-10mbar, con una resolución de 0.01m de precisión. Además, incorpora un sensor de temperatura para corregir los errores debidos a los cambios en la temperatura del ambiente.

Figura 20: Barómetro MS5611

5.6 Tratamiento de imágenes

Para la detección de color y el seguimiento de la persona con el marcador visual será necesario desarrollar un software que mediante las imágenes que recibe desde una cámara, se detecte el color y se siga a ese color en el tiempo. Para ello se utilizarán librerías específicas para el tratamiento de imágenes:

5.6.1 OpenCV

OpenCV [OCV10] es la librería más utilizada de tratamiento de imágenes, debido a su potencial y toda la documentación existente de ésta librería. Puede ser utilizada en diversos lenguajes de programación, como en C++, Java o Python, siendo éste último el más utilizado junto a C++. Puede utilizarse en Linux, Windows, Mac OS, IOs y Android. La librería está escrita en C/C++ y utiliza procesamiento multi-nucleo para un mayor rendimiento.

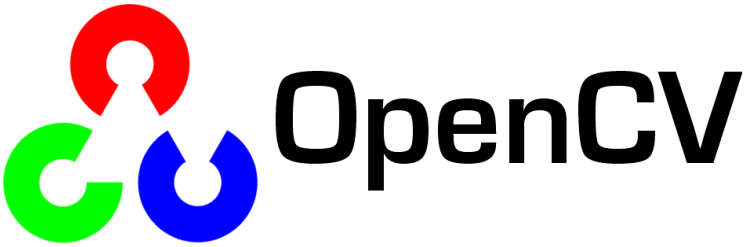

Figura 21: OpenCV Logo

5.6.2 Pixy

Pixy [PIX13] nace de un proyecto iniciado en 1999, CMUCam, en la universidad de Carnegie Mellon, donde la mayoría de los robots del mundo son creados. Con el tiempo y evolución se creó Pixy en 2013, siendo la quinta versión de CMUCam. Se trata de una cámara con una CPU ya integrada que detecta el color y el movimiento de objetos. Se puede conectar a una placa Arduino y la programación se hace directamente en el IDE Arduino mediante librerías optimizadas.

Figura 22: Pixy

6. Análisis de alternativas

6.1 Motores

Analizando ambos motores, claramente el motor brushless es mucho mejor y más eficiente pese a su precio más elevado y la necesidad de un controlador externo. En el dron se necesita mucha precisión y potencia, con la mejor eficiencia posible debido a la energía que necesita para volar, asique un mejor rendimiento, dará lugar a más tiempo de vuelo, que es el punto más débil de los drones.

Desde el punto de vista de la fiabilidad también gana el motor brushless ya que como se ha descrito anteriormente, la ausencia de escobillas da como resultado la supresión o minimización de el mantenimiento de estos.

Además, los motores brushless alcanzan una mayor velocidad y un mayor torque, algo imprescindible para el vuelo del dron.

Una cosa a tener muy en cuenta a la hora de elegir un motor es la calidad del rodamiento ya que, si no es de una calidad suficiente, acabará rompiéndose rápidamente.

Por todo ello se ha decidido utilizar un motor brushless, cuyas especificaciones son descritas a continuación:

Tabla 4: Características A2212

A2212 1000KV Brushless Outrunner Motor

KV	1000	RPM/V
Max. Eficacia	80%	-
Rin	0,09	Ω
Max. Consumo	12	A
Peso	47	g
Tamaño	27,5 x 30	mm^2
Diámetro del eje	3,17	mm
Polos	13	-

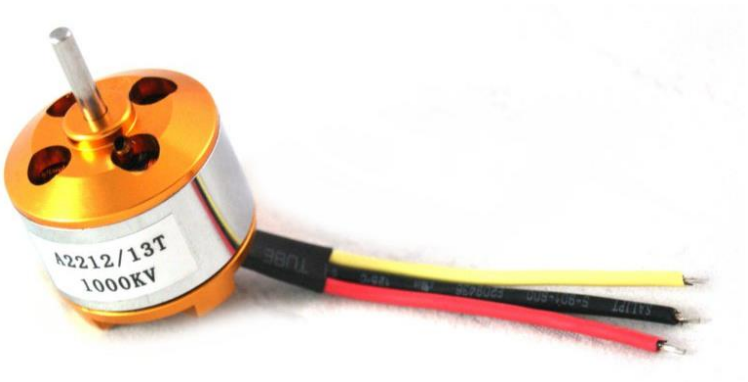

Figura 23: A2212 1000KV Motor

6.2 ESC's

A la hora de seleccionar el ESC hay que tener en cuenta el amperaje que pueden suministrar a los motores. El amperaje que soporta el ESC tiene que ser mayor que el de los motores para evitar sobrecalentamientos en el ESC. En cuanto al sistema BEC, no es necesario para este proyecto ya que la placa microcontroladora ya suministra 5VDC.

Por todo ello se han escogido unos ESC de 30A con sistema BEC con las siguientes características:

Tabla 5: Características ESC 30A

30A ESC Brushless

BEC	5	V
Tensión de corte	4	V
Corriente Max.	30	A
Celdas batería	02-mar	-

Figura 24: ESC 30A Brushless

6.3 Placa controladora de vuelo

Para el controlador de vuelo del dron necesitamos que admita un voltaje de entrada de 12.4V, ya que es el voltaje de la batería con la carga al 100%, por tanto, la Raspberry Pi queda descartada para controladora de vuelo.

Es necesario guardar los datos de calibración inicial de los periféricos como el giroscopio, barómetro y acelerómetro. Para ello será necesaria una memoria no volátil para almacenar esos datos. Las placas Arduino incorporan una EEPROM de 1KB, capacidad suficiente para el almacenaje de los datos de calibración, mientras que la STM32 tan solo tiene memoria flash y SRAM, donde la flash es utilizada para cargar el programa. La SRAM es una memoria volátil, pero poniendo una pila de 3V se puede mantener esa memoria sin borrar.

La velocidad del procesador indica cómo de rápido procesa una instrucción el micro. Esto es importante ya que los ESC's necesitan que se les envíe una señal cada 4 microsegundos. Cuanto mayor sea la velocidad, más operaciones se podrán hacer antes de tener que volver a mandar la señal a los ESC's, pero con 16MHz es suficiente.

En cuanto a programación, Arduino utiliza su propio IDE de Arduino, con programación en C y muchas herramientas que hacen que sea más sencilla la programación. STM32 también puede utilizar el IDE de Arduino mediante el uso de librerías, pero usar librerías reduce el rendimiento considerablemente por lo que se recomienda utilizar otro IDE y, por tanto, más complicado.

Por todo lo anteriormente expuesto se utilizará una placa Arduino, más concretamente una **Arduino UNO** ya que, aunque tenga un mayor peso (poco relevante) dispone de pines de prueba, muy útiles para la interconexión del cableado del dron.

6.4 Placa de seguimiento

La detección de color y el seguimiento de un objeto/persona es una carga de trabajo muy grande, por lo que es necesario un procesador muy potente, capaz de leer y procesar cada imagen captada por la cámara lo más rápido posible y así poder hacer el seguimiento del dron lo más fluido posible.

Es necesario conectar una cámara, por lo que será necesaria una conexión USB para mayor comodidad, ya que es posible conectarla a través de los pines TX y RX, pero sería necesario un adaptador de USB o una cámara especial que encarece el precio.

Analizando las opciones, claramente la placa más recomendable es la **Raspberry Pi 3B+** ya que tiene USB y la velocidad de procesamiento más alta (1.4GHz). El peso es similar a la Arduino UNO, pero despreciable (25g).

6.5 Giroscopio y Acelerómetro

La conexión en todos es posible hacerla mediante I2C o SPI. El giroscopio que integra el MPU-6050 es muy semejante al L3G4200D ya que tienen la misma precisión. En cuanto a acelerómetro el LIS331 tiene un rango de 24g mientras que el que incorpora el MPU-6050 solo tiene

hasta 16g. Para la aplicación que se le va a dar en este proyecto se utilizará 8g por lo que ambos nos proporcionan el rango necesario.

En definitiva, por la comodidad de tener ambos dispositivos en uno y el precio reducido, se ha decidido utilizar el **MPU-6050**. Además, éste incorpora un procesador interno que ejecuta los algoritmos de "MotionFusion" que combinan las mediciones de los sensores internamente, evitando así el uso de filtros externos.

6.6 Barómetro

Se ha decidido utilizar el Barómetro **MS5611** ya que, aunque la altura máxima y mínima no afectan en ninguno de los dos, el MS5611 es mucho más preciso (el triple) que el BMP180 y además incorpora un sensor de temperatura para mejores resultados.

6.7 Software tratamiento de imágenes

Pixy es una buena opción si se busca implementar en un robot ya que está optimizado para ello, pero un dron tiene sus limitaciones ya que hace falta mucha precisión y la cámara que incorpora Pixy no es óptima para distancias de más de 3 metros. Por ello se ha decidido escoger **OpenCV** para el desarrollo software de la detección de color y seguimiento de la persona ya que OpenCV permite implementarse en Linux, en la Raspberry Pi 3B+, teniendo una mayor capacidad de procesamiento que el procesador que incorpora Pixy.

6.8 Cámara

Se ha decidido utilizar una cámara Xiaomi YI puesto que es una cámara de tamaño reducido capaz de grabar a 1080p a un precio muy bajo en comparación con el resto del mercado que graban a la misma calidad.

7. Descripción de la solución

7.1 Control del dron

Para realizar el control del dron se utilizará la placa Arduino UNO, el sensor MPU-6050 y el barómetro MS5611. El controlador está basado en el controlador de Joop Brooking [BRK17].

7.1.1 Comunicación con los ESC's

Para la comunicación de la placa Arduino con lo ESC's se utiliza PWM, que se basa en enviar pulsos de una determinada achura.

Cada ESC necesita un pulso de una duración de 1000-2000µs cada 4ms para poder mandar la potencia necesaria a cada motor y que se mueva a la velocidad adecuada. Para ello se utilizan los pines 4,5,6 y 7 de la placa.

Lo primero que se realiza es la declaración de los pines como salidas en la placa.

```
DDRD |= B11110000;
```

Una vez declaradas como salidas, para crear una señal PWM, pondremos a 1 la salida y esperaremos mediante la función delayMicroseconds(X) el tiempo X en microsegundos que se corresponde con la duración del pulso que se quiere enviar al ESC. Una vez pasado el tiempo se envía un 0.

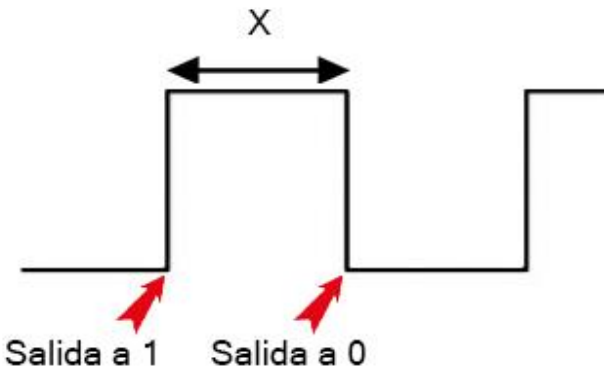

Figura 25: Descripción PWM

Con eso se consigue un pulso PWM sin necesidad de utilizar la función de Arduino analogWrite(Y) estando Y comprendida entre 0-255 que es el valor del duty cycle del pulso.

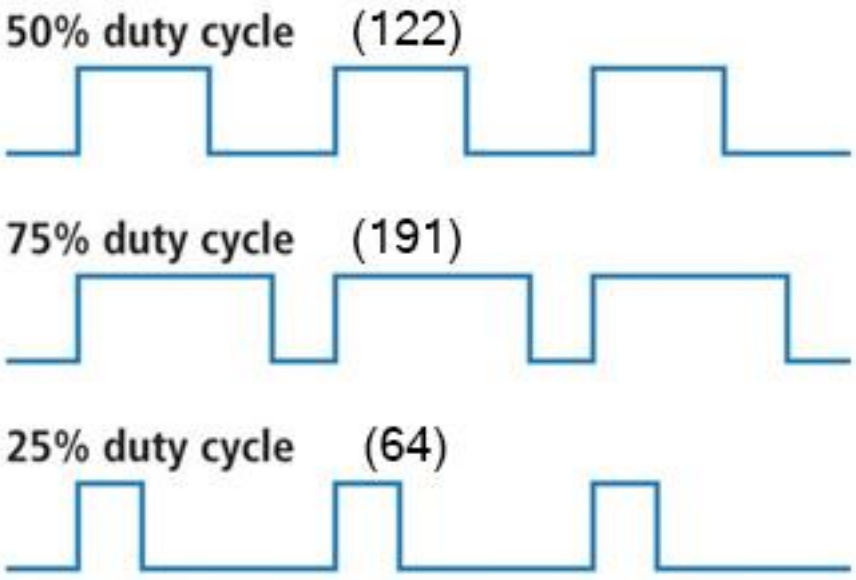

Figura 26: Descripción Duty Cycle

El envío de los pulsos a los ESC's se hace en la sección de "loop()" de Arduino, y este bucle dura exactamente 4ms, ya que es el periodo de actualización del pulso que necesitan los ESC's. Para asegurar que cada 4ms se manda el pulso correspondiente se utilizan temporizadores internos de la placa.

```
while(micros() - loop_timer < 4000);
loop_timer = micros();
```

La función micros() da el valor del tiempo que ha transcurrido desde que el dron se ha encendido. La variable loop_timer ha sido inicializada a 0 en el "setup()", por lo que con esto se consigue que cada vuelta que hace el bucle, espere hasta que se hayan consumido 4ms de vuelta.

Inmediatamente después de haber esperado ese tiempo, se ponen todas las salidas de los ESC's (4,5,6 y 7) a nivel alto y se guarda en una variable para cada ESC el tiempo que tiene que estar a nivel alto el ESC (1000-2000µs) sumado al tiempo de acumulación "loop_timer".

```
PORTD |= B11110000;
timer_channel_1 = esc_1 + loop_timer;
timer_channel_2 = esc_2 + loop_timer;
timer_channel_3 = esc_3 + loop_timer;
timer_channel_4 = esc_4 + loop_timer;
```

Una vez hecho eso, ahora el programa espera hasta que todas las salidas estén a 0 para continuar. Para que una salida se ponga a 0 tiene que cumplirse que el tiempo "timer_channel_x" sea menor o igual que el tiempo actual.

```
while(PORTD >= 16){
  esc_loop_timer = micros();
  if(timer_channel_1 <= esc_loop_timer)PORTD &= B11101111;
  if(timer_channel_2 <= esc_loop_timer)PORTD &= B11011111;
  if(timer_channel_3 <= esc_loop_timer)PORTD &= B10111111;
  if(timer_channel_4 <= esc_loop_timer)PORTD &= B01111111;
```

7.1.2 Recepción órdenes seguimiento

Para la recepción de las órdenes que se reciben desde la placa de seguimiento, se va a hacer uso de la comunicación PWM mandando pulsos de duración 1000-2000µs para indicar un extremo (-45º) u otro extremo (+45º) en cualquiera de los 3 ejes.

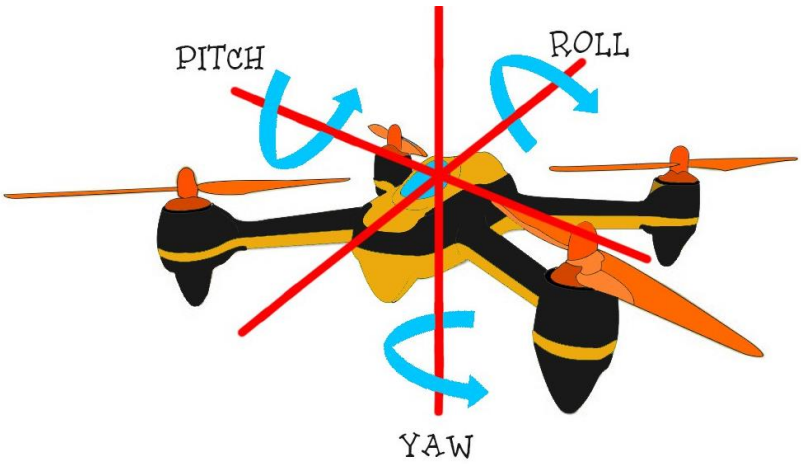

Figura 27: Dron 3 ejes

Para ello se utilizan los pines 8,9,10 y 11 de la placa Arduino como entradas y los pines 21,22,23 y 24 de la Raspberry como salidas. Los pines de Arduino se habilitarán para interrupciones.

```
PCICR |= (1 << PCIE0);
PCMSK0 |= (1 << PCINT0);
PCMSK0 |= (1 << PCINT1);
PCMSK0 |= (1 << PCINT2);
PCMSK0 |= (1 << PCINT3);
```

Una vez habilitadas las interrupciones en cada pin, cuando haya un cambio de estado en alguno de los pines, se entrará en una rutina de atención a interrupciones.

```
ISR(PCINT0_vect){
  current_time = micros();
  //Canal 1
  if(PINB & B00000001){
    if(last_channel_1 == 0){
      last_channel_1 = 1;
      timer_1 = current_time;
    }
  }
  else if(last_channel_1 == 1){
    last_channel_1 = 0;
    receiver_input[1] = current_time - timer_1;
  }
```

En ésta rutina ISR lo que se hace es mirar a qué pin pertenece la interrupción (“PINB & B0000xxxx”). Una vez se sabe el pin de donde viene la interrupción, si es la primera vez que entra, guarda el tiempo actual en una variable y la próxima vez que entre a la interrupción entrará en el “else if” restando el tiempo actual al guardado anteriormente. De ésta manera se consigue saber la longitud del pulso.

La ventaja de utilizar interrupciones en vez de recibir pulsos en el loop() del programa, es que mediante interrupciones no hay necesidad de sincronización con la Raspberry. En el caso de hacerlo en el loop() deberíamos sincronizar la recepción cada 4ms del pulso y el envío en la Raspberry de dicho pulso. Además, si se realiza en el loop(), se malgastarían 1000-2000µs, que no sobran.

7.1.3 PID

El dron tiene que ser estable, pero no basta con que los motores vayan a la misma velocidad, ya que es casi imposible que giren exactamente a la misma velocidad, y el peso del dron no es completamente simétrico. Esto lleva a la necesidad de un algoritmo de estabilización, y aquí se introduce la necesidad del algoritmo PID.

El algoritmo PID es un mecanismo de control por realimentación que calcula la desviación o error entre un valor medido y el valor que se quiere obtener, para aplicar una acción correctora que ajuste el proceso.

Está formado por 3 parámetros: Proporcional, Integral y Derivativo. El proporcional hace efecto al error inmediato, a lo que está ocurriendo. El integral aprende de pasados errores y corrige en función de lo que haya pasado. Por último, el derivativo predice el error, es decir, corrige el error que va a pasar.

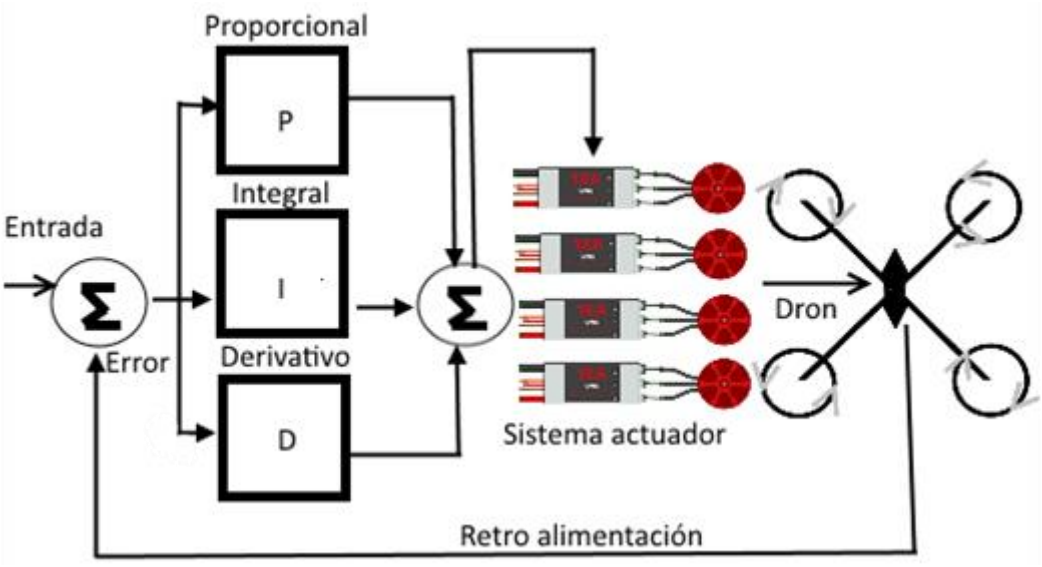

Figura 28: Esquema PID

7.1.3.1 Proporcional

Trata del producto entre la señal de error y la constante proporcional para lograr que el error en estado estacionario se aproxime a cero.

$$K_p e(t)$$

Figura 29: Valor Proporcional

En el dron podría decirse que ésta función es el valor del PID que se encarga de la estabilidad en el dron. A mayor valor de la K_p el dron será cada vez más estable, pero hay un límite en el que empieza a sobreoscilar. La sobreoscilación puede ser de hasta un 30%, a partir de ahí el dron podría llegar a ser peligroso debido a su posible descontrol, aunque no es recomendable que tengo ni siquiera oscilación.

7.1.3.2 Integral

La función Integral tiene como propósito disminuir y eliminar el error en estado estacionario, provocado por perturbaciones exteriores y los cuales no pueden ser corregidos por la proporcional. Esta función hace efecto cuando se crea una desviación entre la variable y el punto de consigna, integrando esa desviación y sumándosela a la proporcional.

$$K_i \int_0^t e(\tau) d\tau$$

Figura 30: Valor Integral

Desde el punto de vista del dron, este valor nos indica la velocidad con la que la acción proporcional se repite. Esto consigue que el movimiento del dron sea mucho más fluido y no de tirones muy bruscos.

7.1.3.3 Derivativo

El derivativo actúa cuando hay un cambio en el valor absoluto del error, asique si el error es constante tan solo actuaran el proporcional y el integral. El error es la diferencia que hay entre el punto que se quiere que esté y en el que realmente está. La función de la acción derivativa es mantener el error al mínimo corrigiéndolo proporcionalmente con la misma velocidad con la que se produce y de esta manera evita que el error vaya a más.

$$K_d \frac{de(t)}{dt}$$

Figura 31: Valor Derivativo

Juntando estos 3 algoritmos en uno se consigue que el dron mantenga una estabilidad muy precisa y sin movimientos bruscos.

Para juntarlos basta con sumarlos y se obtiene la salida del controlador PID:

$$\mathrm{y(t)} = \mathrm{MV(t)} = K_p e(t) + K_i \int_0^t e(\tau)\,d\tau + K_d \frac{de}{dt}$$

Figura 32: Algoritmo PID

7.2 Reconocimiento de color

Para el reconocimiento de color se hará uso de la librería OpenCV con ayuda del IDE Eclipse, en Linux. El código será ejecutado por la Raspberry Pi. El lenguaje de programación utilizado es C++.

Como se va a hacer uso de la librería OpenCV, lo primero que se debe hacer es declarar los paquetes de la librería que queremos utilizar. Para la detección de color han sido necesarios dos paquetes:

```
#include <iostream>
#include "opencv2/highgui/highgui.hpp"
#include "opencv2/imgproc/imgproc.hpp"
```

Como puede verse se han utilizado “highgui.hpp” y “imgproc.hpp”, además del paquete “iostream” de C++.

Hay que declarar una variable tipo “VideoCapture”, perteneciente a la librería OpenCV, que captará imágenes desde la cámara cuando se quiera. El nombre de la variable será “cap”. Si al llamar a la función “isOpened()” dentro de “cap” devuelve *false*, el programa mandará una señal de error a la Arduino, haciendo que el LED conectado a ella se encienda indicando un error.

```
VideoCapture cap(0); //Abrir camara
if ( !cap.isOpened() )
{
     gpio26->setval_gpio("1");
     return -1;
}
```

Una vez inicializada la cámara, se crea un loop infinito con un “while(true)”, para leer y procesar repetidamente cada imagen. Las imágenes se guardan en matrices, por lo que se guardan en una variable al ser leídas desde cámara.

```
Mat imgOriginal;
cap.read(imgOriginal);
```

Por defecto la función “read()” de OpenCV lee la imagen en formato BGR, pero para la detección de color el formato óptimo es HSV. Por ello, se transforma la matriz.

```
Mat imgHSV;
cvtColor(imgOriginal, imgHSV, COLOR_BGR2HSV);
```

Con la imagen en formato HSV se crea otra matriz que solo contendrá 0 o 255, siendo 255 únicamente cuando dicho pixel de la matriz es del color por el que se ha filtrado.

Para filtrar un color se tienen 3 filtros: Hue, Saturación y Valor. Cada uno de ellos tiene un valor máximo y un valor mínimo, de manera que se obtienen rangos de cada variable ya que un color puede variar un poco dependiendo de la luz que le dé al objeto.

- **Hue:** Indica el color. Se le ha dado un valor mínimo y máximo en este proyecto de 30 y 32 respectivamente.
- **Saturación:** Indica la cantidad de color que se mezcla con el color blanco. Se le ha dado un valor mínimo y máximo en este proyecto de 60 y 102 respectivamente.
- **Valor:** Indica la cantidad de color que se mezcla con el color negro. Se le ha dado un valor mínimo y máximo en este proyecto de 110 y 220 respectivamente.

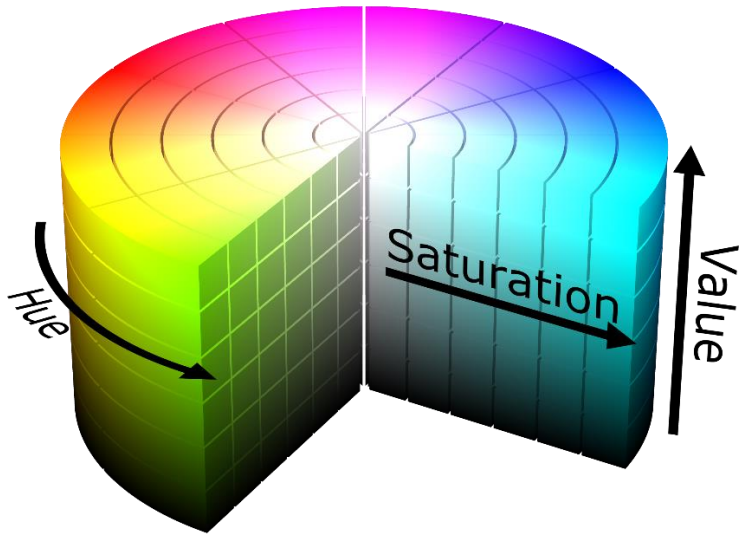

Figura 33: Representación HSV

Para crear la matriz de filtro se hará uso de la función “inRange” de OpenCV. Esta función crea una matriz (imgFiltro) con 255 en las posiciones donde el valor HSV de la matriz de entrada (imgHSV) este dentro del rango seleccionado mínimo y máximo.

```
Mat imgFiltro;
inRange(imgHSV, Scalar(iLowH, iLowS, iLowV), ...
 ... Scalar(iHighH, iHighS, iHighV), imgFiltro);
```

Ésta matriz “imgFiltro” ya contiene la detección del color que se busca, pero las cámaras no captan la realidad al 100%, luego pueden aparecer puntos o zonas donde aparezca información en esta matriz de filtrado. Para evitar esto, existen dos tipos de morfología: La erosión y la dilatación. A partir de estos dos tipos, se pueden combinar consiguiendo diferentes resultados.

- Dilatación: Esta operación consiste en convolucionar una matriz “A” con un kernel “B”. Mientras que el kernel pasa a través de la matriz, se computa el máximo valor de pixel superpuesto por el kernel y se sustituye el valor del pixel de la matriz por su máximo valor. Esto hace que la región crezca por lo que su efecto es de agrandamiento.

Figura 34: Dilatación

- Erosión: Al igual que la dilatación realiza una convolución, pero en vez de computar el máximo valor de pixel, hace el mínimo. De esta manera, se consigue que la región decrezca, luego es un efecto de disminución.

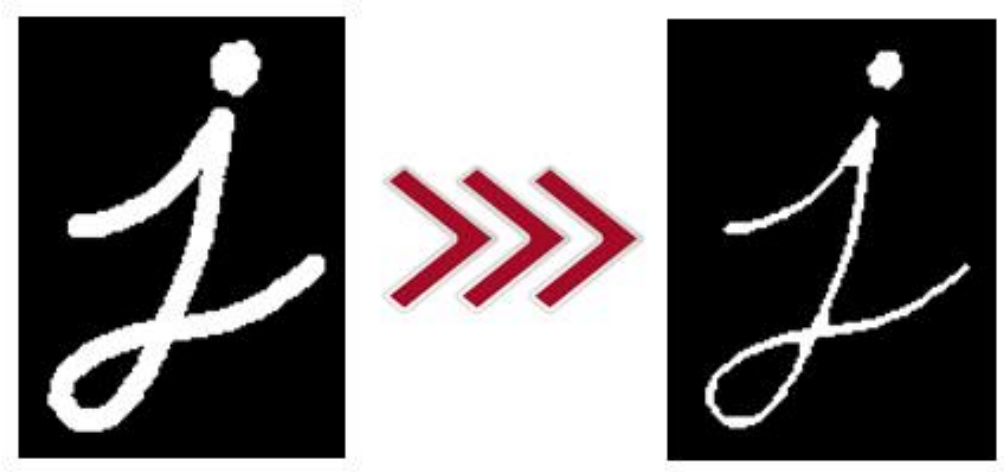

Figura 35: Erosión

Con estas dos morfologías se puede aplicar morfología de apertura (erosionar y después dilatar) y de cierre (dilatar y después erosionar). Primero se aplica morfología de apertura para dejar solo la zona que más predomina de “255” en la matriz (es decir el objeto que se quiere detectar). Seguidamente se aplica morfología de cierre para rellenar los huecos que ha podido crear la de apertura en el objeto principal.

```
//apertura morfológica
erode(imgFiltro, imgFiltro, getStructuringElement(MORPH_ELLIPSE, Size(5, 5)));
dilate(imgFiltro, imgFiltro, getStructuringElement(MORPH_ELLIPSE, Size(5, 5)));

//cierre morfológico
dilate(imgFiltro, imgFiltro, getStructuringElement(MORPH_ELLIPSE, Size(5, 5)));
erode(imgFiltro, imgFiltro, getStructuringElement(MORPH_ELLIPSE, Size(5, 5)));
```

El resultado que se obtiene al procesar una imagen y mostrarla es una imagen en negro, con el chaleco en blanco, marcando que el chaleco es el color que se desea.

Figura 36: Detección chaleco

7.3 Seguimiento

Para realizar el seguimiento del color que se ha detectado anteriormente se hará uso de los “momentos”. La función “moments(Mat source)” incluida en la librería OpenCV ayuda a detectar si ha habido cambios en la imagen respecto a una imagen anterior. Esta función devuelve los momentos espaciales hasta 3er orden sobre una variable de tipo “Moments”.

```
Moments oMoments = moments(imgThresholded);

double dM01 = oMoments.m01;
double dM10 = oMoments.m10;
double dArea = oMoments.m00;
```

Para calcular la posición del chaleco se utiliza el primer orden de los momentos. El plano es de dos dimensiones asique es necesario un eje X y un eje Y. Para el cálculo de la coordenada en X se divide el momento espacial de orden 1 (dm10) sobre el eje X entre el momento central de orden 0 (dArea). De la misma manera, la coordenada Y se calcula dividiendo el momento espacial de orden 1 (dm01) sobre el eje Y entre el momento central de orden 0.

```
int posX = dM10 / dArea;
int posY = dM01 / dArea;
```

Con esto se consigue un seguimiento del objeto, guardando y comparando el anterior punto con el actual.

Para hacer que el dron realice dicho seguimiento se hará uso de los pines 22,23,24 y 25 de la Raspberry que irán conectados a las entradas 8,9,10 y 11 como se explicó anteriormente. Por ello estos pines de la Raspberry son configurados como salidas.

El programa comprueba hacia qué lado se ha desplazado el objetivo, para mandar la orden al dron. Si se mueve en el plano hacia la derecha o izquierda, tendrá que aplicar roll.

```
if(posX>posXant)
{
  gpio22.setval_gpio("1");
  usleep(1600);
  gpio22.setval_gpio("0");
}

if(posX<posXant)
{
  gpio22.setval_gpio("1");
  usleep(1400);
  gpio22.setval_gpio("0");
}
```

La función “gpioXX.setval_gpio()” es de una librería de código abierto para Raspbian [GPIO12]. Mediante el uso de esta función, es posible acceder al estado de los pines de la Raspberry de forma segura.

Se manda una señal PWM de duración 1000-2000µs como se ha explicado anteriormente.

Para saber si la persona se aleja o se acerca al dron se calcula la cantidad de "1" hay en la matriz, comparada con una matriz de referencia. Si es mayor quiere decir que la persona se ha acercado, luego tiene que ir hacia atrás el dron. Al contrario, si es menor, quiere decir que la persona se ha alejado, luego deberá ir hacia delante el dron. La matriz de referencia está diseñada para que el dron mantenga una distancia segura de 7 metros respecto de la persona.

```
int distNueva = count(arrayImagen,arrayImagen+sizeof(arrayImagen),255);

if(distNueva>distRef)
{
  gpio23.setval_gpio("1");
  usleep(1400);
  gpio23.setval_gpio("0");
}

if(distNueva<distRef)
{
  gpio23.setval_gpio("1");
  usleep(1600);
  gpio23.setval_gpio("0");
}
```

7.4 Mantenimiento de altitud

Como añadido, se ha diseñado un código que mediante el uso de un barómetro es posible mantener una altura en el dron, con una desviación de ±0.5m. Éste código se ha integrado en el código de control de vuelo en Arduino.

Al encender el dron en la fase de setup() se toman varias muestras de presión de referencia, para obtener mayor precisión.

```
suma=0;
for(i=0;i<10;i++)
{
  pressure=baro.getPressure();
  ayuda=(double)pressure;
  suma+=ayuda;
}
pini=(suma/10);
```

En el loop, se toma una muestra de presión cada vuelta, realizando las conversiones necesarias para transformar el valor a una altura. Si esa altura es menor de 1.8 metros, la variable throttle será incrementada en 100. Si por el contrario la altura es mayor de 2.2 metros, se disminuirá dicha variable en 100.

```
presion = (double)baro.getPressure();
temperatura=(double)baro.getTemperature()+273.0;
altura=44330.0*(1-pow(presion/pini,1/5.255));
if(altura<1.8)
{
  throttle=throttle+100;
}
else if(altura<2.2)
{
  throttle=throttle-100;
}
```

El efecto que produce la manipulación de la variable throttle es mayor o menor potencia en los motores. De esa manera se consigue la estabilidad de altura del dron.

8. Metodología

8.1 Descripción de tareas

Se ha dividido el proyecto en 4 paquetes principales de trabajo, los cuales se desarrollan en más paquetes. El primer paquete trata de la gestión y seguimiento del paquete. El segundo trata sobre el aprendizaje sobre el dron y tratamiento de imágenes. En el tercer paquete se centra en el desarrollo del proyecto y en el cuarto, en la documentación.

Tabla 6: Fase 1

Fase 1	Inicio	Final	Duración
P1 Gestión del proyecto *Seguimiento para el desarrollo del proyecto.*	22-01-18	5-09-18	227 días
P1.1 Definición del proyecto	22-01-18	22-01-18	1 día
P1.2 Seguimiento del proyecto *Vigilancia y supervisión del desarrollo del proyecto.*	22-01-18	5-09-18	227 días

Tabla 7: Fase 2

Fase 2	Inicio	Final	Duración
P2 Organización del proyecto *Estudio de lo necesario para la realización del proyecto.*	23-01-18	13-03-18	50 días
P2.1 Información del proyecto	23-01-18	10-02-18	19 días
P2.1.1 Información sobre drones *Buscar información sobre UAV para entender los diferentes tipos de drones.*	23-01-18	31-01-18	9 días
P2.1.2 Información sobre software de reconocimiento *Búsqueda de información acerca de reconocimiento de color y seguimiento.*	1-02-18	10-02-18	10 días
P2.2 Búsqueda de proyectos semejantes *Análisis de proyectos ya realizados semejantes que sirvan como guía.*	11-02-18	19-02-18	9 días
P2.3 Estudio de herramientas software *Estudio sobre el posible software a utilizar para el desarrollo de la detección de color y seguimiento.*	20-02-18	5-03-18	14 días
P2.4 Estudio de los componentes *Realizar un análisis de los componentes que necesita el dron.*	6-03-18	13-03-18	8 días

Tabla 8: Fase 3

Fase 3	Inicio	Final	Duración
P3 Desarrollo del proyecto *Realización de las diferentes partes del proyecto.*	14-03-18	31-05-18	79 días
P3.1 Selección de elementos	14-03-18	29-03-18	16 días
P3.1.1 Hardware *Selección y adquisición de los componentes hardware que irán montados en el dron.*	14-03-18	21-03-18	8 días
P3.1.2 Software *Selección del software a utilizar para el desarrollo software del dron.*	22-03-18	29-03-18	8 días
P3.2 Desarrollo software dron *Desarrollo del software para el control del dron.*	30-03-18	26-04-18	28 días
P3.3 Montaje del dron *Montaje de las piezas del dron y soldaduras.*	27-04-18	30-04-18	4 días
P3.4 Desarrollo software seguimiento autónomo	1-05-18	23-05-18	23 días
P3.4.1 Desarrollo software detección de color *Crear un programa que detecte mediante una cámara, un color especifico.*	1-05-18	12-05-18	12 días
P3.4.2 Desarrollo software seguimiento *Diseño de un software capaz de seguir el objeto detectado mediante el color.*	13-05-18	23-05-18	11 días
P3.5 Interconexión software seguimiento con dron *Realizar la conexión del software de seguimiento con el software del control del dron para que el dron siga al objetivo.*	24-05-18	31-05-18	8 días

Tabla 9: Fase 4

Fase 4	Inicio	Final	Duración
P4 Documentación del proyecto *Escritura del documento y presentación.*	1-06-18	5-09-18	97 días
P4.1 Documentación *Redacción del proyecto.*	1-06-18	23-07-18	53 días
P4.2 Presentación Preparación de la presentación oral.	24-07-18	5-09-18	44 días

8.2 Hitos del proyecto

En la tabla se describen los hitos a alcanzar en este proyecto.

Tabla 10: Hitos

Hitos	Fecha a cumplir
H1: Inicio del proyecto	23-01-18
H2: Funcionamiento motores	30-04-18
H3: Reconocimiento de color	12-05-18
H4: Seguimiento objeto	23-05-18
H5: Pruebas Hardware y Software	31-05-18
H6: Vuelo del dron	31-05-18
H7: Finalización documentación	23-07-18
H8: Finalización presentación	5-09-18
H9: Fin del proyecto	5-09-18

8.3 Diagrama de Gantt

A continuación, se muestra el diagrama de Gantt, en el que se observan todos los paquetes de trabajo con sus respectivas fechas de inicio y fin y su duración.

DIAGRAMA DE GANTT

	Modo de	EDT	Nombre de tarea	Duración	Comienzo	Fin	Predecesora
1		**P1**	**Gestión del proyecto**	**227 días**	**lun 22/01/18**	**mié 05/09/18**	
2		P1.1	Definición del proyecto	1 día	lun 22/01/18	lun 22/01/18	
3		P1.2	Seguimiento del proyecto	227 días	lun 22/01/18	mié 05/09/18	
4			Inicio del proyecto	0 días	lun 22/01/18	lun 22/01/18	
5		**P2**	**Organización del proyecto**	**50 días**	**mar 23/01/18**	**mar 13/03/18**	
6		**P2.1**	**Información del proyecto**	**19 días**	**mar 23/01/18**	**sáb 10/02/18**	
7		P2.1.1		9 días	**mar 23/01/18**	mié 31/01/18	
8		P2.1.2		10 días	jue 01/02/18	sáb 10/02/18	7
9		P2.2	Búsqueda de proyectos semejantes	9 días	dom 11/02/18	lun 19/02/18	6
10		P2.3	Estudio de herramientas software	14 días	mar 20/02/18	lun 05/03/18	9
11		P2.4	Estudio de los componentes	8 días	mar 06/03/18	mar 13/03/18	10
12		**P3**	**Desarrollo del proyecto**	**79 días**	**mié 14/03/18**	**jue 31/05/18**	**5**
13		**P3.1**	**Selección de elementos**	**16 días**	**mié 14/03/18**	**jue 29/03/18**	
14		P3.1.1	Hardware	8 días	**mié 14/03/18**	mié 21/03/18	
15		P3.1.2	Software	8 días	jue 22/03/18	jue 29/03/18	14
16		P3.2	Desarrollo software dron	28 días	vie 30/03/18	jue 26/04/18	13
17		P3.3	Montaje del dron	4 días	vie 27/04/18	lun 30/04/18	16
18			Funcionamiento motores	0 días	lun 30/04/18	lun 30/04/18	
19		**P3.4**	**Desarrollo software seguimiento autónomo**	**23 días**	**mar 01/05/18**	**mié 23/05/18**	**17**
20		P3.4.1	Desarrollo software detección de color	12 días	**mar 01/05/18**	sáb 12/05/18	
21			Reconocimiento de color	0 días	sáb 12/05/18	sáb 12/05/18	
22		P3.4.2	Desarrollo software seguimiento	11 días	dom 13/05/18	mié 23/05/18	20
23			Seguimiento objeto	0 días	mié 23/05/18	mié 23/05/18	
24		P3.5	Interconexión software seguimiento con dron	8 días	jue 24/05/18	jue 31/05/18	19
25			Pruebas Hardware y Software	0 días	jue 31/05/18	jue 31/05/18	
26			Vuelo del dron	0 días	jue 31/05/18	jue 31/05/18	
27		**P4**	**Documentación del proyecto**	**97 días**	**vie 01/06/18**	**mié 05/09/18**	**12**
28		P4.1	Documentación	53 días	**vie 01/06/18**	lun 23/07/18	
29			Finalización documentación	0 días	lun 23/07/18	lun 23/07/18	
30		P4.2	Presentación	44 días	mar 24/07/18	mié 05/09/18	28
31			Finalización presentación	0 días	mié 05/09/18	mié 05/09/18	
32			Fin del proyecto	0 días	mié 05/09/18	mié 05/09/18	

Figura 37: Datos del diagrama de Gantt

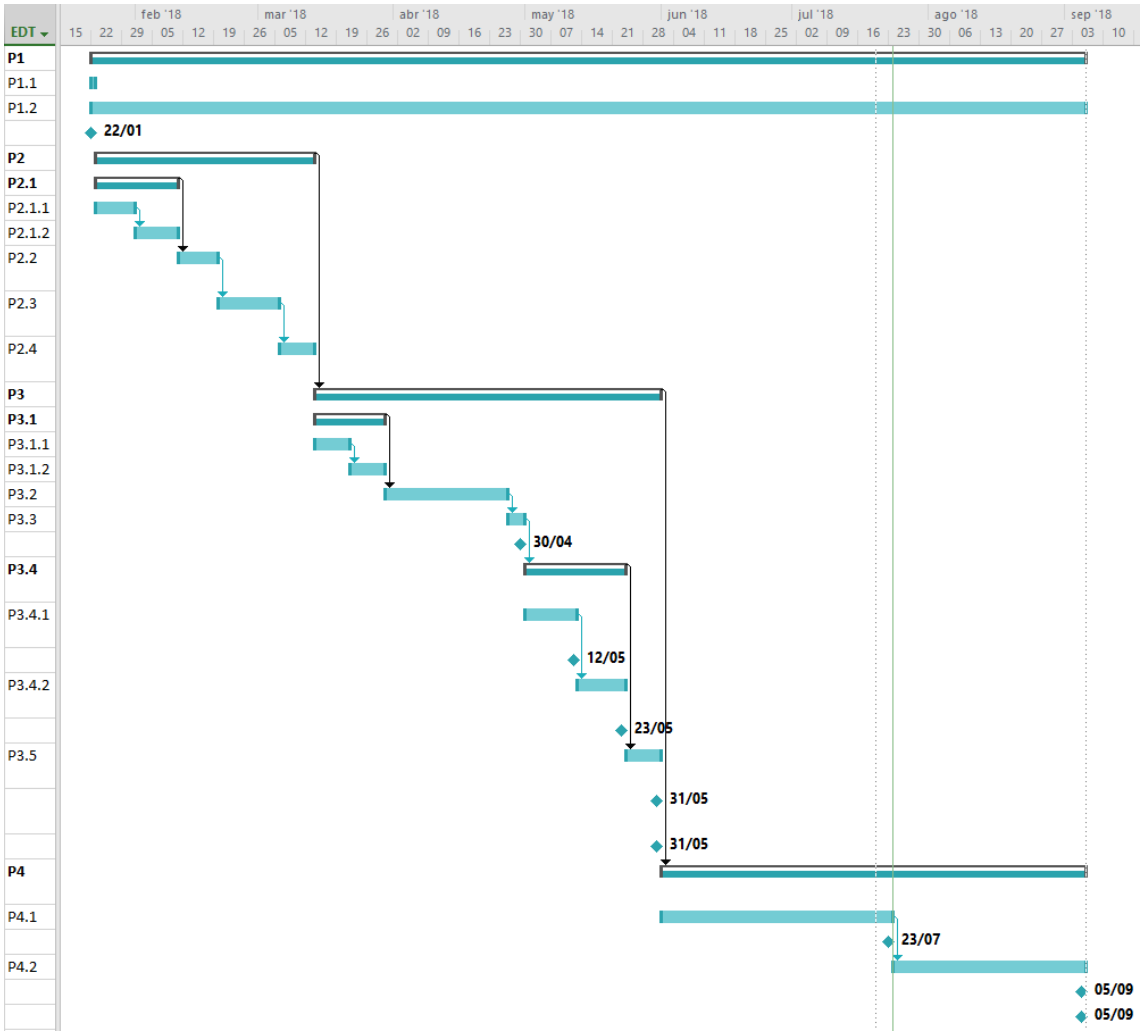

Figura 38: Diagrama de Gantt

9. Presupuesto

En este apartado se realizará el presupuesto del proyecto de manera que se verá reflejados los gastos del proyecto.

9.1 Recursos humanos

Para este proyecto han sido necesarios dos participantes, el director del proyecto y un estudiante de ingeniería de telecomunicaciones. En la siguiente tabla se ve reflejado el gasto que suponen:

Tabla 11: Recursos humanos

Concepto	Uso (horas)	€/hora	Total
Director del proyecto	50	60	3.000€
Estudiante ingeniería	300	40	12.000€
Subtotal			**15.000€**

9.2 Amortizaciones

En la tabla se ven reflejados los gastos de amortización con un uso mensual.

Tabla 12: Amortizaciones

Concepto	Uso (Meses)	Coste	Vida Útil (meses)	Total
Ordenador	7	1.000€	48	145,80€
Office 2016	3	300€	12	75€
Subtotal				**220,80€**

El uso del IDE de Arduino y Eclipse es gratuito, así como el uso de la librería OpenCV de código abierto.

9.3 Gastos generales

A continuación, se muestran los gastos generales del proyecto.

Tabla 13: Gastos generales

Concepto	Coste	Cantidad	Total
Raspberry Pi 3B+	35,66€	1	35,66€
Arduino UNO R3	21,72€	1	21,72€
Frame DJI F450	21,62€	1	21,62€
A2212 1000KV Brushless Outrunner Motor	6,30€	4	25,20€
30A ESC Brushless	4,31€	4	17,24€
Batería 2800mAh	17,34€	2	34,68€
MPU-6050	0,79€	1	0,79€
MS5611	5,06€	1	5,06€
Hélices 10*4.5	1€	10	10€
Cámara Xiaomi YI	41,83€	1	41,83€
Gimbal Cámara 2-axis	36,15€	1	36,15€
Cableado+Soldadura	10€	1	10€
Cable conexión cámara NTSC	2€	1	2€
Cable USB 2.0	5€	1	5€
Subtotal			**266,95€**

9.4 Resumen de gastos

Tabla 14: Gastos totales

Concepto	Gasto
Recursos humanos	15.000€
Amortizaciones	220,80€
Gastos generales	266,95€
TOTAL	**15.487,75€**

10. Conclusiones

Como se ha visto anteriormente, la realización software del seguimiento de la persona mediante un color específico ha sido desarrollada cumpliendo los objetivos marcados.

El control del dron ha sido posible realizarlo de manera que no sea necesaria la interacción con el usuario gracias a la interconexión de ambas placas. Además, que el dron sea capaz de mantener una altura estable, evita preocupaciones por posibles golpes.

También cabe destacar que para la detección de color del dron se ha utilizado una placa aparte del controlador de vuelo para mayor precisión. Se han rechazado así mismo otras opciones más caras y más fiables, debido a que el efecto que producirían sobre el dron es muy pequeño.

Gracias al desarrollo de este proyecto se ha demostrado como el presupuesto para su realización es relativamente bajo. Este dron puede sustituir a personas en sus trabajos y por tanto recortar en gastos. Y no solo personas, sino vehículos como se ha descrito anteriormente.

Finalmente, los beneficios que este proyecto aporta a la sociedad son muchos, y en 5 años este tipo de drones serán lo que más se vea en las calles. En un futuro muy cercano, estos aparatos serán como los coches, los ordenadores o incluso el selfi; algo conocido por todo el mundo y que la gran mayoría utiliza o necesita.

11. Bibliografía

[BRSH15] https://www.4aspas.com.ar/motores-brush-vs-brushless-cual-es-la-diferencia/

[ESC17] http://fpvmax.com/2016/12/21/variador-electronico-esc-funciona/

[STM18] https://www.st.com/en/microcontrollers/stm32f103c8.html

[PI18] https://www.raspberrypi.org/products/raspberry-pi-3-model-b-plus/

[ARD18] https://www.arduino.cc/

[IMU16] https://www.sparkfun.com/pages/accel_gyro_guide

[MPU18] https://www.invensense.com/products/motion-tracking/6-axis/mpu-6050/

[BMP18] https://www.adafruit.com/product/1603

[MS18] http://www.te.com/usa-en/product-CAT-BLPS0036.html

[OCV10] https://opencv.org/

[PIX13] https://pixycam.com/

[GPIO12] http://www.hertaville.com/introduction-to-accessing-the-raspberry-pis-gpio-in-c.html

[BRK17] http://www.brokking.net/ymfc-al_main.html

12. Anexos

Código rutina atención de interrupciones de la comunicación entre la Arduino y la Raspberry Pi:

```
ISR(PCINT0_vect){
  current_time = micros();
  //Canal 1=========================================
  if(PINB & B00000001){
    if(last_channel_1 == 0){
      last_channel_1 = 1;
      timer_1 = current_time;
    }
  }
  else if(last_channel_1 == 1){
    last_channel_1 = 0;
    receiver_input[1] = current_time - timer_1;
  }
  //Canal 2=========================================
  if(PINB & B00000010 ){
    if(last_channel_2 == 0){
      last_channel_2 = 1;
      timer_2 = current_time;
    }
  }
  else if(last_channel_2 == 1){
    last_channel_2 = 0;
    receiver_input[2] = current_time - timer_2;
  }
  //Canal 3=========================================
  if(PINB & B00000100 ){
    if(last_channel_3 == 0){
      last_channel_3 = 1;
      timer_3 = current_time;
    }
  }
  else if(last_channel_3 == 1){
    last_channel_3 = 0;
    receiver_input[3] = current_time - timer_3;

  }
  //Canal 4=========================================
  if(PINB & B00001000 ){
    if(last_channel_4 == 0){
      last_channel_4 = 1;
      timer_4 = current_time;
    }
  }
  else if(last_channel_4 == 1){
    last_channel_4 = 0;
    receiver_input[4] = current_time - timer_4;
  }
}
```

Cálculos del PID:

```
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;

  //Pitch calculations
  pid_error_temp = gyro_pitch_input - pid_pitch_setpoint;
  pid_i_mem_pitch += pid_i_gain_pitch * pid_error_temp;
  if(pid_i_mem_pitch > pid_max_pitch)pid_i_mem_pitch = pid_max_pitch;
  else if(pid_i_mem_pitch < pid_max_pitch * -1)pid_i_mem_pitch = pid_max_pitch * -1;

  pid_output_pitch = pid_p_gain_pitch * pid_error_temp + pid_i_mem_pitch +
  pid_d_gain_pitch * (pid_error_temp - pid_last_pitch_d_error);
  if(pid_output_pitch > pid_max_pitch)pid_output_pitch = pid_max_pitch;
  else if(pid_output_pitch < pid_max_pitch * -1)pid_output_pitch = pid_max_pitch * -1;

  pid_last_pitch_d_error = pid_error_temp;

  //Yaw calculations
  pid_error_temp = gyro_yaw_input - pid_yaw_setpoint;
  pid_i_mem_yaw += pid_i_gain_yaw * pid_error_temp;
  if(pid_i_mem_yaw > pid_max_yaw)pid_i_mem_yaw = pid_max_yaw;
  else if(pid_i_mem_yaw < pid_max_yaw * -1)pid_i_mem_yaw = pid_max_yaw * -1;

  pid_output_yaw = pid_p_gain_yaw * pid_error_temp + pid_i_mem_yaw +
  pid_d_gain_yaw * (pid_error_temp - pid_last_yaw_d_error);
  if(pid_output_yaw > pid_max_yaw)pid_output_yaw = pid_max_yaw;
  else if(pid_output_yaw < pid_max_yaw * -1)pid_output_yaw = pid_max_yaw * -1;

  pid_last_yaw_d_error = pid_error_temp;
}
```

Cálculo de la potencia necesaria para cada motor con la implementación del ajuste del barómetro:

```
throttle = receiver_input_channel_3;

presion = (double)baro.getPressure();
temperatura=(double)baro.getTemperature()+273.0;
altura=44330.0*(1-pow(presion/pini,1/5.255));
if(altura<1.8)
{
  throttle=throttle+100;
}
else if(altura<2.2)
{
  throttle=throttle-100;
}

if (start == 2){
  if (throttle > 1800) throttle = 1800;
  esc_1 = throttle - pid_output_pitch + pid_output_roll - pid_output_yaw;
  esc_2 = throttle + pid_output_pitch + pid_output_roll + pid_output_yaw;
  esc_3 = throttle + pid_output_pitch - pid_output_roll - pid_output_yaw;
  esc_4 = throttle - pid_output_pitch - pid_output_roll + pid_output_yaw;

  if (battery_voltage < 1240 && battery_voltage > 800){
    esc_1 += esc_1 * ((1240 - battery_voltage)/(float)3500);
    esc_2 += esc_2 * ((1240 - battery_voltage)/(float)3500);
    esc_3 += esc_3 * ((1240 - battery_voltage)/(float)3500);
    esc_4 += esc_4 * ((1240 - battery_voltage)/(float)3500);
  }

  if (esc_1 < 1100) esc_1 = 1100;
  if (esc_2 < 1100) esc_2 = 1100;
  if (esc_3 < 1100) esc_3 = 1100;
  if (esc_4 < 1100) esc_4 = 1100;

  if(esc_1 > 2000)esc_1 = 2000;
  if(esc_2 > 2000)esc_2 = 2000;
  if(esc_3 > 2000)esc_3 = 2000;
  if(esc_4 > 2000)esc_4 = 2000;
}

else{
  esc_1 = 1000;
  esc_2 = 1000;
  esc_3 = 1000;
  esc_4 = 1000;
}
```