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 DE UN DRONE MEDIANTE GESTOS VISUALMENTE RECONOCIDOS

Alumno/Alumna: San José, Ibarra, Irene

Director/Directora: Espinosa, Acereda, Koldo

Curso: 2017 - 2018

Fecha: Bilbao, 19 de junio 2018

Resumen

En este proyecto se presenta una solución para controlar un Vehículo Aéreo No Tripulado (VANT), o comúnmente llamado drone, mediante gestos humanos visualmente reconocibles. Se ha desarrollado un programa que es capaz de controlar cámaras y detectar los gestos de una persona que se encuentre en una imagen, analizarlos e interpretarlos para controlar los movimientos del drone.

Para el reconocimiento de gestos del programa, se han empleado algunas técnicas habituales de detección y reconocimiento de objetos en imágenes. Cabe destacar que se han utilizado dos cámaras, en lugar de una única, para simular la visión en estéreo, como la de los humanos. Con esto se consigue hacer uso de características de profundidad que agilizan el reconocimiento y evitan las limitaciones del procesamiento de una única imagen.

Laburpena

Proiektu honek Tripulaziorik Gabeko Aireko Ibilgailuak (TGAIak), edo dronak, ezagutzen direnez, bisualki ezaguterraz keinukin kontrolatzeko metodoa aurkezten du. Dronako mugimenduak kontrolatzeko programa garatu da. Honek, kamarak erabiliz, keinuak antzematen, analizatzen eta interpretatzen ditu.

Keinuak aitortzeko, argazkitan objektu antzemateko eta aitortzeko teknika ohikoak erabili dira. Soluzio honek esterofonikoko ikusmena, pertsonen bezalakoa, alegia egiten du, bi kamara erabiliz. Honela, eszenako sakontasuneko ezaugarriak erabil daitezke, ezagutza bizkortzeko eta argazki bakarra prozesatzea baino informazio gehiago lortzeko.

Summary

This project explains a new method for driving Unmanned Aerial Targets (UAT), or drones, as they are usually called, through visually recognized human gestures. The main part consists on a program that controls cameras and detects, analyzes and interprets human gestures in images to control the drone movements.

For the gesture recognition, typical object detection and recognition techniques in images were used. Furthermore, this method uses two cameras, instead of just one, to simulate stereo vision, similar to human vision, which allows the use of depth features of the scene to increase the recognition speed and to avoid single image processing limitations.

Índice

Tabla de Figuras

Tabla de Contenido

1 Introducción

En este proyecto se va a desarrollar una solución en la que se pueda controlar un VANT mediante gestos humanos de forma visual. Para ello, se va a hacer uso de cámaras y métodos de detección de objetos en imágenes para el reconocimiento de gestos, y se adaptará la información para su posterior interpretación por un drone volador.

Este Trabajo de Fin de Grado se basará en el desarrollo de un proyecto software. Es decir, va a contar con las partes esenciales de éste, como la definición de especificaciones, diseño del software, codificación, pruebas e integración. A esto se le añadirá una fase previa de formación.

En la mencionada fase previa, se van a estudiar distintas técnicas de detección y sus posibilidades de implantación en este caso. También va a ser necesario familiarizarse con el funcionamiento del control de drones, las señales que intervienen en la comunicación, o los parámetros que es necesario tener en cuenta.

2 Contexto

Las técnicas de detección y seguimiento de objetos han supuesto un gran avance en el mundo tecnológico en los últimos años. Algunos de los usos que más se oyen actualmente son la realidad aumentada o el reconocimiento facial. Sin embargo, hace más de una década que se empezaron a utilizar en la vida cotidiana. El ámbito de uso principal fue el ocio, como es el caso de las cámaras de las consolas de videojuegos. Estas técnicas han ido evolucionando en complejidad, velocidad y precisión, y siguen en aumento.

Por otra parte, cada vez se extiende más el uso de drones voladores para determinadas tareas [1], como pueden ser vigilancia y seguridad en las carreteras, agricultura, topografía u ocio. Sin embargo, sus capacidades se ven limitadas, en parte por la legislación, y también por sus funcionalidades actuales. Integrando tecnologías actuales, como métodos de detección de objetos, en estos dispositivos se podría lograr mayor autonomía y beneficiarnos del amplio abanico de posibilidades que aportan. Por ejemplo, en casos en los que la señal de radio entre el mando y el receptor es débil; en caso de que se pierda el control del equipo mediante el mando de radiocontrol, estas técnicas permitirían que otro piloto sin mando pudiera recuperarlo. O llegar a conseguir un control cooperativo, con varios pilotos, y así abarcar mayor área de vuelo, que puede ser de utilidad en operaciones de rescate.

3 Objetivos y alcance

Objetivo: mejorar el interfaz hombre-drone. Se pretende conseguir un control intuitivo, y complementario al manejo por radiocontrol. La alternativa que se propone, además, incluye un manejo de drones cooperativo. De esta forma, se ampliarían las opciones de implantación de los mismos en algunos ámbitos de la sociedad y se podrían aumentar los usos de los mismos.

Para la consecución del objetivo principal, se tiene en cuenta otro secundario; la implementación de técnicas de detección de objetos y procesamiento de imágenes en drones. Incorporando estas técnicas a los equipos se podrían desarrollar nuevas funcionalidades y mayor autonomía.

El proyecto abarca desde la definición del programa de control hasta la implementación en el drone, pasando por el desarrollo y pruebas del mismo.

4 Beneficios

4.1 Beneficios sociales

Una de las principales ventajas de utilizar esta solución es que la interacción hombre-máquina es interactiva e intuitiva. Hasta ahora, el pilotaje se ha llevado a cabo mediante un mando radiocontrol, por lo que el manejo puede resultar difícil. Esta solución proporciona un control semejante a los movimientos humanos habituales para indicar direcciones, y puede ser cooperativo.

Una de las principales funciones que tiene el uso de drones en determinados ámbitos es el aumento de la seguridad de las personas. Se están utilizando estos equipos para la realización de tareas que suponían cierto riesgo para las personas que las llevasen a cabo. Las nuevas capacidades de las que se les dota a los VANTs en esta solución podrían aumentar el número de casos en los que esto es posible, y mejorar así las condiciones de los trabajadores.

4.2 Beneficios técnicos

Este método de control amplía el número de usuarios que pueden controlar un drone, lo que puede aumentar sus capacidades como el alcance máximo, incrementar las posibilidades de uso de estos equipos. También se agilizan determinadas tareas y se podría aumentar el tiempo de vuelo, pues en caso de que sea necesario cambiar la batería no es necesario que vuelva a la posición del piloto original, podría dirigirse a otro más cercano.

Por otra parte, el uso de técnicas de detección y reconocimiento de objetos en un caso como el que se está tratando podría suponer una gran mejora técnica. Es indudable que el tiempo de respuesta del programa debe aproximarse a un valor de respuesta en tiempo real. Por lo que el estudio de la adaptación y uso de estas técnicas para el proyecto podría resultar muy beneficioso para el avance en el desarrollo de las mismas.

4.3 Beneficios económicos

Es bien sabido que los errores, los problemas, los accidentes, y similares suponen ciertas pérdidas y mayores costes. Además, la velocidad con la que se solucionan también resulta esencial. Existen ciertos ámbitos en los que ya se están empleando drones, como la agricultura o la seguridad, en los que se ha conseguido disminuir pérdidas, problemas, o a detectarlos más rápido. Sin embargo, empleando esta nueva solución, se podría optimizar aún más el proceso en los ámbitos ya existentes e incluso en algunos nuevos.

5 Estado del arte

Actualmente se está normalizando el uso de drones en el día a día. Es posible encontrar algunos proyectos de drones basados en una placa Arduino Uno, como el proyecto YMFC-AL [2], que se encuentran en "brokking.net". En este se llevan a cabo ciertas modificaciones en las señales del drone para conseguir que se nivele automáticamente, lo que resulta en un vuelo y control más estables. También es posible encontrar proyectos de drones basados en una placa Raspberry Pi 3, como The drone Pi [3], con los que se puede llegar a una mayor comprensión del funcionamiento general de estos dispositivos. El uso de estas placas para la construcción de drones puede facilitar la implantación de características adicionales a los equipos, o incluso dotarles de mayor inteligencia.

Los VANTs cada vez poseen mayor capacidad para realizar funciones de forma autónoma. Algunas de estas funciones son el vuelo autónomo con una ruta prediseñada o recolección de datos para su uso en mediciones y estadísticas [4].

Respecto a los métodos de reconocimiento de objetos, cabe mencionar que existe una gran variedad de técnicas y algoritmos para llevar a cabo esta función, y algunas de ellas dependen del tipo de cámara empleada. Algunas incluso se han incorporado en librerías de programación, como OpenCV [5], para poder aplicar estos métodos de forma sencilla. Estas técnicas han ido mejorando a lo largo del tiempo, llegando incluso a ser viable su implantación en tiempo real [6].

Los métodos de detección de objetos con cámaras convencionales más habituales se basan en seguimiento de objetos, como la cámara Eye Toy de Play Station de 2003 [7]. Estos métodos requieren definir una región inicial que contiene el objeto a seguir. Si se detecta movimiento o algún cambio en la región, se registra y se actúa en consecuencia. También es posible utilizar marcadores de un determinado color, de tal forma que se filtran los colores de la imagen y/o se identifica el objeto característico según su forma. Otra técnica consta en la sustracción del fondo, de tal forma que si ocurre alguna anomalía en la escena, como algún movimiento o cambio, se detecta de forma sencilla.

Es necesario remarcar que en los casos mencionados se presupone que el objeto a seguir se sitúa en frente de la cámara, pero no siempre es posible identificar el objeto. Métodos como el uso de clasificadores o descriptores sí permiten la detección del objeto. Dos ejemplos son Histogram of Oriented Gradient (HOG) [8] y Haar Cascades [9].

El primer método se basa en el uso de un descriptor que representa al objeto. Este descriptor se genera en función de elementos característicos del objeto a detectar. El proceso para generarlo se basa en el cálculo de los gradientes en una región de interés. Primero se calculan los gradientes vertical y horizontal en cada píxel de la imagen. A partir de estos se calculan la magnitud y el ángulo (entre 0 y 180) del gradiente total. En el caso de imágenes a color, se consideran tres canales de color y se toman como magnitud y ángulo los valores máximos de cada uno.

Una vez calculados los gradientes, se calcula el histograma de los mismos. Para ello, se divide la región de interés en bloques de 8x8. En el histograma se consideran 9 bins, que corresponden a los ángulos: 0, 20, 40, 60, 80, 100, 120, 140 y 160. Por último, cada píxel del bloque contribuye de forma proporcional a cada valor del histograma. Esto es, si el valor del ángulo coincide con el del bin, la magnitud de este se añade a ese valor del histograma, y en el caso de

que se encuentre entre dos valores de bins, se reparte de forma proporcional según la distancia a cada valor de bin, a menor distancia mayor magnitud le corresponde. Una vez creado el histograma, si se normaliza y se visualiza en los bloques de 8x8, se puede ver que la dirección dominante coincide con la forma del objeto.

Los histogramas se normalizan en bloques de 16x16 empleando una ventana que se desplaza por los bloques de 8x8. En cada uno se concatenan los histogramas y se normaliza. Por último, cada bloque de 16x16 se concatena con el resto obteniendo así un vector de características que es el descriptor.

Este método ha inspirado el trabajo de Navneet Dalal y Bill Triggs, que han llevado a cabo una variante del método para detección de personas en imágenes [10], que, según se indica en el artículo citado, su implementación del método mejora la detección y evita falsos positivos. Peyman Hosseinzadeh Kassani, Junhyuk Hyun y Euntai Kim, que también utilizan una variación del método, Soft Histogram of Oriented Gradient (SHOG), han comprobado la aplicación de este para la detección de señales de tráfico [11]. Este método varía respecto a HOG en que hace uso de características de simetría, lo cual, según indican en el artículo, mejora la precisión y el rendimiento para la detección de señales de tráfico

El segundo se basa en el uso de clasificadores, creados a partir de una colección de imágenes positivas y negativas. Se consideran imágenes positivas aquellas que contienen el objeto a detectar y se consideran imágenes negativas aquellas que no lo contienen. A partir de las imágenes negativas se extrae la información que se considera no relevante, como características del fondo, y de las positivas se extraen las características de la imagen relevantes. Para la extracción de características, se utiliza un algoritmo basado en AdaBoost para Machine Learning, en el que se genera una representación de la imagen, a la que denominan imagen integral. A partir de estas se genera el clasificador. Y se emplea el mismo algoritmo para comparar los elementos.

Los primeros resultados no resultaron viables para su uso en tiempo real, pero con el clasificador adecuado, se conseguía una precisión muy buena. Este método también ha sido utilizado por Staffan Reinius para el reconocimiento de objetos en la plataforma iOS [12], sin embargo sus resultado no han resultado favorables debido a falsos positivos o el uso de objetos para los que resulta completo el entrenamiento del clasificador.

Por otra parte, empleando cámaras más complejas, como cámaras de visión en estéreo, cámaras infrarrojas o incluso térmicas, se puede atender a otras características y los elementos de la escena y así agilizar la detección de objetos. Un ejemplo, utilizado también en ocio, es el de la cámara Kinect de Xbox [13]. Esta cámara, además de una cámara convencional tiene otra cámara de infrarrojos, a partir de la cual obtiene características de profundidad. Este método se basa en el cálculo de la distancia, respecto a la cámara, a partir de las longitudes de onda de las señales generadas por la cámara infrarroja. Conocidas las distancias, y a partir de la imagen,es posible reconstruir la escena en tres dimensiones. Por otra parte, las cámaras térmicas ya se han llegado a utilizar en drones [14] para recolección de datos para estudios o en operaciones de rescate.

6 Definición de requisitos

6.1 Esquema general del sistema

La solución propuesta se compone de diferentes elementos hardware y software, tal y como se puede ver en la Figura 1 y en la Figura 2 respectivamente.

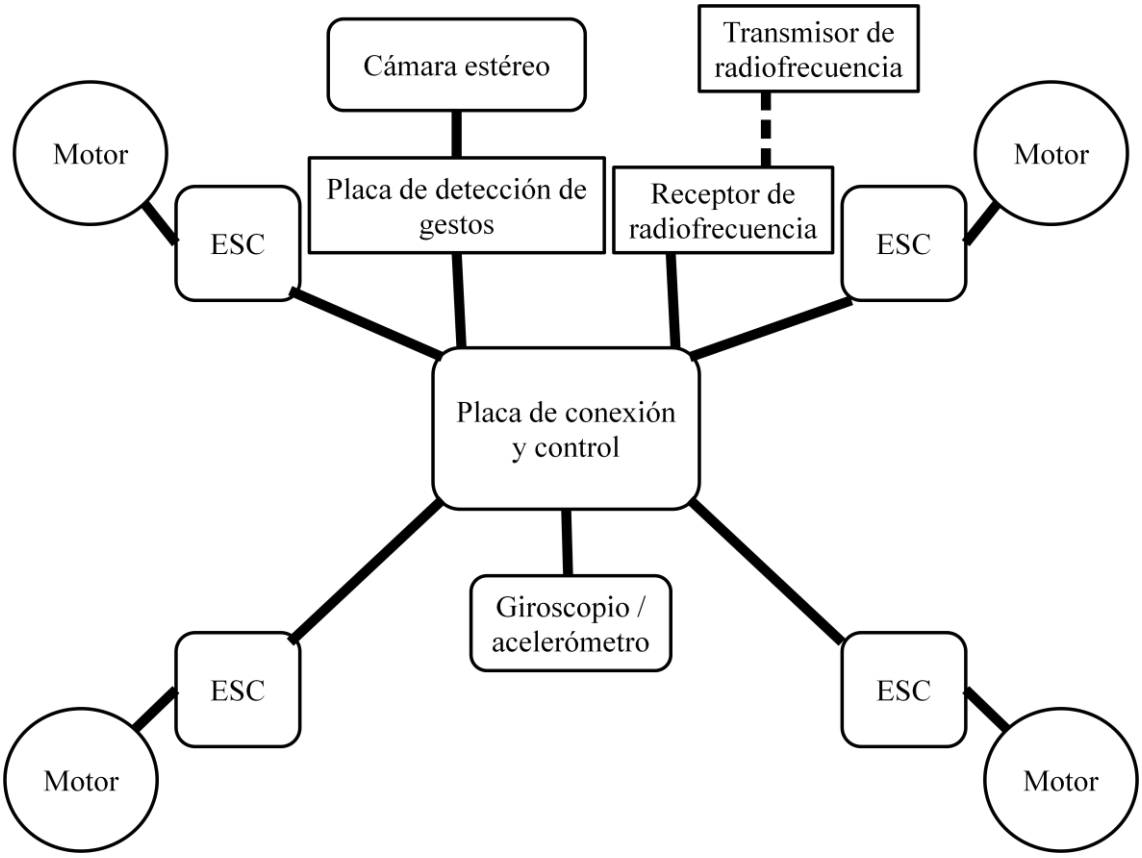


Figura 1: Esquema general de conexión de módulos hardware

Por una parte, el funcionamiento del drone requiere los siguientes módulos hardware:

- **Chasis**: en él se van a instalar los componentes del drone.
- **Batería**: para alimentar los elementos.
- **Motores**: estos pueden ser con o sin escobillas. Dado que se va a tratar de un quadricóptero, se van a necesitar cuatro.
- **Electronic Speed Controllers (ESCs)**: se necesitan tantos como motores haya y deben ser adecuados para el tipo de motor. Estos dispositivos controlan la velocidad y el sentido de rotación de los motores según la longitud de los pulsos PWM que reciban.
- **Hélices**: se necesitan tantas como motores haya, cuatro en este caso; dos para giro en sentido horario y dos para giro en sentido antihorario.
- **Giroscopio y acelerómetro**: estos dispositivos se utilizan para obtener valores sobre la posición del drone y para concocer la rotación de este en el aire respecto a los ejes X, Y y Z. Para mayor precisión debe colocarse en el centro del equipo
- **Transmisor (mando de radiocontrol) y receptor de radiofrecuencia**: estos equipos son necesarios para el manejo del drone. El receptor se conecta al drone, preparado para recibir las señales del transmisor al que esté emparejado. El transmisor envía señales con unos parámetros u otros, en función de lo que indique el piloto.

- **Placa de conexión y control de los elementos del drone**: en esta placa se concentra el control de los distintos módulos. Esta se encarga del envío y recepción de la información y del procesamiento de la misma.

Para la implementación del control del drone mediante gestos, también se necesita lo siguiente:

- **Cámara estéreo**: permite captar imágenes de la escena para su procesamiento.
- **Placa de ejecución del programa de reconocimiento de gestos**: procesa las imágenes, reconoce los gestos de las personas que se encuentren en esta, los interpreta y envía las señales a la placa del drone.

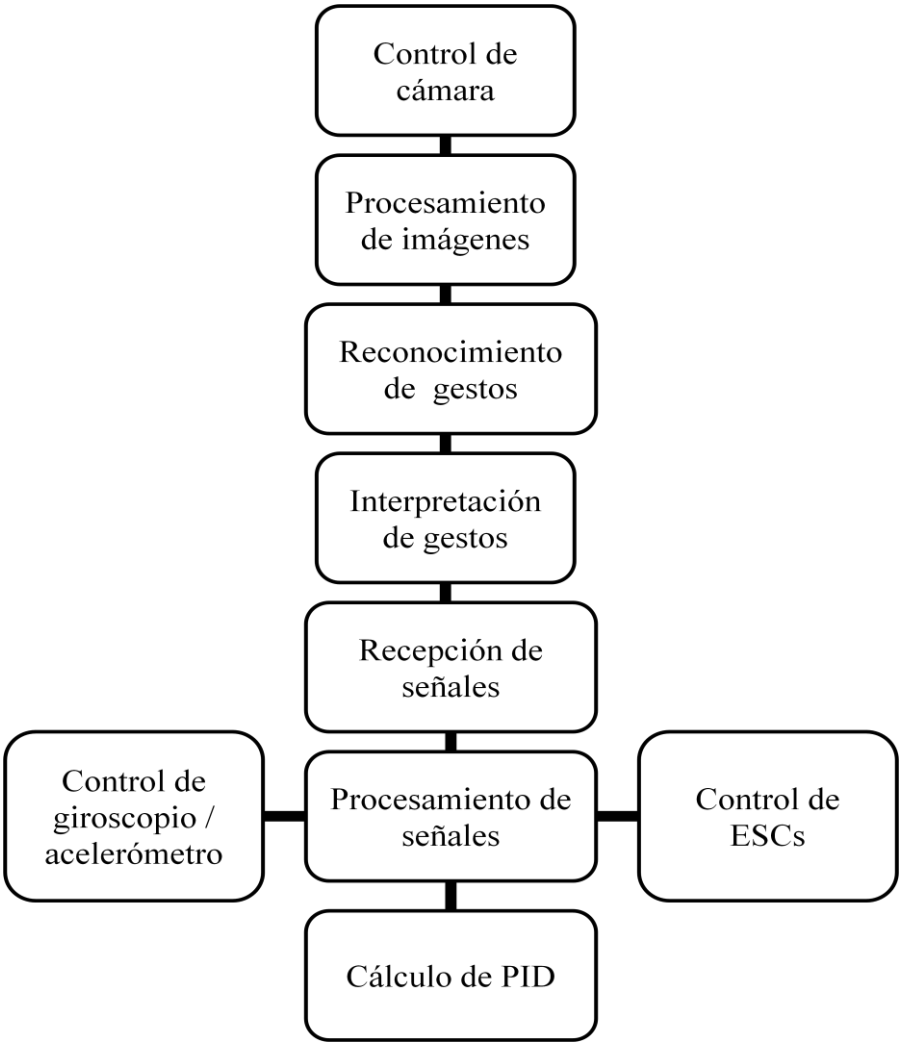


Figura 2: Esquema general de conexión de módulos software

Por otra parte, los mencionados elementos hardware, a su vez, requieren determinados módulos software para controlar algunos de ellos, y realizar operaciones que aportan cierta inteligencia al equipo. Estos módulos se implementan en las placas mencionadas previamente y llevan a cabo las siguientes funciones:

- **Control de cámara**: permite el acceso a la cámara y la captura de imágenes a partir de esta.
- **Procesamiento de imágenes**: se lleva a cabo un tratamiento de imágenes capturadas por la cámara para el proceso de reconocimiento.

- **Reconocimiento de gestos**: A partir del tratamiento de imágenes llevado a cabo, se aplican ciertas operaciones para el reconocimiento de gestos.
- **Interpretación de gestos**: realiza el proceso de traducción de los gestos detectados a las señales de manejo del drone.
- **Recepción de señales**: controla las señales que se reciben para el control del drone.
- **Procesamiento de señales**: gestiona la información que es necesario tener en cuenta para un vuelo controlado.
- **Control de giroscopio / acelerómetro**: calibra el dispositivo y gestiona la información obtenida por el mismo.
- **Cálculo de Proportional Integral Derivative (PID)**: el cálculo de PID resulta esencial para la estabilidad de vuelo de un drone. Todo objeto que se mueve tiene a mantener su movimiento. El movimiento puede ser lineal u oscilatorio, y para poder controlarlos es necesario generar una fuerza contraria a la que genera el movimiento para compensarlo. Las señales PID llevan a cabo esa función de corrección. Se tienen en cuenta:
    - Una parte Proporcional, es decir, que compensa el movimiento generando una fuerza igual en sentido contrario.
    - Una parte Integrativa, que también compensa el movimiento, pero considera una realimentación, es decir, tiene en cuenta valores anteriores. Por ello, resulta en una corrección más lenta.
    - Una parte Derivativa, con la que se tiende a volver a un punto origen o una situación de reposo.
- **Control de ESCs**: se gestionan las señales que se envían a los ESCs para el control de velocidades de los motores.

6.2 Requisitos

Teniendo en cuenta la implementación de la solución, es necesario destacar los siguientes aspectos:

- **Rendimiento del software**: el rendimiento de la solución debe ser adaptable a las características del drone. Esto implica que tenga una velocidad de procesamiento y tiempo de respuesta adecuado para su uso en tiempo real.
- **Solución integrable en el drone**: Es bien sabido que el peso afecta al tiempo de funcionamiento del drone debido al consumo de energía de la batería. Además, el peso debe estar compensado y distribuido por toda la estructura para mayor equilibrio. Por ello, la solución a implementar debe atender a estas características y evitar tener un gran impacto en estos aspectos.

7 Análisis de alternativas

7.1 Aspectos y alternativas

7.1.1 Placa de ejecución del programa de reconocimiento de gestos

Es importante que el dispositivo en el que se ejecute el programa de control cumpla con los aspectos descritos en el apartado "Descripción de requerimientos". Teniendo eso en cuenta se plantean dos opciones:

1. **Placa Rock64**

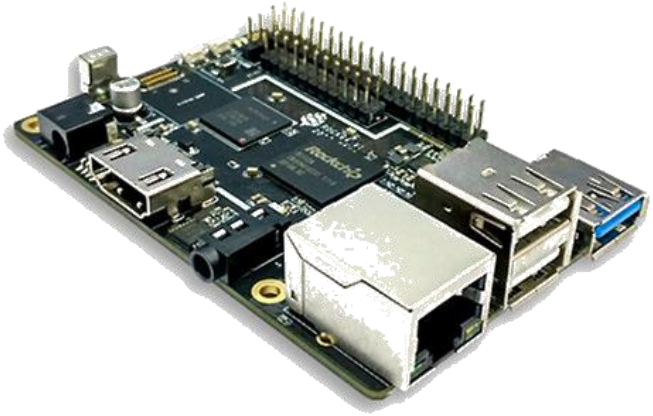


Figura 3: Microordenador Rock64

Observando las características de esta placa (hasta 4 GB de memoria RAM y procesador de 4 núcleos de 1,5 GHz) [15] se puede ver que, en comparación con otras que hay en el mercado, sus características deberían ofrecer mejor rendimiento que otras soluciones. Además, su pequeño tamaño tendría un bajo impacto en el peso del drone. Sin embargo, cabe mencionar que algunos de los usuarios de este equipo han experimentado ciertos fallos que no permiten hacer uso de todas las características de la placa y que aún no se han solucionado.

2. **Placa Raspberry Pi 3 +**

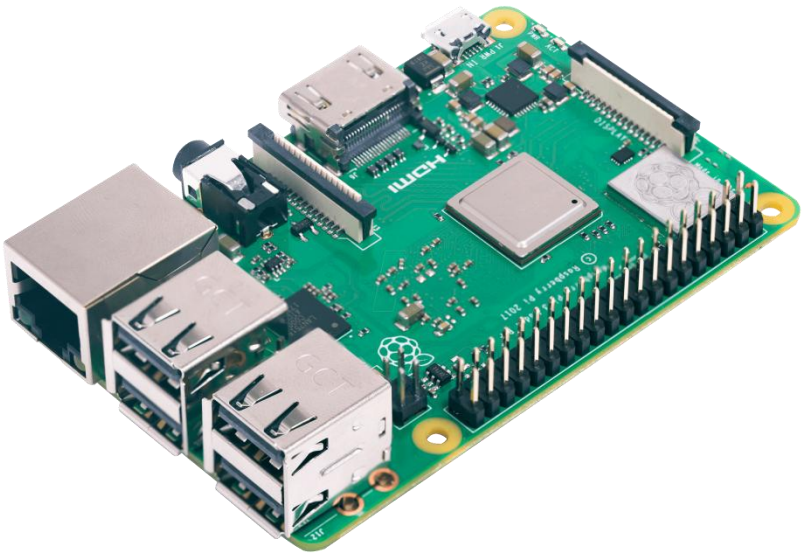


Figura 4: Microordenador Raspberry Pi 3 b+

Esta placa tiene las mismas dimensiones y posee características similares, algo inferiores (1 GB de memoria RAM, procesador de 4 núcleos de 1,4 GHz), a uno de los modelos de la anterior opción [16], pero su precio es algo mayor. Por otra parte, este equipo lleva en el mercado cierto tiempo, su comunidad de usuarios es muy extensa y es posible encontrar opiniones muy positivas sobre el mismo. Es por ello que, en caso de que hubiera algún error, hay mayor probabilidad de que se solucione de forma rápida.

7.1.2 Placa de conexión y control de los elementos del drone

En este proyecto resulta de vital importancia la integración de la solución en el equipo final, es decir, el VANT. Por ello, a la hora de llevar a cabo las pruebas es importante contar con un equipo que pueda soportar dicha solución y que ayude a la comprensión del funcionamiento y a la medición de las señales. Por ello, se están planteando las siguientes alternativas:

1. **Drone basado en una placa Arduino Uno**

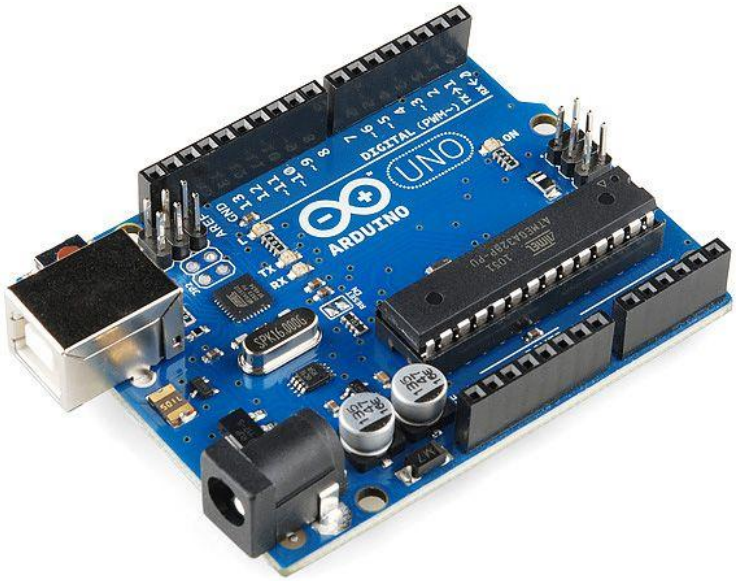


Figura 5: Placa programable Arduino UNO

En el apartado "Análisis del estado del arte" se ha comentado que actualmente existen proyectos de drones basados en placas Arduino Uno. Las placas constan de numerosas entradas y salidas para la conexión de los elementos, y se programan para el control de los mismos. Con esta alternativa es posible comprender en profundidad el funcionamiento de los equipos, así como controlar y personalizar las comunicaciones que se lleven a cabo.

2. **Drone basado en una placa Raspberry Pi 3**

Anteriormente también se ha mencionado la existencia de proyectos basados en placas Raspberry Pi 3. Estos son trabajos muy completos con diversos módulos y componentes. Es necesario remarcar que estos no cuentan con mando radiocontrol para el manejo del drone, sino que el envío de señales se hace mediante otras tecnologías, como WiFi.

7.1.3 Identificación de objetos

Como se ha mencionado en el apartado "Análisis del estado del arte", existe una gran variedad de técnicas de detección de objetos. Es por ello que es necesario hacer una selección del método principal a utilizar que cumpla con los requerimientos del proyecto. La librería OpenCV permite hacer uso de diferentes técnicas de detección de objetos y la programación en diferentes lenguajes.

Las alternativas que se han planteado, que se incluyen en la mencionada librería, son las siguientes:

1. **Identificación de elementos de un color determinado**

Este método se basa en filtrar los elementos de un determinado color en una imagen. Aprovechando las características de color de una determinada imagen, se filtraría un rango limitado de color, preferiblemente de un color característico difícil de hallar en exteriores. Una vez se ha limitado, para la detección del objeto buscado habría que atender a sus características de forma, empleando algún otro método.

2. **Uso de clasificadores**

Los clasificadores atienden a una lista de características que posee el objeto para identificarlo. Con un número mayor de características el método resulta más preciso pero requiere mayor tiempo y capacidad de procesamiento. Los métodos que se están considerando son:

   a. Haar cascades
   b. Histograma de Gradiente Orientado (HOG)

Se compararon ambos métodos, realizando una prueba en la que éstos se utilizaban para la detección de personas en una escena en tiempo real.

Se preparó un programa, utilizando la librería OpenCV, en el que se capturaban las imágenes de una cámara, y se aplicaba uno u otro método para llevar a cabo la detección. Tras la prueba se concluyó que el rendimiento del segundo es mayor que el primero. Éste identifica los objetos de forma más rápida, aunque es posible que sea necesario realizar un filtrado debido a la presencia de falsos positivos. El primero, por otra parte, tiene mayor probabilidad de fallar en la identificación del objeto y el método de detección requiere mayor procesamiento.

3. **Detección del elementos a partir de parámetros adicionales**

En el caso de emplear alguno de estos métodos haría falta alguna cámara especial.

   a. Detección de objeto más cercano

Para detectar el objeto más próximo sería necesaria una cámara capaz de obtener valores de profundidad en una escena, como una cámara estéreo.

   b. Detección de calor

En este otro caso, detectar un objeto que produzca calor, habría que incluir una cámara térmica.

Estos métodos atienden a características adicionales a los de las imágenes convencionales agilizando el proceso de detección de objetos. Sin embargo, la necesidad de uso de cámaras especiales puede encarecer el proyecto.

7.1.4 Lenguajes de programación

Como ya se ha mencionado, la librería OpenCV permite la programación en diferentes lenguajes. Para este proyecto es necesario atender a las características propias de estos para escoger aquel que permita la implementación en el hardware con un rendimiento adecuado.

Los lenguajes que se contemplan en la librería son los siguientes:

1. **C++**

Este lenguaje es bastante versátil, ya que permite el uso de clases y resulta eficiente en la ejecución. La librería contiene una gran cantidad de módulos para su uso en este lenguaje, lo que da un amplio abanico de posibilidades.

2. **Python**

Es un lenguaje que también permite el uso de clases y es un lenguaje bastante utilizado y extendido. La programación con este lenguaje resulta eficiente en términos de tiempo de programación, pero se sacrifica cierta eficiencia en el tiempo de ejecución.

La librería también contiene una gran cantidad de módulos preparados para su uso en este leguaje, derivados de los existentes para C++. Sin embargo cabe la posibilidad de que algunos de ellos no se encuentren en este lenguaje.

3. **JavaScript**

Es un lenguaje muy extendido, sobre todo para otorgar cierto dinamismo a las páginas web. Se trata de un lenguaje bastante ligero, pero la cantidad de módulos de OpenCV desarrollados para este lenguaje resultan limitados y la documentación no es abundante.

7.2 Criterios de selección

Rendimiento: El método a utilizar debe asegurar la viabilidad, pues una gran latencia en la respuesta del programa, o una latencia variable afectarían a la estabilidad en la implementación. Para la valoración del rendimiento de cada alternativa, se va a considerar la comparativa de métodos. Esto implica que aquel método que ofrezca mayor rendimiento tendrá mayor puntuación. Cabe destacar que para obtener un valor de puntuación superior a la mitad, debe tener unos tiempos de procesamiento considerados adecuados para el tiempo real. Este criterio resulta esencial para la viabilidad del proyecto, por ello la ponderación del criterio va a tener mayor peso.
Ponderación del criterio: 2

Rango de valoración: 0 – 3

Valor económico: Algunos de los mencionados métodos podrían encarecer el proyecto por la necesidad de equipo específico adicional. Aquellas alternativas que no supongan un incremento económico tendrán la máxima puntuación. El resto de las alternativas se puntúan por comparación, es decir, aquellas que supongan mayor coste obtendrán menor puntuación, extrapolado al rango de puntuación. Este criterio, aunque relevante, se espera que no tenga un gran impacto en el proyecto, por ello, la ponderación es la que sigue.
Ponderación del criterio: 1

Rango de valoración: 0 – 3

7.3 Selección de alternativas

7.3.1 Selección de la placa de ejecución del programa de reconocimiento de gestos

Tabla 1: Selección de alternativas de equipo de implementación del programa

Placa para reconocimiento de gestos	Rendimiento	Coste económico	Resultado
Rock64	1	3	5
Raspberry Pi 3	2,5	2,5	7,5

La baja puntuación en rendimiento de la placa Rock64 se debe a la baja fiabilidad de ésta. El funcionamiento erróneo afecta en gran medida al rendimiento del equipo, incluso no permitiendo llevar a cabo las funciones del mismo correctamente. Es por ello que aunque su precio sea algo mayor finalmente se ha optado por emplear una placa Raspberry Pi 3.

7.3.2 Selección de la placa de conexión y control de los elementos del drone

Tabla 2: Selección de alternativas de equipo de pruebas

Placa de conexión y control del drone	Rendimiento	Coste económico	Resultado
Drone basado en Arduino	3	3	7
Drone basasdo en Raspberry Pi 3	2,5	3	5

El coste económico de ambas alternativas resulta parecido, debido a que los elementos comunes de ambos equipos son los que encarecen el mismo.

En caso de utilizar un drone basado en una placa Raspberry Pi 3, debido a que la implementación se va a llevar a cabo en un equipo similar, físicamente podría tratarse del mismo equipo. Sin embargo, esto podría tener cierto efecto sobre el rendimiento, mientras que en el caso de emplear un drone basado en Arduino el módulo de control sería independiente. Por ello, a este segundo se le ha otorgado una puntuación mayor.

Teniendo lo anterior en cuenta, y observando opciones de escalabilidad, portabilidad de la solución, posibles mejoras a futuro y las opciones de aprendizaje, se opta por emplear un drone basado en Arduino.

7.3.3 Selección del método de identificación de objetos

Tabla 3: Selección de alternativas de detección de objetos

Identificación de objetos	Rendimiento	Coste económico	Resultado
Identificación por color	2	3	7
Clasificadores – Haar cascades	1	3	5
Clasificadores - HOG	1,5	3	6
Objeto más cercano	3	2	8
Detección de calor	3	1	7

En cuanto al rendimiento, hay alternativas que obtienen una puntuación de 3, la máxima puntuación. Esto se debe a que el efecto que estos tienen en el tiempo de procesamiento es mínimo y es adecuado para el tiempo real. El resto de métodos afecta en mayor o menor medida, por ello se les asigna una puntuación menor. Algunas incluso menor de la mitad, por lo que el impacto de estas en el tiempo de procesamiento puede ser bastante alto.

Atendiendo a los criterios y las ponderaciones otorgadas a cada uno, se opta por una alternativa que, aunque aumente ligeramente el coste económico del proyecto, no tiene un gran impacto en el mismo. Además, cumple con las características de rendimiento, que es esencial para el desarrollo y la viabilidad de la solución.

7.3.4 Selección del lenguaje de programación

Tabla 4: Selección de alternativas de lenguajes de programación

Lenguaje de programación	Rendimiento	Coste económico	Resultado
C++	3	2,5	8,5
Python	2,5	3	8
JavaScript	2	2	6

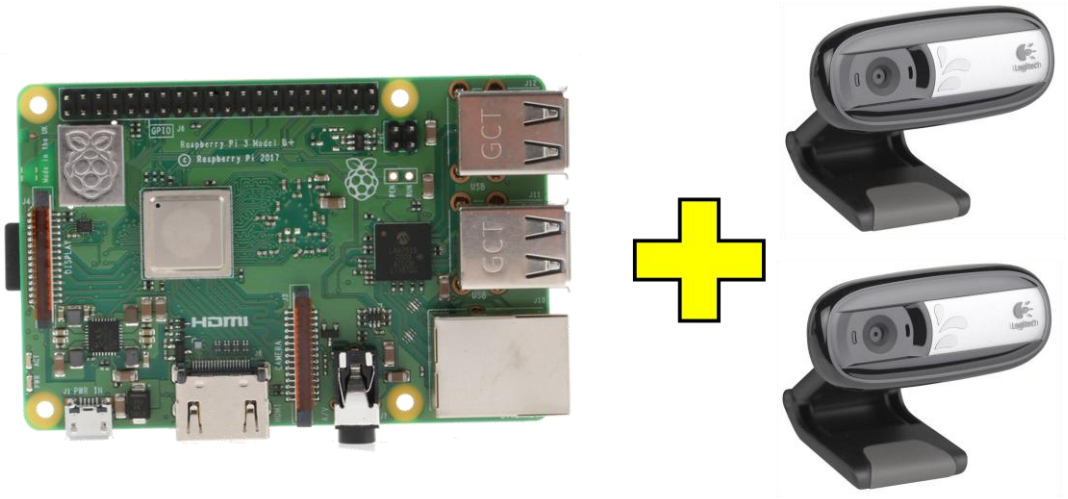
Respecto al rendimiento, comparando las características de los tres lenguajes, se llega a la conclusión de que la ejecución del programa en C++ sería más rápido que en los otros lenguajes, por ello se le da una puntuación de 3. Aunque se espera obtener un rendimiento aceptable si se emplea cualquiera de ellos, es por ello que todas las puntuaciones en este aspecto superan el 1,5. Por otro lado, aquel lenguaje que, en principio, supondría menor coste económico sería Python. Esto se debe a que es un lenguaje que optimiza el tiempo de programación, por lo que en principio, el presupuesto destinado a este aspecto sería menor que en los otros casos. En cuanto a C++, el coste económico aumenta ligeramente debido al tiempo necesario para la programación de éste, pero empleando módulos existentes, el aumento no resulta muy significativo. En el caso de JavaScript, el impacto económico es mayor, debido a que sería necesario destinar mayor presupuesto para la fase de codificación.

Teniendo en cuenta lo anterior, se ha decidido llevar a cabo la codificación en C++.

8 Diseño y descripción de la solución

En base a la alternativa escogida y los requerimientos necesarios, se propone la solución descrita en los siguientes subapartados.

8.1 Módulos Hardware



Raspberry Pi 3 + 2 cámaras USB convencionales

Figura 6: Elementos hardware para detección de gestos

Como se ha mencionado en el apartado Descripción de requerimientos, el peso es importante para la duración de la batería y un vuelo estable. Además, el sistema sobre el que se implante debe poder ofrecer un rendimiento adecuado, con tiempos de respuesta casi inmediatos.

En la Figura 6 se pueden ver los elementos software que se emplean en la solución. Por una parte, se utiliza una placa Raspberry Pi 3+, tal y como se ha seleccionado en el apartado de alternativas, por su pequeño tamaño y características [16].

Por otra parte, en base a la alternativa escogida para la identificación de objetos, será necesario utilizar una cámara en estéreo, o en este caso se plantea el uso de dos cámaras convencionales. De esta forma se tiene mayor control de la información que se utiliza. Se utilizan dos cámaras Logitech iguales, ya que para poder hacer uso de las características de profundidad de la escena, es necesario que las imágenes que se obtienen tengan las mismas características de nitidez, dimensiones, etc.

En base a la elección del equipo sobre el que se va a implementar el control mediante gestos, un drone basado en Arduino, en la Figura 7 se pueden ver los elementos de los que se compone.

Figura 7: Elementos hardware del drone

El chasis escogido es de 450 mm. Las dimensiones de la base la hacen idónea para la solución que se plantea, ya que cuenta con espacio suficiente para el montaje de todos los elementos. La base principal es de fibra de vidrio. Esta distribuye la alimentación a lo largo de la placa, de tal forma que elementos que sea necesario alimentar directamente con la batería, como los ESCs, se pueden conectar de forma sencilla. Por otra parte, las patas sobre las que se colocan los motores son de plástico y cuentan con los espacios necesarios para el montaje de los mismos.

Los motores escogidos son unos motores sin escobillas de 1000 KV. Este tipo de motores resultan más duraderos que los motores con escobillas, y otorgan la fuerza necesaria para levantar el vuelo del equipo. En base a los motores escogidos, se ha optado por unos ESCs compatibles con motores sin escobillas, que requieren una alimentación entre 5,5 y 12,6 V. Y las hélices escogidas son de plástico y de tipo 1045.

El giroscopio/acelerómetro que se utiliza es el MPU6050, que incorpora las funcionalidades de ambos equipos, lo cual es beneficioso para el peso y resulta muy versátil. Además, existen bastantes recursos que facilitan su uso en programación.

Otro de los elementos que se utilizan son el transmisor y receptor de radiofrecuencia. El principal requisito que debe cumplir el receptor, es que debe tener, como mínimo, cuatro canales para el envío de las señales, aunque el que se utiliza en este caso es de seis.

La batería que alimenta a los elementos es de 11,1 V, por lo que entra dentro de los límites para la alimentación de los ESCs. Por otra parte, con esa capacidad podría proporcionar un tiempo de vuelo aceptable.

Por último, la placa de conexión y control de los elementos, tal y como se ha seleccionado en el apartado de Análisis de alternativas, se trata de una placa Arduino UNO, que consta de los pines necesarios para la conexión de los dispositivos y capacidad suficiente para el funcionamiento del mismo.

8.2 Módulos Software

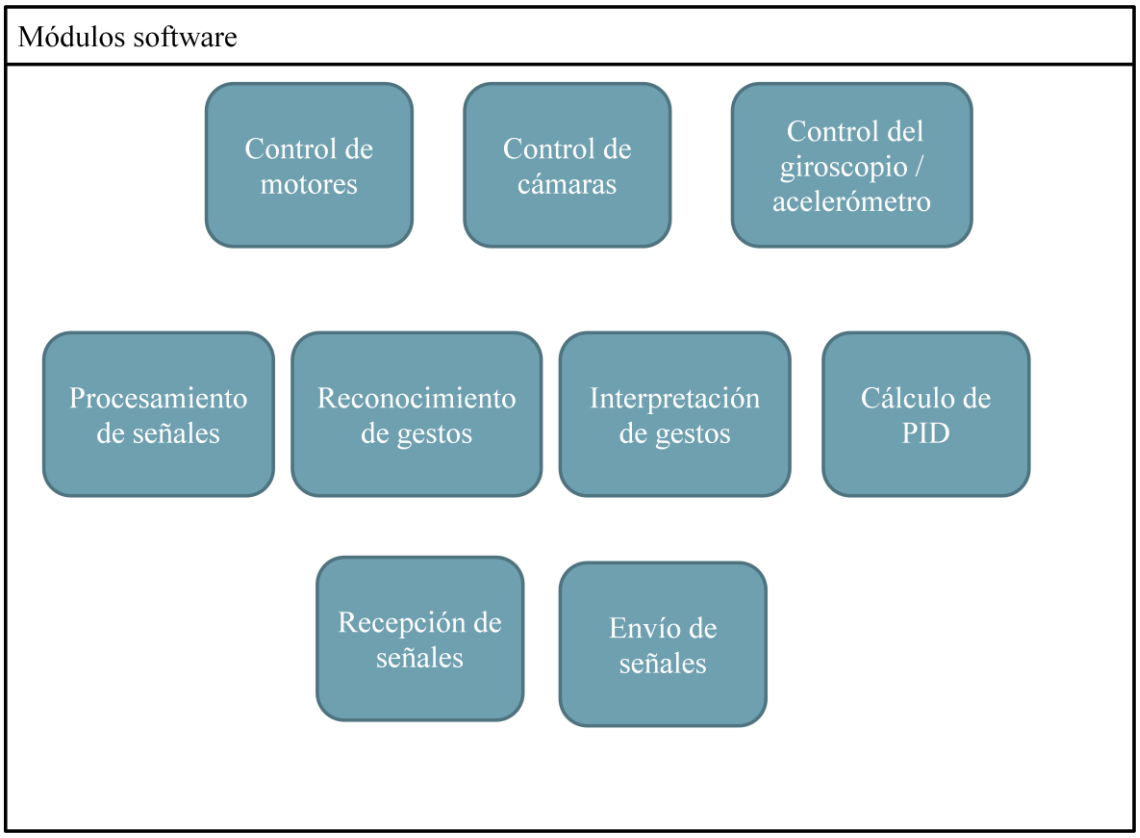


Figura 8: Módulos software de la solución

En base a los elementos software que resultan necesarios para el control de un drone, en la Figura 8 se pueden ver los módulos de la solución.

Es posible identificar que algunos de los módulos tienen como función el control de periféricos o de elementos hardware.

Estos son:

Control de cámaras

Este módulo, entre sus funciones, es esencial que lleve a cabo las siguientes:

- Permite el control de varias cámaras simultáneamente
- Encendido y apagado de las cámaras
- Configuración de la capacidad del bus
- Configuración de dimensiones de las imágenes
- Lectura de imágenes

Control de ESCs

Este módulo controla las señales que se envían a los ESCs para el control de la velocidad de giro de cada motor en cada instante de tiempo. Debe tener en cuenta los valores de PID y como afectan a cada uno de los motores de forma individual. Las funciones que debe llevan a cabo son:

- Calibración de los ESCs
- Generación de señales PWM
- Reconfiguración de los parámetros de las señales

Control del giroscopio / acelerómetro:

Gobierna el manejo del giroscopio/acelerómetro. Debe llevar a cabo las siguientes funciones:

- Calibración del módulo
- Actualización de la información del módulo
- Cálculo de parámetros a partir de las medidas obtenidas

En la Figura 9 [17] se aprecia un ejemplo de la relación entre las señales de control del drone y los ángulos respecto a los ejes X, Y y Z. Es por ello que la información obtenida del giroscopio resulta esencial para el control y estabilidad del vuelo y debe estar actualizada siempre que se vaya a llevar a cabo el cálculo del PID.

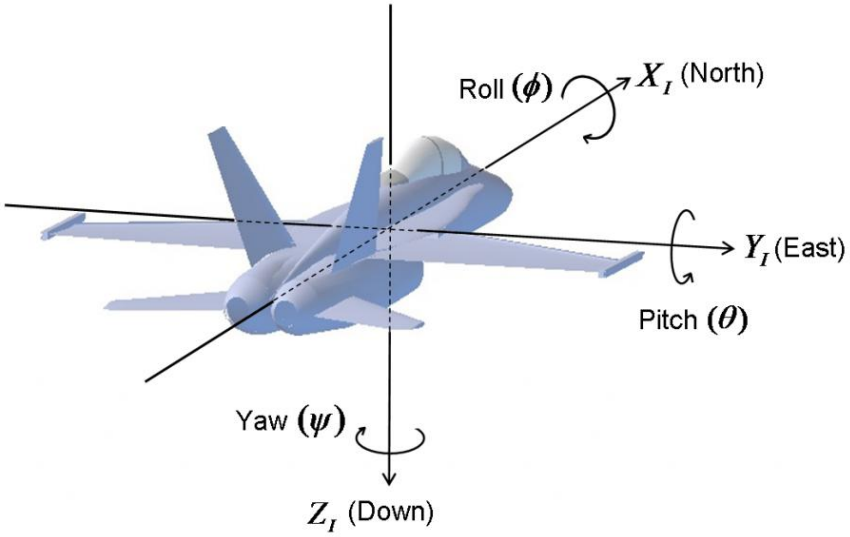


Figura 9: Señales y direcciones

Por otra parte, existen algunos módulos de control de señales de comunicación:

Recepción de señales

Lee las entradas provenientes o del mando o del módulo de reconocimiento de gestos. Estos envían señales PWM. Este módulo debe llevar a cabo la lectura de las señales de control tan pronto como las recibe. La función principal a realizar por este módulo es:

- Medición de longitud de pulsos PWM

Envío de señales

Este debe controlar la comunicación entre las placas de la solución y generar las señales pertinentes. Entre sus funciones, se encuentran:

- Control de apertura y cierre de comunicaciones
- Generación pulsos PWM

Por último, existen algunos módulos que controlan características software o añaden cierta inteligencia mediante el uso de algoritmos.

Reconocimiento de gestos

Procesa las imágenes para la detección de personas y de los gestos que éstas llevan a cabo. En esta etapa se emplea el método de detección seleccionado en el apartado de Análisis de alternativas. Para ello, es necesario que lleve a cabo lo siguiente:

- Tratamiento de imágenes
    - Cambio de espacio de color
    - Difuminar
    - Eliminación de ruido
    - Cambio de tamaño
    - Normalización
    - Filtrado de valores
- Obtención de contornos
- Cálculo del rectángulo mínimo que encierra a un objeto
- Generación de la escena en tres dimensiones
- Detección de objetos
- Reconocimiento de objetos
    - Gestos

Interpretación de gestos

Traduce la información de los gestos detectados según al movimiento que debe llevar a cabo el drone. Esto lo lleva a cabo a partir de:

- Asociación de gestos a movimientos del drone
- Configuración de parámetros de las señales según el movimiento a realizar

Cálculo del PID

Este es un módulo software con el que deben contar los drones para asegurar la estabilidad del vuelo. Para cada una de las señales elevador, timón y alerón, debe tener en cuenta los valores de los módulos receptor y del giroscopio/acelerómetro y debe llevar a cabo lo siguiente:

- Cálculo de la parte Proporcional
- Cálculo de la parte Integrativa
- Cálculo de la parte Derivativa
- Generación de la señal PID

Procesamiento de señales

Este es un módulo en el que se procesan las señales que hay que tener en cuenta para marcar la velocidad de los motores. Tiene en cuenta los valores de las señales PID, de las señales de control recibidas y de las señales del giroscopio/acelerómetro.

8.2.1 Diseño de la solución

Los mencionados módulos se implementan en los Software que se ejecutan en las placas Raspberry Pi 3 y en la placa Arduino UNO, e intervienen en diversos momentos. Los módulos que intervienen en cada uno de los Software se pueden ver en la Figura 10.

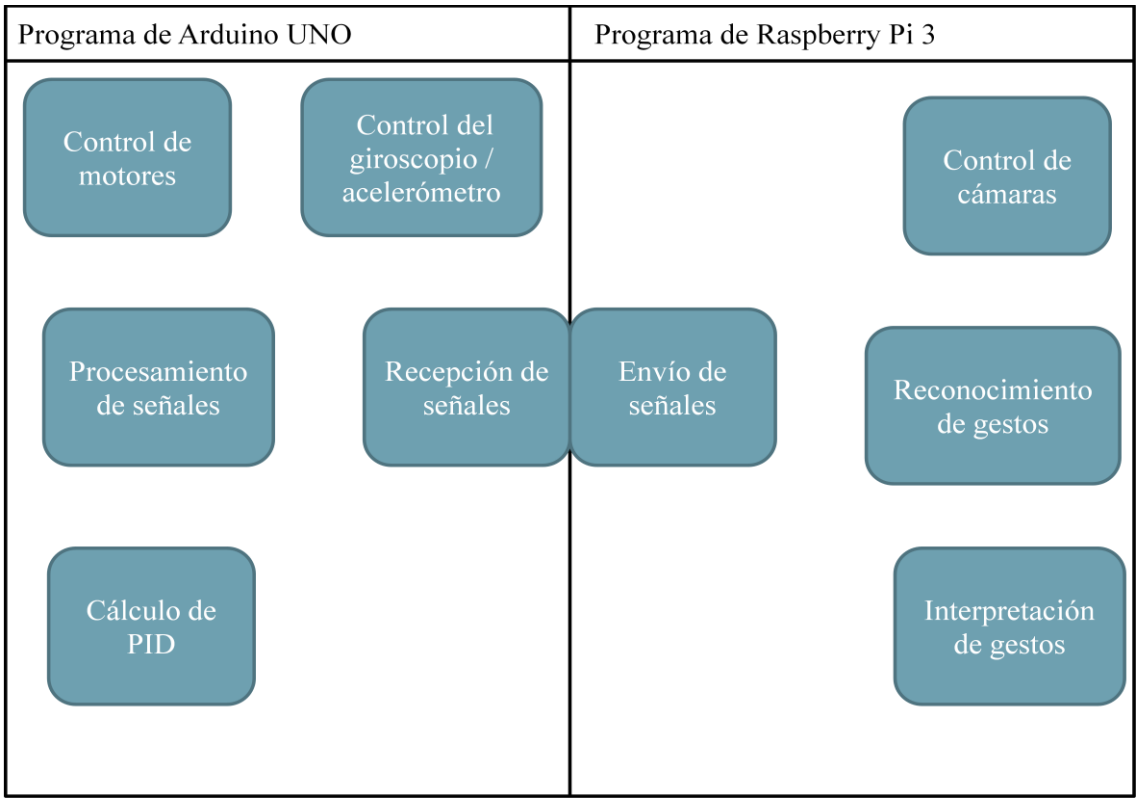


Figura 10: División de módulos por programas

8.2.1.1 Software de reconocimiento de gestos

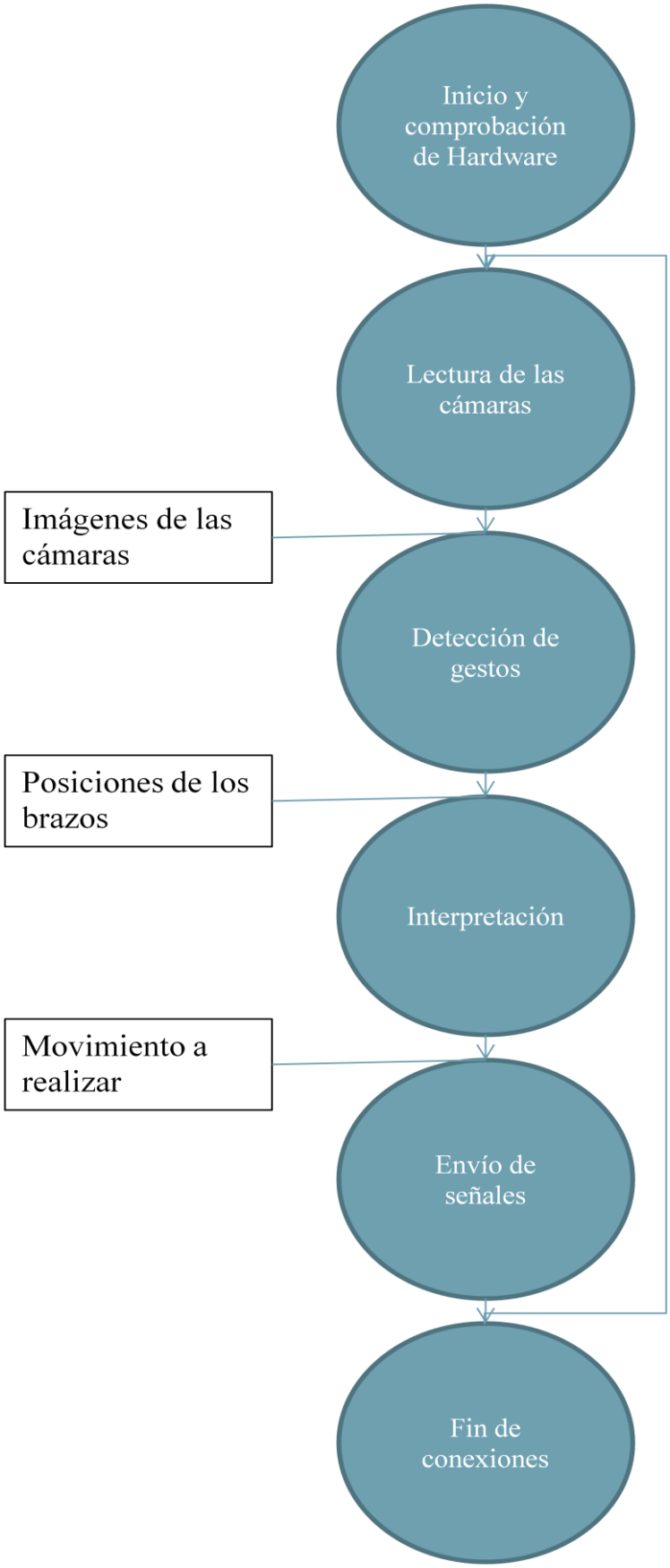


Figura 11: Diagrama de flujo del programa de reconocimiento de gestos

En la Figura 11 se puede ver el diagrama de flujo del funcionamiento del programa de control. Éste estará íntegramente desarrollado en C++. Tal y como se puede observar, hay una serie de etapas claramente diferenciadas. Algunas de ellas se van a repetir una y otra vez hasta que se finalice el programa. Estas son las que se consideran parte del bucle principal, y llevan a cabo la función principal del programa.

8.2.1.1.1 Inicio y comprobación de Hardware

En esta primera etapa, se inicia y se inicializan los módulos, por ello, intervienen la mayor parte de estos. Esta se ejecuta antes del inicio del bucle principal, ya que no es necesario configurar una y otra vez los parámetros de funcionamiento.

Por una parte, se inicializan y comprueban los elementos hardware, es decir, se inician las cámaras y se comprueba que es posible leer de ellas correctamente. También se inicia el módulo de envío de señales. Esto implica que se comienzan a generar pulsos PWM con los valores por defecto.

Por otra parte, se llevan a cabo las configuraciones iniciales para el tratamiento de imágenes. Es decir, se inicializa el módulo de reconocimiento de gestos.

En la Figura 12 se puede ver de forma gráfica diseño de la misma.

Primero se procede con la apertura de las cámaras. Para ello se siguen los siguientes pasos:

- Se configuran las dimensiones de las imágenes a capturar (640 x 480).
- Se configura la capacidad del bus que se va a emplear
- Se indica la cámara con la que se quiere establecer conexión mediante el identificador de la cámara.

En caso de que no se pueda acceder a alguna de ellas, bien porque no estén conectadas o porque se ha producido un error en el equipo, no se seguirá con el proceso, sino que se finalizará el mismo.

Si las cámaras son accesibles, se lleva a cabo una comprobación de lectura de las mismas y así asegurar que es posible capturar imágenes. De la misma forma, en caso de que no se pudiera realizar la lectura se finalizaría el proceso.

Una vez comprobado el correcto funcionamiento de las cámaras, se inicia la comunicación por los pines de la placa Raspberry Pi 3. Esto conlleva las siguientes funciones:

- **Configuración de los pines a utilizar como salidas**: es necesario definir los pines de la placa que se van a utilizar. En este caso habría que hacer uso de cuatro pines GPIO, uno por cada señal a generar (acelerador, timón, alerón y elevador).
- **Configuración de los pulsos PWM en los pines**: cada una de las señales previamente mencionadas se trata de un pulso PWM. La longitud de estos pulsos en estado alto debe estar comprendida entre los valores 1000 a 2000 microsegundos.
- **Inicio de generación de pulsos PWM**: se generan los pulses con los valores de longitud considerados por defecto. Los valores de longitud de los pulsos de las señales son los que se indican en la Tabla 5.

Tabla 5: Valores por defecto de las señales del drone

Señal	Valor por defecto (microsegundos)
Acelerador (Throttle)	1000
Elevador (Pitch)	1500
Alerón (Yaw)	1500
Timón (Roll)	1500

Por último, se inicializa la configuración del módulo de detección. En este es necesario configurar ciertos parámetros para la reconstrucción en tres dimensiones. La reconstrucción se realiza a partir de un mapa de disparidades. Los parámetros más destacables de esta configuración son el número de disparidades, que debe ser un múltiplo de 16, y el tamaño de los bloques sobre los que se aplica.

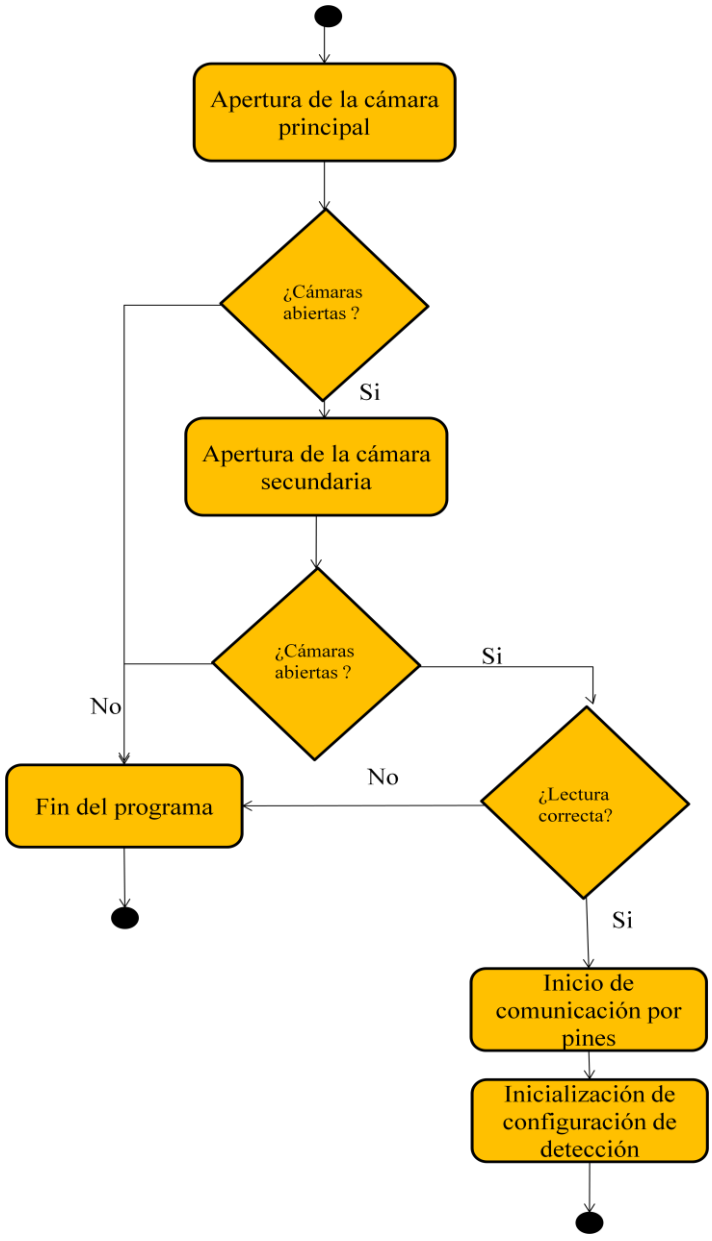


Figura 12: Diagrama de flujo de la etapa de inicio y comprobación del hardware

8.2.1.1.2 Lectura de las cámaras

Esta etapa es la primera que se ejecuta en el bucle principal. En esta se lleva a cabo la lectura de las cámaras, y en consecuencia, el único módulo software que interviene en esta etapa es el módulo de control de las cámaras.

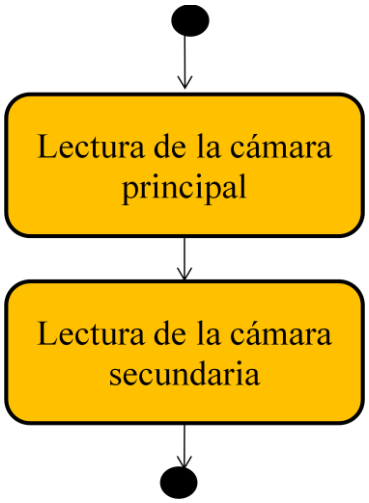


Figura 13: Diagrama de flujo de la etapa de lectura de las cámaras

En esta segunda etapa se lleva a cabo la lectura de las cámaras tal y como se aprecia en la Figura 13. Primero se procede con la lectura de la cámara principal, la cámara izquierda, y luego la de la cámara secundaria, la derecha. La lectura debe ser casi síncrona, es decir, la diferencia de tiempo entre ellas debe ser despreciable, ya que, de esta forma se consiguen mejores resultados en la obtención del mapa de disparidades y mayor estabilidad y fiabilidad. Para lograr que la diferencia de tiempo sea mínima, en lugar de capturar la imagen y acto seguido procesarla, primero se capturan ambas imágenes y después se procesan.

8.2.1.1.3 Detección de gestos

Una vez obtenidas las imágenes de las cámaras, se debe detectar la posición de los brazos. En esta etapa, se lleva a cabo el tratamiento de imágenes y el reconocimiento de gestos, por lo tanto, interviene el módulo software de reconocimiento de gestos. En ella radica la mayor parte del trabajo de ingeniería

Cabe destacar que para el correcto funcionamiento de este módulo se parte de la suposición de que la persona que lleva a cabo el control del VANT será el objeto más cercana al mismo.

Primero se pre-procesan las imágenes para disminuir la información no relevante de la misma que puede afectar a la detección. Para ello se realizan las siguientes funciones:

- **Cambio de mapa de color**: se cambia el mapa de color de las imágenes, de RGB a blanco y negro. El color afecta a la detección y en este caso no proporciona información relevante.
- **Difuminar**: se difuminan las imágenes. Las texturas muy acentuadas dificultan la identificación de objetos al crear del mapa de disparidades.

A continuación, se forma el mapa de disparidades a partir de las imágenes tratadas.

Una vez obtenido, es necesario tratar la imagen para filtrar el objeto más cercano:

- **Obtención del mapa de profundidad**: los valores de la imagen del mapa de disparidades son bajos, y no es posible filtrar los objetos con estos valores. Por ello, para poder distinguir profundidades, se normaliza el mapa a valores entre 0 y 255 para así obtener el mapa de profundidad.
- **Eliminación de ruido**: se reduce el ruido del mapa de profundidad para evitar posibles fallos en la detección.
- **Obtención de contornos y filtrado del más grande**: se obtienen los contornos de la imagen, de tal forma que aquel que sea el contorno más grande será el del objeto que se encuentre más cerca de la cámara, y será el que se tenga en cuenta.
- **Filtrado del objeto**: con el contorno obtenido, se crea una imagen de las mismas dimensiones que la original pero que únicamente contiene el objeto que se encuentra encerrado en el contorno de mayor área.

Una vez se tiene el objeto, se considera que los brazos se encuentran a ambos extremos del cuerpo y que, como máximo, se consideran un treinta por ciento del total que se muestra en la imagen. Por ello, se obtienen imágenes de los extremos del objeto y se obtiene el rectángulo mínimo que encierra al objeto de las imágenes.

A partir del ángulo de inclinación del rectángulo es posible hallar la posición de los brazos.

En la Figura 14 se puede ver el diagrama de flujo de funcionamiento de este módulo de forma gráfica.

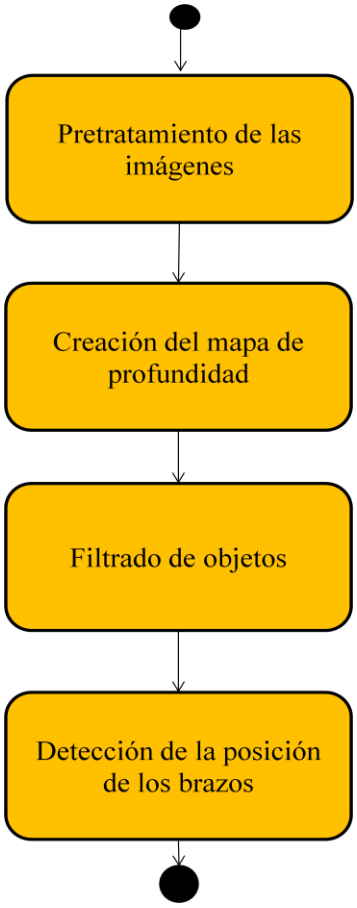


Figura 14: Diagrama de flujo del módulo de detección de gestos

8.2.1.1.4 Interpretación

Conocida la posición de los brazos, se asocia con el movimiento que debe llevar a cabo el drone, función que lleva a cabo el módulo de interpretación de gestos. Esto implica que se lleva a cabo una traducción de los gestos a las longitudes de los pulsos PWM que se deben enviar al VANT.

Los gestos que identifica e interpreta el módulo son los que se detallan en la Tabla 6. Las posiciones de los brazos determinan las longitudes de los pulsos PWM.

Tabla 6: Asociación de gestos con las señales

Gesto del piloto	Señales del drone
Brazo derecho recto	• Acelerador = valor de aceleración • Alerón > 1500 • Elevador = 1500 • Timón = 1500
Brazo izquierdo recto	• Acelerador = valor de aceleración • Alerón < 1500 • Elevador = 1500 • Timón = 1500
Brazos derecho e izquierdo rectos	• Acelerador = valor de aceleración • Alerón = 1500 • Elevador > 1500 • Timón = 1500
Un brazo recto y el otro inclinado hacia abajo	• Acelerador = valor de aceleración • Alerón = 1500 • Elevador < 1500 • Timón = 1500
Brazo derecho inclinado hacia arriba	• Acelerador = valor de aceleración • Alerón = 1500 • Elevador = 1500 • Timón > 1500
Brazo izquierdo inclinado hacia arriba	• Acelerador = valor de aceleración • Alerón = 1500 • Elevador = 1500 • Timón < 1500
Brazos derecho e izquierdo inclinados hacia arriba	• Acelerador > valor de aceleración anterior • Alerón = 1500 • Elevador = 1500 • Timón = 1500
Brazos derecho e izquierdo inclinados hacia abajo	• Acelerador < valor de aceleración • Alerón = 1500 • Elevador = 1500 • Timón = 1500
Brazos abajo, en posición de reposo	• Acelerador = valor de aceleración • Alerón = 1500 • Elevador = 1500 • Timón = 1500

8.2.1.1.5 Envío de señales

Una vez se ha hecho la traducción, se conocen las características de las señales para el control del drone.

Por ello, en esta etapa, se actualizan los valores de longitud de los pulsos PWM según el movimiento indicado.

8.2.1.1.6 Fin de conexiones

En esta etapa, una vez más, intervienen la mayor parte de los módulos. Se liberan las conexiones establecidas entre las placas y con las cámaras, se apagan y se finaliza el programa.

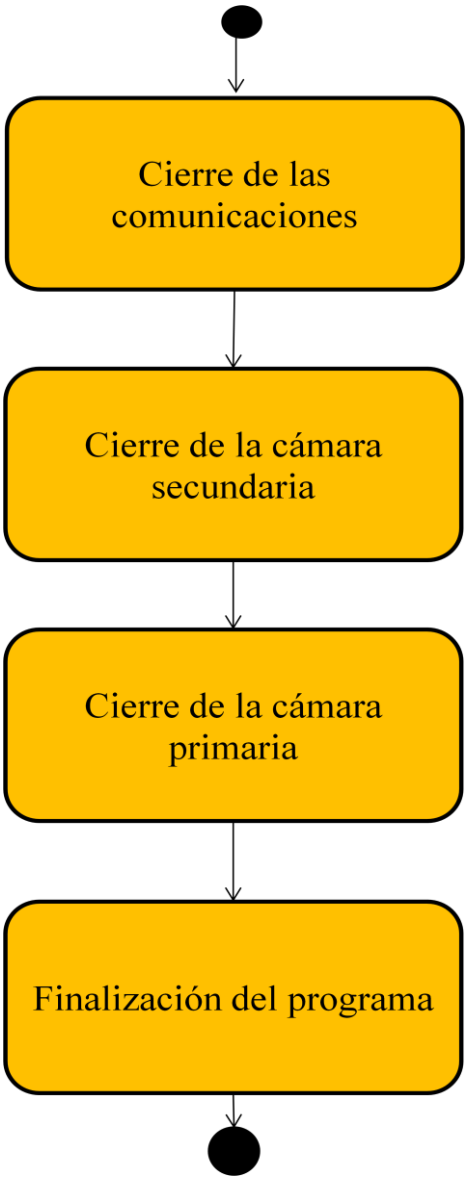


Figura 15: Diagrama de flujo de la etapa de fin de conexiones

Primero, se liberan las conexiones de comunicación con la placa. A continuación se lleva a cabo la liberación y cierre de las cámaras, esta vez en sentido inverso, primero la cámara secundaria y a continuación la principal.

Por último, se cierra el programa y se da por finalizado el proceso.

8.2.1.2 Software de control del drone

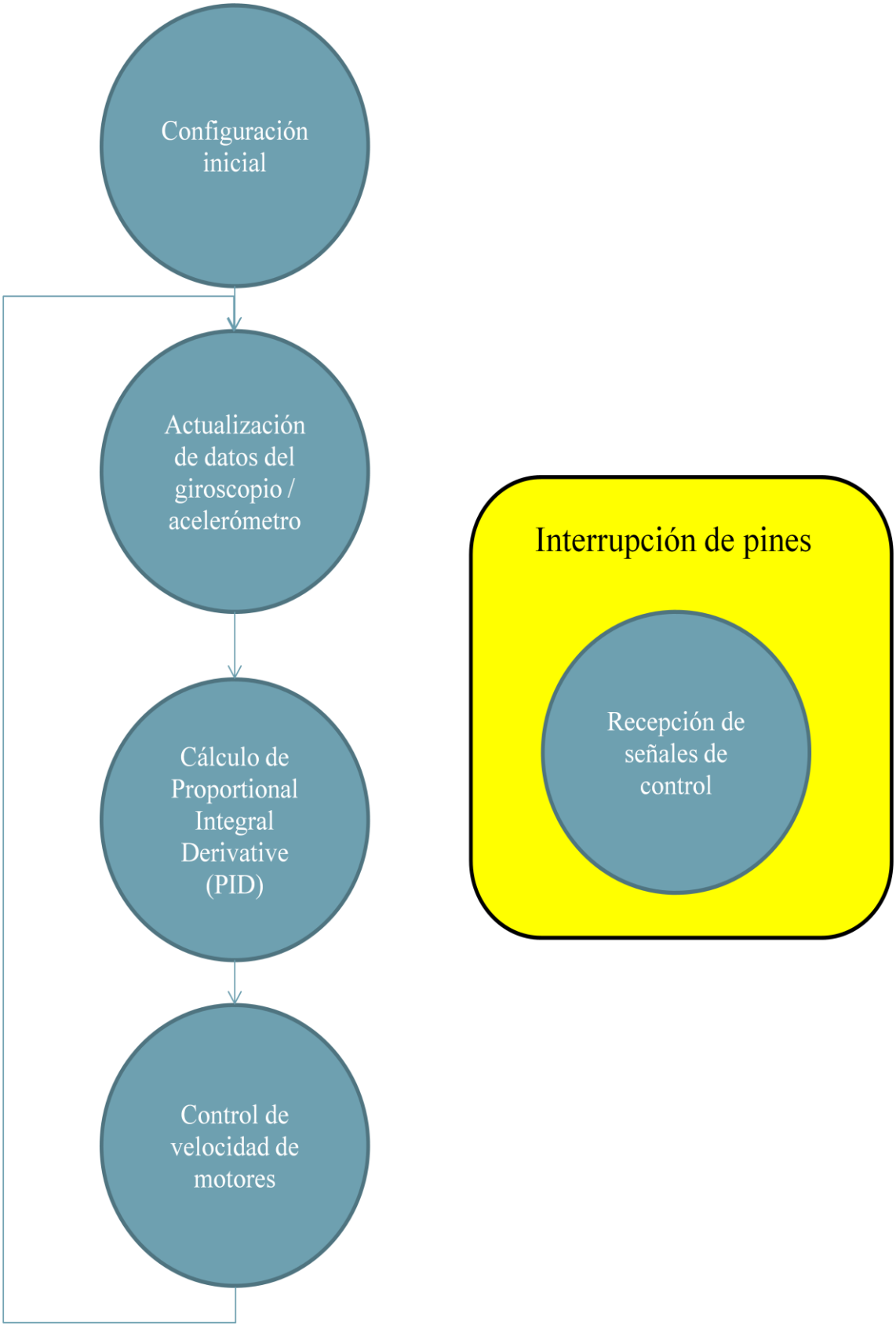


Figura 16: Diagrama de flujo del programa de control del drone

En la Figura 16 se puede ver diagrama de flujo del funcionamiento del programa de control del drone. El programa está íntegramente desarrollado en Arduino.

En base a los módulos que intervienen en el programa, las funciones que llevan a cabo y sus características, estos constan de lo siguiente:

8.2.1.2.1 Configuración inicial

En esta primera etapa intervienen los módulos de control de los elementos hardware: se realiza la calibración del giroscopio/acelerómetro y de los ESCs. Esta tiene lugar antes del bucle principal. También se activan las interrupciones y se indican los elementos que pueden hacer saltar la interrupción.

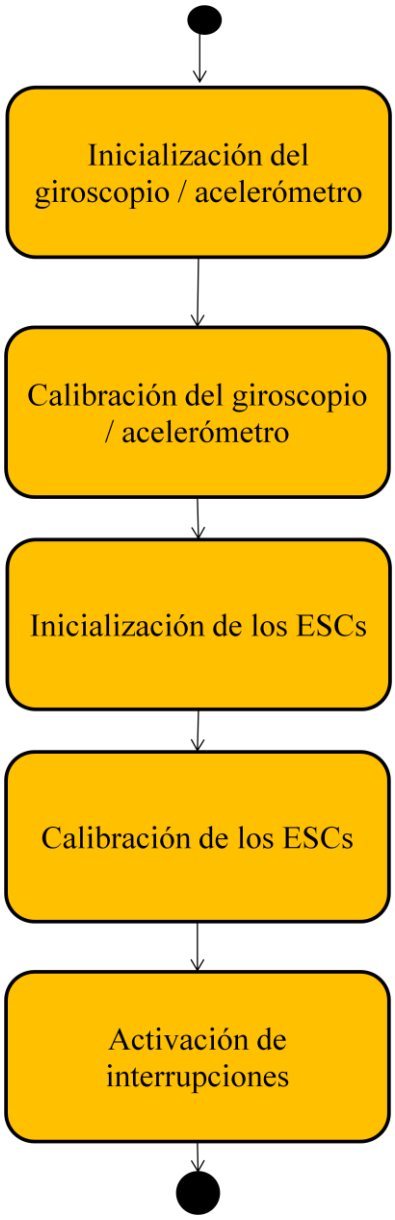


Figura 17: Diagrama de flujo de la etapa de configuración inicial

En la Figura 17 se puede ver el diagrama de flujo de funcionamiento de esta primera etapa.

Se comienza con el giroscopio / acelerómetro. Para inicializarlo, se definen los pines en los que se va a conectar y se configuran como entradas, para poder leer de ellas. Por otra parte, en la calibración se llevan a cabo una serie de medidas sobre la posición y giro respecto a los ejes X, Y y Z para así estimar las posiciones iniciales. Para llevar esto a cabo, el equipo no debe moverse y debe estar en un superficie recta, sobre la que va a despegar.

A continuación, se llevan a cabo las mismas funciones con los ESCs. Primero se inicializan definiendo los pines en los que se van a conectar los ESCs, uno por cada ESC, cuatro en este caso y se definen como salidas. Una vez definidos es necesario calibrarlos. Estos se calibran con la secuencia que sigue:

- Primero se indica el valor máximo del pulso que pueden recibir los ESCs generando pulsos PWM de longitud máxima.
- Y a continuación se indica el valor mínimo del pulso que pueden recibir generando pulsos PWM de longitud mínima.

Una vez calibrados, los ESCs, cuando reciban un pulso PWM entre los valores máximo y mínimo harán girar los motores.

Por último, en esta etapa se inician las interrupciones. En este caso se activan las interrupciones en un rango de pines, de tal forma que, si alguno de los pines del rango cambia de estado, salta la interrupción y se ejecuta una función concreta.

Es importante remarcar que el rango de pines que se activan para el escaneo de interrupciones debe coincidir con los pines a los que se conecten el mando radiocontrol o el módulo de control por gestos.

8.2.1.2.2 Actualización de datos del giroscopio / acelerómetro

En esta etapa toma parte el módulo de control del giroscopio/acelerómetro. En esta se actualiza la información sobre las variables de posición y giro del equipo y se calculan parámetros relevantes a partir de los valores obtenidos.

En cada ciclo de programa, se leen los datos de las entradas a las que está conectado el giroscopio/acelerómetro. A partir de los valores leídos se calculan parámetros como:

- La posición del equipo en los ejes X, Y, Z
- Los valores de giro
- Los ángulos de giro

8.2.1.2.3 Cálculo de Proportional Integral Derivative (PID)

En esta etapa se llevan a cabo las correcciones de las señales timón, alerón y elevador (row, yaw y pitch, en inglés) a partir de las que se reciben y las que se calculan con los valores del giroscopio. Esas correcciones se llevan a cabo mediante el cálculo del PID de cada una de las señales.

Los valores de PID se calculan según la siguiente fórmula:

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

Siendo e(t) la función de error entre la señal recibida por el módulo de control y la señal del giroscopio. Por ello, es necesario tener en cuenta los valores de los ángulos de giro indicados por el giroscopio / acelerómetro. Y los valores K indican el efecto del sumando sobre el resultado.

8.2.1.2.4 Control de velocidad de motores

A partir de los valores PID calculados y los valores recibidos, se controlan los valores de los pulsos PWM que se envían a los ESCs, de tal forma que estos respondan de forma estable y obedezcan a los movimientos del piloto. Se actualizan las longitudes de los pulsos PWM para ajustar la velocidad de cada motor de forma individual.

El valor de la longitud del pulso a comunicar a cada uno de los motores tiene como base el valor de aceleración recibido. Pero es necesario añadir o sustraer los valores de PID según se indica en la Tabla 7. Una vez conocidas las longitudes de los pulsos, se actualizan los valores de los pulsos PWM.

Tabla 7: Cálculo de la longitud del pulso para cada motor

Motor	Cálculo de longitud de pulso
Motor al frente a la derecha	Aceleración – PID_elevador + PID_timón - PID_alerón
Motor atrás a la derecha	Aceleración + PID_elevador + PID_timón + PID_alerón
Motor atrás a la izquierda	Aceleración + PID_elevador - PID_timón - PID_alerón
Motor al frente a la izquierda	Aceleración – PID_elevador - PID_timón + PID_alerón

8.2.1.2.5 Recepción de señales de control

A esta etapa se accede cada vez que se detecta un cambio en los puertos de entrada del dispositivo de control del drone. En éste se mide las longitudes de los pulsos PWM que se reciben.

Esta etapa se activa, mediante interrupciones, cada vez que se detecta un flanco de subida o de bajada en alguno de los pines del módulo de control conectado.

Para el cálculo de las longitudes de los pulsos, cada vez que salta la interrupción:

- Se comprueba si se ha producido algún cambio en los pines

Para cada pin, tal y como se puede ver de forma gráfica en la Figura 18:

- Si éste anteriormente estaba en estado bajo:
  - Se indica que actualmente se encuentra en estado alto
  - Se almacena el instante de inicio del pulso
- Si el pin estaba en estado alto:
  - Se calcula la longitud del pulso

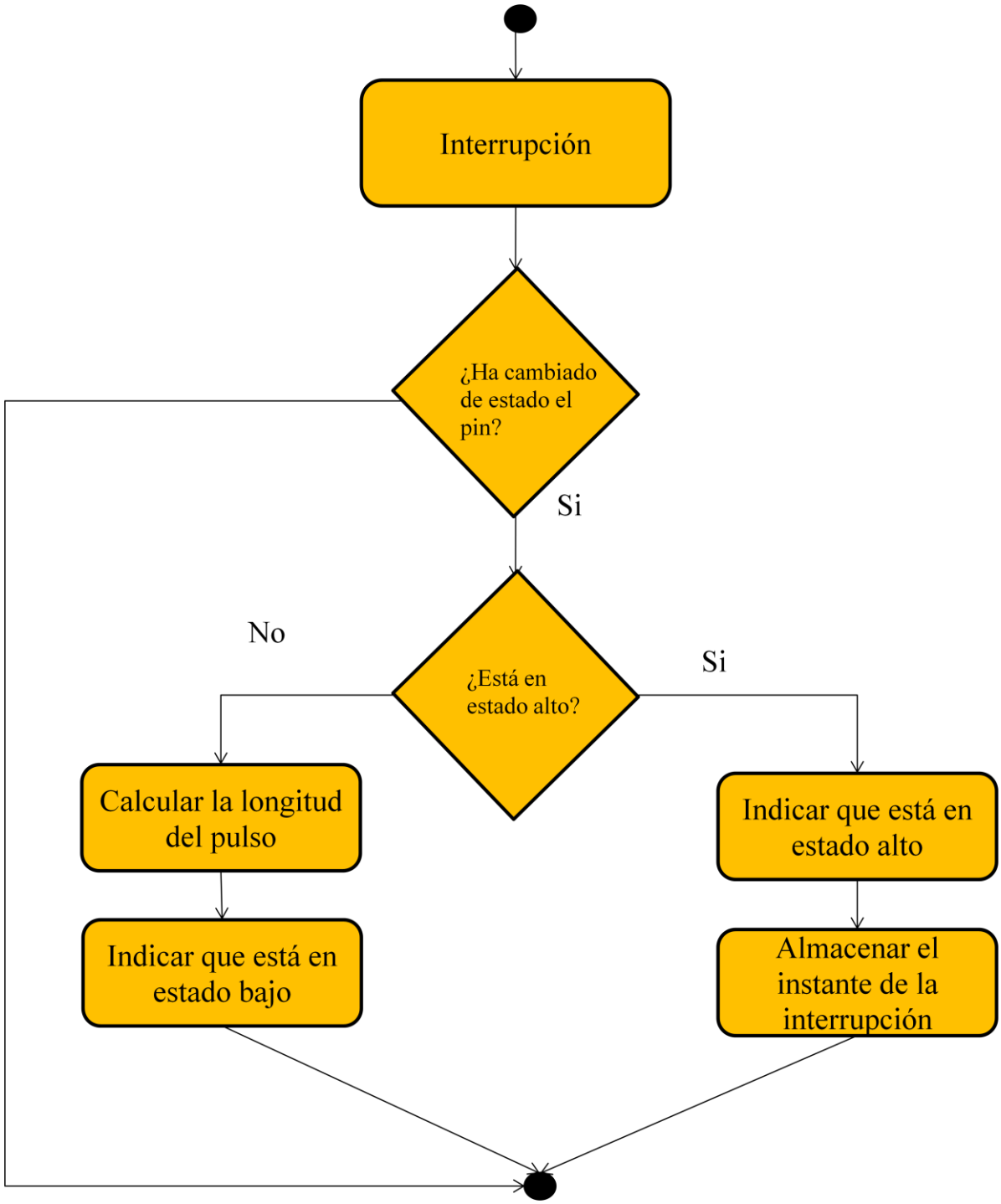


Figura 18: Diagrama de flujo de la comprobación de cada pin en la interrupción

8.2.2 Descripción de la solución

8.2.2.1 Software de reconocimiento de gestos

En base al diseño de programa, los módulos identificados constan de lo siguiente:

8.2.2.1.1 Control de las cámaras

Para el control de las cámaras, se hace uso de la clase *VideoCapture* de la librería OpenCV en C++. Los objetos de esta clase cuentan con métodos para la apertura, lectura y cierre de las cámaras.

Antes de nada, hay que tener en cuenta que la placa Raspberry Pi 3, únicamente cuenta con un único bus para el envío de las imágenes captadas por las cámaras al equipo. Por ello, para utilizar múltiples cámaras de forma simultánea, se divide la capacidad del bus para su uso por diferentes dispositivos.

Por supuesto, antes de iniciar el control, las cámaras deben estar conectadas al equipo, sino saltaría un error indicando que no es posible abrir la cámara. Cabe destacar la importancia del orden de conexión de estas. La primera cámara que se conecte es la denominada *cámara principal* y debe posicionarse a la izquierda. Por lo tanto, la otra cámara será la *cámara secundaria* y se coloca a la derecha. Estas se identifican con los índices 0 y 1 respectivamente.

Para poder controlar estos dispositivos, primero se crean los objetos de la clase *VideoCapture*, un objeto por cada cámara. Una vez creados, se asigna la capacidad del bus a cada uno de ellos y se procede con la apertura de las cámaras. Al finalizar se cierran las cámaras mediante el método *release*.

```
int main(int argc, char** argv)
{
    Mat frameL, frameR; //En estas variables se almacena lo que se
lee de las cámaras
    VideoCapture camL, camR;
    …
    camL = openCam(0); //Se abre la cámara izquierda
    if(camL.isOpened())
    {
        camR = openCam(1); //Se abre la cámara derecha
        if(camR.isOpened())
        {
            …
            camR.release();
            camL.release();
        }
        else
        {
            camL.release();
        }
    }
}
…
VideoCapture openCam(int index)
{
    VideoCapture cap;
//Se indica la velocidad de lectura de los frames, es decir, la
capacidad del bus
    cap.set(CV_CAP_PROP_FPS, 10);
```

```
    …

    //Apertura y comprobación de la cámara
    if(cap.open(index)==false)
    {
        …
        cap.release();
    }
    else
    {
        if(cap.read(frame) == false)
        {
            …
            cap.release();
        }
    }
    return cap;
}
```

Con las cámaras abiertas y comprobadas, en cada ciclo de programa, se procede a la lectura de los frames de la cámara. Hay que tener en cuenta que las imágenes de ambas cámaras deben ser casi síncronas. Por ello, en lugar de utilizar el método *read* del que dispone la clase, se utilizan operadores de desplazamiento.

```
camL >> frameL;
camR >> frameR;
```

8.2.2.1.2 Detección de gestos

Una vez obtenidas las imágenes de la cámara, se detecta si hay una persona en estas y, en caso afirmativo, se identifican los gestos que esté realizando. Pero antes de procesar las imágenes, es necesario pre-tratarlas para filtrar información innecesaria.

Primero se difuminan las imágenes para suavizar texturas, con el método *GaussianBlur*, que puedan afectar a la identificación del objeto más cercano. Y después se cambia el mapa de color a blanco y negro, mediante *cvtColor*, para llevar a cabo el mapa de disparidades.

```
Mat imgTreatment (Mat frame, int tam)
{
    Mat gray = frame;
    GaussianBlur(frame, gray, Size(tam,tam),0,0);
    cvtColor(gray, gray, COLOR_BGR2GRAY);
    return gray;
}
```

Para atender a las características de profundidad de la escena, se hace uso de la clase *StereoBM*. Se crea un objeto *Ptr<cv::StereoBM>* que, una vez configurado, puede crear el mapa de disparidades, que posteriormente se procesa para que se puedan distinguir los valores de profundidad. Para ello, se utiliza el método *normalize*.

```
…
Mat grayL, grayR, disparity;
Ptr<cv::StereoBM> stereo;
…
```

```
stereo -> compute(grayL,grayR,disparity);
…
normalize(disparity, disparity, 0, 255, CV_MINMAX, CV_8U);
```

A continuación se lleva a cabo un filtrado de ciertos valores de profundidad para identificar el objeto más cercano a la cámara, mediante la función *threshold*, y se comprueba que el tamaño es suficiente para que se pueda identificar como una persona.

```
int found = 0, thL = 250;
while(found != 1 && thL>=0)
{
    threshold(disparity, disp, thL, 255, THRESH_BINARY);
    for(int I = 0; I < disp.rows; i++)
    {
        for(int j = 0; j < disp.cols; j++)
        {
            if(disp.at<int>(I,j) == 255)
            {
                num++;
            }
        }
    }
    if(num >= (disp.rows*disp.cols)/200)
    {
        found = 1;
    }
    thL = thL – 5;
    num = 0;
}
```

Con el objeto identificado, se procede a la detección de la posición de los brazos. Los brazos se consideran una proporción del cuerpo, del treinta por ciento en este caso. A partir de esto, se escanea la imagen por columnas hasta obtener los supuestos brazos y se separan en imágenes individuales. Con esas imágenes, se obtiene el contorno de los brazos.

```
//Esta función obtiene el contorno de los brazos
vector< Point > getArmContour(Mat img, Mat *armImg, int arm, double
areaMax, vector< Vec4i > *hierarchy)
{
    Mat temp;
    double area =0;
    int I, lim;
    vector< Point > cont;
    areaMax = areaMax * 0.3;
    if(arm ==0)//Brazo izquierdo
    {
        i=1;
        lim = img.cols;
    }
    else //Brazo derecho
    {
        i=img.cols -1;
        lim = 0;
    }
```

```
    while (( area < areaMax) && i != lim)
    {

        if(arm==0)
        {

            img.colRange(0, i).copyTo(temp);

            i++;
        }
        else
        {
            img.colRange(i, img.cols).copyTo(temp);
            i--;
        }
        cont =getContour(temp, hierarchy);
        if(cont.size()>0)
        {
            area = contourArea(cont);
        }
    }
    temp.copyTo(*armImg);

    return cont;
}
```

Si se obtiene el rectángulo de menor área, empleando el método *minAreaRect* que devuelve un objeto de clase *RotatedRect*, que lo encierra es posible averiguar la posición de los brazos.

```
if(contour.size() > 0)
{
    RotatedRect minRect =minAreaRect( Mat(contour) );

    Point2f point[4];
    minRect.points(point);
 …

int recogniseGesture(RotatedRect rect)
{
    //Point2f point[4];
    //rect.points(point); //El orden de los puntos es: InfIzquierdo
SupIzquierdo SupDerecho InfIzquierdo
    //printf("angulo: %f", rect.angle);
    int pos = 0;
    if(rect.angle >= -5)
    {
        //printf("El brazo está recto\n");
        pos = 1;
    }
    else
    {
        if(rect.angle >= -45)
        {
            //printf("El brazo está inclinado hacia arriba\n");
            pos = 2;
        }
        else
        {
```

```
            if(rect.angle >= -85 )
            {
                //printf("El brazo está inclinado hacia abajo\n");
                 pos = 3;
            }
            else
            {
                //printf("El brazo está hacia arriba\n");
            }
        }
    }

    return pos;
}
```

8.2.2.1.3 Interpretación de gestos

Conocida la posición de los brazos, se lleva a cabo la interpretación de los mismos. Esto se va a ver con un ejemplo; si el brazo derecho se encuentra extendido horizontalmente y el izquierdo en posición vertical, se identifica que el drone debe moverse a la derecha. Esto es equivalente a que, en el mando radiocontrol, el joystick derecho se mueva a la derecha. Es decir, a que la señal alerón tenga una duración de pulso mayor.

8.2.2.1.4 Envío de señales

Para el envío de señales, es necesario que el programa acceda a los pines GPIO de la placa Raspberry Pi 3 y envíe pulsos PWM a través de ellos. Para ello, se hace uso de las librerías "wiringPi.h" y "softPwm.h" respectivamente.

Respecto a la librería "softPwm.h", cabe destacar que cada ciclo de PWM dura 10 milisegundos y que por defecto se divide en intervalos de 100 microsegundos. En este caso, es necesario crear pulsos entre 1000 y 2000 microsegundos. Por lo tanto, para definirlos, será necesario indicar el número de subdivisiones de 100 microsegundos van a estar en nivel alto, entre 10 y 20.

La numeración de los pines seguida para el acceso a los mismos es el que ve en la Figura 19 [20].

P1: The Main GPIO connector							
WiringPi Pin	BCM GPIO	Name	Header		Name	BCM GPIO	WiringPi Pin
		3.3v	1	2	5v		
8	Rv1:0 - Rv2:2	SDA	3	4	5v		
9	Rv1:1 - Rv2:3	SCL	5	6	0v		
7	4	GPIO7	7	8	TxD	14	15
		0v	9	10	RxD	15	16
0	17	GPIO0	11	12	GPIO1	18	1
2	Rv1:21 - Rv2:27	GPIO2	13	14	0v		
3	22	GPIO3	15	16	GPIO4	23	4
		3.3v	17	18	GPIO5	24	5
12	10	MOSI	19	20	0v		
13	9	MISO	21	22	GPIO6	25	6
14	11	SCLK	23	24	CE0	8	10
		0v	25	26	CE1	7	11
WiringPi Pin	BCM GPIO	Name	Header		Name	BCM GPIO	WiringPi Pin

Figura 19: Numeración de los pines de Raspberry Pi

Antes de nada, es necesario iniciar la comunicación a través de los pines. Se definen los pines a utilizar, se inicia la configuración por defecto, se indican que los pines son salidas, con el método *pinMode*, y se crean los pulsos PWM con la longitud que van a tener y el valor con el que se van a iniciar, empleando el método *softPwmCreate*.

```
//Se definen los pines a utilizar para el envío de señales
#define THROTTLE 7
#define ROLL 0
#define YAW 2
#define PITCH 3
…
void initCommunication(int val, int len)
{
    wiringPiSetup(); //Se inicializan los pines con la
configuración por defecto
    //Se configuran los pines a utilizar
    pinMode(THROTTLE,OUTPUT);
    pinMode(ROLL,OUTPUT);
    pinMode(YAW,OUTPUT);
    pinMode(PITCH,OUTPUT);
    //Se inician los pulsos PWM
    softPwmCreate(THROTTLE, val ,len);
    softPwmCreate(ROLL, val ,len);
    softPwmCreate(YAW, val ,len);
    softPwmCreate(PITCH, val ,len);
}
```

Una vez iniciada la comunicación, en cada ciclo de programa, se actualizan los valores de los pulsos según los gestos que se detecten, mediante la función *softPwmWrite*. Por ejemplo, en la posición de reposo, los pulsos de las señales alerón, elevador y timón (yaw, pitch y roll) tendrán una duración de 1500 microsegundos, el valor de acelerador depende del valor que le haya dado el piloto. Sin embargo, si el movimiento es hacia la izquierda, la señal alerón tendrá un valor menor de 1500.

```
void sendSignal(char c) //c indica el movimiento a realizar
{
    //printf("Sendinf %c\n", c);
    if(c=='a') //reposo
    {
        softPwmWrite(THROTTLE, 10);
        softPwmWrite(ROLL, 15);
        softPwmWrite(YAW, 15);
        softPwmWrite(PITCH, 15);
    }
    else
    {
        if(c=='l') //izquierda
        {
            softPwmWrite(THROTTLE, 10);
            softPwmWrite(ROLL, 10);
        }
        else
        {
            if(c=='r') //derecha
            {
                softPwmWrite(THROTTLE, 10);
```

```
                softPwmWrite(ROLL, 20);
            }
            else
            {
                softPwmWrite(THROTTLE, 10);
                softPwmWrite(ROLL, 15);
            }
        }
    }
}
```

8.2.2.2 Software de control del drone

A partir del diseño del programa de control del drone y los elementos adquiridos, que se detallan en el presupuesto, el programa se desarrolla de la siguiente forma:

8.2.2.2.1 Control de la recepción de señales

En esta parte se utilizan las interrupciones para atender de forma inmediata a las señales que se reciben. Estas es necesario activarlas e indicar los casos en los que salta la interrupción. En este caso estas se deben activas cuando hay un cambio, un flanco de subida o de bajada, en uno de los pines a los que está conectado el receptor o el módulo de control que se esté utilizando.

```
…
void setup(
{
    …
   /* Se activan las interrupciones para la recepción de las
señales de control */
  PCICR |= (1 << PCIE0);
  PCMSK0 |= (1 << PCINT0);  //Se activa la interrupción si hay un
cambio en el PIN 8
  PCMSK0 |= (1 << PCINT1);  //Se activa la interrupción si hay un
cambio en el PIN 9
  PCMSK0 |= (1 << PCINT2);  //Se activa la interrupción si hay un
cambio en el PIN 10
  PCMSK0 |= (1 << PCINT3);  //Se activa la interrupción si hay un
cambio en el PIN 11
    …
}
…
```

Cuando salte una interrupción se ejecuta una función con el nombre *ISR(<evento que hace saltar la interrupción>)*. Teniendo en cuenta que en este se está observando si hay algún cambio en cierto rango de pines, el evento es *PCINT0_vect*.

Cada vez que una de las entradas pasa a estado alto, ese cambio se indica con un flag y se almacena el instante de inicio del pulso. Y cuando una entrada pasa a estado bajo, se calcula la diferencia entre ese instante y el de inicio para almacenar la longitud del pulso y se cambia el estado del flag.

```
…
//La duración de los pulsos del receptor se va a almacenar en estas
variables
int channel1; //row
int channel2; //pitch
```

```
int channel3; //throttle
int channel4; //yaw

byte fl1, fl2, fl3, fl4; //flags
unsigned long tm1, tm2, tm3, tm4;
…
ISR(PCINT0_vect)
{
  if (fl1 == 0 && PINB & B00000001)
  {
    fl1 = 1;
    tm1 = micros();
  }
  else if (fl1 == 1 && !(PINB & B00000001))
  {
    fl1 = 0;
    channel1 = micros() – tm1;
  }
  if (fl2 == 0 && PINB & B00000010)
  {

    fl2 = 1;
    tm2 = micros();
  }
  else if (fl2 == 1 && !(PINB & B00000010))
  {
    fl2 = 0;
    channel2 = micros() – tm2;
  }
  if (fl3 == 0 && PINB & B00000100)
  {
    fl3 = 1;
    tm3 = micros();
  }
  else if (fl3 == 1 && !(PINB & B00000100))
  {
    fl3 = 0;
    channel3 = micros() – tm3;
  }
  if (fl4 == 0 && PINB & B00001000)
  {
    fl4 = 1;
    tm4 = micros();
  }
  else if (fl4 == 1 && !(PINB & B00001000))
  {
    fl4 = 0;
    channel4 = micros() – tm4;
  }
}
```

8.2.2.2.2 Control del giroscopio / acelerómetro

Por una parte, el almacenamiento y actualización de la información referente al giroscopio/acelerómetro se lleva a cabo mediante la librería "MPU6050_tockn.h" [19]. Esta librería, a su vez, depende de "Wire.h" la cual también será necesario incluir.

```
#include <MPU6050_tockn.h>
#include <Wire.h>
```

Una vez incluidas, se crea un objeto de la clase *MPU6050* y se inicializa con un objeto de la clase *Wire*:

```
MPU6050 gyro(Wire);
```

Antes de comenzar con la lectura de los datos es necesario inicializar los elementos, tanto *Wire* como el objeto de la clase *MPU6050*. Y una vez inicializados se calibra el dispositivo con el método *calcGyroOffsets*.

```
void setup(
{
    …
    Wire.begin();
    gyro.begin();
    gyro.calcGyroOffsets(false);
    /*En caso de querer mostrar los datos por Serial se
    pasaría como parámetro "true" en lugar de "false"*/
    …
}
```

Una vez inicializado el módulo se procede a su uso en el programa. Los datos que se van a utilizar para estabilizar el vuelo del drone son los ángulos de rotación respecto a los ejes X, Y y Z, ya que las señales alerón, timón y elevador se ven afectadas por ello. Para la obtención de estos valores, el objeto de la clase *MPU6050* cuenta con los métodos *getAngleX*, *getAngleY* y *getAngleZ*. Sin embargo, estos valores no se actualizan automáticamente, por ello, antes de obtener los valores en cada ciclo de programa se realiza una llamada al método *update* del objeto.

```
…
gyro.update();
rollTemp = gyro.getAngleY();
yawTemp = gyro.getAngleX();
pitchTemp = gyro.getAngleZ();
…
```

8.2.2.2.3 Cálculo de PID

Para asegurar la estabilidad del vuelo, las señales timón, alerón y elevador (row, yaw y pitch) deben sufrir ciertas correcciones. Se aplica la misma corrección con todas las señales, por ello, en los ejemplos se va a ver una única señal.

El cálculo mediante la fórmula habitual puede resultar computacionalmente inviable si se lleva a cabo de esta forma. Por ello, se ha optado por simplificarlo [2].

En cada ciclo de programa n, para cada señal, se llevan a cabo las siguientes operaciones:

$$P_n = K_p\,(giroscopio_n - receptor_n)$$

$$I_n = I_{n-1} + K_i(giroscopio_n - receptor_n)$$

$$D_n = K_d\big(giroscopio_n - receptor_n - (giroscopio_{n-1} - receptor_{n-1})\big)$$

$$PID_n = P_n + I_n + D_n$$

```
// Esto es un ejemplo para la señal row
// Estos son los valores K, hay que ajustarlos para cada drone
#define PGAINR 1.3
#define IGAINR 0.04
#define DGAINR 18.0

float iRoll = 0;
float rollErrorPrev = 0;
float rollPid = 0;
…
void calcPID()
{
    float rError, rollTemp;
    …
    rError = rollTemp – channel1;
    iRoll = iRoll + (rError * IGAINR);
    …
    rollPid = rError * PGAINR + iRoll + (rError – rollErrorPrev) *
DGAINR;
  rollErrorPrev = rError;
}
```

8.2.2.2.4 Control de los ESCs

Para el control de los motores, se hace uso de la librería de Arduino "Servo.h". Primero se crean los objetos de la clase *Servo*. Hay que crear un objeto por cada ESC que se va a controlar, en este caso cuatro. También es necesario definir los pines de la placa a los que se van a conectar. Se va a hacer la siguiente asociación:

- Motor 1 = motor al frente a la derecha
- Motor 2 = motor atrás a la derecha
- Motor 3 = motor atrás a la izquierda
- Motor 4 = motor al frente a la izquierda

```
#include <Servo.h>
#define M1PIN 4 //Pin al que se conecta el ESC del motor 1
#define M2PIN 5 //Pin al que se conecta el ESC del motor 2
#define M3PIN 6 //Pin al que se conecta el ESC del motor 3
#define M4PIN 7 //Pin al que se conecta el ESC del motor 4
Servo m1, m2, m3, m4;
```

Una vez definido, se procede a la inicialización. Primero se relaciona el objeto *Servo* con el pin de la placa que le corresponde con el método *attach(<pin de la placa>)*. Una característica de los ESCs es que también es necesario calibrarlos, es decir darles a conocer las longitudes

máxima y mínima de los pulsos PWM que van a recibir. El procedimiento para calibrar los ESCs adquiridos consta de dos pasos:

1. Mandar a los ESCs la señal de aceleración máxima, pulsos PWM cuya duración en estado alto es 2000 microsegundos.
2. Tras un tiempo, mandar a los ESCs la señal de aceleración mínima, pulsos PWM cuya duración en estado alto es de 1000 microsegundos.

Para el envío de los pulsos PWM a los motores se utiliza el método *writeMicroseconds(<longitud de pulso en alto>)*.

```
void setup(
{
    …
    //Se asocian los ESCs con sus respectivos pines
    m1.attach(M1PIN);
    m2.attach(M2PIN);
    m3.attach(M3PIN);
    m4.attach(M4PIN);

    //Calibración de los ESCs
    m1.writeMicroseconds(2000);
    m2.writeMicroseconds(2000);
    m3.writeMicroseconds(2000);
    m4.writeMicroseconds(2000);
    delay(5000); //Se espera tiempo suficiente


    m1.writeMicroseconds(1000);
    m2.writeMicroseconds(1000);
    m3.writeMicroseconds(1000);
    m4.writeMicroseconds(1000);
    delay(5000); //Se espera tiempo suficiente
    …
}
```

Una vez calibrados, se inician los motores en el momento en el que se envíe el siguiente pulso.

En cada ciclo del programa se actualizan la velocidad de los motores a partir de los valores PID calculados y la señal de aceleración. Una vez calculados, se genera el pulso.

```
void writeMotorSignals()
{
    int valM1, valM2, valM3, valM4;
    float throttle = channel3;
    /* La velocidad de los motores se ve afectada por los valores
    de PID */
    valM1 = throttle – pitchPid + rollPid – yawPid;
    valM2 = throttle + pitchPid + rollPid + yawPid;
    valM3 = throttle + pitchPid – rollPid – yawPid;
    valM4 = throttle – pitchPid – rollPid + yawPid;
    …
    m1.writeMicroseconds(valM1);
    m2.writeMicroseconds(valM2);
    m3.writeMicroseconds(valM3);
    m4.writeMicroseconds(valM4);
}
```

9 Planificación del trabajo

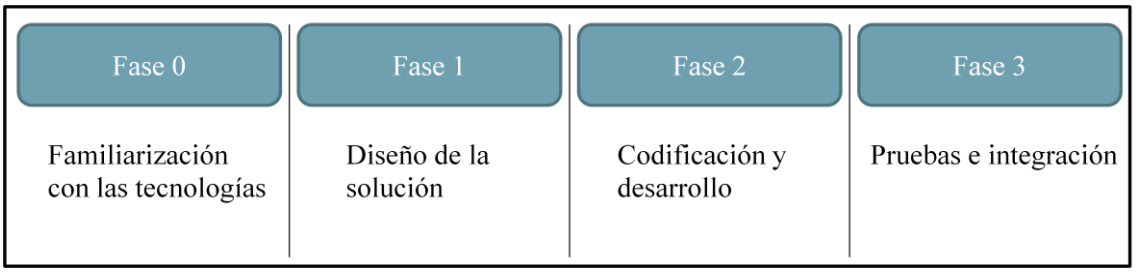


Figura 20: Planificación del proyecto por fases

En la Figura 20 se aprecia la planificación del proyecto dividida por fases. Estas son:

- **Fase 0**: esta es una fase previa en la que se concentra la labor de formación. Se ha visto necesario añadir esta fase debido a la actualidad de las tecnologías empleadas y la falta de conocimientos inicial para poder llevar a cabo de forma satisfactoria el resto de etapas del proyecto.
- **Fase 1**: en esta fase se definen los requisitos del proyecto y las especificaciones. Además se realiza el diseño de la solución, tanto a alto como a bajo nivel.
- **Fase 2**: en esta etapa se lleva a cabo la codificación del programa y se preparan los elementos para la implementación.
- **Fase 3**: se comprueba que la solución cumple con los requerimientos especificados y que lleva a cabo sus funciones con éxito. Se llevan a cabo pruebas tanto de los módulos individuales como del sistema completo.

Como se puede observar, las funciones que se llevan a cabo en las fases 1, 2 y 3 concuerdan con las etapas de un proyecto software. Por ello, se pueden representar tal y como se muestra en la Figura 21.

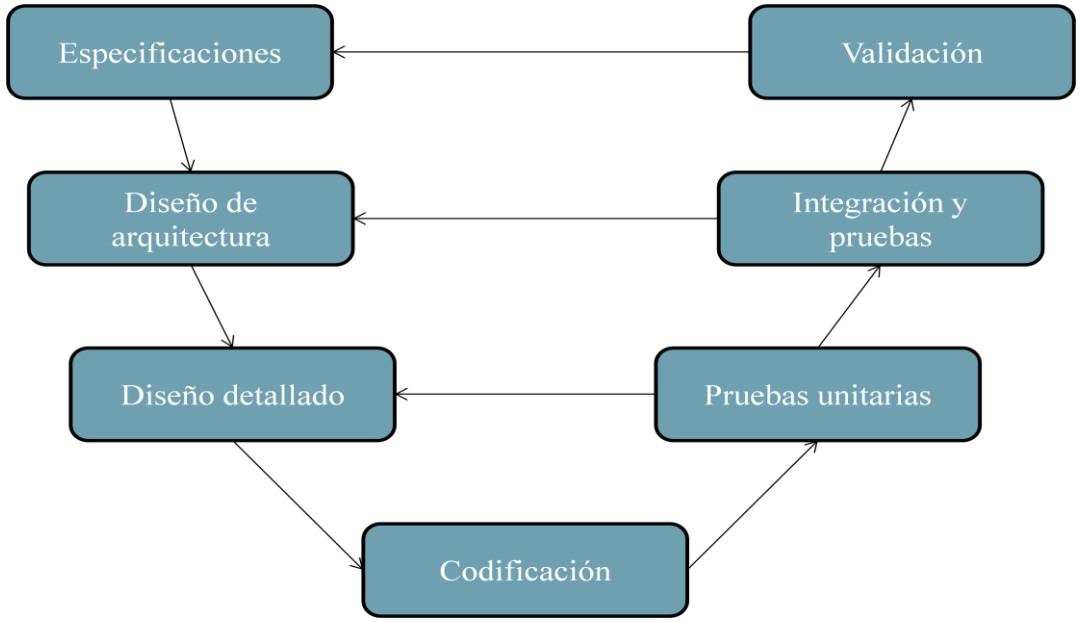


Figura 21: Modelo en V de las etapas del desarrollo software

En este caso el modelo en V se ha considerado adecuado para el desarrollo de este proyecto debido a la clara separación de las etapas y la posibilidad de realimentación.

Tabla 8: Tareas planificadas

Código	Nombre	Descripción
P.T.0	**Familiarización con el ámbito**	**Formación sobre el funcionamiento de drones, técnicas de detección de objetos e implementaciones y usos actuales**
T.0.1	Funcionamiento de drones	Formación sobre información general y funcionamiento de drones
T.0.2	Técnicas de detección de objetos	Formación sobre información general de técnicas de detección de objetos
T.0.3	Usos actuales e implementación de las tecnologías	Formación sobre usos actuales y métodos de implementación de las tecnologías a tratar
P.T.1	**Especificaciones**	**Definición de las funciones del programa y los requisitos que debe cumplir**
T.1.1	Definición de funciones	Definición de la funcionalidad de la solución
T.1.2	Definición de requisitos	Especificación de requisitos a cumplir por la solución para ser considerada como válida
T.1.3	Diseño de las pruebas de validación	Diseño de las pruebas de validación de la solución
P.T.2	**Diseño de la solución**	**Diseño a alto y a bajo nivel de la solución propuesta**
T.2.1	Diseño de arquitectura	Diseño general de la solución
T.2.2	Diseño detallado	Diseño de los componentes individuales de la solución
T.2.3	Diseño de las pruebas unitarias y de integración	Diseño de las pruebas de funcionamiento de los módulos y del sistema
P.T.3	**Desarrollo de la solución**	**Desarrollo de la solución diseñada**
T.3.1	Codificación de componentes software	Desarrollo de los componentes software de la solución
T.3.2	Montaje de los elementos hardware de la solución	Montaje de los componentes hardware de la solución
P.T.4	**Pruebas e integración**	**Comprobación del correcto funcionamiento de los elementos, tanto individuales como del sistema completo**
T.4.1	Preparación de las pruebas	Elaboración o preparación de los elementos necesarios para llevar a cabo las pruebas
T.4.2	Pruebas unitarias	Pruebas de funcionamiento de los componentes individuales
T.4.3	Pruebas de integración	Pruebas de funcionamiento del sistema
P.T.5	**Validación de la solución**	**Comprobación del cumplimiento de requisitos de la solución**
T.5.1	Pruebas de validación	Pruebas de comprobación de los requisitos del sistema
P.T.6	**Documentación**	**Elaboración de la documentación necesaria durante todo el proceso**
T.6.1	Documentación de seguimiento	Elaboración de documentos necesarios para el seguimiento del proceso
T.6.2	Documentación final	Elaboración de documentos para la entrega

Figura 22: Diagrama Gantt de las tareas del proyecto

En la Tabla 8 se puede apreciar la planificación de tareas del proyecto. En ella se indica el nombre de la tarea y la descripción así como un código de identificación. Las tareas se han planificado en base a las etapas previamente identificadas. En la Figura 22 se encuentra el diagrama Gantt en el que se ve, de forma gráfica, la duración estimada de las tareas y el instante de comienzo de cada una de ellas. El diagrama está acompañado de los códigos de las tareas y la duración de las mismas.

10 Presupuesto

El presupuesto está dividido en las partidas: gastos, amortizaciones, horas internas, subcontrataciones, costes indirectos e imprevistos. En este caso, la partida de subcontrataciones tiene un valor de 0, los costes indirectos se consideran el 10 % de los costes directos y los imprevistos son el 5 % de los costes totales. Teniendo lo anterior en cuenta, el presupuesto es el que se muestra en la Tabla 9, Tabla 10, Tabla 11 y Tabla 12:

Tabla 9: Partida de horas internas

Trabajador	Coste Horario	Horas	Coste total
Ingeniero técnico	15 €/h	303 h	4545,00 €
Ingeniero superior	60 €/h	30 h	1800,00 €
		TOTAL	**6345,00 €**

En el Anexo IV – Desglose de horas internas dedicadas se puede ver el desglose de las horas dedicadas a cada tarea.

Tabla 10: Partida de amortizaciones

Concepto	Coste de adquisición	Vida útil	Tiempo de uso en el proyecto	Amortización
Ordenador	1000 €	8 años	190 h	9,12 €
			TOTAL	**9,12 €**

Nota: en la vida útil se supone que en 8 años se utiliza 365 días 8 horas

Tabla 11: Partida de gastos

Concepto	Coste
Raspberry Pi 3	42,32 €
2 Cámaras Logitech	34,30 €
Placa Arduino Uno	7,99 €
4 x (Motores A2212 (1000 kV, sin escobillas) + ESC + Hélices 10*4.5)	54,99 €
Mando radiocontrol + receptor	53,99 €
Giroscopio/Acelerómetro MPU6050	1,89 €
Marco de cuadrocóptero	19,99 €
Batería de litio 3S	21,00 €
Cargador de batería de Litio	11,00 €
Hélices 10*4.5’’	6,78 €
TOTAL	**244,25 €**

Tabla 12: Resumen del presupuesto

Partida	Coste
Horas internas	6345,00 €
Amortizaciones	9,12 €
Gastos	244,25 €
SUBTOTAL	**4798,37 €**
Costes indirectos (10 %)	479,84 €
Imprevistos (5%)	263,91 €
TOTAL	**7342,12 €**

El presupuesto, dado el tiempo dedicado a la elaboración del proyecto y lo que se ofrece, resulta en una cantidad aceptable, 7342,12 €.

11 Conclusiones

La solución desarrollada presenta una forma de manejo complementaria al control por radiocontrol para un drone basado en Arduino. Ésta contempla un control por gestos, que se llevan a cabo con los brazos, y que, en la medida de lo posible, resultan intuitivos. La solución es capaz de identificar las posiciones de los brazos y, a partir de éstas, interpretar el movimiento que debe realizar el drone. Además, no requiere de un marcador, o que el piloto disponga de algún elemento adicional para el control, por lo que es posible un manejo cooperativo, mediante múltiples pilotos.

Por una parte, el Software de reconocimiento de gestos funciona correctamente. Es capaz de identificar la posición de los brazos y a partir de estas generar las señales pertinentes para el control del drone. El tiempo de ejecución del bucle principal aumenta ligeramente si se muestran las imágenes del proceso de detección en pantalla, pero, no afecta al envío de las señales, por lo que en principio resulta viable para el manejo del drone.

Por otra parte, el Software de control del drone gestiona correctamente las señales que recibe y genera las señales adecuadas. Lee correctamente la información del módulo del giroscopio/acelerómetro, gestiona correctamente las señales que recibe tanto si se trata de un mando radiocontrol como si se trata del programa de reconocimiento que se ha desarrollado y, en base a estos, genera las señales para controlar la velocidad de los motores correctamente.

Finalmente, utilizando ambos programas de forma conjunta, es posible apreciar los cambios en la velocidad de los motores según los gestos que se llevan a cabo frente a la cámara. Los cambios de velocidad son coherentes con los movimientos que se espera que lleve a cabo el drone, por lo que se puede concluir que el sistema funciona correctamente.

Observando los aspectos tecnológicos, cabe remarcar que actualmente no es posible encontrar proyectos que hayan integrado reconocimiento de objetos en drones. Por ello, este proyecto puede resultar de utilidad en el avance de estas tecnologías y puede impulsar el desarrollo de nuevos proyectos en los que se utilicen de forma conjunta.

Además, se ha conseguido una forma de control que otorga mayor seguridad, en caso de que se pierda el control del equipo, y que, mediante el control cooperativo, aumenta las capacidades del equipo que pueden resultar útiles en ámbitos como servicios públicos de salvamento.

Demostrada la viabilidad de la integración, se puede concluir que dotando a estos equipos de cierta inteligencia, como la que otorgan las técnicas de reconocimiento de objetos, a futuro, se podrían desarrollar nuevos usos, más allá del manejo del equipo, muy beneficiosos, sobre todo para operaciones de rescate, búsqueda o salvamento.

12 Referencias

[1] Global Mediterránea Geomática. Usos actuales y a futuro de los drones: http://www.globalmediterranea.es/uso-drones-la-actualidad-futuro

[2] Project YMFC-AL - The Arduino auto-level quadcopter (Disponible online): http://www.brokking.net/ymfc-al_main.html (Último acceso 02/06/2018)

[3] The Drone Pi (Disponible online): http://www.instructables.com/id/The-Drone-Pi/ (Último acceso 01/06/2018)

[4] Avansing. Smart Drones. Tecnología revolucionaria en drones autónomos: http://www.avansig.com/smartdrones/ (Último acceso: 04/06/2018)

[5] Documentación de OpenCV versión 3.4.0 (Disponible online): https://docs.opencv.org/3.4.0/ (Último acceso: 30/05/2018)

[6] A Survey on Real Time Object Detection, Tracking and Recognition in Image Processing (Disponible online): https://pdfs.semanticscholar.org/49d3/fe473a78b04b9902828ebecc36ce43506fae.pdf

[7] How the PlayStation camera works (Disponible online) https://electronics.howstuffworks.com/playstation-camera.htm

[8] Histogram of Oriented Gradients (Disponible online): https://www.learnopencv.com/histogram-of-oriented-gradients/

[9] Rapid Object Detection using a Boosted Cascade of Simple features (Disponible online): http://wearables.cc.gatech.edu/paper_of_week/viola01rapid.pdf

[10] Histograms of Oriented Gradients for Human Detection (Disponible online): https://lear.inrialpes.fr/pubs/2005/DT05/hog_cvpr2005.pdf

[11] Application of Soft Histogram of oriented Gradient on Traffic Sign Detection (Disponible online): https://ieeexplore.ieee.org/document/7734067/

[12] Object recognition using the OpenCV Haar cascade-classifier on the iOS platform (Disponible online): http://www.diva-portal.org/smash/get/diva2:601707/fulltext01.pdf

[13] How Motion Detection Works in Xbox Kinect: https://www.wired.com/2010/11/tonights-release-xbox-kinect-how-does-it-work/ (Último acceso: 05/06/2018)

[14] 10 Thermal Vision Cameras For Drones And How Thermal Imaging Works: https://www.dronezon.com/learn-about-drones-quadcopters/9-heat-vision-cameras-for-drones-and-how-thermal-imaging-works/ (Último acceso 05/06/2018)

[15] Características de la placa Rock64 (Disponible online): https://www.pine64.org/?page_id=7147 (Último acceso 06/06/2018)

[16] Descripción de Raspberry Pi 3 model b + (Disponible online): https://www.raspberrypi.org/products/raspberry-pi-3-model-b-plus/ (Último acceso: 30/05/2018)

[17] Relación de las señales con la información del giroscopio / acelerómetro: https://aviation.stackexchange.com/questions/16531/what-is-the-relation-between-roll-angle-and-pitch-angle

[18] Normativa estatal vigente sobre el vuelo de aeronaves (Disponible online): https://www.boe.es/buscar/doc.php?id=BOE-A-2017-15721

[19] Librería de Arduino de control del módulo MPU6050; "MPU6050_tockn.h": https://github.com/tockn/MPU6050_tockn (Último acceso 21/06/2018)

[20] Numeración de los pines de Raspberry Pi para utilizar con la librería wiringPi: http://wiringpi.com/pins/ (Último acceso 24/06/2018)

13 Anexo I - Pliego de condiciones

13.1 Descripción del objeto a tratar

La solución trata de un programa implementado en un módulo que se incorpora a un VANT para controlarlo mediante gestos. Para ello, el módulo, a partir de dos cámaras, capta imágenes que son procesadas por el programa para detectar los gestos del piloto y poder controlar el equipo.

13.2 Especificaciones

La solución debe cumplir los siguientes requisitos:

- No debe violar los aspectos de la normativa vigente.
- Debe ser complementaria al manejo por radiocontrol. Es necesario mantener la posibilidad de manejo habitual y que este sea prioritario.
- La señalización de la solución debe simular la señalización que se lleva a cabo en el manejo por radiocontrol.
- La implementación no debe tener un efecto tal que pueda suponer inestabilidad en el equipo, esto incluye que el rendimiento del programa no se vea afectado por el tiempo de respuesta del mismo.

13.3 Características del proceso

En este proyecto, aunque principalmente se trate de un proyecto software, es necesario tener en cuenta los elementos hardware sobre los que se va a implantar. Por ello, el proceso a llevar a cabo se puede describir como se indica a continuación:

13.3.1 Proceso de elaboración software

El programa de control se va a llevar a cabo en el lenguaje de programación C++. Se va a hacer uso de la librería OpenCV para la codificación los módulos y se va a desarrollar en el entorno de programación NetBeans, que facilita el uso de las librerías, la compilación y ejecución del programa.

13.3.2 Pruebas individuales

Estas pruebas son las primeras que se llevan a cabo. Se pueden realizar en diferentes equipos o con distintos dispositivos, y de esta forma observar opciones de escalabilidad. No es necesario, a priori, emplear en todo momento los equipos que se utilizan en la solución final, ya que su principal finalidad es comprobar si cada módulo lleva a cabo sus funciones correctamente. Pero, sí será necesaria al menos una prueba con los elementos finales y estas son las que se tendrán en cuenta para el proyecto.

13.3.3 Pruebas de integración

Estas pruebas se realizan tras la finalización de las pruebas individuales. Es necesario que se lleven a cabo sobre el equipo o dispositivo sobre el que se va a implementar la solución. Para llevar a cabo las pruebas en las que el VANT debe levantar el vuelo, será necesario hacerlas en un espacio cerrado y controlado.

13.3.4 Pruebas de validación

Las pruebas de validación se llevan a cabo al pasar de forma satisfactoria las pruebas de integración. La solución se considera válida si se cumplen todos los requisitos indicados en el apartado de especificaciones.

13.4 Normativa aplicable

La normativa que engloba el uso de VANTs se contempla en el *Real Decreto 1036/2017, de 15 de diciembre* [18]. En este se indican los requisitos que deben cumplir los equipos, los pilotos, el sistema de control del equipo y, en gran medida, sobre su uso.

13.5 Aspectos vinculantes

13.5.1 Cambios en el proyecto

El proyecto puede sufrir ciertos cambios respecto al diseño original bajo causa justificada por motivos como:

- Cambios en la normativa que exigen determinadas modificaciones
- Mejora de aspectos del proyecto (Rendimiento, métodos de detección, etc.)
- Imprevistos o efectos negativos que surjan

En caso de no ser justificado, no se efectuará el cambio.

13.5.2 Pruebas a realizar

Las pruebas que se deben llevar a cabo son las que se especifican en el plan de pruebas y bajo las condiciones que se indican en este. Es posible llevar a cabo pruebas adicionales, pero carecerán de validez para comprobar el funcionamiento o la validación en el proyecto.

13.6 Plan de pruebas

En este plan de pruebas se recogen las pruebas que es necesario superar para que la solución se considere válida. En él se describen pruebas de funcionamiento de los módulos, pruebas de integración y pruebas de validación.

- **Pruebas de funcionamiento:**
  - Funcionamiento del módulo de control de las cámaras
  - Funcionamiento del módulo de detección de gestos
  - Funcionamiento del módulo de interpretación
  - Funcionamiento del programa de control del drone
- **Pruebas de integración:**
  - Envío de señales
  - Funcionamiento del sistema integrado
- **Pruebas de validación:**
  - Características del sistema

En las tablas (Tabla 13, Tabla 14, Tabla 15, Tabla 16, Tabla 17, Tabla 18, Tabla 19) se encuentra la información de los resultados de cada una de las pruebas mencionadas.

Tabla 13: Prueba de funcionamiento: módulo de control de cámaras

Nombre de la prueba	**Funcionamiento del módulo de control de cámaras**	**Fecha de realización**	
Elementos necesarios	• Equipo de conexión de las cámaras y de ejcución del programa (Raspberry Pi 3) • Dos cámaras de conexión USB (web-cams de Logitech) • Programa en C++ de control de cámaras		
Aspectos a medir	La conexión del programa con las cámaras es correcto	SI	NO
	La lectura de las cámaras se lleva a cabo correctamente	SI	NO
	La lectura de las cámaras se realiza en el orden esperado	SI	NO
	El cierre de conexión con las cámaras se lleva a cabo de forma satisfactoria antes de finalizar el programa	SI	NO
Observaciones			

Tabla 14: Prueba de funcionamiento: módulo de detección de gestos

Nombre de la prueba	Funcionamiento del módulo de detección de gestos	Fecha de realización	
Elementos necesarios	• Equipo de ejecución (Raspberry Pi 3) • Colección de imágenes • Programa en C++ de detección de gestos		
Aspectos a medir	Se detecta correctamente si hay una persona en la imagen	SI	NO
	El filtrado a objeto más cercano se lleva a cabo de forma satisfactoria	SI	NO
	Se reconocen la posición de los brazos en caso de que se encuentre una persona en la imagen	SI	NO
Observaciones	La detección de personas debe ser de cuerpo entero. En caso de que se muestren las imágenes del proceso en pantalla, el tiempo de ejecución aumenta, pero no afecta al envío de señales.		

Tabla 15: Prueba de funcionamiento: módulo de interpretación

Nombre de la prueba	Funcionamiento del módulo de interpretación	Fecha de realización	
Elementos necesarios	• Equipo de ejecución (Raspberry Pi 3) • Programa en C++ de interpretación de gestos		
Aspectos a medir	A partir de la posición de los brazos se identifica correctamente el movimiento que se quiere realizar	SI	NO
	Conocido el movimiento deseado, se traduce a la información necesaria para el envío de señales	SI	NO
Observaciones			

Tabla 16: Prueba de funcionamiento: programa de control del drone

Nombre de la prueba	Funcionamiento del programa de control del drone	Fecha de realización	
Elementos necesarios	• Drone basado en Arduino • Programa en Arduino de control de los elementos del drone		
Aspectos a medir	Lectura correcta de los datos del giroscopio/acelerómetro	SI	NO
	Lectura correcta de las señales del receptor	SI	NO
	Inicio y control de los motores correcto	SI	NO
Observaciones			

Tabla 17: Prueba de integración: envío de señales

Nombre de la prueba	Envío de señales	Fecha de realización	
Elementos necesarios	• Equipo de ejecución (Raspberry Pi 3) • Programa en C++ de señalización • Placa Arduino UNO • Programa en Arduino para reconocimiento de las señales recibidas		
Aspectos a medir	Las señales se envían de forma satisfactoria	SI	NO
	La duración de los pulsos es adecuado en cada caso	SI	NO
Observaciones			

Tabla 18: Prueba de integración: funcionamiento del sistema completo

Nombre de la prueba	Funcionamiento del sistema integrado	Fecha de realización	
Elementos necesarios	• Raspberry Pi 3 • Dos web-cams Logitech • Programa con los módulos integrados • Placa Arduino UNO • Programa en Arduino para reconocimiento de las señales recibidas, similar al de manejo por radiocontrol		
Aspectos a medir	El sistema lleva a cabo su función correctamente	SI	NO
	La información que pasa de un módulo a otro es necesaria y suficiente para el funcionamiento del mismo	SI	NO
	Los módulos integrados llevan a cabo sus funciones correctamente	SI	NO
	El tiempo de ejecución del bucle principal es aceptable en términos de rendimiento	SI	NO
Observaciones			

Tabla 19: Prueba de validación: características del sistema

Nombre de la prueba	Características del sistema	Fecha de realización	
Elementos necesarios	• Drone basado en Arduino • Raspberry Pi 3 • Dos web-cams Logitech • Programa con los módulos integrados		
Aspectos a medir	Los motores del drone responden correctamente a los gestos llevados a cabo	SI	NO
	El tiempo de ejecución del programa es adecuado para el funcionamiento del drone	SI	NO
Observaciones			

14 Anexo II – Esquemas de conexión

14.1 Esquema de conexión de las cámaras

Teniendo en cuenta los elementos hardware necesarios, las conexiones se llevan a cabo como se indica en la siguiente figura.

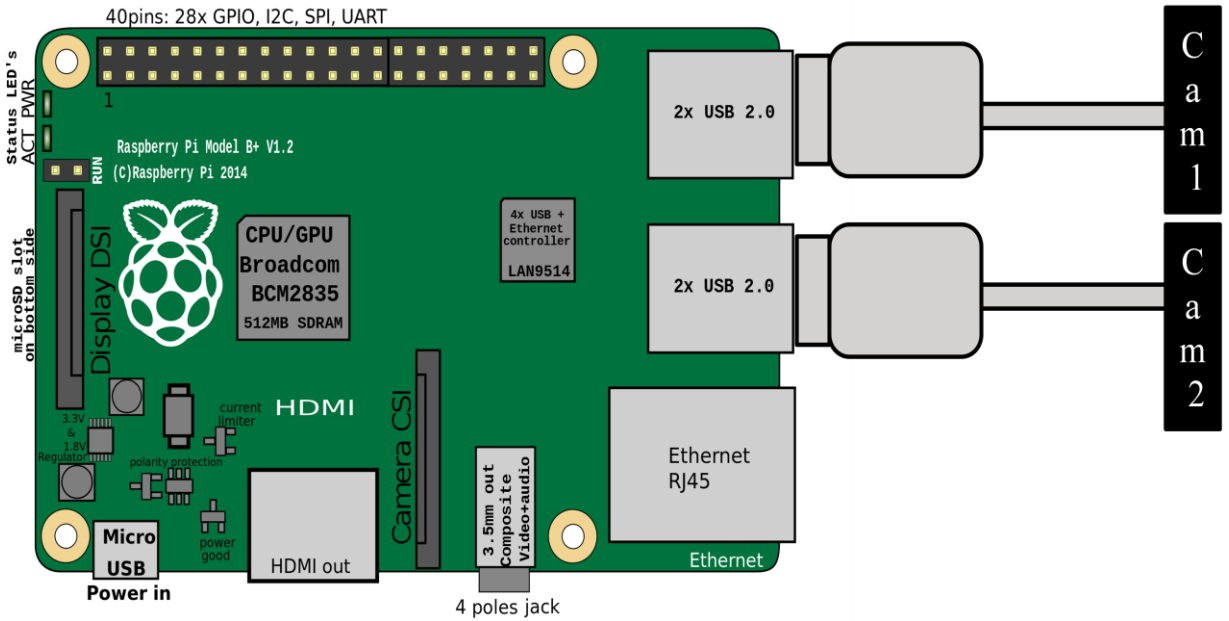


Figura 23: Diagrama de conexión de los elementos hardware

Nota: Teniendo en cuenta la conexión es USB, las cámaras se podrían conectar a cualquiera de los cuatro puertos de los que dispone la placa Raspberry Pi 3. Pero, la conexión debe ser ordenada, es decir, primero la cámara principal y luego la secundaria.

14.2 Esquema de conexión con la placa Arduino Uno

Esta conexión se debe llevar a cabo al conectar el módulo con el equipo de pruebas.

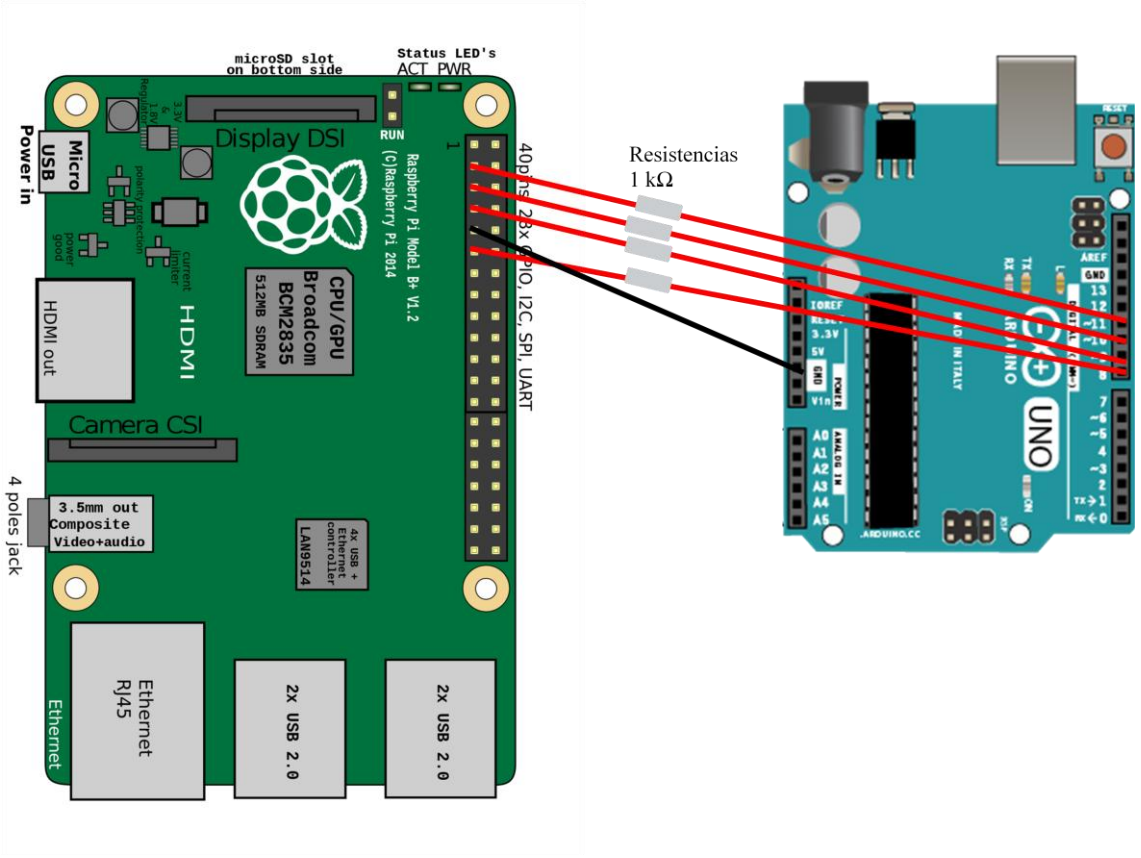


Figura 24: Diagrama de conexión de Raspberry Pi 3 con la placa Arduino Uno

Para la conexión con los pines de ambas placas es necesario añadir resistencias debido a la diferencia de tensión.

Nota: La conexión entre pines puede variar según programación. Se especificarán los pines utilizados.

15 Anexo III - Manual de usuario

Para el control del drone, el usuario debe conocer los gestos que debe llevar a cabo para controlarlo.

Antes de nada, el usuario debe encontrarse frente a la cámara. Con el equipo iniciado, se inicia también el proceso de reconocimiento de gestos. Los gestos que este reconoce son los que se indican en la Tabla 20.

Tabla 20: Gestos para el control de los movimientos del drone

Gesto del piloto	Movimiento del drone
Brazo derecho recto	Hacia la derecha
Brazo izquierdo recto	Hacia la izquierda
Brazos derecho e izquierdo rectos	Hacia adelante
Un brazo recto y el otro inclinado hacia abajo	Hacia atrás
Brazo derecho inclinado hacia arriba	Rotación en sentido antihorario
Brazo izquierdo inclinado hacia arriba	Rotación en sentido horario
Brazos derecho e izquierdo inclinados hacia arriba	Aumenta la altura
Brazos derecho e izquierdo inclinados hacia abajo	Disminuye la altura
Brazos abajo, en posición de reposo	Quieto en el aire

16 Anexo IV – Desglose de horas internas dedicadas

En este apartado es posible encontrar, de forma desglosada, las horas dedicadas a las tareas, así como el tiempo dedicado a cada ámbito; formación, investigación, diseño y desarrollo, pruebas y validación y gestión.

Tabla 21: Desglose de horas dedicadas por paquetes de trabajo y tareas

Código de tarea	Horas a dedicar
P.T.0	**10**
T.0.1	4
T.0.2	5
T.0.3	1
P.T.1	**15**
T.1.1	4
T.1.2	8
T.1.3	3
P.T.2	**20**
T.2.1	5
T.2.2	10
T.2.3	5
P.T.3	**90**
T.3.1	80
T.3.2	10
P.T.4	**90**
T.4.1	10
T.4.2	60
T.4.3	20
P.T.5	**10**
T.5.1	10
P.T.6	**68**
T.6.1	38
T.6.2	30
TOTAL	**303**

Como cabe esperar, la mayor parte del tiempo se dedica a los paquetes de trabajo relacionados con el desarrollo de la solución, P.T.3, y con las pruebas que se llevan a cabo, P.T.4. También se puede apreciar que al paquete de trabajo P.T.6, relacionado con la documentación del proyecto, se le dedica una gran cantidad de tiempo. Esto se debe a la importancia de esta labor y que esta se lleva a cabo durante todo el proyecto.

Por otra parte, puede resultar interesante conocer las horas dedicadas a cada ámbito, para observar posibles cadencias o ver si el presupuesto destinado a horas internas se reparte de la forma esperada. En la Tabla 22 y en la Figura 25 se encuentra la información sobre el tiempo de dedicación a los ámbitos: formación, investigación, diseño y desarrollo, pruebas y validación y gestión.

Tabla 22: Desglose de horas dedicadas por ámbito

	Formación	Investigación	Diseño y Desarrollo	Pruebas y Validación	Gestión
P.T.0	**10**	**1**	**0**	**0**	**0**
T.0.1	4	0	0	0	0
T.0.2	5	0	0	0	0
T.0.3	1	1	0	0	0
P.T.1	**0**	**15**	**0**	**0**	**0**
T.1.1	0	4	0	0	0
T.1.2	0	8	0	0	0
T.1.3	0	3	0	0	0
P.T.2	**0**	**0**	**20**	**0**	**0**
T.2.1	0	0	5	0	0
T.2.2	0	0	10	0	0
T.2.3	0	0	5	0	0
P.T.3	**0**	**0**	**90**	**0**	**0**
T.3.1	0	0	80	0	0
T.3.2	0	0	10	0	0
P.T.4	**0**	**0**	**0**	**90**	**0**
T.4.1	0	0	0	10	0
T.4.2	0	0	0	60	0
T.4.3	0	0	0	20	0
P.T.5	**0**	**1**	**0**	**10**	**0**
T.5.1	0	1	0	10	0
P.T.6	**0**	**0**	**0**	**0**	**68**
T.6.1	0	0	0	0	38
T.6.2	0	0	0	0	30
TOTAL	**10**	**17**	**110**	**100**	**68**

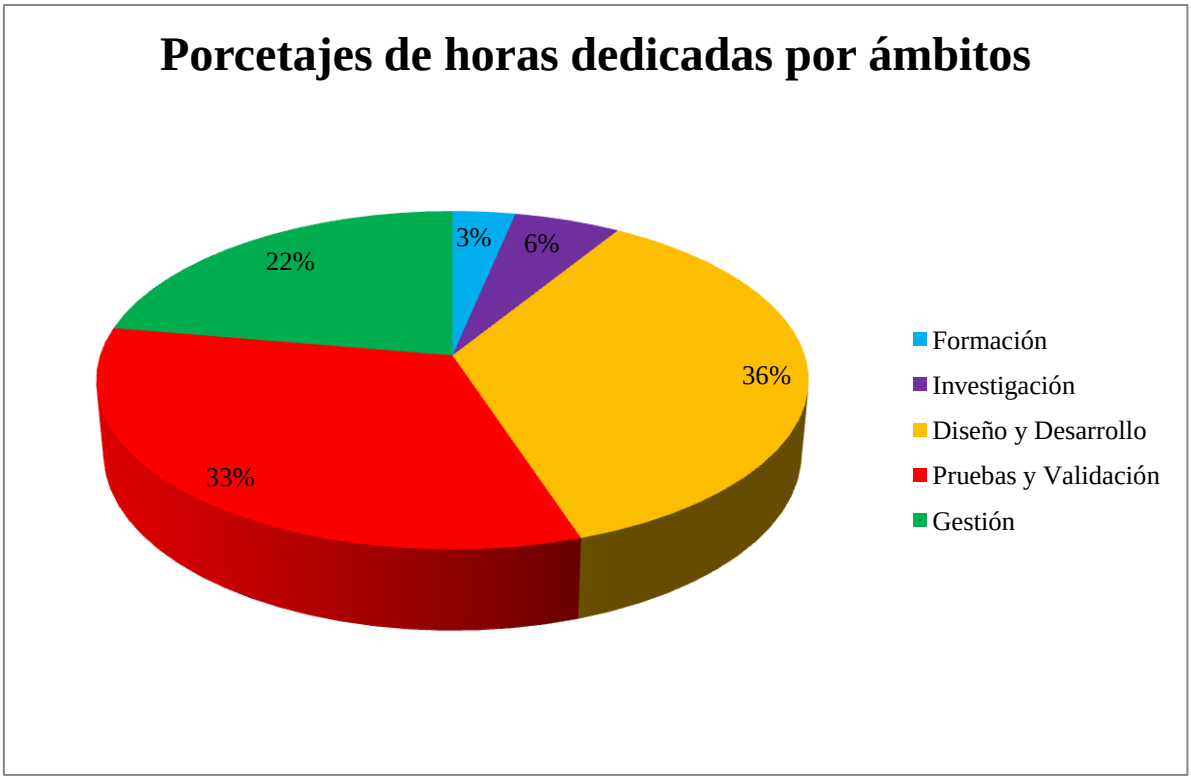


Figura 25: Gráfico circular de porcentajes de horas dedicadas por ámbito

Como se puede comprobar, la mayor parte del tiempo se emplea en la realización de la solución, ya que este se dedica a los ámbitos "diseño y desarrollo" y "pruebas y validación", 90 horas cada uno. El ámbito de gestión también conlleva cierto tiempo, 68 horas, por lo que su porcentaje también es considerable. Y por último, los ámbitos con menor tiempo dedicado son aquellos en los que no se lleva a cabo el desarrollo de la solución, pero resultan esenciales para el proyecto.

Con todo ello se puede concluir que esta distribución de tiempo resulta eficiente, y las horas dedicadas a cada tarea suficientes para su finalización.