UNIVERSITAT POLITÈCNICA DE VALÈNCIA

ESCUELA TÉCNICA SUPERIOR INGENIEROS INDUSTRIALES VALENCIA

TRABAJO FIN DE GRADO EN INGENIERÍA EN TECNOLOGÍAS INDUSTRIALES

DESARROLLO INTEGRAL DE UN QUADROTOR: DISEÑO E IMPLEMENTACIÓN DE LOS SISTEMAS DE TELEMETRÍA E INTERFAZ GRÁFICA (HMI) PARA LA MONITORIZACIÓN Y CONTROL MEDIANTE ENTORNO QT

AUTOR: PABLO ALBIOL GRAULLERA

TUTOR: PEDRO JOSÉ GARCÍA GIL

COTUTOR:

Curso Académico: 2013-14

AGRADECIMIENTOS

Antes de empezar me gustaría dedicar unas líneas de agradecimiento a todas las personas, amigos y familiares, que me han apoyado y motivado durante la realización de todo este proyecto.

En primer lugar, quiero darle las gracias a mis padres y a mis hermanos, que me han apoyado desde pequeño, poniendo gran empeño en hacer de mí la persona que hoy soy.

También quiero agradecer a Pedro, director del proyecto, por darme la oportunidad de realizar un trabajo tan interesante y por su ayuda a lo largo de estos seis meses.

Tampoco puedo olvidarme de mis compañeros de laboratorio Luis y Ricardo por la paciencia que han demostrado, sus consejos y la ayuda prestada durante todo este tiempo.

Por último, a mis compañeros de trabajo Dani y Alberto. Con ellos ha sido todo mucho más fácil y llevadero. Espero que los años que vienen sean igual de buenos.

Sin nada más que añadir, muchas gracias.

Pablo Albiol Graullera

Este Trabajo Final de Grado contiene, por orden, los siguientes documentos:

Memoria

Manual de usuario del HMI

Manual de programación del HMI en Qt

Presupuesto

UNIVERSITAT POLITÈCNICA DE VALÈNCIA

ESCUELA TÉCNICA SUPERIOR INGENIEROS INDUSTRIALES VALENCIA

TRABAJO FIN DE GRADO EN INGENIERÍA EN TECNOLOGÍAS INDUSTRIALES

MEMORIA

AUTOR: PABLO ALBIOL GRAULLERA

TUTOR: PEDRO JOSÉ GARCÍA GIL

COTUTOR:

Curso Académico: 2013-14

ÍNDICE

ÍNDICE DE ILUSTRACIONES

ÍNDICE DE TABLAS

1. OBJETIVOS Y ALCANCE DEL TRABAJO

El trabajo a continuación descrito ha consistido en el desarrollo integral de un vehículo quadrotor. El proyecto se ha dividido en dos partes; una primera de construcción de la plataforma y control de estabilidad del RPAS, y una segunda parte específica, que en el caso de este TFG, es el radioenlace del vehículo con un ordenador en tierra para la monitorización y control del mismo en tiempo real (Ilustración 1).

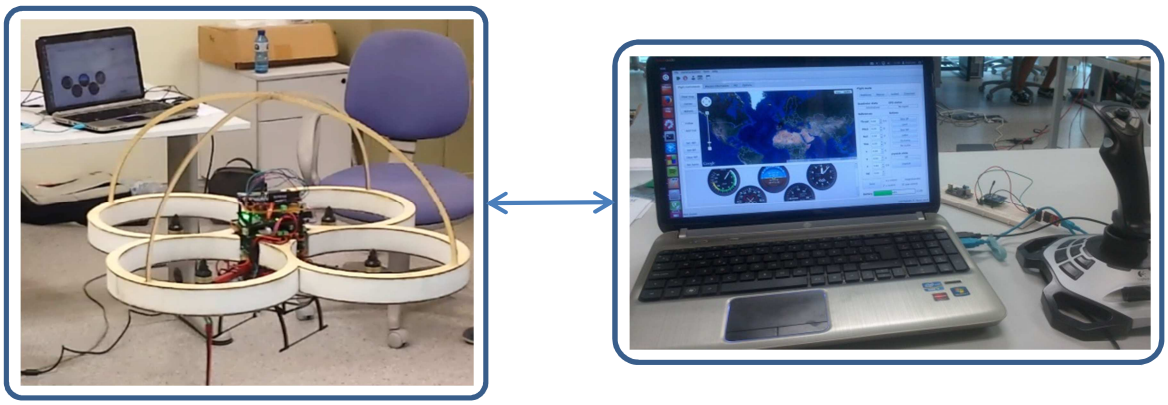

Ilustración 1. Enlace del quadrotor con unidad de control terrestre.

Como se puede ver en la Ilustración 2, este proyecto se divide en tres grandes bloques: implementación de los sistemas de telemetría en el ordenador de a bordo del quadrotor, programación de la unidad de control terrestre (interfaz gráfica HMI) e implementación de otra estrategia para controlar el vehículo, a través de una emisora y receptor de radiocontrol, que funcione como mecanismo de seguridad ante fallo del ordenador de a bordo del quadrotor o del sistema de telemetría principal.

Gran parte del proyecto ha sido llevado a cabo en grupo junto a los estudiantes Alberto Castillo Frasquet (Castillo Frasquet, 2014) y Daniel Verdú Torres (Verdú Torres, 2014). Se ha trabajado conjuntamente, especialmente durante la primera fase del trabajo, no obstante, el trabajo en equipo ha sido constante a lo largo de todo el semestre debido a la coordinación que requiere un proyecto de este tipo.

El objetivo principal del quadrotor y que ha marcado la parte individual y específica de cada Trabajo Final de Grado ha sido la perspectiva de realizar vuelos en exteriores. Para ello y una vez construida la plataforma de vuelo y validado el control de estabilidad se ha pasado a la segunda fase. El estudiante Daniel Verdú ha implementado un control de altura a partir de un sensor barométrico y un control de orientación con las medidas de un magnetómetro (Verdú Torres, 2014) y el estudiante Alberto Castillo ha trabajado en un control de posición a partir de medidas GPS (Castillo Frasquet, 2014). Por último, este proyecto, como se ha comentado, implementa el enlace de telemetría, a través de un módulo de radiofrecuencia, con un ordenador en tierra.

El trabajo realizado durante todo este tiempo ha sido en su mayor parte un trabajo multidisciplinar, práctico y aplicado. Por nombrar algunos ejemplos, se han tratado temas tan diversos como: utilizar y programar diferentes sistemas embebidos, comunicar sistemas

electrónicos, leer sensores, tratar señales, implementar algoritmos de control, utilizar sistemas operativos de tiempo real, realizar buenas conexiones, soldar de manera correcta, etc.

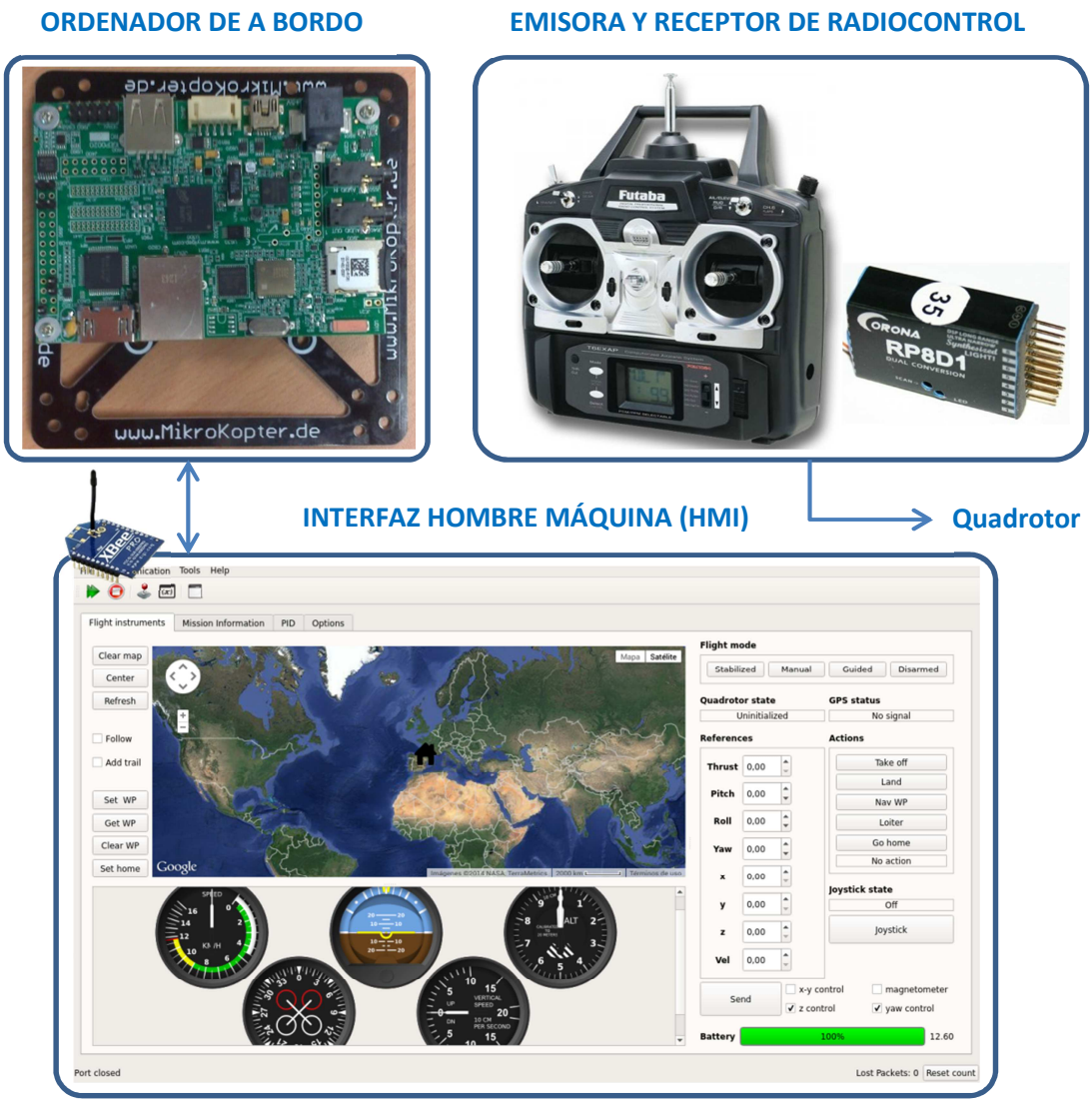

Ilustración 2. Ejes principales del proyecto.

Durante todo el proyecto ha sido necesario utilizar diversos conceptos teóricos asimilados a lo largo de los cuatro años de grado. Además se ha complementado todo esto con nuevos conceptos aprendidos que amplían la formación de un ingeniero industrial.

1.1. Objetivos

1.1.1. Plataforma de vuelo

- Desarrollar una plataforma de vuelo experimental. La plataforma ha de ser robusta y fiable a la vez que flexible y modular.
- Implementar el hardware y los sensores necesarios para realizar vuelos tanto en interiores como al aire libre.
- Desarrollar el software necesario para la gestión del vehículo y la comunicación con los distintos equipos electrónicos.

- Conseguir vuelo estable en actitud tanto en interiores como en exteriores.

1.1.2. Sistemas de telemetría

- Implementar un sistema de telemetría fiable entre el quadrotor y un ordenador en tierra. Además, el enlace ha de ser bidireccional, para monitorizar el "drone" en tiempo real y al mismo tiempo poder controlarlo.
- Desarrollar una interfaz gráfica de usuario para el control del quadrotor desde la unidad de control terrestre.
- Disponer de un mecanismo de seguridad para retomar el control del quadrotor en caso de fallo del ordenador principal.

1.1.3. Interfaz hombre máquina (HMI)

El software de la estación terrestre debe contar con las siguientes características:

- Interfaz gráfica de ususario para visualizar de forma rápida el estado del vehículo con indicadores en forma de instrumentos de vuelo y mapa para posicionar en todo momento el quadrotor.
- Opciones para actuar sobre el vehículo cambiando su modo de vuelo, controlándolo con un joystick o modificando los parámetros de los controladores.
- Facilidades para la ejecución del programa en diferentes sistemas operativos.
- Opciones para guardar datos de vuelo y reproducirlos posteriormente, exportar variables a MATLAB o incluso representarlas gráficamente en tiempo real.

1.2. Tareas Específicas a Realizar

1.2.1. Plataforma de vuelo

- Montaje de la estructura del quadrotor.
- Montaje del sistema de alimentación.
- Elección de los motores, controladores y hélices.
- Implementación de sensores, microcontrolador y ordenador.
- Comunicaciones entre los distintos equipos.
- Programación del software para el control del vehículo.
- Ajuste de la ley de control en actitud y altura.

1.2.2. Sistemas de telemetría

- Elección de los módulos de radiofrecuencia.
- Ensayo de alcance, fiabilidad e interferencias de los módulos RF con otros componentes.
- Implementación de los módulos de radiofrecuencia en el quadrotor.
- Elección de las variables a controlar.
- Programación del protocolo de comunicación entre el ordenador de a bordo del quadrotor y la estación terrestre.
- Implementación de un segundo sistema independiente del ordenador de a bordo para controlar el quadrotor en caso de fallo del ordenador principal.

1.2.3. Interfaz hombre máquina (HMI)

- Elección del lenguaje y entorno de programación.
- Programación del software en la estación en tierra.
- Realización de pruebas para verificar el correcto funcionamiento de todo el sistema.

1.3. Resumen de los Resultados Obtenidos

Tras todo el semestre de trabajo, los resultados que se han obtenido son los siguientes:

- Se ha desarrollado un quadrotor íntegramente.
- Se han implementado diferentes sensores en el vehículo: dos IMUs, un sensor de ultrasonidos, un sensor barométrico, un GPS, un módulo de radio y un receptor de radiocontrol.
- Se ha ajustado la ley de control de forma iterativa para optimizar la respuesta del quadrotor.
- Se ha conseguido realizar un vuelo estable en actitud y altura tanto en interiores como en exteriores.

En cuanto a la parte de la telemetría:

- Se ha conseguido establecer un enlace bidireccional y fiable entre el vehículo y un ordenador en tierra mediante un módulo de radiofrecuencia.
- Se ha desarrollado desde cero una interfaz gráfica para la monitorización y control del quadrotor.
- Se ha implementado un receptor de radiocontrol conectado directamente a un microcontrolador en el vehículo para que en caso de fallo del ordenador de a bordo, poder recuperar el control del vehículo mediante una emisora de radiocontrol.

1.4. Estructura de este Documento

A continuación se hace una descripción breve de los distintos apartados que contiene esta memoria.

En la introducción se realizará una breve descripción del problema. En primer lugar, se nombrarán los motivos que han llevado a la realización de este proyecto y a continuación se explicarán los antecedentes del proyecto (la información a partir de la cual se partía) y la justificación del mismo (dónde se quiere llegar y qué se pretende resolver con este trabajo).

El capítulo llamado “estado del arte”, trata de introducir al lector en los conceptos de quadrotor y telemetría. Además, se pretende explicar estos términos desde un enfoque basado en la actualidad.

Descripción de la plataforma, se describirá de forma general la plataforma de vuelo desarrollada. La descripción se centrará en la parte de hardware, software y estructura del sistema de control dejando la parte de obtención del modelo dinámico teórico para otros trabajos.

A partir de este apartado se describe la segunda parte del trabajo realizado (diseño e implementación de los sistemas de telemetría e interfaz gráfica (HMI) para la monitorización y control mediante entorno Qt).

El capítulo “ámbito de aplicación y rango de soluciones del trabajo” describe los condicionantes de diseño y las alternativas de diseño para el enlace entre el quadrotor y la UCT, la interfaz gráfica HMI y el enlace entre la plataforma y una emisora de radiocontrol.

A continuación, el apartado de descripción de las soluciones adoptadas explica la solución adoptada para cada uno de los tres puntos anteriores (enlace quadrotor UCT, HMI y enlace quadrotor emisora radiocontrol) haciendo énfasis en la justificación de las decisiones tomadas.

El capítulo “conclusiones y trabajos futuros” verifica el cumplimiento de los objetivos expuestos al inicio del trabajo. Al final, también expone una serie de trabajos futuros para mejorar la plataforma desarrollada.

Finalmente hay una sección de bibliografía con referencias a trabajos citados, algún libro, artículo de revista y páginas webs de importancia consultadas a lo largo de este proyecto.

2. INTRODUCCIÓN

Un vehículo aéreo remotamente pilotado, “Remotely Piloted Aircraft Systems” (RPAS) es un vehículo aéreo capaz de realizar misiones en modo semiautónomo. Los quadrotors son una clase de RPAS que han captado la atención de investigadores durante los últimos años debido a su maniobrabilidad y desempeño tanto en interiores como en exteriores.

2.1. Motivación

El campo de los UAS/RPAS es un sector con mucho auge y futuro, prueba de ello es que la Unión Europea considera a los RPAS como un sector estratégico en su programa “HORIZONTE 2020” (HORIZONTE2020). No obstante, estos vehículos ya se están usando en la actualidad para múltiples aplicaciones civiles.

Entre las aplicaciones más utilizadas destacan: agricultura, minería, inspección de infraestructuras, fotografía aérea, cartografía, rescate y salvamento, seguridad, aplicaciones militares, meteorología, investigación de flora y fauna, etc.

En la Ilustración 3, se muestra un ejemplo de una aplicación concreta de un quadrotor, la inspección de una línea eléctrica.

Ilustración 3. Inspección de una línea eléctrica.

Las múltiples aplicaciones que presentan estos vehículos y el hecho de que este sector se encuentra en continua evolución ya suponen motivación suficiente para embarcase en un proyecto de este tipo.

2.2. Antecedentes y Justificación del Trabajo

El proyecto nace en septiembre del pasado año tras la publicación de una primera lista de ofertas de trabajos finales de grados para el año siguiente. Este trabajo, llamó la atención y despertó interés. A raíz de esto, se contactó con Pedro García Gil, el director del proyecto.

El grupo de investigación de Pedro García Gil había trabajado en los últimos años en algoritmos para el control de quadrotors (Ródenas Lorda, 2013). Sin embargo, la mayoría de estos helicópteros sólo habían volado en interiores.

Pronto se vio que el área donde más se podía avanzar y menos trabajo había hecho era en la parte de vuelos en exteriores, pues las plataformas desarrolladas hasta el momento no contaban con todos los sensores necesarios para volar al aire libre. Así pues, éste se convirtió en el objetivo central del trabajo. Llegados a este punto, también era necesario construir una plataforma de vuelo desde cero para albergar los nuevos sistemas.

Se decidió que para poder volar en exteriores era indispensable implementar un sensor de altura preciso, pues hasta el momento se había utilizado un sensor de ultrasonidos con un alcance de unos 3 metros. Además también era necesario contar con un magnetómetro para conocer en todo momento la orientación del vehículo y guiar el mismo. Por último, también era de mucha utilidad disponer de un sensor GPS para conocer la posición del quadrotor e incluso realizar un control de posición a partir de sus medidas.

Por otra parte, las versiones anteriores de la plataforma de vuelo, se comunicaban con un ordenador a través de un enlace unidireccional por medio de WiFi. Este sistema cumplía su función adecuadamente en interiores donde el vehículo no se alejaba de la estación de tierra más de 10 metros, sin embargo, con objeto de tener mayor alcance en exteriores, era imprescindible implementar un nuevo enlace de telemetría. El objetivo era tener un lazo de comunicación bidireccional que sirviera por un lado para actuar sobre el quadrotor modificando el modo de vuelo por ejemplo, y por otro lado monitorizar la información más importante del vehículo en tiempo real.

En cuanto a la interfaz gráfica de usuario (HMI) para controlar el vehículo desde un ordenador en tierra, el grupo de Pedro García contaba con un software que había desarrollado unos años atrás. Sin embargo, como se ha explicado anteriormente, el enlace era unidireccional y por WiFi. Además la interfaz era bastante básica, por ello y más razones se decidió desarrollar una nueva desde cero.

3. ESTADO DEL ARTE

3.1. Concepto de Quadrotor

Se denomina quadrotor, cuadrirrotor o cuadricóptero a un vehículo aéreo no tripulado que está sustentado y propulsado por cuatro rotores para volar a un punto fijo y de forma controlada. Para ello, el control del movimiento del vehículo se consigue variando la velocidad relativa de cada motor. En la Ilustración 4 se muestra uno de los últimos modelos de quadrotors comerciales de la empresa dji.

Ilustración 4. Quadrotor PHANTOM 2 VISION de la empresa dji.

A diferencia de muchos helicópteros, los quadrotors usan dos pares de hélices fijas idénticas, de modo que cada par opuesto de hélices gira en el mismo sentido. Esto proporciona diversas ventajas en cuanto al control y la estabilidad.

No obstante, un quadrotor es un caso particular de un multirotor, que es el término que se aplica para referirse a cualquier configuración y número de hélices. Como se ha comentado en el capítulo anterior, el sector de los multirotor se encuentra en auge, de modo que es posible ver nuevos prototipos y nuevas configuraciones a diario.

En la Ilustración 5, se puede ver el modelo UFOCAM XXL8, un multirotor de 8 rotores de la empresa alemana mikrokopter.

Ilustración 5. Multirotor UFOCAM XXL8.

3.1.1. Estado legal

Los sistemas de aviones dirigidos por control remoto/ sistemas de aeronaves no tripuladas (RPAS/UAS) suponen una revolución del mercado de servicios y un sector estratégico de primer orden. Consciente de ello, la Unión Europa se prepara para el pleno desarrollo de este sector.

En la EU a nivel legal hay dos grandes grupos de RPAS cada uno de los cuales está regulado por diferentes autoridades:

Por una parte están los RPAS con un peso superior a 150 kg los cuales se rigen por la normativa de la "European Aviation Safety Agency" (EASA). Y por otra parte están los de peso inferior a 150 kg los cuales están regulados por la autoridades de aviación civil de cada Estado Miembro.

En España, su uso para fines comerciales se encuentra explícitamente prohibido por el Ministerio de Fomento. De esta manera resulta evidente que hace falta desarrollar tan pronto como sea posible un exhaustivo encaje y armonización de todas las normas que permita un auténtico mercado único de los servicios aéreos.

3.2. Concepto de Telemetría

En todo sistema a controlar, y más en el caso de un vehículo aéreo remotamente pilotado, es necesario disponer de sistemas para monitorizar el estado del vehículo. La telemetría es una técnica altamente automatizada de comunicaciones mediante la cual se efectúa la medición de magnitudes físicas en puntos remotos o inaccesibles y se transmiten a un equipo receptor para el control. La palabra telemetría procede de las palabras griegas "tele", que significa distancia y "metron" que quiere decir medida.

Aunque el término telemetría se aplica comúnmente a las comunicaciones inalámbricas, también se puede utilizar en otros medios (teléfono, redes de ordenadores, enlaces de fibra óptica, etc.).

Las aplicaciones de los sistemas de telemetría son muy diversas. Se utiliza en grandes sistemas como naves espaciales, redes de suministro de gas, agua o electricidad, plantas químicas, medio ambiente… Además, también se utiliza en sectores tan diferentes como la pesca, medicina, agricultura, investigación o incluso para monitorizar el uso eficiente de energía en oficinas, fábricas o residencias.

3.2.1. Telemetría en RPAS

Uno de los subsistemas más importantes de cualquier vehículo aéreo remotamente pilotado es la telemetría, es decir, la comunicación entre el vehículo y una estación terrestre para la correcta monitorización y control del mismo.

Comúnmente, este subsistema de telemetría va unido a una interfaz gráfica de usuario en la unidad de control terrestre y tiene como objetivo disponer de un enlace en tiempo real fiable. Para ello, por lo general, se utilizan módulos de radiofrecuencia con distintas características según la aplicación deseada. Normalmente, estos dispositivos tienen poco ancho de banda (alrededor de 3kb/s) pero pueden transmitir datos muestreados por distintos sensores a alta frecuencia.

Sin embargo, existen muchos otros sistemas diferentes. Por ejemplo, una aplicación muy interesante de estos vehículos es la posibilidad de transmitir video en tiempo real, para ello se pueden utilizar transmisores de radiofrecuencia analógicos.

En la Ilustración 6, se muestra el sistema "on screen display" (OSD) muy utilizado para vuelos en "first person view" (FPV) de la empresa dmd.

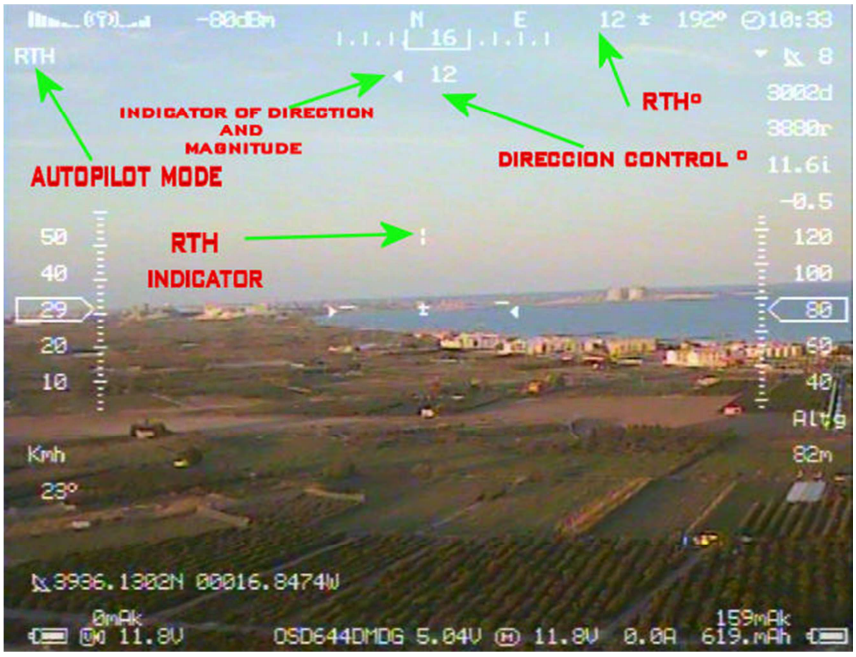

Ilustración 6. Sistema OSD.

En la Ilustración 7, se pueden ver unas "videogafas" que pueden resultar muy útiles junto al sistema OSD comentado en el párrafo anterior para vuelos en FPV.

Ilustración 7. Videogafas para vuelos en FPV.

4. DESCRIPCIÓN DE LA PLATAFORMA

A continuación se describe la plataforma de vuelo desarrollada. En primer lugar se describirá la plataforma en términos de hardware, a continuación se explicará brevemente el software desarrollado y finalmente se explicará la estructura del sistema de control implementado en el vehículo.

A modo de resumen, el esquema de funcionamiento global de la plataforma se muestra en la Ilustración 8.

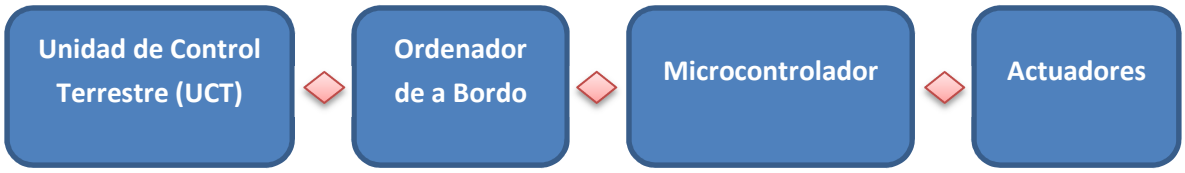

Ilustración 8. Funcionamiento global de la plataforma.

Antes de empezar, cabe resaltar que en ningún momento el objetivo principal de este proyecto es desarrollar una plataforma como producto final. Esto es, el objetivo primordial de la plataforma es disponer de la modularidad y flexibilidad necesaria para poder probar todo tipo de sensores, equipos electrónicos, motores, hélices, etc.

También cabe remarcar, que gran parte del material utilizado en la construcción de la plataforma ha sido reutilizado de proyectos anteriores, siendo el dinero a gastar una restricción importante, es por ello que seguramente, ni los materiales utilizados, ni los equipos electrónicos son los óptimos.

4.1. Hardware de la Plataforma

4.1.1. Estructura del Quadrotor

Para el chasis se ha utilizado un kit que vende la empresa alemana Mikrokopter. Esta estructura está fabricada mayormente en aluminio y tiene una distancia entre los motores de 41 cm.

Este chasis se vende junto a unas placas de fibra de vidrio que facilitan el ensamblaje de toda la estructura mediante tornillos y tuercas. Además, también facilita la instalación posterior de toda la electrónica en niveles sucesivos.

Respecto al tren de aterrizaje, también se ha comprado a la empresa Mikrokopter. Se trata de un tren de aterrizaje de plástico, con espacio suficiente para poder instalar un sensor de ultrasonidos y albergar una batería.

El chasis final del quadrotor, con el tren de aterrizaje ya instalado es el que se muestra en la Ilustración 9.

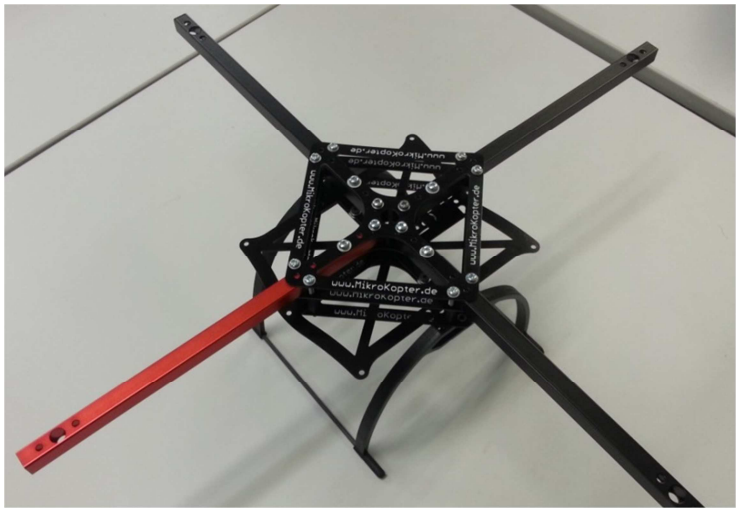

Ilustración 9. Chasis del quadrotor.

4.1.1.1. Protección

Paralelamente se ha diseñado una protección solidaria al chasis del quadrotor para hacer más seguro el vuelo en interiores, tanto para proteger todos los equipos electrónicos embebidos en el quadrotor como para proteger a las personas y objetos de los alrededores.

Además, esta protección permite sujetar con la mano el quadrotor aunque los motores estén en funcionamiento, lo cual resulta muy útil a la hora de ajustar el algoritmo de control y sus constantes experimentalmente.

Se ha realizado una estructura tipo sándwich con madera de chopo y poliestireno expandido. La madera se ha cortado mediante una cortadora láser a partir de un diseño previamente realizado con un programa de CAD. Posteriormente, se ha utilizado el contorno de la madera cortada para moldear el poliestireno con un arco de corte de hilo caliente. Finalmente, se ha pegado todo con cola y unido al chasis del quadrotor mediante bridas.

El resultado final se muestra en la Ilustración 10.

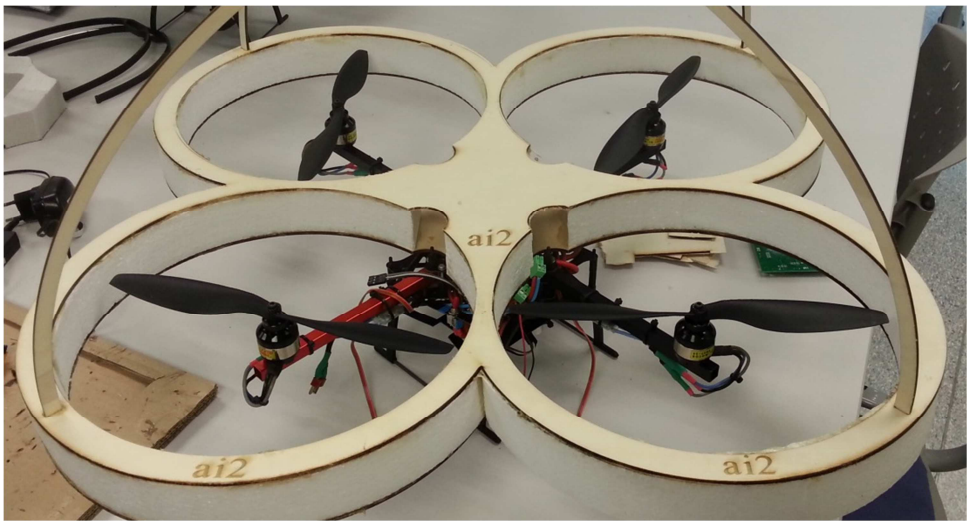

Ilustración 10. Protección del quadrotor.

4.1.2. Sistema de alimentación

La alimentación del quadrotor se divide por un lado en la alimentación de los motores y por otro en el resto de la electrónica. En la Ilustración 11 se muestra el montaje realizado, la alimentación principal se divide en seis cables, cuatro de ellos para alimentar los motores y dos para el resto de la electrónica (uno para el microcontrolador y los equipos electrónicos conectados a él y otro para el ordenador de a bordo del vehículo).

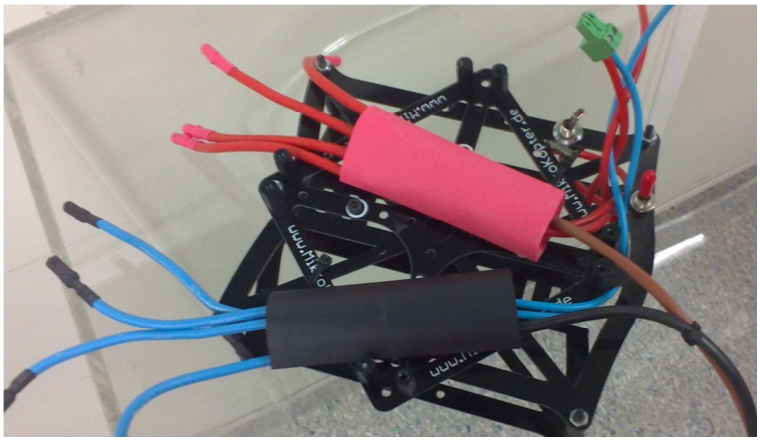

Ilustración 11. Sistema de alimentación de la plataforma.

Además se han añadido dos interruptores, los cuales se pueden ver en detalle en la Ilustración 12. Un interruptor se utiliza para el mini ordenador de a bordo y el otro para el resto de la electrónica, mientras que el microcontrolador es el que actúa como interruptor de los motores.

Por otro lado, los cables se han dimensionado para soportar las intensidades máximas que puede llegar a consumir el vehículo y así evitar que se calienten demasiado pudiendo llegar a fundirse y provocar cortocircuitos. Los cables que alimentan toda la electrónica, así como los cables de comunicaciones, no resultan críticos porque en condiciones normales de funcionamiento consumen en torno a 1 amperio. Sin embargo, los cables que alimentan los motores alrededor de 6 amperios en condiciones normales y pueden llegar a consumir hasta un máximo de 10 amperios cada uno.

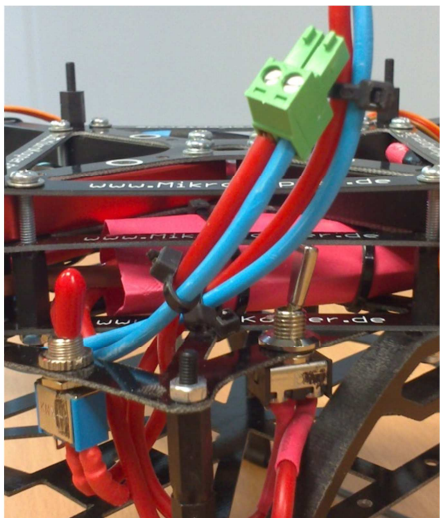

Ilustración 12. Detalle de los interruptores de la plataforma.

Para dimensionarlos, se ha hecho uso de la normativa estándar “American Wire Gauge” (AWG), de uso común para dimensionar cables según su capacidad de soportar corriente. Cabe mencionar que no se han tenido en cuenta factores como la temperatura ambiente o el aislante utilizado pues esta normativa se encuentra del lado de la seguridad.

El cable más crítico, es el que va desde el conector de la batería hasta la bifurcación de los cuatro motores y el resto de la electrónica. Este cable puede llegar a soportar en torno a 40 amperios, es por ello que se ha seleccionado un cable AWG-5 que soporta 47 amperios de forma continua y tiene un diámetro de cobre de 4,6 milímetros. Por último, para cada uno de los cuatro cables que van desde la bifurcación comentada hasta cada motor se ha utilizado un cable AWG-11 que tiene un diámetro de 2,3 milímetros y soporta una corriente de 12 amperios.

Asimismo, se han utilizado conectores tipo bala entre la bifurcación principal y los drivers, y entre los drivers y motores. Esto permite reducir el número de soldaduras al mínimo, al mismo tiempo que facilita la sustitución de motores y controladores. Todo ello se ha aislado con fundas termoretráctiles para reducir el riesgo de cortocircuitos.

4.1.2.1. Alimentación

El quadrotor se alimenta a partir de una batería LiPo de 3 celdas, aunque también se podrían utilizar 4 celdas ya que los motores y controladores soportan ambos tipos de baterías. No obstante, para todas las pruebas realizadas en el laboratorio se utiliza una fuente de alimentación regulable.

Las baterías LiPo son las más utilizadas en quadrotors, pues ofrecen una alta relación carga/peso, alta tasa de descarga y son económicas. Sus desventajas son que quedan casi inutilizables si se descargan por debajo de su voltaje mínimo y que debe tenerse mucho cuidado al cargarlas, ya que no pueden superar las especificaciones de corriente de carga. Además, se debe intentar igualar la carga en las tres celdas para aumentar el rendimiento de la batería y su vida útil. Para ello se utilizan cargadores comerciales especializados.

En la Tabla 1 se muestran las características técnicas de una de las baterías utilizadas en este proyecto.

Capacidad	5300 mAh
Voltaje nominal	11,1 V
Tamaño	139 x 43 x 32 mm
Peso	406 g
Descarga	30 C
Descarga máxima	60 C
Ratio de carga	5 C

Tabla 1. Características técnicas GENS ACE 5300mah 3S1P.

A la hora de elegir una batería, es muy importante tener en cuenta la capacidad y el ratio de descarga de la misma. En este caso, el ratio de descarga de la batería utilizada es de 30 C, lo que

supone 159 amperios continuos, valor muy superior a los 40 amperios que podría llegar a consumir el quadrotor.

Otro factor transcendental, y más en el caso de los quadrotors, es la autonomía. A mayor capacidad y menor velocidad de descarga, se consigue mayor duración de la batería. En este caso, para un consumo típico de 4 C (21,2 amperios), la duración se encuentra en torno a los 15 minutos. Sin embargo, para valores de descargas mayores a 2 C esta relación no es del todo lineal, por lo tanto la duración de la batería se ve aún más reducida.

4.1.3. Actuadores

4.1.3.1. Motores

Los motores utilizados son de la marca alemana Robbe Roxxy. Son motores "OutRunner", muy utilizados para aeromodelismo, sin escobillas, de 14 polos, capaces de proporcionar una potencia máxima de 110 W y combinados con la hélice adecuada pueden levantar hasta 820 gramos de masa. En la Ilustración 13 se muestra este motor.

Ilustración 13. Robbe Roxxy 2827-35.

A continuación, se especifican las características técnicas de este motor (Tabla 2).

Batería	LIPO (2-4 celdas)
Corriente nominal	4-9 A
Corriente máxima (60 s)	10 A
KV	760 rpm/V
Potencia	110 W
Peso	57 g
Diámetro eje	3,17 mm

Tabla 2. Características técnicas Robbe Roxxy 2827-35

4.1.3.2. Variadores de velocidad

Los variadores de velocidad, controladores, drivers o "Electronic Speed Controllers" (ESC) son los dispositivos que se encargan de alimentar los motores y controlar su velocidad de giro. Para variar la velocidad de giro de un motor, se le envía un valor de referencia al driver normalmente por modulación PWM. En función de este valor y a través de un circuito electrónico, el variador genera una señal trifásica con una frecuencia proporcional a la frecuencia de giro del motor.

Los controladores utilizados en este proyecto, los cuales se muestran en la Ilustración 14, han sido los YGE30i de la empresa alemana IGE.

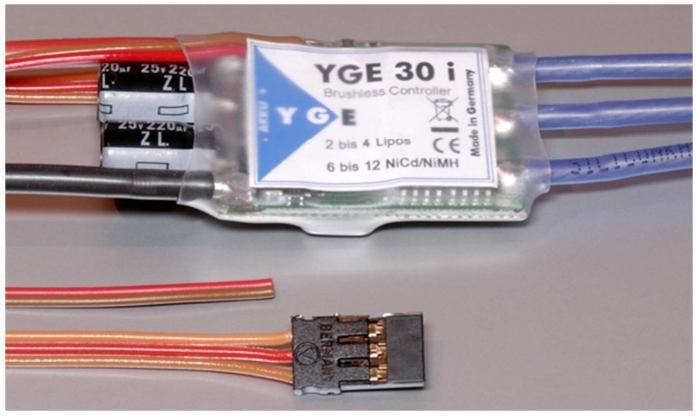

Ilustración 14. Drivers YGE30i utilizados.

Entre las características más importantes de este producto destaca el hecho de que el valor de referencia enviado al ESC puede ser tanto por modulación PWM como mediante el protocolo I2C. Además pueden soportar corrientes de hasta 30 amperios y cuentan con detección de baja tensión, protecciones frente a sobrecorrientes, temperaturas altas, etc.

Normalmente, estos dispositivos están pensados para ser usados directamente conectados a un receptor de radiocontrol, sin embargo, en este trabajo se utilizan conectados a cuatro salidas analógicas de un Arduino. Esto es, los valores de referencia los envía el Arduino. De este modo se consigue que el microcontrolador sea el que gestione la velocidad de cada motor para estabilizar el vehículo.

4.1.3.3. Hélices

El modelo EPP1045 es el que se ha utilizado para las hélices. Se trata de hélices de plástico, bastante flexibles, con un diámetro de 10 pulgadas y un paso de 4,5 pulgadas. En la Ilustración 15 se puede ver el aspecto de estas hélices.

El motivo que ha llevado a esta elección es que son las hélices que recomienda el fabricante del motor Robbe Roxxy 2827-35. Además son económicas y menos peligrosas que otras hélices más rígidas. La desventaja es que generan menos fuerza de sustentación que otras hélices de otros materiales e introducen más vibraciones en la estructura.

Cabe añadir, que para anular el momento generado en el ángulo de yaw por cada hélice, es necesario disponer de dos hélices en sentido horario y otras dos en sentido anti horario.

Ilustración 15. Hélices EPP1045.

4.1.4. Unidades de Control

Con respecto al sistema empotrado, se ha desarrollado una arquitectura en dos capas. El objetivo es que los algoritmos de control más críticos (control de estabilidad) se ejecuten sobre un microcontrolador (Arduino Due) y que las tareas de mayor carga computacional y de comunicación con la unidad de control terrestre (UCT) se lleven a cabo en un ordenador a bordo (Igep v2).

Además, esta arquitectura proporciona una gran flexibilidad para trabajar con distintos sensores, comunicar equipos electrónicos o probar algoritmos de control más avanzados que pueden implementarse tanto en el microcontrolador como en el mini-PC.

Por una parte, el Arduino Due resulta idóneo para comunicarse con todo tipo de sensores, leds, motores, etc. Además, al ser un microcontrolador sin sistema operativo, resulta más fiable que un ordenador a la hora de cumplir tiempos de ejecución estables.

Por otra parte, la Igep cuenta con un sistema operativo y un procesador mucho más potente. Esto facilita implementar algoritmos de control más complejos, guardar datos de vuelo, video, etc. Asimismo cuenta con periféricos adicionales (lector para SD, módulo WiFi, Bluetooth…) que pueden resultar muy útiles.

4.1.4.1. Arduino Due

Arduino es una plataforma de electrónica abierta para la creación de prototipos basada en software y hardware flexibles y fáciles de usar. El Arduino Due es un microcontrolador basado en la CPU Atmel SAM3X8E ARM Cortex-M3.

En la Tabla 3 se muestran sus características más importantes.

Microcontrolador	AT91SAM3X8E
Voltaje de operación	3,3 V
Voltaje de alimentación	6–16 V
Pines digitales	54 (12 PWM)
Pines analógicos de entrada	12
Pines analógicos de salida	2 (DAC)
Memoria flash	512 KB
SRAM	96 KB
Frecuencia del reloj	84 MHz

Tabla 3. Características técnicas Arduino Due.

Además cuenta con cuatro puertos serie TTL, dos puertos I2C, un puerto SPI, 2 puertos USB (uno de ellos OTG) y un puerto de bus CAN.

La siguiente figura, Ilustración 16, muestra el aspecto de un Arduino Due.

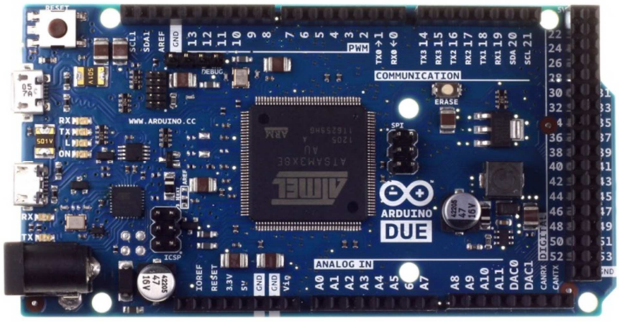

Ilustración 16. Arduino Due.

Como ya se ha comentado, en la versión actual de la plataforma, el Arduino Due se encarga principalmente del control a bajo nivel del quadrotor. Para ello, se hace uso de las siguientes funciones:

- Puerto SerialUSB para comunicarse con un PC
- Puerto Serial1 para la comunicación con la Igep.
- Puerto Serial3 para comunicarse con un Arduino Micro.
- Bus I2C para leer una IMU y pin digital para activar su procesador interno.
- 4 salidas PWM para actuar sobre los motores.
- 3 pines digitales para el uso de un led RGB.
- 1 pin digital para leer un sensor ping de ultrasonidos.
- 1 pin analógico para leer el voltaje de la batería.

Todo ello se ha montado sobre una PCB que había desarrollado el grupo de investigación de Pedro García Gil con anterioridad. Esto facilita mucho el trabajo ya que se reduce el cableado, se reduce el número de soldaduras y se evita el riesgo de cortocircuitos.

La PCB, la cual se muestra en la Ilustración 17, tiene disponibles los conectores necesarios para poder conectar todo tipo de sensores, además cuenta con un regulador de tensión que sirve para alimentar la Igep y dispone de los microchips necesarios para convertir niveles de tensión entre varios voltajes.

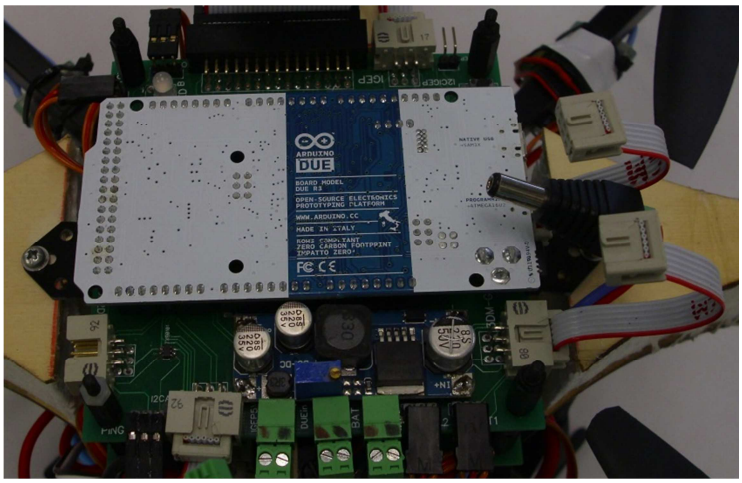

Ilustración 17. PCB utilizada para conectar el Arduino Due.

4.1.4.2. Igep v2

La Igep v2 es un mini ordenador de bajo consumo, placa única y sin ventilador basado en un procesador ARM diseñado y fabricado por la empresa española ISEE. Sus características técnicas se muestran en la Tabla 4.

Procesador	CPU: Cortex A8 at 1GHz
DSP	TMS320DM C64+ 800Mhz
Tarjeta gráfica	SGX530 @200MHz
Voltaje de alimentación	4,2-5,2 V
Memoria RAM	512 MB
Memoria Flash	512 MB
Tamaño	65mm x 95mm

Tabla 4. Características técnicas Igep V2.

Además la Igep cuenta con los siguientes periféricos: tarjeta WiFi integrada con protocolo 802.11 b/g, puerto Ethernet 10/100 MB, bluetooth, conector HDMI, salida de audio, ISP (cámara), 2 USB, lector microSD, 3 puertos serie TTL, bus I2C, SPI, RS232, RS485…

El aspecto de la Igep es el que se muestra en la Ilustración 18.

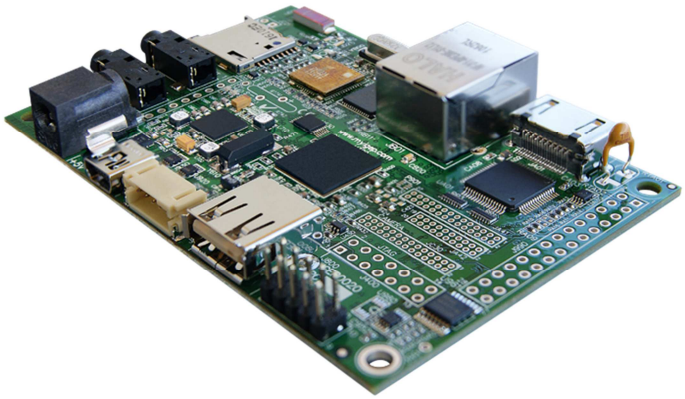

Ilustración 18. Igep V2.

La principal ventaja respecto al Arduino Due es que es posible instalar un sistema operativo y que el procesador que incorpora es mucho más potente. Sin embargo tiene menor número de puertos disponibles para comunicarse con otros sensores y menor flexibilidad en lo que se refiere a alimentar directamente otros equipos y niveles de voltaje para comunicarse con ellos.

En la plataforma actual la Igep se alimenta a partir de un regulador de tensión instalado en la PCB del Arduino Due y se comunica con él a través de un conversor de niveles lógicos de tensión ya que el Arduino Due funciona a 3,3 V y la Igep a 1,8 V.

A nivel de hardware de la Igep, en la plataforma se utilizan las siguientes funciones:

- Puerto serie para la comunicación con el Arduino Due.
- Puerto serie RS232 para comunicarse con un PC.

- Puerto serie para conectar un módulo de radiofrecuencia.
- WiFi para conectarse a la Igep desde un ordenador por medio de protocolo SSH.
- Bus I2C para leer una segunda IMU.

4.1.4.3. Arduino Micro

El Arduino Micro es un microcontrolador basado en el ATmega32u4. Sus características más importantes se muestran en la Tabla 5.

Microcontrolador	ATmega32u4
Voltaje de operación	5 V
Voltaje de alimentación	6–16 V
Pines digitales	20
Pines analógicos de entrada	12
Salidas PWM	7
Memoria flash	32 KB
EEPROM	1 KB
SRAM	2,5 KB
Frecuencia del reloj	16 MHz

Tabla 5. Características técnicas Arduino Micro.

Además cuenta con un puerto serie, un bus I2C y otro puerto SPI. En la Ilustración 19 se muestra su aspecto. Como se puede ver, este microcontrolador, es claramente más limitado que el Arduino Due.

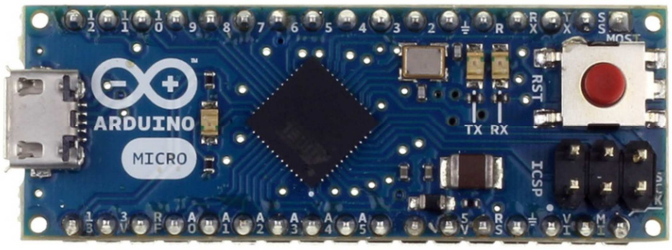

Ilustración 19. Arduino Micro.

El Arduino Micro está montado en una PCB de prototipado en un nivel superior a la PCB del Arduino y la Igep. A nivel de hardware se utiliza:

- Puerto serie para la comunicación con un sensor GPS.
- 2 pines digitales para comunicarse con el Arduino Due mediante la librería “Software Serial”.
- 6 pines digitales para leer señales PWM procedentes de un receptor de radiocontrol.

El motivo de incluir un segundo microcontrolador es para liberar al Arduino Due de más carga computacional con objeto de establecer su tiempo de ciclo en no más de 4 ms. La solución no es óptima, sin embargo el Arduino Micro tiene un peso y un tamaño muy reducido. No obstante, en el apartado de conclusiones y trabajos futuros se comentan soluciones alternativas.

4.1.5. Sensores

Los sensores son unos dispositivos electrónicos imprescindibles en este tipo de proyecto ya que a partir de ellos se obtiene la información necesaria para realizar el control del vehículo. Además, también resultan útiles para monitorizar las variables más importantes en tiempo real.

En este quadrotor se utilizan dos IMUs, un sensor de ultrasonidos, un sensor barométrico, un magnetómetro y un sensor GPS. Todos los sensores tienen sus puntos fuertes y débiles, la conclusión es que la mejor solución es la integración sensorial de todos los dispositivos.

4.1.5.1. Unidad de medida inercial (IMU)

Una unidad de medición inercial o IMU ("inertial measurement unit"), es un dispositivo electrónico que mide e informa acerca de la velocidad, orientación y fuerzas gravitacionales de un aparato, usando una combinación de acelerómetros y giróscopos. La IMU es el componente principal de los sistemas de navegación inercial usados en aviones, naves espaciales, buques y misiles guiados entre otros.

Se ha utilizado el chip MPU6050 diseñado por la empresa Invensense, que integra un giróscopo de tres ejes, un acelerómetro de tres ejes y un procesador interno. En la Ilustración 20 se muestra su aspecto.

Ilustración 20. Chip MPU6050.

La principal ventaja de este chip es que cuenta con un DMP ("digital motion processor") que se encarga de procesar todas las medidas y calcular de forma independiente los ángulos de orientación, sin consumir capacidad de cálculo del dispositivo externo. Además, el DMP proporciona una medida bastante precisa a pesar de las vibraciones en la estructura del quadrotor.

En este proyecto, se ha instalado el chip PU6050 que incorpora en una placa muy pequeña la IMU (MPU6050), un magnetómetro y un sensor barométrico. Los tres sensores se encuentran unidos al mismo bus I2C. En la Ilustración 21 se puede ver el aspecto de este chip.

Por último cabe añadir que en la plataforma se han instalado dos de estos chips, uno para tomar medidas desde el Arduino Due y otro para tomar medidas desde la Igep.

4.1.5.2. Sensor barométrico

Los sensores barométricos son aptos para medir casi cualquier altura en exteriores. Se basan en los cambios de presiones para dar medidas de altura. El inconveniente que presentan es que no

pueden detectar obstáculos por debajo del vehículo y que una ráfaga de viento puede generar una sobrepresión o depresión que afecte a la medida.

El sensor barométrico utilizado en este proyecto ha sido el MS5611 que incorpora el chip PU6050 anteriormente comentado. Este sensor incorpora un sensor de temperatura interno para corregir la medida de presión. Además funciona para rangos de presión entre 10 y 1200 mbar y rangos de temperatura entre -45 y 80º C.

Este sensor, así como la fusión de altura realizada ha sido implementado por el estudiante Daniel Verdú (Verdú Torres, 2014).

4.1.5.3. Magnetómetro

Un magnetómetro es un sensor que indica la posición del norte. Se trata de un sensor necesario a incluir en el quadrotor si se desea conocer la orientación del vehículo con el fin de implementar un control de posición.

El magnetómetro implementado en el quadrotor ha sido el HMC5883L que incorpora el chip PU6050. El problema que presenta este tipo de sensores es que no son válidos para interiores a causa de las interferencias electromagnéticas que existen. Son muy sensibles a ellas, es por ello que se recomienda utilizarlos únicamente en exteriores y lo más alejados posibles de los motores.

Cabe mencionar, que la medida que proporciona la IMU del ángulo de yaw no proviene del magnetómetro sino que procede de la fusión que realiza el DMP interno entre acelerómetros y giróscopos. Esta medida es bastante buena aunque deriva con el tiempo.

Este sensor y la fusión realizada con el yaw del DMP interno de la IMU, ha sido implementado por el estudiante Daniel Verdú (Verdú Torres, 2014).

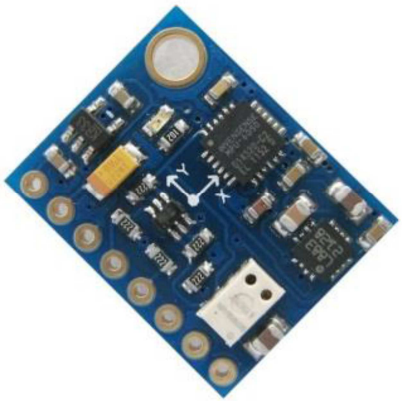

Ilustración 21. 10DOF IMU MPU6050 MS5611 HMC5883L

4.1.5.4. Sensor de ultrasonidos

El segundo sensor utilizado para medir la altura a la que se encuentra el quadrotor ha sido un sensor de ultrasonidos llamado Ping, de la empresa americana Parallax. El funcionamiento de este tipo de sensores se basa en la emisión de un pulso de ultrasonidos y la medición del tiempo que tarda el pulso en rebotar y volver.

En la Ilustración 22 se muestra el sensor implementado.

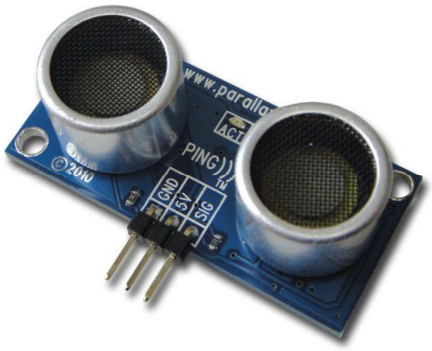

Ilustración 22. Sensor de ultrasonidos Ping.

Este sensor proporciona medidas bastante precisas y con poco ruido, sin embargo tiene algunos inconvenientes:

- Alcance reducido (desde 2 cm a 3 m).
- Funcionamiento lento debido a la propia dinámica del sensor.
- No es apto para superficies que amortigüen el pulso enviado como césped, agua, alfombra, etc.
- No es apto para superficies inclinadas.
- Medidas falseadas en caso de que existan objetos entre el quadrotor y el suelo.

A pesar de esto, el sensor Ping resulta muy útil por la calidad de sus medidas, sobre todo para las maniobras de despegue y aterrizaje.

En este proyecto, el sensor de ultrasonidos se ha instalado atado con unas bridas al tren de aterrizaje del quadrotor. Se alimenta desde el Arduino Due y se comunica con él a través de un pin digital mediante interrupciones.

4.1.5.5. Sensor GPS

El sistema global de navegación por satélite (GNSS) permite determinar en todo el mundo la posición de un objeto, una persona o un vehículo con una precisión hasta de centímetros (si se utiliza GPS diferencial), aunque lo habitual son unos pocos metros de precisión. El sistema GPS está constituido por 24 satélites y utiliza la triangulación para determinar en todo el globo la posición con una precisión de más o menos metros. El problema que tiene es que en interiores no es funcional.

El sensor GPS que se ha incorporado es el Ultimate GPS Breakout v3 de Adafruit. Es capaz de actualizar la posición a un máximo de 10Hz, con un consumo reducido y soporte para antena externa. Además soporta correcciones DGPS/WAAS y EGNOS.

El sensor se ha implementado en la capa superior de la electrónica (encima de la placa de la PCB del Arduino Due y de la Igep). Se alimenta desde el Arduino Micro y se comunica con éste mediante protocolo de comunicación serie TTL. Como se ha comentado anteriormente, no es la solución más óptima. Se ha tomado esta decisión porque no se podía conectar directamente a la Igep ya que no tenía más puertos series disponibles y tampoco se quería conectar al Arduino Due para no aumentar los tiempos de ciclo de su programa.

En la Ilustración 23 se muestra el aspecto de este sensor.

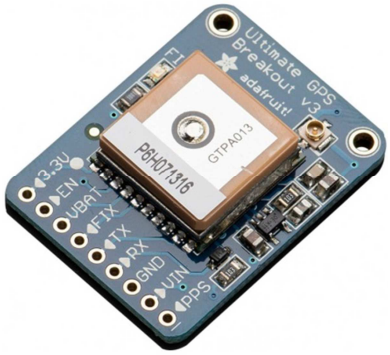

Ilustración 23. Ultimate GPS Breakout v3 de Adafruit.

El sensor GPS y el control de posición x-y ha sido implementado por el estudiante Alberto Castillo (Castillo Frasquet, 2014).

4.1.6. Otros Dispositivos

4.1.6.1. Módulo de radio

La solución adoptada para enlazar el quadrotor con la unidad de control terrestre (UCT) ha sido un módulo de radiofrecuencia. El módulo usado ha sido la Xbee Pro S1 de la empresa Digi, el cual se muestra en la Ilustración 24.

Ilustración 24. Xbee Pro S1.

El módulo de radio se ha conectado por serie a la Igep y se ha instalado en la capa superior de la electrónica.

En los capítulos posteriores se describirá detalladamente la solución tomada, la justificación y la implementación del enlace entre el quadrotor y la estación terrestre.

4.1.6.2. Receptor de radiocontrol

Por último también se ha implementado en la plataforma de vuelo un receptor de radiocontrol. La peculiaridad es que el receptor no se usa directamente sobre los motores sino que se utiliza para enviar ciertas referencias al quadrotor con el fin de controlarlo.

El receptor utilizado ha sido el modelo RP8D1 (Corona), en la Ilustración 25 se puede ver su aspecto.

Ilustración 25. Receptor de radiocontrol Corona RP8D1.

El receptor se ha instalado en la capa superior de la electrónica y la información que proporciona se lee desde el Arduino Micro.

En los capítulos posteriores se explicará la función que realiza este receptor, la razón por la cual es importante en la plataforma de vuelo y la implementación del mismo.

4.1.7. Esquema de funcionamiento de la plataforma

En la Ilustración 26 se muestra un esquema detallado del funcionamiento global de la plataforma.

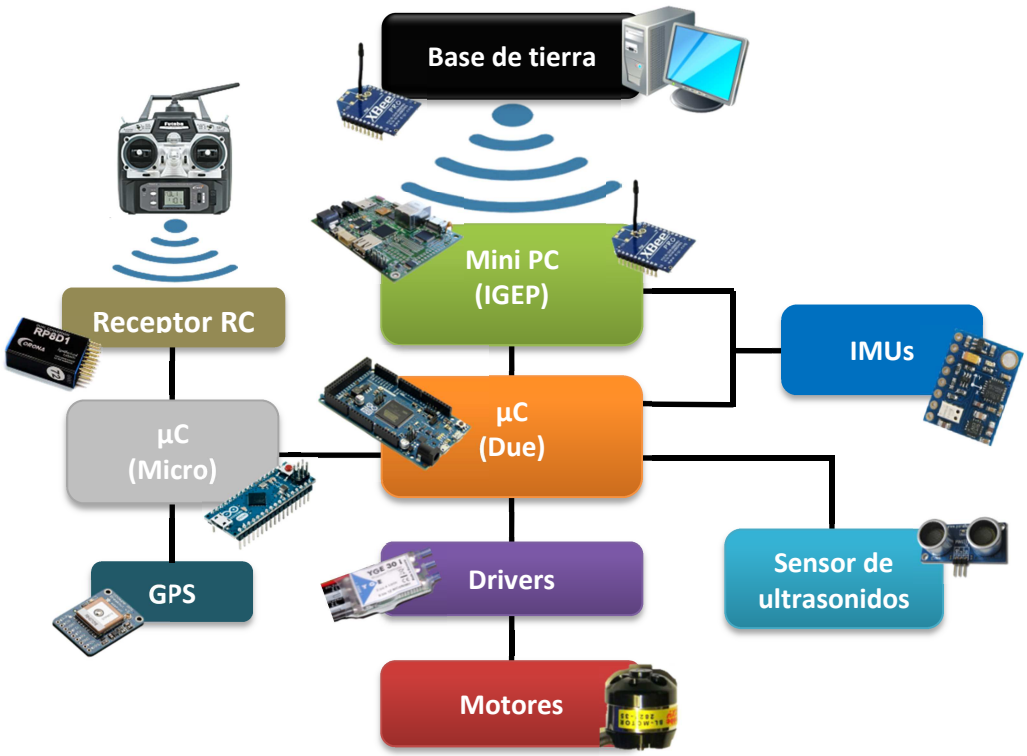

Ilustración 26. Esquema de funcionamiento global de la plataforma.

A continuación se muestra la versión final del quadrotor en la Ilustración 27.

4.2. Software de la Plataforma

A continuación se describe brevemente el software implementado en la plataforma. Para que funcionen todos los sistemas correctamente se han desarrollado tres programas distintos: un programa para el Arduino Due, otro para la Igep y otro para el Arduino micro. Además existe un cuarto programa no perteneciente directamente a la plataforma pero necesario para el control y monitorización de la misma, que es la interfaz hombre máquina (HMI).

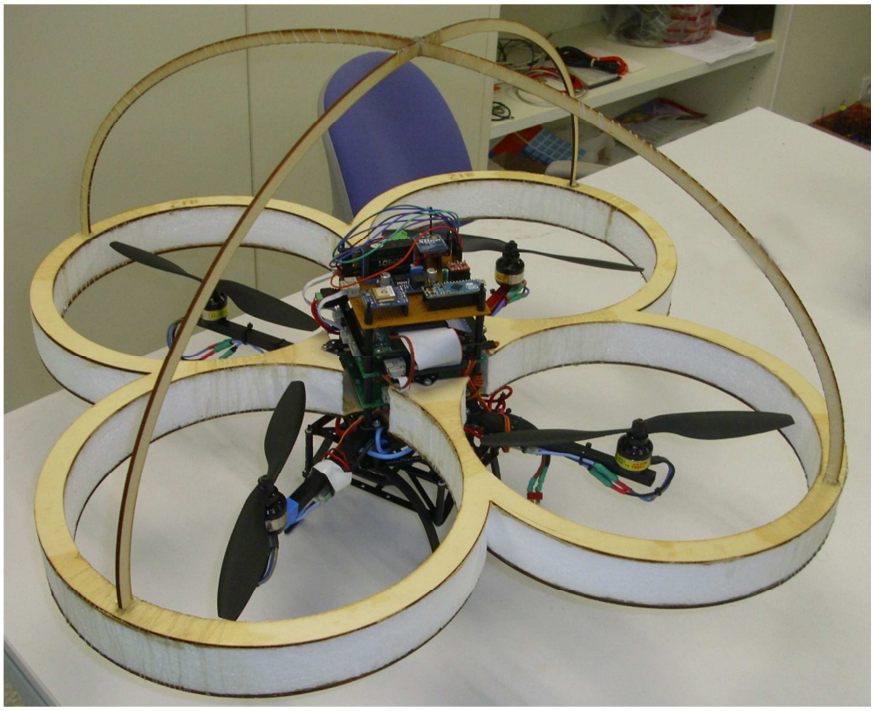

Ilustración 27. Versión final de la plataforma.

No obstante, el programa más importante es el del Arduino Due. Únicamente con éste ya se consigue que el quadrotor sea estable, aunque no es posible actuar sobre el mismo. La combinación de los programas Arduino Due + Arduino Micro ya permite controlar el quadrotor mediante emisora de radiocontrol mientras que la combinación Arduino Due + Igep + HMI permite el control de la plataforma mediante un módulo de radio. Por último, la síntesis de todos los programas ofrece todas las alternativas de control y modos de funcionamiento.

4.2.1. Arduino Due

El código del Arduino Due está programado en lenguaje C utilizando el entorno de programación de Arduino, el Arduino IDE. Realizar modificaciones es muy sencillo, basta con conectar el Arduino al PC con un USB, compilar el programa y subirlo.

El código se divide en tres partes, una parte de declaración de variables globales, otra parte de inicialización del programa (setup) y por último el bucle principal (loop).

El "setup", se ejecuta únicamente al encender el Arduino y realiza las siguientes tareas:

- Definir el estado de los pines digitales (entradas o salidas y niveles altos o bajos).
- Inicializar la comunicación I2C
- Inicializar la comunicación serie con el PC.
- Inicializar la comunicación serie con la Igep.
- Inicializar la comunicación serie con el Arduino Micro.
- Inicializar los motores.
- Inicializar el DMP de la IMU.
- Calcular el offset para el ángulo del yaw.

El “loop” se ejecuta a una frecuencia constante de 250 hercios y realiza, por orden, las siguientes funciones:

- Leer el voltaje de la batería.
- Recibir datos de la Igep.
- Recibir datos del Arduino Micro.
- Leer la IMU.
- Leer el sensor de ultrasonidos y obtener una medida de la altura.
- Calcular las acciones de control a aplicar para estabilizar el quadrotor.
- Comprobar los protocolos de seguridad.
- Enviar las acciones de control a los motores.
- Enviar datos a la Igep.

Cabe destacar que todas las funciones se ejecutan de manera cronometrada para fijar los tiempos de ejecución. Además el código está programado a prueba de errores en las comunicaciones o fallo de sensores para evitar que el programa se bloquee.

4.2.2. Igep v2

A diferencia del Arduino, la Igep puede ejecutar desde una SD un sistema operativo compatible con el procesador ARM que incorpora.

En la plataforma actual, se ha instalado un sistema operativo tipo Linux con un parche de tiempo real llamado Xenomai. Este parche modifica el comportamiento del núcleo para realizar las tareas necesarias ignorando parte de las interrupciones del sistema. En la Ilustración 28 se muestra un esquema. Todo esto ha sido desarrollado durante los últimos años gracias a la colaboración de varios estudiantes e investigadores, siendo el principal encargado el director de este proyecto, Pedro García Gil.

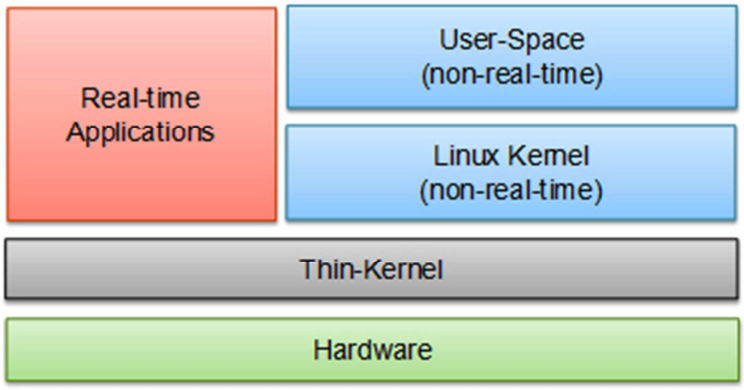

Ilustración 28. Esquema simplificado de la arquitectura de Xenomai.

El código de la Igep está programado en lenguaje C++ y se ha utilizado el entorno de programación de Code::Blocks para su desarrollo. Para compilarlo es necesario instalar un compilador cruzado compatible con la arquitectura del procesador de la Igep que además incluya las librerías de Xenomai.

Una vez que el programa está compilado correctamente, se sube a la Igep. Para ello, existen varias alternativas, en este proyecto se ha optado por la conexión por WiFi mediante protocolo SSH ("Secure Shell") por ser más rápida y cómoda que otras opciones. En este punto, el programa subido se puede ejecutar desde la misma terminal de comandos.

En cuanto al código del programa, se encuentra dividido en distintos hilos de ejecución que se ejecutan de forma paralela. A modo resumen, los hilos que se ejecutan son los siguientes:

- Bucle principal (main del programa) que define algunas variables e hilos.
- Thread de telemetría que gestiona la comunicación con la unidad de control terrestre (UCT). Funciona a 5 Hz.
- Thread de control de posición y modos de vuelo de la plataforma. Funciona a 1000 Hz y es de tiempo real.
- Hilo de comunicación con el Arduino Due. Funciona a 1000 Hz y es de tiempo real.
- Bucle para la lectura de la IMU conectada a la Igep. Funciona a 300 Hz y es de tiempo real.
- Bucle que ejecuta un filtro de Kalman para el control de posición. Funciona a 300 Hz y es de tiempo real.

En la Ilustración 29 se muestra un esquema de los hilos que se ejecutan en el programa de la Igep y las comunicaciones entre ellos.

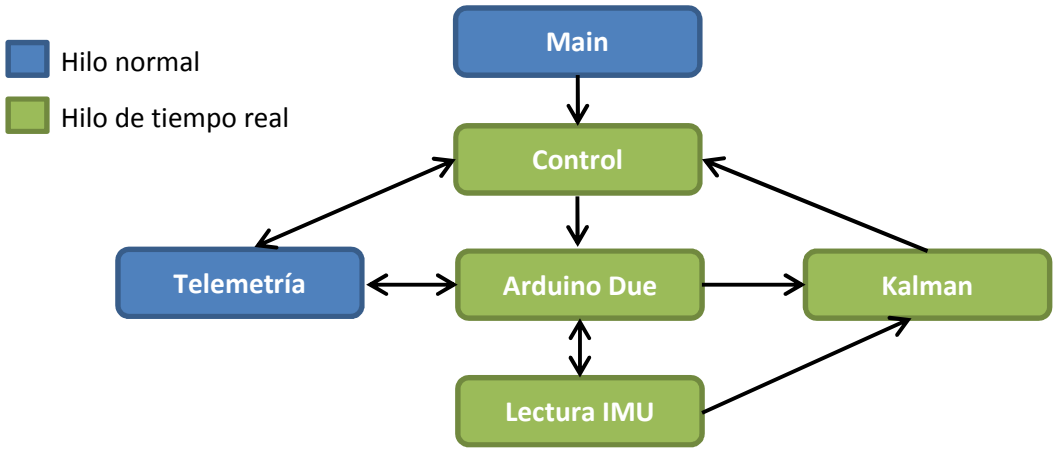

Ilustración 29. Comunicaciones entre hilos.

Hay que añadir que a lo largo de todo el código se ha tenido la precaución de añadir los semáforos necesarios para evitar la lectura y escritura simultánea de variables desde los distintos threads.

4.2.3. Arduino Micro

El código del Arduino Micro también se ha programado en lenguaje C mediante el IDE de Arduino. Al igual que el Arduino Due, el código se divide en tres partes: declaración e inicialización de variables globales, inicialización del programa ("setup") y bucle principal ("loop").

Las funciones que realiza el "setup" son las siguientes:

- Inicializar la comunicación serie con el PC.

- Inicializar la comunicación serie con el Arduino Due.
- Inicializar la comunicación serie con el GPS.
- Inicializar el GPS.
- Definir 6 pines digitales como entradas a leer.

Por otra parte, el bucle principal ("loop") realiza las siguientes tareas a una frecuencia de 5 hercios:

- Comprobar el estado de la emisora de radiocontrol (encendida/apagada).
- Leer las señales PWM procedentes del receptor RC.
- Leer las tramas del sensor GPS.
- Enviar la información al Arduino Due.

Cabe mencionar que se ha añadido un algoritmo que comprueba cuando se ha perdido la señal con la emisora de radiocontrol, esto permite ejecutar un algoritmo de seguridad para que el quadrotor aterrice de la forma más segura posible.

4.3. Estructura del Sistema de Control

El objetivo de este apartado es describir la estructura del sistema de control implementado para la estabilidad del quadrotor de forma breve. Se puede encontrar una exposición más amplia de este apartado en otros trabajos (Castillo Frasquet, 2014), (Verdú Torres, 2014).

4.3.2. Modelo dinámico

Para la obtención del modelo matemático del quadrotor, se utiliza la aproximación de las ecuaciones de Euler-Lagrange. Después de una serie de cálculos y simplificaciones, el modelo queda como (Castillo, Lozano, & Dzul, 2005):

$$\mathrm{m}\ddot{\mathrm{x}} = -\mathrm{u}\sin\theta \tag{4.1a}$$

$$\mathrm{m}\ddot{\mathrm{y}} = \mathrm{u}\cos\theta\sin\phi \tag{4.1b}$$

$$\mathrm{m}\ddot{\mathrm{z}} = \mathrm{u}\cos\theta\cos\phi - mg \tag{4.1c}$$

$$\ddot{\psi} = u_{\psi} \tag{4.1d}$$

$$\ddot{\theta} = u_{\theta} \tag{4.1e}$$

$$\ddot{\phi} = u_{\phi} \tag{4.1f}$$

Donde (x, y, z) denotan la posición del centro de masas del quadrotor, (ψ, θ, ϕ) describen los tres ángulos de Euler (yaw, pitch, roll), m es la masa del quadrotor, g es la aceleración de la gravedad y u el empuje total. Las variables u_{ψ}, u_{θ}, u_{ϕ} se definen como las entradas de control de los ángulos (ψ, θ, ϕ) para el control en equis tales que:

$$u_{\psi} = u_{\psi}(V_1 - V_2 + V_3 - V_4) \tag{4.2a}$$

$$u_{\theta} = u_{\theta}(V_1 + V_2 - V_3 - V_4) \tag{4.2b}$$

$$u_{\phi} = u_{\phi}(V_1 - V_2 - V_3 + V_4) \tag{4.2c}$$

Donde k_ψ, k_θ, k_ϕ son los parámetros de momentos de inercia y V_1, V_2, V_3, V_4 las tensiones aplicadas a los motores, numerados en orden horario.

4.3.1. Ley de control

Como se puede ver en el apartado anterior, el modelo dinámico para los ángulos de Euler es un doble integrador. La ley de control implementada en este vehículo, ha sido obtenida a partir de la literatura (Sanahuja, Castillo, & Sánchez, 2009). Para los tres ángulos (yaw, pitch, roll) queda:

$$u_{PD} = f_s(k_d\dot{\alpha}, \zeta_1) - f_s\big(k_p(\alpha - \alpha_d), \zeta_2\big) \quad (4.3a)$$

$$u_I = f_s(u_I + k_i(\alpha - \alpha_d), \zeta_3) \quad (4.3b)$$

$$u_\alpha = u_{PD} + u_I \quad (4.3c)$$

Donde k_d, k_p, k_i, ζ_1, ζ_2, ζ_3 son constantes positivas, α es el ángulo considerado, α_d es el valor del ángulo deseado y f_s se define como una función de saturación acotada por los valores $\pm\zeta_i$.

Todas estas constantes se han obtenido de forma experimental a partir de pequeños ensayos hasta obtener la respuesta deseada. Para ello, se ha hecho uso de la interfaz HMI que, como se explicará en los capítulos siguientes, tiene un bloque dedicado al ajuste online de los parámetros de control del quadrotor.

Las diferentes acciones de control se suman prestando atención a los signos para obtener los valores de referencia que se envían a los drivers de los motores. Esta suma es diferente según se esté realizando un control en cruz o control en equis. Para un control en equis las ecuaciones a utilizar son:

$$V_1 = \mathrm{u} + u_\psi + u_\phi + u_\theta \quad (4.4a)$$

$$V_2 = \mathrm{u} - u_\psi - u_\phi + u_\theta \quad (4.4b)$$

$$V_3 = \mathrm{u} + u_\psi - u_\phi - u_\theta \quad (4.4c)$$

$$V_4 = \mathrm{u} - u_\psi + u_\phi - u_\theta \quad (4.4d)$$

Cabe añadir que para pequeñas variaciones de los ángulos de orientación, la altura z se puede considerar desacoplada del resto de variables, siendo el modelo resultante un doble integrador. Por lo tanto la ley de control utilizada es la misma que para el control de estabilidad. En este proyecto, el control de altura ha sido implementado por el estudiante Daniel Verdú (Verdú Torres, 2014).

Respecto al control de posición, ha sido implementado por el estudiante Alberto Castillo (Castillo Frasquet, 2014). El control de posición se ha realizado en la Igep mediante un control en cascada sobre los ángulos de pitch y roll.

5. ÁMBITO DE APLICACIÓN Y RANGO DE SOLUCIONES DEL TRABAJO

En el caso de un vehículo aéreo remotamente pilotado (RPAS) se hace necesario disponer de sistemas de telemetría para poder monitorizar las variables más importantes del mismo, así como para poder contralarlo, con objeto de realizar vuelos en modo semiautónomo. Para esto, es necesario tener un enlace entre el vehículo y la unidad de control terrestre (UCT), y un software en la UCT para la monitorización y control del quadrotor. Es por ello que este apartado consta de dos partes.

En la primera parte se analizarán las necesidades de hardware y software que hagan posible la comunicación entre el quadrotor y la unidad de control terrestre, haciendo énfasis en los condicionantes de diseño así como las alternativas disponibles.

En la segunda parte se afrontará el problema del software de monitorización y control del RPAS. Primero se describirán los condicionantes de diseño y a continuación se propondrán una serie de alternativas de lenguajes de programación y entornos de desarrollo, analizando las ventajas y desventajas de cada una de las opciones propuestas.

Por último, también será necesario disponer de otro mecanismo auxiliar para recuperar el control del vehículo en caso de fallo del enlace principal de telemetría o del ordenador de a bordo del quadrotor. Esto se ha resuelto mediante una emisora y un receptor de radiocontrol.

5.1. Enlace del Quadrotor con la Unidad de Control Terrestre

5.1.1. Introducción

El Hardware de comunicación consta de dos partes, un módulo montado en el quadrotor y otro módulo instalado en la estación de tierra, de tal manera que lo que se envíe por un lado sea recibido por el otro y viceversa.

Respecto al software, será necesario definir en qué unidad de control (Arduino Due, Igep o Arduino Micro) se gestiona el envío de los paquetes de telemetría, cómo se lleva a cabo y cómo se comunica con el resto de los sistemas presentes en la plataforma de vuelo.

5.1.2. Condicionantes de diseño

En cuanto a los condicionantes de diseño, lo más importante es disponer de un enlace bidireccional para la monitorización y control del quadrotor desde la UCT en tiempo real. Además de esto, hay cuatro frentes claros que se deben tener en cuenta para decantarse por una opción u otra.

Por un lado, es muy importante que la solución adoptada tenga un tamaño y un peso reducido, de forma que se pueda instalar fácilmente en el quadrotor sin alterar la disposición de los equipos ya instalados. Además, cuanto menor sea el peso del equipo que se instale a bordo del quadrotor, mayor será la duración de la batería. Asimismo, la solución tomada ha de consumir poca energía. Como se ha explicado a lo largo de esta memoria, uno de los puntos débiles de los vehículos cuadrirrotores es su baja autonomía, por lo tanto es requisito imprescindible que los equipos que se añadan al vehículo tengan un consumo de electricidad mínimo.

El segundo frente al que se le debe prestar atención es al alcance del enlace entre el quadrotor y la unidad de control terrestre (UCT). Este requisito es menos crítico que el anterior puesto que los quadrotors se utilizan en general para aplicaciones de vuelos estacionarios, sin embargo, sí que es necesario tener un enlace robusto para distancias de 100 a 200 metros. Además, si se quiere volar el vehículo a distancias mayores, el quadrotor se encontrará lo suficientemente testado para que la pérdida del enlace de telemetría no suponga riesgo alguno.

También hay que tener en cuenta el ancho de banda o velocidad de transmisión de datos. En un principio no es un parámetro muy importante porque no se desea transmitir grandes paquetes de datos como por ejemplo video. No obstante, sí que resulta importante garantizar un enlace bidireccional entre el quadrotor y la unidad de control terrestre con una frecuencia de envío media de 5 Hz.

Por último y no menos importante, la solución tomada se debe poder conectar fácilmente tanto al quadrotor como a cualquier ordenador que se utilice como unidad de control terrestre. Así pues, el protocolo de comunicación serie resulta idóneo para este propósito, ya que tanto la Igep como los Arduinos disponen de múltiples puertos de este tipo. Además, los ordenadores también cuentan con puertos series y puertos USB, los cuales se podrían utilizar por medio de un adaptador.

En resumen, se tienen las siguientes especificaciones:

- Enlace quadrotor - UCT bidireccional.
- Dimensiones reducidas. Máximo 50x50x15 mm.
- Peso reducido. Máximo 50 g.
- Consumo reducido. Máximo 100 mW.
- Ancho de banda mínimo 9600 bps.
- Alcance mínimo 100-200 m.
- Comunicación por protocolo serie.

5.1.3. Alternativas de diseño

Durante la realización de este proyecto se ha tratado de utilizar los equipos existentes en el laboratorio evitando la compra de otros si era posible. Así pues, las posibilidades existentes eran: utilizar el WiFi de la Igep o utilizar dos módulos de radiofrecuencia Xbee Pro S1.

5.1.3.1. WiFi Igep

Como se ha comentado en el apartado de descripción del hardware de la plataforma de vuelo, la Igep v2 dispone de un chipset dedicado a la comunicación por WiFi que cumple el estándar 802.11b/g (Ilustración 30).

Ventajas:

- No es necesario instalar otro dispositivo dedicado a la comunicación entre el quadrotor y la unidad de control terrestre, por lo tanto no se aumenta el peso del vehículo ni el consumo de energía.

- En la estación de tierra se puede utilizar el WiFi de cualquier ordenador sin instalar otro dispositivo específico.
- Alta frecuencia de transmisión de datos (hasta 11 Mbps).

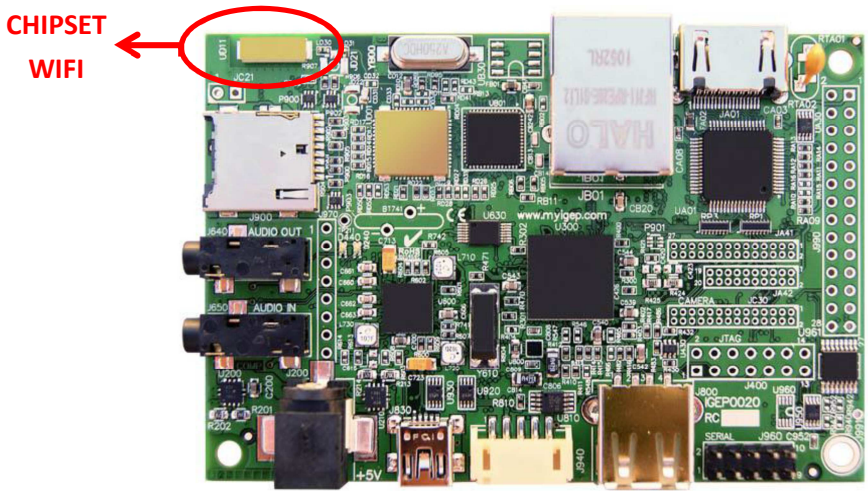

Ilustración 30. Chipset WiFi 802.11b/g de la Igep.

Desventajas:

- El alcance de la cobertura WiFi de la Igep v2 sin antena externa se encuentra en torno a los 10 metros. Con una antena externa puede aumentar hasta 20 metros en interiores y 50 metros en exteriores.
- Si en el futuro se cambia el ordenador de a bordo, habría que buscar otra alternativa para resolver la comunicación entre el vehículo y la UCT.

5.1.3.2. Xbee Pro S1

La Xbee es un módulo de radiofrecuencia de bajo coste, bajo consumo muy utilizado para crear redes de área local inalámbricas que sigue la especificación 802.15.4. A continuación (Tabla 6), se detallan sus características técnicas. En la Ilustración 24 se muestra su aspecto.

Potencia transmitida	10 mW (10dBm)
Sensibilidad del receptor	-100 dBm
Alcance en interiores	60 m
Alcance en exteriores	700 m (LOS)
Ancho de banda	250000 bps
Frecuencia	2,4-2,4835 GHz
Dimensiones	2,438x3,294 cm
Peso	4,5 g
Interfaz de datos	Serie 3,3 V
Opciones de antena	UFL, RPSMA, Wire, Chip

Tabla 6. Características técnicas Xbee Pro S1.

Ventajas:

- Alcance máximo de 700 metros para aplicaciones en exteriores en condiciones de línea de visión (LOS) y 60 metros en interiores.
- Bajo consumo, máximo de 10 mW.
- Comunicación por protocolo serie.
- Velocidad de transmisión de datos de 250000 bps.
- Dimensiones y peso reducido.

Desventajas:

- Es necesario instalar otro dispositivo en el quadrotor y comunicarlo con el resto de equipos.
- La alimentación del módulo de radiofrecuencia tiene que poder suministrar la corriente necesaria para obtener un buen alcance.

5.2. Interfaz Gráfica Hombre Máquina (HMI)

5.2.1. Introducción

Para cerrar el sistema de telemetría principal queda desarrollar el software necesario que permita visualizar de forma rápida y clara el estado del quadrotor en cada una de las fases de vuelo.

A lo largo de la memoria se ha destacado la importancia de conocer las variables más importantes de un sistema con objeto de automatizarlo. En el caso de un quadrotor, variables tales como los ángulos de Euler, posición, altitud o velocidades, resultan fundamentales si se aspira a realizar un control del vehículo.

Asimismo, la interfaz hombre máquina (HMI) cumplirá otra función además de la monitorización, el control del vehículo. Esto es, desde el HMI se podrá actuar sobre el quadrotor, ya sea cambiando el modo de vuelo, cambiando las referencias sobre los ángulos de Euler o incluso cambiando la estrategia de control y sus parámetros característicos.

5.2.2. Condicionantes de diseño

5.2.2.1. Requisitos de la aplicación

La condición más importante del software a desarrollar es que cuente con una interfaz gráfica de usuario que muestre de forma clara y rápida un resumen de las variables más importantes del quadrotor. Además, también se tendrá en cuenta a la hora de decantarse por una opción u otra que la aplicación desarrollada sea compatible con sistemas operativos diferentes.

Asimismo, se quiere que el programa desarrollado sea una aplicación de escritorio. Esto es, la aplicación tiene que contar con un sistema de menús, barra de herramientas, barra de estado, etc.

Respecto a la monitorización del vehículo, las funciones que tiene que ofrecer el programa son las siguientes:

- Instrumentos de vuelo (indicador de velocidad horizontal, horizonte artificial, altímetro, brújula e indicador de velocidad vertical) para visualizar de forma rápida el estado del vehículo.
- Mapa para posicionar en todo momento el quadrotor, dibujar la trayectoria del vuelo realizado y herramientas para introducir "waypoints" sobre el mapa.
- Panel con indicadores varios tales como: modo de vuelo, estado de vuelo, nivel de la batería, cobertura de la señal GPS, calidad del enlace de telemetría, etc.

En cuanto al control del quadrotor, el HMI debe tener las siguientes opciones:

- Opción para cambiar el modo de vuelo del quadrotor (desarmado, estabilizado o guiado). En el modo desarmado el vehículo se encuentra en tierra con los motores apagados, el modo estabilizado se utiliza en vuelo para controlar el quadrotor actuando sobre las referencias de roll, pitch, yaw, x, y, z. Por último, el modo guiado sirve para que el vehículo siga un plan de vuelo previamente establecido en el ordenador de a bordo.
- Apartado para cambiar las referencias sobre los ángulos de Euler del vehículo así como el tipo de control a realizar y los sensores a utilizar.
- Sección para configurar los parámetros PID del vehículo, tanto los referentes al control de estabilidad como al control de posición.
- Opciones de conexión de joystick o similares para controlar el quadrotor.
- Ventana para gestionar el envío de waypoints al vehículo. Aunque no se llegue a realizar en el quadrotor, sí que conviene dejar la interfaz preparada a nivel de telemetría para que en un futuro se pueda implementar.

Además de esto, es interesante que la aplicación permita guardar datos de vuelo para reproducirlos posteriormente. También es de utilidad que el programa ofrezca funciones para exportar las variables más importantes del quadrotor a ficheros *.txt con objeto de realizar un análisis más pormenorizado de ellas o incluso que se puedan representar gráficamente en la misma aplicación en tiempo real.

5.2.2.2. Requisitos para el desarrollo

En primer lugar se va a descartar cualquier tipo de software comercial puesto que a lo largo de la vida del proyecto es probable que se encuentren diferentes necesidades de tal modo que haya que modificar el código para incorporar nuevas funciones. También se va a descartar en un principio cualquier tipo de software de código abierto, ya que no se conoce su código en profundidad y las versiones existentes corren el riesgo de no ser del todo estables. A pesar de esto, sí que se espera que la plataforma de vuelo final sea compatible tanto con el software propio desarrollado como con software de terceras personas.

Como se ha comentado anteriormente, resulta importante que el entorno de desarrollo utilizado permita crear aplicaciones multiplataforma. A lo largo de este proyecto, se ha utilizado tanto Windows para la programación de los microcontroladores Arduino como Linux para desarrollar el código de la Igep, por lo tanto cobra importancia el hecho de tener una aplicación final que se pueda ejecutar en varios sistemas operativos.

Otra característica exigible al entorno de desarrollo es que permita flexibilidad a la hora de realizar modificaciones sobre el código. Sea cual sea el lenguaje escogido, se debe conseguir que el código final esté bien estructurado y comentado, de modo que sea inteligible por nuevos desarrolladores y que, modificaciones en unas partes del código, afecten los menos posible al resto de partes de la aplicación.

Hay que tener en cuenta que el componente gráfico cobra importancia en una aplicación de este tipo, es por ello que se requiere una combinación de lenguaje de programación y entorno de desarrollo que ofrezca herramientas potentes para trabajar con interfaces gráficas de usuario.

Por último, el entorno de desarrollo elegido debe ofrecer facilidades para trabajar con distinto tipo de hardware. Se desea enviar y leer datos por puerto serie si se trabaja con un módulo de radiofrecuencia o enviar y recibir información mediante sockets si se utiliza comunicación WiFi. Además también se quiere leer dispositivos como joysticks o gamepads para controlar el quadrotor. Es por ello que el lenguaje de programación escogido deberá pues ofrecer cierta cercanía al hardware, permitiendo el trato con tramas de bytes y con puertos del ordenador.

5.2.3. Alternativas de diseño

En el punto que sigue, se comparan distintas combinaciones de lenguajes de programación y entornos de desarrollo analizando las ventajas y desventajas de cada una de ellas.

5.2.3.1. Qt

Qt es una biblioteca multiplataforma para desarrollar aplicaciones en C++ con interfaz gráfica de usuario o sin ella y que además cuenta con IDE propio (Qt Creator).

Ventajas:

- Programación en C++ (ejecución rápida).
- Lenguaje orientado a objetos.
- IDE propio (Qt Creator) con opciones para el desarrollo de interfaces gráficas de usuario.
- Multiplataforma en compilación.
- Lenguaje de alto nivel pero posibilidad de trabajar a bajo nivel (programación C/C++).
- Programación dirigida por eventos.
- Programación web, multihilo, bases de datos, etc.
- Software libre y código abierto.

Desventajas:

- Curva de aprendizaje media. Es imprescindible tener conocimientos de programación orientada a objetos C/C++.
- Poco conocido por ingenieros industriales.
- Es necesario instalar las librerías de Qt o distribuirlas con el programa desarrollado.
- Si se desea desarrollar un software comercial y no se quiere compartir el código fuente hay que pagar una licencia.

5.2.3.2. LabView

LabVIEW es una plataforma y entorno de desarrollo para diseñar sistemas de medidas y control, con un lenguaje de programación visual gráfico desarrollado por National Instruments.

Ventajas:

- Curva de aprendizaje muy rápida.
- Múltiples herramientas para crear interfaces gráficas de usuario.
- Facilidades para personalizar el aspecto del programa desarrollado.
- Funciones para implementar y temporizar distintos hilos de ejecución.
- Gran número de funciones implementadas para el manejo de hardware externo.

Desventajas:

- Programación gráfica. El programa es difícil de leer por terceras personas cuando su complejidad es mayor.
- No es multiplataforma ni en compilación ni en ejecución.
- Código poco optimizado.
- Software de pago.

5.2.3.3. Netbeans

NetBeans es un entorno de desarrollo integrado libre, hecho principalmente para el lenguaje de programación Java.

Ventajas:

- Programación principalmente en Java. Lenguaje orientado a objetos.
- Multiplataforma en ejecución.
- Soporte para otros lenguajes de programación como PHP, C/C++, HTML5, etc.
- IDE con opciones para el desarrollo de interfaces gráficas de usuario.
- Software de código abierto.

Desventajas:

- Curva de aprendizaje media.
- Ejecución lenta. El código es interpretado.
- Sintaxis generalmente poco conocida por ingenieros industriales.
- Lenguaje de alto nivel. Alta abstracción del hardware.

5.3. Enlace del Quadrotor con una Emisora de Radiocontrol

5.3.1. Introducción

A lo largo de esta memoria, se ha destacado el hecho de que es necesario contar con las medidas de seguridad necesarias para tener en todo momento el control del vehículo. Esto cobra aún más importancia en un proyecto de este tipo donde se implementan distintos algoritmos de control, se modifica el código de los programas constantemente, etc. Por este motivo, muchas veces resulta impredecible el desempeño que va a tener la plataforma de vuelo. Es por ello que se hace

necesario disponer de algún mecanismo auxiliar para retomar el control del quadrotor a pesar de que falle el ordenador de a bordo o el sistema de telemetría principal.

Esto se ha resuelto con una emisora y un receptor de radiocontrol que funcionan de forma independiente al ordenador de a bordo del quadrotor. La idea es la siguiente: cuando el usuario del quadrotor encienda la emisora y active el modo de control manual entonces el quadrotor sólo obedecerá a los controles de la emisora ignorando la información enviada desde la Igep (el ordenador de a bordo) al Arduino Due.

5.3.2. Condicionantes de diseño

Controlar de forma totalmente manual un quadrotor resulta muy complicado porque se tendría que estar variando la acción aplicada a cada motor en todo momento sin perder la concentración ni un instante de tiempo para que el vehículo se encuentre en equilibrio. Resulta más difícil aún en una plataforma de este tipo donde se han incluido multitud de equipos electrónicos y consecuencia de ello, el centro de gravedad del vehículo está desplazado.

Por lo tanto, es requisito de diseño que la emisora de radiocontrol permita controlar el quadrotor pero que sea el software de éste el que se encargue de estabilizar el vehículo para conseguir un pilotaje fácil, cómodo y seguro.

Atendiendo a las características del enlace entre la emisora y el quadrotor, como se ha explicado en apartados anteriores, no es crítico el alcance del enlace puesto que el vehículo se va a utilizar mayormente para vuelos estacionarios. Sin embargo, sí que es necesario disponer de un buen enlace para distancias de 100 a 200 metros.

Respecto al número de canales necesarios, en principio se necesita: un canal para el gas (acelerador), un canal que controle el ángulo de pitch, otro el ángulo de roll y otro el ángulo de yaw. Además se necesita otro canal que controle el modo de funcionamiento manual/automático y por último se ha utilizado otro canal para realizar un control de altura cuando se encuentra activado.

En cuanto al receptor, el único requisito es que sea compatible con la emisora elegida, que tenga un tamaño reducido de forma que sea fácilmente instalado en la plataforma de vuelo y sea fácil de alimentar.

En resumen, se tienen las siguientes especificaciones:

- Alcance mínimo 100-200 m.
- 6 canales de emisión.
- Receptor de radiocontrol pequeño.

5.3.3. Alternativas de diseño

Respecto a la emisora de radiocontrol se ha utilizado una de las tres que poseía el grupo de investigación de Pedro García Gil. La elección ha sido sencilla pues no había alternativa. Las tres emisoras que había disponibles eran modelos diferentes de la misma marca pero sólo una de ellas contaba con 6 canales de emisión.

La emisora elegida ha sido la Futaba 6EXP que se muestra en la Ilustración 31. Se trata de una emisora utilizada para radiocontrol que funciona en la banda de frecuencias de 35 MHz, posee 6 canales, admite modulación PPM y PCM y junto con el receptor adecuado el alcance puede llegar hasta 1 kilómetro.

Ilustración 31. Futaba 6EXP.

Una vez elegida una emisora que funciona en la banda de los 35 MHz, el receptor a utilizar tiene que utilizar la misma frecuencia. En este caso, sí que ha sido necesario comprar uno nuevo, ya que de los disponibles en el laboratorio ninguno contaba con 6 canales.

Por otro lado, las emisoras y receptores de radiocontrol actuales funcionan en la banda de 2,4 GHz estando la frecuencia de 35 MHz algo desfasada. El único receptor de 35 MHz que se ha encontrado en tiendas de España es el receptor Corona RP8D1, el cual se ha mostrado anteriormente en la Ilustración 25.

Se trata de un receptor de radiocontrol sinterizado. Esto significa que no necesita un cristal para funcionar, automáticamente sintoniza el canal de frecuencia que utiliza la emisora y lo guarda en su memoria. Además, es un receptor pequeño y ligero que admite hasta 8 canales de emisión.

Por último, sí que es necesario que la emisora disponga de un cristal compatible con la banda de frecuencias que admite. En la Ilustración 32 se muestra el cristal utilizado.

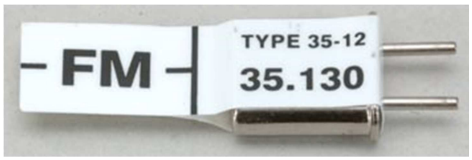

Ilustración 32. Cristal de 35 MHz.

6. DESCRIPCIÓN DE LAS SOLUCIONES ADOPTADAS

6.1. Enlace del Quadrotor con la Unidad de Control Terrestre

6.1.1. Hardware

La solución adoptada en términos de hardware para realizar el enlace entre el quadrotor y la unidad de control terrestre ha sido utilizar los módulos de radiofrecuencia Xbee Pro S1. Uno montado en el quadrotor y otro en la estación de tierra.

La razón principal que ha llevado a esta elección ha sido el alcance del enlace deseado. Utilizar el WiFi de la Igep parece una buena opción a priori, pero presenta un grave inconveniente, el reducido alcance que posee. Los módulos de radiofrecuencia Xbee permiten alcanzar distancias 20 veces mayores.

Respecto a las antenas, los módulos de radio utilizados vienen con la "whip antenna" o "wire antenna". En la Ilustración 33 se muestran los diferentes tipos de antenas disponibles para la xbee.

Ilustración 33. Diferentes tipos de antenas para las Xbee.

Cada antena tiene sus ventajas e inconvenientes. La "chip antenna" resulta apta si el espacio disponible para instalar el módulo de radio es un problema pero tiene la desventaja de que el alcance es más reducido. Por el contrario, si se utiliza un módulo con conector RPSMA se tiene un alcance mucho mayor pero la antena ocupa más espacio y es más pesada. Por último, la "whip antenna" se encuentra a mitad camino de las dos anteriores, es decir, no ocupa mucho espacio y posee un alcance bastante superior a la "chip antenna", no obstante, el alcance del módulo con conector RPSMA es mayor.

En la Ilustración 34 se muestran los patrones de radiación de las antenas "whip" y "chip" normalizados respecto a la antena dipolo. Como se puede ver, se trata de antenas omnidireccionales, esto es, emiten por igual en todas las direcciones en un mismo plano.

En el quadrotor es necesario instalar una antena de este tipo porque la orientación del mismo puede cambiar, por lo que resulta imprescindible que la antena irradie en todas las direcciones. En la unidad de control terrestre sí que se podría utilizar una antena direccional con mayor ganancia si se quiere tener un alcance mayor.

La ganancia de la antena “whip” utilizada es de 1,5 dBi, mientras que la antena “chip” tiene una ganancia de -0,5 dBi y la antena dipolo 2.1 dBi.

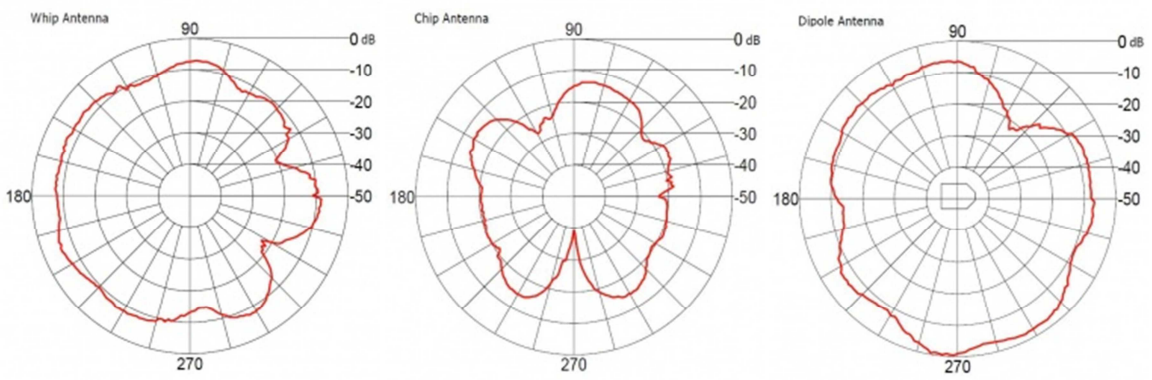

Ilustración 34. Patrones de radiación de las antenas “whip”, “chip” y la antena dipolo.

Además, se han ensayado los dos módulos de radiofrecuencia Xbee Pro S1, con la antena “whip”, que se van a utilizar en el quadrotor. El ensayo se ha realizado enviando paquetes de un tamaño de 35 bytes, que es el tamaño medio de los paquetes que se enviarán desde el quadrotor a la UCT, desde un módulo a otro. Además, el ensayo se ha realizado en condiciones de línea de visión entre los dos módulos de radio. Los resultados se muestran en la Ilustración 35.

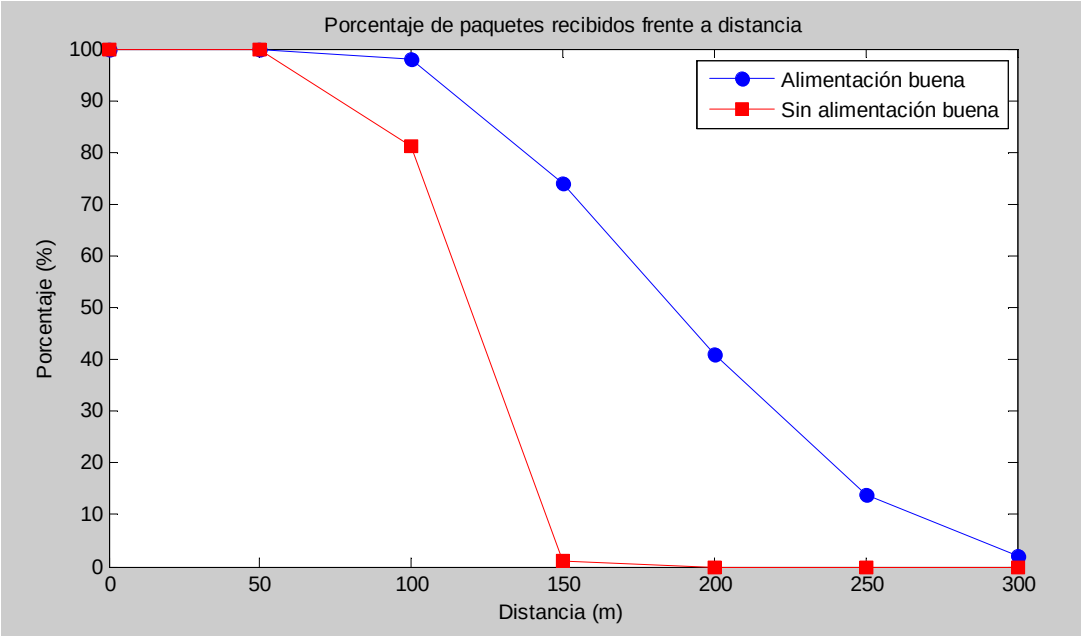

Ilustración 35. Porcentaje de paquetes recibidos frente a distancia.

Como se puede observar, el alcance no llega al máximo de 700 metros que se especifica en el manual de las Xbee Pro S1. Esto se puede deber a múltiples motivos como la presencia de obstáculos entre los dos módulos, interferencias electromagnéticas, etc. Sin embargo, los resultados son bastante satisfactorios, pues el máximo de 700 metros posiblemente se pueda conseguir con unas condiciones meteorológicas perfectas, usando antenas de mayor ganancia y posicionadas a mayor altura.

Sí que conviene destacar el hecho de que es muy importante proporcionar una buena alimentación a los módulos de radio para que estos puedan suministrar la potencia necesaria y se logre un mayor alcance. El ensayo sin alimentación buena se hizo alimentando los módulos de radio mediante un adaptador USB mientras que el ensayo con buena alimentación se realizó alimentando las Xbee con una batería y un regulador de tensión.

6.1.1.1. Implementación de los módulos de radio en la plataforma

Por otro lado, la siguiente decisión a tomar después de haber elegido los módulos Xbee Pro S1 y haber verificado su desempeño, es su disposición en el quadrotor. Las opciones son conectarlo por serie al Arduino Due o conectarlo por serie a la Igep.

La solución tomada ha sido conectarlo a la Igep porque se puede programar en ella un hilo de ejecución dedicado especialmente a tratar con la telemetría. Además, utilizar el Arduino Due para conectar las Xbee supondría aumentar enormemente los tiempos de ciclo lo cual no resulta admisible.

Para ello, se ha diseñado una PCB de prototipado y se ha instalado en el nivel más alto de la electrónica del quadrotor. Esta PCB se ha diseñado junto al estudiante Alberto Castillo e incluye la Xbee, el Arduino Micro, el receptor RC y el sensor GPS. En la Ilustración 36 se muestra su aspecto.

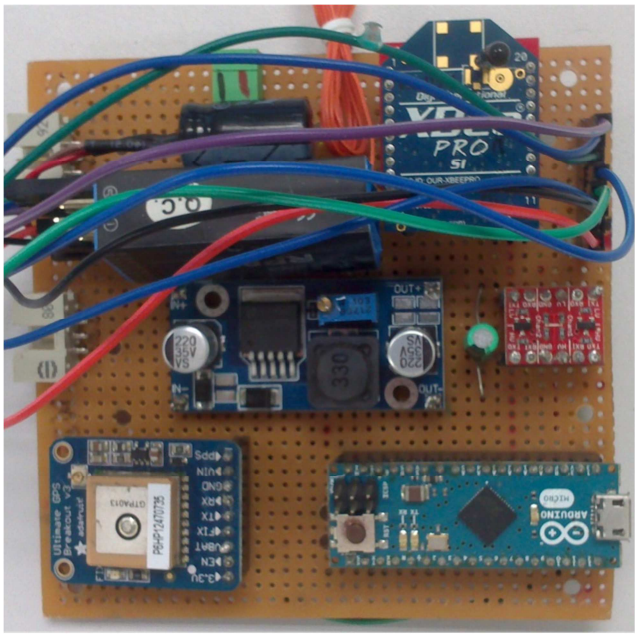

Ilustración 36. PCB diseñada para instalar la Xbee en la plataforma.

Esta placa se alimenta directamente desde la batería instalada en el quadrotor. Sin embargo, la Xbee requiere un voltaje de alimentación de 3,3 V. Es por ello que se ha instalado un regulador de tensión a este efecto. De esta forma se evita alimentar el módulo de radio desde otro dispositivo como un Arduino ya que se corre el riesgo de que éste no pueda proporcionar la corriente necesaria y por lo tanto el alcance de la Xbee se vea reducido.

Otro aspecto a tener en cuenta es el voltaje de las comunicaciones. La Xbee Pro S1 se comunica por protocolo serie TTL a 3,3 voltios, no obstante, la Igep v2 se comunica con unos voltajes de 0 a

1,8 V y puede resultar dañada si el voltaje aplicado es mayor. Para solucionar esto, se utilizan unos microchips, que se encuentran soldados en la PCB del Arduino Due, que permite convertir entre estos niveles de tensión.

6.1.1.1. Implementación de los módulos de radio en la UCT

Para conectar las Xbee al ordenador en la estación terrestre se ha utilizado un adaptador FTDI, el cual se muestra en la Ilustración 37. Este adaptador permite convertir los niveles de tensión de 3,3 V utilizados por las Xbee a los correspondientes para el puerto USB de un ordenador.

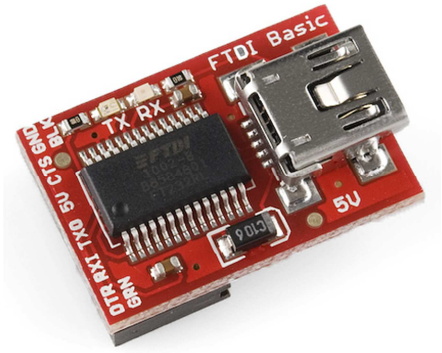

Ilustración 37. Adaptador FTDI.

Como se ha comentado en el punto anterior, resulta muy importante alimentar los módulos de radiofrecuencia correctamente. Por esta razón, en la UCT se utiliza una batería junto con un regulador de tensión para alimentar la Xbee.

En la Ilustración 38 se puede ver la solución a nivel de hardware implementada en la unidad de control terrestre.

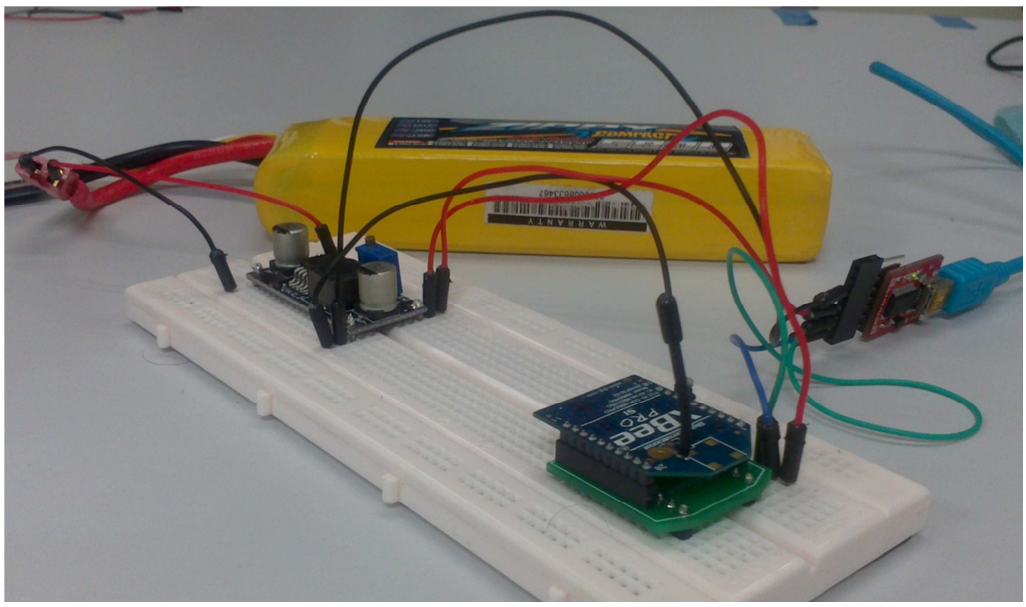

Ilustración 38. Hardware de la unidad de control terrestre.

6.1.2. Software

6.1.2.1. Configuración de los módulos de radio

Los dos módulos de radiofrecuencia se han configurado para funcionar en modo transparente. Esto significa que la xbee funciona como una conexión serial normal, es decir, toda la información que recibe la xbee por el pin RX se pone en cola para ser mandada por radio y cuando llega al módulo de destino, éste la transmite al exterior por el pin TX.

Además, se ha creado una red punto a punto entre los dos módulos de radiofrecuencia. Se trata de la conexión ideal para reemplazar comunicación serial por cable. Sólo se debe configurar la dirección. Para ello se utilizan los comandos MY y DL. La idea, es que se define arbitrariamente una dirección para un módulo, usando el comando MY, el cual se va a comunicar con otro que tiene la dirección DL, también definida arbitrariamente. Con esto cada módulo define su dirección con MY, y escribe la dirección del módulo al cual se desea conectar usando DL. Se muestra un ejemplo en la Ilustración 39.

En este modo, el módulo receptor del mensaje envía un paquete al módulo de origen llamado ACK (Acknowledge) que indica que el mensaje se recibió correctamente.

Ilustración 39. Ejemplo de conexión punto a punto.

Todo esto es fácilmente configurable mediante el software XCTU que proporciona la empresa Digi.

6.1.2.2. Protocolo MAVLink

En cuanto a la programación del software que gestiona la telemetría en el ordenador de a bordo de la plataforma de vuelo se ha optado por utilizar un protocolo para la transmisión de datos ya existente en vez de programar uno propio.

El protocolo utilizado ha sido MAVLink ("Micro air aerial vehicle communication protocol"). MAVLink no es más que un conjunto de librerías en lenguaje C que empaqueta una serie de estructuras en C para mandarlas a la unidad de control terrestre (Ilustración 40). La estructura de las tramas de bytes que genera MAVLink se muestran en la Ilustración 41.

El uso de este protocolo se encuentra bastante extendido entre UAVs de código libre y UAVs comerciales. Algunos de los autopilotos que utilizan el protocolo MAVLink son los siguientes:

- ArduPilotMega.
- pxIMU Autopilot.
- SLUGS Autopilot.

- FLEXIPILOT.
- UAVDevBoard/Gentlenav/MatrixPilot.
- SenseSoar Autopilot.
- SmartAP Autopilot.
- AutoQuad 6 AutoPilot.

Por otro lado, el protocolo MAVLink también se utiliza en las estaciones terrestre al igual que en los autopilotos. Entre los softwares que utilizan el protocolo MAVLink se pueden destacar:

- QGroundControl (Windows/Mac/Linux).
- HK Ground Control Station (Windows).
- APM Planner (Windows/Mac).
- QGroundControl w/ AutoQuad MainWidget (Windows/Mac/Linux).
- Copter GCS (Android).
- AutoQuad GCS (Android).

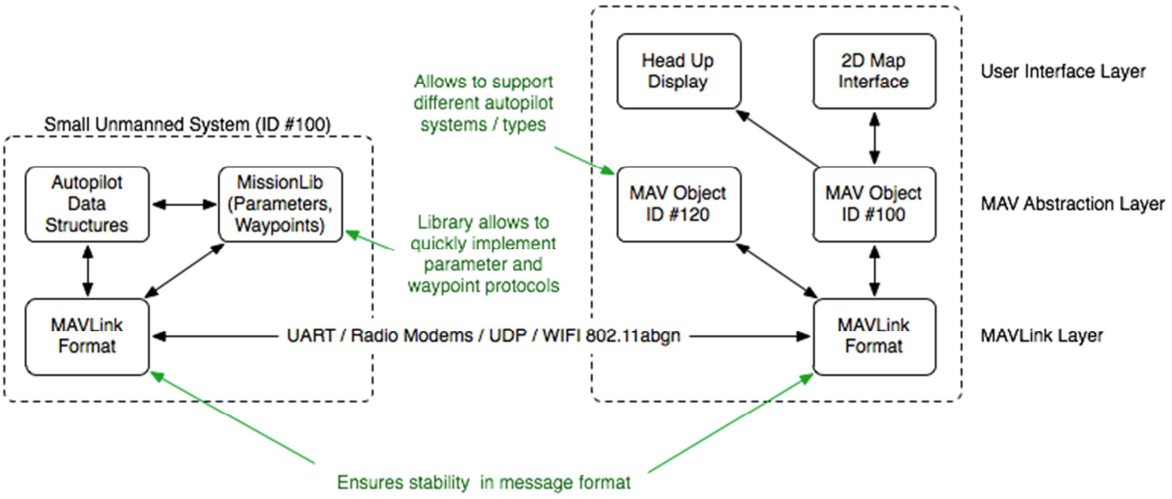

Ilustración 40. Funcionamiento del protocolo MAVLink

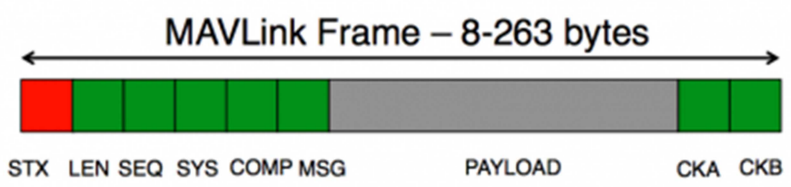

Ilustración 41. Trama de bytes que empaqueta MAVLink.

La ventaja de utilizar este protocolo es que se dota a la plataforma de mayor flexibilidad ya que al utilizar un protocolo que es usado por otros programas, es posible utilizar tanto la interfaz gráfica (HMI) desarrollada como software de terceros para monitorizar y controlar el quadrotor.

Así pues, desde el quadrotor a la UCT se mandan los siguientes mensajes de forma alternada a una frecuencia de 5 Hz:

- HEARTBEAT (#0): Manda información sobre el modo y el estado de vuelo.

- ATTITUDE (#30): Envía los valores de los ángulos de Euler y velocidades angulares.
- GLOBAL_POSITION_INT (#33): Transmite información sobre las coordendas GPS del vehículo.

Por otro lado, desde la UCT a la plataforma de vuelo se mandan los siguientes mensajes cuando el usuario decide actuar sobre el quadrotor.

- SET_MODE (#11): Para cambiar el estado de vuelo.
- SET_ROLL_PITCH_YAW_THRUST (#56): Se envía para cambiar las referencias sobre los ángulos de Euler del vehículo y poder controlarlo.

También se ha implementado todo el protocolo que utiliza MAVLink para enviar y recibir Waypoints aunque esta función sólo está realizada a nivel de telemetría, se espera implementarla en la plataforma en un futuro.

Además de todo esto, MAVLink cuenta con herramientas para implementar mensajes propios. Éstas se han utilizado, por ejemplo, para implementar un mensaje que permita cambiar los parámetros de los controladores PID.

Por último, en la Ilustración 42 se muestra una prueba de telemetría que se realizó en exteriores. Se ha utilizado el software QGroundControl antes mencionado para verificar la compatibilidad del protocolo MAVLink implementado en el quadrotor. La plataforma envía las coordenadas GPS de su localización y el protocolo MAVLink se encarga de empaquetar este mensaje. Por otro lado, en la unidad de control terrestre, la aplicación QGroundControl también utiliza el protocolo MAVLink, por lo tanto puede decodificar el mensaje recibido y representar las coordenadas GPS sobre un mapa de Google Earth.

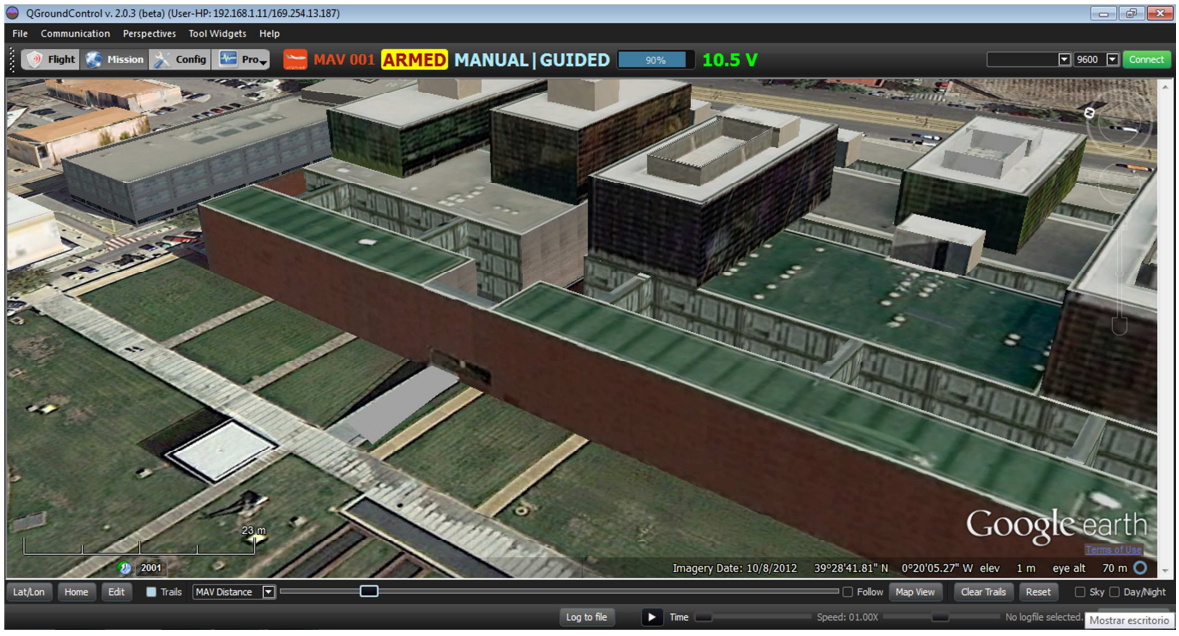

Ilustración 42. Prueba de compatibilidad de la telemetría con el software QGroundControl.

6.2. Interfaz Gráfica Hombre Máquina (HMI)

Para el desarrollo de la interfaz gráfica de usuario (HMI) la opción elegida ha sido Qt. Primero se ha descartado Netbeans porque está orientado a la programación en lenguaje Java, el cual no se ha utilizado a lo largo de este proyecto. A continuación se ha escogido Qt sobre LabView porque es multiplataforma. Además Qt permite mayor flexibilidad a la hora de integrar librerías externas y el código está más estructurado, por lo que resulta más fácil de entender para futuros desarrolladores.

El IDE utilizado ha sido Qt Creator, pues es una de las mayores ventajas de Qt. Qt Creator es un Entorno Integrado de Desarrollo o IDE (editor + compilador + depurador) bastante completo, moderno, potente, fácil de manejar, eficiente, abierto y gratuito, que permite el desarrollo rápido de aplicaciones en entornos MS Windows, Mac OS y Linux. Algunos ejemplos de programas creados con las librerías Qt son: Adobe Photoshop Album, Google Earth, KDE, Opera, Skype, VLC media player.

Las características que han llevado a la elección de Qt Creator son:

- Utiliza el lenguaje de programación orientado a objetos C++.
- Se basa en Qt, una librería multiplataforma y gratuita para la creación de interfaces gráficas, programación web, multihilo, bases de datos, etc.
- Programación visual: el programador centra su atención en diseñar el aspecto gráfico de la aplicación, la distribución de los elementos visuales (llamados widgets: formularios, botones, menús, cuadros de texto, etc.), la interacción entre los mismos, los distintos tipos de ventanas existentes, etc.
- Programación dirigida por eventos: el programador escribe el código que se ejecutará en respuesta a determinados eventos (llamados slots: pulsar un botón, elegir una opción del menú, abrir o cerrar una ventana, etc.). No existe la idea de un control de flujo secuencial en el programa, sino que el programador toma el control cuando se dispara un evento.

Los detalles del programa desarrollado, así como la estrategia de programación seguida para incorporar toda la funcionalidad deseada al HMI se describe tanto en el manual de usuario como en el manual de programación del HMI.

En la Ilustración 43 se muestra el aspecto final del HMI sobre Linux. La captura de pantalla ha sido tomada haciendo pruebas con la plataforma de vuelo en las terrazas de la CPI de la Universidad Politécnica de Valencia.

6.3. Enlace del Quadrotor con una Emisora de Radiocontrol

En cuanto a la emisora y receptor de radiocontrol, como se ha comentado en al capítulo anterior, no se ha hecho ninguna elección puesto que se ha utilizado el material con el que contaba el grupo de investigación de Pedro García Gil.

Sí que ha sido necesario decidir donde conectar el receptor de radiocontrol en el quadrotor. Para leer las acciones enviadas desde la emisora, se ha de tener un dispositivo que posea 6 entradas para leer 6 señales PWM.

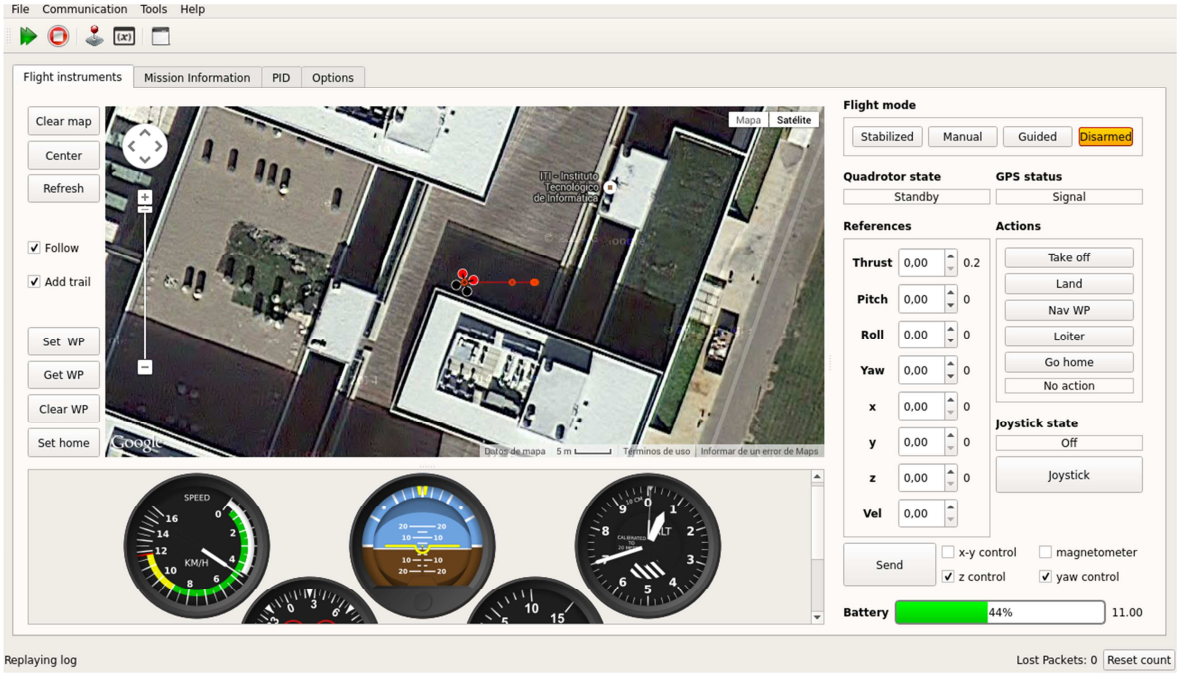

Ilustración 43. Aspecto final del HMI.

La PCB del Arduino se encontraba cerrada en el momento de incorporar este sistema, mientras que la Igep no dispone de las entradas necesarias para comunicarse con el receptor. Es por ello que se ha utilizado el Arduino Micro y se ha instalado todo esto en el último nivel de la electrónica del quadrotor, tal como se muestra en la Ilustración 36.

El Arduino Micro utiliza la función “pulseln” para leer las señales PWM. En concreto, los canales del receptor de radiocontrol se conectan a 6 pines digitales del Arduino Micro y la función “pulseln” mide el tiempo que la señal PWM se encuentra en estado alto o bajo.

El procedimiento es el siguiente:

1. El Arduino Micro lee el canal del receptor correspondiente al interruptor automático/manual.
2. Si el interruptor se encuentra en modo manual, entonces el Arduino Micro lee los otros 5 canales.
3. El Arduino lee un valor de tiempo entre 1000 ms y 2000 ms y este valor se escala. Por ejemplo, en el caso del pitch 1000 ms es una referencia de -10º y 2000 ms una referencia de +10º.
4. Por último se envía esta información al Arduino Due que es el que se encarga de realizar el control modificando las referencias sobre roll, pitch y yaw e ignorando la información que recibe de la Igep.

Cabe destacar la utilidad que tiene el sexto canal introducido. Durante el modo de funcionamiento normal manual, la altura del quadrotor sólo se controla con el acelerador o gas, de forma que resulta difícil mantener el helicóptero a una altura estable. Cuando se activa el sexto canal (control de altura), el quadrotor guarda la altura anterior y la mantiene.

También se ha implementado un mecanismo para que en caso de pérdida de la señal de la emisora con el quadrotor, éste aterrice de forma segura. La secuencia para apagar la emisora es cambiar de modo manual a automático y finalmente desconectarla, de modo que si se pierde la señal con la emisora, el Arduino Micro puede detectarlo y enviarle un mensaje al Arduino Due. El Arduino Due es el que se encarga de ejecutar un algoritmo que va disminuyendo la referencia de altura poco a poco hasta para que el quadrotor aterrice de forma segura.

7. CONCLUSIONES Y TRABAJOS FUTUROS

Al comienzo de la presente memoria se expusieron una serie de objetivos que se debían cumplir tras la ejecución de este proyecto. Tras exponer detenidamente los pasos que se han seguido en su desarrollo, se va a analizar el grado de cumplimiento de estos objetivos.

Plataforma de vuelo:

- *Desarrollar una plataforma de vuelo experimental. La plataforma ha de ser robusta y fiable a la vez que flexible y modular.*
- *Implementar el hardware y los sensores necesarios para realizar vuelos tanto en interiores como al aire libre.*
- *Desarrollar el software necesario para la gestión del vehículo y la comunicación con los distintos equipos electrónicos.*
- *Conseguir vuelo estable en actitud tanto en interiores como en exteriores.*

El principal logro del proyecto ha sido el desarrollo de una plataforma de vuelo quadrotor desde cero y, tras superar uno por uno todos los problemas y contratiempos encontrados, se ha conseguido que funcione de forma robusta. Asimismo se deja a disposición de futuros investigadores una plataforma modular para probar distintos equipos electrónicos, algoritmos de control, etc.

Además se han implementado todos los sensores necesarios tanto para volar en interiores como en exteriores con éxito y se ha desarrollado el software que permite tratarlos y comunicarlos con la unidad de control terrestre.

También se ha conseguido realizar un vuelo estable en actitud en interiores. En exteriores ha sido más complicada la tarea porque debido a la protección que tiene el quadrotor resulta más difícil de controlar si hace algo de viento.

Sistemas de telemetría:

- *Implementar un sistema de telemetría fiable entre el quadrotor y un ordenador en tierra. Además, el enlace ha de ser bidireccional, para monitorizar el "drone" en tiempo real y al mismo tiempo poder controlarlo.*
- *Desarrollar una interfaz gráfica de usuario para el control del quadrotor desde la unidad de control terrestre.*
- *Disponer de un mecanismo de seguridad para retomar el control del quadrotor en caso de fallo del ordenador principal.*

Se ha logrado implementar un enlace robusto y fiable entre la plataforma de vuelo y la unidad de control terrestre (UCT) mediante dos módulos de radiofrecuencia Xbee Pro S1. El enlace implementado es además bidireccional de forma que cumple con los objetivos de poder monitorizar el quadrotor y poder controlarlo.

También se ha implementado con éxito otro enlace para controlar el quadrotor por medio de una emisora y un receptor de radiocontrol. Este sistema cumple con la función deseada, pues ante un fallo en el ordenador de a bordo de la plataforma o del sistema de telemetría principal, es posible recuperar el control del vehículo, ya que este mecanismo funciona de forma independiente al ordenador de a bordo (Igep).

Interfaz hombre máquina (HMI):

El software de la estación terrestre debe contar con las siguientes características:

- *Interfaz gráfica de ususario para visualizar de forma rápida el estado del vehículo con indicadores en forma de instrumentos de vuelo y mapa para posicionar en todo momento el quadrotor.*
- *Opciones para actuar sobre el vehículo cambiando su modo de vuelo, controlándolo con un joystick o modificando los parámetros de los controladores.*
- *Facilidades para la ejecución del programa en diferentes sistemas operativos.*
- *Opciones para guardar datos de vuelo y reproducirlos posteriormente, exportar variables a MATLAB o incluso representarlas gráficamente en tiempo real.*

Éste ha sido uno de los logros principales del proyecto. Se ha desarrollado una aplicación para controlar y monitorizar el quadrotor desde cero en Qt con todas las funcionalidades que se habían pensado en un primer momento.

Al empezar el trabajo, no se había utilizado un lenguaje orientado a objetos y en unos meses se ha aprendido a programar en lenguaje C++ y a utilizar el framework de Qt.

El software desarrollado es una aplicación de escritorio multiplataforma en compilación. Posee una interfaz gráfica con instrumentos de vuelo y un mapa. Además cuenta con paneles que proporcionan información sobre el estado de la batería, señal GPS, etc.

Asimismo es posible actuar sobre el vehículo modificando su estado de vuelo, cambiando las referencias de sus ángulos de Euler o incluso controlándolo con un Joystick. Además también cuenta con una sección muy útil para configurar los parámetros de los controladores PID durante un vuelo y funciones adicionales de estadísticas, guardado de logs, representaciones gráficas, etc.

Por último, también es posible guardar todos los datos de vuelo de una misión para repetir el vuelo posteriormente y analizarlo con detalle.

7.1. Trabajos Futuros

A continuación se enumeran y describen de forma breve diversas tareas que se proponen como trabajos futuros a realizar. Todas están relacionadas directamente con mejorar el funcionamiento de la plataforma desarrollada.

Mejorar la protección del quadrotor:

Aunque la protección diseñada ha resultado muy útil para ajustar las leyes de control y evitar accidentes, si se desean realizar vuelos en exteriores es necesario quitar la protección o diseñar una nueva con otro perfil al que no le afecte tanto el viento.

Diseñar PCB para la capa superior de la electrónica del quadrotor:

La última capa de la electrónica de la plataforma de vuelo se ha realizado sobre una placa de prototipado de baquelita. Se ha validado el buen funcionamiento de esta placa, por lo que en un futuro habría que diseñar una PCB final para reducir el número de soldaduras al mínimo.

Mejorar el alcance del enlace de telemetría entre el quadrotor y la UCT:

En la plataforma desarrollada se tiene un alcance fiable a una distancia no mayor a 150 metros. Si en el futuro se pretende hacer más vuelos en exteriores sería conveniente aumentar el alcance del enlace.

Se propone en un principio probar alguna antena de mayor calidad. También se podría ver la posibilidad de instalar una antena direccional de mayor ganancia en la unidad de control terrestre. Si esto no es suficiente, habría que plantearse utilizar una solución diferente a las Xbee.

Sustituir el Arduino Micro:

Sería conveniente comunicar el receptor de radiocontrol con el Arduino Due en vez del Arduino Micro con objeto de eliminar éste de la plataforma. De este modo, las referencias de la emisora llegarían directamente al microcontrolador donde se realiza el control de estabilidad y se podría utilizar una estrategia mediante interrupciones para leer las señales PWM.

Integración de todos los equipos en una misma placa:

Como se ha comentado en el capítulo de descripción de la plataforma, el objetivo no es tener una plataforma que sea un producto final. No obstante, una vez se validen todos los sistemas y no se quiera desarrollar nada nuevo, el siguiente paso sería integrarlo todo en una placa con el objetivo de reducir peso.

8. BIBLIOGRAFÍA Y REFERENCIAS

Adafruit. (s.f.). *Adafruit*. Obtenido de http://www.adafruit.com

AWG. (s.f.). *PowerStream*. Obtenido de http://www.powerstream.com/Wire_Size.htm

Castillo Frasquet, A. (2014). *Desarrollo integral de un quadrotor: Diseño de un algoritmo de control para la posición x-y basado en señales GPS.*

Castillo, P., Lozano, R., & Dzul, A. (2005). *Modelling and Control of Mini-Flying Machines.* Springer-Verlag in Advances in Industrial Control.

Corona. (s.f.). *Corona RC*. Obtenido de http://www.corona-rc.com/

Digi. (s.f.). *Digi*. Obtenido de http://www.digi.com/xbee/

HORIZONTE2020. (s.f.). *European Comission*. Obtenido de http://ec.europa.eu/programmes

Invensense. (s.f.). *Invensense*. Obtenido de http://www.invensense.com/

ISEE. (s.f.). *ISEE*. Obtenido de https://www.isee.biz/

MAVLink. (s.f.). *QGroundControl*. Obtenido de http://qgroundcontrol.org/mavlink/start

Mikrokopter. (s.f.). *Mikrokopter*. Obtenido de http://www.mikrokopter.es/ufocam-xxl8/

Parallax. (s.f.). *Parallax*. Obtenido de http://www.parallax.com/

Robbe. (s.f.). *Robbe*. Obtenido de http://www.robbe.de/

Ródenas Lorda, L. (2013). *Plataforma de desarrollo para el control de estabilidad en tiempo real de un vehículo aéreo tipo quadrotor.*

Sanahuja, G., Castillo, P., & Sánchez, A. (2009). Stabilization of n integrators in cascade with bounded input with experimental application to a VTOL laboratory system. *International Journal of Robust and Nonlinear Control*.

Verdú Torres, D. (2014). *Desarrollo integral de un quadrotor: Control de la orientación basado en una IMU de bajo coste y control de altura mediante un sensor barométrico.*

Xenomai. (s.f.). *Xenomai*. Obtenido de http://www.xenomai.org/

YGE. (s.f.). *YGE*. Obtenido de http://www.yge.de/index.php

UNIVERSITAT POLITÈCNICA DE VALÈNCIA

ESCUELA TÉCNICA SUPERIOR INGENIEROS INDUSTRIALES VALENCIA

TRABAJO FIN DE GRADO EN INGENIERÍA EN TECNOLOGÍAS INDUSTRIALES

MANUAL DE USUARIO DEL HMI

AUTOR: PABLO ALBIOL GRAULLERA

TUTOR: PEDRO JOSÉ GARCÍA GIL

COTUTOR:

Curso Académico: 2013-14

ÍNDICE

ÍNDICE DE ILUSTRACIONES

1. INTRODUCCIÓN

Este manual se ha realizado con el fin de instruir al usuario en el manejo del programa HMI ("Human Machine Interface"). Este software se ha desarrollado con el objetivo de controlar y monitorizar las variables más importantes de un vehículo aéreo remotamente pilotado (RPAS).

El programa desarrollado es una aplicación de escritorio en Qt. Esto es, cuenta con todos los elementos de una aplicación típica de ordenador; sistema de menús, barra de herramientas, barra de estado, etc. Además, la aplicación es multiplataforma en el sentido de que existen versiones diferentes del programa en función del sistema operativo que el usuario utilice. En el manual de programación del HMI se detalla cómo realizar una compilación con éxito bajo cualquier otro sistema operativo.

Como se ha comentado a lo largo de la memoria del presente proyecto, esta interfaz se ha desarrollado para la plataforma de vuelo construida en este trabajo. Sin embargo, el protocolo utilizado para la recepción de las distintas tramas de bytes ha sido el protocolo MAVLink, de modo que cualquier vehículo que implemente el mismo protocolo para enviar y recibir mensajes es compatible con este programa.

El programa se puede utilizar directamente con el quadrotor desarrollado siguiendo estas instrucciones de forma sencilla. Si se desea utilizar esta interfaz con otro UAV distinto al desarrollado en este proyecto, el usuario final es el responsable de gestionar los mensajes de a bordo del UAV para que se respete el protocolo MAVLink.

Hasta el momento, la aplicación funciona para dispositivos que se comuniquen por medio del puerto serie (o en su defecto el puerto USB) de un PC. Este es el caso más típico, sin embargo, se espera en un futuro que la aplicación también sea compatible con comunicaciones por sockets mediante los protocolos TCP/IP y UDP.

El manual que sigue se va a dividir en varios apartados según las funciones que el software incorpora, con la intención de que el personal a cargo de la monitorización y control, pueda utilizar la aplicación sin ningún conocimiento previo.

A continuación, en la Ilustración 1, se muestra la pantalla principal que aparece al ejecutar el programa.

Como se puede ver en la ilustración, se pueden distinguir varios bloques que a priori realizan tareas diferentes. Este manual se divide en función de estas secciones. La aplicación cuenta con cuatro pestañas principales: Flight Instruments, Mission Information, PID y Options. Cada uno de estos apartados se subdividirá en subapartados para explicar las funciones específicas de cada elemento del programa.

Por último, el programa se entrega bajo licencia La Licencia Pública General de GNU (GNU GPL), esto es, el software cubierto por esta licencia es software libre y garantiza a los usuarios finales (personas, organizaciones, compañías) la libertad de usar, estudiar, compartir (copiar) y modificar el código.

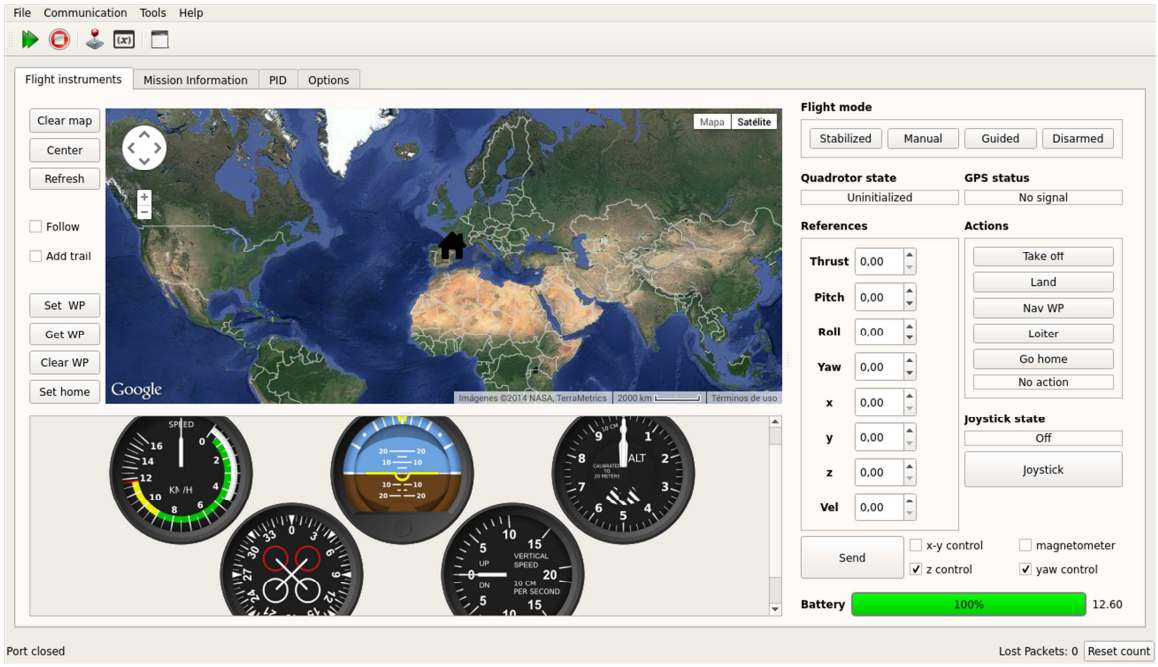

Ilustración 1. Ventana principal de la interfaz HMI.

1.1. Requisitos Mínimos del HMI

- Procesador de 32 bits (x86) o 64 bits (x64) a 1 gigahercio (GHz) o más.
- Memoria RAM de 512 MB (32 bits) o memoria RAM de 1 GB (64 bits).
- Espacio disponible en disco rígido 200 MB.
- Dispositivo gráfico DirectX 9 con controlador WDDM 1.0 o superior.
- Navegador web (Internet explorer, Mozilla Firefox, Google Chrome o Safari).
- OpenSSL 0.9.7 o una versión posterior.

Además, es recomendable contar con conexión a internet si se desea visualizar el mapa de vuelo de Google Maps. Si no se tiene conexión a internet en el momento de volar es recomendable guardar en caché la zona del mapa donde se vaya a volar anteriormente al vuelo.

Por otra parte, el programa se entrega comprimido en formato *.zip. No es necesario instalar ninguna librería ni biblioteca de terceros para el uso del programa. Únicamente es necesario descomprimir la carpeta de la aplicación, dentro de ella ya se encuentra todo incluido.

1.2. Plataformas Compatibles

El programa puede ser ejecutado bajo los siguientes sistemas operativos:

- Ubuntu Linux 10.04 (32-bit)
- Ubuntu Linux 10.04 QWS (x86 32-bit)
- Ubuntu Linux 10.04 (64-bit)
- Ubuntu Linux 11.10 (64-bit)
- Microsoft Windows XP SP3 (32-bit)
- Microsoft Windows 7 (32-bit)
- Microsoft Windows 7 (64-bit)

- Apple Mac OS X 10.6 "Snow Leopard" (64-bit)
- Apple Mac OS X 10.7 "Lion" (64-bit)
- Apple Mac OS X 10.6 "Snow Leopard" Cocoa (32-bit)

2. INICIAR LA MONITORIZACIÓN

Al iniciar el HMI, por defecto, el estado de vuelo es no inicializado, todos los instrumentos se encuentran a cero, el mapa no hace zoom sobre ninguna zona en particular y no aparece ningún vehículo sobre él. Además, en la barra de estado se puede leer “Port closed”, que significa que aún no se ha enlazado el HMI al vehículo.

Para conectar el HMI y que comience la monitorización del vehículo existen dos caminos. Se puede utilizar la barra de herramientas (Ilustración 2) o el menú “Communication” (Ilustración 3).

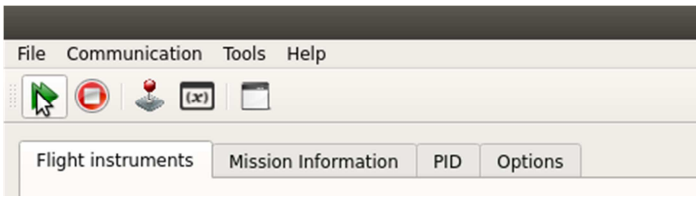

Ilustración 2. Conexión del HMI mediante la barra de herramientas.

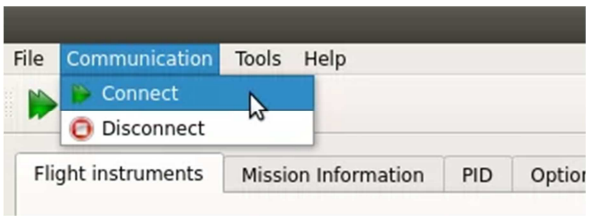

Ilustración 3. Conexión del HMI por medio del menú Communication.

Una vez pulsado el botón “Connect” la interfaz muestra el siguiente cuadro de diálogo (Ilustración 4):

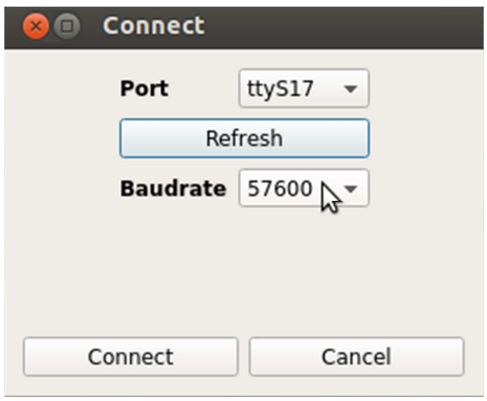

Ilustración 4. Cuadro de diálogo de conexión del HMI.

Hay que seleccionar el puerto serie al que se conecta el equipo que se comunica con el vehículo y la tasa de baudios.

Al pulsar el botón conectar anterior, el programa detecta los dispositivos que hay conectados a los distintos puertos series del ordenador. En caso de que no se haya detectado o no se haya introducido el dispositivo en el puerto serie del PC todavía, se puede utilizar el botón “Refresh” para volver a detectar los dispositivos conectados.

Una vez está todo en orden, se vuelve a pulsar el botón “Connect” y si no hay problemas, el HMI abrirá el puerto serie y en la barra de estado se podrá leer el mensaje “Port openned”. El HMI empezará a recibir mensajes y actualizarlos en los instrumentos de la ventana principal. A partir de este momento, ya se pueden hacer uso de todas las funciones con las que cuenta el HMI.

Si el puerto se abre correctamente pero, por algún motivo inherente al HMI, no llegan mensajes, el HMI mostrará un mensaje de advertencia.

En caso de que se intente abrir el puerto serie, pero por alguna otra razón, no sea posible (por ejemplo, que esté siendo usado por otro programa), el HMI mostrará un mensaje de advertencia (Ilustración 5).

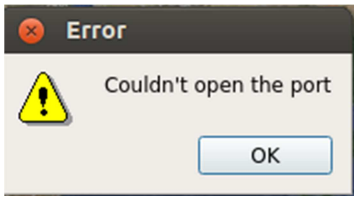

Ilustración 5. Error abriendo el puerto.

Por último, para finalizar la monitorización del vuelo hay que presionar el botón “Disconnect” o simplemente cerrar el programa.

3. INSTRUMENTOS DE VUELO

Esta pestaña es la que se muestra por defecto al abrir el programa (Ilustración 1). El objetivo de esta pantalla es la visualización del estado completo del vehículo aéreo de forma rápida. A grandes rasgos esta sección cuenta con un mapa de Google Maps, una serie de instrumentos de vuelo, un panel para comunicarse con el vehículo y otras herramientas.

Al conectar el HMI al vehículo en cuestión, se empiezan a recibir paquetes de bytes que van actualizando los distintos elementos de esta pantalla.

3.1. Gestión del Mapa

La función del mapa es mostrar la posición absoluta del RPAS sobre un mapa de Google Maps. Por defecto, al iniciar la interfaz, el mapa no está orientado hacia ningún punto en particular. Una vez se reciben las primeras coordenadas válidas, entonces se actualiza el mapa. Después, el mapa se actualiza cada vez que se recibe un mensaje de MAVLink del tipo GLOBAL_POSITION_INT (#33). En la Ilustración 6 se muestra su aspecto.

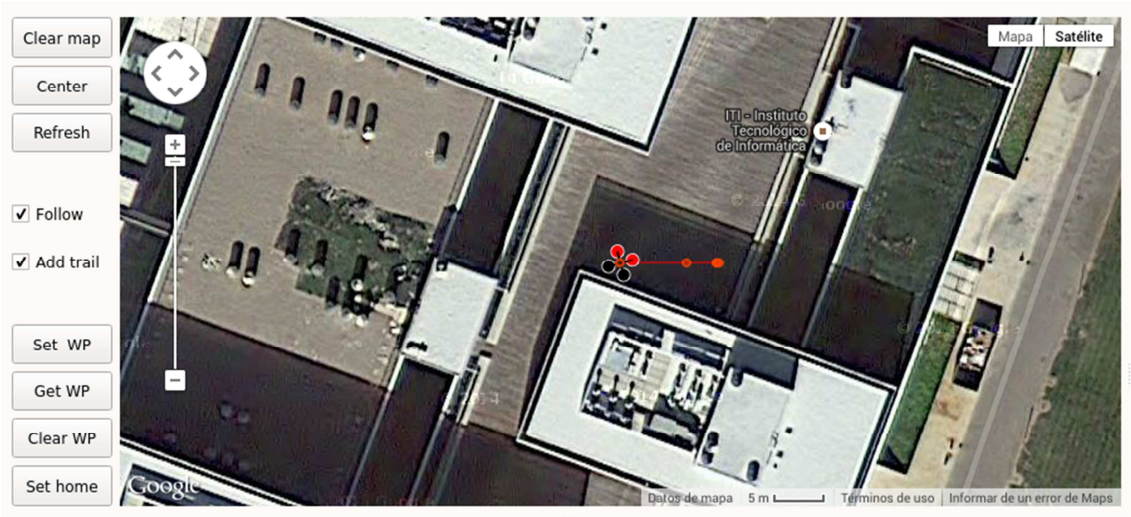

Ilustración 6. Mapa de la aplicación HMI.

Además de mostrar la posición del vehículo, el mapa también muestra la orientación del mismo, pues en el icono que se representa sobre el plano se puede distinguir una parte trasera y otra delantera. La orientación del vehículo se actualiza con un mensaje de MAVLink del tipo ATTITUDE (#30) (campo yaw). También se puede ver la altura del vehículo sobre el mapa, haciendo click sobre el icono del vehículo. Esto muestra un diálogo sobre el vehículo con la altura actual.

El mapa está incrustado de Google Maps, por lo tanto se puede utilizar del mismo modo que se utilizan los mapas de Google en internet. Es decir, se puede interactuar directamente con él haciendo zoom con el "scroll" del ratón, se puede mover manteniendo pulsado el botón izquierdo del ratón, etc.

En la parte izquierda, hay herramientas adicionales para interactuar con el mapa. Sus funciones son las siguientes:

- Clear map: Borra el contenido el mapa.
- Center: Centra el mapa en la posición del vehículo y aplica un zoom sobre él.
- Refresh: Actualiza el mapa en caso de que no se haya cargado bien.
- Follow: Si se marca esta opción, el mapa se desplaza al mismo tiempo que el vehículo para que quede en todo momento centrado en el mapa.
- Add trail: Dibuja una línea del recorrido del vehículo.

Las funciones que restan (Set WP, Get WP, Clear WP, Set home) se explicarán con detalle en el apartado de gestión de "waypoints".

3.2. Instrumentos

Los instrumentos (Ilustración 7) muestran información de forma gráfica sobre las variables más importantes a monitorizar en un vehículo aéreo. Se actualizan conforme el HMI recibe un mensaje u otro del ordenador de a bordo del vehículo en cuestión.

Ilustración 7. Instrumentos de vuelo del HMI.

- Indicador de velocidad en el plano horizontal: Se actualiza con el mensaje de MAVLink GLOBAL_POSITION_INT (#33).
- Horizonte artificial: Se actualiza con el mensaje de MAVLink ATTITUDE (#30).
- Altímetro: Se actualiza con el mensaje de MAVLink GLOBAL_POSITION_INT (#33).
- Brújula: Se actualiza con el mensaje de MAVLink ATTITUDE (#30).
- Indicador de velocidad vertical: Se actualiza con el mensaje de MAVLink GLOBAL_POSITION_INT (#33).

3.3. Panel de vuelo

Este panel permite actuar sobre el vehículo así como cambiar el estado de vuelo del RPAS. Se muestra su aspecto en la Ilustración 8.

En la parte superior se muestra el modo de vuelo. En este programa existen cuatro modos de vuelo:

- Disarmed: El UAV se encuentra en tierra con los motores apagados.
- Stabilized: El vehículo se puede controlar mediante referencias introducidas en la interfaz o desde un joystick a través del HMI.
- Manual: El RPAS se controla mediante una emisora de radiocontrol.

- Guided: El ordenador de a bordo es el que decide las acciones de control a aplicar para seguir unas referencias previamente establecidas.

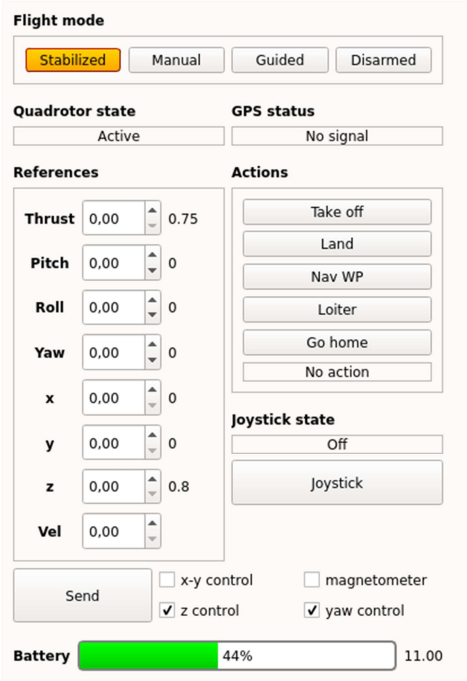

Ilustración 8. Panel de vuelo del HMI.

Para cambiar de modo de vuelo, simplemente hay que hacer click en el modo de vuelo deseado. El UAV tiene que contestar mediante un mensaje de MAVLink tipo HEARTBEAT (#0) para actualizar el modo de vuelo en la interfaz.

El siguiente elemento en el panel es un cuadro que muestra información sobre el estado de vuelo. Se actualiza con el mensaje HEARTBEAT (#0), con el campo system_status. A la derecha se muestra un cuadro con información del tipo de cobertura GPS. En este caso se actualiza con el campo custom_mode del mensaje HEARTBEAT (#0).

El bloque "References" permite enviar referencias de pitch, roll, yaw, x, y, z y velocidades al vehículo. Al mismo tiempo se puede decidir si estará activado el control de posición, el control de altura, el control de yaw y el magnetómetro mediante el envío de cuatro variables booleanas. Para enviar estos valores se hace click en el botón "send" y el quadrotor contestará para confirmar estos datos. Esta estructura de datos se ha implementado mediante un mensaje propio de MAVLink. Si el usuario final utiliza otro vehículo distinto al quadrotor desarrollado tendrá que implementar este mensaje.

El bloque "Actions" permite enviar acciones al vehículo. Se ha utilizado el protocolo MAVLink, en concreto el mensaje COMMAND_LONG (#76). El vehículo ha de confirmar el mensaje recibido con otro mensaje de tipo COMMAND_ACK (#77).

Por último, en la parte inferior se muestra un indicador de la batería del vehículo. Este widget se actualiza mediante un mensaje de MAVLink de tipo SYS_STATUS (#1).

3.4. Otras Herramientas

3.4.1. Control por joystick

La interfaz permite utilizar un joystick para controlar el vehículo. Para ello, hay que hacer click en el botón joystick del panel de vuelo o en el correspondiente botón de la barra de herramientas. A continuación se abrirá la siguiente ventana (Ilustración 9).

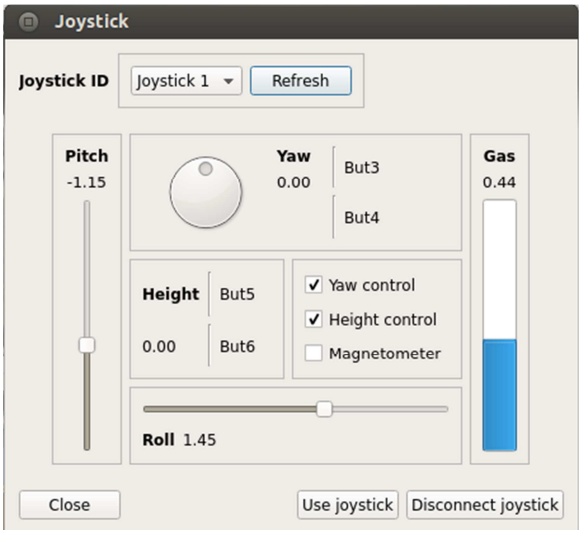

Ilustración 9. Herramienta para el uso de un joystick.

En primer lugar hay que seleccionar, el joystick que se va a utilizar del desplegable superior. Después ya se puede mover el joystick y probar los distintos botones para comprobar que el joystick responde en la interfaz. Una vez se ha probado el correcto funcionamiento del joystick, hay que pulsar el botón “Use joystick” para que el HMI empiece a enviar de forma seguida estos valores al vehículo que se desea controlar. Por último, para dejar de utilizar el joystick, se hará click en “Disconnect joystick”.

Cabe añadir, que como se explicará en el bloque de opciones, es posible configurar la sensibilidad del joystick y otros parámetros.

3.4.2. Visualización de variables de forma numérica

La interfaz también cuenta con una ventana muy sencilla que muestra las variables recibidas del vehículo en forma de texto. Se puede acceder a esta ventana desde la barra de herramientas mediante el botón “Show variables”.

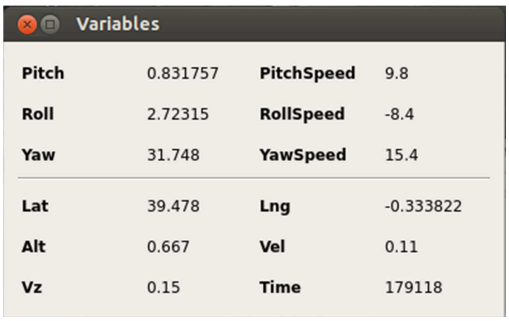

Ilustración 10. Variables en forma numérica.

4. INFORMACIÓN SOBRE EL VUELO

Esta pantalla muestra información variada sobre el vuelo del vehículo. Se distinguen tres bloques principales: un bloque de gestión de "waypoints", otro con un resumen de estadísticas del vuelo y una sección dedicada a exportar y representar gráficamente ciertas variables del RPAS.

Además estos bloques se pueden agrandar o hacer más pequeños según las necesidades del usuario. En la Ilustración 11 se muestra su aspecto.

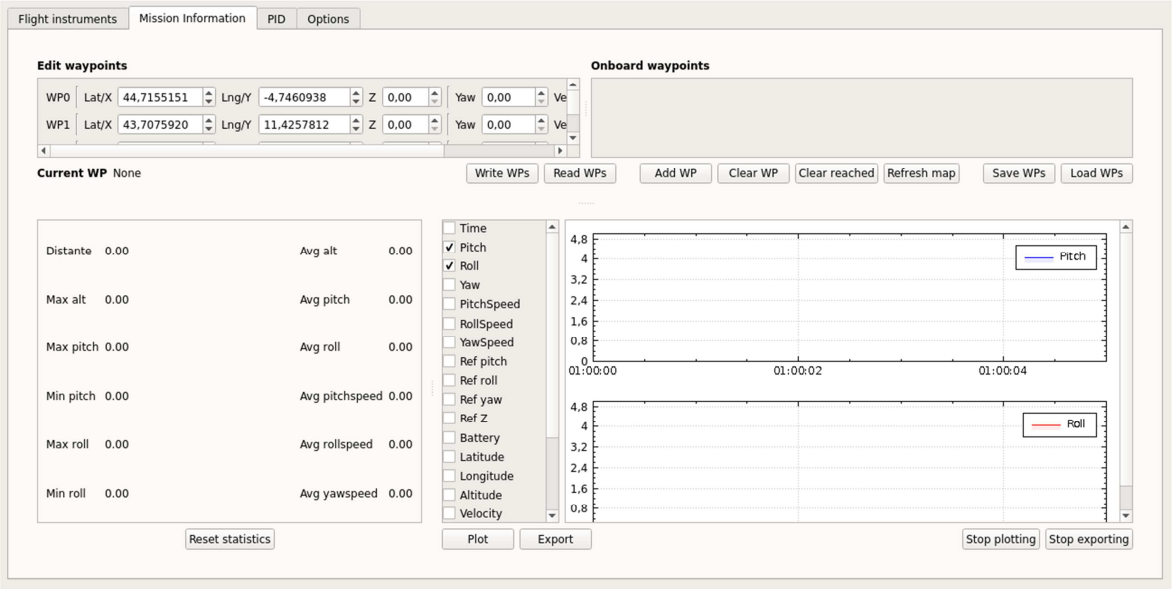

Ilustración 11. Pantalla Mission Information del HMI.

4.1. Gestión de Waypoints

Esta sección permite enviar, leer y borrar "waypoints" del vehículo. En la Ilustración 11 se puede ver un ejemplo.

El procedimiento para enviar "waypoints" al vehículo es el siguiente:

1. Se seleccionan los puntos deseados haciendo doble click sobre el mapa de Google Maps (Ilustración 12).
2. Una vez los puntos deseados son los correctos se pulsa el botón "Set WPs".
3. Se vuelve a la pantalla "Mission information", ahora el bloque de waypoints ya no se encuentra vacío. En este momento se puede editar la altura de cada punto, ángulo de yaw de entrada al WP, velocidad, etc.
4. Finalmente se pulsa el botón "Write WPs" que envía los "waypoints" individualmente al vehículo.
5. Si todo ha funcionado correctamente, los puntos enviados se recibirán de nuevo en la interfaz.

Una vez estos puntos están cargados en el vehículo de forma correcta, esta herramienta está preparada para recibir la ID del "waypoint" actual que sigue el vehículo, marcar los puntos alcanzados o cambiar de destino.

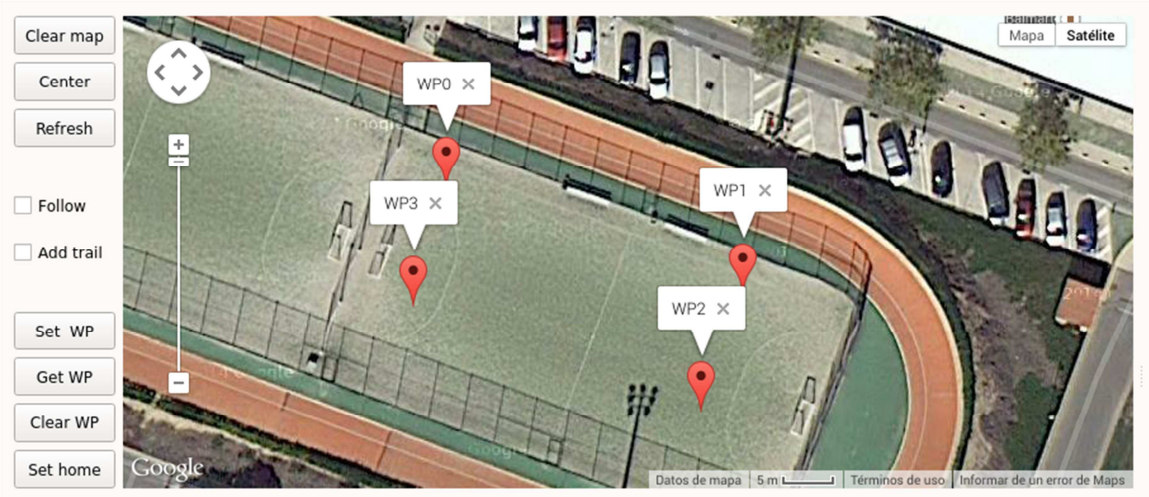

Ilustración 12. Selección de los waypoints.

Para leer los "waypoints" simplemente se hace click en "Read WPs" y el HMI se encarga de mostrarlos por pantalla y actualizarlos en el mapa. Por otro lado "Clear WPs" borra los "waypoints" a bordo del vehículo.

A todo esto cabe añadir que esta pantalla cuenta con opciones para guardar y cargar "waypoints" por medio de archivos de texto.

4.2. Estadísticas de vuelo

Este apartado muestra distintas estadísticas durante el vuelo de un vehículo (distancia recorrida, máxima altura, altura media, máximo ángulo de cabeceo, etc.). Se actualiza cada vez que se recibe un mensaje de MAVLink de tipo ATTITUDE (#30) o GLOBAL_POSITION_INT (#33). Su aspecto se muestra en la Ilustración 13.

Ilustración 13. Estadísticas de vuelo.

4.3. Gráficas

Esta sección permite tanto exportar las variables seleccionadas a archivos de texto como representarlas gráficamente en tiempo real. En la Ilustración 14 se puede ver el funcionamiento de este apartado.

Ilustración 14. Herramientas para exportar y graficar variables.

El uso de esta sección es muy sencillo. En primer lugar, hay que seleccionar las variables que se desean analizar de la lista y después hay que pulsar en “Plot” o “Export” en función de si se desea plotear o exportar estas variables. También cabe la posibilidad de utilizar ambas funciones simultáneamente. En caso de elegir la opción de “Export” se abrirá una ventana para introducir el nombre del archivo y la ruta donde se desea guardar.

Una vez se haya terminado de utilizar estas funciones se pulsarán los botones “Stop plotting”, “Stop exporting” o se cerrará el programa.

5. AJUSTE PID ONLINE

La sección PID permite el envío y recepción de las constantes de control de un vehículo aéreo. Se muestra su aspecto en la Ilustración 15.

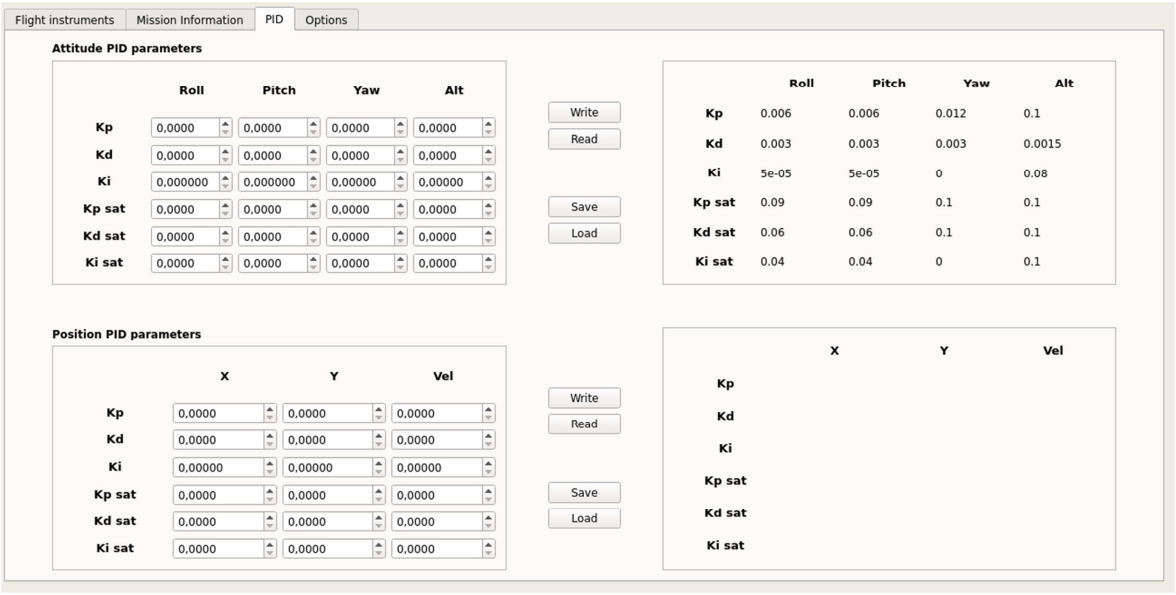

Ilustración 15. Ajuste online de las constantes de control.

Este bloque se ha diseñado específicamente para el quadrotor. Para ello se han implementado una serie de mensajes de MAVLink propios. Si el usuario final desea utilizar estas funciones pero utiliza otro vehículo distinto al quadrotor desarrollado tendrá que implementar estos mensajes.

Esta pantalla se divide en dos secciones, la configuración de los parámetros para el control de estabilidad y altura del quadrotor (el cual se ejecuta en el Arduino Due) y las constantes PID para el control de posición en la Igep.

Además, esta pestaña cuenta con funciones para guardar las constantes en archivos de texto y cargarlas posteriormente. Esto resulta muy útil a la hora de probar estrategias de control diferentes.

6. OPCIONES

Por último, en la pantalla de opciones se pueden configurar distintos parámetros que afectan al funcionamiento de la interfaz HMI. Como las otras ventanas, esta sección se divide en varios bloques. Se muestra su aspecto en la Ilustración 16.

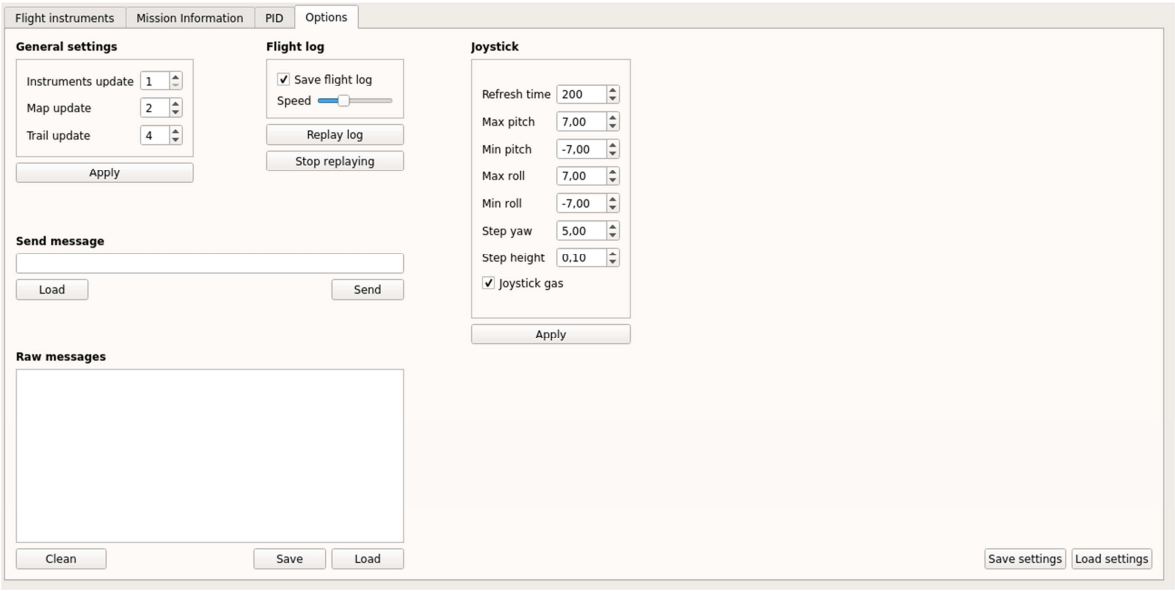

Ilustración 16. Pestaña de opciones del HMI.

El bloque “general settings” permite ajustar la frecuencia de actualización de los instrumentos, del mapa y del recorrido del vehículo que se dibuja en el mapa. Los valores introducidos son múltiplos de la frecuencia de envío de mensajes desde el UAV.

El bloque “flight log” permite guardar un archivo de texto con todos los mensajes recibidos del vehículo aéreo. Este log se guarda con nombre de la fecha y hora del vuelo. Posteriormente, es posible reproducir este log en la interfaz HMI para analizar lo sucedido en un vuelo anterior mediante el botón “Replay log”. Además es posible aumentar o disminuir la velocidad de reproducción o pararlo en cualquier momento.

La sección Joystick permite modificar la configuración de cualquier dispositivo que se conecte como joystick. Es posible modificar la sensibilidad del joystick cambiando los valores máximos y mínimos que éste puede alcanzar. Además, también se puede modificar el incremento que se produce en una referencia de altura o yaw mediante la opción “Step height” y “Step yaw”. Asimismo la opción “Refresh time” representa el tiempo en milisegundos después del cual el HMI lee el joystick y envía la información al vehículo de forma continuada.

UNIVERSITAT POLITÈCNICA DE VALÈNCIA

ESCUELA TÉCNICA SUPERIOR INGENIEROS INDUSTRIALES VALENCIA

TRABAJO FIN DE GRADO EN INGENIERÍA EN TECNOLOGÍAS INDUSTRIALES

MANUAL DE PROGRAMACIÓN DEL HMI EN QT

AUTOR: PABLO ALBIOL GRAULLERA

TUTOR: PEDRO JOSÉ GARCÍA GIL

COTUTOR:

Curso Académico: 2013-14

ÍNDICE

ÍNDICE DE ILUSTRACIONES

1. INTRODUCCIÓN

Este manual se presenta como la guía que deberá seguir un programador experimentado para entender y poder modificar, o incluso replicar, el código de ésta aplicación. La interfaz HMI desarrollada es una aplicación modular con una arquitectura optimizada para futuras extensiones así como contribuciones de terceras personas.

Este software ha sido desarrollado en C++ bajo un entorno Qt. Qt es una biblioteca multiplataforma para desarrollar aplicaciones en C++ con interfaz gráfica de usuario o sin ella y que además cuenta con IDE propio. Para el desarrollo de este programa se ha utilizado el IDE propio de Qt, Qt Creator 3.1 basado en Qt 5.2.1.

El objetivo de este manual no es explicar la estrategia de programación seguida para cada una de las funciones que incorpora el programa. Se escribe este manual con objeto de guiar al programador en algunos de los aspectos claves para el funcionamiento de la interfaz HMI. Es por ello que este manual de programación va dirigido a un ingeniero con experiencia en programación en lenguaje C++.

Así pues, esta guía se dividirá en varios apartados atendiendo a las novedades que incluye Qt respecto al lenguaje C++ y a los bloques que, por la estrategia de programación seguida, sean más difíciles de comprender.

En la Ilustración 1 se muestra el aspecto del IDE de Qt, Qt Creator con el proyecto de la interfaz gráfica HMI abierta.

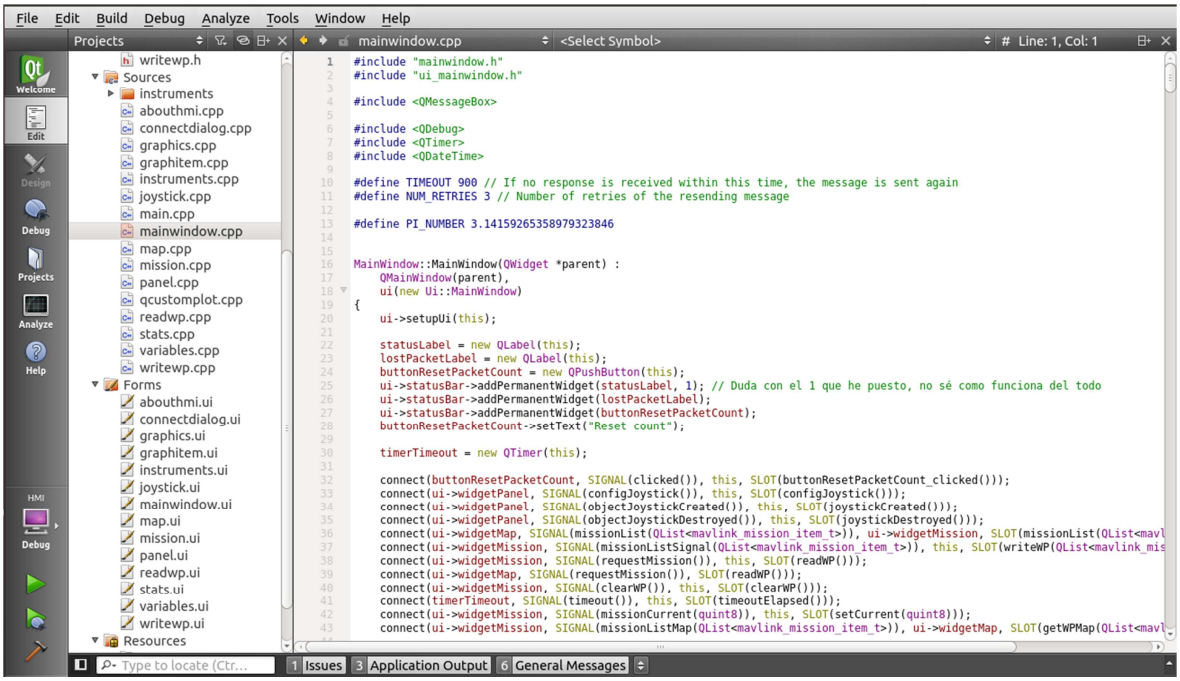

Ilustración 1. Qt Creator 3.1

En primer lugar, se hace una breve introducción a Qt, la programación dirigida por eventos y las diferencias respecto a la programación estructurada. En el tercer apartado se describen las

convenciones generales utilizadas en la programación de este software. A continuación (cuarto capítulo), se explica cómo se utiliza el puerto serie en Qt (abrirlo, configurarlo, escribir y leer en él). Después, en el quinto se describe la estrategia seguida para la gestión de un mapa de Google Maps. A continuación, el capítulo séptimo, explica el funcionamiento de funciones varias del HMI (instrumentos de vuelo, joystick, gráficas y logs de vuelos). El capítulo octavo describe el proceso a seguir para compilar con éxito el programa final en Qt y la distribución del mismo. Por último, en el capítulo número nueve se muestran las referencias de las herramientas utilizadas durante la programación de este software.

2. INTRODUCCIÓN A QT

Antes de comenzar, es necesario saber que todo proyecto con interfaz gráfica de usuario (GUI) en Qt tiene los mismos elementos, los cuales son:

- Ficheros de formularios: Con la extensión *.ui cada fichero describe en lenguaje XML la interfaz gráfica de formularios o ventana de la aplicación. Para diseñar los formularios en este proyecto se ha utilizado el modo “Design” de Qt Creator. En la Ilustración 2 se puede ver un ejemplo para la ventana del panel de vuelo.

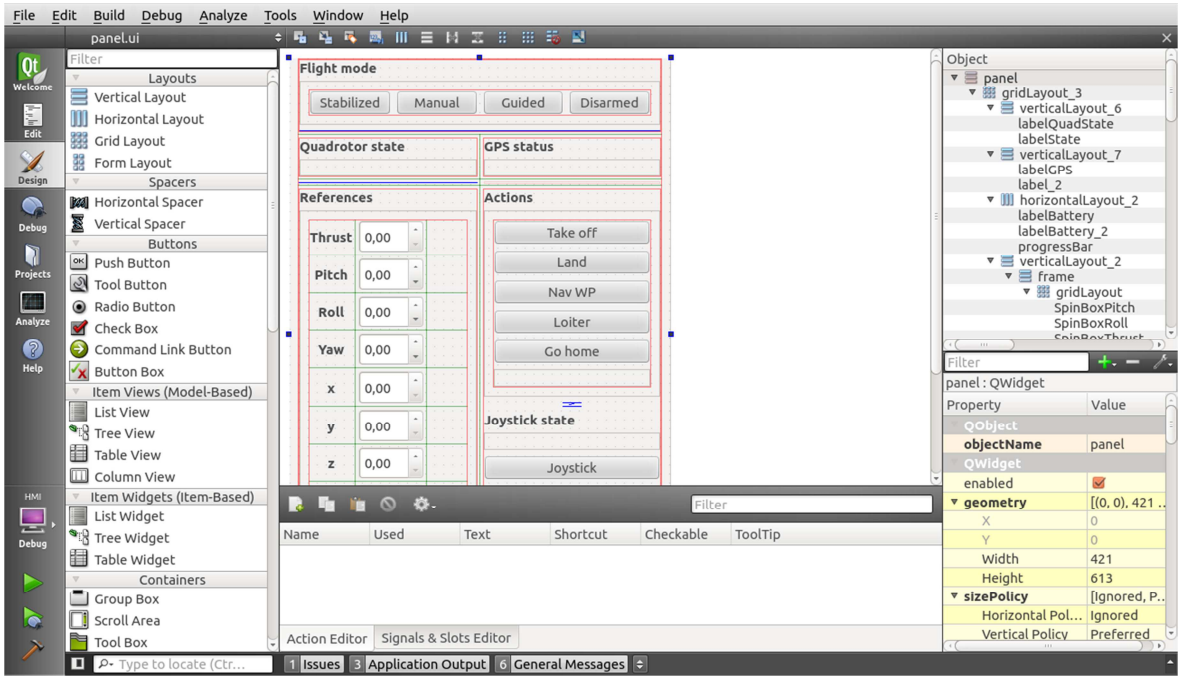

Ilustración 2. Formulario del panel de vuelo del HMI.

- Ficheros de cabezera: Con la extensión *.h. Se recomienda que la declaración de cada clase vaya en un fichero de cabecera por separado.
- Ficheros fuente: Con la extensión *.cpp. Es recomendable que la implementación de cada miembro de una clase esté contenido en un mismo fichero.
- Ficheros de recursos: Con la extensión *.qrc. Contienen los recursos utilizados en el programa (imágenes, iconos, etc).
- Fichero de proyecto: Con la extensión *.pro. Es un fichero que se genera automáticamente al trabajar con Qt Creator. En él se definen todos los elementos que componen el proyecto (headers, sources, forms, resources, librerías externas, etc.).

2.1. Programación Dirigida por Eventos

La programación dirigida por eventos es un paradigma de programación en el que el flujo del programa está determinado por eventos o mensajes desde otros programas o hilos de ejecución. En Qt esto se traduce a que el programador escribe el código que se ejecutará en respuesta a determinados eventos (llamados slots: pulsar un botón, elegir una opción del menú, abrir o cerrar

una ventana, etc.). No existe la idea de un control de flujo secuencial en el programa, sino que el programador toma el control cuando se dispara un evento.

En el HMI esto se implementa de la siguiente forma. Cuando se quiere asignar un trozo de código a un evento, como por ejemplo pulsar un botón, se selecciona el disparador del evento (el botón en este ejemplo) se pulsa el botón secundario del ratón y se selecciona la opción “Go to slot”. Automáticamente Qt Creator abre el editor para introducir el código deseado. A continuación se muestra un ejemplo del “slot” que se ejecuta al pulsar el botón de resetear el número de paquetes perdidos.

```
void MainWindow::on_pushButtonResetPacketCount_clicked()
{
    packetLost = 0;
    globalLost = 0;
    seqPrev = -1;
    lostPacketLabel->setText("Lost Packets: " + QString::number(globalLost));
}
```

Ilustración 3. Código fuente del slot que resetea el número de paquetes perdidos.

2.2. Manejo de la Memoria en Qt

El mal manejo de la memoria junto con el mal uso de los punteros es la fuente de la mayoría de errores y problemas de los programas desarrollados en C/C++.

En Qt, la mejor forma de no tener que poner código para liberar memoria con “delete” es que todos los objetos que compongan la aplicación tengan un padre o bien estén en la pila (“stack”). Esto es lo que se denomina herencia encadenada.

En la Ilustración 4 se muestra el main.cpp del HMI donde se crea una instancia (en el “stack”) de la clase MainWindow que es la ventana principal de la aplicación.

```
int main(int argc, char *argv[])
{
    QApplication a(argc, argv);
    MainWindow w;
    w.show();

    return a.exec();
}
```

Ilustración 4. Código fuente del main del HMI.

En la Ilustración 5 se muestran tres objetos creados dinámicamente que tienen un padre (this).

```
    statusLabel = new QLabel(this);
    lostPacketLabel = new QLabel(this);
    buttonResetPacketCount = new QPushButton(this);
```

Ilustración 5. Código fuente de objetos creados en el montón o "heap".

3. CONVENCIONES UTILIZADAS PARA LA PROGRAMACIÓN DEL HMI

En una aplicación de este calibre hay más de 300 variables diferentes y 200 objetos distintos, por lo tanto se hace necesario seguir una serie de normas y convenciones para que el código final sea entendible por terceras personas.

3.1. Nombres de Objetos

Los objetos que se crean al añadir widgets al formulario, tales como botones, etiquetas, cajas de selección o cuadros de texto, se nombran del siguiente modo. Primero se nombra el tipo de objeto y a continuación la variable que lo identifica, con la primera letra de cada palabra en mayúscula.

Por ejemplo, para un botón: pushButtonSetRoll, pushButtonGetRoll, pushButtonClearMap… En el caso de etiquetas, por ejemplo: labelShowRoll, labelMapInfo, etc.

3.2. Formatos de Archivos de Texto

Todos los archivos de texto que genera el programa ya sean logs, waypoints o variables exportadas están formateados con separaciones de tabulador entre los datos. Esto facilita que estos archivos sean fácilmente importables en programas como Excel o Matlab para analizarlos a posteriori.

3.3. Unidades

Salvo que se exprese lo contrario, todas las variables físicas utilizadas en el HMI tienen las unidades del Sistema Internacional y se representan con precisión IEE-754. Las unidades que no tienen representación física están normalizadas entre 0 y 1 y los porcentajes entre 0 y 100.

El sistema de referencia que se usa para las coordenadas locales es el ENU (x: este, y: norte, z: arriba).

3.4. Código

Como norma general, todas las variables y funciones están declaradas en los archivos de cabecera o “headers” mientras que en los archivos fuentes se implementan las funciones directamente.

Por último, el código del HMI se ha escrito usando el formato “BSD-style code”.

4. GESTIÓN DEL PUERTO SERIE

Las librerías de Qt implementan unas clases que proporcionan acceso al puerto serie. Se puede tanto manejar el puerto serie como obtener información de él. Las clases utilizadas son QSerialPort y QSerialPortInfo.

En la Ilustración 6 se muestra la función que abre el puerto serie.

```
int MainWindow::openPort()
{
    serial = new QSerialPort(this); // Create the serial object

    serial->setPortName(port);
    if (!(serial->open(QIODevice::ReadWrite)))
    {
        return 0;
    }

    // Port configuration
    serial->setBaudRate(baud);
    serial->setDataBits(QSerialPort::Data8);
    serial->setParity(QSerialPort::NoParity);
    serial->setStopBits(QSerialPort::OneStop);
    serial->setFlowControl(QSerialPort::NoFlowControl);

    serial->clear(QSerialPort::AllDirections);

    return 1;
}
```

Ilustración 6. Código fuente de la función que abre el puerto serie.

A continuación, si el puerto serie se ha abierto correctamente se conecta la señal readyRead() con la función receive() tal como se muestra en la Ilustración 7.

```
        // Connection
        if (openPort())
        {
            connected = true;
            statusLabel->setText("Port " + port + " opened");
            connect(serial, SIGNAL(readyRead()), this, SLOT(receive()));
            initialMsgVal();
            channel = MAVLINK_COMM_0;
        }
        else
        {
            QMessageBox::warning(this, "Error", "Couldn't open the port");
            statusLabel->setText("Port closed");
        }
```

Ilustración 7. Código fuente que conecta la lectura del puerto serie a la función Receive().

La parte más importante del código anterior es la línea: connect(serial, SIGNAL(readyRead()), this, SLOT(receive())). Esta línea conecta el objeto puerto serie (serial) a la función receive() del objeto actual (this) cuando se dispara la señal readyRead(). Por otro lado, la señal readyRead() es una señal definida en la clase QIODevice que es heredada por la clase QSerialPort y se emite cada vez que hay nuevos datos disponibles para ser leídos en el puerto serie.

En la Ilustración 8 se muestra una pequeña parte del código de la función que se ejecuta cada vez que hay bytes a leer en el puerto serie.

```
// Receive function (slot)
void MainWindow::receive()
{
    if ((sizRead = serial->bytesAvailable()))
    {
        serial->read((char *)buffRead, sizRead);
```

Ilustración 8. Código fuente de parte de la función Receive().

4.1. Protocolo MAVLink

La interfaz HMI hace uso del protocolo MAVLink para empaquetar las tramas de bytes que se enviarán al vehículo en cuestión y decodificar los mensajes que se reciben del mismo. Para más información sobre las ventajas del uso de este protocolo se puede consultar la memoria de este proyecto.

En primer lugar, tanto para recibir como para enviar datos, se tienen que declarar las estructuras necesarias en un archivo de cabecera (Ilustración 9).

```
    // MAVLink message structures
    mavlink_heartbeat_t heartbeat;
    mavlink_sys_status_t sys_status;
    mavlink_set_mode_t set_mode;
    mavlink_attitude_t attitude;
    mavlink_global_position_int_t global_position_int;
    mavlink_set_local_position_setpoint_t set_local_position_setpoint;
    mavlink_local_position_setpoint_t local_position_setpoint;
    mavlink_set_roll_pitch_yaw_thrust_t set_roll_pitch_yaw_thrust;
    mavlink_set_quad_motors_setpoint_t  set_quad_motors_setpoint;
    mavlink_command_long_t command_long;
    mavlink_command_ack_t command_ack;

    // WAYPOINT PROTOCOL structures
    mavlink_mission_item_t mission_item;
    mavlink_mission_item_t mission_item_aux;
    QList<mavlink_mission_item_t> mission_item_list;
    QList<mavlink_mission_item_t> mission_item_list_read;
    mavlink_mission_request_t mission_request;
    mavlink_mission_set_current_t mission_set_current;
    mavlink_mission_current_t mission_current;
    mavlink_mission_request_list_t mission_request_list;
    mavlink_mission_count_t mission_count;
    mavlink_mission_clear_all_t mission_clear_all;
    mavlink_mission_item_reached_t mission_item_reached;
    mavlink_mission_ack_t mission_ack;
```

Ilustración 9. Código fuente de la declaración de las estructuras de MAVLink.

En la Ilustración 10 se muestra un ejemplo del envío de un mensaje de mavlink de tipo SET_MODE (#11). Donde la variable len es el número de bytes del mensaje y bufWrite es un vector de bytes.

```
mavlink_msg_set_mode_encode(system_id, component_id, &msg, &set_mode);
len = mavlink_msg_to_send_buffer(bufWrite, &msg);
serial->write((char *)bufWrite, len);
```

Ilustración 10. Código fuente del envio del mensaje set_mode.

A continuación se muestra la función que decodifica los mensajes recibidos en la Ilustración 11. Este trozo de código es continuación del descrito en la Ilustración 8.

```
serial->read((char *)buffRead, sizRead);

for (int i = 0; i < sizRead; i++)
{
    if (mavlink_parse_char(channel, buffRead[i], &msg, &status))
    {
        packetLostCount();
        seqPrev = msg.seq;

        switch(msg.msgid)
        {
        case 0:
            mavlink_msg_heartbeat_decode(&msg, &heartbeat);
            ui->widgetPanel->flightModeButtons(heartbeat.base_mode);
            ui->widgetPanel->panelMessage(heartbeat.system_status,
heartbeat.custom_mode);
            break;

        case 1:
            mavlink_msg_sys_status_decode(&msg, &sys_status);
            ui->widgetPanel->setBattery(sys_status.voltage_battery);
            break;

        case 30:
            mavlink_msg_attitude_decode(&msg, &attitude);
            break;

        case 33:
            mavlink_msg_global_position_int_decode(&msg,
&global_position_int);
            break;
        }
    }
}
```

Ilustración 11. Código fuente de la función que decodifica los mensajes recibidos.

Cada vez que hay datos en el puerto serie se leen y se guardan en el vector buffRead. A continuación se pasa cada byte de este vector a la función de MAVLink mavlink_parse_char. En el caso de que se reciba un mensaje válido la función devuelve “true” y se entra en el switch case. Se ejecutará un caso u otro en función del tipo de mensaje recibido.

5. GESTIÓN DEL MAPA

Las librerías de Qt poseen una clase llamada QWebView que proporciona una herramienta para ver y editar documentos web. En este programa se ha utilizado esta clase para implementar un mapa de Google Maps.

Además, para poder interactuar este mapa se ha utilizado la versión 3 del API de Javascript de Google Maps (GoogleMapsAPI) que permite insertar Google Maps en páginas web o dispositivos móviles con un sinfín de opciones distintas.

Todo esto ha sido posible mediante la creación de un documento llamado map.html que se encuentra en la carpeta del ejecutable del HMI. Este documento invoca al API de Google Maps y además cuenta con las funciones Javascript necesarias para manejar el mapa y obtener información de él. De este modo, la interfaz HMI se comunica con el mapa a través de las funciones definidas en el documento map.html.

En la Ilustración 12 se muestra como se carga el documento map.html en la interfaz gráfica HMI.

```
map::map(QWidget *parent) :
    QWidget(parent),
    ui(new Ui::map)
{
    ui->setupUi(this);

    ui->webViewMap->settings()->setAttribute(QWebSettings::JavascriptEnabled,
true);
    QString fileName =
QDir::cleanPath(QDir::current().absoluteFilePath("googleMap/map.html"));

    if (!QFile(fileName).exists())
    {
        QMessageBox::warning(this, "Error", QString("Couldn't load the
map.\nFile not found: %1").arg(fileName));
    }

    ui->webViewMap->load(QUrl::fromLocalFile(fileName));
}
```

Ilustración 12. Código fuente que carga el archivo map.html.

Para comunicarse con el documento map.html se hace uso de la función evaluateJavaScript(). En la Ilustración 13 se pueden ver un ejemplo.

```
void map::on_pushButtonCenter_clicked()
{
    ui->webViewMap->page()->mainFrame()-
>evaluateJavaScript(QString("map.panTo(quad.getPosition());"));
    ui->webViewMap->page()->mainFrame()-
>evaluateJavaScript(QString("map.setZoom(20)"));
}
```

Ilustración 13. Código fuente que centra el quadrotor en el mapa y aplica un zoom.

Por otro lado, en la Ilustración 14 se muestra parte del código del documento map.html que invoca al API de Google Maps y se comunica con el HMI.

```
<script
src="http://maps.googleapis.com/maps/api/js?v=3.exp&sensor=false"></script>
    <script>

var map;
var quad;
var home;
var infoQuad;
var trails = [];
var lines = [];
var listWP = [];
var idWP = [];
var countWP = 0;

function initialize() {
  var mapOptions = {
    zoom: 2,
    center: new google.maps.LatLng(40.41687477895358, -3.703308399999969,
false),
    keyboardShortcuts: false,
    streetViewControl: false,
    mapTypeId: google.maps.MapTypeId.HYBRID,
    panControl: true,
    rotateControl: false,
    heading: 0,
    tilt: 0,
    scaleControl: true,
    disableDoubleClickZoom: true
  };

  map = new google.maps.Map(document.getElementById('map-canvas'),
      mapOptions);

  google.maps.event.addListener(map, 'dblclick', function(newWP) {
    addWP(newWP.latLng);
  });

  createHome();

}
```

Ilustración 14. Código fuente Javascript de parte del documento map.html.

La primera línea del código de la Ilustración 14 es la que incrusta el API de Google Maps en este documento html. A continuación se definen unas variables globales: map es el objeto principal que representa el mapa, quad es el icono del quadrotor que se mueve por pantalla, home es otra marca que representa la ubicación de la unidad de control terrestre, info es un cuadro de información que muestra la altura del vehículo, trails[] y lines[] son vectores que dibujan el recorrido el quadrotor en el mapa, listWP[] es un vector que contiene los waypoints seleccionados, idWP[] es otro vector con los identificativos de los waypoints y por último countWP es una variable que representa el número de waypoints.

Aunque la sintaxis de Javascript es diferente a C++, su uso no es complicado porque al igual que C++ es un lenguaje orientado a objetos. Sin embargo, para la programación de este bloque es necesario consultar la referencia y los ejemplos del API de Google Maps (GoogleMapsAPI).

6. OTRAS FUNCIONES DESTACADAS

6.1. Instrumentos de Vuelo

Los instrumentos de vuelo se han desarrollado a partir de un proyecto GPL que se encontró en internet (QFlightInstruments). El proyecto contaba con una serie de imágenes en formato svg que imitaban los instrumentos de un vehículo aéreo. Estas imágenes son fácilmente modificables con un programa de edición de gráficos vectoriales.

Así pues los instrumentos se basan en una serie de imágenes que giran sobre otras fijas. Esto se puede realizar en Qt mediante la clase QGraphicsItem. En la Ilustración 15 se muestra un ejemplo de empleo de esta clase para el control del instrumento horizonte artificial.

```
void qfi_ADI::updateView()
{
    m_scaleX = (float)width()  / (float)m_originalWidth;
    m_scaleY = (float)height() / (float)m_originalHeight;

    m_itemBack->setRotation( - m_roll );
    m_itemFace->setRotation( - m_roll );
    m_itemRing->setRotation( - m_roll );

    float roll_rad = M_PI * m_roll / 180.0;
    float delta  = m_originalPixPerDeg * m_pitch;

    m_faceDeltaX_new = m_scaleX * delta * sin( roll_rad );
    m_faceDeltaY_new = m_scaleY * delta * cos( roll_rad );
    m_itemFace->moveBy( m_faceDeltaX_new - m_faceDeltaX_old, m_faceDeltaY_new
- m_faceDeltaY_old );

    m_scene->update();
}
```

Ilustración 15. Código fuente que actualiza el instrumento horizonte artificial.

6.2 Uso del Joystick

Qt no cuenta con ninguna clase específica para gestionar un joystick o gamepad, es por eso que se ha utilizado una librería externa multiplataforma. La librería externa utilizada ha sido SFML ("Simple and Fast Multimedia Library"). En la Ilustración 16 hay un ejemplo de su uso.

```
void joystick::getJoystick()
{
    idJoystick = ui->comboBoxJoystickID->itemData(ui->comboBoxJoystickID-
>currentIndex()).toInt();
    sf::Joystick::update();

    xAxis = sf::Joystick::getAxisPosition(idJoystick, sf::Joystick::X);
    yAxis = sf::Joystick::getAxisPosition(idJoystick, sf::Joystick::Y);
    zAxis = sf::Joystick::getAxisPosition(idJoystick, sf::Joystick::Z);
    b3 = sf::Joystick::isButtonPressed(idJoystick, 3)
    b4 = sf::Joystick::isButtonPressed(idJoystick, 4)
    b5 = sf::Joystick::isButtonPressed(idJoystick, 5)
    b6 = sf::Joystick::isButtonPressed(idJoystick, 6)
}
```

Ilustración 16. Código de fuente que obtiene las referencias del joystick.

6.3. Gráficas

Qt tampoco cuenta con ninguna herramienta específica para dibujar gráficas. Para ello se ha utilizado un widget descargado de internet que permite realizar esto a través de las librerías oficiales de Qt. El widget utilizado ha sido QCustomPlot.

El bloque de código que se muestra en la Ilustración 17 se ejecuta cada vez que se plotea en la gráfica un nuevo valor.

```
void graphitem::plotData(float y)
{
    double key = QDateTime::currentDateTime().toMSecsSinceEpoch()/1000.0;
    // add data to lines:
    ui->widget->graph()->addData(key, y);
    // remove data of lines that's outside visible range:
    ui->widget->graph()->removeDataBefore(key-8);
    // rescale value (vertical) axis to fit the current data:
    ui->widget->graph()->rescaleValueAxis();
    // make key axis range scroll with the data (at a constant range size of
8):
    ui->widget->xAxis->setRange(key+0.25, 8, Qt::AlignRight);
    ui->widget->replot();
}
```

Ilustración 17. Código fuente que grafica un nuevo valor.

6.4. Guardado y Reproducción de Datos de Vuelo

La función de guardado, guarda los mensajes recibidos en crudo (sin decodificar) en archivos de texto con nombre de la fecha y hora del vuelo. El contenido del vuelo se puede reproducir posteriormente leyendo el log guardado línea a línea. En la Ilustración 18 se muestra la estrategia seguida para inicializar esta función.

```
void MainWindow::on_pushButtonReplayLog_clicked()
{
    int sliderTimeout;

    QString replayName = QFileDialog::getOpenFileName(this, "Open File",
"log/", "Text Files (*.txt)");
    fileReplayLog.setFileName(replayName);

    timerReplay = new QTimer(this);
    connect(timerReplay, SIGNAL(timeout()), this, SLOT(receiveReplay()));
    sliderTimeout = ui->horizontalSlider->value();
    timerReplay->start((int)(sliderTimeout * 1.6 + 40));
    statusLabel->setText("Replaying log");
}
```

Ilustración 18. Código fuente que inicializa la reproducción de datos.

La variable sliderTimeout es la que marca la frecuencia de reproducción de los datos. Cada vez que se supera este tiempo, el objeto timerReplay, el cual es un contador, emite una señal y se ejecuta la función receiveReplay() que es casi igual a la función receive() de la Ilustración 11 pero en vez de leer el puerto serie, lee el archivo de texto en cuestión.

7. COMPILACIÓN DEL PROGRAMA

Antes de empezar, se recomienda utilizar la versión 5.2.1 de las librerías de Qt que son las que se han utilizado durante el desarrollo de este software. No obstante el código debería poder ser compilado sin mayores problemas con versiones posteriores a la 5.0.

La única dependencia externa que es necesario indicar a Qt para que el código se pueda compilar, es la librería SFML, utilizada para el manejo del joystick. Esto se indica en el archivo HMI.pro como se muestra en la Ilustración 19.

```
# Joystick
LIBS += -L"C:/Users/User/Desktop/QUADROTOR/HMI/SFML-2.1 - windows/lib"
LIBS += -lsfml-window -lsfml-system
INCLUDEPATH += "C:/Users/User/Desktop/QUADROTOR/HMI/SFML-2.1 -
windows/include"
DEPENDPATH += "C:/Users/User/Desktop/QUADROTOR/HMI/SFML-2.1 - windows/include"
```

Ilustración 19. Código fuente del archivo HMI.pro que incluye la librería SFML.

Los proyectos pueden ser construidos en dos modos diferentes: el modo Debug y el modo Release. El modo se selecciona con el botón de seleccionar modo de compilación.

- Modo Debug: Es más adecuado cuando se está en la fase de desarrollo del proyecto. Se genera toda la información de depuración.
- Modo Release: Se usa una vez acabado el proyecto, para entregar el programa al cliente. El código está más optimizado en tiempo y espacio.

Finalmente, para compilar el código es recomendable seguir los siguientes pasos:

1. Clean Project.
2. Run qmake.
3. Build

7.1. Distribución del Programa

Para distribuir el programa es necesario que el sistema operativo y la arquitectura del procesador donde se vaya a ejecutar el HMI sea la misma que donde se ha compilado (Windows 7 32 bits, Windows 7 64 bits, Ubuntu 32 bits, Ubuntu 64 bits, etc.).

Además para que todo funcione sin dar problemas, es necesario distribuir las librerías de Qt y las librerías de SFML. Para ello existen dos opciones:

1. El usuario final debe instalar directamente en su ordenador las librerías de Qt (versión posterior a la 5.0) y las librerías SFML.
2. Entregar al usuario final las librerías necesarias con el ejecutable del programa.

En este proyecto se ha utilizado la segunda opción. Las librerías de Qt y SFML se utilizan dinámicamente durante la ejecución de la aplicación.

8. REFERENCIAS

GoogleMapsAPI. (s.f.). *Google Maps*. Obtenido de https://developers.google.com/maps

MAVLink. (s.f.). *QGroundControl*. Obtenido de http://qgroundcontrol.org/mavlink/start

QCustomPlot. (s.f.). *QCustomPlot*. Obtenido de http://www.qcustomplot.com/

QFlightInstruments. (s.f.). *Sourceforge*. Obtenido de http://sourceforge.net/projects/qfi/

Qt. (s.f.). *Qt Project*. Obtenido de http://qt-project.org/

SFML. (s.f.). *Simple and Fast Multimedia Library*. Obtenido de http://www.sfml-dev.org/

UNIVERSITAT POLITÈCNICA DE VALÈNCIA

ESCUELA TÉCNICA SUPERIOR INGENIEROS INDUSTRIALES VALENCIA

TRABAJO FIN DE GRADO EN INGENIERÍA EN TECNOLOGÍAS INDUSTRIALES

PRESUPUESTO

AUTOR: PABLO ALBIOL GRAULLERA

TUTOR: PEDRO JOSÉ GARCÍA GIL

COTUTOR:

Curso Académico: 2013-14

ÍNDICE

1. INTRODUCCIÓN

El documento que sigue tiene como objetivo proporcionar toda la información detallada del coste de ejecución del trabajo final de grado: "Desarrollo integral de un QuadRotor: Diseño e implementación de los sistemas de telemetría e interfaz gráfica (HMI) para la monitorización y control mediante entorno Qt".

Este presupuesto desglosa el coste total de la primera parte del proyecto (desarrollo de la plataforma de vuelo) y de la segunda parte específica del proyecto (diseño e implementación de los sistemas de telemetría e interfaz HMI). Notar que, para obtener el presupuesto total del trabajo en conjunto (desarrollo integral de un QuadRotor), habría que incluir los presupuestos de los trabajos de los estudiantes Alberto Castillo y Daniel Verdú.

Para ello, este proyecto se ha dividido en 10 unidades de obra. Dichas unidades de obra se han estructurado en 2 capítulos del siguiente modo:

CAPÍTULO 1: Desarrollo de la plataforma de vuelo.

- UO 1.1: Montaje de la estructura de la plataforma:
  Montaje de todos los elementos básicos de la estructura de un quadrotor. Incluyendo chasis, tren de aterrizaje, actuadores, protección, sistema de alimentación etc.
- UO 1.2: Montaje y disposición de la electrónica:
  Montaje de todos los equipos electrónicos, sensores, conexionado y comunicaciones entre equipos para poder controlar el quadrotor.
- UO 1.3: Programación del software embebido:
  Programación de todos los algoritmos y unidades de control necesarios para controlar el quadrotor, incluyendo la programación en el Arduino Due y la Igep.
- UO 1.4: Implementación del control de estabilidad:
  Programación e implementación de las leyes de control para la estabilidad del quadrotor incluyendo el ajuste experimental de las constantes de control.
- UO 1.5: Ensayos y test de vuelo:
  Realización de vuelos de prueba en el laboratorio, en exteriores y comprobación del correcto funcionamiento de todos los subsistemas.

CAPÍTULO 2: Implementación y diseño de los sistemas de telemetría.

- UO 2.1: Implementación del hardware necesario en la plataforma:
  Implementación del hardware para enlazar el quadrotor con la unidad de control terrestre (UCT) incluyendo configuración, instalación y pruebas de los módulos xbee.
- UO 2.2: Programación del software que gestiona la telemetría en el ordenador de a bordo de la plataforma:
  Programación del software en el ordenador de a bordo del vehículo para el control y la correcta monitorización del quadrotor desde un ordenador en tierra. Incluye la programación del puerto serie, implementación del protocolo MAVLink y comunicación con el resto del programa.

- UO 2.3: Programación de la interfaz gráfica HMI en la unidad de control terrestre (HMI):
  Programación de una interfaz gráfica de usuario multiplataforma para controlar y monitorizar el vehículo.
- UO 2.4: Implementación de una emisora y receptor de radiocontrol:
  Implementación de un lazo de control a través de una emisora y un receptor de radiocontrol sobre un microcontrolador. Incluye la implementación en términos de hardware y software.
- UO 2.5: Test de funcionamiento de todos los sistemas:
  Pruebas en el laboratorio y en exteriores del sistema de telemetría implementado incluyendo el enlace a través de los módulos de radio y el enlace por medio de la emisora y el receptor de radiocontrol.

Cabe mencionar que la elaboración del presente proyecto no ha precisado de una aportación económica, puesto que la mano de obra ha sido realizada por un estudiante con el fin de desarrollar su trabajo final de grado. Además, tanto el hardware utilizado, como las licencias de software ya estaban en disposición del Departamento de Ingeniería de Sistemas y Automática de la Universidad Politécnica de Valencia.

2. CUADRO DE PRECIOS DE LA MANO DE OBRA

Para la realización de este proyecto se emplean 3 graduados en ingeniería en tecnologías industriales que realizarán tareas como: implementación del hardware en la plataforma, programación del software embebido, implementación de controladores, ensayos de vuelo, etc. Además también se contrata a un técnico de laboratorio para llevar a cabo tareas de ensamblaje y fabricación principalmente.

El cálculo del coste de la mano de obra por hora se hace partiendo de un salario medio al mes. De este modo, los 3 graduados en tecnologías industriales percibirán un sueldo de 1200 € al mes mientras que el técnico de laboratorio percibirá un salario de 1000 € al mes. Este salario se divide entre las horas laborales de un mes para obtener el coste de la mano de obra por hora.

El cuadro de precios de la mano de obra queda del siguiente modo:

Código	Empleado	Salario mensual (€/mes)	Horas de trabajo al dia	Días labaroables por mes	Salario (€ / h)
MO.GITI	Graduado en Tecnologías Industriales	1200	8	20	7,5
MO.TEC	Técnico de laboratorio	1000	8	20	6,25

3. CUADRO DE PRECIOS DE MATERIALES

Código	Denominación del material		Precio unitario (€)
MT.EST	ud	Estructura Mikrokopter (41 cm diagonal)	62,05
MT.MBR	ud	Tren de aterrizaje de Mikrokopter	33,85
MT.MBR	ud	Motores brushless Robbe-Roxxy 2827-35	51,20
MT.VAR	ud	Variadores YGE 30i	69,00
MT.HEL	ud	Helices 10x4,5 (x2ud)	5,00
MT.CMC	ud	Contrachapado de madera de chopo 160x160 cm	18,70
MT.PEE	ud	Poliestireno expandido 100x100 cm	2,00
MT.COB	ud	Conectores bala 3,5 mm (x100ud)	12,25
MT.CACP	ud	Conectores para cables de potencia (x25ud)	7,60
MT.JTO	ud	Juego de tornillos 330 uds	3,80
MT.CDTB	ud	Conector DeansT para batería (x5ud)	4,90
MT.INT	ud	Interruptores	1,95
MT.BRI	ud	Bridas (x100ud)	1,90
MT.CAS	ud	Cinta aislante	2,30
MT.BAT	ud	Batería LiPo 3S GENS-ACE 5300 mAh	59,00
MT.CPE	ud	Bote de cola para poliestireno expandido	6,75
MT.MCB	ud	Medidor de carga de baterías LiPo	5,99
MT.ARDU	ud	Arduino Due	47,19
MT.IGE	ud	Igep v2	179,00
MT.ARMI	ud	Arduino Micro	24,20
MT.ULT	ud	Sensor de ultrasonidos Ping	18,03
MT.IMU	ud	Drotek IMU 10DOF v2 = MPU6050 + HMC5883 + MS5611	22,90
MT.GPS	ud	Sensor GPS Ultimate v3 Adafruit	29,25
MT.MRA	ud	Módulo de radiofrecuencia Xbee Pro S1	27,78
MT.RRC	ud	Receptor RC Corona RP8D1	18,20
MT.CAL	m	Cables de alimentación	2,32
MT.RTS	ud	Regulador de tensión	1,95
MT.CNL	ud	Convertidor de niveles lógicos	2,16
MT.LED	ud	Led (x20ud)	1,63
MT.BSN	ud	Bobina de estaño	4,54
MT.IDC	ud	Conectores IDC (x10ud)	2,95
MT.PWM	ud	Conectores PWM (x10ud)	1,90
MT.RES	ud	Kit de resistencias	5,90
MT.CCO	ud	Cables para comunicaciones (x40ud)	3,75
MT.CJ	ud	Conector Jack 2,1 x 5,5 mm	1,05
MT.EF	ud	Emisora Futaba T6EX 35MGHz	124,02
MT.JSTK	ud	Joystick logitech xtreme x3 pro	46,21
MT.MSD	ud	Tarjeta micro SD 2GB	2,92
MT.ADPT	ud	Adaptador FTDI	12,95
MT.PBA	ud	Placa baquelita 10 x 10 cm	13,30
MT.PMF	ud	PCB manufacturada	3,00

4. CUADRO DE PRECIOS DE MAQUINARIA

Se parte de la hipótesis de que para desarrollar este proyecto es necesario adquirir la siguiente maquinaria. El cálculo de los rendimientos de la maquinaria se ha realizado en función del número de horas totales de uso de cada herramienta. De este modo, con el supuesto anterior, se consigue que al final del proyecto se haya pagado el coste entero de los equipos.

A continuación se muestra el cuadro de precios de la maquinaria.

Código	Denominación de la maquinaria		Uds	Precio (€)	Horas de uso	Rendimiento
M.OSM	ud	Ordenador completo de sobremesa	1	518,00	150	3,45
M.OSC	ud	Osciloscopio	1	1064,95	10	106,50
M.OP	ud	Ordenador Portátil	2	719,00	500	2,88
M.FA	ud	Fuente de alimentación 40A	1	146,85	100	1,47
M.CDB	ud	Cargador baterías LiPo	1	14,60	50	0,29
M.POL	ud	Polímetro	1	22,80	125	0,18
M.TAL	ud	Taladradora con juego de brocas y limas	1	42,00	20	2,10
M.PTF	ud	Pistola termofusible	1	8,50	2	4,25
M.SDE	ud	Soldador de estaño	1	11,90	30	0,40
M.AHC	ud	Arco cortador de hilo caliente	1	37,81	4	9,45
M.HER	ud	Kit de herramientas	1	53,23	100	0,53
M.CLS	h	Alquiler cortadora laser de control numerico	1	25,00	1	25,00

5. CUADRO DE PRECIOS UNITARIOS

5.1. Desarrollo de la Plataforma de Vuelo

Unidad de Obra	Precio
UO 1.1: Montaje de la estructura de la plataforma.	1519,15
UO 1.2: Montaje y disposición de la electrónica.	1383,13
UO 1.3: Programación del software embebido.	993,22
UO 1.4: Implementación del control de estabilidad.	683,77
UO 1.5: Ensayos y test de vuelo.	767,63

5.2. Implementación y Diseño de los Sistemas de Telemetría

Unidad de Obra	Precio
UO 2.1 Implementación del hardware necesario en la plataforma.	504,89
UO 2.2 Programación del software que gestiona la telemetría en el ordenador de a bordo.	495,33
UO 2.3 Programación de la interfaz gráfica HMI en la UCT.	680,93
UO 2.4 Implementación de una emisora y receptor de radiocontrol.	884,11
UO 2.5 Test de funcionamiento de todos los sistemas.	146,13

6. CUADRO DE PRECIOS DESCOMPUESTOS

6.1. Cálculo de Horas por Tarea

En este apartado se han dividido las unidades de obra en tareas para poder hacer una estimación del número de horas que tienen que dedicar los graduados en ingeniería en tecnologías industriales y el técnico de laboratorio a cada unidad de obra.

Estos cálculos se utilizan en la segunda parte de este apartado para obtener los precios descompuestos.

6.1.1. Desarrollo de la Plataforma de Vuelo

Unidad de Obra 1.1	Montaje de la estructura de la plataforma.			
	GITI (h)	Num GITI	Tec (h)	Num Tec
Ensamblaje del chasis y tren de aterrizaje	9	3	0,5	1
Diseño de la protección	11	2	0	0
Montaje del sistema de alimentación	4	3	0,5	1
Montaje de los actuadores	4	3	0	0
TOTAL		**73**		**1**

Unidad de Obra 1.2	Montaje y disposición de la electrónica.			
	GITI (h)	Num GITI	Tec (h)	Num Tec
Montaje PCB del Arduino	5	2	1	1
Instalación de la IGEP en la plataforma	1	3	1	2
Conexionado de sensores	2	3	0	0
Pruebas de sensores	10	3	0	0
TOTAL		**49**		**3**

Unidad de Obra 1.3	Programación del software embebido.			
	GITI (h)	Num GITI	Tec (h)	Num Tec
Programación del código del Arduino Due	13	3	0	0
Instalación Linux + Xenomai en la IGEP	7	2	0	0
Programación del código de la IGEP	17	3	0	0
TOTAL		**104**		**0**

Unidad de Obra 1.4	Implementación del control de estabilidad.			
	GITI (h)	Num GITI	Tec (h)	Num Tec
Calibración hélices	1	1	0	0
Programación de los variadores	3	2	0	0
Ajuste motores	3	2	0	0
Comprobación señales y comunicaciones	10	3	0	0
Ajuste experimental de las constantes de control	9	3	0	0
TOTAL		**70**		**0**

Unidad de Obra 1.5	Ensayos y test de vuelos.			
	GITI (h)	Num GITI	Tec (h)	Num Tec
Test de vuelos en el laboratorio	12	3	0	0
Test de vuelos en exteriores	13	3	0	0
TOTAL		**75**		**0**

6.1.2. Implementación y Diseño de los Sistemas de Telemetría

Unidad de Obra 2.1	Implementación del hardware necesario en la plataforma.			
	GITI (h)	Num GITI	Tec (h)	Num Tec
Configuración de los módulos de radio	8	1	0	0
Disposición de los módulos en la plataforma	6	1	0	0
Instalación de las xbee en la plataforma	16	1	2	1
Pruebas de alcance de los módulos	10	1	0	0
TOTAL		**40**		**2**

Unidad de Obra 2.2	Programación del software que gestiona la telemetría en el ordenador de a bordo.			
	GITI (h)	Num GITI	Tec (h)	Num Tec
Programación del hilo de telemetría en la Igep	18	1	0	0
Programación del puerto serie en la Igep	6	1	0	0
Implementación del protocolo MAVLink	10	1	0	0
Comunicaciones con el resto de threads	14	1	0	0
TOTAL		**48**		**0**

Unidad de Obra 2.3	Programación de la interfaz gráfica HMI en la UCT.			
	GITI (h)	Num GITI	Tec (h)	Num Tec
Programación de la ventana principal	8	1	0	0
Programación del mapa	12	1	0	0
Programación de los instrumentos de vuelo	6	1	0	0
Programación del panel de vuelo	6	1	0	0
Programación del bloque PID	3	1	0	0
Programación del joystick	3	1	0	0
Programación del bloque de estadísticas	4	1	0	0
Programación del bloque de waypoints	8	1	0	0
Programación del bloque opciones	2	1	0	0
Distribución de la aplicación	6	1	0	0
TOTAL		**58**		**0**

Unidad de Obra 2.4	Implementación de una emisora y receptor de radiocontrol.			
	GITI (h)	Num GITI	Tec (h)	Num Tec
Implementación del hardware necesario	6	1	0	0
Programación del software que lee el receptor	6	1	0	0
Comunicación con Arduino Due	2	1	0	0
Configuración del receptor	1	1	0	0
Configuración de la emisora	2	1	0	0
TOTAL		**17**		**0**

Unidad de Obra 2.5	Test de funcionamiento de todos los sistemas.			
	GITI (h)	Num GITI	Tec (h)	Num Tec
Test en el laboratorio	10	1	0	0
Test en exteriores	2	1	0	0
TOTAL		**12**		**0**

6.2. Precios descompuestos

6.2.1. Desarrollo de la Plataforma de Vuelo

UO 1.1	Montaje de la estructura de la Plataforma.				
Código	Descripción		Precio	Rendimiento	Importe
MO.GITI	h	Graduado en Tecnologías Industriales	7,50	74	555,00
MO.TEC	h	Técnico de Laboratorio	7,00	1	7,00
M.AHC	h	Arco cortador de hilo caliente	9,45	4	37,81
M.PTF	h	Pistola termofusible	4,25	1,5	6,38
M.TAL	h	Taladradora con juego de brocas y limas	2,10	10	21,00
M.HER	h	Kit de herramientas	0,53	75	39,92
M.CLS	h	Alquiler cortadora laser de control numérico	25,00	1	25,00
MT.EST	ud	Estructura Mikrokopter (41 cm diagonal)	62,05	1	62,05
MT.MBR	ud	Tren de aterrizaje de Mikrokopter	33,85	1	33,85
MT.MBR	ud	Motores brushless Robbe-Roxxy 2827-35	51,20	4	204,80
MT.VAR	ud	Variadores YGE 30i	69,00	4	276,00
MT.HEL	ud	Helices 10x4,5 (x2ud)	5,00	2	10,00
MT.CMC	ud	Contrachapado de madera de chopo 160x160 cm	18,70	1	18,70
MT.PEE	ud	Poliestireno expandido 100x100 cm	2,00	1	2,00
MT.COB	ud	Conectores bala 3,5 mm (x100ud)	12,25	1	12,25
MT.CACP	ud	Conectores atornillados para cables de potencia (x25ud)	7,60	1	7,60
MT.JTO	ud	Juego de tornillos 330 uds	3,80	1	3,80
MT.CDTB	ud	Conector DeansT para batería (x5ud)	4,90	1	4,90

MT.INT	ud	Interruptores	1,95	2	3,90
MT.BRI	ud	Bridas (x100ud)	1,90	1	1,90
MT.CAS	ud	Cinta aislante	2,30	1	2,30
MT.BAT	ud	Batería LiPo 3S GENS-ACE 5300 mAh	59,00	1	59,00
MT.CPE	ud	Bote de cola para poliestireno expandido	6,75	1	6,75
MT.CAL	m	Cables de alimentación	2,32	1,5	3,48
MT.BAT	ud	Batería LiPo 3S GENS-ACE 5300 mAh	59,00	1	59,00
MT.CPE	ud	Bote de cola para poliestireno expandido	6,75	1	6,75
MT.CAL	m	Cables de alimentación	2,32	1,5	3,48
	%	Costes directos complementarios		0,01	14,75
	%	Costes indirectos		0,02	29,79
Total unidad de obra					**1519,15**

UO 1.2	**Montaje y disposición de la electrónica.**				
Código	**Descripción**		**Precio**	**Rendimiento**	**Importe**
MO.GITI	h	Graduado en Tecnologías Industriales	7,50	49	367,50
MO.TEC	h	Técnico de Laboratorio	7,00	3	21,00
M.OSC	h	Osciloscopio	106,50	4	425,98
M.OSM	h	Ordenador completo de sobremesa	3,45	20	69,07
M.OP	h	Ordenador Portátil	2,88	20	57,52
M.FA	h	Fuente de alimentación 40A	2,40	10	23,97
M.POL	h	Polímetro	0,18	40	7,30
M.TAL	h	Taladradora con juego de brocas y limas	2,10	10	21,00
M.PTF	h	Pistola termofusible	4,25	0,5	2,13
MT.ARDU	ud	Arduino Due	47,19	1	47,19
MT.IGE	ud	Igep v2	179,00	1	179,00
MT.ARMI	ud	Arduino Micro	24,20	1	24,20
MT.ULT	ud	Sensor de ultrasonidos Ping	18,03	1	18,03
MT.IMU	ud	Drotek IMU 10DOF v2 = MPU6050 + HMC5883 + MS5611	22,90	1	22,90
MT.RTS	ud	Regulador de tensión	1,95	1	1,95
MT.CNL	ud	Convertidor de niveles lógicos	2,16	1	2,16
MT.LED	ud	Led (x20ud)	1,63	1	1,63
MT.BSN	ud	Bobina de estaño	4,54	1	4,54
MT.IDC	ud	Conectores IDC (x10ud)	2,95	1	2,95
MT.PWM	ud	Conectores PWM (x10ud)	1,90	1	1,90
MT.RES	ud	Kit de resistencias	5,90	1	5,90
MT.CCO	ud	Cables para comunicaciones (x40ud)	3,75	1	3,75
MT.MSD	ud	Tarjeta micro SD 2GB	2,92	1	2,92
MT.PBA	ud	Placa baquelita 10 x 10 cm	13,30	1	13,30
MT.PMF	ud	PCB manufacturada	3,00	3	9,00
	%	Costes directos complementarios		0,01	13,43
	%	Costes indirectos		0,02	27,12
		Total unidad de obra			**1383,13**

UO 1.3	Programación del software embebido.				
Código	**Descripción**		**Precio**	**Rendimiento**	**Importe**
MO.GITI	h	Graduado en Tecnologías Industriales	7,50	104	780,00
M.OSM	h	Ordenador completo de sobremesa	3,45	20	69,07
M.OP	h	Ordenador Portátil	2,88	40	115,04
	%	Costes directos complementarios		0,01	9,64
	%	Costes indirectos		0,02	19,47
		Total unidad de obra			**993,22**

UO 1.4	Implementación del control de estabilidad.				
Código	**Descripción**		**Precio**	**Rendimiento**	**Importe**
MO.GITI	h	Graduado en Tecnologías Industriales	7,50	70	525,00
M.OSM	h	Ordenador completo de sobremesa	3,45	10	34,53
M.OP	h	Ordenador Portátil	2,88	25	71,90
M.FA	h	Fuente de alimentación 40A	1,47	20	29,37
M.CDB	h	Cargador baterías LiPo	0,29	10	2,92
	%	Costes directos complementarios		0,01	6,64
	%	Costes indirectos		0,02	13,41
		Total unidad de obra			**683,77**

UO 1.5	Ensayos y test de vuelo.				
Código	**Descripción**		**Precio**	**Rendimiento**	**Importe**
MO.GITI	h	Graduado en Tecnologías Industriales	7,50	75	562,50
M.OSM	h	Ordenador completo de sobremesa	3,45	20	69,07
M.OP	h	Ordenador Portátil	2,88	15	43,14
M.FA	h	Fuente de alimentación 40A	1,47	40	58,74
M.CDB	h	Cargador baterías LiPo	0,29	40	11,68
	%	Costes directos complementarios		0,01	7,45
	%	Costes indirectos		0,02	15,05
		Total unidad de obra			**767,63**

6.2.2. Implementación y Diseño de los Sistemas de Telemetría

UO 2.1		Implementación del hardware necesario en la plataforma.			
Código	**Descripción**		**Precio**	**Rendimiento**	**Importe**
MO.GITI	h	Graduado en Tecnologías Industriales	7,50	40	300,00
MO.TEC	h	Técnico de Laboratorio	7,00	2	14,00
M.OSC	h	Osciloscopio	106,50	1	106,50
M.FA	h	Fuente de alimentación 40A	1,47	2	2,94
M.POL	h	Polimetro	0,18	15	2,74
M.HER	h	Kit de herramientas	0,53	5	2,66
M.SDE	h	Soldador de estaño	0,40	4	1,59
MT.MRA	ud	Módulo de radiofrecuencia Xbee Pro S1	27,78	2	55,56
MT.RTS	ud	Regulador de tension	1,95	1	1,95
MT.CNL	ud	Convertidor de niveles lógicos	2,16	1	2,16
	%	Costes directos complementarios		0,01	4,90
	%	Costes indirectos		0,02	9,90
		Total unidad de obra			**504,89**

UO 2.2		Programación del software que gestiona la telemetría en el ordenador de a bordo.			
Código	**Descripción**		**Precio**	**Rendimiento**	**Importe**
MO.GITI	h	Graduado en Tecnologías Industriales	7,50	48	360,00
M.OSM	h	Ordenador completo de sobremesa	3,45	10	34,53
M.OP	h	Ordenador Portátil	2,88	30	86,28
	%	Costes directos complementarios		0,01	4,81
	%	Costes indirectos		0,02	9,71
		Total unidad de obra			**495,33**

UO 2.3		Programación de la interfaz gráfica HMI en la UCT.			
Código	**Descripción**		**Precio**	**Rendimiento**	**Importe**
MO.GITI	h	Graduado en Tecnologías Industriales	7,50	58	435,00
M.OP	h	Ordenador Portatil	2,88	58	166,81
MT.JSTK	ud	Joystick logitech xtreme x3 pro	46,21	1	46,21
MT.ADPT	ud	Adaptador FTDI	12,95	1	12,95
	%	Costes directos complementarios		0,01	6,61
	%	Costes indirectos		0,02	13,35
		Total unidad de obra			**680,93**

UO 2.4	Implementación de una emisora y receptor de radiocontrol.				
Código	**Descripción**		**Precio**	**Rendimiento**	**Importe**
MO.GITI	h	Graduado en Tecnologías Industriales	7,50	17	127,50
M.OP	h	Ordenador Portátil	2,88	15	43,14
M.OSC	h	Osciloscopio	106,50	5	532,48
M.FA	h	Fuente de alimentación 40A	1,47	4	5,87
M.POL	h	Polímetro	0,18	15	2,74
M.HER	h	Kit de herramientas	0,53	5	2,66
M.SDE	h	Soldador de estaño	0,40	4	1,59
MT.RRC	ud	Receptor RC Corona RP8D1	18,2	1	18,20
MT.EF	ud	Emisora Futaba T6EX 35MGHz	124,02	1	124,02
	%	Costes directos complementarios		0,01	8,58
	%	Costes indirectos		0,02	17,34
		Total unidad de obra			**884,11**

UO 2.5	Test de funcionamiento de todos los sistemas.				
Código	**Descripción**		**Precio**	**Rendimiento**	**Importe**
MO.GITI	h	Graduado en Tecnologías Industriales	7,50	12	90,00
M.OSM	h	Ordenador completo de sobremesa	3,45	5	17,27
M.OP	h	Ordenador Portátil	2,88	10	28,76
M.FA	h	Fuente de alimentación 40A	1,47	4	5,87
	%	Costes directos complementarios		0,01	1,36
	%	Costes indirectos		0,02	2,87
		Total unidad de obra			**146,13**

7. PRESUPUESTO PARCIAL

7.1. Desarrollo de la Plataforma de Vuelo

Unidad de Obra	Cantidad	Precio	Coste Total
UO 1.1 Montaje de la estructura de la plataforma.	1	1519,15	1519,15
UO 1.2 Montaje y disposición de la electrónica.	1	1383,13	1383,13
UO 1.3 Programación del software embebido.	1	993,22	993,22
UO 1.4 Implementación del control de estabilidad.	1	683,77	683,77
UO 1.5 Ensayos y test de vuelo.	1	767,63	767,63
TOTAL			**5346,90**

7.2. Implementación y Diseño de los Sistemas de Telemetría

Unidad de Obra	Cantidad	Precio	Coste Total
UO 2.1 Implementación del hardware necesario en la plataforma.	1	504,89	504,89
UO 2.2 Programación del software que gestiona la telemetría en el ordenador de a bordo.	1	495,33	495,33
UO 2.3 Programación de la interfaz gráfica HMI en la UCT.	1	680,93	680,93
UO 2.4 Implementación de una emisora y receptor de radiocontrol.	1	884,11	884,11
UO 2.5 Test de funcionamiento de todos los sistemas.	1	146,13	146,13
TOTAL			**2711,39**

8. PRESUPUESTO DE EJECUCIÓN MATERIAL, PRESUPUESTO DE INVERSIÓN Y PRESUPUESTO BASE DE LICITACIÓN

Por último, se expone el coste total de cada capítulo y se calculan los presupuestos de ejecución material, de inversión y base de licitación.

Capítulos	Coste
Capítulo 1: Desarrollo de la Plataforma de Vuelo	5346,90
Capítulo 2: Implementación y Diseño de los Sistemas de Telemetría	2711,39
TOTAL	**8058,29**

PRESUPUESTO DE EJECUCIÓN MATERIAL	**8058,29**
15% Gastos generales	1208,74
6% Beneficio industrial	483,50
PRESUPUESTO DE INVERSIÓN	**9750,53**
21% IVA	2047,61
PRESUPUESTO BASE DE LICITACIÓN	**11798,14**

Asciende el presupuesto de ejecución material a la cantidad de EUROS:
OCHO MIL CINCUENTA Y OCHO CON VEINTINUEVE

Asciende el presupuesto de inversión a la cantidad de EUROS:
NUEVE MIL CINCUENTA CON CINCUENTA Y TRES

Asciende el presupuesto base de licitación a la cantidad de EUROS:
ONCE MIL SETECIENTOS NOVENTA Y OCHO CON CATORCE