UNIVERSIDAD
REY JUAN CARLOS

ESCUELA TÉCNICA SUPERIOR DE INGENIERÍA DE TELECOMUNICACIÓN

GRADO EN INGENIERÍA EN SISTEMAS DE TELECOMUNICACIÓN

Curso Académico 2019/2020

Trabajo Fin de Grado

HACIENDO PROGRAMABLE Y ESTABLE CON FPGA UN DRONE COMERCIAL

Autor : Eloy Navarro Morales

Tutor : Dr. José María Cañas Plaza

Cotutor : Juan Ordoñez Cerezo

Resumen

Dada la expansión de los drones en distintas tareas de la actualidad y del uso de las FPGAs como núcleos de procesamiento, resulta natural mezclar dichos dispositivos para beneficio mutuo. A lo largo de este proyecto se han diseñado distintos elementos software y hardware para que un programa en Python ejecutado en un ordenador controle por completo las operaciones y trayectoria de vuelo de un dron comercial. Este es capaz de obedecer las instrucciones de posición indicadas por el programa sin intervención humana de por medio.

Para ello se ha diseñado una estructura basada en: (a) un dron de bajo coste equipado con un conjunto de sensores tridimensionales de posición y una radio adicional para comunicar las medidas a tierra; (b) una estación de tierra basada en FPGA que ejecuta y comunica el control de vuelo con el dron mediante un transceptor radio; y (c) un ordenador para ejecutar la aplicación en Python y conectado por USB a la estación de tierra. Se escogió un dron X5C de Syma como plataforma de vuelo mejorada mediante sensores de BitCraze, junto a una FPGA de la familia ICE40 de Lattice para la estación y procesadores periféricos de Atmel, consiguiendo así un sistema final de bajo coste.

Tras el diseño de los distintos elementos se han realizado ensayos de verificación de cada uno por separado e integraciones de los mismos. La solución completa diseñada y programada se ha validado experimentalmente con éxito. Se han conseguido resultados especialmente satisfactorios con los sistemas en bucle cerrado, tanto en la estabilización como en el control de vuelo.

Summary

Given the expansion of drones in today different tasks and the use of FPGAs as processing cores, it is natural to mix these devices for mutual benefit. During this project, various software and hardware elements have been designed allowing a Python program runing on a computer to completely control the operation and flight path of a commercial drone. It is able to obey the position instructions indicated by the program without human intervention.

For this purpose, has been designed a structure based on: (a) a low-cost drone equipped with a set of three-dimensional position sensors and additional radio to communicate measurements to the ground; (b) an FPGA-based ground station that executes and communicates flight control with the drone via a radio transceiver; and (c) a computer to run the application in Python and connected via USB to the ground station. A Syma X5C drone was chosen as the improved flight platform using BitCraze sensors, along with an FPGA from the Lattice ICE40 family for the base station and Atmel peripheral processors, achieving a low-cost final system.

After the design of the different elements, verification tests have been carried out for each module separately followed by integrations. The complete designed and programmed solution has been experimentally validated successfully. Particularly satisfactory results have been achieved with closed-loop systems, both in stabilization and flight control.

Índice general

Índice de figuras

Capítulo 1

Introducción

El desarrollo de este trabajo fin de grado está orientado a hacer programable y estable un dron comercial usando una FPGA.
En la actualidad se ha extendido con rapidez el uso de drones y de FPGAs [20] más allá del entorno militar, en distintas áreas como vigilancia, cine, ocio, sistemas industriales, etc... Ambos dispositivos se han popularizado debido a su potencia y versatilidad de uso. Esto es gracias a que podemos controlar y guiar al dron con su carga útil. Estas tareas se llevan a cabo habitualmente de diversas maneras. El guiado mediante imagen [28], tele-dirección, ubicación global y controles abiertos o por posición son algunos métodos comunes. En este TFG se hace uso de un cuadricóptero comercial y de FPGAs libres para cubrir las tareas de control y estabilización[19], fomentando además el uso y propagación de código abierto.
A continuación se introducen algunas características de ambos dispositivos, proporcionando así el contexto genérico en el que se encuadra este trabajo.

1.1. Drones

Un dron es a una aeronave voladora no tripulada. Pueden ser pilotadas de manera remota o mediante controles automáticos, dotándolas de distintos elementos según el mecanismo de control escogido y la tarea a cubrir.
Este trabajo hace uso de un cuadricóptero comercial X5C del fabricante Syma. Por tanto se mencionan algunos sistemas propios de los cuadricópteros que no se encuentran presentes necesariamente en otros tipos de drones. Algunos sistemas son específicos de cada aplicación,

como los estabilizadores de cámara usados en aplicaciones de grabación de video. Otros, por el contrario, son comunes a todos los cuadricópteros, como la estructura, sensores, electrónica de control, drivers de motores, motores y sistemas de gestión y almacenamiento de energía (baterías y reguladores). Además existen sistemas que si bien no son estrictamente necesarios para todas las aplicaciones, rara vez se ven excluidos, como los sistemas de comunicaciones.

Las empresas más importantes en el desarrollo y la venta de drones comerciales son principalmente chinas, como Dji, Xiaomi, Hover o Syma junto con la francesa Parrot. En 2019, el fabricante español Hemav, se hizo un hueco, resultando elegida como la cuarta empresa más importante dentro del sector de drones no militares, gracias a sus tareas de inspección y creación de mapas [7].

1.1.1. Aplicaciones de vehículos aéreos no tripulados

Las aplicaciones de los drones son muchas, desde su inicio en el sector militar, como muchas otras tecnologías, se ha extendido su utilización a otros sectores civiles para cubrir tareas variadas.

Una de las áreas con mayor impacto en el desarrollo y aplicación está siendo el sector agrícola. Con hectáreas de cultivos que controlar y cada vez más tipos de sensores, se utilizan desde control de plagas (vehículo fumigando en la Figura 1.1), hasta por ejemplo, exploración de viñedos para determinar el mejor momento para la vendimia.

Figura 1.1: Dron aplicando fitosanitarios desde el aire

A menudo hacen uso de controladores basados en ubicación global a través de GPS para

el control de posición y trayectoria del dron, y de sensores ópticos y multiespectrales para determinar parámetros como la actividad clorofílica o su estrés hídrico. Este es el caso del sensor Sequoia+ de Parrot [21]. Dicho sensor se monta a bordo del dron que va a realizar la misión como el de la Figura 1.2.

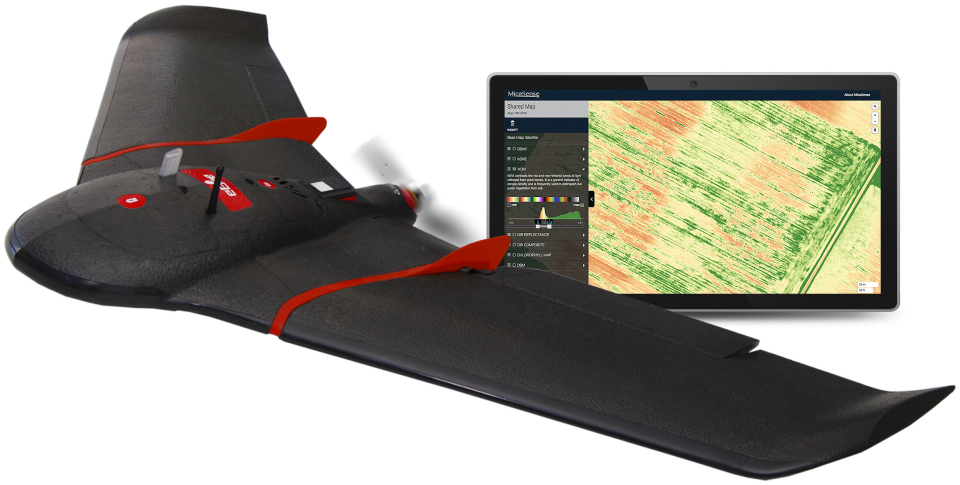

Figura 1.2: Drone senseFly con sensor óptico Sequoia+

De esta manera se analiza desde el cielo, por ejemplo, cuánta luz absorben y reflejan las plantas, permitiendo optimizar la producción del agricultor.

Debido a la ventaja de volar a alturas inferiores que las de un helicóptero, el uso de los drones se ha extendido también al rescate de personas. Como en el caso del montañero Rick Allen, avistado en el Himalaya gracias al uso de un dron (Figura 1.3), por parte de otros dos escaladores [24]. Integrando cámaras de alta resolución en los mismos y sistemas de localización de desaparecidos, como es el caso del proyecto de seguridad “LifeSeeker” [5] de Centum se podría ubicar a personas desaparecidas. El dispositivo Lifeseeker ha sido diseñado para instalarse en aviones no tripulados [1]. El sistema embarcado hace uso de los móviles de los desaparecidos, aun fuera de las zonas de cobertura, como si se tratasen de radiobalizas, permitiendo así su localización.

Otros ejemplos utilización de drones en sistemas de seguridad y rescate son el reparto masivo de medicamentos e incluso envíos urgentes de sangre para transfusiones [12].

También ha habido controversia relacionada con el control aéreo en la aplicación de drones para envíos. La legislación actual dificulta el desarrollo en esta industria a la misma velocidad

Figura 1.3: Imagen tomada desde el dron utilizado durante el rescate de Rick Allen

que se han expandido los drones en otras áreas. No obstante empresas como DHL o Amazon han desarrollado tecnologías y prototipos al respecto [3] y las han probado e implantado en zonas en las que no existen dichas limitaciones. A través de este servicio, Amazon envían paquetes en tiempo inferiores a los 30 minutos usando drones como el de la Figura 1.4. Son sistemas híbridos con despegue vertical y vuelo horizontal. Gozan de navegación autónoma guiada por GPS y controlan el espacio aéreo a su alrededor mediante visión artificial, para evitar obstáculos y localizar su área de aterrizaje.

Figura 1.4: dron de Amazon en prueba real de envío

1.1.2. Estructura

La estructura base o marco de un dron es donde apoyan y montan el resto de componentes. Sus principales requisitos son: amortiguar vibraciones, ser extremadamente rígida y ligera. Si las vibraciones no se viesen adecuadamente atenuadas, ciertas frecuencias podrían resonar mecánicamente a lo largo de toda la estructura, viendo comprometida la estabilidad del dron. Por su lado, la rigidez es fundamental, ya que para un adecuado control del dron, se necesita asumir independencia de ejes. Si partimos de una situación estable, por ejemplo, con el dron en vuelo a 1 metro de altura; si los cuatro motores generan el mismo impulso adicional (asumiéndolos ideales), en una estructura totalmente rígida el dron sencillamente ascendería. El par de rotación de las cuatro aspas se cancelaría entre sí, en caso de establecer el sentido de giro como muestra la Figura 1.5.

Figura 1.5: Sentido de rotación en cuadricóptero

En caso de que la estructura no resulte idealmente rígida durante el empuje ascendente, se producirían momentos idénticos (M1, M2, M3 y M4 de la Figura 1.6) sobre los cuatro motores, pero los ejes que unen dichos motores al marco sufrirían torsiones desiguales, lo que forzaría al dron a girar sobre sí mismo. Dicho error de rotación podría ser corregido por otros sistemas del dron, pudiendo llegar a saturarlos, dificultándoles realizar su tarea adecuadamente.

1.1.3. Sensores

Parte fundamental de todo cuadricóptero es un conjunto de sensores capaz de medir aceleraciones y giros. Para la mayoría de los drones comerciales, se hace uso de acelerómetros y giróscopos MEMS [9], es decir, elementos mecánicos y electromecánicos miniaturizados e integrados en solo chip. Por ejemplo un giróscopo y acelerómetro MEMS son en realidad un capacímetro diferencial conectado a un conjunto de láminas intercaladas, unas fijas al sustrato

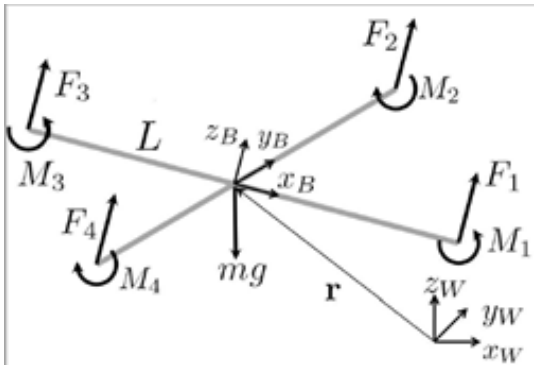

Figura 1.6: Fuerzas asociadas a motores en dron

y otras libres. Cuando se produce una aceleración sobre el chip, este es capaz de cuantificarla.
A menudo se integran acelerómetros y giróscopos de tres ejes en un solo dispositivo, llamado plataforma inercial. De esta manera se facilita su integración en el sistema y se abaratan costes. Esta plataforma mide las aceleraciones y giros respecto de sus ejes de referencia y pone esta información a disposición del dron.
La resolución y precisión son parámetros fundamentales de la calidad del sensor, tanto en la medida de aceleraciones, como en los giros.
Hay un error de desplazamiento asociado a dichos sensores, un giróscopo es incapaz de detectar todo nivel de giro que quede por debajo de su resolución. Es decir, una velocidad de giro suficientemente lenta y constante, por debajo del salto mínimo de velocidad angular que el sensor es capaz de resolver, produciría un giro indetectable y por tanto incorregible para el dron. Dicho error también podría ser corregido por otro tipo de sensor, por ejemplo, un magnetómetro a modo de brújula corrigiendo errores de giro mediante orientación global. Pero a menudo estos no se instalan, debido al incremento del coste.

1.1.4. Electrónica de Control

Este módulo es el encargado de procesar las señales de los sensores, entregando las respuestas necesarias para (a) estabilizar el dron en giro y plano horizontal y (b) en caso de recibir comandos por radio u otro mecanismo, modificar las órdenes a entregar a los sistemas de actuación, para obedecer.
En una situación de vuelo estable el objetivo de esta electrónica es corregir las aceleraciones y giros con las órdenes adecuadas sobre los motores. Por ejemplo, si un sensor detecta una

aceleración a izquierdas, esta electrónica deberá responder generando una fuerza en sentido contrario, para este caso, consistirá en inclinar el dron ligera y proporcionalmente a derechas. Para ello entregará mayor indicación de velocidad de giro a los drivers de los motores izquierdos, y menor a los derechos. Con esto el dron se inclinaría, corrigiendo la aceleración que de otro modo le hubiera desplazado hacia la izquierda. Este es un caso muy simplificado a modo de ejemplo. La electrónica a bordo computa las ecuaciones de movimiento para el cuadricóptero [6] a la frecuencia que le permite su capacidad de proceso, junto con los tiempos de adquisición de los sensores y respuesta de los drivers y motores.

1.1.5. Drivers y motores

Este hardware es el encargado de convertir las órdenes de velocidad de giro en giro real de las aspas conectadas a los motores. Cubren la función de transductores de instrucción a fuerza, a través de etapas de potencia que dependen de la construcción de los motores. La estructura general desde el controlador, hasta los motores, se muestra en la Figura 1.7.

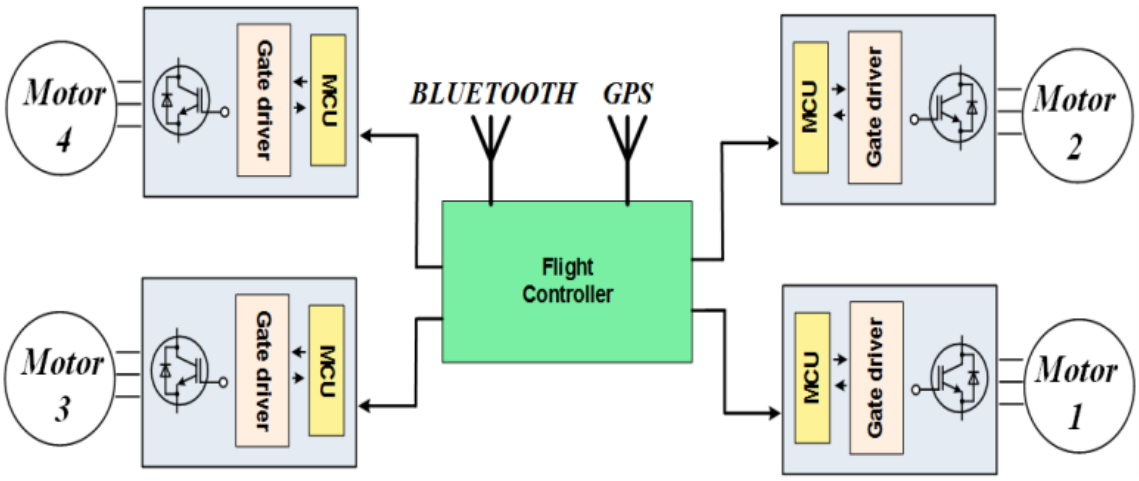

Figura 1.7: Bloques Hardware del sistema de control y drivers de un cuadricóptero

Los motores pueden ser basados en escobillas, o libres de ellas. Cada tipo de motor tiene una forma concreta de ser controlado [2]. Los que disponen de escobillas se manejan mediante métodos de control de motores de corriente continua. Por tanto las etapas de potencia de los drivers se despreocupan de la posición del rotor respecto del estator y transmiten mayor o menor cantidad de corriente al bobinado, a través de puentes en H controlados por PWM. Para el caso

de un motor sin escobillas hay múltiples métodos de control para producir el giro. La dificultad reside en saber en qué instante y a qué velocidad se puede variar el campo magnético en las tres fases del motor. Para ello hay algunos sistemas basados en conocer la posición relativa entre el rotor y el estator mediante sensores de efecto hall. Otros sistemas en cambio, trabajan sin dichos sensores.

1.1.6. Comunicaciones

Dependiendo del tipo de dron, no es estrictamente necesario disponer de un sistema de comunicaciones radio, ya que puede ser sustituido por un conjunto de instrucciones a ejecutar a bordo del dron, sin necesidad de modificar su comportamiento durante el vuelo. Pero tanto para drones teledirigidos, como para el de este TFG, la radio es un componente fundamental, encargada de comunicar la electrónica del dron con la electrónica en tierra que tiene por objetivo dirigir el dron. Aquí rigen múltiples tipos de sistemas y estándares, el más común se encuentra en banda libre de 2.4Ghz y es en el que se apoya este TFG. Se envían comandos radio digitales en forma de tramas de doce paquetes, siendo cada paquete un canal del dron. Los cuatro canales comunes a todo cuadricóptero son: Altura (-Uz), cabeceo (pitch), guiñada (yaw) y alabeo (roll) ordenados como se muestra en la Figura 1.8.

1.2. Computación reconfigurable, FPGAs

Las siglas hacen referencia a un array de puertas lógicas reprogramables in situ (Field Programmable Gate Array). Son la evolución de una tecnología previa similar llamada CPLD, que gracias a una reordenación de los elementos que las componen, consiguen una mayor densidad de puertas lógicas por centímetro cuadrado.
Una FPGA es un dispositivo que agrupa elementos capaces de realizar pequeñas operaciones lógicas y almacenar sus resultados en biestables. Dichas operaciones lógicas y memorias pueden interconectarse de manera muy flexible, permitiendo llevar a cabo operaciones y tareas complejas. Su programación se realiza a través de lenguajes de descripción de hardware, orientados a trabajar a bajo nivel. Para el caso de este trabajo se ha hecho uso de Verilog 2001. En un lenguaje similar a C en aspecto, pero con un concepto de fondo radicalmente distinto.

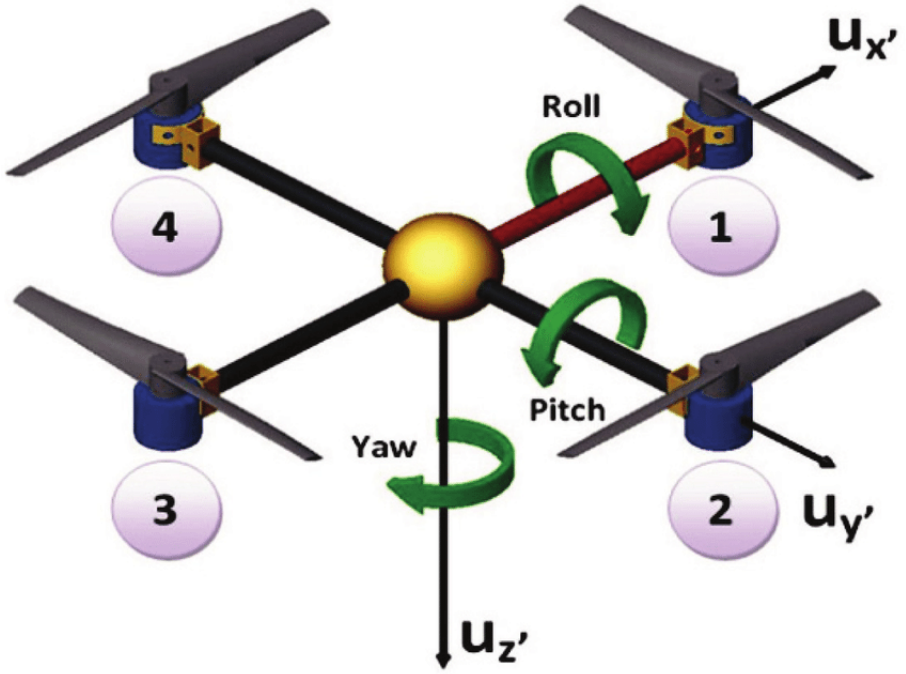

Figura 1.8: Altura (Uz), cabeceo (pitch), guiñada (yaw) y alabeo (roll)

1.2.1. Aplicaciones

Las FPGAs cubren un espectro muy grande de aplicaciones, ya que van desde las típicas sustituciones de sistemas digitales combinacionales a núcleo de sistemas paralelos, en ocasiones inabordables para un procesador secuencial estándar.

Algunas aplicaciones potentes a día de hoy para FPGAs son, por ejemplo, la visión artificial, procesos industriales o procesador de redes neuronales [22]. Estas últimas a menudo se han llevado a cabo por procesadores gráficos que, en su esencia, son una mezcla de muchos pequeños procesadores pensados para realizar cómputos geométricos simples. En ocasiones se están sustituyendo por FPGAs debido a la flexibilidad, potencia y capacidad de re-configuración de las mismas [29]. Los avances en unidades de procesamiento gráfico a menudo no alcanzan a cubrir las necesidades de dichas redes y dispositivos FPGA equivalentes en potencia suelen disponer de mayor eficiencia energética y facilidad para el clustering.

Gracias a su robustez son también utilizadas en entornos de seguridad como los ferroviarios, e incluso comienzan a invadir entornos menos comunes, como el espacial.

1.2.2. Características

Una FPGA dispone de ciertas características que le ofrecen ventaja sobre opciones programables más tradicionales, como los procesadores secuenciales. Estas pueden ser programadas de tal manera que ejecuten un conjunto de instrucciones concreto varias veces en paralelo, sin ver disminuido su rendimiento global, ni impactar en otros módulos pre-programados. Para un procesador secuencial realizar la misma tarea dos veces supone esperar a que la primera acabe para ejecutar la segunda, o disponer de hardware dedicado por duplicado, construido sobre el mismo silicio, opción no siempre disponible. Una FPGA, en cambio, es capaz de construir el mismo sistema hardware tantas veces como recursos hardware disponga, permitiendo que todos ellos ejecuten en paralelo, con un detrimento del rendimiento muy pequeño y controlable. Esta ventaja es de especial importancia en procesos de control cuya tarea puede dividirse en muchas subtareas, como en procesamiento de imágenes, o sistemas de procesado digital de señales.
Otra potente característica es su capacidad de re-configuración dinámica. Esta les ha otorgado popularidad en aplicaciones en las que se requiere rediseñar por completo el sistema de procesado, sin necesidad de un cambio físico sobre el hardware. Permiten modificar, por ejemplo, un filtro completo, no solo el valor de sus coeficientes; o reordenar los módulos de un sistema de telecomunicaciones, dotándolo de funciones completamente diferentes en base a una arquitectura radicalmente nueva. Características como esta hacen que estos dispositivos se popularicen para nuevas aplicaciones.

1.2.3. Desarrollo con FPGAs

Para el desarrollo de sistemas basados en FPGAs se hace uso de herramientas software generalmente distribuidas por el propio fabricante de la FPGA. Suele tratarse de entornos integrados de desarrollo que incluyen la mayor cantidad de comodidades, módulos IP pre-compilados y herramientas posibles. Este es el caso de Quartus, distribuido por Intel, que incluye una versión gratuita muy limitada en prestaciones y capacidades y versiones de pago con todas las utilidades disponibles. Un caso similar ocurre con otros fabricantes, como Xilinx o Atmel.
El proceso parte desde la generación del código de descripción de hardware que definirá el comportamiento del sistema. Este se compila, sintetiza y analiza, en busca de errores de sintaxis (problemas en el uso del lenguaje), construcciones imposibles de generar (hardware ló-

gicamente funcional, pero sin posibilidad de ser construido en la FPGA destino) o problemas de temporización (incumplimientos en tiempos relacionados con la velocidad de ejecución de los bloques lógicos). Una vez hecho esto se puede proceder a ubicar y conectar los elementos sintetizados en los bloques de la FPGA (comprobando disponibilidad y viabilidad). El proceso finaliza con la generación de la trama de bits (BitStream) a cargar en la FPGA.

1.2.4. FPGAs Libres

FPGAs libres son aquellas en las que se encuentra disponible toda la información de su diseño interno, formatos de almacenamiento de datos (BitStream) y en general todo aquello necesario para que un diseñador sea capaz de crear las herramientas de software necesarias para programar dichos dispositivos. En la actualidad muchas de las herramientas de diseño software para FPGAs son software propietario con un elevado coste de adquisición. En contraposición a este tipo de software se encuentra el software libre, o abierto, en general englobado bajo licencias GPL (General Public License) que tienen por objetivo defender su libre distribución sin necesidad de adquirir licencias de pago. Bajo este tipo de licencias se encuentra software como Icestudio (Figuras 1.9), orientado al diseño de software para FPGAs de Lattice.

Figura 1.9: Logo de Icestudio, IDE para desarrollo de software en FPGAs libres

Esta herramienta, dispone de un entorno gráfico para la programación del hardware mediante el uso de bloques como muestra la Figura 1.10. También ofrece soporte para la programación de las FPGAs abiertas, mediante código abierto.

Se trata de una herramienta en constante crecimiento. Una de las familias de FPGAs recientemente cubiertas por este software es la ICE40 UltraPlus, concretamente el integrado ICE40UP5K.

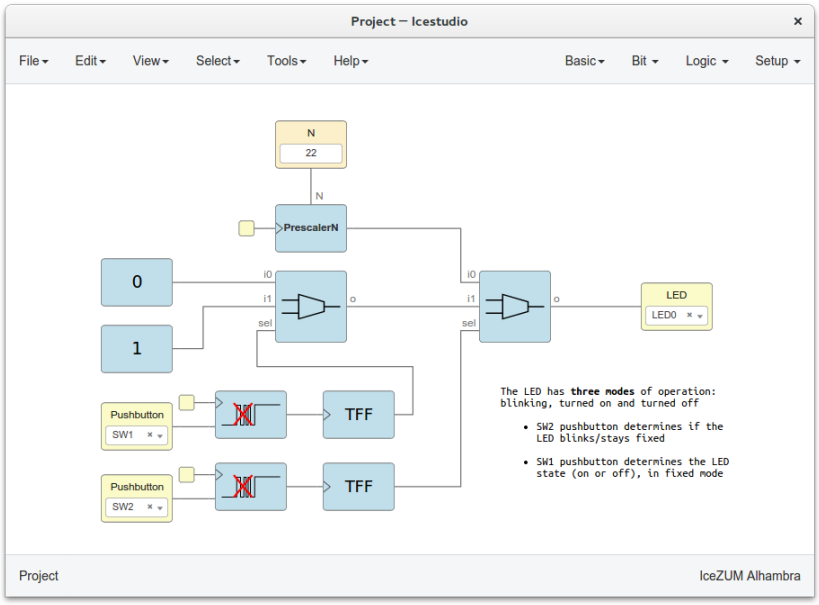

Figura 1.10: Captura del entorno Icestudio

Capítulo 2

Objetivos

Habiendo ubicado en el capítulo anterior el contexto y los elementos principales con los que se trabajará durante este desarrollo, en este capítulo fijamos el problema concreto que se aborda este TFG y los requisitos que la solución a desarrollar tiene también que satisfacer.

2.1. Objetivo principal

En este trabajo se pretende hacer estable y programable el vuelo de un dron comercial de bajo coste haciendo uso de FPGAs libres. El sistema deberá ser capaz de estabilizar el vuelo del vehículo de manera autónoma, haciendo uso de electrónica de sensorización embarcada en el dron y computación sobre la FPGA en tierra. Adicionalmente, el dron deberá obedecer las instrucciones de movimiento ejecutadas en el PC, en comunicación con la FPGA de modo que se puedan ejecutar en el ordenador programas que gobiernen el movimiento estable del dron.

Este objetivo principal lo hemos articulado en tres subobjetivos cuyas funcionalidades, en conjunto, permitirán alcanzar el objetivo principal.

- Se incorporarán al vehículo aéreo los elementos de sensorización y comunicación adecuados a la tarea. Estos permitirán conocer su ubicación de manera local y establecer una comunicación con la estación de tierra, donde se ubicará la FPGA que computará y comunicará las correcciones pertinentes de vuelta.

- Se diseñará una estación de tierra que establezca la comunicación con el dron. Esta FPGA recibirá continuamente los datos de los sensores embarcados en el dron, ejecutará los

controladores de vuelo y le enviará continuamente las órdenes de movimiento a la plataforma voladora. En este subobjetivo se creará el software necesario para conseguir un vuelo estacionario y estable del vehículo.

- La estación de tierra deberá también comunicarse con el PC, permitiendo que programas ejecutando en el ordenador modulen a los controladores ejecutados en la FPGA de la estación. De esta manera, el PC se convertirá en el ordenador de mando indirectamente. Este ejecutará las instrucciones que comandarán los movimientos del dron, siendo la FPGA en la estación de tierra la encargada de que el vehículo obedezca dichas órdenes de manera estable.

2.2. Requisitos

Además de resolver los objetivos planteados, la solución desarrollada se deberá cubrir también unos requisitos mínimos.

- El sistema deberá ser capaz de estabilizar el dron en al menos *tres grados de libertad*; cabeceo, alabeo y altitud.
- Cada grado de libertad se controlará independientemente y permitirán el control en *tiempo real* del vehículo.
- El sistema desarrollado será *reconfigurable*. Para facilitar las tareas de estabilización y otorgar flexibilidad al sistema, los controladores deberán disponer de parámetros reconfigurables que permitan variar la respuesta de comportamiento del dron.
- El control desde el ordenador funcionará también en tiempo real, de modo ágil computacionalmente, con poco retardo desde la ejecución de una instrucción en el ordenador hasta que se materializa en la plataforma voladora.
- El sistema funcionará en un ordenador con Windows 10.
- Las electrónicas, FPGA y dron empleados deberán estar hechos con elementos de bajo coste.

2.3. Metodología

Para realizar el trabajo, se mantendrán reuniones semanales de seguimiento por videoconferencia con los tutores, informando de los avances y posibles problemas. También se informará por correo u otros medios en caso de resultar útil. En estas reuniones se comentarán los avances realizados, los problemas encontrados y se propondrán posibles soluciones y las siguientes tareas a abordar.

El trabajo se organizará segmentando el objetivo semanal en tareas más simples. Se diseñarán soluciones para dichas tareas y posteriormente se implementarán y testearán. Según los resultados de los ensayos, se corregirán errores, se buscarán y aplicarán soluciones alternativas y se volverá a ensayar. En caso de resultado positivo, se preparará el resultado para informar adecuadamente en la reunión, y recibir información de vuelta.

El desarrollo completo, en su descomposición en módulos, se irá subiendo a un repositorio en github[1] con nombre *SP*. El informe se subirá a un repositorio independiente[2]. También se hará uso del canal de YouTube *JdeRobot*[3] para subir vídeos de los resultados conseguidos.

2.4. Plan de trabajo

Para alcanzar el objetivo final, cubriendo los requisitos mínimos, se procederá a abordar el trabajo con la siguiente ordenación temporal de tareas:

- Enlace con el dron: Es el punto de partida del proyecto. Pasa por ser capaces de realizar un enlace radio entre un ordenador y el propio dron. Para esta tarea se hará uso del software libre provisto por *goebish* en su repositorio *nrf24_multipro*[4]. Una vez establecido el enlace es el momento de generar el software necesario en la FPGA de tierra para probar instrucciones simples hacia el dron.

- Diseño de electrónica a bordo: Para corregir el comportamiento en vuelo del dron será necesario que éste disponga de información relativa a su posición local. Para cubrir esta

[1] https://github.com/JdeRobot/FPGA-robotics/tree/master/Projects/Basic-Drone
[2] https://github.com/RoboticsLabURJC/2018-tfg-eloy-navarro
[3] https://www.youtube.com/channel/UCgmUgpircYAv_QhLQziHJOQ
[4] https://github.com/goebish/nrf24_multipro

tarea se instalarán en el dron sensores y sistemas de comunicación capaces de medir parámetros de posición y transmitirlos a la FPGA de la estación de tierra para ser procesados.

- Diseño de la estación de tierra: Para realizar ensayos de bucle cerrado será necesario tener disponible una electrónica en tierra capaz de recibir medidas de los sensores a bordo e instrucciones desde el PC. Deberá entonces poder procesarlas y transmitir la respuesta hacia el vehículo. Dicha respuesta dependerá tanto de las medidas de los sensores del dron como de las indicaciones recibidas desde el ordenador de mando.

- Cierre de bucles para cada eje y experimentación: Se diseñarán bucles de control independientes para cada eje en la FPGA de la estación de tierra. Esto permitirá la estabilización y manejo completo del dron. Tras aplicar cambios en cada controlador se pondrá a prueba el trabajo. Según el resultado se procederá con otro o se iterará sobre el mismo hasta corregir adecuadamente los posibles errores.

- Diseño de librerías de control para PC: Que faciliten el uso del dron y la repetitividad de los ensayos a través de abstraerse de la capa más baja del control del dron. Esto permite realizar ensayos en bucle abierto en trazados más complejos, mediante secuencias de instrucciones simples (despegue y aterrizaje, giros o desplazamientos) facilitarán las siguientes tareas.

Capítulo 3

Infraestructura utilizada

Para alcanzar los objetivos hemos utilizado un amplio conjunto de herramientas software y hardware. Estas han permitido realizar diseños software en distintos lenguajes, compilar, programar y ejecutar, en las plataformas disponibles. Posteriormente, mediante herramientas adicionales, se han podido simular y evaluar los resultados, permitiendo avanzar a través de tareas de depuración y corrección de errores. A continuación se describen dichas herramientas y su uso durante este trabajo.

3.1. Entornos de desarrollo

En este apartado, se comentan las herramientas software que permiten trabajar con código, cubriendo múltiples necesidades (edición, compilación, enlazado, etc...) en un solo paquete.

3.1.1. IceCube2

En su versión 2017.08.27940, es el entorno de desarrollo suministrado por Lattice para la programación de sus familias de pequeñas FPGAs. Integra un editor de texto plano, junto con un sintetizador para código Verilog 2001. También realiza las tareas de ubicación de elementos lógicos utilizados, trazado de conexiones en el array de la FPGA escogida y generación del fichero de programación "*.bin". Se ha utilizado durante todo el desarrollo sobre FPGA del trabajo. Con él se han realizado todas las tareas desde la compilación del proyecto completo hasta la generación del fichero de programación.

3.1.2. Quartus Prime

Herramienta de diseño, test, depuración y programación de software para FPGAs de Altera (Intel) versión 15.1.0 Build 185. Utilizado en este trabajo exclusivamente por su editor de texto con reconocimiento del leguaje Verilog 2001 e identificación de estructuras con color. La comodidad y versatilidad de su editor integrado, frente al entorno de Lattice, lo han convertido en mi elección para el desarrollo del código de cada módulo, previo a su compilación en el proyecto, tarea realizada con el IDE de Lattice.

3.1.3. Arduino IDE

Entorno de desarrollo integrado de Arduino versión 1.8.8 para procesadores Atmel compatibles. Adicionalmente al editor de texto, compilador, enlazador y programador, suministra algunas herramientas adicionales de utilidad: Instalador de librerías integrado. Se ha utilizado en el TFG para facilitar la instalación de paquetes de comunicación por puerto serie y manejo de dispositivos desde librerías en Github. Entre los incluidos, se hace uso en este trabajo de las librerías para puertos serie SPI, I2C y UART y librerías para manejo de sensores. Además incluye un monitor de puerto serie ya configurado para el dispositivo concreto a programar. Realiza la lectura por el mismo puerto USB utilizado para programar el dispositivo, ahorrando tiempo y disminuyendo el peso necesario en la electrónica a bordo para realizar tareas de diagnóstico. Este entorno se ha utilizado mayoritariamente para el desarrollo, depuración y programación de los programas ejecutados en los procesadores Atmel ubicados a bordo del dron (comunicación y sensores) y en la estación de tierra (comunicación por radio).

3.2. Herramienta de Simulación ModelSim

Herramienta para la simulación de lenguajes de descripción de hardware en general. Compatible con Verilog 2001, permite la comprobación de sintaxis del lenguaje, compilación e instanciación de múltiples módulos. Es una herramienta flexible en la creación de cronogramas y selección de señales a observar. Ofrece también completa configuración de resoluciones y tiempos de simulación, junto con la posibilidad de realizar análisis dinámicos del hardware definido. Se ha hecho uso en este TFG de la versión 10.4b, gracias a la cual se ha ahorrado tiempo a la

hora de realizar tareas de limpieza de código y pruebas fuera del vehículo, disminuyendo el tiempo de desarrollo y la cantidad de choques por errores de diseño en los controles de vuelo.

3.3. Herramientas de programación

Una vez generados y simulados los códigos, éstos deben ser programados en los distintos dispositivos que componen el sistema. La programación de los procesadores de Atmel se realiza a través del propio entorno de desarrollo comentado en el apartado 3.1.3, el resto de herramientas necesarias para programación se comentan a continuación.

3.3.1. FT_Prog

Herramienta distribuida por FTDI Chip para reconfigurar la memoria flash de los dispositivos de comunicación USB (versión 3.6.88.402), que permite modificar el comportamiento de cada canal de sus dispositivos; activando, desactivando o modificando su comportamiento para hacerlo acorde a distintos estándares. Se ha utilizado para configurar los dos canales de comunicación USB de la estación de tierra. Configurando el canal A para permitir la programación del dispositivo FPGA y el canal B para comunicar, FPGA con PC mediante USB.

3.3.2. Diamond Programmer

Se trata de la herramienta de programación distribuida por Lattice. Versión 3.10.0.111.2. Permite la programación de sus dispositivos FPGA mediante el uso de ficheros con formato ".bin" generados tras el trazado de la FPGA. Dicho fichero es generado, en este TFG, por la herramienta IceCube2 3.1.1 y posteriormente cargado con este software para programar la FPGA de la estación de tierra.

3.4. Herramientas de depuración

Para facilitar la detección y corrección de errores se ha utilizado una herramienta de análisis de señales digitales, compuesta de un elemento hardware y otro software. A continuación se describen ambos.

3.4.1. Logic Analizer

Tarjeta de adquisición de Saleae. Trabaja como analizador lógico de ocho canales, con muestreo hasta 24MSPS, permite la adquisición y análisis de señales de hasta 12MHz. Sus ocho canales son reconfigurables como señales de entrada y señales de disparo, facilitando la sincronía con tramas y eventos. Esta herramienta trabaja en conjunto con la interfaz de usuario "Logic". Sus ocho conexiones se han realizado en distintos puntos de la estación de tierra. Su función durante el desarrollo ha sido poder obtener las capturas más representativas posibles de lo que estaba ocurriendo en cada momento. Para ello, se han capturando tramas desde el PC de mando hacia la estación de tierra, medidas de los sensores desde el dron hacia la estación de tierra, señales internas de la FPGA y órdenes desde la estación de tierra al dron. Este conjunto de sondas, ha facilitado tanto la depuración en vuelo, como la corrección de distintos parámetros de los controladores. Sin ellas, hubiera resultado imposible avanzar adecuadamente entre las etapas de desarrollo del TFG.

3.4.2. Logic

Esta utilidad de Saleae es su interfaz para almacenamiento y visualización de señales, tomadas con sus herramientas hardware de análisis digital. En nuestro caso, hemos utilizado la versión 1.2.10 junto con el analizador lógico "Logic Analizer". Para el caso de un puerto serie, permite el análisis de la información de un protocolo concreto, decodificando los bits de cada canal y mostrando el resultado en relación al instante de llegada. Esto hace posible tareas de depuración, basadas en observar el efecto concreto de un cambio en el código. Se pueden nombrar y configurar distintos protocolos para cada canal, de esta manera se ha utilizado también para comprobar el comportamiento de puertos UART, SPI y señales internas de la FPGA, analizando los datos transferidos entre dispositivos.

3.5. Plataformas hardware

La arquitectura escogida para el desarrollo de este trabajo, requiere del uso de múltiples tarjetas con distintas capacidades. Estas trabajan juntas para cumplir un mismo objetivo. Tanto en la estación de tierra, como a bordo del vehículo, se han instalado varios elementos hardware

descritos a continuación.

3.5.1. Arduino Uno y Nano

Como procesadores auxiliares, se han utilizado tarjetas de desarrollo que integran un procesador de Atmel de 8 bits modelos ATMEGA8U2-MU (Arduino Uno en la Figura 3.1) y ATMEGA328P-AU (Arduino Nano en la Figura 3.2), un interfaz de UART a USB para comunicaciones, programación y alimentación, conectores hacia las GPIOs de los procesadores y un regulador de 5 a 3,3 voltios. Se pueden alimentar de manera externa sobre la entrada del regulador, o directamente mediante el USB. Se dispone de acceso para programación y comunicaciones a través del mismo USB en ambas tarjetas. La tarjeta "Nano" la hemos utilizado a bordo del dron por una cuestión de dimensión y peso, mientras que la tarjeta "Uno", significativamente más grande y pesada, se ha quedado en la estación de tierra, junto a la FPGA.

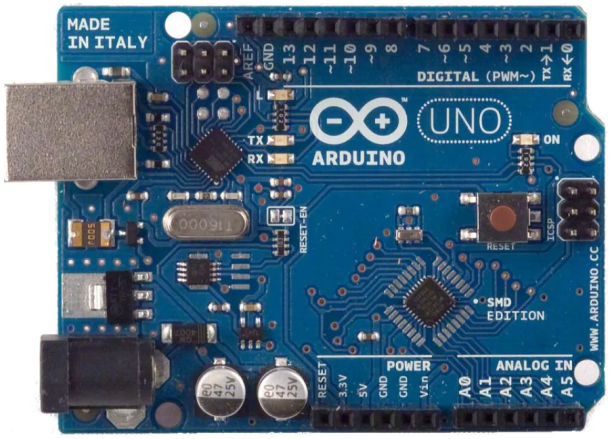

Figura 3.1: Tarjeta de desarrollo Arduino Uno

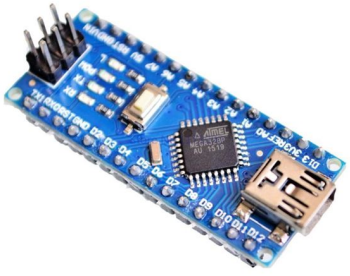

Figura 3.2: Tarjeta de desarrollo Arduino Nano

3.5.2. ICE40 UltraPlus Breakout Board

Tarjeta de desarrollo para la familia UltraPlus de los modelos ICE40 de Lattice (Figura 3.3). Incluye un regulador de tensión, elementos varios de interfaz humana; como Leds, interruptores, jumpers y conexiones de propósito general. También incluye un interfaz USB reprogramable de FTDI Chip y doble canal, FT232HL, con el canal A configurado para convertir de USB a SPI, ocupado en la programación y el canal B libre de uso.

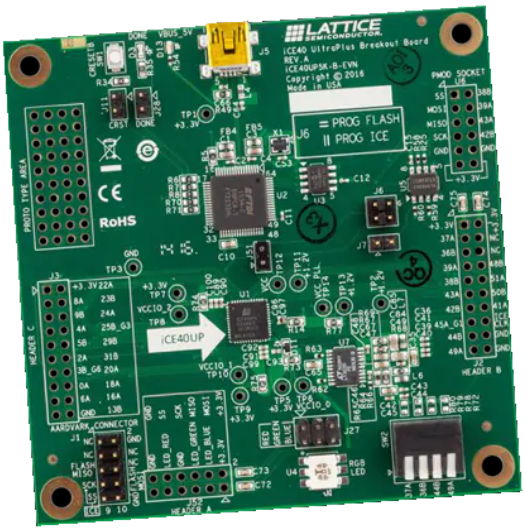

Figura 3.3: Tarjeta de desarrollo ICE40 de Lattice

Como procesador dispone de una FPGA ICE40UP5K-SG48 en su encapsulado QFN de 48 pines, con 5280 celdas lógicas, 120Kbits de RAM embebida y 1024kbits de RAM estática, una PLL y 8 bloques DSP (multiplicadores de 16 bits). Se trata de una FPGA basada en RAM, por tanto, dispone de una memoria interna de configuración basada en RAM (bancos de memoria que determinan el hardware que se construirá dentro de la FPGA) y una memoria flash externa (instalada en esta tarjeta). Esto permite:

- Configurar un programa volátil, directamente en la memoria RAM de configuración de la FPGA. Este sobrescribirá cualquier configuración existente y desaparecerá en caso de apagado.

- Almacenar un programa en memoria flash externa, el cual será cargado en la memoria de configuración RAM, durante el arranque.

Esta plataforma se ha usado como núcleo de la estación de tierra, conectándose con periféricos radio y el PC de mando.

3.5.3. Dron SYMA X5C

Dron cuadricóptero teledirigido fabricado por Syma. Escogido por ser un dron barato y ligero (100gr) para su tamaño (31cm de ancho y largo), con una estructura robusta. Idóneo para aprendizaje y experimentación, gracias a su importante resistencia mecánica. Su electrónica de recepción radio alcanza 50 metros en la banda de 2.4GHz, cuenta con cuatro canales de control, motores de continua y batería de litio. No dispone de sistemas adicional para la corrección del giro, además del propio giróscopo (como podría ser un magnetómetro). Además incluye varios elementos que han sido descartado en este TFG para aliviar peso. Por ejemplo la cámara con almacenamiento en SD-Card, elementos de iluminación y protecciones de aspas. La Figura 3.4 muestra su configuración original y en la Figura 4.4 se observa el resultado tras aligerar y añadir elementos adicionales.

Figura 3.4: Cuadricoptero comercial de bajo coste de Syma

3.5.4. NRF24L01

Este módulo integra el chip de mismo código, junto a condensadores de desacoplo necesarios para su utilización, una antena integrada en la propia PCB y pines de acceso a su interfaz SPI como se muestra en la Figura 3.5. Su utilización durante el trabajo ha facilitado la implementación radio, gracias a la integración en una pequeña PCB de todo lo necesario para

establecer la conexión. Ocupa la banda de 2.4GHz, con 125 posibles canales de 1 MHz de ancho de banda cada uno (disponibles para multiplexación en frecuencia). Al trabajar hasta los 3.6 voltios, es perfecto para su utilización junto con las tarjetas Arduino Uno y Nano, que integran reguladores a dicha tensión. Dispone de tasas de transferencia de hasta 2Mbps y alcances de hasta 30 metros.

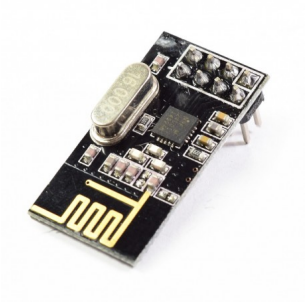

Figura 3.5: Módulo integrado NRF24L01 para enlace radio

3.5.5. Flow breakout board

Se trata de un módulo de BitCraze [4], que integra un sensor de medida de distancias por tiempo de vuelo VL53L0x, accesible por I2C, junto a un sensor de flujo óptico PMW3901, accesible por SPI (Figura 3.6). Incluye la óptica, electrónica periférica para hacerlos funcionar y un cabezal para las conexiones eléctricas. Funciona a 3.6 voltios, por lo que se alimenta directamente desde la salida del regulador de la tarjeta Nano, a bordo del dron. Esto también permite realizar el interfaz eléctrico directamente entre los procesadores y el módulo. Usamos este conjunto de sensores, ya que ofrece medidas del desplazamiento horizontal y vertical, en una sola tarjeta de peso y tamaño adecuados al vehículo escogido.

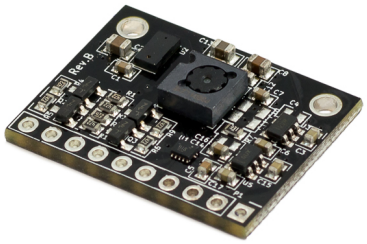

Figura 3.6: Módulo con sensores de medida de distancia y desplazamiento

Capítulo 4

Sistema con dron comercial estable y programable

A continuación se describe la arquitectura del sistema diseñado e implementado. Consta de tres módulos principales; PC, dron y estación de tierra. Estos módulos se descomponen en elementos adicionales, explicados más adelante en sus respectivas secciones. También se detallan los interfaces que comunican dichos módulos.

4.1. Diseño del sistema

El sistema creado en este TFG pretende cubrir los objetivos propuestos en el Capítulo 2 y hace uso de la infraestructura indicada en el Capítulo 3. Para realizar sus tareas, el sistema se compone de tres módulos principales y dos interfaces de alto nivel. El diseño elegido, junto con la relación entre dichos elementos, se muestra en la Figura 4.1.

Los tres módulos principales que lo componen son:

- Ordenador de mando: PC utilizado para enviar hacia la estación de tierra las instrucciones que comandarán el dron.

- Dron: Cuadricóptero que obedece las instrucciones de movimiento indicadas por el ordenador de mando. Va cargado de electrónica propia y de sensores adicionales que mejoran su estabilidad de vuelo.

- Estación de tierra: Actúa como centro neurálgico del sistema mediante el uso de una FPGA como núcleo. Presta atención a las instrucciones indicadas por el ordenador de mando y se encarga de que el vehículo escogido ejecute adecuadamente dichas instrucciones.

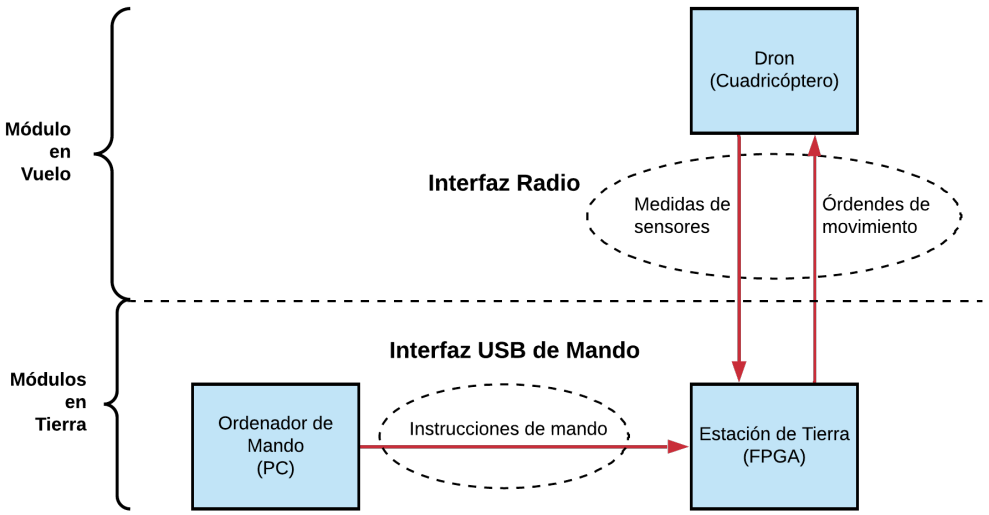

Figura 4.1: Diseño de sistema

Estos elementos se comunican entre sí gracias a dos interfaces externos (cuyas componentes se marcan en rojo en la Figura 4.1):

- Interfaz USB de mando: Conexión unidireccional encargada de portar el flujo de instrucciones desde el ordenador de mando hacia la estación de tierra, mediante USB 2.0. Contiene las instrucciones de movimiento que la estación de tierra deberá hacer efectivas.
- Interfaz radio: Enlace bidireccional que permite comunicar el dron con la estación de tierra y viceversa. Contiene las medidas de los sensores de a bordo y las órdenes de movimiento hacia el dron.

La arquitectura del sistema propuesto basa su funcionamiento en las siguientes premisas:

- La estación de tierra atiende los comandos que puedan llegar desde el ordenador de mando.
- El dron, cargado de sensores, transmitirá periódicamente medidas de posición hacia la estación de tierra.

- La estación de tierra, gracias a las medidas de los sensores recibidas desde el dron, es capaz de ubicar de manera local la posición del vehículo. Dicha ubicación será relativa al punto de despegue.
- La estación de tierra genera un vector de error de posición para el vehículo en vuelo mediante la diferencia entre las medidas de posición indicadas por los sensores del vehículo y las órdenes recibidas desde el ordenador de mando.
- La estación de tierra transmitirá al vehículo en vuelo órdenes de movimiento para corregir el posible error de posición existente.
- El dron realizará cambios en su posición en base a las órdenes de movimiento recibidas.

De manera general, este conjunto de afirmaciones es llevado a cabo, a través del flujo de control expresado en la Figura 4.2.

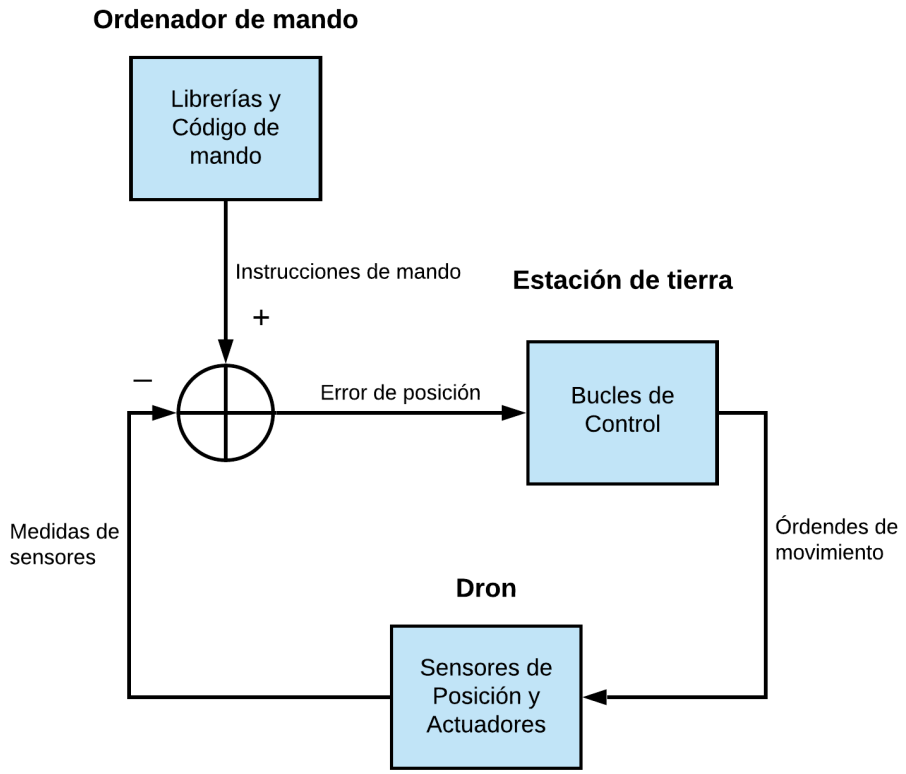

Figura 4.2: Flujo de control general del sistema

4.2. Dron enriquecido

Se trata de la plataforma utilizada como vehículo volador. Para el desarrollo de este TFG se ha optado por el X5C de Syma (descrito en la sección 3.5.3), sobre el que se han realizado modificaciones en su estructura y añadidos a sus capacidades originales.
Estos añadidos le permiten:

- Medir distancia al suelo, para conocer su altura.
- Medir su desplazamiento horizontal, para conocer su ubicación relativa al punto de despegue.
- Comunicarse por radio con la estación de tierra, para hacer llegar su ubicación a la FPGA allí ubicada.

A continuación se comenta la arquitectura que permite la comunicación de estas medidas de posición.

4.2.1. Arquitectura del dron

Para controlar el vuelo del dron desde la estación de tierra es necesario nutrirla de información sobre el vehículo, que este no tenía a su disposición tal y como lo vende el fabricante. Resulta imprescindible tanto que la FPGA de la estación conozca la posición del dron, como que este sea capaz de moverse según se lo indiquen en cada instante. Esto último ya se encontraba presente entre las capacidades originales del dron (módulos verdes de la Figura 4.3). Se ha instalado electrónica adicional en el vehículo que le permite cubrir la primera necesidad, conocer su ubicación.
Cabe destacar que no existen interfaces que comuniquen de manera directa la electrónica propia del dron con electrónica de sensores instalada para este TFG. Esto es debido a que se prefirió aislar dicha electrónica de la instalada originalmente en el dron, por problemas con la alimentación nativa (caídas de tensión abruptas durante aceleraciones). Además, no existe la necesidad específica de comunicar dichos módulos directamente (quedan indirectamente comunicados entre sí a través de la información enviada a la estación de tierra, procesada por esta y recibida de vuelta en el dron). Dicha comunicación directa podría darse para arquitecturas de sistema

diferentes, por ejemplo, aquellas propuestas más adelante en el apartado 6.2.1.
La arquitectura final del dron se muestra en la Figura 4.3. Los módulos azules indican los elementos instalados adicionalmente sobre el vehículo.

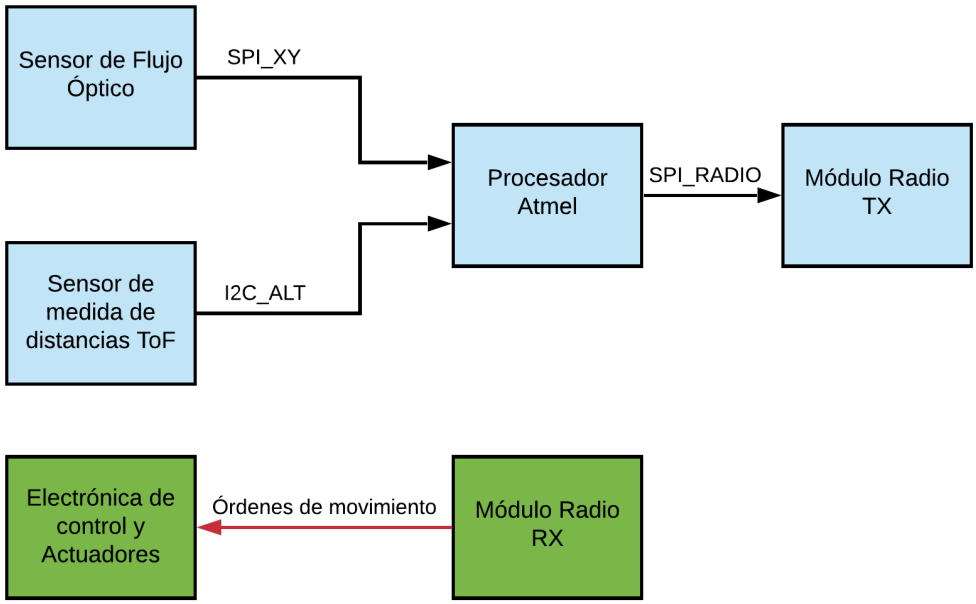

Figura 4.3: Arquitectura del dron

Esta arquitectura hace uso de múltiples elementos como sensores, actuadores, procesadores y radios. La radio de recepción, electrónica de control y actuadores, mostrados en la rama inferior de la Figura 4.3 (en verde) son los elementos propios del vehículo. Estos le permiten obedecer órdenes de movimiento, dirigidas a él a través del interfaz radio de subida (Apartado 4.6.2), detallado más adelante en este informe. Los módulos de la rama superior de la arquitectura (en azul) son los añadidos durante este TFG y que dotan al vehículo, de la capacidad de conocer su posición en tiempo real y transmitírsela a la FPGA de la estación de tierra. Se detallan a continuación.

4.2.2. Módulos del dron

El vehículo se equipa con los siguientes módulos adicionales, utilizados para conocer su propia ubicación en tiempo real.

- Módulo 1: Procesador Atmel.

- Módulo 2: Tarjeta de sensores: Sensor de flujo óptico y sensor de medida de distancias por tiempo de vuelo, integrados en una misma tarjeta.
- Módulo 3: Transmisor radio de enlace con la FPGA.

La instalación final de estos módulos sobre el X5C se muestra en la Figura 4.4. Se alimentan desde un pack de batería de 3.6V de Litio, independiente de la electrónica de control propia del vehículo. Esto es debido a que a pesar del peso adicional que supone una batería separada, los drivers de motores del dron exigen picos de corriente considerables a las baterías. Estas, que no siempre tienen capacidad de descarga suficiente debido al diseño precario del dron, caen en tensión. Estas caídas llegaban a producir apagados eventuales en la electrónica de los sensores, lo que dificulta el control, volviendo el sistema inestable en la mayor parte de los casos. Por este motivo, se optó por un conjunto separado de baterías para alimentar los tres módulos adicionales.

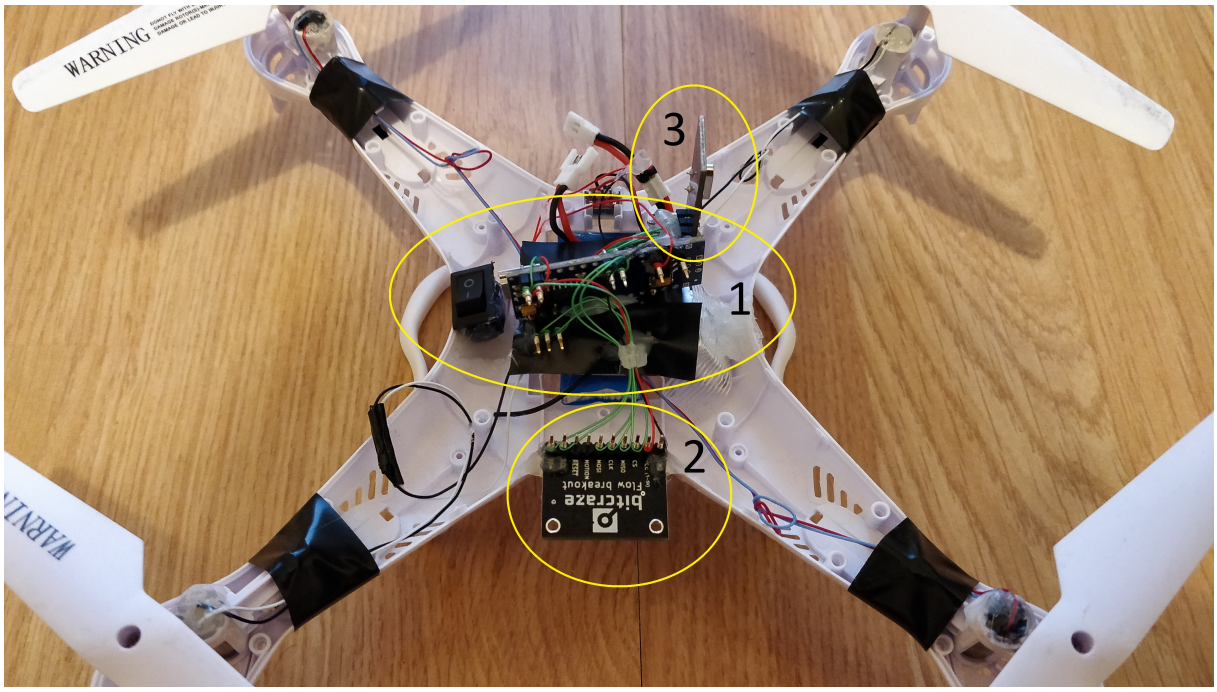

Figura 4.4: Módulos 1, 2 y 3 instalados adicionalmente

Módulo 1: Procesador

Este módulo se encarga de inicializar los dos sensores, altura y desplazamiento, y realizar lecturas de sus medidas por los interfaces I2C_Alt y SPI_XY respectivamente. Una vez tomadas

las medidas, se realizan ciertos procesos sobre los datos. El sensor de medida de altura por tiempo de vuelo (ToF) [26] sufre errores por debajo de los 3 cm, por tanto este módulo se encarga de filtrar dichos valores erróneos, entregando una medida ficticia de 3 cm para toda altura igual o inferior. Esto provoca que el dron, una vez aterrizado, siga indicando una altura de 3 cm, lo cual no supone un problema ya que llegado el momento del aterrizaje, la altura ficticia de 3 cm medidos, ayuda a descargar el bucle integral, apagando los motores por completo. También realiza las lecturas del sensor de velocidad de desplazamiento horizontal (Flow). El procesador se encarga de convertir éstas en medidas de posición. Para ello integra en el tiempo las medidas de velocidad de desplazamiento, leídas del interfaz SPI_XY. Además, el sensor de flujo no mide correctamente para alturas inferiores a 4 cm, por tanto, para un rango inferior a éste no se modifican las medidas de posición.
Una vez filtradas e integradas, el procesador entrama dichas medidas y las envía hacia la FPGA en tierra, a través de la radio del módulo 3, haciendo uso del interfaz SPI_Radio.

Módulo 2: Tarjeta de Sensores

Este módulo se compone de una sola tarjeta que integra dos sensores. Juntos ofrecen lo necesario para llegar a conocer la ubicación en tiempo real del dron relativa al punto de despegue. Incluye electrónica auxiliar y conexiones, además de los módulos de medida de distancia por tiempo de vuelo (Sensor ToF) y medida de velocidad de desplazamiento en dos ejes (Sensor de flujo).

- Sensor ToF: Se trata del sensor VL53L0x, el cual mide la distancia entre su encapsulado y un objeto enfrentado a él, a través de la medida del tiempo de vuelo de una señal laser. El procesador del módulo 1 accede a sus medidas a través del interfaz I2C_Alt, un puerto serie I2C. Tiene un rango de trabajo de entre tres centímetros y dos metros. Este sensor se apunta hacia el suelo, desde su soporte en el dron, con la intención de medir la distancia al suelo desde el punto medio del dron. De esta manera una vez aterrizado, la distancia de medida, es ligeramente superior a 3cm, lo cual minimiza la necesidad de filtrado en el procesador (esta última se mantiene igualmente, ya que los apoyos del dron son flexibles, y el sensor tiene tolerancias de medida). Este sensor tiene un retardo de medida de hasta 30ms, el cual limita en cierta medida la estabilidad del bucle de control de altura ejecutado en la FPGA.

- Sensor de flujo: Como sensor de desplazamiento horizontal se hace uso del PMW3901 [23]. Este sensor de movimiento tiene una óptica diseñada para enfocar a distancias superiores a 8cm. Aunque funciona bien hasta los 4cm si se dispone de buena iluminación. Sus medidas son filtradas como se indica en el proceso del módulo 1 para mitigar errores de medida. Entrega medidas de velocidad de desplazamiento en dos ejes, para el plano captado por su óptica. En el montaje, el sensor se apunta hacia el suelo, desde el mismo soporte que para el sensor de tiempo de vuelo. De esta manera se conocerá la velocidad de desplazamiento horizontal del dron respecto del suelo. Es el módulo 1 el encargado de convertir estas medidas de velocidad de desplazamiento en posición relativa horizontal. La orientación de medida del sensor se muestra en la Figura 4.5. Desde el punto de vista del vehículo, el avance es acorde al eje *-X* y el desplazamiento a izquierdas *+Y*. El sensor tiene un puerto de comunicaciones SPI para inicialización y lectura, sus medidas son leídas por el procesador del módulo 1 a través del interfaz SPI_XY.

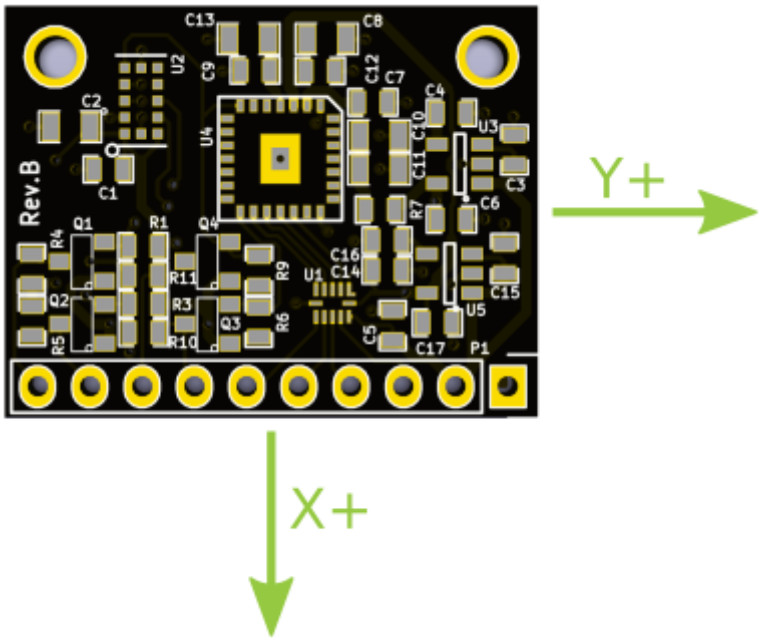

Figura 4.5: Orientación de eje horizontales en Módulo 2

Módulo 3: Transmisor de bajada hacia FPGA

El módulo se compone de una radio NRF24L01 conectada por puerto SPI al procesador del módulo 1, a través del interfaz SPI_Radio. Esta interfaz permite tanto la configuración del dispositivo, como su uso para transmitir tramas. El integrado NRF24L01 se configura como transmisor para enviar las medidas de posición, ya procesadas y entramadas por el procesador del módulo 1, hacia la estación de tierra. Dicha transmisión ocurre de manera periódica, después de realizar cada medida de posición.

4.2.3. Interfaces internos

En este apartado se comentan los interfaces de la rama superior de la arquitectura definida anteriormente en 4.2.1, pintados de negro en la Figura 4.3. Estos se limitan a comunicar información dentro de la electrónica adicional instalada en el vehículo.

- I2C_Alt: Comunica por puerto serie I2C el sensor de medida de altura VL53L0x del módulo 2 con el procesador del módulo 1. Porta las medidas de distancia entre la zona media del dron y el suelo bajo él, consideradas como la altura del dron.
- SPI_XY: Comunica por puerto serie SPI el sensor de flujo PMW3901 del módulo 2 con el procesador del módulo 1. Porta las medidas de velocidad de desplazamiento sobre el plano horizontal en ambos ejes, derivadas del desplazamiento del suelo bajo el sensor.
- SPI_Radio: Puerto serie SPI utilizado para la configuración inicial de la radio NRF24L01, y para la indicación de tramas a enviar. Estas contendrán las medidas de posición ya procesadas. La configuración inicial se realiza de la misma manera que para el módulo receptor en la estación de tierra. En la cual se esperan recibir las tramas enviadas desde esta electrónica a bordo del vehículo, por tanto, la configuración de ambas y el formato de las tramas, deberá ser compatible, según lo especificado en el apartado 4.6.

4.2.4. Interfaces externos

El vehículo aéreo solo se comunica con el resto del sistema mediante dos enlaces radio (comentados en el apartado 4.6), a través del *Interfaz Radio* de la Figura 4.1, que lo conecta contra la estación de tierra. Esto permite llegar las medidas de posición a la FPGA y las órdenes de

movimiento de vuelta hacia el dron.
Adicionalmente, el dron dispone de conexión USB, exclusiva para reprogramación del procesador y extracción de medidas por puerto serie en tareas de depuración. Por ser sólo auxiliar y de desarrollo no se muestra en la arquitectura del sistema del apartado 4.1.

4.2.5. Software a bordo del dron

El software desarrollado para el dron se ejecuta en el procesador Atmel del módulo 1. A través de los interfaces internos (apartado 4.2.3) se encarga de configurar radio y sensores, para posteriormente realizar una medida de cada sensor y transmitirlas por radio. El objetivo es mantener continuamente informada a la estación de tierra de la posición del vehículo. Este software se diseñó para cubrir ciertos requisitos como: Consumir la menor cantidad de energía posible, realizar su función rápidamente (permite aumentar el ancho de banda), entreteniéndose lo menor posible en tareas secundarias (intentando minimizar latencias), y transmitir cada medida rápida e íntegramente. Para cubrir estos objetivos el software se ha diseñado de manera sencilla. Se comienza iniciando los sensores y la radio, sin modificar sus parámetros durante la ejecución y se prepara una estructura de datos en la que quepa la trama entera a transmitir por radio, en grupos de dos bytes:

```
struct TxFrame{
  int16_t H_disp_front; // Frontal displacement counter
  int16_t H_disp_side;  // Side displacement counter
  int16_t altitude;     // Altitude counter
};
```

Esto permite realizar una medida de cada sensor, filtrarla y almacenar el resultado sobre su respectiva ubicación en la variable tipo *TxFrame*. Ésta entonces puede transmitirse inmediatamente por radio, minimizando retardos e iterando esta lógica en bucle, hasta el apagado de la electrónica.

4.3. Estación de tierra basada en FPGA

Este módulo hace posible que el vehículo escogido obedezca las instrucciones ejecutadas en el ordenador de mando (apartado 4.4) y materializa los algoritmos de control que dan estabilidad al vuelo del dron. Su computador principal es una FPGA y está en comunicación tanto con el ordenador de mando como con el dron volador. Para el ajuste de los distintos parámetros que intervienen en el control del vuelo del vehículo resulta fundamental poder realizar capturas de información en tiempo real. Por tanto, se instala un conjunto de electrónica que otorgue suficientes capacidades a la estación de tierra:

- Atender las tramas de mando enviadas desde PC por USB.
- Atender las tramas con medidas de posición enviadas desde el dron por radio.
- Ejecutar algoritmos de control para generar órdenes de movimiento que estabilicen y manejen el vehículo.
- Transmitir por radio órdenes de movimiento hacia el dron.
- Realizar capturas de valores internos de la FPGA y tramas de comunicaciones, enviándolos después hacia el PC para su análisis.

Para cubrir este conjunto de tareas, se ha diseñado la siguiente arquitectura.

4.3.1. Arquitectura de la estación

La estación de tierra se estructura de manera que sea capaz de realizar las tareas comentadas. Hace uso de una FPGA de Lattice ubicada en la tarjeta de desarrollo escogida (sección 3.5.2), dos módulos radio (sección 3.5.4) y un módulo de adquisición para depurar señales digitales (sección 3.4.1). Estos elementos le permiten no solo dirigir el dron, sino además facilitar tareas de ensayo y error, permitiendo ajustar los controladores. En la Figura 4.6 se exponen los módulos que conforman la estación y relaciones entre ellos, que intervienen para cumplir dichas tareas.

A continuación se explican los módulos que hacen posible los cometidos desempeñados por la estación de tierra.

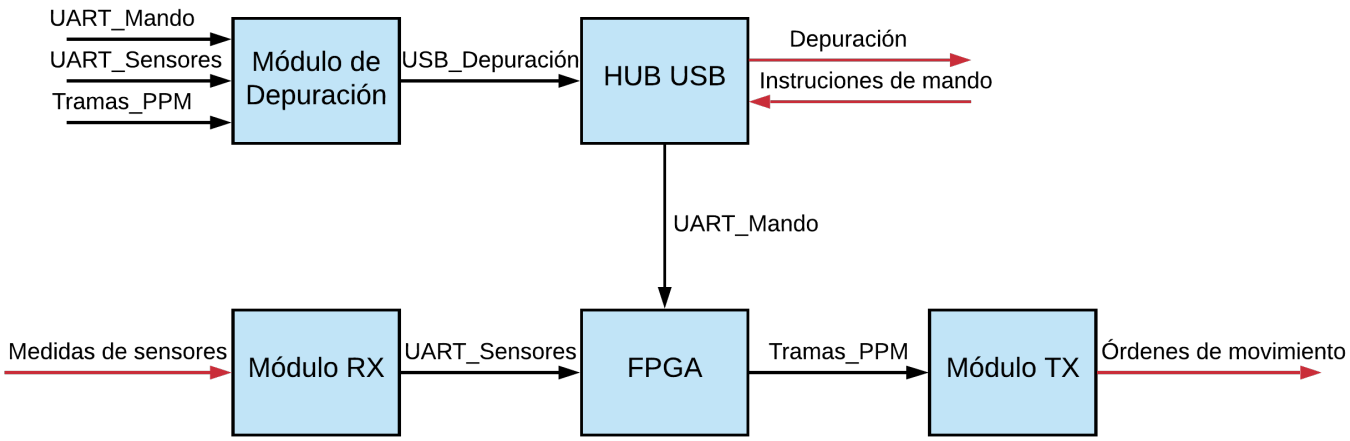

Figura 4.6: Arquitectura de la estación de tierra

4.3.2. Módulos de la estación

Para el control y depuración del vuelo del dron, la estación se equipa con los siguientes módulos:

- Módulo 1: Concentrador USB (HUB USB) como enlace hacia el PC.
- Módulo 2: Tarjeta de procesamiento basada en FPGA (FPGA) para ejecución de algoritmos de control.
- Módulo 3: Receptor radio (Módulo RX) de las medidas de posición del dron.
- Módulo 4: Transmisor radio (Módulo TX) de las órdenes de movimiento hacia el dron.
- Módulo 5: Módulo de Depuración para adquisición de señales digitales.

Cada módulo de la estación se compone de una o varias tarjetas mostradas en el montaje de la Figura 4.7. Cada módulo ejecuta con una función concreta y necesaria para cumplir algún sub-objetivo, permitir a las demás cumplir con el suyo, o facilitar las tareas de depuración.

A continuación se describen los distintos módulos que componen la estación de tierra, enumerados en la Figura 4.7:

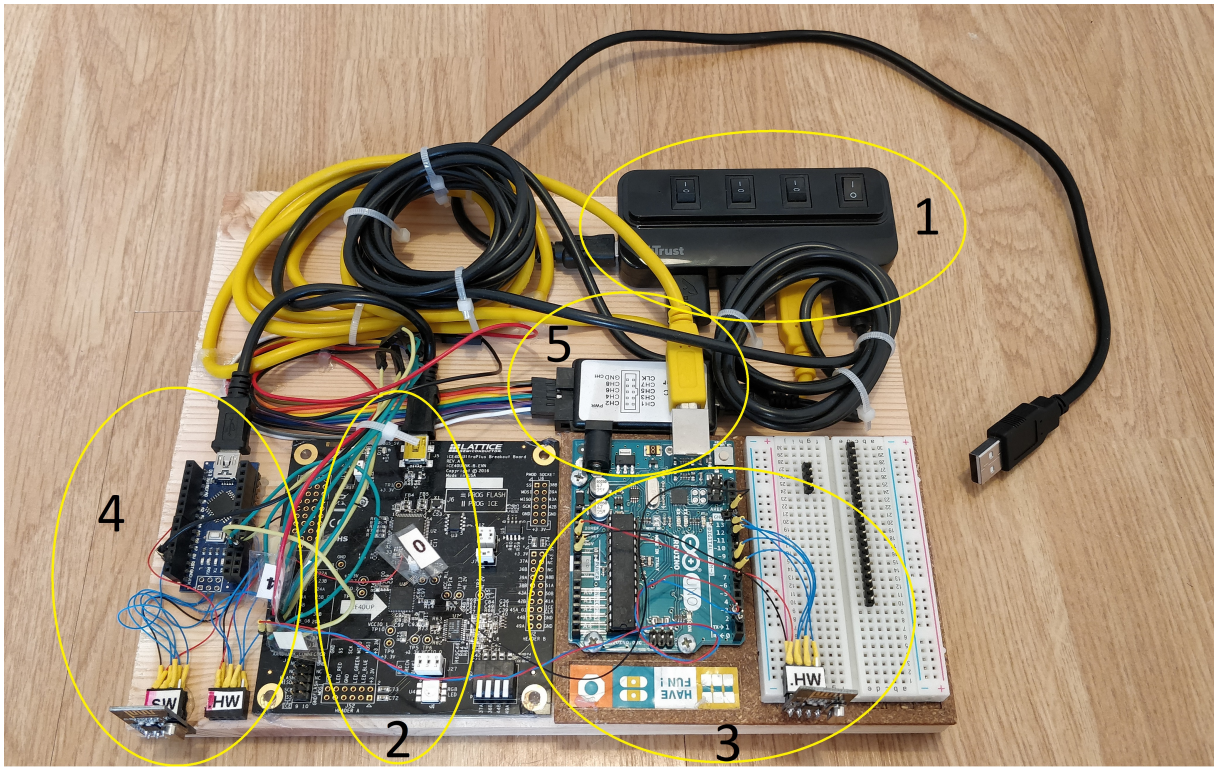

Figura 4.7: Montaje de la estación de tierra

Módulo 1: Concentrador USB

Centraliza las comunicaciones USB, permitiendo tener un único puerto ocupado en el PC y poder acceder desde él a la programación individual de los 3 módulos reprogramables: módulo transmisor, módulo receptor y FPGA. Además se puede acceder a las sondas de depuración, y alimentar simultáneamente todos los sistemas. También permite el apagado individual de cada módulo, muy útil para restablecer la comunicación con el dron reiniciando el módulo transmisor, sin eliminar la configuración en RAM existente en la FPGA.

Módulo 2: Procesador basado en FPGA

Se trata de la tarjeta que integra la FPGA utilizada como procesador principal del sistema. Incluye la Lattice comentada en el Apartado 3.5.2 junto a demás elementos periféricos y auxiliares. Se conecta con el ordenador de mando mediante el interfaz USB integrado en el chip FT232HL, cuyo canal B se encuentra libre. Este convierte de USB a UART, para comunicar las instrucciones de mando desde el PC hacia la FPGA.

La FPGA de este módulo es la encargada de atender las instrucciones de mando y las medidas

de los sensores de a bordo del dron. Además se encarga de ejecutar los bucles de control, generando las órdenes de movimiento que corregirán la trayectoria del vehículo. Estas las modula mediante modulación por posición de pulso, sobre el interfaz Tramas_PPM, hacia el módulo transmisor. Más adelante se describen los algoritmos de control utilizados (Apartado 4.3.5).

Módulo 3: Receptor radio de medidas de posición

Se compone de una radio NRF24L01 conectada por puerto SPI a un procesador de Atmel integrado en la tarjeta Arduino Uno. La radio se alimenta desde la tarjeta del procesador, el cual se alimenta a su vez por USB. También se tiene acceso por dicho USB a la reprogramación del procesador y a su salida de puerto serie para depuración.
Este módulo se encarga de recibir las medidas de posición desde los sensores a bordo del dron. Configura la radio como receptora y se mantiene atento por *polling*, a la espera de recibir nuevas medidas de posición desde el vehículo. Una vez recibidas, las entrama y las dirige (Byte a Byte mediante protocolo UART) hacia la FPGA del módulo 2 mediante la interfaz UART_Sensores del Apartado 4.3.3.

Módulo 4: Transmisor radio de órdenes de movimiento

Este módulo se encarga de demodular el interfaz Tramas_PPM, contenedor de las órdenes de movimiento indicadas por la FPGA. Convierte dicha modulación en una transmisión radio hacia el dron. Consta de un procesador de Atmel integrado en una tarjeta Arduino Nano, junto con una radio NRF24L01. La tarjeta Arduino Nano incluye también regulador de tensión e interfaz USB para cubrir las mismas tareas que en el caso del módulo 3: alimentación, depuración y programación. El procesador configura la radio como transmisor y la enlaza con la electrónica original del dron (Parte inferior de la arquitectura de la Figura 4.3). Tras el enlace comienza a transmitir periódicamente hacia el dron las órdenes de movimiento demoduladas de la trama PPM recibida desde la FPGA.

Módulo 5: Depuración

Se trata de un sistema de adquisición digital diseñado por Saleae, para ser usado como analizador lógico. Dispone de una conexión de masa como referencia y ocho canales. Estos se conectan de tal manera que se tiene acceso a la información de los interfaces: UART_Mando

(Órdenes de movimiento recibidas desde el ordenador), UART_Sensorses (Medidas de posición recibidas desde la electrónica de sensorización del dron), Tramas_PPM (tramas con modulación PPM generadas por la FPGA, encapsulando los resultados de sus bucles de control) y algunas señales internas de la FPGA. De esta manera se puede depurar el sistema completo una vez en funcionamiento. Ya que el sistema de adquisición de Saleae permite capturas predefinidas en tiempo, se configuran adquisiciones de 5 o 10 segundos y se asocia el disparo de captura a la recepción de una nueva trama desde el ordenador de mando. De esta manera se realizan capturas de todos los eventos que se producen en el entorno de la orden enviada. Esto ha hecho posible gran parte de la depuración de errores y fallos en la estabilidad del sistema o en su ejecución, junto con el ajuste de parámetros PID.

4.3.3. Interfaces internos

Para comunicar información entre los módulos propios a la electrónica de la estación de tierra se han definido un conjunto de interfaces mostrados en la Figura 4.6. Estos son:

- UART_Mando: Comunica el USB del ordenador de mando con la FPGA (mediante el intercambiador de protocolo USB a UART, FT23HL). Una vez convertido a protocolo UART, se trata de una interfaz asíncrona sin paridad ni control de flujo, con 10 bits por byte y una tasa de transferencia de 500Kbps. Aquí se incluyen las instrucciones ejecutadas en el PC, que comandarán el vuelo del vehículo.

- UART_Sensores: La segunda de las interfaces asíncronas utilizadas en la estación de tierra, hace uso de una UART configurada idénticamente a la utilizada en el interfaz *UART_Mando*. Esta porta las medidas de posición recibidas desde el dron, por el módulo receptor, con destino en la FPGA.

- Tramas_PPM: Esta interfaz comunica las órdenes de movimiento enviadas por la FPGA, con el módulo transmisor (el cual se hará responsable de hacerlas llegar a su destino final, el dron). Para ello hace uso de un único hilo con modulación por posición de pulso. En una única trama de esta interfaz se encapsulan los cuatro canales que dirigirán el movimiento del dron.

- USB_Depuración: Se trata del USB que conecta el módulo de adquisición de señales

digitales de Logic con el PC. Permite realizar los volcados de su memoria interna para poder realizar posteriores tareas de depuración.

4.3.4. Interfaces externos

La estación de tierra se comunica con el resto del sistema mediante tres vías mostradas en la Figura 4.1. Dos interfaces radio con el dron, uno de subida (*Órdenes de movimiento*) y uno de bajada (*Medidas de sensores*), y un interfaz USB para comandos de control (*Instrucciones de mando*).
Paralelo a estos, existe un segundo interfaz USB desde la estación de tierra hacia el PC, utilizado exclusivamente para tareas de depuración (no presente en la Figura 4.1). Este segundo USB se comunica con la aplicación de análisis de señales digitales de Saleae, *Logic* (3.4.2).

4.3.5. Algoritmos de control sobre la FPGA

La lógica construida en la FPGA se encarga de procesar información sobre la posición actual y la posición deseada para el vehículo, generando órdenes de movimiento que dirijan el dron hacia la posición deseada. Para ello la arquitectura software diseñada se vale de tres puertos de comunicaciones (dos de entrada y uno de salida) y cuatro líneas de procesado paralelas (una para cada grado de libertad del vehículo). La relación entre estos elementos se muestra en la arquitectura de la Figura 4.8.
El conocimiento de la FPGA sobre la posición actual del vehículo procede de las medidas de posición recibidas por radio desde el propio dron y la posición deseada proviene del ordenador de mando. Ambos interfaces internos de la estación de tierra (UART_Sensores y UART_Mando) se describen en el Apartado 4.3.3.

A continuación se describe brevemente la lógica ejecutada dentro de la FPGA. Las medidas de posición actual del dron son recibidas por el módulo de *SINCRONIA Y UART TRAMAS DRON*, el cual las entrega byte a byte al módulo *DECODIFICADOR TRAMAS DRON*. Este se encarga de separar las medidas de altura, avance y desplazamiento lateral, entregándolas a los PIDs correspondientes. Estos tres PIDs requieren también de una consigna para funcionar, es decir, el valor que se desea conseguir en cada caso. Dichas consignas serán la indicación de po-

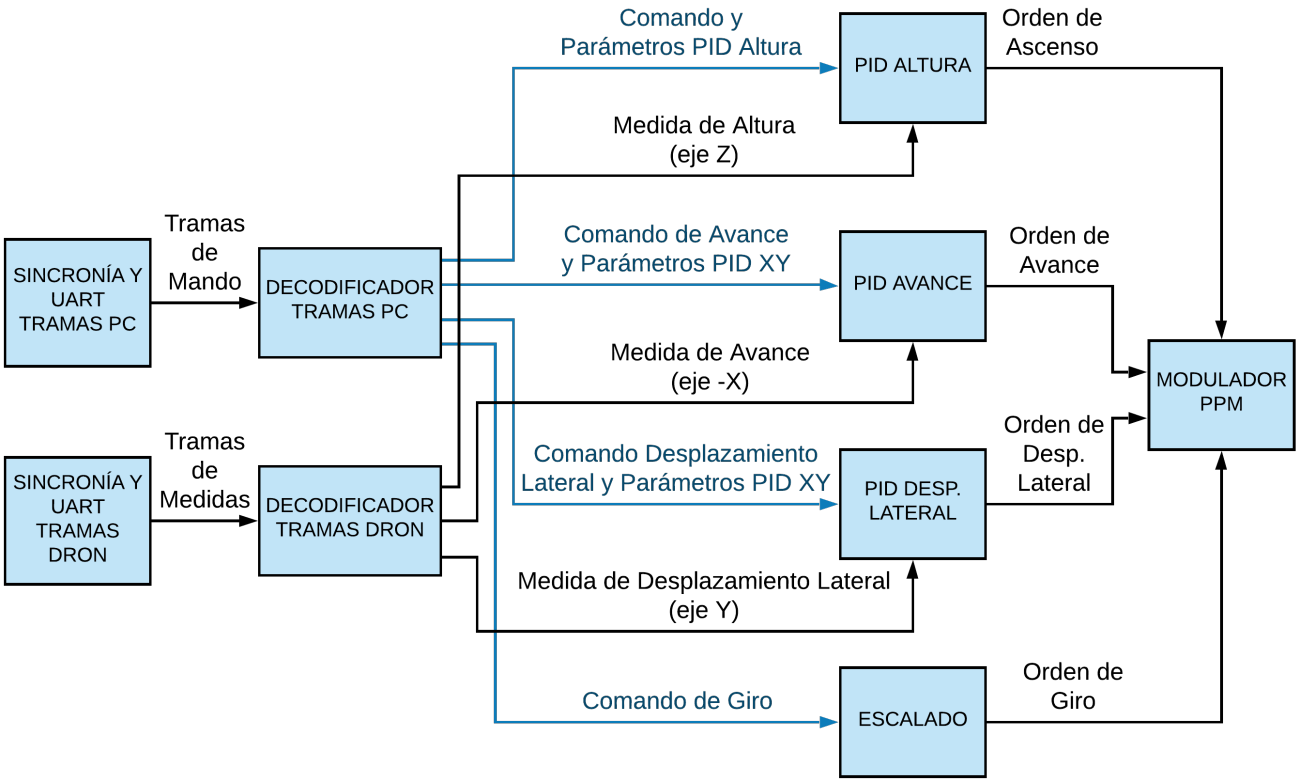

Figura 4.8: Arquitectura del software programado en la FPGA

sición deseada para cada grado de libertad controlado en el vehículo. Estas vienen indicadas por el ordenador de mando, mediante tramas de mando, que viajan a través del módulo *SINCRONIA Y UART TRAMAS PC*, hasta el *DECODIFICADOR TRAMAS PC*. El decodificador separa las consignas y se las comunica a cada módulo PID pertinente. Además, comunica a cada PID sus parámetros Kp, Ki y Kd (también recibidos en la trama de mando). Así cada PID se configurará previo a la ejecución del control. Como último elemento incluido en las tramas de mando está el comando de giro, que es enviado al módulo *ESCALADO*. Éste ajusta el valor del comando de giro recibido al rango dinámico esperado por el *MODULADOR PPM*.

En este punto, los tres controladores PID disponen de lo necesario para realizar su tarea (parámetros, consignas y medidas). El resultado de su ejecución genera las órdenes de movimiento, independientes para cada eje, que son recibidas por el *MODULADOR PPM*, junto con el comando de giro ya escalado. El modulador entrama las cuatro órdenes recibidas en una única trama y la modula mediante posición de pulso, enviándola hacia el transmisor (Módulo TX de la arquitectura 4.3.1).

El ancho de banda del sistema se determina por el elemento más lento en realizar su tarea, dentro de la cadena completa del sistema. Siendo éste las medidas de los sensores. Los módulos aquí descritos están diseñados para realizar su función rápidamente y quedarse a la espera por el siguiente conjunto de tramas entrantes. El cierre del bucle, desde la medida de los sensores hasta la actuación sobre los motores, se realiza a una velocidad de 30.3 Hercios. Esto determina a su vez los tiempo de integración y derivación para los bucles PID quedando estos en 33 milisegundos. El resto de parámetros PID, son recibidos por puerto serie, como se ha comentado. Esta versatilidad en la configuración de los parámetros de control permite al sistema tener un tipo de control concreto para cada fase del vuelo. Por ejemplo, los controladores no se comportarán igual durante el despegue que durante el vuelo o aterrizaje.

4.4. Ordenador de mando

Se trata del PC de la Figura 4.1. Dicho hardware se encarga de ejecutar la aplicación en código Python (Figura 4.9) que dirigirá el vuelo del dron. La ejecución de dicho código, implica que el PC comunicará por USB, hacia la estación de tierra, las órdenes que deberá obedecer el vehículo controlado. Para el caso de este trabajo, el cuadricóptero de Syma.

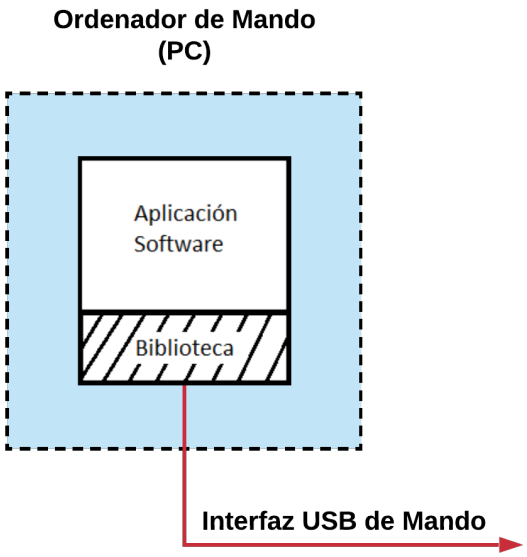

Figura 4.9: Estructura del ordendador de mando

El objetivo de este módulo es permitir el control del vehículo desde PC, ofreciendo la menor cantidad de limitaciones posibles en cuanto a su manejo y configuración. Para ello, el módulo contará con las funciones :

- Ejecución de instrucciones en Python de la aplicación robótica.
- Comunicación hacia la estación de tierra a través de un puerto USB.

En el ordenador se va a ejecutar la aplicación robótica (en Python) que gobierna el comportamiento del dron. Al usuario se le ofrece una biblioteca en Python que permite el control de los movimientos del dron. La implementación de esa biblioteca se comunica vía USB con la estación de tierra. Adicionalmente la biblioteca incluye también funciones para la configuración de los algoritmos de control PID que materializan el control de bajo nivel
Las tramas transmitidas por USB incluirán información de dos elementos:

- Controles sobre el vehículo: órdenes de control de los cuatro grados de libertad del dron (cabeceo, alabeo, guiñada y altitud), con el objetivo de ofrecer el máximo nivel de manejo.
- Controles sobre la estación de tierra: Valores de los parámetros de configuración de los bucles de control PID ejecutados en la estación. Esto permite su reconfiguración incluso en tiempo de ejecución.

4.4.1. Interfaz externo

De manera funcional, este módulo hace uso únicamente de un interfaz con el sistema. El comentado *Interfaz USB de Mando*, dirigido contra la estación de tierra y mostrado en la Figura 4.1, el cual contiene las instrucciones de mando enviadas hacia la estación de tierra.
Como se comentó en el Apartado 4.3.4 también existe un segundo interfaz USB desde la estación de tierra hacia el PC, utilizado exclusivamente para tareas de depuración (no presente en 4.1). Este segundo USB comunica al PC con la herramienta de depuración *Logic Analizer* (3.4.1) ubicada en la estación de tierra, cuyo uso se explica en el Apartado 4.3.2.

4.4.2. Librería Python

El software ejecutado en python en el ordenador de mando es el encargado de enviar las instrucciones de mando hacia la estación de tierra. El código fuente del usuario no lo hace

directamente sino que se apoya en una biblioteca desarrollada en este TFG. De este modo el usuario puede pensar el comportamiento del dron en términos de CONTROL EN POSICIÓN, más abstractos que la implementación. Las funciones de esta biblioteca permiten modificar tanto los parámetros de los bucles PID, como sus consignas de posición, además de inicializar el enlace USB con la UART receptora en la FPGA.

`setPIDValues`	Inicialización de valores de parámetros PID y ajuste de giro.
`setcontrols`	Permite el control directo de los 4 grados de libertad disponibles en el vehículo.
`settrace`	Dirige una trayectoria progresiva entre la posición actual y el punto indicado.
`setcircle`	Dirige una trayectoria progresiva en forma de arco entre la posición actual y el ángulo indicado.
`takeoff`	Arranca los motores y realiza el despegue hasta la altura indicada.
`landing`	Disminuye la altura progresivamente hasta el contacto con el suelo y apagado de motores.

A través de la ejecución de la función `setPIDValues` se cargan los valores de las constantes para los PIDs vertical (altura) y horizontales (cabeceo y alabeo), junto al offset para la corrección del error de giro (guiñada).

def `setPIDValues(alt_kp, alt_ki, alt_kd, xy_kp, xy_ki, xy_kd, offGiroDI):`

- Los parámetros: `alt_kp`, `alt_ki` y `alt_kd`, controlan las constantes, proporcional, integral y derivativa respectivamente, del bucle de control PID de altura.
- Los parámetros: `xy_kp`, `xy_ki`, `xy_kd`, se aplican a las constantes, proporcional, integral y derivativa respectivamente, de ambos bucles de control PID horizontales.
- El parámetro `offGiroDI` es el encargado de permitir corregir el error, si lo hubiere, en el control del giro sobre sí mismo del dron.

Tras al menos una ejecución de esta función se puede enviar instrucciones de mando al dron. Estas pueden enviarse de manera directa, a través de la instrucción `setcontrols`. Esta función permite dirigir el dron a una posición concreta en los tres ejecs controlados por posición, e indicarle una velocidad de giro constante, para el manejo de la guiñada. La función dispone de los siguientes parámetros:

def `setcontrols (arrAb, derIzq, delDet, giroDI, duracion):`

- Los parámetros: `arrAb`, `derIzq` y `delDet` son las consignas para los bucles de control PID en la FPGA. Estos son enviados a sus destinos a través de los flujos de datos *Comando de Altura*, *Comando Desplazamiento Lateral* y *Comando de Avance* de la Figura 4.8, respectivamente.

- El parámetro `giroDI` indica la velocidad de giro deseada para el vehículo. Es enviado directamente hacia el módulo de escalado de rango de la arquitectura de la Figura 4.8.

- `duracion` es el parámetro utilizado para especificar el tiempo de ejecución de cada instrucción, en segundos. Tras este periodo de tiempo se ejecutará la siguiente instrucción de mando.

Adicionalmente al método de control directo, se puede hacer uso de la función `settrace`, la cual genera posiciones intermedias entre la posición actual y la indicada en sus parámetros, para el instante de la ejecución.

def `settrace (arrAb, derIzq, delDet, giroDI, duracion, steps):`

Esta función hace uso de `setcontrols`, para generar puntos intermedios en una trayectoria larga. Los parámetros son los mismos que para `setcontrols`, salvo por el añadido de `Steps`, que indica el número de puntos intermedios a crear entre el punto actual y el destino indicado. Esta función es útil para recorrer distancias largas. Se debe a que en caso de, por ejemplo, despegue y vuelo a una distancia considerable del punto de despegue, se tendría un error de posición grande, lo que provoca un empuje considerable en la dirección de corrección del error. A pesar del posterior suavizado en la aceleración, debido a la componente derivativa, este empuje excesivo puede desestabilizar el control. Además, al inclinar brusca y considerablemente el vehículo se dificulta la adecuada medida de posición por parte del sensor óptico. Ocurre algo similar con el sensor de altura. Para una inclinación considerable del vehículo este mide una distancia mayor, debido al nuevo ángulo de incidencia sobre el suelo. Esto se traduce en un avance tortuoso del dron, avanzando “a tirones” hacia su destino. Para evitar esto se hace uso de `settrace`, que disminuye el crecimiento excesivo de los errores, minimizando este efecto.

La función `setcircle` permite trazar arcos circulares desde la posición actual del vehículo, hasta la indicada en sus parámetros. Funciona de manera similar a `settrace` en cuanto a la generación de una trayectoria con puntos intermedios. Esta trayectoria se define a través de sus parámetros:

def `setcircle (grados_ini, grados_fin, radio, clockwise, duracion_grado):`

- Los parámetros: `grados_ini`, `grados_fin` y `radio` definen el arco a trazar deseado. Dicho arco se trazará partiendo de la posición actual y terminando una vez recorrido el ángulo formado por la diferencia entre grados_fin y grados_ini para el radio indicado.
- El parámetro `clockwise` sirve para trazar el arco a derechas o a izquierdas.
- `duracion_grado` permite controlar la velocidad angular a la que se trazará el arco, teniendo una relación inversamente proporcional.

La función `takeoff` realiza el despegue del vehículo hasta la posición indicada en sus parámetros:

def `takeoff (height, duration):`

- La altura de vuelo deseada se indica a través del parámetro `height` en centímetros.
- El parámetro `duration` controla la velocidad de ascenso del dron hasta la altura indicada previamente.

Esta función garantiza un despegue controlado limitando la corriente inicial exigida por los motores, a través de una gestión adecuada de los parámetros PID durante el ascenso. Esto resulta necesario, ya que un ascenso brusco con los parámetros PID en su valor de vuelo estable, produce un exceso de crecimiento en la derivada del error de altura (al cambiar bruscamente de cero a `height`), pidiendo una descarga excesiva que las baterías del dron no pueden suministrar, resultando en un apagado de la electrónica propia del vehículo. Esta función traza un ascenso suave, a la vez que minimiza estas aceleraciones inadecuadas.

Para completar la librería Python la función `landing` se encarga del aterrizaje del dron hasta el contacto con el suelo, controlando la velocidad de descenso. Para ello hace uso de un solo parámetro:

def `landing (duration):`

- El parámetro `duration` controla el tiempo de descenso del dron hasta el contacto con el suelo.

Transcurrido el periodo de tiempo indicado en el parámetro `duration` se procede al apagado de los motores y descarga de las variables diferenciales del sistema, preparando así el vehículo para un nuevo vuelo, independiente del anterior.

4.5. Interfaz USB entre PC de mando y estación de tierra

Para conseguir la programación de la trayectoria de vuelo del dron desde un PC es necesario comunicar este PC con el elemento encargado del control. En este caso, la FPGA de la estación de tierra. Esta interfaz comunica ambos elementos mediante tramas enviadas por puerto serie. Dichas tramas portan las órdenes de posición que guiarán al vehículo, junto con los parámetros de configuración de los PIDs utilizados por los algoritmos de control de la FPGA. Estas tramas facilitan realizar múltiples ensayos de vuelo consecutivos, con mucha diferencia en el comportamiento de cada uno de ellos, facilitando a su vez la corrección de errores y el ajustes de parámetros.

4.5.1. Características

Se trata de una comunicación unidireccional con origen en el ordenador de mando y destino en la estación de tierra. La comunicación se realiza por puerto serie, y se basa en el uso de los estándares USB 2.0, UART y RS232. Para la conversión del estándar eléctrico existente en el PC (USB) al utilizado por la FPGA (RS232) se hace uso de un driver de USB que es diseño de FTDI Chip. Este driver convierte el protocolo de comunicaciones desde el PC al esperado en la FPGA (USB a UART). Concretamente es el canal B del integrado FT232HL el encargado de realizar dichas tareas, quedando el canal A reservado. Este se encuentra integrado en la misma

placa de desarrollo que la FPGA (ICE40 UltraPlus Breakout Board).
El puerto de comunicaciones (USB a UART) se configura en tiempo de ejecución por la herramienta software ejecutada en el ordenador de mando. En ella se indica el número del puerto, el cual depende de la asignación que realice el ordenador en uso, y la velocidad de transferencia, fijada en 500Kbps. Una vez abierto el puerto con esta configuración, se fragmentan las tramas y se envían byte a byte sin control de flujo, ni paridad. Se hace uso de diez bits por byte: Bit de inicio, 8 bits de datos y bit de parada.

4.5.2. Formato de tramas

Se hace uso de un formato de tramas fijas. La longitud es conocida y siempre la misma, 16 bytes. La posición de cada valor tampoco varía durante toda la ejecución, lo que facilita las tareas de decodificación y sincronía. A lo largo del trabajo se han ido añadiendo o modificando campos de la información transmitida por este interfaz según las necesidades puntuales de ensayos, o las mejoras en el sistema.
Cada trama es enviada en el instante de ejecución de cada instrucción en el ordenador de mando. Cada instrucción se ejecuta durante un tiempo concreto, indicado al ejecutarla. De esta manera, cada trama aquí descrita será enviada hacia la FPGA tras el tiempo de ejecución de cada instrucción. En su versión final, las tramas componen sus 16 bytes con la construcción prefijada de las Tablas 4.1 y 4.2.

Field Name	STx1	STx1	Cmd_ALT	Cmd_L/R	Cmd_F/B	Cmd_T_CW	PID_Alt_Kp	PID_XY_Kp
Value	0xFF	0x5A	Variable	Variable	Variable	Variable	Variable	Variable
Length (Bytes)	1	1	1	1	1	1	1	1
Byte Position	1	2	3	4	5	6	7	8

Cuadro 4.1: Formato de tramas de mando, primera mitad

Field Name	RSV	T_CW_Trimm	PID_Alt_Ki	PID_Alt_Kd	PID_XY_Ki	PID_XY_Kd	RSV	RSV
Value	0	Variable	Variable	Variable	Variable	Variable	0	0
Length (Bytes)	1	1	1	1	1	1	1	1
Byte Position	9	10	11	12	13	14	15	16

Cuadro 4.2: Formato de tramas de mando, segunda mitad

Se comienza la transmisión por el byte de la posición 1, terminando el envío con el byte 16. A continuación se describe cada campo de la trama:

- Los dos primeros se usan de etapa de sincronía. Campos Stx1 y Stx2. Tienen los valores fijos 255 y 90 en base decimal. Siempre son transmitidos de esta manera. Así, el ordenador de mando siempre comienza sus tramas con esta secuencia y la etapa receptora en la FPGA espera por una secuencia de dos bytes consecutivos con valores 255 y 90 para sincronizar con la trama entrante. Tras esto, el receptor comenzará a decodificar los valores de los siguientes 14 bytes en el orden indicado para esta trama. De esta manera, el equipo receptor asociará cada valor con su campo correctamente.

- Cmd_ALT: Comando de altura. Representa el valor de la consigna de posición de altura deseada para el dron. Este valor será posteriormente decodificado por la estación de tierra y enviado al bucle de control de altura como su consigna.

- Cmd_L/R: Comando de desplazamiento lateral. Representa el valor de la consigna de posición lateral deseada para el vehículo. Este valor será decodificado por la estación de tierra y enviado al bucle de control de desplazamiento lateral como su valor de consigna. Se trata del eje de libertad asociado al alabeo.

- Cmd_F/B: Comando de avance. Representa el valor de la consigna de posición frontal deseada para el dron. Este valor será decodificado por la estación de tierra y enviado al bucle de control de avance como su valor de consigna. Se trata del eje de libertad asociado al cabeceo.

- Cmd_T_CW: Comando de giro en sentido horario. Representa el valor deseado de la velocidad de giro del dron sobre sí mismo. Este valor será decodificado por la estación de tierra y enviado directamente al escalador, ya que este grado de libertad no se controla con un bucle cerrado. Se deja su control exclusivamente a la electrónica propia del dron y su giróscopo interno. Se trata del eje de libertad asociado a la guiñada.

- PID_Alt_Kp: Parámetro de la constante proporcional de altura. Representa el valor deseado para la constante proporcional del bucle de control PID de altura. La diferencia entre el valor de consigna de altura, Cmd_ALT, y valor de altura medido por los sensores de

a bordo, genera el valor de error de altura, componente del vector de error. Esta componente del error se multiplicará por el valor recibido en este campo para conseguir así la componente “P” del bucle de control de altura.

- PID_XY_Kp: Parámetro de la constante proporcional de los ejes horizontales (desplazamiento lateral y avance). Representa el valor deseado para las constantes proporcionales de ambos bucles de control PID, avance y desplazamiento lateral. Ya que el dron es simétrico respecto de sus ejes de desplazamiento horizontal, se aplica la misma tensión a ambos bucles de control. Las diferencias entre los valores de las consignas Cmd_L/R y Cmd_F/B y sus respectivas medidas por los sensores de posición de a bordo generan los errores de posición horizontales. Estos errores se multiplicarán por el valor recibido en este campo para conseguir así las componentes “P” de los bucles de control de los ejes horizontales: Avance y desplazamiento lateral.

- RSV: Los bytes de las posiciones 9, 15 y 16 están reservados para posibles usos futuros. Tienen valores constantes a cero.

- T_CW_Trimm: Parámetro de corrección de giro del dron sobre sí mismo. Representa el valor del offset a añadir al valor del comando “Cmd_T_CW” a enviar hacia el dron. El objetivo de este parámetro es corregir de manera constante el valor de la velocidad de giro sobre sí mismo enviado hacia el dron. Ya que este valor se encuentra controlado por un bucle abierto en la electrónica de la estación de tierra, si la electrónica propia del dron tiene un error constante de giro, que no pueda corregir con su giróscopo, se aplicará la corrección necesaria a través del valor de este parámetro.

- PID_Alt_Ki: Parámetro de la constante integral de altura. Representa el valor deseado para la constante integral del bucle de control PID de altura. El valor del error de altura comentado anteriormente se integra y multiplica por el valor recibido en este campo de la trama. De esta manera se consigue la componente “I” del bucle de control de altura.

- PID_Alt_Kd: Parámetro de la constante derivativa de altura. Representa el valor deseado para la constante derivativa del bucle de control PID de altura. El error de altura comentado anteriormente se derivará y multiplicará por el valor recibido en este campo de la trama. De esta manera se consigue la componente “D” del bucle de control de altura.

Este campo, junto con los valores de los campos de las posiciones 7 y 11, completa los valores de los parámetros PID para el bucle de control de la altura, el *PID ALTURA* de la arquitectura mostrada en la Figura 4.8.

- PID_XY_Ki: Parámetro de la constante integral de los ejes horizontales. Representa el valor deseado para las constantes integrales de ambos bucles de control PID de posición horizontal, es decir, el control de posición del avance y el control de posición del desplazamiento lateral. Ya que el dron es simétrico respecto de sus ejes de desplazamiento horizontal, se aplica el mismo peso a las componentes integrales de ambos bucles de control. Los valores de los errores de posición horizontales comentados anteriormente se integran y multiplican por el valor recibido en este campo, para conseguir así las componentes "I" de los bucles de control de los ejes horizontales: Avance y desplazamiento lateral.

- PID_XY_Kd: Parámetro de la constante derivativa de los ejes horizontales. Representa el valor deseado para las constantes derivativas de ambos bucles de control PID de posición horizontal. Ya que el dron es simétrico respecto de sus ejes de desplazamiento horizontal, se aplica el mismo peso a las componentes derivativas de ambos bucles de control. Los valores de los errores de posición horizontales comentados anteriormente se derivarán y multiplicarán por el valor recibido en este campo, para conseguir así las componentes "D" de los bucles de control de los ejes horizontales: Frontal y lateral. Este campo, junto con los valores de los campos de las posiciones 8 y 13, completa los valores de los parámetros PID para los bucles de control del plano horizontal, los *PID AVANCE* y *PID DESP. LATERAL* de la arquitectura mostrada en la Figura 4.8.

4.6. Interfaz radio entre el dron y la estación de tierra

Para ejecutar el control de vuelo del vehículo es necesario establecer comunicación con él desde tierra. Dicha comunicación se realiza exclusivamente por radio, mediante dos canales independientes, como se muestra en la sección *Módulo en Vuelo* de la Figura 4.1.
La interfaz radio incluye ambos, el primero comunica la FPGA de la estación de tierra con los sensores instalados a bordo y el segundo con la electrónica de control de movimiento propia del

dron. Estos son los interfaces *Medidas de sensores* y *Órdenes de movimiento* respectivamente. Su objetivo es que la estación de tierra conozca la posición del dron y en base a esta, realice el control de movimiento.

En su conjunto se trata de un enlace radio bidireccional entre la estación de tierra y el dron, con dos canales trabajando en la banda libre de 2.4GHz, separados en frecuencia. De esta manera se consigue un enlace bidireccional, full-duplex, mediante separación en frecuencia (FDM), que permite transmisión simultanea de información de subida (órdenes) y bajada (medidas), sin colisiones. Existen otras opciones de multiplexación de canal, como radios basadas en separación temporal (TDM), código (CDM) o frecuencias ortogonales (OFDM). Cada una con sus dificultades, como bajada en tasa de transferencia, sincronización y sobre todo, complejidad de implementación.

A continuación se explican ambos canales de comunicación entre la estación de tierra y el vehículo.

4.6.1. Transmisión de medidas de sensores

Esta interfaz define el canal de enlace de bajada, es decir, la conexión radio unidireccional, con origen en el dron y destino en la estación de tierra. Se utiliza para comunicar las medidas de los sensores de posición tomadas a bordo del vehículo, con la FPGA en tierra, cuyos bucles de control requieren de dicha información. Esta información se compone de: una medida de distancia al suelo, o altura, tomada desde la parte central del dron, una medida del desplazamiento frontal, o avance, y una medida del desplazamiento lateral. Todas ellas se entregan en milímetros y referidas al punto de despegue.

Características

La separación en canales entre subida y bajada se configura durante la programación inicial de la radio. Este enlace ocupa el canal 102, asociado a una frecuencia concreta en la banda disponible por el transceptor NRF24L01. Se trata de una transmisión de baja potencia, con un alcance de unos 30 metros, transmitiendo a 1Mbps.

Formato de trama

Se hace uso de un formato de tramas fijas. En este caso la longitud de la trama es de seis bytes. La posición de cada valor tampoco varía durante toda la ejecución, siendo la que se muestra en la Figura 4.3. Cada trama es enviada en el instante posterior a la ejecución de las tres medidas de posición: altura, desplazamiento frontal y desplazamiento lateral.

Field Name	H_Pos_F/B	H_Pos_L/R	Alt
Value	Variable	Variable	Variable
Length (Bytes)	2	2	2
Byte Position	1	3	5

Cuadro 4.3: Formato de tramas de bajada

En estas tramas la sincronía está garantizada como parte del diseño de la transmisión realizada por la radio NRF24L01, por tanto no se hace uso de bytes de sincronía. Los seis bytes incluidos en la trama son tres campos con información útil, de dos bytes cada uno. Estos son:

- H_Pos_F/B: Valor de la medida de avance. Representa los milímetros de desplazamiento frontal desde el origen de coordenadas definido anteriormente. Se almacena en 16 bits consecutivos, y se codifica en complemento a dos.
- H_Pos_L/R: Homólogo al campo “H_Pos_F/B”, representando este caso los milímetros de desplazamiento lateral.
- Alt: Valor de la medida de altura. Representa los milímetros de desplazamiento vertical desde el estado “aterrizado”. Se almacena en 16 bits consecutivos, y se codifica en binario plano, ya que no se contemplan alturas negativas para el marco de este TFG.

Con estas tres medidas la estación de tierra puede ubicar de manera relativa el dron, generando así el vector de posición necesario para realizar posteriormente las correcciones de posición respecto de las órdenes comandadas.

4.6.2. Transmisión de órdenes de movimiento

Las órdenes de movimiento, con origen en el ordenador de mando y destino en el vehículo, componen el enlace de subida. Se trata de una conexión radio unidireccional, para compartir

con el dron las instrucciones de movimiento que éste deberá ejecutar. Además incluye canales auxiliares, que no se han usado en este TFG.

Características

Se utiliza una radio NRF24L01 en la banda de 2.4GHz. Esta interfaz, empaqueta las órdenes provenientes de las tramas PPM, enviadas desde la FPGA a través de *Tramas_PPM* (Apartado 4.3.3), en una única trama radio. Dicha trama es entonces enviada hacia el modulo de recepción del dron.

Formato de trama

Este interfaz se compone de una transmisión de doce canales, de los cuales se hace uso exclusivamente de los cuatro primeros. Los ocho canales restantes portan comandos auxiliares; como la velocidad de reacción deseada para el dron, la cual se mantiene en su valor por defecto y dependiendo del modelo del dron utilizado, a veces disparan fotografías o vídeo de cámaras instaladas. El *Transmisor radio de órdenes de movimiento* (Apartado 4.3.2) se encarga de transmitir la información contenida en la trama de la Figura 4.6.2, por radio, hasta la electrónica del dron.

Field Name	Cmd_THR	Cmd_R/L	Cmd_F/B	Cmd_T_CW	Aux
Value	Variable	Variable	Variable	Variable	Variable
Length (ms)	1 to 2	1 to 2	1 to 2	1 to 2	8 to 16
Channel Position	1	2	3	4	5 to 12

El vehículo, entonces, realizará movimientos en base a los valores contenidos en dicha trama. Sus campos se comentan a continuación:

- Cmd_THR: Valor de velocidad de altura. Ocupa la primera posición en la trama. Representa el impulso deseado para el eje vertical del dron. Un valor de cero no tiene por qué implicar el apagado de los motores, depende de los valores de los siguientes dos canales.

- Cmd_R/L: Valor de velocidad de desplazamiento lateral. Representa la velocidad de desplazamiento asociada al grado de libertad del alabeo.

- Cmd_F/B: Valor de velocidad de avance. Representa la velocidad de desplazamiento frontal deseada para el dron. Está asociado al grado de libertad del cabeceo.
- Cmd_T_CW: Valor de velocidad de giro. Representa la velocidad de giro sobre sí mismo en el plano horizontal. Controla la guiñada del vehículo.
- Aux: Algunos de los ocho canales adicionales tienen valores fijos y otros variables, dependiendo de su funcionalidad y del dron utilizado. Para este trabajo estos canales se mantendrán con su valor por defecto.

Capítulo 5

Validación experimental

En este apartado se describen los experimentos y pruebas realizadas durante el trabajo. Estos permitieron realizar la selección de los componentes finales a utilizar, la corrección de errores y la configuración de parámetros.

5.1. Pruebas con dron Eachine E010

Antes de desembocar finalmente en el dron X5C se exploró el uso del dron Eachine E010 de la Figura 5.1. Se trata de un pequeño cuadricóptero, ligero y de reacciones nerviosas. Cuenta con 8cm de ancho y 8 cm de largo, conductores en las aspas para minimizar vórtices, batería de litio y 30 metros de alcance con su radio de 2.4GHz.

Sobre él se hicieron pruebas de control en bucle abierto, se realizaron las tareas de enlace

Figura 5.1: Dron Eachine E010

de comunicaciones entre la estación de tierra y el dron y se probaron despegues, aterrizajes y vuelos cortos de ida y vuelta en línea recta como el mostrado en la secuencia de imágenes de la Figura 5.2.

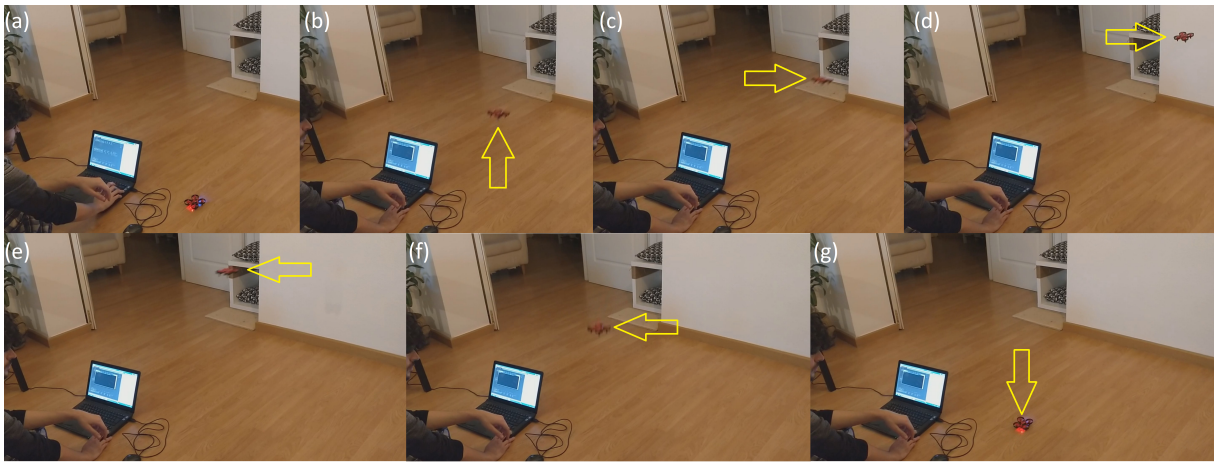

Figura 5.2: Secuencia de vuelo con dron E010: (a) Comando, (b) Despegue, (c)-(f) ida y vuelta y (g) aterrizaje.

El dron mostró buen control de vuelo en tiempo cortos, por tanto fue válido para pruebas de bucle abierto, aplicando algo de pre-énfasis. Desgraciadamente, el vehículo presentaba derivas erráticas en sus planos horizontales (avance y desplazamiento lateral), y grandes derivas en el eje vertical (altura), dependientes, entre otras cosas, del nivel de carga restante en la batería. Estas dificultades hicieron de los vuelos largos en bucle abierto una tarea imposible.
En una segunda fase se planteó instalar el controlador de vuelo de bucle cerrado a bordo de dicho dron. Este controlador requiere de los sensores y procesador comentados en el apartado 4.2.2. Desgraciadamente el peso de esa electrónica adicional era excesivo para los motores del Eachine E010, por tanto se decidió abortar esta vía y buscar otra plataforma de mayor potencia. Por este motivo se seleccionó finalmente el dron Syma X5C.

5.2. Pruebas Unitarias

A continuación se describen las pruebas realizadas sobre distintos módulos del sistema ya con el dron X5C, previas a su integración. Se comienza describiendo los ensayos de enlace radio para el subsistema de bajada. Este abarca desde la electrónica de sensorización y transmisión instalada en el dron, hasta el módulo *Receptor radio de medidas de posición* de la estación de

tierra (módulo 3 del apartado 4.3.2). Se continua con el subsistema de subida, que comprende el módulo *Transmisor radio de órdenes de movimiento* de la estación de tierra (módulo 4 del apartado 4.3.2), y la electrónica original de cada vehículo ensayado. Y se termina con los ensayos unitarios de módulos software en la estación de tierra.

5.2.1. Pruebas del enlace radio del subsistema de bajada

Para ensayar el conjunto de módulos que componen este subsistema se hizo uso del USB del *Receptor radio de medidas de posición* de la estación de tierra. Éste se programó para comunicar hacia la consola de depuración de Arduino las medidas de recibidas en cada trama radio. Con esto hecho y la electrónica de sensorización y transmisión programada para medir periódicamente y transmitir desde el dron, se procedió a encender el conjunto y observar la consola del puerto USB, mostrada en la Figura 5.3. De esta manera se verificó visualmente que el enlace radio funcionaba satisfactoriamente.

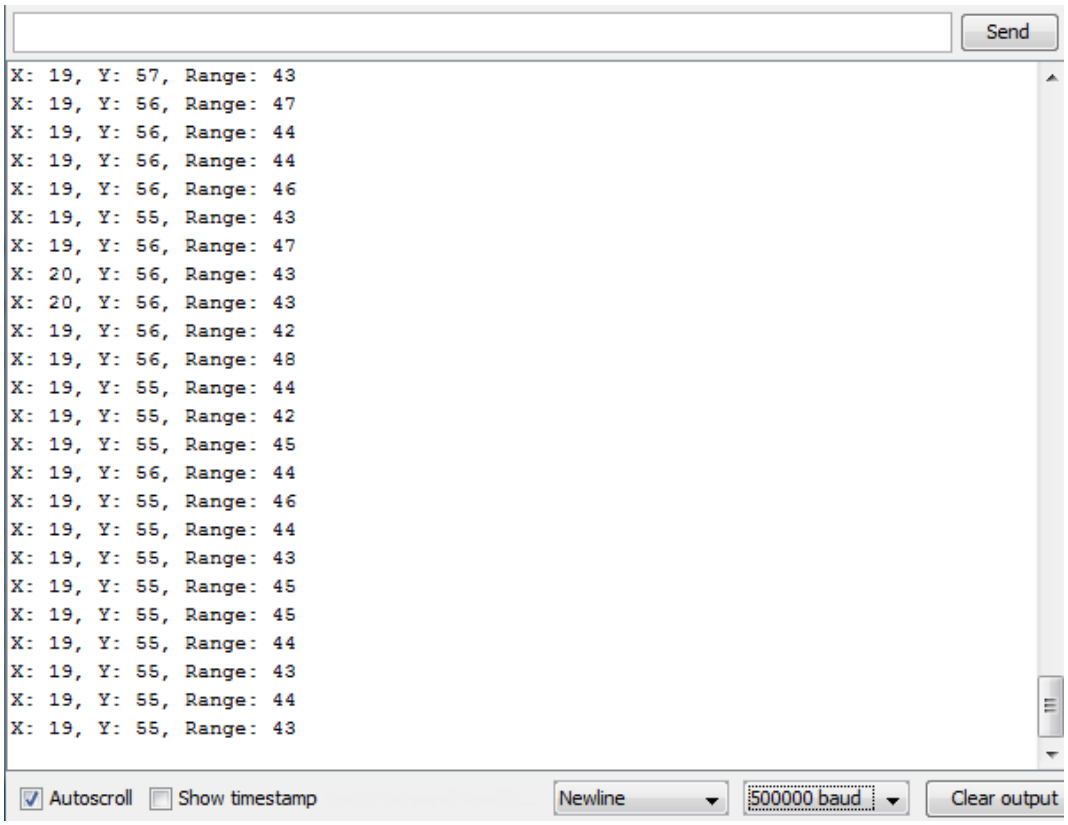

Figura 5.3: Medidas de sensores capturadas en estación de tierra desde el enlace radio

Para comprobar la correcta medida de los sensores y la calidad del enlace radio se obser-

varon las medidas realizadas por la electrónica instalada en el dron, previas a su transmisión. El método escogido fue el mismo, observando esta vez la consola USB del procesador Atmel embarcado (módulo 1 del apartado 4.2.2). Al contrastar dicha observación con la tomada anteriormente en la Figura 5.3, se verificó que no existía pérdida ni corrupción de paquetes en el enlace radio de bajada.

5.2.2. Pruebas del enlace radio del subsistema de subida

Este módulo se verificó a través de las pruebas de enlace con los dos drones utilizados durante el desarrollo del TFG. Estas se realizaron en el punto de partida del proyecto.
Durante el establecimiento del enlace entre el módulo *Transmisor radio de órdenes de movimiento* de la estación de tierra y el dron, las luces de indicación frontales del dron cambian su patrón de parpadeo. En caso de un correcto enlace, dichas luces quedan fijas como en la Figura 5.4. Esta comprobación visual es suficiente para verificar el enlace, y de paso la correcta construcción y programación del módulo 4 de la estación de tierra.

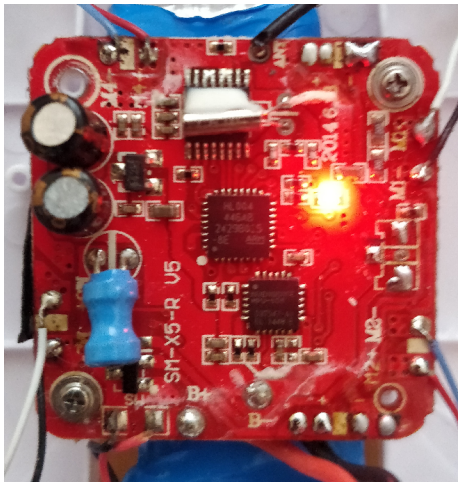

Figura 5.4: Indicación luminosa de enlace radio correcto entre estación y dron

5.2.3. Pruebas de módulos en la estación de tierra

El *Procesador basado en FPGA* de la estación de tierra (módulo 2 del apartado 4.3.2) ejecuta el software principal del sistema, compuesto de múltiples módulos diseñados en Verilog. Los módulos más sencillos se instanciaron tras la codificación, directamente sobre el sistema final.

Los módulos restantes, debido a su complejidad o impacto en la seguridad del vuelo del vehículo, se verificaron de manera individual a través de simulaciones lógicas, previa su integración. A continuación se describen estas simulaciones para los módulos más significativos.

Prueba de módulo de comunicaciones UART

Este módulo es utilizado como receptor de puerto serie de las tramas desde el PC y desde el dron. El objetivo de este ensayo es comprobar que el módulo UART responde al *baud rate* programado y que la decodificación del byte entrante se realiza en el orden adecuado, a efectos de bit más y menos significativos. Se inyecta un byte de valor cero seguido de uno con valor uno, a través de la señal *sdin*, con el bit menos significativo primero. El resultado se puede observar en la señal *data*, síncrona con la señal *data_rdy* de la Figura 5.5, sin errores de trama para ninguno de los dos casos ensayados.

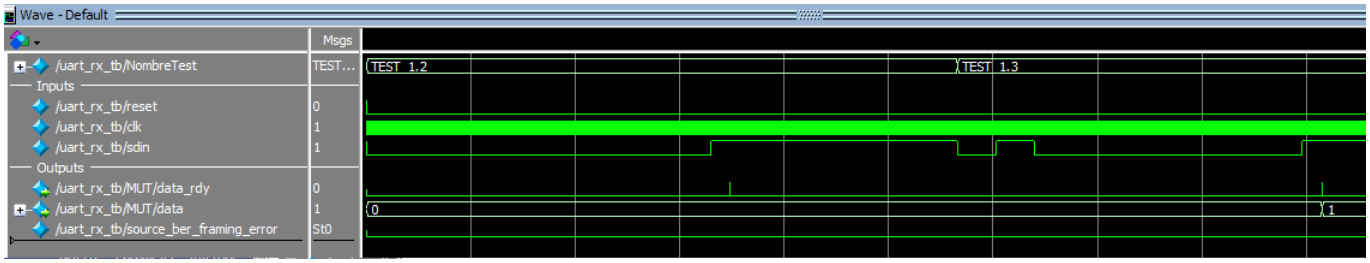

Figura 5.5: Simulación funcional del módulo UART

Prueba del módulo decodificador de tramas desde PC

El test consiste en enviar hacia el decodificador una trama completa para comprobar su correcta decodificación. La trama dispone de los catorce campos definidos en el apartado 4.5.2. Los valores incluidos en la trama van de 1 a 14 con valores crecientes. En la Figura 5.6 se observa la trama enviada en la señal *sink_data* y su correcta decodificación en las señales de nombre source_* junto a la señal de datos válidos (*source_data_valid*) asertada en el instante de recepción del último byte (con valor 0x0E). El test solo captura en sus salidas los ocho primeros bytes. Esto se debe a que el módulo se diseñó originalmente para la recepción de tramas orientadas al control del dron en bucle abierto, usando solo ocho bytes por trama. Una vez comprobada la correcta decodificación de la trama para ocho bytes, su funcionalidad de

aumentó, para decodificar tramas de catorce bytes. Estas incluyen adicionalmente los valores de los parámetros para los PIDs ejecutados en la FPGA.

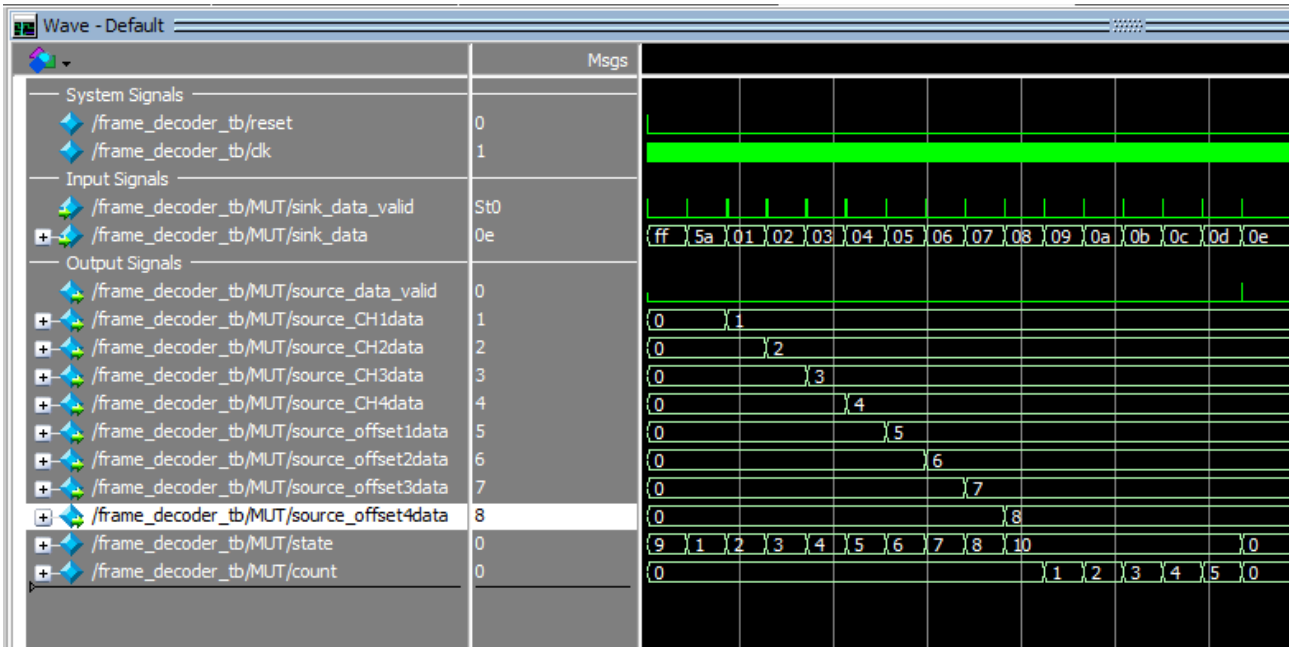

Figura 5.6: Simulación funcional del módulo decodificador de tramas de mando desde PC

Prueba del módulo decodificador de tramas desde dron

De manera similar al test al anterior, con este módulo se ensayó la recepción de una trama completa y posterior comprobación de su decodificación. En este caso, las señales recibidas son valores enteros de dieciséis bits en complemento a dos, separados en grupos de dos bytes. El ensayo envía un valor negativo (-2) como avance, un valor positivo mayor que 255 como desplazamiento lateral (400) y un valor positivo menor que 255 como altura (31). El resultado de la decodificación se muestra en la Figura 5.7, de manera síncrona al pulso en la señal *source_data_valid* en las tres señales posteriores. Observando el -2 del avance en la señal *source_H_disp_front*, el 400 en *source_H_disp_side* y el 31 en *source_altitude*, se verifica el correcto funcionamiento del módulo.

Prueba de módulos PID

Los módulos PID que controlan altura y desplazamientos frontal y lateral hacen uso de una lógica similar, salvando diferencias menores, como el truncamiento de valores negativos

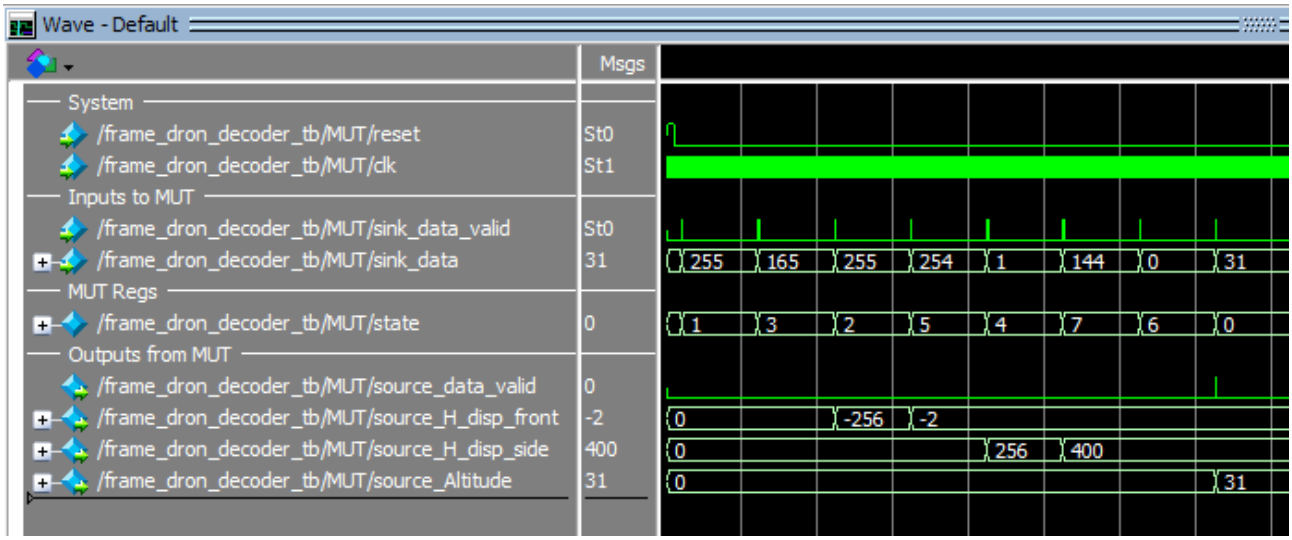

Figura 5.7: Simulación funcional del módulo decodificador de tramas de sensores desde dron

para el caso del PID de altura y el ajuste de salida a los rangos esperados para la generación de las señales PPM. Los ensayos se realizaron en cuatro escenarios numéricos distintos para comprobar el correcto funcionamiento de las distintas componentes:

- TEST_1: Ensaya la componente proporcional en sentido negativo. El resultado del bucle en negativo se observa en la señal *pid_prereg* de la Figura 5.8. Dicho valor es capado posteriormente a cero en el registro de salida del módulo PID *source_pid*.
- TEST_2.1: Ensaya la componente integral en sentido positivo. Su crecimiento se observa sobre el registro *error_i_acumm* con suma resultante sobre *source_pid*.
- TEST_2.2: Tiene por objetivo ensayar el comportamiento del módulo para el caso de la puesta a cero de los parámetros que rigen el PID. Se desea conservar el valor integral sin aplicar sobre el módulo correcciones adicionales o puestas a ceros inadecuadas.
- TEST_3.1: En este caso de ensayo se pone a prueba la componente derivativa. Para ello se genera un error en sentido negativo y se observa el impacto de la derivada sobre la señal *source_pid* de la Figura 5.8.

Prueba de módulo de modulación PPM

Este ensayo permite verificar las correctas transiciones entre los estados de cada canal y entre los 12 canales. En la Figura 5.9 se observa el registro *PPM_STATE*, que almacena el

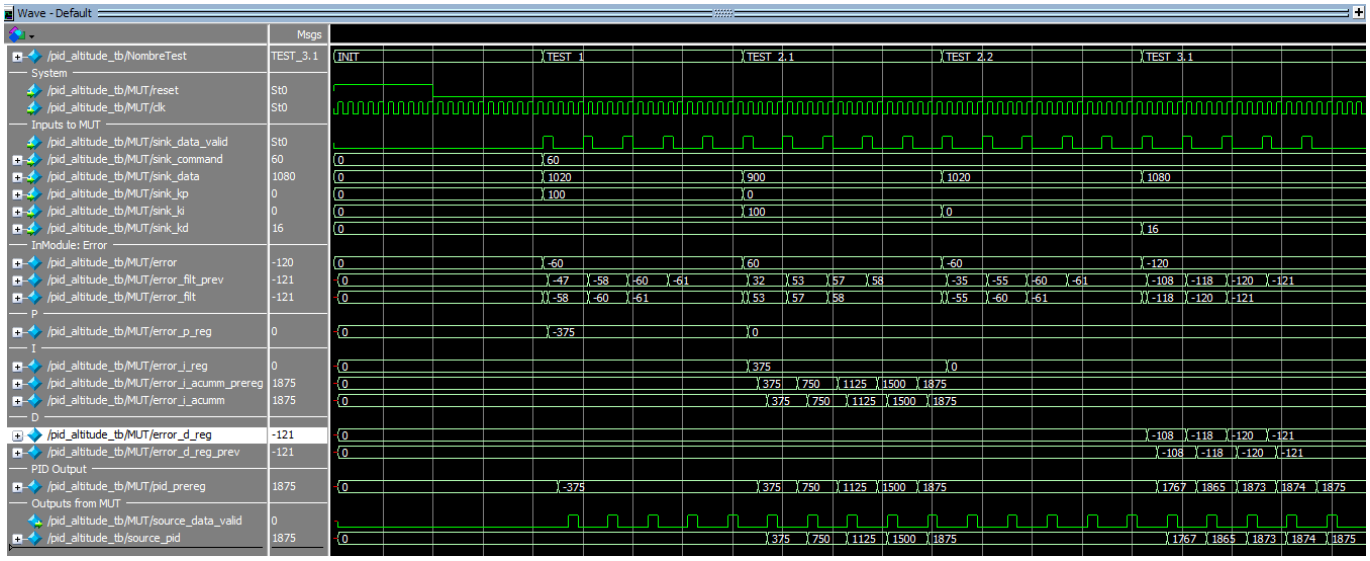

Figura 5.8: Simulación funcional del módulo PID de altura

estado encargado de contabilizar el tiempo a cero de cada pulso de bajada. Por otro lado, el estado *CHOOSE_CHANNEL* se encarga de gestionar las transiciones para cada canal.

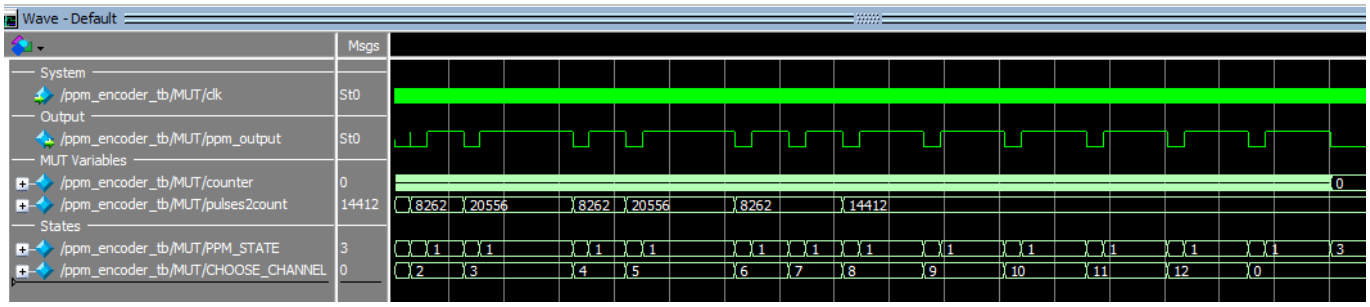

Figura 5.9: Simulación funcional del módulo PPM

Este ensayo permitió corregir errores en las transiciones entre estados, que producían saltos inadecuados, hasta llegar, finalmente, a las transiciones observadas en la Figura 5.9.

5.3. Pruebas de vuelo con los bucles de control

Una característica fundamental de la solución desarrollada en este TFG es que el control de vuelo se materializa en la FPGA de la estación de tierra con información de los sensores que van a bordo del dron. Las órdenes de control generadas suben vía radio desde la FPGA hasta el propio dron. Este diseño introduce latencias no despreciables que condicionan la calidad del

control realizable.

Además, se ha elegido una FPGA barata, de capacidad y velocidad modestas comparadas con otras FPGAs de gama alta. El motivo de elegir este modelo ICE40UP5K es que soporta la herramienta libre IceStudio. Aunque en este TFG la hemos usado con entornos de desarrollo propietarios, esta elección permite continuar este trabajo con desarrollos posteriores ya utilizando entornos de desarrollo libres. Este diseño y la FPGA modesta implicaron limitaciones en el ancho de banda final del sistema y sobre todo en las latencias. Ambos dos tienen relación directa con la calidad del control final.

5.3.1. Pruebas básicas de vuelo del dron Syma X5C

Tras las pruebas con el E010, se comenzó por conseguir el mismo nivel de control sobre el X5C original. Para ello se modificó el código de enlace con el dron, ejecutado en el módulo 4 de la estación de tierra (Transmisor radio de órdenes de movimiento del apartado 4.3.2), para conseguir un control básico sobre el X5C.

Una vez enlazado se hicieron pruebas de control en bucle abierto de igual manera que con el E010. El dron Syma resultó también difícil de controlar en bucle abierto. Este dron presentaba varios problemas. Por un lado, adolecía de las típicas derivas constantes, las cuales se podían disminuir mediante correcciones prefijadas. Sufría también desvíos poco previsibles en su vuelo, lo cual dificultaba las tareas de corrección previas como el pre-énfasis utilizado en el E010. Además, el nivel de carga en sus baterías se degradaba rápidamente, lo que se traducía en una caída de tensión progresiva, lo que a su vez suponía una degradación notable y poco previsible en la potencia de los motores durante la duración de cada vuelo. Esto último, sería posiblemente debido a la baja calidad de las baterías y de los controladores de los motores, junto a la inexistencia de reguladores de tensión. Este problema dificultaba extremadamente su control en altura, haciéndolo casi imposible mediante técnicas de bucle abierto.

Se procedió por tanto a instalar el sistema de medida de posición, que por exceso de peso no toleraba el E010. El X5C sí es capaz de levantar la electrónica embarcada, pero debido al peso adicional sus motores quedaban encendidos casi al máximo de su potencia para mantener el vuelo. En este punto ya se contaba con el bucle cerrado tratando de controlar el dron, pero el exceso de peso continuaba dificultando el correcto control en altura del vehículo, generando

grandes oscilaciones de una amplitud, imposibles de corregir por el sistema en este estado.
Se procedió a aligerar la electrónica embarcada, eliminando elementos innecesarios de las tarjetas disponibles y reducir el peso del dron, con modificaciones mayores. El resultado de estas modificaciones se muestra en la Figura 5.10. Básicamente se libró al dron de todo aquello no estrictamente imprescindible para el vuelo. Se acortaron cables, se eliminaron luces de indicación de frente, soporte de cámara, amarre de batería, topes de motores, protecciones de aspas, tornillos, y media estructura de plástico del dron. Algunos elementos retirados se sustituyeron por puntos de soldadura o pegamento, en pro de aligerar todo lo posible.

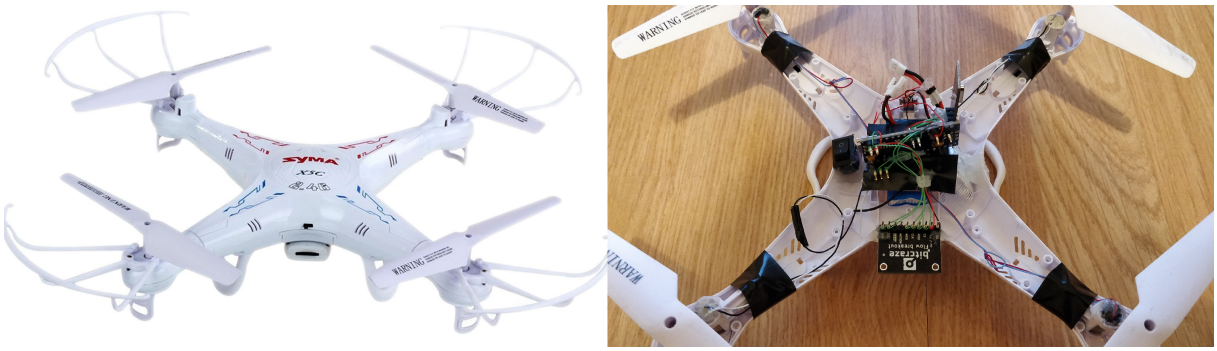

Figura 5.10: Dron X5C Original (izquierda) y tras modificaciones (derecha)

Al retirar media estructura, la resultante pecaba de exceso de flexibilidad y falta de rigidez, lo que eliminó parte de la independencia de ejes, pero a su vez, permitió a los controladores y motores tener más margen de potencia disponible para realizar correcciones. De esta manera, se ganó bastante en cuanto a estabilización del dron.
En este punto el trabajo se centró en mejorar los parámetros PID de los controladores ubicados en la FPGA de la estación de tierra (control en bucle cerrado), ya que fueron diseñados para el dron original, previo al cambio de peso.
Además, se trabajó rotando dos baterías y controlando rigurosamente su nivel de carga. Ya que aunque los bucles de control PID corrigen parte del error debido a la descarga, llegados a un nivel de batería excesivamente bajo, los parámetros de control mismos dejaban de ser igualmente efectivos. Entonces, la velocidad de respuesta sobre el error de posición también disminuía, lo que al final se traducía en el mismo efecto, unos parámetros PID inadecuados. Procurando trabajar con baterías en un rango de carga concreto se minimizó este efecto lo suficiente para garantizar repetitividad entre vuelos.

5.3.2. Ancho de banda

Dos prestaciones fundamentales que condicionan la calidad de los bucles de control son el ancho de banda y las latencias involucradas en un sistema concreto. A continuación se describe el análisis realizado de ellos para el bucle de control del dron desde la estación de tierra.
El ancho de banda alcanzable se ve limitado por la velocidad de ejecución del elemento más lento dentro del sistema. Esta se debate entre dos similares: (a) la velocidad de muestreo de órdenes de movimiento por parte de la electrónica propia del dron, que limita cuántas correcciones de posición puede atender por segundo, y (b) la lectura de los sensores de posición, limitada por la electrónica embarcada y las especificaciones de los propios sensores. Finalmente se estableció una velocidad de lectura máxima de 30.3 Hz, que permitía a los sensores ser leídos adecuadamente (sin errores en las muestras leídas) y a la electrónica propia del dron atender todas las tramas entrantes. Esto limita la agilidad en las reacciones del dron, en caso de correcciones bruscas necesarias y la suavidad del mismo durante ciertos movimientos.

5.3.3. Retardo

Aunque el ancho de banda es un parámetro fundamental en el control de un sistema, para el caso de este TFG, la mayor cantidad de problemas durante el desarrollo y pruebas del sistema fueron originados por falta de estabilidad de vuelo. Esta se pone en riesgo por múltiples factores, algunos ya comentados, siendo de especial importancia los retardos en la ejecución de los bucles de control. El tiempo que transcurre desde que se realiza la medida de posición de un eje hasta que se actúa en consecuencia para corregir el error es fundamental para estabilizar correctamente el vehículo.
En este aspecto, hay varias fuentes de retardo en el sistema, algunas inherentes a algún componente, como los retardos de medida de los sensores, y otras, fruto de la arquitectura escogida. Por ejemplo el hecho de que el proceso de control sea ejecutado en tierra y no a bordo del dron genera retardos asociados a la transmisión de las medidas de posición. Estas no son procesadas a bordo, sino que se transmiten hacia tierra donde se procesan en la FPGA, obteniendo las instrucciones de movimiento como respuesta, que se envían al dron. Este proceso de transmisión a tierra añade retardos indeseados, pero inevitables para el funcionamiento del sistema.
Si tratamos de contabilizar aproximadamente los retardos existentes, tenemos:

- Sensores: 30ms del sensor de medida de altura.
- Enlace radio de subida: 33ms de retardo máximo añadido por la sincronía entre la trama PPM enviada desde la FPGA y su lectura realizada por la electrónica propia del vehículo.
- Enlaces radio de bajada: no más de 120uS sin contar preámbulos ni retransmisiones.
- Puertos serie síncronos: Tres comunicaciones SPI, unos 144us en total.
- Puertos serie asíncronos: Dos transmisiones UART de no más de 240uS.

A falta de contabilizar de manera más fina los retardos, es sabido que existe un mínimo de 63.5ms de retardo entre medida de posición y reacción del vehículo. Esto, sin ser una cifra catastrófica, sí es cierto que implica ciertas limitaciones a la hora de conseguir una suavidad de vuelo concreta.

5.4. Pruebas integrales

Tras las pruebas unitarias realizadas sobre los drones Eachine E010, Syma X5C y módulos software se realizaron ensayos completos de vuelo. Para ello se hizo uso del sistema final, compuesto por el ordenador ejecutando las instrucciones de mando en Python, la estación de tierra con la FPGA programada para ejecutar el control de vuelo y el dron X5C con la electrónica de a bordo transmitiendo su posición.
Los ejercicios realizados incluyen desde la ejecución en PC del software de mando hasta el vuelo y aterrizaje del dron, controlado desde tierra. Dichos ejercicios hacen uso de las funciones contenidas en la librería de Python construida previamente (Apartado 4.4.2). A continuación se describen algunos de los más relevantes realizados de manera integral.
Se ensayó una secuencia controlada de despegue y aterrizaje mostrada en la Figura 5.11 extraída de la grabación del ensayo[1]. El programa de control en Python ejecutado en el ordenador de mando consta de una instrucción de despegue a una altura de 32cm (`takeoff`), seguida de una indicación de vuelo controlado de 6 segundos sobre la misma ubicación (`setcontrols`), para finalizar con una instrucción de aterrizaje (`landing`) cuya ejecución se completa con el apagado de los motores.

[1] https://www.youtube.com/watch?v=U2gZdFM8GPE

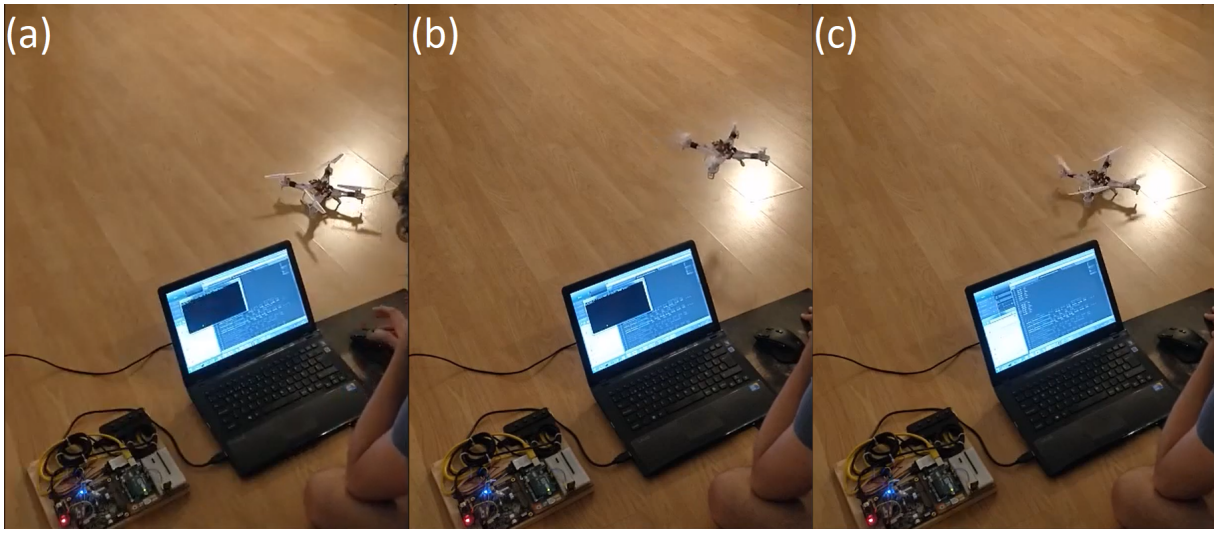

Figura 5.11: Secuencia de despegue y aterrizaje controlado con dron X5C comandado desde PC: (a) Comando de despegue, (b) vuelo estacionario y (c) aterrizaje.

Partiendo de la base del ensayo previo, se extendió su funcionalidad para realizar un viaje de ida y vuelta cuyo resultado se secuencia en las imágenes de la Figura 5.12. La grabación[2] del ensayo muestra el vehículo avanzando hasta la posición indicada para posteriormente iniciar la vuelva al origen. El programa encargado del control consta de una instrucción de despegue a 32cm de altura hecha efectiva en la imagen (a), seguida de una instrucción de avance (`settrace`) hacia una posición 179cm más adelante completada en la imagen (d) y una segunda ejecución de la misma instrucción indicándole en esta ocasión el retorno a la posición de origen (imagen (f)), para terminar con la instrucción de aterrizaje completada en la imagen (g).

Adicionalmente se ensayó la estabilidad del vuelo estacionario y su respuesta ante estímulos externos. Para ello se realizó un despegue hasta los 32cm de altura y posterior indicación de vuelo estacionario, previo al comando aterrizaje. Para facilitar el ensayo, entre la instrucción de vuelo estable y la indicación de aterrizaje se introdujo una lectura por teclado, para disponer del vehículo en vuelo el tiempo necesario. En la grabación del ensayo[3] se observan las dos interferencias en el vuelo del dron (la primera se muestra en la Figura 5.13) a través de dos toques para alterar su posición de manera forzada obligando al sistema a corregirla, junto con la pulsación del teclado que pone fin al ensayo.

[2] https://www.youtube.com/watch?v=EGjc1TjfpkM
[3] https://www.youtube.com/watch?v=V-kKepw79hg

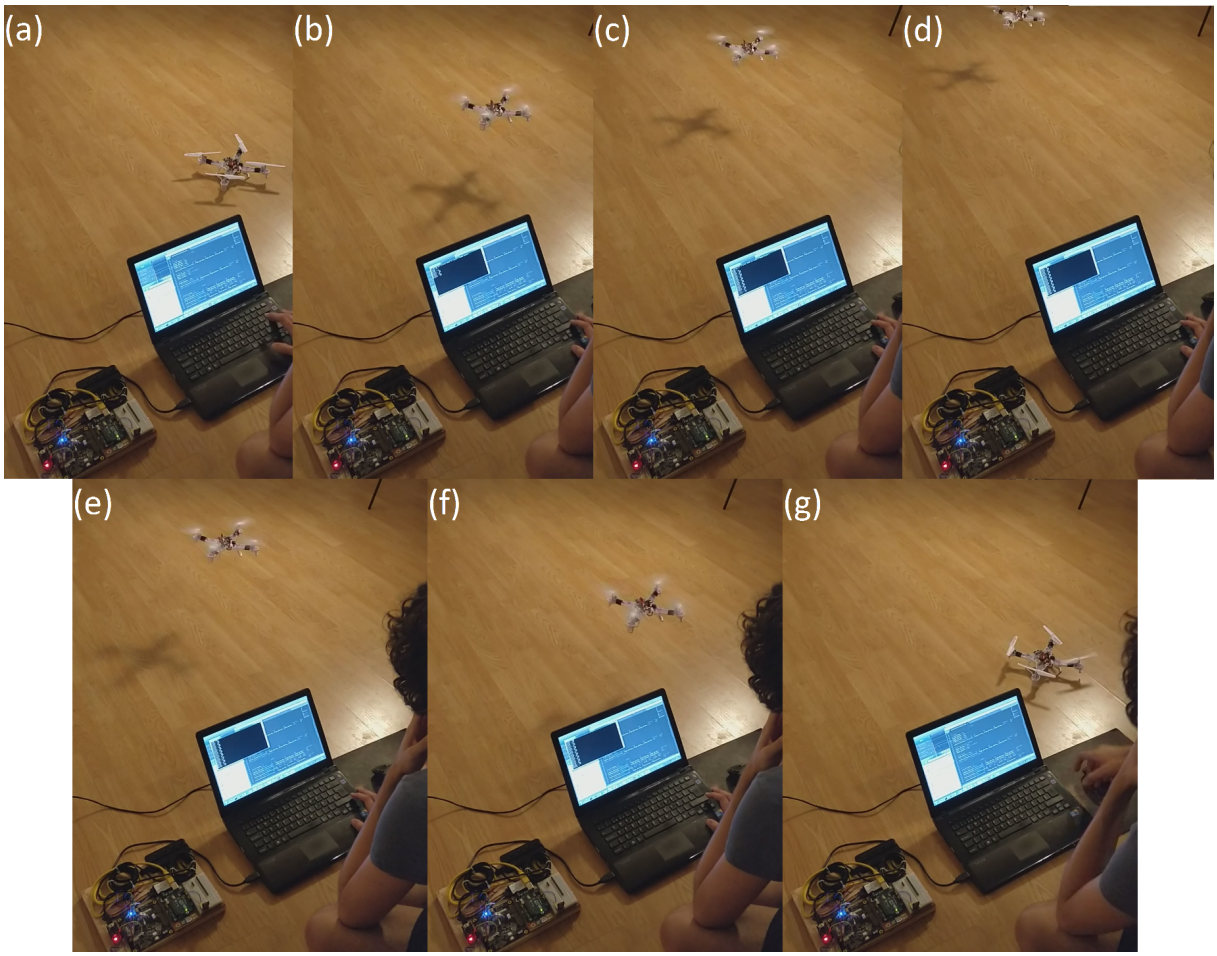

Figura 5.12: Secuencia de vuelo autónomo con dron X5C comandado desde PC: (a) Origen, (b) despegue, (c)-(f) ida y vuelta y (g) aterrizaje.

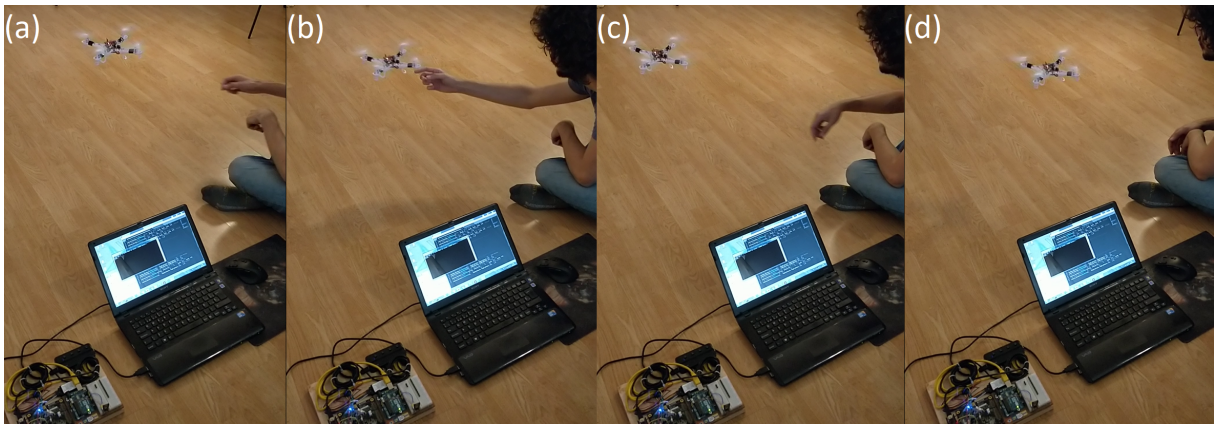

Figura 5.13: Secuencia de vuelo autónomo con dron X5C comandado desde PC forzando respuesta a interferencia: (a) Origen, (b) desplazamiento forzado, (c) retorno y (d) posición final.

Capítulo 6

Conclusiones

Para terminar realizaremos un repaso de los objetivos alcanzados durante el desarrollo de este trabajo, los métodos utilizados y las dificultades encontradas en el proceso. Gracias a ellas se ha aprendido durante todo el proyecto. A cada elemento diseñado se han encontrado mejores formas de hacerlo en un futuro, alternativas con propiedades distintas u opciones interesantes que se recopilan en la sección de Trabajos futuros. En elementos sueltos o incluso en la propia arquitectura, hay cambios posibles que podrían dar resultados diferentes y en ocasiones mejores.

6.1. Conclusiones

El objetivo general del TFG, que es hacer un dron comercial programable desde un ordenador y que tenga vuelo estable, se ha conseguido satisfactoriamente. Para ello se han ido resolviendo los tres subobjetivos planteados en el Capítulo 2, que articulan el objetivo global.
Primero se han incorporado en el vehículo volador los elementos de sensorización necesarios para conocer la posición del mismo. Además se le ha dotado de un enlace radio adicional con la estación de tierra para comunicar dicha posición. Segundo, en la estación una FPGA recibe periódicamente las medidas y ejecuta controladores de vuelo con los datos recibidos. Sus resultados generan órdenes de movimiento que son enviadas a la plataforma voladora, consiguiendo un vuelo estacionario y estable. Tercero, el gobierno del dron se ha completado modulando dichas órdenes de movimiento a través de instrucciones en Python ejecutadas en un PC. Éste se comunica mediante USB con la FPGA de la estación. El software del PC se encarga de dirigir la posición del vehículo haciendo uso de las funciones construidas en la librería de Python, per-

mitiendo así manejar el vuelo mediante un entorno simplificado de alto nivel, siendo la FPGA la encargada de que el vehículo obedezca órdenes de manera estable.

El proceso hasta alcanzar estos objetivos ha incluido el desarrollo de controles de bucle abierto para el vehículo aéreo. El resultado es positivo desde el punto de vista de lo que este tipo de sistemas pueden llegar a ofrecer. Se ha diseñado software y hardware funcional que permiten llevar a cabo el manejo teledirigido del dron mediante instrucciones ejecutadas en el PC, mínimamente tratadas por la FPGA de la estación de tierra. Pero las derivas propias de los drones, especialmente notables en drones de bajo coste, dificultan excesivamente un buen control del mismo mediante esta metodología de controles en bucle abierto. Los resultados así conseguidos permiten controlar un dron comandándole instrucciones que éste obedece, pero dicho control es precario, falto de precisión y repetitividad. Para ejercicios de vuelo estacionario en los que transcurre cierto tiempo las derivas dirigen el dron contra el suelo, paredes u objetos de manera impredecible. La problemática de este método es difícil de subsanar con la infraestructura disponible.
El control de vuelo mediante bucles cerrados ha sido la arquitectura definitiva diseñada para el control del vehículo. Ésta se ha alcanzado mediante los añadidos hardware y la actualización de software en el dron, la estación de tierra y el PC comentados anteriormente. Mediante estos cambios se consiguen vuelos significativamente más estables y similares entre sí. También permiten ampliar la duración de los vuelos, alcanzando los límites impuestos por el nivel de carga en las baterías. Esto también facilita las tareas de ajuste de parámetros PID, experimentación y ofrece vuelos con una calidad de control muy mejorada.

Tras el trabajo realizado se ha conseguido un resultado satisfactorio que cumple el objetivo principal que se tenía, estabilizar y controlar un dron de bajo coste desde un PC, usando FPGAs libres. Este resultado es posible gracias al diseño de una electrónica lo suficientemente compacta y ligera como para embarcarla en el dron, que le permite transmitir su ubicación hacia la estación de tierra. Además, tanto los sistemas de comunicaciones como la FPGA instalada en la estación, permiten el control del vehículo limitando los retardos. Gracias a todo esto, el vuelo del dron obedece a los comandos ejecutados en el PC de mando.

El trabajo realizado se pone a disposición pública, tanto el hardware, el software[1] como el

[1] https://github.com/JdeRobot/FPGA-robotics/tree/master/Projects/Basic-Drone

propio diseño del sistema[2], listos para que cualquiera persona pueda adquirir los elementos por un bajo coste, instalarlos y ponerlos en funcionamiento. Esto abre un gran abanico de posibles modificaciones, mejoras y desarrollos futuros.

6.2. Trabajos futuros

El sistema desarrollado ofrece muchas posibilidades de mejoras y añadidos. En este apartado se repasan algunas de las más interesantes.

6.2.1. Mejoras en la arquitectura del sistema

- La arquitectura del sistema es posiblemente la reconsideración cuyo cambio tendría mayor impacto en el resultado final. La premisa de procesar con la FPGA en tierra tiene ventajas y desventajas. Probablemente la mayor contrapartida sean los tiempos de transmisión y recepción añadidos por los distintos módulos que intervienen en la comunicación de las medidas hacia tierra y las órdenes de vuelta hacia el dron. Estos retardos añadidos empeoran la estabilidad que se puede llegar a conseguir, con el vehículo en vuelo. El cambio más significativo pasaría posiblemente por embarcar la FPGA a bordo del dron, con conexión directa a los sensores y a los drivers de los motores. Esto supondría incluir una cantidad considerable de software en la FPGA, a cambio de disminuir los retardos actuales a más de la mitad. De esta manera se ganaría principalmente en favor de la calidad del control que se ejerce sobre el dron, aunque requeriría motores más potentes y drones más caros. El diseño utilizado en este TFG ha sido escogido no por capricho, sino tomando como restricción fundamental el objetivo de basar el sistema en drones y FPGAs de bajo coste (coste aproximado del dron 24€ y de la FPGA en torno a 5€). La elección de la FPGA tuvo como motivación adicional la compatibilidad con herramientas libres para futuros diseños.

[2]https://github.com/RoboticsLabURJC/2018-tfg-eloy-navarro

6.2.2. Mejoras hardware

- Diseñar una única tarjeta para el sistema embarcado (sensores y radioenlace) es una mejora que aportaría ventajas tanto con la arquitectura del sistema actual, como en caso de llevar la FPGA a bordo. En ambos casos, la disminución de peso y las mejoras en la fijación de la electrónica favorecerían que hubiese menos interacción entre ejes de medida (al disminuir el error de colocación de los sensores de flujo y de altura). La potencia entregada a los motores sería más holgada, mejorando la estabilidad y alargando la duración de los vuelos sin modificar las baterías.

- Aparte de cambios sobre el sistema, se pueden plantear añadidos de interés para el sistema final. Por ejemplo un conjunto de magnetómetros de 2 ejes (ubicados en el plano horizontal) para medir el error de giro, lo que permitiría controlarlo mediante su propio PID. A la vez se ganaría un parámetros de orientación global permitiendo vuelos de largas distancias. También se podría incluir una cámara, dependiendo del peso liberado al rediseñar la electrónica, lo cual mezcla bien con proyectos que hacen uso de ubicación local en base a reconocimiento de etiquetas por imagen [28]. Ambas mejoras ofrecerían más capacidad de vuelo autónomo a través de un mejor conocimiento a bordo sobre la ubicación del propio vehículo.

6.2.3. Mejoras software

- Respecto del diseño software, pueden encontrarse maneras mejores, más eficientes, de construir código igualmente funcional una vez terminado cada módulo. Invertir tiempo en parámetros PID más ajustados, mejoras en el software existente, compactarlo, ejecutando el máximo posible sobre la FPGA (eliminando la necesidad de procesadores periféricos) y cambios de esta índole podrían reducir los retardos de cierre de los bucles, mejorando la respuesta del control.

- Ampliar la librería en python es una mejora que iría en pro de la facilidad de uso. Permitiría la repetición de ensayos con menos esfuerzo.

- Los controladores diseñados actualmente realizan su tarea en base exclusivamente a la posición del vehículo, pero no reparan en limitaciones de velocidad o aceleración. Una

mejora interesante sería ubicar bucles de control de velocidad y aceleración bajo los actuales bucles de control de posición. En principio esto controlaría las brusquedades y dotaría al sistema de un mayor control sobre cómo se desplaza el vehículo.

- Generador de trayectorias: Para acometer un generador de trayectorias completo serían necesarias modificaciones hardware y software en la estación de tierra. Estas deberían permitir a la FPGA informar al ordenador de mando de la ubicación actual del dron (obtenida de las medidas de los sensores embarcados). Con este añadido se permitiría al software ejecutado en el PC crear puntos de paso intermedios en el avance de dron, dependientes tanto de la posición actual, como de la posición de destino. Esto permitiría trazar una trayectoria sin incrementos exagerados en el error de los bucles de control.

Bibliografía

[1] Life seeker. sistema para la localización de personas desde el aire, 2012.

[2] Drone electronic speed controller(esc), 2016.

[3] Amazon. First prime air delivery, 2016.

[4] Bitcraze. Flow breakout, 2019.

[5] S. Centum research & technology. Lifeseeker, 2019.

[6] Charlie. Modeling vehicle dynamics, quadcopter equations of motion, 2019.

[7] dirigentesdigital. Una empresa española entre los mayores fabricantes de drones, 2019.

[8] J. Francisco Martínez Lendech. Datos analógicos, transmisión digital.

[9] J. Ganssle. A designer's guide to mems sensors, 2012.

[10] goebish. nRF24L01 multi-protocol RC transmitter. `https://github.com/goebish/nrf24_multipro`, Feb. 2019.

[11] Jero. Â¿qué son los pids?, 2017.

[12] L. Kelion. Drone-based blood deliveries in tanzania to be funded by uk, 2016.

[13] L. llamas. El bus spi en arduino, 2016.

[14] J. A. C. Montes. First prime air delivery, 2006.

[15] E. Navarro. Basic-Drone. `https://github.com/JdeRobot/FPGA-robotics/tree/master/Projects/Basic-Drone`, July 2019.

[16] E. Navarro. Syma Closed Loop Against Interference. https://www.youtube.com/watch?v=V-kKepw79hg, July 2019.

[17] E. Navarro. Syma Closed Loop Forward and Reverse Movement. https://www.youtube.com/watch?v=EGjc1TjfpkM, July 2019.

[18] E. Navarro. Syma Closed Loop Takeoff and Landing. https://www.youtube.com/watch?v=U2gZdFM8GPE, July 2019.

[19] J. Ordóñez Cerezo. Control system in open fpgas for autonomous robots. 2018.

[20] J. Ordóñez Cerezo, E. Castillo Morales, and J. M. Canas Plaza. Control system in open-source fpga for a self-balancing robot. *Electronics*, 8(2):198, 2019.

[21] S. Parrot. Parrot sequoia+, 2019.

[22] V. Philavanh. Fpgas in neural networks, 2017.

[23] pixart. Optical motion tracking, 2019.

[24] E. P. S.L. El emocionante rescate gracias a un dron de un escalador dado por muerto en el himalaya, 2018.

[25] solitontech. Uart protocol validation service, 2019.

[26] STMicroelectronics. World smallest time-of-flight (tof) ranging sensor, 2019.

[27] teslabem. Fundamentos del protocolo i2c, 2017.

[28] J. Vela Peña and J. M. Canas Plaza. Despegue, navegación y aterrizaje visuales de un drone usando jderobot. 2019.

[29] J. Zhu and P. Sutton. Fpga implementations of neural networks–a survey of a decade of progress. In *International Conference on Field Programmable Logic and Applications*, pages 1062–1066. Springer, 2003.

Apéndice A

Mecanismos de comunicación

A lo largo de este trabajo, se hace uso de diferentes estándares y protocolos de comunicación para enlazar distintos módulos, permitiéndoles comunicarse entre ellos de una manera determinada. Aquí se comentan algunos de los mecanismos empleados en el informe.

A.1. UART

Se trata de un sistema de comunicaciones serie asíncronas (Universal Asynchronous Receiver Transmitter) capaz de realizar una comunicación full dúplex bidireccional con tan solo dos hilos, transmisión y recepción (más un tercero, a modo de referencia, si los dispositivos en comunicación no gozasen de una referencia común de tensión). Se utiliza cada hilo exclusivamente para cada sentido de la comunicación.
En el sistema se utilizan comunicaciones asíncronas en dos ubicaciones distintas. En la Figura A.1 se muestra la transmisión de un byte que incluye bit de paridad [25]. En el trabajo realizado, este bit no está en uso, el resto de la configuración es idéntica a la utilizada. Se comienza con el bus a nivel alto. El primer flanco de bajada marca la llegada del bit de inicio de paquete, “START”. Seguido de los ocho bits de datos comenzando por el bit de menor peso “D0”, y acabando con un último bit de parada “STOP”, este indica el final del paquete. La línea roja del cronograma de la Figura A.1, muestra el instante de detección del bit de inicio, y los posteriores instantes de muestreo de cada bit. Los bits se reciben a la tasa de transferencia preestablecida de 500Kbps, en caso contrario, se perdería la sincronía con el byte en transmisión, pudiendo perder, o recibir prematuramente el bit de parada (dependiendo de si el error en

la tasa de transferencia es por exceso o por defecto), produciéndose un error de “framing”. En caso de transferirse adecuadamente, el valor del byte estaría disponible al recibir el último bit transmitido, “D7”.

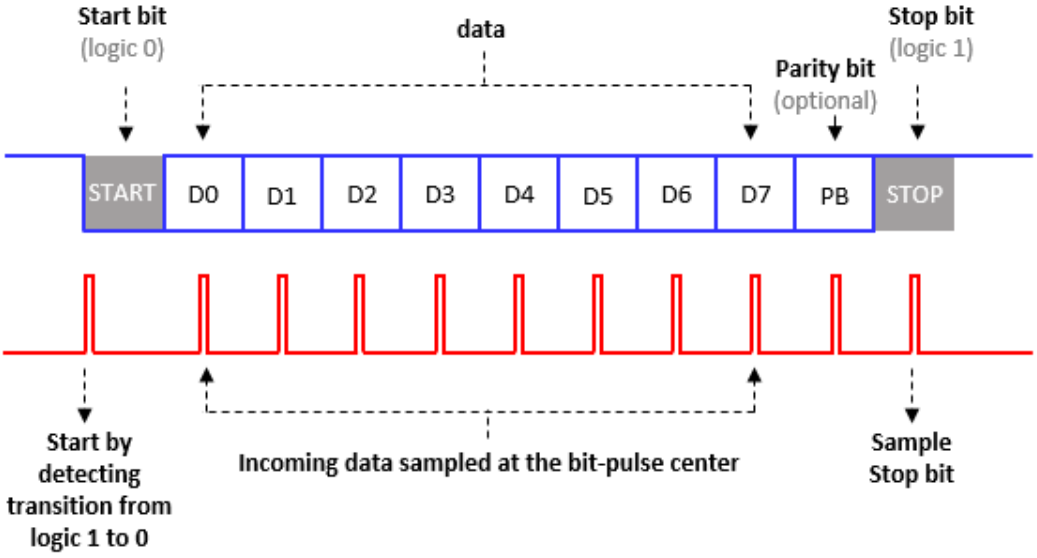

Figura A.1: Formato de paquete UART

A.2. SPI

A lo largo del TFG se hace uso de puertos serie, SPI, en tres ubicaciones distintas, dos en la estación de tierra y la tercera en la electrónica embarcada.

Un puerto SPI se trata de una comunicación maestro esclavo. Esta se basa en una comunicación serie, síncrona de tres o cuatro hilos para un enlace bidireccional, con un único dispositivo cada vez [13]. En la Figura A.2 se muestra un ejemplo de comunicación, y en la Figura A.3 un ejemplo de las conexiones. La única señal opcional es la selección del esclavo, Slave Select (SS). Las tres señales obligatorias son el reloj (SCLK), la salida de datos del maestro y entrada hacia el esclavo (MOSI) y la entrada de datos del maestro y salida del esclavo (MISO).

- SS: En caso de tener conexión a múltiples dispositivos, el maestro puede escoger con cual entablar comunicación, asertando este pin, activo a nivel bajo, en el esclavo deseado.
- SCLK: Se trata de la señal de reloj. Se utiliza para sincronizar las transmisiones. En el

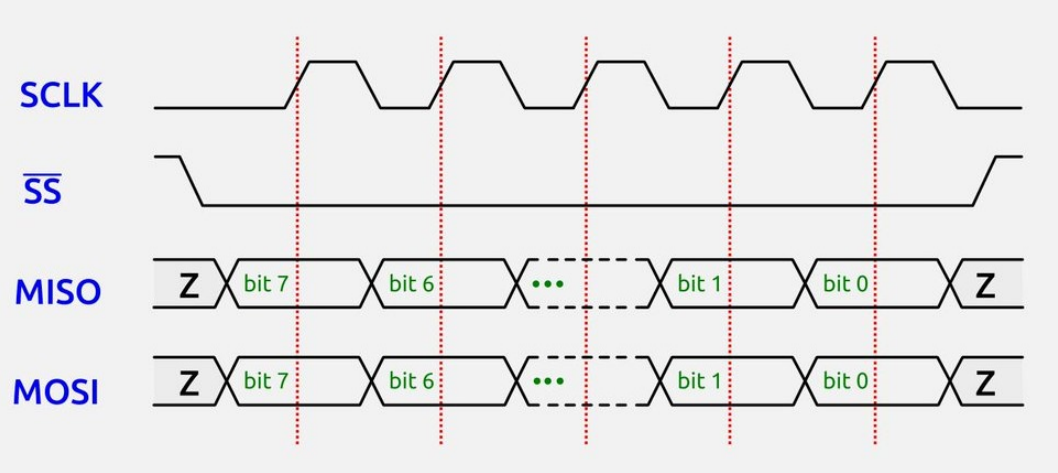

Figura A.2: Funcionamiento de una comunicación SPI

flanco de bajada se deben ubicar los datos en la salida correspondiente, y se deben leer en el flanco de subida.

- MISO: Master Input Slave Output. Convención utilizada para designar el pin de entrada de datos hacia el maestro, desde los posibles esclavos.

- MOSI: Master Output Slave Input. Ofrece la funcionalidad complementaria al pin MISO. En este caso, es el maestro el que utiliza este pin de salida y los esclavos como entrada.

Un esquema típico de conexión con múltiples esclavos se muestra en la Figura A.3. Para este trabajo, se hará uso de comunicaciones con un único esclavo únicamente, ya que se dispone de tres radios NRF24L01 y tres procesadores de Atmel independientes.

A.3. I2C

Se trata de un desarrollo de Philips para la conexión de múltiples circuitos integrados de semejante o distinta naturaleza, sobre un bus común.
De manera similar al estándar de SPI, I2C hace uso de una arquitectura maestro-esclavo, síncrona, bidireccional, en este caso half-dúplex, mediante dos hilos: Reloj (normalmente designado SCL o CLK) y datos (SDA) [27] que se conectan a todos los dispositivos colgados del bus, como se muestra en la Figura A.4.

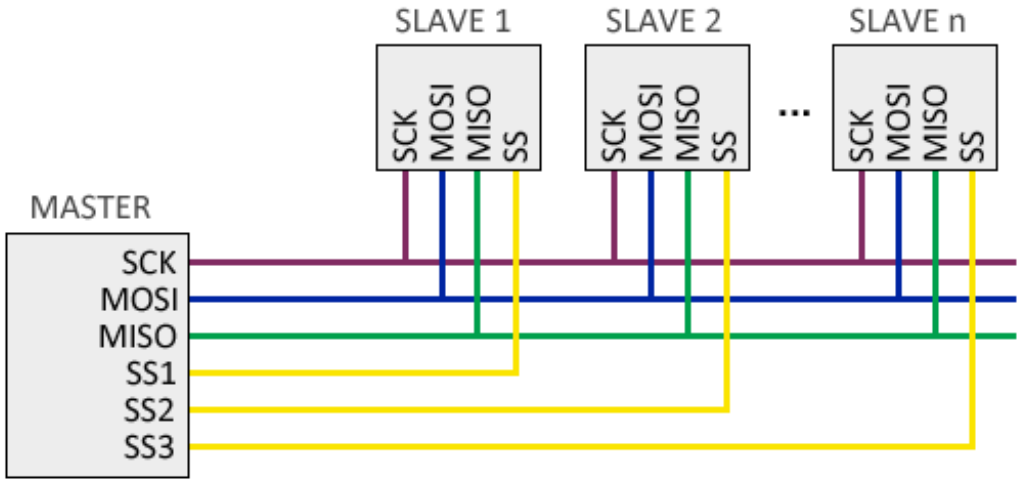

Figura A.3: Esquema de conexión para un puerto SPI

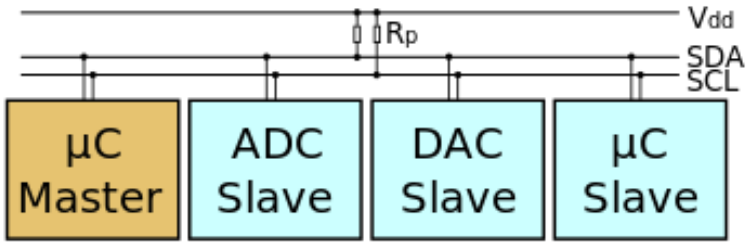

Figura A.4: Arquitectura hardware de un bus I2C

Las líneas de reloj y datos se encuentran, por defecto, a nivel alto gracias a dos resistencias de “pull-up”, Rp en la arquitectura arriba indicada. Para iniciar una comunicación, dichas líneas deben ser comprobadas a fin de evitar colisiones. Si el bus I2C se encuentra libre en ese instante, se puede iniciar una comunicación manteniendo la línea SCL a nivel alto y forzando un flanco de bajada en la línea de datos SDA. Este evento fuerza a los esclavos a atender el siguiente mensaje recibido, por si fuera destinado a ellos. La trama tiene el formato especificado en la Figura A.5. El destinatario se especifica en la trama enviada a través del primer grupo de 7 a 10 bits, dependiendo del dispositivo, seguido del bit que especifica el sentido de la información; se desea enviar información hacia el esclavo, o se quiere solicitar información de él. Tras cada grupo de datos transferidos, se envía un ACK o un NoACK al remitente, para informarle si el paquete ha sido recibido con éxito. La trama se termina con el envío de un bit de parada, señalado por un flanco de subida en la señal SDA, mientras se mantiene la línea SCL a nivel alto.

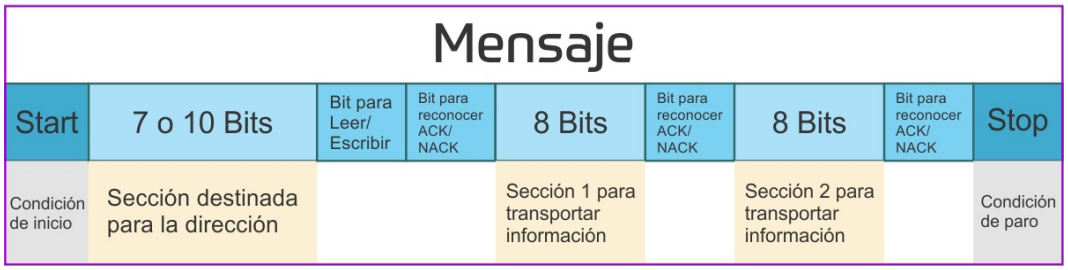

Figura A.5: Formato de trama I2C

A.4. PPM

Las siglas PPM vienen de Pulse Position Modulation. Se trata de una modulación en la que cada canal tiene un instante de llegada concreto, manteniendo constantes la amplitud de los mismos [8], respecto del canal inmediatamente anterior. El desfase que se produzca respecto de dicho instante se transforma en el valor que se asigna a ese canal. De esta manera pueden codificarse todos los canales utilizados sobre un único conductor mediante multiplexación en tiempo. El aspecto general de una trama se muestra en la Figura A.6. En ella se tiene seleccionado el canal 1 para mostrar su temporización.

Una trama completa PPM para el dron SYMA-X5C utilizado (las características básicas de PPM se mantienen entre distintos fabricantes, pero particularidades como el número y orden de los canales puede variar) se compone de doce canales. El inicio de trama se marca con un pulso a nivel bajo de 0.3 milisegundos. A partir de este instante se comienza el conteo de cada canal, indicado por los niveles altos y separados entre ellos a por pulsos de bajada de 0.3ms. La trama termina con un último pulso de bajada de 0.3ms que indica el fin de trama. Cada canal tiene una duración mínima asignada de 1ms y una duración máxima de 2ms contando con su flanco de 0.3ms de bajada. Estos tiempos, sumados al pulso de inicio de trama, completan la duración de una trama con un máximo de 24.3ms. Cada canal por tanto tendrá su valor mínimo para una duración a nivel alto de 0.7ms (canal 1 de la Figura A.6) y su valor máximo para 1.7ms.

En este TFG se hace uso de esta técnica, dado que el módulo que maneja esta interfaz estaba originalmente diseñado para recibir la modulación PPM de un mando de radio control. PPM es la modulación más extendida en la mayoría de mandos de radio control para conectarse a módulos externos de bandas, potencias y otras características distintas a las suyas nativas, por tanto resulta cómodo adaptarse a esta modulación, para comunicarse con el módulo de uplink.

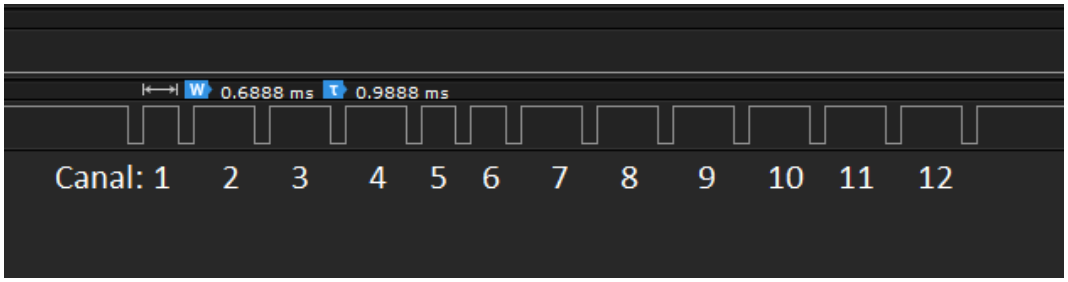

Figura A.6: Señal PPM hacia el módulo de transmisión de uplink

Apéndice B

Sistemas de control

Los sistemas de control de un dron se basan principalmente en bucles de control PID [11], ejecutados por la electrónica de control antes mencionada. Estos sistemas son la herramienta principal para garantizar la estabilidad de vehículo. Los sistemas de control ejecutados dirigen la corriente en cada motor para mantener el sistema estable. Además de los sistemas comunes a todo dron, este TFG hace uso de sus propios bucles de control para conseguir su objetivo. Estos bucles, trabajan de manera simultánea y paralela a los propios del dron. A efectos del trabajo realizado, los bucles de control implementados en este TFG se consideraran los de mayor nivel, ya que son los encargados de comandar a los intrínsecos del dron, considerados de bajo nivel.
Los sistemas de control usados en este TFG son de dos tipos, bucle abierto y bucle cerrado, cada uno utilizado de una manera concreta en distintas partes del sistema.

B.1. Bucle abierto

Los bucles abiertos reciben una señal y producen una respuesta concreta resultado de convolucionar la entrada, con la función de transferencia del bucle. Pueden ser lineales e invariantes en el tiempo, o no, dependiendo de la necesidad concreta que deban cubrir. Su respuesta es enviada hacia los actuadores, o bucles de siguiente nivel como se muestra en la Figura B.1. La respuesta del controlador de bucle abierto es independiente de la situación del dron, respuestas de sensores, posición, ubicación y demás. Por tanto, asumiendo el caso de un bucle de control abierto invariante y sin memoria, si se ejecuta en dos ocasiones con la misma señal de entrada, producirá dos veces la misma señal de salida.

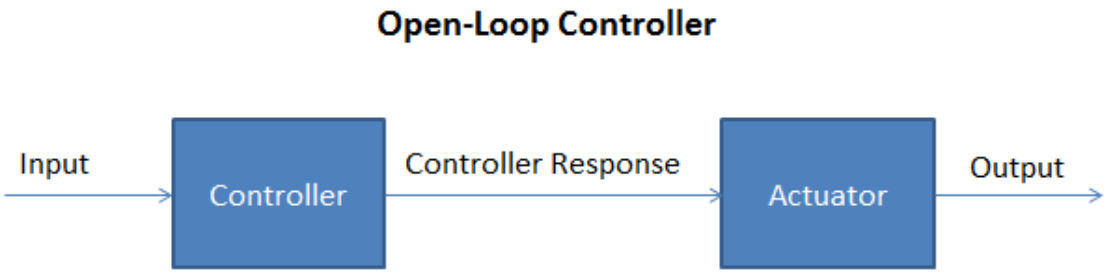

Figura B.1: Esquema básico de un sistema de control de bucle abierto

En relación al trabajo realizado, este tipo de bucle se usa por ejemplo en el control de la guiñada, en la cual, es un bucle cerrado de bajo nivel el que se encarga de girar lo indicado por el bucle abierto de nivel superior programado en la FPGA de control.

El comportamiento de este tipo de bucles difiere de los bucles cerrados mencionados en el siguiente punto.

B.2. Bucle cerrado

Los bucles cerrados hacen uso de dos señales de entrada. Una de ellas es el comando, es la señal que indica el objetivo a alcanzar en algún parámetro concreto por el bucle. Otra es la señal recibida de alguna otra parte del sistema o sensor, capaz de medir e informar sobre el estado del parámetro objetivo. Ambas señales se restan y producen una señal de error que es entregada al controlador [14], como se muestra en la Figura B.2. Este opera de la misma manera que para el bucle abierto, generando una respuesta que es entregada a otro módulo, ya sea un bloque de menor nivel o un actuador.

Estos bucles de control si tienen en cuenta el estado actual del parámetro objetivo para realizar su corrección. Al realizar una medición sobre el mismo, pueden acomodar la respuesta del controlador a valores ajustados a la situación actual del parámetro a modificar. Pudiendo darse el caso, de que para una entrada distinta de cero, la salida del sistema hacia los actuadores, sea cero. Este sería el caso en el que el valor comandado en la entrada, es igual que el valor

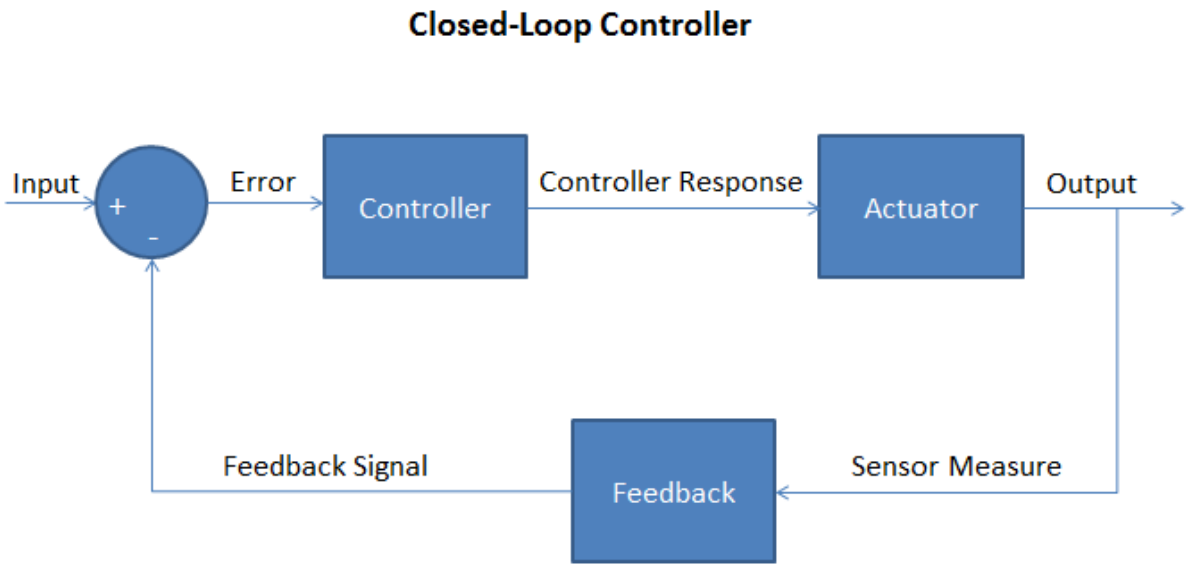

Figura B.2: Esquema básico de un sistema de control de bucle cerrado

medido por el sensor, generando por tanto una señal de error cero (asumiendo por ejemplo una respuesta proporcional del controlador).

B.3. Algoritmos de control PID

Este TFG hace uso de controladores PID. Ejecuta esta clase de sistema de control a través de algoritmos que calculan el error como la diferencia entre una señal de comando (instrucción de mando desde PC) y una señal entrante (medida de un sensor). Dicho error (e(t) en la Figura B.3) entonces pasa por tres procesos paralelos.

- Un proceso de proporción: Su multiplica el error por una constante proporcional según la ecuación B.1. El objetivo es aproximar el error a cero a través de actuar sobre el sistema, de manera proporcional al error existente en cada evaluación del bucle.

$$P[n] = Kp \cdot e[n] \tag{B.1}$$

- Un proceso de integración: El error recibido se suma en una variable de acumulación. Esta integra la medida de error actual, junto con todas las anteriores recibidas según la ecuación B.2. El resultado es multiplicado por una constante de integración. El objetivo de

esta componente, es eliminar error estacionario inalcanzable para el control proporcional. Una vez regulado el valor de la constante proporcional, ocurre que cierto error sigue presente. Este error se suma a sí mismo en cada ejecución del bucle, de tal manera que la acumulación crece lo suficiente como para que se produzca una respuesta sobre los actuadores, capaz de corregirlo. Se puede deducir que una vez corregido en un caso idea, el valor del acumulador dejaría de variar.

$$I[n] = Ki \sum_{0}^{n} e[n] \tag{B.2}$$

- Un proceso de derivación: El error previamente almacenado se resta al error de este instante generando la componente derivativa de la ecuación B.3. El resultado es multiplicado por una constante de derivación. Los dos bucles anteriores tienen el objetivo de corregir el error. En el desarrollo de su tarea, existe la posibilidad de que el sistema oscile. La componente proporcional puede ajustarse para trabajar fuera del régimen de sobreoscilación, pero esto no ocurre con la componente integral, la cual por definición genera un retardo en la respuesta de los actuadores que produce oscilación. Esta puede ser tan pequeña que se vuelva imperceptible, de no ser así, la componente derivativa puede ayudar a minimizar las oscilaciones. Ya que calcula la derivada del error, cuanto este cambia bruscamente, por ejemplo cuando el sistema se acerca rápidamente a la posición de la consigna, la derivada en el tiempo también crecerá, pero en sentido contrario. De tal manera que la componente derivativa ayudará entonces a frenar el sistema, dotando al controlador de un grado de suavidad en el acercamiento hacia el error. Esto disminuye las oscilaciones, por tanto se puede deducir que esta componente resultará en cero en caso de un sistema ideal sin oscilaciones, ya que el error no variaría.

$$D[n] = Kd \cdot (e[n] - e[n-1]) \tag{B.3}$$

El resultado de los 3 procesos previos se suma en una única señal de error construyendo la señal u(t) como se muestra en la Figura B.3.

El resultado de la suma de la Ecuación B.4 será la variable usada para tratar de corregir el error existente. Es decir, será la variable a entregar a los distintos actuadores del sistema, dependiendo de en que parte se encuentre el PID ejecutado.

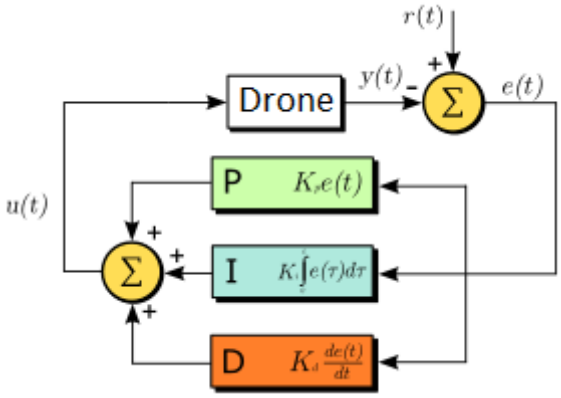

Figura B.3: Esquema de un controlador PID

$$u[n] = P[n] + I[n] + D[n] \tag{B.4}$$