



**ESCUELA TÉCNICA SUPERIOR DE INGENIERÍA DE  
TELECOMUNICACIÓN**

Curso Académico 2019/2020

Trabajo Fin de Grado/Máster

**ESTABILIZACIÓN DE UN DRONE MEDIANTE  
FPGAs LIBRES**

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 de un sistema, no solo como elemento para la agrupación de operaciones lógicas sencillas, resulta natural mezclar dichos dispositivos para beneficio mutuo. A lo largo del proyecto se han diseñado distintos elementos software y hardware para permitirle a un conjunto de instrucciones ejecutadas en un ordenador, controlar por completo las operaciones y trayectoria de vuelo de un drone concreto, tanto en bucle abierto como cerrado. Este es capaz de obedecer las instrucciones de posición indicadas por el programa ejecutado, sin estabilización, control o intervención humana de por medio. Para ello se han diseñado elementos tanto en tierra, como a bordo del drone, que dotan al sistema de control autónomo sobre el vehículo. El control puede realizarse desde cualquier ordenador a través de la instalación de un paquete de librerías y la posterior ejecución de un programa en java. Todo el sistema se ha diseñado en base a FPGAs libres, drones de bajo coste y electrónica de tierra y a bordo del drone diseñados específicamente para la ocasión. De esta manera se dispone del control completo sobre el sistema diseñado, lo que permitiría mejoras y modificaciones futuras completas sobre el sistema, sin necesidad de incurrir en elevados costes de adquisición de software o hardware. Tras el diseño de los distintos elementos software y hardware, se han realizado verificaciones de cada elemento por separado y posteriormente se ha validado el sistema en su conjunto, mediante pruebas experimentales. Se han conseguido resultados especialmente satisfactorios con los sistemas en bucle cerrado, tanto en la estabilización como en el control del drone escogido.



# **Summary**

Here comes a translation of the “Resumen” into English. Please, double check it for correct grammar and spelling. As it is the translation of the “Resumen”, which is supposed to be written at the end, this as well should be filled out just before submitting.



# Índice general

<b>1. Introducción</b>	<b>1</b>
1.1. Sistemas básicos del drone . . . . .	1
1.1.1. Estructura . . . . .	2
1.1.2. Sensores . . . . .	3
1.1.3. Electrónica de Control . . . . .	4
1.1.4. Drivers y motores . . . . .	5
1.1.5. Comunicaciones . . . . .	6
1.2. Aplicaciones de vehículos aéreos no tripulados . . . . .	7
1.2.1. Agricultura . . . . .	7
1.2.2. Seguridad y rescate . . . . .	8
1.2.3. Envíos . . . . .	8
1.3. Sistemas de control . . . . .	9
1.3.1. Bucle abierto . . . . .	10
1.3.2. Bucle cerrado . . . . .	10
1.3.3. Algoritmos de control PID . . . . .	11
1.4. FPGA . . . . .	13
1.4.1. Concepto . . . . .	13
1.4.2. Características . . . . .	14
1.4.3. Aplicaciones . . . . .	15
1.4.4. FPGAs Libres . . . . .	15
1.5. Mecanismos de comunicación . . . . .	16
1.5.1. UART . . . . .	17
1.5.2. SPI . . . . .	18

1.5.3. I2C . . . . .	19
1.5.4. PPM . . . . .	20
<b>2. Objetivos</b>	<b>23</b>
2.1. Objetivo principal . . . . .	23
2.2. Sub-objetivos . . . . .	23
2.2.1. Enlace con el drone . . . . .	23
2.2.2. Comunicación de órdenes fijas al drone . . . . .	23
2.2.3. Diseño de librerías de control para PC . . . . .	24
2.2.4. Comando del drone en bucle abierto . . . . .	24
2.2.5. Diseño de la estación de tierra . . . . .	24
2.2.6. Diseño de electrónica de a bordo . . . . .	24
2.2.7. Cierre de bucles para cada eje . . . . .	24
2.2.8. Experimentación y parches . . . . .	25
2.3. Requisitos . . . . .	25
2.4. Metodología . . . . .	25
2.5. Plan de trabajo . . . . .	26
<b>3. Infraestructura utilizada</b>	<b>27</b>
3.1. IceCube2 . . . . .	27
3.2. Quartus Prime . . . . .	27
3.3. Arduino IDE . . . . .	27
3.4. ModelSim . . . . .	28
3.5. FT_Prog . . . . .	28
3.6. Logic . . . . .	28
3.7. Logic Analyzer . . . . .	29
3.8. Arduino Uno y Nano . . . . .	29
3.9. ICE40 UltraPlus Breakout Board . . . . .	29
3.10. EACHINE E010 . . . . .	30
3.11. SYMA X5C . . . . .	30
3.12. NRF24L01 . . . . .	30
3.13. Flow breakout board . . . . .	30

<b>ÍNDICE GENERAL</b>	<b>VII</b>
<b>4. Arquitectura del sistema</b>	<b>31</b>
<b>5. Interfaces del sistema</b>	<b>33</b>
5.1. Interfaz de mando . . . . .	33
5.1.1. Características . . . . .	33
5.1.2. Formato de tramas . . . . .	34
5.2. Interfaz radio . . . . .	37
5.2.1. Downlink . . . . .	38
5.2.2. Uplink . . . . .	40
<b>6. Módulos del sistema</b>	<b>43</b>
6.1. Estación de tierra . . . . .	43
6.1.1. Descripción . . . . .	43
6.1.2. Objetivo . . . . .	44
6.1.3. Funciones . . . . .	45
6.1.4. Interfaces . . . . .	46
6.1.5. Módulos . . . . .	47
6.2. Sistema embarcado . . . . .	50
6.2.1. Descripción . . . . .	50
6.2.2. Objetivo . . . . .	51
6.2.3. Funciones . . . . .	52
6.2.4. Módulos . . . . .	52
6.2.5. Interfaces externos . . . . .	54
6.2.6. Interfaces internos . . . . .	54
<b>7. Módulos Software</b>	<b>57</b>
7.1. Algoritmos de control sobre la FPGA . . . . .	57
7.2. Software en el módulo de Downlink embarcado . . . . .	59
7.3. Librería Python . . . . .	60
<b>8. Validación experimental</b>	<b>63</b>
8.1. Eachine E010 . . . . .	63
8.2. Syma X5C en Bucle abierto . . . . .	63

8.3. Condiciones de los bucles de control . . . . .	65
8.3.1. Ancho de banda . . . . .	65
8.3.2. Retardo . . . . .	65
<b>9. Conclusiones</b>	<b>67</b>
9.1. Trabajo futuro . . . . .	68
9.1.1. Mejoras en la arquitectura del sistema . . . . .	68
9.1.2. Mejoras hardware . . . . .	69
9.1.3. Mejoras software . . . . .	69

# Índice de figuras

1.1.	Sentido de rotación en cuadricóptero . . . . .	2
1.2.	Error de rotación . . . . .	3
1.3.	Masa y capacímetros de STMicroelectronics . . . . .	3
1.4.	Bloques Hardware del sistema de control y drivers de un cuadricóptero . . . . .	5
1.5.	Altura (Uz), cabeceo (pitch), guiñada (yaw) y alabeo (roll) . . . . .	6
1.6.	Sensor óptico Sequoia+ . . . . .	7
1.7.	Módulo LifeSeeker de Centum . . . . .	8
1.8.	Drone de Amazon en prueba real de envío . . . . .	9
1.9.	Esquema básico de un sistema de control de bucle abierto . . . . .	10
1.10.	Esquema básico de un sistema de control de bucle cerrado . . . . .	11
1.11.	Esquema de un controlador PID . . . . .	13
1.12.	Logo de Icestudio, IDE para desarrollo de software en FPGAs libres . . . . .	16
1.13.	Captura del entorno Icestudio . . . . .	16
1.14.	Formato de paquete UART . . . . .	17
1.15.	Funcionamiento de una comunicación SPI . . . . .	18
1.16.	Esquema de conexión para un puerto SPI . . . . .	19
1.17.	Arquitectura hardware de un bus I2C . . . . .	19
1.18.	Formato de trama I2C . . . . .	20
1.19.	Señal PPM hacia el módulo de transmisión de uplink . . . . .	21
4.1.	Arquitectura de sistema y su bucle de control . . . . .	32
5.1.	Formato de trama de mando, primera mitad de la trama . . . . .	34
5.2.	Formato de trama de mando, segunda mitad de la trama . . . . .	35

5.3.	Formato de trama de downlink . . . . .	39
5.4.	Formato de trama de uplink . . . . .	40
6.1.	Arquitectura e interfaces de la estación de tierra . . . . .	44
6.2.	Montaje de la estación de tierra . . . . .	45
6.3.	Arquitectura e interfaces del sistema sensor de posición ubicado en el drone . .	50
6.4.	Sistema de sensores embarcado . . . . .	51
7.1.	Arquitectura del firmware programado sobre la FPGA . . . . .	59

# **Capítulo 1**

## **Introducción**

En la actualidad se ha extendido con rapidez el uso de drones y FPGAs, 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. Es un factor fundamental en todas las aplicaciones para drones la capacidad de control sobre el mismo. Los drones son útiles porque se les puede estabilizar junto con la carga útil y controlar. Dicho control puede realizarse mediante distintos dispositivos, entre ellos FPGAs. Ciertas características convierten a las FPGAs en una elección sólida como núcleo del sistema de estabilización y dirección para los múltiples grados de libertad de un drone. En este TFG se hace uso de FPGAs libres para cubrir dichas tareas de control y estabilización fomentando además la propagación y uso de código abierto.

A continuación se introducen algunas de estas características, junto con distintas aplicaciones y elementos que toman partido en la construcción del sistema

### **1.1. Sistemas básicos del drone**

Los drones se componen de distintos subsistemas para realizar sus tareas. Estos subsistemas cambian según el tipo de drone. Este TFG concretamente hace uso de un cuadricóptero comercial X5C del fabricante Syma. Por tanto se mencionan algunos sistemas propios de los cuadricópteros que no trabajan igual, o no se encuentran presentes necesariamente en otros tipos de drones. Algunos sistemas son específicos de cada aplicación, como puedan ser los estabilizadores de cámara usados en aplicaciones de grabación de video. Otros, por el contrario, son

comunes a todos los cuadricópteros, como pueden ser la estructura, sensorización, electrónica de control, drivers de motores, motores y sistemas de gestión y almacenamiento de energía como baterías y reguladores. Además existen sistemas que si bien no son estrictamente necesarios para todas las aplicaciones, rara vez se ven excluidos, como son los sistemas de comunicaciones. A continuación se explican algunos de estos sistemas.

### 1.1.1. Estructura

La estructura base de un drone es donde apoyan y montan el resto de componentes sus principales requisitos son reducir 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 drone. Por su lado, la rigidez es fundamental ya que para un adecuado control del drone, se necesita asumir independencia de ejes. Por ejemplo, si asumimos que partimos de una situación estable, con el drone en vuelo a 1 metro de altura, si los cuatro motores generan impulso adicional por encima del necesario para mantenerse a dicha altura, en una estructura totalmente rígida, el drone sencillamente asciende, ya que el par de rotación de las cuatro aspas se cancela entre sí como se muestra en la Figura 1.1 (asumiendo también el empuje de cuatro motores ideales).

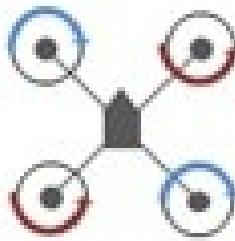


Figura 1.1: Sentido de rotación en cuadricóptero

En caso de que la estructura no sea ideal, si por ejemplo asumimos una ligera asimetría de rigidez en uno de los ejes de apoyo de motores del drone, durante el empuje para ascender, sufriría una fuerza de torsión mayor que los demás, lo que forzaría al drone a girar sobre sí mismo como muestra la Figura 1.2.

Dicho error de rotación podría ser corregido por otros sistemas del drone que se comentan a continuación. Pero solo pueden corregirse hasta cierto punto, además de ocupar dichos sis-

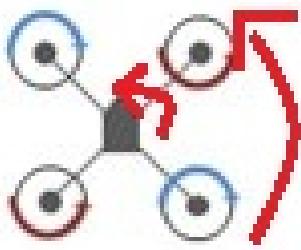


Figura 1.2: Error de rotación

temas en corregir errores que la propia estructura puede paliar si se construye adecuadamente, pudiendo llegar a saturarlos e impedirles realizar su tarea adecuadamente.

### 1.1.2. Sensores

Parte fundamental de todo cuadricóptero es un conjunto de sensores capaz de medir aceleraciones y giros. Para la mayoría de los drones comerciales incluido el utilizado en este TFG, se hace uso de acelerómetros y giróscopos MEMS, es decir elementos mecánicos y electromecánicos miniaturizados e integrados en solo chip. Por ejemplo un giróscopo y acelerómetro MEMS es en realidad un capacímetro diferencial conectado a un conjunto de láminas intercaladas, unas fijas al sustrato y otras libres como se muestra continuación en la Figura 1.3.

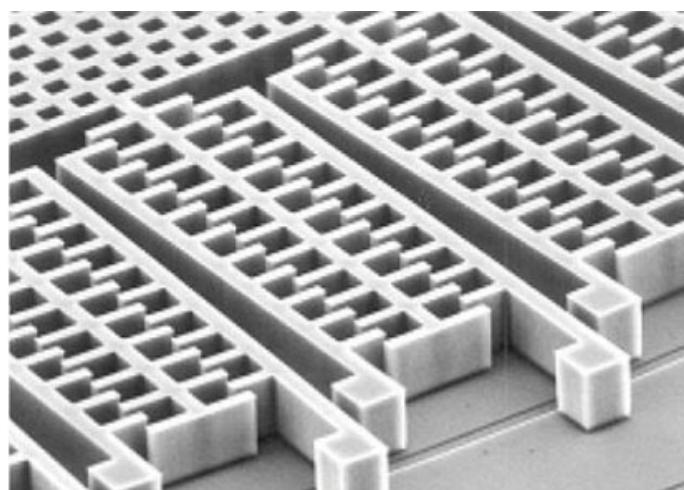


Figura 1.3: Masa y capacímetros de STMicroelectronics

Cuando se produce una aceleración sobre el chip, las láminas fijas al sustrato se desplazan con él, acercándose a ciertas láminas libres y alejándose de las demás, viendo por tanto su me-

dida diferencial de capacidad alterada de manera proporcional a la aceleración, en base a la segunda ley de Newton. Esta construcción se realiza para cada eje y se integra en un solo chip, a menudo llamada plataforma inercial de tres ejes. De esta manera se facilita su integración en el sistema y se abaratan costes.

Como en todo, hay calidades que definen las capacidades finales de un sistema. Para el caso de una plataforma inercial la resolución es un parámetro fundamental, tanto en la medida de las aceleraciones como de los giros.

Por ilustrar un error de desplazamiento asociado a dichos sensores, cuando el sistema de control de drone se basa en un sensor incapaz de detectar cierto nivel de giro, es decir, una velocidad de giro suficientemente lenta y constante como para quedar por debajo del salto mínimo de velocidad que el sensor es capaz de resolver, ocurrirá que el drone, comenzara a girar a una velocidad considerablemente lenta, pero constante en ese sentido. Al no ser detectado, dicho error continuara creciendo, integrándose en el tiempo y posiblemente volviéndose notorio. Como para el caso de la estructura dicho error también podría ser corregido por otro tipo de sensor, pero a menudo estos no se instalan, debido al incremento del coste. Para esta situación, por ejemplo un sensor que aplicase una posible solución sería un magnetómetros a modo de brújula, que corregiría errores de giro considerablemente grandes, pero para el caso de los drones utilizados en este TFG, ninguno integra ningún sistema adicional para la corrección del giro, además del propio giróscopo. Este tipo de errores aplican a todos los ejes de todos los sensores, con mayor o menor impacto.

### 1.1.3. Electrónica de Control

Este módulo es el encargado de procesar las señales de los sensores, entregando las respuestas necesarias para cubrir dos tareas; estabilizar el drone y en caso de recibir comandos por radio u otro mecanismo, modificar las órdenes a entregar a los sistemas de actuación en proporción. Para el caso de los cuadricópteros objeto de este TFG, los elementos de actuación son cuatro motores, y esta electrónica deberá cubrir ambas tareas, estabilización y modificación del comportamiento en base a comandos radio.

En una situación de vuelo estable, el objetivo de esta electrónica es corregir las aceleraciones y giros con las órdenes adecuadas a los motores. Por ejemplo, si un sensor de aceleración detecta una aceleración a izquierdas, esta electrónica deberá responder generando una fuerza en sentido

contrario, para este caso, consistirá en inclinar el drone 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 drone 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 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.

### 1.1.4. 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 instrucción a fuerza, a través de etapas de potencia que dependen de la construcción de los motores. La estructura general se muestra en la Figura 1.4.

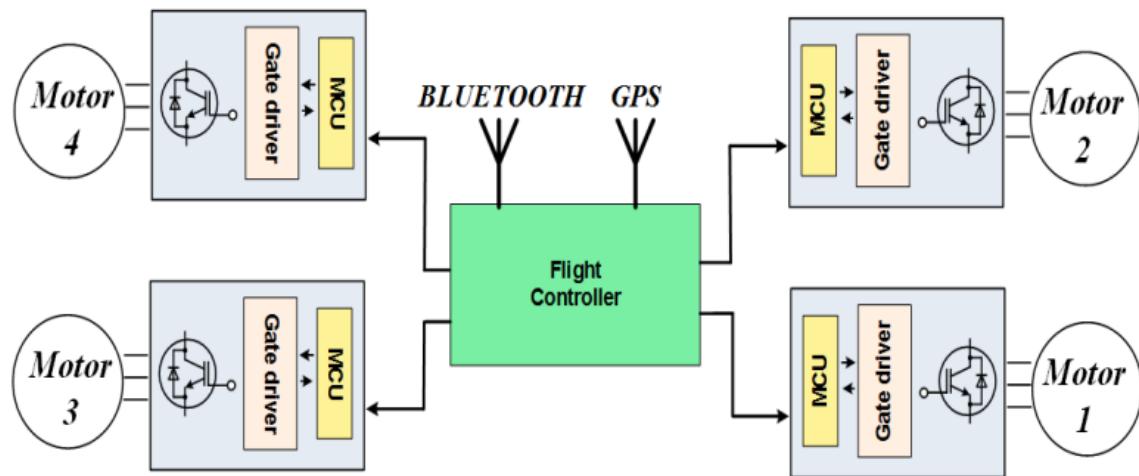


Figura 1.4: Bloques Hardware del sistema de control y drivers de un cuadricóptero

Los motores pueden ser basados en escobillas, o libres de ellas. Cada tipo de motor tiene una forma concreta de ser controlado. 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 desprecian 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. A efectos de este TFG, todos los motores son de corriente continua.

### 1.1.5. Comunicaciones

Dependiendo del tipo de drone, 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 drone, sin necesidad de ver su comportamiento modificado durante el vuelo. Pero tanto para el caso de drones teledirigidos, como para el de este TFG, la radio es un componente fundamental, encargada de comunicar la electrónica del drone, con la electrónica en tierra que tiene por objetivo dirigir el drone. 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 drone. Los cuatro canales comunes a todo drone, y utilizados en esta trabajo son: Altura (Uz), cabeceo (pitch), guiñada (yaw) y alabeo (roll) ordenados como se muestra en la Figura 1.5.

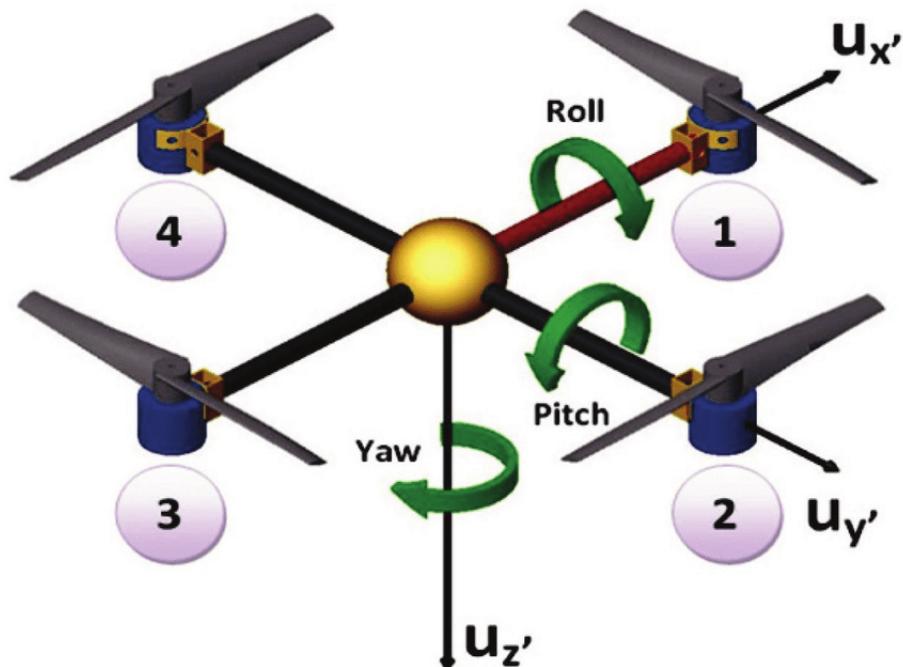


Figura 1.5: Altura (Uz), cabeceo (pitch), guiñada (yaw) y alabeo (roll)

## 1.2. Aplicaciones de vehículos aéreos no tripulados

Un drone tal y como lo conocemos actualmente es una aeronave que vuela sin tripulación. Las aplicaciones de los drones son muchas, desde su inicio en el sector militar como muchas otras tecnologías, se ha extendido su utilización a áreas de otros sectores rápidamente. A continuación se mencionan algunos casos interesantes.

### 1.2.1. Agricultura

Una de las áreas con mayor impacto en el desarrollo y aplicación esta siendo el sector agrícola. Con hectáreas de cultivos que controlar, y cada vez más tipos de sensores, se utilizan desde control de plagas, hasta por ejemplo, exploración de viñedos para determinar el mejor momento para la vendimia. A menudo hacen uso de controladores basados en ubicación global a través de GPS para el control de posición y trayectoria del drone, 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 de la Figura 1.6.



Figura 1.6: Sensor óptico Sequoia+

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

### 1.2.2. Seguridad y rescate

Debido a la ventaja de volar a alturas inferiores que las de un helicóptero, se ha extendido el uso de los drones al rescate de personas. Integrando cámaras de alta resolución en los mismos, y sistemas de localización de desaparecidos, como es el caso del proyecto de seguridad “Life-Seeker” de Centum se podría ubicar a personas desaparecidas. El dispositivo de la Figura 1.7 ha sido diseñado para instalarse en aviones no tripulados. El sistema embarcado hace uso de los móviles de los desaparecidos, aun fuera de las zonas de cobertura, como si se tratases de radiobalizas, permitiendo así su localización.



Figura 1.7: Módulo LifeSeeker de Centum

Otros ejemplos de sistemas basados en drones, los han utilizado para reparto masivo de medicamentos o incluso envíos de sangre para transfusiones.

### 1.2.3. Envíos

En esta aplicación, ha habido controversia relacionada con el control aéreo. La legislación actual dificulta el desarrollo en esta industria a la misma velocidad que se han expandido los drones en otras áreas. No obstante empresas como DHL o Amazon han desarrollado tecnologías al respecto y las han probado e implantado en zonas en las que dichas limitaciones no funcionan de la misma manera. A través de este servicio, Amazon envían paquetes en tiempo inferiores a los 30 minutos usando drones como el de la Figura 1.8. 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 1.8: Drone de Amazon en prueba real de envío

### 1.3. Sistemas de control

Los sistemas de control de un drone se basan principalmente en bucles de control, 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 drone, 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 drone. 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 drone, 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.

### 1.3.1. 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 1.9. La respuesta del controlador de bucle abierto es independiente de la situación del drone, 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.

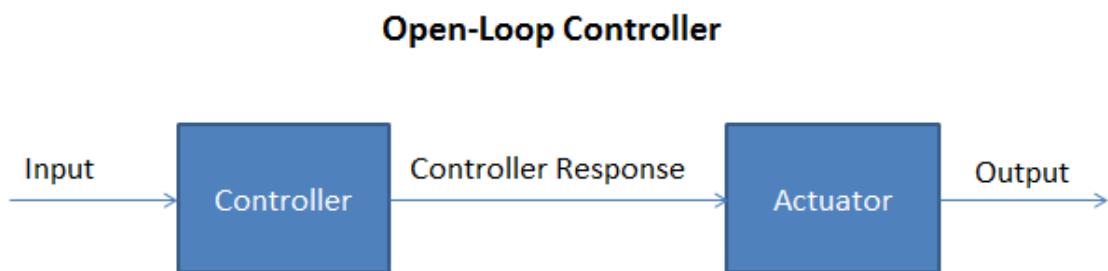


Figura 1.9: Esquema básico de un sistema de control de bucle abierto

En relación al trabajo realizado, este tipo de bucle se usa por ejemplo en el control de la guñada, 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.

### 1.3.2. 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, como se muestra en la Figura 1.10. 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.

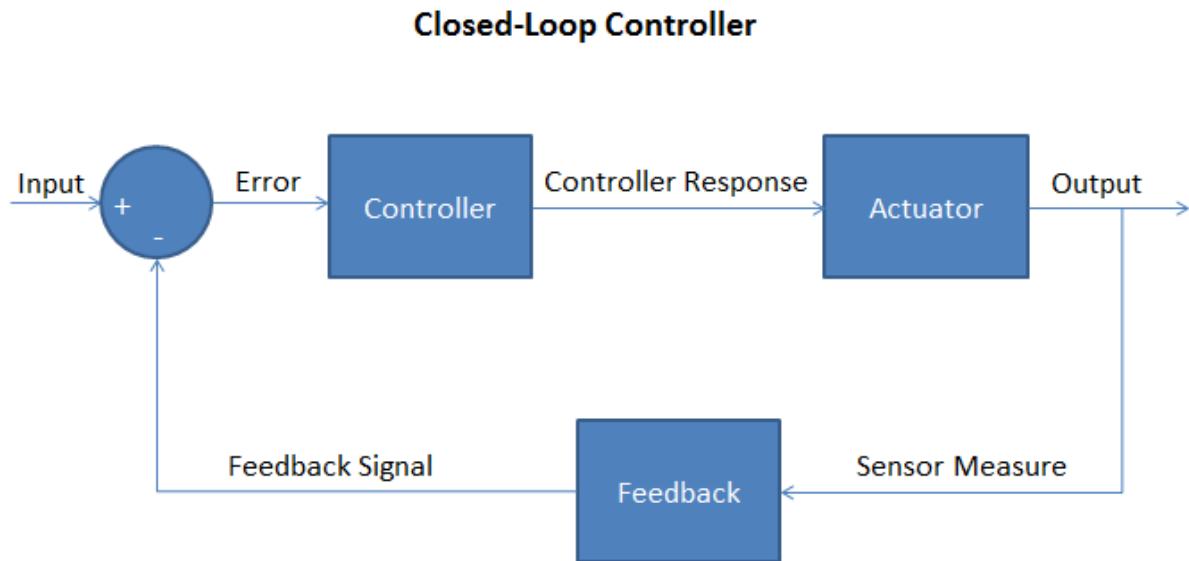


Figura 1.10: Esquema básico de un sistema de control de bucle cerrado

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

### 1.3.3. 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 1.11) entonces pasa por tres procesos paralelos.

- Un proceso de proporción: Su multiplica el error por una constante proporcional según la ecuación 1.1. 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.

$$P[n] = K_p \cdot e[n] \quad (1.1)$$

- 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 1.2. 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 ideal, el valor del acumulador dejaría de variar.

$$I[n] = K_i \sum_0^n e[n] \quad (1.2)$$

- Un proceso de derivación: El error previamente almacenado se resta al error de este instante generando la componente derivativa de la ecuación 1.3. 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.

$$D[n] = K_d \cdot (e[n] - e[n-1]) \quad (1.3)$$

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 1.11.

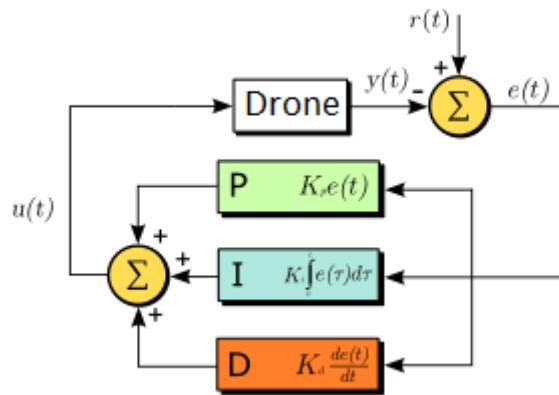


Figura 1.11: Esquema de un controlador PID

El resultado de la suma de la Ecuación 1.4 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.

$$u[n] = P[n] + I[n] + D[n] \quad (1.4)$$

## 1.4. FPGA

### 1.4.1. Concepto

Las siglas hacen referencia a un array de puertas lógicas reprogramables in situ. Son una evolución de una tecnología similar llamada CPLD que disponía de una densidad de puertas lógicas por centímetro cuadrado menor.

Una FPGA es un dispositivo capaz 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 mucho más complejas, a través de una adecuada programación de los elementos más básicos. Dicha programación se realiza en

un lenguaje de descripción de hardware. Para el caso de este trabajo se ha hecho uso de Verilog 2001. Un lenguaje similar a C en aspecto, pero con un concepto de fondo radicalmente distinto.

### 1.4.2. Características

Una FPGA dispone de ciertas características que le ofrecen una ventaja sobre opciones programables más tradicionales, como puede ser un procesador secuencial.

Una FPGA puede ser programada de tal manera que ejecute un conjunto de instrucciones concreto varias veces de manera simultánea sin verse disminuido su rendimiento global. En el caso de un procesador secuencial, si se desea realizar una tarea adicional, o sencillamente ejecutar dos veces la misma sección del código, solo suele haber dos opciones; que haya hardware dedicado por duplicado construido sobre el mismo silicio, cosa que no suele ocurrir para todas las necesidades de un sistema, sino solo en casos puntuales. O realizar una ejecución del código mientras la segunda ejecución espera a que la primera termine. 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, sin un mínimo detrimiento del rendimiento en ninguno de ellos. Esta ventaja es de especial importancia en procesos de control cuya tarea puede dividirse en muchas, como es el caso del procesamiento de imágenes por ejemplo. O en sistemas paralelos de procesado digital de señales. El segundo caso está muy relacionado con este TFG, debido a la ejecución de múltiples sistemas de comunicaciones simultáneamente a la ejecución de cuatro bucles de control paralelos.

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. Esto sobrepasa la idea de por ejemplo reconfigurar los coeficientes de un filtro de una parte concreta de una arquitectura de demodulación, por la capacidad de reconstruir toda la arquitectura. Convirtiendo el nuevo sistema en un demodulador completamente diferente, no solo a efectos de bandas, códigos y demás, sino de idea o concepto completamente nuevo, sobre el mismo hardware.

### **1.4.3. Aplicaciones**

Como se ha mencionado antes, 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 en procesos industriales y en seguridad. Son tareas que a menudo se han llevado a cabo por procesadores gráficos que en su esencia son una mezcla de muchos pequeños procesadores pensados para realizar cálculos geométricos simples, y que actualmente se están sustituyendo por FPGAs debido a la flexibilidad y potencia de las mismas.

También se comienzan a utilizar como aceleradores de redes neuronales convolucionales debido a la que los avances en unidades de procesamiento gráfico no alcanzan a cubrir las necesidades de proceso para dichas redes. Las GPUs de manera individual, tienen mayor capacidad de proceso que una FPGA de grado medio, pero estas últimas ofrecen un consumo mucho más bajo y mayor capacidad de clustering, por estos motivos, son la elección para sustituir a los aceleradores gráficos en este campo.

### **1.4.4. 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. Sobre todo aquellas versiones de los 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 para sus FPGAs que incluye una versión gratuita, muy limitada de prestaciones y con ciertas limitaciones, o Vivado de Xilinx. 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 1.12 y 1.13), orientado al

diseño de software para FPGAs de Lattice, en constante crecimiento. Una de las familias de FPGAs recientemente cubiertas por este software son las ICE40 UltraPlus, concretamente la ICE40UP5K utilizada para desarrollar el núcleo de proceso de la unidad de tierra en este TFG que se detallará más adelante en este informe.



Figura 1.12: Logo de IceStudio, IDE para desarrollo de software en FPGAs libres

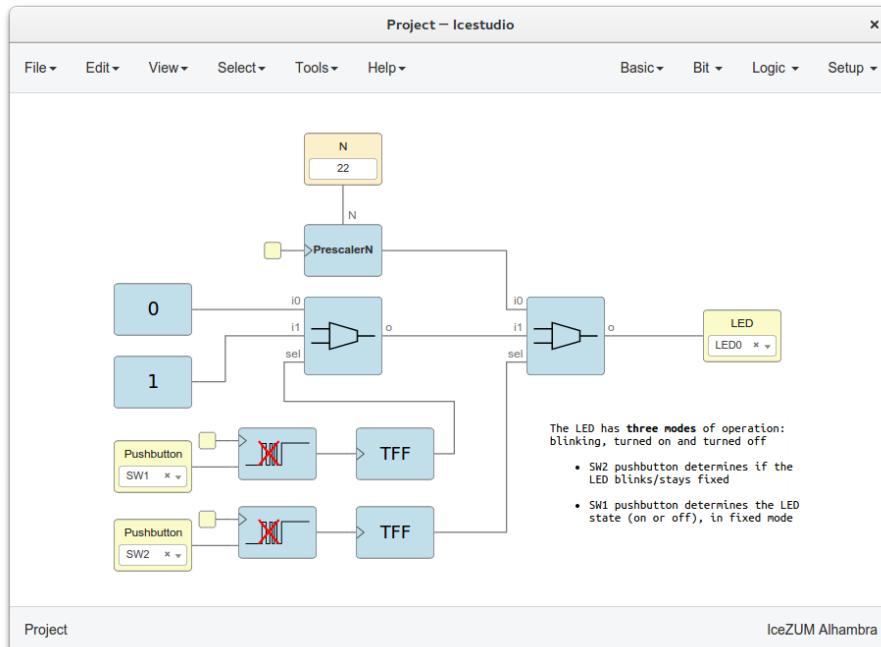


Figura 1.13: Captura del entorno IceStudio

## 1.5. 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.

### 1.5.1. UART

Se trata de un sistema de comunicaciones serie asíncronas (Universal Asynchronous Receiver Transmitter) capaz de realizar una comunicación full duplex 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 1.14 se muestra la transmisión de un byte que incluye bit de paridad. 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 1.14, 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”.

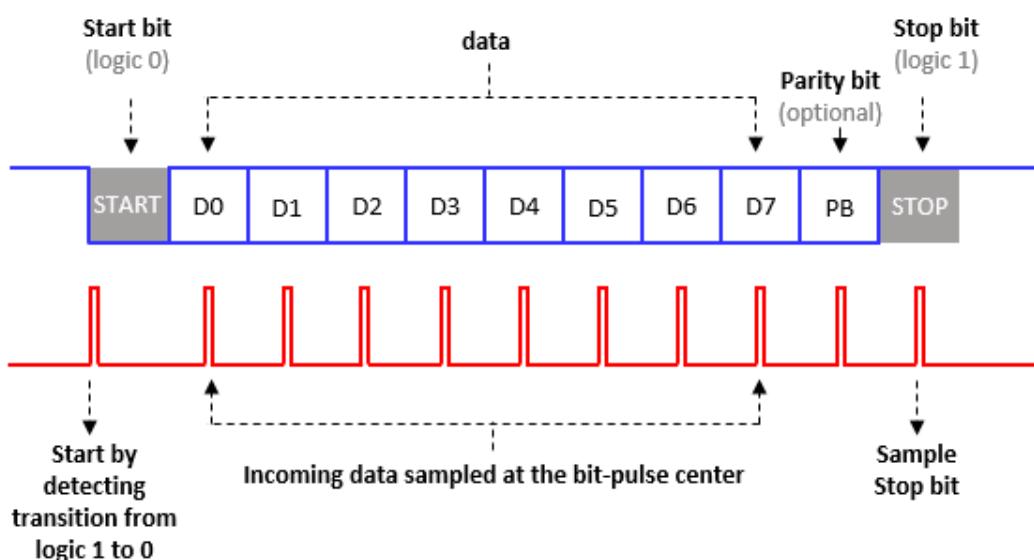


Figura 1.14: Formato de paquete UART

### 1.5.2. 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. En la Figura 1.15 se muestra un ejemplo de comunicación, y en la Figura 1.16 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).

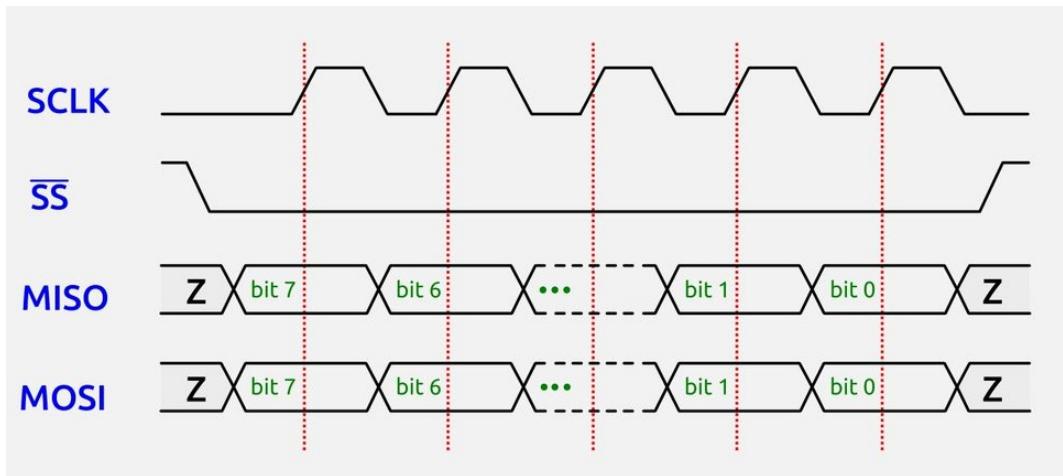


Figura 1.15: Funcionamiento de una comunicación SPI

- SS: En caso de tener conexión a múltiples dispositivos, el maestro puede escoger con cual establecer 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 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 1.16. 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.

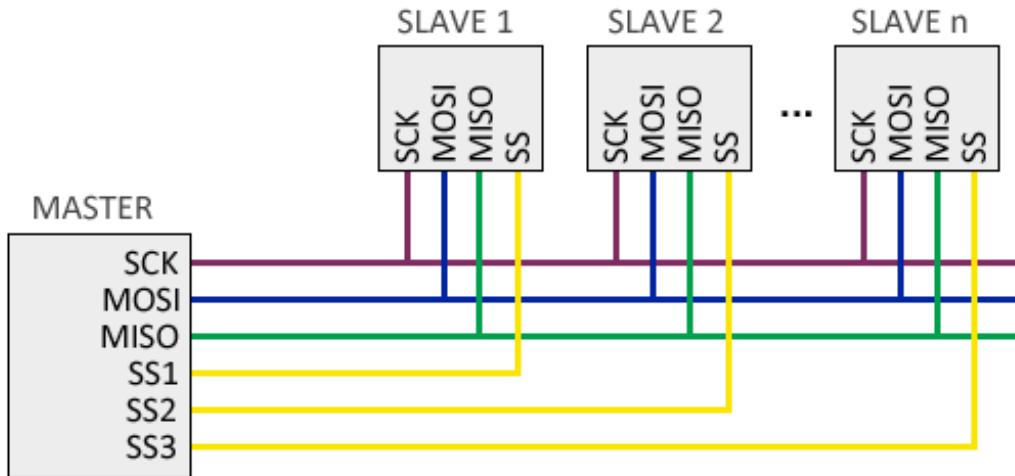


Figura 1.16: Esquema de conexión para un puerto SPI

### 1.5.3. 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) que se conectan a todos los dispositivos colgados del bus, como se muestra en la Figura 1.17.

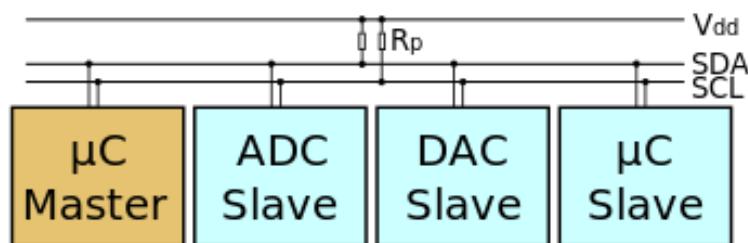


Figura 1.17: Arquitectura hardware de un bus I2C

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 1.18. 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.

Mensaje									
Start	7 o 10 Bits	Bit para Leer/ Escribir	Bit para reconocer ACK/ NACK	8 Bits	Bit para reconocer ACK/ NACK	8 Bits	Bit para reconocer ACK/ NACK	Stop	
Condición de inicio	Sección destinada para la dirección			Sección 1 para transportar información		Sección 2 para transportar información		Condición de paro	

Figura 1.18: Formato de trama I2C

#### 1.5.4. 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, 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 1.19. En ella se tiene seleccionado el canal 1 para mostrar su temporización.

Una trama completa PPM para el drone 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 1.19) y su valor máximo para 1.7ms.



Figura 1.19: Señal PPM hacia el módulo de transmisión de uplink

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.



# **Capítulo 2**

## **Objetivos**

### **2.1. Objetivo principal**

El objetivo de este trabajo es ser capaces de controlar un drone de bajo coste desde un ordenador. Este deberá ser capaz de auto-estabilizarse a través de un conjunto de sensores y procesamiento basado en una FPGA libre. Para ello se ha dividido el trabajo en sub-objetivos comentados a continuación, los cuales en su conjunto permitirán alcanzar el objetivo principal una vez cubiertos.

### **2.2. Sub-objetivos**

#### **2.2.1. Enlace con el drone**

El punto de partida del proyecto, pasa por ser capaces de realizar una comunicación radio entre un ordenador y el propio drone. Para esta tarea se hará uso del software libre provisto por goebish en su repositorio nrf24\_multipro. Habrá que conseguir enlazar el software ejecutado en un procesador de Atmel, con la electrónica propia del drone.

#### **2.2.2. Comunicación de órdenes fijas al drone**

Una vez establecida la comunicación es el momento de generar el software necesario tanto para el procesador de Atmel como para la FPGA. Con esto se podrán probar instrucciones fijas hacia el drone, del tipo, enciende y apaga motores.

### **2.2.3. Diseño de librerías de control para PC**

El objetivo de las librerías, es facilitar el uso del drone y la repetitividad de los ensayos a través de abstraerse de la capa más baja del control del drone. Esto permite pasar de instrucciones de encendido y apagado independientes, a instrucciones de despegue, avance, giro y demás, con el objetivo de facilitar siguientes tareas.

### **2.2.4. Comando del drone en bucle abierto**

En este punto ya se dispone del software para realizar ensayos de vuelo con el drone en bucle abierto. Se encadenarán varias instrucciones de vuelo construidas en la librería previa, con el objetivo de probar el drone en trazados más complejos. Como despegue, avance, retorno y aterrizaje, o estabilidad en vuelo estático.

### **2.2.5. Diseño de la estación de tierra**

Para proceder a realizar ensayos de bucle cerrado, será necesario tener disponible una electrónica de tierra capaz de recibir medidas de los sensores de a bordo, procesar y responder en consecuencia. La respuesta dependerá tanto de las medidas de los sensores, como del comando que se reciba por USB. Estos se enviarán desde un ordenador que secuenciará las instrucciones a ejecutar.

### **2.2.6. Diseño de electrónica de a bordo**

Para poder corregir el comportamiento en vuelo del drone, será necesario que este disponga de información relativa a su posición local. Para cubrir esta tarea, se instalará en el drone electrónica capaz de medir los parámetros que se consideren necesarios y transmitirlos a la estación de tierra para ser procesados.

### **2.2.7. Cierre de bucles para cada eje**

Se deberá diseñar un bucle cerrado de control para cada eje, independiente del resto. Esto permitirá el manejo completo del drone, pudiendo actuar de manera también independiente para cada eje.

## 2.2.8. Experimentación y parches

Este punto cubre una necesidad presente en los 7 sub-objetivos anteriores. Tras aplicar cambios software y/o hardware se testeará el trabajo. Según el resultado, se avanzará a la siguiente fase del desarrollo, o se iterará entre la fase de origen y esta, hasta corregir adecuadamente los posibles errores.

## 2.3. Requisitos

Los objetivos del trabajo deberán ser capaces de cubrir unos requisitos mínimos que permitan discernir cuando una tarea se ha cumplido adecuadamente para avanzar a la siguiente. Para ello se listan a continuación unos requisitos mínimos del sistema.

- El sistema deberá ser capaz de estabilizar el drone con al menos tres grados de libertad; cabeceo, alabeo y altitud.
- Cada grado de libertad controlado deberá implementar un bucle cerrado, retroalimentado, con controlador PID.
- Dichos controladores deberán obedecer a consignas radio que permitan el control en tiempo real desde la plataforma de tierra.
- Para facilitar las tareas de estabilización y otorgar flexibilidad al sistema, los controladores PID deberán disponer de parámetros re-configurables también desde la estación de tierra.

## 2.4. Metodología

Para conseguir alcanzar el objetivo final, cubriendo los requisitos mínimos, se procederá a abordar los sub-objetivos de manera secuencial. Se intentarán tener reuniones de seguimiento semanales por conferencia, para informar de los avances y de posibles problemas. También se informará por correo u otros medios en caso de resultar útil. En estas reuniones se comentará dicho avance y problemas, y se propondrán 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. Segundo 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.

## **2.5. Plan de trabajo**

En base a las herramientas de desarrollo disponibles, mencionadas más adelante en el Capítulo 3, se planteará una arquitectura de sistema, definida en el Capítulo 4. Dicha arquitectura deberá ser capaz de alcanzar el objetivo principal, cubriendo los requisitos mínimos, dentro de las limitaciones impuestas por la propia arquitectura, a su vez limitada por factores como las herramientas disponibles, tiempo de desarrollo y dificultad. El desarrollo se hará en gran medida los fines de semana, añadiendo horas disponibles por las tardes, tras el horario laboral, dependiendo de las exigencias de este. Se cubrirán los sub-objetivos en orden, hasta alcanzar el objetivo principal.

# **Capítulo 3**

## **Infraestructura utilizada**

### **3.1. IceCube2**

En su versión 2017.08.27940, es el entorno de desarrollo suministrado por Lattice para la programación de sus familias pequeñas de dispositivos FPGA. 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 concreto de la FPGA escogida y generación del fichero de programación “\*.bin”. Este software integra la herramienta de programación Diamond Programmer versión 3.10.0.111.2 también distribuida por Lattice para realizar la programación de sus dispositivos con el fichero “.bin” generado tras el trazado de la FPGA.

### **3.2. Quartus Prime**

Versión 15.1.0 Build 185. Herramienta de diseño, test y programación de software para FPGAs de Altera (Intel). Utilizado en este trabajo exclusivamente por su editor de texto con reconocimiento del lenguaje Verilog 2001 e identificación de estructuras con color.

### **3.3. Arduino IDE**

Versión 1.8.8. Entorno de desarrollo integrado de Arduino para procesadores Atmel compatibles. Adicionalmente al editor de texto, compilador, enlazador y programador, todo en uno,

Suministra algunas herramientas adicionales de utilidad: Instalador de librerías integrado. Utilizado para facilitar las tareas de instalación de paquetes para comunicación por puerto serie. Entre los incluidos, se hace uso en este trabajo de las librerías para puertos serie SPI, I2C y UART. 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 de a bordo para realizar tareas de diagnóstico.

### **3.4. ModelSim**

Versión 10.4b. Herramienta para la simulación de lenguajes de descripción de hardware. Compatible con Verilog 2001, permite la comprobación de sintaxis del lenguaje, compilación e instanciación de múltiples módulos. Ofrece también completa configuración de resoluciones y tiempos de simulación, junto con la posibilidad de realizar análisis dinámicos del hardware definido. Permite ahorrar 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 error de control en vuelo.

### **3.5. FT\_Prog**

Versión 3.6.88.402. Herramienta distribuida por FTDI Chip para reconfigurar la memoria flash de los dispositivos de comunicación USB, que permite modificar el comportamiento de cada canal de sus dispositivos, activando, desactivando o modificando su comportamiento para hacerlo acorde a distintos estándares.

### **3.6. Logic**

Versión 1.2.10. Esta utilidad de Saleae es su interfaz para almacenamiento y visualización de señales medidas con sus herramientas de análisis digital. Para el caso de un puerto serie de comunicaciones, permite el análisis de la información para un protocolo concreto, interpretando la información en cada canal y mostrándola en relación al momento de lectura. Se pueden

nombrar y configurar distintos protocolos para cada canal, de esta manera se puede utilizar para comprobar el comportamiento de señales concretas del sistema, o analizar los datos transferidos entre dispositivos. Esta herramienta facilita el debug del sistema a través de tareas de detección y corrección de errores.

### 3.7. Logic Analyzer

Tarjeta de adquisición de Saleae. Trabaja como analizador lógico de ocho canales, con muestreo hasta 24MSPS, permitiendo adquisición y análisis de señales de hasta 12MHz. Canales reconfigurables como señal o disparo para sincronía con tramas y eventos. Esta herramienta trabaja en conjunto con la herramienta para PC ?Logic? para su configuración e interfaz de usuario.

### 3.8. Arduino Uno y Nano

Tarjetas de desarrollo que integran un procesador de Atmel de 8 bits modelos ATMEGA8U2-MU y ATMEGA328P-AU, un interfaz de UART a USB para comunicaciones y alimentación, conectores hacia los GPIOs de los procesadores y un regulador de 5V a 3V3. Se pueden alimentar de manera externa sobre la entrada del regulador, o directamente mediante el USB. Se dispone de acceso para programación y depuración a través del mismo USB.

### 3.9. ICE40 UltraPlus Breakout Board

Tarjeta de desarrollo para la familia UltraPlus de ICE40. Incluye elementos varios de interfaz humana, como Leds, interruptores, jumpers e interfaz hacia los pines de propósito general, GPIOs. También incluye un interfaz USB reprogramable de FTDI Chip y doble canal, FT232HL. Como procesador hace uso de una FPGA ICE40UP5K-SG48, con 5280 celdas lógicas, 120Kbits de RAM embebida y 1024kbits de RAM estática, una PLL y 8 bloques DSP (multiplicadores de 16 bits).

### **3.10. EACHINE E010**

Mini-drone cuadricóptero teledirigido fabricado por Eachine. Cuenta con 8cm de ancho y 8 cm de largo, conductores en las aspas para evitar vórtices en los extremos de las mismas, aumentando así la eficiencia en vuelo, batería de litio y sistema teledirigido de cuatro canales en la banda de 2.4GHz. Dispone de 30 metros de alcance radio, sistema de retorno al punto de partida, y una estructura ligera.

### **3.11. SYMA X5C**

Drone cuadricóptero teledirigido fabricado por Syma. Un drone ligero para su tamaño, con una estructura robusta, perfecto para aprendizaje de principiantes y experimentación gracias a su importante resistencia mecánica. Electrónica de recepción radio con 50 metros de alcance en la banda de 2.4GHz, cuatro canales de control, batería de litio, 31cm de ancho y largo, y 100gr de peso. Incluye una cámara con almacenamiento en SD-Card.

### **3.12. NRF24L01**

Módulo que integra el chip de mismo código, con condensadores de desacoplo necesarios para su utilización, una antena integrada en la propia PCB y pines para acceso a su interfaz SPI. Ocupa la banda de 2.4GHz, con 125 posibles canales de 1 MHz de ancho de banda cada uno. Al trabajar entre 1.9V y 3.6, es perfecto para su utilización junto a tarjetas Arduino Uno y Nano, que integran reguladores con salida a 3.6V. Dispone de tasas de transferencia de hasta 2Mbps y alcances de hasta 30 metros.

### **3.13. Flow breakout board**

Se trata de un módulo de BitCraze 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. Incluye electrónica periférica para hacerlos funcionar y un cabezal para las conexiones eléctricas. Al alimentarse a 3.6V puede ser conectado a la salida de los reguladores de las tarjetas de desarrollo de Arduino.

# Capítulo 4

## Arquitectura del sistema

La arquitectura del sistema se ha planteado con el propósito de cubrir los objetivos propuestos en el apartado Capítulo 2, haciendo uso de la infraestructura disponible, mencionada en el Capítulo 3. El sistema se compone de cuatro módulos principales y dos interfaces de alto nivel. Los cuatro módulos principales que lo componen son:

- Un ordenador para enviar las órdenes de mando.
- Un drone como estructura base y sistema de vuelo con su propia electrónica para la recepción de órdenes de movimiento. La llamaremos electrónica del drone de aquí en adelante, para diferenciarla del siguiente conjunto de electrónica.
- Un conjunto de sensores para realizar medidas de posición y ubicar al drone. De ahora en adelante, sensores de a bordo o embarcados.
- Una estación de tierra como centro de control y verdadero centro neurálgico del sistema.

A nivel de sistema, los cuatro elementos principales se comunican entre ellos gracias a dos grupos de interfaces de sistema:

- Interfaz de mando.
- Interfaz radio.

En la Figura 4.1 se muestra la estructura de más alto nivel de la arquitectura planteada, junto con la relación entre los módulos y los grupos de interfaces. Estos se explicarán en los apartados siguientes.

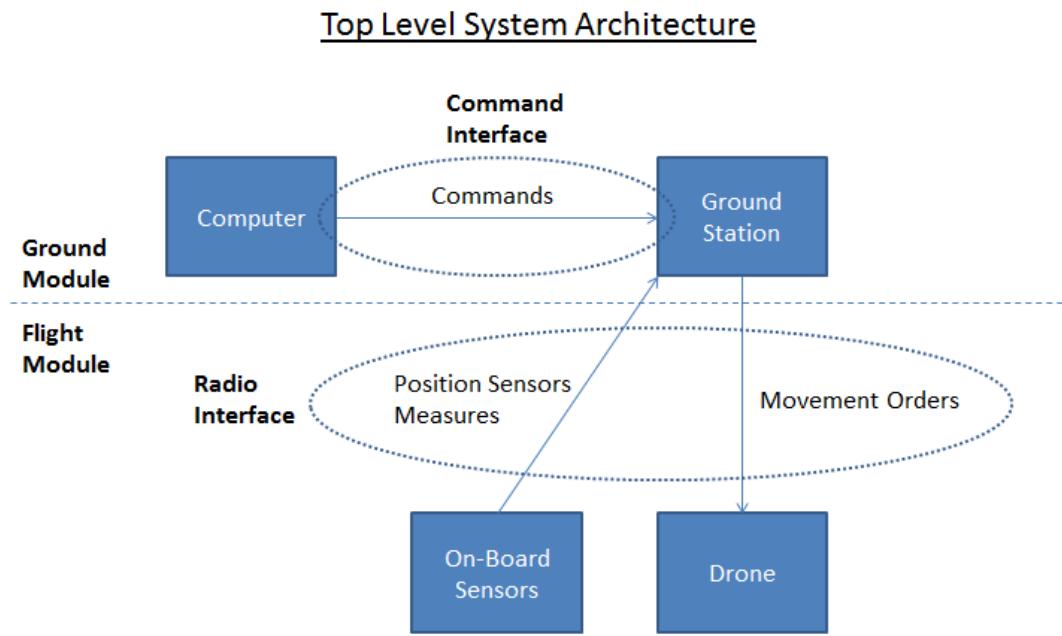


Figura 4.1: Arquitectura de sistema y su bucle de control

La arquitectura del sistema propuesto, se basa en que la estación de tierra atiende los comandos que puedan llegar desde el ordenador de mando, de manera simultánea a las tramas que provengan de los sensores a bordo del drone. El drone, capaz de ubicar de manera local su posición relativa en base a las medidas entregadas por sus sensores, transmite eventualmente por radio su posición hacia la estación de tierra. Ésta calcula el vector de error de posición como la diferencia entre las medidas de posición indicadas por los sensores de posición, y las órdenes recibidas desde el ordenador de mando. En base al vector de error, la estación de tierra computa las correcciones y se las envía por radio al drone en forma de órdenes de movimiento. El drone entonces, obedece dichas órdenes, a continuación vuelve a medir su posición y envía la información de nuevo a la estación de tierra, cerrando así el bucle de control principal del sistema.

A continuación se detallan los interfaces que conectan los cuatro módulos de nivel superior que toman partido en la arquitectura del sistema de la Figura 4.1; estación de tierra, drone, sensores de a bordo y ordenador de mando. Más adelante se detallan también su construcción, funciones y sub-sistemas.

# **Capítulo 5**

## **Interfaces del sistema**

### **5.1. Interfaz de mando**

Se encarga de comunicar únicamente el ordenador de mando con la estación de tierra. Porta información de las instrucciones de posición, que se espera el drone obedezca, y parámetros para los bucles de control. Este interfaz tiene dos objetivos principales: Facilitar la comunicación con la estación de tierra, al mismo tiempo que proveer al sistema de una alta capacidad de reconfiguración para disminuir el tiempo entre pruebas. Por este motivo se incluyen los parámetros de los bucles de control como parte de los datos. Esto facilita realizar múltiples ensayos de vuelo consecutivos, con mucha diferencia en el comportamiento de cada uno de ellos.

#### **5.1.1. 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 y UART y RS232. Para la conversión del estándar eléctrico desde USB a RS232, se hace uso de un driver de USB de FTDI Chip. El cual también convierte del protocolo de comunicaciones USB a UART. Concretamente el canal B del integrado FT232HL. Dicho driver se encuentra integrado en la placa de desarrollo de la FPGA de Lattice. El canal A se reserva para otras tareas comentadas más adelante en la descripción de los interfaces de la propia estación de tierra. El puerto de comunicaciones 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, la cual se fija en 500Kbps. Esta velocidad irá acorde con la tasa de transferencia del mismo interfaz, del lado de la FPGA, explicado en más adelante. Las tramas se envían desde el ordenador de mando a través del USB con la información adecuada para encapsular cada byte en el formato de comunicaciones propio de una UART. De esta conversión también se encarga el módulo FT232HL a través de la información provista en tiempo de ejecución. Una vez abierto el puerto a la velocidad indicada anteriormente, se envía cada 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.

### 5.1.2. Formato de tramas

Se hace uso de un formato de tramas fijas. La longitud es conocida y siempre la misma, 16 bytes. Y 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 pruebas o mejoras a lo largo de la evolución por los distintos objetivos secundarios, hasta alcanzar el objetivo principal. Cada trama es enviada en el instante de ejecución de cada instrucción en el ordenador de mando. Cada instrucción dispone también de un lapso de tiempo indicado al ejecutarla. De esta manera cada instrucción está activa el tiempo indicado, hasta que se produzca la ejecución de la siguiente instrucción, o se termine el programa. En su versión final, las tramas componen sus 16 bytes con la construcción fija mostrada en las Figuras 5.1 y 5.2:

Frame:									
Field Name	STx1	STx2	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	1
Byte Position	1	2	3	4	5	6	7	8	

Figura 5.1: Formato de trama de mando, primera mitad 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, la etapa receptora espera por una secuencia de dos bytes consecutivos con valores 255 y 90 para

Frame:									
Field Name	RSV	T_CW	PID_Alt_K	PID_Alt_K	PID_XY_K	PID_XY_Kd	RSV	RSV	
Value	0	Varia	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	

Figura 5.2: Formato de trama de mando, segunda mitad de la trama

sincronizar con la trama entrante. Tras esto, el receptor comenzará a decodificar los valores de los siguientes 14 bytes con el orden indicado en la trama. Asumiendo que no se produzca un error de comunicaciones, el equipo receptor asociará cada valor a su campo adecuadamente.

- Cmd\_ALT: Comando de vertical, o altura. Representa el valor de la consigna de posición de altura deseada para el drone. Este valor será decodificado por la estación de tierra y enviado al bucle de control de altura como su consigna.
- Cmd\_L/R: Comando de desplazamiento horizontal lateral. Representa el valor de la consigna de posición lateral deseada para el drone. 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 desplazamiento horizontal frontal. Representa el valor de la consigna de posición frontal deseada para el drone. Este valor será decodificado por la estación de tierra y enviado al bucle de control de desplazamiento frontal, 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 drone, sobre sí mismo. Este valor será decodificado por la estación de tierra y enviado directamente al constructor de tramas, ya que este grado de libertad no se controla con un bucle cerrado de alto nivel. Se le deja su control exclusivamente a la electrónica propia del drone 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. Representa el valor deseado para las constantes proporcionales de ambos bucles de control PID de posición horizontal. El control de posición frontal y el control de posición lateral. Ya que el drone es simétrico respecto de sus ejes de desplazamiento horizontal, se aplica la misma tensión a ambos bucles de control. La diferencia entre los valores de las consignas Cmd\_L/R y Cmd\_F/B y sus medidas por los sensores de posición de a bordo, generan los errores de posición horizontales, componentes del vector de error de posición. 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: Frontal y latera.
- 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 drone sobre sí mismo. Representa el valor del offset a añadir al valor del comando “Cmd\_T\_CW” a enviar hacia el drone. El objetivo de este parámetro es corregir de manera constante el valor de la velocidad de giro sobre sí mismo enviado hacia el drone. Ya que este valor se encuentra controlado por un bucle abierto en la electrónica de la estación de tierra, si la electrónica propia del drone, tiene un error constante de giro que no pueda corregir, 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 integrará y multiplicará 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 valor del 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, completan los valores de los parámetros para el bucle de control cerrado PID de la altura.

- **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. El control de posición frontal y el control de posición lateral. Ya que el drone 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 integrarán y multiplicarán por el valor recibido en este campo, para conseguir así las componentes “I” de los bucles de control de los ejes horizontales: Frontal y latera.
- **PID\_XY\_Kd:** Parámetro de la constante derivativo de los ejes horizontales. Representa el valor deseado para las constantes derivativas de ambos bucles de control PID de posición horizontal. El control de posición frontal y el control de posición lateral. Ya que el drone 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 latera. Este campo junto con los valores de los campos de las posiciones 8 y 13, completan los valores de los parámetros para los bucles de control cerrado PID del plano horizontal, Controlador de posición Frontal y Controlador de posición lateral.

## 5.2. Interfaz radio

Se encarga de comunicar la estación de tierra con la electrónica del drone, y los sensores de a bordo, tal como se muestra en la figura 20. Este interfaz porta información fundamental para el desempeño de la tarea de control de posición del drone. Se transmite información tanto de la posición del drone, como de las órdenes de velocidad que este deberá ejecutar. El objetivo de este interfaz es que la estación de tierra conozca la posición relativa del drone, y en base a

esto, y a las instrucciones de mando que reciba, trate de dirigir y corregir en caso de error, dicha posición.

Se trata de un enlace radio bidireccional entre la estación de tierra y el drone. Dividiremos este enlace en dos partes; la bajada, o downlink, que transmite información desde el drone hacia la estación de tierra. Y la subida, o uplink, que transmite información desde la estación de tierra hacia el drone.

Ambos trabajan en la banda libre de 2.4GHz, pero en distintas frecuencias (canales). Así se consigue un enlace bidireccional full-duplex, en base a separación en frecuencia, o FDM. De esta manera ambos pueden estar transmitiendo información simultáneamente, sin colisionar, ni preocuparse de la complejidad y la bajada en la tasa de transferencia propia de un sistema de multiplexación por división en código, CDM.

### 5.2.1. Downlink

Se encarga de enlazar por radio los sensores de posición a bordo del drone con la estación de tierra. Porta la información medida por dichos sensores. Esta información se compone de una medida de distancia al suelo, o altura, tomada desde la parte media del drone. Una medida del desplazamiento frontal relativa al punto de despegue. Y una medida del desplazamiento lateral relativa también al punto de despegue.

Las tres medidas se miden y transmiten por el interfaz radio en milímetros, considerando como origen de coordenadas, tanto para la altura como para los desplazamientos horizontales, la posición de despegue inicial.

#### Características

Ocupa el canal 0x66 dentro de la banda libre de 2.4GHz utilizada 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, como para el caso de las tramas de mando. En este caso la longitud de la trama es de seis bytes. Y la posición de cada valor tampoco varía

durante toda la ejecución. Cada trama es enviada en el instante inmediatamente posterior a la ejecución de las tres medidas de posición; altura, desplazamiento frontal y desplazamiento lateral. El formato de las tramas se muestra en la Figura 5.3.

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

Figura 5.3: Formato de trama de downlink

En estas tramas la sincronía está garantizada como parte del diseño de la transmisión realizada por la radio NRF24L01 y la forma en que esta se comunica con la electrónica de la estación de tierra, que se detallará más adelante. Por tanto los seis bytes de la trama son tres campos útiles de dos bytes cada uno.

- H\_Pos\_F/B: Valor de la medida del desplazamiento frontal. 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” con la salvedad de que este representa los milímetros de desplazamiento lateral desde el origen de coordenadas. Codificado también de la misma manera.
- Alt: Valor de la medida de altura. Representa los milímetros de desplazamiento vertical desde el origen de coordenadas. 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 drone, generando así el vector de posición necesario para realizar posteriormente las correcciones de posición respecto de las órdenes comandadas.

### 5.2.2. Uplink

Se encarga de enlazar por radio la estación de tierra con la electrónica del drone. Comunica las indicaciones de velocidad para cada eje, junto con canales auxiliares cuya función depende de cada drone. En este TFG no se hará uso de dichos canales auxiliares.

#### Características

De igual manera que el interfaz de bajada, hace uso de una radio NRF24L01 en la banda de 2.4GHz. El módulo que gestiona este interfaz, controla la radio NRF24L01. Este módulo está diseñado para recibir los canales a codificar en la trama radio, desde una señal modulada por posición de pulso, PPM. Dicha modulación y sus características se describen más adelante en el apartado de interfaces de la estación de tierra, junto con el módulo que gestiona la radio del uplink.

#### Formato de trama

Este interfaz se compone de una transmisión de doce canales, de los cuales se hará uso exclusivamente de los cuatro primeros. Los ocho canales restantes portan comandos auxiliares, como la velocidad de reacción deseada para el drone, la cual se mantiene en su valor por defecto, y dependiendo del modelo exacto del drone utilizado, a veces se utilizan para disparar fotografías o video de la cámara instalada. El subsistema radio se encarga de transmitir la información contenida en la trama de la Figura 5.4 por radio hacia la electrónica del drone.

Frame:					
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 in Frame	1	2	3	4	5 to 12

Figura 5.4: Formato de trama de uplink

- Cmd\_THR: Valor de velocidad de altura. Ocupa la primera posición en la trama. Representa el impulso deseado para el eje vertical del drone. 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 lateral deseada para el drone. Está asociado al grado de libertad del alabeo.
- Cmd\_F/B: Valor de velocidad de desplazamiento frontal. Representa la velocidad de desplazamiento frontal deseada para el drone. Está asociado al grado de libertad del cabeceo.
- Cmd\_T\_CW: Valor de velocidad de giro. Representa la velocidad de giro sobre sí mismo deseada para el drone. Está asociado al grado de libertad de la guiñada.
- Aux: Algunos de los ocho canales adicionales tienen valores fijos y otros variables, dependiendo de su funcionalidad y del drone utilizado. Para este trabajo, estos canales se mantendrán con su valor por defecto.



# **Capítulo 6**

## **Módulos del sistema**

A continuación se describen los dos sistemas diseñados que junto con el ordenador de mando, componen el sistema completo de este trabajo.

### **6.1. Estación de tierra**

Es el módulo encargado de ejecutar los algoritmos de control y producir las respuestas necesarias para estabilizar y controlar el drone. Para ello procesa los comandos enviados desde el ordenador de mando y las señales medidas por los sensores a bordo. Dicha información se traslada entre sus módulos a través de los interfaces mostrados en la Figura 6.1. El resultado de su proceso es enviado de vuelta hacia el drone a través las órdenes de movimiento del interfaz radio.

Cada módulo realiza su tarea en base a la información previa que pueda contener, junto con la que se transfiere entre él y los módulos con los que tenga comunicación, dentro de la Figura 6.1.

#### **6.1.1. Descripción**

El sistema propuesto se compone de varias tarjetas mostradas en la Figura 6.2. Cada tarjeta o tarjetas, forman un módulo de la arquitectura de la estación de tierra. Cada uno realiza una función concreta y necesaria para cubrir algún sub-objetivo, permitir a las demás cumplir con el suyo, o facilitar las tareas de depuración.

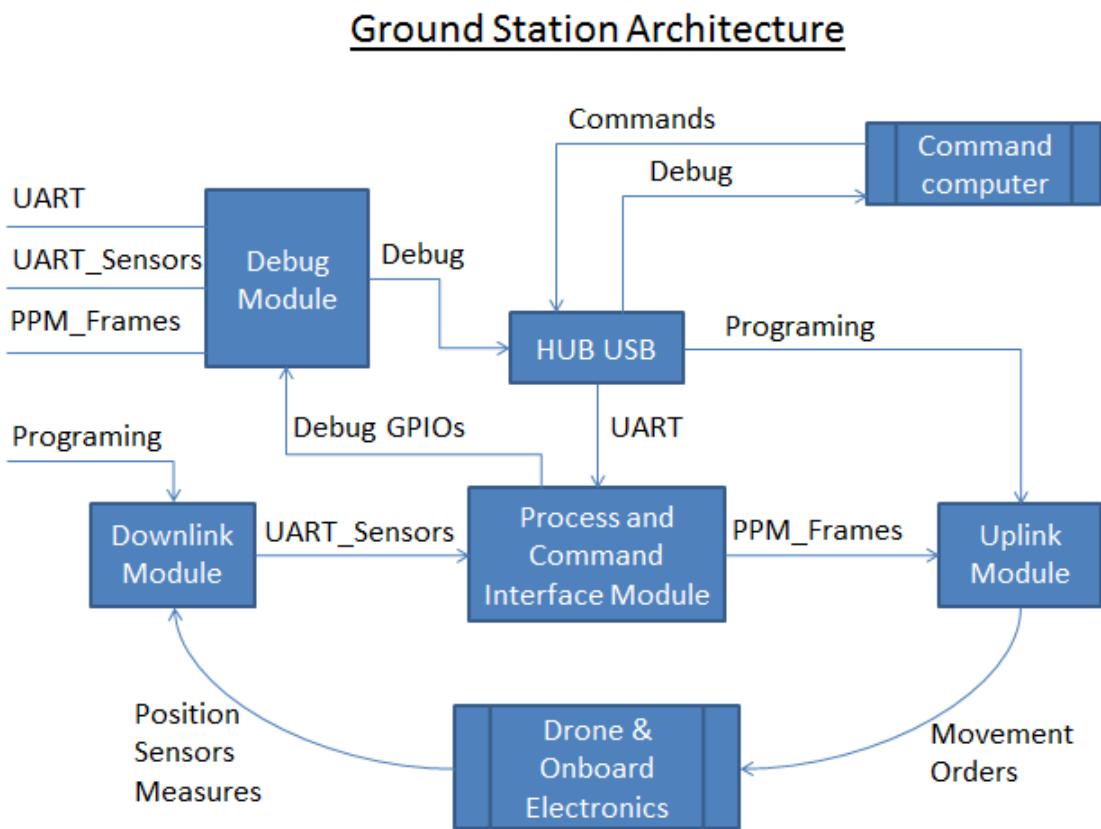


Figura 6.1: Arquitectura e interfaces de la estación de tierra

A continuación se nombran los distintos módulos que componen la estación de tierra, enumerados en la Figura 6.2:

- Módulo 1: Hub USB.
- Módulo 2: Módulo de procesado y comunicaciones con ordenador de mando.
- Módulo 3: Módulo de downlink.
- Módulo 4: Módulo de uplink.
- Módulo 5: Módulo de depuración.

### 6.1.2. Objetivo

La estación de tierra tiene por objetivo conseguir que el drone obedezca las instrucciones de posición que se envían desde el ordenador de mando. Para ello se diseñarán una serie de

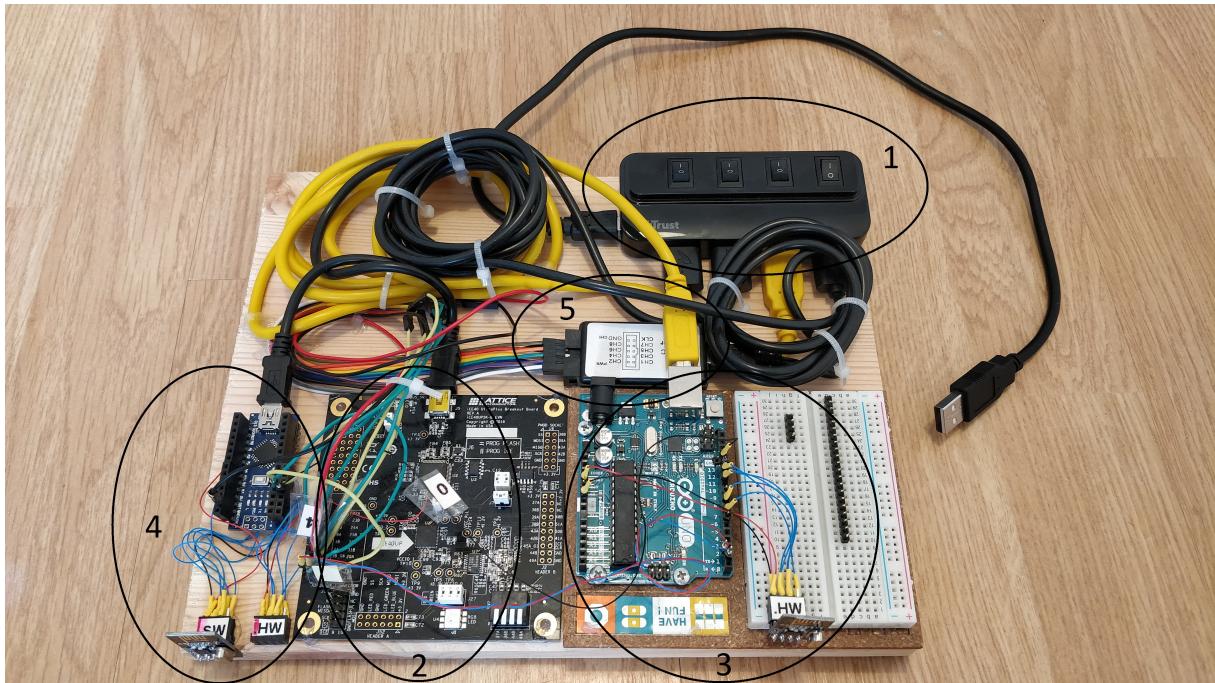


Figura 6.2: Montaje de la estación de tierra

funciones, descritas a continuación, que la estación de tierra ejecutará a través de los módulos disponibles. Toda información que se mueva dentro de la estación de tierra, quedará descrita más adelante en los interfaces. Mientras que la información transmitida entre los módulos de la estación de tierra y el exterior (drone, sensores a bordo y ordenador de mando), queda definida en los interfaces del sistema.

### 6.1.3. Funciones

- Recepción de tramas enviadas desde el drone por el interfaz de downlink. Función ejecutada por el módulo 3. Deberá demodular y decodificar la información de los sensores de a bordo del drone, transmitida por radio. Tras esto generará un paquete con la información recibida hacia el procesador principal, a través del interfaz UART\_Sensors.
- Recepción de comandos desde el ordenador de mando. Función ejecutada por el módulo 2. El módulo 2 incluye el driver de USB, FT232HL, el cual deberá recibir los comandos de posición por USB y realizar la conversión a protocolo UART para hacérselos llegar a la FPGA del módulo 2.
- Procesado de control y dirección del drone. Función ejecutada por el módulo 2. Para la

correcta estabilización y control de la posición del drone, el módulo 2 dispone de una FPGA ICE40UP5K. Esta ejecutará el firmware configurado en su lógica interna para cubrir la función de control y estabilización. Dicho firmware ejecutará, entre otras cosas, los bucles de control definidos en el siguiente capítulo, algoritmos de control.

- Enlace radio con el drone. Como inicio de las comunicaciones entre la estación de tierra y el drone, el módulo 4 se encarga, tras el arranque, de enlazarse con la electrónica del drone.
- Transmisión de comandos de movimiento de vuelta al drone. Tras realizar el enlace, el módulo 4 se encarga de transmitir periódicamente las órdenes de movimiento que haya recibido por el interfaz PPM\_Frames.

#### 6.1.4. Interfaces

##### USB

Se compone de tres flujos de información distintos, programación, depuración y comandos. A través de este interfaz se transmiten los comandos entre el ordenador de mando y la estación de tierra, interfaz definido en la sección Interfaz de mando. También abarca la programación de los 3 dispositivos reprogramables del sistema, módulos 2, 3 y 4 y las medidas tomadas por el módulo de depuración.

##### SPI

La estación de tierra hace uso de dos puertos SPI para leer cada una de las dos radios disponibles. Se trata de las radios de los módulos 3 y 4, downlink y uplink respectivamente. Ambos hacen uso de una radio modelo NRF24L01, cuyo interfaz de escritura y lectura de registros es un puerto serie SPI. Para esta tarea se hace uso de la librería NRF24L01, distribuida de manera abierta en Github y en la herramienta de instalación de librerías del IDE de Arduino. Por el puerto SPI del downlink, viajan los datos de las medidas de los sensores, leídas a bordo del drone. Por el puerto SPI del uplink, viajan las órdenes de movimiento generadas en el módulo 2, que deberán alcanzar al drone.

### UART & UART\_Sensors

En el sistema se utilizan comunicaciones asíncronas en dos ubicaciones distintas, y para portar distinta información. En primer lugar, se utiliza para recibir los comandos desde el ordenador de mando a través del interfaz UART de la figura 25. El protocolo USB 2.0 es convertido a UART por el driver FT232HL del módulo 2. Una vez convertido, es recibido por la FPGA del mismo módulo, la cual se sincroniza con las tramas recibidas con las características y formato de tramas comentados en el apartado Interfaz de mando.

La segunda ubicación donde se hace uso de este protocolo es en la recepción de las medidas de los sensores de a bordo, a través del interfaz UART\_Sensors. Esta interfaz llega desde el módulo de downlink, el cual se encarga de comunicar las medidas de posición a la FPGA del módulo 2, directamente a través de una UART sin paridad ni control de flujo, con 10 bits por byte y una tasa de transferencia de 500Kbps.

### PPM\_Frames

Esta interfaz porta la información de las órdenes de movimiento que deben llegar al drone a través de una comunicación con un solo hilo, desde el módulo 2, hacia el módulo 4. Para ello hace uso de una modulación PPM como la comentada en la introducción. Las órdenes de movimiento han sido calculadas por la FPGA del módulo 2, y se entraman según se especifica en los formatos de trama de la sección del uplink, dentro del interfaz radio de los interfaces del sistema.

#### 6.1.5. Módulos

A continuación se describen los módulos que componen la estación de tierra, mencionados anteriormente y mostrados en la Figura 6.2.

##### Módulo 1: Hub USB

Centraliza las comunicaciones USB, permitiendo tener un único puerto ocupado en ordenador de mando, y poder acceder desde él a la programación individual de los 3 módulos reprogramables, junto a las sondas de depuración, y alimentar simultáneamente todos los sistemas.

También permite el apagado individual, muy útil para restablecer la comunicación con el drone reiniciando el módulo de uplink, sin eliminar la configuración en RAM de los módulos 2 y 3.

### Módulo 2: Procesado y comunicaciones con ordenador

Se trata del procesador principal del sistema, junto a un regulador de tensión, accesos a varios pines de propósito general y un driver de USB. Como procesador principal se hace uso de una FPGA ICE40UP5K de Lattice en su encapsulado QFN de 48 pines conectada a través de una UART al driver de USB, para el cual se hace uso de un FT232HL, capaz de convertir el protocolo USB a distintos estándares de comunicación. Concretamente el canal A convierte de USB a SPI, para la reprogramación de las memorias RAM interna de la FPGA y la memoria FLASH externa de la tarjeta de desarrollo. Y el canal B convierte de USB a UART, para comunicar las instrucciones de mando desde el ordenador hasta la FPGA.

La FPGA de este módulo es la encargada de recibir los comandos desde el ordenador de mando a través de interfaz UART, las medidas de los sensores de a bordo del drone a través del primer interfaz SPI y entrega las órdenes de movimiento por el segundo. Para ello, ejecuta los bucles de control descritos más adelante en el apartado sobre los algoritmos de control de la FPGA. Dichos bucles se nutren de la información recibida por UART originaria del módulo 1 y SPI desde el módulo 3. El resultado de su ejecución es enviado por SPI hacia el módulo 4.

### Módulo 3: Downlink

Se compone de una radio NRF24L01 conectada por puerto SPI a un procesador de Atmel integrado en una 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 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 drone. Configura la radio como receptora y se mantiene atento por pollin, a la espera de recibir nuevas medidas desde el drone. Cuando las recibe, las entrama con un formato de trama idéntico al descrito en el apartado del downlink, con el añadido al principio de dos bytes de valor 255 y 90, como bytes de sincronía. Técnica de sincronía idéntica a la descrita en el interfaz de mando. De esta manera, aunque se falle la recepción de un byte, el receptor tiene un método para volver a sincronizarse con las siguientes tramas. Además el módulo de downlink incluye un CRC de 8

bits en su mecanismo de comunicación radio y sistema de retransmisiones, lo que dificulta la llegada de bytes erróneos a este nivel (Si puede ocurrir que ante la llegada de múltiples bytes erróneos a la radio, uno o varios queden sin entregar al procesador, tras el vencimiento de los reintentos).

Una vez entradas dichas medidas, las reenvía hacia el módulo 2 a través de la UART especificada en el interfaz UART & UART\_Sensors.

#### **Módulo 4: Uplink**

Este módulo se encarga de gestionar el interfaz PPM descrito anteriormente. Se trata de un procesador de Atmel integrado en una tarjeta Arduino Nano, junto con otra radio NRF24L01. La tarjeta Arduino Nano, incluye también regulador de tensión e interfaz USB para cubrir las mismas tareas que en el módulo 3, alimentación propia y de la radio, depuración y programación. El procesador programa la radio como transmisor y lo enlaza con la electrónica del drone. Tras el enlace, comienza a transmitir hacia el drone las órdenes de movimiento demoduladas de la trama PPM recibida periódicamente desde la FPGA del módulo 2.

#### **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 (Órdenes de movimiento recibidas desde el ordenador), UART\_Sensors (Medidas de posición recibidas desde la electrónica de sensorización del drone) y PPM\_Frames (tramas PPM resultado de procesar lo recibido por los dos interfaces previos). 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 inicio de dicha captura a la recepción de una nueva trama de órdenes de mando. De esta manera se realizan capturas de todos los eventos que se producen por ejemplo desde la recepción de la trama de despegue, hasta el aterrizaje. Esto ha hecho posible gran parte de la depuración de errores y fallos en la estabilidad del sistema o en su ejecución.

## 6.2. Sistema embarcado

Para el control del vuelo del drone, es necesario nutrir a la estación de tierra de información sobre el drone. Este sistema se encarga de obtener dicha información y enviársela a la estación de tierra. Para ello dispone de distintos sensores capaces de permitirle ubicar de manera relativa el drone, una radio para comunicar la información, y un procesador para realizar las lecturas de sensores, procesarlas y generar la trama a enviar. Estos módulos y sus interfaces se muestran en la Figura 6.3 y se describen a continuación.

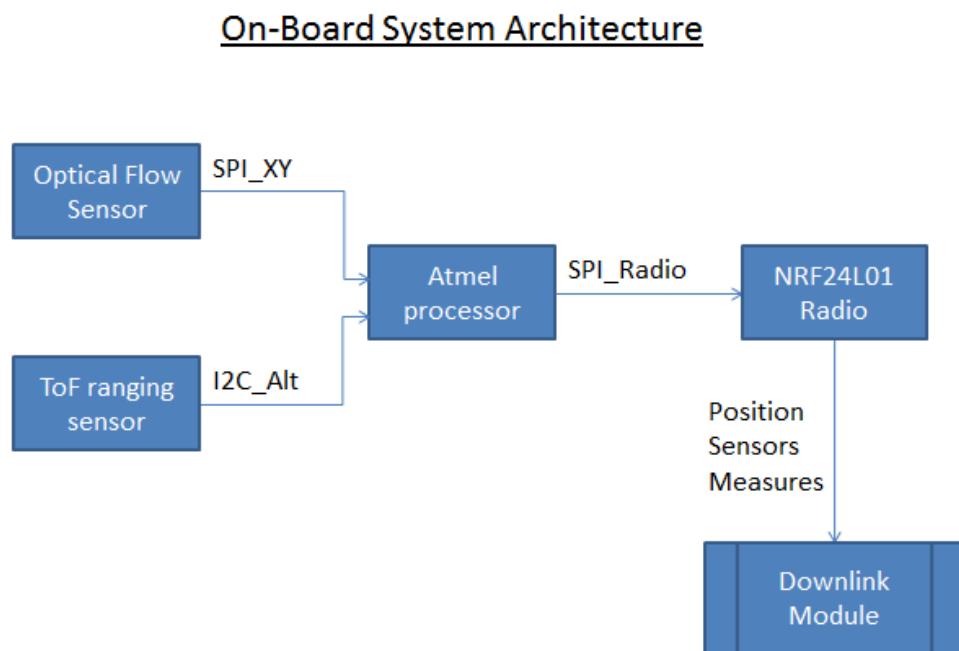


Figura 6.3: Arquitectura e interfaces del sistema sensor de posición ubicado en el drone

### 6.2.1. Descripcion

En la Figura 6.4 se muestran los módulos que componen el montaje de los sensores embarcados.

Este montaje incluye módulos adicionales a los sensores para hacer posible su lectura y comunicación a tierra:

- Módulo 1: Procesador Atmel.

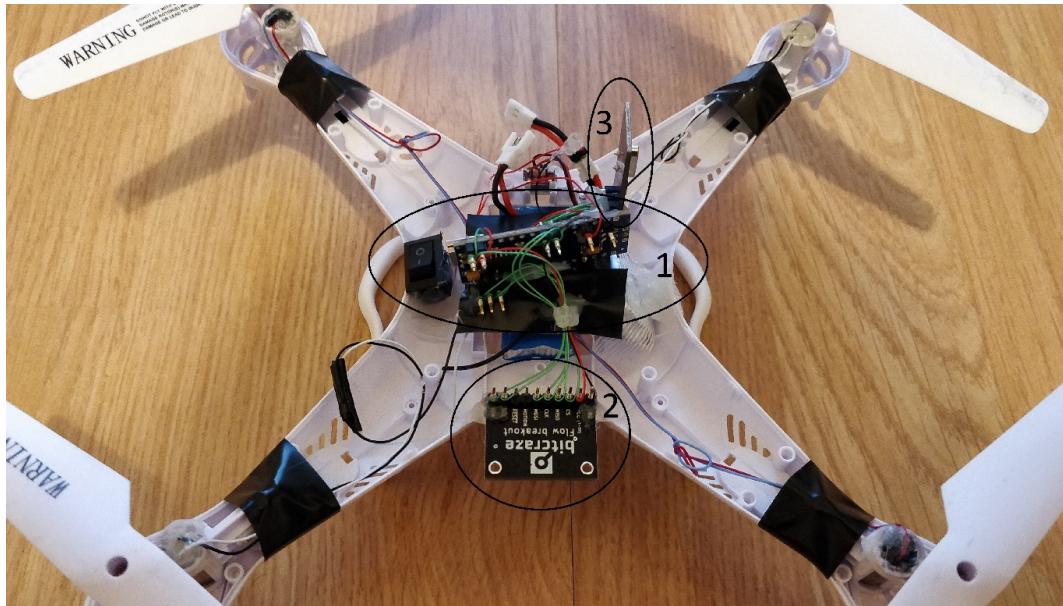


Figura 6.4: Sistema de sensores embarcado

- Módulo 2: Tarjeta de sensores.
- Módulo 3: Transmisor de downlink.

Estos módulos se alimentan desde un pack de batería de 3.6V de Litio, independiente de la electrónica de control del drone. Esto es debido a que a pesar del peso adicional que supone una batería separada, los drivers de motores del drone exigen picos de corriente considerables a las baterías. Estas, que no siempre tienen capacidad de descarga suficiente debido al diseño del drone, caen en tensión. Estas caídas, en caso de alimentar todo el sistema de las mismas baterías, llegan 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.

### 6.2.2. Objetivo

Los sensores embarcados tienen el objetivo de proveer a la estación de tierra, con información sobre la posición del drone en cada momento. Para ello cada módulo cubrirá con unas funciones concretas definidas a continuación y se comunicará la información generada a través de los interfaces de la Figura 6.3, explicados más adelante.

### 6.2.3. Funciones

- Medida de distancia al suelo. Función del módulo 2.
- Medida de velocidad de desplazamiento horizontal. Función del módulo 2.
- Procesado de las medidas hasta conseguir posición relativa del drone y entramado para transmisión. Función del módulo 1.
- Transmisión de paquetes con información de posición del drone hacia la estación de tierra. Función del módulo 3.

### 6.2.4. Módulos

#### 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 altura (ToF) sufre errores por debajo de los 3cm de distancia de medida, por tanto este módulo se encarga de filtrar dichos valores erróneos, entregando una medida ficticia de 3cm para toda altura igual o inferior. Esto hará que en ocasiones el drone, ya aterrizado, siga indicando una altura de 3cm. Esto no supone un problema ya que llegado el momento del aterrizaje, la altura ficticia de 3cm 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 estas, en medidas de posición relativas. Para ello integra en el tiempo las medidas de velocidad de desplazamiento, leídas del interfaz SPI\_XY. Ya que el sensor de flujo no mide correctamente para alturas inferiores a 4cm, se aprovecha la lógica de filtrado, para no modificar las medidas de posición en este rango de alturas.

Una vez filtradas e integradas, entrama dichas medidas y las reenvía hacia el módulo 3 a través del interfaz SPI\_Radio.

#### Módulo 2: Sensores

Este módulo se compone de una sola tarjeta que integra dos sensores. Juntos ofrecen lo necesario para llegar a conocer la ubicación relativa aproximada del drone. También incluye

electrónica auxiliar y conexiones.

- 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. Se accede desde el módulo 1, a través del interfaz I2C\_Alt, un puerto serie I2C. Tiene un rango de trabajo de entre 3cm y 2m. Este sensor se apunta hacia el suelo, desde su soporte en el drone, con la intención de medir la distancia al suelo desde el punto medio del drone. De esta manera una vez aterrizado, la distancia mínima de medida, es superior a 3cm, lo cual minimiza la necesidad de filtrado en el procesador (esta última se mantiene igualmente, ya que los apoyos del drone son flexibles, y el sensor tiene tolerancias de medida). Este sensor tiene un retardo de medida de hasta 30ms, el cual dificultará en cierta medida la estabilidad del bucle de control de altura ejecutado en la FPGA.
- Flow: Como sensor de desplazamiento se hace uso del PMW3901. 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. Entrega medida de velocidad de desplazamiento en dos ejes, para el plano captado por su óptica. El montaje apunta el sensor 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 drone respecto del suelo. Es el módulo 1 el encargado de convertir estas velocidades de desplazamiento en posición relativa. El sensor tiene un puerto de comunicaciones SPI para inicialización y lectura, sus medidas son entregadas al módulo 1 a través del interfaz SPI\_XY.

### Módulo 3: Transmisor de downlink

Se hace uso de una radio NRF24L01 conectada por puerto SPI al módulo 1, a través del interfaz SPI\_Radio. Este módulo transmite las medidas de posición de los sensores del módulo 2, ya procesados y entramados por el módulo 1, hacia la estación de tierra. Se configura la radio como transmisora, con una configuración en los parámetros radio, homóloga a la del módulo de downlink en la estación de tierra. Entonces se transmite una trama, justo después de realizar cada medida de ambos sensores, tiempo de vuelo y flujo. Las tramas son idénticas a las descritas en el apartado del downlink, con el mismo añadido al principio de dos bytes de valor 255 y 90.

Esto lo hace compatible con la configuración del módulo de downlink de la estación de tierra. Incluye los mismos mecanismos de retransmisión y comprobación de redundancia cíclica que la radio de la estación de tierra.

### 6.2.5. Interfaces externos

#### Position Sensors Measures

Como se muestra en la arquitectura de la Figura 6.3, enlaza por radio la electrónica de sensores y la estación de tierra. Porta las medidas de posición tomadas por los sensores del módulo 2 y procesadas por el módulo 1. Sus características y tramas están descritas en el apartado del downlink.

#### USB

Se trata de una comunicación con el ordenador de programación, exclusivo para reprogramación del procesador de Atmel, o extracción de medidas por puerto serie para comprobaciones y depuración.

### 6.2.6. Interfaces internos

#### 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 drone y el suelo bajo él, consideradas como la altura del drone.

#### SPI\_XY

Comunica por puerto serie SPI, el sensor de medida de velocidad de desplazamiento, PMW3901, del módulo 2 con el procesador del módulo 1. Porta las medidas velocidad de desplazamiento sobre el plano horizontal en ambos ejes. En este punto, no existe una referencia concreta, ya que se trata de una medida de la velocidad de desplazamiento instantánea.

**SPI\_Radio**

Puerto serie SPI utilizado para la configuración inicial de la radio NRF24L01, y para la indicación de tramas a enviar. La configuración inicial se parametriza de manera idéntica a la indicada para el downlink, en este caso como radio de transmisión. El formato de tramas coincide de igual manera con el del downlink por compatibilidad. Porta las medidas, ya procesadas por el módulo 1, de los interfaces I2C\_Alt y SPI\_XY.



# Capítulo 7

## Módulos Software

El sistema hace uso de 4 procesadores distintos. Tres de ellos ejecutan código C++ para tareas de gestión de puerto serie, inicialización de módulos y comunicación de datos. El cuarto es la FPGA, cuya lógica se define en Verilog 2001 y cubre tareas de procesamiento de señales mediante bucles de control, diseñados en base a bucles cerrados PID y comunicación de las medidas y resultados por puertos serie PPM respectivamente.

A continuación se explican los mecanismos ejecutados en cada caso.

### 7.1. Algoritmos de control sobre la FPGA

La lógica de la FPGA se encarga procesar información procedente de dos fuentes distintas, ordenador de mando y drone.

- Por un lado, recibe información desde el ordenador de mando, a través de un puerto serie asíncrono con destino en el módulo, UART PC Frames & Sync de la Figura 7.1. Las características y el formato de las tramas recibidas por esta vía se describen en el apartado Interfaz de mando. De cada trama se decodifica información para los tres bucles de control PID. La información decodificada por el módulo PC Frame Decoder incluye información de cada PID. Esta es enviada hacia cada controlador según corresponda. A través del flujo de datos Alt Command & Alt KpKiKd, se envía la información del comando de altura recibido, es decir, la altura deseada para el drone en ese instante, junto con los parámetros proporcional, integral y derivativo del bucle de control de altura. La misma información pero particularizada a cada uno de los dos ejes horizontales es portada

por los interfaces Forward Command & XY KpKiKd y Side Command & XY KpKiKd. Con la particularidad de que estos dos comparten parámetros PID. De esta manera puede actualizarse casi por completo el comportamiento de los bucles de control, según lo que se reciba en cada trama de mando enviada. Adicionalmente la trama incluye información sobre la orden de giro a entregar directamente al módulo de escalado. Este parámetro, como se ha comentado anteriormente, no pasa por un PID ejecutado en la FPGA, sino que se transfiere directamente hacia el drone, por no disponer de un elemento de medida a bordo. Dicho elemento de medida no se introdujo ya que los errores de giro son suficientemente pequeños como para despreciarlos.

- Por otro lado se recibe información desde los sensores embarcados en el drone en el módulo UART Drone Frames & Sync, decodificada en este caso por el módulo Drone Frame Decoder. Las medidas tomadas son entregadas a través de los flujos de datos X, Y, Z Measures, a los bucles de control correspondientes. Estos hacen referencia al desplazamiento lateral (Bucle Side PID), frontal (Bucle Forw PID) y en altura (Bucle Alt PID), respectivamente. En este punto, los bucles y el módulo de escalado de giro, tienen toda la información necesaria para trabajar.

Cada bucle de control de cada eje se diseña en base a un controlador PID. Cada uno Tendrá entonces como entrada de comando, el valor recibido desde el decodificador de tramas de PC para su eje concreto. Como entrada de retroalimentación, tendrán la medida del sensor de desplazamiento del eje acorde, recibido desde el decodificador de tramas del drone. Calculando la diferencia entre el comando y la medida del sensor, se tendrá el error a procesar por el PID de dicho eje, de manera independiente de los demás ejes. Los bucles integral y derivativo, realizan su tarea diferencial a la velocidad de cierre del bucle. Esta es de 30.3Hz. Por tanto el tiempo de derivación y el tiempo de integración serán constantes con un valor de 33ms. Por el contrario como se ha comentado anteriormente, los parámetros PID, por los que se multiplican las componentes del error proporcional, el error integrado en el tiempo y el error derivado respecto del tiempo, son variables recibidas en las tramas de mando. Dichas variables suelen modificarse en momento clave de la ejecución de las instrucciones de mando, como despegue, vuelo y aterrizaje, permitiendo cierta flexibilidad a la hora de realizar distintas tareas.

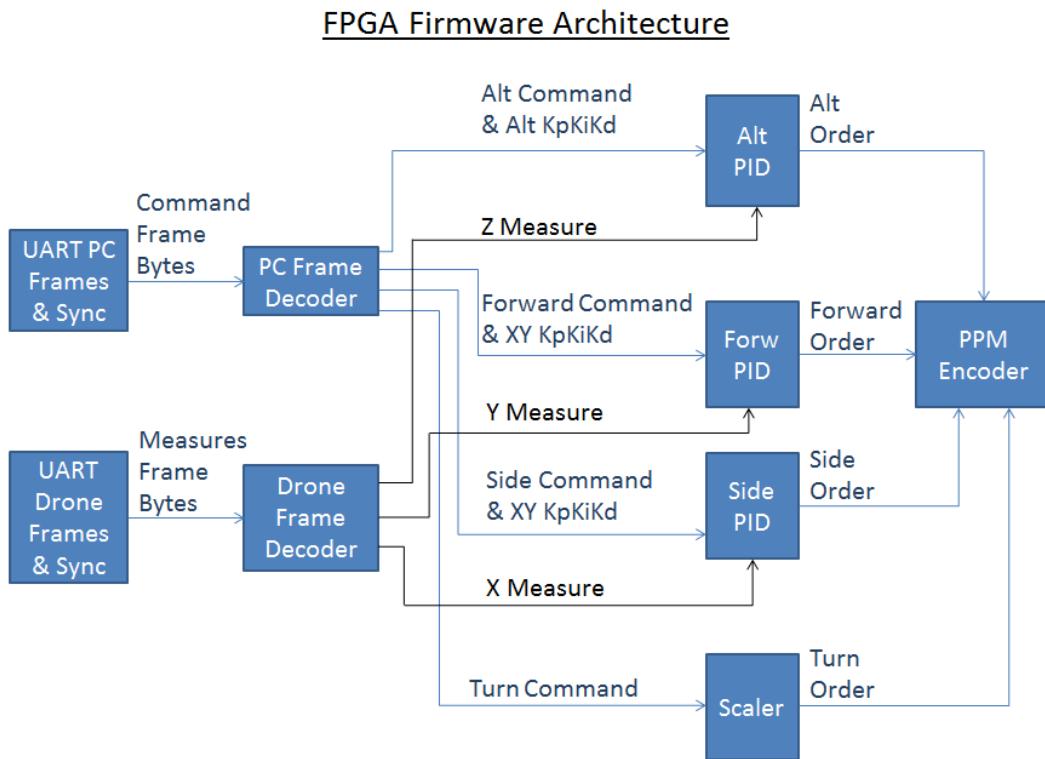


Figura 7.1: Arquitectura del firmware programado sobre la FPGA

## 7.2. Software en el módulo de Downlink embarcado

El software planteado a bordo del drone se encarga de inicializar radio y sensores, para posteriormente realizar una medida de cada sensor y transmitirlas por radio. El objetivo es informar a la estación de tierra de las medidas de a bordo. Sus requisitos son, consumir la menor cantidad de energía posible, realizar su función rápidamente, entreteniéndose lo menor posible en tareas secundarias, y transmitir las medidas con la menor cantidad de latencia posible. Para cubrir estos objetivos, el software se ha diseñado de manera sencilla. Se comienza inicializando 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
}

```

```
};
```

Entonces se realiza una medida de cada sensor, se filtran las medidas y el resultado se almacena en la estructura TxFrame. Esta es transmitida por radio a 2 Mbps, y se comienza de nuevo el bucle hasta que se apague la electrónica.

### 7.3. Librería Python

El software ejecutado en python en el ordenador de mando es el encargado de enviar las tramas de mando hacia la estación de tierra. Este se encarga de permitir 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.

A través de la ejecución de la función *setPIDValues* dentro del módulo control\_functions.py se cargan los valores de las constantes para los PIDs vertical y horizontales, junto al offset para la corrección del error de giro.

```
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 bucle 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 drone.

Tras al menos una ejecución de esta función, se puede proceder a enviar instrucciones de mando al drone. Estas pueden enviarse de manera directa a través de la instrucción setcontrols:

```
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 su destino a través de los flujos de datos Alt\_Command, Side\_Command, Forward\_Command de la Figura 7.1, respectivamente.

- El parámetro GiroDI, es enviado hacia el módulo de escalado de rango, Scaler, de la arquitectura de la Figura 7.1.
- Duracion, es el parámetro utilizado para especificar el tiempo mínimo de espera para la ejecución de esta instrucción, en segundos. Tras este periodo de tiempo se podrá ejecutar la siguiente instrucción de mando.

Adicionalmente al método de control directo, se puede hacer uso de la función settrace.

```
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 en casos en los que se quiera recorrer distancias de más de pocos centímetros. Esto es debido a que en caso de por ejemplo, despegue y posterior indicación de vuelo a una distancia considerable del punto de origen, los bucles de control partirían de un error grande, lo que se intenta resolver con un empuje considerable en la dirección de corrección del error. A pesar del posterior suavizado en la aceleración, de la componente derivativa, este empuje excesivo en ocasiones puede desestabilizar el control de los bucles PID y dificultar la adecuada medida de la ubicación por parte del sensor óptico. Además al inclinar bruscamente el drone, el sensor de altura mide una distancia mayor, debido al nuevo ángulo de incidencia sobre el suelo. Esto se traduce en un avance tortuoso del drone, como avanzando “a tirones” hacia su destino. Para evitar esto, *settrace* genera la ristra de avances de menor distancia, evitando el crecimiento excesivo de los errores calculados por los bucles.



# **Capítulo 8**

## **Validación experimental**

### **8.1. Eachine E010**

En un inicio se hico uso del drone Eachine E010. Un pequeño cuadricóptero, ligero y de comportamiento nervioso. Sobre él se hicieron pruebas de control en bucle abierto. Se realizaron las tareas de enlace de comunicaciones entre la estación de tierra y el drone y se probaron despegues y vuelos cortos en línea recta.

El drone mostró buen control de vuelo en tiempo cortos, por tanto fue válido para pruebas con algo de pre-énfasis. Sabido su desvió a derechas y hacia abajo, se prepararon vuelos de avance y despegue en los que progresivamente se aumentaba la fuerza de los motores hacia la izquierda y hacia arriba. Estas dificultades hicieron los vuelos largos en bucle abierto imposibles, por tanto se intentó instalar el controlador de vuelo de bucle cerrado en dicho drone. Desgraciadamente el peso de la electrónica disponible era excesivo y hubo que abortar los intentos y buscar una plataforma de mayor potencia. Por este motivo se utilizó el drone Syma X5C finalmente, frente al pequeño E010.

### **8.2. Syma X5C en Bucle abierto**

Tras las pruebas con el E010, inicialmente se tuvo que conseguir el mismo control sobre el X5C. Para ello se modificó el código de enlace con el drone, ejecutado en el módulo de uplink, para conseguir control básico sobre el X5C. Una vez enlazado se hicieron pruebas de control en bucle abierto de igual manera que con el E010. El drone Syma, resultó también difícil de

controlar en bucle abierto para vuelos de más de 10 o 15 segundos. A parte de sus desvíos poco previsibles en cada eje, lo cual dificultaba ejecutar correcciones en base a ?trimado?, este drone presentaba una degradación considerablemente notable en la potencia de los motores a medida que se gastaba la batería. 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 en bucle abierto. Se procedió por tanto rápidamente a instalarle el sistema de medida de posición. El X5C si era capaz de levantar la electrónica embarcada, pero sus motores quedaban encendidos casi al máximo de su potencia para mantener el vuelo. Esto continuaba dificultando el correcto control en altura, generando oscilaciones de una amplitud imposible de corregir para el PID de altura. En este punto, al menos existía un mínimo de control sobre el drone, con escasa estabilidad. Por tanto se procedió a aligerar la electrónica con cambios menores, y aligerar el drone, con modificaciones mayores. Básicamente se libró al drone de todo aquello estrictamente innecesario 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 drone. Algunos elementos retirados se sustituyeron por puntos de soldadura o pegamento, en pro de aligerar todo lo posible. 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 permitió a los controladores, ganar bastante en cuanto a estabilización del drone. En este punto el trabajo pendiente se centró en mejorar los parámetros PID de los controladores diseñados para el drone previo al cambio de peso, ya que los anteriores no eran correctos con el nuevo sistema. Además se trabajó rotando dos baterías y controlando rigurosamente su nivel de carga, ya que aunque los bucles de control PID corrigen parte de la disminución de potencia en los motores debido a la descarga de las baterías, llegado un nivel de carga bajo, los parámetros de control mismos, dejaban de ser igual de efectivos, y la velocidad de respuesta también disminuía, lo que al final se traducía en el mismo efecto, parámetros PID inadecuados. Procurando trabajar con baterías en un rango de carga concreto, se minimizó lo suficiente este efecto como para poder mejorar la estabilidad y control sobre el drone.

## 8.3. Condiciones de los bucles de control

Desde un punto de vista más alejado que solo centrarse en los parámetros PID de los bucles, existen algunos factores importantes que limitan los resultados obtenibles con una arquitectura concreta. Para el caso de este trabajo, la arquitectura elegida permitió usar tarjetas de desarrollo, en vez de hardware propietario. Pero a su vez implica ciertas limitaciones en el ancho de banda final del sistema, y sobre todo en las latencias. Ambos dos impactan en el resultado del control final.

### 8.3.1. Ancho de banda

El ancho de banda se vio limitado por las velocidades de ejecución más lentas dentro del sistema, estas son la velocidad de recepción de tramas de la electrónica propia del drone, la cual limita cuantas órdenes puede atender por segundo, y la lectura de los sensores, limitada por la electrónica embarcada y las especificaciones de los propios sensores. Finalmente se estableció una velocidad de lectura de 30.3 hercios, que permitía a los sensores ser leídos adecuadamente, y a la electrónica propia del drone, atender todas las tramas entrantes. Esto limita la brusquedad de las reacciones del drone, en caso de correcciones necesarias, y la suavidad del mismo durante su vuelo.

### 8.3.2. 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. Esta se pone en riesgo por múltiples factores, algunos ya comentados, pero principalmente por 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 drone, genera retardos asociados a la transmisión de los valores. Las medidas no son procesadas a

bordo sino que se transmiten hacia tierra, donde se procesan para obtener una respuesta, y dicha respuesta se devuelve al drone. Este proceso de transmisión a tierra añade retardos indeseados, pero necesarios para el funcionamiento del sistema.

Si tratamos de contabilizar a groso modo los retardos existentes, tenemos:

- 30ms del sensor de medida de altura, ToF.
- Retardo máximo añadido por la sincronía entre la trama PPM enviada desde la FPGA, y la lectura realizada por la electrónica de uplink, 33ms.
- Enlaces radio, no más de 120uS sin contar preámbulos.
- Tres comunicaciones SPI, unos 144us.
- 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 la posición y reacción. Esto, sin ser una cifra catastrófica, si es cierto que implica ciertas limitaciones a la hora de conseguir una suavidad de vuelo concreta.

# Capítulo 9

## Conclusiones

Para terminar, realizaremos un repaso de los objetivos alcanzados, los métodos y las dificultades encontradas en el proceso. Gracias a ellas se ha sufrido y aprendido durante todo el proyecto. A cada elemento diseñado, se han encontrado formas mejores de hacerlo en un futuro, alternativas con propiedades distintas y opciones a elementos sueltos o la propia arquitectura misma, que podrían dar resultados diferentes y mejores. Finalmente se ha conseguido un resultado satisfactorio con el objetivo principal que se tenía, estabilizar y controlar un drone de bajo coste desde un ordenador. El trabajo podría dividirse en dos grandes bloques, control del drone mediante bucles abiertos y control mediante bucles cerrados.

- Control mediante bucles abiertos: Esta fue la primera aproximación a tener un control mínimo de los vehículos desde el ordenador. El resultado fue positivo desde el punto de vista de lo que este tipo de sistemas pueden llegar a ofrecer. Se diseñó software y hardware completamente funcional, permitiendo el enlace con el drone y su obediencia ante las instrucciones dadas desde el pc. Pero las derivas propias de los drones, especialmente notables en drones de bajo coste, dificultaron excesivamente un buen control del mismo mediante esta metodología. Los resultados conseguidos permitían controlar un drone, comandándole instrucciones que este obedecía, pero dicho control era precario, faltó de precisión y repetitividad. Para un ejercicio de despegue y parada en vuelo, llegaba un momento en que dichas derivas dirigían el drone contra el suelo, contra una pared, etc. La problemática de este método era difícil de subsanar con la infraestructura disponible, por este motivo se procedió a evolucionar todo el sistema, hardware y software, tanto en tierra como a bordo, dotándolo de una nueva dimensión, conocimiento en el drone de la propia

ubicación relativa del drone. Esto supuso poder avanzar al siguiente tipo de control.

- Control mediante bucles cerrados: Esta fue la arquitectura definitiva diseñada para el control del drone. Se diseñó electrónica y software para permitir al drone medir su propia ubicación respecto del punto de despegue y transmitirla al sistema en tierra. La estación de tierra recibió añadidos hardware, y una completa actualización de software para recibir, procesar y corregir la ubicación del drone mediante PIDs, enlaces serie y radio, satisfactoriamente. El sistema en este punto permitió recibir instrucciones de posición desde el ordenador y controlar el drone para obedecerlas. Mediante estos cambios se consiguieron vuelos significativamente más estables y similares entre sí. Los cambios permitieron además ampliar la duración de los vuelos hasta los límites impuestos por los niveles de carga en las baterías, lo que facilitó las tareas de ajuste de parámetros PID y experimentación.

El trabajo realizado pone a disposición pública el hardware, software y diseño del sistema, listos para que cualquiera persona pueda adquirir los elementos por un bajo coste, instalarlos y ponerlos en funcionamiento. Esto ofrece un gran abanico de posibles modificaciones, mejoras y desarrollos futuros.

## 9.1. Trabajo futuro

El sistema desarrollado ofrece muchas posibilidades de mejoras y añadidos. En este apartado se pretenden repasar algunos de los más interesantes.

### 9.1.1. Mejoras en la arquitectura del sistema

- La arquitectura del sistema es posiblemente la consideración de diseño cuyo cambio tenga un mayor impacto en el resultado final. La premisa de procesar con la FPGA en tierra, tiene ventajas y desventajas. Probablemente la peor contrapartida sea 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 drone. Estos retardos añadidos, empeoran la estabilidad que se puede llegar a conseguir, con el vehículo en vuelo. El cambio más significativo pasaría posiblemente por trasladar la FPGA a la electrónica embarcada, 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 a más de la mitad, ganando en favor de la calidad del control que se ejerce sobre el drone.

### 9.1.2. Mejoras hardware

- Diseñar una única tarjeta para el sistema embarcado 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 (debido al error de colocación de los sensores de flujo y de altura) y que la potencia entregada por los motores fuese más holgada, mejorando la estabilidad y ahorrando batería.
- A parte de cambios sobre el sistema, se pueden plantear añadidos de interés para el sistema final, como podrían ser un conjunto de magnetómetros de 2 ejes (ubicados de manera horizontal) para medir el error de giro, lo que permitiría corregirlo, a la vez de ganar un parámetros de orientación global. 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. Ambas mejoras ofrecerían más capacidad de vuelo autónomo, a través de un mejor conocimiento sobre la ubicación del vehículo.

### 9.1.3. Mejoras software

- Respecto del diseño software, es difícil no encontrar 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, ganando en estabilidad.
- 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 actuales realizan su tarea en base exclusivamente a la posición del vehículo, pero no reparan en limitaciones de velocidad o aceleración. Una mejora potente sería ubicar bucles de control de velocidad y aceleración bajo los actuales bucles de control de posición. Esto controlaría las brusquedades y dotaría al sistema de un mayor control sobre cómo se desplaza el vehículo.
- Generador de trayectorias: Para poder acometer un generador de trayectorias completo, serían necesarias modificaciones hardware y software en la estación de tierra. Estas deberían permitir a la estación de tierra informar al ordenador de mando de la ubicación actual del drone (obtenida de las medidas de los sensores embarcados). Con este añadido se permitiría al software en el pc crear verdaderos puntos intermedios en el avance de drone, dependientes tanto de la posición actual, como de la posición de destino, trazando así una trayectoria sin incrementos exagerados en el error de los bucles de control.

## Bibliografía

<https://www.parrot.com/soluciones-business/profesional/parrot-sequoia>  
<http://www.centum-rt.com/en/lifeseeker/>  
<http://www.ingenieros.es/noticias/ver/life-seeker-sistema-para-la-localiz>  
<https://www.bbc.com/news/technology-38450664>  
<https://www.digikey.com/en/articles/techzone/2012/jul/a-designers-guide-t>  
<https://charlestytler.com/quadcopter-equations-motion/>  
[https://www.infineon.com/dgdl/Infineon-Application-Motor\\_Control-Drone\\_El](https://www.infineon.com/dgdl/Infineon-Application-Motor_Control-Drone_El)  
<http://fpvmax.com/2017/08/09/pids-ajuste-drones/>  
<https://www.arrow.com/es-mx/research-and-events/articles/fpgas-in-neural->  
<https://core.ac.uk/download/pdf/154797518.pdf>  
<https://www.luisllamas.es/arduino-spi/>  
[https://twitter.com/obijuan\\_cube/status/1135920285205401600](https://twitter.com/obijuan_cube/status/1135920285205401600)  
<https://www.solitontech.com/uart-protocol-validation-service/>  
<http://www.latticesemi.com/-/media/LatticeSemi/Documents/DataSheets/iCE/i>  
<https://www.mouser.es/pdfdocs/enDM00270461.pdf>  
<https://www.bitcraze.io/flow-breakout/>  
[https://wiki.bitcraze.io/\\_media/projects:crazyflie2:expansionboards:pot01](https://wiki.bitcraze.io/_media/projects:crazyflie2:expansionboards:pot01)  
<https://hipertextual.com/presentado-por/vodafone-one/paola-santana>  
<https://www.amazon.com/Amazon-Prime-Air/b?ie=UTF8&node=8037720011>  
[http://www.automatas.org/hardware/teoria\\_pid.htm](http://www.automatas.org/hardware/teoria_pid.htm)  
<https://teslabem.com/nivel-intermedio/fundamentos-del-protocolo-i2c-apren>