Universidad Zaragoza
1542

# Trabajo Fin de Grado

## Navegación autónoma de un multi-rotor: control automático de la altura mediante sensor láser

## Autonomous navigation of a multi-rotor: automatic altitude control with laser sensor


Autor

David Pont Esteban

Directores

Luis Montano Gella
José Luis Villarroel Salcedo



Escuela de Ingeniería y Arquitectura
Zaragoza, Septiembre de 2017

TRABAJOS DE FIN DE GRADO / FIN DE MÁSTER

Escuela de Ingeniería y Arquitectura Universidad Zaragoza

# DECLARACIÓN DE AUTORÍA Y ORIGINALIDAD

(Este documento debe acompañar al Trabajo Fin de Grado (TFG)/Trabajo Fin de Máster (TFM) cuando sea depositado para su evaluación).

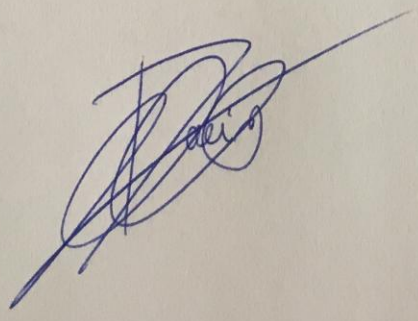
D./Dª. David Pont Esteban,

con nº de DNI 48250698X en aplicación de lo dispuesto en el art. 14 (Derechos de autor) del Acuerdo de 11 de septiembre de 2014, del Consejo de Gobierno, por el que se aprueba el Reglamento de los TFG y TFM de la Universidad de Zaragoza,

Declaro que el presente Trabajo de Fin de (Grado/Máster) Grado en Ingeniería Electrónica y Automática, (Título del Trabajo) Navegación autónoma de un multi-rotor: control automático de altura mediante sensor láser,

es de mi autoría y es original, no habiéndose utilizado fuente sin ser citada debidamente.

Zaragoza, 1 de septiembre de 2017



Fdo: David Pont Esteban

# RESUMEN

# Navegación autónoma de un multi-rotor: control automático de altura mediante sensor láser

Se ha desarrollado el control de altura de un *quadcopter* mediante el uso del escáner láser Hokuyo URG-04LX, para medir la altura del drone respecto al suelo. El usuario será capaz de fijar la consigna de altura del drone desde un ordenador, de manera que el drone se adapte a dicha altura y consiga estabilizarse una vez alcanzada. Los demás grados de libertad del drone pueden manipularse mediante el mando de control.

Para implementar dicho control ha sido fundamental todo el trabajo de puesta a punto del drone, dejándolo en las condiciones más óptimas posibles para facilitar la labor del controlador en un sistema con gran tendencia a desestabilizarse.

La implementación de la aplicación se ha realizado en el microcontrolador TMS320F28377S de Texas Instruments y se ha construido sobre el sistema de tiempo real SYSBIOS facilitado por el entorno de programación Code Composer Studio. Se ha trabajado con las últimas versiones de todos estos paquetes.

Finalmente se ha procedido a la implementación del control en el drone real, realizando varios ensayos para ajustar correctamente los parámetros del controlador y se han obtenido resultados satisfactorios.

# ÍNDICE

# Capítulo 1

# Introducción

## 1.1. Contexto y estado del arte

Los UAVs [ 14 ] (del inglés *Unmanned Aerial Vehicle*), también conocidos UAS (*Unmanned Aerial System*) como *drones* o VANT (*Vehículo Aéreo No Tripulado*), son aeronaves capaces de volar sin tripulación a bordo. Los UAS constan de dos partes bien diferenciadas: el dispositivo aéreo (la aeronave) y la estación terrestre de control (*Ground Control Station*). Estos sistemas pueden ser controlados remotamente de forma manual o pueden estar programados para realizar rutas de vuelo de forma totalmente automática mediante la secuenciación de *waypoints* sin necesidad de pilotarlos.

Al igual que gran parte de los avances tecnológicos de la sociedad, el desarrollo de los drones tiene su origen en el campo militar. El primer vuelo de un drone data del año 1917, el cual tuvo una duración de más de 50 millas. El rudimentario UAV era un modelo derivado del avión de combate *U.S Navy Curtis N-9 trainer*. En la actualidad, y como ya viene siendo desde hace unos 8 años, los drones han dado el salto del campo militar al campo civil, generándose un gran abanico de nuevas aplicaciones para ellos, como por ejemplo la cartografía aérea, vigilancia, mantenimiento y revisión de infraestructuras, control medioambiental entre muchas otras.

Una de las líneas de investigación en cuanto a drones que está actualmente en mayor auge es el vuelo en interiores. En interiores no se dispone de sistema GPS, que es el medio de localización que utilizan los drones convencionales, por lo que es necesario utilizar otros sistemas que permitan la localización, como por ejemplo sistemas de visión por computador, sensores inerciales o escáneres láser para evitar las colisiones con objetos incluso en entornos desconocidos.

El proyecto se va a realizar en esta línea de investigación. Para poder volar en interiores desconocidos es vital poder mantener una altura fija respecto al suelo que permita al drone desplazarse por el espacio de forma segura y estable.

Este proyecto se lleva a cabo dentro del grupo de Robótica, Percepción y Tiempo Real de la Universidad de Zaragoza. Éste es uno de los grupos de investigación del Instituto Universitario

de Investigación en Ingeniería de Aragón (I3A) y es considerado Grupo de Investigación por el Gobierno de Aragón. Dicho grupo tiene las siguientes líneas de trabajo:

- Localización y Mapeado Simultáneo.
- Visión por Computador y Percepción.
- Comunicaciones y redes ad-hoc.
- Exoesqueletos y procesamiento de bioseñales.
- Aprendizaje: en robótica, optimización Bayesiana, interfaces cerebro-ordenador...
- Robótica Móvil. Planificación y navegación.

El proyecto desarrollado se enmarca en ésta última línea de trabajo.

## 1.2. Objetivo y alcance

El objetivo del proyecto es el diseño de una aplicación capaz de controlar la altura de vuelo del quadcopter *F450 FlameWheel* de la marca DJI. El control a implementar se realizará un nivel por encima el autopiloto NAZA M Lite, que será el encargado de estabilizar el drone respondiendo a las consignas de los cuatro grados de libertad que se suelen poder manipular en un drone comercial: el nivel del acelerador (*throttle*) y los tres ángulos de orientación. El controlador diseñado en el proyecto dará al autopiloto la consigna de *throttle* adecuada en cada momento para que la altura sea la deseada.

Para poder implementar el control de altura se hará uso de dos sensores: un escáner láser Hokuyo URG-04LX que será el encargado de proporcionar al sistema una medida precisa de la distancia el drone respecto al suelo y el IMU Razor 9dof, del cual se obtendrán las medidas de los ángulos *roll* y *pitch* que permitirán corregir la medida de distancia obtenida por el láser de tal manera que se podrá obtener una estimación de la distancia perpendicular del drone al suelo. Se deberá diseñar el driver para el láser de la forma más modular posible dado que habrá que experimentar con distintas formas de realizar la medida de altura puesto que a priori no se conoce la forma más adecuada.

El drone será dotado de un sistema de comunicaciones con el ordenador que se implementará mediante un módulo XBee. De esta manera se le darán al drone las consignas de altura y también se podrán realizar ajustes en los parámetros de control. Por otro lado el drone enviará periódicamente la altura actual al ordenador.

Finalmente se pretenden realizar ensayos con el drone para analizar las prestaciones del esquema de control planteado y observar qué resultados es capaz de ofrecer después de ajustar los parámetros de control.

El sistema de control será implementado en el microcontrolador TMS320F8377S y será programado sobre el sistema de tiempo real SYSBIOS mediante el entorno Code Composer Studio.

Se cuenta con la información realizada en otros trabajos de fin de carrera anteriores [ 3 ] en la misma línea como base.

## 1.3. Organización de la memoria

La memoria se divide en 6 capítulos:

- Hardware y software: en este capítulo se describe el hardware utilizado en el proyecto, así como los distintos programas que se han utilizado.

- Arquitectura del sistema: en este capítulo se muestra el diagrama de bloques del sistema implementado y se explica la interconexión entre los distintos dispositivos.

- Ley de control: en este capítulo se exponen los distintos controladores diseñados, las simulaciones realizadas y los controladores implementados finalmente.

- Implementación: en este capítulo se describen las soluciones adoptadas para poder hacer posible el control, tanto a nivel software como hardware.

- Pruebas de la aplicación: en este capítulo se muestra la prueba más destacada realizada con el drone junto con sus resultados.

- Conclusiones: en este capítulo se extraen las conclusiones del proyecto realizado.

# Capítulo 2

# Hardware y software

En este capítulo se describen en detalle el hardware y el software empleados en la realización del proyecto.

## 2.1. Hardware

### 2.1.1. Drone utilizado

El drone del que se dispone es un *quadcopter* Flame Wheel F450 de DJI (Figura 2.1). Al drone se le han añadido varios elementos, con lo que el resultado final varía sustancialmente respecto a lo que se puede apreciar en la figura 2.1.

**Figura 2.1**

#### 2.1.1.1. Estructura

El *quadrotor* está formado por 4 brazos de plástico resistente pero lo suficientemente flexible como para que no se rompa ante impactos de baja o media envergadura. Dos de los brazos están pintados de color rojo y los otros dos de color blanco, siendo así distinguible a simple vista la parte frontal de la aeronave, la roja, facilitando así el pilotaje. Los cuatro brazos se unen mediante dos tornillos cada uno a las dos placas centrales del drone, de tal manera que a la vez

que se aporta rigidez a la estructura por la presión de las placas centrales sobre los 4 brazos, también queda espacio libre entre ambas para albergar el hardware necesario. La placa inferior tiene preparados puntos de soldadura para facilitar la distribución de la alimentación principal del drone a los diversos dispositivos que lo requieran.

Tal y como se puede observar en la figura 1.1 el chasis del drone cuenta en el extremo de cada uno de los brazos con un pie de apoyo para el aterrizaje. No obstante, dado el hardware que va a ser necesario embarcar en el drone y que se desea aumentar la resistencia a los impactos, se ha optado por añadir un tren de aterrizaje comercial. Dada la complejidad del montaje de los distintos elementos a embarcar al drone y su influencia sobre el resultado final, se profundizará más en la estructuración del mismo en el apartado

*Figura 5.18*

*5.2.3.* Montaje del drone.

### 2.1.1.2. Autopiloto

El autopiloto utilizado es el NAZA M Lite (Figura 2.2). Tiene cuatro entradas para el control del drone: las de los tres ángulos de orientación y la del acelerador. Las consignas para cada uno de los canales mencionados se transmiten mediante una señal PWM. El algoritmo de control implementado en el autopiloto se encarga de la estabilización del drone mediante las medidas obtenidas por su giróscopo, acelerómetro y barómetro internos.


**Figura 2.2**

### 2.1.1.3 Batería

La batería utilizada para la aplicación es la Trunigy 5.0 4S 30C (Figura 2.3). Tiene una tensión a plena carga de 16.8V y un peso de 631g. La batería constituye la fuente de alimentación principal, a partir de la cual se extraerán los niveles de tensión necesarios para alimentar los diversos componentes. Para alimentar el autopiloto se ha utilizado el ·3DR Power Module, que a partir de la entrada de la batería pone a su salida una tensión de 5.3V.


**Figura 2.3**

### 2.1.1.4. Motores

El drone está equipado con cuatro motores DJI 2212/920Kv (Figura 2.4). Cada motor ofrece un empuje máximo de 1.2kg [ 10 ]. Para la batería utilizada (4 celdas), las hélices adecuadas para el motor recomendadas por el fabricante, y que por tanto las que se han montado, son de 10 pulgadas.

**Figura 2.4**

### 2.1.1.5. Equipo de radiocontrol

La mando de radiocontrol utilizado para pilotar el drone es la Futaba 6J [ 13 ]. Dicha radio tiene 6 canales y trabaja a una frecuencia de 2.4GHz. El mando emisor se comunica con el receptor R2006GS de doble antena embarcado en el drone. En la aplicación en cuestión se van a utilizar únicamente 5 canales, de los cuales 4 se utilizan para dar manualmente las consignas de roll, pitch, yaw y throttle, y el quinto canal se utiliza para la selección del modo de vuelo, así como para conectar el mando o la salida pwm del microcontrolador a la entrada de throttle del autopiloto. En el capítulo *19*, se detalla este tema.

### 2.1.2. Microcontrolador

El microcontrolador utilizado es el TMS320F28377S, diseñado por Texas Instruments. Se hace uso de la placa de desarrollo LAUNCHXL-F28377S (Figura 2.5), que básicamente embarca el microcontrolador sacando sus puertos a conectores externos para que sea posible el acceso a ellos. Además la placa incorpora el módulo de *debugging* XDS100v2 para testear la aplicación desde el entorno de programación.

**Figura 2.5**

El microcontrolador tiene una tensión nominal de alimentación de 3.3V, una CPU de 32 bits y una frecuencia máxima de reloj de 200MHz. Tiene una memoria RAM de 164KB y una FLASH de 1MB. Entre sus periféricos destacan dos conversores AD de hasta 16 bits con hasta 14 entradas, 15 canales PWM, 2 módulos $I^2C$ y 3 módulos SCI.

El F28377S tiene capacidad para ejecutar aplicaciones basadas en el sistema operativo de tiempo real SYS/BIOS. El microcontrolador se programa con el entorno de programación Code Composer Studio.

### 2.1.3. Sistema de medida de altura

Para la medida de la altura se utiliza el escáner láser URG-04LX de la marca Hokuyo. El láser es uno de los elementos centrales del proyecto. Debido a que el sensor utilizado no es unidireccional sino que es un escáner y va ubicarse solidario al drone y orientado horizontalmente, es necesario diseñar un elemento mecánico para redirigir los rayos emitidos por el sensor hacia el suelo. El sistema de medida láser se basa, por tanto, en los elementos que aparecen a continuación.

### 2.1.3.1. Escáner láser Hokuyo URG-04LX

El escáner láser URG-04LX (Figura 2.6) se basa en un emisor láser de longitud de onda de 785nm. El cálculo de la medida se fundamenta en la medida de la diferencia de fase entre el rayo emitido y el recibido. El rango máximo de escaneo del laser es de 240º, realizando una nueva medida cada 0.36º. La resolución del sensor es de 1mm y las distancias medibles se comprenden en el rango de 20mm a 4m. Es posible comunicarse con el sensor mediante comunicación USB o RS232 a velocidad configurable entre un amplio rango de valores. Su peso es de 160g las dimensiones son de 50x50x70mm. La tensión de alimentación es de 5V y el consumo de corriente puede llegar hasta los 800mA.


**Figura 2.6**

### 2.1.3.2. Adaptador RS232

Puesto que el microcontrolador se alimenta con una tensión de 3.3V y el láser con una de 5V es necesario introducir un módulo que adapte los niveles de tensión entre ambos elementos con el fin de que la comunicación sea factible. Para ello se dispone de una placa (Figura 2.7) fabricada con anterioridad que está basada en el circuito integrado MAX3222. Dicho circuito consta de dos canales de comunicación, aunque únicamente se utiliza uno de ellos. El MAX3222 transforma la tensión proveniente del microcontrolador del rango 0V a 3.3V al rango -5V a 5V para que llegue al láser con los niveles de tensión que éste requiere. Análogamente reduce la tensión proveniente del láser del rango -5V a 5V al rango 0V a 3.3V en el que trabaja el microcontrolador. La tensión a la que se alimenta el circuito es de 3.3V y la velocidad máxima de comunicación es 250kbps.

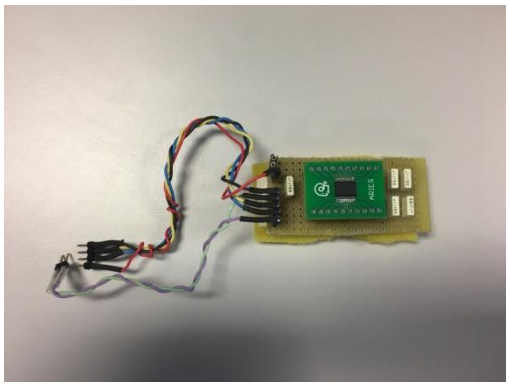


**Figura 2.7**

### 2.1.3.3. Estructura del láser

Se ha diseñado y fabricado una estructura (Figura 2.8) que permita acoplar el láser al drone y que a la vez permita redirigir hacia el suelo los rayos emitidos por el sensor. La estructura consta de una plancha que se atornilla al drone por la parte superior y se la atornilla el láser por la parte inferior. De la plancha salen 3 brazos con espejos en sus extremos que redirigen los rayos al suelo. En el apartado *5.2.1. Soporte para el láser* se explica el papel que juega la estructura en detalle.

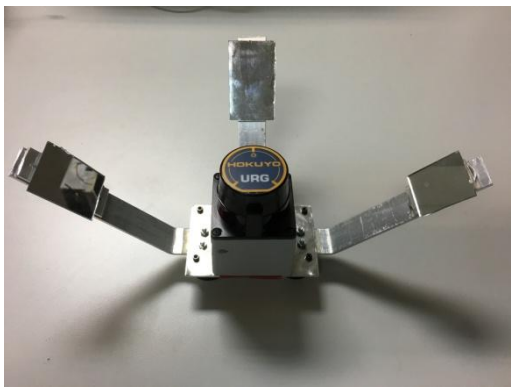



**Figura 2.8**

## 2.1.4. Unidad de medida inercial (IMU)

Se utiliza el IMU 9DOF Razor (Figura 2.9) que consta de un giróscopo de 3 ejes, un acelerómetro de 3 ejes y un magnetómetro de tres ejes. Un microcontrolador gestiona la medida de todas esas variables y las transmite por comunicación SCI  a una velocidad 57600bps con un periodo de 25ms. En el proyecto únicamente se utilizan las medidas de los ángulos roll y pitch para corregir la medida de la altura.

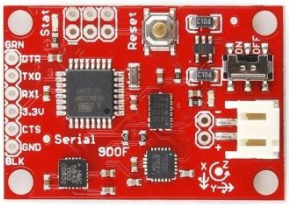



**Figura 2.9**

### 2.1.5. Comunicaciones

Se dispone de dos XBee (Figura 2.10). Uno de ellos va embarcado en el drone y el otro conectado al ordenador para implementar las comunicaciones entre ambos. El XBee [ 12 ] se conecta al microcontrolador mediante uno de los módulos SCI de los que éste dispone. La velocidad de comunicación que se ha elegido es de 9600bps, la tensión de alimentación nominal es de 3.3V y la frecuencia de radio es de 2.4GHz.

Las comunicaciones implementadas son bidireccionales, de tal forma que se le pueden enviar paquetes de datos al drone, como por ejemplo la consigna de altura, así como recibir información del mismo, como la altura actual.


**Figura 2.10**

### 2.1.6. Regulador de tensión de 5V

Con el fin de alimentar los distintos elementos electrónicos que van a bordo del drone, se debe reducir la tensión de salida de la batería a 5V y 3.3V. Se ha conectado a la batería un conversor

reductor de tensión. Este tipo de elemento es comúnmente conocido en el mundo del radiocontrol como BEC (*Battery Eliminator Circuit*) ya que evita la necesidad de añadir una batería adicional, en este caso de 5V. El conversor (Figura 2.11) reduce la tensión de la batería a 5V para alimentar el láser, dando la corriente suficiente a tensión constante para que éste funcione correctamente.

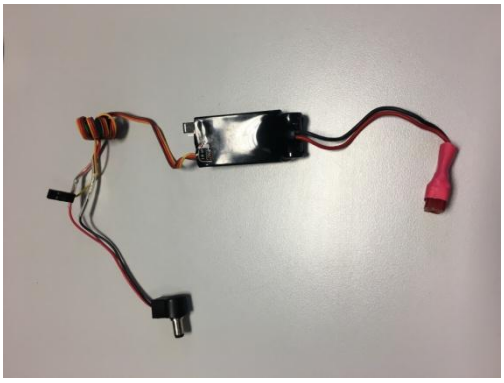


**Figura 2.11**

### 2.1.7. Placa de conexionado

En el drone se han embarcado varios dispositivos que deben alimentarse a distintos niveles de tensión y conectarse entre sí mismos, y de cara a facilitar esa tarea se ha diseñado una placa de conexionado (Figura 2.12). La placa se alimenta con la tensión de 5V proveniente de la salida del BEC. La placa a partir de ese nivel de tensión genera otro nivel de 3.3V para alimentar el modulo XBee, el MAX3222 y el IMU. Además de los pines de la alimentación también se han colocado los pines necesarios para el conexionado de los distintos puertos de los citados dispositivos.

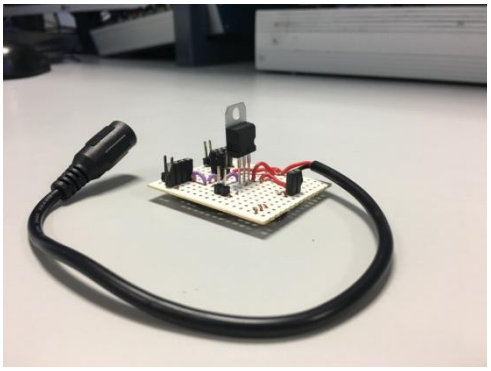


**Figura 2.12**

### 2.1.8. Multiplexor de modo de vuelo

El drone debe poder ser capaz de volar tanto totalmente radiocontrolado desde el mando de control como de volar con el control de altura activado, controlándose el resto de grados de libertad desde el mando. Para ello se utiliza el Pololu 4-Channel RC Servo Multiplexer [ 8 ] (Figura 2.13). A dicho multiplexor se conectan la entrada de throttle del autopiloto, el canal de throttle del receptor radio y la acción de control de throttle del microcontrolador. La selección de la entrada de throttle que se conecta al autopiloto se realiza mediante la entrada SEL del multiplexor. A dicha entrada se conecta el quinto canal del receptor radio, de tal manera que se puede controlar la señal multiplexada desde un interruptor del mando de control.

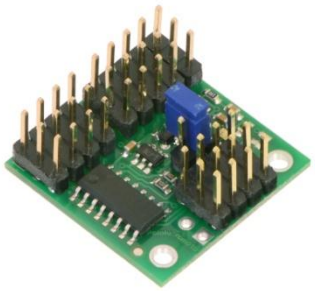


**Figura 2.13**

Además, el quinto canal de la radio también está conectado a la entrada de modo de control del autopiloto. De esta forma, cuando la consigna de throttle que llega al drone es la del propio mando, el autopiloto está en modo Manual, controlándose en velocidad los ángulos de orientación del drone. En cambio, cuando la consigna que llega al drone es la proveniente del control de altura del microcontrolador, el modo de vuelo cambia a Attitude Mode, modo en el que se controlan los ángulos de la orientación en posición.

## 2.2. Software

En esta sección se explica el software utilizado en el desarrollo del proyecto.

### 2.2.1. Programación del microcontrolador

Para programar el microcontrolador se ha hecho uso del entorno de programación de Texas Instruments así como de su sistema operativo de tiempo real. Se ha utilizado la última versión que había disponible en la fecha de comienzo del proyecto de ambos productos.

#### 2.2.1.1. Code Composer Studio (CCS)

Es el entorno de programación que se ha utilizado para programar el microcontrolador. El programa consta de dos modos principales: el modo edición de código y el modo *debugging* para depurar el programa realizado. Es un entorno de programación realmente potente. La versión utilizada es la v7.1.

#### 2.2.1.2. SYS/BIOS

SYS/BIOS [ 4 ] es el sistema operativo de tiempo real, diseñado por Texas Instruments, sobre el que se ha implementado el programa realizado en el proyecto. Ofrece una serie de primitivas ya programadas que permiten al usuario trabajar de forma más sencilla con una aplicación de tiempo real. El protocolo de acceso implementado por SYS/BIOS es el de herencia de prioridad. La versión que se ha utilizado es la v6.46.

### 2.2.2. DIGI XCTU

El XCTU de la empresa DIGI es el programa utilizado para la comunicación entre los módulos XBee. Además de permitir configurar los XBee, el programa tiene una consola serie desde la que se pueden enviar y recibir mensajes. Dicho registro de mensajes se puede posteriormente almacenar para el procesamiento de los datos obtenidos. También existe la posibilidad de crear un conjunto de mensajes y guardarlos con formato xml para cargarlos en otra sesión y no tener que crearlos cada vez que se vaya a ejecutar la aplicación.

### 2.2.3. URG BENRI Standard

Este programa permite conectar el escáner láser al ordenador y visualizar de forma tanto gráfica como numérica las medidas que el escáner está tomando. Además permite abrir un terminal serie entre el ordenador y el láser para poder configurarlo.

### 2.2.4. DJI NAZAM Lite Assistant 1.00

Permite conectar el autopiloto al ordenador para ver el estado en el que se encuentra el equipo y comprobar que todo funciona correctamente. Además, desde el programa [ 7 ] se pueden configurar diversos parámetros del drone.

### 2.2.5. Matlab

Es un programa matemático que ofrece un entorno de desarrollo integrado. Se ha hecho uso de su herramienta Simulink para simular los comportamientos de los distintos reguladores.

# Capítulo 3

## Arquitectura del sistema

En el presente capítulo se explica la interacción y el conexionado entre los distintos elementos que constituyen la aplicación. En la Figura 3.1 se muestra un diagrama de bloques del sistema.

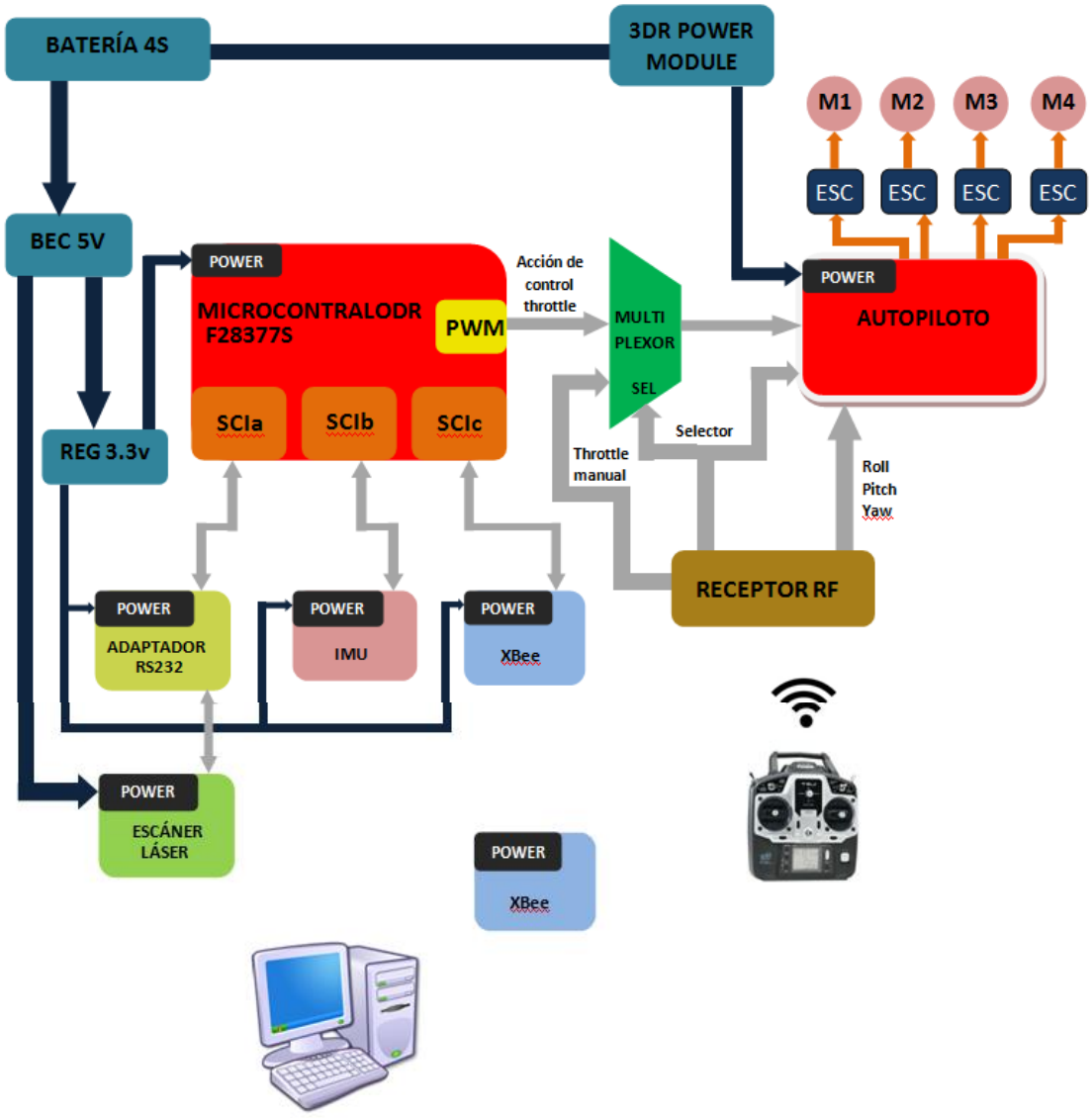



**Figura 3.1**

## 3.1. Alimentación

La fuente de alimentación principal del drone es una batería LiPo de 4 celdas, que a plena carga se traduce en una tensión de 16.8V. Según los fabricantes de los motores y los ESCs (controladores de velocidad) tanto unos como otros pueden trabajar perfectamente a esa tensión. Sin embargo, es necesario obtener tensiones de alimentación de 5V y de 3.3V para alimentar los diversos dispositivos electrónicos.

Conectando el BEC a la batería se obtiene una tensión fija de 5V. La salida del BEC y el terminal neutro de la batería se llevan a la placa de conexionado. Estos dos terminales están conectados con los pines de alimentación del láser y con los terminales de 5V y GND del regulador de tensión de 3.3V que se ha soldado a la placa. Finalmente, los terminales neutro y de salida del regulador de tensión se llevan a los distintos pines de alimentación que se han distribuido por la placa para alimentar el microcontrolador, el módulo XBee y el módulo del MAX3222. El IMU, que también se alimenta a 3.3V, se conecta directamente a dos de los terminales de 3.3V y GND de la placa del microcontrolador ya que la corriente que consume es baja.

## 3.2. Conexión de los dispositivos

En este apartado se dividen los distintos elementos que forman el sistema según el módulo del microcontrolador con el que interactúan. Se entrará en detalle en la programación y los aspectos más relevantes de cada módulo en el *30.*

### 3.2.1. Láser, IMU y XBee

Estos tres elementos se comunican con el microcontrolador por SCI. El microcontrolador dispone de 3 canales SCI distintos, por lo que ha habido que utilizarlos todos. La velocidad de comunicación en cada canal es configurable por separado, sin embargo la frecuencia de reloj (la señal de reloj de baja frecuencia del microcontrolador) que llega a los tres canales es la misma, por lo que se debe fijar una frecuencia de reloj que permita establecer simultáneamente las velocidades de comunicación de los tres periféricos.

El escáner láser está conectado (mediante el adaptador de tensiones para comunicación RS232) al módulo SCIa con una velocidad inicial de 115.2kbps por defecto, que una vez establecida la comunicación entre el microcontrolador y el láser se incrementará hasta los 250kbps. El IMU está conectado al módulo SCIb con una velocidad 57600bps, recibiéndose una trama que

contiene las aceleraciones en los tres ejes y los ángulos girados también respecto a los tres ejes. El IMU está programado para enviar una medida de las 6 magnitudes cada 25ms. Por último, el XBee está conectado al módulo SCIc con una velocidad de 9600bps.

### 3.2.2. Autopiloto

El microcontrolador da la consigna de throttle al autopiloto a través del canal EPWM2 del bloque PWM. La señal debe tener un ancho de pulso limitado entre 1ms y 2ms, con un periodo de 13.60ms para de esta manera emular un canal del receptor de radiofrecuencia. Esta consigna es llevada al canal 2 del lado de los esclavos del multiplexor Pololu y de allí al autopiloto cuando la entrada SEL lo determina. Además de la señal PWM, es necesario llevar al multiplexor (y por ende al autopiloto) el nivel de alimentación y la masa del canal PWM, por lo que también se han llevado al multiplexor dos cables adicionales: uno proveniente de uno de los pines de 3.3V de la placa del microcontrolador y otro de uno de los pines de GND.

# Capítulo 4

# Ley de control

En este capítulo se expone el proceso seguido a la hora de diseñar e implementar los controladores de altura que se han probado. Se expondrán las conclusiones extraídas de las simulaciones previas y los esquemas de control que se han implementado.

## 4.1. Simulación

Se parte de un simulador de control para drone [ 1 ]. Aunque el simulador no se va a utilizar para obtener los valores numéricos de los parámetros de control dado que los propios módulos implicados en la cadena de control tienen sus propias ganancias (por ejemplo la conversión de la acción de control a una señal PWM), se van a realizar ciertas simulaciones para ver cómo influye la variación de las distintas ganancias.

En un principio se ha planteado el uso de un controlador proporcional derivativo (PD) con prealimentación de la perturbación gravitatoria ya que éste es uno de los controladores más utilizados para este tipo de aplicación, y una muestra de ello es que es el regulador implementado en el simulador del que se dispone. La parte proporcional acercará al drone a la altura de consigna y la parte derivativa debe oponerse a las variaciones de altura para que el drone se mantenga estable en régimen permanente. Se realimenta la altura a la que se encuentra el drone. En la Figura 4.1se muestra el modelo en Simulink del controlador. Además del control de altura hay otros controladores implementados en el simulador para los demás grados de libertad del drone. Los parámetros de esos controladores se dejan fijos y únicamente se cambian los parámetros del controlador de altura (líneas rojas).

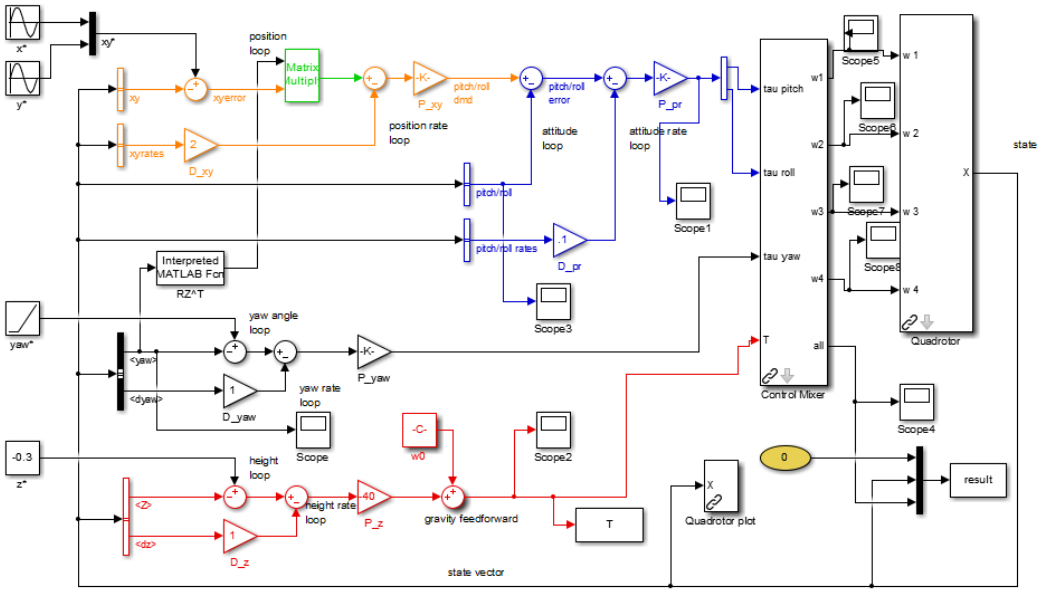



**Figura 4.1**

A continuación se exponen los resultados de algunas de las simulaciones realizadas con los parámetros $k_p$ y $k_d$ seleccionados en cada una de ellas. En las simulaciones se le ha dado al sistema una consigna de altura de 1m.

Inicialmente se ha anulado la ganancia derivativa para estudiar el comportamiento de la ganancia proporcional. Como se observa en la Figura 4.2 para un valor de $k_p = 20$ (que produce un tiempo de respuesta relativamente rápido), el drone no alcanza un régimen permanente y oscila continuamente.

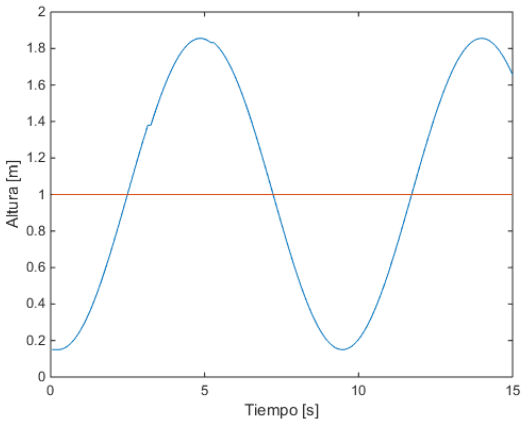



**Figura 4.2**

Al añadir una $k_d = 10$, del orden de magnitud de la $k_p$, se consigue alcanzar el régimen permanente aunque hay oscilaciones, tal y como se muestra en la Figura 4.3, con lo que se constata la necesidad de aplicar una acción derivativa.

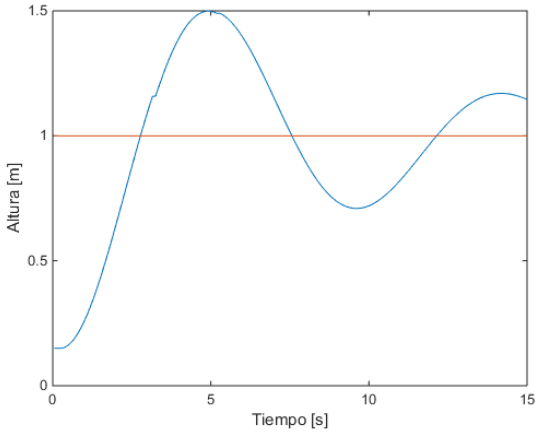



**Figura 4.3**

Se han ajustado ambos parámetros a través de distintas simulaciones ($k_p = 150$ y $k_d = 140$) hasta obtener una respuesta sin sobrepasamiento y con error de posición nulo (Figura 4.4).

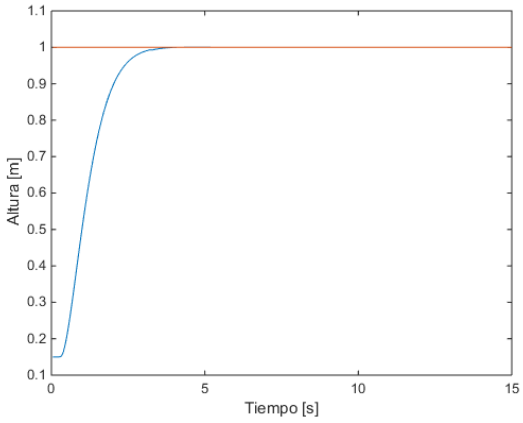



**Figura 4.4**

Las conclusiones extraídas de las distintas simulaciones sobre el efecto de cambiar los valores de $k_p$ y $k_d$ se basan en que para eliminar las oscilaciones, los valores de ambas ganancias deben ser parecidos, una $k_d$ demasiado grande aumenta considerablemente el tiempo de respuesta y una demasiado baja hace que el sistema oscile. Cuanto más se aumenta la $k_p$, menor es el tiempo de respuesta.

## 4.2. Implementación de los controladores

Se ha medido con el osciloscopio que la consigna de *throttle* que el microcontrolador da al drone es un pulso de una anchura entre 1ms y 2ms y con un periodo de 13.60ms, y hay que diseñar el algoritmo que traduzca la consigna generada por el propio controlador en una señal de ese tipo.

El controlador que se ha diseñado trabaja con la magnitud de la fuerza, aunque en el mundo de los drones, y concretamente de los motores para drone, se suele denominar empuje. La consigna enviada al drone es una imagen de la fuerza que se le está demandando a los motores que produzcan. Siendo $C_t$ el coeficiente de sustentación de las hélices, $\rho$ la densidad del aire y $R$ el radio de cada hélice, el empuje ($T$) y la velocidad angular del motor están relacionados a través de la siguiente ecuación [ 2 ]:

$$\omega_{motor} = \sqrt{\frac{T}{4C_t \rho \pi R^4}} \quad (1)$$

A continuación se establece una relación aproximada entre el empuje y el valor del registro de comparación del módulo PWM que determina la anchura del pulso generado. Conocer esta relación es fundamental para tener una idea de en qué rango se van a mover los valores de los distintos parámetros de control.

Los motores montados en el drone tienen una relación velocidad/tensión de $920\frac{rpm}{V}$. Partiendo de la base de que la batería tiene 4 celdas con una carga máxima de 4.2V cada una, la tensión de la batería cargada es:

$$V_{bat} = 4 \cdot 4.2 = 16.8V \quad (2)$$

Por tanto, la máxima velocidad a la que puede girar un motor es:

$$\omega_{máx} = 920 \cdot 16.8 = 15456rpm \quad (3)$$

Si se tuviese conocimiento del coeficiente de sustentación de la hélice, sabiendo que el radio es de 12.7cm se podría calcular el empuje máximo que puede generar cada motor. Este dato se desconoce, sin embargo, en la hoja de características del motor aparece el dato del empuje máximo de cada motor, que es de 1.2kgf, equivalente a 11.76N por motor. Teniendo en cuenta que ese es el empuje máximo, conociendo los valores a escribir en el registro de comparación del módulo PWM para obtener pulsos de $T_{min} = 1ms$ y $T_{máx} = 2ms$ (calculados en el apartado *5.1.1.1. Configuración hardware*) y sabiendo, porque se ha medido con el osciloscopio, que a

menor anchura de pulso más tensión se le aplica al motor, se plantea el siguiente sistema de ecuaciones:

$$\begin{cases} PWM_{máx} = T_{min} \cdot m + n \\ PWM_{min} = T_{máx} \cdot m + n \end{cases} ; \qquad \begin{cases} 4166 = 0 \cdot m + n \\ 2082 = 11.76 \cdot m + n \end{cases} \quad (4)$$

Resolviendo el sistema de ecuaciones se halla el valor a escribir en el registro de comparación del bloque PWM en función del empuje $T$ deseado:

$$PWM = -177.21\, T + 4166 \quad (5)$$

### 4.2.1. Control PD

El diagrama de bloques del controlador PD con prealimentación de consigna y saturación implementado sobre el drone se muestra en la Figura 4.5.

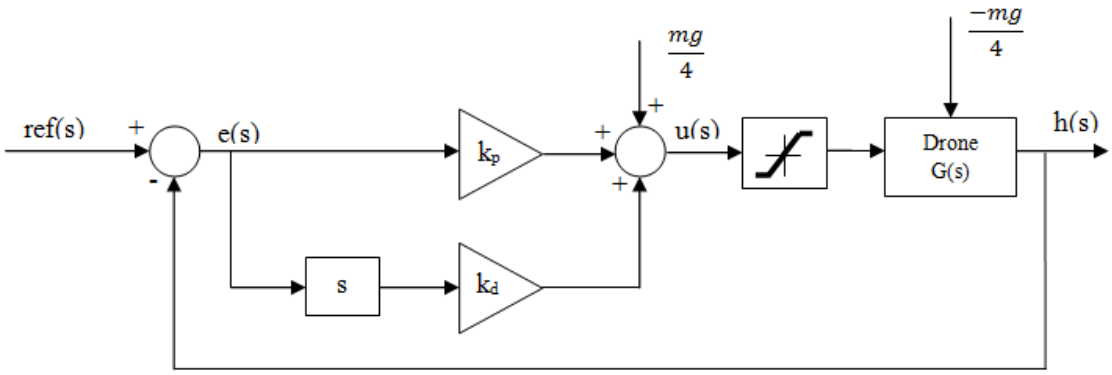



**Figura 4.5**

El empuje o fuerza ascensional $T$ (en Newtons), de acuerdo a este controlador, se calcula de la siguiente manera:

$$T = k_p \cdot e + k_d \cdot e_d + \frac{mg}{4} \quad (6)$$

La prealimentación gravitatoria se divide entre 4 ya que el empuje calculado es el de un motor por separado, que es el que debe generar cada uno.

Los parámetros de control que hay que fijar son $k_p$, $k_d$, y la prealimentación de la gravedad. Dado que no se conoce la función de transferencia del sistema, no se puede calcular un regulador estableciendo requisitos de tiempo de respuesta y sobreoscilación por los métodos clásicos, por lo que se han realizado unas estimaciones numéricas para obtener unos valores

orientativos de los distintos parámetros de control y posteriormente se han ajustado experimentalmente de acuerdo con el comportamiento observado en las simulaciones.

La prealimentación gravitatoria se podría haber calculado pesando el drone, sin embargo se ha realizado un experimento más exacto en el que se ha ido aplicando como acción un pulso PWM cada vez menor (se recuerda que a menor anchura de pulso mayor es el empuje solicitado) hasta que el drone ha estado a punto de elevarse, y ese valor PWM se ha traducido a empuje sustituyendo en la ecuación ( 5 ) y así se obtiene el valor en Newtons de la prealimentación gravitatoria. Se ha buscado que la prealimentación gravitatoria no sea suficiente para elevar el drone, simplemente la necesaria como para que un poco más de acción ya haga ascender al drone.

En cuanto a los parámetros $k_p$ y $k_d$, la estimación se ha basado en buscar que se produzca una acción de control razonable ante determinados valores de error. Por ejemplo, es razonable pensar que el máximo error diferencial (teniendo en cuenta que se obtiene una medida de altura cada poco más de 100ms) que se puede tener es de 5cm. Por otro lado y bajo esa premisa, también es razonable imponer que para un error de 1m la acción sature. El valor de la prealimentación gravitatoria, es de unos 6.6N. Unificando todas estas condiciones y sustituyendo en la ecuación ( 6 ) se pueden obtener unos valores base de las ganancias:

$$11.76 = 1 \cdot k_p + 0.05 \cdot k_d + 6.6; \qquad k_p + 0.05 \cdot k_d = 5.16 \tag{7}$$

Buscando darle más peso a la parte proporcional que a la derivativa se impone $k_d = 10\frac{N}{m}$ para que tan sólo aporte 0.5N y $k_p$toma por tanto un valor de $k_p = 4.66\frac{N}{m}$. Estos valores se tomarán como punto de parida para las pruebas con el drone.

### 4.2.2. Controlador PID

Como se expone en el *Anexo I: Proceso experimental*, el control PD no ha funcionado como se esperaba. Las oscilaciones eran muy exageradas pero el mayor problema se encontraba a la hora de llegar al régimen permanente. Por estos motivos se ha diseñado un controlador PID, que debería facilitar alcanzar el régimen permanente. En la Figura 4.6 se muestra el esquema de dicho controlador.

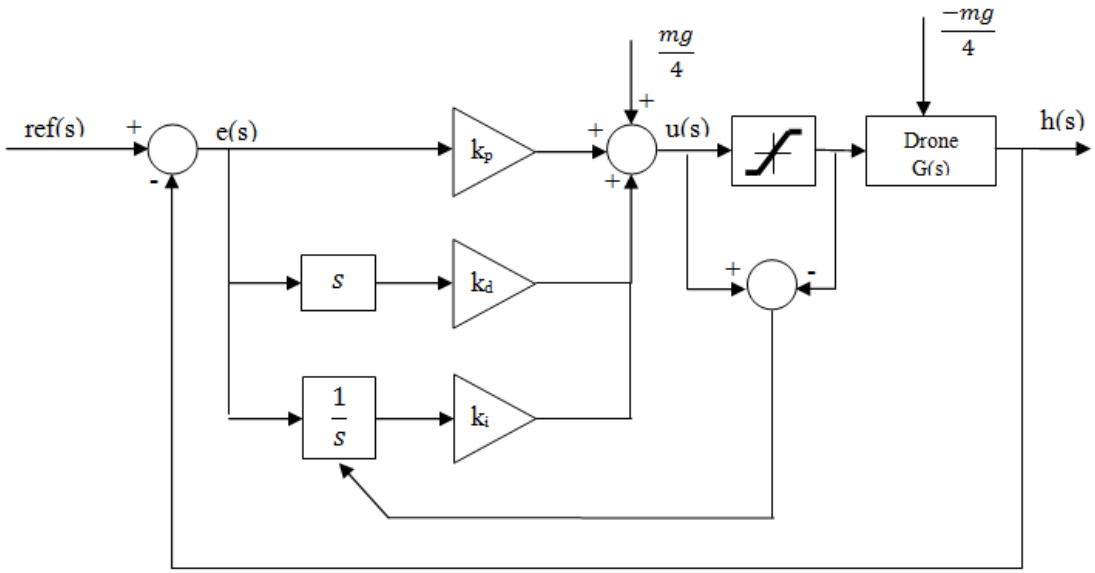



**Figura 4.6**

Todo lo expuesto en relación al controlador PD es igualmente válido, sin más que la adición de la ganancia integral, con lo que ahora la acción de control se calcula:

$$T = k_p \cdot e + k_d \cdot e_d + k_i \cdot e_i + \frac{mg}{4} \quad (8)$$

Como parámetros iniciales se han tomado los mismos que en el control PD, y para la ganancia proporcional $k_i$se ha tomado un valor bajo en relación a las otras ganancias para comenzar con el ajuste experimental, de tal manera que esta ganancia no haga que el sistema se inestabilice.

Se ha implementado un sistema de saturación *anti-windup* para que el regulador no siga integrando el error una vez la acción está saturada tanto superior como inferiormente. Si la acción está saturada se deshabilita la integración, manteniéndose la última acción integral aplicada antes de que se diese la saturación. Añadir un sistema de saturación integral es imprescindible. Si no se añadiese, el error integral se seguiría acumulando y se generarían unas oscilaciones cada vez de mayor amplitud que harían que el sistema se descontrolase.

### 4.2.3. Saturación

En ambos reguladores se ha incluido un bloque de saturación como se muestra en sus correspondientes diagramas de bloques. En este sistema el bloque de saturación es de vital importancia. La acción que se puede aplicar al drone es un pulso entre 1ms y 2ms, pero experimentalmente se ha comprobado que es necesario saturar mucho la acción porque si no el drone se vuelve inestable aunque las ganancias sean bajas. En realidad este comportamiento es normal, porque es como si se intentase estabilizar manualmente el drone a una altura concreta

moviendo el *stick* del acelerador a lo largo de todo el recorrido que permite el mando, cuando realmente lo que se hace es mover el *stick* ligeramente en torno a un punto de equilibrio.

El sistema de saturación que se ha propuesto pretende emular ese comportamiento y es relativo a la magnitud de la prealimentación de consigna. Los valores de saturación superior e inferior de la acción de control vienen dados por:

$$U_{máx} = \frac{mg}{4} + margen_{ascensión} \quad \textbf{(9)}$$

$$U_{min} = \frac{mg}{4} - margen_{descenso} \quad \textbf{(10)}$$

Estos márgenes se han estimado experimentalmente y los valores que toman son $margen_{ascensión} = 1N$ y $margen_{descenso} = 0N$, es decir, la acción nunca puede ser menor que la prealimentación gravitatoria. Se recuerda que la prealimentación gravitatoria es la justa para que el drone casi empiece a sustentarse, pero sin llegar a hacerlo. De esta forma el comportamiento del drone es mucho más suave. Los resultados han mejorado mucho con este esquema de saturación obteniéndose respuestas con muchas menos oscilaciones.

# Capítulo 5

# Implementación

En este capítulo se presentan las soluciones implementadas para la superación de los distintos frentes que se han ido presentando a lo largo de la elaboración del proyecto. La implementación del control de altura del *quadrotor* tiene una carga elevada a nivel software como cabría esperar. Sin embargo el trabajo a realizar a nivel hardware y de ajustes del drone ha sido también muy tedioso e igual de importante.

## 5.1. Diseño software de la aplicación

El control de altura del drone conlleva la utilización de varios elementos hardware distintos, los cuales se han presentado de forma breve anteriormente. El programa debe acceder a todos esos elementos tanto para leer la información recibida como para darles órdenes. Además, la información de los distintos periféricos debe combinarse en el propio programa con distintos fines como se explicará a continuación. Adicionalmente, se requiere un control riguroso del tiempo en el que se ejecutan ciertas secciones del código así como la periodicidad con la que se ejecutan, como es el caso del claro ejemplo del bucle de control. Por estos motivos se ha implementado el programa sobre un sistema operativo de tiempo real para microcontroladores, SYS/BIOS.

El programa, por tanto, se ha estructurado en una serie de tareas que ejecutan funciones aisladas y que se comunican entre ellas mediante los diversos servidores que se han diseñado, los cuales permiten el acceso seguro a los distintos recursos compartidos, como por ejemplo los datos obtenidos por los sensores.

A continuación se van a analizar los distintos aspectos de la implementación del programa, comenzando por la configuración hardware y software del microcontrolador, continuando con las funciones implementadas en las distintas tareas y con los servidores programados.

### 5.1.1. Configuración del microcontrolador

#### 5.1.1.1. Configuración hardware

En primer lugar se han configurado los distintos puertos del microcontrolador de acuerdo con el hardware que debe conectarse al mismo.

La frecuencia principal de reloj del microcontrolador se ha configurado en 200MHz, que es la máxima alcanzable. Se ha elegido esta frecuencia porque existirán porciones de código importantes que deberán procesarse lo más rápido posible y el consumo del microcontrolador no es un problema en la aplicación, ya que comparado con el de los motores es prácticamente inexistente. La fuente de reloj elegida ha sido la del oscilador de la placa en la que va embarcado el microcontrolador y el reloj interno del microcontrolador ya que la fuente externa tiene una precisión superior (de 30 ppm).

El bloque PWM debe generar una señal de frecuencia $f_{throttle} = 73.53Hz$, lo cual equivale a un periodo de 13.60ms, con un tiempo en alto variable entre 1ms y 2 ms. Arbitrariamente se elige una precisión temporal muy elevada en el PWM, de 0.5µs (equivalente a una frecuencia de $f_{tick} = 2MHz$), para poder dar una consigna de calidad al autopiloto. Las distintas ecuaciones aplicadas respecto a la configuración de los módulos del microcontrolador se han extraído del manual de usuario [ 5 ]. En dicho manual también se puede ver el uso de cada registro utilizado en las fórmulas. El valor real de la frecuencia de tick se calcula de acuerdo a:

$$f_{tick} = \frac{EPWMCLK}{HSPCLKDIV \cdot CLKDIV} = \frac{\frac{f_{CPU}}{2}}{HSPCLKDIV \cdot CLKDIV} = \frac{100MHz}{HSPCLKDIV \cdot CLKDIV} \quad \textbf{(11)}$$

De tal forma que siendo la frecuencia de tick del PWM de $f_{tick} = 2MHz$:

$$HSPCLKDIV \cdot CLKDIV = \frac{100}{2} = 50MHz \quad \textbf{(12)}$$

Se toman HSPCLKDIV = 6 y CLKDIV = 8, obteniendo un producto de 48 y por tanto una frecuencia real de tick de:

$$f_{tick} = \frac{EPWMCLK}{HSPCLKDIV \cdot CLKDIV} = \frac{100}{48} = 2.083333MHz \quad \textbf{(13)}$$

Finalmente, los valores del registro de periodo y de los valores máximo y mínimo del registro de comparación se calculan de la siguiente manera:

$$TPRD = tiempo \cdot f_{tick} - 1 = 13.6 \cdot 10^{-3} \cdot 2.083333 \cdot 10^{6} - 1 \approx 28332 \quad (14)$$

$$COMP_{máx} = tiempo \cdot f_{tick} - 1 = 2 \cdot 10^{-3} \cdot 2.083333 \cdot 10^{6} - 1 \approx 4166 \quad (15)$$

$$COMP_{min} = tiempo \cdot f_{tick} - 1 = 1 \cdot 10^{-3} \cdot 2.083333 \cdot 10^{6} - 1 \approx 2082 \quad (16)$$

En el módulo SCI, además de los temas de paridad, *start bit* y número de *stop bits* que se fijan para que se ajusten a lo que aparece en la hoja de características de cada dispositivo, es necesario configurar 5 parámetros: la frecuencia de la señal de reloj que llega al bloque SCI, las velocidades de comunicación del IMU y el XBee y las dos velocidades de comunicación del escáner láser (la de arranque y la de funcionamiento). La velocidad de comunicación se fija mediante el registro BRR de cada canal de acuerdo a la siguiente fórmula:

$$BRR = \frac{LSPCLK}{8 \cdot bitrate} - 1 \quad (17)$$

Interesa que el valor del BRR sea lo más elevado posible para que la velocidad de comunicación se ajuste lo máximo posible a la teórica después de redondear el resultado de la igualdad anterior al entero más cercano. También de acuerdo con la fórmula anterior, a mayor valor de *bitrate* la frecuencia del reloj de baja velocidad, *LSPCLK*, debe ser mayor. Esta frecuencia se obtiene dividiendo con un *prescaler* la frecuencia $f_{CPU}$, y su máximo valor es de 100MHz según la hoja de características del microcontrolador. Dado que la velocidad máxima de comunicación que se va a establecer es de 250kbps, la frecuencia elegida para la señal de reloj que llega al módulo SCI es *LSPCLK* = 100MHz. Los valores a escribir en los registros BRR son:

$$BRR_{Láser115.2k} = \frac{100 \cdot 10^{6}}{8 \cdot 115200} - 1 \approx 107 \quad (18)$$

$$BRR_{Láser250k} = \frac{100 \cdot 10^{6}}{8 \cdot 250000} - 1 = 49 \quad (19)$$

$$BRR_{IMU} = \frac{100 \cdot 10^{6}}{8 \cdot 57600} - 1 \approx 216 \quad (20)$$

$$BRR_{XBee} = \frac{100 \cdot 10^{6}}{8 \cdot 9600} - 1 \approx 1301 \quad (21)$$

#### 5.1.1.2. Configuración software

Desde el entorno CCS, una vez generado el proyecto [ 9 ] con SYS/BIOS, es necesario configurarlo incluyendo los módulos que se vayan a utilizar. En el proyecto se han incluido

varios módulos, entre los que destacan *Clock* (para contabilizar tiempos y poder implementar tareas periódicas), *Semaphore* (para bloquear y desbloquear las tareas tanto de forma periódica como esporádica), *SWI* (para lanzar interrupciones software ante ciertos eventos que determinen la necesidad de ejecutar cierto código de elevada prioridad), *HWI* (para configurar de forma sencilla las distintas fuentes de interrupción hardware), *GateMutexPri* (para implementar servidores que trabajen con bloqueo por herencia de prioridad) y *Boot* (para configurar el reloj de sistema).

Además de incluir y configurar los módulos anteriormente mencionados también se han fijado el tamaño de la pila del sistema, el tamaño de la pila de cada tarea y la *Heap* del sistema, donde se alojan los elementos de SYS/BIOS creados dinámicamente, como las distintas tareas e interrupciones hardware y software.

## 5.2. Diseño del software

En esta sección se va a explicar la forma en que se ha implementado el programa. A continuación se exponen las tareas implementadas junto con la descripción general de los drivers implementados para los distintos periféricos a los que accede cada una de ellas y finalmente los servidores propuestos para el acceso seguro a los recursos compartidos.

### 5.2.1. Tareas

Las tareas programadas, más allá de su contenido, se pueden diferenciar en dos grupos según el tipo de activación asociado a cada una de ellas.

- Tareas periódicas: estas tareas son activadas periódicamente por un objeto *Clock* que al alcanzar el número de conteos establecidos en su creación ejecuta una función que desbloquea el semáforo de la tarea en cuestión, dejándola activa para que el sistema operativo la ejecute cuando sea necesario.

- Tareas esporádicas: estas tareas se activan cuando se produce cierto evento externo que las desbloquea. El motivo de haber elegido esta implementación es que de esta forma se tiene acceso mucho más rápidamente a la información captada que con una implementación periódica ya que el sincronismo es prácticamente perfecto más allá de la posible latencia asociada a que deban ejecutarse tareas de mayor prioridad antes de las esporádicas que nos ocupan. Las tareas que se han implementado como esporádicas son las asociadas a la recepción de información por parte del IMU, el escáner láser y el XBee. Los tres dispositivos coinciden en que se conectan al microcontrolador por

comunicación SCI, y el mecanismo de desbloqueo de la tarea de recepción de datos de cada uno de los tres elementos es el siguiente. Las tramas de caracteres que se reciben de cada dispositivo tienen unas características concretas, pero en los tres casos, a grandes rasgos, hay un carácter esperado de inicio de trama y uno de fin de trama. Cada vez que los sensores envían un carácter al microcontrolador, se activa una interrupción hardware que almacena en un *buffer* dicho carácter si es el carácter que se espera de inicio de trama o si, sin haberse superado la longitud esperada del mensaje, ya se ha recibido anteriormente el carácter de inicio de trama. En el momento en que se recibe el carácter de fin de trama se genera una interrupción software que por un lado almacena una copia del mensaje recibido en la variable en cuestión que esté dentro del alcance de la tarea que la va a leer y por otro lado desbloquea el semáforo asociado a la tarea que va a procesar la información recibida.

La planificación de tareas implementada se basa en el criterio *Rate Monotonic*, que da mayor prioridad a las tareas más frecuentes, es decir, a las que tienen menor periodo. El análisis de tiempo real del sistema de tareas se encuentra en el *Anexo IV: Análisis de tiempo real*. En la Figura 5.1 se muestra el esquema de tareas y servidores y posteriormente se explica la función que tiene cada tarea.

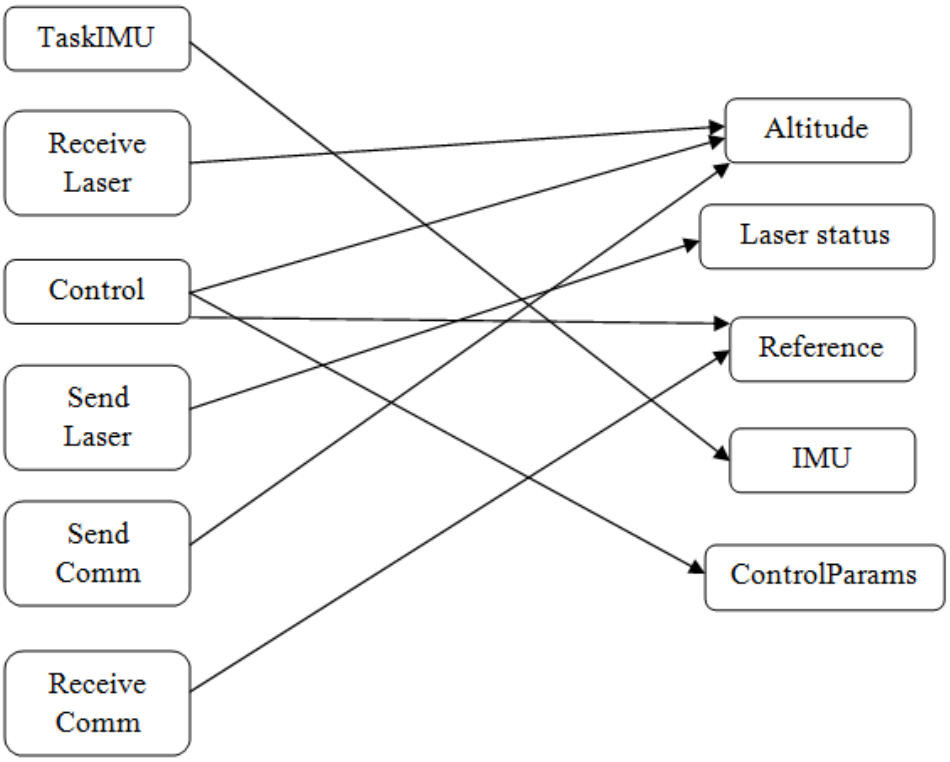



**Figura 5.1**

#### 5.2.1.1. Lectura del IMU

La tarea esporádica *TaskIMU* se activa cuando se ha recibido correctamente un mensaje del IMU y se encarga de extraer la información de los ángulos *roll*, *pitch* y *yaw* del mensaje en cuestión. El ángulo *yaw* no se utiliza en la aplicación, pero se ha creído conveniente almacenarlo por si en un futuro pudiese ser de utilidad.

El mensaje está estructurado de tal manera que primero se envían las aceleraciones y posteriormente los ángulos. Puesto que no es necesario tener medida de las aceleraciones, el mensaje se empieza a almacenar a partir de la llegada del carácter 'Y' (de *yaw*), y la recepción sigue hasta la llegada del carácter '\n' (retorno de carro), que indica que la trama ha concluido. Los ángulos vienen dados en grados, pero de cara a los cálculos que hay que hacer con ellos, la propia función de decodificación del mensaje los devuelve en radianes.

La tarea, a partir de las medidas obtenidas en las primeras 20 ejecuciones, calcula el offset medio de cada ángulo y lo almacena. Una vez superadas esas primeras ejecuciones, la tarea lee el mensaje, extrae los valores de los tres ángulos, les resta el correspondiente offset, aplica un filtro de mediana móvil a las últimas 5 medidas tomadas y posteriormente almacena dichos valores en el servidor *IMU_server*. El filtro es de mediana para eliminar los valores espurios que de vez en cuando da el sensor y se ha realizado con 5 medidas porque el bucle de control se ejecuta cada 108ms.

#### 5.2.1.2. *Tarea de control*

La tarea *TaskControl* se encarga del control de altura del drone. El periodo de control es de 108ms ya que por motivos inherentes al escáner láser no se ha podido reducir más, como se explica en la siguiente tarea. La tarea lee la altura a la que se encuentra el drone del servidor *Altitude_server* y la consigna de altura a alcanzar del servidor *Reference_server*. A partir de esas variables aplica la ley de control, que calcula y escribe el valor necesario en el registro de comparación del módulo PWM.

Los parámetros de control (ganancias, prealimentación y saturaciones) se pueden modificar desde el ordenador enviando comandos al microcontrolador mediante el XBee.

#### 5.2.1.3 Envío al láser

*La* tarea *TaskSendLaser* se encarga de enviar al láser los comandos necesarios para configurarlo al inicio de la aplicación y el comando que ordena al láser realizar una medida. En las siguientes

líneas y junto con la próxima tarea, *TaskReceiveLaser*, se va a explicar la gestión software del escáner láser que se ha implementado.

El diseño del driver que se ha hecho desde cero para el láser ha sido realmente complejo dado, entre muchas otras cosas, que se pueden enviar varios tipos de mensajes diferentes de longitudes distintas entre sí. La estructura comentada anteriormente para la recepción de caracteres en las interrupciones hardware de los dispositivos conectados a los canales SCI del microcontrolador, en este caso es más compleja dado que tanto el carácter de inicio de trama como la longitud esperada de la respuesta son variables. Para superar esta dificultad se ha creado una estructura de datos en la que se almacenan el tipo de comando, la cabecera, el carácter de inicio esperado, la longitud de la respuesta esperada y la longitud de la cabecera del último mensaje que se ha enviado al láser. Asimismo, se han creado funciones específicas de envío de cada tipo de mensaje que, además de enviar el mensaje en cuestión, cargan en la estructura de datos comentada anteriormente todos los parámetros propios del mensaje recientemente enviado. De esta forma queda encapsulado el problema de la variabilidad de las características estructurales y de contenido de los distintos mensajes a enviar al láser.

Los mensajes que en la aplicación se envían al láser son los siguientes: parada del láser en protocolo SCIP1.1, cambio de protocolo a SCIP2.0, cambio de *bitrate*, parada del láser en protocolo SCIP2.0 y solicitud de obtener medidas continuamente. Existe la posibilidad de solicitar medidas individuales, pero se ha decidido que las medidas lleguen continuamente ya que de esta forma el tiempo de transmisión es menor y por tanto se obtienen las medidas más rápidamente puesto que el láser trabaja de forma que cada vez que éste recibe un mensaje, envía una respuesta de conformidad o disconformidad al maestro.

En la Figura 5.2 se muestra la máquina de estados que controla el láser. El láser al encenderse se encuentra en el protocolo de comunicación SCIP1.1 el cual ofrece poca flexibilidad de manejo, por lo que se ha decidido cambiar el láser al modo SCIP2.0, protocolo que lanzó posteriormente Hokuyo y que ofrece muchas opciones de configuración. La velocidad de comunicación por defecto del láser es de 115200bps y se va aumentar en tiempo de ejecución hasta los 250000bps para obtener más rápidamente las medidas. En primer lugar es necesario destacar que se ha observado experimentalmente que el sistema del láser junto con el adaptador RS232 necesita un tiempo de espera desde la conexión de la batería hasta que se pueda establecer la comunicación, por lo que antes de comenzar la ejecución de las tareas el programa tiene un tiempo de espera de 10 segundos. Una vez transcurrido ese tiempo y ya iniciado el sistema operativo, la máquina de estados configura el láser secuencialmente y finalmente le envía el comando de solicitud de medida continua. Esta tarea está ligada muy estrechamente con la tarea *TaskReceiveLaser*,

puesto que para determinar si se envía el siguiente mensaje es necesario tener la certeza de haber recibido la respuesta esperada del láser, de tal forma que se le envía el mismo comando hasta que se recibe por parte del láser la confirmación de que el mensaje lo ha recibido bien. Se ha creado un tipo de dato, *LASER_STATUS*, que tiene la doble misión de guiar la configuración del escáner en la secuencia de arranque y de determinar si el mensaje enviado por el láser se ha recibido correctamente o no.

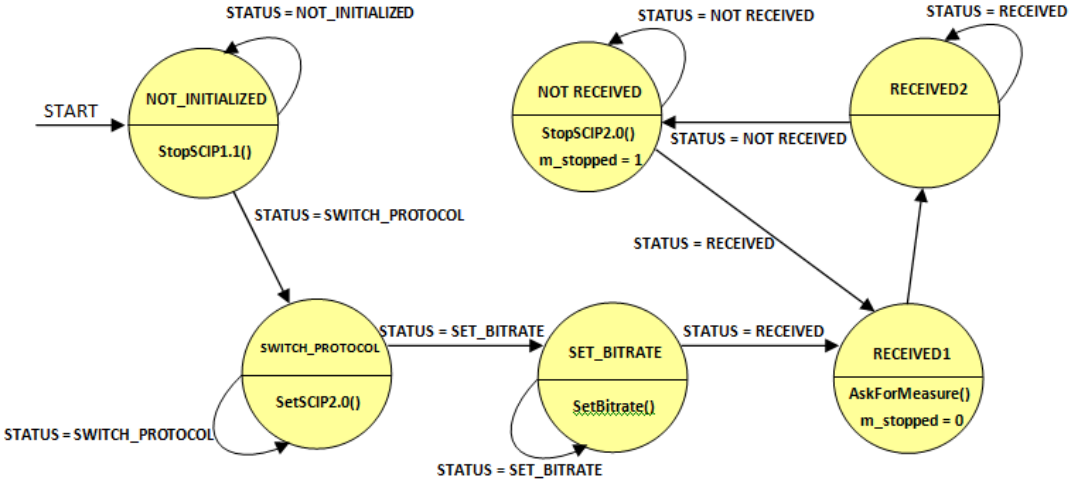



**Figura 5.2**

En la *datasheet* del protocolo del láser [ 6 ] no se explica la secuencia de inicialización, por lo que más allá de leer los tipos de comandos que hay y cómo trabajar con ellos se han realizado varias pruebas hasta llegar a la secuencia que configura el láser correctamente.

En la Figura 5.3 se muestra el esquema de medidas del láser. El escáner tiene un máximo ángulo de barrido de 240º y toma una medida cada 0.36º, lo que hace un total de 682 medidas. A cada medida del láser se le denomina *step*, y cuando se envía al láser el comando de solicitud de medidas, entre otros parámetros se le indica el intervalo de *steps* que se desea recibir, fijando el *step* inicial y el final.

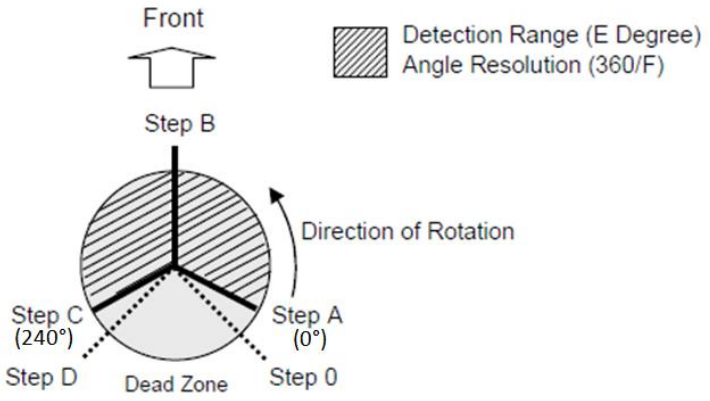



**Figura 5.3**

En la Figura 5.3 se pueden apreciar las referencias de 0° y 240° que se han colocado arbitrariamente en esas posiciones para referenciar los ángulos en el proyecto. Dado que al *Step A* le corresponde según la hoja de características del sensor un valor de 44, la ecuación para hallar la relación ángulo – *step*, redondeando siempre al entero más cercano es:

$$step = 44 + \frac{ángulo}{0.36} \qquad (22)$$

Para medir la altura no es necesario utilizar todas las medidas que puede captar el sensor, sino sólo algunas de ellas. Sin embargo, se ha diseñado un driver flexible y modular que permite variar el rango medido simplemente cambiando una línea de código. Inicialmente se había propuesto un rango de medida de 90º (y tanto el programa como el hardware están diseñados están preparados para ello) con su bisectriz alineada con la línea frontal del láser (*Step B*) para poder obtener dos grupos de medidas determinados por dos haces de rayos de rango angular de 10º cada uno, tal y como se observa en la Figura 5.4.

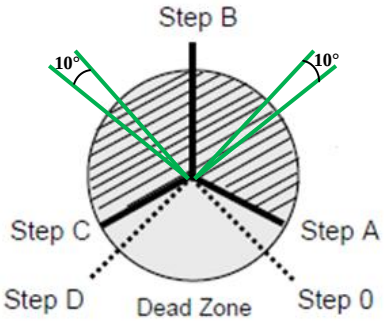



**Figura 5.4**

Sin embargo existe un problema que impide obtener esos dos haces de medidas, y no es otro que el tiempo de medida del láser. El escáner tarda 100ms en realizar un barrido de medidas, independientemente de si se le solicita todo el rango de medidas o sólo un único rayo. Se ha intentado reducir este tiempo de medida aumentando la velocidad del motor del láser pero esto no ha tenido ningún efecto beneficioso. Se ha contactado con Hokuyo preguntando si existe alguna posibilidad de reducir ese tiempo y no la hay. A parte de esos 100ms que cuesta realizar una medida hay que sumar el tiempo que cuesta transmitir los datos obtenidos por RS232. Es necesario tener disponible la medida de la altura del drone en el menor tiempo posible para poder implementar el bucle de control con un periodo lo suficientemente bajo, por lo que se han tomado dos medidas correctoras para disminuir el tiempo de transmisión, que es el único que se puede variar. En primer lugar se ha reducido el ángulo de barrido de los 90º propuestos inicialmente a aproximadamente 12º (Figura 5.5) y por otro lado se ha aumentado la velocidad de comunicación de 115.2kbps a 250kbps (y no se ha aumentado más porque el MAX3222 no permite trabajar a mayores velocidades). Implementadas las dos mejoras, el tiempo que transcurre entre la llegada de dos medidas al láser es mucho menor.

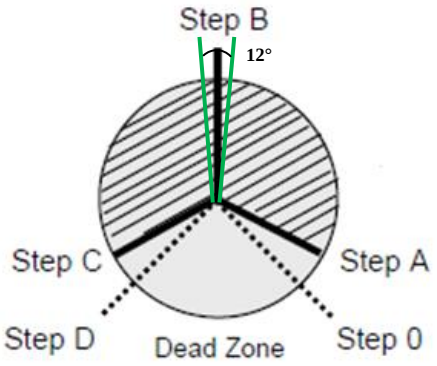



**Figura 5.5**

A continuación se calcula el tiempo que tarda en transmitirse el mensaje que contiene las distancias medidas por los distintos rayos del láser con y sin las reducciones de tiempo implementadas. Para ello, en primer lugar se calcula la longitud en caracteres del mensaje esperado, posteriormente se traduce a número de bits y finalmente se divide entre la frecuencia con la que llegan los bits.

La longitud en bytes del mensaje tiene la siguiente fórmula:

$$longitud = 26 + 2 \cdot num_{medidas} + 2 \cdot \left\lceil \frac{num_{medidas}}{64} \right\rceil + 1 \qquad (23)$$

El número de medidas se calcula como la diferencia entre el *step* final y el inicial:

$$num_{medidas} = step_{fin} - step_{ini} + 1 \quad (24)$$

Dado que el protocolo de comunicación con el láser tiene un *stopbit*, un *startbit* y no tiene *bit* de paridad, el número de bits de un mensaje es:

$$n_{bits} = (8 + 1 + 1) \cdot longitud = 10 \cdot longitud \quad (25)$$

El tiempo que transcurre durante la transmisión del mensaje es:

$$t = \frac{n_{bits}}{bitrate} \quad (26)$$

Por tanto, el tiempo de transmisión de que se planteó en un principio de un barrido de 90° a 115.2kbps es:

$$t_1 = \frac{10 \cdot \left(26 + 2 \cdot (519 - 249 + 1) + 2 \cdot \left\lceil \frac{(519 - 249 + 1)}{64} \right\rceil + 1\right)}{115200} = 50.26ms \quad (27)$$

Sin embargo, obteniendo medidas de un barrido de 12° a 250kbps el tiempo de transmisión se reduce a:

$$t_2 = \frac{10 \cdot \left(26 + 2 \cdot (406 - 372 + 1) + 2 \cdot \left\lceil \frac{(406 - 372 + 1)}{64} \right\rceil + 1\right)}{250000} = 3.96ms \quad (28)$$

En la Figura 5.6 se muestra un cronograma en el que se representan el tiempo que le cuesta al láser tomar la medida (los 100ms que no se pueden variar), el tiempo de transmisión con las características que determinan $t_1$ y el tiempo de transmisión con las características que determinan $t_2$.

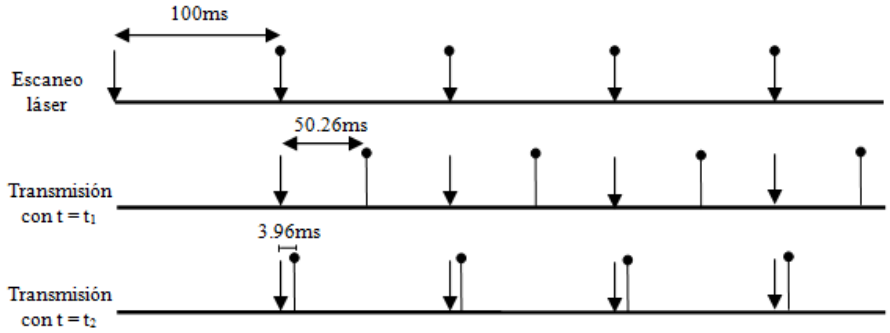



**Figura 5.6**

Se observa que se obtiene una medida cada 100ms pero el retraso con el que se obtiene con $t = t_2$ es mucho menor que con $t = t_1$. Se ha elegido un periodo de control de 108ms, ligeramente superior al tiempo total de obtención de medida de altura, para que en cada ejecución el bucle de control se valga de la nueva medida tomada por el láser.

#### 5.2.1.4. Recepción del láser

La tarea *TaskReceiveLaser* se activa cuando se ha recibido un mensaje completo del escáner láser. La tarea lee el mensaje y lo decodifica, obteniendo el valor de la distancia medida por el sensor así como el estado en el que se encuentra el láser. El estado del láser, como se ha dicho anteriormente, es indicativo de si el mensaje se ha recibido correctamente o no. También se lee el valor de los ángulos roll y pitch. Con estas tres medidas se calcula la altura perpendicular al suelo del drone. Esta altura corregida es la que se almacena en el servidor *Altitude_server*. El estado del láser se escribe en el servidor *LaserStatus_server*.

La función que decodifica los mensajes recibidos también se vale de la estructura de datos comentada anteriormente que almacena la información relevante del último mensaje enviado. La función en primer lugar distingue el tipo de comando enviado y actúa de una forma u otra según el comando que sea. En las recepciones de mensajes de configuración lo que se hace es comparar lo recibido con lo esperado, y si concuerdan ambas cadenas de caracteres entonces la función devuelve el *LASER_STATUS* adecuado para que se pueda enviar el siguiente mensaje de configuración desde la tarea *TaskSendLaser*. En el caso de la medida de altura, además de comprobar si el mensaje recibido es correcto se debe calcular la altura a partir de todos los caracteres recibidos. Para ello, y en busca de un diseño flexible, se van traduciendo los datos recibidos al *step* correspondiente del láser. Si ese dato se encuentra en uno de los grupos de *steps* de los que se quiere obtener medida, éste se almacena en el *buffer* correspondiente a ese grupo y si no se desecha. De esta forma se puede modificar radicalmente el rango que se desea medir del láser simplemente ajustando los *steps* inicial y final de los diferentes grupos en el código fuente. Una vez se llega al final del mensaje se calcula la media de las medidas asociadas a los rayos comprendidos en cada sección de *steps*. La medida implementada finalmente (Figura 5.5) consta de un único grupo de *steps* con un total de 35 medidas.

Una vez obtenida la medida en crudo del láser es necesario corregirla con la orientación para obtener la distancia perpendicular desde el tren de aterrizaje hasta el suelo. Si no, la inclinación del drone falsearía la altura real y el control se haría imposible. La corrección de altura ha sido implementada para la medición del conjunto frontal de 12° de la Figura 5.5. Para ello se analizan por separado el caso de giro respecto al eje x y giro respecto al eje y, representando de

forma esquemática el drone, y posteriormente se combinan ambos efectos. El espejo frontal está situado a lo largo de uno de los ejes de simetría del drone.

En la Figura 5.7 se analiza el efecto que tiene el ángulo roll en la medida. Las líneas negras representan el cuerpo del drone y el tren de aterrizaje en la posición de equilibrio, las líneas rojas representan el drone con un cierto ángulo *roll* no nulo y la línea verde representa el suelo. El espejo está representado en naranja junto al trapecio gris que representa el láser.

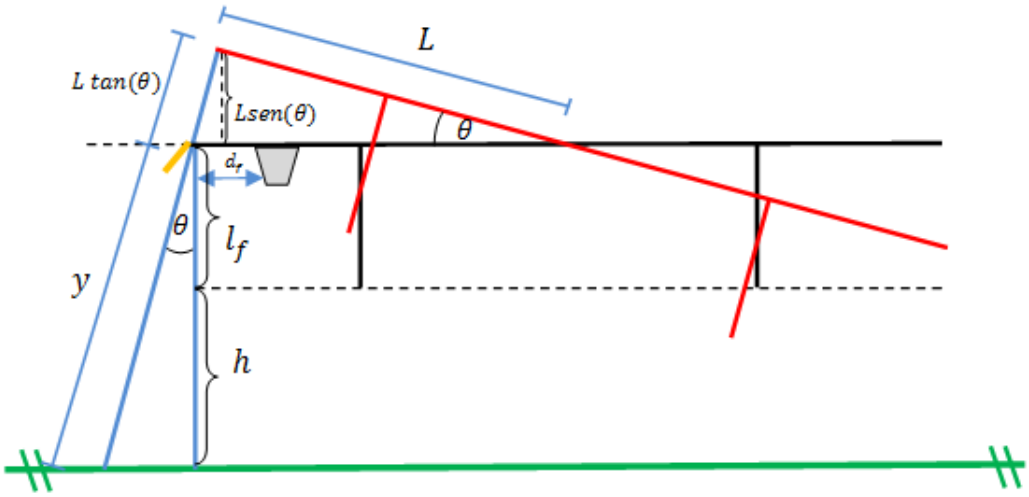



**Figura 5.7**

Sea $y$ la distancia medida por el láser, $l_f$ la distancia perpendicular al suelo entre el punto en el que se reflejan los rayos y la parte más baja del tren de aterrizaje, $L$ la distancia del centro del drone al punto donde se reflejan los rayos, $d_f$ la distancia del origen del láser al espejo y $\theta$ el ángulo *roll*, la altura $h$ a la que se encuentra el drone se calcula de la siguiente forma:

$$h = (y - d_f)\cos(\theta) - L\,sen(\theta) - l_f \qquad (29)$$

En la Figura 5.8 se analiza el efecto que tiene el ángulo pitch en la medida. El código de colores es el mismo que en la anterior. El hecho de que el láser esté situado en el centro en esta proyección simplifica la corrección a realizar.

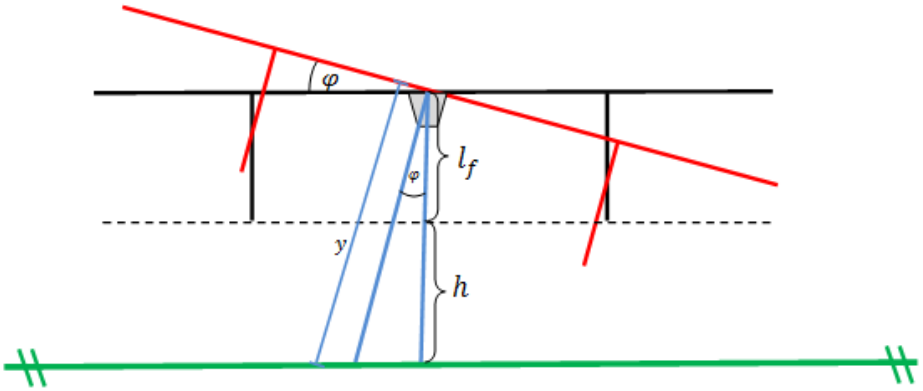



**Figura 5.8**

En este caso, la altura $h$ se calcula de la forma:

$$h = \left(y - d_f\right)\cos(\varphi) - l_f \quad \textbf{(30)}$$

Si se combinan los efectos de ambos ángulos, el valor de la altura corregido es:

$$h = \left(y - d_f\right)\cos(\varphi)\cos\left(\theta\right) - L\,sen(\theta) - l_f \quad \textbf{(31)}$$

#### 5.2.1.5. Envío de datos al ordenador

La tarea periódica *SendComm* se encarga de enviar los datos deseados del drone al ordenador. La información más relevante a enviar es la altura a la que se encuentra el drone, pero también se envían otros datos auxiliares para comprobar que el láser ha iniciado la comunicación con el microcontrolador correctamente. La información es escrita en el XBee y éste se encarga de enviarla secuencialmente.

#### 5.2.1.6. Recepción de datos del ordenador

La tarea *ReceiveComm* se encarga de recibir la información enviada desde el ordenador al drone a través del XBee. Los mensajes que se pueden enviar al drone son la consigna de altura y los valores de los parámetros de control. La estructura de los comandos consta de un carácter de inicio de trama (‘#’), el contenido del mensaje (que tiene una estructura del tipo descriptor del mensaje y a continuación valor numérico) y por último el carácter de fin de trama (‘\n’). Una máquina de estados es la encargada de distinguir entre los distintos mensajes recibidos y posteriormente decodificarlos para almacenar el valor recibido en el servidor que corresponda. En el *Anexo II: Comandos de comunicación* se muestra el listado de comandos que se pueden enviar al drone.

### 5.2.2. Servidores

Se han implementado una serie de servidores para almacenar los recursos compartidos entre las distintas tareas.

- *Altitude_server:* almacena la altura a la que se encuentra el drone.
- *Laser_status_server*: almacena el estado del láser.
- *Reference_server*: almacena la consigna de altura que se le ha dado al drone a través de las comunicaciones inalámbricas.
- *IMU_server*: almacena la orientación medida por el IMU.
- *Kp_server*: almacena la ganancia proporcional, modificable a través del XBee.
- *Kd_server*: almacena la ganancia derivativa, modificable a través del XBee.
- *Ki_server*: almacena la ganancia integral, modificable a través del XBee.
- *Ascension_sat_server*: almacena el margen de acción que se le permite dar al control para que el drone ascienda, modificable a través del XBee.
- *Descension_sat_server*: almacena el margen de acción que se le permite dar al control para que el drone descienda, modificable a través del XBee.
- *Gravity_compensation_server:* almacena la prealimentación que compensa la prealimentación gravitatoria, modificable a través del XBee.

## 5.2. Montaje físico

Una parte fundamental del proyecto ha sido el montaje de todos los componentes en el drone. Esta tarea ha sido realmente complicada dados todos los problemas que han surgido no sólo propios del propio proyecto, sino del drone del que se partía como base.

En primer lugar se va explicar el hardware diseñado propiamente para el proyecto y posteriormente se expondrá la fase de ensamblaje y montaje de los distintos elementos en el drone junto con los problemas que han ido surgiendo y las soluciones que se han adoptado.

### 5.2.1. Soporte para el láser

El láser se ha decidido colocar de tal forma que los rayos vayan paralelos al suelo para, mediante su reflexión al suelo, poder obtener varios grupos de medidas distintas. Si se hubiese colocado con los rayos apuntando al suelo directamente, se podría haber obtenido una única medida perpendicular al suelo, suficiente para la aplicación y más visto lo que sucede con el tiempo de medida del sensor, pero aun así mucho menos flexible de cara futuras mejoras.

Como se muestra en la Figura 5.9, el tren de aterrizaje que se ha montado en el drone tiene las barras superiores más largas de lo que en principio podría necesitar el cuerpo del mismo. Se ha aprovechado esta longitud adicional para colocar el láser. La localización es buena ya que es una zona despejada que permite la ubicación de los espejos.

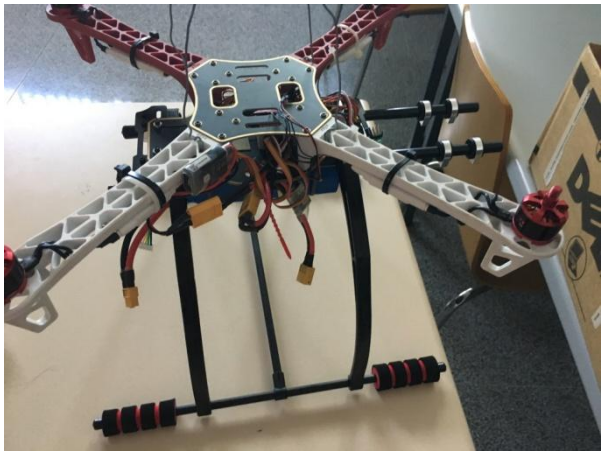


**Figura 5.9**

El láser tiene una serie de tornillos en la base que ensamblan la carcasa. Dos de esos tornillos han sido extraídos y se han utilizado para atornillar el láser al soporte. Junto con el tren de aterrizaje comprado venían dos placas perforadas con diferentes formas de cara a poder colgar del tren de aterrizaje los elementos que se puedan necesitar. En un principio se planteó utilizar una de esas placas (la otra directamente se vio que no sería útil) como soporte para el láser. Sin embargo dada la geometría de los agujeros de dicha placa (Figura 5.10) no era posible atornillar el láser.

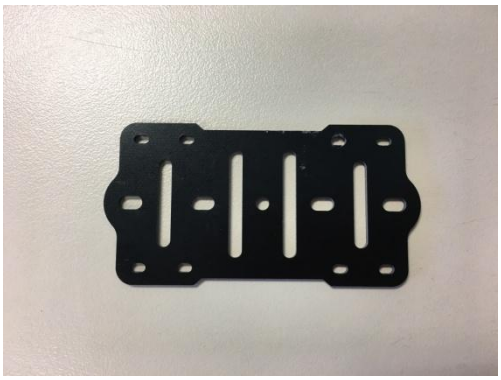


**Figura 5.10**

Se pueden apreciar 8 agujeros situados en la periferia de la placa. Esos agujeros permiten atornillar la placa a las barras del tren de aterrizaje mediante unos elementos de sujeción que forman parte del kit del tren de aterrizaje. Lo que se ha hecho ha sido utilizar dicha placa como plantilla para fabricar un soporte en chapa de aluminio. Se han realizado los 8 agujeros para atornillar el soporte al tren de aterrizaje y además los dos agujeros necesarios para atornillar el láser (se han hecho 4 para elegir el sentido de orientación del láser). Tal y como se muestra en la Figura 5.11, se han añadido 3 capas de cinta de doble cara en la cara del soporte que contacta con la base del láser con una doble misión: dar altura a esa cara del soporte para que el láser tenga más superficie de contacto que los tornillos de sujeción al tren de aterrizaje y reducir ligeramente las posibles vibraciones que se puedan generar.


**Figura 5.11**

Una vez decidida la posición en la que se iba a montar el láser y construido el soporte había que encontrar la forma de reflejar los rayos al suelo. Según la ley de la reflexión [ 11 ], un rayo incidente en una superficie reflectante, será reflejado con un ángulo igual al ángulo de incidencia, ambos ángulos medidos con respecto a la normal a la superficie (Figura 5.12).

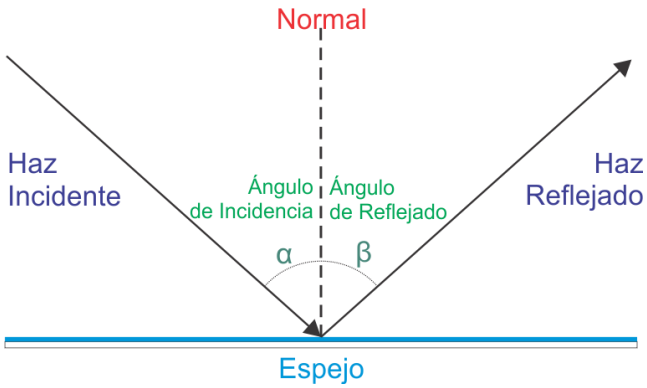



**Figura 5.12**

Al ser los rayos del láser paralelos al suelo y querer redireccionarlos de tal forma que sean perpendiculares al mismo (cuando el drone está posado en el suelo), es necesario colocar el elemento reflectante con un ángulo de 45° respecto al rayo, tal y como se muestra en la Figura 5.13. Como elemento reflectante se ha comprobado que los espejos estándar (Figura 5.14) funcionan correctamente, por lo que no ha sido necesario recurrir a material óptico mucho más caro.

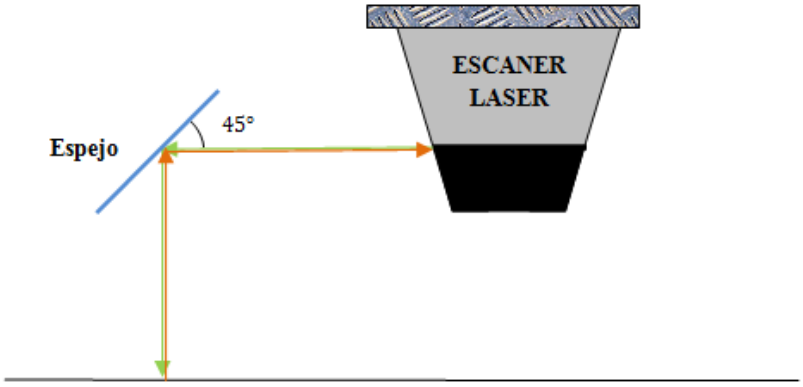



**Figura 5.13**

**Figura 5.14**

La siguiente fase requería de la colocación de los espejos. En principio se iba a realizar el montaje de la Figura 5.4 (la de los 2 espejos). Para ello se pensó en orientar el láser apuntando hacia el centro del drone y situar de alguna forma los espejos en el tren de aterrizaje. En primer lugar hubo que recortar los espejos con un cortavidrios hasta un tamaño más adecuado. Una vez se tuvieron los cristales más pequeños, se sujetaron al tren de aterrizaje mediante cable atado a la estructura tensando hacia lados opuestos para mantener el ángulo firme. Se pegaron con cinta de doble cara a las patas del tren de aterrizaje con una capa de espuma para orientar correctamente los espejos y reducir las vibraciones en la medida de lo posible. El resultado es el montaje de la Figura 5.15.

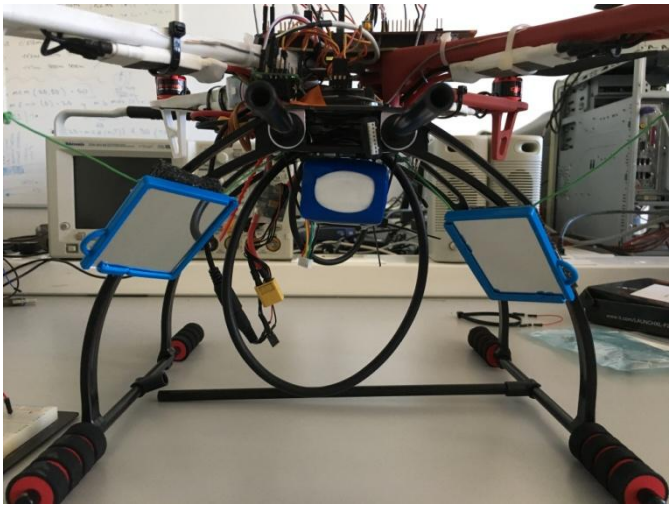


**Figura 5.15**

Sin embargo, después de algunas pruebas de vuelo se observó que los espejos se desorientaban ligeramente ante las turbulencias e impactos de aterrizaje y había que reajustarlos. Por ello se ideó un sistema más robusto (Figura 5.16). Se buscó también una mayor modularidad del bloque láser de forma que se pudiese montar y quitar del drone como un único bloque. Para ello se han atornillado tres brazos de plancha de aluminio al soporte del láser y en cada uno de esos

brazos, que tienen una inclinación de 45°, se ha colocado un espejo. Los espejos (cortados con el cortavidrios) se han pegado a los brazos mediante cinta de doble cara. Entre los brazos y el soporte del láser, además de los tornillos también se ha colocado cinta de doble cara, todo ello para intentar reducir las posibles vibraciones. En este caso se han colocado tres espejos porque en la fase en la que el proyecto se encontraba en ese momento ya se vio que podría haber problemas con la lentitud de medida del láser. La idea inicial fue utilizar los espejos de los extremos pero se colocó un tercer espejo en la parte frontal del láser por si únicamente se podía utilizar uno de ellos y, en cualquier caso, para aportar mayor flexibilidad al sistema. Este soporte se ha comprobado que es resistente a vibraciones y no sufre alteraciones en los impactos por aterrizaje.

**Figura 5.16**

Aunque los brazos se han diseñado para estar centrados en los ángulos de 30°, 120° y 210° que corresponden respectivamente a los *steps* de 127, 377 y 627, ha sido necesario un ajuste experimental para comprobar qué rango abarca cada espejo. Para ello se ha utilizado el programa URG BENRI Standard. Se han observado todas las medidas tomadas por el láser dejándolo a una altura conocida y se han podido observar los *steps* que comprende cada espejo. Para cada espejo se ha dejado un margen de seguridad de unos 5 *steps* por cada lado para que ante movimientos bruscos en vuelo no pueda haber lecturas anómalas. El recorrido que realizan los rayos del láser con el soporte montado es el de la Figura 5.17. Se ha representado únicamente el brazo frontal visto desde uno de los perfiles.

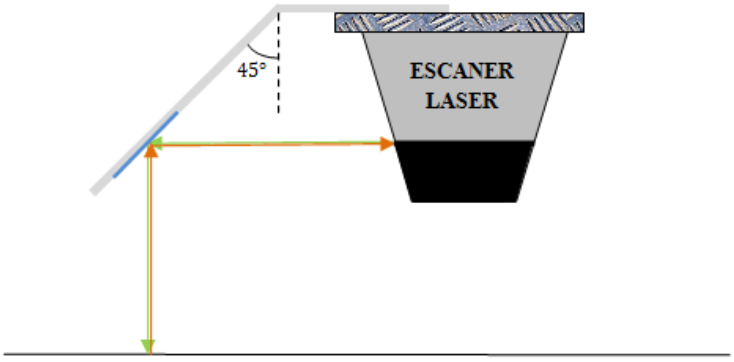



**Figura 5.17**

### 5.2.2. Placa de conexionado

La placa de conexionado (Figura 5.18) se ha diseñado para facilitar la conexión de los periféricos al microcontrolador así como para alimentarlos. Se ha utilizado placa de baquelita perforada para poder soldar los distintos componentes. La placa se alimenta desde el BEC de 5V a la salida del cual se ha soldado un conector DC circular macho. A la placa de conexionado se ha soldado un cable DC hembra para posibilitar la conexión con el BEC. A la placa se ha soldado el regulador de tensión LD33V. Los 5V son llevados al regulador de tensión que produce a su salida una tensión de 3.3V para alimentar el microcontrolador, el módulo RS232 y el XBee. El IMU, dada su colocación en el drone, se conecta directamente al microcontrolador, aunque su fuente de alimentación indirectamente también es el regulador de 3.3V. Se han colocado conectores macho para el microcontrolador y conectores hembra para el resto de periféricos.

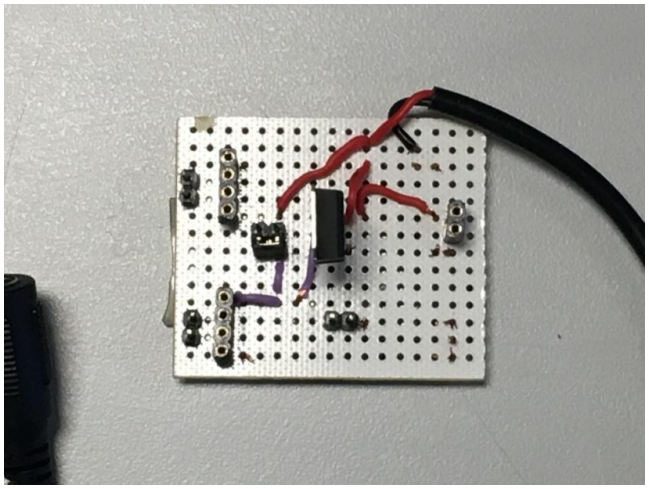


**Figura 5.18**

### 5.2.3. Montaje del drone

Una vez establecidas las conexiones que debe haber entre los distintos periféricos y construido el hardware comentado anteriormente es necesario encontrar la forma más adecuada de embarcar todos los elementos en el drone. Además de colocar los distintos elementos ha sido necesario realizar varios cambios en el drone del que se partía ya que había ciertos aspectos que debían mejorarse.

En primer lugar se ha montado un tren de aterrizaje nuevo puesto que el que había era poco robusto y no ofrecía muchas opciones a la hora de colocar la batería y el láser. El nuevo tren es más grande y amortigua mucho mejor los impactos. El tren de aterrizaje va atornillado a la plancha inferior del cuerpo del drone. En la Figura 5.19 se muestra el drone con el tren de aterrizaje antiguo y en la Figura 5.20 con el tren nuevo sin más componentes montados que el autopiloto y la batería.


**Figura 5.19**


**Figura 5.20**

Como se puede observa en la Figura 5.20, la plancha superior del drone estaba de inicio totalmente despejada. Se ha decidido ubicar sobre esa superficie  la placa del microcontrolador

sujetada con una brida. A su lado se ha colocado el módulo RS232 pegado con cinta de doble cara. La ubicación de estos elementos se muestra en la Figura 5.21.

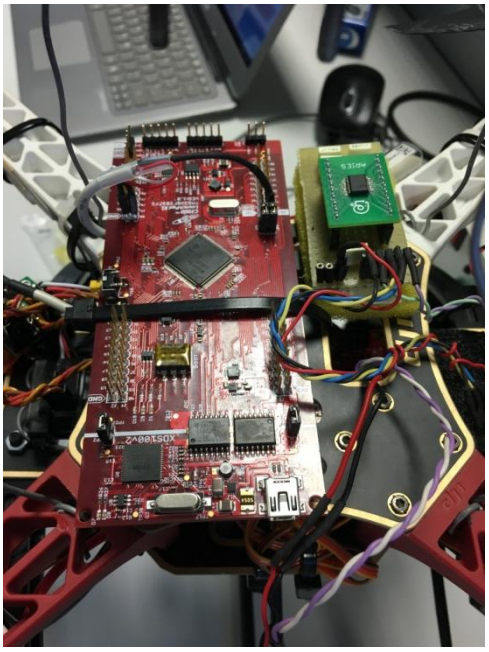


**Figura 5.21**

El IMU se ha situado sobre la placa inferior (Figura 5.22), paralelo al autopiloto y, por tanto, alineado con el sistema de coordenadas del drone. Debajo del IMU se ha pegado una esponja con cinta de doble cara para amortiguar las vibraciones que el chasis le pueda transmitir. A través de las aperturas existentes en la placa superior del cuerpo del drone se han pasado los cables del IMU (transmisión, recepción y alimentación) y se han conectado directamente al microcontrolador.

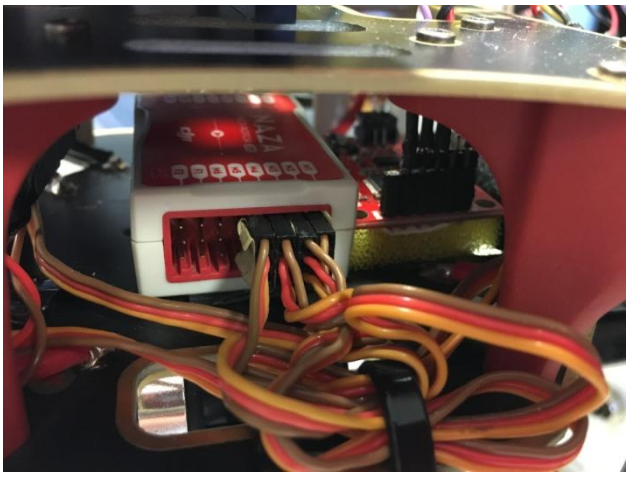


**Figura 5.22**

La placa de conexionado se ha situado en uno de los salientes de la placa inferior del cuerpo del drone para que se puedan conectar y desconectar de forma sencilla los cables. El XBee se ha

pegado con cinta de doble cara al tren de aterrizaje (Figura 5.23), bien cerca de la placa de conexionado ya que los cables de los que dispone el XBee son de longitud reducida.

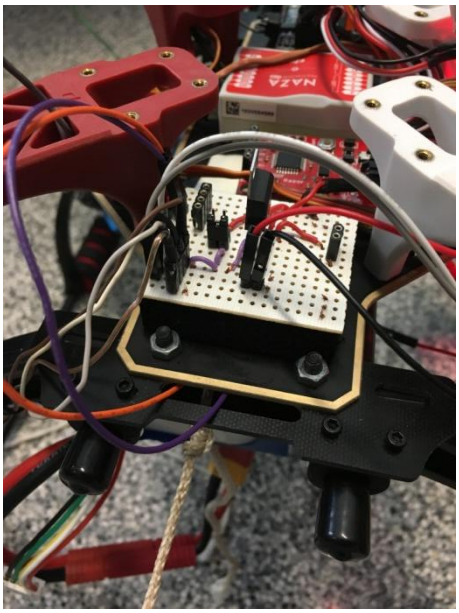


**Figura 5.23**

El láser con su soporte se ha colocado en el saliente de las barras del tren de aterrizaje (Figura 5.24), en el lado derecho del drone. El cable de comunicación RS232 era de una longitud considerable (entorno a 1m) y se ha enrollado entre las barras del tren de aterrizaje, de tal manera que no se ve ni estorba a los demás elementos.

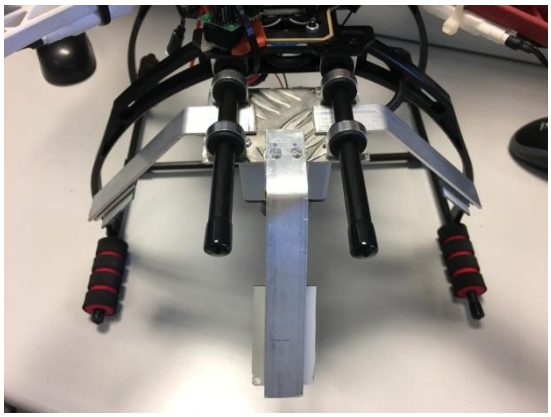


**Figura 5.24**

En cuanto a la batería, ésta se ha colocado debajo del tren, centrada pero un poco movida hacia el lado izquierdo para compensar el peso del láser, colocado en el lado opuesto, y así equilibrar el peso del drone, haciendo que el centro de gravedad se encuentre lo más cerca posible del autopiloto. Cada vez que se realiza un cambio de batería, la ubicación de la misma se ajusta para que el drone esté lo más equilibrado posible. La batería se acopla al drone con dos bridas que la sujetan a las barras del tren de aterrizaje (Figura 5.25).

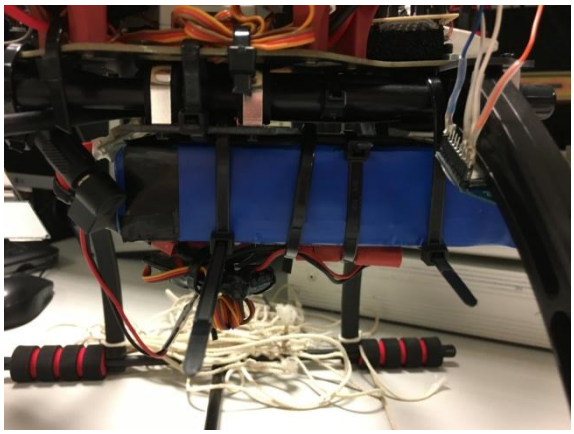


**Figura 5.25**

A la propia batería se le ha colocado en la parte baja el BEC mediante otro par de bridas. La batería tiene un único terminal de salida que en principio debe ir conectado al drone. Sin embargo el BEC también debe alimentarse directamente desde la batería, por lo que ha habido que añadir un elemento que bifurque la salida de la batería y así se pueda conectar a ambos conectores (Figura 5.26).

**Figura 5.26**

Los ajustes que ha habido que realizar a la base de la que se partía han sido imprescindibles para mejorar el vuelo del drone. Cuando el drone se voló por primera vez haciendo uso del mando radiocontrol se observó que "bailaba" mucho. Había que hacer muchas correcciones con el mando y era prácticamente imposible que se quedase más o menos estable en una posición fija. Para intentar solucionar estos problemas se utilizó el programa DJI NAZAM Lite Assitant 1.00 para ver si el IMU interno del autopiloto estaba mal calibrado. El diagnóstico del programa fue que el IMU no necesitaba calibrarse de nuevo, lo que indicaba que la calibración era correcta. Sin embargo se observó que el autopiloto no estaba situado en el centro del drone, sino que estaba considerablemente desplazado hacia uno de los lados. Se decidió colocarlo justo en el centro y el vuelo del drone mejoró sustancialmente ya que de esta forma el autopiloto estaba colocado donde su programa interno contaba con estar colocado.

Otro de los problemas que se encontró tuvo que ver con el receptor de radio. En algunas ocasiones mientras se estaba volando, se perdía el control del drone desde el mando y el LED conectado al autopiloto se ponía a parpadear en rojo. Se investigó hasta descubrir que ese parpadeo podía indicar, entre otras cosas, algún problema con la radio. Se observó que cuando el drone estaba alimentado, si se tocaba con la mano los cables de los distintos canales del receptor de la radio, el LED se ponía a parpadear en rojo. Se conectó el autopiloto al ordenador y efectivamente aparecía un mensaje de que había algún tipo de problema con la radio. Además, desde el propio programa se puede ver el estado de los canales de la radio (Figura 5.27) y se observó que las señales de los canales pasaban de un canal a otro.

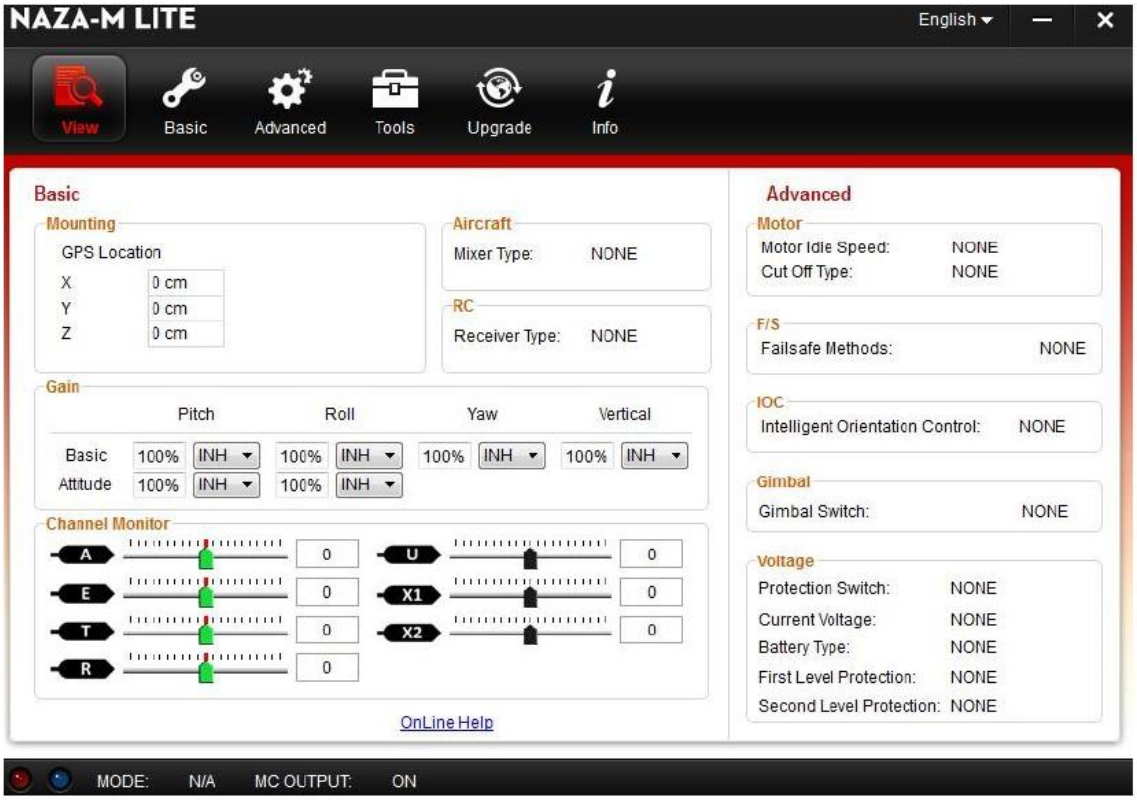



**Figura 5.27**

Se desmontó prácticamente todo el drone para comprobar que el cableado estaba bien hecho y libre de cortocircuitos. Se distribuyeron mejor los cables pero el problema seguía sucediendo de vez en cuando. Finalmente se ha llegado a una solución que, si bien no ha hecho que el problema no haya vuelto a repetirse, sí que ha conseguido que surja realmente muy pocas veces. Los cables del receptor de radio, al igual que el resto de cables conectados al autopiloto, son los típicos cables de tres conductores que llevan alimentación, masa y la señal en cuestión. Lo que se ha hecho ha sido enrollar entre sí los tres conductores en cada cable procedente del receptor

de la radio dejando a su vez los cables lo más separados posibles entre sí. De esta manera se han reducido drásticamente las interferencias entre canales.

Por último se quiere destacar el hecho de que muy a menudo ha habido que realizar ajustes y reparaciones en el montaje de los distintos componentes debido a las distintas pruebas de vuelo que se han realizado. Desde reubicar componentes que se habían movido hasta reparar soldaduras. A base de las distintas pruebas de vuelo y el comportamiento observado se ha llegado a la distribución final de las siguientes figuras.

**Figura 5.28**

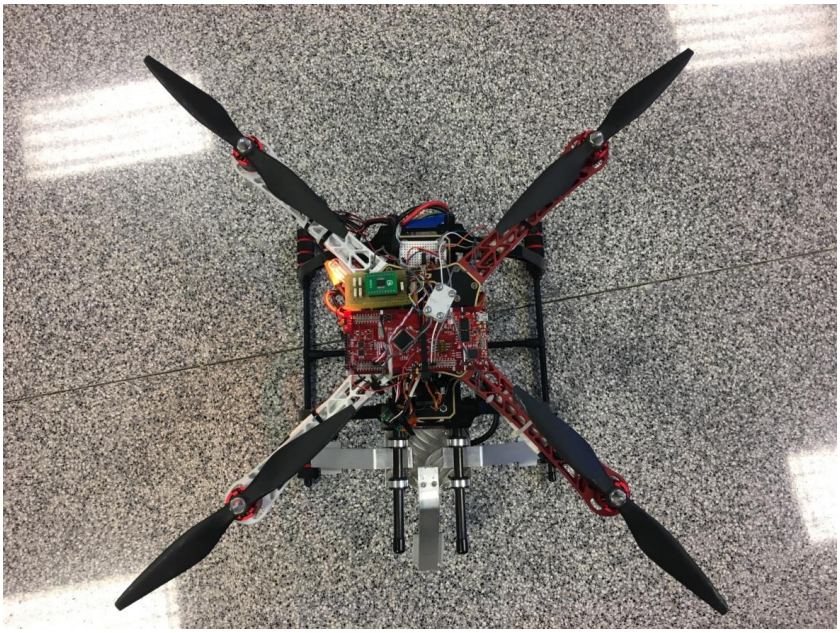


**Figura 5.29**

# Capítulo 6

# Pruebas de la aplicación

Las pruebas que se han realizado han seguido una gran evolución a medida que se ha ido consiguiendo un sistema de mejores prestaciones. En este capítulo se exponen las pruebas que se realizaron finalmente, que son las de mayor complejidad pero a la vez las que mejores resultados han dado. La parte del proceso experimental (junto con algunas gráficas relevantes obtenidas en algunas pruebas) que ha llevado a la superación de las pruebas finales se expone en el *Anexo I: Proceso experimental*.

El objetivo de esta prueba ha sido conseguir que el drone vuele a una altura fija de 60cm haciéndolo avanzar en línea recta. Se ha colocado una rampa apoyada en el suelo que conduce a una plataforma. El drone debe incrementar su altura al encontrar los obstáculos y disminuirla una vez éstos han sido superados. En definitiva, se busca ver que el control de altura implementado se adapta al perfil del terreno. La Figura 6.1 muestra un esquema del montaje de la prueba.

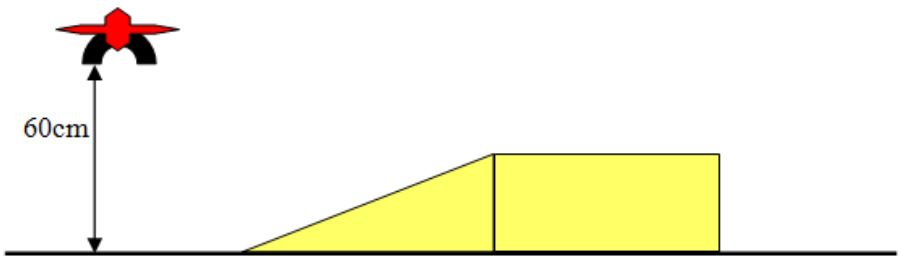



**Figura 6.1**

La prueba se ha efectuado con el control PID con prealimentación de perturbación gravitatoria y saturación *anti_windup*. Los valores de los parámetros de control han sido: $k_p = 0.015\frac{N}{mm}$, $k_d = 0.008\frac{N}{mm}$, $k_i = 0.01\frac{N}{mm}$, $prealim_{grav} = 7N$, $margen_{asc} = 1N$ y $margen_{desc} = 0N$.

Al drone se le ha enviado mediante el XBee la consigna de altura. Éste ha despegado y mediante los controles de pitch, roll y yaw se ha controlado el drone tanto como para avanzar como para realizar las correcciones direccionales necesarias. Es necesario destacar que esta prueba se ha realizado con el drone volando de forma libre, si ningún tipo de atadura con cuerdas, lo cual ya

de por sí es muy significativo. Además la prueba no ha funcionado una única vez, sino que se ha realizado con éxito varias veces.

Los resultados que se han obtenido son satisfactorios. El drone ha sido capaz de alcanzar la altura de consigna, mantenerla durante el movimiento en línea recta, subir de altura al encontrarse con la plataforma y volver a bajar una vez superada la plataforma. Es cierto que existen oscilaciones en la altura (sobre todo cuando se producen cambios en el nivel del suelo) y que el régimen permanente no se alcanza con una precisión centimétrica, pero se ha demostrado que el drone es capaz de volar a la altura de consigna y adaptarse a la geometría del terreno. La Figura 6.2 contiene el muestreo de la altura realizado en la prueba.

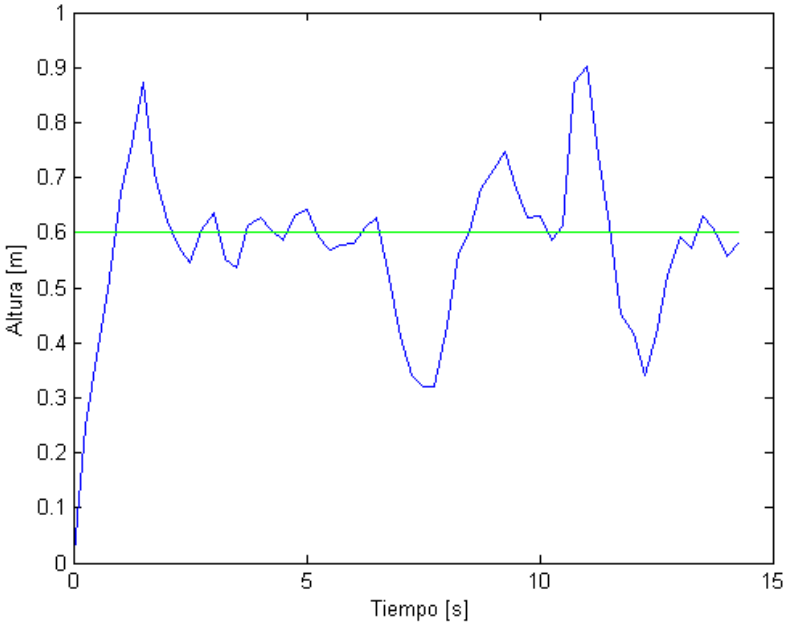



**Figura 6.2**

El pico descendiente que se produce en torno a los 7.5 segundos se corresponde con la llegada del drone al obstáculo. La siguiente sobreoscilación se debe a la acción generada para alcanzar la altura de 60cm de nuevo. El pico de altura entorno a los 11 segundos corresponde a la superación de la plataforma y la detección de nuevo del suelo. El drone desciende sobreoscilando para volver finalmente a la consigna de altura.

Las Figura 6.3 es representativa de la prueba realizada.

**Figura 6.3**

# Capítulo 7

# Conclusiones

Se ha realizado el control de la altura de un drone sobre el terreno capaz de adaptarse a los cambios de consigna de altura y de la altura sobre un terreno irregular, no exclusivamente plano. Se han ajustado los controladores para conseguir un control preciso de altura, sin o con poca sobreoscilación. El control final obtenido presenta algunas oscilaciones entorno a la consigna, que son debidas a diferentes motivos acumulados, principalmente: retardo en las medidas del sensor láser, errores de reflexión del láser sobre las superficies no perpendiculares a él, diseño de los dispositivos mecánicos realizados, control imperfecto del autopiloto de las otras variables, perturbaciones en el vuelo, y perturbaciones introducidas por el propio control manual de las otras variables. Las fuentes de error son múltiples, aunque a pesar de ello se ha conseguido realizar un control satisfactorio y estable de la altura del drone. Queda para trabajo futuro el abordar todos estos problemas en la medida de lo posible para mejorar el resultado final.

La realización del control de altura ha sido posible gracias a la precisión con que se han diseñado todos los módulos tanto hardware (incluyendo las modificaciones realizadas en el drone, como el equilibrado de pesos o el cambio de posición del autopiloto) como software del sistema. Un drone es un aparato muy poco estable por naturaleza, por lo que ha habido que ser todo lo riguroso y preciso posible en todas las facetas del proyecto para no aumentar esa inestabilidad con errores propios.

El diseño del driver del láser ha sido realmente complicado puesto que es un sensor que maneja gran cantidad de información, tiene un protocolo de comunicación potente pero complejo y se ha querido sacar de él todo el potencial que ofrece. El driver se ha diseñado de forma muy flexible y entendible para que se pueda hacer uso de él en futuros proyectos que requieran del uso de este sensor.

Considero que el proyecto ha sido muy completo. No sólo ha habido que abordar la programación de un microcontrolador, sino que también se ha diseñado la arquitectura de un sistema formado por varios dispositivos que se deben conectar y coordinar entre sí, se ha diseñado un dispositivo electrónico encargado de la distribución de alimentación y conexiones a los distintos dispositivos y además se ha realizado el diseño y montaje de un dispositivo mecánico para adaptar la medida del sensor láser al objetivo deseado del control de altura.

A nivel personal estoy muy contento con el trabajo realizado, con el cumplimiento de los objetivos y con todo lo aprendido. Creo que ha sido una experiencia muy enriquecedora, ya que ante la gran cantidad de problemas que han ido surgiendo a lo largo del proyecto he tenido que ir encontrando soluciones, que en el fondo es para lo que estamos los ingenieros.

# Anexo I: Proceso experimental

En este anexo se van a explicar las distintas pruebas que se han realizado para hallar los parámetros óptimos de control. Al final se va a mostrar la gráfica obtenida en una de las pruebas finales de vuelo estático previas a la prueba de vuelo en movimiento expuesta en el *Pruebas de la aplicación*.

Una vez escrito el programa y realizado el montaje de todos los elementos en el drone llegó el momento de comenzar a realizar las pruebas de vuelo. En primer lugar se ató el drone mediante unas cuerdas a las patas de unas mesas de uno de los laboratorios de pruebas para impedir que el drone se pudiese descontrolar ante mi inexperiencia en el vuelo de drones. Inicialmente se ató el drone muy en corto para evitar la posibilidad de que pudiese volcar. Las primeras pruebas consistieron en una primera toma de contacto controlando el drone mediante el mando de radiocontrol. El drone era muy inestable, se movía continuamente de lado a lado y no impactaba con nada porque las cuerdas lo impedían. A pesar de esta inestabilidad se decidió probar el control de altura pero el drone ni siquiera se levantó. El motivo fue que no se había implementado la prealimentación de perturbación gravitatoria en el regulador, y la ganancia proporcional no era ni mucho menos lo suficientemente grande como para que el drone se elevase.

Para obtener el valor PWM de la prealimentación gravitatoria a aplicar se hizo un programa que reducía en 30 el registro de comparación del módulo PWM cada 5 segundos. Se dejó correr el programa hasta que se observó que el drone estaba a punto de levantarse, momento en el que se tomó nota del valor del PWM y se abortó la ejecución. Es necesario destacar que el valor de la prealimentación gravitatoria depende del estado de carga de la batería, y se ha observado que puede tomar valores entre $6.4N$ y $7N$, por lo que en las distintas pruebas, a medida que se veía que el drone tenía más dificultades para despegar se iba aumentando la prealimentación gravitatoria en este rango.

Una vez aplicada en el control la prealimentación gravitatoria se volvió a probar el controlador y funcionaba realmente mal. El drone se movía mucho de lado a lado cuando teóricamente debía únicamente elevarse. Se descubrió que el autopiloto estaba mal situado y se colocó en el centro. Con esta medida que se tomó el drone era mucho más estable cuando se volaba con el mando.

Posteriormente apareció el problema de las interferencias entre los distintos canales de la radio que hacían que el drone en pleno vuelo se descontrolase y se estrellase. Por este motivo el drone sufrió varios golpes, pero gracias a las cuerdas no sufrió daños mayores ni causó daños a otros

objetos o personas. Como se ha comentado en el apartado *5.2.3. Montaje del drone*, este problema se redujo enrollando los conductores de cada cable entre sí.

Una vez solventados estos problemas se pudieron por fin centrar los esfuerzos en el ajuste de los parámetros de control mediante multitud de pruebas. La consigna de altura de dichas pruebas osciló entre los 20cm y los 40cm. Inicialmente se empezó a trabajar con el controlador PD. El valor de la prealimentación gravitatoria ya se había obtenido y faltaba por determinar los valores de las ganancias proporcional y derivativa. Dado que en el programa las alturas se trabajan en unidades de mm, las ganancias en vez de tener unidades de $\frac{N}{m}$ tienen unidades de $\frac{N}{mm}$. Por tanto, las ganancias iniciales propuestas en el apartado *25* se dividieron entre mil.

En primer lugar se probó a utilizar únicamente la ganancia proporcional para ajustar el permanente dejando nula la ganancia derivativa. El resultado fue malo per coherente con las simulaciones ya que el drone no paraba de oscilar, llegando a rebotar en el suelo, sin equilibrarse en ningún momento. Entonces se decidió aplicar también una ganancia derivativa para que se opusiera a esas oscilaciones. A partir de esos parámetros iniciales se fue aumentando y disminuyendo el valor de esas ganancias en relación a lo que se iba observando en las pruebas. Si se aumentaba la ganancia proporcional el resultado eran grandes oscilaciones y si se reducía no se alcanzaba la altura de consigna. Si la derivativa era demasiado pequeña había oscilaciones, si se aumentaba disminuían y si era demasiado grande las oscilaciones inestabilizaban el sistema. Los mejores resultados se obtuvieron con $k_p = 0.0075$ y $k_d = 0.004$, aunque no eran buenos ya que las oscilaciones eran exageradas y no se alcanzaba de forma correcta la altura de referencia. Después de muchas pruebas con este controlador se llegó a dos conclusiones: era necesario saturar la acción para reducir las oscilaciones para así no tener que reducir la ganancia proporcional tanto como para que el drone no tuviese suficiente fuerza como para ascender hasta la altura de consigna y que se iba a añadir una parte integral al regulador para que ayudase alcanzar el régimen permanente.

En primer lugar se trataron de ajustar los márgenes de acción tanto de ascensión como de descenso. Se comenzó por valores elevados, pero finalmente se comprobó que para reducir las oscilaciones lo suficiente había que establecer unos márgenes bajos, que finalmente fueron $margen_{asc} = 1N$ y $margen_{desc} = 0N$. Las oscilaciones no se reducen únicamente de este modo, el ajuste que se ha hecho de las ganancias también es crucial. Con la saturación, los resultados obtenidos con el control PD mejoraron, pero se seguía teniendo mucha dificultad para alcanzar la referencia y las oscilaciones seguían siendo excesivas. La Figura A1.1 muestra una prueba de vuelo con una consigna de altura de 35cm implementando control PD con

prealimentación gravitatoria y saturación. Los valores de los parámetros de control en esta prueba fueron $k_p = 0.01\frac{N}{mm}$, $k_d = 0.004\frac{N}{mm}$, $margen_{asc} = 1N$ y $margen_{desc} = 0N$.

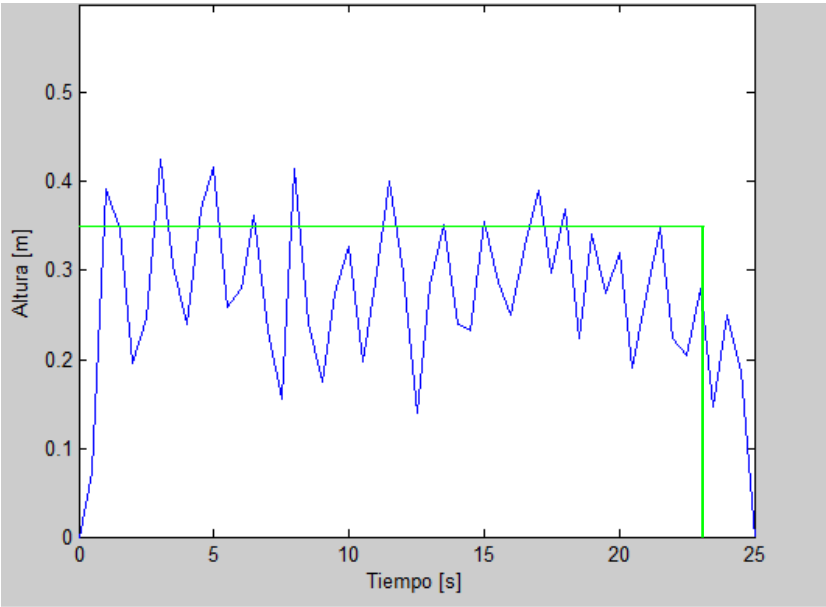



**Figura A1.1**

Con la implementación del controlador PID con saturación *anti-windup* los resultados fueron mucho mejores. Dentro de que siguió habiendo oscilaciones, éstas eran menores y se alcanzaba mucho mejor la altura de consigna. Se empezó a ajustar el regulador con los valores de $k_p$ y $k_d$ que mejor habían funcionado en el regulador PD y con un valor de $k_i = 0.001$. Se fueron modificando los distintos parámetros hasta obtener los que han dado mejores resultados y que se han utilizado también en la prueba de avance con obstáculos del drone, que son: $k_p = 0.015\frac{N}{mm}$, $k_d = 0.008\frac{N}{mm}$, $k_i = 0.01\frac{N}{mm}$, $margen_{asc} = 1N$ y $margen_{desc} = 0N$.

Todas estas pruebas se realizaron también con el drone atado con cuerdas por las patas, pero a medida que iba funcionando mejor el sistema, se le fue dando más margen de cuerda al drone ya que las propias cuerdas cuando el drone se mueve le dan tirones y los desestabilizan.

Se probó otro montaje, el de la Figura A1.2, en el que se pretendía atar el drone mediante una única cuerda a una barra saliente de la pared del laboratorio. Se colocaron unas bridas en el drone de tal manera que guiasen el descenso de la cuerda a medida que el drone ascendía, pero cuando el drone se movía un poco lateralmente, la cuerda ya no descendía por la guía que se le había hecho y chocaba con las hélices, por lo que rápidamente se descartó este montaje y se volvió el de las cuerdas atadas al tren de aterrizaje.

**Figura A1.2**

A continuación se muestra el resultado de una prueba (Figura A1.3) que se realizó con los parámetros que se han indicado anteriormente. La consigna de altura que se estableció fue de 40cm.

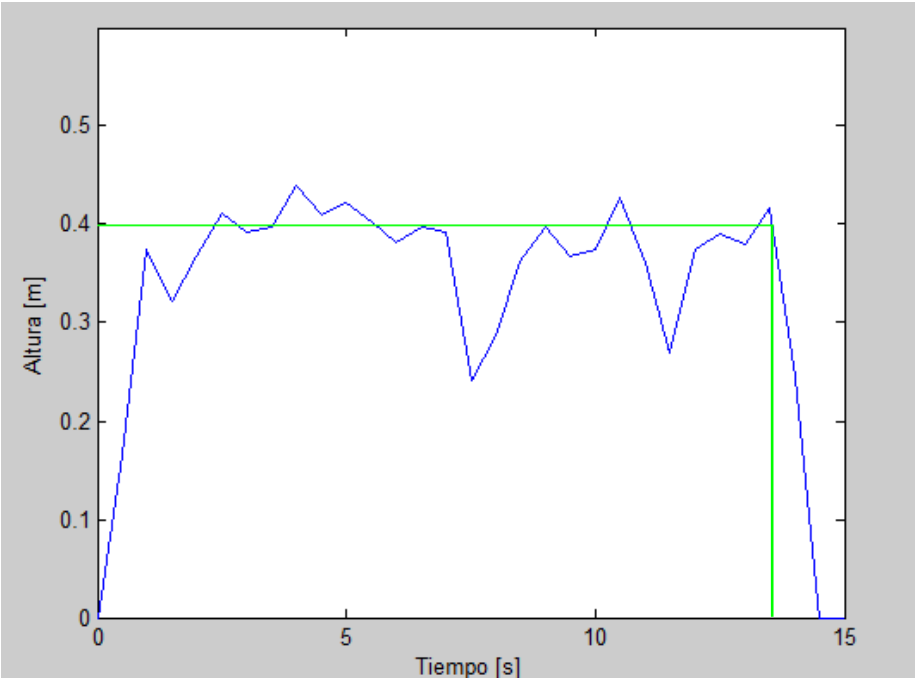



**Figura A1.3**

Se observa que aunque de vez en cuando se produce una perturbación, que pudo ser entre otros factores por el tirón de alguna de las cuerdas de sujeción, se alcanzó la altura de consigna tanto inicialmente como después de esas perturbaciones.

Los últimos resultados conseguidos son los mostrados en el *57*.

# Anexo II: Comandos de comunicación

En el presente anexo se muestra el listado de comandos que se pueden enviar al drone desde el programa DIGI XCTU junto con la estructura de cada uno de ellos para que el usuario pueda utilizar el sistema diseñado. Todos los comandos tienen un carácter de inicio de trama (‘#’) y un carácter de fin de trama (‘\n’). Es imprescindible añadir dicho carácter al final de cada mensaje para que éste pueda ser procesado correctamente.

- Consigna de altura: este comando fija la consigna de referencia con la altura en metros. *Ejemplo: #SETR 0.5*

- Ganancia proporcional: este comando modifica la ganancia proporcional del regulador en unidades de $\frac{N}{mm}$. *Ejemplo: #KP 0.015*

- Ganancia derivativa: este comando modifica la ganancia derivativa del regulador en unidades de $\frac{N}{mm}$. *Ejemplo: #KD 0.008*

- Ganancia integral: este comando modifica la ganancia integral del regulador en unidades de $\frac{N}{mm}$. *Ejemplo: #KI 0.00*

- Margen de saturación de ascensión: este comando modifica el margen de saturación superior de la acción de control en newtons. *Ejemplo: #USAT 1*

- Margen de saturación de descenso: este comando modifica el margen de saturación inferior de la acción de control en newtons. *Ejemplo: #DSAT 0*

- Prealimentación gravitatoria: este comando modifica el valor de la prealimentación de compensación de la fuerza gravitatoria en newtons. *Ejemplo: #GRAV 7*

# Anexo III: Parámetros obtenidos

En el presente anexo se concentran los parámetros que se han obtenido de forma experimental durante el desarrollo del proyecto de cara a facilitar la realización de futuras ampliaciones.

- Ganancia proporcional óptima $k_p = 0.015 \frac{N}{mm}$.
- Ganancia derivativa óptima $k_d = 0.008 \frac{N}{mm}$.
- Ganancia integral óptima $k_i = 0.01 \frac{N}{mm}$
- Margen de saturación de ascensión óptimo $marg_{asc} = 1N$.
- Margen de saturación de descenso óptimo $marg_{desc} = 0N$.
- Prealimentación gravitatoria óptima $prealim_{grav} \in [6.4, 7]N$, según carga de la batería.
- Distancia del láser al espejo frontal $d_f = 105mm$
- Distancia del espejo frontal al suelo $l_f = 100mm$
- Distancia del láser al espejo 1 al $d_1 = 115mm$
- Distancia del espejo 1 al suelo $l_1 = 98mm$
- Distancia del láser al espejo 2 $d_2 = 107mm$
- Distancia del espejo 2 al suelo $l_2 = 105mm$
- Distancia paralela al suelo del centro del láser al centro del drone $L = 220mm$

# Anexo IV: Análisis de tiempo real

En el presente anexo se analiza el cumplimiento de los requisitos de tiempo real del sistema de tareas implementadas. Se ha diseñado una estructura de tareas basada en el criterio *Rate Monotonic* por lo que se ha elegido para todas las tareas que el plazo de respuesta sea igual al periodo. Este criterio da mayor prioridad a las tareas de menor periodo. Para gestionar el acceso a los recursos compartidos se utiliza el protocolo de bloqueo por herencia de prioridad. La Figura A4.1 muestra el listado de las distintas tareas e interrupciones hardware y software del programa junto con su periodo (P), plazo de respuesta (D), tiempo de cómputo (C), tiempo de bloqueo ($B_{hp}$) y número de bloqueos que se han calculado. El tiempo de acceso a un servidor se ha medido en 9.3µs y el tiempo de dos cambios de contexto es de 57µs.

| Tarea | Prioridad | P[ms] | D[ms] | C[ms] | Bhp[ms] | Número de bloqueos |
|---|---|---|---|---|---|---|
| Reloj (HWI) | HW | 1 | - | 0,015 | - | - |
| SCIa_RX (HWI) | HW | 0,04 | - | 0,0024 | - | - |
| SCIa_TX (HWI) | HW | 0,04 | - | 0,002 | - | - |
| SCIb_RX (HWI) | HW | 0,173 | - | 0,0024 | - | - |
| SCIb_TX (HWI) | HW | 0,173 | - | 0,002 | - | - |
| SCIc_RX (HWI) | HW | 0,938 | - | 0,0024 | - | - |
| SCIc_TX (HWI) | HW | 0,938 | - | 0,002 | - | - |
| IMU_SWI | SW | 25 | - | 0,081 | - | - |
| Laser_SWI | SW | 100 | - | 0,302 | - | - |
| XBee_SWI | SW | 300 | - | 0,124 | - | - |
| TaskIMU | 6 | 25 | 25 | 0,16 | 0,0093 | 1 |
| TaskReceiveLaser | 5 | 100 | 100 | 1,02 | 0,0186 | 2 |
| TaskControl | 4 | 108 | 108 | 0,4 | 0,0279 | 3 |
| TaskSendLaser | 3 | 150 | 150 | 0,0068 | 0,0186 | 2 |
| TaskSendComm | 2 | 200 | 200 | 1 | 0,0093 | 1 |
| TaskReceiveComm | 1 | 300 | 300 | 0,83 | 0 | 0 |

**Figura A4.1**

Aplicando el Teorema 5 de Sha, Rajkumar y Lehoczky, el cual establece que en un sistema de n tareas periódicas con prioridades asignadas en orden de frecuencia, que se comunican mediante servidores, se cumplen todos los plazos de respuesta, para cualquier desfase inicial de las tareas, si:

$$\forall i, 1 \leq i \leq n, min_{t \in S_i} \left( \sum_{j=1}^{i} \frac{C_j}{t} \left\lceil \frac{t}{C_j} \right\rceil \right) + \frac{b_i}{t} \leq 1 \quad (32)$$

Para comprobar que se cumple esta condición suficiente, se calcula el tiempo de respuesta en el plazo de cada tarea para el nivel de prioridad de cada tarea:

| Wi(Di) | Wi(Di) [ms] | | Di [ms] | |
|---|---|---|---|---|
| W1(D1) | 4,6151 | < | 25 | OK |
| W2(D2) | 18,061 | < | 100 | OK |
| W3(D3) | 20,7323 | < | 108 | OK |
| W4(D4) | 22,5766 | < | 150 | OK |
| W5(D5) | 24,8147 | < | 200 | OK |
| W6(D6) | 29,7308 | < | 300 | OK |

**Figura A4.2**

Con esto queda demostrado que la estructura de tareas creada cumple los plazos previstos.

# Bibliografía


**[ 1 ]** Corke, P.: Robotics, Vision and Control. Fundamental Algorithms in MATLAB®. Second, Completely Revised, Extended and Updated Edition. Springer Trackts in Advanced Roboitcs, vol. 118 (2017)

**[ 2 ]** Beatriz Frisón. Modelado y control de un helicóptero de cuatro motores. Proyecto Final de Carrera. Universidad de Zaragoza, CPS, 2008/2009.

**[ 3 ]** Kilian Nicolás Pascual. Planificación de misiones y navegación autónoma de un quadcopter. Trabajo de Fin Grado. Universidad de Zaragoza, CPS, 2015/2016

**[ 4 ]** Texas Instruments, SYS/BIOS (TI-RTOS Kernel) v6.46 User's guide (2016)

**[ 5 ]** Texas Instruments, TMS320F2837xS Delfino Microcontrollers Technical Reference Manual (2014, revised in 2016)

**[ 6 ]** Hokuyo, Communication Protocol Specification For SCIP2.0 Standard (2006)

**[ 7 ]** DJI, NAZA-M Lite User Manual v2.00 2014.04.21 Revision

**[ 8 ]** Pololu RC Switch User's Guide, © 2001–2015 Pololu Corporation

**[ 9 ]** Texas Instruments, F2837xS Firmware Development Package User's Guide (2016)

**[ 10 ]** Robomart, Especificaciones motor DJI 2212/920Kv URL: https://www.robomart.com/dji-2212-920kv-brushless-motor-for-multicopter

**[ 11 ]** Hyperphysics, Principio de Fermat URL: http://hyperphysics.phy-astr.gsu.edu/hbasees/phyopt/Fermat.html

**[ 12 ]** DIGI International Inc., XBee®/XBee-PRO® RF Modules Product manual (2009)

**[ 13 ]** Futaba Corporation, INSTRUCTION MANUAL for Futaba 6J-2.4GHz (2011)

**[ 14 ]** David Pont Esteban, Los UAVs y sus aplicaciones civiles, Zaragoza, 2012