Universidad
Carlos III de Madrid

UNIVERSIDAD CARLOS III DE MADRID
DEPARTAMENTO DE INGENIERÍA DE SISTEMAS Y AUTOMÁTICA

TESIS DE MASTER

Quadcopter: Construcción, control de vuelo y navegación GPS

Autor: Eduardo Parada Pardo
Director: Mohamed Abderrahim Fichouche

MASTER OFICIAL EN
ROBÓTICA Y AUTOMATIZACIÓN

LEGANÉS, MADRID
OCTUBRE - 2012

UNIVERSIDAD CARLOS III DE MADRID

MASTER OFICIAL EN ROBÓTICA Y AUTOMATIZACIÓN

El tribunal aprueba la tesis de Máster titulada “QUADCOPTER: CONSTRUCCIÓN, CONTROL DE VUELO Y NAVEGACIÓN GPS.” realizada por Eduardo Parada Pardo.

Fecha: Octubre 2012

Tribunal: _____

_____

_____

A mis padres

Agradecimientos

Quiero agradecer en primer lugar a mi novia por haberme preguntado sin parar como iba y apoyarme en todo momento. A mis amigos por haberme apoyado a seguir y sobre todo para saber si me había atacado una vez más el robot.

A mi tutor por haberme guiado y ayudado en todo lo que ha podido. Y la profesora Meriem Nachidi, por proporcionarme toda la información y ayuda referente a control.

A Jonathan por haberme ayudado tanto con muchísima información, guiándome y dándome consejos.

A mi familia, porque su apoyo incondicional es lo más valioso que tengo y lo que me da fuerzas para luchar por alcanzar cualquier meta.

Índice general

Índice de figuras

Capítulo *2*

Capítulo *3*

Capítulo 4

Capítulo 5

Resumen

En este trabajo final del máster se presenta una plataforma casi completa de un robot UAV (Unmanned Aerial Vehicle), concretamente un quadcopter. La finalidad de este proyecto será presentar una plataforma capaz de realizar un vuelo estable y a su vez realizar una navegación a través de POIs (Points Of Interests) dados.

Dicho quadcopter tiene como finalidad servir como plataforma de lucha anti-incendios, consiguiendo así una herramienta de bajo coste y altamente eficaz capaz de controlar extensas zonas arboladas.

Esta versión de la plataforma casi completa solo le faltaría añadir una cámara térmica capaz de medir la temperatura mientras realiza vuelos rutinarios. Dichos vuelos estarían controlados con una ruta predefinida. Esta ruta seria marcada a través de POIs introducidos previamente.

Ya que posee una capacidad de transmisión de datos SMS, si encontrara una temperatura superior a la definida como peligrosa, el quadcopter enviaría un mensaje al centro de guardabosques informando de la temperatura, la longitud y la latitud.

Este trabajo de máster, por tanto, unifica 4 puntos. El diseño y construcción de una plataforma de vuelo de 4 motores, un control de vuelo, una navegación GPS y por último una comunicación SMS.

Abstract

This master project presents an almost complete platform for a UAV (Unmanned Aerial Vehicle) robot, quadcopter.

The first aim of this project is to construct a platform of a quadcopter and the second aim is to design control methodologies for stabilization and trajectory control of the quadcopter.

This quadcopter is intended to serve as a platform for fire-fighting, obtaining a low cost and highly effective platform capable of controlling extended forested areas.

This version of the platform is almost complete; it only needs to add a thermal imaging camera able to measure the temperature while conducting routine flight. These flights will be controlled by a predefined path. This route will be marked by POIs (Points Of Interests) previously entered.

In addition, our system takes into account data transmission through SMS. Thus, if the detected temperature is above a predefined threshold, the quadcopter sends a warning message informing to a control center reporting the temperature, longitude and latitude.

This project therefore, unifies 4 points: the design and construction of a flying platform with 4 electrical motors, flight controller design, GPS navigation and an SMS communication.

Capítulo 1

Introducción

Desde el origen de la robótica se lleva buscando obtener robots con capacidades que ayuden a la raza humana en situaciones adversas, situaciones fuera del alcance del cuerpo humano ó en su defecto situaciones peligrosas para el mismo.

Últimamente, a causa de la facilidad de conseguir las herramientas para construir robots aéreos, se ha generado un gran interés en este área de la robótica, entre ellos hay un rama denominada quadcopter o quadrotor.

Un quadcopter es un helicóptero de 4 motores. Dichos motores están dirigidos hacia arriba en una formalización cuadrada, siempre con la misma distancia al centro de masas del quadcopter. El quadcopter se controla ajustando las velocidades angulares de los motores, que se hacen girar mediante motores eléctricos. Este tipo de robots son usados en vigilancia, búsqueda, inspección en la construcción, etc.

Los quadcopter aportan una serie de ventajas muy útiles para este tipo de tareas mencionadas anteriormente. Al ser un "Unmanned Aerial

Vehicle“ (UAV), en caso se estrellarse a causa de fenómenos medioambientales adversos, los daños serian materiales y al ser eléctricos, la posibilidad de provocar un incendio serian mínimas. Otra ventaja es la capacidad de estos robots de permanecer estacionarios en vuelo, y con ello obtener imágenes nítidas. Ya que dichos sistemas serian implementados en los bosques, donde no se pueden disponer de extensas zonas donde despegue un UAV de tipo avión, el uso de quadcopter consigue minimizar el impacto medioambiental a la hora de integrarlos en la protección de los bosques. Con los robots de tipo UAV se pretende solventar la carencia del ser humano de poder ver a simple vista extensas zonas, de detectar temperaturas altas entre la densidad de los bosques o controlar zonas donde el incendio está activo y poder informar de la posición del fuego a corta distancia sin correr peligro personas.

Por el contrario, estas plataformas plantean algunas desventajas que están principalmente relacionadas con la tecnología actual. El principal problema reside en la autonomía de las baterías. Actualmente un quadcopter equipado con una cámara dispone de una autonomía de aproximadamente 15 minutos. Otro problema es la poca capacidad para integrar herramientas extras, como cámaras ó actuadores.

En los últimos años, tanto universidades, empresas como gente particular han invertido tiempo y dinero en el desarrollo de quadcopter, como el Ar.Drone de Parrot, con esto se muestra que tanto empresas como particulares se han fijado en esta nueva tecnología, a su vez muchos departamentos de investigación trabajan sobre este tema. Por ejemplo el laboratorio de General Robotics, Automation, Sensing and Perception (GRASP) de la universidad de Pensilvania, donde se estudia los quadcopters que son capaces de realizar tareas cooperativas. Respecto al objetivo planteado en este trabajo, disponer de esta capacidad optimizaría el tiempo

que necesitarían los quadcopters para realizar la vigilancia de un área delimitada. Otras universidad como ETH en Zürich, desarrollan algoritmos de control capaces de realizar maniobras complicadas de vuelo, como evasión de obstáculos en espacios reducidos o maniobras acrobáticas. Estos comportamientos mencionados son explicados detenidamente en el capítulo siguiente.

En este proyecto, nos interesamos en la constricción y el control de un quadcopter. Este tipo de UAV presenta ventajas relacionadas con la finalidad para la que planteamos este proyecto. Una de ellas es la capacidad de volar incluso contra fuertes vendavales [1]. Además, con su debida modificación pueden volar incluso con lluvia y realizar las tareas definidas sin problemas. Por consecuencia, se amolda perfectamente a las condiciones climatológicas de los bosques.

La plataforma que se propone en este proyecto es una plataforma de dimensiones reducidas y sobretodo de bajo coste comparado con otros sistemas de lucha contraincendios como helicópteros ó avionetas de reconocimiento. Analizando los requisitos técnicos que requiere la implementación de un sistema de navegación de este tipo, se apuesta por dos soluciones adecuadas para garantizar el control de vuelo de este quadcopter. Una posible solución está enfocada al diseño de un controlador PID y la otra solución utiliza la estrategia de control Fuzzy.

1.1 Motivación

La motivación de este trabajo es conseguir una plataforma plenamente operativa, de bajo coste y eficaz que permita vigilar a los bosques de incendios. Es algo que se debería plantear claramente a la hora de definir

como vigilar las zonas forestales, ya que son estos los que mantienen la vida de todos los seres del planeta.

Actualmente a causa de los problemas económicos, los sistemas de detección de incendios se han visto reducidos drásticamente. Esta plataforma resuelve dicho problema y además mejora los controles anti incendios existentes, ya que la vigilancia realizada desde un helicóptero necesita de un piloto humano que necesita descansar un número de horas en el cual, se pueden iniciar incendios. Al contrario, los quadcopters solo necesitan recargar sus baterías en un corto periodo de tiempo (30 min aproximadamente).

1.2 Objetivos

El trabajo expuesto en esta memoria trata tres objetivos principales, descritos a continuación:

- Realizar una plataforma de vuelo tipo quadcopter, el objetivo de no comprar directamente esta plataforma y diseñarla, desarrollarla y construirla desde cero, tiene una doble finalidad. Calcular la dificultad y coste de la plataforma y intentar mejorar las plataformas existentes.
- Implementar el algoritmo de control de vuelo que sea capaz de realizar los requisitos de vuelos necesarios para completar un vuelo estable en cualquier tipo de condición climatológica.
- Crear un sistema de navegación y comunicación. Conseguir que los AUV envíen la posición donde se encuentra el foco del fuego y la evolución del mismo, reforzaría las posibilidades de controlar antes de su propagación.

Con este proyecto se pretende conseguir una plataforma de bajo coste, estable y equipada, comparada con otros modelos de quadcopters que

existen a la venta. Por ejemplo, el 3DR Arducopter Quad-C de DiyDrones (449$) ó el Arducopter Quad V1.1. de jDrones (549$).

1.3 Estructura del documento

El **capítulo 2** es una descripción del estado del arte en vistas generales de los diseños que otros autores han empleado cuando se ha tratado de hacer proyectos relacionados con quadcopters y control de vuelo.

En el **capítulo 3** se hace una descripción general de la plataforma de vuelo, y posteriormente una guía paso a paso de la evolución que ha sufrido el UAV hasta completar su diseño final.

En el **capítulo 4** se lleva a cabo la implementación del software de control de vuelo. Con sus 2 variables y comparando sus ventajas e inconvenientes.

El **capítulo 5** detalla el funcionamiento y uso de la navegación GPS y de la comunicación SMS.

El **capítulo 6** detalla los resultados experimentales, mostrando las diferentes pruebas realizadas y los datos obtenidos.

Finalmente, en el **capítulo 7** se presentan las conclusiones obtenidas y los trabajos futuros que hay interés en desarrollar.

Capítulo 2

Estado del arte

En los últimos años, dentro del campo de los UAV, los quadcopters se han convertido en una línea de investigación a seguir a causa de sus numerosas ventajas.

Hasta la fecha desarrollar tecnologías relacionadas con UAV era muy rara debido al alto coste que requería para construir y mantener este tipo de robots. Actualmente se ha reducido el coste de los componentes necesarios (sensores, motores, actuadores), por lo que se ha podido comenzar a desarrollar nuevas investigaciones relacionadas con los UAV.

Los UAV cuentan con diferentes sistemas de reconocimiento del medio, ya sea por medio de cámaras, laser [2], sonares [3], etc. Actualmente el medio más usado son las cámaras, como por ejemplo, las cámaras para la estimación de posición [4], [5]. Últimamente se han desarrollado nuevos avances usando sistemas como “Kinect” [6] donde ha demostrado su capacidad [7]. Ya que los micro controladores que portan los quadcopter son poco potentes (obligado para que tengan un bajo consumo), la capacidad que dispone de tratamiento de imágenes en los quadcopter es muy limitado, por lo que el uso de estos sistemas se ve condicionado.

En algunas investigaciones [8] se ha optado por incluir una cámara independiente al sistema que transmite directamente a un receptor provocando que el radio de alcance sea reducido. En otros casos, la cámara solo realiza la grabación guardándose en la memoria de la misma, para su posterior tratamiento. En nuestro caso, la cámara solo se usaría en un momento indicado cuando la plataforma hubiera alcanzado el POI marcado para no usar demasiado tiempo de CPU.

2.1. UAV

A la hora de definir qué tipo de UAV se requiere para cada situación, primero se realiza un estudio del entorno, ya que el tipo de UAV a elegir difiere drásticamente según la situación

Dentro del tipo de UAV de tipo avión podemos diferenciarlos en 3 tamaños: pequeños, medianos y grandes.

Dentro de los UAV grandes, los EEUU disponen de RQ-4 Global Hawk. Este UAV tiene 116 pies de envergadura, es capaz de realizar vuelos de hasta 36 horas, con lo que le permite despegar desde EEUU realizar la misión designada y aterrizar desde la misma base militar que despegó. Para realizar las tareas de vigilancia dispone de un SAR (radar de apertura sintética), capaz de mapear extensas aéreas de terrero y visualizar el terreno a través de las nubes o de tormentas de arena. También dispone de lente ópticas e infrarrojos. El Global Hawk proporciona información de 103.600 km^2 de terreno al día (equivalente a Portugal) y vuela a 19.812m de altura.

El principal problema de este tipo de UAV es su elevado coste, 218 millones de dólares.

Figura 2.1: *Global Hawk*

Dentro de la clase media de tamaño, uno de los UAV más importantes de los que dispone el ejerció de EEUU es el RQ-1 Predator, actualmente dispone de más de 60.

Figura 2.2: *Predator*

Su uso más habitual es de realizar misiones de reconocimiento. Su rango es de 450 millas y 16 horas de vuelo. Este tipo de UAV cuenta con una potente carga de pago, con la cual es capaz de realizar grabaciones de video de alta definición. Dispone de cámaras de visión infrarrojas y de un SAR (radar de apertura sintética) con el que es capaz de mapear amplias secciones de tierra. El coste de este modelo de UAV es inferior (40 millones de dólares).

Por último, dentro de los UAV definimos los de tamaño pequeño, estos son con lo que realizaremos la comparación a la hora de elegir entre un UAV de tipo avión o de tipo hélice.

EN RQ-11 Raven es un modelo de UAV con un alcance máximo de 10 km y 110 minutos de autonomía, su techo está situado en los 500 pies.

Figura 2.3: *RQ-11 Raven*

Los UAV de tipo avión disponen de la ventaja de tener una autonomía superior ya que requieren de menos gasto energético a la hora de conseguir su sustentación. A su vez un AUV de este tipo puede alcanzar altitudes superiores, con lo que puede mapear una mayor área.

La desventaja que cuenta este tipo de AUV, es que requieren una larga pista de despegue y unas condiciones climatológicas favorables. Los quadcopters, son capaces de realizar un despegue vertical y volar hasta con viento de más de 80k/h. Estos poseen la ventaja de no solo ser más baratos económicamente, sino que sus capacidades de maniobrabilidad son muy superiores. Poseyendo mayor cantidad de grados de libertada.

Ya que nuestra zona operativa donde trabajara nuestro UAV, es una zona boscosa con gran facilidad para variar es estado climatológico, se requiere un UAV versátil y que no requiera provocar un impacto medioambiental a la hora de implantar dicha tecnología.

2.2 Quadricopter

Actualmente en mercado existen alta gama de modelos de helicópteros, ya sean tricopter, quadcopter u optocopter, todos ellos siguen el mismo patrón de vuelo y control.

Estos tipos de UAV no tienes aspas de cola para controlar la rotación, sino que en el caso de los quadcopter 2 motores rotaran en un sentido horario y los otros 2 motores giraran en sentido anti horario, con la finalidad que el quadcopter compense la rotación y se mantenga estable.

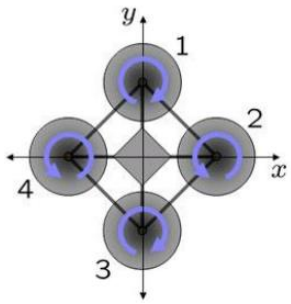


Figura 2.4: *Rotación de los motores en un quadcopter*

Los quadcopter poseen son usado para gran cantidad de aplicaciones, desde uso militar para vigilancia, hasta uso civil para realizar grabaciones en lugares de difícil acceso. El coste bajo de este tipo de UAV permite que su implantación se extienda con facilidad en el mundo científico y actualmente este abriéndose camino entre usuarios que disfrutan del aeromodelismo.

2.2.1 Control de vuelo

Los quadcopter han recibido una atención considerable por parte de los investigadores, los fenómenos complejos del quadcopter han generado diversas áreas de interés. El modelo dinámico del quadcopter es el punto de partida para la mayoría de los estudios. En la literatura, diferentes métodos de control han sido investigados, incluyendo controladores PID, el control

Backstepping, H ∞ no lineal de control, los controladores LQR, y no lineales controladores con saturaciones anidadas. Los métodos de control requieren una información exacta de la posición y la actitud de las mediciones realizadas con un giroscopio, un acelerómetro, y otros aparatos de medición, como el GPS.

En este documento, nos interesamos a la estabilización y al control de trayectoria de un quadcopter usando dos estrategia de control que son PID y la lógica borrosa

2.2.1.1 Vuelo acrobático usando un controlador Fuzzy

Investigadores como Markus Hehn [9] (estudiante de ETH Zürich, Switzerland) desarrollan algoritmos de control capaces de realizar tareas complejas en vuelo y vuelo acrobático.

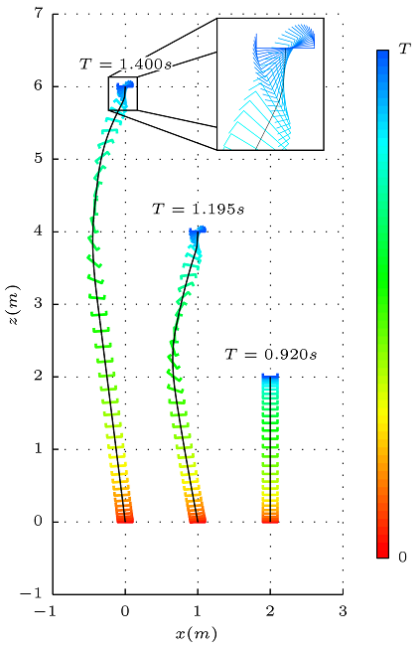


Figura 2.5: *Modelos de elevación vertical*

En los últimos años se han conseguido realizar algoritmos que permitan realizar vuelos más complejos [10] o maniobras acrobáticas [11]. Estos algoritmos son capaces de generar trayectorias de tipo lineal, curvas o rizadas.

El mayor problema que plantea realizar este tipo de trayectorias complejas es la generación de la trayectoria en tiempo real entre 2 estados. Una de los métodos es la programación no-lineal y los algoritmos genéticos [12]. Este tipo de sistemas minimizan numéricamente el tiempo necesario que requiere el algoritmo para procesar los datos entre 2 estados, asumiendo que los datos de control de entrada son recibidos por partes y no de una forma constante.

Para mejorar esto, Markus Hehn (entre otros) diseñaron un algoritmo capaz de realizar el control del quadcopter y calcular la trayectoria entre 2 estados de una forma más eficiente.

Para ello se centran en la realización de un modelo inicial del quadcopter, centrándose en solo 2 dimensiones con 3 grados de libertad. El eje horizontal x, el vertical z y el ángulo de inclinación entre ambos ejes Θ, como se muestra en la Figura 2.3

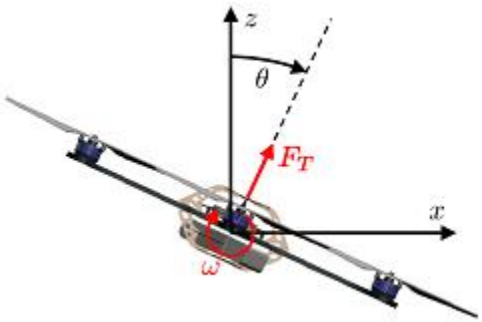


Figura 2.6: *Entradas del modelo del quadcopter*

El control del quadcopter es controlado por 2 entradas, la suma total de fuerzas (F_T) y el ángulo de inclinación (ω).Ambas entradas están controladas para que tengan saturación.

$$\underline{F_T} \leq F_T \leq \overline{F_T}\,, |w| \leq \bar{w}$$

Ya que durante el vuelo es imposible que un motor gire en sentido contrario al realizado en la marcha inicial, se asume que $\underline{F_T} > 0$. La ecuación de movimiento resultante es:

$$\ddot{x} = \frac{F_T}{m}\sin\theta\,, \ddot{z} = \frac{F_T}{m}\cos\theta - g, \theta = \omega$$

Donde *g* es la aceleración de la gravedad y *m* el peso del quadcopter.

Para describir el sistema se introduce una transformación no-dimensionalizada en la ecuación anterior y la frecuencia de rotación.

$$\hat{t} = \bar{\omega}t\,, \hat{x} = \bar{\omega}^2\,\frac{x}{g}\,, \hat{z} = \omega^2\,\frac{z}{g}$$

Por último se define el vector de estado x y el vector control u:

$$\dot{x} = \begin{pmatrix} \dot{\hat{x}} \\ \ddot{\hat{x}} \\ \dot{\hat{z}} \\ \ddot{\hat{z}} \\ \dot{\theta} \end{pmatrix} = f(x,u) = \begin{pmatrix} \dot{\hat{x}} \\ u_T \sin\theta \\ \dot{\hat{z}} \\ u_T \cos\theta - 1 \\ u_R \end{pmatrix}$$

Una vez obtenido el modelo, se realizan las operaciones necesarias para obtener el mínimo principio para conseguir controlar el quadcopter en tiempo óptimo (BVP).

Obtenida la ecuación:

$$\dot{x}_a = \int_a (t, x_a),$$
$$x^*(0) = x_0,$$
$$x^*(T) = x_T$$

El resolver el algoritmo consistirá en 3 pasos. Primero conseguir una maniobra que nos lleve desde el estado inicial actual al estado final deseado

usando la ecuación de optimización del tiempo computación (STO). La maniobra es obtenida variando la duración T y el tiempo que reciben los datos las entradas de control, hasta que el quadcopter alcanza el estado final con una aceptable precisión.

Segundo, basado en la maniobra resultante anteriormente y usando la constante *c = (c1, c2, c3, c4)* para la ecuación BVP explicada anteriormente. Estas ecuaciones son usadas para obtener la contante desconocida c.
Por último, teniendo una buena conjetura inicial sobre la duración T de la maniobra, la constante c obtenida en el paso anterior y la ecuación BVP obtendremos el control de vuelo.

2.2.1.2 Vuelo y transporte de quadcopters cooperativo a través de un controlador Fuzzy

Alex Kushleyev, Daniel Mellinger y Vijay Kumar son investigadores de la Universidad de Pennsylvania (GRASP Lab). Dichos investigadores centran su investigación en el desarrollo de algoritmos de control cuya finalidad es realizar un vuelo cooperativo con otros quadcopters, realizando formaciones de vuelo, trabajos simultáneos y solventando obstáculos conjuntamente [13].

Figura 2.7: *Vuelo en formación de quadcopters.*

Los quadcopter cuentan con una gran versatilidad a la hora de realizar tareas, si a esto le sumamos que dichos robots sean capaces de trabajar

conjuntamente y dispongan de los actuadores necesario para realizar tareas, estos se pueden convertir una herramienta claramente beneficiosa. Para el equipo del laboratorio de GRASP, era un objetivo fundamental el que los robots fueran capaces de transportar objetos de un punto a otro, pero ya que los quadcopters no disponen de una gran capacidad de carga, se buscó un método alternativo. Para ello se dispuso de múltiples quadcopter para cargar una única estructura.

Primero se definirá un modelo simple de quadcopter con un grupo de quadcopter rígidamente atados a la carta de pago que deseaban portar. Segundo se definirá el control de un único quadcopter, este realiza la operación de inclinación respecto a la carta de pago y estabiliza en 3 dimensiones la trayectoria a realizar.

A la hora de manipular objetos desde la altura se plantean una serie de problemas analizados en [14], [15] estos centraban su problema en conseguir un equilibrio estático en la carta de pago. El equipo de GRASP lab. busca un sistema diferente a la hora de transportar la carga.

Primero se generó el modelo, para ello se definió un marco de trabajo W. Donde B es la estructura incluyendo los quadcopter rígidamente anclados como se muestra en la Figura 2.5. Aunque los quadcopter este rígidamente anclados a la carga a transportar, cada quadcopter tendrá su propio marco de individual de trabajo Qi.

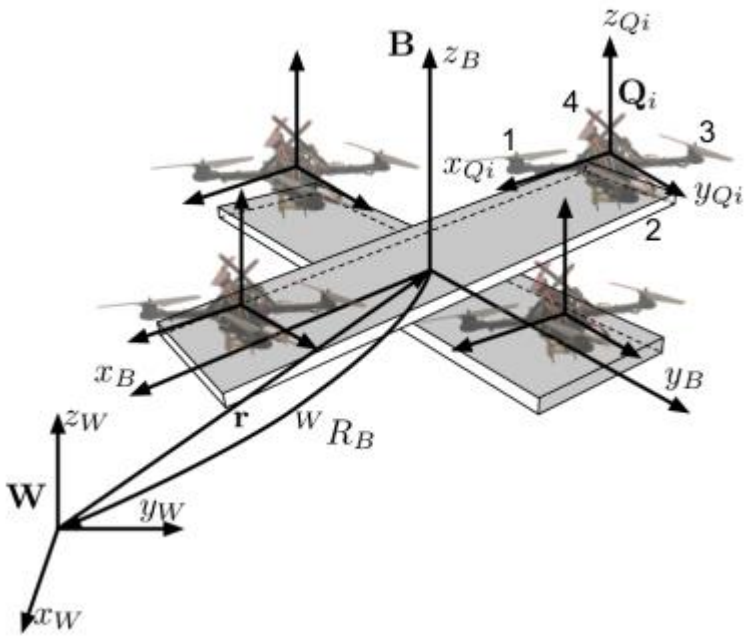


Figura 2.8: *Sistemas de coordenadas*

Para cada motor de cada quadcopter produce una fuerza independiente denominada $F_{i,j}$ y un momento angular $M_{i,j}$ en una dirección z_{Qi} . Todas las fuerzas de los motores pueden ser unificadas en una única fuerza [13].

$$\begin{bmatrix} F_{q,i} \\ M_{xq,i} \\ M_{yq,i} \\ M_{zq,i} \end{bmatrix} = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 0 & L & 0 & -L \\ -L & 0 & L & 0 \\ \frac{k_M}{k_F} & -\frac{k_M}{k_F} & \frac{k_M}{k_F} & -\frac{k_M}{k_F} \end{bmatrix} \begin{bmatrix} F_{i,1} \\ F_{i,2} \\ F_{i,3} \\ F_{i,4} \end{bmatrix}$$

Por último la matriz del momento de inercia de todo el sistema referenciado es denotada por *I*. Asumen que $x_B - y_B$ - z_B se eligen buscando que *I* sea diagonal. La aceleración angular es determinada por las ecuaciones de Euler [13]:

$$I \begin{bmatrix} \dot{p} \\ \dot{q} \\ \dot{r} \end{bmatrix} = \begin{bmatrix} M_{xB} \\ M_{yB} \\ M_{zB} \end{bmatrix} - \begin{bmatrix} p \\ q \\ r \end{bmatrix} xI \begin{bmatrix} p \\ q \\ r \end{bmatrix}$$

Para realizar el control de vuelo, usan un control proporcional y derivativo, basadas en las siguientes leyes:

$$M_{xB}^{des} = k_{p,\emptyset}(\emptyset^{des} - \emptyset) + k_{d,\emptyset}\,(p^{des} - p)$$

$$M_{yB}^{des} = k_{p,\theta}\,(\theta^{des} - \theta) + k_{d,\theta}(q^{des} - q)$$

$$M_{zB}^{des} = k_{p,\varphi}\,(\varphi^{des} - \varphi) + k_{d,\varphi}\,(r^{des} - r)$$

Para conseguir controlar la posición de x_W e y_W usan el ángulo de aladeo y cabeceo. Con M_{zB}^{des} se controlara el ángulo de guiñada del quadcopter y con F_B^{des} el control de posición a lo largo de z_w. Resolviendo la ecuación anteriormente planteada se obtiene la variable u, que contendrá las 4 entradas de control para cada quadcopter.

$$u = F_B^{des}\,u_F + M_{xB}^{des}u_{Mx} + M_{yB}^{des}u_{My} + M_{zB}^{des}u_{Mz}$$

Para mejorar el rendimiento del control, el equipo del laboratorio de Grasp descentralizo las leyes control de los quadcopters. Si se usa un sistema con un control centralizado, el sistema estima que los quadcopters encargados de portar la carta de pago son combinados creando una única entidad. Esto reduce el ruido en las estimaciones del control de la plataforma conjunta y consigue como resultado un control limpio.

Grasp lab centra su estudio en realizar una solución combinada centralizando y descentralizando las formulas. De esta manera, la posición, velocidad y orientación estimada de todo el control es calculada como un sistema centralizado. La velocidad angular es medida directamente de cada quadcopter, de esta forma las leyes de control depende de la velocidad angular que ha sido calculada usando un método descentralizado.

A la hora de definir el sistema de agarre la estructura a desplazar debe ser siempre horizontal aunque su distribución de pesos no necesita ser uniforme.

En la Figura 2.9 se representa como se distribuyen los quadcopter, el punto en el centro en las gráficas representa el error medio y las líneas representan la desviación.

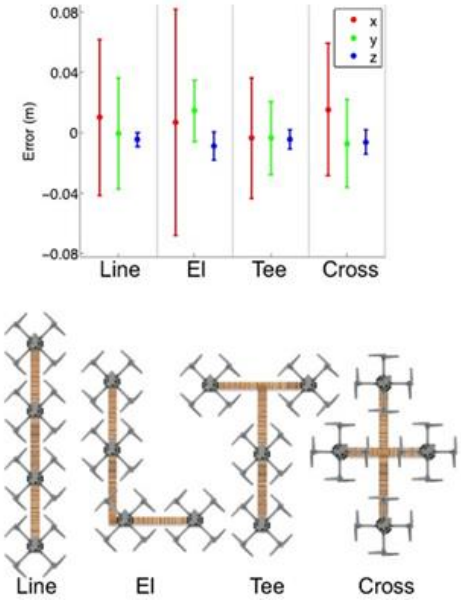


Figura 2.9: *Ordenaciones de quadcopter.*

Las pruebas experimentales se desarrollaron combinando el cálculo computacional para obtener el control de vuelo, realizado desde un PC y los datos recogidos desde el quadcopter (velocidad angular). El software usado fue el Matlab recogidos desde VICON. En la Figura 2.10 se muestra la primera prueba realizada, usando un grupo de 4 quadcopter rígidamente anclados a la carga de pago. Cada quadcopter pesa 500g y porta una carga de pago de 1.2kg. Los 4 quadcopter recogieron y transportar la estructura de madera con un peso de 320 gr una distancia de 0.8m. Como se aprecia en los diagramas se realizó con un vuelo estable a 0’5m/s. Una vez realizado el circuito los quadcopter posaron la estructura en el suelo finalizando así la prueba.

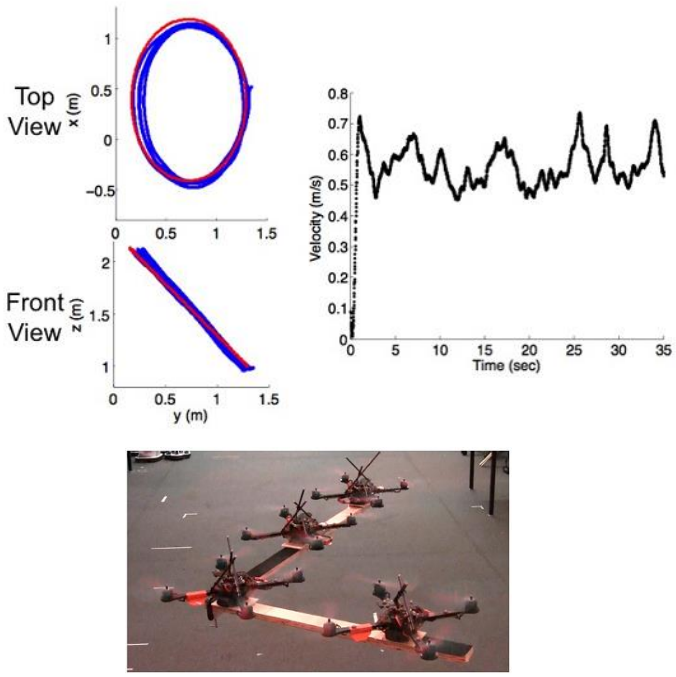


Figura 2.10: *Experimentación de vuelo con junto*

2.2.1.3 Cross- Entropy basado en un controlador Fuzzy

Existen otros modelos de enfoques a la hora de realizar el control de vuelo de los quadcopter, entre ellos el Cross-Entropy (CE) [9],[17], un método de optimización de un controlador Fuzzy. Este es un enfoque nuevo de optimización estadístico y simulación. Dicho método se desarrolló como un método eficiente para la estimación de probabilidades. Este estudio se realizó para obtener una optimación de un controlador fuzzy.

El proceso de optimización fue realizado usando el simulador ROS-Gazebo 3D, con una ampliación añadida para desarrollar el experimento.

El control visual se calcula a través del procesamiento de imágenes usando el algoritmo Camshift que permite la detección de objetos en la zona de trabajo.

Una vez realizadas las pruebas experimentales, el algoritmo fue trasmitido a un pequeño quadcopter.

El uso del método de Cross-entropy es una manera fácil de estimar si los resultados futuros en la plataforma real van a ser óptimos. Este método se basa en la captura de datos a través de la cámara equipada en quadcopter. Una vez capturada esta información es enviada junto con los datos de vuelo usando una comunicación wifi 802.11. Toda esta información es procesada "en tierra" y reenviada a la plataforma.

A la hora de detectar y caracterizar obstáculos, son diferenciados por colores según la proximidad y tamaño. Este algoritmo adopta una estrategia de adaptarse con el entorno constantemente (CamShift) [17]. La finalidad de este algoritmo es conseguir dejar en el borde de la imagen (izquierda o derecha) el color más representativo (rojo), que identifica los obstáculos.

En la Figura 2.11, se muestra el comportamiento que realiza el quadcopter con el algoritmo de CamShift.

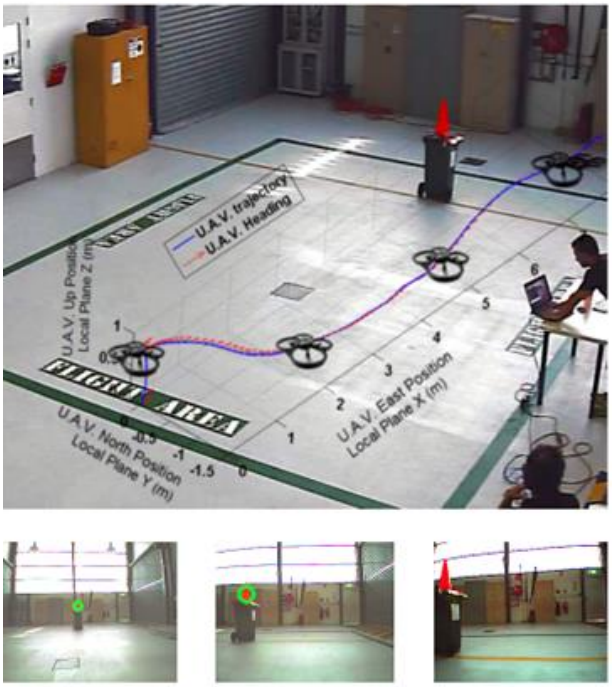
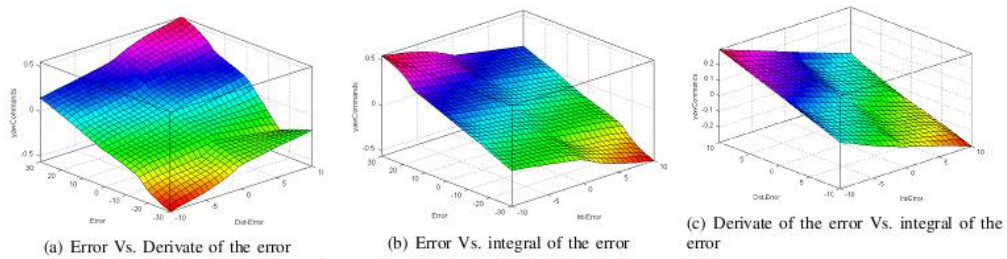


Figura 2.11: *Ejemplo de vuelo*

El objetivo del controlador Fuzzy es generar el control de estabilización basado en el plano de la imagen. Este es un controlador de 3 entradas y una salida.

La primera entrada mide el error en grados entre el quadcopter y la referencia (el objetivo detectado). La segunda entrada es la derivada del error, y la tercera entrad la integral del error. La salida es la inclinación desead para poder evitar el obstáculo.

Este sistema de lógica borrosa cuenta con 45 reglas. Para facilitar la comprensión del comportamiento Fuzzy, en la Figura 2.12 definimos como tercera dimensión la salida.



(a) Error Vs. Derivate of the error

(b) Error Vs. integral of the error

(c) Derivate of the error Vs. integral of the error

Figura 2.12: *Comportamiento fuzzy (Cross-Entropy)*

Para optimizar estos resultados se aplicar el método de Cross-Entropy(CE) . Este es un método iterativo y basado en la generación de datos de ejemplo aleatorios.

Primero inicializara el sistema, posteriormente generara un ejemplo de *N* controladores, realizara las operaciones de menor a mayor. Actualizara el computo de datos y repetirá desde el paso 2 hasta que converja o acabe. Cuando converjan los datos se habrá encontrado el valor óptimo.

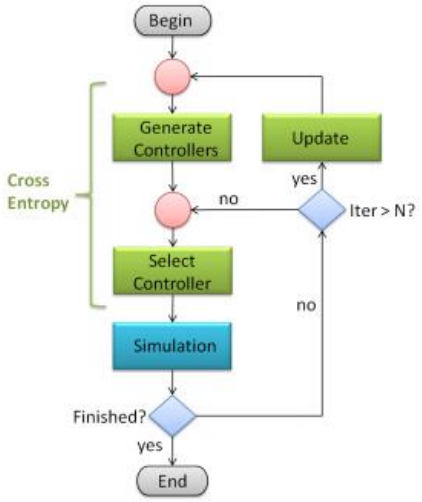


Figura 2.13: *Diagrama de flujo del proceso de optimización.*

Para comprobar si es funcionamiento de este algoritmo es correcto, se realiza la siguiente prueba de simulación. Para ello usan el ROS (Robotics Operative System) y el 3D simulation Gazebo.

A la hora de optimizar el proceso usando la simulación, se define una prueba de 3 segundos para cada controlador. El quadcopter empieza enfocado hacia el obstáculo. A los 5 segundos el controlador debe reorientar la plataforma dejando a la izquierda de la imagen el obstáculo.

Para evaluar cada test se usa el ITAE (Integral Time Absolute Error), y obtenemos el coste computacional.

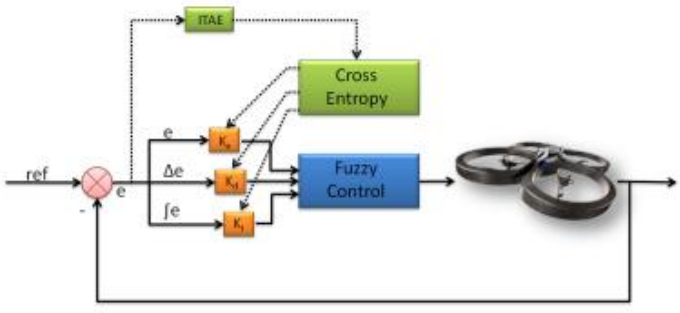


Figura 2.14: *Bucle de control con la optimización del método de CE.*

Los resultados finales de muestran que desarrollando un control de lógica borrosa y optimizándolo con el método de Cross-Entropy, se consigue reducir significativamente el número de reglas que permiten a la plataforma volar.

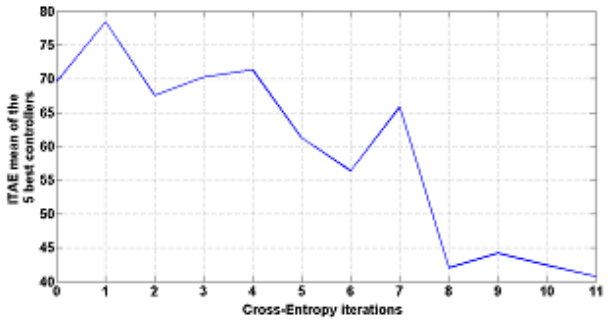


Figura 2.15: *Evolución ITAE durante 12 iteraciones del método CE*

2.2.1.4 Planificador de trayectorias adaptado a UAV basados en computación evolutiva.

Los sistemas de control y navegaciones basados en la planificación de trayectorias son comunes en la robótica, por lo que existen adaptaciones de este tipo de algoritmos eficientes a plataformas AUV [18].

Para realizar misiones de largo recorrido en UAV es importante que el sistema sea capaz de responder a situaciones dinámicas. A esto se suma que la respuesta del software debe ser eficiente para conseguir una plataforma estable. La planificación que se describirá a continuación junto al algoritmo de generación de trayectoria esta creado para conseguir una arquitectura completamente autónoma. El sistema además conseguir un control estable y ser capaz de enfrentarse a situaciones adversas, debe ser capaz de realizar una serie de tareas. En la Figura 2.16 se muestra el desglose de la arquitectura.

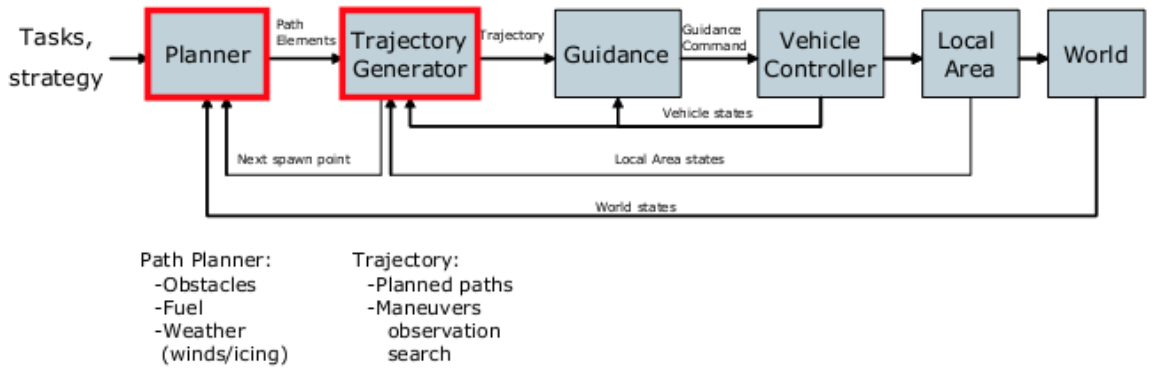


Figura 2.16: *Arquitectura de un sistema autónomo*

A la hora de buscar las soluciones para realizar las tareas, un posible sistema es realizar las tareas sin buscar la solución óptima, ya que esta suele tener un coste computacional alto. Para ello se usan las técnicas de Computación Evolutiva (EC), dichas técnicas se basan en conseguir una solución viable en corto plazo de tiempo. Dicha técnica resulta muy útil a la hora de proporcionar resultados para realizar el Path Planning donde tenemos un entorno complejo.

Par entender el funcionamiento del sistema, primero el algoritmo buscara una batería de soluciones aleatorios que propongan una soluciona factible. Una vez recogido estas posibles soluciones se seleccionaran las que posean mayores aptitudes con mayor población en dichas soluciones. Esto se repetirá cíclicamente hasta encontrar una solución.

A la hora de exportar este sistema a un UAV, requeriremos que nuestras soluciones puedan cambiar, es decir requerimos que nuestro Path Planning sea dinámico. Una ventaja del EC es que se adapta dinámicamente a los cambios sin necesitar de re calcular completamente una solución, por lo cual es computacionalmente eficiente.

El Path Planning centra su objetivo de trabajo en la región donde va a realizar la iteración con el medio. Como el sistema posee una planificación dinámica, este solo informa al vehículo de una parte de la planificación

realizada, es decir solo proporciona una trayectoria, si durante el vuelo se requiere una modificación de dicha trayectoria, esta no se necesita recalcular de forma completa, sino solo esa iteración, como se describe en la Figura 2.17

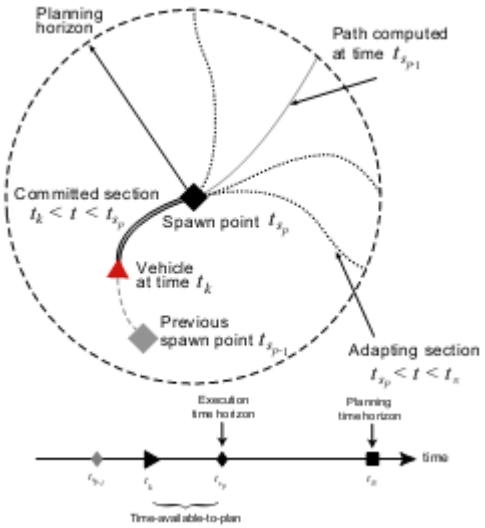


Figura 2.17: *EC con Path Planning*

Capítulo 3

Plataforma de vuelo

En el diseño propuesto en esta plataforma de vuelo ha variado progresivamente según se necesitaban nuevas funcionalidades o ampliaciones relacionadas con la introducción de nuevos componentes hardware. A continuación se describirán uno por uno las diferentes partes que compondrá el quadcopter, definiendo técnicamente cada componente y el porqué de su elección.

Posteriormente y a lo largo del resto del capítulo, explicare el desarrollo de la plataforma según se implementaban diferentes soluciones a los problemas que se presentaban.

Inicialmente la plataforma se diseñó basándose en otros modelos de quadcopter como los descritos [19], [20], pero ya que no se contaba con el mismo presupuesto y se buscaba un diseño propio que cumpliera los objetivos descritos anteriormente, se desarrolló una plataforma basándose en los quadcopter descritos y fue modificándose progresivamente.

3.1 Componentes hardware del quadcopter

En este apartado se van a describir los componentes que forman nuestra plataforma, definiendo especificaciones técnicas y porque se han comprado.

Arduino:

La placa Arduino Uno es el encargado de procesar todos los datos que recibe de los diferentes sensores a través de sus entradas analógicas y a su vez de transmitir las señales PWM a los motores a través de las salidas digitales.

Esta placa cuenta con un micro controlador ATmega328 con una frecuencia de reloj de 16MHz y 32k de memoria flash.

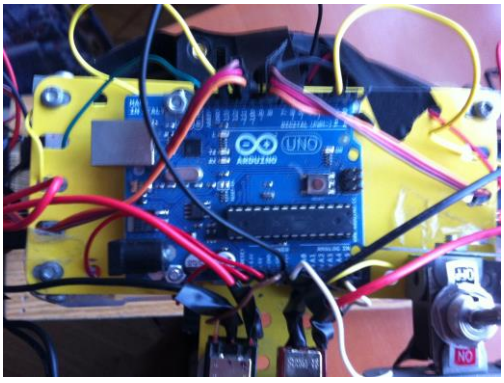


Figura 3.1: *Placa Arduino Uno.*

Arduino Uno cuenta con 6 entradas analógicas con las que recibe datos del giroscopio, acelerómetro, de los ultrasonidos, etc. Y a su vez con 14 salidas digitales (en en PWM) para transmitir datos, en nuestro caso, a los motores. Por último, Arduino Uno cuenta con una sola salida serie.

Aunque esta placa cuenta con una cantidad de entradas y salidas escasas, son las necesarias para realizar la tarea designada para este

proyecto, con lo que demuestra que es una placa compacta, de bajo coste y lo suficientemente potente como para realizar el control de vuelo.

Uno de los principales defectos que cuenta la placa Arduino Uno, es que no es capaz de trabajar con un comportamiento multi-Thread, con el grave problema que conlleva.

Arduino Mega 2560:

La placa controladora Arduino Mega, es una versión más potente de Arduino Uno. Esta placa dispone de 4 puertos UART, con un micro controlador ATmega2560 con una velocidad de reloj de 16 MHz. Cuenta con 54 pines digitales (14 con PWM) y 16 entradas analógicas.

Figura 3.2: *Placa Arduino Mega 2560.*

La causa de usar esta placa controladora reside en la IMU Razor 9. Dicha IMU requiere un puerto serie para la comunicación con Arduino. Ya que Arduino requiere 1 salida para el mismo, se requirió una placa con más de 1 salida serie.

Acelerómetro (ADXL 335):

El ADXL335 es un acelerómetro de 4 ejes que nos permite medir aceleraciones de rango de +3G -3G.

La interfaz para leer esta medida es un sensor analógico de 3 pines. Cada uno de estos voltajes es proporcional a la aceleración de cada eje. Este acelerómetro es alimentado con 3.3v.

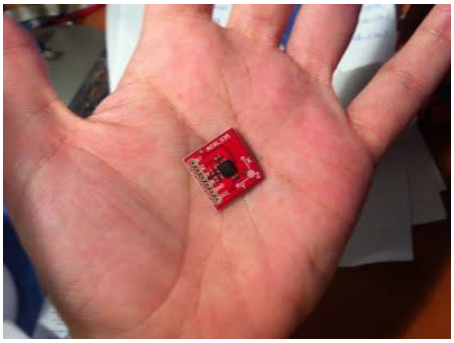


Figura 3.3: *Acelerómetro ADXL 335.*

Los acelerómetros nos proporcionaran la medida de las aceleraciones lineales que sufra el quadcopter. Esta será medida en la unidad "G".

Giroscopio (IDG 1215):

Este giroscopio nos proporciona la velocidad angular del eje X e Y. Es alimentado entre 3 y 7v y posee una sensibilidad de 15mv por cada grado/sg. Su rango de trabajo está comprendido entre -67 grado/sg y +67 grado/sg. Dichos datos son enviados en el paso bajo de la señal.

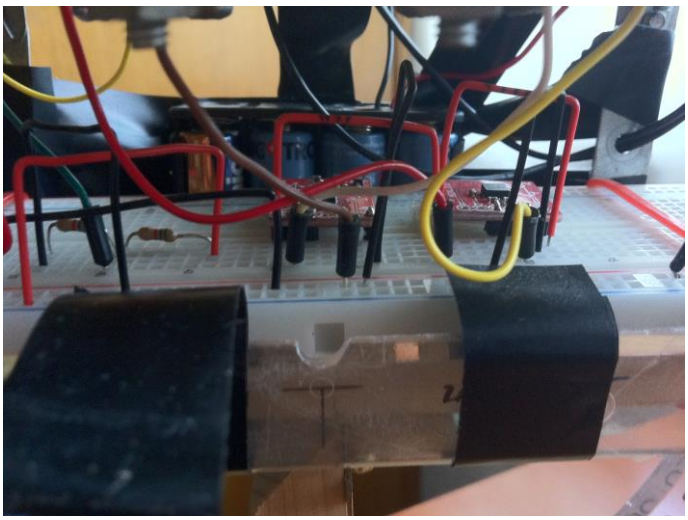


Figura 3.4: *Giroscopio IDG 1215.*

IMU Razor 9 y chip USB-Serie:

El sistema de medición inercial 9DOF Razor IMU dispone de 3 sensores: un giroscopio de 3 ejes ITG3200, un acelerómetro ADXL345 de 3 ejes y un compás/magnetómetro HMC5883L de 3 ejes.

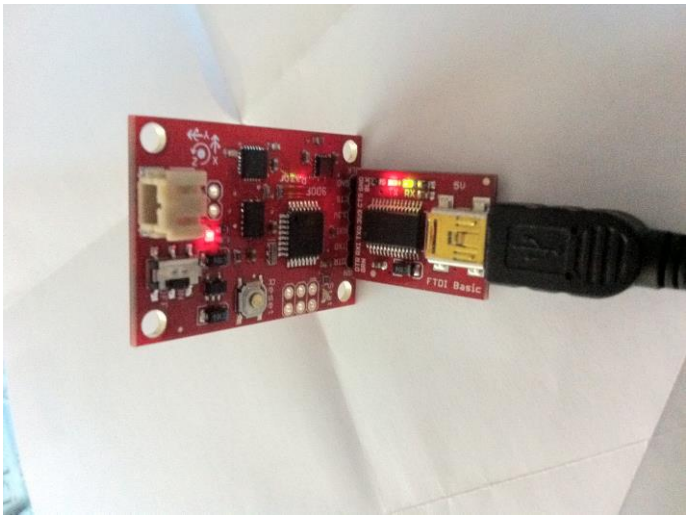


Figura 3.5: *IMU Razor 9 (izquierda), chip USB-Serie (derecha).*

El conjunto proporciona 9 grados para medición inercial. Todas las salidas de los sensores son procesadas por un ATMega328 que envía a su vez la información por un puerto serie UART para poder recuperarlos de forma rápida.

La velocidad de transmisión de los datos está fijada originalmente en 38400bps, pero se mejoró por software hasta los 57600 bps.

Gracias a esta IMU obtendremos los valores de X, Y y Z, tratando los datos recibidos por los sensores a través del algoritmo de DCM.

Ultrasonidos:

La finalidad de los ultrasonidos es controlar la distancia al suelo a la hora de aterrizar.

Para realizar esta tarea se estudiaron 2 modelos de ultrasonidos: Los primeros emiten y escuchan cada "X" microsegundos. La segunda versión, emiten constantemente señal y tú decides cuando deseas escuchar. Se decidió la segunda versión ya que estar emitiendo y recibiendo consumía tiempo de la CPU del micro controlador y de esta manera solo se activaría la escucha del ultrasonido cuando el barómetro nos indique que estamos cerca de la primera medida de altitud tomada.

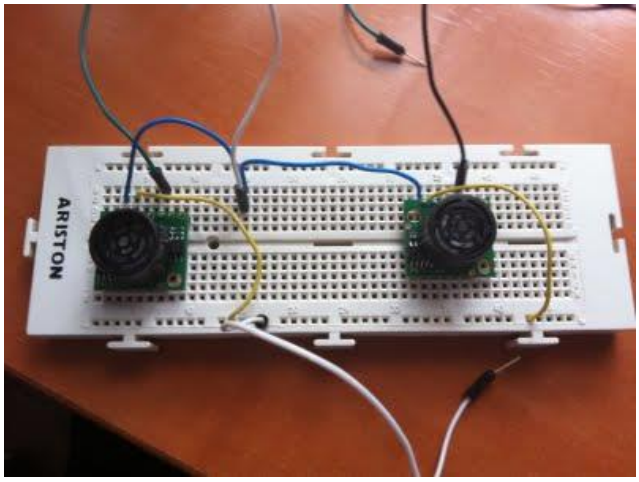


Figura 3.6: *Ultrasonido LV-MaxSonar-EZ0.*

El modelo es el LV-MaxSonar-EZ0. Dicho ultrasonido muestrea una distancia de hasta 6 metros con un ancho de onda de hasta 3 metros.

Arduino Protoboard:

La protoboard de Arduino se incluyó principalmente para reducir el peso en el quadcopter y para reducir el ruido de la señal con los cables.

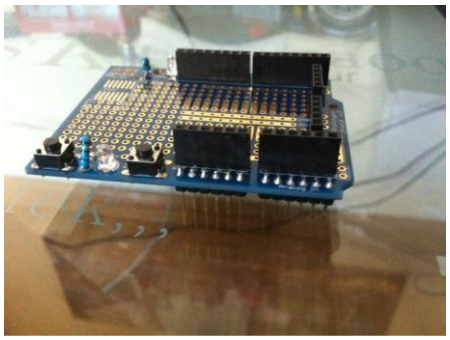


Figura 3.7: *Arduino Protoboard.*

GPS/GPRS/GSM Module V2.0:

El modulo combina la tecnología de navegación GPS y de comunicación GSM, esta última trabaja a 850 MHz. EL modulo GPS posee una precisión de 10 metros aunque es capaz de mejorar su precisión si se activa su modo AGPS (GPS asistido por las antenas de telefonía).

Figura 3.8: *GPS/GRPS/GSM Module V2.0.*

Variador Bushless:

Este componente hardware es el encargado de recibir la señal PWM del micro controlador Arduino y transformarla en voltaje.

Dicho componente tiene una frecuencia de entrada de PWM de 8 KHz y es capaz de proporcionar una corriente media constante de 20A a picos máximos de 25A

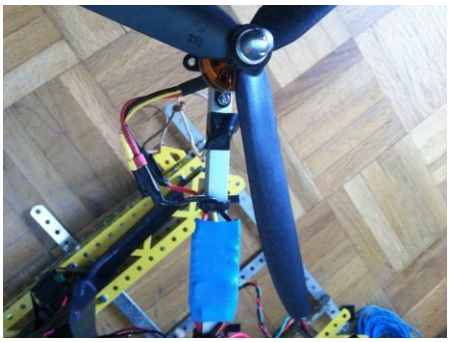


Figura 3.9: *Variador Brushless.*

Motores Bushless:

Los motores empleados para el vuelo son 4 motores de 1490kv. Dichos motores cuentan con una corriente máxima de entrada de 12A/60sg.

El variador elegido anteriormente admite hasta un máximo de 20 A, lo que permitirá al variador no está en su límite y sobrecalentarse y al motor no sufrir daños.

Figura 3.10: *Motor Brushless.*

Sensores de carga:

Las baterías Lipo tiene una carga de 11.1v, dichas baterías se dañan cuando caen por debajo de un umbral de carga (incluso llegando a explotar). El sistema de control usado para controlar el nivel de carga, son 2

módulos que unen un sistema de led y un piezoeléctrico para informarnos de la capacidad de la batería.

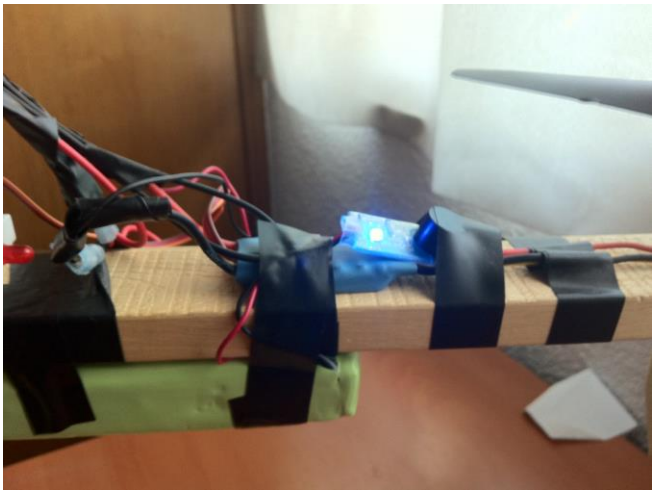


Figura 3.11: *Sensores de carga.*

Baterías Lipo:

La batería usada para proporcionar energía a los motores es una batería Lipo de 2200mAh de 11’3v. El tiempo medio de duración de esta batería es de 15 minutos.

Figura 3.12: *Batería Lipo*

3.2 Desarrolo del diseño de la plataforma

Para desarrollar un quadcopter, se realizó una investigación previa de la dificultad que planteaba construir un quadcopter desde cero. Una vez realizada se dedujo que careciendo de los conocimientos previos de control y aerodinámica, la mejor solución fue simplificar al máximo el problema para poder abordarlo de la forma más óptima.

Inicialmente se realizó una toma de contacto con una plataforma más sencilla de solo 1 eje, reduciendo así la complejidad del problema en gran medida.

Dicha construcción inicial se realizó con materiales de bajo coste, sencillos de obtener y fáciles de manipular.

3.2.1 Plataforma de un eje

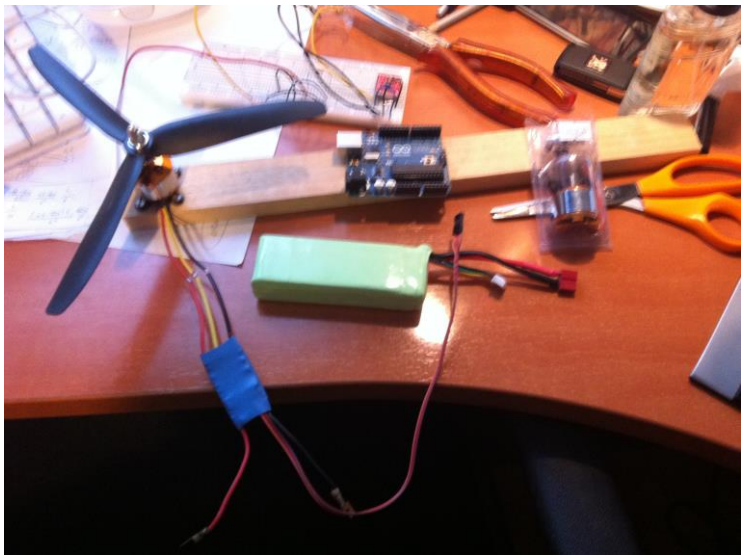


Figura 3.13: *Primera versión de la plataforma*

Para ello se ha construido una plataforma basada en la de desarrollada por *miKuadripcopter* [19]. Primero, se compró un juego de motores brushless, 2 variadores brushless y el Arduino Uno.

Se realizaron una serie de pruebas de fuerza y respuestas para conocer el peso que podía levantar, la frecuencia de envió, recepción y muestreo de Arduino. Comprobado que eran satisfactorias, se procedió a construir la plataforma descrita anteriormente [19].

Dicha plataforma posee una forma similar a un "balancín". La primera versión se usó una fuente externa para alimentar a los motores, pero aunque el cable estaba holgado falseaba el comportamiento del quadcopter para mantenerse estable en la estructura de pruebas mostrada en la Figura 1.2.

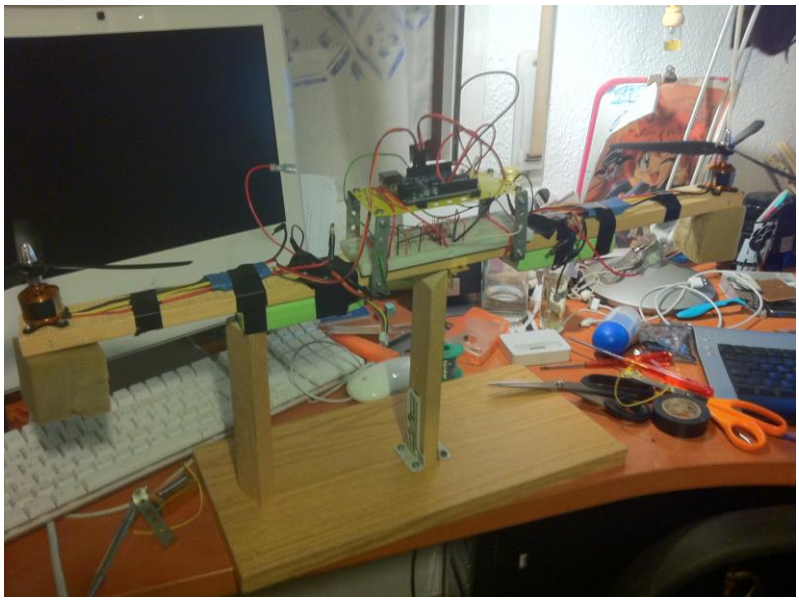


Figura 3.14: *Estructura de pruebas*

Para solventarlo se añadieron 2 baterías Lipo por cada motor y así conseguir un comportamiento completamente autónomo.

En la parte superior se situó el micro controlador y en la parte inferior la protoboard con el acelerómetro, el giroscopio y una serie de sensores para identificar el comportamiento del algoritmo de control.

Para finalizar esta primera aproximación se le añadieron un sistema de interruptor para cortar la corriente en caso de seguridad.

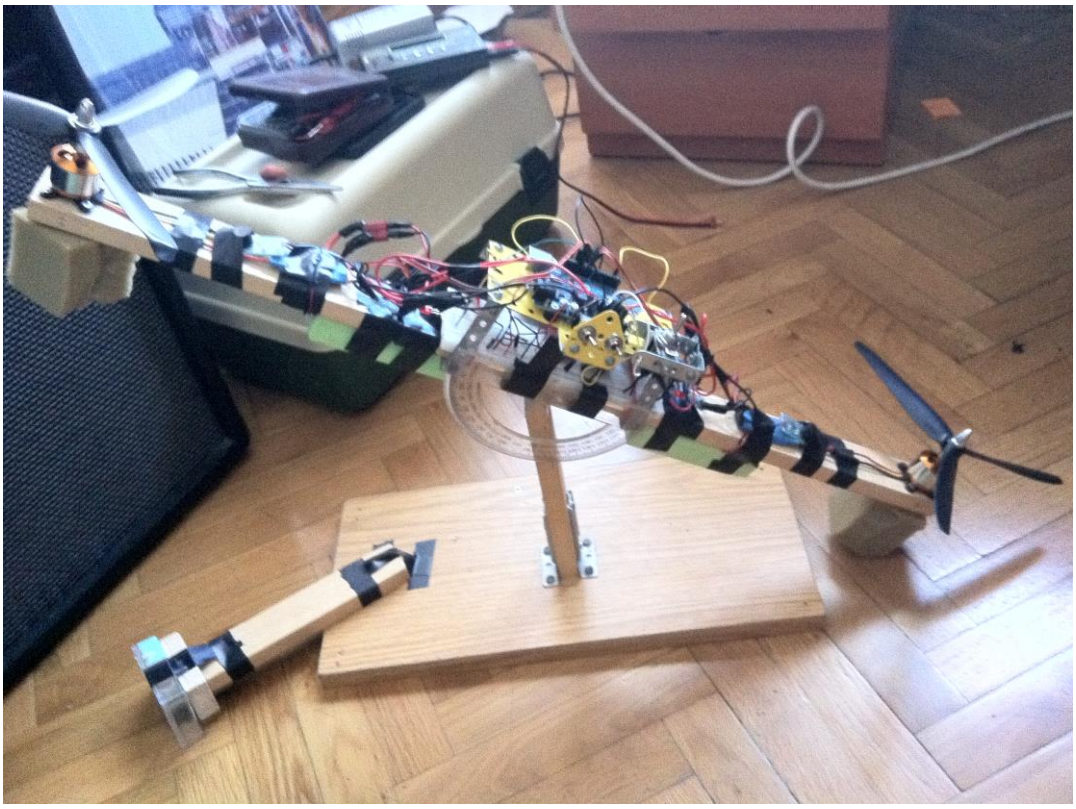


Figura 3.15: *Plataforma final de un eje*

3.2.2 Plataforma de tres ejes

Se estudiaron diferentes versiones chasis de tipo quadcopter para la finalidad que se buscaba. Al final me decante por una versión mixta entre aluminio y plástico. Se buscaba un chasis robusto, poco pesado y que tuviera capacidad para almacenar todo el hardware usado.

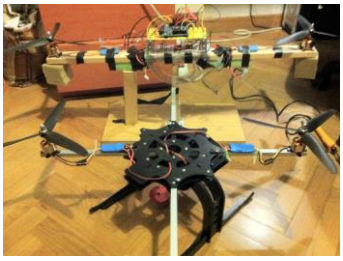


Figura 3.16: *Proceso de migración*

Acabada la migración, se añadió una plataforma de pruebas de despegue anclada por gomas al final de cada brazo y unido al techo por un cable central. De esta manera se buscaba que el algoritmo de control de vuelo no necesitara controlar el eje Z.

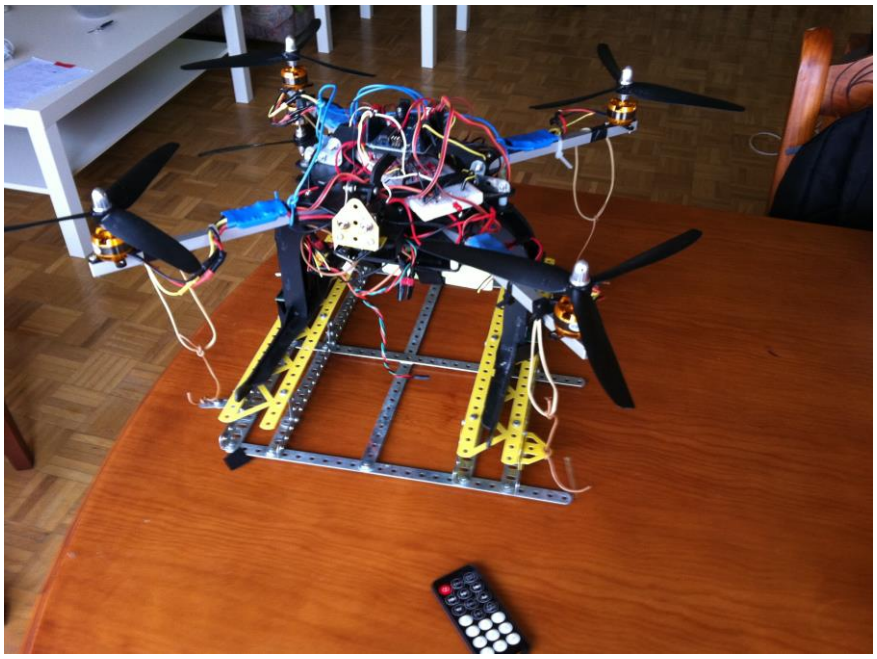


Figura 3.17: *Migración completada de la plataforma*

A su vez se añadió un control de apagado de emergencia por infrarrojos.

Realizadas las pruebas pertinentes con esta nueva plataforma, se detectaron una serie de problemas.

1) La plataforma pesaba demasiado, al intentar realizar el despegue, los motores trabajaban casi al 90% de su capacidad, por lo que conseguir corregir era inviable.

2) EL consumo de la CPU del micro controlador en la emisión y recepción del sensor infrarrojos provocaba que los motores perdieran un tiempo de corrección que impedía conseguir el equilibrio.

Para solventar estos problemas, la plataforma fue desmontada nuevamente, las modificaciones fueron las siguientes:

a) Se eliminó el tren inferior para aligerar peso.
b) Se eliminaron 2 de las 3 baterías.
c) Se eliminaron todos los componentes metálicos y la protoboard.
d) Se unifico todo el control del quadcopter sobre la misma base, usando la protoboard de Arduino. Esta ahorraba espacio y reducía ruido en la señal.

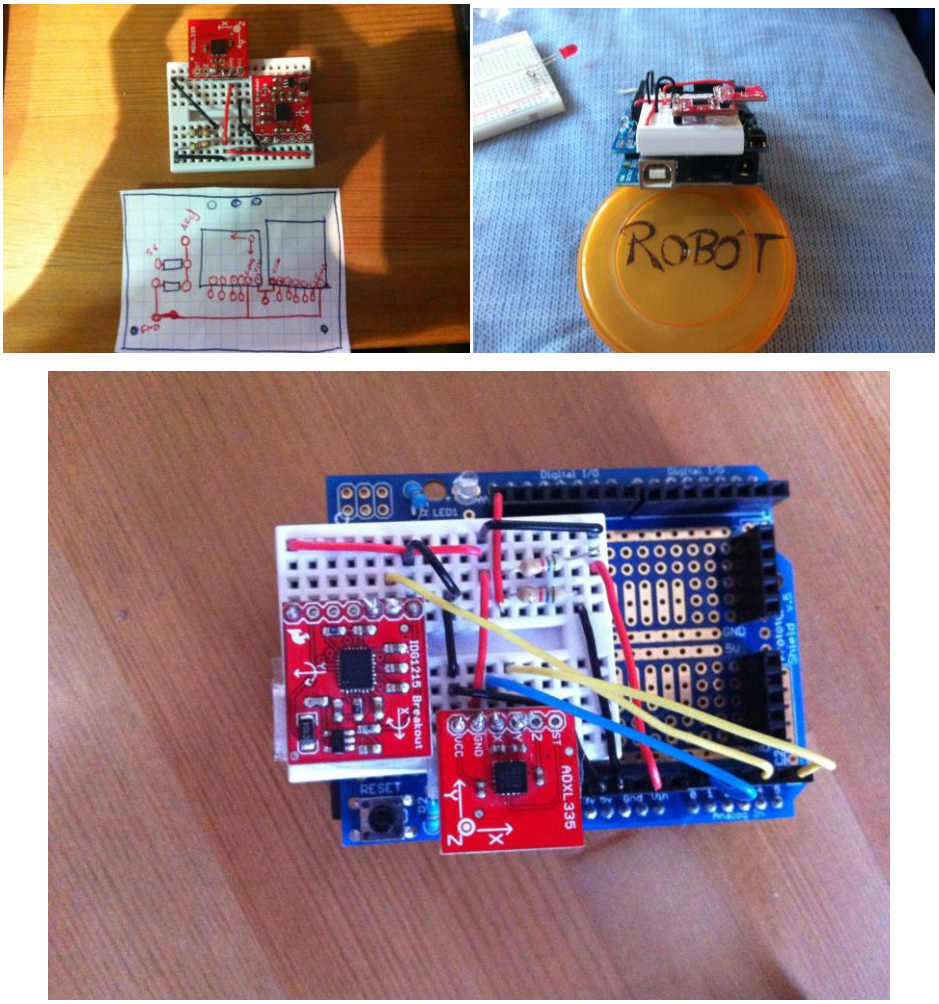


Figura 3.18: *Protoboard Arduino*

Una vez realizadas las modificaciones en el chasis, la antigua plataforma de pruebas no se podía usar ya que ya no contábamos con el tren de aterrizaje. Para ello se construyó una nueva plataforma que controlara con mayor grados de libertad y a su vez menos factores que interfirieran en el vuelo (como por ejemplo el cable USB que enlazaba el quadcopter con el PC, ahora estaba sujeto a la parte superior de la estructura lo que no condicionaba el vuelo).

Figura 3.19: *Estructura de pruebas de tres ejes*

El resultado es el siguiente:

Figura 3.20: *Plataforma modificada*

Realizando las respectivas pruebas de esta nueva plataforma se detectaron los siguientes problemas:

1) Durante las pruebas, el giroscopio empezó a presentar medidas erróneas.

2) El centro de gravedad no estaba centrado lo que provocaba que los motores giraran a revoluciones diferentes para compensar este sobrepeso, lo que dificultaba el algoritmo de control.

Se estudió si era mejor comprar el mismo componente o buscar otro componente que nos proporcionara mejores resultados. Tras varias comparativas entre diversos componentes hardware se llegó a la conclusión de invertir algo más y comprar una IMU con su propio procesador.

De esta manera el micro controlador de la IMU se encargaría de procesar los datos de los 3 sensores de los que dispone y enviar los datos en grados de la inclinación en la que se encuentra el quadcopter.

Para solventar el segundo problema, se desarmo el quadcopter. La plataforma fue reconstruida con el centro de gravedad en el centro y situándolo lo más abajo posible.

Otras modificaciones fueron cambiar el sistema de vuelo y la inclusión del módulo de GPS.

Se cambió la posición imaginaria de lo que representaba la parte delantera del quadcopter. Antes situado entre 2 motores y ahora sobre un único motor (como se muestra en la Figura 3.21).

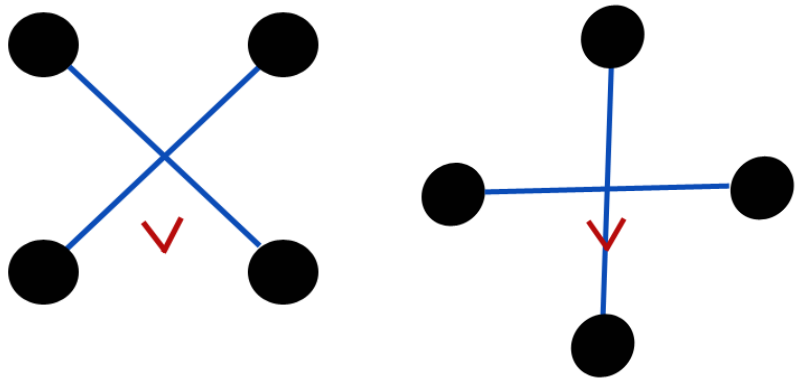


Figura 3.21: *Redistribución de la plataforma*

Con este cambio se conseguía que antes, para poder corregir el desequilibrio en el eje X, debía modificarse la potencia en los 4 motores. Ahora, solo se necesitaría corregir en 2, reduciendo la complejidad del algoritmo de control y evitando la posibilidad de desequilibrar el eje Y.

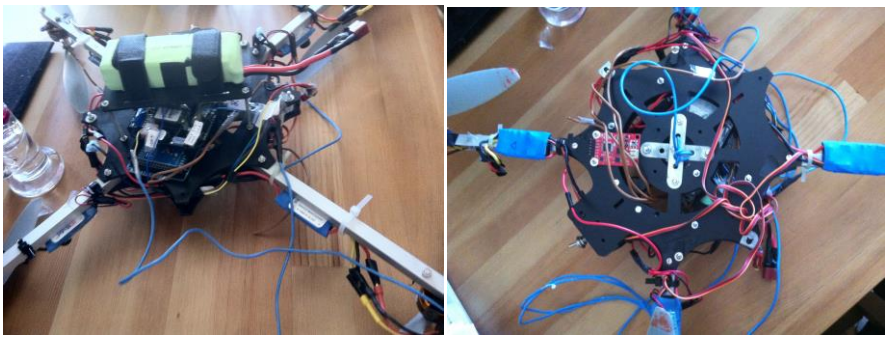


Figura 3.22: *Plataforma final*

Capítulo 4

Diseño e implementación del controlador

En este capítulo se describirán 2 algoritmos para el control de vuelo.

El sistema de control está formado por una máquina de estados como se muestra en la Figura 4.1. En primer lugar inicializa el sistema calculando inclinación inicial (con el cual obtiene su ángulo cero de inclinación). Posteriormente realiza el comportamiento establecido, ya sea el protocolo de despegue, mantenerse en equilibrio dada una posición GPS, enviar coordenadas vía GSM o desplazarse.

Una de las ventajas obtenidas con respecto a otros sistemas expuestos en capítulos anteriores, es que el sistema de control implementado (PID) se realiza en el propio quadcopter, lo que nos independiza de un sistema externo como un PC. Otra ventaja del segundo sistema de control descrito (controlador Fuzzy), es que al ser un sistema non lineal permite amoldarse al control de la plataforma aunque esta varíe en estructura.

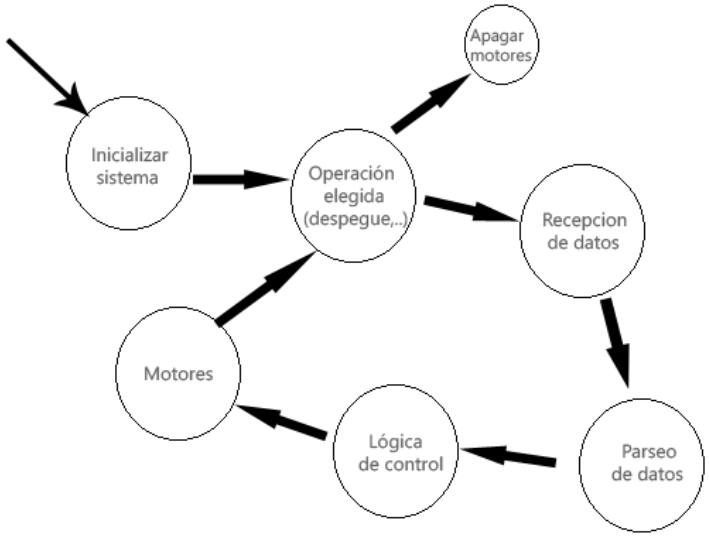


Figura 4.1: *Maquina de estados del control del quadcopter*

Durante la mayor parte del desarrollo de la plataforma de vuelo, el algoritmo usado ha sido el PID ya que casi cumplía con los objetivos marcados, a su vez se realizó la investigación de un segundo sistema de control basado lógica fuzzy.

4.1. Controlador PID

El algoritmo PID (Proporcional, Integral, Derivativo), es un conocido algoritmo de control para sistemas lineales. La finalidad de este algoritmo es conseguir que tu dato de salida sea igual al dato de referencia que introduzcas. A su vez constará de 3 constantes (K_p, K_i, K_d), que se deberán definir según el comportamiento deseado.
La ley de control PID viene dada por la siguiente ecuación:

$U_{PID} = K_p$*error + K_i * (Sumatorio de errores) + K_d* de/dt
error = Angulo de referencia – ángulo actual

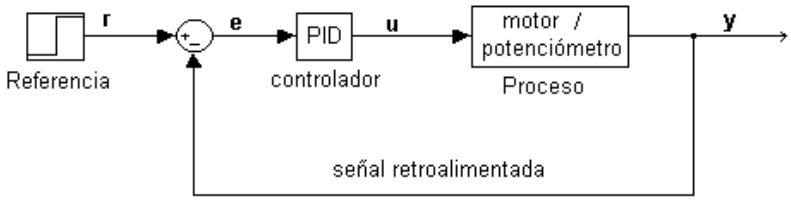


Figura 4.2: *Esquema de comportamiento del controlador PID*

Dicho algoritmo cuenta con una entrada y una salida. La entrada era el error obtenido, de la diferencia entra la señal de referencia que deseamos obtener y la señal que estamos recibiendo (el ángulo actual). La salida será la potencia que necesitaremos aplicar al motor para corregir la desviación del error que hayamos obtenido.

La salida mencionada anteriormente se genera a través de las 3 partes que componen el algoritmo, la parte Proporcional, la Integrativa y la Derivativa.

La parte proporcional se obtiene de multiplicar el error de entrada por la constante definida anteriormente (K_p). Esta parte es la que se define como acción-reacción.

La segunda parte es la integral. Se obtiene de multiplicar la integral del error por la constante K_i. Esta parte del algoritmo se calcula sumando los errores que hemos ido obteniendo, con la finalidad de compensar el error hasta cancelarlo.

La última parte, la derivativa, se obtiene de multiplicar la derivada del error por la constante denominada K_d. La finalidad de este es corregir la sobreactuación que hayamos aplicado para no tener oscilaciones, ya que se anticipa a la salida para disminuir la salida que hayamos obtenido con las 2 anteriores partes.

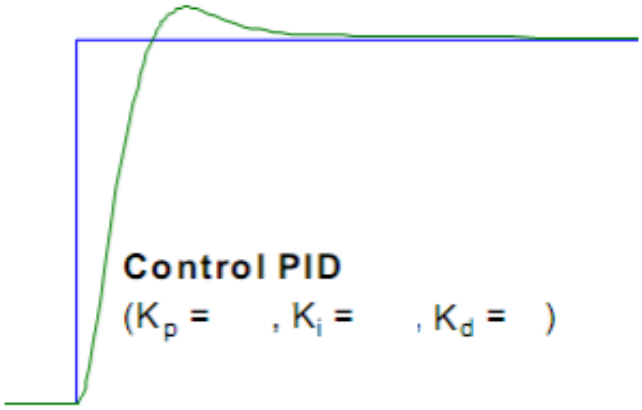


Figura 4.3: *Ejemplo de comportamiento PID*

4.1.1 PID adaptado a nuestro quadcopter

La primera implementación del PID, fue sobre la plataforma de un solo eje. El comportamiento fue el esperado, consiguiendo un equilibrio a cero grados. Las pruebas que se realizaron simularon efectos de viento constante y ráfagas de viento, siendo capaz de recuperar la posición de equilibrio.

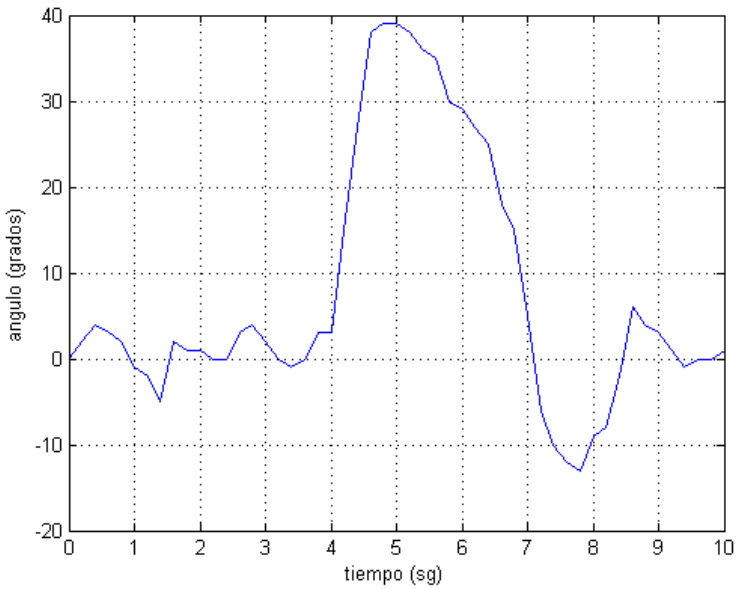


Figura 4.4: *Controlador PID en un eje*

Como se muestra en la figura anterior, el quadcopter mantiene una estabilización aproximada a 0 grados, hasta que sufre una fuerza externa (se fuerza el desequilibrio), provocando la desestabilización de la plataforma y su consiguiente corrección. Una vez finalizadas las pruebas con la plataforma de un eje se realizó la migración a la plataforma de tres ejes.

Cuando se implementó el algoritmo en la plataforma que soportaba el vuelo en 3 ejes, se diseñó un PID por cada eje tomando en consideración el acoplamiento en los ejes.

Una vez unificados los 3 PID, el comportamiento a la hora de estabilizarse era incorrecto, ya que nuestro sistema es un sistema non lineal de tipo *multiple inputs multiple outputs* (MIMO) y acoplado no era capaz de corregir y volver a la zona de equilibrio.

Este problema mencionado anteriormente es la mayor desventaja del uso del control PID, ya que al no ser un sistema non lineal, al salirse de nuestra zona de control, este perdía su eficacia.

Para solventarlo se ha propuesto el diseño de un controlador modificado basado en el algoritmo PID. Este PID ha sido modificado para tener en cuenta el acoplamiento de entradas del sistema.

Para ello, primero se diseñó un sistema de puntos, por el cual según la respuesta del PID del eje que se deseara, se le asignaba una cantidad de puntos, hasta un máximo de 10 por eje. Sumamos todos los puntos y obtenemos los puntos de cada motor.

Estos puntos se transforman en señales PWM que son enviados al motor.

Usando este algoritmo no se conseguían los resultados deseados, ya que en los casos en los que uno de los ejes estuviera en equilibrio recibía la misma importancia que los demás ejes.

Para ello se propone una solución basada en la asignación de pesos. Con este método si un eje está en equilibrio tendrá 0 de peso a la hora de

corregir y el peso de ese eje, seria asignado al otro eje, para conseguir una corrección más rápida.

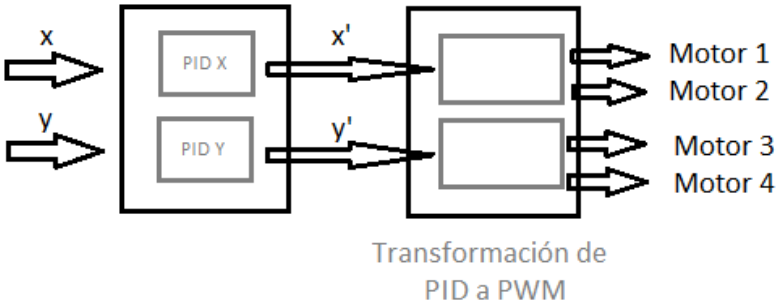


Figura 4.5: *Descripción del controlador PID*

$$U_x + U_y = U_T$$

$$PWM_{M1} = \frac{\Delta_1 * U_x}{U_T} + U_o$$

$$PWM_{M2} = \frac{\Delta_1 * U_x}{1.2 * U_T} + U_o$$

$$PWM_{M3} = \frac{\Delta_1 * U_y}{U_T} + U_o$$

$$PWM_{M4} = \frac{\Delta_1 * U_y}{1.2 * U_T} + U_o$$

U_x,U_y : Controlador PID en X e Y.
U_T: Suma de valores de los controladores PID.
PWM_{M1}, PWM_{M2}, PWM_{M3}, PWM_{M4}: Potencia enviada al motor1, 2, 3 y 4.
Δ_1 : Diferencia entre potencia máxima de los motores y la de equilibrio.
U_o: Potencia de equilibrio.

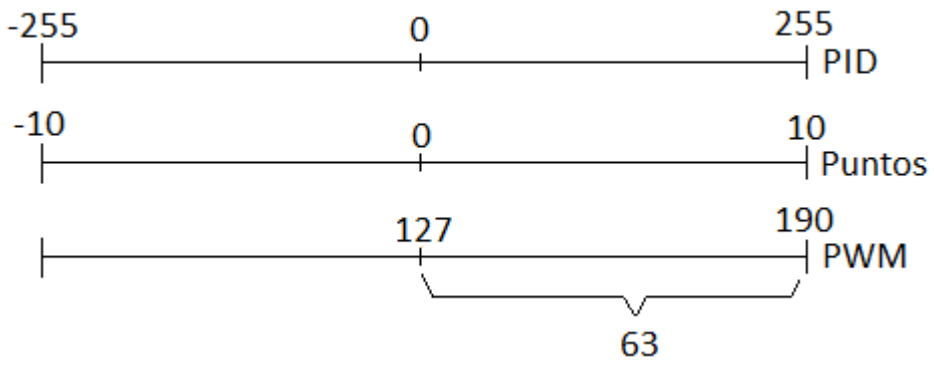


Figura 4.6: *Conversión de datos*

La implementación del PID en código Arduino fue la siguiente:

```
//------ X ----
errorAnguloProp = setPoint - angulo;
errorIntegral =  errorIntegralAcumulado + errorAnguloProp ;
errorDerivativo = anguloAnt - angulo;
//------ Y ----
errorAnguloProp_T = setPoint_T - angulo_Y;
errorIntegral_T =  errorIntegralAcumulado_T + errorAnguloProp_T;
errorDerivativo_T = anguloAnt_T - angulo_Y;
//------ PID ----
pid = kP*errorAnguloProp + kI*errorIntegral + kD*errorDerivativo;
      pid_T = kP*errorAnguloProp_T + kI*errorIntegral_T + kD*errorDerivativo_T;
              …
double numPuntos = auxPid * 0.04;
  puntosMaxX = numPuntos;

  if (pid >0){
    puntosMotor9 += numPuntos;
    puntosMotor10 += numPuntos/1.5;
  }
  else{
    puntosMotor9 += numPuntos/1.5;
    puntosMotor10 += numPuntos;
  }
               …


veloDelantero11 = ((puntosMotor11 * difTrasero) / sumaDePotMax) + eqTraseros;
velotrasero3 = ((puntosMotor3 * difDelantero) / sumaDePotMax) + eqTraseros;
veloIzq10 = ((puntosMotor10 * difTrasero) / sumaDePotMax) + eqTraseros;
veloDer9 =  ((puntosMotor9 * difDelantero) / sumaDePotMax) + eqTraseros;
```

En algunos casos los datos proporcionados por la IMU, contienen errores, como por ejemplo recibir datos incorrectos, para controlar esto, se implementaron 2 filtros:

Filtro de cancelación de errores de medida, para ello si el ángulo obtenido es superior a 100º, se utilizaba el ángulo anterior. Este tipo de datos errores son causados por el ruido en el canal de datos de la IMU a Arduino.

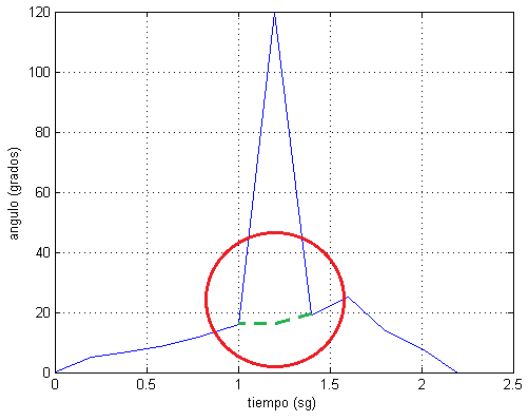


Figura 4.7: *Filtro de cancelación de errores de medida.*

Filtro atenuador de perturbaciones externas, en el caso que la diferencia de ángulos entre, el ángulo actual y el ángulo anterior es superior o inferior a 20º, se obvia ese dato y se utiliza el anterior.

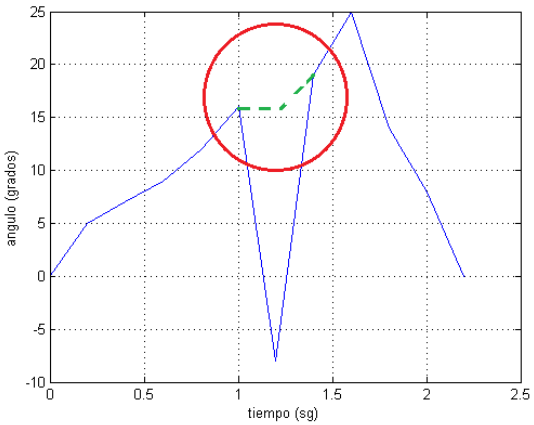


Figura 4.8: *Filtro de cancelación de perturbaciones externas.*

4.1.1 Experimentaciones

A la hora de realizar las pruebas sobre este tipo de control, se realizaron un total de 20 simulaciones en la plataforma real. El quadcopter consigue estabilizarse dentro del are de trabajo satisfactoriamente.

En la Figura 4.9 el comportamiento normal sin perturbaciones externas, como se aprecia en la figura el controlador corrige la inclinación del quadcopter hasta conseguir un equilibrio cercano a 0 grados.

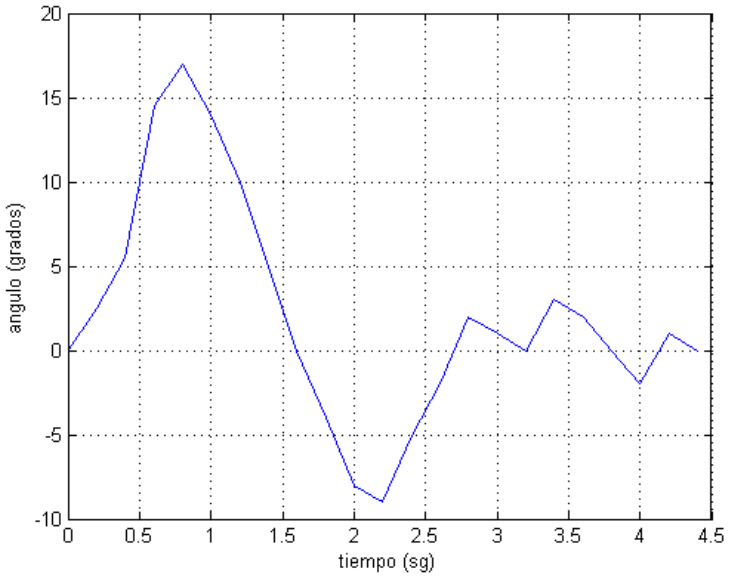


Figura 4.9: *Evolución del ángulo del eje X.*

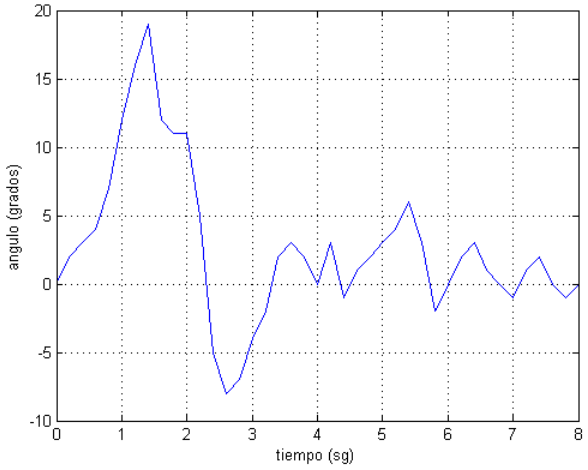


Figura 4.10: *Evolución del ángulo del eje X.*

En el caso de la Figura 4.11, golpeamos al quadcopter en la parte frontal provocando una inclinación en el eje Y.

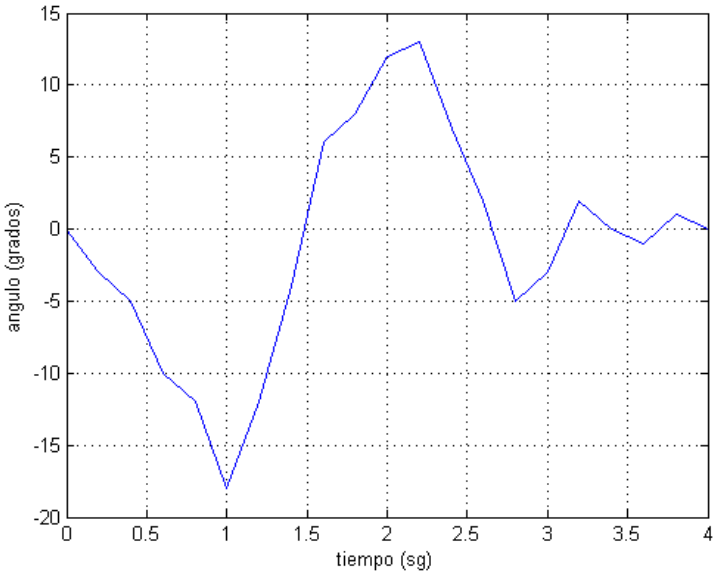


Figura 4.11: *Evolución del ángulo del eje Y.*

Dado los datos aportados, definimos que los resultados obtenido son aceptables, pero buscando conseguir optimizar el control del quadcopter, se realizó un estudio sobre los sistemas non lineales, en este caso un controlador Fuzzy.

4.2. Control Fuzzy

El controlador Fuzzy ha demostrado ser una herramienta útil para aquellos sistemas que presentan un modelo matemático difícil de obtener o en el que existe una capacidad de cómputo limitada [21],[22],[23],[24],[25].

Uno de los mayores problemas en las teorías de los sistemas dinámicos es la necesidad de garantizar la estabilidad en todo momento. El problema se agrava cuando el control del sistema es involucrado.

Aunque los controladores Fuzzy han sido desarrollados hace tiempo y han sido satisfecho las necesidades del control correctamente se han continuado realizando mejoras en este control recientemente [26],[27],[28],[29]. [30], [31], [32].

Estos sistemas han dado lugar a nuevos enfoques prácticos y de relativa facilidad de comprensión para el control de sistemas no lineales complejos.

También se utilizan los controladores Fuzzy para hacer frente a los procesos complejos que varían su linealidad en el tiempo (LTV). Los sistemas LTV también pueden ser el resultado de linealizar los sistemas no lineales durante un periodo de tiempo durante una trayectoria.

El sistema de control mencionado anteriormente (PID), es muy usado para sistemas lineales, y ya que nuestro sistema es un sistema no lineal se necesita estrategias de control que pueden manejar la non linealidad de dicho sistema. Se decantó por el desarrollo de un controlador Fuzzy para controlar el sistema no lineal. El enfoque del control Fuzzy consiste en dividir la dinámica del sistema non lineal en varios subsistemas interpolados entre ellos a través de funciones non lineales que se llaman las funciones de partición, después asignar a cada subsistema un regulador.

Para poder realizar el controlador Fuzzy, necesitábamos como prerrequisito, los datos de entrada que van a ser tratados pro nuestro controlador. Dichos datos de entrada son en ángulo actual a tiempo real del eje X e Y. El encargado de proporcionarnos estos datos es el micro controlador Arduino. Para facilitar la compresión de la obtención de estos

datos, se ha incluido una figura que se muestra a continuación con la comunicación bidireccional entre Arduino y Matlab.

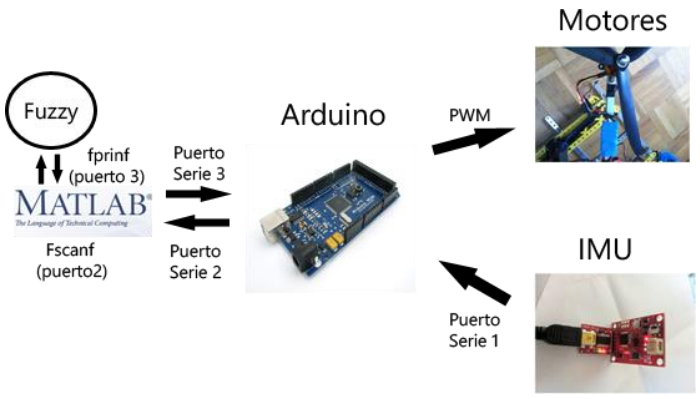


Figura 4.12: *Comunicación bidireccional Arduino-Matlab*

En primer lugar se transmiten los datos recibidos desde la IMU, que son recibimos por el puerto serie1. Estos datos son transmitidos por el puerto serie2. Desde Matlab recibimos los datos abriendo un canal con el mismo puerto abierto desde Arduino y con su misma frecuencia de muestreo. Se escanea la entrada de datos y se recogen dichos datos.

Una vez tratados por nuestro controlador Fuzzy, enviamos los datos por un nuevo canal de salida, que son recogidos desde Arduino por un nuevo puerto serie3.

A continuación el código en Matlab de la comunicación Arduino-Matlab:

```
delete (instrfind({'Port'},{'COM6'}));
placa = serial( 'COM6' , 'BaudRate', 57600 );
fopen( placa );
M1=0;
M2=0;
M3=0;
M4=0;
```

```
while (true)
  loQueRecibo = fscanf( placa );
  imprimir = regexp(loQueRecibo,'\|','split')
  X = [tiempo,imprimir{1}]
  Y = [tiempo,imprimir{2}]

  aux = [num2str(M1),'-',num2str(M2),'-',num2str(M3),'-',num2str(M4)]
  fprintf(placa,'%s',aux)
end
```

4.2.1 Descripción del controlador Fuzzy

Una vez obtenidos los datos entrantes relativos a la inclinación de la plataforma, estos son enviados a nuestro esquemático, definido en la Figura 4.13. Como se ha mencionado anteriormente el controlador Fuzzy tendrá como entradas la inclinación de X y de Y, como salidas el PWM de cada motor.

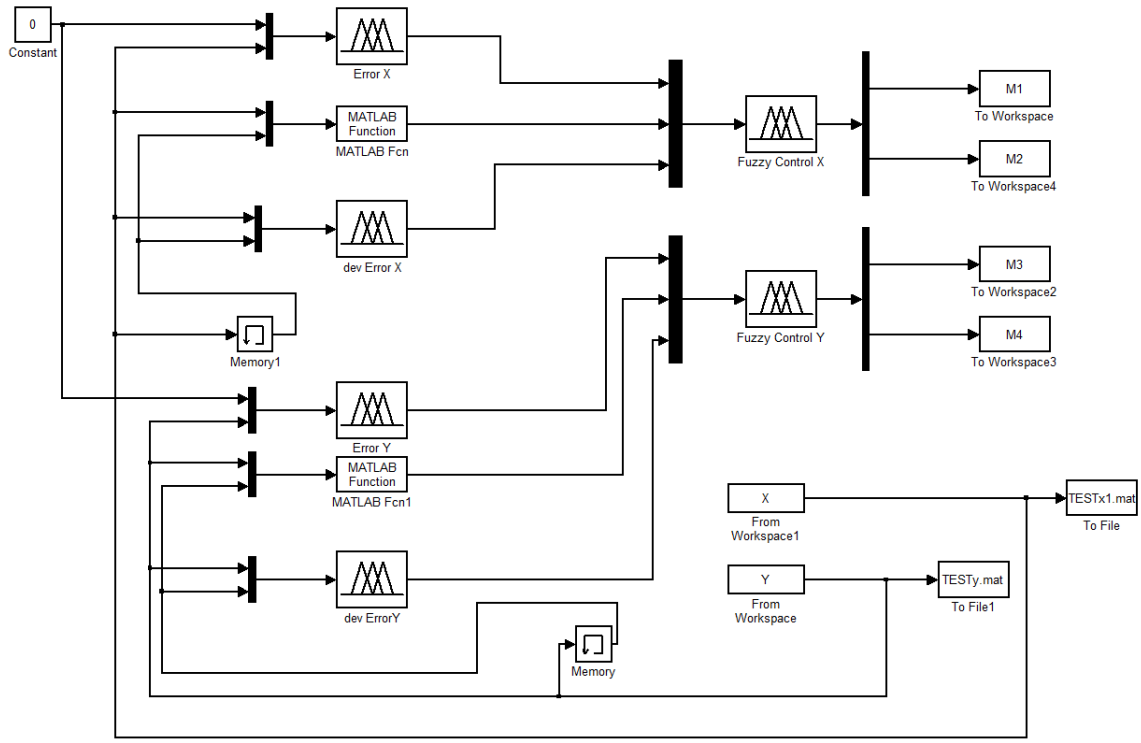


Figura 4.13: *Esquemático en Simulink*

El sistema a controlar es un sistema de tipo MIMO y es de cuatro entradas (motor1, motor2, motor3 y motor4) y dos salidas (inclinación de X y de Y). Para calcular la ganancia del controlador Fuzzy, se usa los errores y sus derivadas. Con la intención de conseguir simplificar el controlador y sus reglas, la configuración de módulos se dividió en 2 bloques, uno de ellos para modulación de los datos de las salidas obtenidas y otro bloque para realizar el control.

Ya que el sistema cuenta con un retardo provocado por la transmisión de datos desde la IMU a nuestro Arduino y de este a Matlab, puede ocurrir que en este transcurso de tiempo se produzca una sobre corrección, en estos casos ha de comprobarse si la corrección ha sido correcta o no. Para ello primero calculamos (a través de las 2 funciones mostradas en el esquemático del Simulink), si la corrección realizada en el ciclo anterior ha sido correcto.

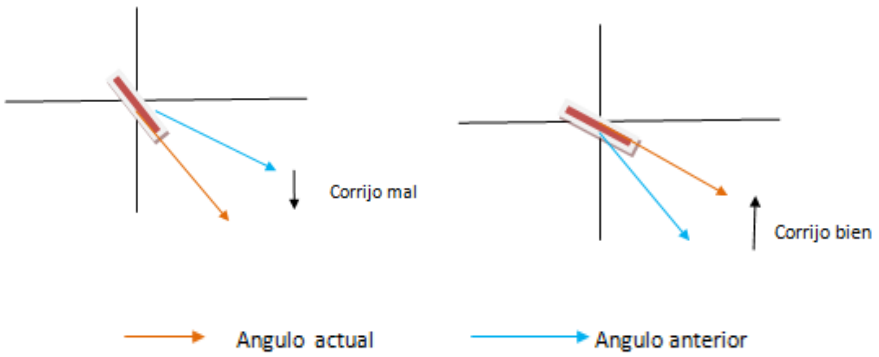


Figura 4.14: *Correcciones de ángulos.*

A la hora de obtener las entradas del control principal, es necesario saber cómo ha sido la corrección, ya que cuando el quadcopter se inclina hacia la derecha y se obtiene en la derivada del error un resultado negativo y realizamos la misma operación pero para la izquierda y obtenemos un

resultado negativo (en ambos casos el resultado es -20º). En el primer caso la recuperación está siendo incorrecta y en el segundo está siendo correcta.

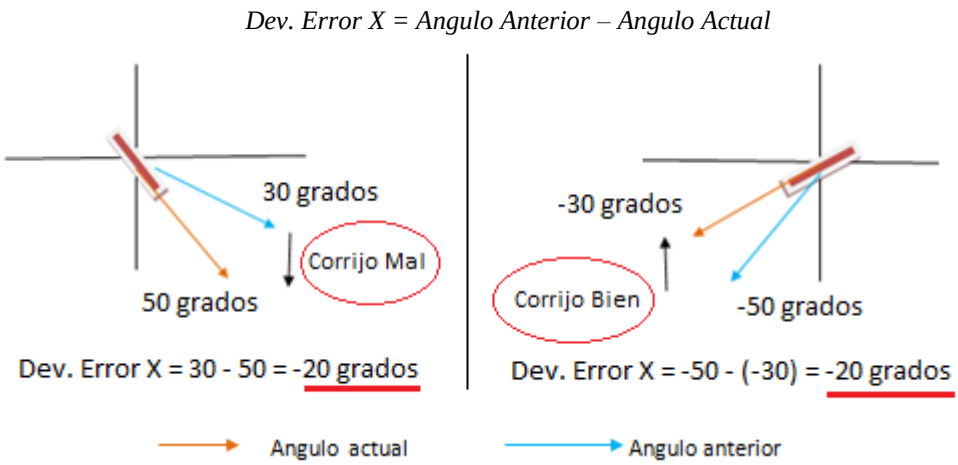


Figura 4.15: *Control recuperación correcta.*

Por ello requerimos de las 2 funciones mencionadas anteriormente para desambiguar esto. Por último, unificaremos los resultados con un módulo Fuzzy con el que obtendremos la potencia requerida a cada pareja de motores, los datos de entrada serán: el error, la derivada del error y hacia qué lado se está inclinando.

4.2.2 Reglas Fuzzy

Basado en lo explicado anteriormente, se ha generado un total de 464 reglas para realizar el controlador Fuzzy. 22 de esta reglas han sido necesarias para identificar los casos del error de X e Y, 142 para identificar la derivada del error de X e Y y 300 para calcular la potencia (en señales PWM) de cada motor.

Dicho sistema cuenta con una combinación de componentes Fuzzy que regulan la inclinación con un rango de 10º, desde -50º hasta 50º.

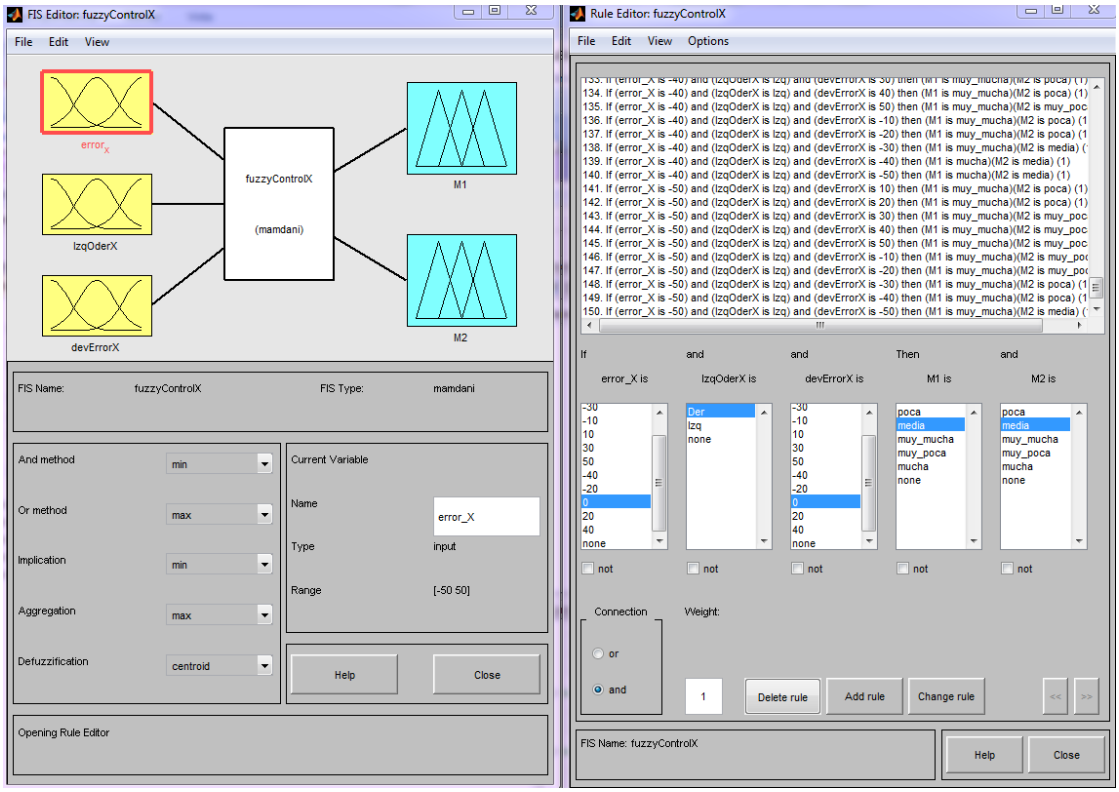


Figura 4.16: *Editor para el control del eje X*

Las reglas en el controlador Fuzzy principal a la hora de crearlas, se dividieron en 2 tipos. El primero, solo variaba el error y la derivada del error quedaba constante a cero.

Primer tipo de regla:

If (error_X is -20) and (IzqODer is Der) and (devErrorX is 0) then

(M1 is media)(M2 is mucha)(1)

El segundo tipo de reglas, es aquellas que variaba la derivada del error, teniendo en cuenta si se corregía bien o mal.

Segundo tipo de regla:

If (error_X is 40) and (IzqODer is Der) and (devErrorX is -20) then

(M1 is poca)(M2 is mucha)(1)

If (error_X is 40) and (IzqODer is Der) and (devErrorX is 20) then

(M1 is media)(M2 is mucha)(1)

error_X: error obtenido de calcular ángulo de referencia – ángulo actual.

IzqODer: Identifica hacia qué lado se está inclinando la plataforma.
devErrorX: derivada del error de X, obtenido del error anterior – error actual.
M1, M2: motores.
Muy poca, poca, media, mucha, muchísima: PWM desde 130 hasta 190.

En el caso que el error es cercano a 0, y la derivada del error es muy alta (40º), quiere decir que la corrección es muy fuerte y la respuesta dada es frenar la aceleración de ese motor para que no provoque el efecto de bamboleo.

Por falta de tiempo a la hora de finalizar las simulaciones y poder describir meticulosamente los resultados, no se han incluido en la memoria.

Capítulo 5

Seguimiento de tracyectorias y comunicación

Para realizar el sistema de navegación y comunicación se empleará el modulo GPS/GRPS/GSM v2.0 de DFRobot. Al ser un módulo polivalente capaz de realizar tanto la comunicación como proporcionar la posición GPS, nos permite ahorrar tanto peso como ruido en el caso que dispusiéramos 2 de módulos para cada finalidad.

Este módulo consume 100mAH lo cual no repercute de manera importante en nuestras baterías de 2200mAH. Al ser un módulo Arduino, la instalación y el uso del mismo es muy rápido y sencillo, con lo que facilita la integración con la plataforma.

En la siguiente figura se muestra la distribución de los componentes de nuestro modulo.

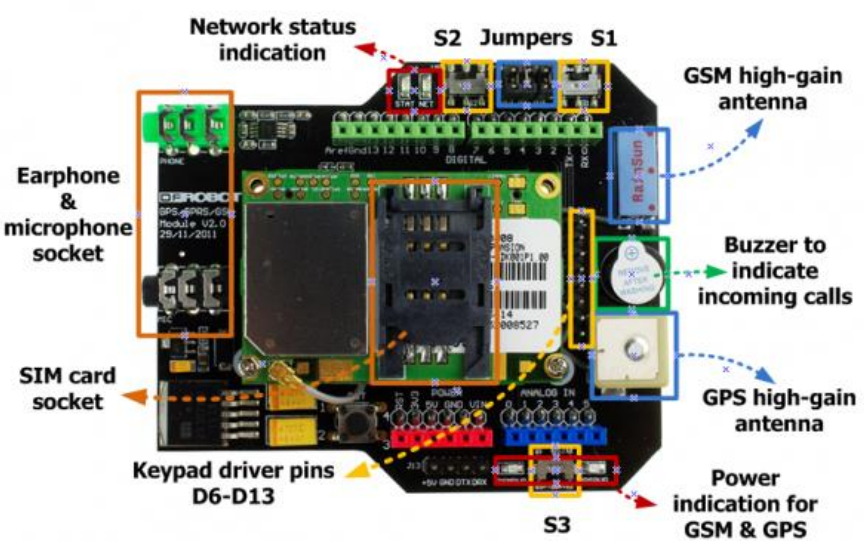


Figura 5.1: *Descripción modulo GPS/GRPS/GSM v2.0*

5.2. Navegación GPS

El chip usado por nuestro módulo de navegación es el SIM548C de SIMCom. El sistema permite una combinación AGPS (GPS asistido) para mejorar la precisión, pero para nuestra navegación no requerimos de más de la precisión obtenida por el GPS.

Para realizar nuestra navegación primero marcaremos una serie de POIS a los queremos que se desplace, a continuación a través de este módulo seremos geolocalizados obteniendo las latitud y la longitud actual. En ese preciso momento la máquina de estados iniciara el sistema y generara una ruta entre nuestra posición inicial y el primer POI.

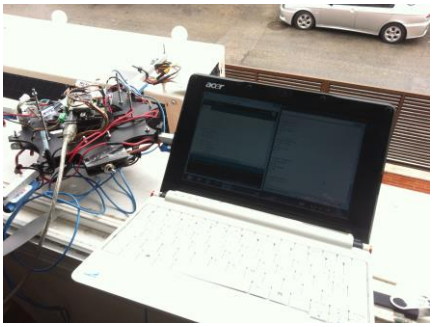


Figura 5.2: *Conexión quadcopter-Pc*

A la hora de realizar las pruebas con el modulo GPS, primero se tuvo que transportar la plataforma a una zona exterior, ya que el sistema GPS requiere estar al aire libre para poder triangularizar la señal GPS.

La comunicación entre Arduino y este módulo, se obtiene utilizando el cuarto canal de comunicación disponible por Arduino Mega. A fin de no colapsar al micro controlador, los datos obtenidos por el módulo GPS con el que se transmite la posición geolocalizada es enviada cada segundo.

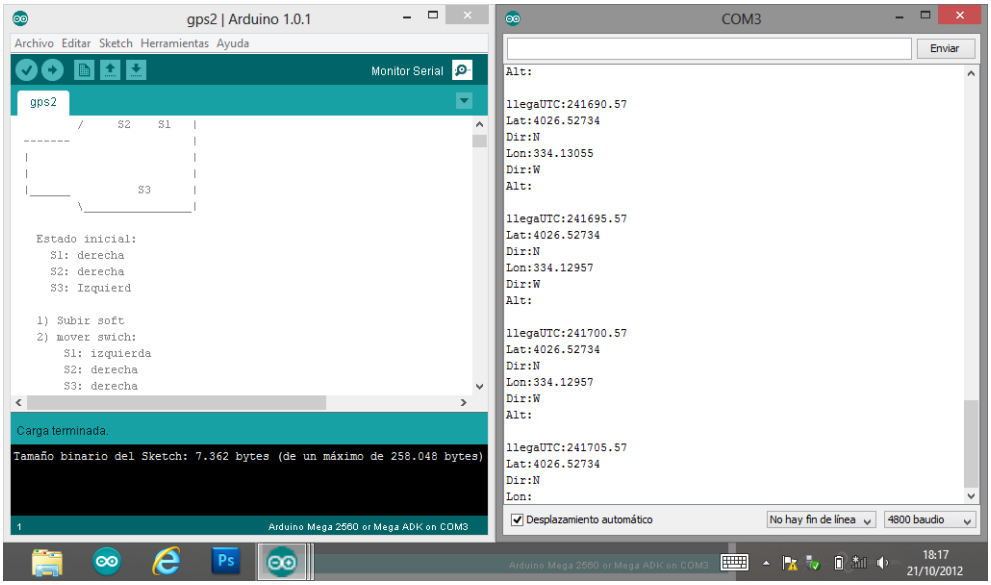


Figura 5.3: *Datos obtenidos por el modulo GPS*

Una vez geolocalizados, se procede a calcular la distancia entre los puntos geográficos, para ello se usara la *Fórmula de Haversine*. Cuando se desea buscar la distancia entre 2 puntos sobre un plano es sencillo, pero cuando se realiza sobre la esfera terrestre no, ya que al no ser una distancia lineal se debe tener en cuenta la curvatura terrestre. La tierra al no ser redonda, el reacio ecuatorial es de 6378km mientras que el radio polar es de 6357km. En nuestro caso se usara el radio ecuatorial, aunque no es la situación exacta en la tierra, el error en distancias menores a 20 km es despreciable con este método.

La *Fórmula de Haversine* es la siguiente:

$$\Delta lat = lat2 - lat1$$

$$\Delta long = long2 - long1$$

$$a = \sin^2(\frac{\Delta \text{lat}}{2}) + \cos(\text{lat1}) * \cos(\text{lat2}) * \sin^2(\frac{\Delta \text{long}}{2})$$

$$c = 2 * A\tan2(\sqrt{a}, \sqrt{(1-a)})$$

$$d = R * c$$

R : radio de la tierra

lat1,*lat2*: latitud quadcopter y latitud POI

long1,*long2*: longitud quadcopter y latitude POI

d: distancia

A continuación se añade el código en lenguaje C de Arduino:

```
double difLatitud = transfRadianes(latitudDestino - latitudOrigen);
double difLongitud = transfRadianes(longitudDestino - longitudOrigen);

double a = sin ((difLatitud/2)*(difLatitud/2)) + cos(transfRadianes(latitudOrigen)) * cos(transfRadianes(latitudDestino)) * sin(difLongitud*difLongitud);

double c = 2* atan2 (sqrt(a), sqrt(1-a));
double d = R * c;
```

5.3. Comunicación GSM

Dicho dispositivo permite realizar una comunicación cuadri-banda GSM/GRPS, el dispositivo trabajar a frecuencias EGSM a 900 MHz, DCS a 1800 MHz, GSM a 850MHz y PCS a 1900 MHz. El consumo de este módulo cuando realiza operaciones de comunicación GSM es de 200mAH,

pero ya que esta tarea solo se realizara cuando llegan a la posición geográfica marcada, el consumo es despreciable.

El sistema permite tanto realizar llamadas como enviar mensajes. Para realizar los objetivos marcados en este proyecto, solo se requerirá usar la funcionalidad de envío de mensajes.

Para realizar las pruebas de su correcto funcionamiento se usó una tarjeta Orange de prepago. A causa de ser un módulo comprado en el extranjero, la tarjeta SIM no es compatible con esta parte del módulo GPS/GSM y por ello no respondía a las instrucciones descritas por el fabricante.

A continuación se muestra el código usado para realizar las pruebas referentes a la comunicación GSM, para ello describimos el código realizado:

```
byte gsmDriverPin[3] = { 3,4,5};
 void setup(){
  InicializarGSM();
}

void InicializarGSM (){
  for(int i = 0 ; i < 3; i++){
    pinMode(gsmDriverPin[i],OUTPUT);
  }
  digitalWrite(3,LOW);
  digitalWrite(4,HIGH);
  //Tiempo de salida del Modulo GSM
  digitalWrite(5,HIGH);
  delay(1500);
  digitalWrite(5,LOW);
}
```

```
void loop(){
  delay(100);
}
```

Una vez cargado el código en memoria, para realizar las pruebas con el módulo GSM se han de enviar vía consola. Las pruebas realizadas sobre el modulo permitían cargar el código PIN de la tarjeta, pero a la hora de enviar un mensaje fallaba, por ello no se ha podido completar esta parte de los objetivos.

Capítulo 6

Conclusiones y líneas futuras

6.1. Conclusiones

A la hora de describir las conclusiones, se define el estado actual de la quadcopter. La plataforma ha sido completada satisfactoriamente incluyendo todos los componentes descritos, consiguiendo construir una plataforma con los componentes casi más baratos del mercado, con un centro de gravedad cercano a la perfección y correcta funcionalidad del sistema de navegación.

El peso de la plataforma es aproximadamente de 850gr., con un coste total de 482€. Dicho precio comparado con la plataforma más barata encontrada en el mercado (Arducopter Quad-C valorada en 449$, que carece de componente GPS y con motores de inferior potencia), se ha conseguido una de las plataformas de mayor bajo coste y con componentes competitivos.

Referido al control de vuelo, se ha conseguido equilibrarse en la estructura de pruebas dentro del espacio de trabajo usando nuestro controlador PID.

El sistema de navegación funciona perfectamente comunicando al microcontrolador la latitud y la longitud actuales de la plataforma en tiempo real. Con una mejora, podría implementarse un sistema 3G que permitiera conocer desde un móvil la posición exacta de cada quadcopter cada momento.

6.2. Líneas futuras

Aunque la construcción de la plataforma está casi completada, una de las carencias que presenta es la falta de la inclusión de la cámara térmica, encargada de realizar las mediciones de temperatura. Dicho componente hardware sería la última parte a añadir.

Otra ampliación seria la capacidad de realizar vuelos cooperativos con otros quadcopter, consiguiendo así patrullas de vigilancia anti-incendios más eficientes.

Por último, la inclusión de un componente 3G que permitirá la geolocalizacion de los quadcopter vía internet desde cualquier PC o móvil.

Referencias

[1] http://www.youtube.com/watch?v=upko_2Z-z0Q,
59 millas/hora (aprox. 95 km/h) 17/10/2012

[2] O. Meister, N. Frietsch, C. Ascher, y G. Trommer, *Adaptive path planning for a vtol-uav, in Position, Location and Navigation Symp-sium*, IEEE/ION, mayo 2008.

[3] A. Moses, M. Rutherford, y K. Valavanis, R*adar-based detection and identification for miniature air vehicles, in Control Applications (CCA)*, IEEE International Conference, septiembre 2011. Pp 933-940

[4] A. E. R. Shabayek, C. Demonceaux, O. Morel, y D. Fofi, *Vision based uav attitude estimation: Progress and insights"* . Volumen 65, numerous del 1-4, pp. 295-308.

[5] C. Martinez, I. Mondragon, M. Olivares-Mendez, y P. Campoy, *On-board and ground visual pose estimation techniques for uav control*. Springer, Paises Bajos, 2011. Pp. 301-320

[6] "kinect," 2012. *http://en.wikipedia.org/wiki/Kinec*

[7] Hibrid system laboratory, berkeley university, quadrotor and kinect, *http://hybrid.eecs.berkeley.edu*

[8] Miguel A. Olivadres-Mendez, Luis Mejias. Pascual Campoy y Ignacio Mellado-Bataller, *See-and-Avoid Quadcopter using Fuzzy Control Optimized bay Cross-Entropy,* Universidad Politécnica de Madrid

[9] Markus Hehn and Raffaello D'Andrea. *Real-Time Trajectory Generation for Interception Maneuvers with Quadrocopters.* IEEE/RSJ (International Conference on Intelligent Robots and Systems), 2012.

[10] D. Mellinger, N. Michael, y V. Kumar, *Trajectory Generation and Control for Precise Aggressive Maneuvers with Quadrotors*, en el Simposio Internacional de Robótica Experimental, 2010.

[11] S. Lupashin, A. Sch¨ollig, M. Sherback, y R. D'Andrea, *A Simple Learning Strategy for High-Speed Quadrocopter Multi-Flip,* en IEEE (International Conference on Robotics and Automation), 2010.

[12] L.-C. Lai, C.-C. Yang, y C.-J. Wu, *Time-Optimal Control of a*

Hovering Quad-Rotor Helicopter, Diario de Sistemas Inteligentes de Robótica, 2006.

[13] Daniel Mellinger, Michael Shomin, Nathan Michael, Vijay Kumar. *Cooperative Grasping and Transport usingMultiple Quadrotors*, Universidad de Pensilvania (GRASP Lab.)

[14] N. Michael, J. Fink, y V. Kumar, "*Cooperative manipulation and transportation with aerial robots*," en Proc. de Robótica: Ciencia y Sistemas, Seattle, WA, Junio 2009.

[15] J. Fink, N. Michael, S. Kim, y V. Kumar, "*Planning and control for cooperative manip-ulation and transportation with aerial robots*," en el Simposio Internacional de Robótica Experimental, Luzern, Suiza, Agosto. 2009.

[16] R. Y. Rubinstein y D. P. Kroese, *The Cross Entropy Method: A Unified Approach To Combinatorial Optimization, Monte-carlo Simulation (In-formation Science and Statistics)*. Secaucus, NJ, USA: Springer-Verlag New York, Inc., 2004.

[17] G. R. Bradski, *Computer vision face tracking for use in a perceptual user interface*, http://citeseer.ist.psu.edu/585206.html

[18] Miguel A. Olivadres-Mendez, Luis Mejias. Pascual Campoy and Ignacio Mellado-Bataller, *Adaptive Path Planning for Autonomous UAV Oceanic Search Missions*, Polytechnic Universidad de Madrid, 2004

[19] MiKuadricoptero, *sites.google.com/site/mikuadricoptero/*

[20] Óscar Higuera. *QUAV (Quadruple Unmanned Aerial Vehicle).* Universidad Rey Juan Carlos I

[21] Johanyák, Z.C. y S. Kovács (2006).*Fuzzy rule interpolation based on polar cuts, Computational Intelligence, Theory and Applications*. Springer Verlag, Berlin, Heidelberg, New York. Bernd Reusch (Ed.).

[22] Johanyák, Z.C., D. Tikk, S. Kovács y K. K. Wong (2006). *Fuzzy rule interpolation Matlab toolbox - FRI toolbox*. Actas del Congreso Mundial de la IEE de Inteligencia Computacional (WCCI'06), 15th, Canada. Pp.1427-1433.

[23] Skrjanc, I., S. Blazie, S. Oblak y J. Richalet (2004). *An approach to predictive control of multivariable time-delayed plant: Stability and design issues.* ISA Transactions 43(4) pp 585-595.

[24] Vascák, J. (2008). *Fuzzy cognitive maps in path planning.Acta Technica Jaurinensis, Series Intelligentia Computatorica*, Universidad Széchenyi Istvan de Györ, Hungría. Pp 467-479.

[25] Vascák, J. (2009). *Using neural gas networks in track navigation.Acta Technica Jaurinensis, Series Intelligentia Computatorica*, Universidad Széchenyi Istvan de Györ, Hungría, Pp. 203–215.

[26] Zhao, Z.-Y., W.-F. Xie y H. Hong (2010). *Hybrid optimization method of evolutionary parallel gradient search.International* Diario de la Inteligencia Artificial. Pp 1-16.

[27] Kruszewski, A., R. Wang y T.M. Guerra (2008). *Nonquadratic stabilization conditions for a class of uncertain nonlinear discrete time ts fuzzy models: A new approach.* IEEE Transactions on Automatic Control.

[28] Chin-Tzong, P. y L. Yung-Yih (2008). *On the stability of takagi-sugeno fuzzy systems with time-varying uncertainties.* IEEE Transactions on Fuzzy Systems 74(5). Pp.453-464

[29] Precup, R.-E., M.-L. Tomescu, St. Preitl y E.M. Petriu (2009). *Fuzzy logic-based stabilization of nonlinear time-varying systems*. Revista Internacional de Inteligencia Artificial.

[30] Haber, R.E., R.M. del Toro y A. Gajate (2010). *Optimal fuzzy control system using the cross-entropy method. a case study of a drilling process*. Ciencias de la Información.

[31] Kurnaz, S., O. Cetin y O. Kaynak (2010). *Adaptive neuro-fuzzy inference system based autonomous flight control of unmanned air vehicles*. Sistemas Expertos con aplicaciones. Pp 1229-1234.

[32] Precup, R.-E., M.-L. Tomescu, St. Preitl y E.M. Petriu (2010). *Fuzzy logic-based stabilization of a magnetic ball suspension system.* Revista Internacional de Inteligencia Artificial. Pp 56-66.