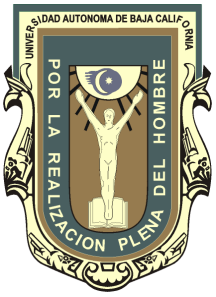
Universidad Autónoma de Baja California

Facultad de Ingeniería, Arquitectura y Diseño



Maestría y Doctorado en Ciencias e Ingeniería

Control de motores de un cuadróptero

TESIS

que para cubrir parcialmente los requisitos necesarios para obtener el grado de

MAESTRO EN INGENIERÍA

Presenta

ALEJANDRO UREÑA ACUÑA

Ensenada, Baja California, Agosto del 2015.

UNIVERSIDAD AUTÓNOMA DE BAJA CALIFORNIA

FACULTAD DE INGENIERÍA, ARQUITECTURA Y DISEÑO

CONTROL DE MOTORES DE UN CUADRÓPTERO

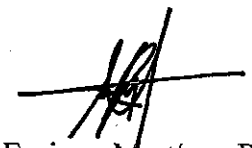
TESIS

Que para obtener el grado de maestría en ingeniería presenta:

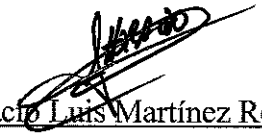
Alejandro Ureña Acuña

Aprobada por:

Manuel Moisés Miranda Velasco
Director de tesis

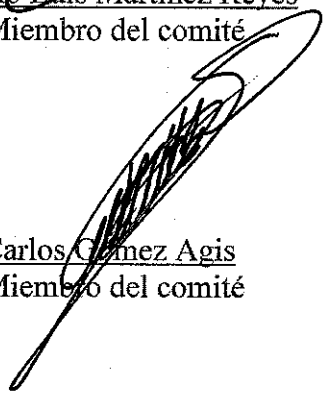


Miguel Enrique Martínez Rosas
Miembro del comité



Horacio Luis Martínez Reyes
Miembro del comité

Humberto Cervantes de Ávila
Miembro del comité



Carlos Gómez Agis
Miembro del comité

Ensenada, Baja California, México. Agosto de 2015

Resumen de la tesis de **Alejandro Ureña Acuña**, presentada como requisito parcial para la obtención del grado de MAESTRO EN INGENIERÍA del programa de Maestría y Doctorado en Ciencias e Ingeniería (MYDCI) de la UABC. Ensenada Baja California, México, Agosto del 2015.

Control de motores de un cuadróptero

Resumen Aprobado por:

Manuel Moisés Miranda Velasco
Director de Tesis

En este trabajo se desarrolló e implementó una plataforma para el control de los motores de un vehículo aéreo no tripulado (VANT), en configuración cuadróptero con un bajo costo.

Durante el análisis de la dinámica de vuelo del cuadróptero, se encontraron muchas irregularidades durante el vuelo, como la sensibilidad a perturbaciones externas como viento y donde la acción correctora no es eficiente debido a que el VANT no resguarda su posición y se observa una deriva importante.

Para corregir esto, se propuso una plataforma de control, donde el usuario sea capaz de poder desarrollar e implementar sus propios algoritmos de control, asi como nuevos sensores capaces de optimizar la calidad del vuelo y obtener mas datos de navegación.

Palabras Clave: *cuadróptero, drone, control PID, Plataforma para el control de motores*

Dedicatoria

A mi amada esposa Monserrat por su amor,su paciencia, sus consejos, su apoyo y motivación durante la realización de mis estudios.

A mis padres María Santos y Jose Luis por su apoyo. A mis hermanos.

A mi amigo Tizoc Fernando por su paciencia y sabios consejos.

Agradecimientos

Quiero expresar mi agradecimiento:

Al Dr. Manuel Moisés Miranda Velasco, mi director de tesis, por sus sabios consejos, por guiarme oportunamente en la realización de mi proyecto y por estar al pendiente de mis necesidades y las de mi familia.

A mis sinodales Dr. Miguel Enrique Martínez Rosas, Dr. Horacio Luis Martínez Reyes, Dr. Humberto Cervantes de Avila y al M.C. Carlos Gómez Agis por sus consejos, criticas y apoyo durante el desarrollo de esta tesis.

A mis compañeros y amigos: Ernesto Martínez Sandoval, Miguel Pérez, Edgar Cadena, Victor Velasquez, Francisco Villalpando, Ismael H. Capuchín por su amistad y los grandes momentos que pasamos juntos.

Al personal administrativo y docente de la Universidad Autónoma de Baja California por su atenciones, amabilidad y disposición para ayudarme.

Índice general

Índice de figuras

Índice de tablas

Capítulo 1

Introducción

Un cuadróptero o cuadricóptero,-es una aeronave que entra en vuelo gracias a la fuerza de cuatro rotores que usualmente se montan en forma de cruz. Es un vehículo totalmente diferente si se compara con un helicóptero, principalmente por la manera en que ambos son controlados. Los helicópteros son capaces de cambiar el ángulo en el que ejercen fuerza sus hélices, mientras que un cuadróptero no. Otra diferencia entre ellos es que los cuadrópteros no requieren conexiones mecánicas para variar el ángulo de paso de las palas del rotor a medida que giran. Esto simplifica el diseño y mantenimiento del vehículo, también, el uso de cuatro rotores le permite que cada rotor posea un diámetro menor que el equivalente de rotor de helicóptero, lo que genera que tengan menos energía cinética durante el vuelo[1].

La mayor limitación de los helicópteros es la necesidad de un mantenimiento extensivo y costoso para un correcto funcionamiento en el aire. Los VANT y los vehículos aéreos miniatura no están exentos de esta limitación, por ello se busca simplificar la estructura de estas máquinas lo cual trae beneficios tanto operativos como económicos.

En la actualidad hay poca diversidad de vehículos aéreos no tripulados (VANT) capaces de realizar un vuelo con algoritmos realizados por los usuarios, en el mercado nacional. Sin embargo, hay un gran interés por parte de investigadores en desarrollar investigaciones relacionadas con los VANT del tipo cuadróptero, para poder implemen-

tar algoritmos propios que puedan controlar la posición y las distintas variables que puedan afectar su operación durante el vuelo.

A la fecha, muchos de los cuadrópteros que se desarrollan están basados en juguetes voladores y están destinados a la investigación o a usarse para el entrenamiento y recreación. Aunque dichos sistemas pueden ser utilizados como prototipos, no son lo suficientemente fuertes para servir en experimentos. Muchos de estos vehículos de cuatro motores son construidos a partir de componentes de juguetes a control remoto. Esto resulta en que el rendimiento y la fiabilidad práctica de estos prototipos no sea el esperado por lo que es necesario contar con algorítmos de control robustos.

Actualmente, los cuadrópteros se están diseñando en tres áreas principales: militares, transporte (tanto personas como bienes) y VANT. Los VANT se pueden clasificar en otros dos grupos: ligeros y pesados. Así mismo, estos dos grupos se pueden dividir en muchos otros dependiendo el tipo de motores que utiliza, forma de despegar, materiales y muchos otros parámetros. Los vuelos a baja altura en ambientes con obstáculos representan un alto riesgo para la vida de los pilotos, por esto que los vehículos no tripulados despiertan mucho interés en aplicaciones como inspección de edificios, inspección de líneas de distribución de energía eléctrica en alta tensión, tareas de rescate en zona de desastres o de alta montaña, filmación, etc. Otra razón por la que estos vehículos resultan atractivos es por su bajo costo de operación [2].

El despegue y aterrizaje vertical de los VANT como los cuadrópteros tiene sus ventajas en comparación a los aviones de ala fija. Pueden moverse en cualquier dirección y son capaces de volar a bajas velocidades y mantenerse estáticos en una ubicación. Otra ventaja que tienen es la capacidad de aterrizar en casi cualquier terreno mientras que los aviones requieres una pista previamente preparada para despegar y aterrizar. Gracias a estas características, los cuadrópteros pueden ser utilizados en misiones de búsqueda y rescate, meteorología, exploración de entornos peligrosos y muchas otras aplicaciones a las cuales puede adaptarse. Además, están tomando un papel importante en áreas de la investigación como en la ingeniería de control, donde sirven como prototipos en aplicaciones de la vida real[3].

El principio por el que vuelan parece muy sencillo, aunque su funcionamiento interno no lo es tanto. Una placa electrónica de control reparte el voltaje entre los cuatro reguladores o variadores de velocidad, para dar mayor o menor número de vueltas a sus respectivos rotores, lo que de forma automática y computarizada va a permitir que se muevan hacia arriba, abajo, cualquier lado, o permanezcan estables en el aire. Los más simples cuentan con giróscopos de 3 a 6 ejes que les sirven para mantener una posición horizontal con mayor o menor estabilidad.

Los más avanzados son auténticos robots voladores, cuentan con múltiples ayudas a la navegación, ya no solo giroscopios, si no altímetros, acelerómetros y GPS(del inglés Global Positioning System), que les sirve para mantener una posición estable como si estuvieran colgados de un hilo, o un rumbo con una gran exactitud. Los cuadrópteros a menudo pueden volar de forma autónoma. Muchos controladores de vuelo modernos utilizan un software que permite al usuario marcar puntos en el mapa, a los que el cuadróptero volará y realizará su tarea, tales como el aterrizaje o ganar altitud. Otras aplicaciones de vuelo incluyen el control de multitudes de vuelo entre varios cuadrópteros, donde se utilizan datos visuales del dispositivo para predecir a donde se moverá la multitud y a su vez dirigir al cuadróptero al siguiente punto correspondiente[4].

Otra forma en la que los cuadrópteros pueden volar es con el sistema FPV (del inglés First Person View), este pequeño sistema cumple la función de permitirle al usuario o dueño de un cuadróptero, ver exactamente lo que la nave no tripulada ve una vez que empieza su vuelo. En esencia el usuario se convierte en el piloto o en los ojos del drone. La ventaja de utilizar este pequeño sistema es que el usuario toma el control total de la vista desde un cuadróptero y al ver casi en persona lo que ocurre en las alturas, mientras la máquina sigue su curso, las grabaciones de vídeos cuentan con un aspecto más refinado y cinematográfico, esto gracias al control del ángulo perfecto o la perspectiva que el usuario desee capturar[5].

1.1. Justificación

El empleo de las nuevas tecnologías en nuestra vida diaria es muy importante y se requiere estar al tanto de los avances tecnológicos. Un ejemplo es el uso de los helicópteros multi-rotor con cuatro brazos conocidos como cuadrópteros o cuadricópteros, que ha ido en aumento y siguen surgiendo nuevas utilidades para este innovador artefacto, para poder analizar todas las ramas de utilidades que puede tener, es necesario adquirir un conocimiento amplio acerca del funcionamiento y las limitaciones que puedan llegar a tener estos artefactos. Desde su función profesional en el ámbito militar, hasta su empleo en el ocio, los cuadrópteros son muy útiles para la captura de fotos o videos de ángulos y alturas que de otra manera, no se podrían controlar con facilidad. Aún así estas aplicaciones aún se ven limitadas. Dichas limitaciones pueden ir reduciendo la vida o la utilidad de este, pero con el empleo de mejores tecnologías y materiales que ayuden a prolongar el tiempo de uso y la cantidad de fuerza de los cuadrópteros. Estas nuevas tecnologías no solo aumentarán el rendimiento de los usos actuales de los cuadrópteros, si no que darán paso a nuevos usos. Aprender cómo funciona un cuadróptero, es útil para poder implementar satisfactoriamente los nuevos usos que se les pueden dar a las áreas relacionadas con el desarrollo, ya que se requieren conocimientos de diferentes áreas como aviación, electrónica, programación. Personas con interés en los cuadrópteros deberán poséer conocimientos generales acerca de campos cómo controlar el vuelo, estabilizarlo, a qué altura puede llegar, los movimientos que puede realizar, el funcionamiento del motor, tipo de software para mejorar su rendimiento o si el rendimiento se ve afectado por éste, entre otro tipo de factores.

Para lograr lo anterior es necesario contar con una plataforma experimental que permita una interacción a diferentes niveles con el prototipo de cuadróptero utilizado, dicha plataforma debe contar con acceso al hardware y software del sistema.

1.2. Objetivo General

Realizar un sistema de telemetría que gestione la interacción entre el hardware y software a través de señales de mando para su aplicación en un cuadricóptero comprendiendo los sistemas de control de estabilidad y dirección necesarios, abarcando desde las estaciones de vuelo hasta la implementación de código libre, del mismo modo obtener de manera precisa el conocimiento adecuado para el manejo de los motores de un cuadricóptero.

1.3. Objetivos específicos

- *Relacionar el sistema de control de los movimientos básicos de un cuadricóptero, con su estabilidad en el aire en el vuelo automático y durante los comandos del piloto.*
- *Determinar cómo se realiza la programación de un cuadróptero.*
- *Saber cómo están relacionados los sistemas electrónicos y mecánicos.*
- *Conocer los movimientos que puede realizar el cuadróptero.*
- *Conocer que factores pueden afectar el funcionamiento de los motores.*
- *Determinar cómo se realiza la programación de un cuadróptero.*
- *Manipulación de los modos de vuelo de un cuadricóptero haciendo uso del código abierto en sus sistemas de control.*
- *Implementación de un controlador PID diferente al que el fabricante propone.*

1.4. Secuencia de la tesis

Este trabajo está dividido en los siguientes capítulos:

Capítulo 2 Se presenta el marco teórico, incluyendo definiciones y la descripción detallada del objeto bajo estudio, así como las etapas que constituyen el desarrollo del control de los motores.

Capítulo 3 Describe el diseño e implementación del código desarrollado, se incluye la información considerada para la implementación física en el sístema y el tipo de señales obtenidas en cada etapa.

Capítulo 4 En éste capítulo se presentan los resultados obtenidos en la aplicación práctica del cuadróptero, se indican las condiciones en que fueron realizadas las mediciones y se da una interpretación de las mismas.

Capítulo 5 Se dan las conclusiones del trabajo.

Capítulo 2

Antecedentes

2.1. Vehículos aéreos

2.1.1. Historia de los vehículos aéreos

La aparición de los vehículos aéreos empezó en el siglo V con el diseño del primer aparato volador: El cometa o papalote. En el siglo XIII el monje inglés Roger Bacon tras años de estudio, llego a la conclusión que el aire podría soportar un ingenio de la misma manera que el agua soporta a un barco.

A comienzos del siglo XVI Leonardo da Vinci analizó el vuelo de los pájaros y anticipó varios diseños que después resultaron realizables. Entre sus importantes contribuciones al desarrollo de la aviación se encuentra el tornillo aéreo o hélice y el paracaídas. Concibió diferentes tipos de inventos más pesados que el aire.

El ornitóptero, máquina con alas como las de un pájaro que se podían mover mecánicamente; el helicóptero diseñado para elevarse mediante el giro de un rotor situado en el eje vertical y el planeador en el que el piloto se sujetaba a una estructura rígida a la que iban fijadas las alas diseñadas a imagen de las grandes aves.

La primera persona que escribió sobre el principio de la elevación fue Sir George Cayley. Para 1799 había el hecho el descubrimiento más importante en la historia de la aviación, descubrió que el aire que fluye por encima de un ala curvada y fija crea

elevación, una fuerza hacia arriba que hace que el ala se eleve. Otra persona que continuó avanzando el conocimiento del vuelo por planeamiento, fue Otto Lilienthal, quien diseñó y construyó planeadores.

El 17 de diciembre de 1903, por primera vez en la historia, los hermanos Wright pudieron remontar un aparato que era más pesado que el aire, se trataba de un biplano, una máquina movida por fuerza propia y capaz de viajar sin perder velocidad. Los alemanes también hacían sus ensayos en ese momento, aunque ellos desarrollaron otro tipo de naves: los zeppelines, globos metálicos que contaban con un motor que permitía dirigirlos y no dejarlos a merced del viento como ocurría con sus antecesores. Fueron estas dos experiencias las que convencieron a las autoridades que estaban frente a un invento que, bien desarrollado, podía constituir una eficaz solución de transporte.

En la Segunda Guerra Mundial se desarrollaron en masa los helicópteros. El *Focke-Achgelis Fa 61*, fue el primer helicóptero completamente controlable, hizo su primer vuelo en 1936. Fue un helicóptero experimental de la compañía alemana Focke Achgelis.

El exterior era similar al de un avión ligero de ala fija, equipado con un motor radial Bramo Sh.14A de 160 cv instalado en el morro, cuya misión básica era la de accionar los dos rotores tripalas contrarrotatorios emplazados a ambos costados por medio de un complejo sistema de montantes; también movía una hélice convencional de escaso diámetro cuya función era la de refrigerar el motor.

Figura 2.1: Avión d guerra controlable Focke-Achgelis Fa 61

Al finalizar las guerras fue cuando el transporte aéreo alcanzo un lugar destacado

en todos los países, sobre todo con líneas aéreas británicas y estadounidenses montando aviones con propulsores más grandes y eficientes. Aparte de los aviones supersónicos, un gran avance en los viajes aéreos fue la introducción, en 1970, del Boeing 747, el llamado reactor Jumbo, que puede llevar desde 360 hasta más de 500 pasajeros en vuelos regulares.

2.1.2. Tipos de Vehículos Aéreos

2.1.3. Aviones

También denominado aeroplano, es una aeronave de ala fija con mayor densidad que el aire, dotado de alas y un espacio de carga capaz de volar, impulsado por ninguno, uno o más motores. Pueden clasificarse por su uso como aviones civiles (que pueden ser de carga, transporte de pasajeros, entrenamiento, contra incendios, etc.) y aviones militares (carga, transporte de tropas, cazas, bombarderos, de reconocimiento o espías, de reabastecimiento en vuelo, etc.).

Su principio de funcionamiento se basa en la fuerza aerodinámica que se genera sobre las alas, en sentido ascendente, llamada sustentación.

Esta se origina por la diferencia de presiones entre la parte superior e inferior del ala, producida por la forma del perfil alar.

Figura 2.2: Avión comercial

2.1.4. Helicópteros

Un helicóptero es una aeronave que es sustentada y propulsada por uno o más rotores horizontales, cada uno formado por dos o más palas donde las alas giratorias sirven para proporcionar elevación, propulsión y control. Los helicópteros están clasificados como aeronaves de alas giratorias, comparado con otros tipos de aeronave como el avión, el helicóptero es mucho más complejo, tiene un mayor coste de fabricación, uso y mantenimiento, es relativamente lento, tiene menos autonomía de vuelo y menor capacidad de carga. No obstante, todas estas desventajas se ven compensadas por otras de sus características, como su gran maniobrabilidad y la capacidad de mantenerse estático en el aire, girar sobre sí mismo y despegar y aterrizar verticalmente[6].

Figura 2.3: Helicóptero

Las aspas del rotor giran alrededor de un eje vertical, describiendo un disco en un plano casi horizontal. Las fuerzas aerodinámicas son generadas por el movimiento relativo de la superficie de un ala con respecto al aire. El helicóptero puede generar estas fuerzas con sus alas giratorias incluso cuando la velocidad de éste es cero. El helicóptero por su parte, tiene la capacidad de vuelo vertical, incluyendo descenso vertical y aterrizaje. El logro eficiente del vuelo vertical es la característica fundamental del helicóptero[7].

En comparación con otros aparatos aeronáuticos tiene la facilidad de poder trabajar

fácilmente en tres dimensiones, al poder producir un movimiento en cualquiera de los tres ejes de una manera segura y precisa[8].

Helicópteros Multirotor

Dentro del segmento de aeronaves denominadas drones, los que más popularidad han alcanzado los últimos años han sido los "Multi rotores" (helicópteros de más de 2 rotores de sustentación), y entre ellos mayoritariamente los de 4 hélices (denominados comúnmente cuadricópteros), gracias a su bajo coste, su facilidad de construcción (gracias a su simple mecánica) y su gran maniobrabilidad[9].

La razón principal por la cual se opta por un helicóptero de cuatro rotores, es que este tipo de modelo elimina la complicación mecánica de la construcción de un rotor principal de un helicóptero de un solo rotor. Generalmente la elevación y el resto de los movimientos de los helicópteros convencionales dependen, no solo de la velocidad de giro del rotor, sino también del ángulo de ataque de las aspas y de su inclinación mediante el plato cíclico.

En un helicóptero de cuatro rotores, todos los movimientos, y desplazamientos posibles están controlados con las velocidades de giro de cada rotor, concretamente con las combinaciones de velocidades de giro de los cuatro rotores. De esta manera simplificamos los problemas mecánicos que encontramos en la construcción de rotores convencionales[7]. Al ser un tipo de helicóptero, tanto la sustentación como la propulsión se basan en el aire impulsado por las hélices de sus rotores. En cambio, a diferencia de los helicópteros convencionales que varían su ángulo de incidencia (inclinación) de las palas de las hélices para maniobrar, los multicópteros basan sus maniobras en el cambio de revoluciones de sus distintos motores, usando así, hélices con palas de paso fijo que simplifican en gran medida la mecánica de este tipo de aeronaves[9].

2.1.5. VANT

Es una aeronave que vuela sin tripulación. Tienen una amplia variedad de formas, tamaños, configuraciones y características en el diseño. En este sentido se han creado dos variantes: Algunos son controlados desde una ubicación remota, y otros vuelan de forma autónoma sobre la base de planes de vuelo pre-programados. Se pueden clasificar en seis tipos:

- Blanco.- sirven para simular aviones o ataques enemigos en los sistemas de defensa de tierra o aire.
- Reconocimiento.- sirven para enviar información militar.
- Combate.- para combatir y llevar a cabo misiones que suelen ser muy peligrosas.
- Logística.- diseñados para llevar carga.
- Investigación y desarrollo.- en ellos se prueban los sistemas que están en desarrollo.
- VANT comerciales y civiles.- son diseñados para propósitos civiles, filmar películas y entretenimiento[10].

2.2. Cuadrópteros

Los cuadrópteros o cuadricópteros que ya se definieron previamente en la introducción de este trabajo como vehículos aéreos no tripulados con la capacidad de elevarse y desplazarse con la propulsión de 4 motores instalados al final de un marco en forma de cruz. Para realizar el control de un avión se emplean superficies de contacto con el aire denominados alerones, por otro lado el control del cuadricóptero se realiza por medio de la variación de la potencia de los motores, lo que provoca variaciones en sus momentos, dando lugar al movimiento que se desea obtener. El sentido de giro de las hélices en dos motores es en sentido de las manecillas del reloj mientras que en los otros dos es a la inversa, de esa forma se conserva el par motor y evita que gire sobre sí mismo.

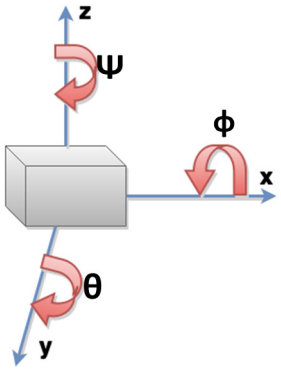


Figura 2.4: Esquema cartesiano del cuadróptero

2.2.1. Historia del Cuadróptero

En el siglo XX el científico francés Charles Richet en conjunto con Louis Bréguet construyeron un pequeño helicóptero pilotado que era una aeronave de alas giratorias, donde la velocidad aerodinámica responsable de la sustentación proviene principalmente del giro de las palas del rotor, sin embargo este modelo que constaba con 4 alas giratorias, se levantó del suelo en 1907, pero carecía de una suficiente estabilidad y capacidad de mando para efectuar un vuelo controlado. Dicho aparato fue denominado "El giro plano" de Bréguet-Richet, que básicamente es un cuadróptero con propulsores de 8.1 metros de diámetro. Un peso de 578 kg incluido los pilotos y con sólo un motor de combustión interna de 50 HP que manejaba los rotores a través de una transmisión de correa y polea. La máquina levantó el vuelo en 1907, con varios testigos se elevó cerca de 1.5 metros y se mantuvo en vuelo por pocos momentos aterrizando inmediatamente.

Posteriormente en 1922 en Francia se crea un prototipo, llamado Convertawings Modelo A, fue diseñado para uso civil y militar. El diseño tenía dos motores que controlaban cuatro rotores. No tenía rotor de cola, ya que usaba la diferencia las velocidades de giro para lograr el desplazamiento.

A finales de los años 90 la empresa de helicópteros Bell y Boeing trabajaron en conjunto en el desarrollo del Quad Tiltrotor. Es un cuadricóptero de despegue horizon-

tal capaz de modificar la dirección de sus cuatro rotores, llamados *proprotors*, hacia delante para tener un vuelo vertical como un avión convencional. Es capaz de llevar una gran carga útil, alcanzando altas velocidades, emplea un pequeño espacio tanto para el despegue como para el aterrizaje y puede modificar la dirección de sus cuatro rotores.

2.2.2. Cuadricópteros VANT

Es la nueva generación de cuadricópteros, que utiliza mayoritariamente la energía eléctrica de una batería, además de usar sistemas de control y sensores electrónicos para estabilizar al vehículo.

El Parrot AR.Drone 1.0 es un famoso VANT de uso recreativo civil. Su funcionamiento consta de 4 motores eléctricos en configuración cuadricóptero, en contraste con otros modelos radiocontrolados, es que cuenta con un microprocesador y una cadena de sensores entre los cuales contienen 2 cámaras que le permiten capturar lo que pasa a su alrededor y un conector Wi-Fi integrado[11].

2.2.3. Características del Cuadróptero

Un dron es un sistema de vuelo que es dependiente de comunicaciones de datos locales de sus sensores, sus efectos de a bordo , y en los enlaces de telecomunicaciones a través de la cual recibe los datos de alimentación y comandos de alimentación a partir de fuentes terrestres y tal vez en el aire y de los satélites. Un dron actúa sobre el mundo, y por lo tanto es un robot. Los pilotos remotos, y los operadores de las instalaciones de drones, como cámaras, dependen de herramientas de alta tecnología que interpretan los datos que son transmitidos y recibidos en forma de imagen y video generado y aumentado, y que permiten la composición de los comandos[12].

2.2.4. Características del Parrot AR. Drone

Un cuadricóptero entra en la categoría de ala rotativa, a nivel “mini”. Esta categoría tiene las características de alcance menor a 10 km, una altitud de vuelo menor a 300

m, una autonomía menor a 2 horas con la capacidad de llevar una carga máxima de 5 kg. Aparte de las características técnicas propias de la categoría a la cual pertenece se pueden destacar:

- Maniobrabilidad

  Los cuatro rotores del vehículo permiten una mayor exactitud en su control, permitiendo ser empleado en sitios con espacio reducido y navegación en interiores

- Capacidad de vuelo vertical

  Al igual que los helicópteros, los cuadricópteros poseen esta capacidad, esta característica resulta ventajosa cuando se desea realizar un vuelo estacionario en lugar de un vuelo horizontal. La autonomía de vuelo de los cuadrópteros a nivel mini no suele ser muy buena, siendo una de las limitaciones por las que los UAV tardaron un cierto tiempo en avanzar.

  Actualmente se están realizando avances importantes en las baterías proporcionando más capacidad y reduciendo los tamaños.

2.2.5. Movimientos longitudinales y ángulos de navegación

Cualquier VANT cuenta con 6 grados de libertad y es capaz de realizar 3 diferentes giros alrededor de sus 3 ejes perpendiculares entre sí, donde su punto de intersección o coordenada (0,0,0) está situado sobre el centro de gravedad de la aeronave. Existirán tres ejes que se utilizarán para describir las 6 maniobras que es capaz de hacer el VANT. Los moviemientos que describen los 6 grados de libertad son:

Movimiento en el eje longitudinal "x"

Este movimiento se realiza variando las velocidades de un par de motores: el frontal y derecho para dirigirse a la derecha ó el izquierdo y trasero para dirigirse a la izquierda. Para lograr este movimiento se deben disminuir un poco las velocidades de los pares de

motores antes mencionados para poder tener los movimientos correspondientes sobre el eje longitudinal.

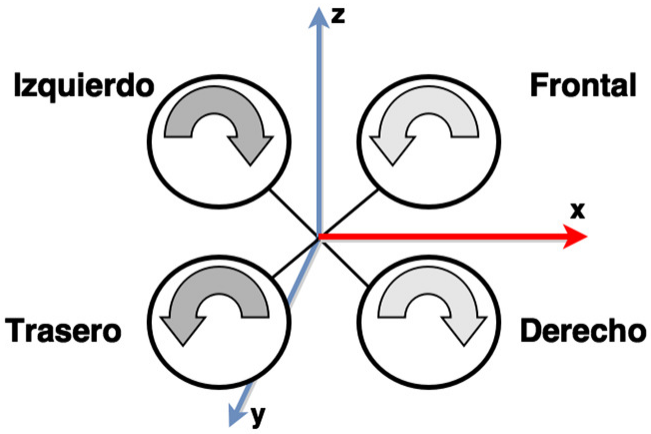


Figura 2.5: Movimiento en el eje longitudinal

Movimiento en el eje transversal z"

Se logra disminuyendo un poco las velocidades de los pares de motores frontal e izquierdo para dirigirlo hacia adelante, ó trasero y derecho para dirigirlo hacia atrás a lo largo del eje transversal.

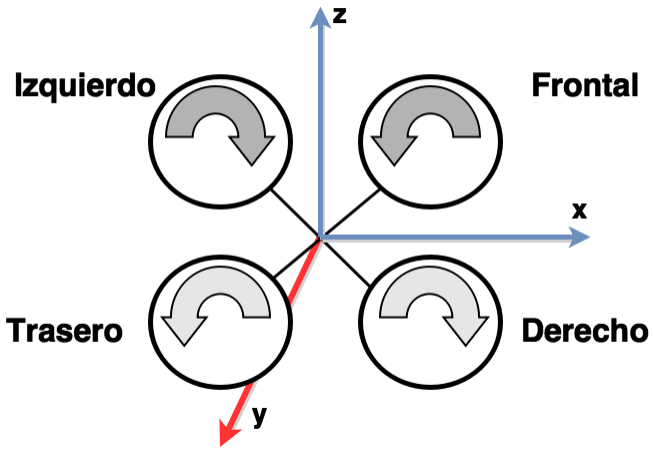


Figura 2.6: Movimiento en el eje transversal

Movimiento en el eje vertical "z"

Este movimiento ocurre cuando todos los motores giran a la misma velocidad. Si hay una variación en las velocidades de los motores, el cuadróptero se desplazará a lo largo del eje vertical. Un aumento en las velocidades implica un movimiento ascendente y una disminución en las velocidades dará como resultado un movimiento descendente.

Cabe mencionar que los ángulos de navegación deben permanecer estables para lograr estos movimientos.

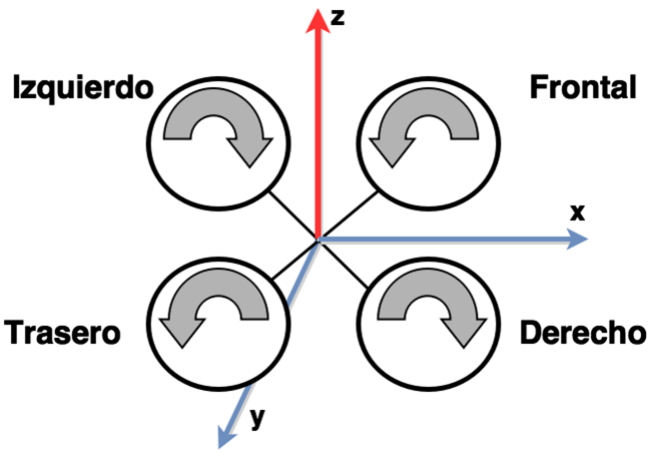


Figura 2.7: Movimiento en el eje vertical

Pitch o Cabeceo

Es el movimiento que se realiza alrededor del eje transversal. Se emplea como actuador el elevador que permite el control de los grados de inclinación durante la maniobra de cabeceo. Para lograrlo es necesario disminuir la velocidad del motor frontal, además de disminuir en menor forma las velocidades de los motores izquierdo y derecho como se muestra en la figura 2.8.

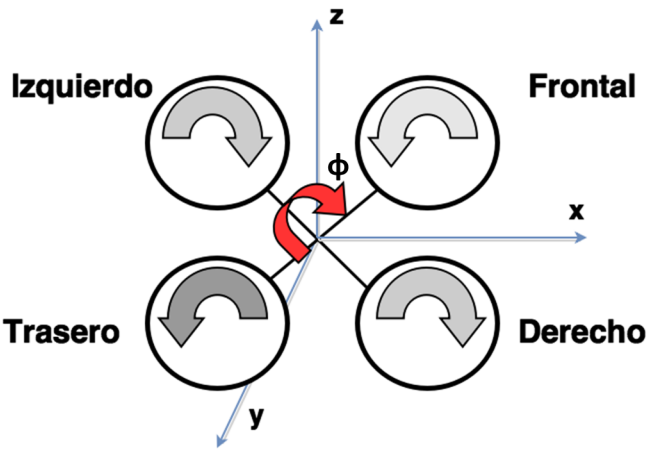


Figura 2.8: Movimiento pitch o cabeceo

Roll o Alabeo

Es el movimiento que se realiza al producirse una variación alrededor del eje longitudinal del VANT. El actuador que se utiliza para este movimiento son las velocidades de los motores derecho, frontal y trasero. El motor derecho es el que debe bajar su velocidad en mayor proporción que los motores frontal y trasero.

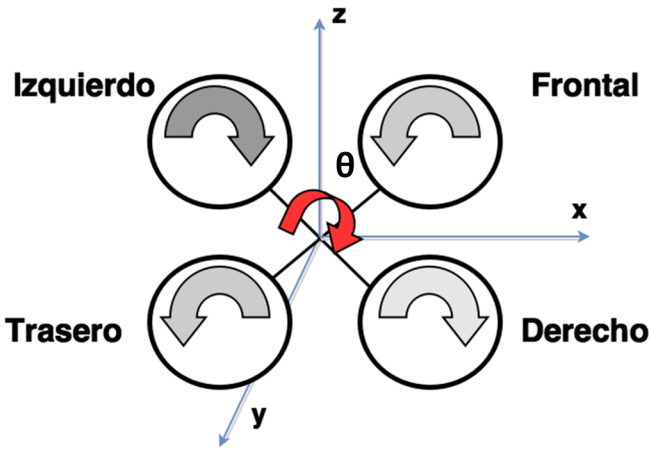


Figura 2.9: Movimiento Roll o Alabeo

Yaw o Guiñada

Se produce cuando se gira alrededor del eje vertical, este eje es perpendicular a los otros dos ejes. Los motores frontal y trasero deben girar a una velocidad menor con respecto a los motores izquierdo y derecho lo cual dara como resultado un movimiento de giro a favor de las manecillas del reloj. Si se desea hacer un giro contrario, los motores izquierdo y derecho deberan tener velocidades menores con respecto a los motores frontal y trasero.

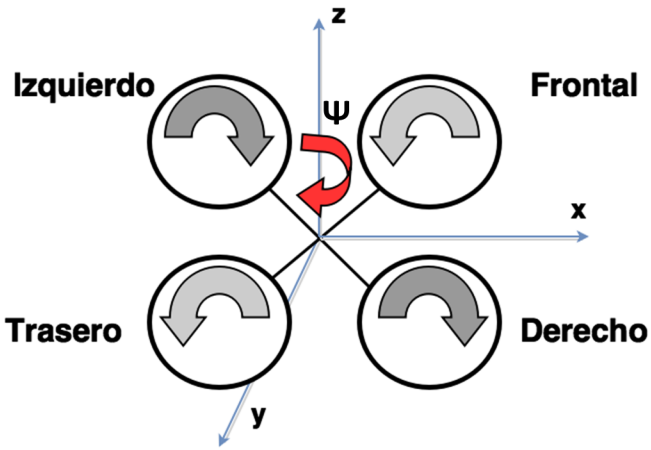


Figura 2.10: Movimiento Yaw o guiñada

2.3. Usos

Existe un amplio rango de usos comunes para un cuadróptero y tiene potencial para muchos más. Los que se mencionarán en esta sección son de áreas donde tiene mayor potencial, las aplicaciones que se mencionarán son:

Envíos (correo, paquetería, libros, etc.), transporte de pasajeros, hobby o entretenimiento.

2.3.1. Envíos

En vuelos recientes, los cuadrópteros han sido utilizados para el reabastecimiento de municiones, rescate de heridos, entrega de medicamentos, pizzas y libros.

Muchas de las características y tecnología contemporánea de los vehículos aéreos no tripulados son la mayor limitante en usos para el desplazamiento de cargas. Una indicación de que esta aplicación es la más probable a destacar es proporcionada por Japón, donde, por la década de 1990, más de 1000 vehículos aéreos con capacidad de 20 a 30 kg, fueron utilizados para fumigaciones. Según algunas fuentes la capacidad práctica de estos vehículos estaba limitada a 1 hectárea por día; aunque dentro de este contexto en particular parece ser altamente rentable.

Esta aplicación está limitada a ciertas áreas, con poca población, pocos edificios, mediana infraestructura y pocos o nula actividad aérea. La ausencia de tales amenazas combinado a la facilidad de manejo a través de una computadora, han reducido en gran medida la demanda de habilidad por parte del piloto. Por otro lado, cuando existen tales amenazas, cabe la posibilidad de riesgo tanto para las personas como para los bienes.

2.3.2. Transporte de pasajeros

Hasta la fecha, todos los aviones de pasajeros llevan un piloto humano, sin embargo los pilotos dependen en gran medida de la tecnología, ningún vuelo de pasajeros es totalmente autónomo. Hay ciertos factores que deben abordarse antes de que los vehículos aéreos sean controlados por pilotos remotos, como puede ser la tecnología para evitar colisiones y la aceptación del público. En el Reino Unido un avión comercial modificado ha realizado un vuelo por espacio aéreo compartido, controlado después del despegue y antes de aterrizar por un piloto remoto.

Es evidente que la seguridad, incluso entre los aviones grandes, es hasta la fecha la mayor preocupación para la aceptación pública.

2.3.3. Hobby o Entretenimiento

La mayor parte del uso de los drones ha sido por individuos que actúan con cierto cuidado y responsabilidad, usualmente por miembros de algún club en las que establecen limitaciones, cubren un seguro, y utilizan un espacio exclusivo y aislado. Los costos para adquirir y operar un dispositivo aéreo han disminuido considerablemente y algunos de los modelos menos costosos se han convertido en un regalo para navidad dentro del rango para un adolescente. La época de los carros a control remoto que aterrorizaban los vecindarios puede estar a punto de dar paso a una ola de dispositivos remotos que no solo se limite a los terrestres, con lo que podría atraer algunos riesgos ya que por la velocidad y altura a la que operan, pueden llegar a provocar daños a propiedades o a personas si es que se salen de control[12].

2.4. Arquitectura del cuadróptero

Para analizar la arquitectura del cuadróptero se dividirá en bloques como se aprecia en la figura

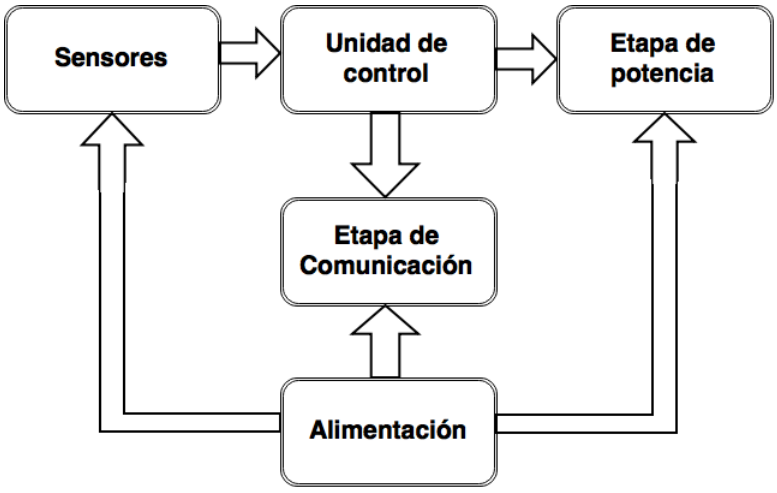


Figura 2.11: Arquitectura de un cuadróptero

2.4.1. Sensores

La instrumentación que permitirá leer las magnitudes físicas involucradas para mantener estable y dar movimiento al cuadróptero y además convertirlas a un valor digital para poder controlar los motores son los siguientes:

- Acelerómetro.- mide la aceleración asociada a la variación de posición.
- Giroscopio.- mide la aceleración angular.
- Magnetómetro.- o brújula digital, empleado para no perder la orientación.
- Sensores ultrasónicos.- permiten detectar y estimar la proximidad de un objeto, generalmente son usados tanto para el despegue como para el aterrizaje, y estimar la altura del vehículo.
- GPS (del inglés *Global Positioning System*).- posiciona al dron en un espacio geográfico determinado, es utilizado para trazar rutas de vuelo de un punto a otro.
- Cámaras de video.- Permiten que el dispositivo pueda grabar durante un vuelo. En coordinación con los otros sensores permite detección de objetos y evasión de obstaculos.

La señal de los sensores es de tipo analógica (variaciones de voltaje), para poderlas procesar es necesario digitalizarlas a través de un conversor A/D. Un microprocesador será el encargado de procesar las señales digitales y con los resultados calcular la respuesta que se debe enviar a los motores. Este conjunto forma un sistema de lazo cerrado pues la realimentación se produce a través del entorno físico por el que se mueve, variando sus magnitudes dependiendo de los requerimientos de altura o posición que sean demandados[13].

2.4.2. Unidad de control

Los VANT y en especial los cuadrópteros requieren de sensores inerciales para controlar tanto los movimientos como la estabilidad. Los datos provenientes de los sensores son procesados en varios microprocesadores de alta velocidad para evitar centralizar el control del sistema, ganando así velocidad de respuesta. El uso de cámaras resulta atractivo hoy en día, los controles basados en visión artificial son temas de investigación en muchos laboratorios del mundo.

Tanto la parte sensorial como el sistema de video integrado al cuadróptero actúan en conjunto para brindar estabilidad y controlabilidad al sistema, incluso se han desarrollado colaboradores automáticos para hacerlo más fácil de manejar y hacerlo atractivo a usuarios inexpertos.

2.4.3. Etapa de potencia

Los cuadrópteros generalmente emplean motores de tipo sin escobillas(brushless) para dar propulsión al sistema. Las escobillas son los elementos que hacen contacto en el colector de un motor común. En los motores de DC más pequeños, son de una aleación de cobre y en motores más grandes son de un compuesto a base de carbón. Los motores brushless en vez de funcionar con DC funcionan con AC, la mayoría se alimenta con una señal trifásica, esta señal idealmente debería ser sinusoidal, pero en la práctica son pulsos, sin embargo se los clasifica como motores de DC porque al igual que los motores comunes tienen imanes permanentes. Los imanes son atraídos por la polaridad de un campo magnético generado en las bobinas, las cuales como se explicó anteriormente reciben pulsos en un patrón específico. Si se desea que el motor gire más rápido, simplemente se hace girar el campo magnético secuencial a mayor velocidad, es decir sería necesario aumentar la frecuencia de los pulsos. Un control de velocidad adecuado de los motores permitirá al cuadróptero desplazarse en cualquier dirección[14]. Los cuadrópteros utilizan este tipo de motores debido al alto rendimiento en revoluciones por minuto que estos tienen, además del poco desgaste mecánico lo cual

los hace mas duraderos.

2.5. Elementos y especificaciones técnicas del AR Drone v1.0

2.5.1. Batería

El A.R. Drone utiliza una batería recargable LiPo (Lithium Polymer) de 3 celdas de 11.1 V y 1000 mAh para alimentarse. Durante el vuelo la batería disminuye la tensión de carga completa que es 12.5 V a 9 V, el A.R. Drone convierte esta tensión en un porcentaje de duración de la batería.

Cuando el UAV detecta un bajo voltaje de la batería, primero envía una advertencia para el usuario, de forma automática. Si la tensión alcanza un nivel crítico el sistema se apagará con el fin de evitar cualquier comportamiento inesperado.

En conjunto con el cuadricóptero se tiene un cargador LiPo balanceado para recargar la batería y mantenerla siempre en perfectas condiciones, alargando su vida útil. Incluye distintos adaptadores para poderlo usar en varios países que no usan el mismo tipo de enchufe. La desventaja es que la batería dura tan sólo 12 minutos durante el vuelo, y se necesitan 90 minutos para recargarla[15].

Figura 2.12: Batería LiPo recargable

2.5.2. Sistema informático

El AR. Drone, al contrario de la mayoría de helicópteros y aviones teledirigidos convencionales, tiene un sistema informático completo, que supera en potencia a algunos

teléfonos móviles actuales. Cuenta con un microprocesador ARM 9 a 468 MHz de 32 bits y dispone de 128 MB de memoria RAM DDR a 200 MHz, un chip WiFi b/g el cual se usa para crear una red wifi al cual se tendrá acceso para controlar el AR.Drone.

A nivel de software, usa sistema operativo Linux; el software es totalmente actualizable y permite que sea personalizado. Mediante la interfaz USB, se puede cargar un software propio.

2.5.3. Sensores

Con respecto a los sensores, el AR. Drone dispone de:

- Sensor de ultrasonidos para medir los cambios de altitud.
- Sensor de altitud para correcciones.
- Acelerómetro digital de tres ejes para monitorizar los movimientos de posición con presición de 50mg.
- Giróscopio de dos ejes y giróscopo preciso piezoeléctrico para medir el giro y en cabeceo.
- Magnetómetro de tres ejes para la orientación.
- Sensor de presión que responde ante cualquier diferencia de presión tiene una tolerancia de 10 Pa.

Todos estos sensores están impresos en un circuito montado sobre un esqueleto de fibra de carbono. En su totalidad resultan cruciales para determinar la posición y orientación del drone en el vuelo.

Figura 2.13: Altímetro

2.5.4. Motores

El AR. Drone cuenta con 4 motores de 15 W sin escobillas(brushless) que funcionan a 35.000 RPM. Cada motor cuenta con leds de color rojo, naranja y verde, que indican el estado del motor y permiten verificar si hay algún problema, además de permitir diferenciar la parte trasera de la delantera mientras está en vuelo. El control es realizado por un microcontrolador incorporado en el VANT, el mismo que detecta automáticamente a los motores que están conectados y ajusta el control de los mismos, también detecta si todos los motores están girando o están detenidos, en caso de que una hélice que esté girando encuentre algún obstáculo, el AR. Drone detecta si alguna de las hélices se bloquea y en tal caso todos los motores se detienen de inmediato. Este sistema de protección impide repetidos choques y el consecuente daño de los motores. El cuadricóptero también cuenta con 4 hélices de alta eficacia y una estructura de tubos, ambos elaborados a base de fibra de carbono.

Figura 2.14: Motor del Parrot AR.Drone.

2.5.5. PWM

La Modulación por ancho de pulso (PWM) (pulse width modulation) de una señal o fuente de energía es una técnica en la que se modifica el ciclo de trabajo (D) de una señal periódica (una sinusoidal o cuadrada), ya sea para transmitir información a través de un canal de comunicaciones o para controlar la cantidad de energía que se envía a una carga. El ciclo de trabajo de una señal periódica es el ancho relativo de su parte positiva en relación con el período.

La Regulación por Ancho de Pulso de un motor de CC está basada en el hecho de que si se recorta la corriente de alimentación en forma de una onda cuadrada, la energía que recibe el motor disminuirá de manera proporcional a la relación entre la parte alta (habilita corriente) y baja (cero corriente) del ciclo de la onda cuadrada. Controlando esta relación se logra variar la velocidad del motor de una manera bastante aceptable.

Para controlar la velocidad de un motor DC se necesita un voltaje variable DC de la fuente de alimentación. Sin embargo si se usa un motor de 12 Voltios y se conecta la alimentación, el motor empezará a aumentar su velocidad; los motores no responden inmediatamente, necesitan un pequeño intervalo de tiempo para alcanzar su velocidad máxima. Si se apaga la alimentación en algún momento antes que el motor alcance su máxima velocidad, se notará una disminución de ésta y si se enciende la alimentación y se apaga rápidamente, el motor tomará una velocidad comprendida entre velocidad cero

y velocidad máxima. Esto es exactamente lo que hace un controlador PWM: alimentar el motor suministrándole una serie de pulsos. Para controlar la velocidad del motor se varía (modula) el ancho de los pulsos, y como el motor siempre se encuentra alimentado a su tensión nominal nunca se verá reducido su torque nominal.

2.5.6. Cámara frontal

Esta pequeña cámara se encargará de grabar y transmitir todo lo que ve. Cuenta con una lente de gran angular a 93°, sensor CMOS y resolución VGA (640 x 480), con una velocidad de video de 15 fps (tramas por segundo). Esta cámara además de permitir ver en tiempo real lo que está viendo el AR. Drone, también permitirá detectar otros objetos, como por ejemplo, otros AR. Drone[16].

Figura 2.15: Cámara Frontal

2.5.7. Cámara vertical

Para la navegación del AR. Drone cuenta con una cámara vertical de alta velocidad a 60 fps con un ángulo de 64°. Esta cámara permite que el sistema informático lo estabilice incluso con viento leve (velocidad del viento máxima de 15 km/h). También se tiene la posibilidad de conectar con esta cámara desde la aplicación de control, una opción realmente útil para aterrizarlo correctamente.

Figura 2.16: Cámara Vertical

2.6. Seguridad

Las primeras versiones del modelo de Parrot, permitían una fácil accesibilidad a poder modificar de manera completa el vehículo, es relativamente sencillo poder agregar o modificar el hardware, y referente al software, es basado en Linux y código abierto, lo que permite que usuarios con conocimientos técnicos puedan modificar el código predeterminado del drone y así poder realizar, incluso, acciones que originalmente no estaba diseñado para hacer.

En la versión 2.2 del dron de Parrot se consigue mayor seguridad y es difícil acceder a el código o modificarlo remotamente, sin embargo el código sigue siendo en base Linux y por lo tanto se pueden acceder a permisos de "súper usuario"(root) y así poder modificar completamente el código y funcionamiento del vehículo. Se puede acceder a esta vulnerabilidad mediante la wifi integrada en el drone, así mismo, se pueden realizar otras manipulaciones mediante un puerto USB, incluso se pueden introducir virus para que infecte a futuras USB. Una medida de seguridad ofrecida por Parrot es vincular el app de un smartphone al drone. Así se vincula la dirección MAC y se mantiene segura, bloqueando cualquier tráfico no autorizado. Esta manera es una de la más sencilla, pero también de las más efectivas contra hackers amateurs.

La manera más eficaz de poder proteger de ataques a nuestro dron, es haciendo la red de este más segura, encriptando nuestro código, cambiando el tipo de seguridad a

algo como WPA2. Lo que requiere compilar un código que será corrido en arquitectura ARM y cambiar las especificaciones de la red integrada del dron.

2.7. Protocolos de transporte

2.7.1. TCP

TCP es un protocolo de transporte orientado a conexión enormemente extendido en Internet. Las aplicaciones de red más populares (ftp, telnet, acceso web,etc) lo utilizan en sus comunicaciones.

La función principal del nivel de transporte dentro de la arquitectura de protocolos TCP/IP es la de permitir la comunicación extremo a extremo entre dos aplicaciones de forma económica y fiable. La unidad básica de transferencia se denomina segmento, de tamaño máximo el denominado MSS (Maximum Segment Size) expresado en octetos, que como veremos más adelante se negociará por los extremos de la comunicación en el establecimiento de la misma[17].

2.7.2. UDP

El protocolo de datagrama de usuario (User Datagram Protocol (UDP)) es un protocolo del nivel de transporte basado en el intercambio de datagramas. Permite el envío de datagramas a través de la red sin que se haya establecido previamente una conexión, ya que el propio datagrama incorpora suficiente información de direccionamiento en su cabecera. Tampoco tiene confirmación ni control de flujo, por lo que los paquetes pueden adelantarse unos a otros; y tampoco se sabe si ha llegado correctamente, ya que no hay confirmación de entrega o recepción.

Su uso principal es para protocolos protocolos en los que el intercambio de paquetes de la conexión/desconexión son mayores, o no son rentables con respecto a la información transmitida, así como para la transmisión de audio y vídeo en tiempo real, donde no es posible realizar retransmisiones por los estrictos requisitos de retardo que se tiene

en estos casos[18].

2.7.3. Comparaciones entre los protocolos TCP y UDP

- UDP: proporciona un nivel de transporte no fiable de datagramas, ya que apenas añade la información necesaria para la comunicación extremo a extremo al paquete que envía al nivel inferior. Lo utilizan aplicaciones como NFS (Network File System) y RCP (Remote Procedure Call) el cual es un comando que sirve para copiar ficheros entre ordenadores remotos, pero sobre todo se emplea en tareas de control y en la transmisión de audio y video a través de una red. No introduce retardos para establecer una conexión, no mantiene estado de conexión alguno y no realiza seguimiento de estos parámetros. Así, un servidor dedicado a una aplicación particular puede soportar más clientes activos cuando la aplicación corre sobre UDP en lugar de sobre TCP.

- TCP: es el protocolo que proporciona un transporte fiable de flujo de bits entre aplicaciones. Está pensado para poder enviar grandes cantidades de información de forma fiable, liberando al programador de la dificultad de gestionar la fiabilidad de la conexión (retransmisiones, pérdida de paquetes, orden en el que llegan los paquetes, duplicados de paquetes, etc.) que gestiona el propio protocolo. Pero la complejidad de la gestión de la fiabilidad tiene un coste en eficiencia, ya que para llevar a cabo las gestiones anteriores se tiene que añadir bastante información a los paquetes que enviar. Debido a que los paquetes para enviar tienen un tamaño máximo, cuanta más información añada el protocolo para su gestión, menos información que proviene de la aplicación podrá contener ese paquete (el segmento TCP tiene una sobrecarga de 20 bytes en cada segmento, mientras que UDP solo añade 8 bytes). Por eso, cuando es más importante la velocidad que la fiabilidad, se utiliza UDP. En cambio, TCP asegura la recepción en destino de la información para transmitir.

2.8. Wi-Fi

El protocolo IEEE 802.11 o Wi-Fi es un estándar de protocolo de comunicaciones del Instituto de Ingeniería Eléctrica y Electrónica (IEEE, por sus siglas en inglés) que define el uso de los dos niveles mas bajos del modelo OSI (capa física y de enlace de datos), especificando sus normas de funcionamiento en una WLAN. En general los protocolos de la rama 802.11x definen la tecnología de redes de área local.

El estándar original de este protocolo data de 1997, fue el IEEE 802.11, tenía velocidades de 1 hasta 2 Mbps y trabajaba en la banda de frecuencia de 2.4 GHz con una modulación de señal de espectro expandido por secuencia directa (DSSS), o con espectro expandido por salto de frecuencia (FHSS).

Una red Wi-Fi permite conectar servidores, PC, impresoras, etc., con la particularidad de hacerlo sin necesidad de cableado.

De manera purista vale a decir que el acrónimo Wi-Fi se utiliza para identificar los productos que incorporan cualquier variación de la tecnología sin cables (wireless) de los estándares IEEE 802.11, que permiten la creación de redes de área local sin cables conocidas como WLAN4, y que son plenamente compatibles con los de cualquier otro fabricante que utilice estos estándares. Las características generales de funcionamiento de una red Wi-Fi son las mismas que las de una red con cableado. La particularidad es que el Wi-Fi utiliza el aire como medio de transmisión.[19]

2.9. Características de los estándares IEEE 802.11

2.9.1. Estándar 802.11a

- Velocidad máxima de hasta 54 Mbps.
- Opera en el espectro de 5 GHz.

  Menos saturado.
- No es compatible con las normas 802.11b y 802.11g.

- Modulación OFDM.

  Transmisión exterior

- Valor máximo a 30 metros 54 Mbps
- Valor mínimo a 300 metros 6Mbps.

  Transmisión interior.

- Valor máximo a 12 metros 54 Mbps.
- Valor mínimo a 90 metros 6 Mbps.

2.9.2. Estándar 802.11b

- Velocidad máxima de hasta 11 Mbps.
- Opera en el espectro de 2.4 GHz.
- Conocido como WI-FI.
- Modulación DSSS.
- Compatible con los equipos DSSS del estandar 802.11a.

2.9.3. Estándar 802.11g

- Velocidad máxima de hasta 54 Mbps.
- Opera en el espectro de 2.4 GHz.
- Compatible con 802.11b.
- Modulación DSSS y OFDM.

Estándar	802.11b	802.11a	802.11g
Velocidad	11 Mbps	54 Mbps	54 Mbps
Canales no solapados	3	8	3
Costo	Barato	Caro	Barato
Banda de frecuancia	2.4-2.497 GHz	5.15-5.35 GHz; 5.425-5.675 GHz	2.4-2.497 GHz
Cobertura	100 m en interior, 300 a 400 m en exterior	150 m en interior y exterior	100 m en interior, 300 a 400 m en exterior
Compatibilidad	compatible con 802.11g	incompatible	compatible con 802.11b
Modos de datos	1,2,5.5,11 Mbps	6,9,12,18,24,36,48,54 Mbps	1,2,5.5,11 Mbps
Modulación	DSSS	OFDM	OFDM y DSSS

Tabla 2.1: Comparación de los estándares 802.11

2.10. Telnet

TELNET (TELecommunication NETwork) es un protocolo de red que bajo el protocolo TCP/IP, sirve para controlar otro dispositivo remotamente, siempre y cuando, el dispositivo a controlar reconozca dicho protocolo. Telenet es un protocolo del tipo cliente servidor en el cual el cliente se conecta a través del puerto 23. Telnet esta diseñado para acceder en modo terminal. Para conectarse al cuadróptero se necesita un cliente TELNET, cuando el cuadróptero se encuentra conectado a través de la red Wi-Fi, el cliente se puede conectar vía telnet a través de la direccion IP del cuadróptero la cual es 192.168.1.1.

```
Microsoft Windows [Versión 6.3.9600]
(c) 2013 Microsoft Corporation. Todos los derechos reservados.

C:\Users\Alejandro>telnet 192.168.1.1
```

Figura 2.17: TELNET

2.11. Compilador Cruzado

Un compilador cruzado o cross compiler es un compilador capaz de generar un código ejecutable para una plataforma distinta a la que se desea implementar.

Dichos compiladores son muy utilizados en sistemas empotrados en donde los recursos que se tienen son muy limitados.

La desventaja de este tipo de compiladores es que cada arquitectura utiliza un compilador específico, si se utiliza un cross compiler diferente, el sistema enviará un error de segmentación y el programa compilado no se ejecutará.

Para instalarlo se utiliza el comando *sudo apt-get install gcc-arm-linux-gnueabi.*

Para entrar en ambiente de desarrollo se escribe en la terminal de Ubuntu el comando *codesourcery-arm-2009q3.sh*

Una vez en el entorno de desarrollo del cross compiler,escribimos el archivo que quermos cambiar a arquitectura ARM mediante el comando: *arm-none-linux-gnueabi-gcc archivo.c -o archivoarm*

Es asi como tendremos un archivo compatible con la arquitectura ARM y listo para ejecutarse.

2.12. FTP

El protocolo de transferencia de archivos (File Transfert Protocol) o FTP por sus siglas en inglés, es un protocolo de comunicaciones destinado al intercambio informatico de archivos sobre una red TCP/IP. Permite, desde una computadora, copiar archivos de una computadora a otra dentro de la misma red.

Para transferir archivos al cuadróptero, se debe utilizar un cliente FTP, para este proyecto se utilizará *FILEZILLA*, el cual es un programa gratuito y de código abierto.

La transferencia de archivos hacia el cuadróptero se realizará sobre el puerto 5551, dicho puerto es el que esta configuarado para utilizar el protocolo FTP en el cuadróptero.

Para realizar dicha tarea, es necesario contar con instrumentos que posean interfaces que permitan su operación con ayuda de una computadora personal, en el caso de éste trabajo se implementó un servidor FTP creado exclusivamente para transferir archivos al cuadróptero para poder manipularlo fácilmente.

El servidor FTP implementado permite transferir los programas realizados a través del puerto 5551.

2.13. Controladores

Un controlador es un instrumento que compara el valor medido con el valor deseado, en base a esta comparación calcula un error(diferencia entre el valor medido y deseado), para luego actuar a fin de corregir este error. Tiene por objetivo elaborar la señal de control que permita que la variable controlada corresponda a la señal de referencia[20].

Existen distintos tipos de controladores que se usan comúnmente en los cuadrópteros, los cuales se describen a continuación:

2.13.1. Controlador de acción Proporcional

La acción de control proporcional (P), da una salida del controlador que es proporcional al error, es decir: u(t) = $K_P e(t)$, que descrita desde su funcion transferencia queda:

$$C_P(s) = K_P \tag{2.1}$$

Donde K_P es una ganancia proporcional ajustable. Un controlador proporcional

puede controlar cualquier planta estable, pero posee un desempeño limitado y error en régimen permanente.

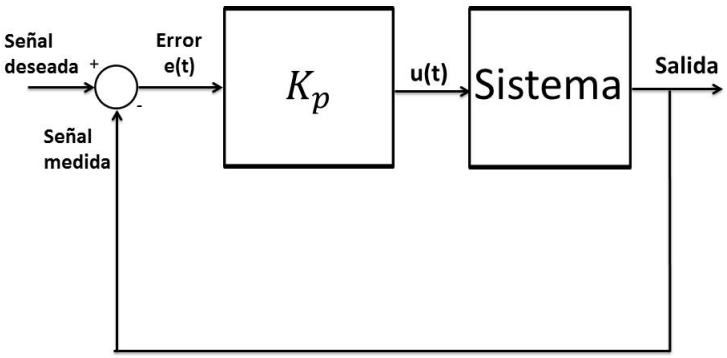


Figura 2.18: Esquema de un controlador proporcional

2.13.2. Controlador de acción Integral

La acción de control integral (I) da una salida del controlador que es proporcional al error acumulado, lo que implica que es un modo de controlar lento.

$$u(t) = K_i \int_{x=0}^{x=t} e(\tau) \cdot d\tau \tag{2.2}$$

La señal de control u(t) tiene un valor diferente de cero cuando la señal de error e(t) es cero. Por lo que se concluye que dada una referencia constante, o perturbaciones, el error en régimen permanente es cero.

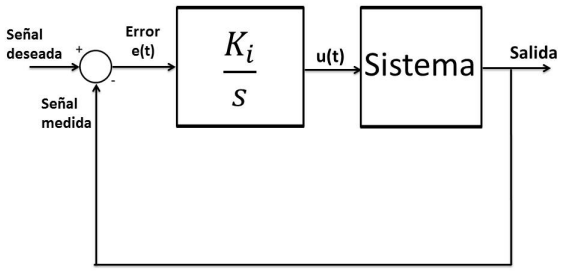


Figura 2.19: Esquema de un controlador integral

2.13.3. Controlador de acción Proporcional-Integral

La acción de control proporcional-integral (PI), se define mediante:

$$u(t) = K_P e(t) + K_I \int_{x=0}^{x=t} e(\tau) \cdot d\tau \tag{2.3}$$

donde Ki se denomina constante de tiempo integral y es quien ajusta la acción integral. La función de transferencia resulta

$$C_{PI}(s) = K_P + \frac{K_I}{s} \tag{2.4}$$

Con un control proporcional, es necesario que exista error para tener una acción de control distinta de cero. Con accion integral, un error pequeño positivo siempre nos dará una acción de control creciente, y si fuera negativo la señal de control será decreciente. Este razonamiento sencillo nos muestra que el error en régimen permanente será siempre cero.

Muchos controladores industriales solo tienen acción PI. Se puede demostrar que un control PI es adecuado para todos los procesos donde la dinámica es esencialmente de primer orden.

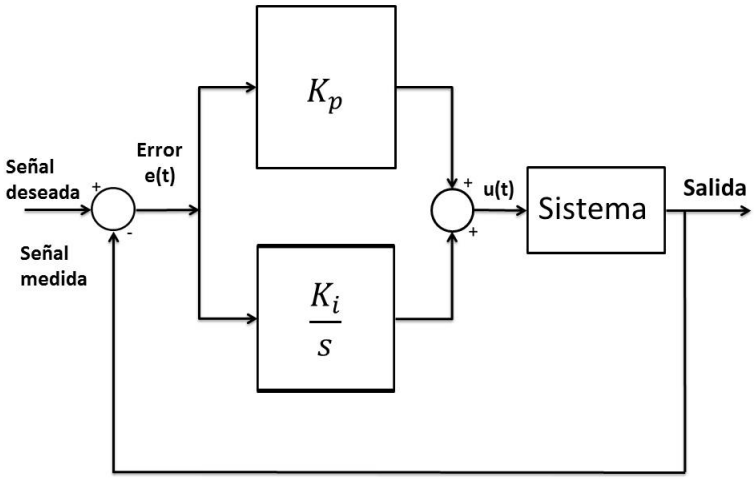


Figura 2.20: Esquema de un controlador proporcional-integral

2.13.4. Controlador de acción Proporcional-Derivativa

La acción de control proporcional-derivativa (PD), se define mediante:

$$u(t) = K_P e(t) + K_D \frac{de(t)}{dt} \tag{2.5}$$

donde Td es una constante de denominada tiempo derivativo. Esta acción tiene carácter de previsión, lo que hace más rápida la acción de control, aunque tiene la desventaja importante que amplifica las señales de ruido y puede provocar saturación en el actuador. La acción de control derivativa nunca se utiliza por sí sola, debido a que solo es eficaz durante períodos transitorios. La función de transferencia de un controlador PD resulta:

$$C_{PD}(s) = K_P + K_D s \tag{2.6}$$

Cuando una acción de control derivativa se agrega a un controlador proporcional, permite obtener un controlador de alta sensibilidad, es decir que responde a la velocidad del cambio de error y produce una corrección significativa antes de que la magnitud

del error se vuelva demasiado grande. Aunque el control derivativo no afecta en forma directa al error ea estado estacionario, añade amortiguamiento al sistema y, por lo tanto, permite un valor más grande que la ganancia K, lo cual provoca una mejora en la precisión en estado estable.

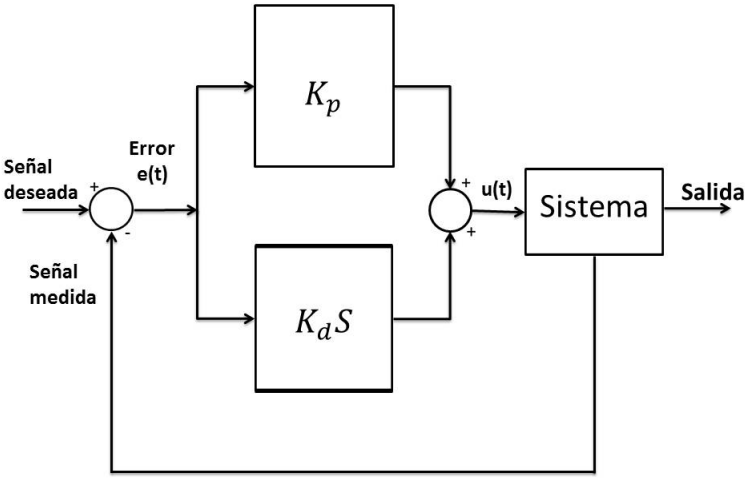


Figura 2.21: Esquema de un controlador proporcional-derivativo

2.13.5. Controlador de acción de control Proporcional-Integral-Derivativa

Esta acción combinada reune las ventajas de cada una de las tres acciones de control individuales. La ecuación de un controlador con ésta acción combinada se obtiene mediante:

$$u(t) = K_P e(t) + K_D \frac{de(t)}{dt} + K_I \int_{x=0}^{x=t} e(\tau) \cdot d\tau \tag{2.7}$$

y su función transferencia resulta:

$$C_{PID}(s) = K_P + \frac{K_I}{s} + K_D s \tag{2.8}$$

El objetivo de los ajustes de los parámetros PID es lograr que el bucle de control corrija eficazmente y en el mínimo tiempo los efectos de las perturbaciones; se tiene que

lograr la mínima integral de error. Si los parámetros del controlador PID (la ganancia del proporcional, integral y derivativo) se eligen incorrectamente, el proceso a controlar puede ser inestable, por ejemplo, que la salida de este varíe, con o sin oscilación, y está limitada solo por saturación o rotura mecánica. Ajustar un lazo de control significa ajustar los parámetros del sistema de control a los valores óptimos para la respuesta del sistema de control deseada. El comportamiento óptimo ante un cambio del proceso o cambio del $set - point$ varía dependiendo de la aplicación.

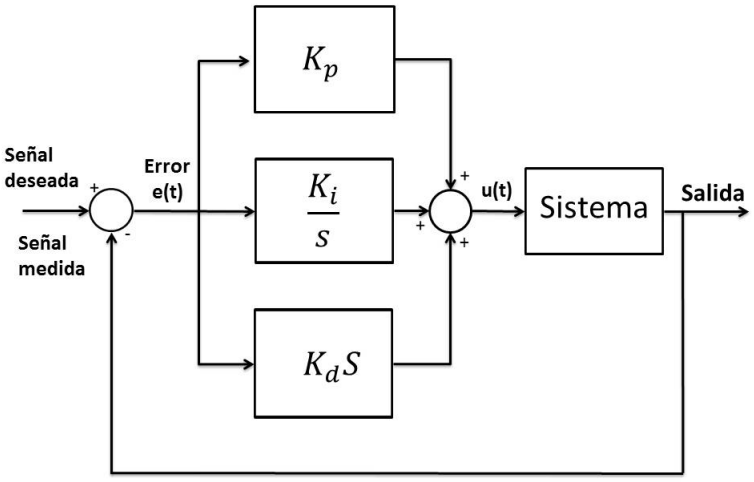


Figura 2.22: Esquema de un controlador PID

2.14. Modelo matemático del cuadróptero

Existen diversos modelos matemáticos para describir la dinámica del cuadróptero. El modelo matemático en el que se basa este trabajo es bajo la metodología de Newton-Euler [21]. Partiendo de que se tienen 6 grados de libertad los cuales son:

$$q = (x, y, z, \phi, \theta, \psi) \tag{2.9}$$

Donde (x, y, z) denotan la posición del centro de masa del cuadróptero relativo

al marco;(ψ, θ, ϕ) son los ángulos de Euler (cabeceo, alabeo y guiñada)y representan la orientación del vehículo aéreo. Por lo tanto podemos representar la ecuación en coordenadas rectangulares y rotacionales de la siguiente forma:

$$\xi = (x, y, z)\,, \eta = (\phi, \theta, \psi) \tag{2.10}$$

Mediante el análisis matemático de Newton-Euler podemos obtener el modelo completo de la dinámica del cuadróptero con el movimiento en x, y, z como consecuencia de la rotación es:

$$m\ddot{x} = (\cos\phi \sin\theta \cos\psi + \sin\phi \sin\psi)\, U_1, \tag{2.11}$$

$$m\ddot{y} = (\cos\phi \sin\theta \sin\psi + \cos\phi \sin\psi)\, U_1, \tag{2.12}$$

$$m\ddot{z} = -mg + \cos\phi \cos\theta U_1, \tag{2.13}$$

$$I_{xx}\ddot{\phi} = \dot{\theta}\dot{\psi}\,(I_{yy} - I_{zz}) - J_r\dot{\theta}\Omega + lU_2, \tag{2.14}$$

$$I_{yy}\ddot{\theta} = \dot{\phi}\dot{\psi}\,(I_{zz} - I_{xx}) - J_r\dot{\phi}\Omega + lU_3, \tag{2.15}$$

$$I_{zz}\ddot{\psi} = \dot{\phi}\dot{\theta}\,(I_{xx} - I_{yy}) + U_4 \tag{2.16}$$

Donde encontramos que:

$$U_1 = b\left(\Omega_1^2 + \Omega_2^2 + \Omega_3^2 + \Omega_4^2\right), \tag{2.17}$$

$$U_2 = b\left(\Omega_4^2 - \Omega_2^2\right), \tag{2.18}$$

$$U_3 = b\left(\Omega_4^2 - \Omega_1^2\right), \tag{2.19}$$

$$U_4 = d\left(\Omega_2^2 + \Omega_4^2 - \Omega_1^2 - \Omega_3^2\right), \tag{2.20}$$

$$\Omega = (\Omega_2 + \Omega_4 - \Omega_1 - \Omega_3) \tag{2.21}$$

Los efectos físicos que actuan sobre el cuadróptero se describen en la siguiente tabla:

Momentos debido al alabeo	Expresión matemática
Efecto giroscópico del cuerpo	$\dot{\theta}\dot{\psi}\,(I_{yy} - I_{zz})$
Efecto giroscópico de las propelas	$J_r\dot{\theta}\Omega$
Momentos debido al cabeceo	Expresión matemática
Efecto giroscópico del cuerpo	$\dot{\phi}\dot{\psi}\,(I_{zz} - I_{xx})$
Efecto giroscópico de las propelas	$J_r\dot{\phi}\Omega$
Momentos debido al guiño	Expresión matemática
Efecto giroscópico del cuerpo	$\dot{\phi}\dot{\theta}\,(I_{xx} - I_{yy})$
Otros	Expresión matemática
Energia potencial	U_i
Velocidad del rotor	Ω_i
Factor de arrastre	b

Tabla 2.2: Efectos físicos que actuan sobre el cuadróptero.

Capítulo 3

Control de motores del cuadróptero

Para realizar la parte experimental, se realizará la siguiente medolología con el proposito de cumplir los objetivos planteados en el capítulo 1.

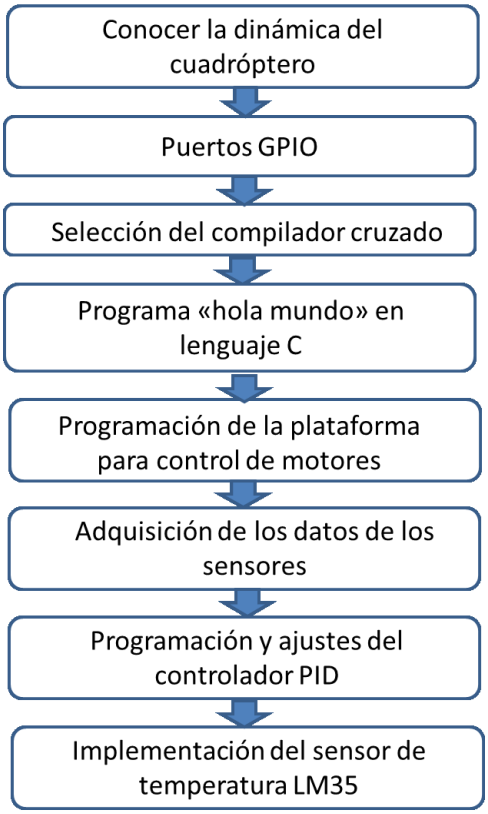


Figura 3.1: Metodología de experimentación

3.1. Cuadróptero Parrot AR Drone

El cuadróptero Parrot AR Drone es un vehículo aéreo no tripulado radiocontrolado de uso recreativo. Funciona propulsado por cuatro motores eléctricos en configuración cuadricóptero y es similar en su estructura básica y aerodinámica a otros modelos radiocontrolados, pero se diferencia de todos ellos en que cuenta con un microprocesador y una serie de sensores entre los cuales se incluyen dos cámaras que le permiten captar lo que ocurre a su alrededor, más un conector Wi-Fi integrado que le permite vincularse a dispositivos móviles personales que cuenten con los sistemas operativos iOS, Android o Linux. Esto permite controlar al cuadricóptero directamente desde un dispositivo móvil, mientras se reciben por medio de este dispositivo las imágenes y datos de telemetría de lo que los sensores del drone están captando.

El drone fue diseñado originalmente para ser controlado por medio de los productos de Apple con iOS tales como los iPhone, iPad y iPod Touch y para dispositivos Android.2 Pero debido a que la empresa fabricante del drone liberó los applet de control bajo código abierto pronto aparecieron aplicaciones para otros dispositivos tales como el Samsung BADA y también algunas aplicaciones no oficiales para Symbian,3 actualmente està disponible un paquete de desarrollo oficial, de descarga libre, que permite utilizar el drone por medio de una PC portátil equipada con Linux.

Figura 3.2: Parrot AR Drone

3.1.1. Sistema operativo del cuadróptero Parrot AR Drone

El sistema operativo con el que cuenta el cuadróptero Parrot AR drone es linux 2.6.32, una vez accediendo , a través de TELNET, podemos encontrar el siguiente conjunto de directorios propios de un sistema linux.

bin	data	dev
etc	factory	firmware
home	lib	licenses
mnt	proc	root
sbin	sys	tmp
update	usr	var

Tabla 3.1: Directorios del sistema operativo del AR Drone

3.2. Programación de la PWM

Como se mencionó en el capítulo 2, la implementación de una PWM para cada motor fue necesaria para poder controlar sus velocidades. Se plantearon casos críticos para el control: motores en reposo, motores al máximo, motores al mínimo, motores en incremento o decremento.

Esta función se encuentra en el archivo *mot.c* que es el que se encarga de las variables que afectan a los motores.

La función *mot SetPWM* adquiere la señal PWM y la escribe sobre un vector de 8 bits, debido a que la velocidad de los motores puede ser incrementada o decrementada sobre un vector de esa longitud.

```
void mot_SetPWM(u16 m0, u16 m1, u16 m2, u16 m3) {
  pthread_mutex_lock(&mot_mutex);
  mot.pwm[0] = m0 & 0x1ff;
  mot.pwm[1] = m1 & 0x1ff;
```

```
  mot.pwm[2] = m2 & 0x1ff;
  mot.pwm[3] = m3 & 0x1ff;
  pthread_mutex_unlock(&mot_mutex);
 }
```

La función *mot Stop()* surge de la necesidad de frenar los motores en cualquier momento o en una emergencia. Cuando esta función sea llamada, los valores PWM de los motores serán cero, por lo que la actividad de los motores cesará.

```
void mot_Stop()
{
  pthread_mutex_lock(&mot_mutex);
  mot.pwm[0] = 0;
  mot.pwm[1] = 0;
  mot.pwm[2] = 0;
  mot.pwm[3] = 0;
  mot.mot[0]=0;
  mot.mot[1]=0;
  mot.mot[2]=0;
  mot.mot[3]=0;
  pthread_mutex_unlock(&mot_mutex);
}
```

La función mas importante para el control de las velocidades de los motores es *mot Run()*. Esta función modifica el ancho de la PWM para variar la velocidad de un motor en específico. El valor mínimo de arranque es del 40 %, y puede descender hasta el 33 % de velocidad. Si la velocidad desciende aún más, los motores frenarán.

```
void mot_Run(float m0, float m1, float m2, float m3)
{
```

```
  mot.mot[0]=m0;
  mot.mot[1]=m1;
  mot.mot[2]=m2;
  mot.mot[3]=m3;

  float pwm[4];
  for(int i=0;i<4;i++) {
    if(mot.mot[i]<0.0) mot.mot[i]=0.0;
    if(mot.mot[i]>1.0) mot.mot[i]=1.0;
    pwm[i]=mot_pwm_min + mot.mot[i]*(mot_pwm_max-mot_pwm_min);
if(pwm[i]<mot_pwm_min) pwm[i]=mot_pwm_min;
if(pwm[i]>mot_pwm_max) pwm[i]=mot_pwm_max;
  }
```

3.3. GPIO

Los puertos GPIO (General Purpose Input/Output, Entrada/Salida de Propósito General) son pines genéricos en un circuito integrado los cuales pueden ser controlados por el usuario.

Para poder controlar cada uno de los elementos del Parror AR Drone primeramente se deberán conocer dichos puertos. Una vez conociéndolos, se procederá a controlar los puertos que estan designados para los motores y LED's

Algunos de los puertos GPIO con los que se cuenta son:

Puero GPIO	Configuración	Descripción
43	Salida	Reset de WLAN 1=reset
59	Salida	Cámara vertical 1=Habilitado
63	Salida	LED Rojo 1=encendido
64	Salida	LED verde 1=encendido
68	Salida	Motor 1 ; 0=seleccionado, 1=seleccionado
69	Salida	Motor 2 ; 0=seleccionado, 1=seleccionado
70	Salida	Motor 3; 0=seleccionado, 1=seleccionado
71	Salida	Motor 4 ; 0=seleccionado, 1=seleccionado
101	Salida	Cámara Horizontal
106	Entrada	Corte de Motor
107	Salida	Habilitar Motor; 1=Habilitado
108	Entrada	Botón de Reset 1=Reset
109	Salida	Cámara Horizontal 1=Habilitada
127	Salida	Conector USB
130	Salida	Datos de Navegación
131	Salida	Datos de Navegación
132	Salida	Datos de Navegación
158	Entrada	Bóton de paro

Tabla 3.2: Descripción de los puertos GPIO

La configuración de los motores con los puertos GPIO se observa a continuación:

```
//reset del flip-flop IRQ
gpio_set(106,-1);
gpio_set(107,0);
gpio_set(107,1);

//Seleccionar los puertos de los motores
```

```
gpio_set(68,1);
gpio_set(69,1);
gpio_set(70,1);
gpio_set(71,1);

//Configurar motores
int retval=0;
u08 reply[256];
for(int m=0;m<4;m++) {
gpio_set(68+m,-1);
motorboard_cmd(0xe0,reply,2);
if(reply[0]!=0xe0 || reply[1]!=0x00)
{
printf("motor%d pwm=0x%02x \n",m+1,(int)reply[0],(int)reply[1]);
retval=1;
}
motorboard_cmd(m+1,reply,1);
gpio_set(68+m,1);
}
```

En el código anterior podemos observar la coordinación de los puertos GPIO correspondientes a cada motor, así mismo el flip flop IRQ es el encargado de guardar en memoria los datos de fabrica, por lo que es necesario un *reset* para escribir los nuevos algoritmos e instrucciones de control.

3.4. Compilador cruzado

Como se mencionó en el capítulo 2, un compilador cruzado es capaz de transformar un programa escrito en cualquier lenguaje a otra arquitectura, en este caso,*ARM*. Es

importante seleccionar un compilador cruzado adecuado, debido a que ciertos microprocesadores ARM cuentan con una arquitectura diferente, por lo que una mala elección arroja *errores de segmentación*.

El compilador cruzado utilizado para el cuadróptero fue el *arm-none-linux-gnueabi-g++.exe* el cual puede descargarse de la página de *codebench*.

Para poder ejecutar el complilador cruzado se debe escribir en la terminal de ubuntu ó de windows: *arm-none-linux-gnueabi-g++.exe -lpthread -o archivoacompilar.c*

3.4.1. Programa Hola mundo

Se realiza el programa "hola mundo", con el proposito de encontrar el compilador correcto bajo el método de prueba y error. Si el compilador no es el correcto, al ejecutar el programa se observara un error que aparecerá como *segmentation fault*. Si el compilador es el correcto, el programa se ejecutará correctamente.

```
alejandro@alejandro-Inspiron-1011:~$ /opt/CodeSourcery/
Sourcery_CodeBench_Lite_for_ARM_GNU_Linux/bin/
arm-none-linux-gnueabi-gcc -march=armv7-a hello.c -o hello.elf
```

3.4.2. Compilación cruzada del la plataforma de control de motores

Una vez seleccionado el compilador cruzado correcto, se realizará la complilación de los distintos programas que componen la plataforma de control. Para lograr esto es necesario escribir en la terminal (windows ó linux) el siguiente comando:*arm-none-linux-gnueabi-g++.exe-lpthread -o../bin/motorboard/motorboard.c mot.c../gpio/gpio.c main-motorboard.c*

Donde todos los programas seras transformados a lenguaje binario para que el microcontrolador ARM pueda interpretar las instrucciones.

Una vez hecha la compilación se obtendrá un archivo binario llamado "motorboard", sin embargo este no puede ser ejecutado debido a que se deben dar los permisos de ejecución con el comando *chmod 777 motorboard*. Es así como obtendremos el archivo listo para ejecutarse.

3.4.3. Creación del servidor FTP

Para la transferencia de archivos hacia el cuadróptero, fue necesario crear un servidor FTP, dicho servidor fue creado en el sistema operativo Ubuntu 13.10 bajo los siguientes pasos:

Se escribió el comando sudo aptitude install vsftpd para comenzar con la instalacion del servidor. Una vez finalizada la instalación, se procedió a configurar el servidor editando el archivo */etc/vsftpd.conf*. Para editar este archivo se debe escribir el comando *gksu gedit /etc/vsftpd.conf*

Ya configurado el servidor, se debe denegar el acceso FTP anónimos cambiando la línea de código *anonymous enable=YES* por *anonymous enable=NO*.

Permitir a los usuarios locales acceder al servidor, cambiando la línea de código *local enable=NO* por *write enable=YES*.

Así mismo, para permitir que el usuario suba archivos al servidor se debe cambiar la línea de código *write enable=NO* por *write enable=YES*.

Una vez hechos estos cambios, se debe reiniciar el servidor.

3.5. Formas de comunicarse con el cuadróptero

Con respecto a la comunicación del AR Drone con los diferentes dispositivos, la transmisión se realiza cuando éstos se conectan a la wifi del drone. Según los tipos de datos transmitidos, la emisión se realiza mediante un vía u otra.
Existen cuatro vías principales:
Para controlar y configurar el AR Drone se utilizan los AT Commands, que se transfieren

por el puerto UDP 5556 con una frecuencia de 30 veces por segundo.

La información sobre el estado del AR Drone está contenido en *Navdata*, bajo puerto UDP 5554 y en dos frecuencias: 15 veces por segundo en modo *demo* y 200 veces por segundo en modo *debug*. Éste último se utiliza en la creación de juegos de realidad aumentada.

La transmisión de video se realiza bajo TCP en el puerto 5555. Las imágenes que recibe el dispositivo pueden decodificarse usando el códec incluido en el SDK.

Los Datos críticos van al puerto de Control TCP 5559.

La transferencia de archivos se realiza a través del puerto TCP 5551.

Puerto	Número	Función
UDP	5556	Control con comando AT
UDP	5554	Datos de navegación
TCP	5555	Transmisión de video
TCP	5559	Datos críticos
FTP	5551	Transferencia de archivos

Tabla 3.3: Descripción de puertos del microcontrolador ARM

3.6. Control de los motores

3.6.1. Aceleración

El motor recibe corriente a través del sistema eléctrico del cuadróptero, dicha corriente pasa del procesador central hasta el motor a través del controlador del motor. Como consecuencia de la transmisión de corriente al motor, éste hace girar el eje del mismo, que a su vez, hace mover los engranes a través del conjunto mecánico, formado por: el piñón, la corona, el eje y las ruedas.

3.6.2. Desaceleración

Consiste en la carencia de suministro de corriente al motor, o bien, en un menor suministro de corriente al controlador del motor, en cualquiera de los dos casos, debido a la inercia del conjunto engrane-motor, el motor tarda en desacelerar o detenerse. Sin embargo, mientras el engrane principal se desplaza sin corriente, o con menos corriente de la necesaria, según el caso, el motor deja de funcionar como tal, pues al continuar girando por las ruedas pasa a funcionar como generador, ya que se suministra menos corriente al motor de la que necesita para alcanzar su velocidad deseada, por lo cual genera corriente en vez de ser absorbida. Todo ello hace, que aparezca una corriente en la pista generada por dicho motor que ahora funciona como generador, la corriente generada, pasando un espacio de tiempo, durante el cual, el sistema vuelve al equilibrio, será entonces, cuando el motor funcionará como tal, dejando de hacerlo como generador.

3.6.3. Frenado

El frenado consiste en consumir la energía generada por el motor durante la desaceleración lo más rápidamente posible, debido a que cuanto mayor sea la demanda de energía eléctrica por el generador, mayor será la inercia necesaria para que éste, siga en movimiento. Como la inercia es aproximadamente la misma, siempre que desaceleremos, si aumentamos la demanda de energía eléctrica conseguiremos que el engrane principal se detenga antes, pues obligamos al generador a consumir más movimiento para mantener esa energía eléctrica. La forma de incrementar la demanda de energía eléctrica, será provocando un cortocircuito temporal en el motor. Este cortocircuito temporal, se puede controlar por medio de un interruptor o mediante el teclado de la computadora, como en el caso de este proyecto.

Bajo estos conceptos se programa la plataforma de control de motores, en la cual los motore inician a un 50 % de su velocidad para asegurar el correcto arranque (la velocidad mínima de arranque es 40 %).

```
if(c=='q') break;
```

```
if(c==’1’) {
printf("\r Motor1 al 50%           ");
throttle1 = .50;
throttle2 = 0;
throttle3 = 0;
throttle4 = 0;
mot_Run(throttle1,throttle2,throttle3,throttle4);
}
if(c==’2’) {
printf("\r Motor2 al 50%           ");
throttle1 = 0;
throttle2 = .50;
throttle3 = 0;
throttle4 = 0;
mot_Run(throttle1,throttle2,throttle3,throttle4);
}
if(c==’3’) {
printf("\r Motor3 al 50%           ");
throttle1 = 0;
throttle2 = 0;
throttle3 = .50;
throttle4 = 0;
mot_Run(throttle1,throttle2,throttle3,throttle4);
}
if(c==’4’) {
printf("\r Motor4 al 50%           ");
throttle1 = 0;
throttle2 = 0;
throttle3 = 0;
```

```
throttle4 = .50;
mot_Run(throttle1,throttle2,throttle3,throttle4);
}
if(c=='5') {
printf("\r Todos los motores al 50%           ");
throttle1 = .50;
throttle2 = .50;
throttle3 = .50;
throttle4 = .50;
mot_Run(throttle1,throttle2,throttle3,throttle4);
}
```

3.7. Adquisición de los datos de los sensores

Una vez que se realiza un vuelo, los sensores trabajan en conjunto para poder tener la mayor estabilidad posible. Es posible obtener los datos de cada uno de los sensores para poder interpretar la dinámica del vuelo y poder realizar e implementar acciones de control y corrección. Los datos de los sensores se pueden adquirir a través del puerto UDP 5556, a través de la dirección IP 127.0.0.1 como lo describe el código siguiente.

```
#include <unistd.h>
#include <stdlib.h>
#include <string.h>
#include <stdio.h>
#include "udp.h"

void diep(const char *s)
{
  perror(s);
  exit(1);
```

```
}

int main(void)
{
  udp_struct udp;

  int msglen;
  char buf[512];

  if(udpClient_Init(&udp,"127.0.0.1",9930)) diep("udpClient_Init");

  for (int i=0; i<10; i++) {
    printf((char*)"Enviar paquete %d\n", i);
    msglen=sprintf(buf, (char*);
    if (udpClient_Send(&udp, buf, msglen)) diep("Enviar");
    usleep(100000);
  }

  udpClient_Close(&udp);
  return 0;
}
```

3.8. Implementación de un controlador PID en el AR Drone

Para mejorar la posición del dron ante cualquier perturbación, se propone implementar un controlador PID. La función de transferencia del controlador es:

$$C_{PID}(s) = \frac{K_D s^2 + s(K_P + \beta K_I) + K_I}{s(\beta s + 1)} \tag{3.1}$$

en donde: K_P es la constante proporcional, K_D es la constante derivativa. K_I es la constante integral.β es la constante de amortigamiento.

Para realizar la fase experimental se contó con un modelo en el cual las constantes fueron:

K_P=0.6, K_D=1.5, K_I=0.0795, β=0.01.

Sin embargo, en la implementación del controlador en el cuadróptero se observó una ligera deriva en el eje longitudinal debido a perturbaciones externas como viento y brisa marina.

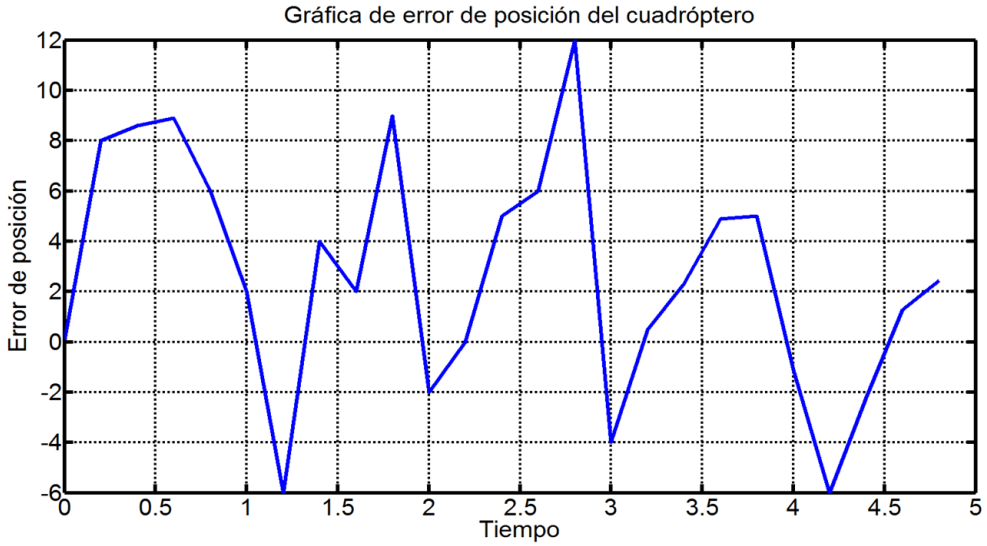


Figura 3.3: Error de posición del cuadróptero

Por esta razón, se optó por cambiar las variables del controlador bajo el metodo de *prueba y error*. La primera variable que se modificó fue K_D debido a que es la constante que controla la deriva en vuelo. Enseguida se modificó K_P con el objetivo de reducir el sobretiro inicial.La constante de amortiguamiento β se redujo en un orden de magnitud ya que se observó que el cuadróptero tardaba mucho tiempo en estabilizarse.

Las constantes encontradas para la optimización del vuelo son: K_P=0.499, K_D=1, K_I=0.0795, β=0.001.

Sustituyendo lo valores en la función de transferencia obtendremos:

$$C_{PID}(s) = \frac{s^2 + 0.5s + 0.0795}{s(0.001s + 1)} \tag{3.2}$$

Cuya respuesta es:

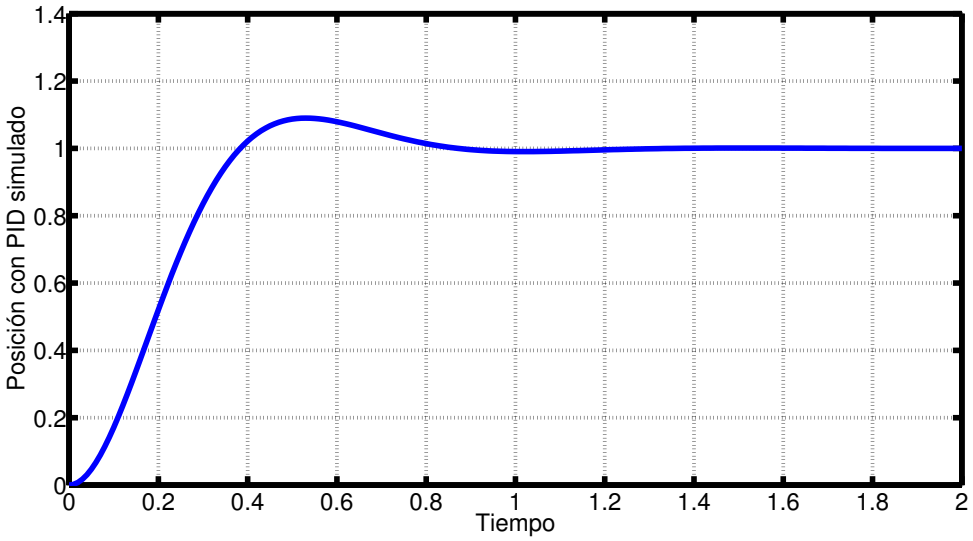


Figura 3.4: Simulación del controlador PID

En la figura 3.4 se puede apreciar que el sobretiro es de 9.1 % y el tiempo de asentamiento de aproximadamente 1 segundo lo cual es una respuesta deseable y se buscará llevar a cabo experimentalmente.

Definimos la estructura ya mencionada del controlador PID en el programa *pid.C* para poder implementarlo en el cuadróptero.

```
void pid_Init(pid_struct *pid, float kp, float ki, float kd, float i_max)
{
pid->kp=0.499;
pid->ki=0.0795;
pid->kd=1;
pid->beta=0.001;
```

```
pid->i=0;
pid->e_prev=0;
}
float pid_CalcD(pid_struct *pid, float error, float dt, float d)
{
pid->i += error * dt;
if(pid->i > pid->beta) pid->i = pid->beta;
if(pid->i < -pid->beta) pid->i = -pid->beta;
float out = pid->kp * error + pid->ki * pid->i + pid->kd * d;
pid->e_prev = error;
return out;
}
```

3.9. Sensor de temperatura en el AR Drone

El AR Drone cuenta con múltiples sensores capaces de medir variables ambientales, sin embargo, dicho drone no cuenta con un sensor de temperatura, por esta razón, se propone implemetar el sensor LM35 debido a que es fácil de instrumentar y no requiere de una calibración muy exhaustiva.

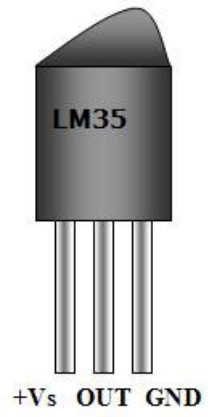


Figura 3.5: Sensor de temperatura LM35

La implementación de este sensor se realizó con la ayuda de la plataforma de hardware libre llamada Arduino, la cual cuenta con un microcontrolador ATmega 328, puertos de entrada y salida, además de un entorno de desarrollo que implementa el lenguaje de programación basado en C.

Figura 3.6: Placa de dasarrollo Arduino

EL AR Drone 1.0 cuenta con unos puertos de expansión los cuales nos ayudarán a comunicar la tarjeta madre con la placa de desarrollo arduino a través del puerto serial. Estos puertos desarrollan funciones específicas las cuales se describen en la tabla 3.4.

Puerto	Función	Valores y descripción
1	Vusb	+5V (cable rojo del USB)
2	Vbat	Voltaje de batería +11.7 V
3	USB D-	Cable blanco del USB
4	RX	Entrada del puerto serial del drone
5	USB D+	Cable verde del USB
6	TX	Salida del puerto serial del drone
7	GND	Tierra (cable negro del USB)
8	Nulo	Nulo

Tabla 3.4: Descripción de puertos de expansión del drone.

Para poder implementar correctamente el sensor de temperatura LM35 fue necesario

utilizar un factor de corrección proporcionado en la hoja de datos del fabricante el cual se describe en el programa como *correcFactor = 0.768744354*

El programa que permite que se pueda adquirir la temperatura por el cuadróptero es:

```
int analTempPin = A0 ;
float temp = 0 ;
floatcorrectFactor=0.768744354;

void setup ( ) {
Serial.begin (9600);
}
void loop ( ) {
temp = analogRead(analTempPin) ;
temp = ((5.0*c o r r e cFa c t o r) * temp * 100.0)/1024.0;
Serial.println(temp);
delay (200);
}
```

Capítulo 4

Pruebas y resultados

En éste capítulo se presentan las pruebas y resultados obtenidos con el prototipo del programa de control de motores desarrollado. Para comprobar el funcionamiento del programa se procedió a transferir los programas realizados a través del programa *Filezilla* por el puerto FTP 5551 del Parrot AR Drone.

4.1. Realización del primer programa

El primer programa realizado es un "hola mundo". El proposito de este programa es encontrar el compilador cruzado adecuado bajo el metodo de prueba y error, sobre todo que coincida con la arquitectura ARM del cuadróptero.

4.1.1. Programa de prueba

Para evaluar el funcionamiento del cross compilador y vigilar que el proceso de transferencia del servidor FTP hacia el cuadróptero sea el correcto, se implemento el programa *Hola Mundo* siguiendo la metodología propuesta.

El primer paso fue la elaboración del código en lenguaje C. Una vez compilado, se comprueba que no contenga errores y se carga dicho programa por el compilador cruzado para obtener el archivo ejecutable compatible con la arquitectura ARM del

microprocesador del cuadróptero.

Como siguiente paso, se transfiere el arhivo ejecutable a través del puerto FTP 5551 del cuadróptero.

Cuando el proceso de transferencia termina, se procede a abrir una sesión TELNET bajo la ip 192.168.1.1, como se mencionó en el capítulo 2.

El archivo ejecutable que se transfiere se encontrará en el folder *UPDATE*, al localizarlo basta con escribir en la terminal el comando *chmod 777 hello.elf* para que el archivo pueda ejecutarse correctamente.

Para ejecutar el archivo se escribe el comando *./hello.elf*, de esta forma el cuadróptero arrojará la siguiente respuesta:

```
BusyBox v1.14.0 () built-in shell (ash)
Enter 'help' for a list of built-in commands.

# cd update
# cd bin
/bin/sh: cd: can't cd to bin
# cd
# cd update
# ls -l
-rw-r--r--    1 root     root          5701 Jan  1 00:03 hello.elf
-rw-r--r--    1 root     root             7 Jan  1 00:00 version.txt
# chmod 777 hello.elf
# ls -l
-rwxrwxrwx    1 root     root          5701 Jan  1 00:03 hello.elf
-rw-r--r--    1 root     root             7 Jan  1 00:00 version.txt
# ./hello.elf
Hola Mundo#
```

Figura 4.1: Ejecución del programa Hola mundo

4.2. Limitaciones de comunicación

Como se mencionó en el capitulo 3, el cuadróptero Ar.Drone 1.0 utiliza el estándar de comunicación IEEE 802.11 b/g. Una de las principales desventajas de este protocolo es la baja eficiencia de conexión en el exterior debido a agentes dispersores de la señal (aire, brisa, arboles, edificios). Teóricamente, este protocolo puede mantener conexiones hasta una distancia de 400 metros en exterior. Sin embargo, durante las pruebas realizadas, se observó que se puede mantener comunicación con el cuadróptero hasta una distancia

aproximada de 50 metros en dirección planar y de 15 metros en dirección vertical. Esta limitación se debe tomar en cuenta cuando se pretenda realizar una rutina de vuelo que exceda estas dimensiones.

4.2.1. PWM de los motores

Con el objetivo de controlar cada motor, se implementó una PWM para cada motor inicializándolos en 0. Así mismo, una de las ventajas de esta modulación es que puede controlar la velocidad de los motores, sin embargo, cuando se inicie un vuelo, los motores deberán trabajar al menos al 40 % de su velocidad, de otra forma el cuadróptero no despagará.

Si se supera el umbral del 40 % los motores arrancarán por lo que su velocidad podra ascender hasta el 100 % y descender hasta el 33 %, el cual es el valor mínimo de operabilidad de los motores.

Para asegurar la correcta operabilidad en modo de vuelo, se estableció como parámetro de inicio la velocidad de los motores al 50 %, de esa forma se asegura que el cuadróptero inicie correctamente su vuelo a una velocidad adecuada.

```
BusyBox v1.14.0 (2012-06-01 16:35:43 CEST) built-in shell (ash)
Enter 'help' for a list of built-in commands.

# cd update
# cd bin
# ./motorboard
programa control de motor ... Presione q para salir
Motor: 1,2,3,4=correr cada motor al 50% 5=correr todos los motores al 50% ,=desa
celerar al 1% .0=acelerar al 1 8.401459E-310spacio=parar motores
Leds:  a=leds apagados  s=modo vuelo  d=modo emergencia
motor1   pwm[1]=0
motor2   pwm[2]=0
motor3   pwm[3]=0
motor4   pwm[4]=0
```

Figura 4.2: Inicialización de PWM de los motores

4.2.2. Control de los motores

El control autónomo de los motores se realizó exitosamente llevando a cabo una prueba en donde los motores giran independientemente. Se estableció que el arranque estuviera por encima del 40 % donde al acelerar se llegó al 100 % de la velocidad de

los motores. Al descender en la velocidad se observó un frenado repentino cuando los motores llegaron al 32 % de su velocidad, esto se debe a que el valor minimo de operabilidad es el 33 %.

```
BusyBox v1.14.0 (2012-06-01 16:35:43 CEST) built-in shell (ash)
Enter 'help' for a list of built-in commands.

# cd update
# cd bin
# killall program.elf
# ./motorboard
programa control de motor ... Presione q para salir
Motor: 1,2,3,4=correr cada motor al 50% 5=correr todos los motores al 50% ,=desa
celerar al 1% .0=acelerar al 1 8.401459E-310spacio=parar motores
Leds:  a=leds apagados  s=modo vuelo  d=modo emergencia
    Motor1 50   1
```

Figura 4.3: Control del motor 1

```
BusyBox v1.14.0 (2012-06-01 16:35:43 CEST) built-in shell (ash)
Enter 'help' for a list of built-in commands.

# cd update
# cd bin
# killall program.elf
# ./motorboard
programa control de motor ... Presione q para salir
Motor: 1,2,3,4=correr cada motor al 50% 5=correr todos los motores al 50% ,=desa
celerar al 1% .0=acelerar al 1 8.401459E-310spacio=parar motores
Leds:  a=leds apagados  s=modo vuelo  d=modo emergencia
    Motor2 50   2
```

Figura 4.4: Control del motor 2

4.3. Control PID

Para comprobar el funcionamiento de la plataforma desarrollada, se implementó un controlador PID para controlar los ángulos y posición del cuadróptero.

Las variables del controlador PID utilizadas fueron: $K_P = 0.499$, $K_I = 0.0795$, $K_D = 1$,$\beta = 0.001$ y puestas en marcha en lenguaje C.

Se realizaron prueba en el exterior bajo condiciones de mucho viento y en un lugar cerrado para comprobar la acción del controlador implementado.

Los datos fueron obtenidos a través del puerto UDP 5556, el cual se encarga de recopilar todos datos del cuadróptero durante el vuelo.

4.3.1. Pruebas en exterior

Se realizó una prueba al aire libre bajo condiciones de mucho viento y brisa marina, dicha prueba fue realizada en el estacionamiento de la Facultad de Ciencias Marinas de la Universidad Autónoma de Baja California obteniendo los siguientes datos:

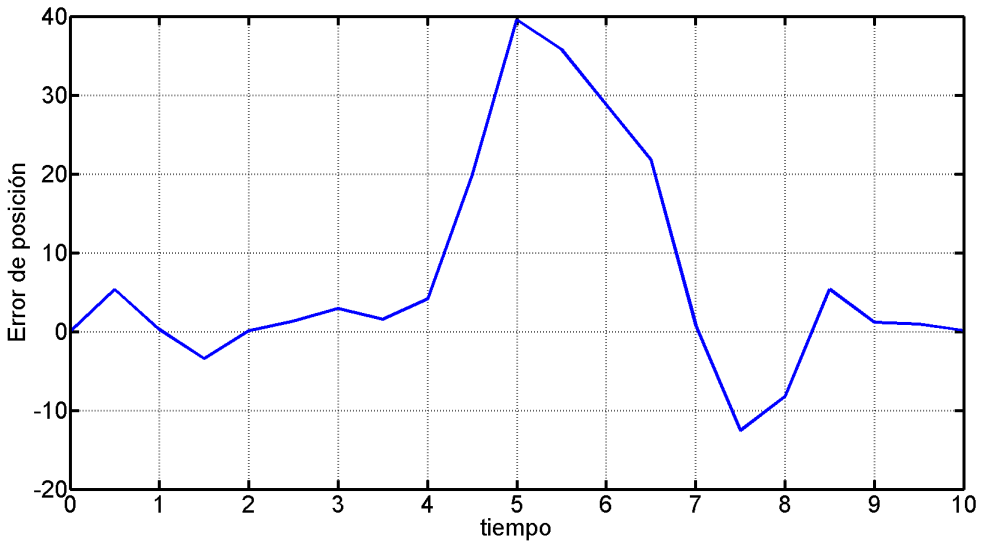


Figura 4.5: Error de posición

En la fígura 4.5 se tiene el error de posición, se puede percibir que el controlador estabiliza la nave ante perturbaciones como las que se pueden observar en la gráfica aproximadamente a los 5 segundos de vuelo.

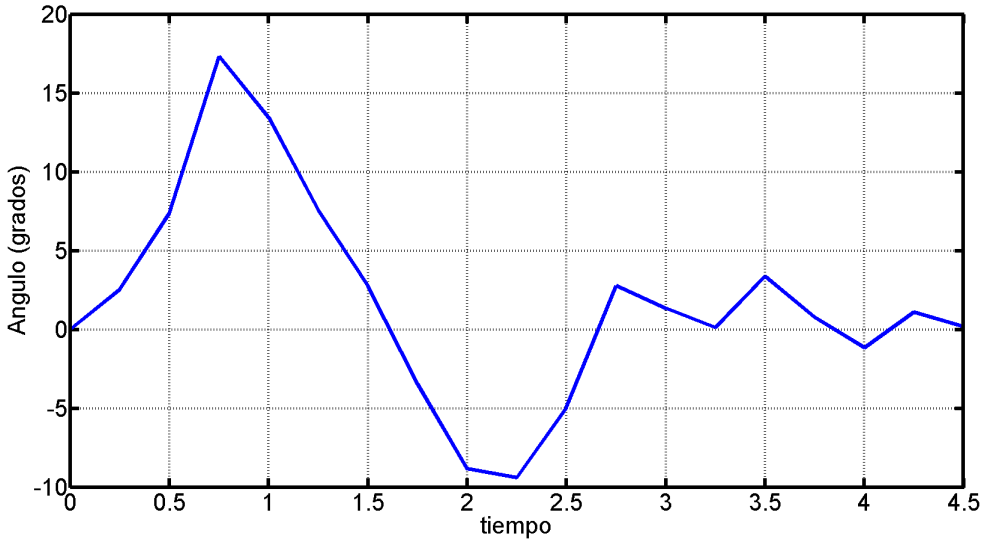


Figura 4.6: Ángulo de alabeo (roll)

En la fígura 4.6 se observa una perturbación inicial debido al viento y brisa marina, el controlador actua rapidamente estabilizando el drone a los 0°.

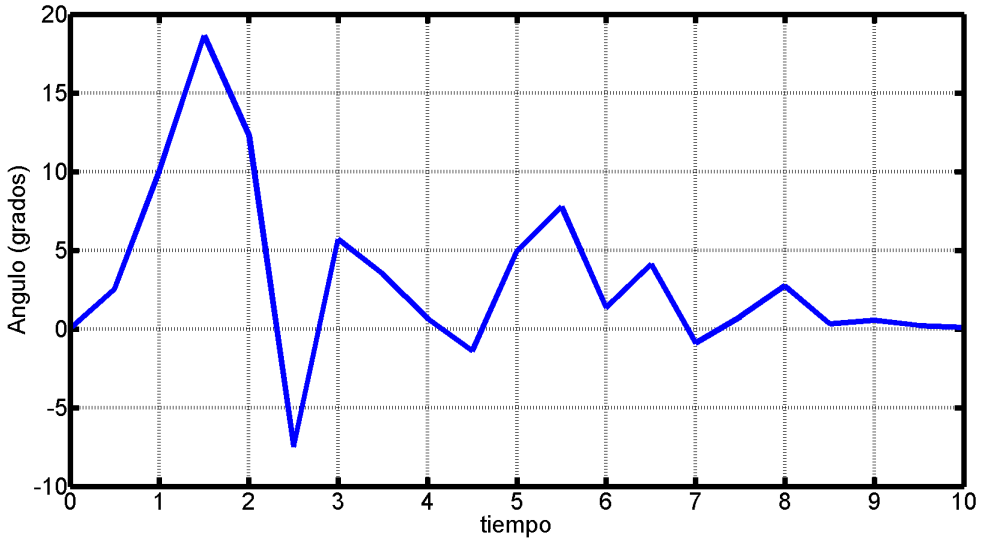


Figura 4.7: Ángulo de guiñada (yaw)

La fígura 4.7 muestra las perturbaciones que existen al inicio del vuelo, se observa

que el drone comienza a estabilizarse alrededor de 6 segundos de vuelo.

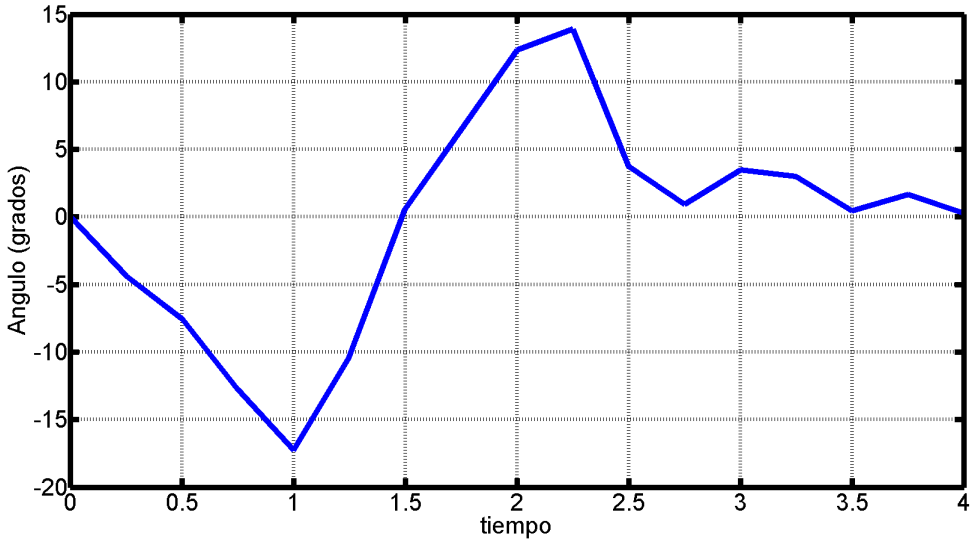


Figura 4.8: Ángulo de cabeceo (pitch)

La fígura 4.8 muestra las perturbaciones en el ángulo de cabeceo, se observa que alrededor de los dos segundos hay una perturbación que afecta la posición del cuadróptero. El controlador PID entra en acción inmediatamente posicionando el vehículo en la referencia (0°).

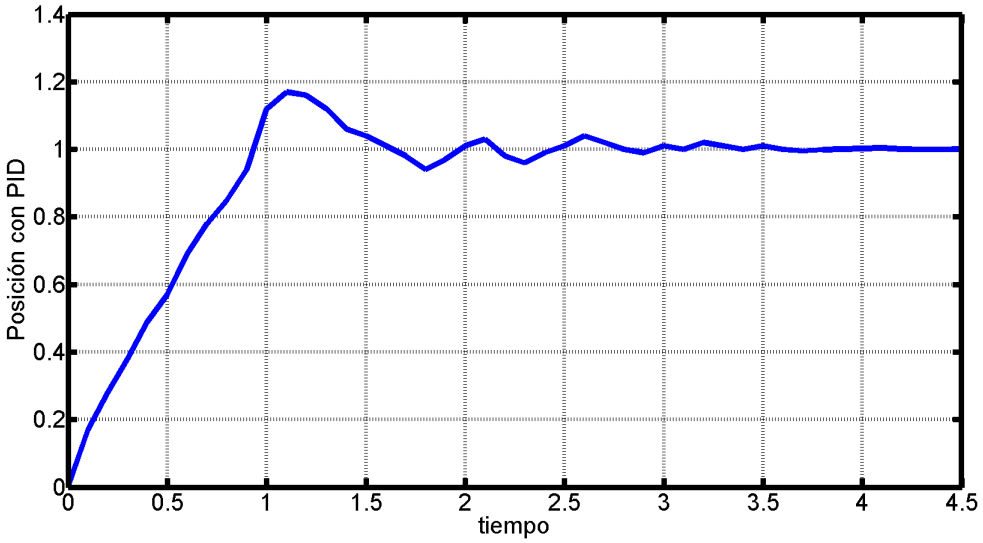


Figura 4.9: Posición con el controlador PID

En la fígura 4.9 se observa la acción del controlador PID. En esta prueba se posicionó el drone a un metro de altura para observar fácilmente el sobretiro. Dicho sobretiro es del 18 % y tiene un tiempo aproximado de asentamineto de 2.5 segundos.

4.3.2. Pruebas en interior

Se realizaron pruebas de vuelo en un ambiente sin perturbaciones, en particular, en el laboratorio de comunicaciones ópticas de la Facultad de Ingeniería, Arquitectura y Diseño (FIAD) en la Universidad Autónoma de Baja California, obteniendo los siguientes resultados:

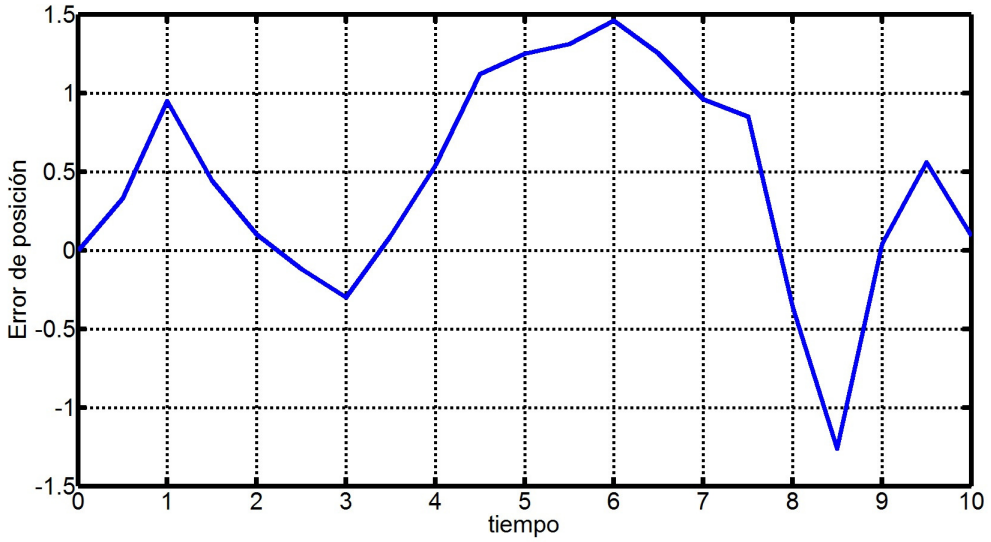


Figura 4.10: Error de posición con el controlador PID en interior

En la fígura 4.10 se puede observar el error de posición al inicio del vuelo, se percibe que, con respecto a la prueba en exterior, el error en grados es mucho menor, obteniendo un pico maximo de 1.5 cm y uno mínimo en -1.36 cm siempre cercano a la referencia.

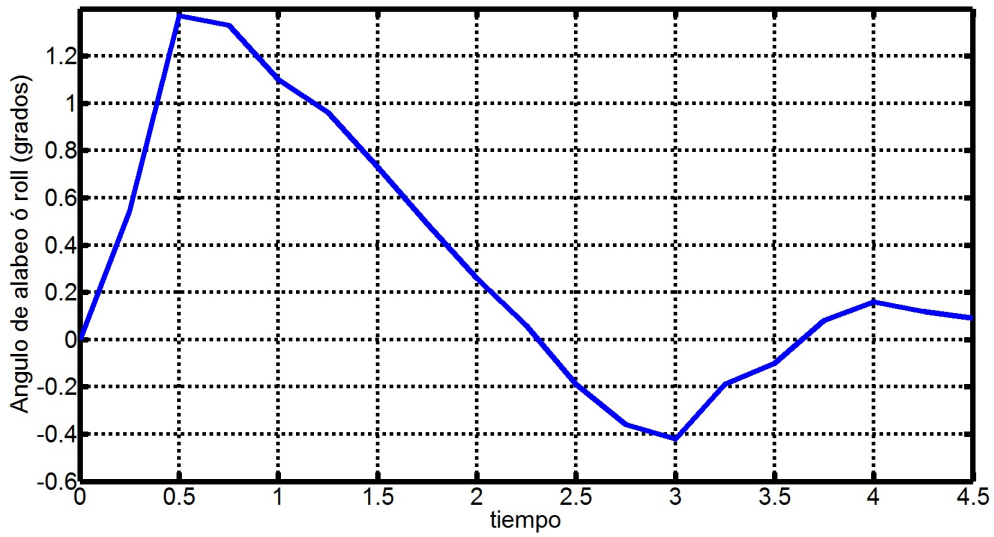


Figura 4.11: Ángulo de alabeo (roll)

En la fígura 4.11 se observa la dinámica del ángulo de alabeo, la diferencia con

respecto a la prueba en el exterior es que las perturbaciones son mínimas, es por esta razón que el ángulo varía entre 1.4°y -0.42°, es decir, muy cercano a la referencia.

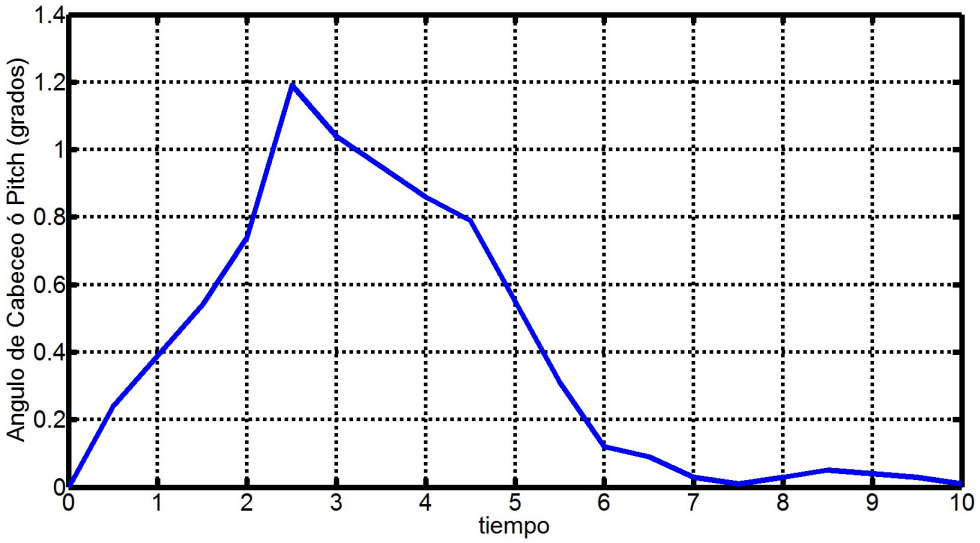


Figura 4.12: Ángulo de cabeceo (pitch)

En la fígura 4.12 se observa el comportamiento del ángulo de cabeceo o pitch, el pico maximo esta a 1.2°, lo cual es muy difícil de percibir durante la ejecución del vuelo, tambien se observa que el cabeceo nunca esta por debajo de la referencia, lo cual es una respuesta favorable debido a que se busca que el drone se comporte de manera estable durante el vuelo.

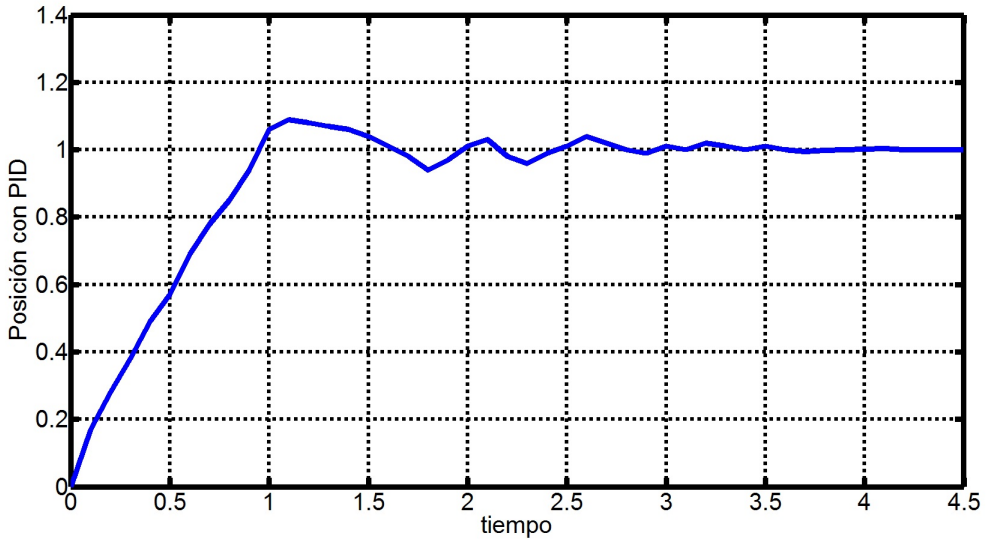


Figura 4.13: Posición con el controlador PID

En el experimento que muestra la gráfica de la figura 4.13, se realizó un procedimiento similar al de la prueba en exterior, posicionando el cuadróptero a un metro de altura para poder medir el sobretiro que existe cuando se ejecuta un vuelo. Se puede observar que el sobretiro es de aproximadamente 10 % y un tiene un tiempo de asentamiento aproximado de 2.5 segundos. En comparación con la prueba que se realizó en el exterior, ésta respuesta muestra menor sobretiro, esto se debe a que existen menos perturbaciones y a que los sensores se encuentran en condiciones mas adecuadas.

En comparación con la simulación del capitulo 3, donde el sobretiro es del 9.1 %, se observa que el sobretiro es similar, aproximadamente 10 %, sin embargo el tiempo de asentamiento es mucho mas largo lo cual se debe al desgaste de los motores durante las pruebas y los distintos factores de desgaste del cuadróptero.

4.4. Implementación de un sensor de temperatura en el AR Drone

Como se propuso en el capítulo 3, se implementa un sensor de temperatura embebido en el AR drone 1.0 a través del puerto serial con la ayuda de la placa de desarrollo de hardware libre Arduino. La figura 4.14 muestra el esquema para conectar el sensor.

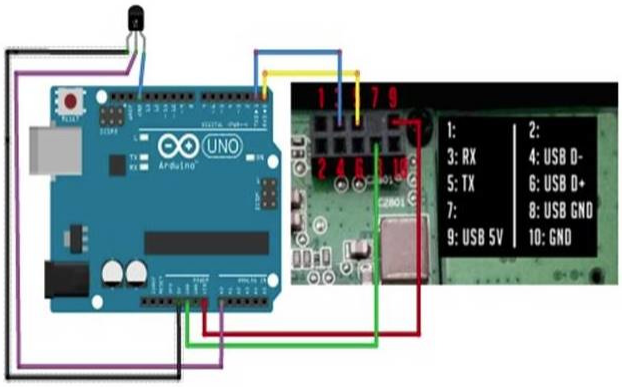


Figura 4.14: Esquema de conexión del sensor LM35

Al ejecutar el programa del puerto serial podemos observar en la pantalla la lectura de la temperatura. En el programa se utilizó un factor de corrección el cual fue obtenido en el datasheet del sensor de temperatura LM35.

```
Microsoft Windows [Versión 6.3.9600]
(c) 2013 Microsoft Corporation. Todos los derechos reservados.

C:\Users\Alejandro>pserial.js

25.8

25.8

25.8

25.6
```

Figura 4.15: Lectura de temperatura con el sensor LM35

Al implementar este sensor se pretende demostrar que se pueden agregar mas sensores al vehículo para poder mejorar su rendimiento en vuelo. Así mismo, se pueden

implementar sensores de humedad, sensores bioquimicos o mejorar los sensores con los que cuenta el propio cuadróptero.

Capítulo 5

Conclusiones Generales

5.1. Conclusiones

En base a la información obtenida, se dividen las conclusiones de acuerdo a los objetivos específicos descritos en el primer capítulo. Se analizarán las conclusiones y se establecerán las propuestas para un trabajo a futuro.

La información descrita deja claro los componentes del cuadróptero y el material del que están elaborados, esto da una idea más clara de lo que puede realizar y a lo que podría estar expuesto con respecto a las variaciones del clima.

Se presentó la metodología para el diseño e implementación de una plataforma realizada en lenguaje C para el control de los motores del Parrot AR Drone a través de los puertos GPIO del microprocesador ARM. Se produjo exitosamente el accionamiento de cada uno de los motores, de todos los motores en conjunto, así como la acelereación, desaceleración y frenado de los motores.

Los movimientos del cuadróptero y sus aplicaciones se relacionan entre sí, debido a que las tareas que realiza dependen de estos movimientos, por esta razón se buscó otorgar la mayor libertad de movimientos al cuadróptero para que pueda realizar mayor variedad de actividades.

Cabe Mencionar que el código realizado no depende del SDK (software Development

Kit) de Parrot, lo que lo hace más independiente y maniobrable. Las funciones del SDK no incluyen el control de los motores por separado, ni el control de la aceleración y desaceleración.

Uno de los problemas con los que se encontró en este proyecto fue sin duda la falta de documentción, debido a que Parrot no proporciona datos específicos de puertos de control, accionamiento de cámaras y obtención de datos por lo que muchos de los datos se obtuvieron por el metodo de prueba y error.

La implementación del control PID se realizó exitosamente, los parámetros fueron establecidos de tal forma que la nave guarda la posición ante cualquier perturbación externa.

5.2. Aportaciones

Entre las principales aportaciones de éste trabajo se pueden mencionar:

- Realización de un código independiente del SDK de la empresa Parrot para controlar los motores del cuadróptero.
- Implementación de un código en lenguaje C para aceleración y desaceleración de los motores.
- La realización de una plataforma para controlar los motores a traves de una computadora portátil.
- La implementación de un controlador PID diferente al de fábrica.
- La implementación de un sensor de temperatura embebido en el cuadróptero.

5.3. Trabajo a futuro

- Implementar un software para tratamiento de imágenes y video.

- Realizar un análisis de los datos obtenidos para verificar la confiabilidad del sistema en campo.
- Implementar mas sensores, como humedad relativa y/o bioquimicos, al cuadróptero.
- Implementar un controlador PID con mejor comportamiento al se propuso en este trabajo.

Bibliografía

[1] Paul Pounds, Robert Mahony, and Peter Corke. Modelling and Control of a Quad-Rotor Robot. *East*, 4:44, 2006.

[2] John Macdonald, Robert Leishman, Randal Beard, and Timothy McLain. Analysis of an improved IMU-based observer for multirotor helicopters. *Journal of Intelligent and Robotic Systems: Theory and Applications*, 74(3-4):1049–1061, 2014.

[3] Jmb Domingues. Quadrotor prototype. *Instituto superior tecnico, universidade tecnica de ...*, page 129, 2009.

[4] S.L. Waslander. Flight PID Controller Design for a UAV Quadrotor - Documents.

[5] J. Engel, J. Sturm, and D. Cremers. Camera-Based Navigation of a Low-Cost Quadrocopter. *Proc. of the International Conference on Intelligent Robot Systems (IROS)*, 2012.

[6] Javier Ignacio García Santos. Lineas aéreas ecuatorianas, transportación de pasajeros y carga al exterior; efectos económicos por la impuntualidad; caso lan ecuador 2006-2010. 2011.

[7] Wayne Johnson. *Helicopter theory*. Courier Corporation, 2012.

[8] L Acosta, J Toledo, GN Marichal, S Hernández, JN Rodríguez, and S Torres. Diseño e implementación de una maqueta de un helicótero de 4 rotores para un laboratorio de control. *XXV Jornadas de Automática*, 2004.

[9] Omar Luque Rodriguez. *Estudio y desarrollo de un sistema de control de un cuadricoptero*. PhD thesis, 2014.

[10] Daniel Garijo Verdejo, Jesús Ismael López Pérez, and Isaac Pérez Estrada. Control de un vehículo aéreo no tripulado. 2009.

[11] Tulo D E L Tfc, Rodrigo Alberto, Mayorga Rodr, Jaime Oscar, and Casas Piedrafita. Trabajo de final de carrera. 2009.

[12] Roger Clarke. What drones inherit from their ancestors. *Computer Law and Security Review*, 30(3):247–262, 2014.

[13] Veronica L. Foreman, Francesca M. Favaró, Joseph H. Saleh, and Christopher W. Johnson. Software in military aviation and drone mishaps: Analysis and recommendations for the investigation process. *Reliability Engineering & System Safety*, 137:101–111, 2015.

[14] Juan Manuel Fern, Ribao Tutor, and Gregorio Robles Mart. Autor: Juan Manuel Fernández Ribao Tutor: Gregorio Robles Martínez.

[15] Jose F. Almendariz. *Control del movimiento de un quadrotor mediante un sensor de profundidad kinect*. PhD thesis.

[16] Alessandra Capolupo, Stefania Pindozzi, Collins Okello, Nunzio Fiorentino, and Lorenzo Boccia. Photogrammetry for environmental monitoring: The use of drones and hydrological models for detection of soil contaminated by copper. *Science of the Total Environment*, 514:298–306, 2015.

[17] JJ Delgado Sotés. Otros sistemas operativos en red.

[18] Alejandro Alarcon Quigua. Estudio, implementación y análisis de tráfico de una red voip bajo el protocolo sip. 2013.

[19] Pablo Gonzalez Garcia et al. Diseño de una herramienta de planificación de sistemas wimax. 2006.

[20] Katsuhiko Ogata. *Ingeniería de control moderna*. Pearson Educación, 2003.

[21] RAUL RASCON CARMONA. *ANALISIS DE CONTROLADORES DE ESTRUCTURA VARIABLE PARA HELICOPTEROS DE CUATRO ROTORES CON RETROALIMENTACION DE SALIDA*. PhD thesis, 2009.

Apéndice A

Programas principales de la plataforma de control de los motores

A.1. Programa principal de control

```
#include <stdio.h>
#include <unistd.h>
#include <pthread.h>
#include <ctype.h>
#include <math.h>

#include "../util/type.h"
#include "../util/util.h"
#include "mot.h"


int main()
```

```
{
  printf("programa control de motor ... Presione q para salir\r\n");
  printf("Motor: 1,2,3,4=correr cada motor al 50%  5=correr todos los motores al 5C
  printf("Leds:  a=leds apagados  s=modo vuelo  d=modo emergencia  \r\n");

  mot_Init();

  float throttle1 = 0;
  float throttle2 = 0;
  float throttle3 = 0;
  float throttle4 = 0;
  float step=0.01;


  while(1) {

int c=tolower(util_getch());
if(c==’q’) break;
if(c==’1’) {
printf("\r Motor1 al 50%           ");
throttle1 = .50;
throttle2 = 0;
throttle3 = 0;
throttle4 = 0;
mot_Run(throttle1,throttle2,throttle3,throttle4);
}
if(c==’2’) {
printf("\r Motor2 al 50%           ");
throttle1 = 0;
```

```
throttle2 = .50;
throttle3 = 0;
throttle4 = 0;
mot_Run(throttle1,throttle2,throttle3,throttle4);
}
if(c==’3’) {
printf("\r Motor3 al 50%            ");
throttle1 = 0;
throttle2 = 0;
throttle3 = .50;
throttle4 = 0;
mot_Run(throttle1,throttle2,throttle3,throttle4);
}
if(c==’4’) {
printf("\r Motor4 al 50%            ");
throttle1 = 0;
throttle2 = 0;
throttle3 = 0;
throttle4 = .50;
mot_Run(throttle1,throttle2,throttle3,throttle4);
}
if(c==’5’) {
printf("\rTodos los motores al 50%           ");
throttle1 = .50;
throttle2 = .50;
throttle3 = .50;
throttle4 = .50;
mot_Run(throttle1,throttle2,throttle3,throttle4);
}
```

```
if(c==',') {
printf("\rDesaceleración            ");
if(throttle1>step) throttle1 -= step;
if(throttle2>step) throttle2 -= step;
if(throttle3>step) throttle3 -= step;
if(throttle4>step) throttle4 -= step;
mot_Run(throttle1,throttle2,throttle3,throttle4);
}
if(c=='.') {
printf("\rAceleración            ");
if(throttle1>0) throttle1 += step;
if(throttle2>0) throttle2 += step;
if(throttle3>0) throttle3 += step;
if(throttle4>0) throttle4 += step;
mot_Run(throttle1,throttle2,throttle3,throttle4);
}
if(c==' ') {
printf("\r Frenar           ");
mot_Stop();
}
if(c=='a') {
printf("\rLeds  apagados           ");
mot_SetLeds(MOT_LEDOFF,MOT_LEDOFF,MOT_LEDOFF,MOT_LEDOFF);
}
if(c=='s') {
printf("\r Modo vuelo            ");
mot_SetLeds(MOT_LEDGREEN,MOT_LEDGREEN,MOT_LEDGREEN,MOT_LEDGREEN);
}
```

```
if(c=='f') {
printf("\r Modo emergencia            ");
mot_SetLeds(MOT_LEDRED,MOT_LEDRED,MOT_LEDRED,MOT_LEDRED);
}

pthread_yield();
  }
  mot_Close();
  printf("\nSalir...\n");
  return 0;
}
```

A.2. Programa del controlador PID

```
#include "pid.h"

void pid_Init(pid_struct *pid, float kp, float ki, float kd, float i_max)
{
pid->kp=0.499;
pid->ki=0.0795;
pid->kd=1;
pid->beta=0.001;
pid->i=0;
pid->e_prev=0;
}

float pid_CalcD(pid_struct *pid, float error, float dt, float d)
{
pid->i += error * dt;
```

```
if(pid->i > pid->i_max) pid->i = pid->i_max;
if(pid->i < -pid->i_max) pid->i = -pid->i_max;
float out = pid->kp * error + pid->ki * pid->i + pid->kd * d;
pid->e_prev = error;
return out;
}

float pid_Calc(pid_struct *pid, float error, float dt)
{
return pid_CalcD(pid,  error, dt, (error - pid->e_prev)/dt);
}
```

A.3. PWM de los motores

```
#include <stdio.h>
#include <string.h>
#include <unistd.h>
#include <fcntl.h>
#include <errno.h>
#include <termios.h>

#include "../gpio/gpio.h"
#include "motorboard.h"

int mot_fd; /* File descriptor for the port */


int motorboard_cmd(u08 cmd, u08 *reply, int replylen) {
```

```
write(mot_fd,&cmd,1);
return read(mot_fd,reply,replylen);
}

int motorboard_Init() {
//abrir puertos
mot_fd = open("/dev/ttyPA1", O_RDWR | O_NOCTTY | O_NDELAY);
if (mot_fd == -1)
{
perror("open_port: Puerto_no_valido /dev/ttyPA1 - ");
return 201;
}
fcntl(mot_fd, F_SETFL, 0);
//Reset del flipflop IRQ
gpio_set(106,-1);
gpio_set(107,0);
gpio_set(107,1);

//Seleccionar puertos
gpio_set(68,1);
gpio_set(69,1);
gpio_set(70,1);
gpio_set(71,1);

//Configurar motores
int retval=0;
u08 reply[256];
for(int m=0;m<4;m++) {
gpio_set(68+m,-1);
```

```
motorboard_cmd(0xe0,reply,2);
if(reply[0]!=0xe0 || reply[1]!=0x00)
{
printf("motor%d pwm=0x%02x \n",m+1,(int)reply[0]);
retval=1;
}
motorboard_cmd(m+1,reply,1);
gpio_set(68+m,1);
}
void motorboard_Close()
{
close(mot_fd);
}
```

A.3.1. Configuración de las funciones de los motores

```
#include <stdio.h>
#include <string.h>
#include <unistd.h>
#include <errno.h>
#include <stdlib.h>
#include <pthread.h>

#include "mot.h"
#include "motorboard.h"

struct mot_struct
{
  float mot[4];
```

```
  u16 pwm[4];
  u08 led[4];
  u08 NeedToSendLedCmd;
};
pthread_t mot_thread;
pthread_mutex_t mot_mutex;
mot_struct mot;

void *mot_main(void* data)
{
mot.NeedToSendLedCmd=1;
while(1) {
usleep(5000);
pthread_mutex_lock(&mot_mutex);
motorboard_SetPWM(mot.pwm[0],mot.pwm[1],mot.pwm[2],mot.pwm[3]);
if(mot.NeedToSendLedCmd) {

motorboard_SetLeds(mot.led[0],mot.led[1],mot.led[2],mot.led[3]);
mot.NeedToSendLedCmd=0;
}

pthread_mutex_unlock(&mot_mutex);
}
}

void mot_SetPWM(u16 m0, u16 m1, u16 m2, u16 m3) {
  pthread_mutex_lock(&mot_mutex);
  mot.pwm[0] = m0 & 0x1ff;
  mot.pwm[1] = m1 & 0x1ff;
```

```
  mot.pwm[2] = m2 & 0x1ff;
  mot.pwm[3] = m3 & 0x1ff;
  pthread_mutex_unlock(&mot_mutex);
}
void mot_Stop()
{
  pthread_mutex_lock(&mot_mutex);
  mot.pwm[0] = 0;
  mot.pwm[1] = 0;
  mot.pwm[2] = 0;
  mot.pwm[3] = 0;
  mot.mot[0]=0;
  mot.mot[1]=0;
  mot.mot[2]=0;
  mot.mot[3]=0;
  pthread_mutex_unlock(&mot_mutex);
}

void mot_Run(float m0, float m1, float m2, float m3)
{
  mot.mot[0]=m0;
  mot.mot[1]=m1;
  mot.mot[2]=m2;
  mot.mot[3]=m3;



  float pwm[4];
  for(int i=0;i<4;i++) {
    if(mot.mot[i]<0.0) mot.mot[i]=0.0;
```

```
    if(mot.mot[i]>1.0) mot.mot[i]=1.0;
    pwm[i]=mot_pwm_min + mot.mot[i]*(mot_pwm_max-mot_pwm_min);
if(pwm[i]<mot_pwm_min) pwm[i]=mot_pwm_min;
if(pwm[i]>mot_pwm_max) pwm[i]=mot_pwm_max;
  }


  void mot_Close()
{
  motorboard_Close();
}
```