CENTRO DE INVESTIGACIÓN Y DE ESTUDIOS AVANZADOS
DEL INSTITUTO POLITÉCNICO NACIONAL

UNIDAD ZACATENCO
DEPARTAMENTO DE COMPUTACIÓN

Diseño e implementación de un sistema de control para un cuadricóptero.

Tesis que presenta

Antonio Pico Villalpando

Para Obtener el Grado de

Maestro en Ciencias

En

Computación

Director de la Tesis

Dra. Xiaoou Li

Codirector de Tesis

Dr. Rogelio Lozano

México, Distrito Federal **Diciembre, 2012**

Resumen

En décadas recientes, ha habido un gran interés en el estudio de vehículos aéreos no tripulados (en inglés: *UAVs* o *unmanned aerial vehicles*), ya que han demostrado ser una herramienta versátil y de relativo bajo costo para un gran número de aplicaciones.
Los últimos avances en la tecnología han impulsado el desarrollo y la operación de este tipo de vehículos. Los nuevos sensores, microprocesadores y sistemas de propulsión, son ahora más pequeños, más ligeros y más capaces que nunca.
En este trabajo de tesis, construimos un vehículo aéreo de ala rotativa llamado cuadricóptero o cuadrirotor. Estos vehículos tienen la ventaja de poder quedarse suspendidos en el aire y de tener alta maniobrabilidad, lo que los hace útiles en misiones de robótica.
El vehículo cuenta con una pequeña computadora a bordo con Linux embebido y Xenomai, una plataforma para dar soporte en tiempo real.
Diseñamos el control automático de postura y altura basándonos en controladores PID. Este tipo de controladores requiere de un ajuste fino de sus parámetros para su correcto funcionamiento por lo que, viendo ésto como un problema de optimización, utilizamos el algoritmo de evolución diferencial para hacer el mejor ajuste posible, obteniendo resultados positivos.
El cuadricóptero puede comunicarse con una computadora (usada para configuración y apoyo en la evolución diferencial) y una *tablet-PC* con Android, que sirve para el control en tierra. Ésto a través de una red *Wi-Fi*.
Palabras clave: vehículos aéreos no tripulados, cuadricóptero, evolución diferencial, control PID, sistemas embebidos.

Abstract

In recent decades, there has been a great interest in the study of unmanned aerial vehicles (UAV's), as they have proven to be a versatile and relative low cost tool for a large number of applications.
Recent advances in technology have boosted the development and operation of such vehicles. The new sensors, microprocessors and propulsion systems, are now smaller, lighter and more capable than ever.
In this thesis, we construct a rotary-wing aerial vehicle called quadcopter or quadrotor. These vehicles have the advantage of being able to hover and having high maneuverability, making them useful in robotic missions.
The vehicle has a small on-board computer running embedded Linux and Xenomai, a development framework for real-time support.
We design an automatic control of attitude and altitude based on PID controllers. Such controllers require the fine tuning of their parameters for proper operation, so seeing this as an optimization problem, we use the differential evolution algorithm to do the best possible fit, with positive results.
The quadcopter can communicate with a computer (used for configuration and differential evolution support) and a tablet-PC with Android, which serves as ground control. All this through a Wi-Fi network.
keywords: unmanned aerial vehicles, quadcopter, differential evolution, PID control, embedded systems.

Agradecimientos

A mis padres Martha y Antonio, por apoyarme siempre en todos mis emprendimientos.

A mis tías Lidia y Lourdes, por ofrecerme su ayuda siempre que lo he necesitado.

A mis asesores de tesis, la Dra. Xiaoou Li y el Dr. Rogelio Lozano, por permitirme trabajar con ellos y brindarme el equipo apropiado para el desarrollo del proyecto.

A mis revisores de tesis, la Dra. Sonia Mendoza y el Dr. Sergio Salazar, por tomarse el tiempo de examinar mi trabajo y por sus valiosos consejos.

A los profesores y compañeros del departamento de computación, por compartir sus conocimientos.

A sofía Reza, por brindarme su apoyo durante mi estancia en el CINVESTAV.

Al CINVESTAV, por darme la oportunidad de estudiar una maestría.

Al Consejo Nacional de Ciencia y Tecnología (CONACYT), por el apoyo económico brindado durante el posgrado.

Índice general

Índice de figuras

Índice de cuadros

Capítulo 1

Introducción

En los últimos años ha habido un gran interés en el desarrollo de vehículos aéreos no tripulados, ya que poseen características únicas como: tamaño pequeño, gran maniobrabilidad y relativo bajo precio, haciéndolos atractivos para uso tanto militar como civil en áreas como vigilancia, reconocimiento e inspección en ambientes complejos o peligrosos.

Los últimos avances en la tecnología han impulsado el desarrollo y la operación de este tipo de vehículos. Los nuevos sensores, microprocesadores y sistemas de propulsión son más pequeños, ligeros y más capaces que nunca, llevando a niveles de resistencia, eficiencia y autonomía que sobrepasan las capacidades humanas [Nonami et al., 2010].

1.1. Antecedentes y motivación

El primer vehículo aéreo no tripulado (en inglés: UAV o *unmanned aerial vehicle*) fue construido en 1916, pero debido a su inmadurez no tuvo ningún uso práctico [Barnhart et al., 2011].
El desarrollo de UAVs empezó a crecer a finales de los años 1950's, contando con investigación a gran escala hasta finales de los 1970s, siendo utilizados por primera vez en forma práctica por Estados Unidos en la guerra del golfo (1991).
Por otro lado, la Agencia estadounidense del Espacio y la Aeronáutica (en inglés: *NASA* o *National Aeronautics and Space Administration*) hacía investigaciones para usos civiles en ese mismo periodo, desarrollando vehículos de medición de variables ambientales [Nonami et al., 2010].
A partir de entonces, se ha desarrollado una gran variedad de este tipo de vehículos para usos como: fotografía y video aéreos, instrumentación en la agricultura, inspección de cables de alta tensión, monitoreo de tráfico vehicular, etcétera.
Un tipo particular de UAVs, el cuadricóptero, ha sido objeto de numerosas investigaciones en los ultimos años, debido a que tienen la ventaja de poder quedarse suspendidos en el aire y ser altamente maniobrables, lo cual habilita el despegue y

aterrizaje vertical. Estas capacidades los hacen especialmente útiles en misiones de robótica, lo que llevó a su utilización en este trabajo de tesis.

Si bien, el desarrollo de vehículos aéreos no tripulados ha tenido un gran avance en Estados Unidos, Japón y Europa [Nonami et al., 2010]; en países como México aún se encuentra en su etapa inicial. Hay pocos centros de investigación dedicados a este tema y aún no se cuenta con ninguna implementación comercial.
Esta situación me ha llevado a realizar este proyecto, además de mi interés personal en la aplicación de tecnología para la agricultura en países en vías de desarrollo, que es a donde va enfocado el trabajo a mediano y largo plazo.

1.2. Planteamiento del problema

Un cuadricóptero es un helicóptero con cuatro rotores, donde cada uno está colocado en la extremidad de una cruz. El control del cuadricóptero se hace modificando las velocidades relativas de los rotores, ya sea para cambiar su altitud, orientación, avance, retroceso, etcétera.
Si un operador humano simplemente manda señales de velocidad a cada motor, el control del vehículo se hace casi imposible, primero porque es un sistema dinámico que requiere ajustes en el orden de los milisegundos y segundo porque el número de variables a controlar sobrepasa las capacidades de una persona normal (6 grados de libertad). Debido a ésto, se deben incluir etapas de control automático que aligeren la carga del operador.
El diseño de una buena interfaz de usuario también es importante. Una interfaz sencilla e intuitiva puede acelerar enormemente el proceso de aprendizaje del manejo de estos vehículos, facilitando su uso en lugares donde no haya disponible personal especializado.
Otro aspecto a tomar en cuenta es el diseño del *hardware*. Un diseño pensado en extender las funcionalidades del vehículo, puede ser muy útil para investigaciones futuras.

1.3. Objetivos

General

Diseñar e implementar un sistema de control para un cuadricóptero, que sea fácil de manejar por usuarios inexpertos y que sea lo suficientemente escalable para utilizarlo como base en futuras investigaciones.

Particulares

- Construir un cuadricóptero.
- Implementar el control automático de la postura y altura del vehículo.
- Crear una interfaz de control de usuario intuitiva, aprovechando las características que ofrecen las computadoras en tableta.
- Diseñar un sistema básico, pero pensado para ser extendido con características como: despegue y aterrizaje autónomo, planeación de trayectorias, visión, etcétera.

1.4. Trabajos relacionados

Se han implementado varios tipos de cuadricópteros con base en diferentes modelos de su dinámica y controles, que van desde PID [Li and Li, 2011] hasta lógica difusa [Santos et al., 2010] y redes neuronales [Bouhali and Boudjedir, 2011].
En [Ulbrich et al., 2011] desarrollaron un sistema de control modular para facilitar la investigación en diferentes campos como los sistemas de tiempo real, sistemas distribuidos y la robótica. Además de buscar modularidad en *hardware*, se buscó modularidad en *software* basándose en un microcontrolador lo suficientemente poderoso para atender varias tareas a la vez. Se han hecho pruebas con sistemas de seguimiento de trayectorias.
En [Hafner et al., 2010] implementaron un vehículo para el desarrollo de estrategias de navegación bio-inspiradas. El sistema utiliza un *Gumstix* para su control. Un aspecto interesante de este trabajo es que utiliza sensores bio-inspirados, como una cámara construida con un lente especial para darle visión omnidireccional y una brújula basada en el patrón de polarización de la luz solar.
Una plataforma para planeamiento de trayectorias es implementada en [Bai et al., 2011]. Este sistema, además de contar con una unidad de masa inercial y *GPS*, está equipada con seis sensores de proximidad ultrasónicos, con el fin de sensar obstáculos y detectar la mejor ruta a tomar. Un sistema *bluetooth* integrado sirve para comunicar varios vehículos entre sí en ambientes colaborativos. Como unidades de procesamiento, utiliza dos microcontroladores *ARM*, uno para el control de bajo nivel (postura) y el otro para ejecutar los algoritmos de planeamiento de trayectorias y comunicación.
Ya ha habido implementaciones de vehículos aéreos no tripulados en la agricultura. En [Berni et al., 2009] desarrollaron un sistema de instrumentación multiespectral, montado en un helicóptero, con el fin de incrementar la resolución de las mediciones en cultivos con respecto a los datos obtenidos por satélites. Este sistema es capaz de seguir trayectorias de manera autónoma y mandar los datos tomados a una estación en tierra, basándose en un sistema de posicionamiento global.
Otro helicóptero diseñado para instrumentación multiespectral se implementa en

[Xiang and Tian, 2011]. Éste cuenta con tres canales de comunicación: un radiocontrol para el uso manual del vehículo, un sistema *WiFi* para el envío de datos en tiempo real y un transmisor de video, para tener una perspectiva del vehículo desde tierra.

1.5. Organización del documento

La tesis se organiza en 6 capítulos con la siguiente estructura:

- Capítulo 2: define qué son los vehículos aéreos no tripulados, presentando su historia, características y clasificación. Además, se explica el principio de funcionamiento del cuadricóptero.
- Capítulo 3: da una introducción a los sistemas de control, enfocándose en el controlador PID. También describe el diseño del sistema de control del cuadricóptero, además del método de ajuste de parámetros PID con evolución diferencial.
- Capítulo 4: hace una descripción de los componentes de *hardware* que conforman el sistema de control diseñado, así como los pasos seguidos para la construcción del cuadricóptero.
- Capítulo 5: da una breve explicación de las plataformas de desarrollo utilizadas para la implementación del *software* del sistema, para despúes hacer una descripción detallada de cada una de las aplicaciones creadas.
- Capítulo 6: presenta las conclusiones que se obtuvieron en el desarrollo de la tesis, así como el trabajo a futuro para alcanzar la meta final de este desarrollo.

Capítulo 2

Cuadricóptero

En los últimos años ha habido un gran interés en el desarrollo de vehículos aéreos no tripulados, ya que poseen características únicas como: tamaño pequeño, gran maniobrabilidad y relativo bajo precio, haciéndolos atractivos para uso tanto militar como civil en áreas como vigilancia, reconocimiento e inspección en ambientes complejos o peligrosos.

Los últimos avances en la tecnología han impulsado el desarrollo y operación de este tipo de vehículos. Los nuevos sensores, microprocesadores y sistemas de propulsión son más pequeños, ligeros y más capaces que nunca, llevando a niveles de resistencia, eficiencia y autonomía que sobrepasan las capacidades humanas [Nonami et al., 2010]. En este capítulo, se definirá lo que es un vehículo aéreo no tripulado, se hará una breve reseña histórica y se describirán sus características. Se pondrá especial énfasis en un tipo particular de estos vehículos, el cuadricóptero, cuya historia y principio de funcionamiento serán explicados.

2.1. Vehículos aéreos no tripulados

Un avión no tripulado es definido con los términos siguientes: vehículo aéreo no tripulado (en inglés: *UAVs* o *Unmanned Aerial Vehicles*), avión operado de forma remota (en inglés: *ROA* o *Remotely Operated Aircraft*), y vehículo piloteado de forma remota (en inglés: *RPV* o *Remotely Piloted Vehicle*) [DoD, 2002]. El Instituto Americano de la Aeronáutica y de la Astronáutica (en inglés: *AIAA* o *The American Institute of Aeronautics and Astronautics*) define un vehículo aéreo no tripulado como: "Un avión que es diseñado o modificado para no cargar a un piloto humano y es operado por un controlador de vuelo o por un sistema de control de vuelo autónomo a bordo, que no requiere la intervención del controlador de vuelo."

Aunque no hay una definición estricta de la diferencia entre UAV y MAV (del inglés: *Micro Aerial Vehicles*), de acuerdo con la Agencia de Proyectos de Investigación

Avanzados de Defensa de los Estados Unidos (en inglés: *DARPA* o *Defense Advanced Research Projects Agency*), Los MAV tienen dimensiones de quince centímetros o menos.

2.1.1. Historia

El primer vehículo aéreo no tripulado fue construido por Lawrence y Elmer Sperry en 1916 (figura 2.1). Ellos desarrollaron un giroscópio para estabilizarlo. Pudieron volarlo a una distancia de casi 100 metros. Sin embargo, debido a su inmadurez para usos prácticos, este vehículo no fue utilizado en la primera guerra mundial [Barnhart et al., 2011].

Figura 2.1: Primer vehículo aéreo no tripulado [Barnhart et al., 2011].

El desarrollo de UAVs empezó a crecer a finales de los 1950s aprovechando la guerra de Vietnam y la guerra fría, contando con investigación a gran escala hasta finales de los 1970s. Después de la guerra de Vietnam, Estados Unidos e Israel empezaron a construir vehículos más pequeños y ligeros, los cuales utilizaban pequeños motores como los utilizados en motocicletas, cargaban cámaras de video y transmitían imágenes al operador. Parece que el prototipo de los UAVs actuales puede ser encontrado en este periodo [Nonami et al., 2010] .

Estados Unidos los utilizó en forma práctica en la guerra del Golfo en 1991. A partir de entonces, los vehículos aéreos para aplicaciones militares se han desarrollado rápidamente. Por otro lado, la Agencia estadounidense del Espacio y la Aeronáutica (en inglés: *NASA* o *National Aeronautics and Space Administration*) hacía investigaciones para usos civiles en ese mismo periodo, desarrollando vehículos de medición de variables ambientales (figura 2.2).

Figura 2.2: Vehículo aéreo no tripulado de uso civil [Nonami et al., 2010].

2.1.2. Características

Algunas características de los vehículos aéreos no tripulados son equivalentes a las de los tripulados (materiales, sistemas de propulsión, aerodinámica, etc.), pero algunas son específicas a éstos. Las más importantes son [Nonami et al., 2010]:

- Sensores de navegación y microprocesadores: los sensores representan uno de los elementos más caros de este tipo de sistemas y son necesarios para la navegación y la ejecución de tareas. Los procesadores permiten a los UAVs volar misiones enteras de manera autónoma con poca o nula intervención humana.
- Sistemas de comunicación: los problemas principales de las tecnologías de comunicación son flexibilidad, adaptabilidad, seguridad y controlabilidad de flujo de datos.
- Estación de control en tierra (ECT): es un centro de control en tierra o mar que provee las facilidades para el control humano de los vehículos no tripulados en el aire o el espacio. Las ECT varían en tamaño, pueden ser tan pequeñas como un dispositivo de mano o tan grandes como estaciones de trabajo con múltiples trabajadores en ellas. Una de las metas principales para futuras operaciones será la capacidad de controlar múltiples aeronaves desde una ECT [Barnhart et al., 2011] .
- Inteligencia a bordo: la inteligencia que puede ser "empacada" dentro de un UAV está directamente relacionada a la complejidad de las tareas que puede manejar e inversamente relacionada con la cantidad de supervisión requerida por operadores humanos.

2.1.3. Clasificación

Según Nonami et al, un vehículo aéreo no tripulado generalmente entra en una de las siguientes cuatro categorías [Nonami et al., 2010]:

- Ala fija: se refiere a aeroplanos que requieren un recorrido horizontal para despegar o aterrizar, o un lanzamiento en catapulta. Generalmente tienen largos tiempos de vuelo y pueden volar a altas velocidades.
- Ala rotativa: tienen la ventaja de poder quedarse suspendidos en el aire y de tener alta maniobrabilidad. Estas capacidades los hacen útiles en misiones de robótica, especialmente en aplicaciones civiles. Un UAV de ala rotativa puede tener varias configuraciones, con rotores principal y de cola (helicóptero convencional), rotores coaxiales, multirotores, etc.
- Dirigibles: como globos o zepelines, que son más ligeros que el aire. Tienen largos tiempos de vuelo, vuelan a bajas velocidades y generalmente son de tamaño grande.
- UAVs de aleteo (en inglés: *flapping-wing*): poseen alas flexibles inspiradas en pájaros y pequeños insectos.

2.1.4. Aplicaciones

A continuación se enlistan algunos usos en los que se aplican o se pueden aplicar los UAVs [Austin, 2010].
Usos civiles:

- Fotografía aérea
- Agricultura
- Guardia costera
- Conservación
- Compañias de electricidad
- Detección de incendios
- Forestal
- Monitoreo de ganado
- Monitoreo de contaminación atmosférica
- Servicios meteorológicos

- Compañias petroleras
- Monitoreo de tráfico vehicular
- Búsqueda de personas perdidas
- Monitoreo de tuberías

Usos militares:

- Protección de puertos
- Desvío de misiles
- Reconocimiento
- Vigilancia de actividad enemiga
- Localización y destrucción de minas terrestres
- Monitoreo de contaminación química, biológica o nuclear
- Eliminación de bombas sin explotar

2.2. Cuadricóptero

El cuadricóptero o cuadrirotor (figura 2.3) es el vehículo aéreo multirotor más popular. Éste intenta alcanzar una suspensión en el aire estable y vuelos precisos, balanceando las fuerzas producidas por sus cuatro rotores.

Figura 2.3: Cuadricóptero [Ulbrich et al., 2011].

Una de las ventajas de utilizar un helicóptero multirotor es su capacidad de carga, como tiene más fuerza de empuje, puede soportar cargas más pesadas. Los cuadricópteros son altamente maniobrables, lo cual habilita el despegue y el aterrizaje vertical, así como también el acceso a áreas difíciles de alcanzar. Tienen como desventaja el incremento de peso del helicóptero y el incremento en consumo de energía debido a los motores extra. Como éstos son controlados con cambios de velocidad en los rotores, es más adecuado el uso de motores eléctricos [Castillo et al., 2005].

2.2.1. Historia

Louis y Jacques Bréguet, dos hermanos que trabajaban para el profesor Charles Richet, fueron los primeros en construir un cuadricóptero, al cual nombraron "Bréguet Richet Gyroplane No. 1" (figura 2.4). La primera demostración de vuelo fue conseguida el 29 de septiembre de 1907.

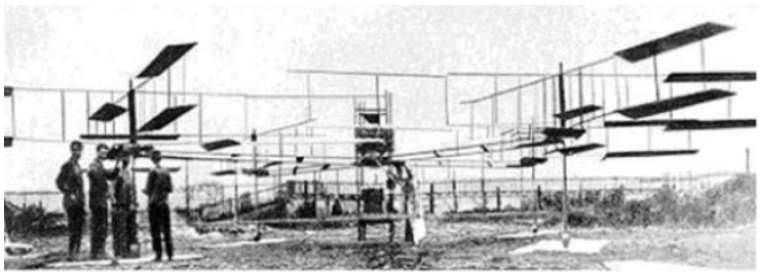


Figura 2.4: Primer cuadricóptero [Raza and Gueaieb, 2010].

Después, dos diseños adicionales fueron desarrollados y se realizaron vuelos experimentales. El primero, construido por Georges de Bothezat e Ivan Jerome en 1922, tenía los rotores colocados en cada extremo de una estructura en forma de X, como se muestra en la figura 2.5.

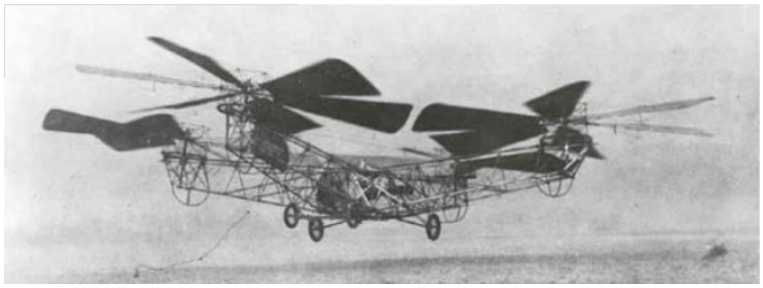


Figura 2.5: Cuadricóptero diseñado por George De Bothezat [Raza and Gueaieb, 2010].

El segundo, mostrado en la figura 2.6, fue construido por Étienne Œ hmichen en 1924, y rompió records de distancia, incluido el de realizar el primer vuelo de un kilómetro por un helicóptero.

Actualmente, además de ser utilizados en aplicaciones militares, los cuadricópteros son utilizados para tareas de vigilancia en áreas contaminadas o inaccesibles [Puls et al., 2009], para actividades de búsqueda y rescate [Naidoo et al., 2011], investigación académica, etc.

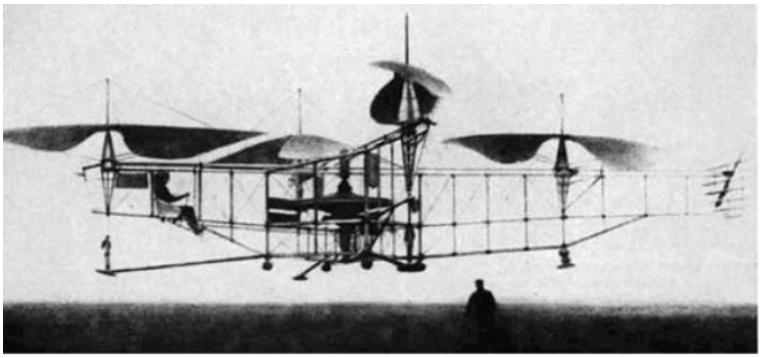


Figura 2.6: Cuadricóptero diseñado por Œ hmichen en 1924 [Raza and Gueaieb, 2010].

2.2.2. Principio de operación

Similar a un helicóptero tradicional, un cuadricóptero es un sistema de seis grados de libertad (x, y, z, *pitch*, *roll* y *yaw*), multivariable, fuertemente acoplado y subactuado. Las fuerzas principales y momentos que actúan en un cuadricóptero son producidas por sus rotores. Dos pares de motores rotan en direcciones opuestas para balancear el torque total del sistema. La figura 2.7 muestra un diagrama de un cuadrirotor típico.

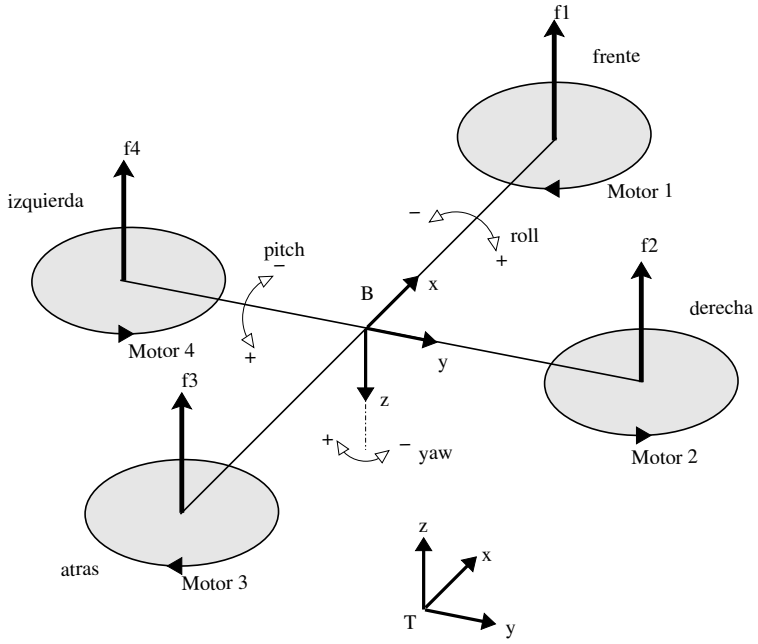
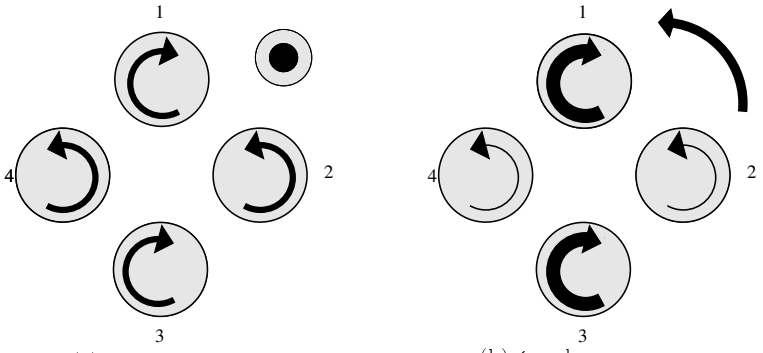


Figura 2.7: Diagrama de un cuadrirotor.

Como se puede observar, se utilizan dos sistemas de referencia: el sistema T fijo a la Tierra y el sistema B fijo al vehículo.
Un cuadrirotor es controlado, manipulando las fuerzas de empuje de los rotores, así co-

mo balanceando el torque entre ellos. Para suspenderlo en el aire, todos los rotores aplican una fuerza constante como se muestra en la figura 2.8(a). Para controlar el movimiento vertical, la velocidad de los motores se incrementa o decrementa simultáneamente. De este modo se obtiene una mayor o menor fuerza de empuje, pero sin afectar el balance.



(a) sustentación (b) ángulo *yaw*

(c) ángulo *pitch* (c) ángulo *roll*

Figura 2.8: Dinámica de un cuadrirotor, (a) sustentación y altura; (b) diferencia en torque para manipular *yaw* (ψ); (c) diferencia en torque para manipular *pitch* (θ); (d) diferencia en torque para manipular *roll* (ϕ).

Para el control de postura, el ángulo *yaw* (ψ) puede ser controlado, manipulando el balance del torque (aumentando la velocidad en un par de rotores con el mismo sentido de giro y disminuyéndola en igual magnitud en el otro par). La fuerza total de empuje permanece balanceada por lo que la altitud se conserva, como se puede ver en la figura 2.8(b).
De una forma similar, el ángulo *pitch* (θ) y el ángulo *roll* (ϕ) pueden ser manipulados, aplicando empuje diferencial en rotores opuestos, como se ilustra en las figuras 2.8(c) y (d) respectivamente [Naidoo et al., 2011].
Variando el ángulo *pitch*, se obtiene un avance o retroceso del vehículo, con respecto al eje x del sistema de referencia T, mientras que variando el ángulo *roll* se obtiene un movimiento lateral (eje y) sobre el mismo sistema T.

Capítulo 3

Control del cuadricóptero

Como se vio en el capítulo anterior, el control de un cuadricóptero se hace cambiando las velocidades de sus motores, ya sea para modificar su altitud, orientación, avance, retroceso, etcétera. Si un operador humano simplemente manda señales de velocidad a cada motor, el control del vehículo se hace casi imposible, primero porque es un sistema dinámico que requiere ajustes en el orden de los milisegundos y segundo porque el número de variables a controlar sobrepasa las capacidades de una persona normal (6 grados de libertad).
Debido a ésto, se deben incluir etapas de control automático que aligeren la carga del operador humano.
En este capítulo, después de una breve introducción a los sistemas de control, se describirá el diseño del controlador del cuadricóptero desarrollado.

3.1. Introducción a los sistemas de control

Un sistema de control está definido como un conjunto de componentes que pueden regular su propia conducta o la de otro sistema, con el fin de lograr un funcionamiento predeterminado. Los componentes básicos de un sistema de control se pueden describir mediante:

1. Objetivos de control (entradas o salidas actuantes u).
2. Componentes del sistema de control.
3. Resultados o salidas (variables controladas).

La relación básica entre estos tres componentes se muestra en la figura 3.1. En general, el objetivo de un sistema de control es controlar las salidas en alguna forma prescrita, mediante las entradas, a través de los elementos del sistema de control [Kuo, 2003].

Según su comportamiento, los sistemas de control pueden clasificarse en sistemas en lazo abierto o sistemas en lazo cerrado.

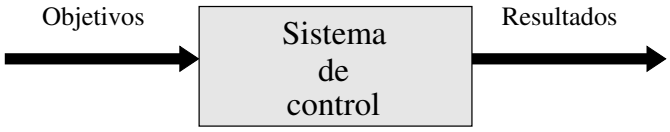


Figura 3.1: Componentes de un sistema de control.

3.1.1. Sistemas de control en lazo abierto

Los sistemas en que la salida no afecta la acción de control se denominan sistemas de control en lazo abierto.
En cualquier sistema en lazo abierto, la salida no se compara con la entrada de referencia. Por tanto, a cada entrada de referencia le corresponde una condición operativa fija; como resultado, la precisión del sistema depende de la calibración. Ante la presencia de perturbaciones, un sistema de control en lazo abierto no realiza la tarea deseada [Ogata, 1997].
Los elementos de un sistema de control en lazo abierto se pueden dividir en dos partes: el controlador y el proceso controlado (figura 3.2). Una señal de entrada o comando r se aplica al controlador, cuya salida actúa como señal actuante u; la señal actuante controla el proceso controlado, de tal forma que la variable controlada y se desempeñe de acuerdo con estándares preestablecidos.

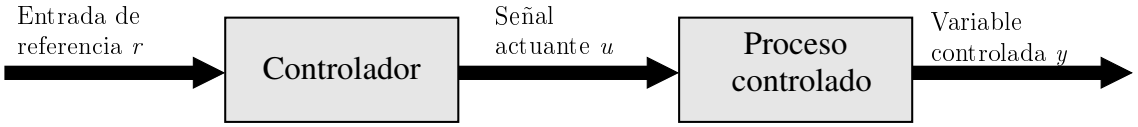


Figura 3.2: Elementos de un sistema de control en lazo abierto.

Debido a la simplicidad y economía de los sistemas de control en lazo abierto, se les encuentra en muchas aplicaciones no críticas [Kuo, 2003].

3.1.2. Sistemas de control en lazo cerrado

Lo que hace falta en el sistema de control en lazo abierto para que sea más exacto y más adaptable es una conexión o retroalimentación desde la salida hacia la entrada del sistema.
Para obtener un control más exacto, la señal controlada y debe ser retroalimentada y comparada con la señal de referencia, y se debe enviar una señal actuante proporcional a la diferencia de la entrada y la salida, a través del sistema para corregir el error. Un sistema con una o más trayectorias de retroalimentación se denomina sistema en lazo cerrado.
Este tipo de control es esencial para mantener la variable controlada estable, a pesar de perturbaciones y variaciones en la dinámica del sistema [Kuo, 2003].
La figura 3.3 muestra un diagrama a bloques de un sistema de control en lazo cerrado

típico, donde P es el proceso controlado, C es el controlador, r es la señal de referencia, y es la variable controlada, $e = r - y$ es el error, u es la señal actuante (variable de control), d es una señal de perturbación y n es una señal de ruido en la medición [Visioli, 2006].

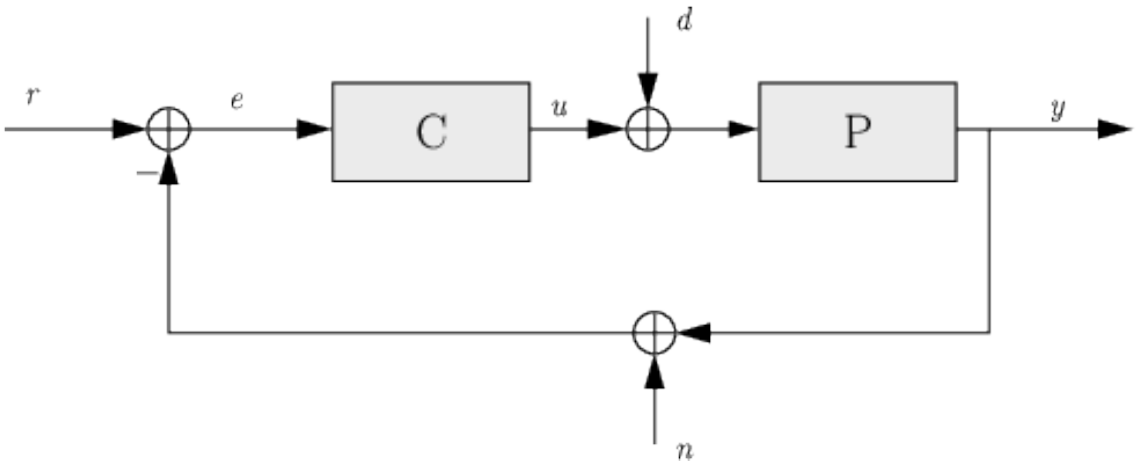


Figura 3.3: Elementos de un sistema de control en lazo cerrado.

3.1.3. Controlador PID

Un controlador PID es un controlador de tres términos que tiene una larga historia en el campo de control automático. El nemónico PID se refiere a las primeras letras de los nombres de sus términos individuales, éstos son P por la parte proporcional, I por la parte integral y D por la parte derivativa.
Los controladores PID probablemente sean los más usados en la industria, ésto es debido a su relativa simplicidad y su desempeño satisfactorio en un amplio rango de procesos. Éstos han venido evolucionando conforme al progreso de la tecnología y en la actualidad es implementado muy frecuentemente en forma digital.
El éxito de los controladores PID también es debido al hecho de que usualmente representan el componente fundamental de sistemas de control más sofisticados, que pueden ser implementados cuando una ley de control básica no es suficiente para obtener el desempeño requerido.
La ley de control PID consiste en aplicar correctamente la suma de tres tipos de acciones de control: una acción proporcional, una acción integral y una acción derivativa (figura 3.4), que serán explicadas a continuación.

Acción proporcional

La acción de control proporcional es proporcional al error de control actual de acuerdo con la expresión:

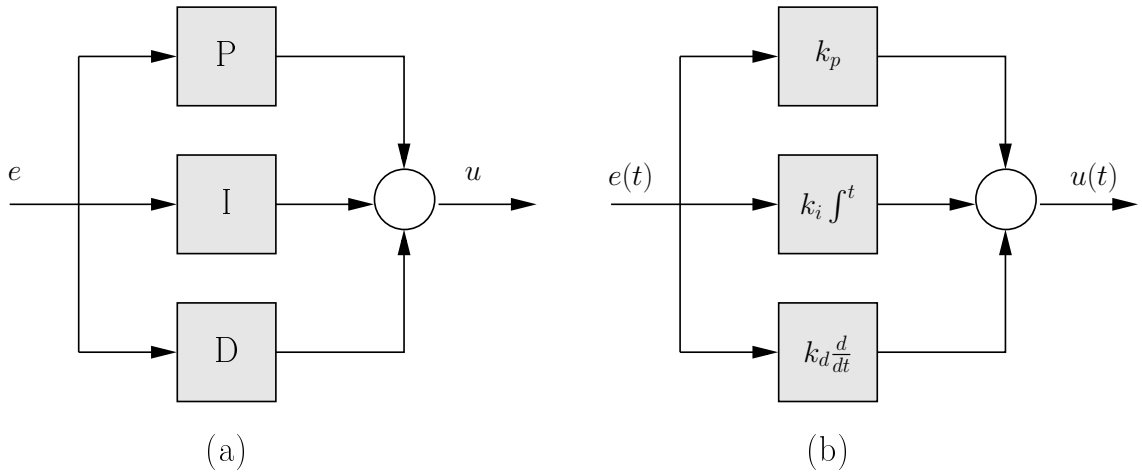


Figura 3.4: Elementos de un sistema de control PID: (a) forma simbólica y (b) forma en el dominio del tiempo

$$u(t) = k_p e(t) = k_p(r(t) - y(t)) \tag{3.1}$$

Donde k_p es la ganancia proporcional.
El significado de la acción proporcional es bastante sencillo, ya que implementa la típica operación de incrementar la variable de control, cuando el error de control es grande (con el signo apropiado).
La principal desventaja de usar un controlador proporcional puro es que éste produce un error en estado estable.

Acción integral

La acción de control integral es usada cuando se requiere que el controlador corrija el error en estado estable. Éste se denota con la expresión:

$$u(t) = k_i \int_0^t e(\tau) d\tau \tag{3.2}$$

Donde k_i es la ganancia integral.
La acción integral actúa sobre los valores pasados del error de control. Debe señalarse que cuando una acción integral está presente puede ocurrir un fenómeno llamado *windup*.Este fenómeno sucede cuando hay un cambio grande en el valor de referencia y la parte integral acumula un error más grande que el valor máximo de la variable de control, saturando la salida.
Este problema puede ser resuelto mediante las técnicas siguientes:

- Inicializar el controlador integral a un valor deseado.

- Incrementar el valor de referencia en forma gradual.
- Limitar el tiempo en el cual el error integral es calculado.
- Evitar que el término integral se acumule estableciendo límites predeterminados.

Acción derivativa

Mientras la acción proporcional está basada en el valor actual del error de control y la acción integral está basada los valores pasados del error de control, la acción derivativa se basa en la predicción de los valores futuros del error de control. La ley derivativa de control puede expresarse como:

$$u(t) = k_d \frac{de(t)}{dt} \tag{3.3}$$

Donde k_d es la ganancia derivativa.
La acción derivativa tiene un gran potencial de mejorar el desempeño del controlador, ya que ésta puede anticiparse a una tendencia incorrecta del error de control y actuar contra ello. Sin embargo tiene algunos problemas críticos, como la sensibilidad al ruido, que la hace no ser adoptada en muchos casos prácticos [Visioli, 2006] [Johnson et al., 2005]

La suma de estas tres acciones de control nos da la siguiente fórmula básica del controlador PID:

$$u_c(t) = k_P e(t) + k_I \int^t e(\tau)\,\mathrm{d}\tau + k_D \frac{de}{dt}. \tag{3.4}$$

3.1.4. Controlador PID digitalizado

Si se utiliza una implementación digital del controlador, las fórmulas mostradas anteriormente tienen que ser discretizadas.
Considerando la fórmula del controlador PID ideal:

$$u_c(t) = k_P e(t) + k_I \int^t e(\tau)\,\mathrm{d}\tau + k_D \frac{de}{dt}. \tag{3.5}$$

La parte integral puede ser aproximada utilizando diferencias finitas como:

$$\int^t e(\tau)\,\mathrm{d}\tau = \sum_{i=1}^{k} e(t_i)\Delta t, \tag{3.6}$$

donde $e(t_i)$ es el error en tiempo continuo en el i-ésimo instante de muestreo.

Aplicando también diferencias finitas al término derivativo, éste resulta en:

$$\frac{de(t_k)}{dt} = \frac{e(t_k) - e(t_{k-1})}{\Delta t}. \tag{3.7}$$

Por consiguiente, la ley de control discretizada es:

$$u(t_k) = K_P e(t_k) + k_I \sum_{i=1}^{k} e(t_i)\Delta t + K_D \frac{e(t_k) - e(t_{k-1})}{\Delta t}. \tag{3.8}$$

3.2. Control del cuadricóptero

El movimiento de un cuadricóptero tiene seis grados de libertad, definidos por el vector:

$$q = (x, y, z, \theta, \phi, \psi) \in \mathbb{R}^6 \tag{3.9}$$

donde $p = (x, y, z) \in \mathbb{R}^3$ denota la posición del centro de masa del vehículo con respecto a un eje de referencia en tierra y $\alpha = (\theta, \phi, \psi) \in \mathbb{R}^3$ son los ángulos de Euler *pitch*, *roll* y *yaw* que representan la postura.
La figura 3.5 muestra un diagrama a grandes rasgos del sistema de control en lazo cerrado para el cuadricóptero.

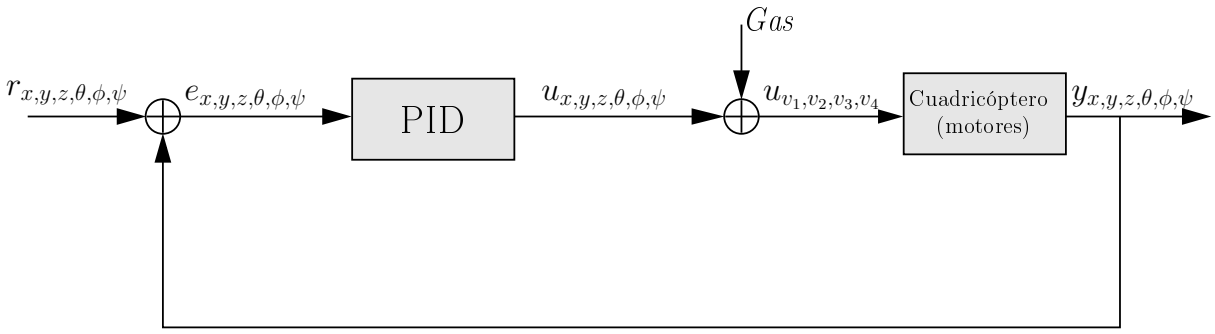


Figura 3.5: Diagrama de control del cuadricóptero.

La señal de referencia $r_{x,y,z,\theta,\phi,\psi}$ indica la postura y posición deseada del vehículo, mientras que $y_{x,y,z,\theta,\phi,\psi}$ es la variable manipulada, que es la posición y postura reales del cuadricóptero, obtenidas por medio de sensores. La señal de error $e_{x,y,z,\theta,\phi,\psi}$ es la diferencia entre la posición y postura reales y deseadas $e = r - y$. El proceso controlador es un control (controles) PID cuya variable de control $u_{x,y,z,\theta,\phi,\psi}$ es sumada a la variable *Gas* , obteniendo $u_{v1,v2,v3,v4}$ que son las señales actuantes aplicadas a cada motor del cuadricóptero.
La variable *Gas*, es un valor de referencia que indica la velocidad base utilizada por los motores. Ésta se aplica con la misma magnitud en cada motor y, a partir de ese valor, las señales actuantes de los controladores PID son sumadas (o restadas).

En este trabajo implementamos el control automático de la postura (θ, ϕ, ψ) y altura (z), mientras el control de las variables x y y se hace de forma manual.

3.2.1. Control de postura

El control automático de postura se subdivide en tres controladores PID: uno para el *pitch*, uno para el *roll* y otro para el *yaw*. La figura 3.6 muestra el diagrama de control automático de postura, donde se toman como referencia r los ángulos (θ, ϕ, ψ) deseados, que por medio de la retroalimentación y se obtienen los errores e_θ, e_ϕ, e_ψ. Cada señal de error es tomada como entrada de cada controlador PID del ángulo correspondiente, obteniéndose así tres variables de control, que combinadas resultan en la señal actuante de postura $u_{\theta,\phi,\psi}$, la cual se suma a la variable *Gas* y a la señal actuante de posición para después ser aplicada a los motores.

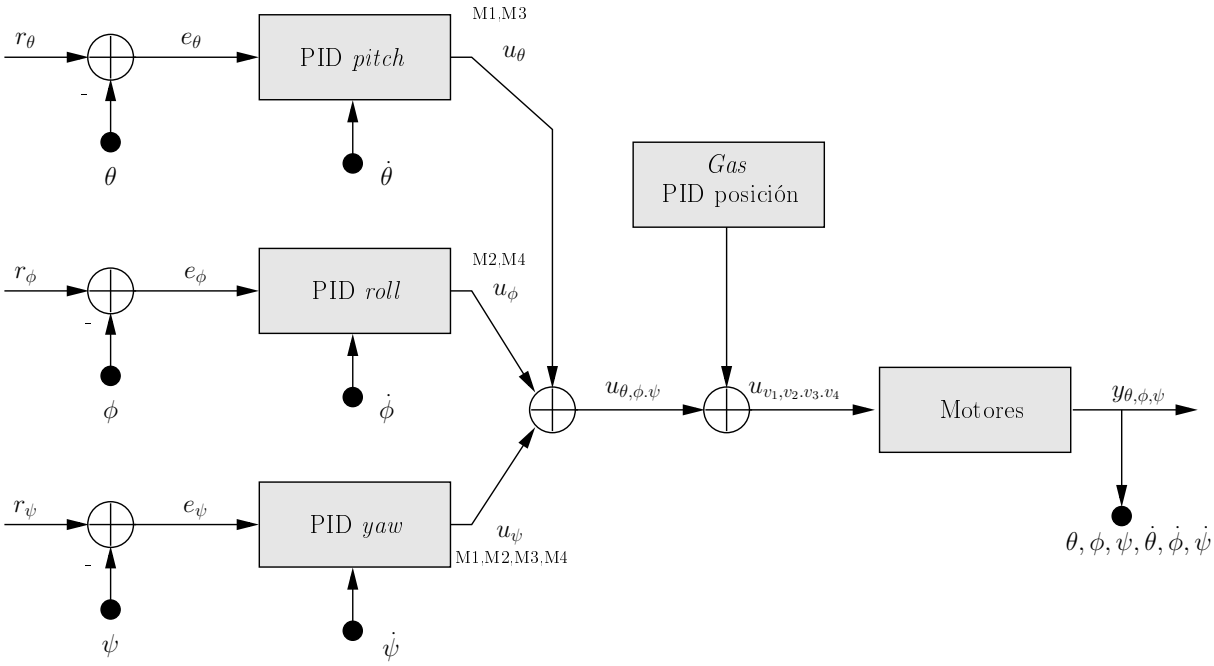


Figura 3.6: Diagrama del control de postura del cuadricóptero.

Control de *pitch* (θ)

Como puede verse en la figura 3.7, el ángulo θ representa los grados de rotación en el eje y del sistema de coordenadas del cuadricóptero. Al subir el frente del vehículo, se obtienen ángulos positivos y al bajarlo ángulos negativos. Con ésto se puede observar que el control automático de θ sólo actúa sobre los motores 1 y 3.
Para elevar el frente, se aumenta la velocidad en el motor 1 y se disminuye en la misma magnitud en el motor 3, haciendo lo contrario para bajarlo.

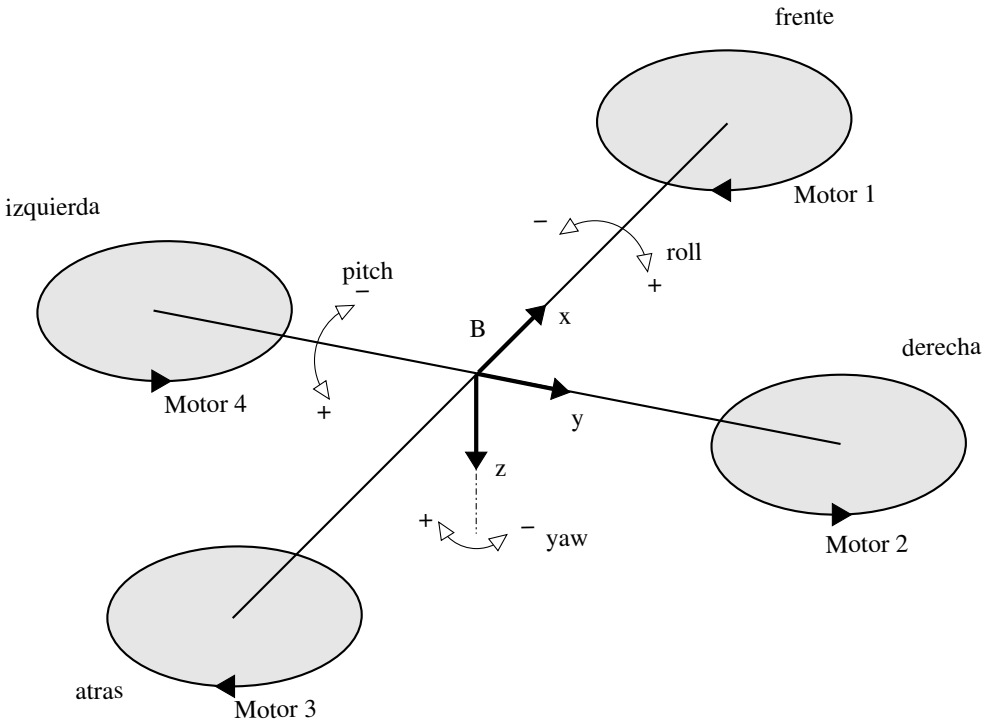


Figura 3.7: Ángulos θ, ϕ, y ψ determinan la postura del cuadricóptero.

La figura 3.8 muestra el sistema de control PID para el ángulo θ. Para éste y los demás controladores de postura se utilizó una versión modificada del PID básico. En ésta, el control proporcional e integral utilizan la señal de error e como entrada, pero la parte derivativa utiliza la derivada de la señal de retroalimentación y.

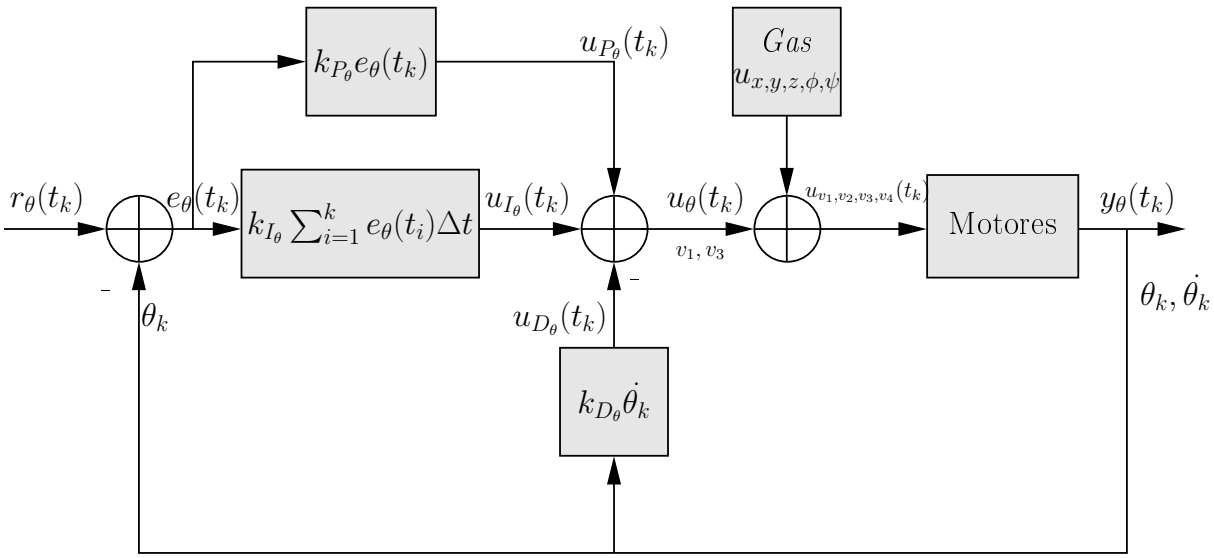


Figura 3.8: Diagrama del controlador PID de *pitch*.

Este cambio se hace para evitar el fenómeno denominado "patada derivativa" , el cual sucede en ciertos sistemas al cambiar repentinamente la entrada de referencia r, lo que provoca un pico en la señal de error, el cual se transmite a la salida de la señal de control derivativo, provocando cambios bruscos en la acción de control.
Al utilizar como entrada la derivada de la señal de retroalimentación en el término derivativo, los cambios repentinos en la señal de referencia no afectan el desempeño del control derivativo, obteniendo un sistema más estable.
Una ventaja más de esta configuración es que la derivada de la señal de salida (variable manipulada, posición angular) es la velocidad angular, que puede ser obtenida directamente de los sensores del vehículo evitando la necesidad de hacer aproximaciones basadas en la posición.
Continuando con el diagrama, vemos que la entrada de referencia r_θ corresponde los grados de inclinación deseados y la señal $e_\theta(t_k) = r_\theta(t_k) - y_\theta(t_k)$ es el error en el instante de tiempo t_k, que es utilizado por los términos proporcional e integral.
La variable de control u_{P_θ} obtenida del término proporcional queda de la siguiente manera:

$$u_{P_\theta}(t_k) = K_{P_\theta} \cdot e_\theta(t_k), \tag{3.10}$$

donde K_{P_θ} es la constante proporcional en θ y $e_\theta(t_k)$ es el error en el instante de tiempo t_k

La variable de control u_{I_θ} aportada por el término integral se obtiene con la fórmula:

$$u_{I_\theta}(t_k) = k_{I_\theta} \sum_{i=1}^{k} e_\theta(t_i)\Delta t, \tag{3.11}$$

donde k_{I_θ} es la constante integral en θ, $e(t_i)$ es el error en tiempo continuo en el i-ésimo instante de muestreo y Δt es el intervalo de muestreo.

La acción de control u_{D_θ} del término derivativo es la siguiente:

$$u_{D_\theta}(t_k) = K_{D_\theta} \cdot \dot{\theta}(t_k), \tag{3.12}$$

donde K_{D_θ} es la constante derivativa en θ y $\theta(t_k)$ es la velocidad angular en el eje y en el instante de tiempo t_k.

Por lo que la señal actuante en θ es:

$$u_\theta(t_k) = u_{P_\theta}(t_k) + u_{I_\theta}(t_k) - u_{D_\theta}(t_k) = K_{P_\theta} \cdot e_\theta(t_k) + k_{I_\theta} \sum_{i=1}^{k} e_\theta(t_i)\Delta t - K_{D_\theta} \cdot \dot{\theta}(t_k). \tag{3.13}$$

En conclusión, el aporte de la variable de control $u_\theta(t_k)$ sobre las velocidades de los motores es:

$$u_{v_1}(t_k) = u_{n_{v_1}} + u_\theta(t_k)\ ,\ u_{v_3}(t_k) = u_{n_{v_3}} - u_\theta(t_k). \tag{3.14}$$

Donde $u_{v_1}(t_k)$ y $u_{v_3}(t_k)$ son las señales actuantes en los motores 1 y 3, y u_n representa las señales actuantes de los demás controladores.

Control de *roll* (ϕ)

El ángulo ϕ representa la rotación en el eje x en el sistema de coordenadas del cuadricóptero. Elevar el lado derecho del vehículo corresponde a un giro negativo, mientras que bajarlo (elevar el lado izquierdo) corresponde a un giro positivo.
Para elevar el lado derecho, se aumenta la velocidad en el motor 2 y se disminuye en la misma proporción en el motor 4, para bajarlo se hace lo contrario.
Con ésto se puede ver que el controlador del ángulo ϕ en el cuadricóptero sólo actúa sobre los motores 2 y 4.
La figura 3.9 muestra el diagrama del controlador PID de *roll*. El esquema es el mismo que en el controlador de *pitch*, con la diferencia de que actúan sobre diferentes motores.

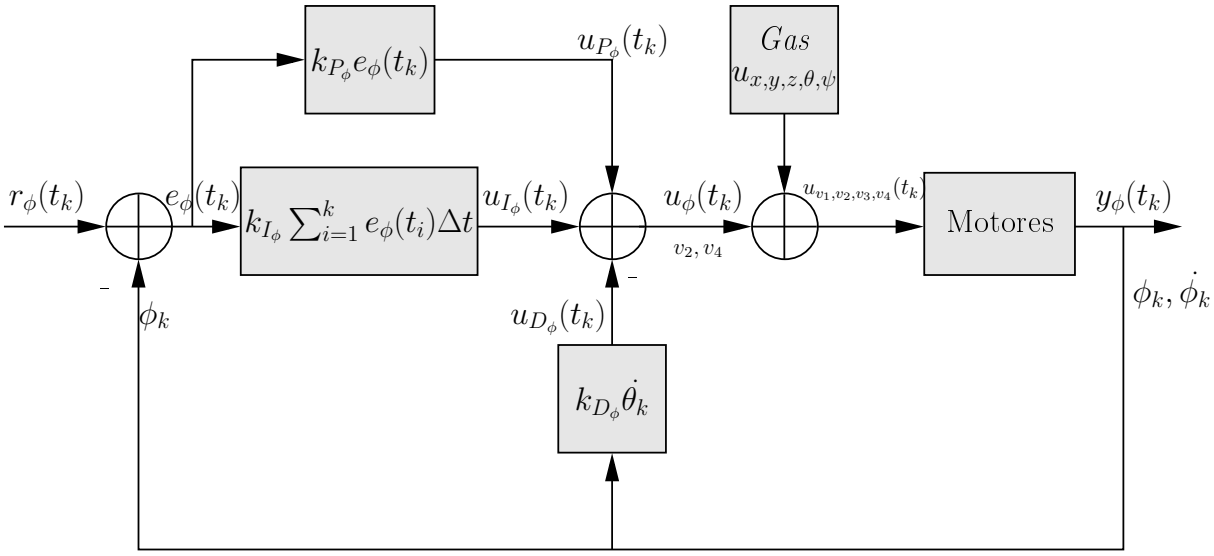


Figura 3.9: Diagrama del controlador PID de *roll*.

La fórmula del controlador PID para ϕ es la siguiente:

$$u_\phi(t_k) = u_{P_\phi}(t_k) + u_{I_\phi}(t_k) - u_{D_\phi}(t_k) = K_{P_\phi} \cdot e_\phi(t_k) + k_{I_\phi} \sum_{i=1}^{k} e_\phi(t_i)\Delta t - K_{D_\phi} \cdot \dot{\phi}(t_k). \tag{3.15}$$

Donde:
$u_\phi(t_k)$ es la variable de control en ϕ.

$u_{P_\phi}(t_k), u_{I_\phi}(t_k), u_{D_\phi}(t_k)$ son las variables de control proporcional, integral y derivativo en ϕ.
$e_\phi(t_k)$ es la señal de error en ϕ.
$K_{P_\phi}, k_{I_\phi}, K_{D_\phi}$ son las constantes proporcional, derivativa e integral en ϕ.
$\dot{\phi}(t_k)$ es la velocidad angular en el eje x en el sistema de coordenadas del vehículo.

El aporte de la variable de control $u_\phi(t_k)$ sobre las velocidades de los motores es:

$$u_{v_2}(t_k) = u_{n_{v_2}} - u_\phi(t_k) \ , \ u_{v_4}(t_k) = u_{n_{v_4}} + u_\phi(t_k). \tag{3.16}$$

Donde $u_{v_2}(t_k)$ y $u_{v_4}(t_k)$ son las señales actuantes en los motores 2 y 4, y u_n representa las señales actuantes de los demás controladores.

Control de *yaw* (ψ)

El ángulo *yaw* representa un movimiento de rotación en el eje z del sistema de coordenadas del vehículo (figura 3.7). El giro es negativo en el sentido de las manecillas del reloj y positivo en contra de las manecillas.
Al aumentar la velocidad en los motores 1 y 3, y disminuirla en la misma magnitud en los motores 2 y 4, se produce un giro negativo, mientras que para producir un giro positivo se hace lo contrario.
A diferencia de los controladores de los ángulos θ y ϕ, el controlador PID del ángulo ψ actúa sobre los cuatro motores del cuadricóptero.

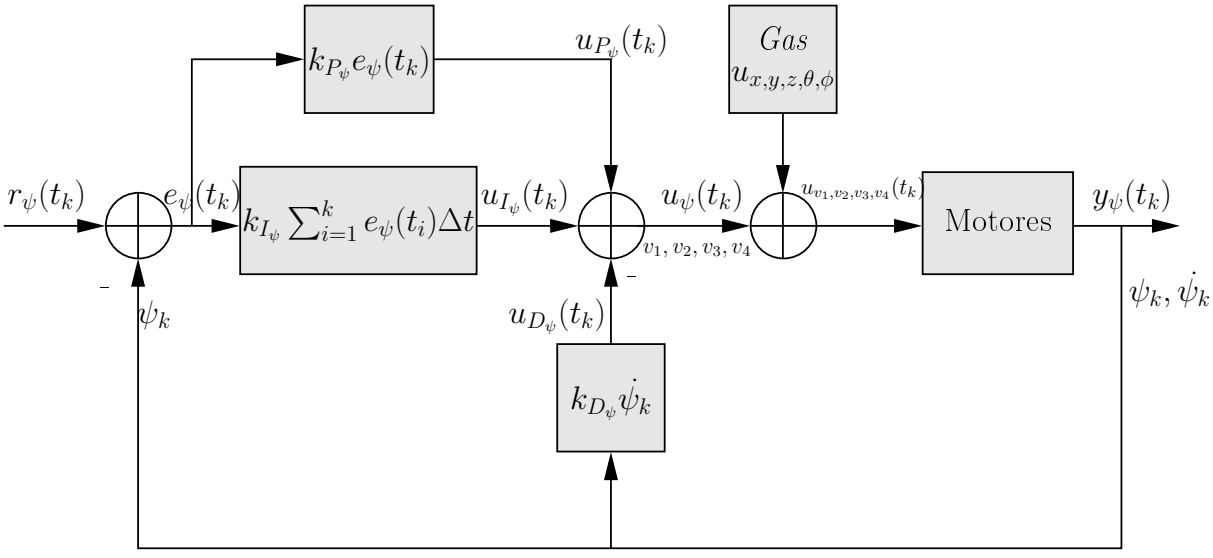


Figura 3.10: Diagrama del controlador PID de *yaw*.

De la figura 3.10 obtenemos la ecuación del controlador PID para ψ, la cual se define de la siguiente manera:

$$u_\psi(t_k) = u_{P_\psi}(t_k) + u_{I_\psi}(t_k) - u_{D_\psi}(t_k) = K_{P_\psi} \cdot e_\psi(t_k) + k_{I_\psi} \sum_{i=1}^{k} e_\psi(t_i)\Delta t - K_{D_\psi} \cdot \dot{\psi}(t_k). \tag{3.17}$$

Donde:
$u_\psi(t_k)$ es la variable de control en ψ.
$u_{P_\psi}(t_k), u_{I_\psi}(t_k), u_{D_\psi}(t_k)$ son las variables de control proporcional, integral y derivativon en ψ.
$e_\psi(t_k)$ es la señal de error en ψ.
$K_{P_\psi}, k_{I_\psi}, K_{D_\psi}$ son las constantes proporcional, derivativa e integral en ψ.
$\dot{\psi}(t_k)$ es la velocidad angular en el eje z en el sistema de coordenadas del vehículo.

El aporte de la variable de control de $u_\psi(t_k)$ sobre las velocidades de los motores es:

$$u_{v_1}(t_k) = u_{n_{v_1}} - u_\psi(t_k) \ , \ u_{v_3}(t_k) = u_{n_{v_3}} - u_\psi(t_k). \tag{3.18}$$

$$u_{v_2}(t_k) = u_{n_{v_2}} + u_\psi(t_k) \ , \ u_{v_4}(t_k) = u_{n_{v_4}} + U_\psi(t_k). \tag{3.19}$$

Donde $u_{v_1}(t_k)$, $u_{v_2}(t_k)$, $u_{v_3}(t_k)$ y $u_{v_4}(t_k)$ son las señales actuantes en los motores 1, 2, 3 y 4, y u_n representa las señales actuantes de los demás controladores.

3.2.2. Control de posición

El control de posición se divide en dos subcontroles: un control automático de altura y un control manual de posición en los ejes x y y con respecto al sistema de coordenadas en tierra.
Como se muestra en la figura 3.11, se utiliza un controlador PID para la altura. Para mover el vehículo en las direcciones x y y, se varían los ángulos de la postura (por ejemplo, para ir hacia adelante se cambia el valor de referencia del controlador de *pitch* a un ángulo negativo), por lo que el control manual consiste en enviar los valores de referencia r_θ, r_ϕ y r_ψ al controlador de postura.
Al combinar las variables de control de postura, altura y *Gas*, se obtiene la señal actuante total del sistema.

Control de altura (z)

El control de altura se hace variando la velocidad de todos los motores con la misma magnitud. Para elevar el vehículo se incrementa y para bajarlo se decrementa.
Como puede verse en la figura 3.12, se implementó un control PID clásico donde $r_z(t_k)$

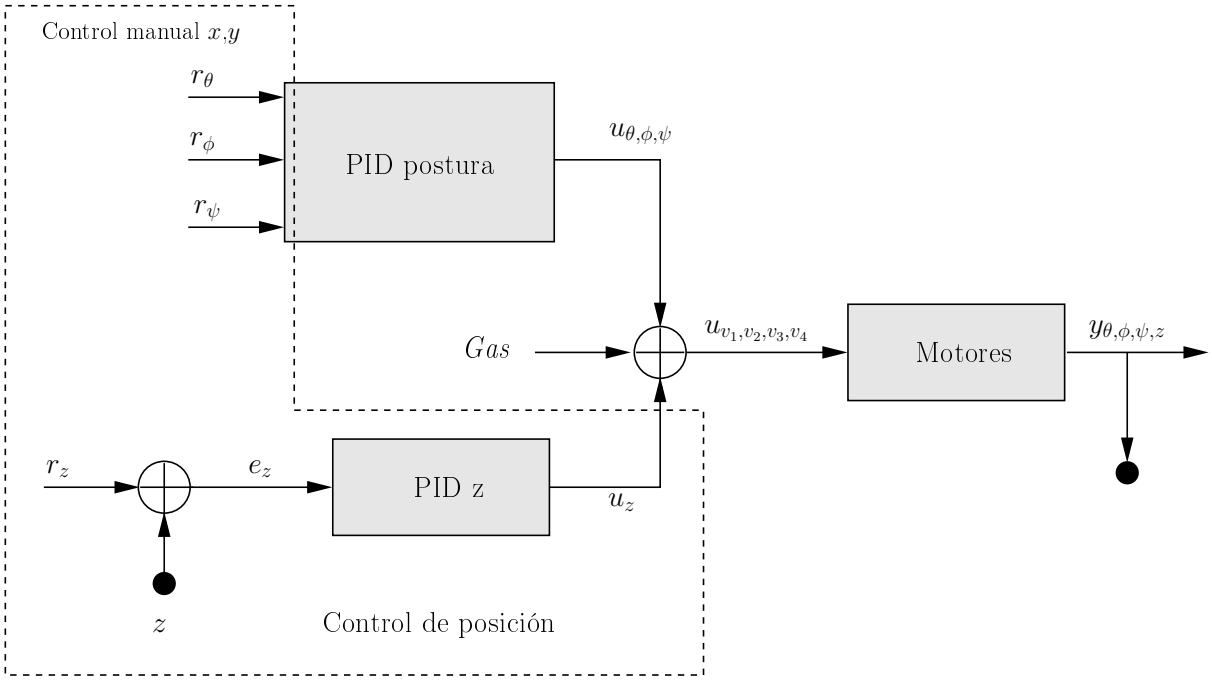


Figura 3.11: Diagrama del control de posición.

es la altura deseada y la señal de error $e_z(t_k) = r_z(t_k)$ - $y_z(t_k)$ es la diferencia entre la altura deseada y la altura real del vehículo medida por los sensores. Esta señal de error es tomada por las partes de control proporcional, integral y derivativa, de las cuales se obtienen las variables de control $u_{P_z}(t_k)$, $u_{I_z}(t_k)$ y $u_{D_z}(t_k)$. La suma de estas variables resulta en la señal actuante de altura, que junto con las variables de control de postura y *Gas*, es aplicada a los motores del cuadricóptero.

La variable de control u_{P_z} obtenida del término proporcional queda de la siguiente manera:

$$u_{P_z}(t_k) = K_{P_z} \cdot e_z(t_k), \tag{3.20}$$

donde K_{P_z} es la constante proporcional en z y $e_z(t_k)$ es el error en el instante de tiempo t_k

La variable de control u_{I_z} aportada por el término integral se obtiene con la fórmula:

$$u_{I_z}(t_k) = k_{I_z} \sum_{i=1}^{k} e_z(t_i)\Delta t, \tag{3.21}$$

donde k_{I_z} es la constante integral en z, $e_z(t_i)$ es el error en tiempo continuo en el

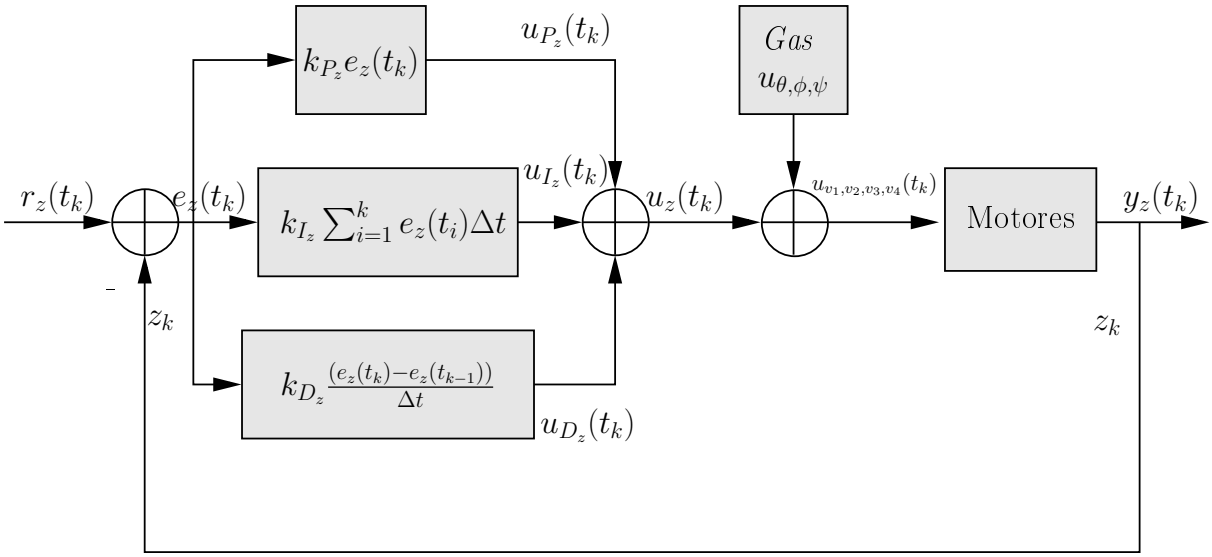


Figura 3.12: Diagrama del controlador PID de altura.

i-ésimo instante de muestreo y Δt es el intervalo de muestreo.

La acción de control U_{D_z} del término derivativo es la siguiente:

$$u_{D_z}(t_k) = K_{D_\theta} \cdot \frac{e_z(t_k) - e_z(t_{k-1)}}{\Delta t}, \tag{3.22}$$

donde K_{D_z} es la constante derivativa en z ,$e_z(t_k)$ es el error en el instante de tiempo t_k y $e_z(t_{k-1})$ es el error en el muestreo anterior.

Por lo que la variable de control en z es:

$$u_z(t_k) = u_{P_z}(t_k) + u_{I_z}(t_k) - u_{D_z}(t_k) = K_{P_z} \cdot e_z(t_k) + k_{I_z} \sum_{i=1}^{k} e_z(t_i)\Delta t - K_{D_z} \cdot \frac{e_z(t_k) - e_z(t_{k-1)}}{\Delta t}. \tag{3.23}$$

En conclusión, el aporte de u_z sobre las velocidades de los motores es:

$$u_{v_1}(t_k) = u_{n_{v_1}} + u_z(t_k) \ , \ u_{v_2}(t_k) = u_{n_{v_2}} + u_z(t_k). \tag{3.24}$$

$$u_{v_3}(t_k) = u_{n_{v_3}} + u_z(t_k) \ , \ u_{v_4}(t_k) = u_{n_{v_4}} + u_z(t_k). \tag{3.25}$$

Donde $u_{v_1}(t_k)$, $u_{v_2}(t_k)$, $u_{v_3}(t_k)$ y $u_{v_4}(t_k)$ son las señales actuantes de los motores 1, 2, 3 y 4 respectivamente, y u_n representa las señales actuantes de los demás controladores.

3.2.3. Control general del sistema

La figura 3.13 muestra el diagrama del sistema de control del cuadricóptero. Éste cuenta con cuatro controladores PID: tres para la postura y uno para la altura. Cada señal actuante de estos controladores tiene un aporte sobre la velocidad de cada uno de los cuatro motores del vehículo.
La variable de control total en cada motor es la suma de los aportes de todos los controladores PID más la variable de referencia *Gas*.

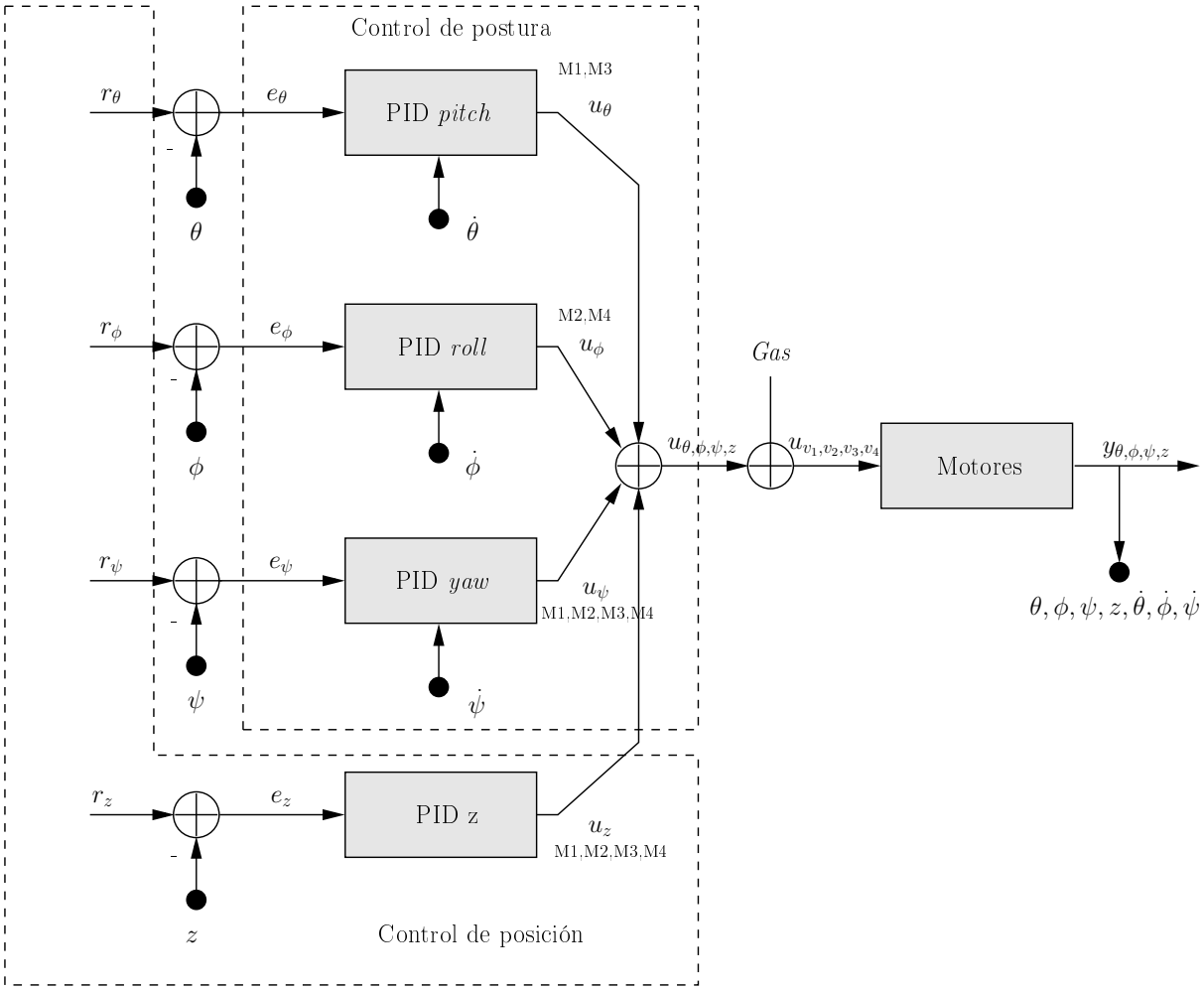


Figura 3.13: Diagrama del control general del sistema.

Uniendo las variables de control de las ecuaciones 3.14, 3.16, 3.18, 3.19, 3.24 y 3.25 obtenemos:

$$u_{v_1}(t_k) = Gas + u_\theta(t_k) - u_\psi(t_k) + u_z(t_k). \tag{3.26}$$

$$u_{v_2}(t_k) = Gas - u_\phi(t_k) + u_\psi(t_k) + u_z(t_k). \tag{3.27}$$

$$u_{v_3}(t_k) = Gas - u_\theta(t_k) - u_\psi(t_k) + u_z(t_k). \tag{3.28}$$

$$u_{v_4}(t_k) = Gas + u_\phi(t_k) + u_\psi(t_k) + u_z(t_k). \tag{3.29}$$

Donde:
u_{v_1}, u_{v_2}, u_{v_3} y u_{v_4}, son las variables de control aplicadas a los motores 1, 2, 3 y 4 respectivamente.
u_θ, es la variable de control en *pitch*.
u_ϕ, es la variable de control en *roll*.
u_ψ, es la variable de control en *yaw*.
u_z, es la variable de control de altura.

3.3. Ajuste de parámetros del controlador PID por medio de evolución diferencial

Uno de los factores cruciales para el buen funcionamiento del controlador PID es el ajuste de sus parámetros. Un correcto ajuste de las constantes proporcional, integral y derivativa es la diferencia entre un buen desempeño y un desempeño pobre del controlador.
Ésto puede verse como un problema de optimización, donde se trata de minimizar la función $e(k_P, k_I, k_D)$ que es la señal de error, tomando a k_P, k_I y k_D como las variables de decisión.
En la sección anterior vimos que el sistema de control del cuadricóptero consta de cuatro controladores PID, por lo que debe hacerse el ajuste de doce constantes.
Una forma de resolver este problema es usando algoritmos evolutivos, ya que no es necesario conocer el modelo del sistema para llegar a resultados aceptables.
En esta sección se dará una breve introducción a los algoritmos evolutivos, después se explicará el algoritmo de evolución diferencial (una rama de la computación evolutiva), para terminar explicando el método de obtención de las constantes del controlador del cuadricóptero utilizando dicho algoritmo.

3.3.1. Algoritmos evolutivos

Los algoritmos evolutivos son métodos de optimización y búsqueda de soluciones basados en los postulados de la evolución biológica. Existen varios tipos de algoritmos evolutivos, pero la idea básica detrás de estas técnicas es la misma: dada una población de individuos, la presión del ambiente causa una selección natural (supervivencia del más apto), que produce un aumento en la aptitud de la población.
Dada una función de calidad a ser maximizada, se crea aleatoriamente un conjunto de soluciones candidatas y se aplica la función de calidad como una medida de desempeño. Basándose en su aptitud, algunos de los mejores candidatos son escogidos para

crear la siguiente generación, mediante su recombinación y/o mutación.
La recombinación es un operador aplicado a dos o más candidatos (padres), del cual resultan uno o más candidatos nuevos (hijos). La mutación es aplicada a un candidato y resulta en un nuevo candidato. Por medio de estos operadores se obtiene un nuevo conjunto de candidatos (descendencia) que compiten, basados en su desempeño, contra los viejos por un lugar en la nueva generación. Este proceso puede ser iterado hasta que es encontrado un candidato con suficiente calidad (una solución) o cierto criterio de paro es alcanzado.
En este proceso hay dos fuerzas fundamentales que forman la base de los sistemas evolutivos:

- Operadores de variación (recombinación y mutación) que crean la diversidad necesaria.
- Selección, que actúa como una fuerza que incrementa la calidad.

La aplicación combinada de variación y selección generalmente lleva al mejoramiento de los valores de desempeño en generaciones consecutivas. Durante la selección, los individuos más aptos tienen más oportunidad de ser seleccionados que los menos aptos, aunque los individuos débiles también tienen oportunidad de ser padres o sobrevivir. La selección de qué partes de los individuos serán mutadas o recombinadas se hace de manera aleatoria [Eiben and Smith, 2008].
Suele hablarse de tres paradigmas principales de los algoritmos evolutivos:

- Programación evolutiva
- Estrategias evolutivas
- Algoritmos genéticos

Cada uno de estos paradigmas se originó de manera independiente y con distintas motivaciones. Actualmente, los algoritmos evolutivos tienden a combinar características de estos tres. Por ejemplo, la evolución diferencial, los algoritmos culturales y los algoritmos meméticos, entre otros.
En este trabajo de tesis se utilizó la evolución diferencial, que será descrita en la siguiente sección.

3.3.2. Evolución diferencial

La evolución diferencial (ED) es un método de optimización perteneciente a la categoría de computación evolutiva, aplicado en la resolución de problemas complejos. Al igual que otros algoritmos de esta categoría, la ED mantiene una población de soluciones candidatas, las cuales se recombinan y mutan para producir nuevos individuos, los cuales serán elegidos de acuerdo al valor de su función de desempeño. Lo

que caracteriza a la ED es el uso de vectores de prueba, los cuales compiten con los individuos de la población actual a fin de sobrevivir.
Fue creado en 1995 por Kenneth Price y Rainer Storn, y desde entonces ha ganado la reputación de ser un optimizador global muy efectivo. Su historial de desempeño robusto y confiable, lo han llevado a ser utilizado de manera exitosa en muchos problemas del mundo real [Price et al., 1997].
Este algoritmo codifica todos los parámetros como números de punto flotante, manipulándolos con operadores aritméticos. Ésto ofrece ventajas significantes sobre los algoritmos genéticos, que codifican sus parámetros como cadenas de bits y utilizan operadores lógicos para manipularlos.
Algunas de estas ventajas son:

- Facilidad de uso
- Utilización de memoria eficiente
- Menor complejidad computacional (escala mejor en problemas grandes)
- Menor esfuerzo computacional (convergencia más rápida)

La ED se compone básicamente de cuatro pasos:

- Inicialización
- Mutación
- Recombinación
- Selección

Estos pasos serán explicados a continuación con más detalle.

Inicialización

Antes de que la población pueda ser inicializada, se tienen que especificar los límites superior e inferior de cada parámetro. Estos valores se pueden guardar en dos vectores $\mathbf{L}_I$ y $\mathbf{L}_S$, donde los subíndices I y S indican los límites inferior y superior, respectivamente.
Una vez especificados los límites, se asignan aleatoriamente los valores de cada parámetro, respetando los rangos preestablecidos.
Por ejemplo, el valor inicial ($g = 0$) del j-ésimo parámetro del i-ésimo vector es:

$$X_{j,i,0} = rand(0,1) \cdot (L_{j,S} - L_{j,I}) + L_{j,I} \tag{3.30}$$

Donde el generador de números aleatorios $rand_j(0,1)$ regresa un número aleatorio uniformemente distribuido dentro del rango [0,1). El subíndice j indica que se genera un nuevo valor aleatorio para cada parámetro.

Mutación

Una vez inicializada, la ED muta y recombina la población para producir otra población de vectores de prueba. La mutación diferencial suma la diferencia escalada de dos vectores tomados al azar a un tercer vector, también tomado al azar, obteniendo el vector mutado $\mathbf{V}_{i,g}$.

$$\mathbf{V}_{i,g} = \mathbf{X}_{r0,g} + F \cdot (\mathbf{X}_{r1,g} - \mathbf{X}_{r2,g}). \tag{3.31}$$

Donde:
$\mathbf{V}_{i,g}$, es el vector mutante i en la generación g.
$\mathbf{X}_{r0,r1,r2,g}$, son los vectores tomados al azar de la población actual g (diferentes).
F, es el factor de escala (0,1+).

El factor de escala es un número real positivo que controla la velocidad a la cual la población evoluciona. No hay un límite superior para este valor, pero rara vez hay valores efectivos mayores que 1. El vector base $\mathbf{X}_{r0,g}$ puede determinarse de varias maneras, aunque también puede tomarse al azar. Los vectores $\mathbf{X}_{r1,g}$ y $\mathbf{X}_{r2,g}$ también son seleccionados al azar una vez por mutante.

Recombinación

En la recombinación se forman vectores de prueba, a partir de valores de parámetros que han sido copiados de dos vectores diferentes. En este caso se cruza cada vector con un vector mutante.

$$\mathbf{U}_{i,g} = U_{j,i,g} = \begin{cases} V_{j,i,g} \text{ si } (rand_j(0,1) \leq C_r) \text{ ó } j = j_{rand} \\ X_{j,i,g} \text{ de otra manera} \end{cases} \tag{3.32}$$

Donde:
$V_{j,i}$, es el parámetro j del vector mutante i.
$X_{j,i}$, es el parámetro j del vector i.
C_r , es la probabilidad de cruza [0,1].

La probabilidad de cruza C_r, es un valor definido por el usuario que controla la fracción de valores de los parámetros que son copiados del vector mutante.
Para determinar qué vector contribuirá con un parámetro, se compara C_r con la salida de un generador de números aleatorios uniformemente distribuidos $rand_j(0,1)$. Si el número aleatorio es menor o igual a C_r, el parámetro de prueba es heredado por el vector mutante $\mathbf{V}_{i,g}$. De otra manera, el parámetro es copiado del vector $\mathbf{X}_{i,g}$. Además, un parámetro de prueba seleccionado al azar (j_{rand}) se toma del vector mutante, esto asegura que al menos un parámetro mutante será heredado, evitando así que el vector de prueba $\mathbf{U}_{i,g}$ sea igual al vector $\mathbf{X}_{i,g}$.

Selección

Si el valor de la función objetivo del vector de prueba $\mathbf{U}_{i,g}$ es igual o menor al vector $\mathbf{X}_{i,g}$, éste reemplaza a $\mathbf{X}_{i,g}$ en la siguiente generación. De otra manera, el vector $\mathbf{X}_{i,g}$ permanece en su lugar.

$$\mathbf{X}_{i,g+1} = \begin{cases} \mathbf{U}_{i,g} \text{ si } f(\mathbf{U}_{i,g}) \leq f(\mathbf{X}_{i,g}) \\ \mathbf{X}_{i,g} \text{ de otra manera} \end{cases} \tag{3.33}$$

Donde:
$f()$, es la función objetivo.
$\mathbf{U}_{i,g}$, es el vector de prueba i.
$\mathbf{X}_{i,g}$, es el vector i en la población actual.

Una vez que la nueva población es seleccionada, el proceso de mutación, recombinación y selección es repetido hasta que el óptimo es localizado, o un criterio de paro preestablecido es alcanzado, por ejemplo, el número de generaciones alcanza un máximo preestablecido [Price et al., 1997].

3.3.3. Ajuste de parámetros del controlador del cuadricóptero

Como se vió en secciones anteriores la fórmula de un controlador PID es:

$$u(t_k) = K_P e(t_k) + k_I \sum_{i=1}^{k} e(t_i)\Delta t + K_D \frac{e(t_k) - e(t_{k-1})}{\Delta t}, \tag{3.34}$$

donde se tienen que encontrar los valores de las constantes proporcional (K_P), integral (k_I) y derivativa (k_D) más adecuados para el correcto funcionamiento del controlador. Ésto puede verse como un problema de optimización donde se trata de minimizar la función $e(k_P, k_I, k_D)$, que es la señal de error del sistema, tomando como variables de decisión las constantes del PID.
El control de postura del vehículo consta de tres controladores PID por lo que hay que encontrar nueve constantes, mientras que el control de altura, con un PID, requiere del ajuste de tres constantes.

Ajuste de parámetros con evolución diferencial

Para simplificar el problema y poder realizar más fácilmente las pruebas de laboratorio, se dividió el problema de ajuste de parámetros en dos: ajuste de parámetros de postura y ajuste de parámetros de altura.
Para aplicar el algoritmo de evolución diferencial, primero se definen los individuos como los vectores:

$$postura : \mathbf{X} = [k_{P_\theta}, k_{I_\theta}, k_{D_\theta}, k_{P_\phi}, k_{I_\phi}, k_{D_\phi}, k_{P_\psi}, k_{I_\psi}, k_{D_\psi}] \in \mathbb{R}^9, \tag{3.35}$$

$$altura : \mathbf{X} = [k_{P_z}, k_{I_z}, k_{D_z}] \in \mathbb{R}^3, \tag{3.36}$$

donde:
k_P en θ, ϕ, ψ, z son las constantes proporcionales de los controladores de *pitch*, *roll*, *yaw* y altura.
k_I en θ, ϕ, ψ, z son las constantes integrales de los controladores de *pitch*, *roll*, *yaw* y altura.
k_D en θ, ϕ, ψ, z son las constantes derivativas de los controladores de *pitch*, *roll*, *yaw* y altura.

Los límites inferior y superior de cada parámetro fueron definidos experimentalmente y guardados en los vectores L_S y L_I en $\mathbb{R}^9$ para la postura, y L_S y L_I en $\mathbb{R}^3$ para la altura.
Una población consta de treinta individuos, con un criterio de paro del algoritmo de 200 generaciones. Cada evaluación del individuo se hace en tiempo real, ésto es, se cargan los parámetros del individuo a evaluar en el algoritmo del controlador del vehículo, y éste se enciende por diez segundos mientras se guardan las mediciones de las señales de error de *pitch* (e_θ), *roll* (e_ϕ) y *yaw* (e_ψ), en el caso de la postura y de e_z, en el caso de la altura.
El periodo de muestreo en el caso de las señales e_θ,e_ϕ y e_ψ, es de 10 milisegundos, por lo que resultan tres vectores en $\mathbb{R}^{1000}$. Mientras que el periodo de muestreo en e_z es de 100 milisegundos, quedando un vector en $\mathbb{R}^{100}$.
La función de desempeño para los individuos del control de postura es:

$$f(\mathbf{X}) = \frac{1}{3}\sum_{i=0}^{59} |e_\theta(t_i)\cdot 0.0166i| + |e_\phi(t_i)\cdot 0.0166i| + |e_\psi(t_i)\cdot 0.0166i| + \frac{1}{3}\sum_{i=60}^{999} |e_\theta(t_i)| + |e_\phi(t_i)| + |e_\psi(t_i)|, \tag{3.37}$$

donde la sumatoria de los primeros 60 elementos de los vectores de error, es una suma ponderada de los valores absolutos de cada elemento. Ésto se hace para que el error encontrado en los primeros 600 milisegundos tenga menor importancia que el error encontrado en el resto del tiempo, ya que las posiciones iniciales (error inicial) pueden variar un poco entre un individuo y otro.
En la segunda sumatoria, se toman los valores absolutos de las señales de error en el instante i de los 9.4 segundos de prueba restantes.

La función de desempeño para los individuos del control de altura es:

$$f(\mathbf{X}) = \sum_{i=0}^{9} |e_z(t_i)\cdot 0.1i| + \sum_{i=10}^{99} |e_z(t_i)|, \tag{3.38}$$

La cual es bastante semejante a la de altura, pero ésta sólo evalúa la señal de error e_z

Teniendo ya todos los elementos necesarios para utilizar la evolución diferencial, se implementan los pasos de inicialización, mutación, recombinación y selección, descritos anteriormente, para encontrar las constantes óptimas. El algoritmo implementado será descrito más adelante en el capítulo 5, que trata lo referente a la implementación del *software*.

Capítulo 4

Implementación del *hardware*

El propósito de este trabajo de tesis es, además del control de un cuadricóptero, crear un sistema base que sea versátil y escalable al que se le puedan añadir nuevas características con facilidad. Una pieza clave para lograr este objetivo es el uso de una placa computadora (en inglés: *Single Board Computer* o *SBC*) empotrada en el vehículo en lugar de un simple microcontrolador. Ésto trae grandes ventajas, ya que además de contar con más recursos como memoria RAM y velocidad de procesamiento, se le puede cargar un sistema operativo (en este caso Linux) el cual, al ser una capa intermedia entre el *hardware* y el *software* de aplicación, permite la generación de código portable e independiente de la arquitectura utilizada, pudiendo así actualizar el sistema con mayor facilidad y contando con la ventaja de la reutilización de código fuente.

4.1. Descripción general del sistema

La figura 4.1 muestra un diagrama a grandes rasgos del sistema de control a bordo del cuadricóptero, que consta de cinco módulos principales: sensores, controlador, etapa de potencia, actuadores y comunicación.

Un vehículo aéreo no tripulado puede contener una gran variedad de sensores, altímetros (barómetros) para medir la altitud, sistemas de posicionamiento global (en inglés: *Global Positioning system* o *GPS*) para determinar la ubicación en el espacio, sensores de proximidad para evasión de obstáculos y hasta cámaras de video en modelos más sofisticados. El sensor primordial para el control automático de la postura, es la unidad de medición inercial (UMI), con la cual se pueden determinar los tres ángulos de rotación del cuadricóptero (*pitch, roll, yaw*), así como también la velocidad angular en la que éstos rotan. El módulo de sensores para este proyecto cuenta con una UMI y un sensor ultrasónico para la altitud.

El módulo controlador es el encargado de leer los datos provenientes de los sensores y aplicar una acción de control sobre los actuadores, en este caso cuatro motores. La

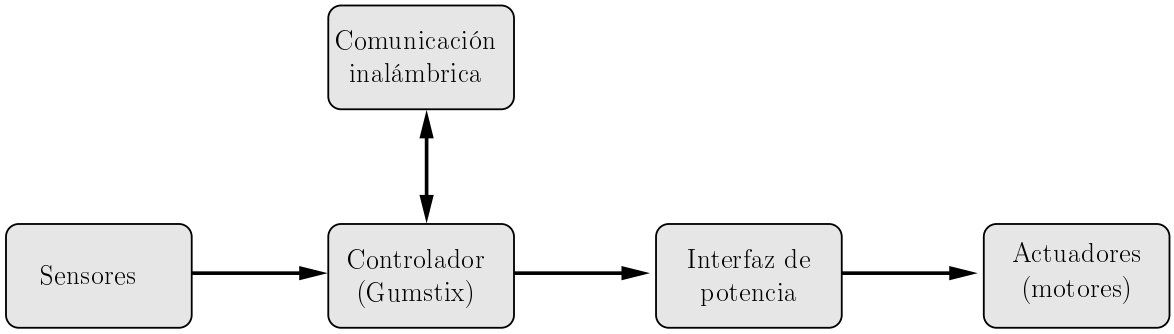


Figura 4.1: Sistema de control a bordo del cuadricóptero.

etapa de potencia se encarga de decodificar las órdenes de control enviadas por el controlador, para luego activar los motores según lo indicado por éste. El controlador también se encarga de recibir, procesar, y enviar datos hacia otros dispositivos conectados a la red de control.

La figura 4.2 muestra el sistema de comunicación entre el cuadricóptero y la estación de control en tierra (o estaciones). Éste está basado en una configuración cliente-servidor por *WiFi* a través de un ruteador, donde el vehículo aéreo actúa como servidor, atendiendo las peticiones de los clientes. Para este sistema se utilizan dos clientes: una computadora personal y una computadora en tableta. La computadora personal tiene la función de configurar aspectos internos del control de postura, así como de hacer una recopilación y análisis de datos tomados del servidor. La computadora en tableta sirve como control a distancia de la posición del cuadricóptero en el espacio.

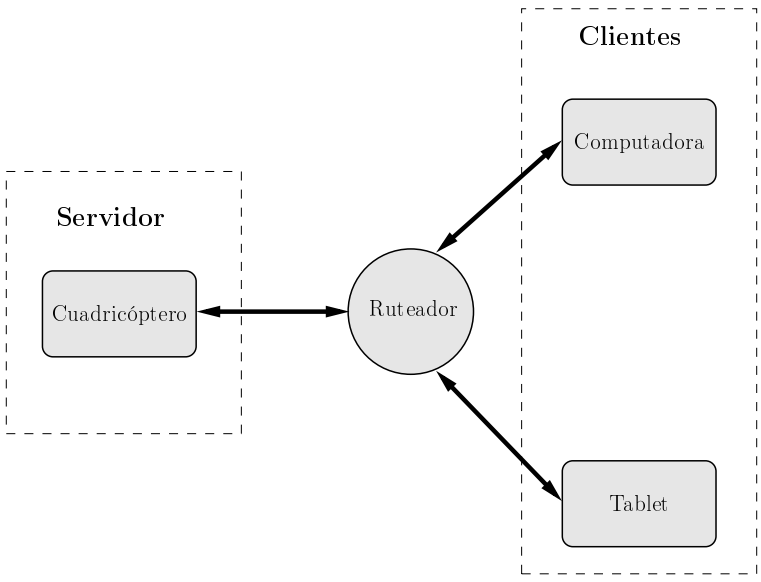


Figura 4.2: Sistema de comunicación.

En las siguientes secciones se explicará con mas detalle cada uno de los elementos que conforman este sistema.

4.2. Descripción del *Hardware*

El sistema de control a bordo fue montado sobre un chasis de la marca *Mikrokopter*, modelo *Quadro XL-Frame*. Éste consiste en cuatro barras de aluminio sujetadas por dos láminas de fibra de carbono (figura 4.3). Tiene una longitud de 47 centímetros de punta a punta y pesa aproximadamente 230 gramos . Las hélices utilizadas son de plástico con una longitud de 25 cm.

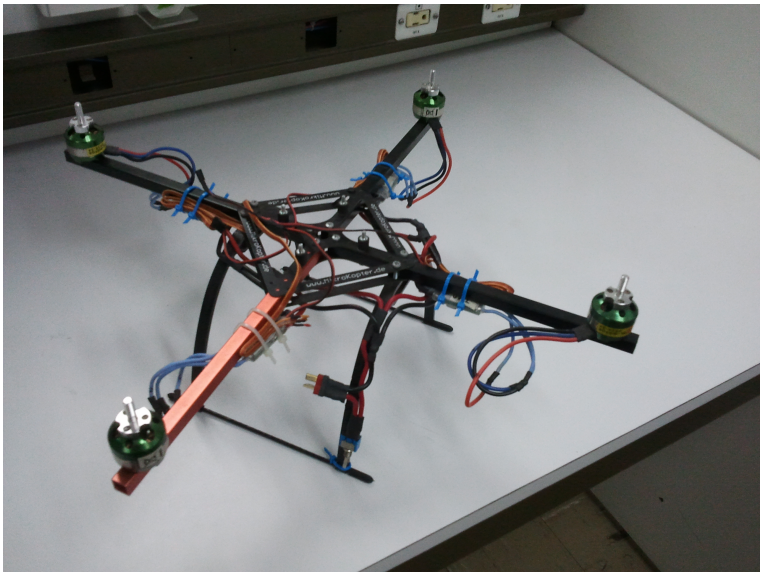


Figura 4.3: Chasis del cuadricóptero con motores.

4.2.1. Controlador

Se utilizó una placa computadora de la marca *Gumstix* (figura 4.4) como controlador principal. Estas computadoras son muy pequeñas (17 x 58 x 4.2 mm) por lo que resultan ideales para usarse en vehículos aéreos no tripulados. Estas tarjetas pueden montarse en diferentes placas de expansión, que le dan diferentes funcionalidades al sistema. La placa utilizada para este proyecto fue el modelo *Summit* (figura 4.5), que cuenta con interfaces como: puerto serie, bus I^2C (en inglés *Inter-Integrated Circuit*), bus SPI (en inglés: *Serial Peripheral Interface*) y USB (en inglés : *Universal Serial Bus*).

Las características principales de la placa computadora son:

- Arquitectura: ARM Cortex-A8.

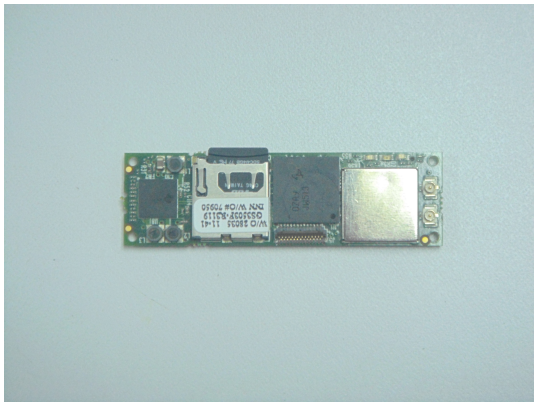


Figura 4.4: Computadora *Gumstix*.

- Procesador: OMAP3530 de Texas Instruments.
- Velocidad del procesador: 720 MHz.
- Memoria RAM: 512 Megabytes.
- Memoria: entrada para tarjeta SD.
- Comunicación inalámbrica: *Bluetooth*, *WiFi*.
- Sistema Operativo: Linux (Angstrom).

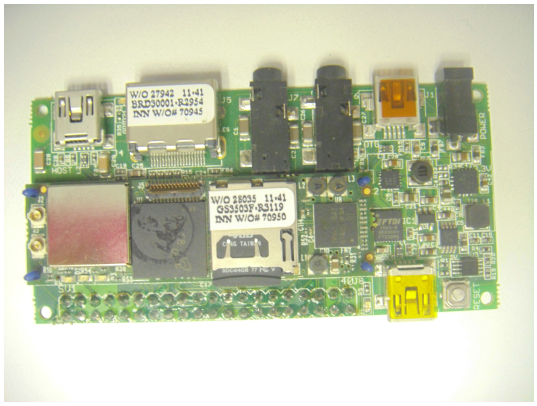


Figura 4.5: *Gumstix* montado en placa de expansión *Summit*.

Como se puede notar, esta tarjeta cuenta con bastantes recursos, por lo que se pueden agregar funcionalidades al sistema sin hacer cambios al *Hardware*.

4.2.2. Sensores

Unidad de medición inercial (UMI)

Un sensor UMI (en inglés *IMU* o *Inertial Measurement Unit*) es un dispositivo electrónico que mide velocidad, orientación y fuerzas gravitacionales usando una combinación de acelerómetros, giroscópios y magnetómetros.
Una unidad de medición inercial usualmente contiene tres acelerómetros, tres giroscópios y tres magnetómetros. Los acelerómetros son colocados de tal forma que sus ejes de medición sean ortogonales entre ellos, por lo que se mide la aceleración inercial (también conocida como fuerzas G) en cada eje.
Los tres giroscópios están colocados en un patrón ortogonal similar, midiendo la velocidad angular respecto a cada eje. Los magnetómetros sirven para detectar hacia dónde apunta el vehículo, tomando como referencia el norte magnético.
Cuando se usan en conjunto con un procesador para calcular la orientación relativa en el espacio (*pitch*, *roll*, *yaw*), las UMIs suelen denominarse sistemas de referencia de actitud y rumbo (en inglés *AHRS* o *Attitude and Heading Reference Systems*).

El dispositivo AHRS utilizado en este proyecto es el modelo CHR-6d de *CH Robotics* (figura 4.6). Este dispositivo incluye un procesador ARM Cortex a 64 MHz, el cual utiliza un filtro de Kalman extendido para obtener las estimaciones del *pitch* y *roll*.

Figura 4.6: Unidad de medición inercial CHR-6d de *CH Robotics*.

Las características principales de este sensor son las siguientes:

- Ángulos de Euler o cuanterniones como salidas.
- Comunicación por TTL UART (en inglés: *Universal Asynchronous Receiver Transmitter*) o por bus SPI.
- Velocidades ajustables de salida (20Hz-300Hz) y de comunicación (hasta 115200 baudios).

- Actualizaciones internas a 500Hz.
- Regulador de 3.3 Voltios integrado.
- Precisión en *pitch* y *roll* mejor de dos grados.
- Precisión en *yaw* mejor de cinco grados.
- Velocidad angular máxima medible de +/- 2000 °/s.
- Aceleración máxima medible de +/- 2G.
- *Firmware* de código abierto con herramientas de desarrollo gratuitas.
- *Software* de computadora de código abierto para visualización de datos y configuración.

Sensor ultrasónico

Los sensores ultrasónicos trabajan excitando un transductor acústico con pulsos de voltaje, haciendo que éste vibre y produzca sonidos ultrasónicos. Estas oscilaciones son dirigidas hacia un objetivo y, midiendo el tiempo que tarda en regresar el eco al transductor, se puede calcular la distancia a la que se encuentra el objetivo [Wilson, 2005].
Se utilizó el sensor MB310 de *XL-Maxsonar* (figura 4.7) que cuenta con las siguientes características:

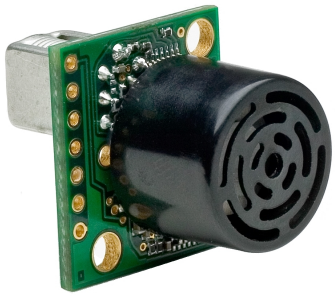


Figura 4.7: Sensor ultrasónico.

- Dimensiones de 22.1mm x 19.9mm x 25.11mm.
- Voltaje de entrada de 3.3 a 5.5 Voltios.
- Salida serial (9600 baudios) o analógica.
- Actualización de mediciones a 10Hz (tarda 99ms en hacer cada medición).

- Rango de distancias de 20 a 765 cm con resolución de 1 cm.
- Operación a 42KHz.
- Autocalibración en cada medición.

4.2.3. Motores

Usualmente los llamados motores sin escobillas (en inglés:"brushless motors") son utilizados en vehículos aéreos no tripulados, ya que éstos presentan muchas ventajas con respecto a los motores eléctricos convencionales (con escobillas), como:

- Son más baratos de fabricar.
- Pesan menos.
- Requieren menos mantenimiento.
- Son más eficientes.
- Generan menor interferencia electromagnética.

Una desventaja es que su control es mucho más complejo, aunque debido a los avances de la tecnología, esta complejidad se ha eliminado con controladores electrónicos.

Los motores utilizados en el vehículo son de la marca *Robbe ROXXY* modelo *BL-Outrunner 2824-34* (figura 4.8), los cuales cuentan con las siguientes características:

- Peso: 48 gr.
- Dimensiones: 28.8 x 26 mm.
- Número de celdas: 2-3 Lixx / 6-10 Nixx.
- RPM/V: 1100.
- Corriente: 4–8 A.
- Corriente máxima: 9 A. (30 segundos).
- Potencia: 90 W.
- Empuje Máximo: 580 gr.

Figura 4.8: Motor sin escobillas.

4.2.4. Interfaz de potencia

La interfaz de potencia se encarga de convertir las señales de salida del controlador en señales apropiadas para el manejo de los motores. En el caso de los motores sin escobillas, se cuentan con controladores eléctronicos comerciales que usualmente toman como entrada una señal de pulso modulada o una serie de datos transmitidos por bus i^2c.
Los controladores utilizados para el cuadricóptero son de la marca *YGE* modelo *30 i* (figura 4.9), que cuentan con las siguientes características:

- Comunicación por i^2c.
- Se pueden conectar hasta ocho controladores en un mismo bus (direcciones 0x29 a 0x30).
- Maneja corrientes de hasta 30 A.

4.2.5. Comunicaciones

Como se dijo anteriormente, el sistema de comunicaciones consiste en una red inalámbrica (*Wi-Fi*) donde el controlador a bordo del cuadricóptero es el servidor y una computadora personal o en tableta pueden ser los clientes.
Esta comunicación se hace a través de un ruteador de la marca *Linksys* modelo *WRT54G*, mostrado en la figura 4.10. Cualquier computadora con sistema operativo Linux y tarjeta de red inalámbrica puede utilizarse como cliente. Se utilizó una computadora en tableta *Galaxy tab 10.1* (figura 4.11) de la marca *Samsung* como controlador en tierra.

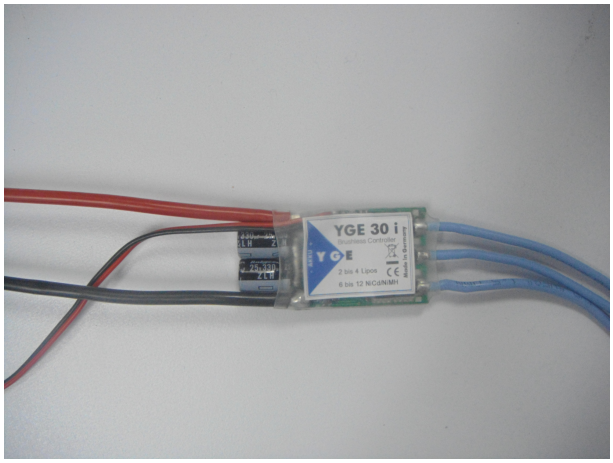


Figura 4.9: Manejador de motores sin escobillas.

Figura 4.10: Ruteador.

4.2.6. Energía

En las pruebas de laboratorio, se utilizó una fuente de computadora de 500 Watts para alimentar a los circuitos y los motores del cuadricóptero. Estas fuentes son más baratas que una fuente de poder lineal y pueden otorgar hasta 25 Amperes con voltajes de 12 Voltios.
En la figura 4.12 se pude ver la fuente utilizada. Se unieron todos los cables de 12v para dar toda la capacidad de corriente.

En las pruebas de vuelo se utilizó una batería de litio polímero (LiPo) de tres celdas y capacidad de 2200 miliAmperes/hora (figura 4.13). Las baterías de litio polímero son muy utilizadas en vehículos aéreos, ya que son muy ligeras y tienen gran capacidad de descarga.

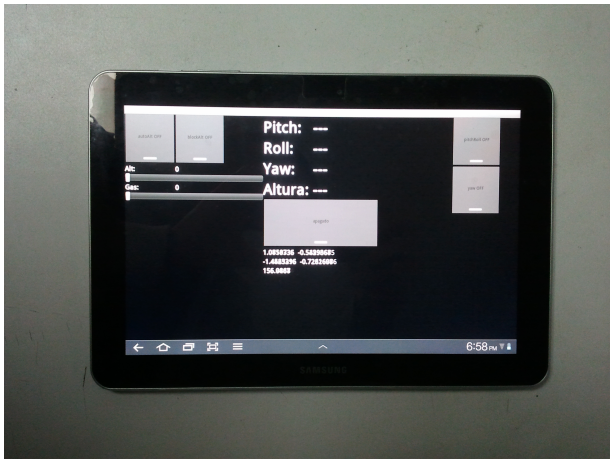


Figura 4.11: Computadora en tableta utilizada como control en tierra.

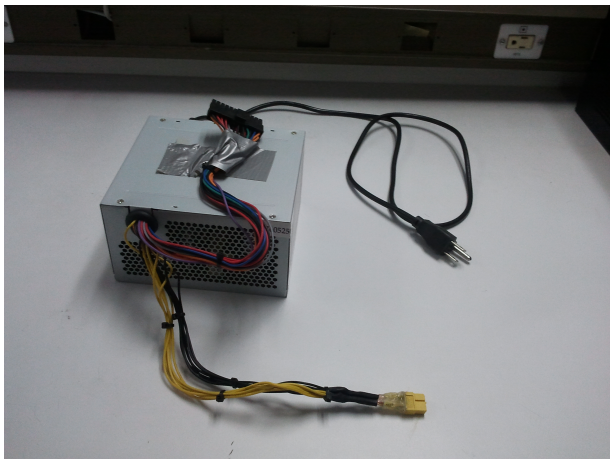


Figura 4.12: Fuente de poder de una computadora.

4.2.7. Otras interfaces

Los componentes que integran al sistema funcionan con diferentes voltajes. Tanto el controlador como los sensores requieren una alimentación de 5 voltios, mientras que los motores y sus manejadores necesitan 12v.
Esta diferencia de voltajes también sucede con las señales de comunicación. La computadora (Gumstix) utiliza señales de 1.8v en su puerto serial y en el bus i^2c, mientras que los manejadores de los motores utilizan 5v en el bus i^2c y la UMI funciona con tensiones de 3.3v en su puerto serial.
Para resolver este problema, se diseño y construyó un circuito electrónico que convierte y regula los voltajes de los componentes para poderlos acoplar entre sí. La figura 4.14 muestra el diagrama esquemático del circuito. Éste consiste en un regulador de voltaje (12 a 5v) y dos "traductores" de voltaje (uno de 1.8 a 5v y otro de 1.8 a 3.3v).

El regulador de voltaje toma como entrada 12v proporcionados por la batería del

Figura 4.13: Batería de litio polímero de tres celdas.

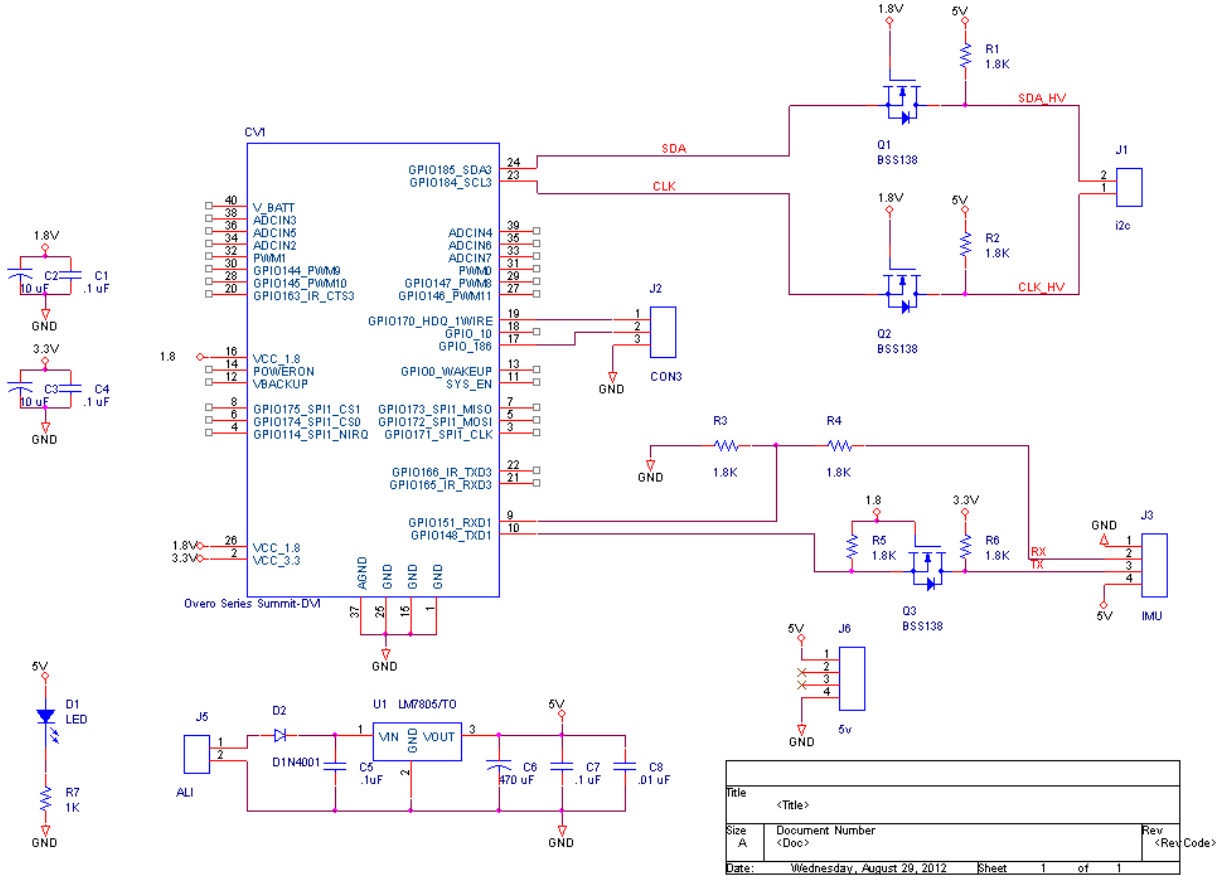


Figura 4.14: Diagrama esquemático del circuito regulador-traductor de voltaje.

vehículo y los convierte a 5v, necesarios para alimentar a la computadora y los sensores. Los traductores de voltaje acoplan las tensiones de las señales de comunicación del controlador con las de los sensores.

La tablilla ya construida puede verse en las figuras 4.15 y 4.16.

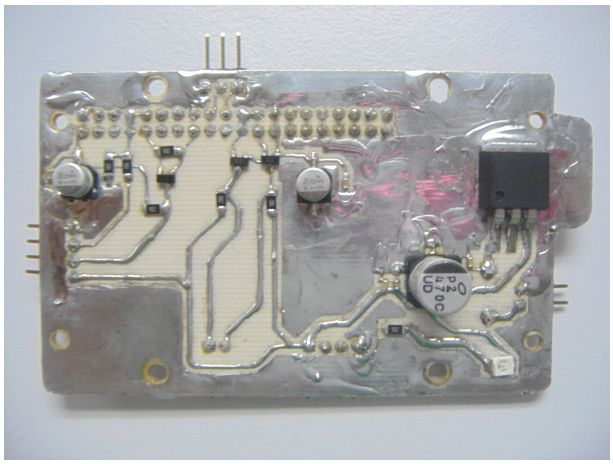


Figura 4.15: Tablilla del circuito regulador-traductor vista por abajo.

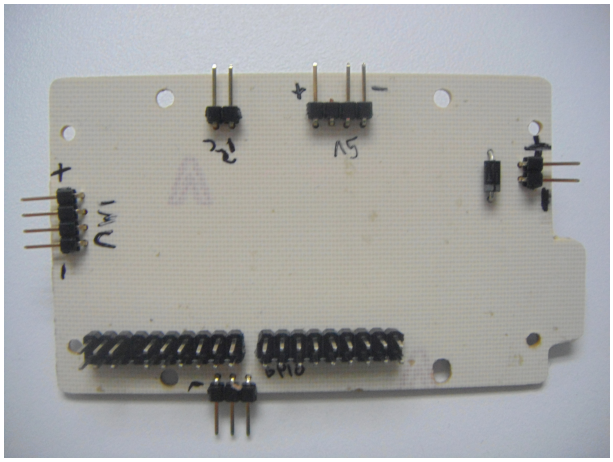


Figura 4.16: Tablilla del circuito regulador-traductor vista por arriba.

Otro problema encontrado en el sistema fue que la computadora sólo tiene un puerto serial disponible, sin embargo, se necesitaban dos puertos: uno para la UMI y otro para el sensor ultrasónico. Ésto se resolvió utilizando un convertidor usb-serial conectado a través de un concentrador USB genérico (figuras 4.17 y 4.18), dando capacidad de conectar hasta tres dispositivos extras.

4.2.8. Estructuras para pruebas del vehículo

Se utilizaron dos estructuras para contener el vehículo en las pruebas de evolución: una para la postura y otra para la altura.
Para las pruebas de postura se utilizó un tripié, como lo muestra la figura 4.19. Éste

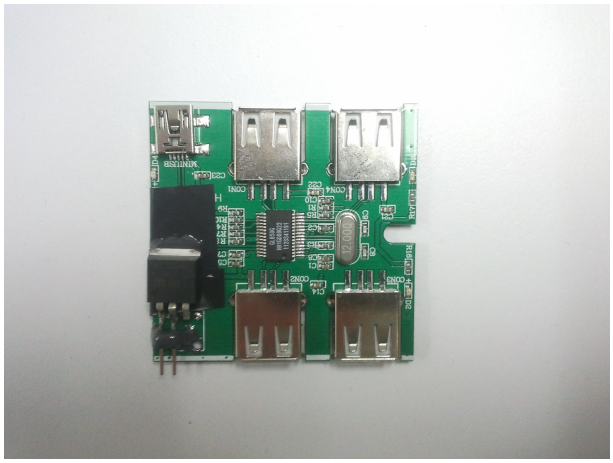


Figura 4.17: Concentrador USB (sin carcasa).

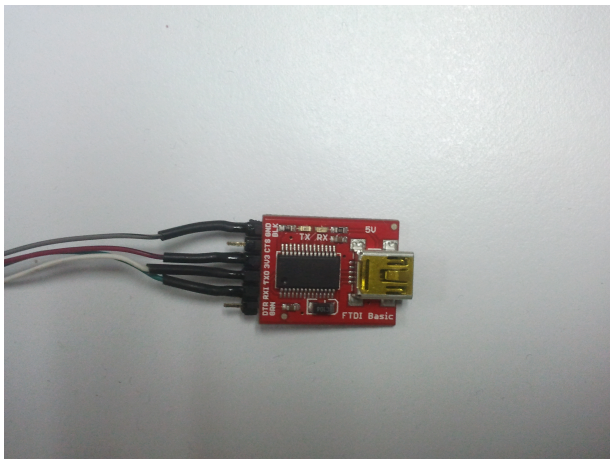


Figura 4.18: Convertidor USB a serial.

cuenta con un pivote que contiene tres baleros que permiten un movimiento con libertad en los ángulos de *pitch*, *roll* y *yaw*; mientras mantienen al vehículo fijo en los ejes x, y y z (figura 4.20).
El cuadricóptero se fija en este pivote y, de esta manera, se hacen las pruebas con seguridad.

Para las pruebas de altura se utilizó un poste con una base pesada, como lo muestra la figura 4.21. A este poste se le unió un palo de madera con cinta elástica para darle una función de "brazo". El vehículo se unió al extremo de esta estructura con cinta elástica. Esta configuración permite el vuelo del cuadricóptero con cierta libertad alrededor del poste y hasta una altura delimitada, probando su utilidad para el entrenamiento de altura.

Figura 4.19: Tripié para pruebas de control de postura.

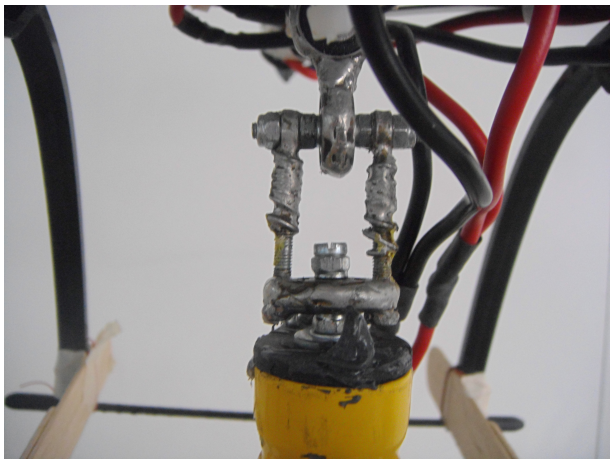


Figura 4.20: Pivote para movimiento en *pitch*, *roll* y *yaw*.

4.3. Construcción del cuadricóptero

En esta sección se describirá la construcción del cuadricóptero, utilizando los componentes mostrados en la sección anterior.
La figura 4.22 nos muestra un diagrama del sistema integrado. Todo esto va a bordo del vehículo, con excepción del ruteador y el cliente.

En el centro está el componente principal, el controlador (computadora *Gumstix*), al cual están conectados todos los demás dispositivos.
A la derecha del diagrama está la conexión del controlador con los motores. Ésta se hace a través del bus i^2c, pasando antes por un traductor de voltaje hacia los manejadores de potencia.
En la figura 4.23 se puede ver una imagen del chasis del vehículo armado con los motores y sus manejadores ya montados.

Figura 4.21: Base para pruebas de control de altura.

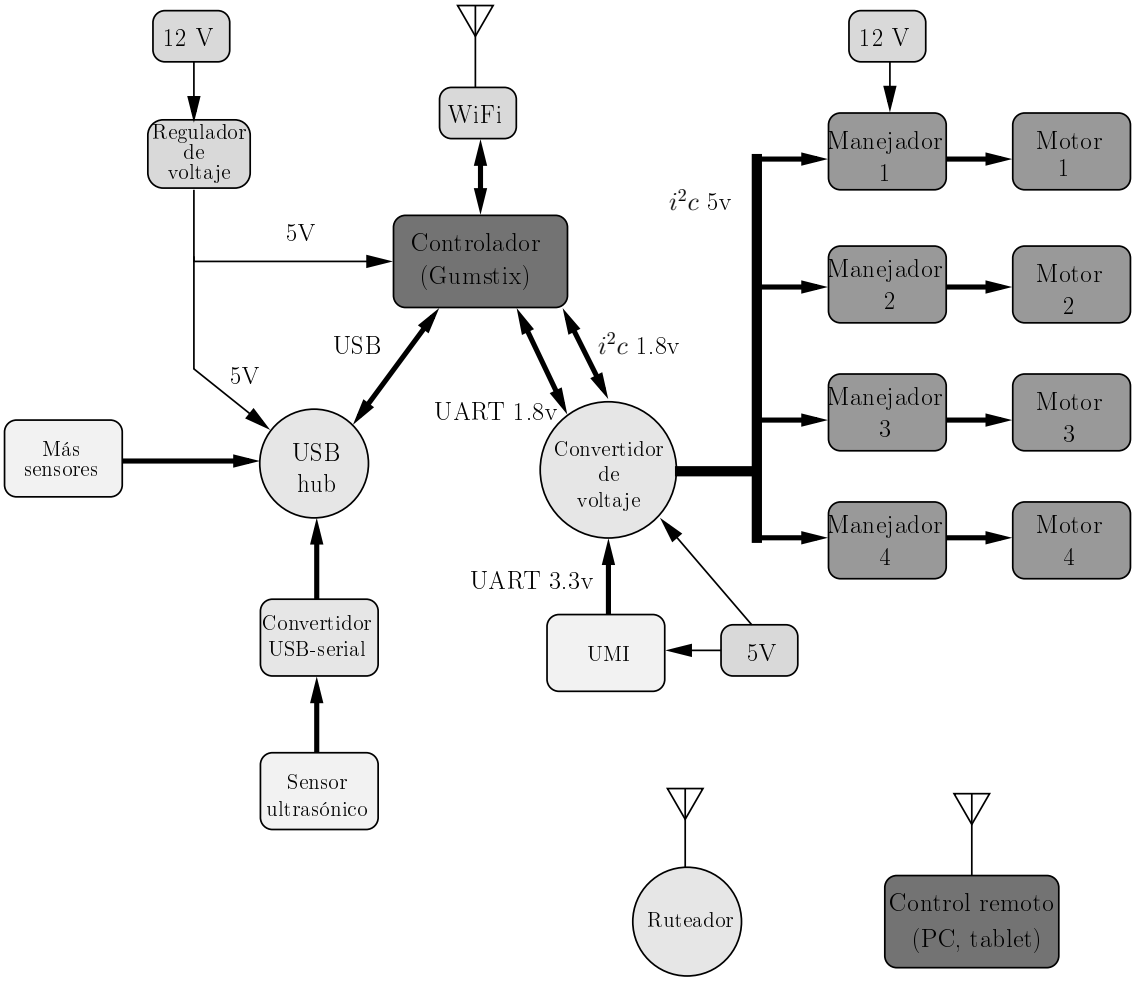


Figura 4.22: Diagrama del *hardware* del sistema de control del cuadricóptero.

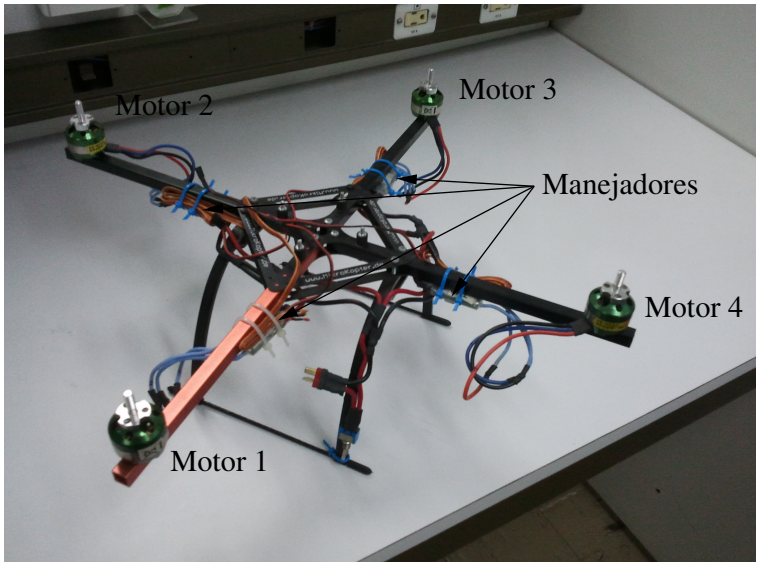


Figura 4.23: Vehículo con motores y manejadores de motores montados.

Para conectar la unidad de medición inercial (UMI) con el controlador, las señales del puerto serial deben pasar antes por un traductor de voltaje de 1.8 a 3.3v.
El controlador, así como también la UMI y el traductor de voltaje, son alimentados con 5v proporcionados por el regulador de voltaje.
La tablilla electrónica con el regulador y los traductores de voltaje se usa además como base para montar la computadora, como se ve en las figuras 4.24 y 4.25. Sobre estas dos placas se monta una más, que contiene la unidad de medición inercial.

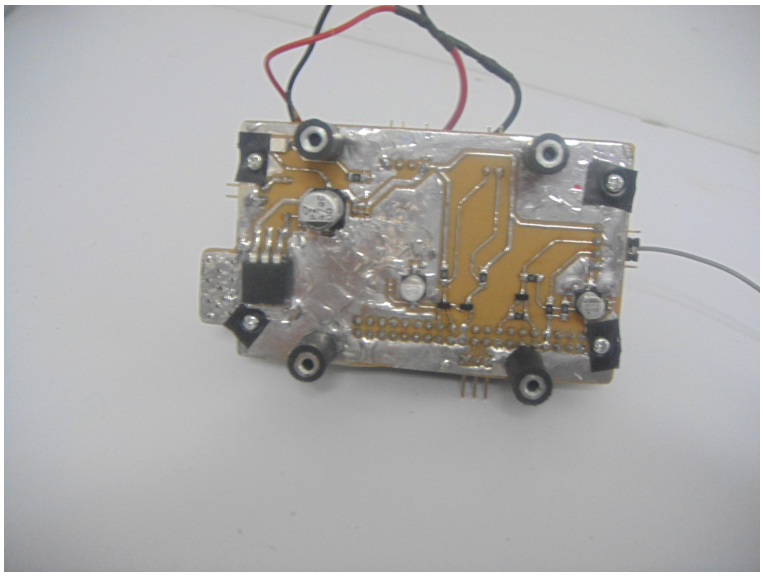


Figura 4.24: Placa del regulador-traductor de voltaje utilizada como base (vista inferior).

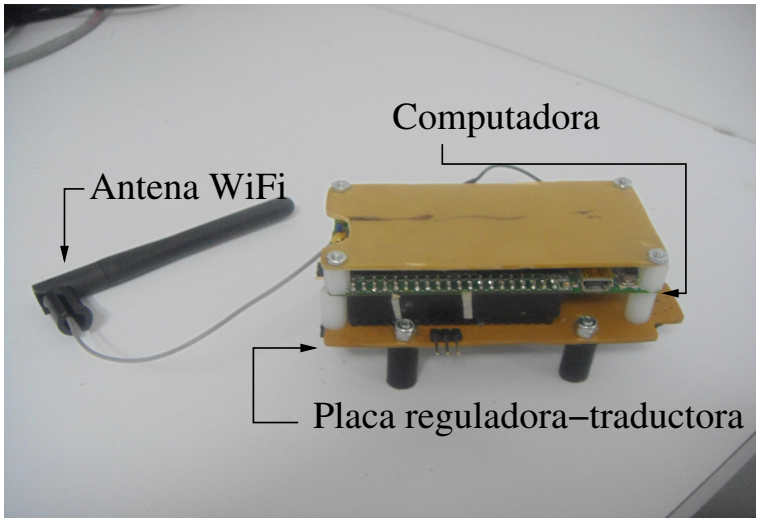


Figura 4.25: Computadora montada sobre placa reguladora-traductora de voltaje.

Estas tres placas, montadas una sobre la otra, se atornillan al centro del cuadricóptero como lo muestra la figura 4.26.

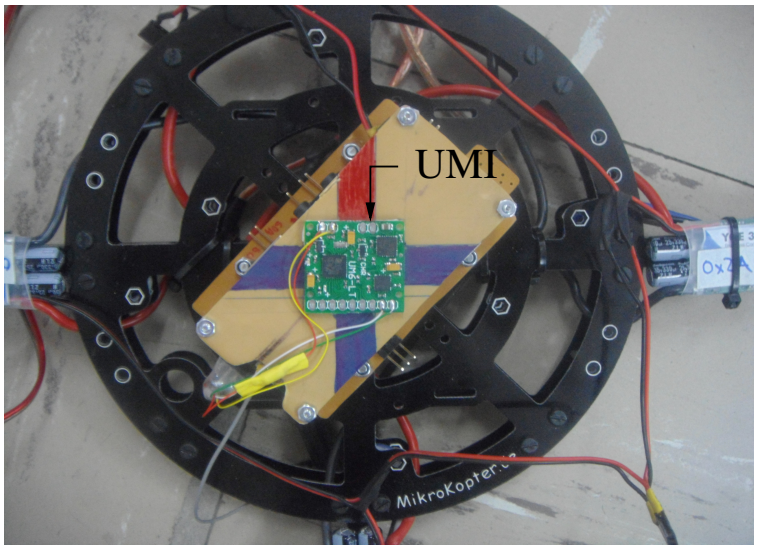


Figura 4.26: Placas montadas en el cuadricóptero.

La conexión del sensor ultrasónico se hace a través de un convertidor usb-serial, el cual está conectado a un concentrador USB, como lo muestra la figura 4.27.

Como medida de protección para las hélices del cuadricóptero y las personas, se construyó una carcasa de madera (figura 4.28). Ésta también sirve para facilitar la manipulación del vehículo en la etapa de pruebas.

Finalmente, la figura 4.29 muestra el cuadricóptero ya ensamblado, con la computadora y la unidad de medición inercial al centro del vehículo, el concentrador USB

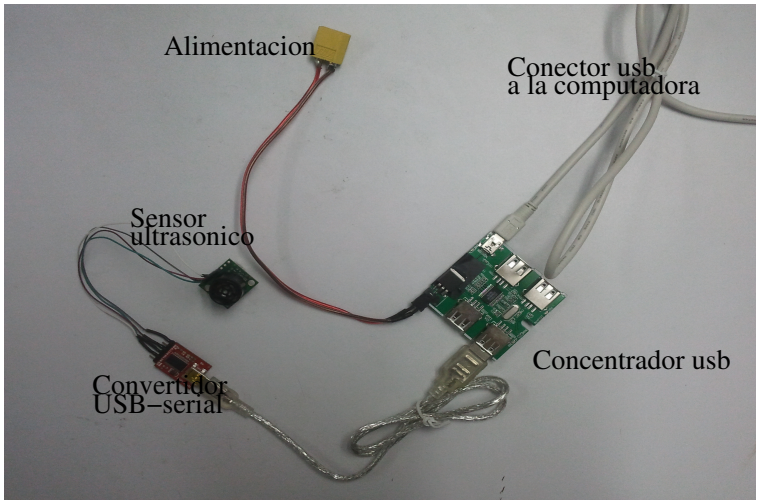


Figura 4.27: Conexión del sensor ultrasónico.

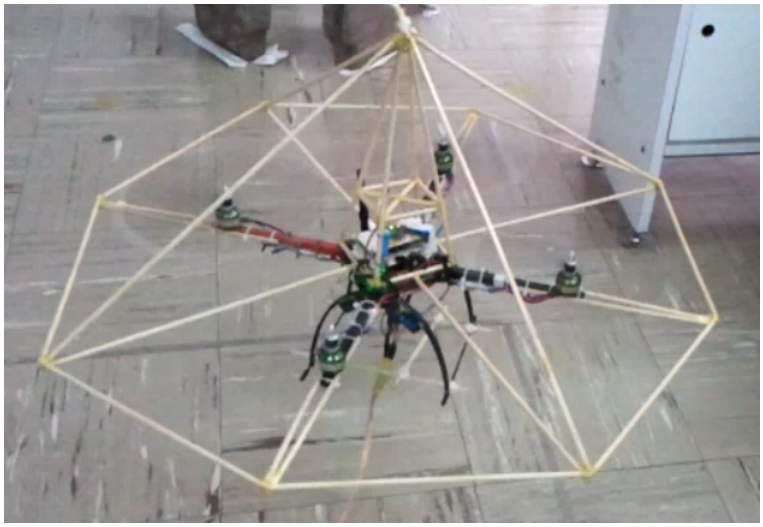


Figura 4.28: Cuadricóptero con carcasa de madera.

un poco más arriba sobre la carcasa de madera y el sensor ultrasónico en la base del vehículo. La batería es colocada justo debajo del vehículo, en el centro de masa de éste.

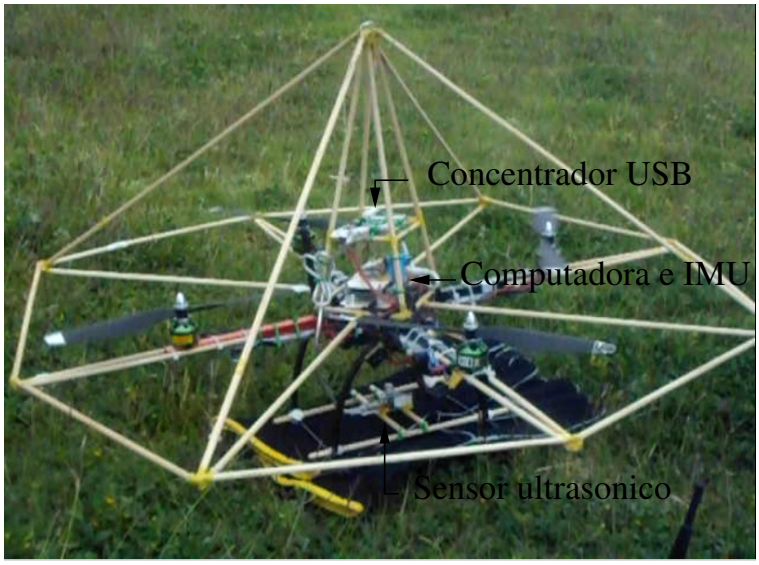


Figura 4.29: Cuadricóptero.

Capítulo 5

Implementación del *software*

En este capítulo se describirá el *software* desarrollado para el control y la configuración del cuadricóptero. Primero se dará una descripción de las plataformas de desarrollo utilizadas, como el sistema operativo Linux y el parche de tiempo real Xenomai, utilizados en la computadora a bordo del vehículo, así como también el sistema operativo Android. Luego se mostrará el sistema de control a bordo, el cual interactúa con el *software* de configuración de la computadora personal y con el control en tierra, realizado en una computadora en tableta. Finalmente, se describirá la implementación del algoritmo de evolución diferencial para el ajuste de los parámetros del controlador PID del vehículo.

5.1. Plataformas de desarrollo

Como se ha dicho en capítulos anteriores, el avance en la tecnología ha dado un gran impulso al desarrollo de vehículos aéreos no tripulados. Circuitos integrados más pequeños, más capaces y con bajo consumo de energía son fácilmente montables en vehículos pequeños, dotando a éstos con mayores recursos como memoria y capacidad de procesamiento.
Gracias a ésto, ha sido posible incluir sistemas operativos (SO) en diseños donde antes se programaba directamente en el procesador. El uso de SO en sistemas empotrados puede dar muchas ventajas con respecto a una programación directa en el procesador: abstracción de *hardware*, capacidad de multiprocesamiento y reutilización de *software*, entre otras.

5.1.1. Linux y tiempo real

Linux es un sistema operativo originalmente escrito por Linus Torvalds. El kernel de Linux provee una variedad de facilidades requeridas por cualquier sistema basado en linux para operar correctamente. El *software* de aplicación depende de las características específicas de este kernel, como el manejo de dispositivos de *hardware*,

así como también de una variedad de abstracciones fundamentales como la memoria virtual, tareas (o procesos), *sockets*, archivos, etc.
Este sistema operativo puede ser encontrado en una gran variedad de dispositivos, desde el más pequeño dispositivo de mano hasta en un *cluster* de supercómputo.
Algunos ejemplos del amplio rango de aplicaciones que utilizan Linux son: televisiones digitales, grabadoras, teléfonos celulares, servidores de sitios web como Google, etcétera [Yaghmour et al., 2008].
Algunas características importantes de linux son:

- Código abierto.
- Hay versiones que requieren poca memoria (2MB).
- Es un sistema maduro y estable.
- Gran variedad de software disponible.

El planificador de Linux está diseñado para tener la mejor respuesta promedio. Sin embargo, ésto no garantiza que alguna tarea en particular siempre corra dentro de un tiempo límite. Una tarea puede ser suspendida por un largo tiempo arbitrario, por ejemplo, cuando un manejador de dispositivos da servicio a una interrupción de disco.
Los sistemas operativos en tiempo real (en inglés: *RTOS* o *Real Time Operating Systems*) ofrecen garantías en el planificador de tareas. Sistemas como QNX, Lynxos o VxWorks son usados típicamente para aplicaciones de control o de comunicaciones, no para propósito general.
Linux ha sido adaptado para dar soporte en tiempo real. Estas adaptaciones han sido llamadas "Linux en tiempo real" (en inglés: *Real-Time Linux* o *RT Linux).*
Hay numerosas versiones de Linux en tiempo real disponibles, algunas libres, otras comerciales. Tres versiones comúnmente disponibles son:

- RTLinux. Desarrollado por New Mexico Tech y ahora mantenido por Wind River Systems.
- RTAI. Desarrollado por la Universidad Politécnica de Milán y disponible en https://www.rtai.org/.
- Xenomai. Es un derivado de RTAI. Mientras RTAI se enfoca en las más bajas latencias posibles, Xenomai además considera extensibilidad, portabilidad, y mantenibilidad como metas importantes. Está disponible en www.xenomai.org.

Debido a sus características, en este trabajo de tesis ha sido utilizado Xenomai.

5.1.2. Xenomai

El proyecto Xenomai fue lanzado en agosto de 2001. Éste está basado en un núcleo abstracto, utilizable para crear todo tipo de interfaces en tiempo real. Estas interfaces, llamadas "pieles" (en inglés: *skins*), se comunican de una manera específica con las aplicaciones usando los servicios del núcleo genérico.
Han sido implementadas las siguientes pieles:

- POSIX.
- pSOS+.
- VxWorks.
- VRTX.
- Nativa. La piel de Xenomai.
- uITRON.
- RTAI.

La idea general de Xenomai es que un pequeño kernel de tiempo real (TR) corra bajo Linux, lo cual significa que el kernel de TR tiene mayor prioridad que el kernel de Linux (figura 5.1). Las tareas en tiempo real son ejecutadas por el kernel de TR y los programas normales de Linux corren cuando no hay tareas en TR a ser ejecutadas.

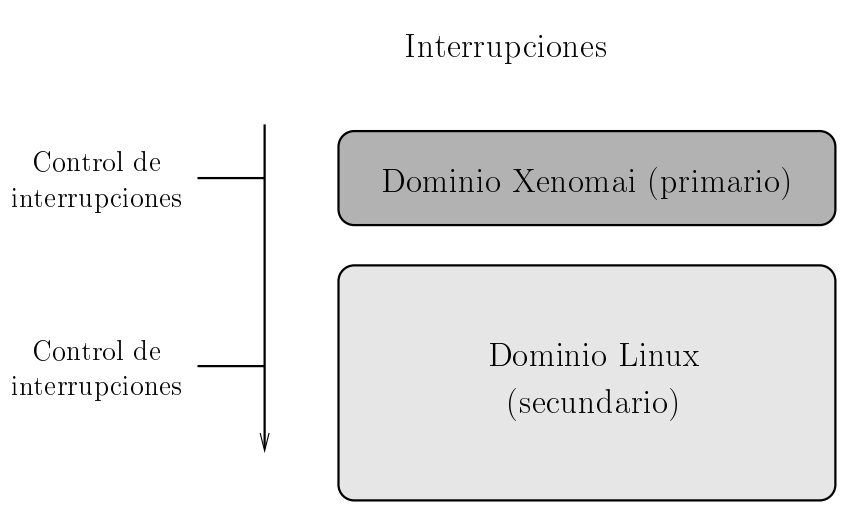


Figura 5.1: El kernel de tiempo real (primario) tiene mayor prioridad que el kernel de Linux (secundario).

Linux puede considerarse como una tarea inactiva del planificador de tiempo real. Cuando esta tarea inactiva corre, ejecuta su propio planificador y maneja los procesos

del Linux normal.
Como el kernel de tiempo real tiene mayor prioridad, una tarea de Linux es detenida cuando una tarea de TR está lista para correr, la cual es ejecutada inmediatamente.

Xenomai está escrito en C, por ello C es el lenguaje preferido para escribir programas de Linux en tiempo real.

5.1.3. Android

Android es una pila de *software* de código abierto, que incluye el sistema operativo, *middleware* y un conjunto de APIs.
Este sistema fue creado para correr en teléfonos celulares, pero ya se ha expandido su uso para computadoras en tableta.
Específicamente, Android está hecho de varias partes, incluyendo las siguientes:

- Un kernel de Linux que provee una interfaz de bajo nivel con el *Hardware*, manejo de memoria y control de procesos.
- Bibliotecas de código abierto para el desarrollo de aplicaciones, incluyendo SQLite, Webkit, OpenGL, etc.
- La máquina virtual Dalvik, utilizada para ejecutar las aplicaciones. Está diseñada para ser pequeña y eficiente para uso en dispositivos móviles.
- Un marco de trabajo que ofrece servicios a la capa de aplicación, incluyendo el manejador de ventanas, bases de datos, telefonía y sensores.
- Un conjunto de aplicaciones preinstaladas.
- Un kit de desarrollo de *software* (en inglés: *SDK* ó *Software Development Kit* usado para crear aplicaciones, incluyendo las herramientas relacionadas y documentación [Meier, 2012].

Las aplicaciones en Android se escriben habitualmente en lenguaje Java, aunque están disponibles otras herramientas de desarrollo para aplicaciones en C o C++.
Una ventaja que tiene Android sobre otros sistemas operativos como iOS o Windows phone, es que posee plataformas de desarrollo gratuitas y no se requieren licencias para desarrollar *software*. Además, sus APIs permiten un acceso total al *hardware*, un requerimiento importante para este trabajo.

5.2. Control y configuración del cuadricóptero

Para el control y configuración del cuadricóptero se desarrollaron tres aplicaciones. La primera, escrita en C, es la aplicación a bordo del vehículo. Ésta se encarga del control automático del mismo y además atiende las peticiones de los clientes, utilizando

hilos en tiempo real. La segunda fue escrita en Python y Qt para computadoras con Linux. Es una interfaz de configuración de las constantes del controlador, así como de lectura de los sensores a bordo. Finalmente, se desarrolló una aplicación en Java para computadora en tableta que funciona como control manual en tierra.
A continuación se describirán estas aplicaciones con más detalle.

5.2.1. Controlador a bordo

Esta aplicación se compone básicamente de cuatro hilos: el hilo principal, el hilo de control de postura, el hilo de control de altura y el hilo servidor (uno o más). Estos cuatro hilos interactúan a través de la memoria compartida, la cual contiene las principales variables requeridas por todos ellos, como las mediciones de los sensores, los valores de referencia del controlador, el modo de control, etcétera (figura 5.2). Para evitar la corrupción de los datos compartidos se utiliza la exclusión mutua.

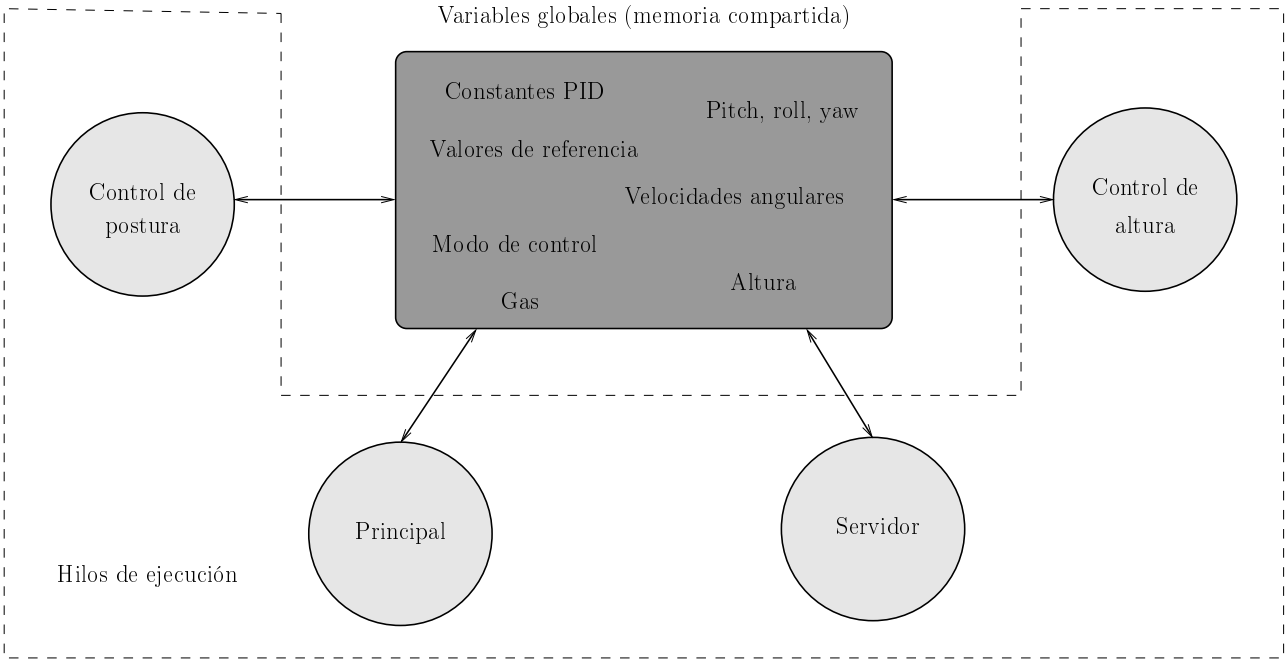


Figura 5.2: Hilos básicos del *software* a bordo del vehículo y memoria compartida.

Hilo principal

Este hilo, que corre en el dominio de Linux, es el punto de entrada de la aplicación. En él se inicializa el sistema y se crean los otros hilos.
La figura 5.3 muestra el diagrama de flujo. El primer paso es inicializar la unidad de medición inercial, donde se calibran los giroscópios con velocidad inicial de cero. En esta etapa el vehículo debe permanecer inmóvil por al menos tres segundos.

Después de ésto, se inicializan los motores, uno a uno, en modo manual con un valor de 20 (de un rango de 0 a 255). Ésto se hace para lograr un arranque más suave cuando se incremente la velocidad.
A continuación, se crean y corren los hilos de control de postura y altura, pero no empezarán a trabajar, ya que el sistema aún se encuentra en modo manual.
A partir de ese momento, el hilo principal se mantiene esperando por peticiones de conexión de clientes. Para cada petición se crea un hilo servidor que se encarga de ejecutar los comandos requeridos por el cliente.

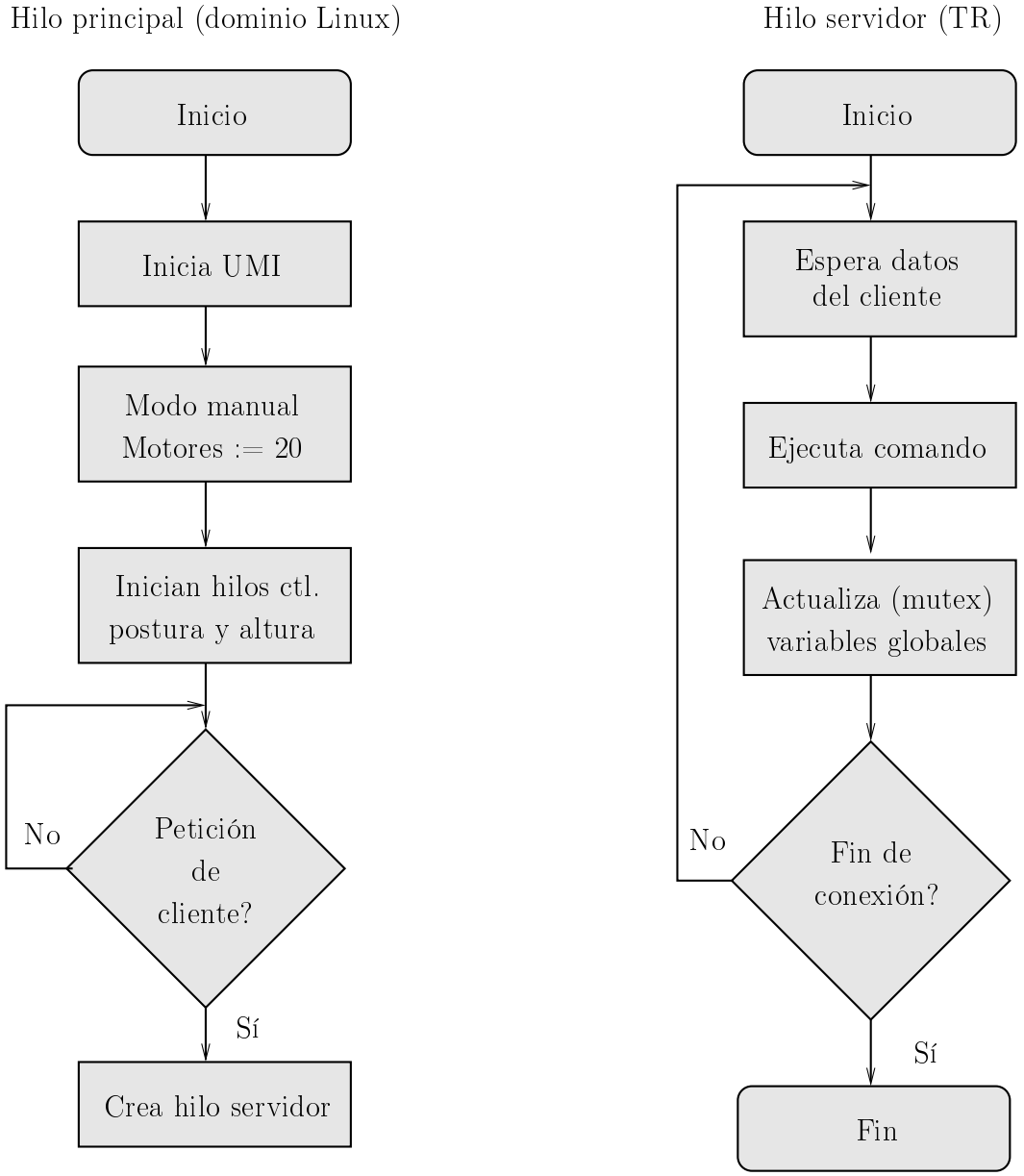


Figura 5.3: Hilo principal e hilo servidor de la aplicación de control a bordo.

Hilo servidor

Este hilo (o hilos), con la más baja prioridad de los hilos en tiempo real, es el encargado de atender los comandos enviados por las aplicaciones clientes, ya sea la interfaz de configuración de la computadora o el control en tierra de la computadora en tableta.
El cliente y el servidor se comunican a través de *sockets* con el protocolo *TCP/IP*.
Lo primero que hace este hilo al ser creado, es esperar el envío de un paquete de datos del cliente. Un paquete de datos, como lo muestra la figura 5.4, consta de tres partes: un byte que contiene el número de bytes enviados por el cliente, un byte que contiene el comando a ejecutar y una cadena de bytes (opcional) que contiene los parámetros relacionados con el comando.

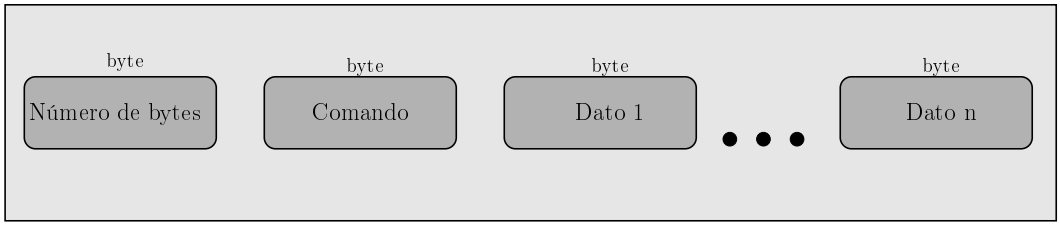


Figura 5.4: Estructura de los paquetes de datos enviados entre el servidor y el cliente.

Después de recibir el paquete de datos enviado por el cliente, se ejecuta el comando contenido, para después responder al cliente con otro paquete de datos, ya sea con los valores requeridos por el cliente o sólo regresando el mismo comando recibido como un aviso de que ya se ha ejecutado.
La tabla 5.1 muestra el conjunto de comandos que puden ser ejecutados por el servidor.

Al terminar de ejecutar el comando, el hilo servidor actualiza las variables globales que hayan cambiado para que los demás hilos puedan disponer de la nueva información. Este ciclo se realiza hasta que el cliente se desconecte de la red y el hilo se destruya.

Hilo de control de postura

Este hilo en tiempo real tiene la prioridad de interrupción más alta. Es el encargado de calcular las variables de control de los ángulos de *pitch*, *roll* y *yaw*, y el único que puede manipular las salidas a los motores después de la inicialización en el hilo principal. Básicamente, este hilo consiste en un ciclo de lectura de sensores, cálculo de variables de control y salida a motores que se repite cada 10 milisegundos. La Figura 5.5 muestra el diagrama de flujo del hilo de control de postura. Como primer paso, se leen los ángulos y velocidades actuales de la UMI, los cuales son actualizados en la memoria compartida para su uso en otros hilos. De la memoria compartida también se leen los valores de referencia y constantes a utilizar por los controladores PID, así como el modo de operación del vehículo.

Código	Comando
97	Enviar últimos valores leídos de la IMU al cliente
109	Cambiar velocidades de los motores 1,2,3 y 4
49	Cambiar velocidades de los motores 1 y 3
50	Cambiar velocidades de los motores 2 y 4
19	Enviar al cliente el valor de la variable *Gas*
20	Cambiar *Gas* a valor especificado por el cliente
21	Cambiar valores de k_P, k_I y k_D del control de *pitch*
22	Cambiar valores de k_P, k_I y k_D del control de *roll*
23	Cambiar valores de k_P, k_I y k_D del control de *yaw*
24	Cambiar valores de referencia de los PID de *pitch*, *roll*, *yaw*
25	Guardar en archivo constantes PID actuales
26	Enviar constantes PID guardadas en archivo al cliente
27	Enviar constantes PID en ram al cliente
28	Cambiar valores de k_P, k_I y k_D del control de altura
30	Cambiar modo de control
40	Enviar último valor leído de sensor ultrasónico al cliente
41	Cambiar valor de referencia de control PID de altura

Tabla 5.1: Lista de comandos ejecutables por el hilo servidor.

Si el modo de operación es manual, los valores de los motores son ajustados directamente por el usuario a través de la memoria compartida. Si el modo de operación es automático, se lee la variable de control de altura calculada en otro hilo (si el modo autómatico de control de altura está activado) y se ejecutan los algoritmos de control PID descritos en el capítulo 3. De esta forma se determinan las variables de control de los ángulos *pitch*, *roll* y *yaw*.
Finalmente, las aportaciones de las variables de control son enviadas a cada motor y se esperan 10 milisegundos para empezar el ciclo de nuevo.

Hilo de control de altura

Este hilo de tiempo real tiene la segunda prioridad de interrupción más alta. Es el encargado de calcular la variable de control de altura, que es leída por el hilo de control de postura para sumarla a la salida de los motores. La estructura de este hilo es muy semejante a la del de control de postura, siendo un ciclo de lectura, control y salida que se repite cada 100 milisegundos.
Como se ve en la figura 5.5, primero se lee el sensor ultrasónico para determinar la altura actual del vehículo. Este valor se pasa por un filtro pasabajas de segundo orden para eliminar el ruido en las lecturas. El valor resultante se pasa luego a la memoria compartida para su uso en otros hilos. En este paso se leen también los valores de referencia, las constantes del controlador PID y el modo de control a utilizar.
Si el control automático de altura está activado, se calcula la variable de control

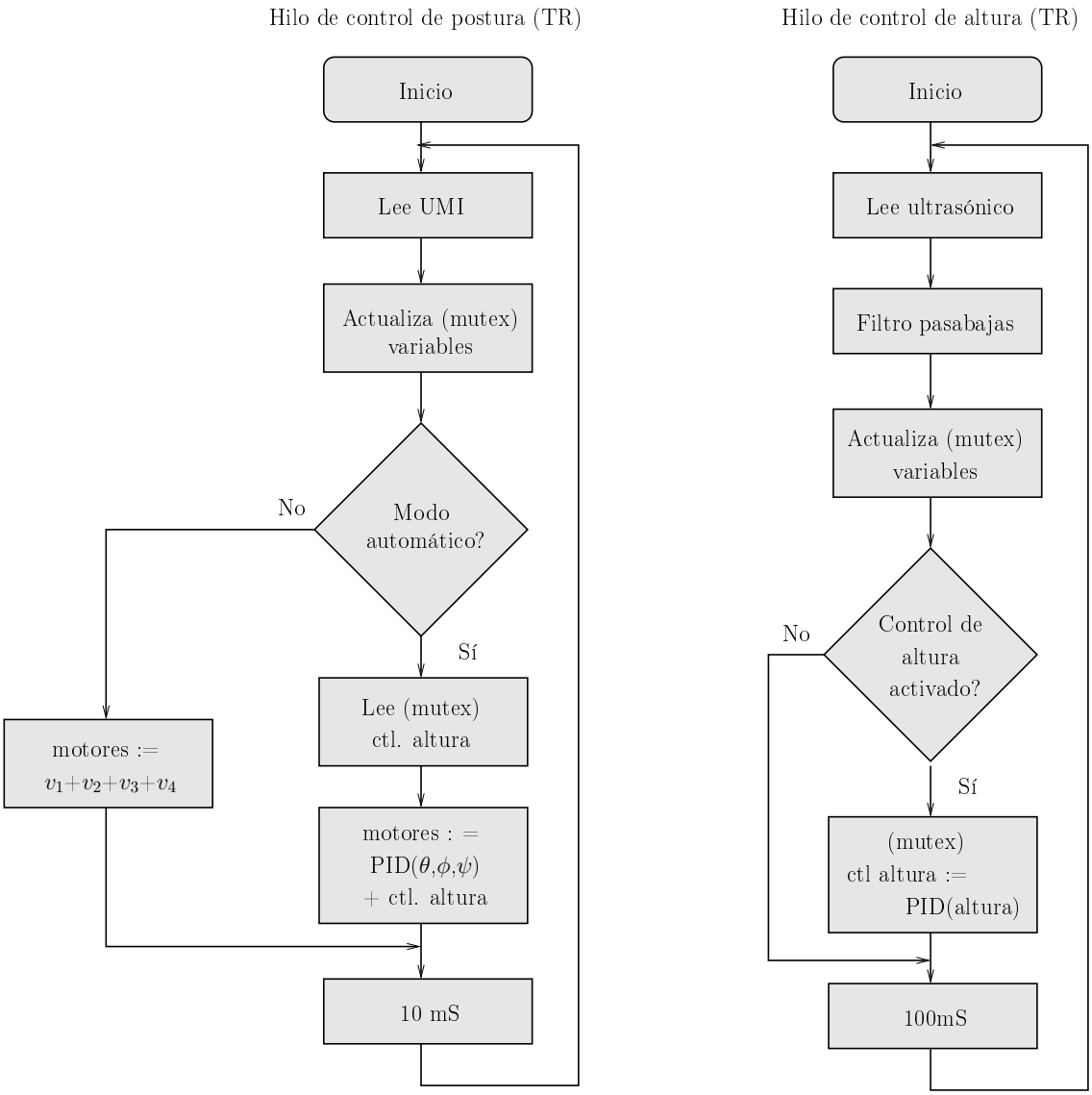


Figura 5.5: Hilos de control de postura y altura de la aplicación de control a bordo.

de altura utilizando el algoritmo de control PID descrito en el capítulo 3. Una vez obtenida esta variable, es actualizada en la memoria compartida para que pueda ser aplicada sobre los motores en el hilo de control de postura. Si el modo de control es manual, no se hace ningún cálculo. Este ciclo continúa cada 100 milisegundos.

5.2.2. Aplicación de configuración en computadora

Esta aplicación fue escrita en Python y se utilizó Qt para crear la interfaz gráfica de usuario. Funciona como cliente y sirve para configurar las constantes de los controladores PID que conforman el sistema a bordo, así como también para probar estas

configuraciones en el laboratorio.
La figura 5.6 muestra la interfaz de usuario de la aplicación. En la esquina superior derecha están los botones de conexión y desconexión con el servidor; y de encendido y apagado del vehículo. Un poco más abajo se pueden ver los botones para leer y guardar las constantes del controlador PID a bordo. En la misma columna se encuentra un cuadro para configurar la variable *Gas* y un botón de paro de emergencia.
A la izquierda se encuentran los cuadros de edición de los valores de las constantes de los controladores. Éstos se pueden activar o desactivar de manera individual para hacer pruebas en alguno en particular.
Finalmente, al centro se encuentran las gráficas de los valores obtenidos de la unidad de medición inercial y del sensor ultrasónico, las cuales son actualizadas cada 10 milisegundos. Todos los valores leídos se guardan en un archivo de texto para su estudio posterior.

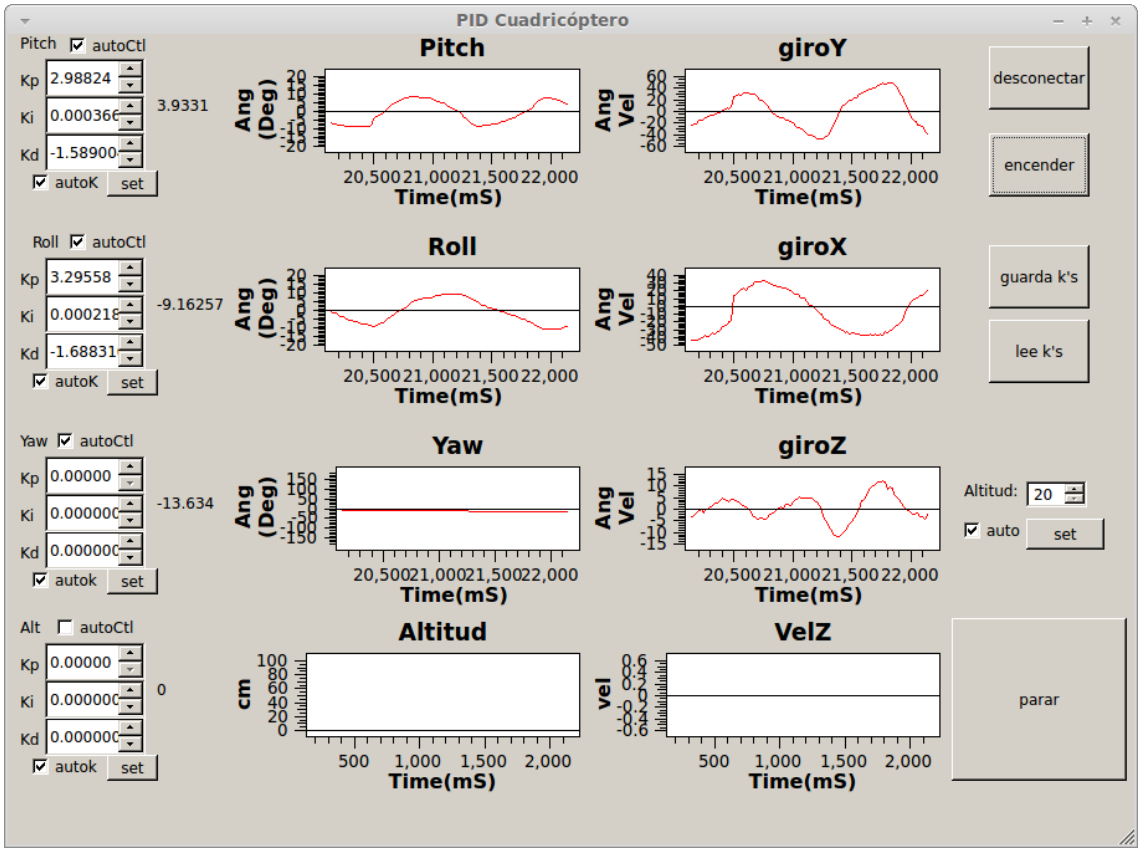


Figura 5.6: Interfaz de usuario de la aplicación de configuración.

5.2.3. Control en tierra con computadora en tableta

Para el control en tierra del vehículo se utilizó una computadora en tableta. Ésta tiene grandes ventajas con respecto a un ordenador convencional, como lo son: peso

más ligero, mayor movilidad y un conjunto de sensores que pueden enriquecer la forma de interactuar con el usuario. Esta aplicación aprovecha los acelerómetros, giroscópios y magnetómetros incluidos para crear un control más intuitivo para el usuario.
El control se basa en cambios de posición del dispositivo, el cual tiene un sistema de coordenadas fijo igual al sistema de coordenadas del cuadricóptero. Como se puede ver en la figura 5.7 un cambio en los ángulos de la tableta equivale a un cambio proporcional en los ángulos de referencia en el controlador del vehículo, provocando un movimiento semejante al de la tableta. Podría decirse que el cuadricóptero "imita" la posición de la tableta.

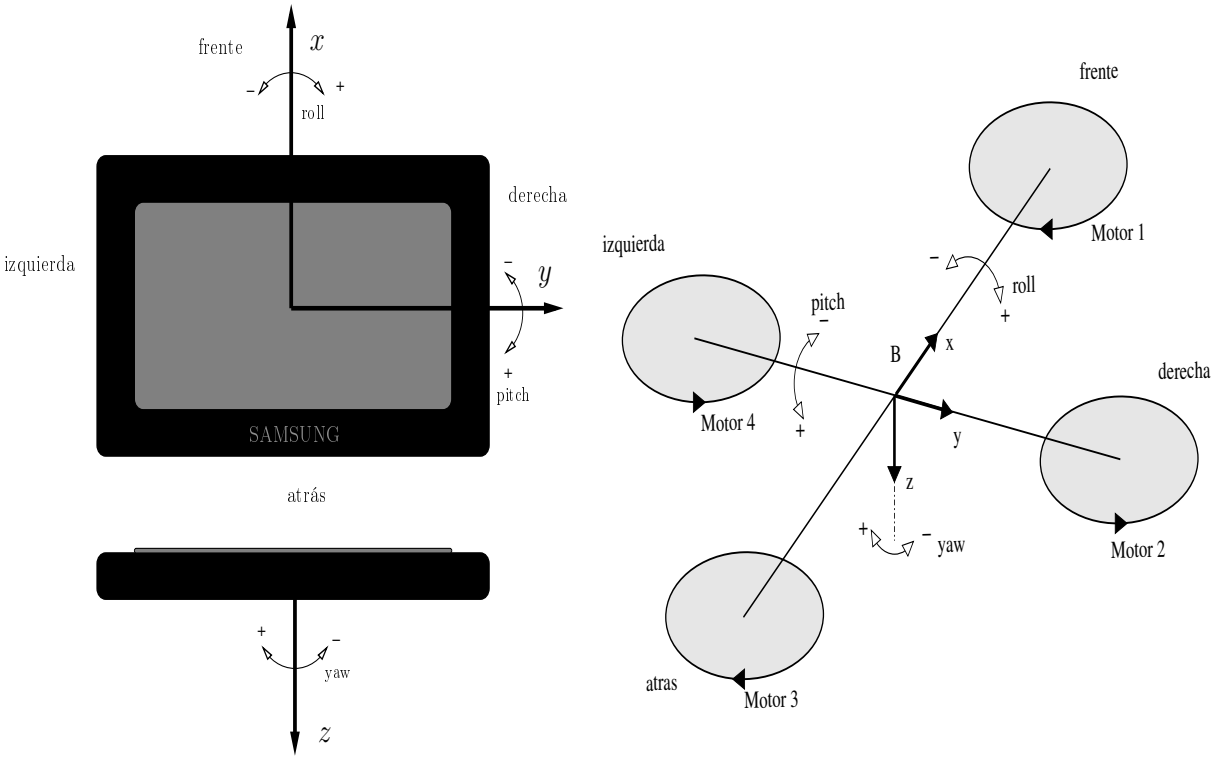


Figura 5.7: Sistema de coordenadas de tableta y vehículo.

Por cada medio grado que se mueva la tableta en los ángulos *pitch* y *roll*, el cuadricóptero tendrá un cambio de un grado, esto es: $\text{pitchVehículo} = (0.5 \cdot \text{pitchtableta})$. La relación de grados del movimiento de *yaw* de la tableta con respecto al *yaw* del vehículo es de uno a uno, por cada grado que rote la tableta, el cuadricóptero rotará un grado. Por seguridad, se pueden hacer giros máximos en *pitch* y *roll* de ± 10 grados

La figura 5.8 muestra la interfaz de usuario de la aplicación. Es una interfaz sencilla, ya que la mayoría del control se hace con el movimiento.
Al centro de la pantalla se pueden ver los valores actuales de las mediciones de los sensores a bordo del vehículo. También se encuentra el botón de apagado y encendido

de los motores.

Figura 5.8: Interfaz de la aplicación de control en tierra.

A la izquierda de la pantalla se encuentran dos barras, las cuales sirven para ajustar el valor de referencia del control de altura y la variable *Gas*. El botón "blockalt" sirve para habilitar o inhabilitar las barras. Ésto es útil para evitar hacer cambios accidentales en esos valores.
El botón "autoAlt" activa o desactiva el control automático de altura. Si está desactivado, el control manual de altura se hace modificando los valores en la barra de *Gas*.
A la derecha de la pantalla se encuentran los botones "pitchRoll" y "yaw". Cuando el botón "pitchRoll" está activado, la rotación en los ejes x y y de la tableta modifican los valores de referencia de *pitch* y *roll* del vehículo. Cuando está desactivado, los valores de referencia se ajustan a cero grados y el movimiento de la tableta no afecta al cuadricóptero. Lo mismo sucede con el botón "yaw"que sólamente modifica los valores de referencia cuando está activado.

5.3. Ajuste de parámetros del PID con evolución diferencial

Para el ajuste de parámetros del controlador PID se implementó el algoritmo de evolución diferencial en la computadora a bordo del vehículo, pero además se desa-

rrollo una aplicación cliente en Python y Qt para la captura de datos, supervisión y control del estado de la evolución.
Como se mencionó en el capítulo 3, el problema de ajuste de parámetros se separó en dos: ajuste de parámetros de altura y ajuste de parámetros de postura. La estructura de los dos programas es la misma, con la diferencia de que los parámetros de entrada son de diferente longitud, se afecta a diferentes controladores PID, la función de desempeño de los individuos es distinta y el criterio de paro es de menos generaciones en el control de altura.
A continuación se describirá más a detalle el desarrollo de estas aplicaciones.

5.3.1. Evolución diferencial en computadora a bordo

Esta fue escrita en C y es la encargada de encontrar los parámetros adecuados para un correcto funcionamiento de los controladores PID.
Como se detalló en el capítulo 3, el algoritmo de evolución diferencial se basa en tener una población de individuos, que por medio de mutación y recombinación se crean nuevas generaciones mejores que las anteriores, todo esto con base en una función de desempeño.
Los individuos de postura y altura se definen de la siguiente manera:

$$postura : \mathbf{X} = [k_{P_\theta}, k_{I_\theta}, k_{D_\theta}, k_{P_\phi}, k_{I_\phi}, k_{D_\phi}, k_{P_\psi}, k_{I_\psi}, k_{D_\psi}] \in \mathbb{R}^9, \tag{5.1}$$

$$altura : \mathbf{X} = [k_{P_z}, k_{I_z}, k_{D_z}] \in \mathbb{R}^3, \tag{5.2}$$

donde:
k_P en θ, ϕ, ψ, z son las constantes proporcionales de los controladores de *pitch*, *roll*, *yaw* y altura.
k_I en θ, ϕ, ψ, z son las constantes integrales de los controladores de *pitch*, *roll*, *yaw* y altura.
k_D en θ, ϕ, ψ, z son las constantes derivativas de los controladores de *pitch*, *roll*, *yaw* y altura.

Los límites superior e inferior de sus posibles valores, se encontraron experimentalmente, utilizando el programa de configuración en la computadora.
Tomando estos vectores y usando una población de 30 individuos se puede iniciar el proceso de evolución diferencial.
Las figuras 5.9, 5.10 y 5.11 muestran el diagrama de flujo de la aplicación. En este diagrama no aparecen, pero también existen los hilos de control de postura, altura y servidor descritos en la aplicación de control a bordo. El sistema de memoria compartida también es el mismo.

El programa comienza en el hilo principal (figura 5.9), el cual espera una petición

de conexión del cliente. Al conectarse el cliente (programa de control de evolución diferencial de la computadora), la aplicación espera una señal de inicio para comenzar el algoritmo de evolución diferencial.
El primer paso es inicializar la población. La figura 5.11 muestra el diagrama de flujo de esta función. Un individuo es generado aleatoriamente, dentro de los límites especificados. Después de eso se actualizan los valores obtenidos (número de individuo, generación, parámetros) en la memoria compartida para que el cliente pueda leer los datos. Una vez que los datos son leídos por el cliente, se evalúa el individuo generado para determinar su aptitud. Esta variable también se pasa a la memoria compartida y el algoritmo se detiene de nuevo hasta que el cliente la lea. Todo ésto se repite hasta formar la primera población de treinta individuos.
A continuación, es necesario crear otra serie de individuos temporales para que compitan con aquellos recién creados. Primero se eligen al azar tres padres de la población actual, el individuo temporal es generado mediante los procesos de mutación y recombinación explicados anteriormente. Los valores obtenidos son actualizados en la memoria temporal y se espera a que el cliente los lea. Después, se obtiene la aptitud del individuo temporal por medio de la función de desempeño y una vez más, espera a que el cliente la lea.
El nuevo individuo, de la siguiente generación, se obtiene comparando las aptitudes de los individuos temporal y de la generación actual. El que tenga la mejor aptitud (valor más cercano a cero) es el que domina y gana la nueva posición.
Al terminar de hacer esta selección con toda la población, se determina el mejor y peor individuo de la generación, para luego esperar a que el cliente lea estas variables. Este proceso se hace para doscientas generaciones, que es cuando el algoritmo se detiene y toma como resultado al mejor individuo obtenido.
La forma de evaluación de un individuo se muestra en el diagrama de flujo de la figura 5.11. Esta función toma al individuo como parámetro, esté individuo (constantes proporcional, integral y derivativa) se carga al controlador PID del vehículo. El vehículo se enciende por 10 segundos, mientras la señal de error de los ángulos o de altura (según sea evolución de postura o altura) es muestreada y guardada en un vector. Al terminar de hacer el muestreo, el controlador del vehículo se apaga y se determina la aptitud del individuo según la función de desempeño, que toma como parámetros las señales de los errores.

Recordando el capítulo 3, la función de desempeño para la evolución diferencial de postura es:

$$f(\mathbf{X}) = \frac{1}{3}\sum_{i=0}^{59} |e_\theta(t_i)\cdot 0.0166i| + |e_\phi(t_i)\cdot 0.0166i| + |e_\psi(t_i)\cdot 0.0166i| + \frac{1}{3}\sum_{i=60}^{999} |e_\theta(t_i)| + |e_\phi(t_i)| + |e_\psi(t_i)|, \tag{5.3}$$

y para la evolución diferencial de altura:

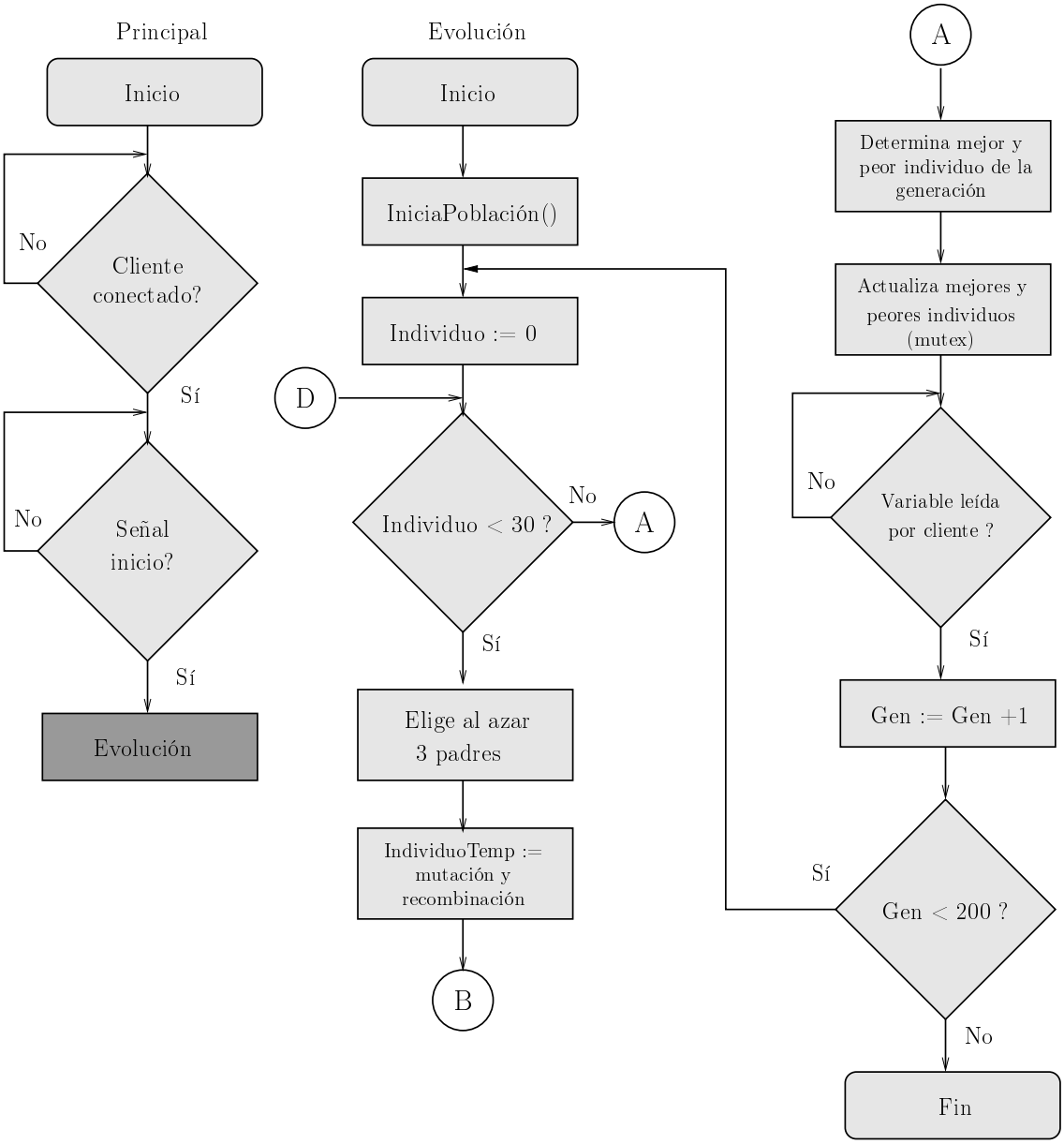


Figura 5.9: Diagrama de flujo de ED en computadora a bordo, parte 1.

$$f(\mathbf{X}) = \sum_{i=0}^{9} |e_z(t_i) \cdot 0.1i| + \sum_{i=10}^{99} |e_z(t_i)|. \tag{5.4}$$

donde: e_θ, e_ϕ, e_ψ y e_z son los errores de la posición del vehículo con respecto a los valores de referencia en el instante de tiempo i.

Las evaluaciones de los individuos se hacen con el vehículo montado en las estructuras de prueba descritas en el capítulo 4. En la figura 5.12 se puede ver, a la izquierda, la estructura para hacer las pruebas de la evolución de postura y, a la derecha, la

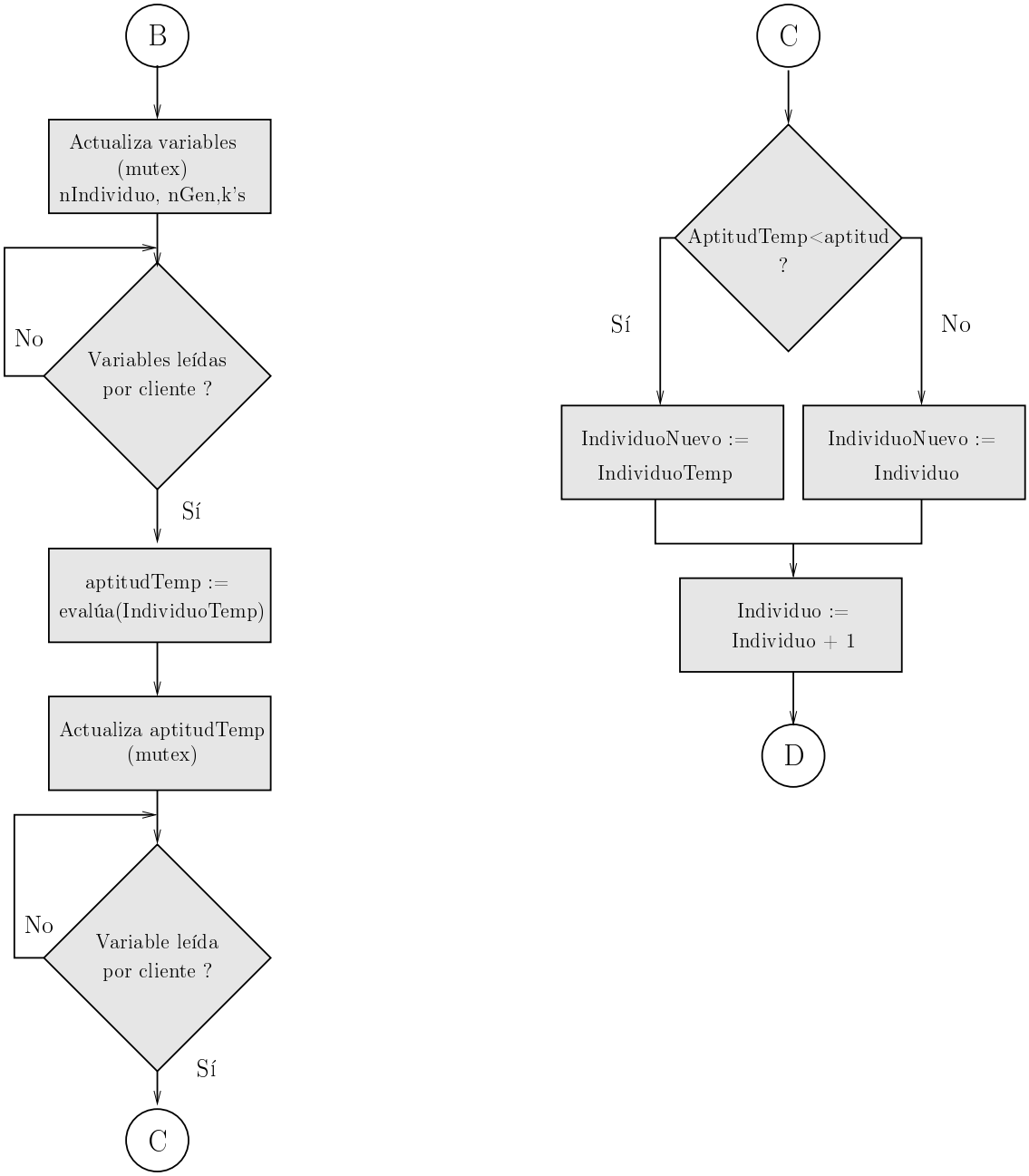


Figura 5.10: Diagrama de flujo de ED en computadora a bordo, parte 2.

estructura de pruebas de la evolución de altura.

5.3.2. Control de evolución diferencial en computadora

Esta aplicación se desarrolló con el fin de tener una interfaz de control y seguimiento del algoritmo de evolución diferencial ejecutado en la computadora a bordo. A través

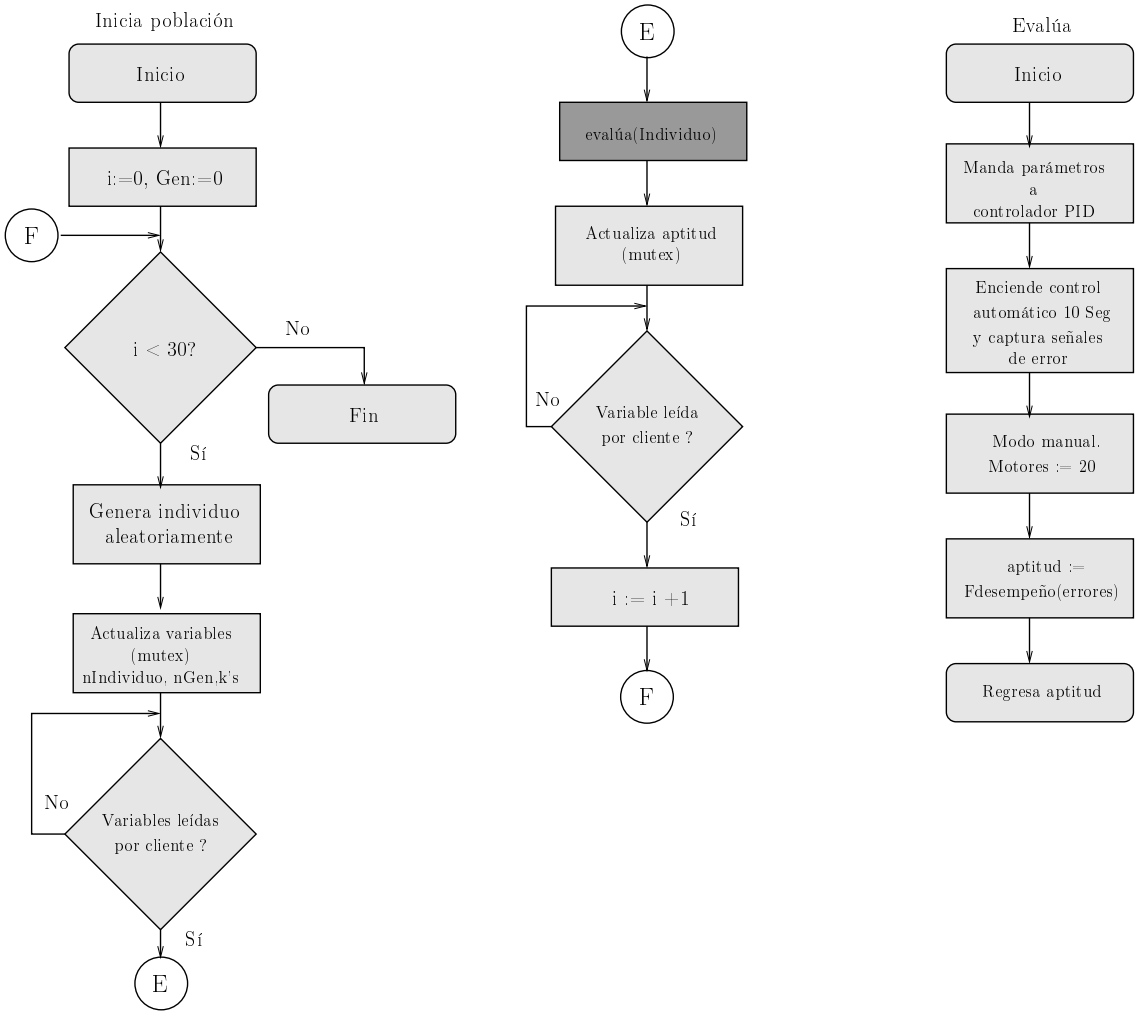


Figura 5.11: Diagrama de flujo de ED en computadora a bordo, parte 3.

de ella se puede iniciar o detener el proceso, así como visualizar y guardar los datos generados.

La figura 5.13 muestra la interfaz gráfica para la evolución de postura. En la esquina superior izquierda de la ventana se muestran los datos del individuo actual: el número de individuo, la generación y los parámetros que se están evaluando. Al término de cada evaluación se muestra la aptitud obtenida.

En la misma columna, un poco más abajo, se encuentran los botones de inicio (>) y pausa (||) de la evolución. Estos son necesarios para detener y reiniciar la evaluación del vehículo en caso de que sea necesario.

El resto de la interfaz consta de dos gráficas: al fondo está la gráfica de las señales de errores en *pitch*, *roll* y *yaw*, que va actualizando los datos según se van produciendo en el cuadricóptero. En la parte de arriba está la gráfica de las evaluaciones, donde se van graficando las aptitudes de los mejores y peores individuos de cada generación, así como también la mediana. Está gráfica sirve para conocer el avance de la evolución

Figura 5.12: Estructuras para las pruebas de evolución diferencial de postura y altura del cuadricóptero.

diferencial y determinar si el algoritmo está convergiendo.

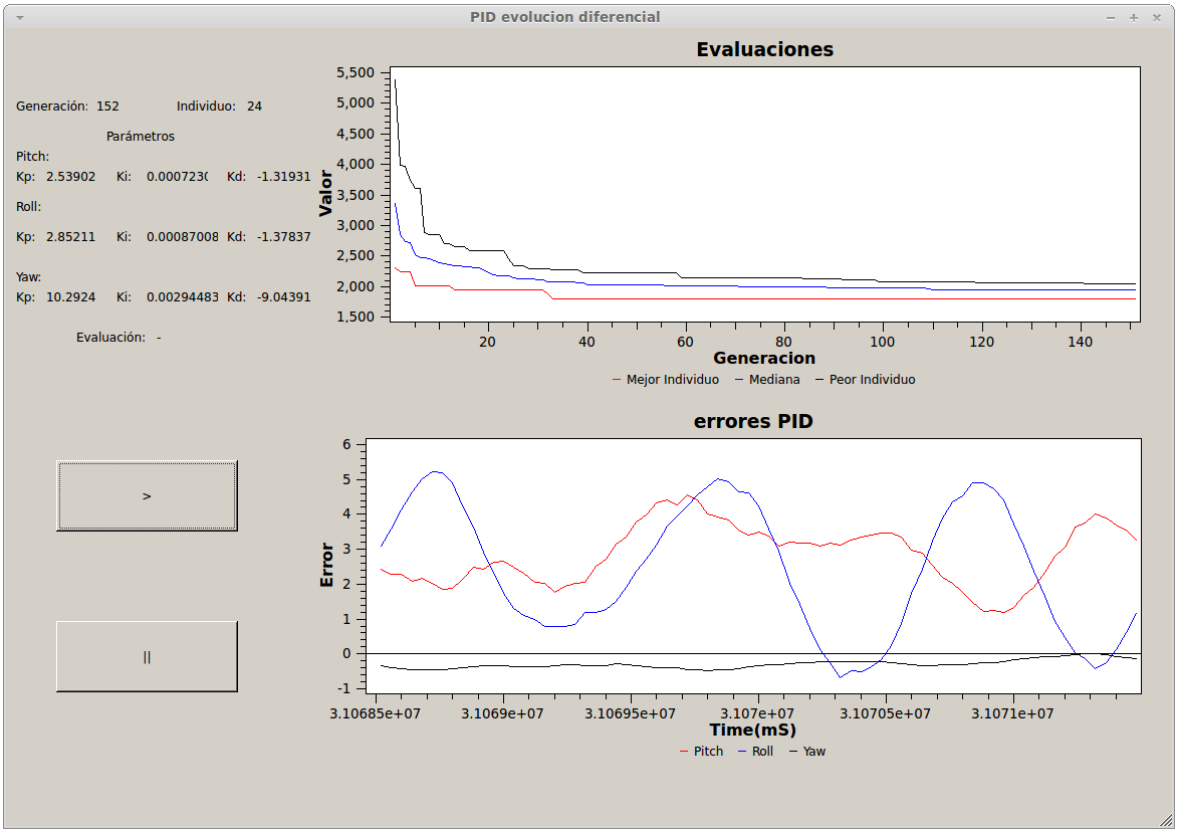


Figura 5.13: Interfaz de usuario de la aplicación de evolución diferencial en la computadora.

Esta aplicación además genera cuatro archivos de texto donde guarda los datos

generados por el control a bordo del vehículo. El primer archivo guarda los parámetros de todos los individuos evaluados, hayan sido seleccionados o no. En otro par de archivos, se guardan los parámetros de los mejores y peores individuos de cada generación. En el cuarto archivo se guardan los valores de las aptitudes de los mejores y peores individuos, además de la mediana de cada generación.

La figura 5.14 muestra el diagrama de flujo de la aplicación. Al iniciar el programa, hace la conexión con el servidor (cuadricóptero) y envía la señal de inicio para que comience la evolución diferencial. Después, lee los parámetros del primer individuo a evaluar (número de individuo, generación y constantes PID), envía la señal de recibido al servidor para que comience la evaluación y espera a que ésta termine de ejecutarse.
En el mismo tiempo de espera, lee las señales de error que van siendo generadas por el vehículo durante la evaluación y se grafican en la interfaz de usuario.
Al terminar la evaluación, hace una pausa de 1.5 segundos para que el vehículo detenga su movimiento y lee la aptitud resultante del individuo evaluado. Ya teniendo los datos completos del individuo, éstos se guardan en un archivo de texto para su posterior procesamiento.
En el siguiente paso, verifica si el usuario ha presionado el botón de pausa. Si lo hizo, el programa se mantiene esperando a que el botón de reinicio sea presionado, manteniendo al cuadricóptero detenido. Si no lo hizo, el programa continúa su ejecución normal.
Este ciclo de lectura-evaluación-lectura de individuos se hace hasta completar una generación, que es cuando se guardan en los otros tres archivos los mejores y peores individuos; así como las aptitudes peores, mejores y la mediana de de la generación.
El programa continúa hasta que se haya satisfecho la condición de paro, que es la evaluación de doscientas generaciones de individuos.

5.3.3. Resultados de la ejecución de la evolución diferencial

Después de varias corridas del algoritmo de evolución diferencial para el control de postura y de altura, nos pudimos dar cuenta de que en pocas generaciones ya se obtiene un individuo suficientemente bueno. Aunque ésto puede tardar algunas horas debido a que cada evaluación tarda 10 segundos.
La figura 5.15 muestra la gráfica de las aptitudes de los mejores y peores individuos de la evolución de postura, así como la mediana de cada generación evaluada.
Se puede notar que, en las primeras generaciones, los valores de las aptitudes de los individuos son muy diferentes, pero conforme se va realizando la evolución estos empiezan a converger a un sólo valor. A partir de la generación 90, los valores mejor, peor y mediana se mantienen estables hasta que llega la condición de paro (200 generaciones).
La evaluación de las doscientas generaciones de individuos tomó alrededor de 19 horas, aunque encontrar un individuo suficientemente bueno tomó menos de la mitad

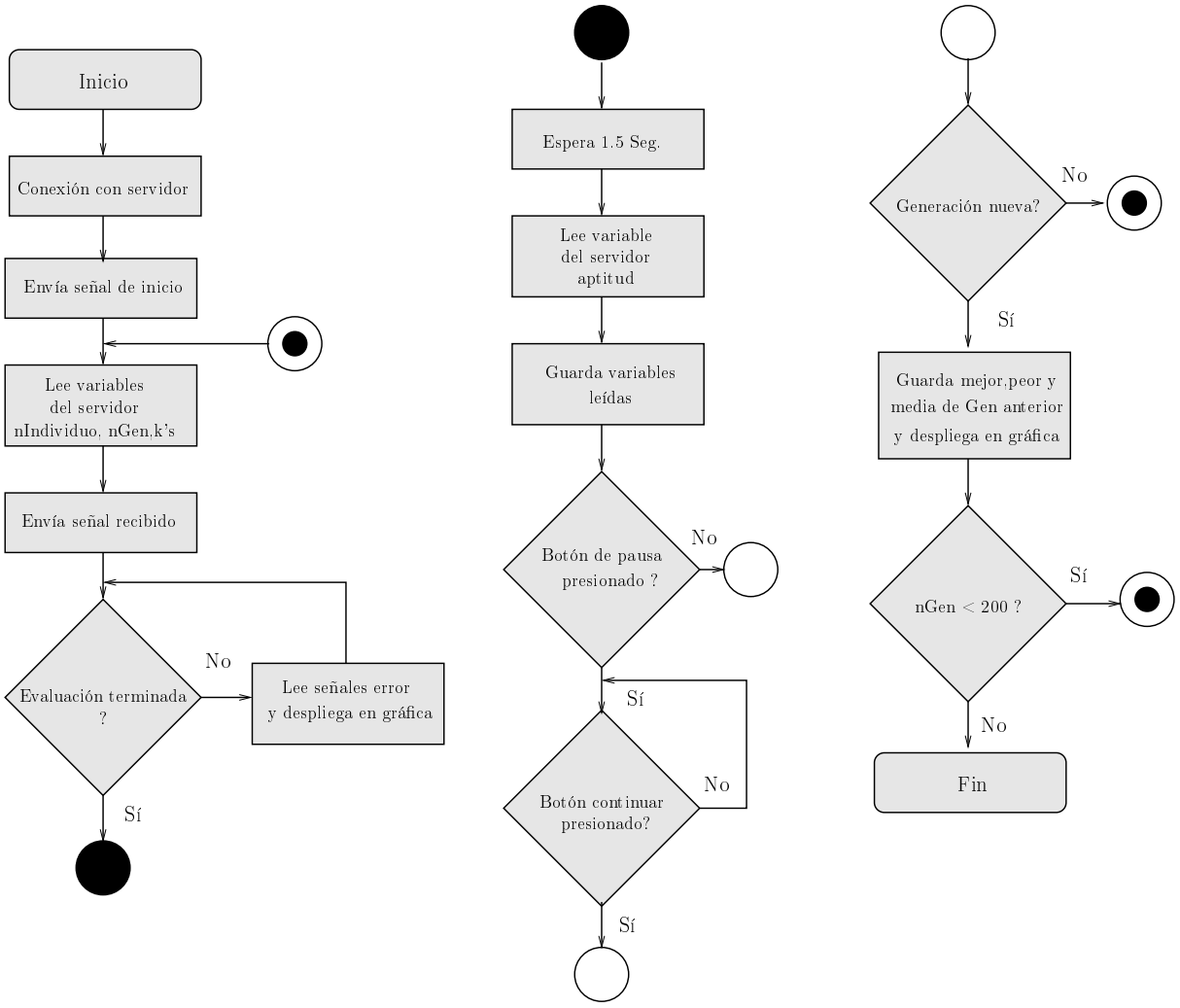


Figura 5.14: Diagrama de flujo de la aplicación de control de ED en computadora.

del tiempo.

La figura 5.16 muestra la gráfica de aptitudes de los individuos de control de altura. Para la ejecución de este algoritmo se requiere de supervisión humana, por lo que sólo se hicieron evaluaciones de 39 generaciones. Sin embargo, puede verse una estabilización de las aptitudes a partir de la generación 28.
El tiempo de evaluación de las 39 generaciones fue de alrededor de 4 horas.

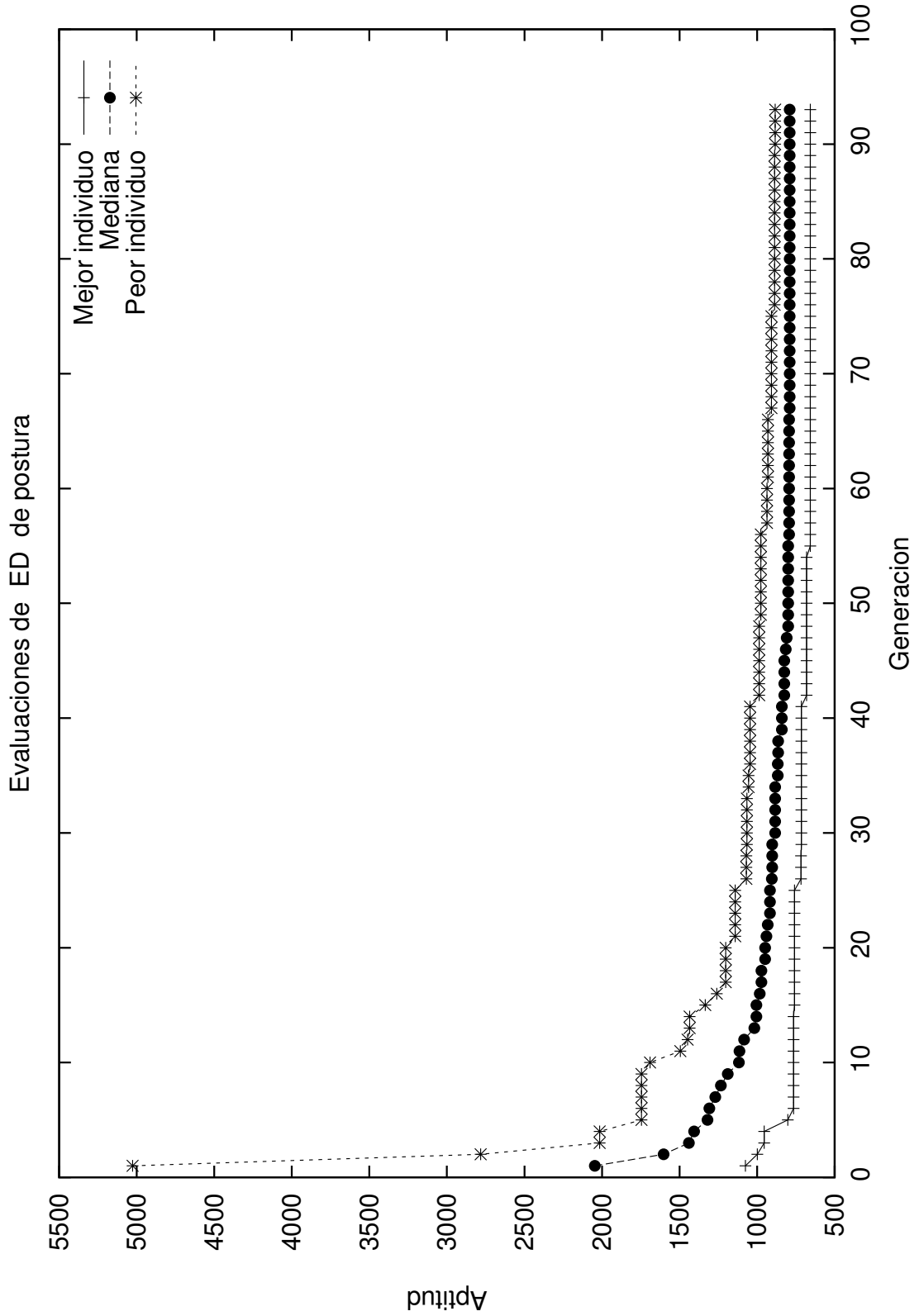


Figura 5.15: Gráfica de evaluaciones de individuos de control de postura.

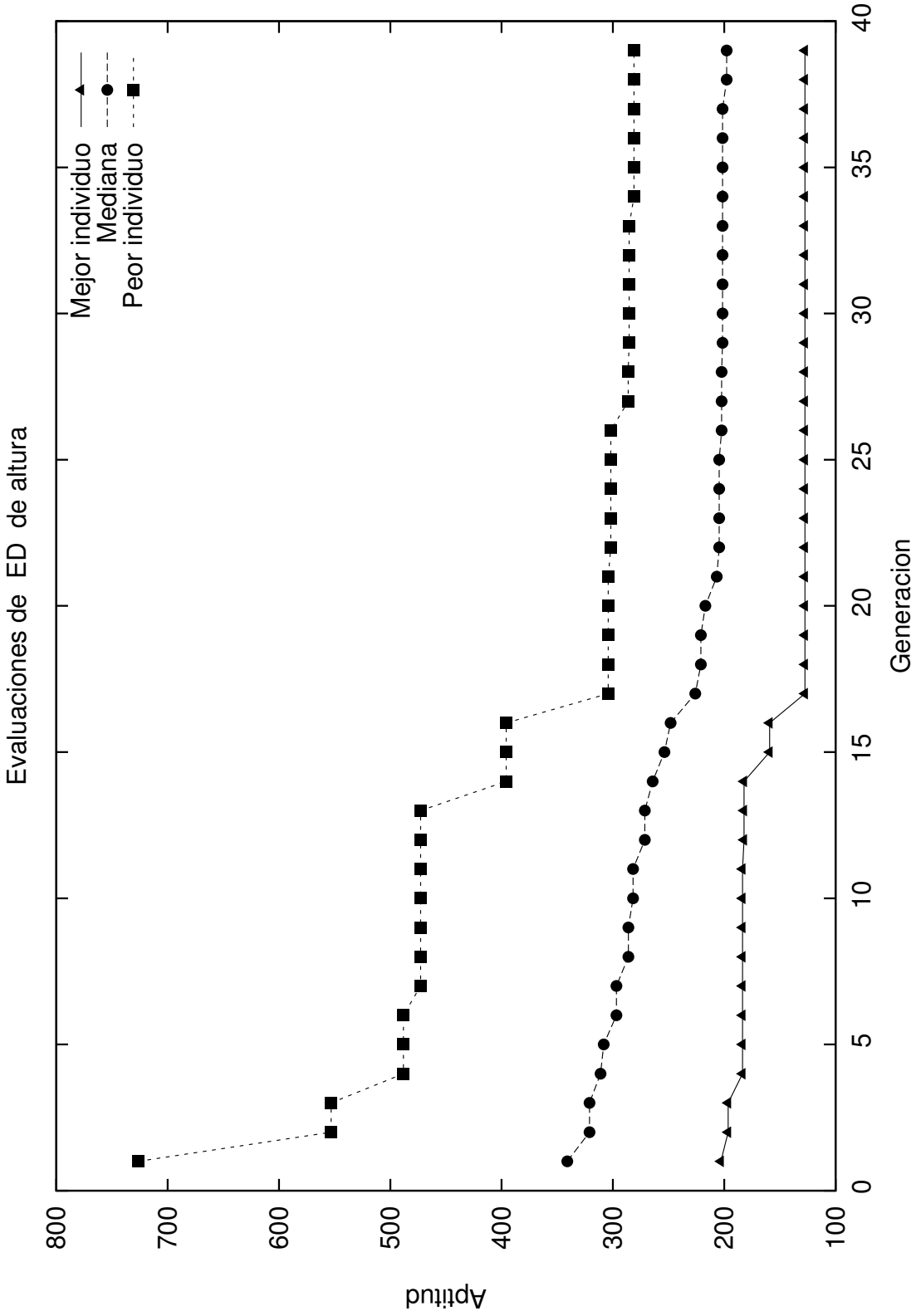


Figura 5.16: Gráfica de evaluaciones de individuos de control de altura.

Capítulo 6

Conclusiones y trabajo a futuro

En este trabajo de tesis se diseñó e implementó un sistema de control para un cuadricóptero. Se diseño el control automático de postura y altura, basándose en controladores PID, utilizando evolución diferencial para el ajuste de sus parámetros. El sistema a bordo fue implementado en una placa computadora con el sistema operativo Linux y Xenomai, mientras que el control en tierra se hizo con una computadora en tableta con Android.
En general, los resultados obtenidos fueron buenos, cumpliendo con los objetivos propuestos. A continuación, se expondrán las conclusiones generadas en diferentes partes del proyecto, así como el trabajo a futuro.

6.1. Conclusiones

Linux embebido

El uso de una computadora a bordo con Linux embebido fue de gran ayuda en el desarrollo del proyecto, ya que además de contar con bastantes recursos, como memoria RAM y velocidad de procesamiento, se tienen las ventajas inherentes del sistema operativo como: manejo de varios procesos a la vez, herramientas de red, manejo de archivos, abstracción entre el *hardware* y el *software* de aplicación, etcétera. Ésto llevó al desarrollo de programas más portables, lo que facilitará la actualización del *hardware* cuando sea necesario.
Los contras que se pudieron encontrar, son que se requiere más esfuerzo, tiempo y conocimiento para poner el sistema en funcionamiento, que utilizando un microcontrolador convencional, además de que un *software* complejo es más propenso a contener errores, algo que puede ser crítico al usar vehículos aéreos.

Control automático y evolución diferencial

El ajuste de parámetros de los controladores PID con evolución diferencial dio buenos resultados, especialmente en el control de postura. El control de altura es bueno cuando el cuadricóptero está en postura horizontal, pero llega a tener muchas variaciones con respecto a su valor de referencia cuando se inclina el vehículo. Ésto es debido a que el sensor ultrasónico también se inclina y mide diferentes distancias. Aún así, el control fue suficientemente bueno para manipular el vehículo con facilidad.
Otro problema con el control de altura es que, mientras la carga de la batería va disminuyendo se empieza a perder altura, por lo que es necesario modificar la variable *Gas* manualmente para aumentar la potencia.

Computadora en tableta como control en tierra y comunicaciones

El uso de una computadora en tableta como control en tierra demostró ser muy eficaz. La forma y tamaño de estos dispositivos los hace fáciles de manipular en ambientes exteriores, por lo que pueden ser transportadas a zonas de difícil acceso.
Además, su rica variedad en recursos de *hardware* (pantalla táctil, acelerómetros, giroscópios, magnetómetros, comunicación inalámbrica, etc.), ayuda a crear aplicaciones mucho más intuitivas, haciendo el control del vehículo más sencillo que utilizando controles remotos convencionales.
El uso de *Wi-Fi* como medio de comunicación entre la tableta y el cuadricóptero fue ventajoso, debido a que ambas computadoras cuentan con adaptadores de red inalámbrica integrados. Sin embargo, el alcance de este tipo de redes es limitado, por lo que para aplicaciones donde se requiera una navegación en espacios mayores deberá utilizarse otro sistema.

Tolerancia a fallos

Un error del sistema cuando el vehículo está en pleno vuelo usualmente lleva a accidentes, por lo que éste debe estar diseñado para recuperarse de los problemas que puedan surgir.
El sistema diseñado incluyó un botón de paro de emergencia y en etapas de prueba en laboratorio se diseño *hardware* específico para esta tarea. Sin embargo, aún hay mucho trabajo por hacer para mejorar la respuesta a fallos.
Durante la realización de este proyecto ocurrieron dos accidentes, uno con la pérdida total del vehículo. Es por eso que en investigaciones futuras se pondrá especial énfasis en este tema.

6.2. Trabajo a futuro

A continuación, se mencionan algunas propuestas de trabajo a futuro para que el sistema pueda ser utilizado en problemas del mundo real.

- Hacer pruebas con controladores no lineales. Los controladores PID dieron buenos resultados en el control del cuadricóptero pero, debido a que se pretende usarlo en exteriores, un control no lineal podría ser más apropiado para tener una mayor robustez contra cambios en el clima, como ráfagas de viento. Un sistema de control neuro-difuso podría resultar ventajoso.

- Transmisores-receptores inalámbricos con mayor alcance. Se pretende utilizar los módulos inalámbricos *Xbee-pro*, que tienen rangos de alcance de hasta 1 kilómetro. Para eso se deberá diseñar una interfaz para acoplar estos dispositivos con la computadora en tableta.

- Mejorar el control de altura. Agregar otros tipos de sensores como barómetros y láser ayudará a controlar el vehículo a mayores alturas.

- Planificación de trayectorias. Lograr la planeación de trayectorias requerirá del control automático de la posición, por lo que habrá que agregar un GPS al sistema, entre otros sensores para la evasión de obstáculos.

- Despegue y aterrizaje automáticos.

- Sistema tolerante a fallos. Se requerirán ajustes en *software* y *hardware* para lograr una mayor tolerancia a fallos.

- Visión por computadora. Aplicar algoritmos de visión y procesamiento de imágenes puede ayudar a mejorar el desempeño del vehículo, así como también puede ser útil para el sistema de instrumentación.

Bibliografía

[Al-Younes and Jarrah, 2008] Al-Younes, Y. and Jarrah, M., *Attitude stabilization of quadrotor uav using backstepping fuzzy logic & backstepping least-mean-square controllers*, in Mechatronics and Its Applications. ISMA 2008. 5th International Symposium on, IEEE, pp. 1–11, Amman, Jordan, May 2008.

[Austin, 2010] Austin, R., *Unmanned Air Systems: UAV Design, Development and Deployment*, Wiley, Hoboken, NJ, USA, 2010.

[Bai et al., 2011] Bai, H., Shao, S. and Wang, H., *A vtol quadrotor platform for multi-uav path planning*, in Electronic and Mechanical Engineering and Information Technology (EMEIT), 2011 International Conference on, IEEE, pp. 3079–3081, Heilongjiang, China, August 2011.

[Barnhart et al., 2011] Barnhart, R., Shappee, E., Marshall, D. et al., *Introduction to Unmanned Aircraft Systems*, CRC Press, Boca Raton, FL, USA, 2011.

[Berni et al., 2009] Berni, J., Zarco-Tejada, P., Suárez, L. and Fereres, E., *Thermal and narrowband multispectral remote sensing for vegetation monitoring from an unmanned aerial vehicle*, in Geoscience and Remote Sensing, IEEE Transactions on, IEEE, pp. 722–738, Ann Arbor, MI, USA, March 2009.

[Bouhali and Boudjedir, 2011] Bouhali, O. and Boudjedir, H., *Neural network control with neuro-sliding mode observer applied to quadrotor helicopter*, in Proc. Int Innovations in Intelligent Systems and Applications (INISTA) Symp, IEEE, pp. 24–28, Istanbul, Turkey, June 2011.

[Castillo et al., 2005] Castillo, P., Lozano, R. and Dzul, A., *Modelling and control of mini-flying machines*, Springer-Verlag New York Inc, New York, NY, USA, 2005.

[DoD, 2002] DoD, *Unmanned Aerial Vehicles Roadmap 2002 - 2027*, Oficce of the Under Secretary of Defense for Acquisition Technology and Logistics, Washington DC, USA, 2002.

[Donahoo and Calvert, 2009] Donahoo, M. and Calvert, K., *TCP/IP sockets in C: practical guide for programmers*, Morgan Kaufmann Pub, Burlington, MA, USA, 2009.

[Eiben and Smith, 2008] Eiben, A. and Smith, J., *Introduction to evolutionary computing*, springer, New York, NY, USA, 2008.

[Fowers et al., 2007] Fowers, S., Lee, D., Tippetts, B., Lillywhite, K., Dennis, A. and Archibald, J., *Vision aided stabilization and the development of a quad-rotor micro uav*, in Computational Intelligence in Robotics and Automation. CIRA 2007. International Symposium on, IEEE, pp. 143–148, Jacksonville, FL, USA, June 2007.

[Ghadiok et al., 2011] Ghadiok, V., Goldin, J. and Ren, W., *Autonomous indoor aerial gripping using a quadrotor*, in Intelligent Robots and Systems (IROS). IEEE/RSJ International Conference on, IEEE, pp. 4645–4651, San Francisco, CA, USA, September 2011.

[Goela et al., 2009] Goela, R., Shahb, S., Guptac, N. and Ananthkrishnanc, N., *Modeling, simulation and flight testing of an autonomous quadrotor*, in Proceedings of the IISc Centenary International Conference and Exhibition on Aerospace Engineering, ICEAE, pp. 18–22, Bangalore, India, May 2009.

[Hafner et al., 2010] Hafner, V., Bachmann, F., Berthold, O., Schulz, M. and Müller, M., *An autonomous flying robot for testing bio-inspired navigation strategies*, in Robotics (ISR), 2010 41st International Symposium on and 2010 6th German Conference on Robotics (ROBOTIK), VDE, pp. 1–7, Munich, Germany, June 2010.

[Hallinan, 2010] Hallinan, C., *Embedded Linux primer: a practical, real-world approach*, Pearson Education, Boston, MA, USA, 2010.

[Hong et al., 2005] Hong, W., Lee, J., Rai, L. and Kang, S., *Rt-linux based hard real-time software architecture for unmanned autonomous helicopters*, in Embedded and Real-Time Computing Systems and Applications. Proceedings. 11th IEEE International Conference on, IEEE, pp. 555–558, Hong Kong, China, August 2005.

[Johnson et al., 2005] Johnson, M., Moradi, M. and Crowe, J., *PID control: new identification and design methods*, Springer Verlag, New York, NY, USA, 2005.

[Kira, 2009] Kira, Z., *Transferring embodied concepts between perceptually heterogeneous robots*, in Intelligent Robots and Systems. IROS 2009. IEEE/RSJ International Conference on, IEEE, pp. 4650–4656, St. Louis, MO, USA, October 2009.

[Kirli et al., 2010] Kirli, A., Omurlu, V., Buyuksahin, U., Artar, R. and Ortak, E., *Self tuning fuzzy pd application on ti tms320f28335 for an experimental stationary quadrotor*, in Education and Research Conference (EDERC), 2010 4th European, IEEE, pp. 42–46, Nice, France, December 2010.

[Kuo, 2003] Kuo, B.C., *Automatic control systems*, Wiley, Hoboken, NJ, USA, 2003.

[Lendek et al., 2011] Lendek, Z., Berna, A., Guzmán-Giménez, J., Sala, A. and García, P., *Application of takagi-sugeno observers for state estimation in a quadrotor*, in Decision and Control and European Control Conference (CDC-ECC). 50th IEEE Conference on, IEEE, pp. 7530 – 7535, Orlando, FL, USA, December 2011.

[Li and Li, 2011] Li, J. and Li, Y., *Dynamic analysis and pid control for a quadrotor*, in Proc. Int Mechatronics and Automation (ICMA) Conf, IEEE, pp. 573–578, Beijing, China, August 2011.

[Maranhao and Alsina, 2009] Maranhao, D. and Alsina, P., *Project of a hardware and software architecture for an unmanned aerial vehicle*, in Robotics Symposium (LARS), 2009 6th Latin American, IEEE, pp. 1–6, Valparaiso, Chile, October 2009.

[Meier, 2012] Meier, R., *Professional Android 4 application development*, Wrox, Indianapolis, IN, USA, 2012.

[Naidoo et al., 2011] Naidoo, Y., Stopforth, R. and Bright, G., *Development of an uav for search & rescue applications*, in AFRICON, 2011, IEEE, pp. 1–6, Livingstone, September 2011.

[Nonami et al., 2010] Nonami, K., Kendoul, F., Suzuki, S., Wang, W. and Nakazawa, D., *Autonomous Flying Robots: Unmanned Aerial Vehicles and Micro Aerial Vehicles*, Springer, New York, NY, USA, 2010.

[Ogata, 1997] Ogata, K., *Modern control engineering*, Prentice Hall, Upper Saddle River, NJ, USA, 1997.

[Price et al., 1997] Price, K., Storn, R. and Lampinen, J., *Differential evolution*, Springer, New York, NY, USA, 1997.

[Puls et al., 2009] Puls, T., Kemper, M., Kuke, R. and Hein, A., *Gps-based position control and waypoint navigation system for quadrocopters*, in Intelligent Robots and Systems, 2009. IROS 2009. IEEE/RSJ International Conference on, IEEE, pp. 3374–3379, St. Louis, MO, USA, October 2009.

[Raharijaona and Bateman, 2011] Raharijaona, T. and Bateman, F., *Robust control for an off-centered quadrotor*, in Control & Automation (MED), 2011 19th Mediterranean Conference on, IEEE, pp. 1253–1258, Corfu, Greece, June 2011.

[Raza and Gueaieb, 2010] Raza, S.A. and Gueaieb, W., *Intelligent flight control of an autonomous quadrotor*, in Motion Control, pp. 245–264, IN-TECH, Rijeka, Croatia, 2010.

[Santos et al., 2010] Santos, M., López, V. and Morata, F., *Intelligent fuzzy controller of a quadrotor*, in Intelligent Systems and Knowledge Engineering (ISKE), 2010 International Conference on, IEEE, pp. 141–146, Hangzhou, China, November 2010.

[Ulbrich et al., 2011] Ulbrich, P., Kapitza, R., Harkort, C., Schmid, R. and Schröder-Preikschat, W., *I4copter: An adaptable and modular quadrotor platform*, in Proceedings of the 2011 ACM Symposium on Applied Computing, ACM, pp. 380–386, TaiChung, Taiwan, March 2011.

[Visioli, 2006] Visioli, A., *Practical PID control*, Springer Verlag, New York, NY, USA, 2006.

[Wilson, 2005] Wilson, J., *Sensor technology handbook*, Elsevier, Oxford, UK, 2005.

[Xiang and Tian, 2011] Xiang, H. and Tian, L., *Development of a low-cost agricultural remote sensing system based on an autonomous unmanned aerial vehicle (uav)*, in Biosystems Engineering, Elsevier, pp. 174–190, Champaign, USA, February 2011.

[Yaghmour et al., 2008] Yaghmour, K., Masters, J., Ben-Yossef, G. and Gerum, P., *Building embedded Linux systems*, O'Reilly Media, Sebastopol, CA, USA, 2008.