Diseño, construcción y control de un cuadricóptero– parte 2: software, estimación y control

MICHELL P EHRLICH, LUIS F LUPIÁN

***Resumen*—Se presenta el proceso completo de programación de la computadora de control de vuelo para un cuadricóptero pequeño con un diámetro de 275 mm con fines académicos y de investigación dentro del Laboratorio de Robótica Móvil y Sistemas Automatizados de Universidad La Salle. El trabajo presentado hace énfasis en los algoritmos de estimación de variables, así como en la arquitectura y algoritmos de control implementados. Se presentan pruebas experimentales del resultado obtenido con los algoritmos de estimación y control. Se detallan además los módulos de recepción, interpretación y generación de señales de control. En su estado actual, el cuadricóptero es capaz de volar y autoestabilizarse de manera satisfactoria.**

I. Introducción

Un cuadricóptero es una herramienta de investigación y desarrollo versátil y divertida que presenta oportunidades de investigación que van más allá de conseguir que levante vuelo y se autoestabilize, si no que puede ser utilizada para probar y desarrollar otras tecnologías como algoritmos de navegación por reconocimiento de patrones, implementar distintos tipos de sensores para que el aparato pueda reaccionar a cambios en su ambiente, entre otros. Los objetivos fundamentales de este proyecto son desarrollar (a) el hardware y construcción física del cuadricóptero, y (b) el software y sistema de control de vuelo. Debido a que se buscaba sobre todo aprender cómo desarrollar un cuadricóptero se puso como meta adicional el utilizar el mínimo posible de partes prefabricadas.

En la última década, los cuadricópteros o cuadrirotores han alcanzado gran popularidad tanto por parte del público en general como de la comunidad científica. Actualmente, los cuadrirotores se usan en algunas instituciones como caso de estudio en cursos introductorios de la ingeniería con enfoque multidisciplinario [1]. Desde el punto de vista científico, hay varios problemas interesantes que se pueden investigar como el diseño y control [2], [3], [4], [5] y la estimación de pose en tiempo real [6], [7].

Una vez que se tiene diseñado y construido un cuadricóptero estable y maniobrable se pueden desarrollar muchas aplicaciones con él. Destacan en el medio científico los logros de Raffaello D'Andrea quien usando técnicas novedosas de control [8] ha realizado impresionantes acrobacias dinámicas con conjuntos de cuadricópteros como jugar badminton, balancear, lanzar y capturar una varilla, e incluso la construcción de formas arquitectónicas complejas, todo esto de manera autónoma [9].

MICHELL P EHRLICH pertenece a la carrera INGENIERÍA MECATRÓNICA de la FACULTAD DE INGENIERÍA y realizó el proyecto dentro del LABORATORIO DE ROBÓTICA MÓVIL Y SISTEMAS AUTOMATIZADOS (Email: michell_pro@hotmail.com).

El proyecto fue asesorado por LUIS F. LUPIÁN.

Figura 1: Tercera iteración de construcción del cuadricóptero

Dentro del presente proyecto, el objetivo de desarrollar el sistema de control de vuelo se logra con una arquitectura de controladores PID. Existen antecedentes del uso de esta técnica de control para estabilizar el vuelo de cuadricóperos como el trabajo de Tayebi [4].

La Fig. 1 muestra el cuadricóptero desarrollado dentro del presente proyecto en su estado actual. La aeronave ya es capaz de volar y auto-estabilizarse de manera satisfactoria como se puede ver en el video anexo [10]. El presente proyecto se divide en dos partes: "hardware y construcción física" y "software, estimación y control".

II. Descripción del problema

Escribir un controlador de vuelo para un cuadricóptero no es tarea fácil, ya que tiene muchas partes que deben interactuar armónicamente para que opere de manera eficiente y sin problemas. Los dos componentes más importantes son la estimación de variables a partir de la información ruidosa de los sensores y los algoritmos de control que comandan la velocidad de las hélices para lograr el efecto deseado.

El objetivo específico de control en el presente proyecto es controlar de manera retroalimantada los tres grados de libertad esenciales del cuadricóptero: los ángulos de inclinación conocidos como *pitch* (α) y *roll* (φ), así como la velocidad angular de giro aledeor de la vertical ($\dot{\theta}$) también conocido como ángulo de *yaw* (θ), los cuales están orientados

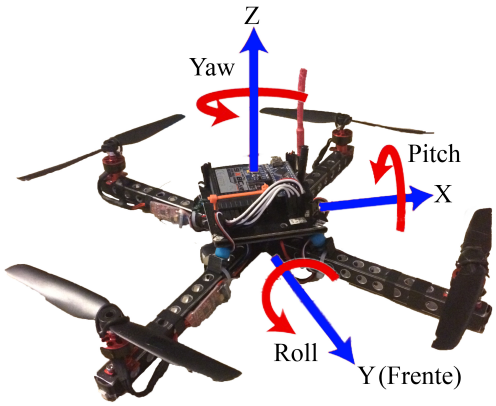


Figura 2: Ejes de rotación y marco de referencia local

con respecto al cuerpo de la nave de acuerdo a lo indicado en la Fig. 2. Los ángulos de inclinación de *pitch* y *roll* le permiten a la nave lograr desplazamientos hacia adelante/atrás e izquierda/derecha, respectivamente. Hay un cuarto grado de libertad que corresponde al desplazamiento en dirección perpendicular al plano de los rotores, el cual no está siendo controlado de manera automática debido a que en su versión actual el cuadricóptero no cuenta con un sensor que le permita estimar la altura. La razón por la que la mayoría de los multicópteros se construyen con cuatro hélices es precisamente porque esta es la mínima cantidad de grados de libertad controlables requeridos para controlar de manera holonómica los cuatro grados de libertad de la nave.

III. Recepción e interpretación de señal

Una vez que el transmisor y receptor están funcionando, el siguiente paso es comunicarle a la computadora Arduino la señal que está saliendo del receptor e interpretarla para obtener información significativa de lo que está comandando el piloto. Para lograr esto primero hay que entender qué señal está saliendo del receptor. La señal que sale de la mayoría de los receptores se usa de manera estándar en servos de radio control, y es una clase de señal PWM. En esta señal hay un pulso alto de entre 1 ms y 2 ms, siendo el primero la mínima señal y el segundo la máxima, seguido de 18 ms–19 ms, lo necesario para completar 20 ms y después este proceso se repite, dando así actualizaciones a 50 Hz. Para leer la señal PWM desde la computadora Arduino se utiliza normalmente la función `pulseIn()`, que devuelve la duración en μs del pulso alto, pero resulta que esta función es muy ineficiente. El método que utiliza es comenzar a "escuchar" el pin que se le indica hasta que haya un pulso alto, esperar a que acabe y después devolver la duración. Esto es terriblemente ineficiente ya que si tenemos la mala fortuna de comenzar a "escuchar" justo cuando acaba de terminar un pulso alto, tendremos que esperar entre 18 ms y 19 ms para escuchar el siguiente, tiempo en el que el procesador está inactivo y se podría utilizar para hacer cálculos o leer sensores, así que fue necesario un método más eficiente.

La mejor forma de hacer esto para no tener tiempo muerto en el procesador es utilizar interrupciones. La Arduino Nano tiene sólo dos interrupciones por hardware, pero se pueden hacer interrupciones por software utilizando la librería PinChangeInt [11], tal que cuando se detecte algún cambio en el estado del pin, comience una rutina de servicio de interrupción, o ISR por sus siglas en inglés. Esta rutina responde a las transiciones de nivel en el pin de entrada seleccionado, si hay una transición de BAJO a ALTO, guarda en una variable el tiempo actual que lleva el microprocesador activado y si pasó de ALTO a BAJO, resta el tiempo que había guardado en la rutina anterior con el tiempo actual, resultando así en el tiempo en μs que estuvo el pin en estado ALTO. Esto es mucho más eficiente ya que solamente utiliza unas cuantas líneas para hacer la asignación de tiempo y la substracción, dejando todo el tiempo entre rutinas para que siga corriendo el ciclo de control. Originalmente se utilizaba la función `micros()` que devuelve el tiempo en μs, pero esta sólo tiene una resolución de 4 μs, por lo que para hacer una medición más precisa se utilizó directamente el timer 1 del microcontrolador, el cual trabaja en "counts" o unidades de ejecución. Cada uno de estos equivale a 0.5 μs, por lo que lo hace mucho más preciso.

El módulo de interpretación de señal se desarrolló con base en lo que describe Duane B [12]. Una vez que se tuvo confianza en que nuestra propia implementación funcionaba y que se entendía claramente su funcionamiento, se decidió utilizar la librería que distribuye el mismo autor, la cual que hace exactamente lo descrito en esta sección pero empacado limpiamente.

Para generar la señal que se le manda al controlador de motor se utiliza un método muy similar al usado para leer la señal del receptor, pero al revés. La interrupción en vez de estar ligada a un pin se liga al timer, así cuando llegue a cierto tiempo se dispara una interrupción y cuando pase cierto tiempo otra, que ponen el pin en ALTO y en BAJO, respectivamente.

IV. Estimación de variables de medición inercial

La unidad de medición inercial (IMU) seleccionada es un MPU6050, con un acelerómetro y un giroscopio de 3 ejes cada uno para dar así 6 grados de libertad. Se eligió éste por su existencia en tiendas locales y bajo precio.

Para leer este IMU desde la computadora Arduino, se conectó a los puertos de I^2C y se utilizó la librería escrita por Jeff Rowberg [13] para obtener los valores "crudos", es decir, los valores numéricos tal como los dan los sensores sin ningún tipo de filtro ni procesamiento. Lo primero que se tiene que hacer con estos valores es convertirlos a una escala útil, por lo que se multiplican por alguna constante de proporcionalidad según el nivel de sensibilidad, como se especifica en la hoja técnica del fabricante [14].

La información que se requiere obtener del IMU dentro del ciclo de control es velocidad angular y ángulo con respecto a la horizontal. La primera se obtiene directamente del giroscopio, mientras que para obtener el ángulo con respecto a la horizontal es necesario un pre-procesamiento. El IMU provee dos fuentes de información que permiten la estimación del ángulo con respecto a la horizontal: (a) el acelerómetro mide el vector de aceleración de la gravedad, que por medio de trigonometría se usa para calcular el ángulo deseado, (b) el giroscopio mide

la velocidad angular, cuya integral numérica con respecto al tiempo da una estimación del ángulo de giro desde el inicio de la integración. La primera fuente de información es altamente ruidosa debido a las aceleraciones causadas por la vibración de los motores por lo que es muy poco confiable en lapsos cortos pero en el promedio a lo largo del tiempo es muy confiable. La segunda fuente de información es muy confiable en lapsos cortos, pero acumula error conforme avanza el tiempo. La fusión de la información proveniente de estas dos fuentes es un problema clásico de estimación que tiene soluciones igualmente clásicas, entre las que destaca el filtro de Kalman [15], que se caracteriza por encontrar la estimación óptima. El algoritmo de fusión de sensores que se usó en este trabajo es una aproximación de estado estacionario al filtro de Kalman (o filtro de Wiener) llamado filtro complementario [16], [17]. Aún cuando la estimación del filtro complementario no es óptima, tiene la ventaja sobre el filtro de Kalman de que se puede diseñar sin conocer la descripción estadística de las fuentes de información, además de que en casos de segundo orden o mayor su costo computacional es menor. El primer antecedente que se tiene del uso del filtro complementario para estimar la pose de una máquina voladora es la patente de Klein en 1987 [18]. Concretamente, el filtro aplicado para obtener la estimación del ángulo en este proyecto es el indicado en (1)

$$\hat{\beta}_k = \gamma(\hat{\beta}_{k-1} + \dot{\beta}_{giro}T) + (1-\gamma)\beta_{acel} \tag{1}$$

donde $\hat{\beta}_k$ es la estimación del ángulo producida por el filtro complementario en el instante k, T es el periodo de muestreo o duración del ciclo de control ($\frac{1}{286}$ s para los experimentos aquí presentados), γ es un parámetro adimensional de ponderación del término de integración de la velocidad angular, $\dot{\beta}_{giro}$ es la velocidad angular calculada del giroscopio y β_{acel} es el ángulo calculado del acelerómetro. Estas últimas dos variables se calculan de acuerdo a (2) y (3)

$$\dot{\beta}_{giro} = \frac{W_{giro}}{131} \tag{2}$$

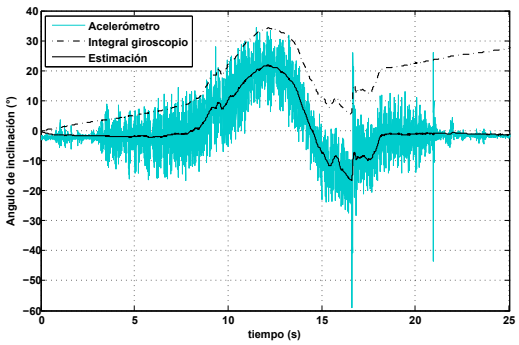
$$\beta_{acel} = \frac{180^\circ}{\pi}\arctan\left(\frac{A_y}{A_z}\right) \tag{3}$$

donde W_{giro} es el valor numérico obtenido directamente del giroscopio, $1/131$ es el factor de proporcionalidad especificado por el fabricante del MPU6050 [14] para convertir W_{giro} a unidades de $^\circ$/s, A_y y A_z son la proyección del vector de aceleración con respecto a los ejes Y y Z del acelerómetro, respectivamente, y $180^\circ/\pi$ es el factor de proporcionalidad para convertir de radianes a grados. La implementación del filtro complementario descrito en esta sección está inspirada en la descripción presentada por Van de Maele [19].

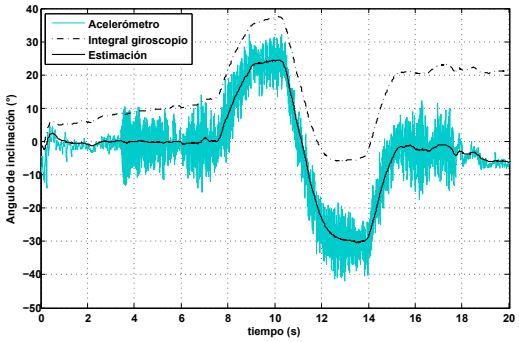
La recursión (1) representa en realidad un par de filtros, un paso bajas para el acelerómetro y un paso altas para la integral del giroscopio [20]. Ambos filtros tienen constante de tiempo igual a τ y frecuencia de corte $f_c = 1/\tau$. El parámetro γ se obtiene a partir de

$$\gamma = \frac{\tau}{\tau + T} = \frac{1}{1 + T/\tau} = \frac{1}{1 + T f_c} \tag{4}$$

La frecuencia de corte f_c se elige de tal forma que se rechace el ruido de alta frecuencia del acelerómetro. Para nuestra aplicación se encontró que $f_c = 2\,\text{Hz}$ da un resultado aceptable ya que las oscilaciones causadas por los motores son de mucho más alta frecuencia, mientras que su inverso (0.5 s) es un tiempo suficientemente pequeño para garantizar que la integración del giroscopio no ha acumulado demasiado error y por lo tanto sigue siendo confiable. Tomando en cuenta que el periodo de muestreo $T = \frac{1}{286}$ s y substituyendo en (4) se obtiene que $\gamma = 0.993$, que para nuestros propósitos se redondeó a $\gamma = 0.99$ dando así una frecuencia de corte real $f_c = 2.88\,\text{Hz}$.



(a) Primer experimento



(b) Segundo experimento

Figura 3: Resultados del filtro complementario

En la Fig. 3 se presentan los resultados de dos experimentos en los que se evalúa la efectividad del filtro complementario diseñado. En cada una de las gráficas se muestra como función del tiempo: el ángulo calculado con el acelerómetro, el ángulo calculado de la integral del giroscopio, y la estimación del ángulo producida por el filtro complementario. En ambos casos puede apreciarse que el ángulo calculado con el acelerómetro es demasiado ruidoso y por lo tanto no es recomendable utilizarlo. Por otro lado, aunque el ángulo obtenido como la integral del giroscopio es bastante limpio en cuanto a ruido, claramente este ángulo se desvía del valor promedio conforme avanza el tiempo, por lo tanto por sí solo tampoco es recomendable utilizarlo. Finalmente, la estimación del ángulo que produce el filtro complementario sigue fielmente el promedio de la señal ruidosa del acelerómetro pero lo hace limpiamente como la integral del giroscopio, por lo que queda claro que fusiona muy bien la información de ambos sensores.

V. Implementación del algoritmo PID digital

Un controlador proporcional, integral y derivativo (PID) es, descrito brevemente, un tipo de ciclo de control con retroalimentación que toma en cuenta una señal de error, dada por la diferencia entre la medida actual de una variable y el objetivo al que se quiere que se llegue esta. Con esta señal de error se generan tres componentes: el proporcional que es únicamente la señal de error multiplicada por alguna constante, el integral que es la suma de los errores con el tiempo y el derivativo que describe como está cambiando el error con respecto del tiempo. Esto se utiliza para controlar sistemas de manera precisa y con un comportamiento específico, y este tipo de controlador es el que se utiliza en la vasta mayoría de los cuadricópteros. La forma de implementar un ciclo de control con PID en una computadora Arduino es descrita con detalle por Beauregard [21], el desarrollador de la librería de PID para Arduino.

Para este proyecto primero se desarrolló una implementación propia del algortitmo del PID digital en lenguaje C++ para verificar que se tenía una comprensión clara de la teoría y aplicación. Una vez alcanzado este objetivo, se remplazó nuestra implementación del PID por la de la librería desarrollada por Beauregard [21] para conseguir un código más limpio, y se verificó que en ambos casos se obtenía el mismo resultado.

VI. Sistema de control de vuelo

Por medio del transmisor, el piloto cuenta con cuatro señales de referencia que puede manipular a su conveniencia. Dos de ellas se usan para comandar el *pitch* (α_d) y *roll* (φ_d) deseados, una tercera se usa para comandar la velocidad angular de *yaw* ($\dot{\theta}_d$) deseada, y la cuarta se usa como acelerador para hacer que varíe la velocidad promedio de las hélices (ω_d) y hacer de esta forma que la nave suba o baje. Cabe hacer notar que debido a que el cuadricóptero no cuenta con ninguna forma de retroalimentación de la altura es responsabilidad del piloto visualmente manipular el desplazamiento en dirección vertical.

VII. Arquitectura de controladores PID

La primera implementación que se hizo con PID tenía uno de estos controladores para cada eje (*pitch* y *roll*). Se mapeó la palanca del transmisor de $-20°$ a $20°$ y eso se le alimentó al PID como objetivo comparándolo contra la estimación del ángulo que produce el filtro complementario correspondiente. Todas las implementaciones de PID se probaron en un soporte de prueba, que sólo le permitía al cuadricóptero girar alrededor de un eje (ángulo de *roll*), para analizar el comportamiento más claramente.

Esta primera implementación de PID, aunque lograba balancearse, le faltaba fuerza, tardaba en regresar a la horizontal y tenía pequeñas oscilaciones, sin importar cuántas combinaciones de coeficientes se probaron. Esto fue mucho más evidente al momento de intentar un vuelo, ya que le faltaba potencia que lo centrara por lo que aunque no se volteaba completamente, no se lograba estabilizar bien por lo que se movía impredeciblemente de lado a lado. Al no tener éxito experimental se regresó a la investigación para encontrar la razón por la que no funcionaba correctamente.

Después de analizar distintas implementaciones y códigos de PID para control de vuelo de cuadricópteros, se descubrió que todos utilizaban dos controladores PID en cascada por eje. Uno como el descrito anteriormente y otro que utilizaba como objetivo la salida del primer PID comparado contra la velocidad angular medida por el giroscopio ($\dot{\alpha}_m$ y $\dot{\varphi}_m$). Al probar únicamente el segundo PID se notó un comportamiento mucho más estable, al mover la palanca el cuadricóptero rotaba a cierta velocidad y al regresarla al centro éste se detenía. Es posible despegar así, pero la constante derivación del giroscopio combinado con el hecho de que al rotar el cuadricóptero éste se detiene donde se quedó, efectivamente avanzando en esa dirección hasta que se gire en sentido contrario hace que pilotearlo de esta manera sea algo muy complicado, reservado sólo para los pilotos acrobáticos más experimentados.

Cuando se probó esta implementación, con los dos PID, uno alimentando al otro se tuvo un mucho mejor resultado. Al salir de balance el cuadricóptero regresaba rápidamente al origen. La arquitectura de dos controladores PID en cascada se puede apreciar en el diagrama de bloques de la Fig. 4.

VIII. Esquema de control completo

El diagrama de bloques de la Fig. 4 muestra el esquema de control completo. En el lado izquierdo se encuentra el transmisor, donde se pueden apreciar las cuatro variables de referencia disponibles para el piloto: *throttle* (ω_d), velocidad angular de *yaw* ($\dot{\theta}_d$), ángulo de *roll* (φ_d), ángulo de *pitch* (α_d). En el lado derecho están los cuatro controladores de motores cuya función es recibir una señal que representa el nivel de velocidad deseado para el motor y producir la acción de control necesaria. La velocidad angular de las hélices del cuadricóptero, al producir más o menos empuje según giren más rápido o más lento, se traduce en un movimiento que se interpreta como un cambio en los valores reales de α, φ, θ, así como las primeras derivadas con respecto al tiempo de cada una. Los cambios en estas variables son a su vez medidos indirectamente por el IMU, el cual entrega datos crudos contaminados con ruido. Los filtros complementarios de *pitch* y *roll* se encargan de fusionar los datos del IMU para generar una estimación mejorada de estas dos variables ($\hat{\alpha}$ y $\hat{\varphi}$). Cada una de estas dos estimaciones es comparada con las correspondientes señales comandadas por el transmisor (α_d y φ_d) para producir las correspondientes señales de error ($\alpha_d - \hat{\alpha}$ y $\varphi_d \hat{-} \varphi$), cada una de las cuales entra a una primera etapa de controlador PID digital cuya función es producir una señal de referencia de velocidad angular ($\dot{\alpha}_d$ y $\dot{\varphi}_d$) que busque reducir el error en el ángulo correspondiente. Estas señales de referencia son comparadas contra las correspondientes mediciones de velocidad angular que reporta el IMU directamente ($\dot{\alpha}_m$ y $\dot{\varphi}_m$) para producir señales de error que entrarán a una segunda etapa de controlador PID digital cuya función es generar la acción de control necesaria para mantener la velocidad angular de referencia. Por otro lado, la velocidad angular de *yaw* ($\dot{\theta}_d$) comandada desde el transmisor

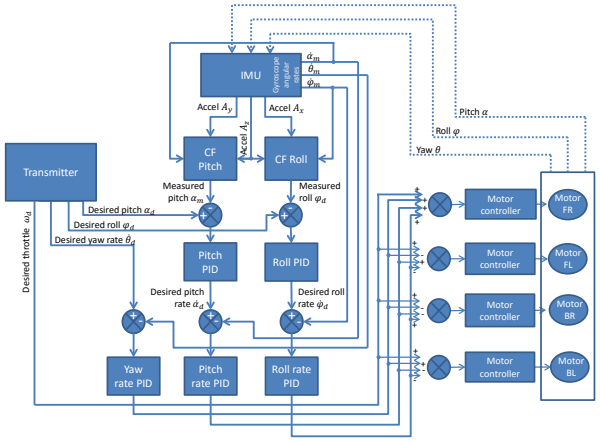


Figura 4: Diagrama de bloques del sistema de control (abreviaturas: CF=complementary filter, FR=front right, FL=front left, BR=back right, BL=back left)

es comparada contra la velocidad angular de *yaw* ($\dot{\theta}_m$) medida por el IMU para producir una señal de error que entra a un PID digital cuya función es producir la acción de control necesaria para mantener la velocidad angular deseada. Finalmente, las cuatro acciones de control producidas se suman (con signos apropiados de acuerdo a la ubicación relativa de cada motor) para producir una acción de control específica para cada motor. Por ejemplo, el empuje vertical (*throttle*) implica un aumento de velocidad en todos los motores, pero para que la nave gire en la dirección positiva de *roll* es necesario que disminuya la velocidad de los motores izquierdos al mismo tiempo que aumenta la de los derechos.

IX. Sintonización de controladores PID

Para sintonizar los parámetros de los controladores PID más finamente se utilizó el siguiente procedimiento:

- Primero se incrementa el coeficiente K_P, hasta que comience a haber oscilaciones rápidas, y se decrementa un poco. En este punto, ya es posible despegar en vuelo.
- Después, se incrementa el coeficiente K_I hasta que el cuadricóptero regrese a la posición deseada después de una perturbación rápida y se resista a quedarse en algún punto fuera del centro. Se sigue incrementando hasta que haya oscilaciones de más baja frecuencia, entonces se decrementa un poco.
- Finalmente, se incrementa el coeficiente K_D hasta que el cuadricóptero regrese rápidamente al origen sin pasarse. Este coeficiente depende mucho del tipo de vuelo deseado, incrementar más para un vuelo más estable y menos responsivo, y decrementar para un vuelo más ágil pero un poco menos estable.

X. Experimentos de control

Usando el procedimiento de sintonización descrito se diseñaron varios controladores PID con distintos grados de éxito.

En la Fig. 5 se muestra el comportamiento de un PID con un valor de K_P muy bajo, con K_I y K_D igual a 0. El cuadricóptero no alcanza el ángulo deseado y tiene un

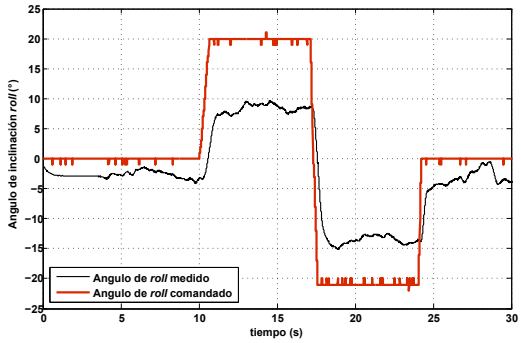


Figura 5: Control proporcional con ganancia baja ($K_P = 0.3$, $K_I = 0$, $K_D = 0$)

error en estado estacionario muy grande, aparte de que tiene una oscilación de $\pm 3°$ cuando alcanza el punto de equilibrio. Claramente se necesita incrementar el valor de K_P. Si se intenta volar el cuadricóptero en este estado tendría una respuesta extremadamente débil, siendo casi inservible por la falta de estabilidad.

La Fig. 6 ilustra el comportamiento con un valor de K_P muy alto. Nótese cómo hay oscilaciones de muy alta frecuencia, con aproximadamente $5°$ de amplitud. El error en estado estacionario se redujo en gran medida, pero aún existe en algunos puntos. Este tipo de oscilación es muy característico de que se debe decrementar el valor de K_P. A partir de este punto ya es posible volar el cuadricóptero, pero se tienen oscilaciones de alta frecuencia que dificultan el control.

En la Fig. 7 se muestra el efecto que tiene un valor de K_P que se acerca al valor nominal. Al alcanzar el equilibrio existen muy pocas oscilaciones, menores a un grado de amplitud. Hay un error en estado estacionario más grande que con un valor de K_P muy alto, pero esto se eliminará una vez agregado el término K_I. Es posible un vuelo estable en este punto, pero si hay alguna fuerza externa como por ejemplo de una corriente de aire, no opondrá suficiente resistencia y será arrastrado.

Utilizando el valor de K_P previamente encontrado que tiene resultados satisfactorios, se agrega ahora el término integral cuyo efecto se puede apreciar en la Fig. 8. El error en estado estacionario se ha reducido hasta ser casi inexistente, y las oscilaciones son mínimas, teniendo una amplitud menor a $1°$. En este punto el cuadricóptero se puede volar de manera muy estable y ágil, logrando los objetivos de este proyecto de investigación.

El término D se puede agregar según el tipo de vuelo que se desee, incrementándolo para una respuesta menos agresiva al acercarse al objetivo, lo que se traduce en mayor estabilidad pero menor agilidad. Para los objetivos de este proyecto, se decició dejar el valor de K_D en 0.

XI. Conclusiones y trabajo futuro

En su estado actual, el cuadricóptero ya es capaz de volar y autoestabilizarse de manera muy satisfactoria. Se cumplieron los objetivos ya que aparte de conseguir una aeronave funcional, se aplicaron conceptos sobre programación, teoría de control, diseño de piezas y selección de materiales.

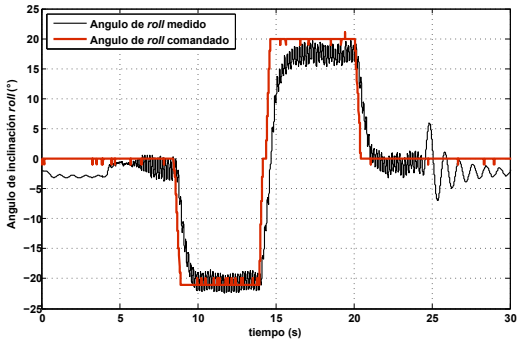


Figura 6: Control proporcional con ganancia alta ($K_P = 5$, $K_I = 0$, $K_D = 0$)

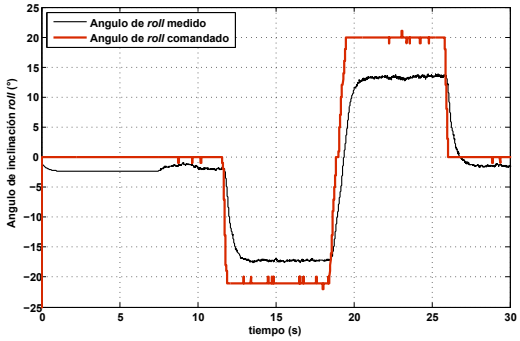


Figura 7: Control proporcional con ganancia nominal ($K_P = 1$, $K_I = 0$, $K_D = 0$)

Se siguen buscando mejores y más eficientes algoritmos de control y estimación para poder ejecutar el ciclo de control con mayor frecuencia y precisión aumentando la estabilidad e implementar distintos sensores como barómetro o ultrasónico e integrarlos al código para desarrollar control de altura automático, ya que al momento del pilotaje mantener la altura es una de las operaciones que más práctica requieren. El avance más necesario está en la interpretación de la orientación del cuadricóptero, ya que en la implementación actual se realiza con funciones trigonométricas, las cuales dejan de dar resultado en ciertos ángulos cuando el cuadricóptero gira más de 90°, impidiendo hacer acrobacias que requieran vuelo invertido. Para esto se utilizarán cuarterniones o matrices de rotación consiguiendo así una representación precisa de la orientación sin importar el ángulo.

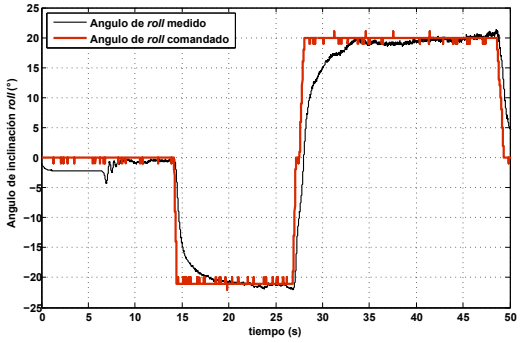


Figura 8: Control PI con ganancia proporcional nominal y ganancia integral nominal ($K_P = 1$, $K_I = 1$, $K_D = 0$)

REFERENCIAS

[1] I. Gaponov and A. Razinkova, "Quadcopter design and implementation as a multidisciplinary engineering course," in *Teaching, Assessment and Learning for Engineering (TALE), 2012 IEEE International Conference on*, Aug 2012, pp. H2B–16–H2B–19.

[2] S. Bouabdallah, P. Murrieri, and R. Siegwart, "Design and control of an indoor micro quadrotor," in *Robotics and Automation, 2004. Proceedings. ICRA '04. 2004 IEEE International Conference on*, vol. 5, April 2004, pp. 4393–4398 Vol.5.

[3] R. Mahony, V. Kumar, and P. Corke, "Multirotor aerial vehicles: Modeling, estimation, and control of quadrotor," *Robotics Automation Magazine, IEEE*, vol. 19, no. 3, pp. 20–32, Sept 2012.

[4] A. Tayebi and S. McGilvray, "Attitude stabilization of a four-rotor aerial robot," in *Decision and Control, 2004. CDC. 43rd IEEE Conference on*, vol. 2, Dec 2004, pp. 1216–1221 Vol.2.

[5] H. Voos, "Nonlinear control of a quadrotor micro-uav using feedback-linearization," in *Mechatronics, 2009. ICM 2009. IEEE International Conference on*, April 2009, pp. 1–6.

[6] M. Earl and R. D'Andrea, "Real-time attitude estimation techniques applied to a four rotor helicopter," in *Decision and Control, 2004. CDC. 43rd IEEE Conference on*, vol. 4, Dec 2004, pp. 3956–3961 Vol.4.

[7] J. Hall, N. Knoebel, and T. McLain, "Quaternion attitude estimation for miniature air vehicles using a multiplicative extended kalman filter," in *Position, Location and Navigation Symposium, 2008 IEEE/ION*, May 2008, pp. 1230–1237.

[8] O. Purwin and R. D'Andrea, "Performing aggressive maneuvers using iterative learning control," in *Robotics and Automation, 2009. ICRA '09. IEEE International Conference on*, May 2009, pp. 1731–1736.

[9] S. Lupashin, M. Hehn, M. W. Mueller, A. P. Schoellig, M. Sherback, and R. D'Andrea, "A platform for aerial robotics research and demonstration: The flying machine arena," *Mechatronics*, 2014.

[10] M. P. Ehrlich and L. F. Lupián, "Prueba de vuelo de cuadricóptero," Publicación electrónica: *http://youtu.be/A0CQ2GqSkhw* Consultada 2014/12/05.

[11] Arduino SA, "PinChangeInt Library," Publicación electrónica: *http://playground.arduino.cc/Main/PinChangeInt* Consultada 2014/12/01.

[12] Duane B, "Arduino Projects, Libraries and Tutorials," Publicación electrónica: *http://rcarduino.blogspot.mx/* Consultada 2014/12/01.

[13] J. Rowberg, "I^{2}Cdevlib: MPU-6050 6-axis accelerometer/gyroscope," Publicación electrónica: *http://www.i2cdevlib.com/devices/mpu6050* Consultada 2014/12/01.

[14] *MPU-6000/MPU-6050 Product Specification*, 3.4 ed., InvenSense, Inc., USA, Aug. 2013.

[15] R. E. Kalman, "A New Approach to Linear Filtering and Prediction Problems," *Trans. ASME, J. Basic Eng.*, vol. 82 (Series D), pp. 35–45, 1960.

[16] W. Higgins, "A Comparison of Complementary and Kalman Filtering," *Aerospace and Electronic Systems, IEEE Transactions on*, vol. AES-11, no. 3, pp. 321–325, May 1975.

[17] M. Euston, P. Coote, R. Mahony, J. Kim, and T. Hamel, "A complementary filter for attitude estimation of a fixed-wing UAV," in *Intelligent Robots and Systems, 2008. IROS 2008. IEEE/RSJ International Conference on*, Sept 2008, pp. 340–345.

[18] R. W. Klein, "Flight control system employing complementary filter," May 21 1987, WO Patent App. PCT/US1986/002,281. [Online]. Available: http://www.google.com/patents/WO1987002964A1?cl=en

[19] P.-J. Van de Maele, "Reading a IMU Without Kalman: The Complementary Filter," Publicación electrónica: *http://www.pieter-jan.com/node/11* Consultada 2014/12/01.

[20] H. Gi-Min and E. Tae-Jeung, "Complementary Filter Design for Angle Estimation using MEMS Accelerometer and Gyroscope," Publicación electrónica: *http://www.academia.edu/6261055* Consultada 2014/12/08.

[21] B. Beauregard, "Improving the Beginner's PID – Introduction," Publicación electrónica: *http://brettbeauregard.com/blog/2011/04/improving-the-beginners-pid-introduction/* Consultada 2014/12/01.