Diseño de una computadora de navegación, guiado y control: Aplicación a un vehículo aéreo no tripulado

Claudio Pose[a], Federico Roasio[a] y Juan Giribet[a,b]
[a] Grupo de Procesamiento de Señales, Identificación y Control – GPSIC
Departamento de Ingeniería Electrónica, Universidad de Buenos Aires.
[b] Instituto Argentino de Matemática – CONICET.
cdpose88@gmail.com, federoasio@gmail.com, jgiribet@fi.uba.ar

***Resumen*-El propósito de este trabajo es presentar un prototipo de computadora de navegación, guiado y control que ha sido desarrollado en el GPSIC. El aporte principal del trabajo es el desarrollo de la arquitectura y firmware del sistema. Como ejemplo de aplicación se muestran resultados en donde esta computadora se utiliza en un vehículo aéreo no tripulado del tipo multirotor.**

I. INTRODUCCIÓN

Cuando se trabaja con sistemas de navegación, guiado y control (NG&C) suele establecerse cierto compromiso entre la capacidad de cómputo y el determinismo en la ejecución de las tareas [1]. Por un lado, los algoritmos de navegación integrada, suelen requerir mucho cómputo para tareas como procesar imágenes o correr filtros de fusión de datos. Mientras tanto, el sistema de control impone restricciones sobre el tiempo de ejecución de la ley de control. Además, para garantizar estabilidad, es necesario cierto determinismo en los tiempos de ejecución del lazo de control.

Por otro lado, el filtro de fusión de datos del sistema de navegación no requiere determinismo en su tiempo de ejecución, admitiendo que haya cierto retardo para entregar el dato preciso de navegación, siempre y cuando se sepa a qué instante de tiempo corresponden todos los datos de los sensores. El grado de precisión que se consiga en las etiquetas de tiempo de la información que alimenta al algoritmo de fusión, impactará directamente en el desempeño del sistema de navegación [2].

En este trabajo se presenta el desarrollo de una computadora de NG&C que contempla este compromiso que se presenta cuando se trabaja con este tipo de sistemas. La computadora tiene como objetivo ser utilizada en vehículos aéreos no tripulados pequeños, en particular ha sido utilizada en un hexarotor F550 de la empresa DJI.

Un hexarotor es un helicóptero dotado de seis conjuntos motor-hélice que generan empuje de aire; la capacidad de ejercer torques de control sobre el vehículo, es resultado de la diferencia de empuje que se puede hacer realizar a los motores. Puede decirse que como sistema de control es subactuado, ya que para variar la posición del vehículo es necesario aplicar un cambio en la inclinación del mismo, en *pitch* y/o *roll*. Por otro lado, cualquier cambio en la inclinación generará un cambio en la posición del vehículo.

El sistema de NG&C propuesto se divide en dos partes, una de bajo nivel (con restricciones fuertes para los tiempos de ejecución de las tareas) y una de alto nivel (con requerimientos fuertes para la capacidad de cómputo). En este trabajo se detalla la computadora de bajo nivel, la cual ejecuta tres lazos de control PID para estabilizar los ángulos de navegación del vehículo. Pero cabe destacar que la computadora de alto nivel es una parte central del sistema de NG&C, de hecho es ésta la computadora encargada de dar una estimación precisa de la posición, velocidad y orientación del vehículo (mediante un filtro de Kalman Extendido que fusiona mediciones de sensores inerciales, magnetómetros, GPS, barómetros y cámaras). Por su parte, ejecuta un lazo de control externo por encima de los PID, basado en una ley control robusto H∞, que garantiza la estabilidad del vehículo, mejorando el desempeño y encargándose además del control de posición. La computadora de alto nivel trabaja con un sistema Linux embebido y por lo tanto no puede garantizar los tiempos de ejecución de las tareas. Esto es el objetivo de la computadora de bajo nivel (a la cual nos referiremos de aquí en adelante simplemente como computadora de NG&C).

II. SISTEMA DE NG&C

El procesador de la computadora de NG&C es un LPC1769 de la empresa NXP con un conector JTAG que permite su programación. Es un procesador ARM Cortex M3 de 120MHz. La unidad de medidas inerciales es un MPU6000 que se conecta al procesador principal a través de un protocolo SPI. Además, un magnetómetro de Honeywell (HMC5883L) y un barómetro de la firma Bosch (BMP180) se comunican con el LPC1769 mediante un bus I2C. el receptor GPS utilizado es un ET332, el cual emplea una conexión estándar RS232. Las señales de EOC de dichos sensores (*End Of Conversion*) y PPS (*Pulse Per Second*) del GPS se conectan a entradas de interrupciones externas. Estas señales tienen un rol central en la sincronización de la información de los diversos sensores que componen el sistema de navegación.

Las señales de PWM (Pulse Width Modulation) son utilizadas para controlar los ESC (Electronic Speed Controller) de los motores brushless del hexarotor. La placa y el esquema de sensores y entradas/salidas de la computadora de NG&C pueden verse en la Fig. 1 y Fig. 2, respectivamente.

El sistema se alimenta mediante una fuente switching diseñada utilizando un TPS5430 de Texas Instruments. La elección de

este módulo para alimentar la computadora se debe a que utilizando la misma batería LiPo utilizada para alimentar los motores, se debe alimentar la computadora basada en el LPC1769 así como una computadora de más alto nivel, basada en un Cortex A8 (que no se detallará en este trabajo), cuyo consumo es elevado y hace ineficiente el uso de un regulador lineal.

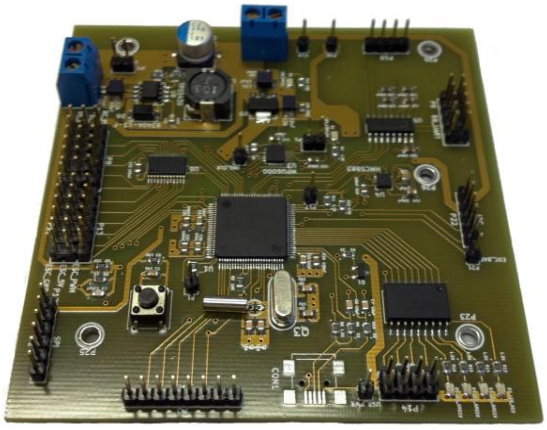

Fig. 1 Computadora de navegación, guiado y control

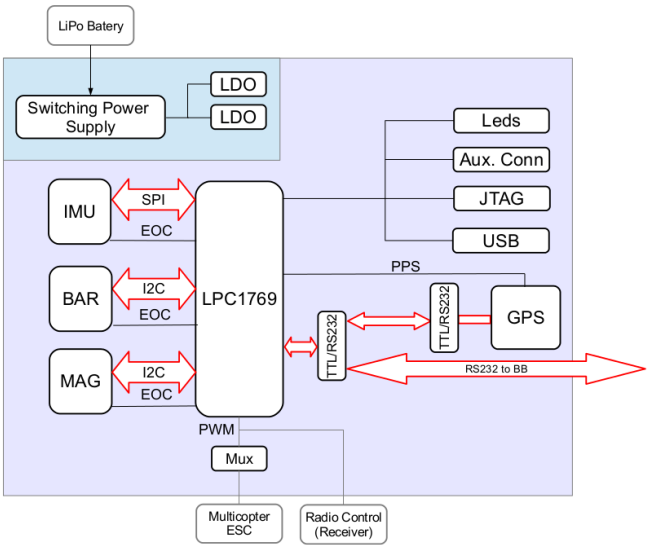

Fig. 2 Esquema de componentes y entradas/salidas de la computadora de navegación, guiado y control.

III. FIRMWARE

El firmware ha sido desarrollado en el lenguaje C mediante el IDE LCPXpresso de NXP. Utiliza un crosscompilador GCC para microcontroladores ARM Cortex M3. Este IDE no sólo permite la codificación, compilación y carga del firmware sino también herramientas para debugging.

El firmware ha sido diseñado en tres capas. En el nivel más bajo se utiliza la interfaz estándar para microcontroladores Cortex desarrollada por ARM y NXP, los periféricos se manejan a través de la librería desarrollada por NXP y finalmente el tercer nivel es el de aplicación el cual se basa en un conjunto de librerías y el programa de aplicación que ha sido desarrollado en el GPSIC.

Como se mencionó anteriormente, los sistemas de NG&C usualmente tienen fuertes requerimientos respecto al tiempo de ejecución, esto viene dado principalmente por el sistema de control. Para garantizar el correcto funcionamiento del sistema de control usualmente se establecen cotas sobre su tiempo de ejecución que deben ser garantizadas (incluso en ciertos sistemas se establecen condiciones duras de tiempo real). Pero no es el único sistema que impone restricciones fuertes respecto al tiempo de ejecución. Si bien el sistema de navegación no establece restricciones duras de tiempo real, es necesario el sincronismo de los distintos sensores, más aún es necesario conocer los instantes de tiempo a los que corresponden cada uno de los sensores para que el algoritmo pueda ejecutarse correctamente sin perder desempeño. Por ejemplo, el sistema de navegación puede utilizar un dato de GPS con retardo, siempre que este retardo sea tenido en cuenta. Es decir que el algoritmo de adquisición de datos también impone restricciones sobre el tiempo de ejecución, quizás no tan fuertes como las de navegación (muchas veces se puede hablar de restricciones firmes o incluso suaves, dependiendo de la aplicación). Por otro lado, es necesario conocer la posición, velocidad y orientación del vehículo para resolver el control. Esto fija dos niveles de algoritmos de navegación (y sensores), uno que satisface requerimientos fuertes como los del control y otros más débiles.

Uno de los requisitos para el sistema de adquisición es recolectar datos de sensores inerciales a una frecuencia de 800Hz, lo cual es algo difícil de lograr utilizando un sistema operativo de tiempo real en un microcontrolador Cortex M3. Un RTOS podría reducir el determinismo en el tiempo de muestreo (*jitter*), es por esto que se decidió utilizar un firmware que no se basa en RTOS y se desarrolló un pequeño sistema de *schedulling* diseñado ad-hoc para esta aplicación.

El firmware se desarrolló utilizando un estilo de programación por eventos para aprovechar el desempeño del microcontrolador. Este paradigma de programación es muy útil en sistemas de tiempo real porque cada periférico puede realizar su tarea y cuando finaliza envía una señal (evento) al microcontrolador el cual procesa la información enviada por el periférico. De esta forma el microcontrolador no se *bloquea* mientras espera que el periférico finalice con la realización de una tarea, pudiendo así aprovechar el tiempo para realizar otras tareas.

El sistema diseñado desempeña dos tareas principales: por un lado la recolección de datos y su posterior envío al sistema de procesamiento, y por otro lado la recepción de comandos por parte del sistema de procesamiento de datos y la posterior ejecución de dichos comandos. Ambas tareas responden a un

problema clásico de software denominado *productor y consumidor*. El problema consta de tres actores, un productor de datos, un consumidor de los datos producidos, y un medio de comunicación entre ambos, que transporta los datos de un lado al otro. Típicamente el medio de comunicación es una cola de datos. El problema suma complejidad cuando los productores y consumidores de datos son múltiples (en este caso múltiples productores (sensores) y un consumidor (microcontrolador), dado que es posible que más de un productor o consumidor intenten acceder simultáneamente a la cola. La solución clásica al problema consiste en utilizar mecanismos de exclusión mutua, lo que evita que más de un proceso intente escribir simultáneamente en la cola. Esta solución es bloqueante por lo tanto no se pueden utilizar en este caso, ya que los procesos se ejecutan dentro de interrupciones. Si la interrupción que posee el mecanismo de exclusión es interrumpida a su vez por un evento de mayor prioridad, entonces este último no podría completar su tarea sobre la cola. Esto implicaría que el sistema se bloquee indefinidamente.

La otra solución implica utilizar mecanismos no bloqueantes, los cuales pueden ser utilizados con interrupciones. Existen numerosas investigaciones sobre colas no bloqueantes en el campo de la programación concurrente [3][4]. Las soluciones propuestas hacen uso de hardware específico de los procesadores, mediante operaciones de acceso exclusivo. Los procesadores *ARM* incluyen operaciones de escritura, lectura, y borrado exclusivo para poder implementar este tipo de algoritmos. Son especialmente utilizadas en aplicaciones multinúcleo.

Si bien esta solución sería factible, las estructuras de datos no bloqueantes son complejas, y muchas veces más lentas que su versiones bloqueantes. En este caso se decidió utilizar otro mecanismo provisto por el microcontrolador para solucionar el problema: el controlador del vector de interrupciones enlazadas. Este controlador permite entre otras cosas, crear grupos de interrupciones con distintas prioridades y también acumular interrupciones con un sistema de colas propio.

Si durante la ejecución de una interrupción se dispara otra del mismo grupo, ésta no interrumpirá la primera, sino que el sistema pondrá en una cola a la misma y la ejecutará una vez finalizada la interrupción actual. Entonces las solución propuesta consiste en que todas las interrupciones que quieran escribir en la misma cola se encuentren en el mismo grupo. De esta manera no habrá peligro de que se corrompan los datos sobre la cola por culpa de un intento de acceso múltiple y el sistema se comportará como si tuviera un único productor.

Si el productor y el consumidor tratan de acceder simultáneamente al recurso, también pueden dañarse los datos. En este caso el consumidor se encontrará en el programa principal y los productores escribirán en la cola desde interrupciones. El objetivo es que si el consumidor está accediendo a la cola y la tarea es interrumpida por un productor que también quiere acceder a la cola, lo pueda hacer y los datos no se corrompan. El problema de una cola con un solo productor y un solo consumidor, es un caso particular que puede resolverse sencillamente por software de forma no bloqueante. La implementación usual es utilizar un buffer circular con dos índices distintos para la escritura y lectura. Luego, cada índice se actualiza una vez que la lectura o escritura se haya concretado según corresponda.

La solución propuesta permite lidiar con el problema de concurrencia en una forma simple, gracias a las herramientas de ARM. La desventaja es que cierto determinismo se pierde cuando un dato debe ser guardado para su posterior procesamiento. La marca de tiempo no se etiquetará hasta que la interrupción sea atendida.

Cada medición de los sensores debe ser etiquetada con una marca de tiempo precisa. De esta forma es posible utilizar los datos de los sensores de navegación en una computadora de más alto nivel (con mayor capacidad de cómputo) sin perder precisión en la marca de tiempo de los sensores. Además, la etiqueta de tiempo puede ser utilizada para la verificación de la integridad de los datos. En este caso se utiliza un *timer* de 32 bits para etiquetar las mediciones que llegan al microcontrolador. El *drift* del *timer* del microcontrolador puede degradar el desempeño del cálculo de navegación. En ciertas aplicaciones puede no ser apreciable, pero es notorio en aplicaciones que requieran una alta precisión en el sistema de navegación[1]. Es por esto que para compensar el *drift* se reinicia el *timer* cada vez que una señal de PPS llega. Dado que la señal de PPS proveniente del GPS es precisa es posible acotar el *drift* del *timer* del microcontrolador. Si no hay señal de PPS, por ejemplo en ambientes cerrados en donde la señal de GPS no está disponible, no es posible evitar el *drift* del *timer*, sin embargo se supone que los vuelos en ambientes cerrados no son tan prolongados como para afectar significativamente el desempeño del sistema de navegación. Si la aplicación lo requiriese, se podría poner una referencia de tiempo precisa.

IV. APLICACIÓN A UN VEHÍCULO MULTIROTOR

El propósito de esta sección es mostrar el desempeño de la computadora de NG&C de bajo nivel, en un vehículo aéreo no tripulado del tipo multirotor. Si bien la computadora es flexible y puede ser utilizada en diversas aplicaciones, siempre es necesario emplear técnicas de navegación, guiado y control ad-hoc para la aplicación en la que se utilizará. En particular en este ejemplo de aplicación, se supone que el multirotor se encuentra cerca de la zona de *hovering*. Esta suposición se utilizará para estimar la orientación del vehículo y así poder efectuar el control del mismo. Como se ha mencionado anteriormente, el algoritmo de control impone las restricciones

[1] Cabe aclarar que puede no ser el caso de un vehículo multirotor dado que el tiempo de vuelo del mismo es corto como para que este *drift* sea apreciable, pero en vuelos prolongados el impacto de este *drift* puede ser notorio. Esto también depende de la calidad de los sensores utilizados, por ejemplo esta plataforma puede ser empleada con sensores inerciales MEMS de alta precisión, por ejemplo la gama más alta de la serie ADIS de Analog Devices.

más fuertes de tiempo real y, esto hace que al menos una parte del sistema de navegación deba ejecutarse con restricciones tan fuertes como las del control. En un lazo de más alto nivel (y corriendo en otro procesador) habrá un sistema de navegación (y posiblemente un sistema de control) que se encarguen de obtener información de navegación de mejor calidad (y de mejorar el desempeño del sistema de control) corriendo con restricciones no tan fuertes sobre sus tiempos de ejecución.

En la Fig. 3 se presenta la definición para los ángulos de *pitch* (θ), *roll* (ϕ), and *yaw* (ψ) necesarios para efectuar el control del multirotor. Estos ángulos se estiman a partir de la medición de los sensores inerciales (suponiendo válida la hipótesis que el vehículo está cercano a la zona de *hovering*). Dado que los sensores MEMS presentan importantes ruidos y sesgos elevados es necesario utilizar técnicas de filtrado que permitan mitigar los efectos de estos errores. Si bien es cierto que los sensores inerciales se ven afectados por otras fuentes de error como el *rate ramp random walk*, los factores de escala, la falta de ortogonalidad entre ejes, no-linealidades, entre otros, son factores que afectan apreciablemente a este tipo de sensores. Se hace imposible contemplar todos estos parámetros en un filtro simple que pueda ejecutarse en tiempo real en un microcontrolador como el Cortex M3. Estos factores adicionales son estimados y compensados en un filtro de más alto nivel, haciendo que a bajo nivel sólo el sesgo y ruido blanco sean los factores a tener en cuenta como los errores de los sensores inerciales.

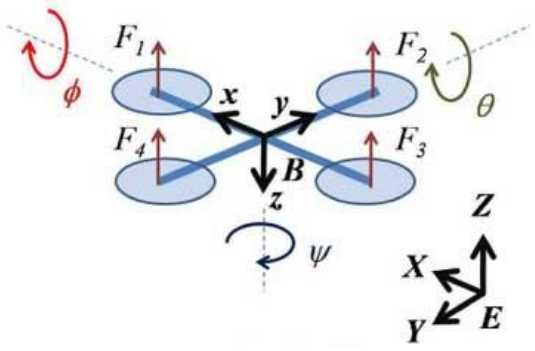

Fig. 3 Definición de los ángulos de pitch, roll y yaw para el sistema de NG&C

En los ensayos que han sido realizados los acelerómetros presentaron un sesgo con una componente sistemática considerablemente estable, por lo que es posible realizar una calibración en el laboratorio para cada sensor y compensarla en el microcontrolador. En cambio los giróscopos presentan inestabilidades del sesgo considerables por lo que es necesaria una calibración al inicio de cada experimento. Esto no presenta mayor dificultad dado que en la mayoría de los experimentos el vehículo se inicializa con velocidad angular nula (despreciando la rotación terrestre que no puede ser medida por este tipo de instrumentos).

El primer filtro que se utiliza es un pasa-bajos (simplemente un promediado pesado de muestras) para disminuir el efecto del ruido de los sensores. Teniendo en cuenta que la tasa de adquisición de los sensores inerciales estaba fijada en 800Hz, se promedian 10 muestras para lograr una frecuencia del control de 80Hz, que mostró ser una frecuencia aceptable para realizar el control del vehículo.

Cabe destacar que previo al filtro digital implementado, la placa de NG&C se instaló sobre un amortiguador mecánico con el fin de reducir la incidencia de las vibraciones en las mediciones de los acelerómetros. Estas vibraciones se hacen particularmente apreciables cuando el vehículo no está bien balanceado (las hélices están en mal estado o los motores no están bien centrados). En la Fig. 4 se puede apreciar las mediciones de los acelerómetros, normalizadas a la gravedad donde **g** es la aceleración de la misma ($9.8 m/s^2$).

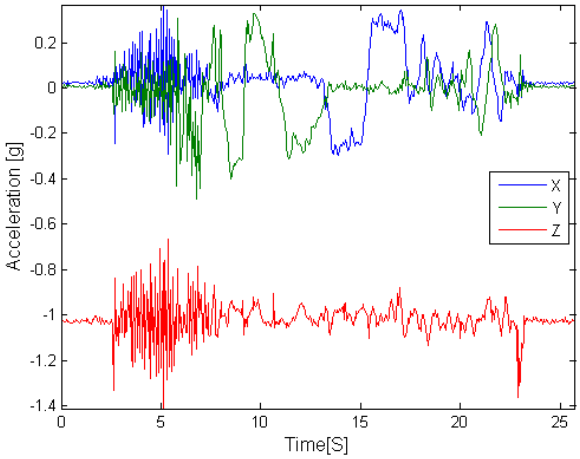

Fig. 4 Mediciones de los acelerómetros en uno de los ensayos realizados con el vehículo multirotor

Un segundo filtro se utiliza para obtener una estimación de ángulo. Este filtro es un esquema clásico de un filtro compensador, el cual basa su funcionamiento en tres hechos: a) la suposición de que el vehículo se encuentra en posición de *hovering*, de esta manera los acelerómetros aportan información de ángulo comparando la fuerza específica medida respecto de la gravedad local (que se supone conocida), b) los acelerómetros proveen buena información en bajas frecuencias (la fuerza específica se mantiene constante) pero no en alta (debido a que están afectados por mucho ruido y vibraciones), c) los giróscopos son buenos para dar información angular en altas frecuencias, es decir en cortos períodos de tiempo (integrando la medición de velocidad angular) pero no en bajas frecuencias (debido a la deriva en el cálculo que aporta el sesgo). De esta forma se filtra la información de los acelerómetros mediante un filtro pasa bajos y con su filtro complementario (un pasa altos) la información de los giróscopos.

En la ecuación (1) se muestra una estimación de los ángulos de pitch y roll utilizando solamente las mediciones de los acelerómetros, donde θ y ϕ se suponen lo suficientemente

pequeños como para considerar sen(θ)≈θ y sen(ϕ)≈ϕ. Los factores a_i se expresan en veces la fuerza de gravedad **g**.

$$\theta_a = \frac{a_x}{a_x^2 + a_y^2 + a_z^2};\ \phi_a = \frac{a_y}{a_x^2 + a_y^2 + a_z^2} \qquad (1)$$

Combinando esta estimación con la integración de la información provista por los giróscopos se obtiene la ecuación (2) expresada para el caso del ángulo de pitch. El término $\omega_{y_{acc}}$ representa la integral de ω_y y α es un factor que se relaciona con el tiempo entre muestras (12.5μS at 80Hz) y la frecuencia de corte $1/T_f$ para el filtro de las mediciones de los acelerómetros (3), en este experimento 1Hz. De esta forma α resulta aproximadamente 0.02.

$$\hat{\theta} = (1 - \alpha) * \left(\omega_{y_{acc}} + T_s * \omega_y\right) + \alpha * \theta_a \qquad (2)$$

$$\alpha = \frac{T_s}{T_s + T_f} \qquad (3)$$

Respecto al algoritmo de control de bajo nivel implementado, se basa en un control de *pitch* y *roll* implementados utilizando dos lazos PID, como se muestra en la Fig. 5. Para el control de yaw se realiza un procedimiento similar, pero utilizando las mediciones del magnetómetro en lugar de las de los acelerómetros.

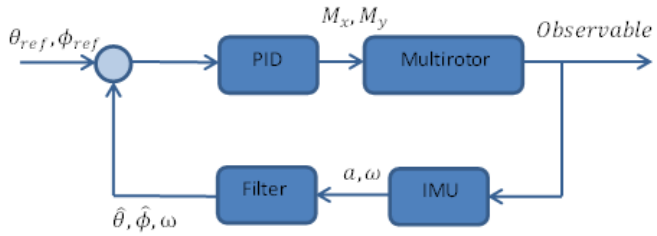

Fig. 5 Lazo de control de pitch y roll.

Las referencias para el control PID son los ángulos de *pitch* y *roll*, y las entradas son los errores en los ángulos respecto de la estimación del filtro complementario (parte proporcional), la integral del error del ángulo (parte integral) y la velocidad angular (parte diferencial). El sistema contempla además el uso de la derivada de la velocidad angular para mejorar el desempeño respecto a perturbaciones de aerodinámicas [5], aunque no llego a utilizarse en las pruebas que se presentan aquí. La frecuencia adoptada definitiva fue de 80Hz, mediante experimentos de campo donde se evaluó el desempeño del vehículo, en lo que resulta un tiempo entre ejecuciones del lazo de 12.5 μS.

Para garantizar el correcto funcionamiento del lazo de control, el tiempo entre muestras debe mantenerse constante. Mientras las tareas de adquisición de los sensores se ejecutan en diversos niveles de prioridad de interrupción, el lazo de control corre en el *main*, o en la prioridad más baja, con lo cual dicha rutina puede ser (y será) interrumpida por otras tareas. Por ello se realizaron diversas pruebas para comprobar que el tiempo entre diferentes ejecuciones se mantenga dentro de ciertos límites.

En la Fig. 6 y 7 se exhiben los resultados donde el eje y representa el tiempo entre ejecuciones del lazo de control para dos de los experimentos realizados.

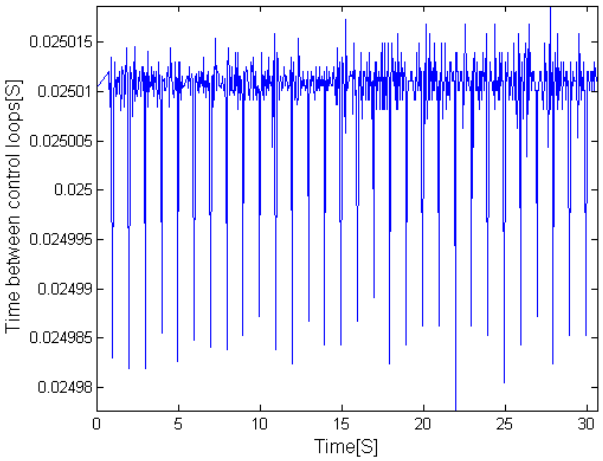

Fig. 6 Tiempo entre ejecuciones del lazo de control a 40Hz

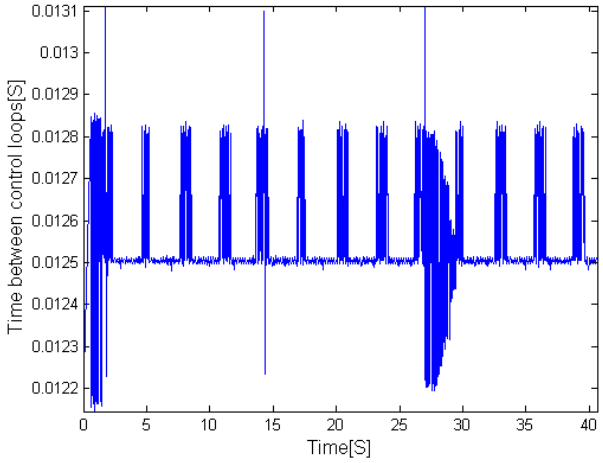

Fig. 7 Tiempo entre ejecuciones del lazo de control a 80Hz

Para el lazo ejecutado a 40Hz, el valor medio del tiempo entre ejecuciones es de 25.0096μS, llegando a un máximo de 25.0172μS y a un mínimo de 24.9776μS, lo cual significa que, en el peor caso medido, la variación porcentual del *timing* es de 0.16%.

Por otro lado, al ejecutar el lazo a 80Hz, el valor medio del tiempo cambia a 12.5469μS, llegando a valores máximos y mínimos de 12.8632μS y 12.1569μS. En este caso, entonces, la variación es del 5.65%, sin tomar en cuenta los picos ocasionales.

Claramente, al incrementar la tasa de adquisición de datos de la IMU de 400Hz a 800Hz se permite tener mayor cantidad de datos para mejorar la estimación de los ángulos de navegación

y/o la frecuencia y la calidad del controlador, pero al costo de deteriorar la precisión en tiempo de la ejecución del lazo de control. En la Fig. 6 pueden apreciarse períodos de tiempo donde el tiempo entre ejecuciones aumenta, producto de grandes cantidades de datos que deben ser leídos y tienen mayor prioridad, además de otras tareas que pueden ser retrasadas para atender la lectura.

V. DESARROLLO FUTURO

Actualmente se está desarrollando una nueva versión de la computadora de NG&C, contemplando diversas limitaciones y mejoras que surgieron durante las pruebas del diseño presentado.

Hasta el momento las mejoras implementadas se relacionan con modificaciones al hardware existente y el diseño físico para que sea compatible con la plataforma de desarrollo BeagleBone Black. Las mejoras al software serán analizadas una vez que se disponga de un nuevo modelo armado, entre las cuales se analizan la utilización de un sistema RTOS, sus ventajas y limitaciones.

VI. CONCLUSIONES

En este trabajo se resumen los resultados obtenidos respecto a los timings de una computadora de NG&C implementada en un microcontrolador Cortex-M3 LPC1769 que ejecuta las tareas de adquisición de datos de diversos sensores para obtener las variables de orientación del sistema y utilizarlas en tres lazos PID para estabilizar al vehículo.

El determinismo necesario para la ejecución de dichos lazos es alcanzado de forma satisfactoria en el firmware desarrollado, garantizando una variación del tiempo de ejecución menor al 6%; dicho error es debido a las otras tareas de mayor prioridad que se ejecutan en el microcontrolador y demoran al lazo de control.

En ensayos posteriores a la presentación original de este trabajo, se cambiaron ligeramente las funciones que se ejecutaban mayor cantidad de veces, como la lectura de la IMU a 800Hz, reemplazando funciones de *C* con la lectura y escritura utilizando el acceso a registros, con lo cual se logró una variación de tiempo de ejecución menor al 1% y un diagrama de tiempos similar al de la Fig. 6. Esto representa una gran mejora respecto del paso previo, sin embargo, ya que la computadora de NG&C se concibió como una plataforma de desarrollo, es conveniente el uso de funciones de *C* para mejorar la comprensión en la lectura y modificación del firmware.

REFERENCIAS

. [1] M. Ella, A. R. M. Sanner, QoS tradeoffs for guidance, navigation, and control, Aerospace Conference Proceedings, 2002. IEEE, Vol. 7.

. [2] Kai-Wei Chiang, Thanh Trung Duong, and Jhen-Kai Liao, The Performance Analysis of a Real-Time Integrated INS/GPS Vehicle Navigation System with Abnormal GPS Measurement Elimination, Sensors (Basel). Aug 2013; 13(8): 10599–10622.

. [3] M. M. Michael and M. L. Scott, “Simple, Fast, and Practical Non-Blocking and Blocking Concurrent Queue Algorithms,” PODC 9´6 Proceedings of the fifteenth annual ACM symposium, pp. 267–275, 1996.

. [4] L. Groves, “Verifying Michael and Scott’s lock-free queue algorithm using trace reduction,” Proceeding CATS ’08 Proceedings of the fourteenth symposium on Computing: the Australasian theory, vol. 77, pp. 132–142, 2008.

. [5] GM Hoffmann, SL Waslander, CJ Tomlin. "Quadrotor helicopter trajectory tracking control", AIAA Guidance, Navigation and Control Conference, 2008