Diseño e Implemantación de un Cuadricóptero de Vuelo Autónomo

Alan Kharsansky, Federico Roasio, Ezequiel Espósito,
Claus Rosito, Daniel Schermuk & Ariel Lutenberg
Laboratorio de Sistemas Embebidos,
Facultad de Ingeniería
Universidad de Buenos Aires
lse@fi.uba.ar

***Resumen*—Se presenta el diseño y la implementación de un cuadricóptero como plataforma voladora de investigación y desarrollo en temas de electrónica, sistemas embebidos y control. Si bien en los últimos años ha habido muchos trabajos referidos a cuadricópteros, en general se centraban en aplicaciones sobre cuadricópteros comerciales. En cambio, el foco del presente trabajo está puesto en obtener un sistema abierto y escalable que sirva como base para futuros desarrollos.**

Para eso se siguieron tres líneas de trabajo paralelas: el diseño mecánico está realizado de forma modular, la implementación en hardware se hizo utilizando productos comerciales off the shelf con su debida especificación, y el software de control se separó en diferentes capas de abstracción.

De esta manera se logró una plataforma repetible, ya que al estar toda la información disponible cualquier laboratorio, grupo o individuo puede armar uno o más cuadricópteros de características similares. Y por otro lado, también se apunta a que sea una plataforma común de desarrollo, donde varios grupos trabajando de manera concurrente sobre diferentes aspectos de la mecánica, el hardware y el software del cuadricóptero se complementen y amplíen las posibilidades y el alcance de la plataforma.

***Index Terms*—uadrotor, Control, Telemetría, Navegación inercial uadrotor, Control, Telemetría, Navegación inercial Q**

I. Introducción

Gracias a los avances de los últimos años en diferentes tecnologías, principalmente en motores brushless, sensores mems, baterías y microprocesadores, los cuadricopteros (también llamados çuadrotores", o "quadrotor.[en] inglés) se han popularizado entre hobbyistas y grupos de investigación en áreas de control y navegación.

Si bien algunos cuadricópteros de hobbistas [1] tienen interfases directas para ser controlados por teléfono celular, la mayoría son vehículos a control remoto, y dependen de la capacidad del piloto para ser comandados desde tierra y seguir trayectorias o realizar acrobacias.

En cambio, en varios grupos de investigación se está trabajando sobre el control de a bordo del cuadricóptero, es decir que la misma computadora de abordo es responsable en cierto grado de la navegación y el guiado, comportándose como un vehículo autónomo, es decir que no depende de los comandos enviados remotamente para volar. En la univerisdad de Stanford [2] y Pennsylvania [3], por ejemplo, usan sistemas de visión para la navegación, mientras que para el guiado simulan la computadora de vuelo en tierra, mandándole los comandos directamente al cuadricóptero.

En general, la mayor parte de los trabajos publicados con respecto a los cuadricópteros hacen uso de unidades comerciales, centrándose en los aspectos de los algoritmos de control y áreas relacionadas [?] [?], o bien en aplicaciones como por ejemplo adquisición de imágenes y su respectivo procesamiento [?], relegando los aspectos de la ingeniería del cuadricóptero a segundo plano.

Desde el Laboratorio de Sistemas Embebidos de la Facultad de Ingeniería de la Universidad de Buenos Aires abordamos el problema de diseñar e implementar un cuadricóptero de vuelo autónomo partiendo desde cero, con la visión de que esto nos permitiría ganar concocimiento y control sobre cada una de las partes que componen el sistema. A su vez, vimos que es un plataforma muy rica para generar desarrollos en cada uno de sus subsistemas, mejorando la performance del conjunto y abriendo nuevas áreas de trabajo.

En particular, se trabajo con el objetivo de crear una plataforma abierta y escalable, que le permita a cualquier individuo o grupo empezar a trabajar en el área nutríendose de los desarrollos presentados y generando un marco para el trabajo en conjunto sobre mejoras y avances. En cuanto al aspecto abierto, se logró utilizando un sistema de documentación tipo wiki, de acceso público, de manera que la información esté disponible universalmente para los individuos o grupos que quieran ponerse a trabajar en la plataforma. Y el aspecto escalable se logró separando el sistema en subsistemas modulares, de forma que cada subsistema puede ser cambiado y mejorado individualmente sin perjudicar el funcionamiento del conjunto. Se tuvo especial cuidado en la interconección de la computadora de abordo con los sensores y en el desarrollo de una arquitectura de capas de abstracción para la computadora de a bordo, que logra la independencia progresiva del software con respecto al hardware.

La estructura del presente trabajo es la siguiente: primero se describe el diseño y la implementación mecánica y de hardware, luego la arquitectura del software tanto de tierra como de abordo, luego se resume el estado actual del desarrollo y por último se presenta una breve descripción de las líneas de trabajo que se desprenden del presente trabajo.

II. Arquitectura de Hardware

Para el desarrollo del Quadrotor se comenzó por el diseño de la mecánica estructural del mismo. Considerando que la plataforma debe ser fácilmente reproducible, se optó por utilizar planchas de corte y una cortadora láser. Esta decisión impactó fuertemente en las posibilidades de diseño, prohibiendo el uso de estructuras tridimensionales y limitando la fabricación a piezas planas. Para la forma del vehículo se propuso una geometría simétrica con dos ejes largos y un octógono central con el fin de colocar la electrónica en el mismo. Teniendo el objetivo de que el proceso de armado del Quadrotor fuera lo más simple posible, se propuso un sistema de encastre, lo que evita grandes cantidades de elementos de sujeción, reduciendo el peso y a su vez aportando

a la simplicidad del diseño. Teniendo en cuenta la restricción en el diseño de las piezas, se elaboró el armazón mostrado en la Figura ¿?.

Para el diseño de la electrónica de abordo, se analizaron los requerimientos mínimos para que la plataforma fuera funcional. Como resultado de este análisis, se concluyó que el sistema debería constar de 4 partes:

- Microcontrolador Principal
- Sensores de Navegación
- Sistema de Comunicación
- Controlador de Velocidad para Motores Brushless

El microcontrolador principal elegido fue el LPC1769 de NXP Semiconductors, el cual cuenta con un núcleo ARM Cortex M3. Este microcontrolador es capaz de ejecutar el algoritmo de control de orientación del vehículo y dispone de un tiempo ocioso de más del 80 %, lo cual permite la implementación de funcionalidades adicionales sin modificar la estructura de abordo. Para la elección de los sensores de navegación se consideró que sería interesante que el microcontrolador recibiera información procesada acerca de la orientación del Quadrotor. Esta elección resultó en la necesidad de utilizar un segundo procesador para el análisis y filtrado de los sensores. Dados estos requerimientos, se agregó la necesidad de evitar las derivas temporales, ocasionadas por los sensores inerciales relativos, tales como los acelerómetros y giróscopos, los cuales aportan información relativa al cambio de posición y no a la posición absoluta respecto de un eje de referencia dado. Considerando este último requisito y teniendo en cuenta que el Quadrotor posee un gran espectro de utilidad en recintos cerrados, se descartó el uso de GPS, optando por el uso de magnetómetros, los cuales brindan información acerca de la posición del vehículo respecto al campo magnético del planeta. Finalmente se optó por el desarrollo Razor 9DoF (Degrees of Freedom) IMU (Inertial Measurement Unit), el cual cuenta con sensores de aceleración, giróscopos y magnetómetros alineados para la medición de las magnitudes en una terna ortogonal directa. Esta placa cuenta además con un ATmega328 para el procesamiento de los sensores.

En cuanto a los requerimientos del sistema de comunicación física, se estableció la necesidad de un control de colisiones por parte del hardware, simplificando la programación del microcontrolador principal y liberando tiempo de cálculo para los algoritmos de control y de funcionalidades adicionales. Dado este requisito y estableciendo un rango de utilización de unos pocos cientos de metros, se optó por utilizar los módulos XBee Pro de Digi International, los cuales poseen un control de colisión de paquetes tienen una comunicación sencilla con el microcontrolador, por puerto serie. Esta elección se vio respaldada por la disponibilidad local de los módulos, favoreciendo a la posibilidad de repetición de la plataforma. Para la elección de los controladores de velocidad de los motores, se optó por la utilización de ESC (Electronic Speed Control) para modelismo. Los mismos son capaces de proporcionar la corriente necesaria para el manejo de los motores y poseen un control del tipo PWM, lo cual aporta a la simplicidad de uso.

Una vez definidos los módulos a utilizar, se diagramó un esquema de interconexión entre los mismos. Dado que el ATmega328 de la placa de sensores se comunica por puerto serie, se designó la conexión del mismo hacia el microcontrolador por ese puerto. Análogamente se fijó la conexión entre el módulo de comunicaciones y el microcontrolador por otro puerto serie. Por último los controladores de velocidad se conectaron a pines del microcontrolador capaces de utilizar PWM por hardware.

III. Arquitectura de Software

Recordando que uno de los principales objetivos del proyecto es la capacidad de reproducción y la escalabilidad del mismo, resulta de vital importancia la independización entre el software y el hardware. Esto permitirá futuras actualizaciones de hardware, con pequeños retoques en el software. Con el fin de lograr este objetivo, se diagramó una estructura de software embebido guiado por la siguiente estructura:

- HAL (Hardware Abstraction Layer)
- API (Hardware Abstraction Layer)
- Aplicación

El HAL es la capa encargada de controlar el hardware, siendo la única específica para el mismo y aportando funciones de manejo de los periféricos de microcontrolador. Esta capa es la única que tendrá acceso directo al hardware y es la única que se deberá modificar ante un eventual cambio del mismo.

La API es la capa que provee al usuario los métodos de programación. Esta capa es intermedia entre el usuario y el HAL. Esta capa no tiene acceso al hardware sino que se ve limitada al uso de las funciones otorgadas por el HAL.

La capa de aplicación es la cual programa el usuario. Esta capa hace uso solamente de la API, no pudiendo acceder directamente al HAL o al hardware. La aplicación será el algoritmo de control, navegación o cualquier otra funcionalidad que el usuario quiera agregar.

Esta división por capas evidencia una mayor complejidad en la programación de las mismas, pero aporta a la portabilidad del código escrito, independizándolo del hardware utilizado. Esta independencia nos permite la migración entre plataformas diferentes, sin la necesidad de reescribir completamente el programa. Esta estructura favorece, además, a la simplicidad de la escritura del código por parte del usuario, utilizando funciones que elevan el nivel de abstracción de la programación.

Como herramienta adicional para aumentar la fiabilidad de las comunicaciones, se desarrolló un protocolo capaz de detectar errores en las comunicaciones y de transportar paquetes de información entre nodos. Este sistema se subdividió en 4 capas: Aplicación, Transporte, Enlace, Física.

La capa física es íntegramente controlada por los módulos XBee, dejando los modos de direccionamiento a las capas de nivel superior.

La capa de transporte se encarga de que tanto los datos salientes como entrantes cumplan con un determinado formato. Este conformación del paquete permite la detección de errores y la separación de paquetes en función de su utilidad.

En la Figura ?? se muestra el formato de la trama establecida por el protocolo implementado. En la misma se aprecia que la carga útil de la misma es de 0-255Bytes, dejando encargada del fragmentado de los paquetes a la capa de aplicación.

La capa de transporte es además encargada de verificar la trama mediante las direcciones de fuente y destino, numero de secuencia y checksum. Dado que el protocolo permite interconexión entre diferentes nodos, en ambos sentidos, se requiere un tipo de dato que especifique el carácter de la transmisión (control, telemetría, uplink, downlink).

La capa de aplicación implementa la utilidad del protocolo de comunicaciones. Es posible implementar has 256 tipos diferentes de datos, diferenciándolos en su propósito. Los tipos de datos implementados hasta la fecha son: • System: transmite comandos de sistema, alternando entre modos de funcionamiento del vehículo. • Control: transmite información acerca del control de vuelo, tales como cambio en la posición, navegación, acciones, etc. • Debug: transmite mensajes de baja prioridad, a interpretarse desde la consola terrestre como caracteres ASCII. • Telemetry: transmite paquetes de datos relacionados con la telemetría.

Como es apreciable de la descripción del protocolo de comunicaciones, el canal estará inundado por diferentes tipos de paquetes, transportando información que será útil a distintos puestos de trabajo. Dado que estos puestos pueden estar ubicados

en diferentes locaciones, se implementó un servidor de mensajes, denominado RadioServer. Este servidor se encarga de canalizar las comunicaciones entre los dispositivos de tierra y el Quadrotor. Ante el arribo de un mensaje, se direccionará el mismo al usuario en cuestión, utilizando puertos TCP. Esta estructura de comunicaciones brinda una gran flexibilidad en cuanto al espectro de utilización del dispositivo, pudiendo desplegar simultáneamente funciones en campo y en laboratorio. En la Figura ¿? se ilustra el funcionamiento del servidor.

IV. Conclusion

The conclusion goes here.

Acknowledgment

The authors would like to thank...

Referencias

[1] "Parrot." [Online]. Available: http://www.parrot.com/

[2] G. Hoffmann, D. Rajnarayan, S. Waslander, D. Dostal, J. Jang, and C. Tomlin, "The stanford testbed of autonomous rotorcraft for multi agent control (starmac)," in *Digital Avionics Systems Conference, 2004. DASC 04. The 23rd*, vol. 2, oct. 2004, pp. 12.E.4 – 121–10 Vol.2.

[3] N. Michael, D. Mellinger, Q. Lindsey, and V. Kumar, "The grasp multiple micro-uav testbed," *Robotics Automation Magazine, IEEE*, vol. 17, no. 3, pp. 56 –65, sept. 2010.