

Diseño e Implementación 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. 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 componentes disponibles comercialmente 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—“Cuadricóptero”, Control, Telemetría, Navegación inercial

I. INTRODUCCIÓN

Gracias a los avances de los últimos años en diferentes tecnologías, principalmente en motores sin escobillas (“brushless”), sensores MEMS, baterías y microprocesadores, los cuadricópteros (también llamados “cuadrotores”, o “quadrotor” en inglés) se han popularizado entre hobbyistas y grupos de investigación en áreas de control y navegación.

Si bien algunos cuadricópteros de hobbyistas [1] tienen interfaces directas para ser controlados por teléfono celular, la mayoría son vehículos a radio control, y todos 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 de modo que la misma computadora de a bordo 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 universidad 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.

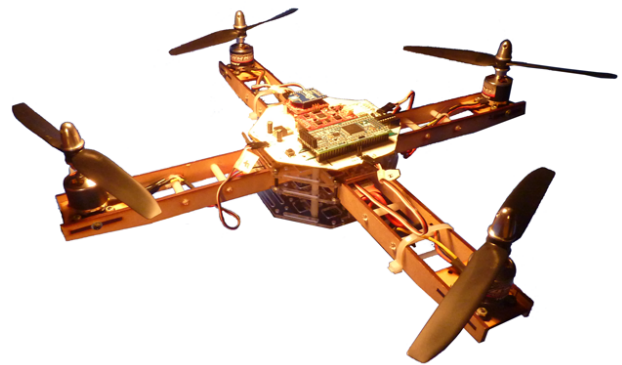


Figura 1. Prototipo del cuadricóptero

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 [2] [3], o bien en aplicaciones como por ejemplo adquisición de imágenes y su respectivo procesamiento [3], 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 conocimiento y control sobre cada una de las partes que componen el sistema. A su vez, vimos que es una 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 trabajó con el objetivo de crear una plataforma abierta y escalable, que le permita a cualquier individuo o grupo empezar a trabajar en el área nutriéndose 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. 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 interconexión de la computadora de a bordo 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 a bordo, 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. DISEÑO MECÁNICO

Para el desarrollo del cuadricóptero se comenzó por el diseño de su mecánica estructural. Para que la plataforma sea fácilmente reproducible, se optó por utilizar planchas de corte y una cortadora láser. Esta decisión impactó fuertemente en las posibilidades de diseño, evitando 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. Con el objetivo de que el proceso de armado del cuadricóptero 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. Una foto del prototipo se puede ver en la Fig. 1.

III. ARQUITECTURA DE HARDWARE

Para el diseño de la electrónica de a bordo, 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 subsistemas principales:

- 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 actitud del vehículo y dispone de un tiempo ocioso estimado en más del 80 %, lo cual permite la implementación de funcionalidades adicionales sin modificar la estructura de a bordo.

Para la elección de los sensores de navegación se consideró que sería interesante que el microcontrolador recibiera información pre-procesada acerca de la actitud del cuadricóptero. Esta elección resultó en la necesidad de utilizar un segundo procesador para el análisis y filtrado de estos sensores. Dados estos requerimientos, se agregó la necesidad de evitar las derivas temporales, ocasionadas por los sensores

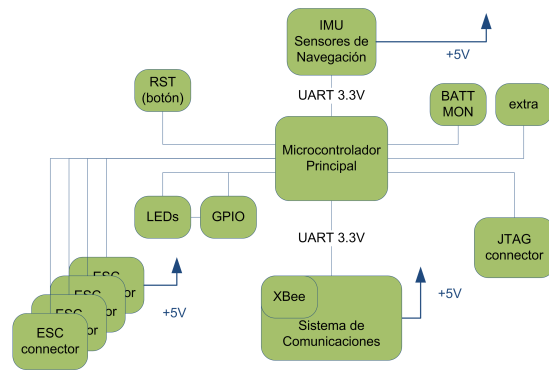


Figura 2. Esquema de interconexión

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 cuadricóptero 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) [4], 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. En la Figura 2 se muestra un esquema de conexionado de las partes.

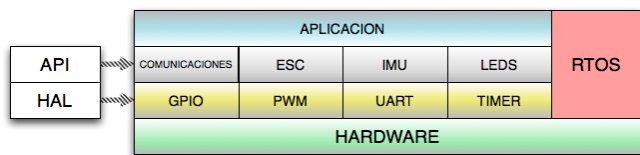


Figura 3. Estructura de software

HEADER	SRC	DST	TIME/SEQ	TYPE	LEN	PAYLOAD	CHECKSUM
1B	1B	1B	1B	1B	1B	0-255B	1B

Figura 4. Trama del protocolo de comunicaciones

Considerando los costos de mecánica y de electrónica se estimó un costo de fabricación estimado de 400 dólares.

IV. ARQUITECTURA DE SOFTWARE

Dado que uno de los principales objetivos del proyecto es que éste sea de fácil reproducción y escalabilidad, resulta de vital importancia lograr independencia 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 compuesto por las siguientes partes:

- HAL (Hardware Abstraction Layer)
- API (Application Programming Interface)
- 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 implica una mayor complejidad en la programación de las mismas, pero facilita la portabilidad del código escrito, independizándolo del hardware utilizado, como se mencionó anteriormente. Esta independencia nos permite la migración entre plataformas diferentes, sin la necesidad de reescribir completamente el programa. Esta estructura favorece, además, la escritura del código por parte de terceros, utilizando funciones que elevan el nivel de abstracción de la programación.

En la Figura 3 se muestra un diagrama de la implementación en las capas previamente descritas.

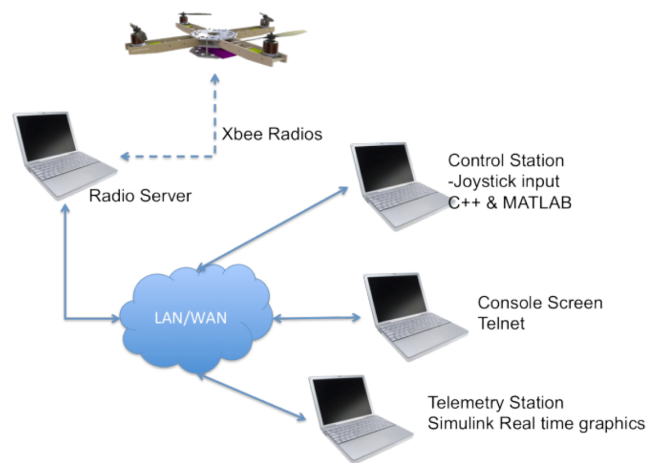


Figura 5. Esquema de funcionamiento del servidor

V. ARQUITECTURA DE COMUNICACIONES

Para aumentar la confiabilidad y versatilidad del sistema se desarrolló una arquitectura completa para las comunicaciones. Para esto se diseñó un protocolo capaz de detectar errores en las comunicaciones y de transportar paquetes de información entre nodos. Este subsistema se subdividió en cuatro capas: Aplicación, Transporte, Enlace, Física.

Las capas físicas y de transporte son íntegramente controladas 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. El módulo de comunicaciones de la capa API de la estructura de software se ocupa de este rol mediante la implementación de una máquina de estados. La comunicación con los módulos XBee la realiza a través del módulo UART de la capa HAL de la estructura de software.

En la Figura 4 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, número 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). El contenido del payload es presentado a la aplicación de la capa superior que requiere este servicio mediante un modelo publisher/subscriber.

La capa de aplicación implementa la utilidad del protocolo de comunicaciones. Es posible implementar hasta 256 tipos diferentes de datos, diferenciándolos en su propósito. Los tipos de datos implementados actualmente 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.

VI. IMPLEMENTACIÓN DEL SERVIDOR DE MENSAJES

Como se aprecia en 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 cuadricóptero. 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 5 se ilustra el funcionamiento del servidor, junto con algunos ejemplos de clientes que se comunican con el cuadricóptero.

VII. TRABAJO FUTUROS

A partir del presente desarrollo queda abierta la posibilidad de comenzar con trabajos sobre temas más puntuales de la ingeniería y operación del cuadricóptero.

Una línea de trabajo es la mejora mecánica de la plataforma, alivianándola y aumentando la carga útil del cuadricóptero. Esto abre la puerta a nuevas aplicaciones, ya que se pueden transportar más sensores y subsistemas a bordo, como por ejemplo cámaras de fotos, sensores de radiación, nodos de comunicaciones, etc.

Otra línea de trabajo es la mejora de los algoritmos de navegación, lo que le daría más precisión y confiabilidad a la ubicación del cuadricóptero en el espacio y redundaría en más utilidad para los sensores, ya que se contaría con más precisión acerca de dónde fue obtenido el dato. Y también implicaría una mejora en el control del cuadricóptero, de la mano de una mejora en los algoritmos de control de actitud y guiado.

Los algoritmos control de actitud y de guiado también son materia de desarrollo actualmente, ya que una mejora en el control de actitud traería asociado más calidad en los datos obtenidos por los sensores, en particular los ópticos como cámaras de fotos, y posibilitaría la autonomía total de la aeronave.

Así mismo se desprende una línea de trabajo consistente en vuelos en formación de más de un cuadricóptero, en la que los controles de las diferentes aeronaves se coordinan entre sí para lograr formaciones estables. Una posible aplicación

del vuelo en formación es cubrir un área a sensar con muchos cuadricópteros en un tiempo corto.

Por último, el segmento terreno, es decir los sistemas en tierra para planificar los vuelos, darle las indicaciones necesarias al cuadricóptero, y procesar los datos recolectados también está abierto a desarrollarse, ya que actualmente solo están implementados los programas necesarios para controlar mínimamente a la aeronave y para mostrar la telemetría recibida.

Actualmente se están desarrollando estas líneas de trabajo tanto en el Laboratorio de Sistemas Embebidos como en el Grupo de Procesamiento de Señales, Identificación y Control de la Facultad de Ingeniería, Universidad de Buenos Aires.

CONCLUSIÓN

Se obtuvo un cuadricóptero totalmente documentado y abierto, que sienta las bases para futuros trabajos de investigación y desarrollo en áreas de control, electrónica y sistemas embebidos. Su naturaleza abierta, repetible y escalable lo convierte en una plataforma ideal para la adaptación relativamente rápida a posibles aplicaciones de sensado y reconocimiento.

AGRADECIMIENTOS

Los autores quieren agradecer al Dr. Ing. Juan I. Giribet y al Sr. David Vilaseca por sus valiosos consejos y aportes.

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.
- [4] "9 degrees of freedom - razor imu - ahrs compatible." [Online]. Available: <http://www.sparkfun.com/products/9623>