Diseño de un autopiloto para pequeños vehículos no tripulados

Design of an autopilot for small unmanned vehicles

Leonardo Garberoglio*[1], Claudio Pose†[2], Ignacio Mas†‡§[3] and Juan Giribet†‡[4]
*Grupo de Estudio de Sistemas de Control, Facultad Regional San Nicolás, Universidad Tecnológica Nacional
Buenos Aires, Argentina
[1]lgarberoglio@frsn.utn.edu.ar
†Grupo de Procesamiento de Señales, Identificación y Control, Facultad de Ingeniería, Universidad de Buenos Aires
Paseo Colón 850, Ciudad Autónoma de Buenos Aires, Argentina
[2]cldpose@fi.uba.ar
[3]imas@fi.uba.ar
‡Consejo Nacional de Investigaciones Científicas y Técnicas
Godoy Cruz 2290, Ciudad Autónoma de Buenos Aires, Argentina
[4]jgiribet@conicet.gov.ar
§Centro de Sistemas y Control - Instituto Tecnológico de Buenos Aires
Madero 351, Ciudad Autónoma de Buenos Aires, Argentina

***Resumen*—En este trabajo se presenta el diseño de un autopiloto para pequeños vehículos autónomos. El trabajo está enfocado principalmente en la arquitectura del sistema, pero se presentan también algunos detalles del firmware desarrollado para el autopiloto. En particular, el firmware incluye el paquete *rosserial*, que permite una conexión simple con ROS (Robot Operating System), resultando beneficioso para diversas aplicaciones.**

El desarrollo se valida experimentalmente en un vehículo aéreo no tripulado (UAV[1]) del tipo multi-rotor y en un vehículo acuático de superficie (ASV[2]). Estos vehículos han sido desarrollados por nuestro grupo, y en este trabajo puede encontrarse información sobre estos proyectos, lo cual puede ser de utilidad para los interesados en el desarrollo de UAV o ASV.

Palabras clave: Autopiloto, Vehículos no tripulados, Navegación Guiado & Control.

***Abstract*— In this work, the design of an autopilot for small unmanned vehicles is presented. It is focused mainly on the system's architecture, while presenting some important remarks of the developed firmware. In particular, the firmware uses *rosserial*, a package that allows a simple connection with ROS (Robot Operating System), which is useful for a number of applications.**

The work is validated experimentally on an multirotor-type unmanned aerial vehicle (UAV), and on an autonomous surface vehicle (ASV). These vehicles have been developed by our group, and this work provides information about such projects, which may be useful for those interested in the development of this kind of vehicles.

Keywords: Autopilot, Unmanned vehicles, Navigation Guidance & Control

I. Introducción

En los últimos años ha ido creciendo el interés en el diseño, construcción y despliegue de pequeños vehículos no tripulados. Los vehículos autónomos acuáticos (tanto los de de superficie (ASV) como los submarinos), los terrestres y los vehículos aéreos no tripulados (UAV), han demostrado ser de utilidad para diversas aplicaciones, logrando reducir costos operativos e incluso realizando tareas riesgosas, sin la necesidad de exponer a los pilotos a situaciones peligrosas.

Los avances en la ingeniería electrónica han permitido reducir los costos de fabricación y el desarrollo de vehículos cada vez más pequeños y económicos. En este sentido, es notable la potencia computacional de nuevas arquitecturas de microcontroladores (como la de los ARM Cortex M de 32bits), la miniaturización y mejora en el desempeño de los sensores inerciales (giróscopos y acelerómetros) de tipo MEMS[3], además de la estabilidad y sensibilidad de los nuevos barómetros MEMS (utilizados como altímetros) y magnetómetros. Por otro lado, con las nuevas constelaciones de satélites para GNSS[4], la mejora en la precisión y la reducción de los costos de los receptores satelitales, es cada vez más fácil obtener un sistema de posicionamiento global; incluso existen en el mercado soluciones económicas que permiten lograr precisión centimétrica para aquellas tareas que requieran tal precisión. Finalmente, las nuevas tecnologías y materiales utilizados para la fabricación de baterías recargables con gran densidad de energía, junto al desarrollo de actuadores cada vez más eficientes, en particular los motores brushless, permiten dotar a los vehículos de una autonomía cada vez mayor. Sin embargo, aún es la autonomía un factor limitante para ciertos escenarios en donde se deben efectuar misiones prolongadas. En estos escenarios la propulsión eléctrica no es una alternativa viable y se hace necesario recurrir a vehículos propulsados por combustible, los cuales suelen ser de mayor tamaño y costo, con una mecánica más compleja y la necesidad de un

[1]Unmanned Aerial Vehicles.
[2]Autonomous Surface Vehicles.
[3]Microelectromechanical systems.
[4]Global Navigation Satellite Systems.

mantenimiento periódico.

Existen diversas empresas que comercializan pequeños vehículos no tripulados. Sin embargo, asociado a estas tecnologías aún quedan muchos problemas por resolver. Esto hace que hoy día diversos grupos de investigación alrededor del mundo estén trabajando con este tipo de vehículos. Por ejemplo, en [1]–[4] se han desarrollado pequeños ASV con fines de investigación. Estos vehículos han demostrado ser de gran utilidad en tareas tales como el monitoreo medioambiental, la toma de muestras, la confección de mapas batimétricos entre otros, todos ellos sin el costo de desplegar grandes vehículos y personal. De igual modo, se han desarrollado UAVs para gran variedad de tareas específicas, como en [5]–[8], donde se observan aplicaciones en seguridad, agrimensura, tareas de monitoreo en zonas de difícil acceso, como bosques, volcanes, zonas de incendio o de desastres naturales. Cada vez son más las aplicaciones que se ven beneficiadas con el uso de los ASV o UAV y de pequeños vehículos terrestres.

Una de las limitaciones que se presenta al trabajar con pequeños vehículos es su capacidad de carga reducida, lo que no permite instalar un gran número de sensores de observación de peso apreciable. Para aquellas misiones que necesitan un gran conjunto de estos sensores de observación, se pueden utilizar varios vehículos, distribuyendo entre éstos los sensores. Estas arquitecturas segmentadas han despertado el interés de la comunidad, y en particular el estudio de la coordinación y control de la flota de vehículos es objeto de investigación en la actualidad. [9].

Mayor interés aún despierta el desarrollo de técnicas de control para la coordinación de vehículos que funcionen cada uno en un dominio diferente (vehículos acuáticos, junto con aéreos o terrestres). Dado que cada vehículo está dotado con un reducido número de sensores, y teniendo en cuenta que la percepción del entorno de cada uno de ellos es diferente y limitada, combinar datos provenientes de los distintos vehículos es de particular interés. En base a estos desarrollos, es posible contar con una arquitectura distribuida de robots, cada uno diseñado para cumplir una tarea simple. Finalmente, en función a la tarea más compleja que se requiera, es posible seleccionar y desplegar los robots más adecuados [9].

Todo vehículo autónomo cuenta con un hardware de bajo nivel, denominado Piloto Automático (AP por sus siglas en inglés), el cual se encarga de adquirir, sincronizar y acondicionar la información de los instrumentos de navegación, para luego estimar la orientación y la posición del vehículo, decidir su trayectoria y ejecutar los comandos correspondientes (es decir realizar la navegación, el guiado y el control del vehículo). A su vez, muchas veces esta electrónica es capaz de recibir instrucciones de un operador o de una computadora de alto nivel y comandar los actuadores del vehículo. Los AP están presentes en todos los vehículos autónomos, y muchas veces incluyen varios subsistemas interconectados, pero para pequeños vehículos autónomos, el AP debe tener un tamaño reducido y por lo general incluir los sensores de navegación. De estos sensores, uno de los más importantes, es la unidad de mediciones inerciales (IMU, por su siglas en inglés), compuesta por tres acelerómetros y tres giróscopos (situados en cada uno de los ejes). Otros sensores que suelen estar presentes, en la misma placa o como módulos externos, son los magnetómetros (para estimación de orientación), barómetros (utilizados para estimar altitud), receptores de GPS y diferentes tipos de sensores de velocidad y proximidad.

Cada clase de vehículo autónomo (ASV, UAV o terrestres) posee necesidades de control de bajo nivel particulares. Un UAV del tipo multi-rotor posee 6 grados de libertad (6 DOF[5]) mientras que un ASV posee solo 3 DOF. Los multi-rotores pueden ser actuados por medio de 4, 6 y más motores, mientras que los ASV suelen tener 1 ó 2, por su parte vehículos de ala fija suelen tener un motor para propulsión y servos para controlar la trayectoria. No obstante, muchas características son comunes a ambas clases de vehículos. Por ejemplo, para la estimación de la posición y la orientación del vehículo, suelen utilizarse algoritmos similares para fusionar los datos de los sensores de navegación. Los actuadores suelen comandarse mediante el mismo tipo de señales (PWM[6]), los comandos suelen provenir de radio-controles (RC) que utilizan protocolos similares. Teniendo en cuenta todas estas similitudes y diferencias, es de particular interés poder contar con un AP versátil, esto simplifica el desarrollo cuando se trabaja con varios vehículos de diferentes clases.

Existe una variedad de AP comerciales disponibles en el mercado, cuyos diseños buscan optimizar el peso, tamaño, costo, potencia computacional y periféricos, así como también diferentes tipos sensores. Dentro de los AP más populares, proyectos como Pixhawk [10] y APM [11] son *open-hardware* y *open-software*, con una enorme comunidad de usuarios y desarrolladores. El proyecto Pixhawk se presenta en diferentes versiones de su AP y el software es constantemente actualizado teniendo en cuenta los requerimientos de sus usuarios, ya que el mismo es muy utilizado en el ámbito académico y por usuarios aficionados. Con características similares se encuentra la línea de AP desarrolladas por DJI [12], los cuales se enfocan en UAV del tipo multi-rotor y que, comparado con otros proyectos como el Pixhawk, tienen un mayor costo y suelen ser instaladas en vehículos *listos para volar*. La principal desventaja es que, tanto su hardware como su software, son propietarios, por lo que la flexibilidad para proyectos académicos es más limitada. Cuando se requiere mayor capacidad de cómputo o el uso de sensores más avanzados se pueden utilizar AP como los desarrollados por Acutronic Robotic (ex Erle-Robotic), basados en una Raspberry Pi Zero. Otra opción es la controladora de vuelo Aereo, basada un microprocesador Intel Atom. Para realizar proyectos con mayor capacidad de cómputo, pueden utilizarse mini computadoras como la Intel NUC, pero estas alternativas suelen ser más costosas, con un considerable incremento en el peso y consumo respecto al Pixhawk o APM. La aplicación en la que se esté trabajando definirá cuál es el vehículo, así como el AP adecuado. En [13] se presenta una comparación entre distintos autopilotos que suelen encontrarse, principalmente en UAV de tipo multi-rotor.

[5]Degree Of Freedom.
[6]Pulse With Modulation.

II. Desarrollo de hardware del autopiloto

En el año 2018, los laboratorios GPSIC[7] y GESIC[8] desarrollaron un AP de bajo costo y versátil, para ser utilizado en pequeños vehículos autónomos de diversos tipos. El diseño se basó en un AP fabricado previamente en el GPSIC en 2013 y 2015, denominado Choriboard. La primera (año 2013) y segunda versión (año 2015) de la Choriboard [16], [17] utilizaban un procesador ARM Cortex-M3, sin unidad de punto flotante, con un clock de hasta 120MHz y 150 DMPIS (Drystone Millions of Instructions per Second), en un encapsulado LQFP de 100 pines de 14x14mm. En la Facultad de Ingeniería de la UBA se habían construido proyectos similares de autopilotos [18].

Las primeras versiones de la Choriboard utilizaban una IMU con un chip MPU6000 de Invensense. Este chipset, actualmente discontinuado, ofrecía la posibilidad de utilizar un firmware propietario llamado Digital Motion Processor (DMP), que permitía realizar la fusión y filtrado de datos dentro del mismo chip para obtener una estimación de la orientación del vehículo mediante un cuaternión. Si bien la versión MPU6050 del mismo dispositivo era mucho más popular, el MPU6000 presentaba la opción de comunicación por SPI hasta 20MHz, que presenta ventajas cuando se necesita procesar una gran cantidad de datos. El dispositivo se presentaba en un encapsulado QFN de 24 pines de 4x4mm.

Para completar la estimación de orientación, se agregó un magnetómetro HMC5883L de Honeywell. En la primera versión, este sensor se encontraba sobre la misma placa, pero su medición se veía fuertemente afectada por el campo magnético producido por los motores de los vehículos, lo cual hacía difícil la instalación de la placa en vehículos del tipo multi-rotor. Para solucionar este inconveniente, se utilizaba un magnetómetro adicional, ubicado entre 10cm y 15cm por encima del plano de los motores.

La primera versión de la Choriboard fue diseñada sobre una placa de 93x93mm, con suficiente espacio entre los diversos componentes para permitir la verificación de errores de diseño. Para la segunda versión, el factor de forma fue tomado en cuenta, y se redujeron las dimensiones de la placa a 100x60mm, como puede observarse en la figura 1.

Ambas versiones de la placa incluyen una fuente switching con un amplio rango de voltaje de entrada (7-40V), para permitir el uso de baterías LiPo de entre 2 y 6 celdas (7.4-22.2V), y con una salida de 5V-3A. Esta fuente permite alimentar todos los componentes de la placa (mediante un regulador lineal de bajo dropout de 3.3V), a la vez que habilita el uso de una computadora externa para tareas de alto nivel. De hecho, para la segunda versión de la Choriboard se diseñó una placa que permite adosarle una Intel Edison y así poder implementar algoritmos de navegación y control más complejos, por ejemplo aquellos basados en imágenes [19]–[21]. En ambas versiones, todos los conectores para las entradas y salidas de PWM, así como también para los sensores externos, puertos de comunicación y programación/debugging, se implementaron utilizando tiras de pines con un pitch de 2.54mm (0.1”).

[7]Grupo de Procesamiento de Señales, Identificación y Control, Facultad de Ingeniería de la UBA [14].

[8]Grupo de Estudio de Sistemas de Control, Facultad Regional San Nicolás, Universidad Tecnológica Nacional [15].

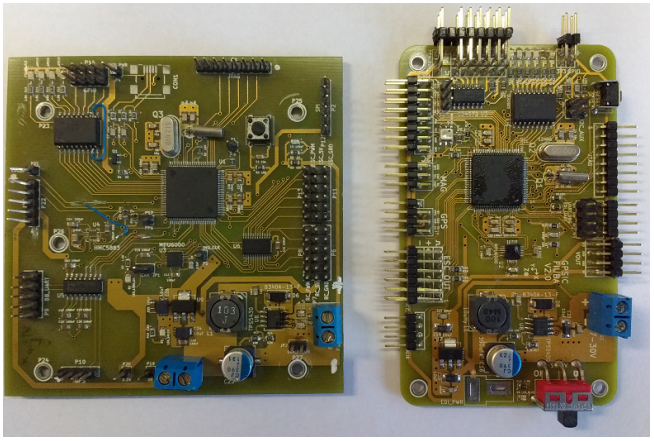

Figura 1. Versión I de la Choriboard (izq) y versión 2 (der)

II-A. Nueva propuesta de diseño del autopiloto

La tercera versión de la Choriboard (año 2018) se diseñó con dos objetivos principales:

1. Reducir el tamaño y peso de la placa, para poder ser utilizada en vehículos aéreos muy pequeños.
2. Actualizar el modelo de microcontrolador y sensores, para mejorar el rendimiento, y para remplazar dispositivos obsoletos y/o discontinuados.

El microcontrolador utilizado es un STM32F722, en un encapsulado LQFP de 64 pines de 10x10mm. Es un dispositivo ARM Cortex-M7 de alto desempeño, que puede operar hasta 216MHz y posee unidad de punto flotante integrada, logrando hasta 462 DMIPS, el triple que la versión previa. Cuenta con 512KB de memoria flash y 256KB de memoria RAM; 64KB de esta última son usados para almacenar datos de tareas tiempo real críticas y otros 16KB para ejecutar dichas tareas. Este chip fue elegido para obtener el mejor balance entre el tipo y tamaño de encapsulado (maximizar el número de pines utilizados, facilidad para soldadura manual), poder computacional (es uno de los mejores de la serie Cortex-M), documentación de firmware, disponibilidad de librerías de bajo nivel, y las comunidades de soporte.

Para remplazar el ya discontinuado MPU6000, se eligió otro dispositivo de Invensense, el ICM20602, similar a su antecesor. Una desventaja es que no soporta el uso del firmware DMP, razón por la cual la fusión y filtrado de los datos debe realizarse directamente en el microcontrolador. Sin embargo esto no es un inconveniente dado que, el firmware desarrollado para las primeras versiones del AP incluía rutinas para la estimación de orientación, las cuales han sido validadas con el DMP.

En el Cuadro I se muestra una comparación general entre los dos sensores, donde puede observarse que los giróscopos presentan 20 veces menos offset en ausencia de movimiento rotacional (ZRO), 20 % menos de ruido y tres veces mejor sensibilidad, mientras que los acelerómetros presentan la mitad de offset y cuatro veces menos ruido.

Para reducir el peso y el tamaño de la placa, todos los conectores fueron reemplazados por modelos de montaje superficial de Hirose, de la línea DF13, los mismos que se

Cuadro I
COMPARACIÓN ENTRE IMUS

PRODUCTO	ICM 20602	MPU6000
Encapsulado	2x2x0,75mm LGA 16 leads	4x4x0,9mm QFN 24 Leads
GIRÓSCOPOS		
FSR (dps)	±250/500/1000/2000	±250/500/1000/2000
ZRO @25°C	±1dps	±20dps
ZRO over Temp	±0,02dps/°C	±0,16dps/°C
Sens. Tol. @25°C	±1 %	±3 %
Sens. Over Temp	±2 %	±2 %
Gyro Noise	0.004dps/$\sqrt{Hz}$	0.005dps/$\sqrt{Hz}$
ACELERÓMETROS		
FSR	±2/4/8/16g	±2/4/8/16g
Offset @25°C	±25mg	X,Y : ±50mg Z: ±80mg
Offset Over Temp	X,Y: ±0.5mg/°C Z: ±1mg/°C	X,Y: ±0.5mg/°C Z: ±85mg/°C
Sens. Tolerance	±1 %	±3 %
Sens. Over Temp	±1,5 %	±2,5 %
Accel Noise	100$\mu g/\sqrt{Hz}$	400$\mu g/\sqrt{Hz}$

usan en el autopiloto Pixhawk. Al momento del diseño, los pines se han asignado de forma tal que todos los módulos utilizados en el Pixhawk pueden ser conectados directamente en la Choriboard.

Además, la fuente switching fue removida de la placa principal, dejando un conector de alimentación, también compatible con los que se utilizan en el Pixhawk. Estos módulos permiten el sensado de los niveles de tensión y consumo de corriente de la batería.

El barómetro BMP180, también discontinuado, fue reemplazado por el modelo MS5611 de TE Conectivity, que cuenta con un conversor ADC de 24 bits en lugar del de 19 bits de su antecesor, esto permite medir la altura con una sensibilidad de hasta 10cm. El barómetro se comunica con el microcontrolador mediante una interfaz I2C.

Para habilitar la conexión de diversos sensores externos, otro conector de bus I2C se encuentra disponible. Como dicha interfaz suele ser utilizada por un magnetómetro, se agrega un pin extra para ser asignado a la señal de fin de conversión de la medición. Si es necesario conectar múltiples sensores, se utiliza un extensor externo a la placa.

En el mismo bus I2C, pero montado sobre la placa principal, se encuentra una memoria EEPROM para almacenar información no volátil, como datos de calibración de sensores, o configuración del sistema.

El diseño cuenta con un conector adicional con un bus SPI diferente a aquel usado para la IMU, para ser utilizado con otra IMU de mayor calidad en pos de realizar calibraciones o comparaciones, o para comandar un On-Screen Display (OSD), que agrega los datos de vuelo a un transmisor de video.

Además, se provee un conector de un puerto UART para utilizarse con un módulo GPS, el cual cuenta no sólo con las líneas de transmisión y recepción, sino también con una entrada de PPS (Pulse Per Second - Pulso Por Segundo), que suele utilizarse para una precisa sincronización de tiempos con el sistema satelital.

Para la recepción de comandos se provee de una entrada para señales tipo PPM. Dado que se desea la reducción del tamaño, el encoder PWM a PPM no se incluye en la placa pero puede agregarse de forma externa de ser necesario. Además, se agrega una entrada para señales DSM de la línea de radiocontroles Spektrum (UART), para conectar directamente los micro-receptores Satellite de la marca. Por otro lado, seis salidas PWM están disponibles para controlar los actuadores del vehículo, los cuales en general son controladores electrónicos de velocidad (ESC) para motores trifásicos sin escobillas, o servomotores. Todas las señales de este tipo se encuentran en un único conector de 10 pines (6 salidas PWM con una masa común, y la entrada de PPM con +5V y masa).

Para almacenar una cantidad de datos masiva en tiempo real (por ejemplo, los datos sin procesar de los sensores), se utiliza una memoria microSD. Como indicadores visuales se agregan tres leds (rojo, verde, azul) para diversos usos.

El microcontrolador permite la conexión directa a un puerto mini USB para la actualización del firmware, o como una interfaz UART para la comunicación con una placa de alto nivel. También se provee un conector de 4 pines Serial Wire Debug (SWD), como otra opción para actualizar el firmware, y, además, para realizar debugging del mismo.

En la figura 2 se puede observar la comparación física entre la segunda y tercera versión de la Choriboard.

Figura 2. Último diseño, versión III (izq), y versión II (der).

Con respecto al hardware, las características de la placa diseñada son considerablemente similares a las del Pixhawk. La última versión del Pixhawk recientemente se actualizó e incluye un Cortex-M7. También es cierto que la compatibilidad entre el hardware de ambos autopilotos fue un factor importante al momento del diseño. Una de la razones detrás de dichas elecciones es la capacidad de desarrollar tareas de investigación de grado, master y doctorado, a la vez que se pueden crear trabajos conjuntos entre diferentes universidades, aprovechando la gran disponibilidad de los productos comerciales. Otra razón es la capacidad de utilizar sensores comerciales, que han sido probados minuciosamente, para comprobar la calidad del diseño realizado, mientras se desarrollan versiones propias de cada uno de ellos. Por otro lado, existen diversas prestaciones en los autopilotos comerciales que no son óptimas o incluso no tienen uso en diversas tareas de investigación, sin embargo consumen recursos del microcontrolador. La capacidad de poder eliminar las partes innecesarias puede liberar recursos útiles para asignar a otras tareas, o para aliviar la carga del microcontrolador. Por último, la selección de los componentes utilizados en base a

su disponibilidad y precio, y el diseño de un PCB simple con posibilidad de ser soldado a mano, otorga un mayor control sobre el costo de la placa finalizada. En la figura 3 puede verse un diagrama en bloques del hardware del autopiloto.

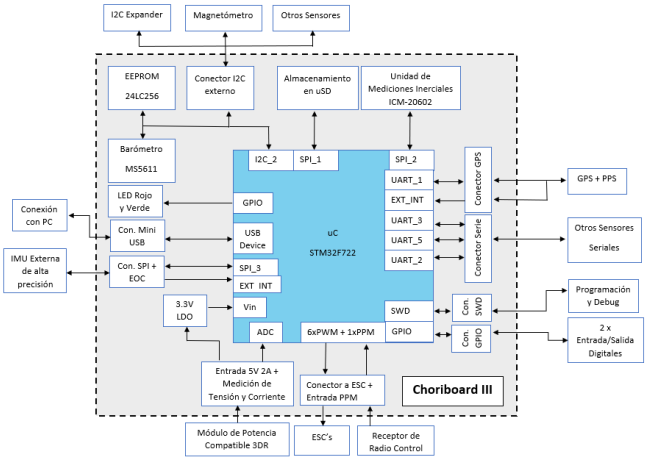

Figura 3. Diagrama en bloques del hardware desarrollado

III. Desarrollo del Firmware

El firmware, el cual está basado en las versiones desarrolladas para las versiones anteriores del AP, fue dividido en dos partes. La primera es una abstracción de los periféricos de bajo nivel, mientras que la segunda es la aplicación del usuario. Los drivers de bajo nivel tienen por objetivo inicializar el microcontrolador y sus periféricos (temporizadores, buses SPI-I2C, UARTS, etc.), mientras que la aplicación de usuario provee las funciones de alto nivel para realizar la configuración y lectura de los sensores, y ejecutar los algoritmos de navegación y control. En general, la adquisición de datos de los diferentes sensores se realiza a intervalos regulares, pero los sensores tienen distintas tasas de muestreo en el cual entregan información. Por ejemplo, la IMU puede realizar mediciones a una frecuencia entre 1KHz y 8KHz, el barómetro y magnetómetro a 10Hz, el receptor de radio control a 50Hz y el GPS a 1Hz (5Hz o 10Hz), entre otros. Los actuadores suelen funcionar a frecuencias de hasta 400Hz. Por lo tanto, se necesita de un sistema que pueda administrar la lectura de sensores y la fusión de datos teniendo en cuenta las características del sistema.

Para realizar la implementación del software, se prefirió utilizar un sistema *bare-metal* en lugar de basarse en un *RTOS* (sistema operativo de tiempo real). Aunque existen en la comunidad distintas visiones sobre cuál de estas opciones es más conveniente, hay un claro consenso sobre las ventajas que cada una presenta. Las ventajas que ofrece el desarrollo de un RTOS son diversas, en particular, el uso de un scheduler o un middleware facilita considerablemente el desarrollo. Si bien es cierto que se sacrifica memoria del microcontrolador, no parece una limitación para esta aplicación. Sin embargo, usar un RTOS significa, de alguna manera, perder cierto determinismo en la ejecución de las tareas, o al menos que se haga más complejo el análisis de tiempos. Ésta es la razón para la elección tomada, obtener un mayor control sobre los tiempos de respuesta ante diferentes eventos, normalmente dados por los sensores, como puede ser la llegada de un pulso de PPS, o el fin de la conversión del ADC en la IMU o el magnetómetro. Cuando se desea dar pruebas formales sobre la estabilidad de un lazo de control, teniendo en cuenta la implementación en el hardware, resulta conveniente poder describir con cierto determinismo el comportamiento del sistema puesto que facilita el análisis de los algoritmos. Por ejemplo, el uso del PPS para sincronización de tiempos, el cual es una señal con precisión del orden de los nanosegundos, requiere su atención con la mayor prioridad posible, relegando otras posibles tareas en ejecución.

Para este fin, en los microcontroladores utilizados se encuentra disponible un controlador dedicado a la organización y atención de interrupciones, llamado Nested Vectored Interrupt Controller (NVIC). El mismo permite agrupar a las diferentes fuentes de interrupción, tales como interrupciones por timers internos, canales de comunicación o cambio de flancos o nivel en determinados pines, en diferentes niveles de prioridades y sub-prioridades. De esta manera, interrupciones con un alto nivel de prioridad pueden pausar temporalmente la ejecución de interrupciones de menor prioridad, y retomarlas una vez hayan sido finalizadas. Por otro lado, interrupciones con el mismo nivel de sub-prioridad son puestas en una cola y atendidas secuencialmente.

Además, se hace uso de otra prestación de esta línea de microcontroladores, el acceso directo a memoria (DMA - Direct Memory Access). Dicha prestación permite habilitar, entre otras cosas, el acceso directo de los puertos de comunicación a la memoria donde son almacenadas las variables, ejecutándose dicho proceso en paralelo al programa principal. Dado que se hace uso de varios canales de comunicación para los diferentes sensores, y para transmitir datos de forma masiva, dicho recurso permite alivianar la carga del procesador.

Para el desarrollo del firmware se ha utilizado el paquete *STM32CubeMX*. Esta aplicación gráfica permite configurar cada pin del microcontrolador para asignarle la función específica requerida. También provee una forma simple de configurar el complejo sistema del reloj interno del microcontrolador, permitiendo elegir la frecuencia de trabajo deseada y configurando automáticamente los PLLs. Esta herramienta también genera el código para la inicialización de los periféricos configurados y crea un proyecto para diferentes IDEs. El IDE utilizado es el provisto por el fabricante del microcontrolador, el cual está basado en Eclipse.

III-A. ROS y ROS Serial

ROS (Robot Operating System) es un entorno de trabajo (framework) para el desarrollo de software orientado a robots, que provee la funcionalidad de un sistema operativo. ROS provee los servicios estándar de un sistema operativo (más precisamente funciona como middleware) tales como abstracción del hardware, control de dispositivos de bajo nivel, implementación de funcionalidad de uso común, paso de mensajes entre procesos y mantenimiento de paquetes.

En los últimos años, ROS se ha convertido en uno de los estándares más usados en grupos de investigación dedicados a la robótica. El funcionamiento es por medio de nodos, programados generalmente en C++ o Python, los cuales

se comunican entre sí compartiendo mensajes a través de tópicos. Cada nodo suele tener una función específica y básica en el entorno. Es posible ejecutar nodos en distintos dispositivos, formando una red entre ellos. Por ejemplo, una computadora puede correr el nodo maestro de ROS (roscore) y algún nodo que procese datos mientras que en otra computadora se pueden correr los nodos encargados de la adquisición de los datos. Esta segunda computadora puede estar, incluso, a bordo de un vehículo autónomo. ROS necesita un sistema operativo, generalmente Linux, para poder ejecutarse. En el caso en donde la adquisición de datos se realiza en alguna computadora de bajo nivel, en la que no se puede instalar un sistema operativo, es necesario hacer uso de algún paquete que permita la comunicación entre el sistema embebido de bajo nivel y la computadora de alto nivel. En este trabajo se hace uso del paquete *rosserial* [22].

Rosserial es una interfaz de comunicación que permite intercambiar mensajes del tipo ROS (los mismos que se utilizan en la comunicación interna de los nodos) a través del protocolo serial. Para que esta comunicación se lleve a cabo es necesario ejecutar el paquete *rosserial_python* en la computadora principal y desarrollar un cliente en el sistema embebido de bajo nivel. Actualmente existen clientes para Arduino, Mbed, Tivac y otros. En este trabajo se ha desarrollado un cliente de dicha librería para el microcontrolador de la Choriboard.

La integración de un AP en ROS permite dividir el sistema de control de un vehículo más complejo en pequeños sistemas embebidos capaces de crear nodos, publicar y suscribirse a tópicos del sistema así como hacer uso de los servicios y parámetro del entorno. Como se ha comentado anteriormente, el esqueleto del firmware de la Choriboard se crea con la herramienta CubeMX. ROS Serial necesita una comunicación serie con la PC de alto nivel. Es por ello que se dedica la UART3 para este fin.

En este proyecto se han utilizado los mensajes IMU y BatteryState. Dichos mensajes forman parte de los std_sensor message. Para utilizarlos se crea un node handler y dos objetos publisher, uno para cada mensaje. Luego, periódicamente se actualizan los valores de los datos de dichos mensajes y se publican en el nodo maestro de ROS a través de rosserial.

En la sección de validación experimental se verá la utilidad de haber implementado rosserial, y cómo permitió utilizar de manera simple información del AP para procesar la información adquirida por un LIDAR instalado en un ASV.

IV. Bancos de prueba para la validación experimental

Para validar experimentalmente el desempeño del AP se realizaron varias pruebas. En particular se presentan aquí los resultados obtenidos con un vehículo aéreo no tripulado del tipo multi-rotor (más precisamente un quadrotor) y con un vehículo acuático de superficie. Ambos vehículos fueron desarrollados por el grupo de investigación y usualmente son utilizados como banco de pruebas para validar los algoritmos de navegación, guiado y control. A continuación se presenta una introducción de estos vehículos.

IV-A. Diseño y construcción de los vehículos

En la actualidad, no sólo la electrónica de los vehículos autónomos se encuentra disponible de forma masiva y a un bajo costo. Con la aparición y proliferación de la tecnología de impresión 3D en una gran variedad de materiales, también las estructuras de este tipo de vehículos se han vuelto accesibles para el público general. Es posible encontrar en sitios dedicados al almacenamiento de modelos 3D (Thingiverse, GrabCAD, etc.) no sólo piezas de repuesto y accesorios personalizados para modelos comerciales de estructuras, sino también modelos propios diseñados para facilitar la producción y armado por usuarios particulares que no disponen de complejas maquinarias industriales (en particular para UAV del tipo multi-rotor).

Por ejemplo, para algunas pruebas de control con múltiples vehículos aéreos en el GPSIC, se diseñó un vehículo quadrotor con características similares a los vehículos comerciales, con una distancia diagonal entre motores de 500mm. Este diseño es el mismo que se utiliza para realizar los vuelos de validación en este trabajo, y el mismo puede descargarse del repositorio utilizado por el grupo[9]. Los actuadores están compuestos por cuatro motores RCTimer 2212/920Kv y hélices plásticas 9443. Se utilizaron un transmisor Spektrum DX7 y un receptor AR6100, de la misma marca, para controlar de manera remota el vehículo. Para la alimentación del vehículo se utilizó una batería LiPo de cuatro celdas. La figura 4 muestra la Choriboard instalada en el UAV.

Figura 4. UAV utilizado para la validación experimental del autopiloto.

En 2017 comenzó una colaboración entre los laboratorios GPSIC y GESIC con el propósito de desarrollar un ASV de bajo costo capaz de funcionar en aguas calmas, como las de un arroyo o un lago. Como resultado de esta colaboración se obtuvo el ASV Yaguarón [23]. En la figura 5 se puede ver una foto de este vehículo navegando en un arroyo de la zona de Ramallo, provincia de Buenos Aires.

La configuración elegida fue tipo catamarán con 2 pontones. Esta configuración otorga una gran estabilidad y permite una considerable capacidad de carga útil aceptable para un vehículo de este porte. Los pontones fueron desarrollados a partir de tubos de PVC y las punteras, así como las tapas traseras de los pontones fueron construidas con piezas moldeadas en fibra de vidrio y resina epoxi. La estructura central, que une ambos pontones, fue construida en

[9]https://gitlab.com/gpsic/modelos-3d/quad

aluminio. Para soportar la electrónica y algunos sensores se construyó un segundo marco desmontable, también de aluminio, ubicado en la parte posterior del ASV y montado en forma vertical.

La propulsión del ASV está dada por dos impulsores de la firma BlueRobotics, modelo T100, instalados uno en cada pontón. La configuración de propulsión en modo diferencial permite el control de *heading* sin la necesidad de timones móviles, facilitando la mecánica. Cada motor es manejado por un Controlador Electrónico de Velocidad (ESC por sus siglas en inglés) del mismo fabricante. Estos ESC permiten la operación del vehículo hacia adelante y en reversa. La fuente de energía utilizada son tres baterías de 12V y 7Ah, lo que le da una autonomía de 2 horas. Éstas fueron montadas en la parte delantera del ASV para balancear el peso del mismo.

Todos los componentes electrónicos del Yaguarón ASV fueron montados en un gabinete plástico, estanco, en el marco vertical del vehículo. En la figura 6 puede verse el montaje de la Choriboard III utilizando una pieza plástica fabricada con impresora 3D, el receptor de Radio Control, los ESCs y el módulo de alimentación. En esta figura también puede verse una Raspberry Pi 3, que se utilizó en las pruebas para adquirir información de un sensor de distancia láser y de los sensores de navegación del AP.

Figura 5. Yaguaron ASV.

IV-B. Control de los vehículos

En esta sección se presentan los algoritmos de control que se implementan en el firmware del AP. El propósito no es presentar en detalle el algoritmo, sino dar una explicación general, para que pueden comprenderse los resultados que se presentan más adelante.

IV-B1. Estabilización del UAV: La distribución de los motores con la dirección de giro y la convención de los ejes son representados en la figura 7.

Los ángulos de navegación utilizados se definen como la rotación sobre cada uno de los ejes: *roll*, para la rotación sobre el eje *x*, *pitch*, para la rotación sobre el eje *y*, y *yaw*, para la rotación sobre el eje *z*.

Por otro lado, el vehículo posee sólo 4 DOF, ya que puede actuar independientemente sobre los tres ángulos de navegación, además de la aceleración vertical (respecto a la terna del vehículo). Cualquier movimiento en el plano *xy*

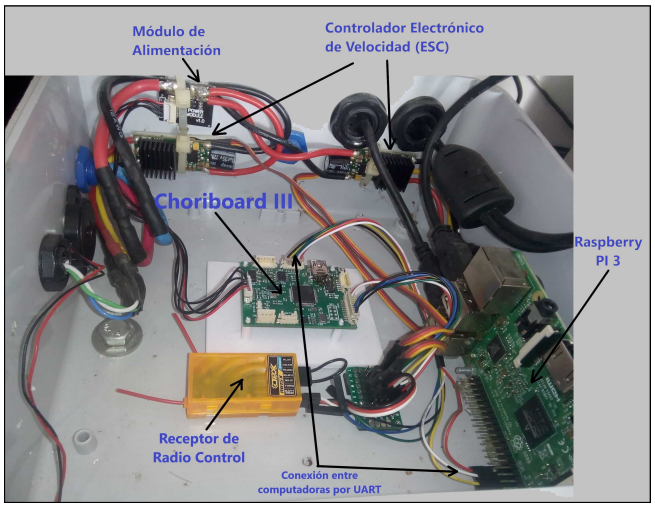

Figura 6. Interconexión de los componentes del sistema electrónico.

(respecto de una terna inercial fija a la tierra), estará ligado a variaciones de los ángulos de pitch y roll.

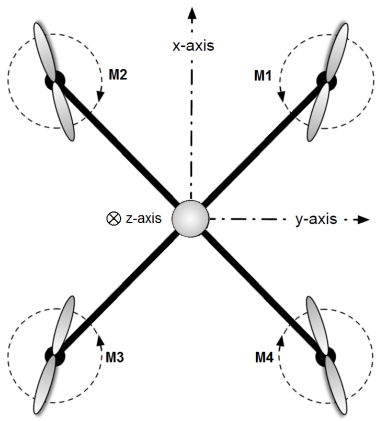

Figura 7. Distribución de los motores y ejes del UAV.

Para poder volar un vehículo del tipo multi-rotor, el AP debe ejecutar varias tareas. Primero, para estabilizar la orientación del vehículo, los datos de los tres acelerómetros y giróscopos son adquiridos y procesados para obtener una estimación filtrada de los ángulos de pitch y roll. Los acelerómetros están afectados por ruido de alta frecuencia, pero (en una situación de vuelo estático - *hovering*) entregan una buena medición de la dirección del vector de gravedad. Por otro lado, los giróscopos tienen una buena respuesta en alta frecuencia, pero suelen estar afectados por un sesgo. Para extraer las mejores características de cada uno de los sensores, se realiza una calibración inicial para eliminar el sesgo de los giróscopos y los ángulos de pitch y roll son obtenidos usando un filtro complementario [24]. El magnetómetro, previamente calibrado, es usado para obtener el ángulo de yaw respecto del norte magnético.

Para guiar el vehículo y enviar los comandos de referencia de los ángulos, se utiliza un radio control de 7 canales marca Spektrum operando en la frecuencia de 2.4GHz. Cada canal del control envía una señal PWM a 50Hz, con un tiempo en alto de entre 1 y 2mS. El receptor saca una señal PPM compuesta por todas las señales PWM una detrás de la otra

e ingresa a un pin dedicado para tal fin en la FC. Cinco canales son usados, uno para el empuje vertical y tres para las referencias de pitch, roll y yaw y un switch del tipo on/off para habilitar/deshabilitar el control y los motores.

Para estabilizar el vehículo, se utilizan tres controladores PID independientes para cada uno de sus ángulos. Cuando se implementan los controladores PID se debe notar que actuando sobre el error en los ángulos de pitch, roll y yaw, la salida será el torque necesario en cada eje para realizar la acción deseada. Para convertir los torques en fuerzas individuales para cada motor se deben establecer las relaciones entre dichas magnitudes. El conjunto de los torques de cada motor más el empuje vertical se relaciona con las fuerzas individuales de cada motor de la siguiente forma:

$$\begin{bmatrix} M_x \\ M_y \\ M_z \\ F_z \end{bmatrix} = A \cdot \begin{bmatrix} f_1 \\ f_2 \\ f_3 \\ f_4 \end{bmatrix} \quad (1)$$

donde, para el multi-rotor descripto en la figura 7:

$$A = \begin{bmatrix} -\frac{\sqrt{2}}{2}l & \frac{\sqrt{2}}{2}l & \frac{\sqrt{2}}{2}l & -\frac{\sqrt{2}}{2}l \\ \frac{\sqrt{2}}{2}l & \frac{\sqrt{2}}{2}l & -\frac{\sqrt{2}}{2}l & -\frac{\sqrt{2}}{2}l \\ \tilde{k}_t & -\tilde{k}_t & \tilde{k}_t & -\tilde{k}_t \\ -1 & -1 & -1 & -1 \end{bmatrix}$$

La constante $\tilde{k}_t$ representa la relación entre el torque producido en el eje Z del vehículo por el giro de los motores a una distancia l del centro, y la fuerza es ejercida en el mismo eje.

Para obtener las fuerzas individuales de los motores dados los torques, es suficiente con usar la pseudoinversa de A (ver por ejemplo [25]), ya que produce solución con la mínima energía y por ende el mínimo uso de potencia.

Más información acerca del análisis de la dinámica y el control de un quad-rotor puede encontrarse en [26].

IV-B2. Guiado del Yaguaron ASV: Como se mencionó anteriormente, gran parte de los periféricos utilizados para estabilizar un UAV son los mismos que se utilizan para navegar con el ASV. La lectura de los comandos del Radio Control, el manejo de los motores (por medio de los ESC), la telemetría con una PC en tierra utilizando radio enlace y la estimación de la actitud del vehículo son ejemplos de estas semejanzas. Es por ello que se reutilizó gran parte del firmware del UAV, siendo necesario reescribir las funciones que vinculan los comandos del Radio Control con las señales enviadas a los motores para la navegación en modo manual.

V. Resultados

En esta sección se presentan los resultados obtenidos utilizando la Choriboard en el UAV y el ASV. En [27] puede verse un video de las pruebas realizadas con los vehículos. Primero se presentan los resultados de los experimentos en un vuelo realizado con el multi-rotor, en donde se realiza un análisis del control interno (de hovering) y el desempeño del control ante perturbaciones. El propósito de la segunda prueba es mostrar el correcto funcionamiento del AP para controlar el ASV y además mostrar la utilidad de haber incluido el rosserial en el firmware.

V-A. Desempeño del AP en un multi-rotor

Se realizaron varios vuelos con el multi-rotor para evaluar el desempeño de la Choriboard. Todos los vuelos fueron realizados en exteriores, con vientos moderados.

En las figuras 8, 9 y 10, se muestran los valores medidos y de referencia para los ángulos de pitch, roll y yaw durante uno de los vuelos. Puede observarse que el desempeño es aceptable, manteniendo los ángulos de pitch y roll con 0.5 grados respecto de la referencia en estado de vuelo estacionario; mientras que las maniobras las sigue apropiadamente. Por otro lado, para el ángulo de yaw también se observa un buen comportamiento, siguiendo la referencia correctamente.

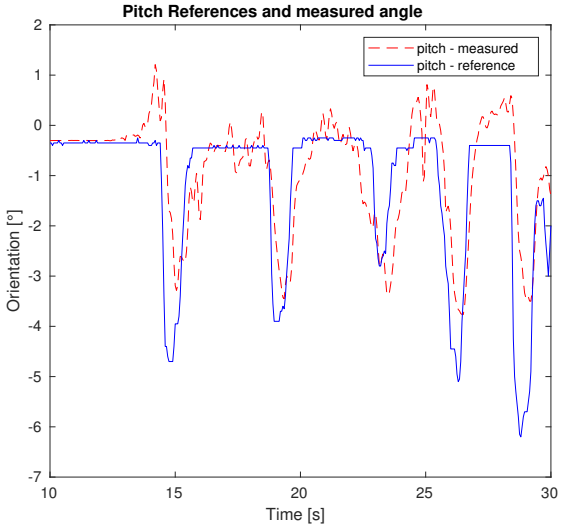

Figura 8. Ángulo de pitch: Medido y Referencia.

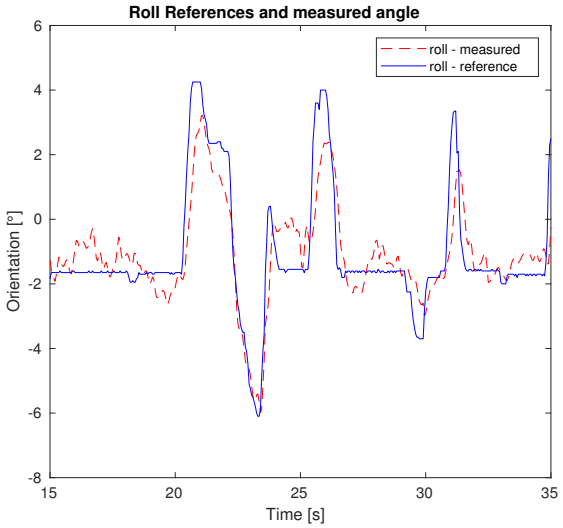

Figura 9. Ángulo de roll: Medido y Referencia.

En cuando a la robustez del control, en la figura 11 se muestra el rechazo a una perturbación que se introdujo golpeando uno de los brazos del vehículo durante un vuelo. Puede apreciarse que el vehículo se recupera bien, sin oscilaciones y en un tiempo adecuado.

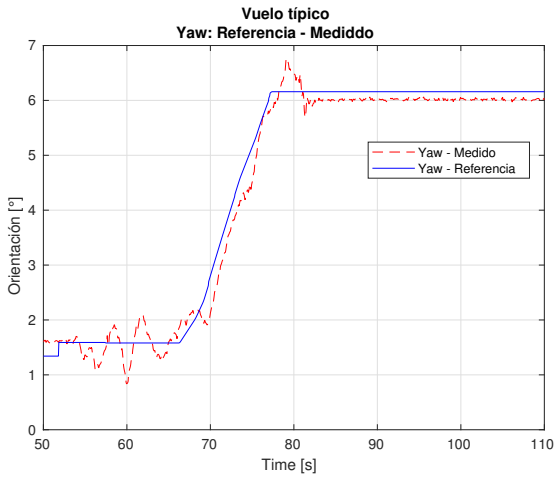

Figura 10. Ángulo de yaw: Medido y Referencia.

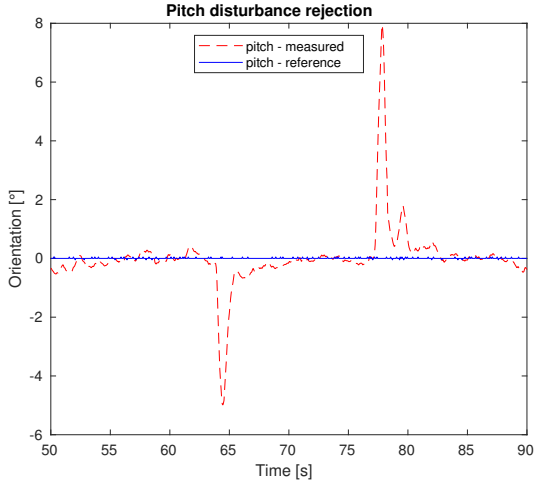

Figura 11. Rechazo a perturbaciones en pitch.

V-B. Desempeño del AP en un vehículo acuático

El segundo experimento fue realizado utilizando un vehículo acuático de superficie. Como validación del sistema desarrollado se propone realizar el relevamiento topográfico de la costa del arroyo "Las Hermanas", ubicado en la ciudad de Ramallo, provincia de Buenos Aires, Argentina. Para esta tarea se utiliza un sensor láser para medir distancia (LIDAR), los datos de una IMU, provenientes del AP desarrollado, y un sistema basado en ROS. El LIDAR utilizado es de la firma Hokuyo, posee un rango de funcionamiento máximo de 5 metros. Este sensor es montado en el marco vertical del vehículo y se lo ubica apuntando hacia el costado derecho del ASV. De este modo el SLAM se produce utilizando solo una costa del arroyo. En la figura 12 se observa el montaje del LIDAR en el vehículo.

En estas pruebas el vehículo es controlado por un operador en la costa, en modo manual, a baja velocidad y cerca de la costa, de modo que el LIDAR obtenga datos válidos. Todos los datos son grabados para un posterior procesamiento en modo off-line.

A bordo del vehículo se instaló una Raspberry Pi 3B+, que corre un sistema operativo Linux con el sistema ROS. Esta placa se utiliza para recolectar la información del LIDAR y almacenarla en un archivo bag-file de ROS. Además se adquiere información proveniente de la IMU de la Choriboard, que se utiliza para compensar el movimiento del sistema y mejorar el cálculo de la trayectoria del vehículo en el procesamiento de la información del LIDAR. El rosserial simplifica el trabajo, dado que permite adquirir la información de la IMU directamente en un nodo de ROS en la Raspberry y guardar esta información junto con la del LIDAR para su post-procesamiento. Los datos almacenados en la Raspberry son post-procesados en una PC usando el paquete Google Cartographer [28].

Figura 12. Montaje del sensor principal del ASV.

En la figura 13 se muestra la superposición del mapa obtenido utilizando el Google Cartographer, sobre un mapa de un servicio satelital, en este caso Google Maps.

Figura 13. Mapa obtenido por Google Cartographer e imagen satelital.

VI. Conclusiones

En este trabajo se presentó una actualización del autopiloto Choriboard, el cual es más liviano y posee mayor potencia de cómputo que las versiones previas.

El tamaño compacto y la compatibilidad de sus conectores con dispositivos y sensores comerciales le dan una gran versatilidad, permitiendo su uso en pequeños vehículos aéreos no tripulados y otros robots móviles. Como validación experimental, se instaló el autopiloto en un vehículo

aéreo no tripulado y un vehículo acuático de superficie, mostrando su correcto funcionamiento. El firmware utilizado para controlar los vehículos también ha sido desarrollado por nuestro grupo.

El uso de un diseño propio de hardware y software, permite un mayor control sobre el funcionamiento del autopiloto. Esta flexibilidad permite tener un control a bajo nivel de los tiempos de ejecución de cada una de las tareas, o poder introducir herramientas de validación de software, que permiten evaluar el desempeño de algoritmos y técnicas de diseño, principalmente con un propósito académico.

Teniendo en cuenta las similitudes con otros autopilotos como el Pixhawk, es pertinente preguntarse sobre la necesidad de desarrollar una nueva versión del autopiloto Choriboard. La primera versión de la Choriboard fue desarrollada en 2013, en el marco de un Proyecto de Desarrollo Tecnológico y Social, financiado por la Universidad de Buenos Aires. El propósito de este proyecto era demostrar la factibilidad de desarrollar en el país esta tecnología. Pero además, este desarrollo generó varias tesis de grado y posgrado, diversas investigaciones relacionadas con algoritmos de navegación, guiado y control, estudio de sistemas tolerantes a fallas, y métodos de detección y corrección de errores, entre otros proyectos. Esto demuestra la importancia de incentivar este tipo de desarrollos en el país, y el impacto positivo que han tenido para el desarrollo de estas tecnologías. El conocimiento generado Más aún, esta nueva actualización del autopiloto generó una fructífera colaboración entre dos grupos de investigación, que se espera que crezca y que se sumen en el futuro otros grupos de investigación.

REFERENCIAS

[1] S. Bertram, C. Kitts, D. Azevedo, G. D. Vecchio, B. Hopner, G. Wheat, and W. Kirkwood, "A portable asv prototype for shallow-water science operations," in *OCEANS 2016 MTS/IEEE Monterey*, Sept 2016, pp. 1–6.

[2] G. G. Acosta, B. Menna, R. de La Vega, L. Arrien, H. Curti, S. Villar, R. Leegstra, M. D. Paula, I. Carlucho, F. Solari, and A. Rozenfeld, "Macabot: prototipo de vehículo autonomo de superficie," in *XI Jornadas Argentinas de Robótica*, Nov. 2017.

[3] S. Manjanna, A. Q. Li, R. N. Smith, I. Rekleitis, and G. Dudek, "Heterogeneous multi-robot system for exploration and strategic water sampling," in *IEEE International Conference on Robotics and Automation (ICRA)*, 2018.

[4] J. Moulton, N. Karapetyan, A. Q. Li, and I. Rekleitis, "External force field modeling for autonomous surface vehicles," in *International Symposium on Experimental Robotics*, Nov. 2018.

[5] G. Jiang, R. M. Voyles, and J. J. Choi, "Precision fully-actuated uav for visual and physical inspection of structures for nuclear decommissioning and search and rescue," in *2018 IEEE International Symposium on Safety, Security, and Rescue Robotics (SSRR)*, Aug 2018, pp. 1–7.

[6] J. Rojas, C. Devia, E. Petro, C. Martinez, I. Mondragon, D. Patino, M. C. Rebolledo, and J. Colorado, "Aerial mapping of rice crops using mosaicing techniques for vegetative index monitoring," in *2018 International Conference on Unmanned Aircraft Systems (ICUAS)*, June 2018, pp. 846–855.

[7] D. C. Guastella, L. Cantelli, C. D. Melita, and G. Muscato, "A global path planning strategy for a ugv from aerial elevation maps for disaster response," in *ICAART*, 2017.

[8] P. Addabbo, A. Angrisano, M. L. Bernardi, G. Gagliarde, A. Mennella, M. Nisi, and S. Ullo, "A uav infrared measurement approach for defect detection in photovoltaic plants," in *2017 IEEE International Workshop on Metrology for AeroSpace (MetroAeroSpace)*, June 2017, pp. 345–350.

[9] T. Fossen, K. Pettersen, and H. Nijmeijer, *Sensing and Control for Autonomous Vehicles.* Springer, 2017.

[10] "Pixhawk, an open source autopilot." [Online]. Available: https://pixhawk.org/

[11] "ArduPilot, an open source autopilot." [Online]. Available: http://ardupilot.org/

[12] "DJI Naze v M-2." [Online]. Available: https://www.dji.com/naza-m-v2

[13] Z. Yang, F. Lin, and B. Chen, "Survey of autopilot for multi-rotor unmanned aerial vehicles," in *42nd Annual Conference of the IEEE Industrial Electronics Society*, 10 2016, pp. 6122–6127.

[14] "Grupo de Procesamiento de Señales, Identificación y Control." [Online]. Available: http://psic.fi.uba.ar

[15] "Grupo de Estudio de Sistemas de Control." [Online]. Available: https://www.facebook.com/gesic.frsn/

[16] F. Roasio, *Diseño de un sistema de navegación integrada para un UAV.* Tesis de Grado, Facultad de Ingeniería de la Universidad de Buenos Aires, 2013.

[17] C. Pose, *Desarrollo de algoritmos de navegación y control para un vehículo aéreo autónomo.* Tesis de Grado, Facultad de Ingeniería de la Universidad de Buenos Aires, 2014.

[18] A. Kharsansky, *Diseño e implementación de un sistema embebido de control de actitud para aeronaves no tripuladas.* Tesis de Grado, Facultad de Ingeniería de la Universidad de Buenos Aires, 2013.

[19] J. Luiso, *Desarrollo de un sensor de flujo óptico.* Tesis de Grado, Facultad de Ingeniería de la Universidad de Buenos Aires, 2016.

[20] J. E. Luiso and J. I. Giribet, "Sensor de flujo óptico," in *Reunión de Procesamiento de la Información y Control (RPIC)*, 2017.

[21] A. Tournour, *Control de un vehículo aéreo no tripulado utilizando información de flujo óptico.* Tesis de Grado, Facultad de Ingeniería de la Universidad de Buenos Aires, 2018.

[22] "ROS serial: a protocol for wrapping standard ROS serialized messages." [Online]. Available: http://wiki.ros.org/rosserial

[23] L. Garberoglio, P. Moreno, J. I. Giribet, and I. Mas, "Autonomous vehicles for outdoor multidomain mapping," in *IEEE Biennial Congress of Argentina (ARGENCON)*, 2018.

[24] J.-M. P. R. Mahony, T. Hamel, "Nonlinear complementary filters on the special orthogonal group," *IEEE Transaction on Automatic Control*, vol. 53, no. 5, pp. 1203–1218, 2008.

[25] G. Strang, *Introduction to Linear Algebra.* Wellesley-Cambridge Press., 2016.

[26] J. Li and Y. Li, "Dynamic analysis and pid control for a quadrotor," in *Mechatronics and Automation (ICMA), 2011 International Conference on*. IEEE, 2011, pp. 573–578.

[27] "Choriboard V3 - Autopilot." [Online]. Available: https://www.youtube.com/watch?v=MgewmB_4lAg

[28] W. Hess, D. Kohler, H. Rapp, and D. Andor, "Real-time loop closure in 2d lidar slam," in *2016 IEEE International Conference on Robotics and Automation (ICRA)*, May 2016, pp. 1271–1278.