4^to Congreso Argentino de Ingeniería Aeronáutica

DISEÑO Y DESARROLLO DE PLATAFORMA PARA AUTOPILOTO DE PARACAÍDA IMPLEMENTADO EN COMPUTADORA INDUSTRIAL ABIERTA ARGENTINA (CIAA)

C. Alberoni[a], J. Fernandez[a] y M. Busnardo[a]

[a]*Dpto. Electrónica y Telecomunicaciones, Facultad de Ingeniería, Instituto Universitario Aeronáutico Av. Fuerza Aérea 6500 (IX5010JMX) Córdoba, Argentina. http://www.iua.edu.ar*

Palabras Clave : Piloto automático, UAV, paracaída, autopiloto, computadora industrial abierta Argentina (CIAA), sistemas embebidos

Resumen

La entrega aérea o lanzamiento con paracaídas, es un tipo de puente aéreo desarrollado para el re-abastecimiento de tropas o entrega de suministros para ayuda humanitaria cuando es inaccesible de otros modos. La carga que desciende del paracaídas sigue una trayectoria fijada según las condiciones por la que fue lanzada. Sin embargo, a medida que desciende, sufre de modificaciones debido a la intensidad y dirección de vientos entre otros factores, y por lo tanto, se puede alejar del punto fijado varios kilómetros.

El presente trabajo tiene por objetivo el diseño y desarrollo de una plataforma para el desarrollo de un piloto automático que pueda comandar un paracaída y a través de este sistema, es aumentar la precisión del sistema es decir disminuir sustancialmente el radio de caída del mismo.

La plataforma utilizada para la implementación de la lógica del piloto automático es la Computadora Industrial Abierta Argentina (CIAA) en su versión EDU-CIAA-NXP. Se hace uso del RTOS basado en el estándar OSEK-OS 2.2.3.

Sobre la placa madre EDU-CIAA-NXP, se diseña un circuito impreso (PCB) que contiene interfaces de entrada y salida para los sensores y actuadores requeridos por el sistema.

e-mail: calberoni@gmail.com

1. INTRODUCCIÓN

Como se ha comentado, en el presenta trabajo se desarrollará y construirá el sistema electrónico que adquiere la información de los sensores necesarios, y genera las señales de control a los actuadores correspondientes, permitiendo comandar al paracaídas a través de un sistema de control.
Existen actualmente estudios, elementos tecnológicos y productos comerciales que permiten desarrollar paracaídas comandados autónomos que, lanzados desde cierta posición y altitud, pueden hacer llegar una carga dada (por ejemplo medicamentos) a un punto de destino preestablecido, por ejemplo, mediante coordenadas geográficas. Con un equipo de este tipo, no resultaría necesario volar a baja altura y sería posible realizar el lanzamiento desde una distancia lejana a una región hostil o terreno donde la geografía no lo permite. Este sistema posibilita colocar a todas las cargas en un único punto (dentro de cierto radio) o entregar cargas a múltiples y diferentes puntos de destino lanzándolas desde una única aeronave y, dependiendo del caso, hasta en un único viaje, tal como se muestra en la figura 1. Con base en lo anterior, surge la iniciativa de desarrollar un dispositivo autónomo para la entrega de cargas con precisión, utilizando un paracaídas comandado, guiado por un sistema de navegación automático.

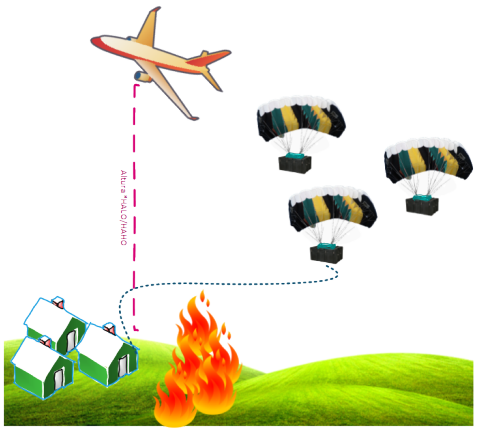


Figura 1: Ilustración de ejemplo de funcionamiento del Sistema en caso de incendio con zona anegada

1.1. Aplicaciones

El alcance del siguiente proyecto tiene como principales aplicaciones la *Entrega de viveres*, *Entrega de armas*, *Entrega de medicamentos/primeros auxilios*, *Herramientas necesarias para zonas de desastres*. Es importante destacar que también puede ser aplicado en *Aplicaciones de autopiloto para aeronaves / drones*, *Aplicaciones de autopiloto para automoviles*, *Aplicaciones de datalogger de parámetros atmosféricos para globos meteorológicos* u cualquier otra aplicación que requiera de la captura y almacenamiento de los parametros sensados.

2. DESARROLLO

En la figura 2(a) se muestran las partes que componen todo el sistema, como son el paracaída cuadrado, el sistema de autopiloto con su batería y actuadores integrados en su gabinete y por último la carga útil, es decir medicamentos, alimentos, etc hasta 100Kg.
Por otro lado en la figura 2(b) se ve en detalle las partes que componen al piloto automático CIAAPilot en su versión Poncho y su distribución en el PCB, cuales responden a las características que se muestran a continuación y que debe cumplir el sistema:

- Captura de sensores de plataforma inercial (acelerometro, girómetro), sensor de presión barométrica, magnetómetro y posterior conversión y almacenamiento de ángulos de Euler (yaw, pitch, roll) y altura.
- Captura de sensor de presión diferencial y posterior conversión y almacenamiento de velocidad KTAS o TAS (según corresponda).
- Captura, conversión y almacenamiento de mediciones de corriente consumida y tensión de batería.
- Captura de señales de control proveniente de receptor de radio control y posterior procesamiento de manera de tener control manual desde tierra del paracaída.

- Dar salida a actuadores, ya sea, servomotores o controladores de velocidad (ESC – Electronic speed control).
- Estado visual de sistema: Dar salida mediante leds del estado del piloto automático (normal funcionamiento - heartbeat, funcionamiento HIL, estado de sensores, etc.).
- Dar salida digitales para funcionalidades extras no definidas aún, como por ejemplo eyección de paracaída.
- Recepción y almacenamiento de posición global (GPS/GLONASS/Galileo)
- Envío de telemetría de datos almacenados (Ángulos de Euler, altura, posición global, etc.).
- Almacenamiento masivo de la información de sensores para posterior análisis.

Como CPU del sistema se decide utilizar la plataforma **EDU-CIAA-NXP** [1] la cual contendrá toda la lógica del sistema incluido los calculos adicionales de filtros digitales y sistema de control.

2.1. Modelo de Hardware

En esta sección se desarrolla el proceso de diseño del hardware, dando como resultado un módulo adicional encastrable (Poncho) sobre la parte superior de la EDU-CIAA-NXP.
Según la planificación del proyecto, se decide diseñar para esta etapa, un "Poncho"que contenga todos los sensores que sean necesarios para un autopiloto genérico, de manera que pueda ser utilizado posteriormente por otros desarrolladores en distintas aplicaciones. Cabe destacar que el Poncho diseñado será el target de desarrollo del proyecto, esto quiere decir que los desarrollares del firmware tendrán su EDU CIAA con el correspondiente Poncho para desarrollo en paralelo de las diferentes funcionalidades y posteriores ensayos de las mismas.
Como se observa en la Figura 3, para el desarrollo del hardware se opta por utilizar un modelo de diseño modular y utilizar la mayor cantidad de módulos comerciales que sean adecuados para esta aplicación de forma de acelerar el proceso en esta fase.

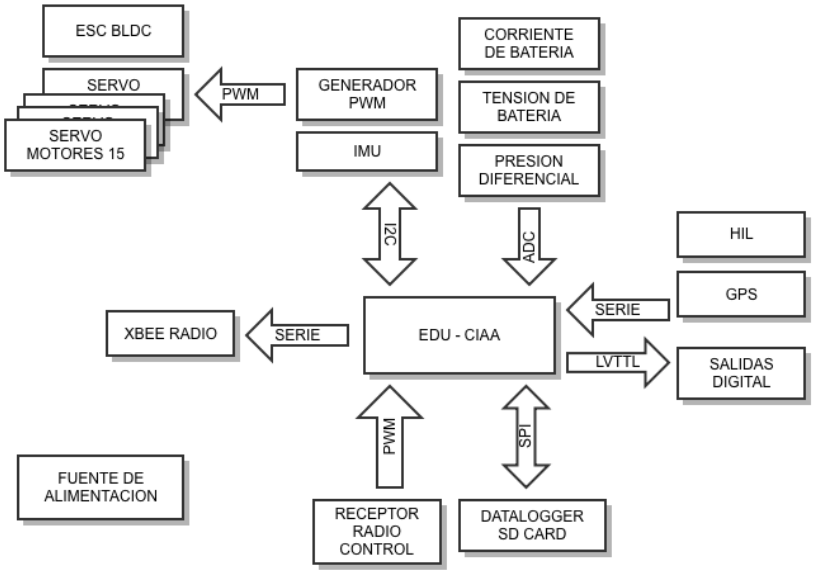
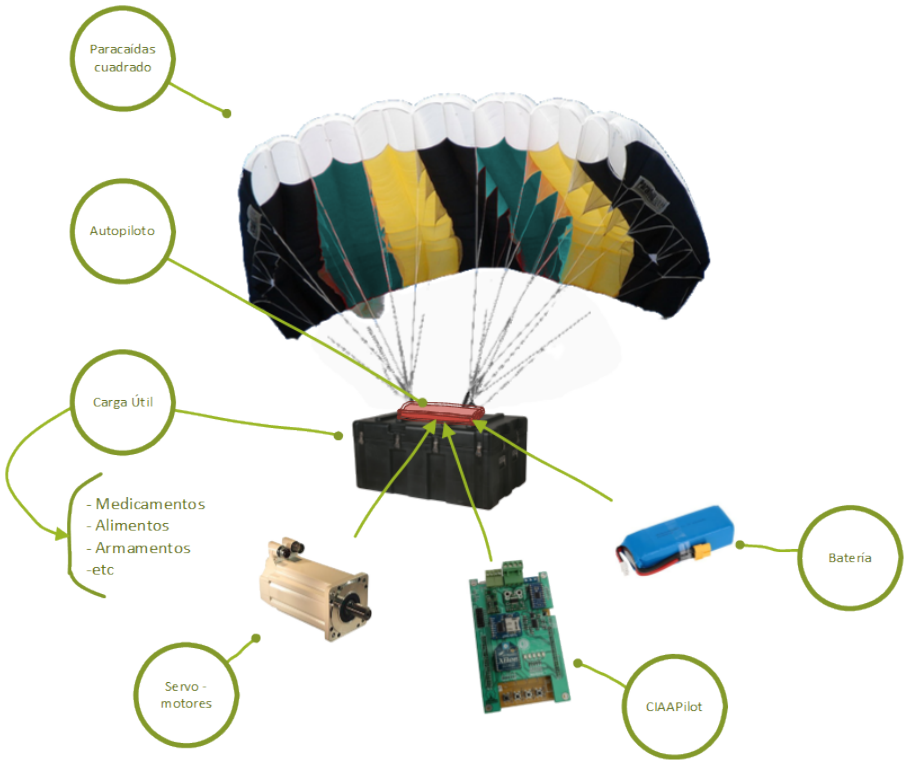


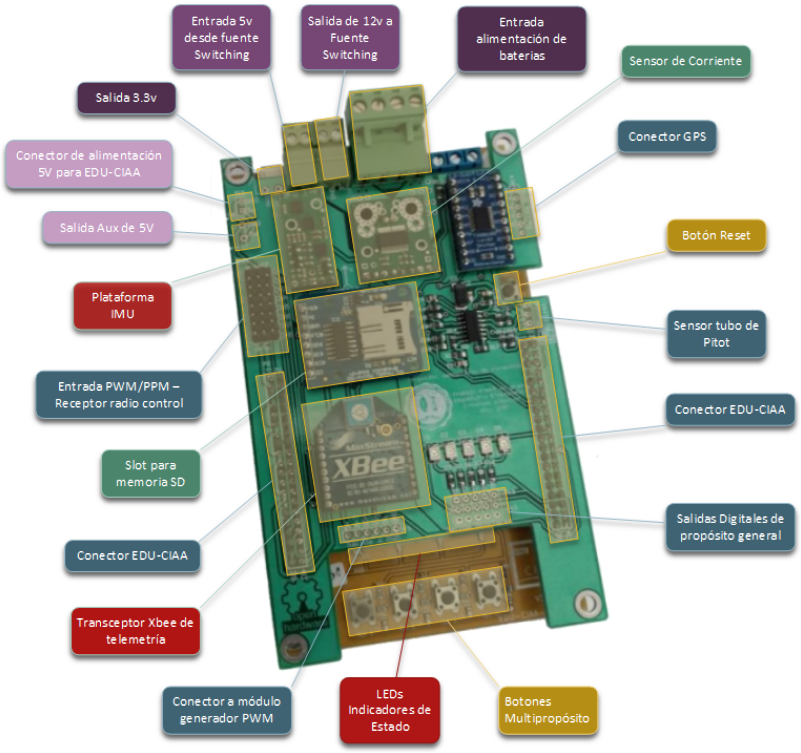
Figura 3: Diagrama general de Autopiloto

El hardware del piloto automático se encuentra dividido en 10 módulos como lo muestra la Figura 3 y son listados a continuación:

- Módulo alimentación
- Módulo imu



(a) Partes que componen el sistema



(b) Partes del Autopiloto

Figura 2: Esquemas de partes que componen el sistema

- Módulo pitot
- Módulo Batsens
- Módulo GPS
- Módulo Zigbee
- Módulo Sd Car
- Servos y controlador de motor brushless
- Salidas digitales de propósito general
- Entradas receptor radioRX

2.2. Modelo de Firmware

En esta sección se describe el proceso de desarrollo del firmware para el Autopiloto. Como se puede observar en la Figura 4, la lógica del sistema y las bibliotecas desarrolladas de encuentran en las capas de "Application" y "Library" respectivamente y pertenecen al desarrollo de este trabajo.
Las capas pertenecientes al CIAA Firmware, es decir, la implementación del RTOS, kernel layer, POSIX layer e Interface layer no fue necesario modificar y permanecen estándar. Por el contrario la capa Driver fue la única que se modificó por necesidades de asignación en la funcionalidad de los pines de salida para la aplicación de piloto automático.

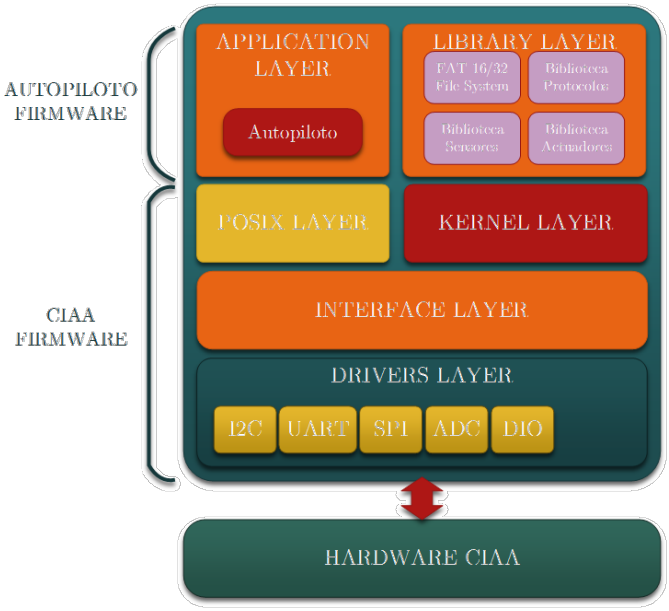


Figura 4: Diagrama del modelo de firmware empleado para Autopiloto

Modos de funcionamiento

El Autopiloto presenta dos modos de funcionamiento, *Modo Misión* (MissionMode) y *Modo HIL*. Por defecto el autopiloto se iniciará en *Modo Misión* y se ejecutará así durante toda su misión no pudiendo cambiar de modo. Para iniciar en *Modo HIL* es necesario usar una combinación de teclas de la *botonera multipropósito*.
El *Modo Misión* (MissionMode) ejecuta las tareas para adquisición de los sensores, procesamiento de los datos, procesamiento del sistema de control, telemetría y control de servos.
Por otro lado el *Modo HIL* coloca al autopiloto en espera para iniciar la funcionalidad de *Hardware In the Loop*, es decir que ejecuta las tareas de adquisición de comandos HIL por puerto USB/Serie, realiza el procesamiento de los datos, el procesamiento del sistema de control, telemetría, control de servos y realiza el envío de resultados sistema de control por USB/Serie.

Tareas de Inicialización

Las tareas de inicialización tienen la características que se ejecutan solo una vez al comienzo de la aplicación y su función principal es la de configurar los periféricos a utilizar por el sistema.

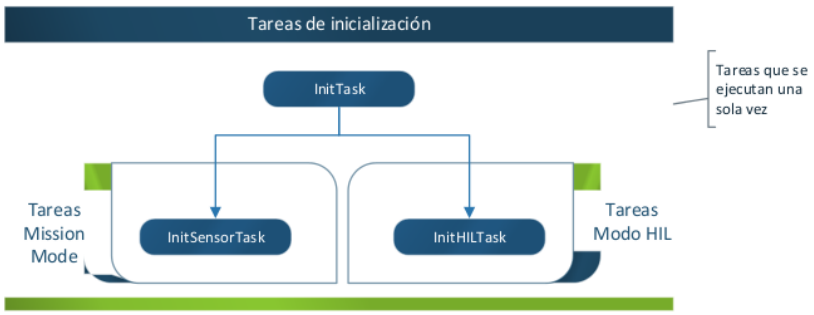


Figura 5: Tareas de Inicialización

La tarea InitTask inicializa el kernel CIAA y los dispositivos que van a ser usados en toda la aplicación independientemente del modo en que se inicie. A su vez se encarga leer si el botón HIL está o no presionado.
La tarea InitSensorTask inicializa los sensores en Modo Mision (MissionMode).
La tarea InitHILTask inicializa las variables y el puerto USB/serie que se usa para transmitir y recibir los parámetros HIL. Solo funciona en modo HIL.

Tareas de ejecución

Las tareas de ejecución comprenden a las tareas cíclicas que se ejecutan cada 10ms, 20ms y 50ms activadas con alarmas, las tareas de ejecución en Modo Mision (MissionMode), las tareas de ejecución en Modo HIL, las tareas referentes al cálculo del sistema de control (tarea cíclica) y las tareas de modificación de posición de servos y telemetría.

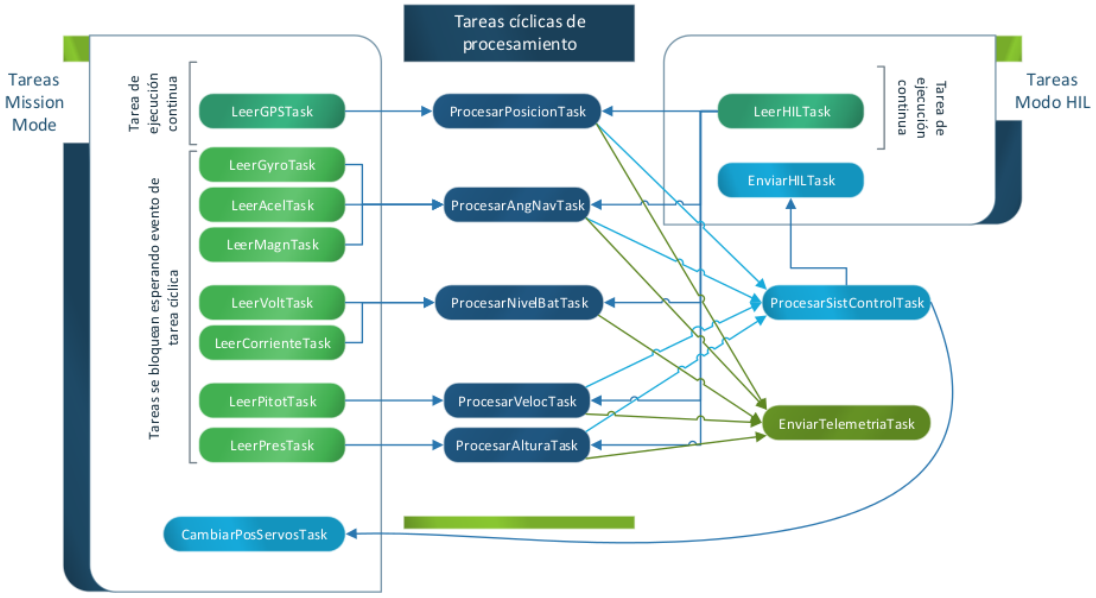


Figura 6: Diagrama de tareas de ejecución

Las tareas cíclicas tiene las menores prioridades por lo que al esperar un evento y pasar al estado Waiting, las tareas de lectura como por ejemplo LeerGyroTask, de mayor prioridad, pueden ejecutarse y generar el evento esperado por alguna tarea cíclica. De esta forma se evita que se produzca un Deadlock (bloqueo mutuo).

Secuencia de operación

La secuencia de operación se expone con el fin de resumir lo visto en los diagramas de las tareas anteriores, son puramente descriptivos de manera de facilitar la comprensión de funcionamientos.

Como se puede ver en la Figura 7 y en la Figura 8, las tareas pasan a un estado de Ready en función del modo de inicio del sistema (Normal o HIL). Es decir, por ejemplo, que en la inicialización solo se ejecuta la tarea de "Inicialización de sensores" si se está en Modo Mision (MissionMode).

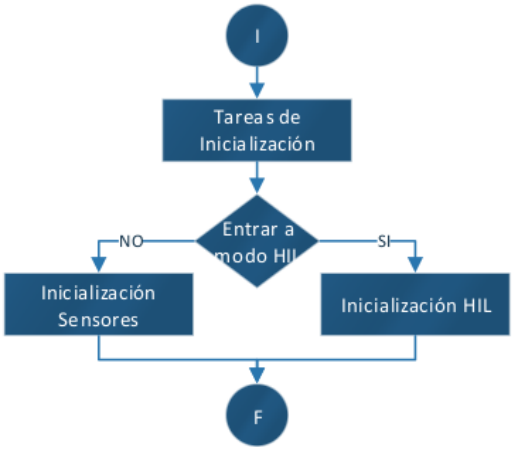


Figura 7: Secuencia de operaciones de inicialización

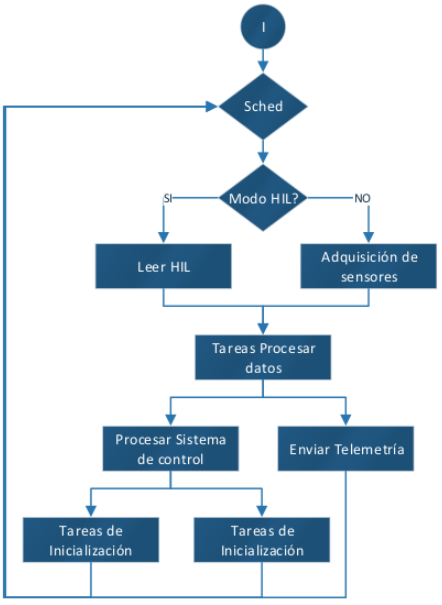


Figura 8: Secuencia de operaciones de proceso

Filtro complementario

El girómetro presenta derivas para tiempos largos y el acelerómetro presenta mediciones fuertemente afectadas por la estructura donde se presenta el modelo. Es por esta razón que surge la necesidad de utilizar las mediciones del girómetro en tiempos cortos y realizar la corrección de la deriva con las mediciones del acelerómetro.

Por lo tanto, es necesario hacer una combinación de las mediciones. Para ello, el método que se implementará es un filtro complementario.

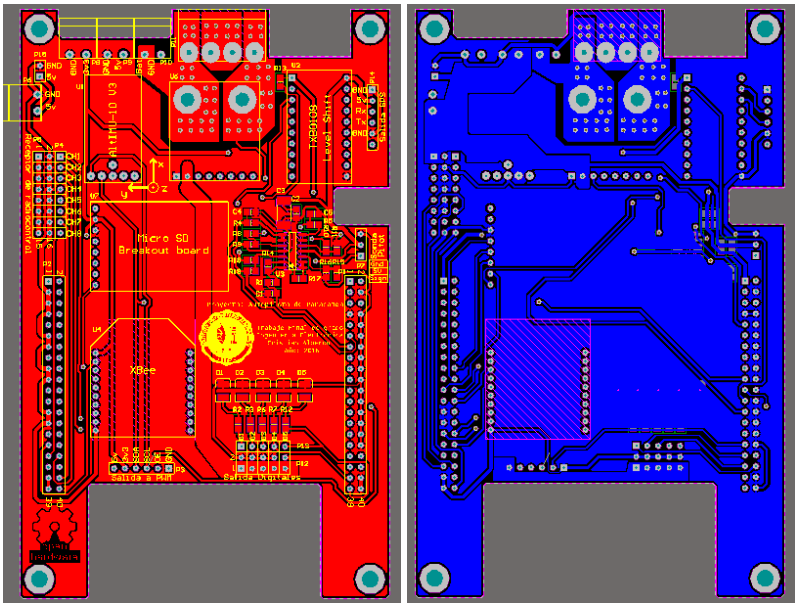
La fórmula resultante de combinar (complementar) los dos filtros se muestra en la ecuación 1

$$\theta = 0{,}98(\theta + \theta_G \Delta t) + 0{,}02\theta_A \tag{1}$$

3. RESULTADOS

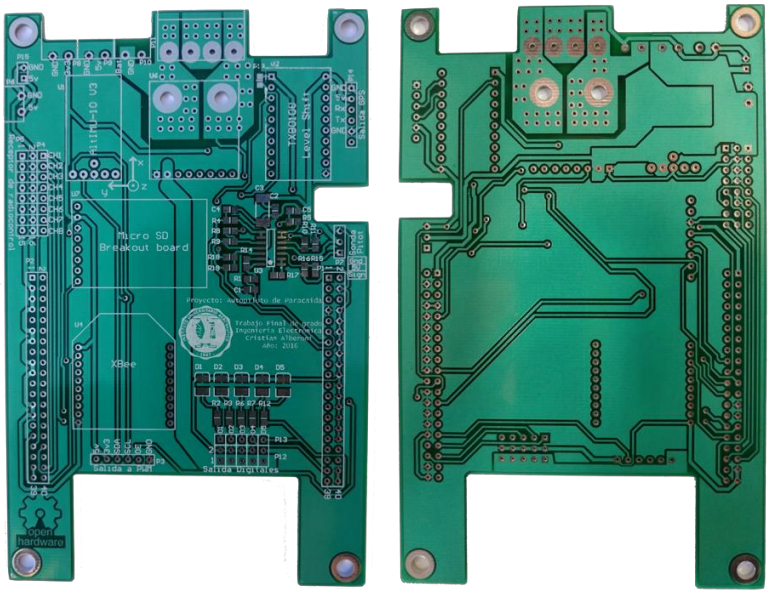
3.1. Poncho CIAAPilot

En la Figura 9, se muestran las capturas de pantalla del diseño del circuito impreso realizado en Altuim Designer®. Como resultado del proceso se muestra en la Figura 10. El circuito impreso fabricado previo al ensamblaje final. Se fabricaron tres placas de manera de poder montar dos placas completas y tener al menos una de repuesto en caso de presentarse algún inconveniente o se rompa el circuito en pruebas.



(a) Capa Top (b) Capa Bottom

Figura 9: Diseño de Circuito Impreso

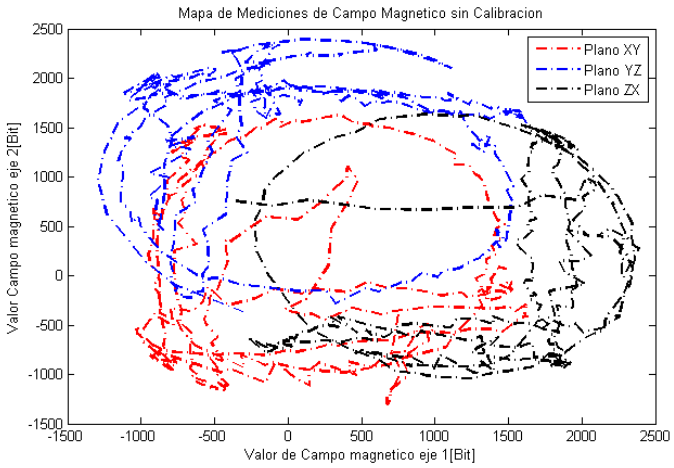


(a) Capa Top (b) Capa Bottom

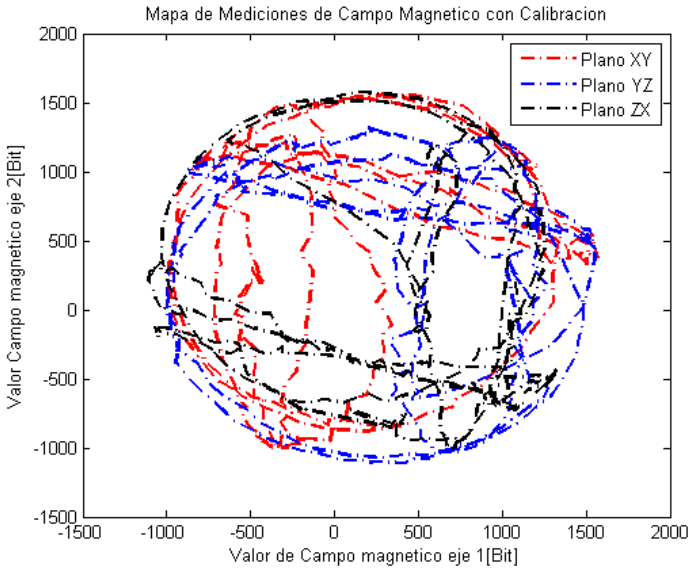
Figura 10: Circuito Impreso fabricado

3.2. Calibración de Magnetómetro

Debido a que el compás magnético se comporta como una brújula, es susceptible a interferencias de campo magnético a cuerpos ferrosos. Este tipo de distorsión es conocido como hard-iron producido por materiales que presentan una adición constante en el campo magnético de la tierra, generando de ese modo error a la salida de cada uno de los ejes del magnetómetro.
Para obtener estos valores es necesario ejecutar el programa de calibración una vez montado el piloto automático en su estructura final.
En la Figura 11(a) realizada en Matlab®, se pueden apreciar las mediciones sin calibración y en la Figura 11(b), se ven las mediciones calibradas.



(a) Sin calibración



(b) Con calibración

Figura 11: Mediciones de campo magnético con sensor calibrado y sin calibrar

3.3. Medición de filtro complementario

Como se puede observar en la Figura 12, se muestra los resultados obtenidos del ángulo calculado con el acelerómetro, luego con el girómetro y finalmente aplicando el filtro complementario [2].
Claramente se puede apreciar la deriva propia del girómetro y por otro lado las mediciones variables del acelerómetro que generan ruido al cálculo. Finalmente la linea de color negro perteneciente a las mediciones de ángulo aplicando el filtro, se ve suavizada y no presenta deriva, tal como se esperaba.

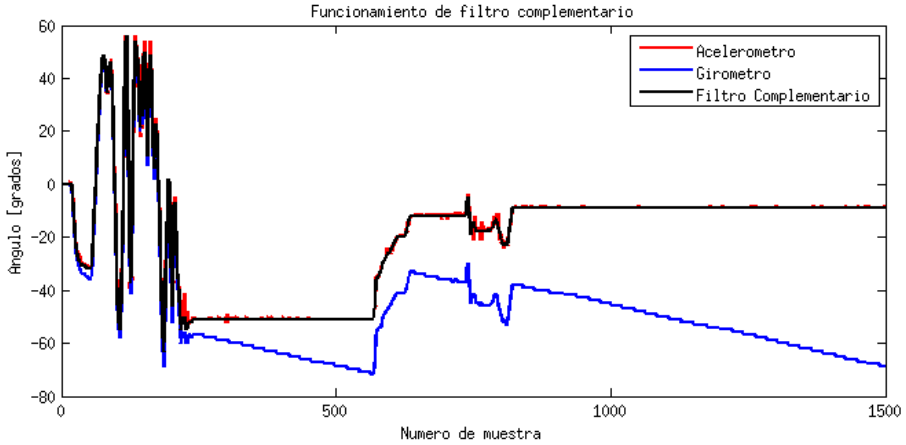


Figura 12: Gráfico comparativo de mediciones con y sin filtro

3.4. Medición de ángulos de navegación

Para la verificación de los ángulos de navegación se decidió construir un dispositivo que permite tener una base estable y garantizar la medición de cada ángulo por separado, es decir tener un solo grado de libertad.
En la Figura 13 se muestra el gráfico de linealidad de las mediciones de los ángulos de roll, pitch y yaw.

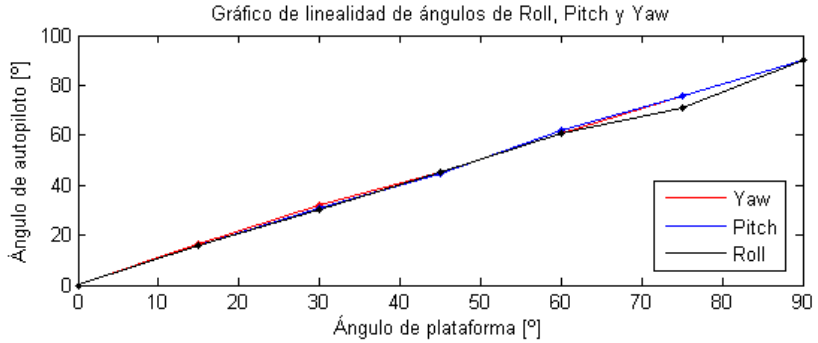


Figura 13: Linealidad de ángulos de navegación (Roll, Pitch y Yaw)

3.5. Medición de consumo de corriente

Se realizó una medición del consumo de corriente del circuito completo, el cual integra la fuente switching, la plataforma EDU-CIAA y el poncho CIAAPILOT montado sin actuadores (servomotores o motores brushless).
Los resultados se observan en la Figura 14, muestran la variación de consumo de corriente [mA] en función a la tensión de entrada.
En resumen, el circuito presenta un consumo de corriente media de 160 mA y un pico de corriente 210mA en 7,5V de tensión de entrada.

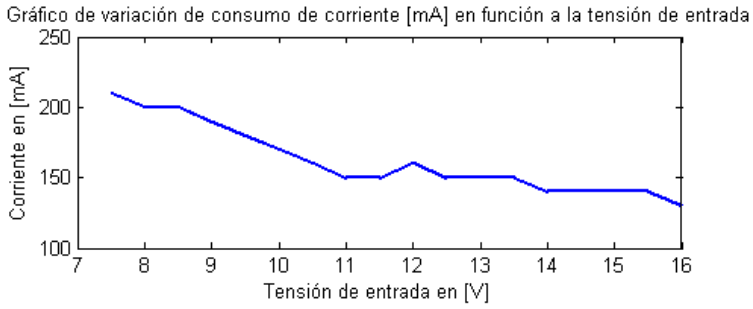


Figura 14: Variación de consumo de corriente en función de la tensión de entrada

4. CONCLUSIONES

La primera parte del presente trabajo consiste en el estudio de requerimientos y alcances del proyecto. Para esto, se analizan las variables a obtener y cómo medirlas. El paso siguiente es realizar un estudio de caso de uso y en función de este último, dividir el proyecto en módulos de desarrollo. Una vez fijado esto, se procede con la asignación de los tiempos y responsabilidades que cada tarea necesitará.
Paralelamente al desarrollo del firmware, se decide montar un target de desarrollo en una placa experimental de propósitos múltiples, de manera de trabajar en simultáneo con el desarrollo de hardware del Poncho. Este último será la versión de target final a utilizar por el equipo de desarrollo de firmware y en las pruebas iniciales de campo.
Luego se procede a diseñar cada uno de los módulos a nivel firmware y hardware, sometiendo cada uno de los mismos a pruebas de manera independiente.
Una vez superada satisfactoriamente la etapa anterior, se realiza el ensamble de cada uno de los módulos para obtener el sistema de control automático. A partir de éste se realizan las pruebas y ensayos del conjunto obteniéndose los resultados finales del trabajo.
Se corroboró que el uso por separado de los ángulos obtenidos por medio del acelerómetro y del girómetro no son viables para la aplicación de control de paracaída. Por otro lado la implementación del filtro complementario combina las mediciones anteriores y generan valores que pueden ser utilizados en la aplicación real.
Para finalizar se listan la serie de trabajos a futuros que siguen en la linea de desarrollo del proyecto.

- Update del sistema operativo a la versión CIAA Firmware UPA 1.0.0 LTS que se encuentra actualmente disponible, compilar todo el sistema y testear lo desarrollado.
- Implementar sistema de archivo FAT en version de RTOS LTS.
- Implementación de protocolo MavLink para telemetría.
- Lectura de trama RMC de NMEA en biblioteca de GPS.
- Implementar en driver de DIO el manejo del recurso de entrada digital por interrupciones para lectura por flancos de subida se señal PWM.
- Implementación de filtros de Kalman para cálculo de ángulos de navegación.
- Implementar modelo de sistema de control y realizar test de comportamiento.

REFERENCIAS

[1] ACSE-CADIEEL. (2014) EDU-CIAA-NXP computadora industrial abierta argentina. [Online; accessed 2016-09-09]. [Online]. Available: http://www.proyecto-ciaa.com.ar/devwiki/doku.php?id=desarrollo:edu-ciaa:edu-ciaa-nxp

[2] J. Gaydou, D. Redolfi and A. Henze, "Filtro complementario para estimación de actitud aplicado al controlador embebido de un cuatrirrotor," in *CASE, Congreso Argentino de Sistemas Embebidos, 2011*, Mar 2011, pp. 120–125.

[3] M. Barr and A. Massa, *Programming Embedded Systems: With C and GNU Development Tools, 2nd Edition*. O'Reilly Media, 2006.

[4] J. Catsoulis, *Designing Embedded Hardware*. O'Reilly Media, 2005.

[5] G. Stringham, *Hardware/Firmware Interface Design: Best Practices for Improving Embedded Systems Development*. Newnes, 2009.