INSTITUTO UNIVERSITARIO AERONAUTICO

Trabajo Final de Grado

Facultad de Ingeniería - Ingeniería Electrónica

Título: *"Diseño y desarrollo de Autopiloto de paracaída implementado en Computadora Industrial Abierta Argentina (CIAA)"*

REVISIÓN: C

Autor:

Cristian Alberoni

Tutor:

Ing. Javier Fernandez

Córdoba, 01 de agosto de 2016

HISTORIAL DE REVISIONES		
Revisión	**Fecha**	**Comentarios**
A	28/07/2016	Revisión Inicial
B	01/08/2016	Se agregan correcciones en conclusión y punto 9.1.2. Revisó: Ing. Javier Fernandez
C	18/10/2016	Se da formato a bibliografía según normas APA

DEDICATORIA

A mis Familiares y amigos,

A mis colegas de trabajo,

AGRADECIMIENTOS

A mis padres, por su entrega, apoyo, sacrificio, y ejemplo.

A mi hermano, por compañía y cariño.

A Ing. Javier Fernández por haberme acompañado y orientado durante el desarrollo de mi trabajo.

A Ing. Pablo Sonna por haberme orientado en la elección del tema desarrollado.

A Ing. José Duclox por su colaboración y predisposición permanente.

A los Ingenieros del Departamento de Mecánica Aeronaútica, Ing. Diego Llorens, Ing. Germán Weht, Ing. Andres Liberatto, Ing. Esteban Gonzalez.

A la Lic. Valeria Maurizi por su desinterezada colaboración al momento de realizar la revisión y corrección de la redacción.

A Ing. Juan Galleguillo por haberme guiado durante mi carrera, por su permanente apoyo y generosa entrega.

A Ing. Marcela Busnardo por las innumerables oportunidades ofrecidas tan generosamente. Por su visión y asesoramiento.

Al personal de la institución.

A mis amigos y compañeros.

Título: *"Diseño y desarrollo de Autopiloto de paracaída implementado en Computadora Industrial Abierta Argentina (CIAA)"*

INDICE

1. GLOSARIO, LISTADO DE TÉRMINOS Y CONVENCIONES

Api (Application Programming Interface)

C (Lenguaje De Programación)

C++ (Lenguaje De Programación)

Csp (Comunicating Sequential Processes)

Dos (Sistemas Operativos Para Pc)

Faa (Fuerza Aerea Argentina)

Fifo (First In First Out - Primero En Entrar Primero En Salir)

Gcc (Gnu Compiler Collection)

Gnu (Proyecto Para Un Sistema Operativo Totalmente Libre)

Gpio (General Propouse Input Output)

Gps (Sistema De Posisionamiento Global)

Ieee (Instituto De Ingenieros Eléctricos Y Electrónicos)

Iua (Instituto Universitario Aeronautico)

Lan (Local Area Network)

Mil-Std (Militar Standard)

Os (Sistema Operativo)

Piddef (Proyecto Investigación De Defensa)

Pc (Personal Computer)

Rt (Remote Terminal)

Rtos (Sistema Operativo De Tiempo Real)

Tfg (Trabajo Final De Grado)

Figuras y Tablas

2. RESUMEN

La entrega aérea o lanzamiento con paracaídas, es un tipo de puente aéreo desarrollado para el reabastecimiento de tropas o entrega de suministros para ayuda humanitaria cuando es innaccesible 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 dedido 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 un piloto automático para comandar un paracaída y a traves de este sistema, es aumentar la precisión del sistema es decir disminuir sustancialmente el radio de caida 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.

3. PALABRAS CLAVES

Piloto automático, paracaída, autopiloto, computadora industrial abierta Argentina (CIAA), sistemas embebidos.

4. INTRODUCCION

El presente trabajo final tiene como propósito el diseño y desarrollo de un sistema electrónico de precisión para el comando autónomo de paracaídas, destinado a la entrega de suministros críticos a las unidades desplegadas en el campo de batalla, y para aplicaciones fuera del ámbito militar: como situaciones de catástrofes y misiones de ayuda donde se necesite la entrega de sumistros (Figura 1).

Este proyecto se está llevando a cabo en el marco del programa de investigación y desarrollo para el ministerio de defensa (PIDDEF).

Figura 1 - Lanzamiendo de Cargas desde C-17 Globemaster III

El proyecto tiene por finalidad en esta etapa, la creación de un sistema embebido capaz de adquirir datos durante la caída de la carga y mediante estos parámetros

poder ser procesados por el sistema de control implementado y tomar las decisiones de vuelo según se hayan planificado.

Es por ello que en esta etapa, se destaca la importancia de la utilización de la Computadora Industrial Abierta Argentina (Proyecto CIAA) como plataforma de Hardware de desarrollo embebido para los prototipos (Figura 2). A esta plataforma se la proveerá de todos los sensores necesarios que requiere para su funcionamiento.

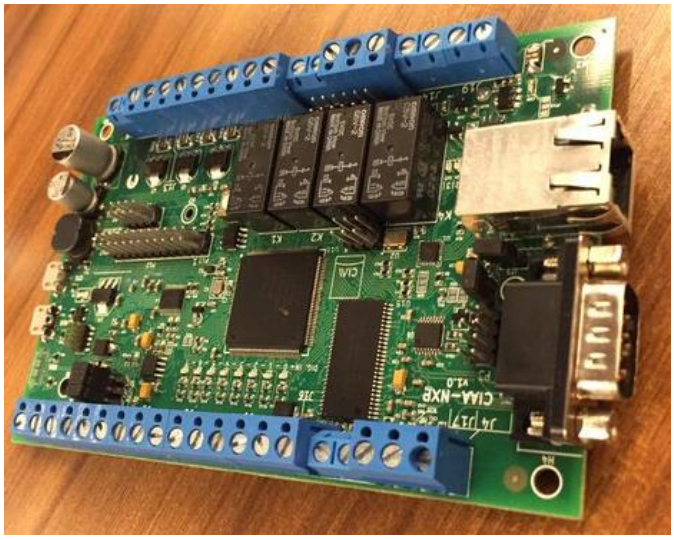


Figura 2 - Plataforma CIAA en su versión CIAA-NXP

En el presente trabajo se desarrollará el software de aplicación embebida que funcionará en la plataforma EDU-CIAA, excluyendo a los algoritmos de control, los cuales son desarrollados por el Departamento de Mecánica Aeronáutica del IUA.

5. OBJETIVOS DEL PROYECTO

5.1 OBJETIVOS GENERALES

Diseñar e Implementar un sistema de Autopiloto para paracaída utilizando la Computadora Industrial Abierta Argentina (CIAA) en su versión EDU-CIAA.

5.2 OBJETIVOS ESPECÍFICOS

- Desarrollar módulo IMU (Unidad Inercial) - adquisición de los sensores de presión barométrica, aceleraciones, velocidad angular, campo magnético terrestre, temperatura de aire.
- Desarrollar módulo Pitot - adquisición de sensor de presión dinámica.
- Desarrollar módulo de Telemetría
- Desarrollar módulo GPS
- Desarrollar módulo SD Card – almacenado de información relevante en memoria SD Card (Secure Digital Memory Card)
- Desarrollar módulo de control de motor y servomotores
- Desarrollar módulo de monitoreo de corriente y tensión de Baterías
- Diseñar la plataforma que contendrá todos los sensores.
- Construir PCB de la plataforma de sensores que pueda ser montada sobre la EDU-CIAA (llamada "Ponchos" en proyecto CIAA).
- Unificar los módulos desarrollados a fin de obtener el control automático del sistema.
- Ensayar el comportamiento del sistema.

- Analizar los resultados obtenidos.

6. DESTINATARIOS

Este trabajo se enmarca dentro de un proyecto de desarrollo e investigación del Departamento de Electrónica y Telecomunicaciones del Instituto Universitario Aeronáutico que tiene como destinatario al proyecto PIDDEF "Paracaídas Comandado Autónomo para Entrega de Cargas" perteneciente al Departamento de Mecánica Aeronáutica del mismo Instituto. A su vez este proyecto cuenta con la aprobación del MINCYT (Ministerio de Ciencia y Técnica de la Nación).

Así también puede ser beneficiario cualquier otra entidad que realice, actividades relacionadas con misiones de ayuda, elementos y víveres a personas en situación de aislamiento producida como por ejemplo por inundaciones (islas del Paraná, norte de Formosa y pequeñas aldeas en regiones selváticas del noroeste Argentino y sur de Bolivia) y brindar apoyo al Plan Nacional de Manejo del Fuego, asistiendo a rescatistas mediante el envío de agua, equipos y medicamentos a regiones anegadas por el fuego (Bariloche, Córdoba, San Martín de los Andes y RosarioVictoria).

7. BENEFICIOS ESPERADOS

Los beneficios esperados con el diseño e implementación de este sistema son:

- Aumento de la capacidad de procesamiento para cálculos del sistema de control ya que se cuenta originalmente con un piloto automático basado en la plataforma Arduino (Figura 3) de inferior capacidad de procesamientos de información.

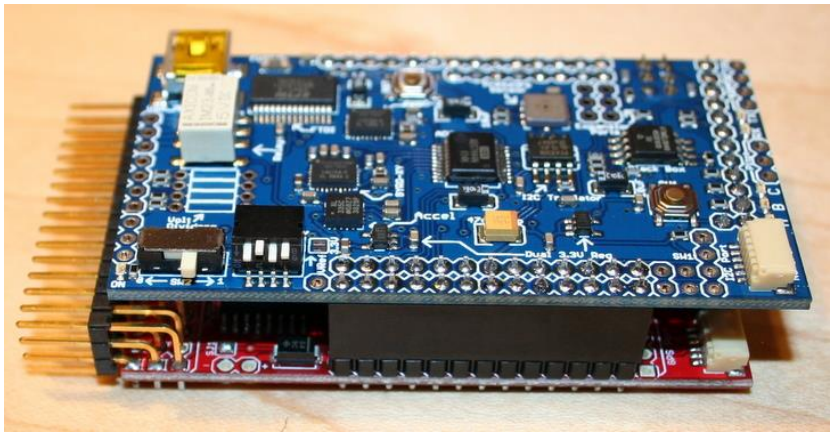


Figura 3 - ArduPilot Mega (APM) and the IMU Sensor Board (APM Shield/Oilpan)

- Contar con plataforma de hardware y firmware propietario.

A pesar de que la plataforma actual es de Hardware y Software libre en su mayoría, sigue siendo de uso genérico y diseñado para hobbistas.

- Escalabilidad del sistema, tanto de hardware como firmware.
- Al plantear un esquema de desarrollo modular, tanto en Hardware como en Software, permite que el sistema se escale en función a la evolución del proyecto.Lograr un sistema robusto y estándar bajo normas militares.
- Aumento de eficiencia del sistema.
- Generar la documentación necesaria para poder recrear el modelo en cualquier momento y bajo las mismas normas.

8. ESTUDIO TECNICOS

Como se ha comentado en el punto 4 - INTRODUCCION, en el presenta trabajo se desarrollará y construirá el sistema electrónico que adquiera la información de

los sensores necesarios, procese esta información y genere las señales de control a los actuadores correspondientes, permitiendo comandar al paracaídas.

Figura 4 - Sistema de paracaída no tripulado

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 a punto de destino preestablecido, por ejemplo, mediante coordenadas GPS. 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. También permitiría 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, quizás en un único viaje.

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.

8.1 PARACAÍDAS

El paracaídas es, como su nombre indica, un dispositivo utilizado para reducir la velocidad de un objeto en caída libre.

Hay dos elementos básicos para que los paracaídas retarden el descenso: *sustentación* y *resistencia* al avance. Un paracaídas redondo crea resistencia al avance frenando la carga, simplemente capturando tanto aire como puede. Pero un paracaídas rectangular crea sustentación, la cual impulsa a la vela en una dirección particular determinada por el diseño del ala y su presentación al fluido en el que mueve. Controlar el flujo de aire encima del ala es el arte del piloto de velamen.

Sustentación

Un velamen produce sustentación de dos maneras. La propia forma del ala produce alguna sustentación. Las alas se conforman para que el aire fluya más rápidamente por encima de ella que por debajo. Cuando la velocidad del aire aumenta, su presión disminuye. Creando un área de baja presión en la parte superior, y una presión más alta debajo de ella. Así el ala es "llevada" hacia el área de presión baja.

La deflexión de aire es el segundo tipo de sustentación. Si el aire se desvía hacia un lado, debe haber una reacción igual en la dirección opuesta, el mismo principio que nos hace girar, "trackear", y desarrollar otras maniobras de caída libre.

El balance entre deflexión y sustentación es complejo. Si la deflexión fuera la fuerza principal de sustentación, en un giro con comandos hacia la derecha (el borde derecho del ala tirado hacia abajo) el aire deflectado empujaría el lado derecho del velamen, en una inclinación a la izquierda y crearía un giro a la izquierda. Pero de hecho, tirando hacia abajo el comando derecho reduce la sustentación, porque aumenta la resistencia en ese lado. Con el lado derecho moviéndose más lentamente, se crea menos sustentación. El velamen gira a la derecha.

El principal uso de la deflexión en el paracaidismo es en el momento del planeo final "flare". Cuando se ejecuta un "flare", algo del aire se deflecta hacia abajo con la consiguiente elevación del velamen. Pero esto también aumenta la resistencia retardando la velocidad de avance del paracaídas. El piloto suspendido, teniendo más masa y menos resistencia, no reduce su velocidad tan rápido y gira hacia delante. Esto cambia completamente el ángulo de ataque, mayormente aumentando la deflexión del aire mientras exista cualquier velocidad. Se verá más de cerca este uso de la deflexión cuando se discuta el ángulo de ataque, y en los capítulos de técnicas prácticas de vuelo.

Resistencia

La otra fuerza principal que actúa en un velamen es la resistencia. La resistencia también tiene además dos manifestaciones que se llamarán resistencia de forma y resistencia parásita. De un modo simple, la resistencia de forma es el resultado de la fricción entre el flujo de aire y el ala.

Es una penalidad en que todas las alas incurren de algún modo y puede pensarse como una sustentación hacia la parte de atrás. La resistencia parásita en cambio es el resultado de las interrupciones del flujo de aire por las irregularidades propias del ala. Las celdas abiertas crean turbulencia. Las costuras, cintas de plegado "packing tabs", cuerdas "lines" y sus puntos de vinculación, el pilotín, el pañal "slider", e incluso el piloto, contribuyen a la resistencia pero nada a la sustentación. Los paracaídas nunca han sido alas muy eficientes comparados con los aviones porque su propia estructura crea una gran cantidad de resistencias parásitas.

Sustentación y resistencia, entonces, son ambos resultados del flujo de aire encima de un ala. Porque es el flujo de aire encima del ala el que crea éstas fuerzas de vuelo, más flujo implica más fuerza. La sustentación y resistencia aumentan en proporción geométrica a la velocidad: dos veces la velocidad, cuatro veces la sustentación, y lo mismo para la resistencia. Esto indica que la velocidad aérea es crucial para la performance. Ir más rápido, significa en este punto mayor sustentación y una inmediata respuesta de los comandos. Significa también que la resistencia aumenta, por lo que los velámenes más rápidos tienen distintas características de diseño, para reducir la resistencia, como pilotines colapsables, pañales removibles, y cuerdas de pequeño diámetro.

Tipos de paracaídas

- *Parasol:* este primer caso era utilizado por los artistas chinos en las actuaciones para simular una caída lenta. Era una acción en pequeña escala de lo que vendría en los siguientes años.

- *Cónico:* puede decirse que este es el primer paracaídas que existió. Con forma de cono fue creado en el año 1470 en Italia con las bases del gran Leonardo Da Vinci. El objetivo por el que se fabricó dicho paracaídas fue para que en caso de incendio de un edificio las personas pudieran saltar de el con su ayuda.

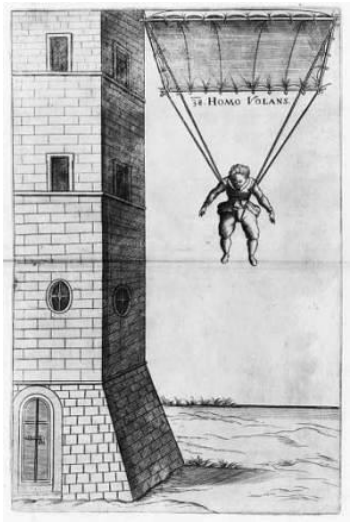


Figura 5 - Paracaída cónico

- *Piramidal:* este paracaídas tiene forma de pirámide (Figura 6 - Paracaída piramidal) formada por maderas en sus aristas y en la base, que conectadas entre ellas por telas creaban un entorno cerrado que permitía frenar la caída. Este caso también está basando en proyectos de Leonardo Da Vinci.

Figura 6 - Paracaída piramidal

- *Redondo:* En este caso pueden dividirse en dos tipos, manejables o no manejables. Estos últimos (

Figura 7 - Paracaida redondo) se destinan al lanzamiento de mercancías en misiones principalmente, ya que al no poder ser controlados, solo siguen las corrientes de aire hasta llegar al suelo. Estos tipos de paracaídas ya casi no se encuentran, al menos los manejables, ya que su origen fue en el sigo XIX, pero utilizados en la segunda guerra mundial y compuestos por seda o nylon. El diámetro en el caso de ser usado por un ser humano rondaba los 7 metros.

Figura 7 - Paracaida redondo

- *Cuadrados:* son actualmente los que se usan en modalidad deportiva ya que su forma ayuda a un mejor control de la dirección en el aire a la vez de obtener mayor velocidad y de planear más. Estos paracaídas están compuestos por dos paneles grandes paralelos colocados en horizontal y unidos por otros más pequeños verticalmente, creando cuadrados por donde circula el aire. La parte posterior de esta formación está cerrada para lograr planear mejor e incluso en ciertas ocasiones en que se alcanza bastante velocidad se puede ascender.

Figura 8 - Paracaída rectángulo

- *Forma de anillo:* son usados para frenar aviones o coches que van a grandes velocidades. Normalmente otro paracaídas no soportaría y acabaría rompiéndose, ya que en este caso lleva un agujero en el centro para disminuir la presión.

Figura 9 - Paracaida de Anillo

8.2 AUTOPILOTO – PILOTO AUTOMÁTICO

La función básica del piloto automático es la de controlar el vuelo de la aeronave y mantener en una ruta predeterminada en el espacio sin que se requiera ninguna acción por parte del piloto. El piloto automático puede por lo tanto, aliviar al piloto de la fatiga y la tediosa tarea de tener que mantener el control continuo de la trayectoria de vuelo de la aeronave en un vuelo de larga duración. De esta forma el piloto es libre para concentrarse en otras tareas y en la gestión de la misión.

Un sistema de piloto automático bien diseñado que esté integrado correctamente con el sistema de control de vuelo de la aeronave puede alcanzar una respuesta más rápida y mantener una trayectoria de vuelo más preciso que el piloto. Aún

más importante, la respuesta de un piloto automático es siempre constante, mientras que la respuesta de un piloto puede verse afectada por la fatiga, carga de trabajo y estrés.

Para continuar el entendimiento de lo que se busca, debe introducirse el concepto de ángulos de navegación.

Ángulos de navegación

Los ángulos de navegación, llamados en matemáticas ángulos de Tait-Bryan, son tres coordenadas angulares que definen un triedro rotado desde otro que se considera el sistema de referencia. Se definen matemáticamente de forma similar a los ángulos de Euler, pero en vez de usar como línea de nodos el corte entre dos planos homólogos (por ejemplo el XY es el homólogo del xy), se utilizan dos planos no homólogos (por ejemplo XY e yz).

Por ejemplo, en el dibujo adjunto que usa el convenio ZXY, se definen mediante los planos xy (plano perpendicular al eje "z" usado en la primera rotación) del sistema de referencia, y el plano ZX del sistema móvil (plano perpendicular al eje "Y" de la última rotación).

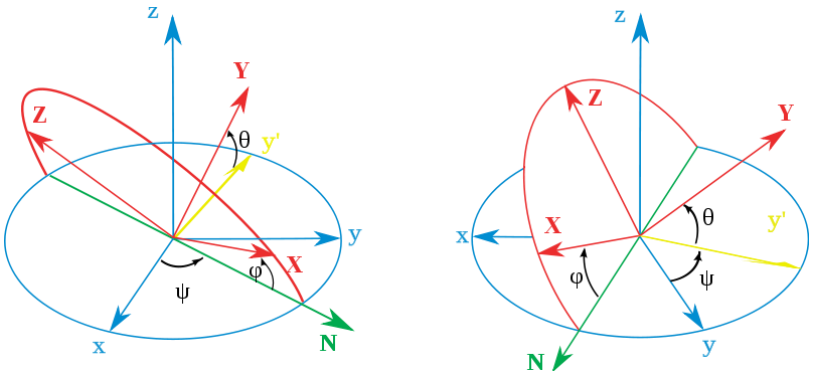


Figura 10 - Ángulos de navegación según convención ZXY definidos estáticamente

Los tres ángulos son la dirección (heading o yaw), elevación (pitch) y ángulo de alabeo (roll).

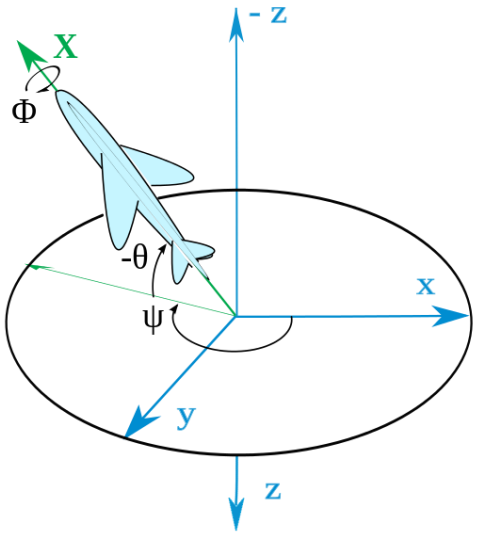


Figura 11 - Ángulos de navegación según convención ZYX

Estos tres ángulos son equivalentes a tres maniobras consecutivas. Dado un sistema de tres ejes fijos en el aeroplano, llamados eje de guiñada (yaw en inglés), de cabeceo (pitch) y de alabeo (roll). Existen tres rotaciones principales, normalmente llamadas igual que el eje sobre el que se producen, que permiten alcanzar el sistema del aeroplano desde el sistema de referencia. Tienen que venir dadas en ese orden y ser realizadas en ese orden, ya que el resultado final depende del orden en que se apliquen.

- Cabeceo: es una inclinación del morro del avión, o rotación respecto al eje ala-ala.
- Alabeo: rotación respecto de un eje morro-cola del avión.
- Guiñada: rotación intrínseca alrededor del eje vertical perpendicular al avión.

Principio de funcionamiento básico

Los pilotos automáticos modernos usan sistemas informáticos para controlar la aeronave. En una aeronave, el sistema de navegación calcula su posición actual y envía estos datos al sistema de gestión de vuelo que, a su vez, envía las correciones pertinentes de rumbo, y altitud, entre otros, al piloto automático, el cual hace actuar las superficies de vuelo del aparato. En un sistema de este tipo, además de los controles de vuelo clásicos, muchas aeronaves incorporan la capacidad de gestionar el empuje mediante el autothrottle, para controlar el flujo de combustible de los motores y optimizar la velocidad de crucero, descenso y ascenso.

El piloto automático lee la localización y posición de la aeronave de un sistema de guía inercial. Estos sistemas acumulan errores con el tiempo, por lo que incorporan sistemas de reducción de error, como el sistema carrusel que gira una vez por minuto de forma que los errores se disipen en diferentes direcciones y tengan un efecto global nulo. El error en los giróscopos se conoce como deriva y se debe a las propiedades físicas del sistema, ya sea mecánico o guiado por láser, que corrompen los datos de posición. Las diferencias entre los dos se resuelven con la ayuda del procesamiento digital de señales, normalmente con un filtro de Kalman hexadimensional. Las seis dimensiones suelen ser balanceo (roll), inclinación (pitch), orientación (yaw), altitud, latitud y longitud. La aeronave puede volar rutas que tienen un factor de rendimiento exigido, por lo que la cantidad de error o factor de rendimiento real debe ser monitorizado para poder volver dichas rutas particulares. Cuanto más largo sea el vuelo, mayor será el error acumulado en el sistema.

Las ayudas de radio, tales como DME, actualizaciones DME y GPS, pueden usarse para corregir la posición de la aeronave. Las unidades de referencia inercial, por ejemplo giróscopos, son la base del cálculo de localización a bordo (ya que el GPS y otros sistemas de radio dependen de un tercero que proporcione información). Dichas unidades son totalmente autocontenidas y usan la gravedad y la rotación terrestre para determinar su posición inicial. De esta forma, miden la aceleración para calcular dónde se encuentran en relación a donde empezaron. A partir de la aceleración puede calcularse la velocidad y de ésta la distancia. En cuanto se sabe la dirección (gracias a acelerómetros), las unidades de referencia inercial pueden determinar dónde están (con ayuda de software adecuado).

8.3 FORMULACIÓN Y VALORACIÓN DE ALTERNATIVAS

A continuación se presentan las alternativas de pilotos automáticos tanto comerciales como así también proyectos de hardware y software libre disponibles en el mercado.

Alternativas comerciales:

Piccolo autopilots

Figura 12 - Autopiloto Piccolo II

Especificaciones

Tabla 1 - Especificaciones autopiloto Piccolo II

MECHANICAL	
Dimensions	5.59 x 1.81 x 2.46 inches (142.00 x 46.00 x 62.60 mm) Unflanged
Weight	7.7 oz. (200 g) with 900 MHz radio
Enclosure/Mounting	EMI shielded carbon, flanged, unflanged
ENVIRONMENTAL & POWER	
Operating Case Temperature	-40 ° C to 80 ° C (calibrated range, no case)
Power Requirements	VIN: 8 - 20 volts Power: 4 W (typical including 900 MHz radio)
GENERAL	
RS232 Payload Interface	5
CAN	Simulation / General interface
Flight Termination	Deadman output
Digital/Analog I/O	16 configurable GPIO lines. 4 GPIO lines can be configured as analog inputs, 0-5V input, 10 bit conversion

Integrated RF Data Link Options	900 MHz unlicensed ISM. 900 MHz Australian band. 2.4 GHz unlicensed ISM. 310-390 MHz discrete. 1350-1390 MHz discrete. 1670-1700 MHz discrete.
GPS	4 Hz uBlox module GPS receiver, 5 volt
Pressure Sensors	Ported static. 15-115 KPa-ported pitot. 4 Kpa differential. 155 kts max indicated airspeed.
Waypoint Navigation	1000 waypoints saved in autopilot
Inertial Sensors	3 axis gyroscopes, 300°/sec. 3 axis acceleration, 10g
Supported Peripherals	Transponders, secondary comms radios, Iridium SatComm, TASE gimbals, servo PTZ gimbals, magnetometers, laser altimeters, payload passthrough, RTK GPS.
SOFTWARE OPTIONS	
Radio Options	900 MHz unlicensed ISM 900 MHz Australian band 2.4 GHz unlicensed ISM 310-390 MHz discrete 1350-1390 MHz discrete 1670-1700 MHz discrete
Standard Feature Set + Peripherals	Adds new support for pan-tilt servos, improved GPS/INS performance, and more flexibility in configuring payload ports.
Laser Altimeter Autoland	Provides accurate altitude information allowing the vehicle to perform a soft flared landing (Laser altimeter hardware sold separately).
DGPS Autoloand	Extends the autoland performance by using 2 cm accuracy DGPS. Supports autonomous taxi, rolling take-off, stationary and moving net recovery. Uses NovAtel DGPS equipment.

DGPS and Moving Net Recovery	Adds support of moving net recovery needed for shipboard and other moving capture applications. Uses NovAtel DGPS equipment and associated antennas.
Helicopter Operations (VTOL)	Includes take-off and landing, precision hover, and automated path following along with autopilot assisted manual steering modes.

Micropilot

Figura 13 - Autopiloto MicroPilot

Tabla 2 - Especificaciones autopiloto MicroPilot

Specifications

Servos
- Elevon, flaperons, 4 servo flap/aileron, separate flaps, v-tail, x-tail, split rudders
- 12 servo outputs
- 50 Hz servo update rate
- Separate servo and main battery power supply
- Separate voltage monitor for main and servo battery power supplies
- Integrated RC override
- 11 bit servo resolution

Control System
- 30 Hz PID loop update rate
- Gain scheduling for optimum performance
- Rudder aileron feed forward for improved turn performance
- Aileron/elevator feed forward for improved altitude hold during turns
- Autonomous takeoff and landing
- User definable PID feedback loops
- User definable table lookup functions

Navigation
- 4 Hz GPS update rate
- Move servo at waypoint
- Change altitude at waypoint
- Change airspeed at waypoint
- User definable holding patterns
- User definable error handlers
- RPV and UAV modes
- Supports GFPS accuracy
- 1,000 waypoint command buffer

Telemetry, Datalog and Video
- Telemetry (100 user definable fields transmitted each second)
- 5 to 30 Hz telemetry update rate
- Onboard datalog: 52 standard fields, 24 customizable, 3 MB
- 5 or 30 Hz datalog update rate
- Can record for approximately 4.5 hours at 5 Hz

Sensors
- Airspeed max speed: 500kph
- Altimeter max altitude 12,000m
- 2G, 3-axis accelerometers
- 3-axis rate gyro
- Max angular rate: 300° per sec

Physical Characteristics
- Embedded long-range data communication link, frequency-hopping, spread spectrum 2.4 GHz, 900 MHz, other frequency optional
- 8 high current drivers,
- 8 analog sensor inputs to be displayed on the GCS
- 2 control modes
  - Autopilot mode (UAV/RVP)
  - Manually piloted mode for emergency response
- Wide range of input voltage (6.5-30v)
- Weight 413 g
- L146 mm x W81.7 mm x H46 mm
- Failsafe watchdog timer

Kestrel autopilot

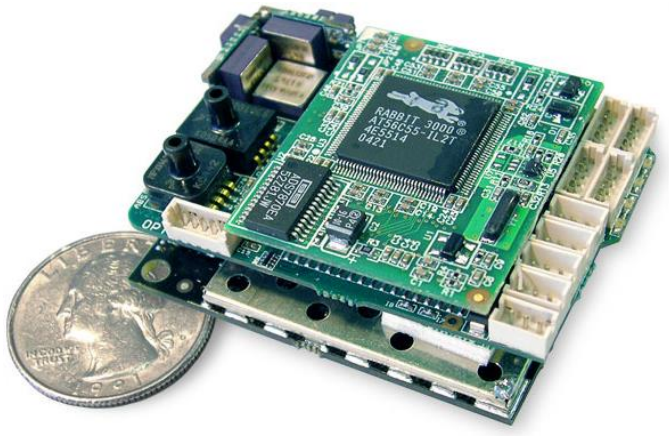


Figura 14 - Autopiloto Kestrel

Kestrel™ Autopilot v3.0 **Q409**

- Fixed Wing flight control avionics
- Heli / tri-rotor / quad-rotor (Advanced PID)
- 500mhz DSP, 32Mb flash, 32Mb RAM
- 11 servos supported onboard
- Integrated 3axis magnetometer
- High quality pitot / static systems
- Factory calibrated, temp. compensated IMU
- 17 state EKF navigation solution
- High accuracy gimbal pointing / geo-location
- 51 pin Omnetics connector
- May be used with other ground station software

** Fixed wing & vtol versions priced differently

Alternativas de software y hardware libre:

Paparazzi Apogee:

Figura 15 – Paparazzi Apogee Autopilot

- STMicroelectronics STM32F405RGT6 Cortex M4 168MHz processor featuring a Floating point unit (FPU), up to 192k of RAM and 1024k of FLASH.
- 9(6) DOF integrated IMU MPU-9150(6050) based
- 1 x Barometer/altimeter MPL3115A2 (I2C, MPU slave capability)
- 1 x MicroSD card slot, 4 bit SDIO interface (high speed data logging)

- 1 x USB : DFU mode (download) or USB storage (direct access to MicroSD card)
- 6 x Servo PWM outputs
- 1 x R/C receiver PPM frame input
- 1 x R/C receiver serial input with inverter (Futaba S.BUS, Spektrum, etc.)
- 3 x UART
- 2 x I2C bus
- 1 x SPI bus
- RTC with backup capacitor
- SWD(ARM download/debug interface)
- 4 x Auxiliary I/O (General Purpose and/or ADC and/or servo PWM)
- 5v / 1.5A switching power supply (input voltage range 5.5V min → 17.0v max)
- 3.3v / 1A linear regulator
- 1 x 5v / 500mA power switch
- 4 x status LEDs
- 10.4 grams (0.37 oz)
- 53 x 25mm (2.1" x 0.98"), shares the same external dimensions and mounting points as UmarimLite
- 4 layers PCB design

ArduPilot - Pixhawk

Figura 16 - Pixhawk autopilot

Specifications

Processor

- 32-bit ARM Cortex M4 core with FPU
- 168 Mhz/256 KB RAM/2 MB Flash
- 32-bit failsafe co-processor

Sensors

- MPU6000 as main accel and gyro
- ST Micro 16-bit gyroscope
- ST Micro 14-bit accelerometer/compass (magnetometer)
- MEAS barometer

Power

- Ideal diode controller with automatic failover
- Servo rail high-power (7 V) and high-current ready
- All peripheral outputs over-current protected, all inputs ESC protected

Interfaces

- 5x UART serial ports, 1 high-power capable, 2x with HW flow control
- Spektrum DSM/DSM2/DSM-X Satellite input
- Futaba S.BUS input (output not yet implemented)
- PPM sum signal
- RSSI (PWM or voltage) input
- I2C, SPI, 2x CAN, USB
- 3.3 and 6.6 ADC inputs

AeroQuad

Figura 17 - AeroQuad autopilot

Hardware components

- STM32F405VGT6 CPU, 168MHz Cortex M4 32 bit core including FPU, 1MB flash ROM, 192KB RAM (datasheet)
- MPU6000 gyro/accelerometer, SPI interface, selectable gyro range 250/500/1000/2000dps, selectable accel range +-2/4/8/16g (datasheet)
- MS5611-01BA03 barometer, 10cm resolution, 24 bit ADC, stainless steel cap (datasheet)
- HMC5983-TR magnetometer (datasheet)
- Key board features
- Connectors
- 8 PWM receiver inputs
- 8 PWM ESC/motor outputs
- 4 PWM servo outputs
- 3 USARTs
- 1 SPI
- 1 I2C
- 6 ADC (12bit)
- 1 mini USB device port for PC connection
- 1 µSDCard slot
- 1 I2C shared with onboard components
- 4 LED driver switching raw power input to connectors
- 2 5V outputs from on board 5V regulator
- 2 3.3V outputs from on board 3.3V regulator

- 1 DAC (12bit)
- SWD debug port
- Daughterboard connector holes
- All PWM lines are controlled by hardware timer for low jitter
- Options
- Servo 2+3 lines could be used as UART4
- ESC 3+4 lines could be used as USART6
- On board features
- CPU internal bootloader for flashing from USB and USART1
- Reset button
- 3.3V low drop voltage regulator
- 5V voltage regulator to power receiver
- 2 switchable LEDs to signal board status
- 2 status LEDs for 3.3V and USBVDD
- Dedicated SPI for MPU6000
- Dedicated SPI for SDCard
- Measuring of input voltage
- Data ready line for MPU6000
- Data ready line for HMC5983-TR

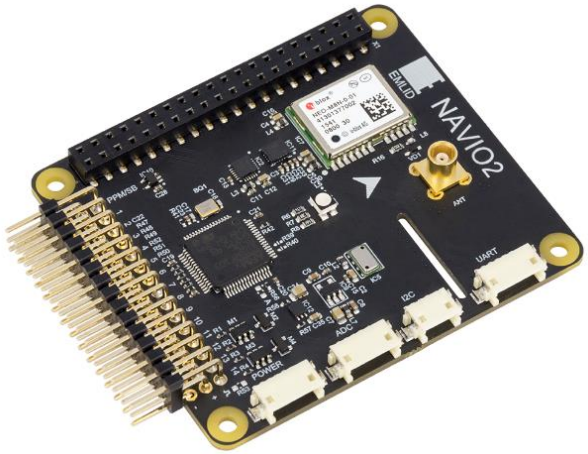
Power

- Typically driven by the motor's electronics speed controllers (ESC) +5V output, so no more extra connections from your power harness is required. The additional ESC power outputs power the optional camera stabilization servos and ultrasonic sensor.
- Board can be powered from USB, 5V output will have ~4.7V in this case, which should work with most receivers

Dimensions

- PCB size 50x50mm
- 4 mounting holes with 45mm distance, each 3mm in diameter

Navios 2



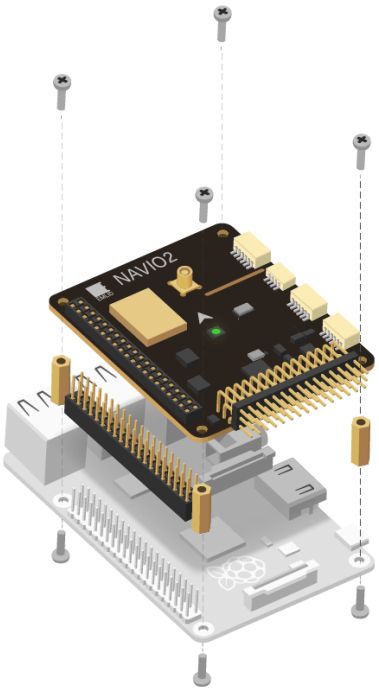


Figura 18 - Navios 2 Autopilot

FEATURES

- MPU9250 9DOF IMU
- LSM9DS1 9DOF IMU
- MS5611 Barometer
- U-blox M8N Glonass/GPS/Beidou
- RC I/O co-processor
- HAT EEPROM
- RGB LED

SPECIFICATIONS

- 14 PWM servo outputs
- PPM/S.Bus input
- Triple redundant power supply
- Power module connector
- UART, I2C, ADC for extensions
- Size: 55x65mm
- Weight: 23gr

En todos los casos anteriores, se denota que los autopilotos son de uso genérico, siendo necesario ser adaptados para el uso específico que requiere la aplicación del paracaida.

Los autopilotos comerciales presentan características deseables para el proyecto pero su problema radica en su excesivo costo, no siendo acorde para el presupuesto del proyecto actual.

En el caso de los autopilotos libres, donde su diseño se basa en el reducido tamaño y el bajo costo, su hardware a pesar de encontrarse disponible en la red, fue pensado para uso principalmente amateur. Por lo que para el uso en este proyecto, se requiere de un PCB robusto preparado para uso profesional.

Por estas razones se decide diseñar un Autopiloto específicamente orientado al control de paracaída, de manera de poder obtener un hardware robusto a medida, escalable y totalmente programable acorde a los requerimientos.

A su vez, se decide utilizar como placa madre al proyecto CIAA, ya que permite que se comience a desarrollar desde una base de hardware y firmware preexistente. Finalmente, para disminuir el tiempo de desarrollo se decide utilizar módulos comerciales.

8.4 INTRODUCCION A LA CIAA

La CIAA (Computadora Industrial Abierta Argentina) es una plataforma creada para impulsar el desarrollo tecnológico, generar conocimiento, y darle la posibilidad a PyMEs de poder basar sus trabajos en una plataforma testeada y

con la mayor cantidad de funcionalidades posible. Reduciendo así el tiempo y la inversión necesaria para poder llegar a un producto final.

La CIAA es la primera iniciativa a nivel mundial que cumple con las características de ser industrial, abierta y estar basada en procesadores de distintas marcas.

Actualmente está disponible la primera versión, que es la CIAA-NXP (Figura 19 - CIAA-NXP), que puede ser adquirida a las distintas empresas que la comercializan.

A su vez un equipo de trabajo conformado por alrededor de 100 personas está elaborando otras versiones de la plataforma, que tienen distinto grado de avance: CIAA-FSL (Figura 20 - CIAA FSL (con uC MK60FX512VLQ15)), CIAA-ACC,CIAA-Safety, CIAA-INTEL (Figura 21 - CIAA EDU-INTEL (Intel Edison)) y CIAA-PIC.

Figura 19 - CIAA-NXP

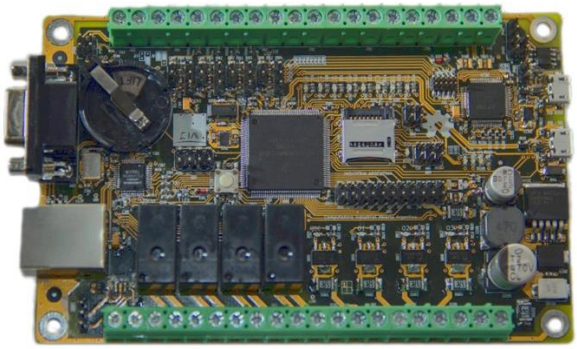


Figura 20 - CIAA FSL (con uC MK60FX512VLQ15)

Figura 21 - CIAA EDU-INTEL (Intel Edison)

Al mismo tiempo están abiertas las puertas para crear otras versiones de la plataforma a medida que aparezcan interesados.

La CIAA tiene el soporte de una comunidad de más de 4.000 desarrolladores de sistemas embebidos y por más de 50 universidades de todo el país, articuladas en la Red Universitaria de Sistemas Embebidos (RUSE) del CONFEDI.

Por todas estas razones, la CIAA es una plataforma adecuada para ser utilizada en aplicaciones profesionales, contando con amplio soporte y con diferentes alternativas de implementación que son compatibles entre sí. Todo lo cual, minimiza riesgos tecnológicos, logísticos y económicos, y brinda un contexto adecuado para el desarrollo tecnológico.

Hardware

El proyecto CIAA cuenta hoy en día con más de 6 plataformas de HW. En la sección de Hardware se explican las características de cada plataforma, el procesador, los periféricos, los esquemáticos y todo lo relevante para poder trabajar y adaptar la CIAA al proyecto en el cual sea utilizada.

Firmware

El firmware es el programa que se ejecuta en la CPU del microcontrolador. Éste comprende los módulos de código de programas para realizar una aplicación determinada, e interactúa directamente con los periféricos internos y otros componentes físicos de la computadora industrial, dispositivos electrónicos, e interfaces de comunicación. La CIAA tiene tanto la posibilidad de correr Linux como un bm. El Firmware también incluye un IDE para poder desarrollar desde un entorno confortable y amigable.

Software-PLC

El Software-PLC de la CIAA, es el entorno de desarrollo integrado (IDE) industrial que se ejecuta en una PC (Windows / Linux / MAC OS X compatible) y permite utilizar la CIAA como un PLC industrial. De esta forma es posible programar la placa en base al estándar IEC 61131-3.

EDU-CIAA-NXP

La EDU-CIAA-NXP (Figura 22 - EDU-CIAA-NXP) es una versión de bajo costo de la CIAA-NXP pensada para la enseñanza universitaria, terciaria y secundaria.

Figura 22 - EDU-CIAA-NXP

Esta versión es de reducido costo y es por esto que se deja disponible solo los componentes electrónicos necesarios para uso educativo. Como característica a destacar para el uso de desarrollos genéricos, se puede observar que se deja disponibles al desarrollador 80 pines de usos multiples.

8.5 HARDWARE DE EDU-CIAA-NXP

La EDU-CIAA-NXP cuenta con un Microcontrolador NXP LPC 4337 JDB 144 (Dual-core Cortex-M4 + Cortex-M0 @ 204MHz) , alimentación por puerto USB e implementa un circuito USB-to-JTAG FT2232H que soporta OpenOCD para el proceso de Debug (Figura 23 - Diagrama de bloque EDU-CIAA-NXP).

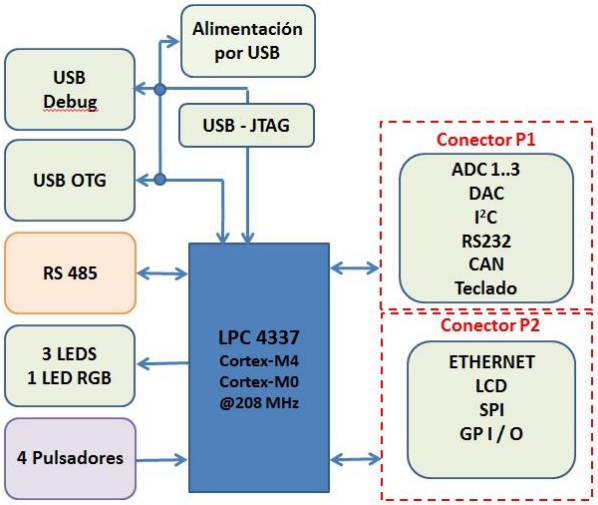


Figura 23 - Diagrama de bloque EDU-CIAA-NXP

La EDU-CIAA cuenta también con los siguientes módulos:

- 2 puertos micro-USB (uno para aplicaciones y debugging, otro para alimentación).
- 4 salidas digitales implementadas con leds RGB.
- 4 entradas digitales con pulsadores.
- 1 puerto de comunicaciones RS 485 con bornera.
- 2 conectores de expansión:
    - P1:
        - 3 entradas analógicas (ADC0_1,2y3),
        - 1 salida analógica (DAC0),
        - 1 puerto I2C,
        - 1 puerto asincrónico full duplex (para RS-232).
        - 1 puerto CAN,
        - 1 conexión para un teclado de 3×4,
    - P2:
        - 1 puerto Ethernet,
        - 1 puerto SPI,

- 1 puerto para Display LCD con 4 bits de datos, Enable y RS.
- 9 pines genéricos de I/O.

Pin Out de EDU-CIAA

En la Figura 24 - PinOut EDU-CIAA, se muestra un diagrama general de los pines pre asignados con sus funciones correspondientes.

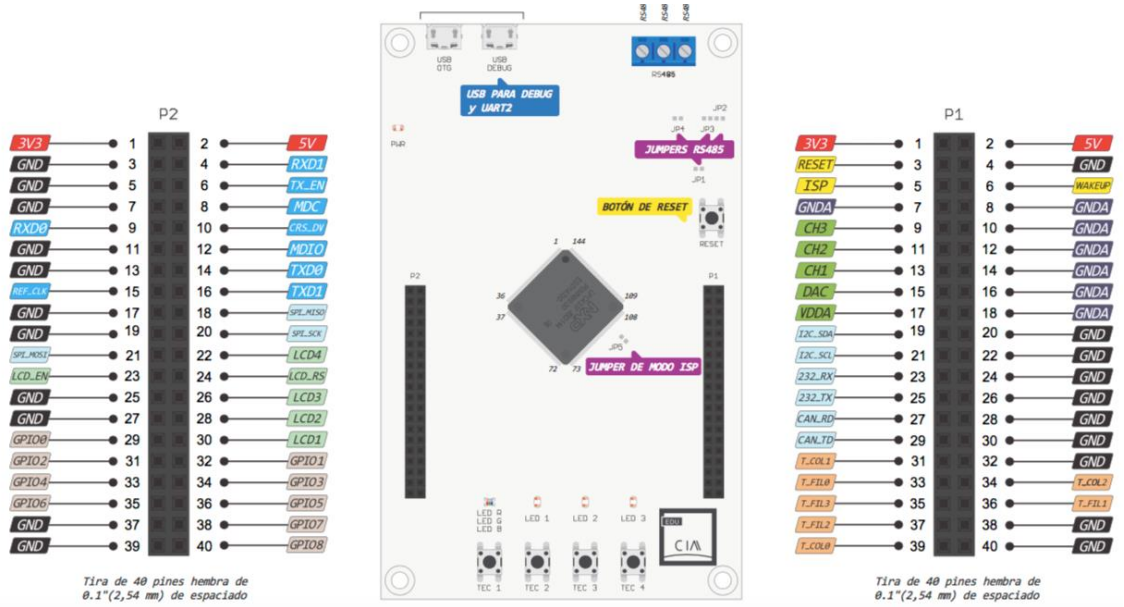


Figura 24 - PinOut EDU-CIAA

A pesar de que la EDU-CIAA presente funcionalidades pre asignadas en las salidas, el desarrollados puede reasignar según su necesidad lo requiera en función del hardware (Figura 25 - PinOut 2 EDU-CIAA).

Figura 25 - PinOut 2 EDU-CIAA

Información detalla sobre el pinout y diagramas de mejor calidad se encuentran en la página del proyecto CIAA[1].

8.5.1 uCONTROLADOR LPC4373

El LPC4337 es un microcontrolador basado en ARM Cortex-M4 para aplicaciones integradas que incluyen un coprocesador ARM Cortex-M0, de hasta 1 MB de Flash y 136 KB de SRAM en el chip, 16 KB de memoria EEPROM, periféricos configurables avanzados, como el temporizador configurable de Estado (SCT) y la E/S estándar seriales de propósito generales (SGPIO), dos controladores USB de alta velocidad, Ethernet, LCD, un controlador de memoria externa, y varios periféricos digitales y analógicos. El LPC4337 funcionan a frecuencias de CPU de hasta 204 MHz.

[1] http://www.proyecto-ciaa.com.ar/devwiki/lib/exe/fetch.php?media=desarrollo:edu-ciaa:edu-ciaa-nxp_pinout_a4_v4r2_es.pdf - Autor: Ing. Eric Pernia

El procesador ARM Cortex-M4 es un núcleo de 32 bits de nueva generación que ofrece mejoras en el sistema, tales como bajo consumo de energía, características mejoradas de depuración, y un alto nivel de integración de bloque de soporte. El procesador ARM Cortex-M4 incorpora una tubería ("pipeline") de 3 etapas, utiliza una arquitectura de Harvard con buses separados de instrucciones y datos locales, así como un tercer bus de periféricos, e incluye una unidad de captación previa interna que soporta ramificación especulativa. El procesador ARM Cortex-M4 es compatible con un solo ciclo de procesamiento digital de señales e instrucciones SIMD. Un procesador de punto flotante se integra en el núcleo.

El coprocesador ARM Cortex - M0 es un núcleo de 32 bits de fácil uso y eficiente energéticamente. Este fue diseñado como un reemplazo para microcontroladores de 8/16 bits existente, ofrece hasta 204 MHz con un set de instrucciones simples.

Aunque el sistema operativo de la plataforma CIAA en sus versiones NXP, ya cuente actualmente con soporte multiCore, al momento de comenzado el desarrollo no se encontraba disponible y es por esto que se decidió no utlizar el Core Cortex –M0. En la sección de trabajos futuros se deja como propuesta implementar en el firmware del "Piloto Automático" el soporte multiCore.

SIMD:

SIMD (del inglés Single Instruction, Multiple Data, en español: "una instrucción, múltiples datos") es una técnica empleada para conseguir paralelismo a nivel de datos (Figura 26 - SIMD Single Instruction, Multiple Data).

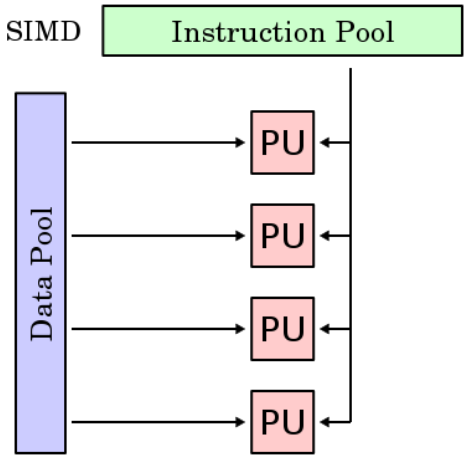


Figura 26 - SIMD Single Instruction, Multiple Data

Los repertorios SIMD consisten en instrucciones que aplican una misma operación sobre un conjunto más o menos grande de datos. Es una organización en donde una única unidad de control común despacha las instrucciones a diferentes unidades de procesamiento. Todas éstas reciben la misma instrucción, pero operan sobre diferentes conjuntos de datos. Es decir, la misma instrucción es ejecutada de manera sincronizada por todas las unidades de procesamiento.

8.5.1.1 ARQUITECTURA CORTEX M4

El procesador ARM ® Cortex® - M4 es un procesador embebido de alto rendimiento con instrucciones DSP desarrollados para hacer frente a los mercados de control de señales digitales que exigen una mezcla eficiente, de fácil uso de las capacidades de procesamiento de señal y control .

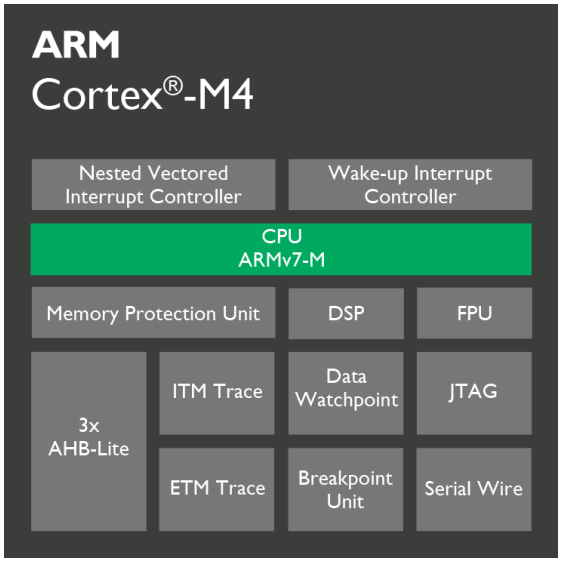


Figura 27 - Arquitectura Cortex M4

El procesador es altamente configurable, permite una amplia gama de implementaciones adaptandose a las aplicaciones que requieren operaciones de punto flotante, de protección de memoria, etc.

Beneficios

Obtener las ventajas de un microcontrolador con instrucciones integrados DSP, SIMD y MAC que simplifican el diseño general del sistema, desarrollo de software y de depuración.

Procesa operaciones aritméticas de precisión simple en punto flotante, hasta 10 veces más rápido que si se utilizara las biblioteca de software de entero equivalente, con la unidad de coma flotante opcional (FPU).

Logra excepcional rendimiento de 32 bits con baja potencia dinámica, la entrega que lleva la eficiencia energética del sistema debido a los modos integrados controlados por software de sleep, extensa y temporización de las compuertas de retención de estado opcional.

8.5.1.2 DIAGRAMA DE BLOQUE LPC4337

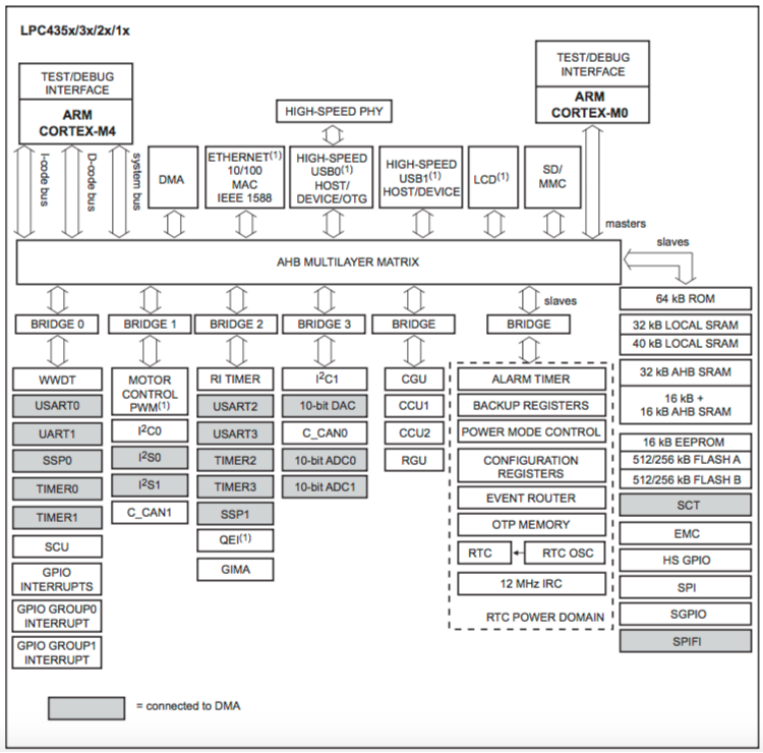


Figura 28 - Diagrama de bloque de LPC4337

A continuación se detallan las principales caractarísticas que presenta el microcontrolador LPC4337 de NXP:

Cortex-M4 Processor core

- ARM Cortex-M4 processor, running at frequencies of up to 204 MHz.

- ARM Cortex-M4 built-in Memory Protection Unit (MPU) supporting eight regions.
- ARM Cortex-M4 built-in Nested Vectored Interrupt Controller (NVIC).
- Hardware floating-point unit.
- Non-maskable Interrupt (NMI) input.
- JTAG and Serial Wire Debug (SWD), serial trace, eight breakpoints, and four watch points.
- Enhanced Trace Module (ETM) and Enhanced Trace Buffer (ETB) support.
- System tick timer.

Cortex-M0 Processor core

- ARM Cortex-M0 co-processor capable of off-loading the main ARM Cortex-M4 application processor.
- Running at frequencies of up to 204 MHz.
- JTAG, Serial Wire Debug, and built-in NVIC.

On-chip memory

- Up to 1 MB on-chip dual bank flash memory with flash accelerator.
- 16 kB on-chip EEPROM data memory.
- 136 kB SRAM for code and data use.
- Multiple SRAM blocks with separate bus access. Two SRAM blocks can be powered down individually.
- 64 kB ROM containing boot code and on-chip software drivers.
- 32 bit general-purpose One-Time Programmable (OTP) memory.

Configurable digital peripherals

- Serial GPIO (SGPIO) interface.
- State Configurable Timer (SCT) subsystem on AHB.
- Global Input Multiplexer Array (GIMA) allows to cross-connect multiple inputs and outputs to event driven peripherals like the timers, SCT, and ADC0/1.

Serial interfaces

- Quad SPI Flash Interface (SPIFI) with four lanes and up to 60 MB per second.
- 10/100T Ethernet MAC with RMII and MII interfaces and DMA support for high throughput at low CPU load. Support for IEEE 1588 time stamping/advanced time stamping (IEEE 1588-2008 v2).
- One High-speed USB 2.0 Host/Device/OTG interface with DMA support and on-chip high-speed PHY.
- One High-speed USB 2.0 Host/Device interface with DMA support, on-chip full-speed PHY and ULPI interface to external high-speed PHY.
- USB interface electrical test software included in ROM USB stack.
- One 550 UART with DMA support and full modem interface.
- Three 550 USARTs with DMA and synchronous mode support and a smart card interface conforming to ISO7816 specification. One USART with IrDA interface.
- Two C_CAN 2.0B controllers with one channel each.
- Two SSP controllers with FIFO and multi-protocol support. Both SSPs with DMA support.
- One SPI controller.
- One Fast-mode Plus I²C-bus interface with monitor mode and with open-drain I/O pins conforming to the full I²C-bus specification. Supports data rates of up to 1 Mbit/s.
- One standard I²C-bus interface with monitor mode and with standard I/O pins.
- Two I²S interfaces, each with DMA support and with one input and one output.

Digital peripherals

- External Memory Controller (EMC) supporting external SRAM, ROM, NOR flash, and SDRAM devices.
- LCD controller with DMA support and a programmable display resolution of up to 1024 H x 768 V. Supports monochrome and color STN panels and TFT color panels; supports

1/2/4/8 bpp Color Look-Up Table (CLUT) and 16/24-bit direct pixel mapping. Available on parts LPC4357/53 only.

- Secure Digital Input Output (SD/MMC) card interface.
- Eight-channel General-Purpose DMA (GPDMA) controller can access all memories on the AHB and all DMA-capable AHB slaves.
- Up to 164 General-Purpose Input/Output (GPIO) pins with configurable pull-up/pull-down resistors.
- GPIO registers are located on the AHB for fast access. GPIO ports have DMA support.
- Up to eight GPIO pins can be selected from all GPIO pins as edge and level sensitive interrupt sources.
- Two GPIO group interrupt modules enable an interrupt based on a programmable pattern of input states of a group of GPIO pins.
- Four general-purpose timer/counters with capture and match capabilities.
- One motor control Pulse Width Modulator (PWM) for three-phase motor control.
- One Quadrature Encoder Interface (QEI).
- Repetitive Interrupt timer (RI timer).
- Windowed watchdog timer (WWDT).
- Ultra-low power Real-Time Clock (RTC) on separate power domain with 256 bytes of battery powered backup registers.
- Alarm timer; can be battery powered.

Analog peripherals

- One 10-bit DAC with DMA support and a data conversion rate of 400 kSamples/s.
- Two 10-bit ADCs with DMA support and a data conversion rate of 400 kSamples/s. Up to eight input channels per ADC.

Unique ID for each device.

Clock generation unit

- Crystal oscillator with an operating range of 1 MHz to 25 MHz.
- 12 MHz Internal RC (IRC) oscillator trimmed to 1 % accuracy over temperature and voltage.
- Ultra-low power Real-Time Clock (RTC) crystal oscillator.
- Three PLLs allow CPU operation up to the maximum CPU rate without the need for a high-frequency crystal. The second PLL is dedicated to the High-speed USB, the third PLL can be used as audio PLL.
- Clock output.

Power

- Single 3.3 V (2.2 V to 3.6 V) power supply with on-chip DC-to-DC converter for the core supply and the RTC power domain.
- RTC power domain can be powered separately by a 3 V battery supply.
- Four reduced power modes: Sleep, Deep-sleep, Power-down, and Deep power-down.
- Processor wake-up from Sleep mode via wake-up interrupts from various peripherals.
- Wake-up from Deep-sleep, Power-down, and Deep power-down modes via external interrupts and interrupts generated by battery powered blocks in the RTC power domain.
- Brownout detect with four separate thresholds for interrupt and forced reset.
- Power-On Reset (POR).

8.5.1.3 CIRCUITO

En este apartado se muestra especificamente el circuito implementado en la EDU-CIAA-NXP para el microcontrolador LPC4337.

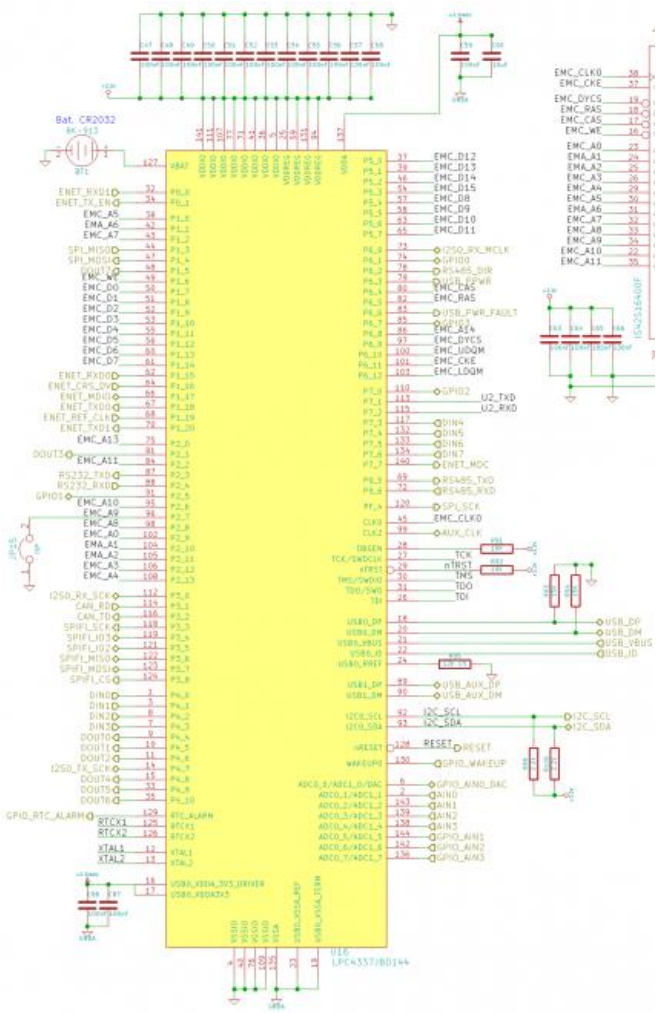


Figura 29 - Circuito implementación de LPC4337 en EDU-CIAA

8.5.2 JTAG

JTAG es un acrónimo para Joint Test Action Group. Es el nombre común utilizado para la norma IEEE 1149.1 titulada Standard Test Access Port and Boundary-Scan Architecture para test access ports utilizada para testear PCBs utilizando escaneo de límites.

JTAG se estandarizó en 1990 como la norma IEEE 1149.1-1990. En 1994 se agregó un suplemento que contiene una descripción del boundary scan description language (BSDL). Desde entonces, esta norma fue adoptada por las compañías electrónicas de todo el mundo. Actualmente, Boundary-scan y JTAG son sinónimos.

Diseñado originalmente para circuitos impresos, actualmente es utilizado para la prueba de submódulos de circuitos integrados. Es muy útil también como mecanismo para depuración de aplicaciones embebidas, puesto que provee una puerta trasera hacia dentro del sistema. Cuando se utiliza como herramienta de depuración, un emulador en circuito que usa JTAG como mecanismo de transporte permite al programador acceder al módulo de depuración que se encuentra integrado dentro de la CPU. El módulo de depuración permite al programador corregir sus errores de código y lógica de sus sistemas.

8.5.2.1 CIRCUITO DEBUGGER

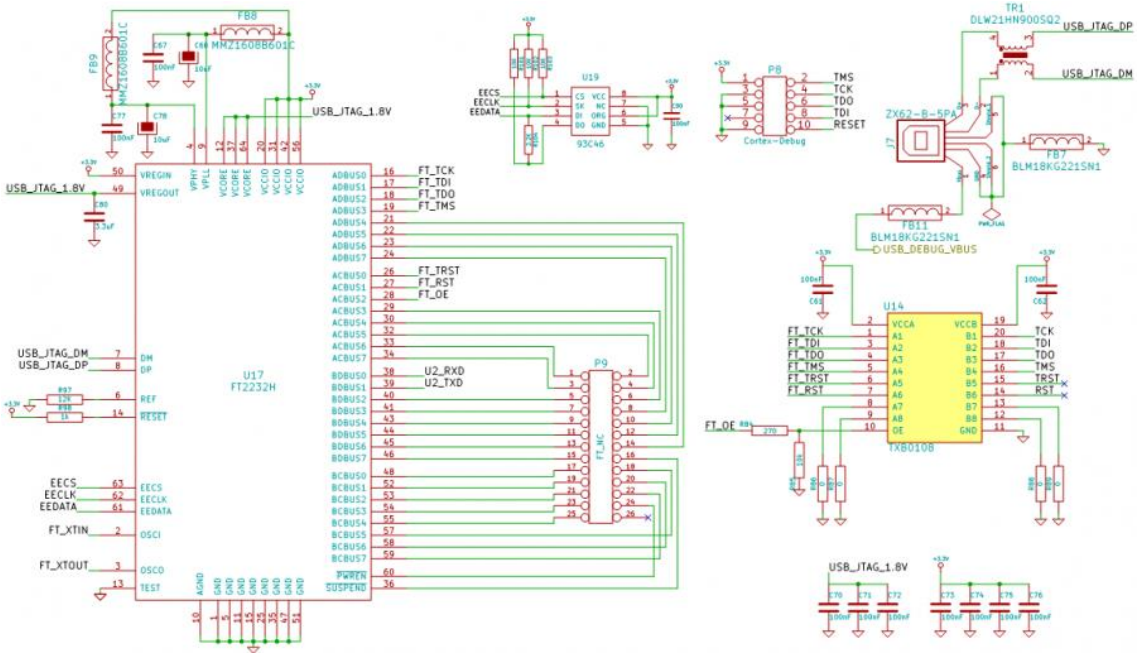
La plataforma EDU-CIAA-NXP incorpora un circuito de depuración completo compatible con Open On-Chip Debugger (OpenOCD[2]), el circuito integrado FT2232H también genera un puerto serie virtual en la PC disponible para ser usado por la EDU-CIAA.

Componentes principales:

- USB-to-JTAG FT2232H[3]. Soportado por OpenOCD[2].
- Memoria EEPROM (utilizada por el FT2232H) AT93C46DN.

[2] http://openocd.org

- Conector Cortex-Debug para Debugger externo.
- Buffer TXB0108 para desconectar el FT2232H del bus JTAG en caso que se use Debugger externo.



La CIAA cuenta con un conector Cortex-Debug que se encuentra disponible para uso externo de un debugger externo. Es por esta razón que se coloca el integrado TXB0108 (convertir de nivel lógico bidireccional), de manera de tener aislados los circuitos del debugger interno como el externo por medio de la activación del pin OE de dicho chip.

8.5.2.2 CIRCUITO INTEGRADO FT2232H

Figura 30 - Circuito integrado FT2232H

Referencia a hoja de dato del componente [3].

El FT2232H es un circuito integrado que adapta USB 2.0 Hi-Speed (480Mb/s) a UART/FIFO. Tiene la capacidad de ser configurado en una variedad de estandares seriales industriales o en interfaces paralelas.

Este circuito integrado tiene dos MPSSE (multi-protocol synchronous serial engines), que permiten la comunicación con JTAG , I2C y SPI en dos canales simultáneamente.

Algunas de las caraceristicas a destacar de este chip se detallan a continuacion:

- Un chip USB con 2 puertos seriales / paralelos con variedad de configuraciones.
- Manejados de protocolo USB completo implementado en chip. No requiere de firmware especifico.
- USB 2.0 High Speed (480Mbits/second) y Full Speed (12Mbits/second) compatible.
- Dual Multi-Protocol Synchronous Serial Engine (MPSSE) que simplificando el diseño con protocolos síncronos seriales (USB a JTAG, I2C, SPI, etc.).

[3] http://www.ftdichip.com/Support/Documents/DataSheets/ICs/DS_FT2232H.pdf

- Puerto dual e independiente UART o FIFO configurable usando MPSSEs.
- Generador de Baud rate independiente.
- Genera automaticamente señal de control Enable (EC) para aplicaciones que usen RS485, usando el TXDEN.
- Soporta alimentación por bus.
- Fuente de alimentación simple +3.3V.

8.5.3 RS485

RS-485 o también conocido como EIA-485, que lleva el nombre del comité que lo convirtió en estándar en 1983. Es un estándar de comunicaciones en bus de la capa física del Modelo OSI.

Está definido como un sistema de bus diferencial multipunto, es ideal para transmitir a altas velocidades sobre largas distancias (35 Mbit/s hasta 10 metros y 100 kbit/s en 1200 metros) y a través de canales ruidosos, ya que el par trenzado reduce los ruidos que se inducen en la línea de transmisión. El medio físico de transmisión es un par trenzado que admite 32, 128 o 254 estaciones en 1 solo par, con una longitud máxima de 1200 metros operando entre 300 y 19 200 bit/s y la comunicación half-duplex (semiduplex) dependiendo del consumo de cada driver. La transmisión diferencial permite alcanzar mayor distancia con una notable inmunidad al ruido, siempre que el bus de comunicación conserve las características de bus balanceado dando la posibilidad de una configuración multipunto.

Desde 2003 está siendo administrado por la Telecommunications Industry Association (TIA) y titulado como TIA-485-A.222.

8.5.3.1 CIRCUITO

En la Figura 31 - Circuito RS485 se muestra el circuito integrado utilizado, como así también las protecciones correspondiente a la salida de la misma. Cabe aclarar que la implementación utilizada en la EDU-CIAA-NXP es Half-dupplex.

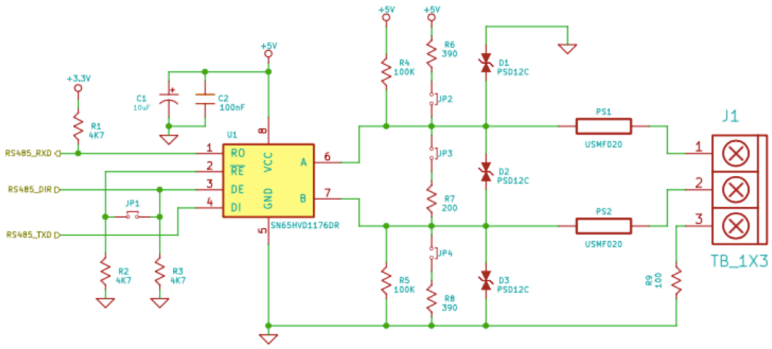


Figura 31 - Circuito RS485

8.5.4 ENTRADAS Y SALIDAS DIGITALES

8.5.4.1 PULSADORES (SWITCH)

Esta placa cuenta con cuatro pulsadores como se muestra en la Figura 32 - Circuito pulsadores, los mismos contienen un circuito antirrebote (debounce) RC.

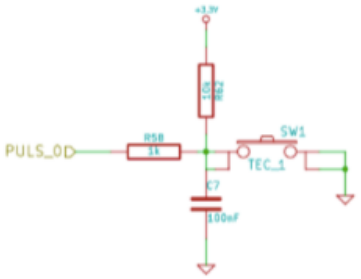


Figura 32 - Circuito pulsadores

Como se puede ver en la Figura 32 - Circuito pulsadores, cuando SW1 se encuentra abierto, o sea en estado normal, el uControlador registra a su entrada un nivel lógico alto. Por el contrario, cuando se presiona SW1, pasado un tiempo establecido por la constante de tiempo del R-C, el uControlador registrará un nivel lógico bajo.

8.5.4.2 DIODOS LED

La EDU-CIAA cuenta con un diodo LED RGB como se muestra en la Figura 33 - Circuito Diodo LED RGB, a su vez cuenta con tres diodos LED mas de colores rojo, verde y naranja como se muestra en la Figura 34 - Circuito Diodos LED.

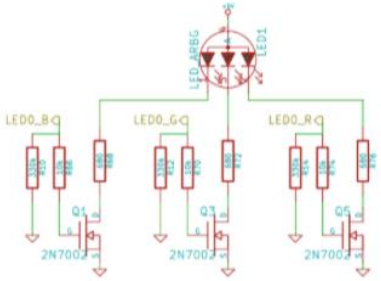


Figura 33 - Circuito Diodo LED RGB

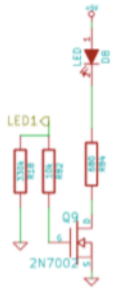


Figura 34 - Circuito Diodos LED

8.6 FIRMWARE

Sistemas operativos[4]

Los sistemas embebidos son plataformas con recursos muy limitados en comparación con una PC. Es por esto que generalmente no tienen un sistema operativo completo, sino sólo el subconjunto de estos que pueden manejar ventajosamente. Incluso en algunos casos el OS no es un programa en sí mismo sino que es un conjunto de funciones que se ejecutan solo en momentos determinados del programa.

Una tarea importante de los sistemas operativos es la de asignar tiempo de ejecución a todos los programas que tiene cargados en base a un juego de reglas conocido de antemano. A estos subprogramas se los llama tareas. Con esto se logra la ilusión de que múltiples programas se ejecutan simultáneamente, aunque

[4] https://2mp.conae.gov.ar/attachments/article/1313/SAE-MAN-0009-B%20-%20Manual%20de%20FreeOSEK.pdf

en realidad sólo pueden hacerlo de a uno a la vez (en sistemas con un sólo núcleo, como es el caso general de los sistemas embebidos).

Un RTOS es un sistema operativo de tiempo real. Esto significa que hace lo mismo que un OS común pero además ofrece herramientas para que los programas de aplicación puedan cumplir compromisos temporales definidos por el programador. El objetivo del mismo es diferente de un OS convencional. Un RTOS se emplea cuando hay que administrar varias tareas simultáneas con plazos de tiempo estrictos.

CIAA-Firmware + FreeOSEK[5]

CIAA-Firmware utiliza OSEK-OS, que es un estándar de RTOS (del inglés Sistema Operativo de Tiempo Real) internacional creado por la industria automotriz y de libre licencia, su ventaja respecto de otros es que es un estándar y por ende existen muchas implementaciones (open source y cerradas). Además de que hay muchas empresas que proveen soporte. Al ser un estándar abierto, quien quiera es libre de usar el estándar, implementarlo, venderlo, comprarlo, etc. Específicamente, el OSEK de CIAA-Firmware está basado en FreeOSEK.

FreeOSEK es un RTOS escalable para sistemas embebidos basado en las especificaciones OSEK-VDX y que a diferencia de otros sistemas operativos como Linux y Windows es un sistema operativo estático. Esto significa que las tareas, sus prioridades, cantidad de memoria que utilizan y etc. es definido antes de

[5] https://2mp.conae.gov.ar/attachments/article/1313/SAE-MAN-0009-B%20-%20Manual%20de%20FreeOSEK.pdf

compilar el código, en un proceso que se llama generación. Esta característica ofrece entre otras cosas:

- Determinístico
- Más performance
- Menos recursos
- Testeabilidad

Lo que lo hace un SO optimo para aplicaciones industriales. OSEK-VDX es un comité de estandarización creado en 1994 por las automotrices europeas. OSEKVDX incluye varios estándares que se utilizan en la industria automotriz, entre ellos los más relevantes. El estandard OSEKVDX es utilizado con exito desde hace mas de una decada en la industria automotriz.

8.6.1 ESTRUCTURA DEL FIRMWARE CIAA

Módulos

El CIAA-Firmware está dividido en módulos. Un módulo es un conjunto de archivos .h/.c y Makefiles, que implementa una funcionalidad específica. Cada módulo contiene una relativa gran cantidad de llamadas dentro del mismo, pero mantiene acotadas las dependencias a otros módulos, y de ser posible no tiene dependencias de otros módulos.

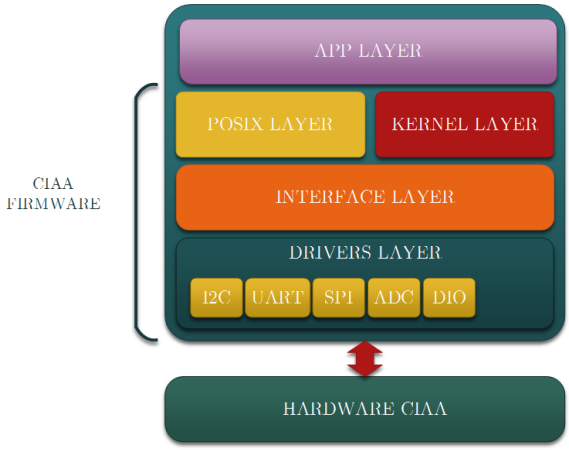


Figura 35 - Diagrama estructura de CIAA firmware

Capas

Los módulos del CIAA-Firmware están separados en 4 capas:

- Aplicación
- Servicios
- Interfaz
- Drivers

Aplicación

No es en sí parte del Firmware sino la aplicación en sí misma que utiliza el Firmware. Una aplicación podría ser la Aplicación Ladder, que de estar cargada transforma la CIAA en un PLC.

Servicios

Esta capa de software provee a la Aplicación interfaces estandarizadas a ella.

Interfaz

Abstrae la funcionalidad de los drivers para la capa de Servicios. A su vez, implementa toda la funcionalidad posible para dejar la implementación de los drivers a su mínima expresión, permitiendo la mayor re utilización posible entre dos hardwares incompatibles.

8.6.2 CAPA DE DRIVERS

Drivers

Esta capa implementa todos los drivers de la CIAA. Es la única capa que depende del hardware. Es desarrollada de forma de minimizar su funcionalidad. Toda funcionalidad adicional debe estar implementada en la capa de Interfaz.

- ciaaDriverUart
- ciaaSocket
- ciaaDriverFlash
- ciaaDriverGpio

8.6.3 CAPA DE KERNEL

8.6.3.1 ESTANDAR OSEK

OSEK-VDX es comité de estandarización creado en 1994 por las automotrices europeas. OSEK- VDX incluye varios estándares que se utilizan en la industria automotriz, entre ellos los más relevantes:

- OSEK OS
- OSEK COM
- OSEK NM

- OSEK Implementation Language
- OSEK RTI
- OSEK Time Trigger Operating System

En este documento se tratará únicamente OSEK-OS y OSEK Implementation Language (OIL). Algunas versiones especificas de estos documentos fueron estandarizados en la ISO17356. Además de los estándares en si, OSEK-VDX a travéz del proyecto MODISTARC realizó las especificaciones de los tests. Estos documentos también se encuentran disponibles en la página de OSEK-VDX. Los mismos especifican paso a paso como testear un sistema para certificar de que el mismo sea conforme al estándard.

OSEK-VDX es un estándard libre y abierto (Cerdeiro, 2015).

8.6.3.2 CONFIGURACIÓN

Al ser OSEK-OS un sistema estático es necesario configurarlo, indicar cuantas tareas hay definidas, que prioridad tienen estas tareas, etc. etc. Para ello OSEK-VDX definió otro estándard llamado OSEK Implementation Language comunmente llamado OIL.

Es un lenguaje textual muy fácil de interpretar similar al C donde uno indica las caracterîsitcas del OS, Tareas, etc. Por ejemplo:

```
TASK InitTask {

        PRIORITY = 1;

        SCHEDULE = NON;

        ACTIVATION = 1;

        STACK = 128;
        TYPE = BASIC;
```

```
        AUTOSTART = TRUE {
                APPMODE = ApplicationMode1;
        }
}
```

Este ejemplo muestra la definición de una tarea llamada InitTask que tiene una prioridad 1, 128 bytes de stack y se auto inicia al comienzo. El resto de la información se irá viendo en el transcurso del documento. Luego de configurar el OS, debe generarse el sistema operativo. Una vez generado el OS y antes de compilar, se debe crear el código C con la tarea a ejecutar, por ejemplo un main.c con el siguiente código:

```
int main (void) {
StartOs(ApplicationMode1)
}
TASK(InitTask) {
/* perform some actions */
TerminateTask();
}
```

Como se puede observar desde la función *main* se inicial el sistema operativo, el mismo correrá automáticamente la tarea InitTask y finalmente terminará con la llamada TerminateTask (Cerdeiro, 2015).

8.6.3.3 TAREAS

A diferencia de otros sistemas operativos donde las tareas corren por un tiempo indeterminado hasta ser terminadas, en OSEK-VDX y por lo general en sistemas de tiempo real, las tareas realizan su cometido y terminan. No se utilizan estructuras como while(1) para mantener la tarea corriendo, ni sleeps para dormir entre activaciones. Sino que se inicia la tarea, realiza su cometido y termina. Ya

sea leer la temperatura de un sensor, recibir un paquete de comunicación, procesar datos de audio, sin importar lo que sea: se comienza, se procesa y se termina.

En los casos en donde una tarea realmente debe esperar y no puede terminar OSEK-OS provee la utilización de eventos, estos se verán más adelante.

Para el control de tareas OSEK-OS ofrece las siguientes interfaces:

- ActivateTask: activa una tarea.
- ChainTask: realiza la combinación de ActivateTask seguido de TerminateTask.
- TerminateTask: termina una tarea

8.6.3.4 ESTADOS

Cada tarea en OSEK-VDX se encuentra siempre en uno de los siguientes cuatro estados:

• running: la tarea se encuentra corriendo, está utilizando los recursos del procesador en este mismo momento. En cada momento una única tarea puede encontrarse en este estado.

• ready: en este estado están todas las tareas que se encuentran esperando los recursos del procesador para poder correr. No se encuentran en el estado running por que una tarea de mayor prioridad se encuentra corriendo.

• waiting: la tarea se encontraba corriendo y decidio esperear la ocurriencia de un evento, hasta que el evento que espera ocurra la misma se encontrará en el estado waiting.

- suspended: es el estado por defecto de las tareas, la tarea esta momentaneamente desac- tivada.

La Figura 36 - Posibles estados de una tarea (imagen de OSEK-VDX OS Standard http://www.osek-vdx.org) esquematiza los estados y sus transiciones:

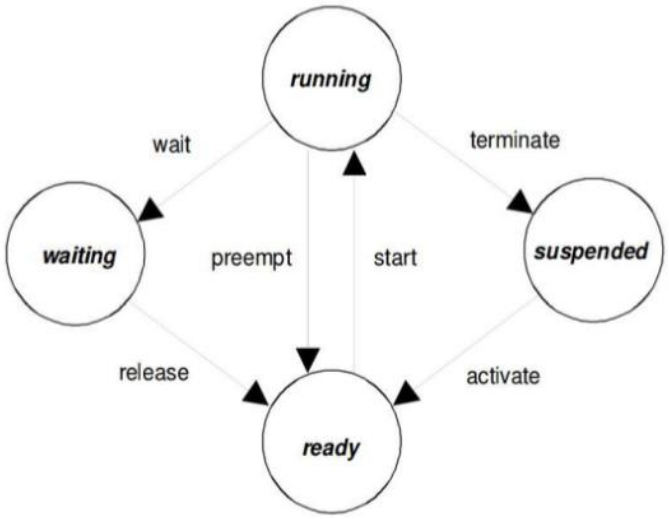


Figura 36 - Posibles estados de una tarea (imagen de OSEK-VDX OS Standard http://www.osek-vdx.org)

8.6.3.5 TIPOS DE TAREAS

El sistema OSEK-VDX permite 2 tipos de tareas, BASIC y EXTENDED. Las tareas básicas (BASIC) no tienen eventos y por ende no pueden permanecer en el estado waiting. A diferencia las extendidas (EXTENDED) pueden tener uno o más eventos y esperar . (Cerdeiro, 2015)

8.6.3.6 PRIORIDADES

En OSEK OS las prioridades de las tareas se definen de forma estática en el OIL. La prioridad es un número entero entre 1 y 255. Mayor sea el número mayor es la prioridad. Si dos tareas tienen la misma prioridad son ejecutadas según su

ordern de llegada FIFO. Una tarea que se encuentra corriendo nunca va a ser interrumpida por una de menor prioridad ni por una de la misma prioridad.

Es importante destacar, que a diferencia de otros sistemas operativos donde el tiempo de procesamiento es repartido de forma ponderada según la prioridad, en OSEK-OS no se reparte el tiempo de ejecución, para pasar a correr una tarea de menor prioridad la tarea de mayor prioridad debe o terminar o pasar al estado *waiting*. (Cerdeiro, 2015).

8.6.3.7 SCHEDULING

OSEK-OS provee 2 formas de scheduling que se pueden configurar a cada tarea:

- NON PREEMPTIVE
- PREEMPTIVE

Las tareas NON PREEMPTIVE no pueden ser interrumpidas nunca por otra tarea, sin importar que prioridad tengan, las tareas NON PREEMPTIVE se ejecutan sin interrupción hasta que terminan, pasan al estado waiting esperando un evento o llaman a la función Schedule.

Para configurar una tarea como NON PREEMPTIVE o PREEMPTIVE se utiliza en OIL el parametro SCHEDULE.

La función Schedule es una alternativa para forzar al OS de verificar si hay tareas de mayor prioridad por correr.

Las tareas NON PREEMPTIVE se utilizan generalmente en dos situaciones.

- Tareas de corta ejecución

- Sistemas deterministicos

8.6.3.8 ALARMAS

Las alarmas son utilizadas para realizar una acción luego de determinado tiempo. Las alarmas de OSEK-OS pueden realizar tres tipos de acciones:

- Activar una tarea
- Setear un evento de una tarea
- Llamar una callback en C

Para la implementación de las alarmas, ya sea 1 o n, el sistema operativo necesita un contador de HW. Con un contador es posible configurar la cantidad de alarmas que sea necesario. Cada alarma debe ser definida en el formato OIL así como la correspondiente acción.

8.6.3.9 SISTEMA OPERATIVO FreeOSEK

El RTOS implementado en el CIAA-Firmware está basado en FreeOSEK [6] y a su vez este cumple con el estándar OSEK-OS 2.2.3 .

8.6.4 CAPA POSIX

8.6.4.1 QUE ES POSIX

POSIX es una norma escrita por la IEEE. Dicha norma define una interfaz estándar del sistema operativo y el entorno, incluyendo un intérprete de comandos (o "shell"), y programas de utilidades comunes para apoyar la portabilidad de las aplicaciones a nivel de código fuente. El nombre POSIX surgió de la

[6] http://opensek.sourceforge.net/

recomendación de Richard Stallman, que por aquel entonces en la década de 1980 formaba parte del comité de IEEE.

Una serie de pruebas acompañan al estándar POSIX. Son llamadas "PCTS" en alusión al acrónimo "Posix Conformance Test Suite". Desde que la IEEE empezó a cobrar altos precios por la documentación de POSIX y se ha negado a publicar los estándares, ha aumentado el uso del modelo Single Unix Specification. Este modelo es abierto, acepta entradas de todo el mundo y está libremente disponible en Internet. Fue creado por The Open Group.

8.6.4.2 IMPLEMENTACIÓN POSIX-LIKE

El CIAA Firmware adopta algunos conceptos de POSIX para facilitar al programador de aplicaciones un acceso a los recursos de hardware fácil y principalmente portable. Por este motivo es posible compilar las aplicaciones tanto para arquitectura x86 y ejecutarlas en la PC como para Cortex M4 y ejecutarlas en la ECU-CIAA-NXP. Como no se adopto de forma completa el estándar POSIX se dice que CIAA- Firmware es "POSIX Like" [7].

La definición de los prototipos de las funciones se encuentra en el archivo "ciaaPOSIX_stdio.h" dentro de la carpeta "Firmware\modules\posix\inc\" [7].

El acceso a dispositivos está pensado de forma similar a POSIX, con las siguientes funciones:

- ciaaPOSIX_open

[7] https://2mp.conae.gov.ar/attachments/article/1313/SAE-MAN-0008-B%20-%20Manual%20POSIX.pdf

- ciaaPOSIX_close
- ciaaPOSIX_read
- ciaaPOSIX_write
- ciaaPOSIX_ioctl
- ciaaPOSIX_lseek

8.7 SOFTWARE

8.7.1 ENTORNO DE DESARROLLO EN LINUX

El proceso de instalación del entorno de desarrollo, ya sea para Windows o Linux, se encuentra documentado en los siguientes enlaces [8] [9] [10] y se adjunta en la documentación adicional del presente trabajo.

8.7.2 PROCESO DE COMPILACIÓN Y DEBUG

En este punto se describirá brevemente el proceso de compilación por consola, ya que inicialmente se trabajó así por comodidad y velocidad.

[8] https://2mp.conae.gov.ar/attachments/article/1313/SAE-MAN-0002-D%20-%20Manual%20instalacion%20IDE%20y%20compilacion.pdf - Instalación en Linux, Programa 2mp CONAE (Comisión Nacional de Actividades Espaciales)
[9] http://www.proyecto-ciaa.com.ar/devwiki/lib/exe/fetch.php?media=docu:fw:bm:ide:installciaa_ide_linux_v1.0.pdf - Instalación en Linux
[10] http://www.proyecto-ciaa.com.ar/devwiki/lib/exe/fetch.php?media=docu:fw:bm:ide:installciaa_ide_windows_v1.0.pdf - Instalación en MS Windows

Luego para el proceso de debug, se decidió configurar a Eclipse[11] y utilizar este IDE para todo el desarrollo.

Configuración de Makefile

```
cristian@cristian-N61Jq: ~/CIAA/Firmware
cristian@cristian-N61Jq:~$ cd CIAA/Firmware/
cristian@cristian-N61Jq:~/CIAA/Firmware$ ls
doc       externals  Makefile.config  Makefile.mine~  out       readme.md
examples  Makefile   Makefile.mine    modules         projects
cristian@cristian-N61Jq:~/CIAA/Firmware$
```

Con el Makefile.mine creado y configurado, se debe ejecutar la siguiente linea:

```
cristian@cristian-N61Jq: ~/CIAA/Firmware
cristian@cristian-N61Jq:~/CIAA/Firmware$ make clean
```

Como resultado se verá :

```
cristian@cristian-N61Jq: ~/CIAA/Firmware
cristian@cristian-N61Jq:~/CIAA/Firmware$ make clean
Removing libraries
Removing bin files
Removing RTOS generated files
Removing mocks
Removing Unity Runners files
Removing doxygen files
Removing ci outputs
Removing coverage
Removing object files
cristian@cristian-N61Jq:~/CIAA/Firmware$
```

Luego se debe ejecutar:

```
cristian@cristian-N61Jq: ~/CIAA/Firmware
cristian@cristian-N61Jq:~/CIAA/Firmware$ make generate
```

Esto generará los archivos .c y .h apartir del .oil de configuración

[11] https://eclipse.org/kepler/

```
cristian@cristian-N61Jq: ~/CIAA/Firmware
INFO: list of configuration files:
INFO: configuration file 1: projects/3dPitchRollYaw/etc/3dPitchRollYaw.oil
INFO: list of files to be generated:
INFO: generated file 1: ./modules/rtos/gen/inc/Os_Internal_Cfg.h.php
INFO: generated file 2: ./modules/rtos/gen/inc/Os_Cfg.h.php
INFO: generated file 3: ./modules/rtos/gen/src/Os_Cfg.c.php
INFO: generated file 4: ./modules/rtos/gen/src/Os_Internal_Cfg.c.php
INFO: generated file 5: ./modules/rtos/gen/src/cortexM4/Os_Internal_Arch_Cfg.c.p
hp
INFO: generated file 6: ./modules/rtos/gen/inc/cortexM4/Os_Internal_Arch_Cfg.h.p
hp
INFO: output directory: ./out/gen
INFO: reading projects/3dPitchRollYaw/etc/3dPitchRollYaw.oil
INFO: generating ./modules/rtos/gen/inc/Os_Internal_Cfg.h.php to ./out/gen/inc/O
s_Internal_Cfg.h
INFO: generating ./modules/rtos/gen/inc/Os_Cfg.h.php to ./out/gen/inc/Os_Cfg.h
INFO: generating ./modules/rtos/gen/src/Os_Cfg.c.php to ./out/gen/src/Os_Cfg.c
INFO: generating ./modules/rtos/gen/src/Os_Internal_Cfg.c.php to ./out/gen/src/O
s_Internal_Cfg.c
INFO: generating ./modules/rtos/gen/src/cortexM4/Os_Internal_Arch_Cfg.c.php to .
/out/gen/src/cortexM4/Os_Internal_Arch_Cfg.c
INFO: generating ./modules/rtos/gen/inc/cortexM4/Os_Internal_Arch_Cfg.h.php to .
/out/gen/inc/cortexM4/Os_Internal_Arch_Cfg.h
INFO: Generation Finished with WARNINGS: 0 and ERRORS: 0
cristian@cristian-N61Jq:~/CIAA/Firmware$
```

Si todo se hizo correctamente, se verá al final el mensaje que dice "Finished with WARNINGS: 0 and ERRORS: 0"

Esto significa que se puede continuar con la compilación y se debe ejecutar:

```
cristian@cristian-N61Jq: ~/CIAA/Firmware
cristian@cristian-N61Jq:~/CIAA/Firmware$ make
```

Como resultado exitoso, se obtiene lo siguiente:

```
cristian@cristian-N61Jq: ~/CIAA/Firmware
================================================================================
Post Building 3dPitchRollYaw

arm-none-eabi-objcopy -v -O binary ./out/bin/3dPitchRollYaw.axf ./out/bin/3dPitc
hRollYaw.bin
copy from `./out/bin/3dPitchRollYaw.axf' [elf32-littlearm] to `./out/bin/3dPitch
RollYaw.bin' [binary]
cristian@cristian-N61Jq:~/CIAA/Firmware$
```

Finalmente se debe cargar el binario generado y para esto se debe ejecutar:

```
cristian@cristian-N61Jq: ~/CIAA/Firmware
cristian@cristian-N61Jq:~/CIAA/Firmware$ make download
```

8.7.3 ESTACIÓN DE CONTROL DE TIERRA

En el nivel más básico, una estación de control en tierra (GCS) es un programa diseñado para recibir flujos de datos desde la aeronave y representar gráficamente en una computadora que se encuentra en tierra. Para utilizar este GCS no necesariamente se necesita un piloto automático, como mínimo se usaría una unidad GPS conectada a un transmisor, como por ejemplo X-Bee, en su avión y otra unidad X-Bee conectado a su computadora portátil en el suelo. Esto proporcionaría latitud, longitud, velocidad, rumbo y datos de altitud. Para cabeceo y balanceo (y posiblemente de guiñada) se necesita de una plataforma inercial IMU (por sus siglas en ingles: Innertial measure unit) o un piloto automático con un IMU.

El software de estación de control de tierra seleccionado para el presente trabajo es HappyKillmore's Ground Control Station, ya que es el programa con el que se trabaja actualmente en el departamento de mecánica aeronáutica.

Figura 37 - HappyKillmore's Ground Control Station

Características:

- Soporta protocolos: NMEA, uBlox, SiRF, MediaTek (custom binary for DIYDrones.com), ArduPilot ascii, ArduPilot Mega binary, ArduIMU binary, MAVlink, FY21AP II, etc.
- Video en tiempo realLive real time video
- Instrumental de vuelo gráfico.

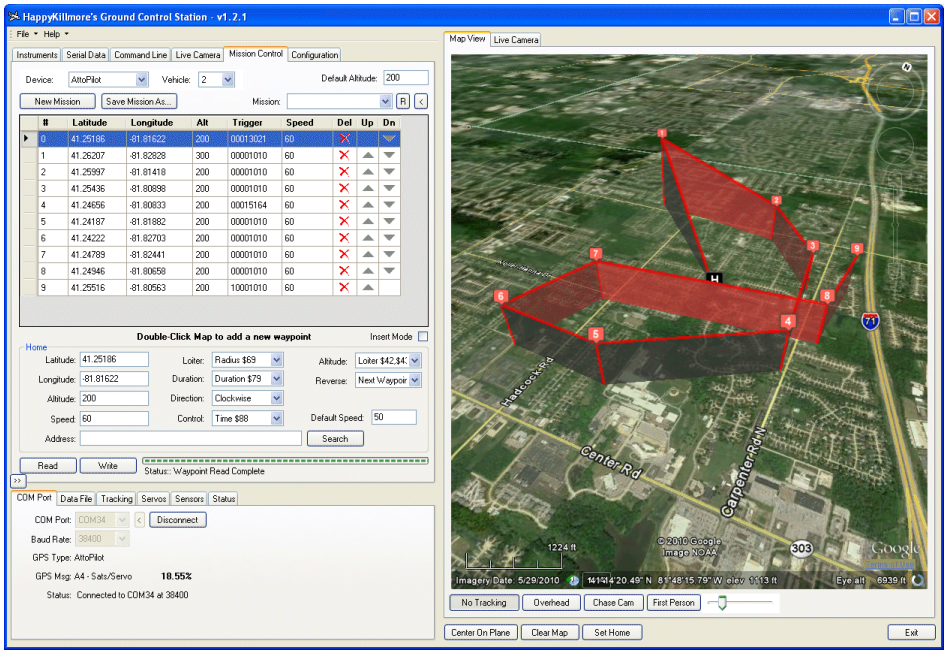


Figura 38 - Control de misión con waypoints

Como se puede observar en la Figura 38 - Control de misión con waypoints, este software cuenta la posibilidad de trazar y visualizar los puntos que se desea seguir en una misión (waypoints).

Por otro lado, cabe destacar que haciendo clic en el botón de conexión y poniendo en marcha la comunicación de datos, éste selecciona el protocolo automáticamente por lo que solo es necesario fijar el Baud Rate y el puerto de comunicación.

8.8 COMUNICACIÓN

8.8.1 INTER-INTEGRATED CIRCUIT (I2C)

I²C, en inglés Inter-Integrated Circuit, en español pronunciado como I-cuadrado-C o en inglés I-Squared-C, es un bus de datos serial desarrollado en 1982 por Philips Semiconductors (hoy NXP Semiconductors).

El I²C está diseñado como un bus maestro-esclavo. La transferencia de datos es siempre inicializada por un maestro; el esclavo reacciona. Es posible tener varios maestros (Multimaster-Mode). En el modo multimaestro pueden comunicar dos maestros entre ellos mismos, de modo que uno de ellos trabaja como esclavo. El arbitraje (control de acceso en el bus) se rige por las especificaciones, de este modo los maestros pueden ir turnándose.

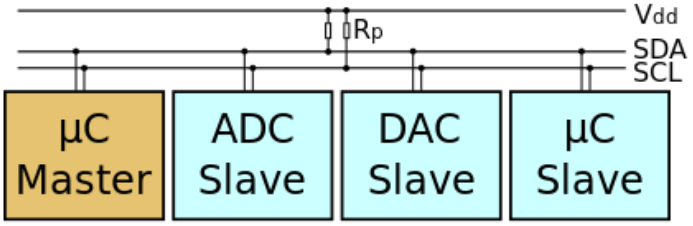


Figura 39 - Diagrama de conección Maestro – Esclavo

El I²C precisa de dos líneas de señal: reloj (CLK, Serial Clock) y la línea de datos (SDA, Serial Data). Ambas líneas precisan resistencias de pull-up hacia VDD.

Modos y velocidades

En la Tabla 3 - Tabla de velocidades y modos - I2C, se muestran los porcentajes máximos permisibles de reloj.

Tabla 3 - Tabla de velocidades y modos - I2C

Modo	**Velocidad de transmisión máxima**	**Dirección**
Standard Mode (Sm)	0,1 Mbit/s	Bidireccional
Fast Mode (Fm)	0,4 Mbit/s	Bidireccional

Fast Mode Plus (Fm+)	1,0 Mbit/s	Bidireccional
High Speed Mode (Hs-mode)	3,4 Mbit/s	Bidireccional
Ultra Fast-mode (UFm)	5,0 Mbit/s	Unidireccional

8.8.2 SERIAL PERIPHERAL INTERFACE (SPI)

El Bus SPI (del inglés Serial Peripheral Interface) es un estándar de comunicaciones, usado principalmente para la transferencia de información entre circuitos integrados en equipos electrónicos. El bus de interfaz de periféricos serie o bus SPI es un estándar para controlar casi cualquier dispositivo electrónico digital que acepte un flujo de bits serie regulado por un reloj (comunicación sincrónica).

Incluye una línea de reloj, dato entrante, dato saliente y un pin de chip select, que conecta o desconecta la operación del dispositivo con el que uno desea comunicarse. De esta forma, este estándar permite multiplexar las líneas de reloj.

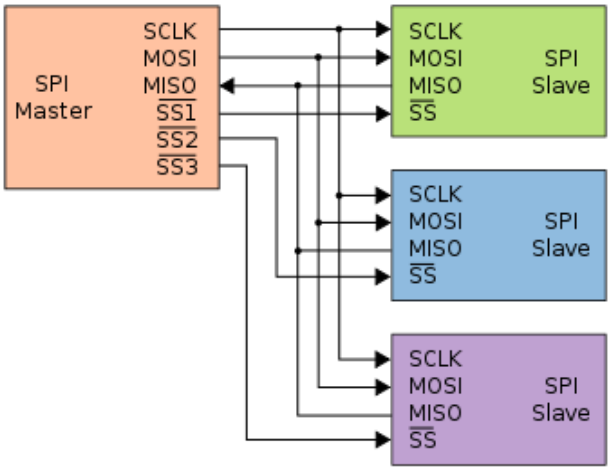


Figura 40 - Bus SPI

El SPI es un protocolo síncrono. La sincronización y la transmisión de datos se realiza por medio de 4 señales:

- SCLK (Clock): Es el pulso que marca la sincronización. Con cada pulso de este reloj, se lee o se envía un bit. También llamado TAKT (en Alemán).
- MOSI (Master Output Slave Input): Salida de datos del Master y entrada de datos al Slave. También llamada SIMO.
- MISO (Master Input Slave Output): Salida de datos del Slave y entrada al Master. También conocida por SOMI.
- SS/Select: Para seleccionar un Slave, o para que el Master le diga al Slave que se active. También llamada SSTE.

La Cadena de bits es enviada de manera síncrona con los pulsos del reloj, es decir con cada pulso, el Master envía un bit. Para que empiece la transmisión el Master baja la señal SSTE ó SS/Select a cero, con esto el Slave se activa y empieza la transmisión, con un pulso de reloj al mismo tiempo que el primer bit es leído. Nótese que los pulsos de reloj pueden estar programados de manera que la transmisión del bit se realice en 4 modos diferentes, a esto se llama polaridad y fase de la transmisión:

- Con el flanco de subida sin retraso.
- Con el flanco de subida con retraso.
- Con el flanco de bajada sin retraso.
- Con el flanco de bajada con retraso.

8.8.3 PROTOCOLO NMEA 0183

NMEA 0183 es una especificación combinada eléctrica y de datos entre aparatos electrónicos marinos y, también, más generalmente, receptores GPS.

El protocolo NMEA 0183 es un medio a través del cual los instrumentos marítimos y también la mayoría de los receptores GPS pueden comunicarse los unos con los otros. Ha sido definido, y está controlado, por la organización estadounidense National Marine Electronics Association [12].

A continuación se muestra un ejemplo de la trama GGA, donde se puede observar que para todas las tramas se utiliza la cabecera "$GP" seguido de letras (en este caso GGA) y finaliza con una coma ",".

```
$GPGGA,123519,4807.038,N,01131.000,E,1,08,0.9,545.4,M,46.9,M,,*47

Where:
     GGA          Global Positioning System Fix Data
     123519       Fix taken at 12:35:19 UTC
     4807.038,N   Latitude 48 deg 07.038' N
     01131.000,E  Longitude 11 deg 31.000' E
     1            Fix quality: 0 = invalid
                               1 = GPS fix (SPS)
                               2 = DGPS fix
                               3 = PPS fix
                          4 = Real Time Kinematic
                          5 = Float RTK
                               6 = estimated (dead reckoning) (2.3 feature)
                          7 = Manual input mode
                          8 = Simulation mode
     08           Number of satellites being tracked
     0.9          Horizontal dilution of position
     545.4,M      Altitude, Meters, above mean sea level
     46.9,M       Height of geoid (mean sea level) above WGS84
                      ellipsoid
     (empty field) time in seconds since last DGPS update
     (empty field) DGPS station ID number
     *47          the checksum data, always begins with *
```

[12] https://www.nmea.org/content/nmea_standards/nmea_0183_v_410.asp

8.8.4 PROTOCOLO ARDUPILOT ASCII (legacy)

Este protocolo utilizado inicialmente por el proyecto Ardupilot y posteriormente reemplazado por el protocolo MavLink que ofrece mejores prestaciones.

Está caracterizado por estar implementado a nivel de caracteres ASCII por lo que sus tramas de datos son armadas como se muestran en el ejemplo a continuación.

Ejemplo:

```
!!!LAT:35300071,LON:-120663928,SPD:14.23,CRT:3.31,ALT:335,ALH:327,CRS:82.58,BER:14,WPN:0,DST:877,BTV:16.64***
```

En la Figura 41 - Trama protocolo de telemetría, se detalla como se forma el vector de datos de manera genérica para ser transmitida y también se incluye la cantidad de bytes que se destina a cada identificador y dato.

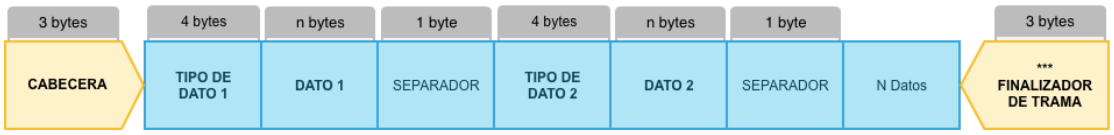


Figura 41 - Trama protocolo de telemetría

La comunicación comienza con una cabecera de 3 bytes la cual identifica a los tipos de datos que se contendrán, ver Tabla 4 - Cabeceras de mensaje. En el cuerpo del paquete se envia el identificador del tipo seguido del dato propiamente dicho en formato ASCII, es decir que por ejemplo el integer 2468 debe convertirse en ASCII para luego ser enviado.

El "SEPARADOR" entre cada tipo de dato es el carácter "," coma o en hexadecimal 0x2C.

Para indicar el "FINALIZADOR DE TRAMA" del paquete se envia en hexadecimal 0x2A2A2A o en codificación ASCII los caracteres correspondientes "***" a tres asteriscos.

Tipos de cabeceras de mensaje

Tabla 4 - Cabeceras de mensaje

!!!	**Position**	**Low rate telemetry**
+++	Attitude	High rate telemetry
###	Mode	Change in control mode
%%%	Waypoint	Current and previous waypoints
XXX	Alert	Text alert - NOTE: Alert message generation is not localized to a function
PPP	IMU Performance	Sent every 20 seconds for performance monitoring

8.8.5 XBEE

XBee es el nombre comercial de una familia de módulos de radio compatibles con factor de forma de Digi International. Las primeras radios XBee se introdujeron bajo la marca MaxStream en 2005 y se basa en el estándar IEEE 802.15.4-2003[13] diseñada para conexiones punto a punto o comunicaciones estrellas inalambricas a velocidades de transmisión 250 kbit/s.

[13] http://standards.ieee.org/getieee802/download/802.15.4-2003.pdf

Los modelos introducidos inicialemente se diferencian en la potencia de transmisión. Uno de costo más bajo, 1 mW "Xbee" y otro de mayor potencia de 100 mW "XBee-PRO".

Los radios XBee se pueden usar con el número mínimo de conexiones: Alimentación de 3.3 V, masa, datos de entrada y salida de datos (UART).

Además, la mayoría de las familias XBee contienen puertos extras de control de flujo, de entrada/salida (I/O), conversión analógica-digital (A/D) y líneas indicadoras incorporada.

9. DESARROLLO

9.1 RESUMEN TÉCNICO

Como se vio en secciones anteriores introductorias, el presente trabajo tiene por objetivo el diseño y desarrollo de un piloto automático para paracaida.

A continuación se detallan las características 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 bateria.

- 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 mediantes 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.

En la Figura 42 - Diagrama general de sistema Autopiloto simplificado se puede observar la interacción de los subsistemas que componen al Autopiloto.

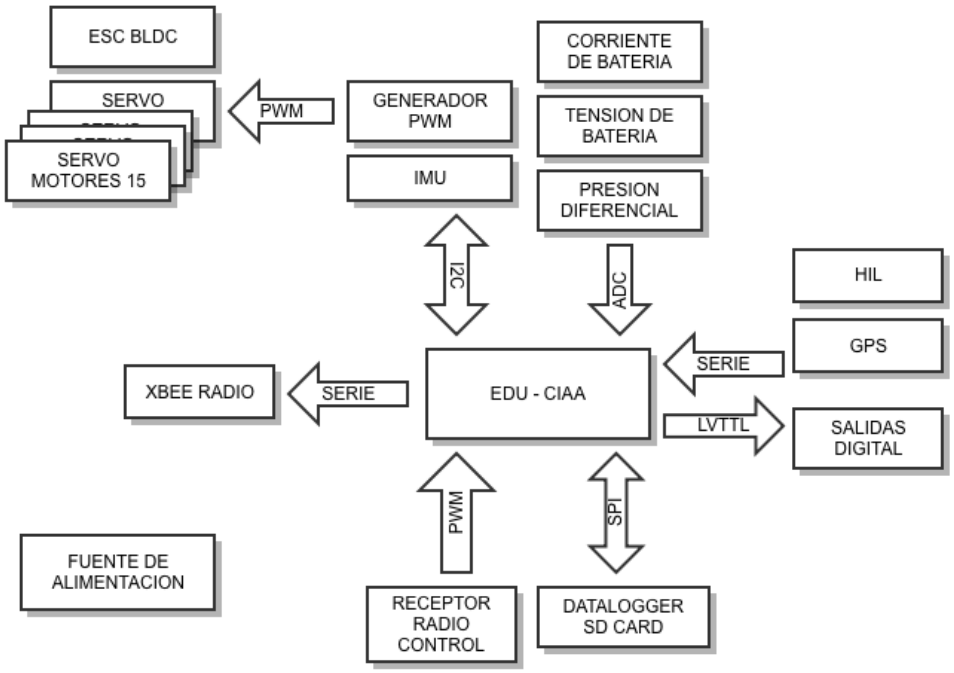


Figura 42 - Diagrama general de sistema Autopiloto simplificado

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

En la sección 9.1.1 se entra en detalle sobre el hardware de cada módulo componente y en la sección 9.1.2 se detalla el firmware desarrollado a nivel de aplicación e implementaciones de cada módulo sobre el sistema operativo.

Sistema de control

Debido a que esta plataforma es de carácter experimental, su finalidad es la de ensayar diferentes modelos de sistemas de control y concluido esta etapa, ejecutar el sistema óptimo seleccionado en el modelo de vuelo final. Es por esto que el

algoritmo del sistema de control no será implementado en este trabajo, diseñando toda la aplicacion orientada a contener este algoritmo.

9.1.1 HARDWARE

En esta sección se desarrolla el proceso de diseño del hardware, dando como resultado un módulo adicional encastrable (Poncho[14]) sobre la parte superior de la EDU-CIAA, similar a lo que es un "Shield" para el proyecto Arduino.

Figura 43 - Logo Poncho para EDU-CIAA

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 44 - Diagrama general de Autopiloto, para el desarrollo del hardware se opta por utilizar un modelo de diseño modular y utilizar

[14] Llamamos "Ponchos" a los módulos conectables para montarse sobre el módulo procesador EDU-CIAA. El nombre "Poncho" se utiliza como "Shield" en Arduino entre la comunidad del proyecto CIAA.

la mayor cantidad de módulos comerciales que sean adecuados para esta aplicación de forma de acelerar el proceso en esta fase.

A continuación, en la Figura 44 - Diagrama general de Autopiloto, se muestra un diagrama detallando los módulos y la forma en que estos son conectados a la EDU-CIAA.

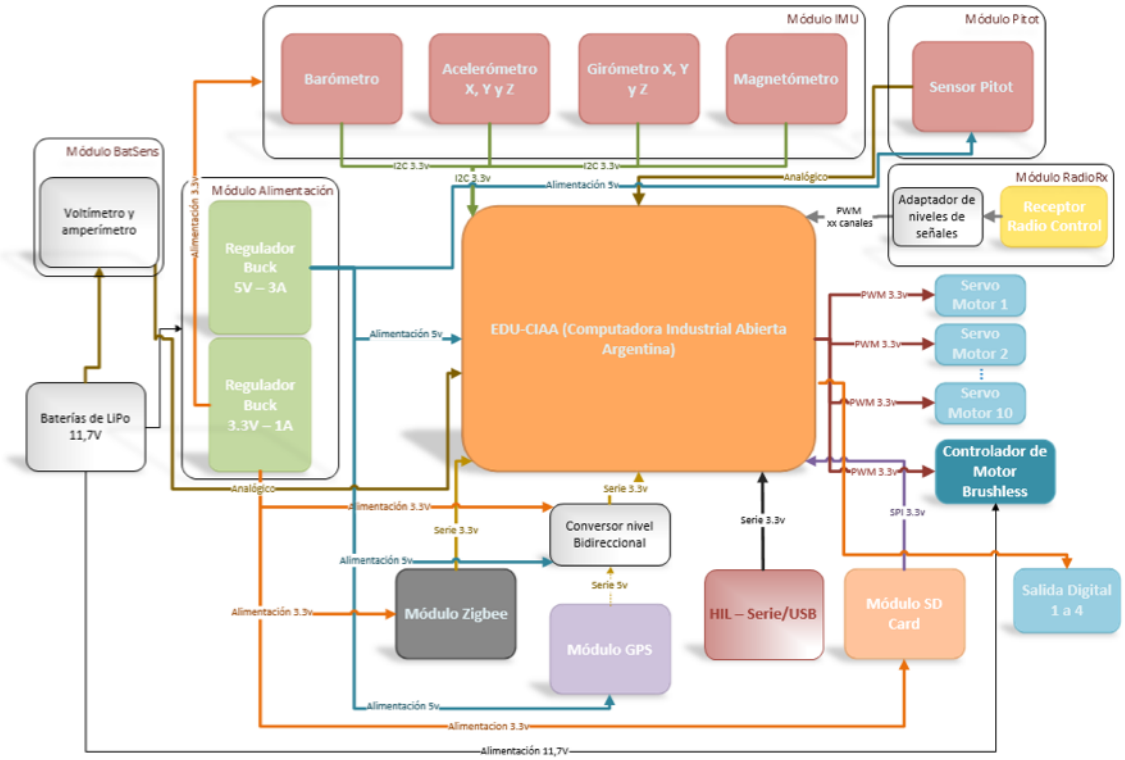


Figura 44 - Diagrama general de Autopiloto

9.1.1.1 MÓDULO ALIMENTACIÓN

Se encarga de proveer las tensiones estabilizadas y constantes a cada sub módulo asegurando que no sucedan picos ni bajas que pudiesen alterar el correcto funcionamiento de todo el sistema.

El módulo debe alimentar a la EDU-CIAA y al Poncho del piloto automático y cumplir con los siguientes requerimientos:

Requerimientos de diseño

- Tensión de alimentación de entrada: de 11,1 a 14,8V (baterías LiPo de 3S a 4S)
- Tensión de alimentación de salida: 5v
- Consumo de corriente máximo: 1A
- Baja potencia dispada
- Rendimiento alto, por ser un sistema alimentado a baterias.

A partir de los requisitos anteriores se decide utilizar una fuente switching del tipo Buck, encontrando en el mercado el siguiente circuito basado en el integrado XL4015 tal como se muestra en la Figura 45 - Fuente switching Buck. Este circuito es un conversor DC/DC del tipo buck con frecuencia de trabajo fija de 180KHz y capaz de entregar hasta 5A con tensión de entrada de entre 8V a 36V y tensión de salida de 1,25V a 32V con eficiencia superior al 85%.

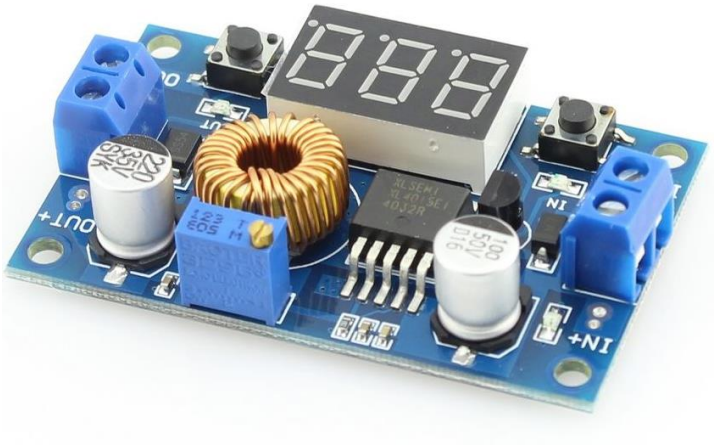


Figura 45 - Fuente switching Buck

9.1.1.2 MÓDULO IMU

El módulo de Unidad Inercial (IMU por sus siglas en Inglés) se encarga de contener la electrónica necesaria para el correcto funcionamiento de los sensores de presión barométrica, aceleraciones, velocidad angular, campo magnético terrestre y temperatura de aire.

El módulo IMU debe cumplir con los siguientes requerimientos:

Acelerometro

- Rango de medición: ±3.6g.
- Sensibilidad de señal de salida: 300mV/g.

Girómetro

- Rango de medición: ±500°/s.
- Sensibilidad de señal de salida: 2mV/°/s.

Magnetómetro

- Rango de medición: ±8 gauss.
- Resolución: 2mGauss.

Barómetro

- Rango de medición: 300 a 1100hPa.
- Resolución: 0.01hPa.
- Precisión: ±1.0hPa.

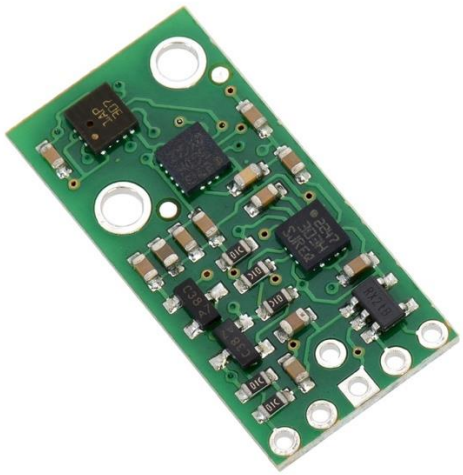


Figura 46 - AltIMU-10

Especificaciones:

- Tension de alimentación: 2.5 V to 5.5 V
- Corriente consumida: 6 mA
- Comunicación: I2C
- Girómetro: one 16-bit reading per axis
- Acelerómetro: one 16-bit reading per axis
- Magnetómetro: one 16-bit reading per axis

- Barómetro: 24-bit pressure reading (4096 LSb/mbar)

Rangos de sensivilidad:

- Girómetro: ±245, ±500, or ±2000°/s
- Acelerómetro: ±2, ±4, ±6, ±8, or ±16 g
- Magnetómetro: ±2, ±4, ±8, or ±12 gauss
- Barómetro: 260 mbar to 1260 mbar (26 kPa to 126 kPa)

En la Figura 47 - Circuito esquemático AltIMU se muestra el circuito implementado. El módulo se alimenta con una tensión de 3,3V dejando sin utilidad a los transistores Q1 y Q2 y al regulador lineal de 3.3v LDO.

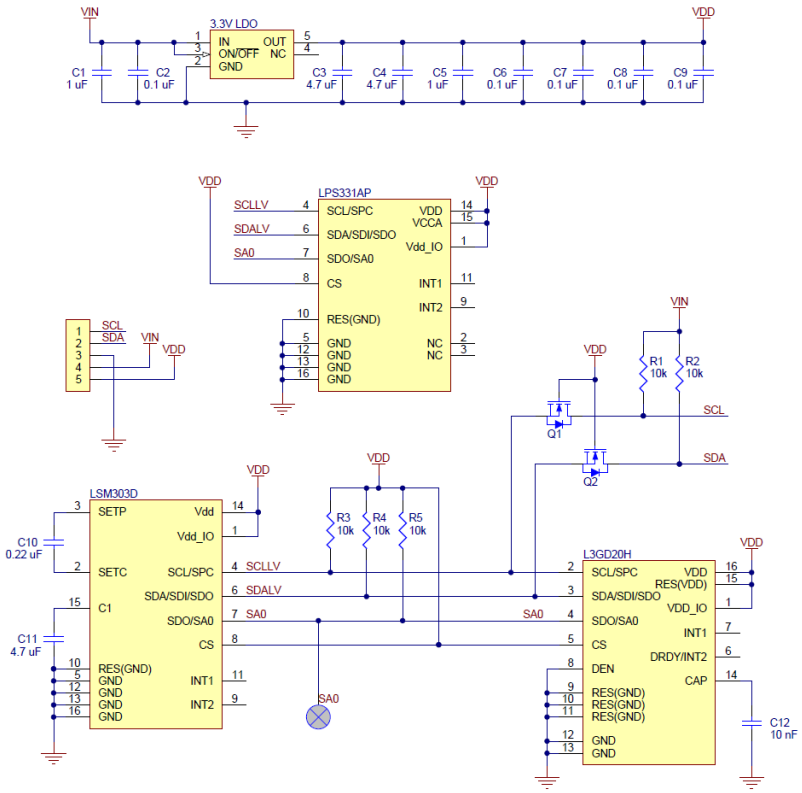


Figura 47 - Circuito esquemático AltIMU

Como se puede apreciar, el módulo IMU cumple con los requerimientos de diseño establecidos, dejando un margen en el rango de los sensores para otras posibles aplicaciones.

9.1.1.3 MÓDULO PITOT

El módulo Pitot, se encarga de contener la electrónica necesaria para la adquisición de la presión dinámica, utilizada para el cálculo de la velocidad del móvil.

El módulo Pitot debe cumplir con los siguientes requerimientos:

Velocidad mínima: 5 [m/s]

Velocidad máxima: 30 [m/s]

El cálculo de la presión dinámica se realiza a través de la ecuación de Bernoulli, y para flujo de fluido incompresible se tiene que:

$$P_{dim} = \frac{1}{2}.\rho.v^2$$

Ecuación 1

Donde:

- P_{dim} es presión dinámica,
- ρ es densidad del aire,
- v es velocidad del aire

Rango de medición de presión dinámica:

$$P_{dim_{MAX}} = 540[Pa]$$

$$P_{dim_{MIN}} = 15[Pa]$$

Se decide utilizar el mismo sensor con el que cuenta el Departamento de Mecánica Aeronáutica ya que se ha comprobado su correcto funcionamiento y cumple ampliamente con los requerimientos.

El sensor seleccionado se muestra en la Figura 48 - Sensor de presión diferencial MPVX7002DP y a continuación se describen sus caracteristicas.

Figura 48 - Sensor de presión diferencial MPVX7002DP

- Rango de medición: ±2kPa.
- Precisión: ±2.5 – 6.25 %VFSS.
- Tiempo de respuesta: 1ms.
- Sensibilidad: 1V/kPa.
- Vs: 5V, Is: 10mA.

Para la adquisición de la señal proveniente del sensor es necesario verificar si es necesario adaptar a esta al rango del conversor analógico/digital del microcontrolador.

Según la hoja técnica del sensor:

$$V_{out} = V_s(0{,}2\ P + 0{,}5) \pm 6{,}25\%\ V_{FSS}$$

Ecuación 2

Donde:

- V_{out} es tensión de salida de sensor
- V_s es tensión de alimentación de sensor (5v)
- P es presión en KPa

Por lo tanto:

$$V_{out_{MAX}} = 3{,}04\ [V]$$

$$V_{out_{MIN}} = 2{,}515\ [V]$$

Calculo del LSB:

$$LSB = \frac{3{,}3V - 0V}{2^{10} - 1} = 0{,}0032258\ [V]$$

Como se puede observar la excursión de la señal de entrada varía desde 2,515V hasta 3,04V y según el cálculo anterior el LSB es de 0,0032 V para una resolución máxima del Conversor Analógico Digital de 10 bit. De manera que para poder obtener una resolución adecuada en la medición de presión, se necesita que la señal excursione en todo el rango del ADC. El circuito implementado para esta función se muestra en la Figura 49 - Circuito de entrada señal Sensor Pitot

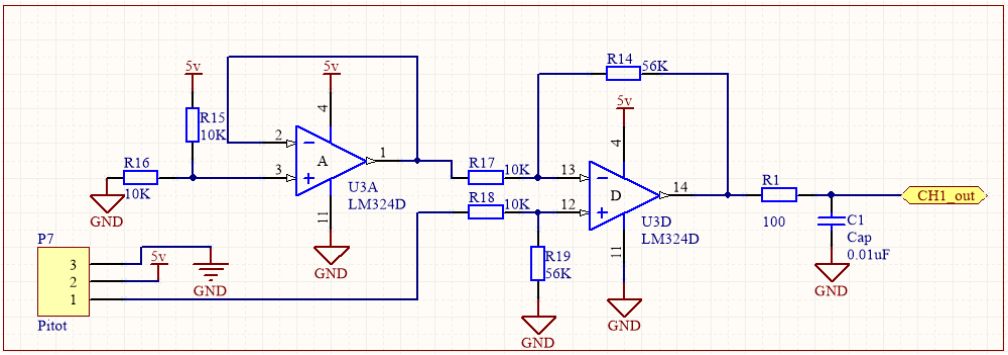


Figura 49 - Circuito de entrada señal Sensor Pitot

Se decide utilizar un circuito buffer no inversor con un circuito restador con ganancia que funcione como offset de la señal del sensor, restando 2,5V a este y multiplicando la diferencia por 5,6 veces, de manera que la excursión de la señal sea en el rango dinámico máximo del conversor AD.

Cálculo de componentes:

$$V_{out} = R_3 \left(\frac{V_2}{R_2} - \frac{V_1}{R_1} \right)$$

Ecuación 3

Si $R_2 = R_1 = R$

$$V_{out} = \frac{R_3}{R} (V_2 - V_1)$$

Ecuación 4

Si V_2=3,04[V], V_1= 2,5[V] y V_{out}=3,3[V] entonces:

$$\frac{R_3}{R} = \frac{3,3}{0,54}$$

$$R_3 = 6,11\, R$$

Si se elige de R= 10K

$R_3 = 56K$

Luego de realizar los cálculos correspondientes de los componentes del circuito, se decide realizar una simulación para verificar el comportamiento deseado.

Circuito a simular:

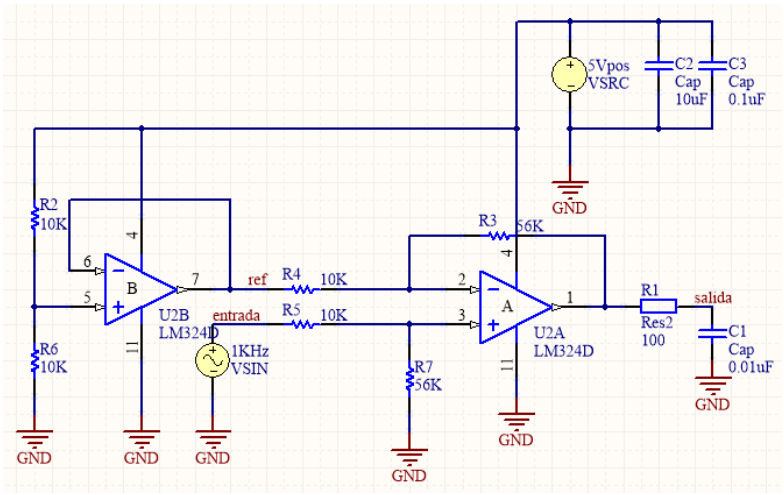


Figura 50 - Circuito a Simular en Altium - Entrada sensor Pitot

Simulación:

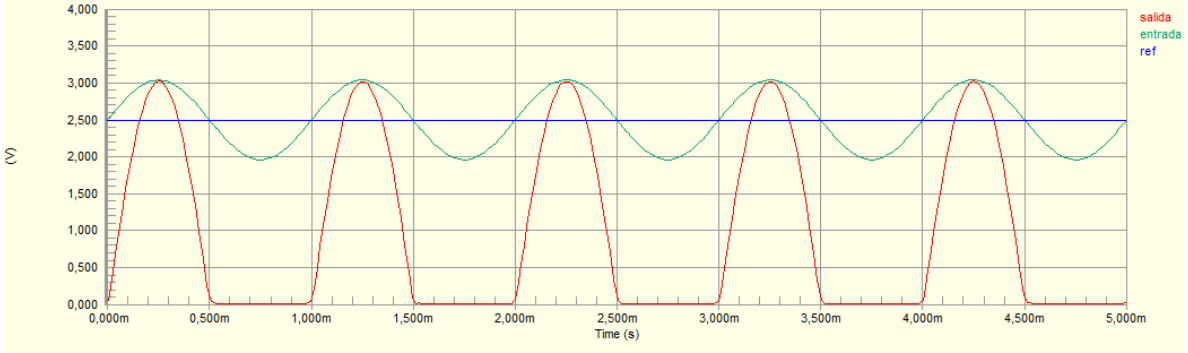


Figura 51 - Resultado de simulación con Entrada = 3,04

Como se observa en la Figura 51 - Resultado de simulación con Entrada = 3,04, con la configuración utilizada, se obtiene a una entrada de V_{outMAX} = 3,04 V, es decir con una $V_{pico\text{-}pico}$ = 0,525 una excursión de salida de 0v a 3,5v ya que el amplificador operacional se satura.

La respuesta en frecuencia del circuito, se muestra en la Figura 52 - Respuesta en frecuencia de circuito sensor de presión diferencial.

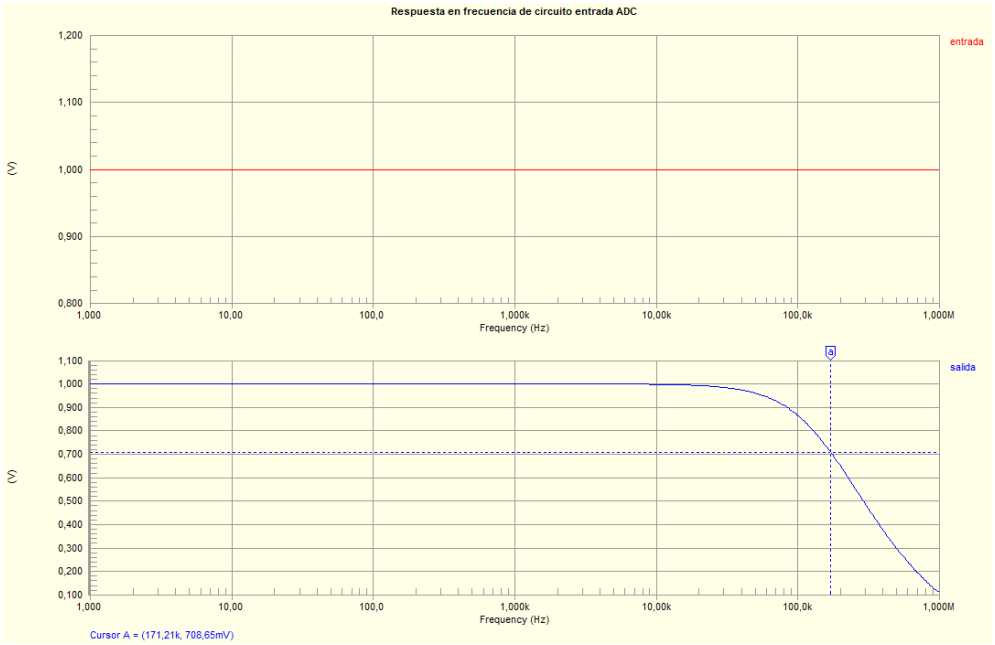


Figura 52 - Respuesta en frecuencia de circuito sensor de presión diferencial

Se decide realizar un estudio de variación de temperatura para observar el comportameinto del circuito ante cambios térmicos que va ser sometido. Es por esto que se toma como parámetro inicial de temperatura a los -10ºC hasta 40ºC cada pasos de 5ºC.

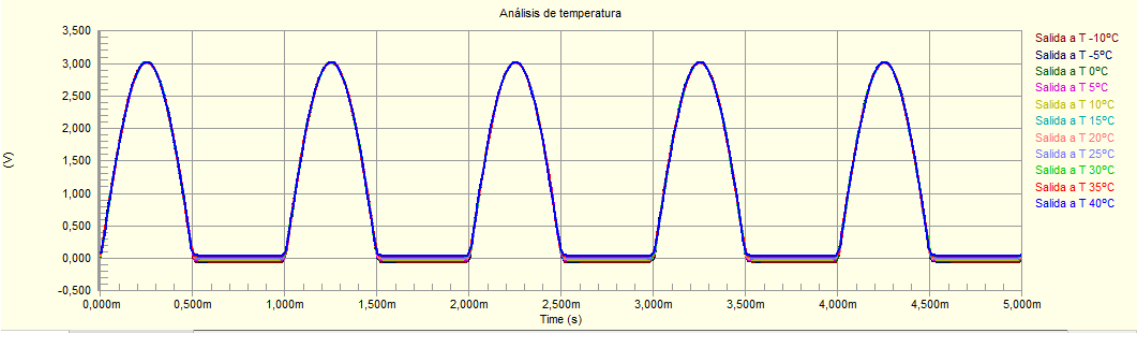


Figura 53 - Simulación de variación de temperatura en circuito sensor Pitot

Como se observa en la Figura 53 - Simulación de variación de temperatura en circuito sensor Pitot, en el peor caso posible no se ve afectado el valor máximo de la señal, pero si el valor mínimo para una temperatura de 40ºC. Analizando detalladamente este último punto, se ve que para 40ºC se tiene un desplazamiento del cero de 58mV lo que genera un corrimiento de 18 cuentas del ADC, representando un recorte del rango de medición de 10,3[Pa] en su punto mínimo. Esto no es significativo, ya que el rango de presiones es de 540[Pa] a 15[Pa].

Adicionalmente se decide hacer una simulación de Montecarlo para estudiar la variabilidad del circuito ante cambios en los componentes.

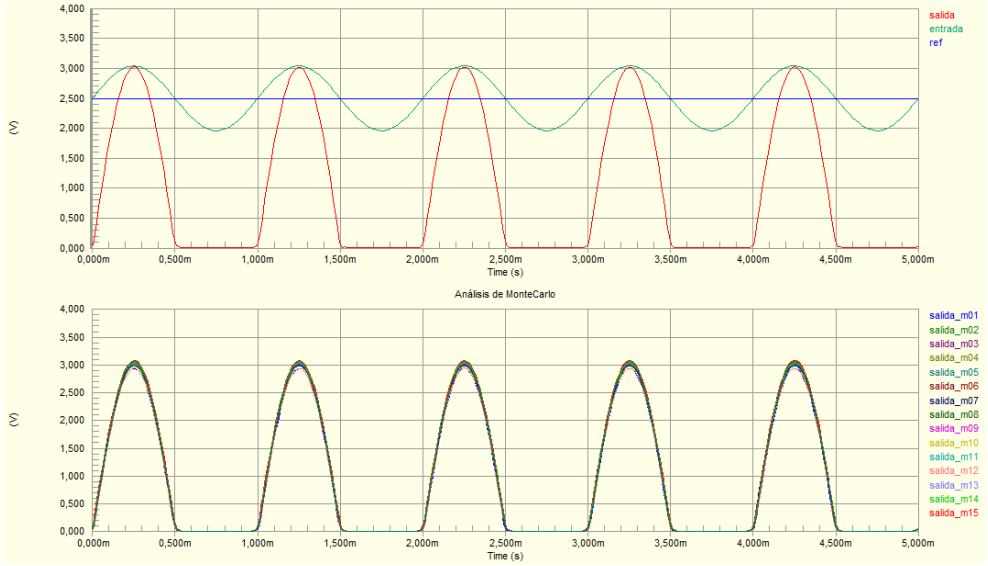


Figura 54 - Simulación de Montecarlo sobre circuito entrada sensor Pitot

Como se observa en la Figura 54 - Simulación de Montecarlo sobre circuito entrada sensor Pitot, analizando los casos mas desfavorables se obtienen los siguientes valores de V de salida, Vmax1 = 3,1 V y Vmax2 = 2,94 V. Dado estos valores se corresponden a 553,5[Pa] y 525[Pa] respectivamente, por lo que se puede deducir un delta de 28,5[Pa] donde el valor máximo se puede encontrar.

Para corregir esto, se toma la decisión de caracterizar el circuito una vez ensamblado y correguir este valor en el firmware.

9.1.1.4 MÓDULO BATSENS

Este módulo contiene la electrónica necesaria para la correcta adquisición de los valores de tensión y corriente del pack de batería que alimenta a todo el sistema.

Requisitos de diseño:

- Tensión de medición: 10V a 16V
- Corriente de medición: 0A a 40A

Ya que el sistema se diseña para realizar entre otras cosas, ensayos de vuelos, se hace necesario medir corrientes superiores a 30A que corresponden al consumo de un motor BLDC que propula la aeronave para realizar el despegue, llegar a nivel de vuelo de ensayo y posteriormente se apaga para comenzar el ensayo.

Se decide utilizar el sensor de corriente por efecto Hall ACS709 de la firma Allegro como lo muestra la Figura 55 - Sensor de corriente ACS709 75A. Para

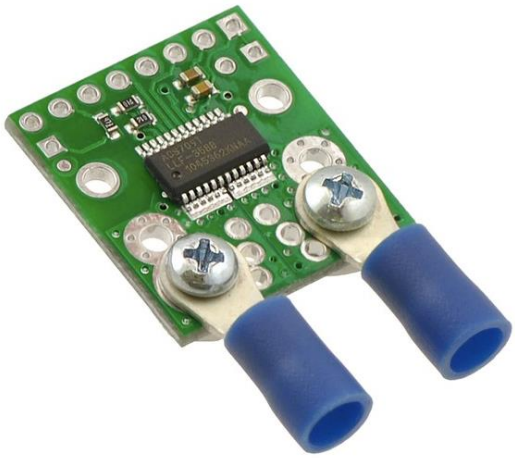


Figura 55 - Sensor de corriente ACS709 75A

Carecterísticas del sensor:

Tensión de operación: 3 - 5.5 V

Sensivilidad: 18.5 mV/A cuando Vcc es 3.3 V o 28 mV/A cuando Vcc es 5 V

Rango de corriente de entrada: rango de detección lineal de -75A a 75A.

Medición de tensión de alimentación

Se decide realizar un circuito utilizando un amplificador operacional en configuración buffer no inversora, atenuando la señal de entrada con un divisor resistivo y se coloca a la salida un filtro antialiasing con frecuencia de corte fc = 16KHz aproximadamente, como se muestra la Figura 56 - Circuito entrada medición tensión de baterias.

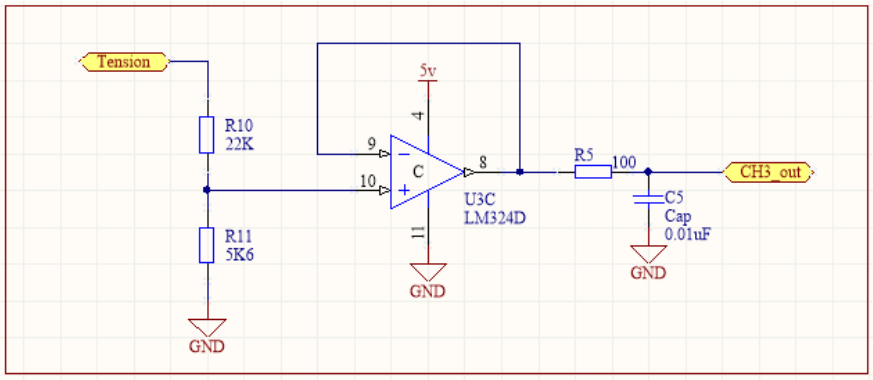


Figura 56 - Circuito entrada medición tensión de baterias

9.1.1.5 MÓDULO GPS

El módulo GPS se encarga de contener la electrónica necesaria para la adquisición de los parámetros de posición y el correcto funcionamiento del receptor de GPS.

Caracterîsticas mínimas correspondiente a módulo receptor EM-406ª que se decide usar:

GPS Receiver

Chipset	SiRF Star III/LP Single
Frequency	L1, 1575.42 MHz
Code	1.023 MHz chip rate
Protocol	Electrical Level: TTL level, Output Voltage Level: 0V~2.85V Baud Rate: 4800 bps Output Message: NMEA 0183 GGA, GSA, GSV, RMC (VTG, GLL optional)
Channels	20
Sensitivity	-159dBm
Cold Start	42 seconds average
Warm Start	38 seconds average
Hot Start	8 second average
Reacquisition	0.1 second average
Accuracy	Position: 10 meters, 2D RMS 5 meters, 2D RMS, WAAS enabled Velocity: 0.1 ms Time: 1 μs synchronized to GPS time
Maximum Altitude	18,000 meters (60,000 feet) max
Maximum Velocity	515 meter/second (1000 knots) max

Maximum Acceleration	4G
Datum	WGS-84
Jerk Limit	20m/sec **3
Physical Characteristic	
Dimensions	1.2" x 1.2" x 0.4" (30mm x 30mm x 10.5mm)
DC Characteristics	
Power Supply	4.5V~6.5V DC Input
Backup Voltage	+2.5V to +3.6V
Power Consumption	44mA (Continuous Mode) 25mA (Trickle Power Mode)
Backup Current	10uA typical
Environmental Range	
Humidity Range	5% to 95% non-condensing
Operation Temperature	-40F to +176F (-40C to 85C)

Tabla 5 - Características receptor GPS EM-406a

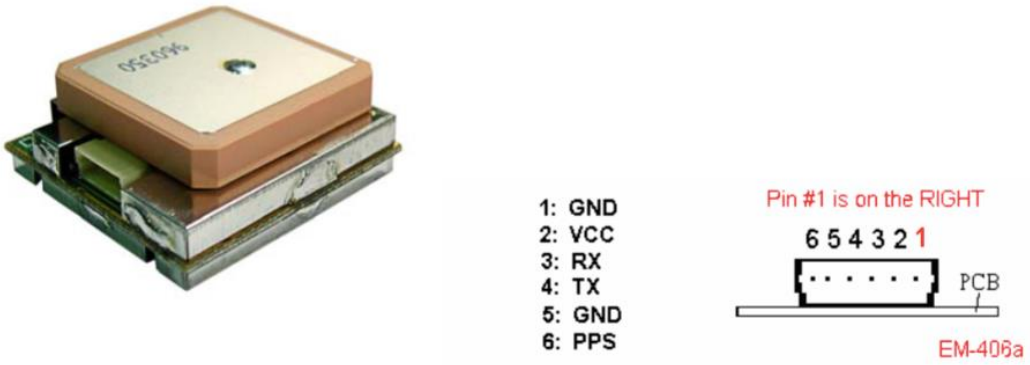


Figura 57 - Módulo GPS EM-406A

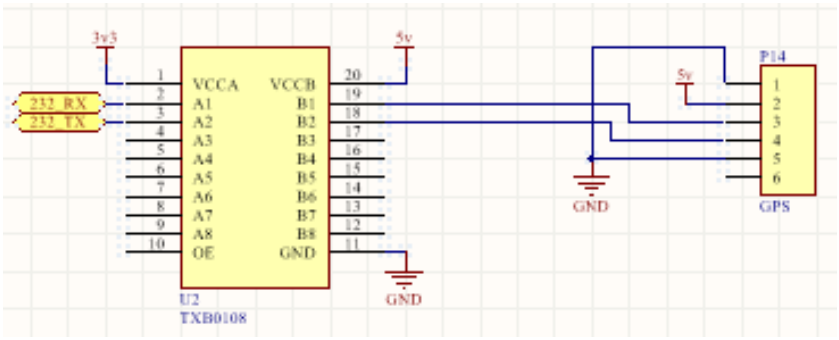


Figura 58 - Circuito adaptador de nivel lógico para GPS

En la Figura 58 - Circuito adaptador de nivel lógico para GPS se observa que para adaptar el nivel lógico de 5V a 3,3V necesario para el microcontrolador se utiliza el intergrado TXB0108 de Texas Instruments.

9.1.1.6 MÓDULO ZIGBEE

Este módulo contiene la electrónica necesaria para el correcto funcionamiento del transceptor digital Xbee utilizado para realizar la telemetría de los valores sensados de todo el sistema.

Características:

- 3.3V @ 50mA

- 250kbps Max data rate
- 1mW output (+0dBm)
- 300ft (100m) range
- Fully FCC certified
- 6 10-bit ADC input pins
- 8 digital IO pins
- 128-bit encryption
- Local or over-air configuration
- AT or API command set
- Trace Antenna

Figura 59 - XBee 1mW serie 1

Es necesario solamente un zócalos para conectar este módulo al microcontrolador.

9.1.1.7 MÓDULO SD CARD

Este módulo contiene la electrónica necesaria para el correcto funcionamiento de una memoria del tipo Secure Digital Card (SD Card), la que almacena los valores sensados de todo el sistema.

Se decide utilizar el módulo Adafruit Micro SD breakout board como se muestra en la Figura 60 - Micro SD Breakout Board.

Características:

- Regulador onboard 5V->3V @ 150mA
- La utilización de un chip adaptador de nivel de 3V a 5V permite interconexión con lógicas de 3V o 5V (Hexa Buffer 74HC4050)
- 3 o 4 pines digitales permiten leer y escribir más de 2GB de almacenamiento.
- LED de actividad indica si la tarjeta está siendo accedida
- Zócalo push-push con la tarjeta ubicada ligeramente fuera del borde del circuito impreso para simplificar las operaciones de inserción y extracción

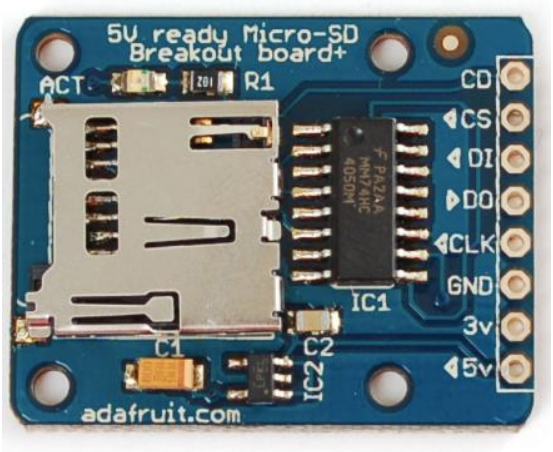


Figura 60 - Micro SD Breakout Board

Es necesario solamente un zócalos para conectar este módulo al microcontrolador.

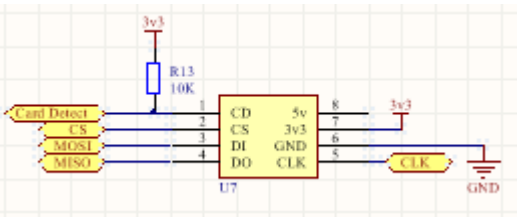


Figura 61 - Implementación de módulo SD-Card

9.1.1.8 SERVOS Y CONTROLADOR DE MOTOR BRUSHLESS

Este módulo contiene la electrónica necesaria para la generación de las señales PWMs (Pulse Width Modulation) las que son necesarios para indicar la posición angular de los servomotores y la velocidad de rotación al Driver del motor brushless.

Se decide utilizar el módulo comercial Adafruit 16-channel 12-bit pwm/servo driver - i2c interface como se ve en la Figura 62 - Módulo salidas PWM.

Características:

- Señal de salida: PWM
- Rango de frecuencia: hasta 1,6 KHz
- Resolución de paso: 12 bit
- Canales de salida: 16 canales
- Interfaz de comunicación: I2C

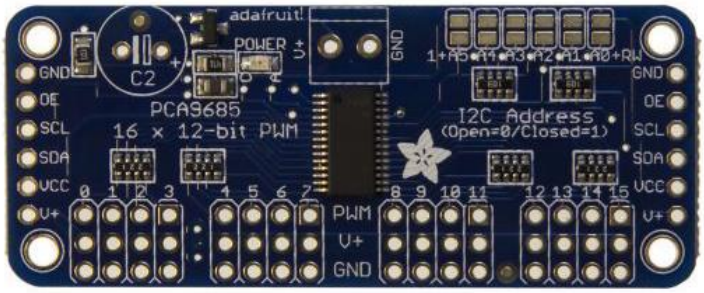


Figura 62 - Módulo salidas PWM

El controlador PCA9685[15] fue diseñado para controlar leds por PWM, pero también nos permite controlar servos, ya que estos también se controlan por PWM, aplicación que actualmente es muy usada.

El Módulo Controlador de servos PCA9685 tiene la placa diseñada para el control de servos, y tiene los pines en el orden correcto para simplemente conectar los servomotores. Además, posee una bornera para la alimentación de los servos y conectores para la alimentación de la parte lógica junto con los pines I2C para comunicarse con arduino.

Se puede establecer la dirección I2C soldando los puentes A0-A5 con esto podemos usar el mismo bus I2C para controlar más módulos PCA9685 u otros dispositivos I2C.

EL PCA9685 nos permite controlar individualmente 16 salidas PWM con 12 bits de resolución y con frecuencia máxima de 1600Hz.

Básicamente lo se tiene que establecer es la frecuencia de la señal PWM, frecuencia que será la misma para las 16 salidas PWM. Para establecer el ciclo de

[15] http://cache.nxp.com/documents/data_sheet/PCA9685.pdf?pspll=1

trabajo (Duty) tenemos que manipular el flanco de subida (Up) y flanco de bajada (Down), esto se configura individualmente para cada salida PWM.

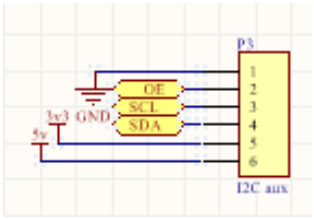


Figura 63 - Salida de I2C aux para módulo PWM

9.1.1.9 SALIDAS DIGITALES DE PROPOSITO GENERAL

Estas salidas se dejan disponibles para uso de "propósito generales", las que pueden ser utilizadas por ejemplo, para activar un dispositivo de apertura de paracaída.

Se decide dejar conectores "header" con las señales disponibles directamente desde el microcontrolador tal como se ve en la Figura 64 - Circuito salidas digitales.

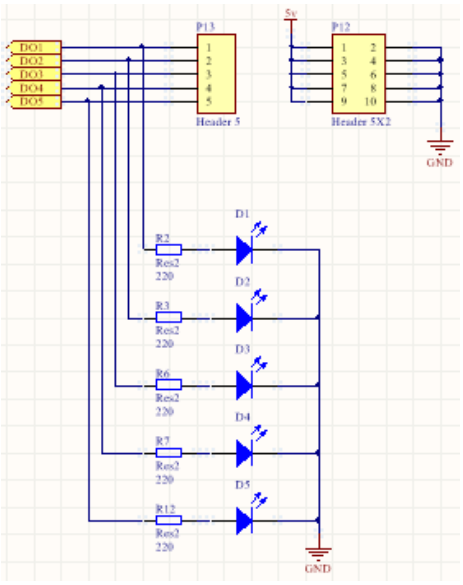


Figura 64 - Circuito salidas digitales

9.1.1.10 ENTRADAS RECEPTOR RadioRX

Las señales de mando provenientes del receptor de radio control Futaba R6008HS[16], del cual se obtienen las salidas de control PWM correspondiente a cada canal de los actuadores (servomotores/motores BLDC), tienen niveles lógicos TTL (0-5v).

[16] https://www.assembla.com/spaces/de-iua/documents/btocR6shOr5ykpacwqjQXA/download/btocR6shOr5ykpacwqjQXA

Figura 65 - Receptor Futaba R6008HS

Según la tabla 11 - Static characteristics del data sheet[17] del microcontrolador LPC4337 (Pag. 93), se obtiene que el V_{IH} máximo es de 5,5V (tolera niveles TTL). Es por esta razón que se decide colocar un conector "header" sin ningún circuito adaptador de nivel como entradas para 8 canales de PWM independientes, tal como se muestra en la Figura 66 - Conector entradas Radio Control.

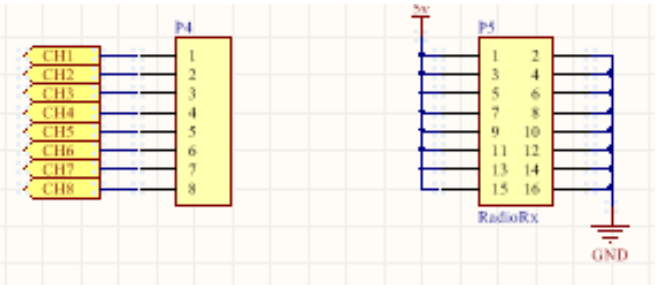


Figura 66 - Conector entradas Radio Control

[17] http://cache.nxp.com/documents/data_sheet/LPC435X_3X_2X_1X.pdf?pspll=1

9.1.2 FIRMWARE

En esta sección se describe el proceso de desarrollo del firmware para el Autopiloto. Como se puede observar en la Figura 67 - Diagrama general del firmware del Autopilot, 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.

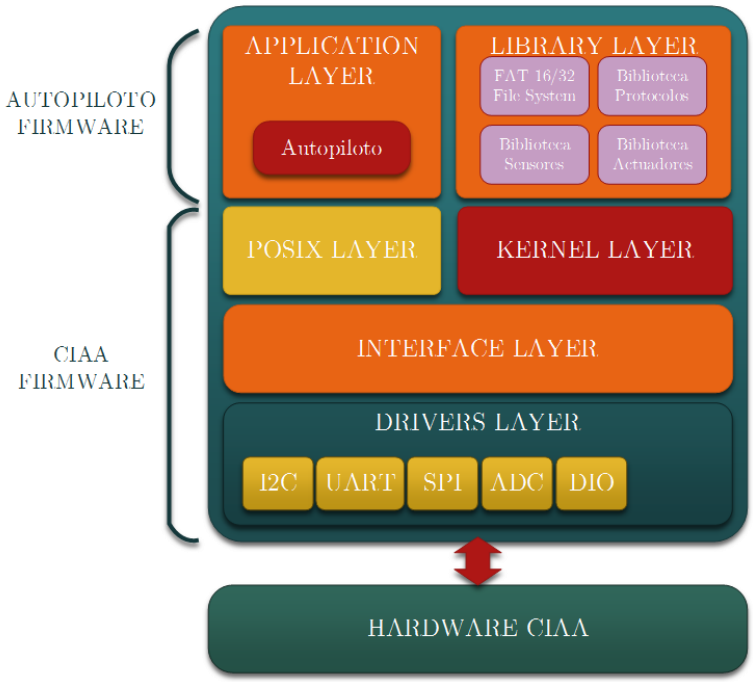


Figura 67 - Diagrama general del firmware del Autopilot

En el punto 9.1.2.1 se explica el funcionamiento de la capa de aplicación, es decir la lógica del piloto automático.

En los puntos 9.1.2.4, 9.1.2.5 y 9.1.2.6 se explican los componentes de la capa de Bibliotecas. Esta tiene por función contener implementaciones de protocolos de comunicaciones y hardware, como sensores o drivers, que se utilicen para el RTOS.

Esta modularidad permite que el desarrollo del sistema sea escalable en características facilmente y que la inteligencia del sistema permanezca independiente del hardware.

9.1.2.1 FUNCIONAMIENTO GENERAL DE TAREAS

Con el fin de describir la implementación general del Aupiloto, inicialmente se parte estudiando el problema y se definen diagramas de caso de uso para las principales tareas.

En la Figura 68 - Diagrama de caso de uso - Registro de datos, se describe la interacción del Autopiloto en el proceso de registro de datos de los sensores y almacenamiento en la memoria RAM interna y en la memoria SD-Card.

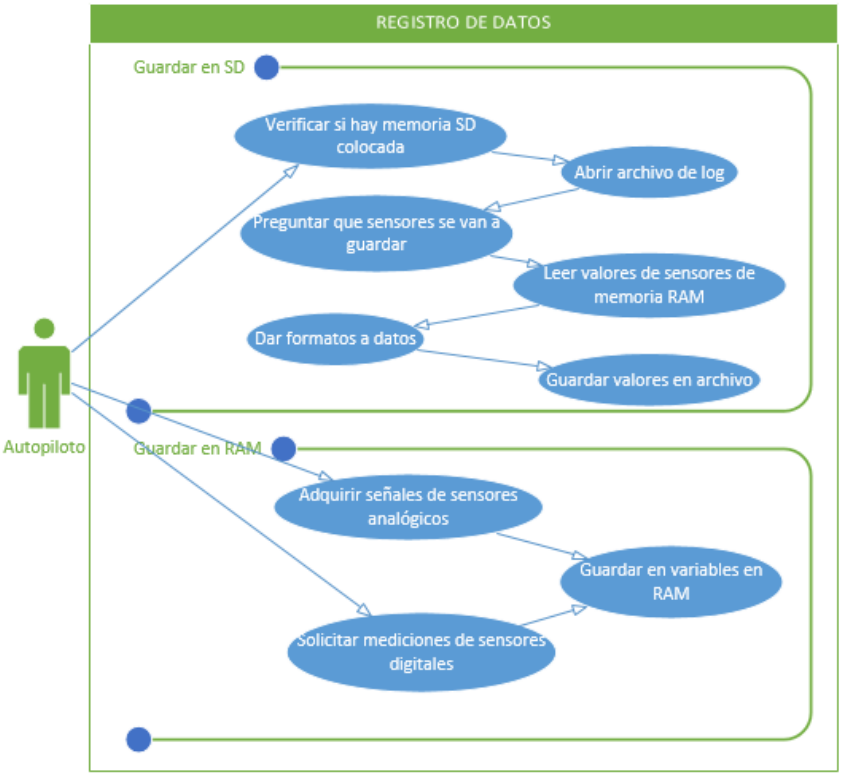


Figura 68 - Diagrama de caso de uso - Registro de datos

En la Figura 69 - Diagrama de caso de uso - Telemetría (autopiloto) y en la Figura 70 - Diagrama de caso de uso - Telemetría (Estación de control en tierra), se describe la interacción del Autopiloto y la Estación de control en tierra en el proceso de telemetría de datos respectivamente.

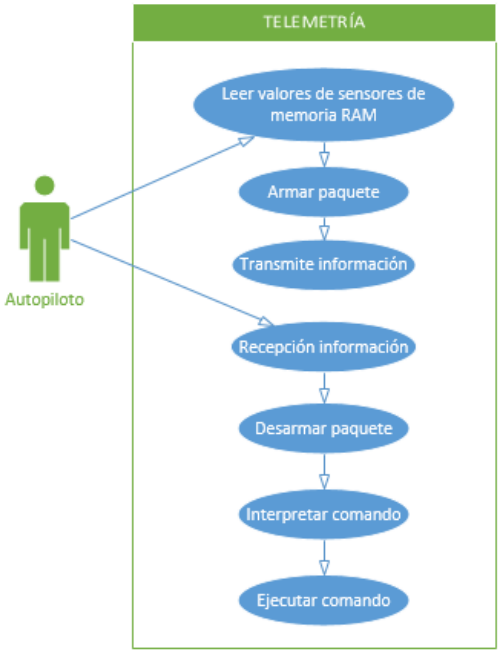


Figura 69 - Diagrama de caso de uso - Telemetría (autopiloto)

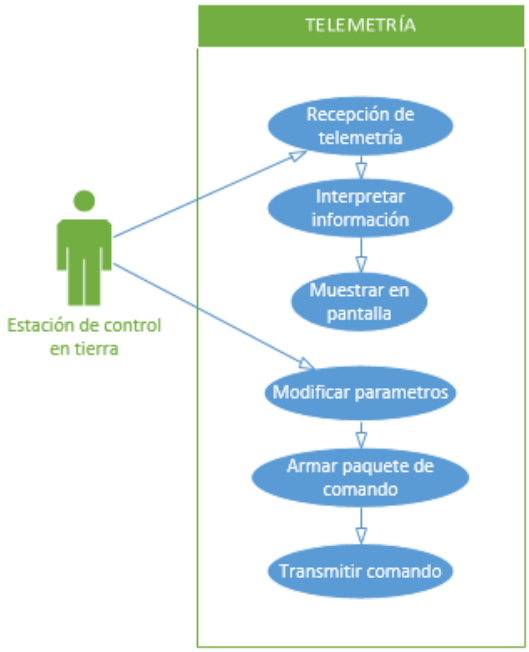


Figura 70 - Diagrama de caso de uso - Telemetría (Estación de control en tierra)

En la Figura 71 - Diagrama de caso de uso – HIL, se describe la interacción del Autopiloto y el usuario cuando se desea iniciar en el modo Hardware in the Loop.

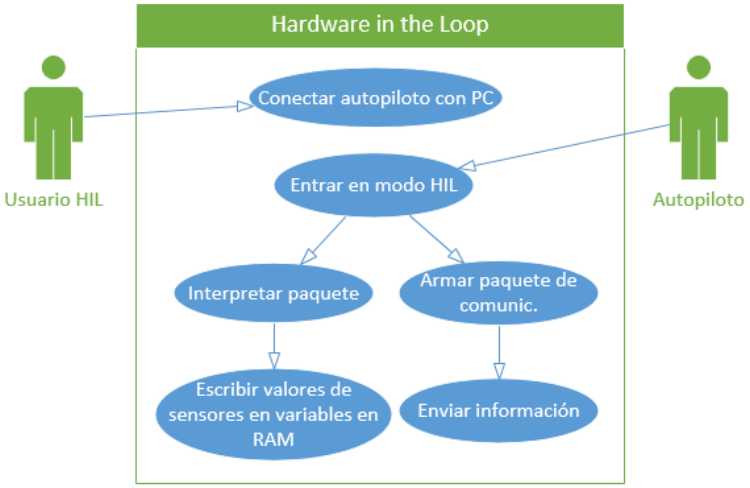


Figura 71 - Diagrama de caso de uso – HIL

Figura 72 - Diagrama de caso de uso - Control, se describe la interacción del Piloto en tierra y el Autopiloto con el sistema de control de los actuadores. En el caso de tener activado la funcionalidad de Autopiloto, este toma realiza los calculos a traves del sistema de control implementado y como resultado actualiza los valores correspondiente a los ángulos de rotación de los servomotores.

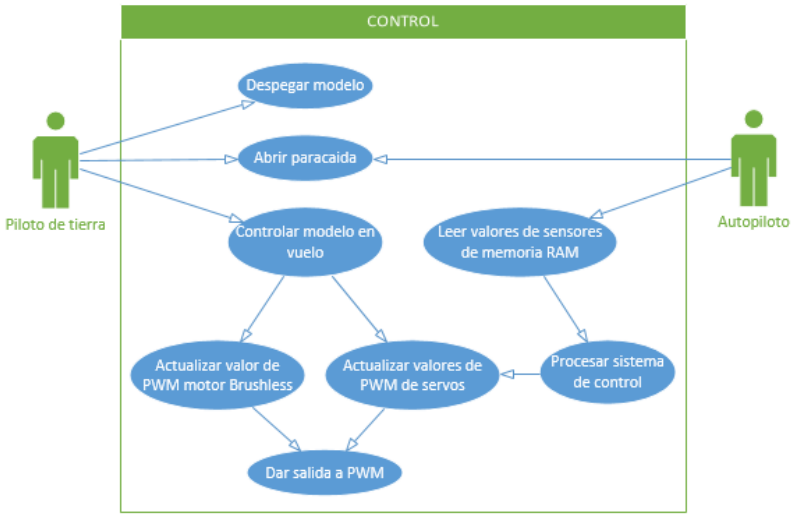


Figura 72 - Diagrama de caso de uso - Control

Modos de funcionamiento

El Autopiloto presenta dos modos de utilización, Modo Misión (MissionMode) y Modo HIL.

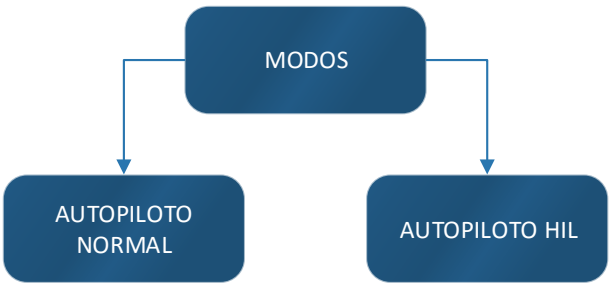


Figura 73 - Modos de funciomiento de Autopiloto

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 para realizar Hardware In the Loop. Ejecuta las tareas de adquisición de comandos HIL por puerto USB/Serie, procesamiento de los datos, procesamiento del sistema de control, telemetría, control de servos y envio de resultados sistema de control por USB/Serie de ser necesario.

Tareas de Inicialización

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

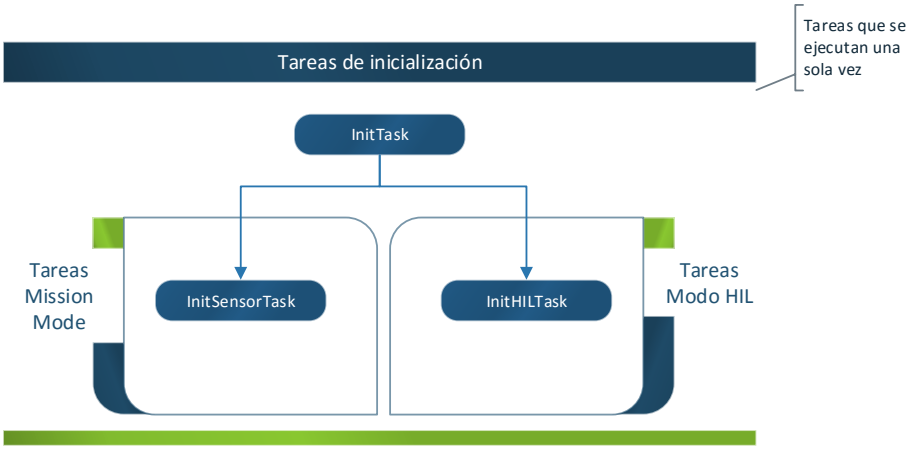


Figura 74 - Tareas de Inicialización

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.

Ejemplo de configuración .OIL para tarea InitTask:

```
TASK InitTask {
   PRIORITY = 10;
   ACTIVATION = 1;
   AUTOSTART = TRUE {
      APPMODE = MissionMode;
   }
   STACK = 512;
   TYPE = BASIC;
   SCHEDULE = NON;
```

```
    RESOURCE = POSIXR;
    EVENT = POSIXE;
}
```

Tarea de prioridad 10, que se puede ejecutar solo una vez y se autoejecuta. Se le asigna un stack de 512bytes y no puede ser interrumpida por el scheduler.

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.

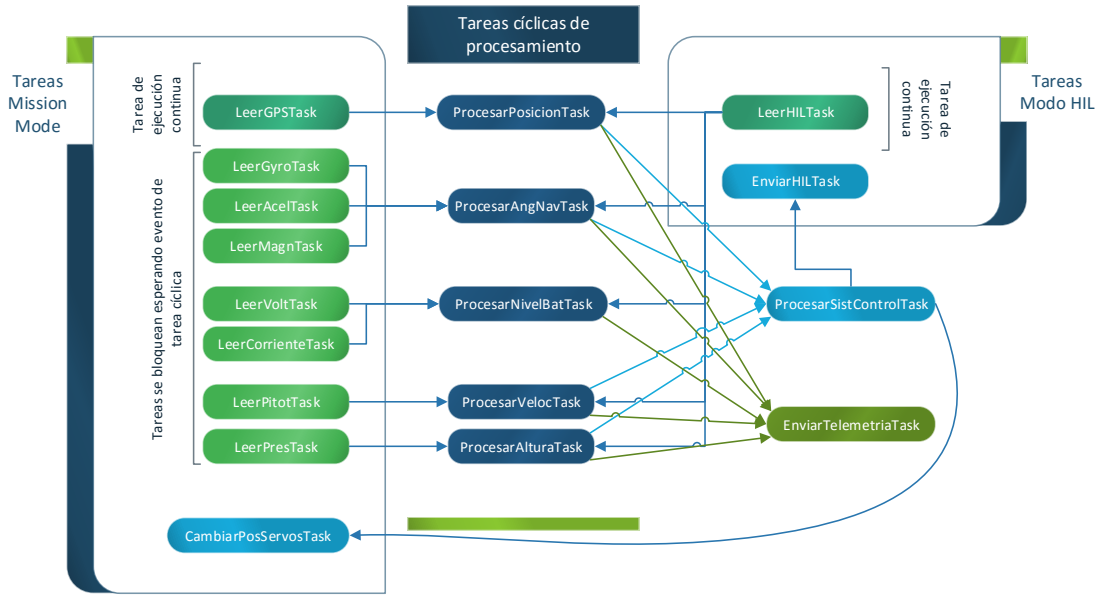


Figura 75 - Diagrama de tareas de ejecución

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 76 - Secuencia de operaciones de inicialización y en la Figura 77 - Secuencia de operaciones de proceso, 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).

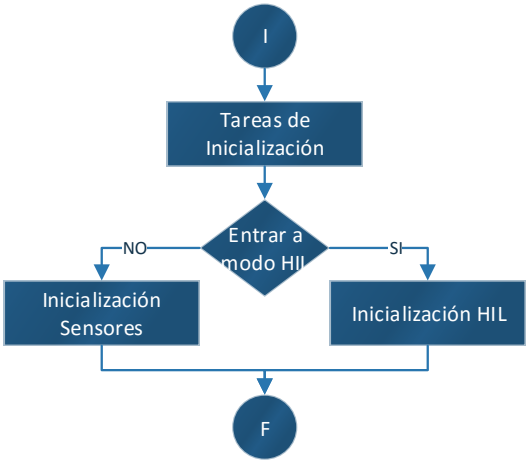


Figura 76 - Secuencia de operaciones de inicialización

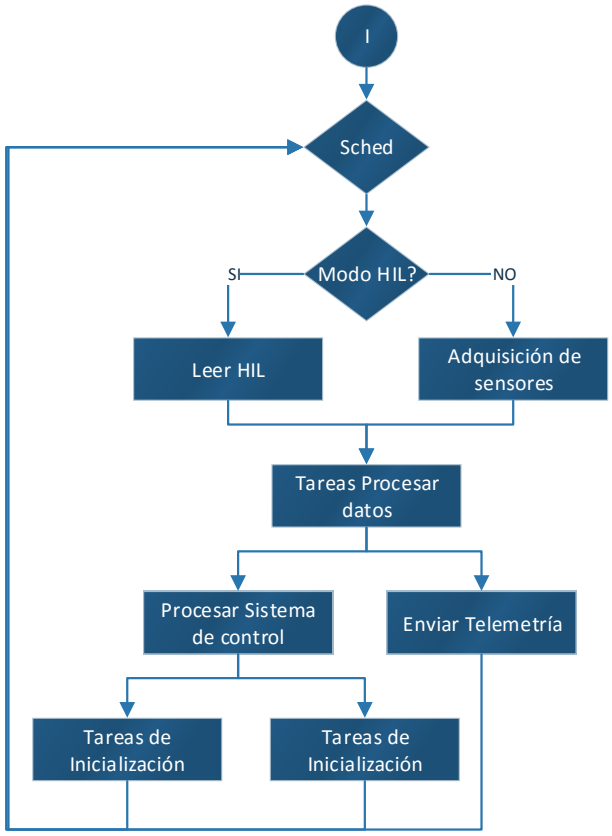


Figura 77 - Secuencia de operaciones de proceso

9.1.2.2 CÁLCULO DE ÁNGULOS DE ROLL, PITCH y YAW

En el presente punto se muestra el método de obtención de los ángulos de navegación (Roll, Pitch y Yaw) para una condición estática.

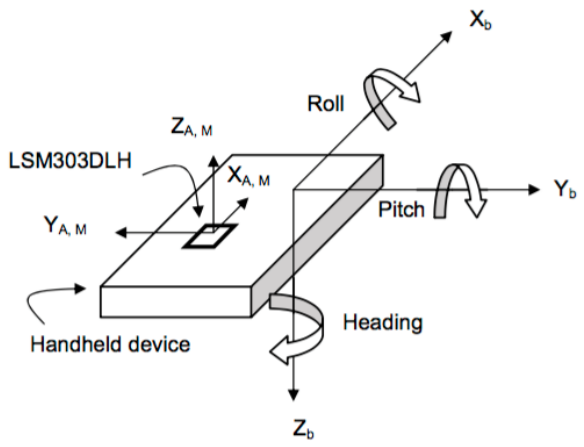


Figura 78 - Sistema coordenado compás magnético

YAW:

El ángulo de guiñada o yaw se calcula utilizando las mediciones del magnetómetro de 3 ejes LSM303H. Inicialmente se realiza la calibración del mismo y luego se normalizan sus componentes X-Y-Z, este vector de campo se proyecta en el plano horizontal y se realiza compensación de inclinación con las mediciones del acelerómetro, como se muestra en la Figura 79 - Calculo ángulo Yaw.

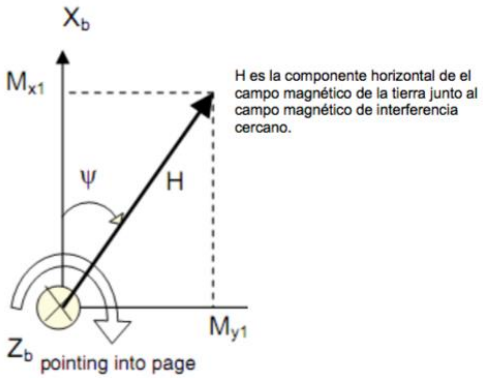


Figura 79 - Calculo ángulo Yaw

Calibración:

Debido a que el compas 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. La distorsión presente es visiblemente identificada por un desplazamiento del origen del círculo ideal desde (0, 0) a Hx-Hy, como se muestra en la Figura 80 - Desviación medición de magnetómetro.

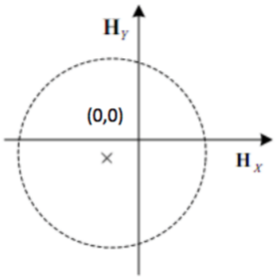


Figura 80 - Desviación medición de magnetómetro

La correcciones de hard-iron se determinan normalmente mediante la rotación del sensor a través de un mínimo de 360° realizando movimientos dentro de una esfera imaginaria, entonces la determinación de la distancia a partir de (0,0) en el centro del círculo mediante la identificación de la media de los valores máximos y mínimos para cada uno de los ejes. Para obtener estos valores es necesario ejecutar el programa de calibración (ver archivo adjuntos con el presenta trabajo – "CalibMag") una vez montado el piloto automático en su estructura final.

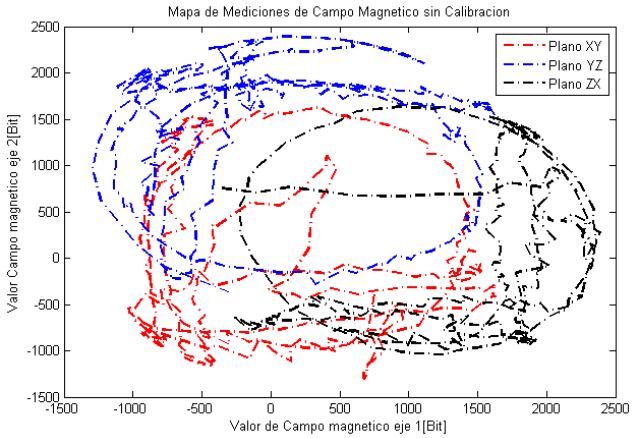


Figura 81 - Gráfico de mediciones sin calibrar – Magnetómetro

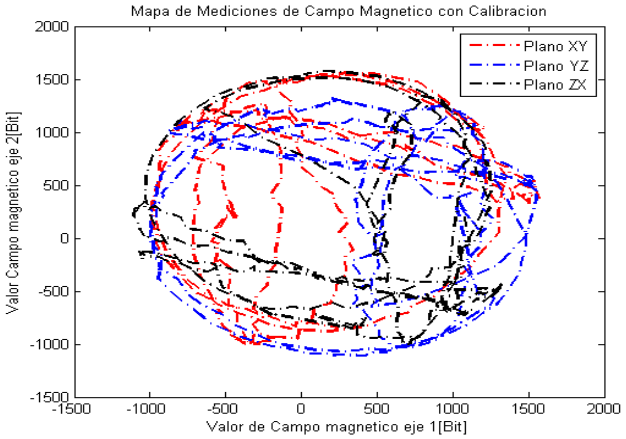


Figura 82 - Gráfico de mediciones calibradas – Magnetómetro

En la **¡Error! No se encuentra el origen de la referencia.** realizada en atlab, se puede apreciar a la izquierda las mediciones sin calibración y a la derecha, en la Figura 82 - Gráfico de mediciones calibradas – Magnetómetro , se ven las mediciones calibradas.

Cálculo de ángulo:

Siguiendo la nota de aplicación[18] con la Figura 78 - Sistema coordenado compás magnético, y si se gira los ejes $X_b/Y_b/Z_b$ hasta $X''_b/Y''_b/Z''_b$ a traves del eje de roll(ϕ) seguido de una rotación a traves del eje de pitch(θ) se tiene,

$$\begin{bmatrix} X_b \\ Y_b \\ Z_b \end{bmatrix} = R_\phi^{-1} R_\theta^{-1} \begin{bmatrix} X_b'' \\ Y_b'' \\ Z_b'' \end{bmatrix} = \begin{bmatrix} \cos\theta & 0 & \sin\theta \\ \sin\phi\sin\theta & \cos\phi & -\sin\phi\cos\theta \\ -\cos\phi\sin\theta & \sin\phi & \cos\phi\sin\theta \end{bmatrix} . \begin{bmatrix} X_b'' \\ Y_b'' \\ Z_b'' \end{bmatrix}$$

Ecuación 5

[18] https://www.pololu.com/file/0J434/LSM303DLH-compass-app-note.pdf

Por lo tanto, siendo M_{x1}, M_{y1} y M_{z1} las mediciones de los sensores magnéticos normalizadas y después de aplicar la corrección por distorsión hard-iron con los parámetros de calibración en el sensor magnético a las mediciones sin calibración M_x, M_y y M_z en nuevas posiciones $X''_b/Y''_b/Z''_b$. A partir de la ecuación, las componentes M_{X2}, M_{Y2}, y M_{Z2} con la compensación por inclinación se puede obtener como:

$$M_{X2} = M_{X1}\cos\theta + M_{Z1}\sin\theta$$

Ecuación 6

$$M_{Y2} = M_{X1}\sin\phi\sin\theta + M_{Y1}\cos\phi - M_{Z1}\sin\phi\cos\theta$$

Ecuación 7

$$M_{Z2} = -M_{X1}\cos\phi\sin\theta + M_{Y1}\sin\phi - M_{Z1}\cos\phi\cos\theta$$

Ecuación 8

Donde los ángulos de giro(θ) y elevación(ϕ) son hallados con el acelerómetro con las Ecuación 16 y Ecuación 17.

$$\psi = \tan^{-1}\frac{M_{Y2}}{M_{X2}}$$

Ecuación 9

PITCH y ROLL:

Los ángulos de pitch (θ) y roll (ϕ) se calculan utilizando las mediciones de aceleraciones del LSM303 y velocidades angulares del L3GD20. Cuando el dispositivo se encuentra en una posición arbitraria con los ejes X'_b, Y'_b y Z'_b, hay algunos procedimientos de rotación para hacer girar el marco de referencia desde

el nivel local X_b, Y_b y Z_b, a la posición 3D. Diferentes procedimientos de rotación resultan en matriz de rotación diferente.

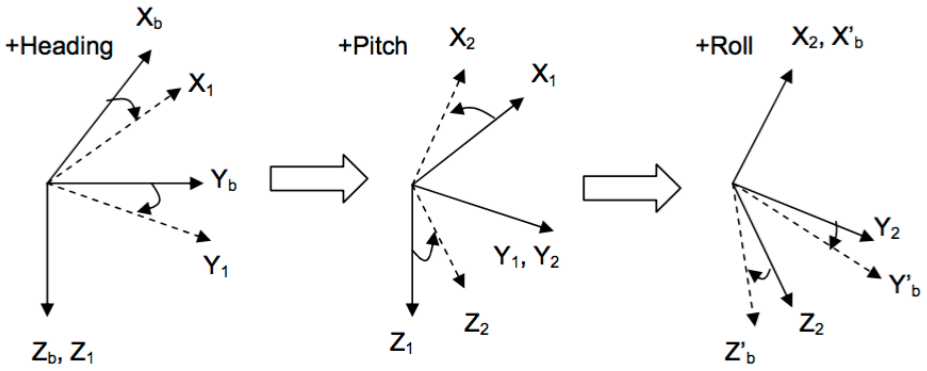


Figura 83 - Procedimiento de rotación

En primer lugar, se gira el eje Z_b en sentido de las agujas del reloj un ángulo(ψ) con la vista desde el origen a la baja. Luego se gira el eje alrededor de Y_b en un ángulo(θ) con X_b hacia arriba. Finalmente se gira el eje alrededor X_b en un ángulo (ϕ) con Y_b hacia abajo. Entonces los nuevos ejes de referencia se convierten en X'_b, Y'_b, y Z'_b, como se muestra en la Figura 9.

Cada matriz de rotación puede definirse según el ángulo como:

$$R_\psi = \begin{bmatrix} \cos\psi & \sin\psi & 0 \\ -\sin\psi & \cos\psi & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Ecuación 10

$$R_\theta = \begin{bmatrix} \cos\theta & 0 & -\sin\theta \\ 0 & 1 & 0 \\ \sin\theta & 0 & \cos\theta \end{bmatrix}$$

Ecuación 11

$$R_\phi = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\phi & \sin\phi \\ 0 & -\sin\phi & \cos\phi \end{bmatrix}$$

Ecuación 12

Y la relación entre X'b, Y'b, Z'b y Xb, Yb, Zb es:

$$\begin{bmatrix} X'_b \\ Y'_b \\ Z'_b \end{bmatrix} = R_\phi \quad R_\theta \quad R_\psi \begin{bmatrix} X_b \\ Y_b \\ Z_b \end{bmatrix}$$

Ecuación 13

En el plano horizontal local mostrado en la Figura 78 - Sistema coordenado compás magnético, $X_b = Y_b = 0$, $Z_b = 1g$. En X'_b / Y'_b / Z'_b, las mediciones en bruto del acelerómetro son A_x, A_y y A_z. Se utilizan los valores normalizados del acelerómetro tipo float y representados por A_{x1}, A_{y1} y A_{z1} se convierten en menos de 1 en términos de g(gravedad de la tierra).

Reemplazando en la Ecuación 13 se convierte en:

$$\begin{bmatrix} A_{X1} \\ A_{Y1} \\ A_{Z1} \end{bmatrix} = R_\phi \quad R_\theta \quad R_\psi \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}$$

Ecuación 14

Reemplazando en la Ecuación 14 las matrices de rotación, se tiene:

$$\begin{bmatrix} A_{X1} \\ A_{Y1} \\ A_{Z1} \end{bmatrix} = \begin{bmatrix} \cos\theta\cos\psi & \cos\theta\sin\psi & -\sin\theta \\ \cos\psi\sin\theta\sin\phi - \cos\phi\sin\psi & \cos\phi\cos\psi + \sin\theta\sin\phi\sin\psi & \cos\theta\sin\phi \\ \cos\psi\sin\theta\cos\phi + \sin\phi\sin\psi & -\sin\phi\cos\psi + \sin\theta\cos\phi\sin\psi & \cos\theta\cos\phi \end{bmatrix} \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}$$

Ecuación 15

Por lo tanto, el ángulo de pitch(θ) y roll(ϕ) se pueden calcular como:

$$\theta = \sin^{-1}(-A_{X1})$$

Ecuación 16

$$\phi = \sin^{-1}\left(\frac{A_{Y1}}{\cos\theta}\right)$$

Ecuación 17

Los ángulos de pitch y roll encontrados, como se vió hasta ahora, se calcularon con las mediciones del acelerometro. Es por esto de que ahora en adelante serán llamadas θ_A y ϕ_A, ya que para los ángulos calculados apartir del girometro serán llamados θ_G y ϕ_G.

La velocidad angular se define como:

$$\vec{\omega} = \frac{\partial\theta}{\partial t}$$

Ecuación 18

Donde θ es el ángulo de rotación.

Para el cálculo de θ_G y ϕ_G se debe utiliza un método numérico para realizar la integración. Se decide utilizar el método de integración de Euler como se muestra en la Ecuación 19.

$$\theta_G(t1) = \theta_G + \Delta t \,.\, \omega_{gyro}$$

Ecuación 19

Donde :

$$\Delta t = (t_1 - t_0)$$

FILTRO COMPLEMENTARIO:

Como se describió anteriormente 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.

Un filtro complementario es en sí, un filtro de Kalman de estado estacionario para una cierta clase de problemas de filtrado. Este no considera ninguna descripción estadística del ruido que corrompe las señales y es obtenido por un simple análisis en el dominio de la frecuencia. Para su implementación se utiliza la función de transferencia de la Ecuación 20.

$$H_{a(s)}G_{(s)} + H_{g(s)}\left(1 - G_{(s)}\right) = 1$$

Ecuación 20

Donde Ha(s) y Hg(s) son las funciones de transferencia del acelerómetro y del girómetro respectivamentes y G(s) es la función de transferencia de un filtro pasa bajo de primer orden.

$$G_{(s)} = \frac{\alpha}{s + \alpha}$$

Ecuación 21

Por lo tanto queda aplicado un filtro pasa bajo a las mediciones de aceleración y un filtro para alto a las mediciones del girómetro.

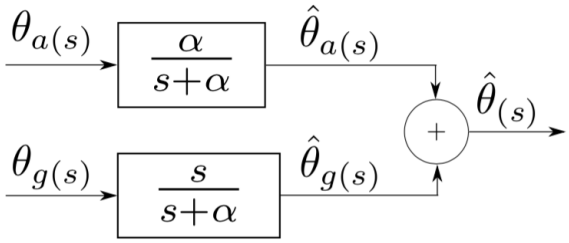


Figura 84 - Filtro complementario[19]

La fórmula resultante de combinar (complementar) los dos filtros se muestra en la Ecuación 22.

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

Ecuación 22

9.1.2.3 DRIVERS CIAA

Se introjeron algunas modificaciones en los drivers de la CIAA con el fin de cambiar las funcionalidades asignadas por defecto a los pines y adaptar estas a las necesidades del proyecto.

ciaaDriverDio.c

En la función ciaa_lpc4337_gpio_init() se eliminan las siguientes líneas de código, las cuales configuran a los pines como salidas digitales.

```
1   static void ciaa_lpc4337_gpio_init(void)
2   {
3      Chip_SCU_PinMux(6,4,MD_PUP|MD_EZI,FUNC0);   /* GPIO3[3], GPIO1 */
4      Chip_SCU_PinMux(6,5,MD_PUP|MD_EZI,FUNC0);   /* GPIO3[4], GPIO2 */
```

[19] http://proyectos.ciii.frc.utn.edu.ar/cuadricoptero/export/9ed95816c90cc7d83e32fd2e13b032dc515c0d7a/documentacion/informe_final/paper_case.pdf

ciaaDriverUart.c

En la función ciaaDriverUart_hwInit() se debe agregan las siguientes líneas de código, las cuales mapean y configuran la UART0 en los pines P9_5 y P9_6, que corresponde a los pines GPIO1 y GPIO1 de la EDU-CIAA:

```
5   static void ciaaDriverUart_hwInit(void)
6   {
7      /* UART0 (RS485/Profibus) */
8      Chip_UART_Init(LPC_USART0);
9      Chip_UART_SetBaud(LPC_USART0, 115200);
10
11     Chip_UART_SetupFIFOS(LPC_USART0, UART_FCR_FIFO_EN |
   UART_FCR_TRG_LEV0);
12
13     Chip_UART_TXEnable(LPC_USART0);
14
15     Chip_SCU_PinMux(9, 5, MD_PDN, FUNC7);              /* P9_5:
   UART0_TXD */
16     Chip_SCU_PinMux(9, 6, MD_PLN|MD_EZI|MD_ZI, FUNC7); /* P9_6:
   UART0_RXD */
17
18     Chip_UART_SetRS485Flags(LPC_USART0, UART_RS485CTRL_DCTRL_EN |
   UART_RS485CTRL_OINV_1);
19
20     Chip_SCU_PinMux(6, 2, MD_PDN, FUNC2);              /* P6_2:
   UART0_DIR */
```

9.1.2.4 BIBLIOTECAS DE SENSORES

Implementacion de sensores en el RTOS

LPS331AP – sensor de presión estática

Para realizar la correcta comunicación del dispositivo se hace necesario inicializar a este correctamente. Esta función la cumple la rutina lps331ap_init() .

```
1   //Initialize a given LPS331AP pressure sensor
2   extern int32_t * lps331ap_init (void);
```

La rutina lps331ap_read_temp() lee los registros correspondientes a la temperatura adquirida y retorna el valor el miligrados Celsius.

```
1   //Read a temperature value from the given sensor, returned in m°C
2   extern int   lps331ap_read_temp (int32_t);
```

La rutina lps331ap_read_press() lee los registros correspondientes a la presión barométrica adquirida y retorna el valor el milibares.

```
1   //Read a pressure value from the given sensor, returned in mbar
2   extern int   lps331ap_read_press (int32_t);
```

La rutina lps331ap_whoAmI() escribe en el registro 0x0F y espera el valor de retorno 0xBB, el cual indica que el dispositivo se encuentra en el bus y funcionando correctamente.

```
1   extern int32_t lps331ap_whoAmI (int32_t fildes);
```

LSM303D – sensor de aceleraciones 3D y campo magnético 3D

Para realizar la correcta comunicación del dispositivo se hace necesario inicializar a este correctamente. Esta función la cumple la rutina lsm303d_init() .

```
1   //Initialize a given LSM303D accelerometer/magnetometer sensor
2   extern int32_t * lsm303d_init (void);
```

La rutina lsm303d_read_axis_accel() lee las aceleraciones en los 3 ejes y guarda estos valores.

```
1   //Read axis
2    void lsm303d_read_axis_accel (int32_t fildes, int16_t * x, int16_t *
    y, int16_t * z);
```

La rutina lsm303d_read_axis_magn() lee los valores de campo magnético en los 3 ejes y guarda estos valores.

```
1   /** Funcion que devuelve */
2    void lsm303d_read_axis_magn (int32_t fildes, int16_t * x, int16_t * y,
    int16_t * z);
```

La rutina lsm303d_whoAmI() escribe en el registro 0x0F y espera el valor de retorno 0x49, el cual indica que el dispositivo se encuentra en el bus y funcionando correctamente.

```
1   extern int32_t lsm303d_whoAmI (int32_t fildes);
```

L3GD20H – sensor velocidad angular 3D

Para realizar la correcta comunicación del dispositivo se hace necesario inicializar a este correctamente. Esta función la cumple la rutina l3gd20h_init().

```
1   //Initialize a given l3gd20h gyro sensor
2   extern int32_t * l3gd20h_init (void);
```

La rutina l3gd20h_read_axis() lee los valores de velocidad angular que corresponden a los 3 ejes y los guarda.

```
1   //Read axis
2    void l3gd20h_read_axis (int32_t fildes, int16_t * x, int16_t * y,
    int16_t * z);
```

La rutina l3gd20h_whoAmI() escribe en el registro 0x0F y espera el valor de retorno 0xD7, el cual indica que el dispositivo se encuentra en el bus y funcionando correctamente.

```
1   extern int32_t l3gd20h_whoAmI (int32_t fildes);
```

9.1.2.5 BIBLIOTECA PROTOCOLO DE COMUNICACIÓN

EM-406A – receptor de GPS TRAMA NMEA

Es necesario configurar el dispositivo y el recurso que este utiliza y es en la rutina em406a_init() donde se realiza esta tarea.

```
1   //Initialize a given GPS receptor
2   extern int32_t * em406a_init (void);
```

La rutina em406a_read_gga() lee los datos del receptor GPS, mas explícitamente la trama GGA y devuelve los parámetros de latitud, longitud, altura y número de satélites usados.

```
3   //Read GGA parameters
4    void em406a_read_gga (int32_t fildes, int16_t * lat, int16_t * long,
    int16_t * alt, int16_t * satNum);
```

Protocolo de telemetría

Se decide utilizar los mismos nombres de funciones que implementa el proyecto el ArduPilot para utilizar el presente protocolo.

La rutina print_attitude() envía los parametros velocidad de aire, ángulo de roll, ángulo pitch y posición de acelerador, este último parametro no utilizado en el proyecto actual.

```
1   void print_attitude(int32_t fildes, char * strPitch, char * strRoll);
```

La rutina print_control_mode() envía el modo de control que se está ejecutando. Actualmente se encuentra fijado en "TESTMODE".

```
1   void print_control_mode(int32_t fildes);
```

La rutina print_position() envía los datos de posición como Latitud, Longitud, Altura, Velocidad respecto de tierra, ángulo yaw, etc.

```
1   void print_position(int32_t fildes, char *strYaw);
```

Rutina que en función del "id" llama a las funciones print_position(), print_control_mode() o print_attitude().

```
1   void send_message(int32_t fildes, int16_t id, long param, char
    *strPtrYaw, char *strPitch, char *strRoll);
```

Única rutina llamada necesariamente para enviar los datos. Esta llama a send_message() y en función al "id" seleccionado será el mensaje enviado

```
1   void send_message_GCS(int32_t fildes, int16_t id, char *strPtrYaw, char
    *strPitch, char *strRoll);
```

9.1.2.6 BIBLIOTECA ACTUADORES

El PCA9685, circuito integrado que se utiliza para generar las señales de PWM de control hacia los servomotores y variadores de velocidad de motores BLDC. Para comunicarse, inicializar el CI y controlarlo, utiliza el bus I2C. Es por esta razón la necesidad de la rutina pca_init(), de inicializar.

```
1   extern int32_t * pca9685_init (void)
2   {
3       char tmp;
4       int32_t fildes;
5
6       // Hacer un try del recurso para asegurar que se capturó
7       fildes = ciaaPOSIX_open("/dev/i2c/0/pca9685", ciaaPOSIX_O_RDWR);
8
9       ciaaPOSIX_ioctl(fildes,ciaaPOSIX_IOCTL_SET_SLAVEADD, (void
    *)PCA9685_DEFAULT_ADDRESS );
10      ciaaPOSIX_ioctl(fildes,ciaaPOSIX_IOCTL_SET_REGISTERADDWIDTH, (void
    *) ciaaPOSIX_IOCTL_REGISTERADDWIDTH_8bits);
11      ciaaPOSIX_ioctl(fildes,ciaaPOSIX_IOCTL_SET_CLOCKRATE, (void *)
    ciaaPOSIX_IOCTL_CLOCKRATE_400000);
12
13        /* Reset the device */
14      ciaaPOSIX_ioctl(fildes,ciaaPOSIX_IOCTL_SET_REGISTERADD, (void
    *)PCA9685_MODE1 );
15        tmp= 0x01;
16      ciaaPOSIX_write(fildes, &tmp, 1);
17      pca9685_setPwmFreq(fildes, PCA9685_DEFAULT_FREQ_HZ);
18      return (int32_t *)fildes;
19  }
```

Se hace un open al recurso "/dev/i2c/0/pca9685", se configura la dirección del dispositivo, el ancho de los registros y el clock del bus. Luego se hace un reset del dispositivo y se configura la frecuencia de trabajo de las señales PWM.

La rutina pca9685_setPwmFreq() configura la frecuencia de trabajo de la señal de PWM de salida en Hz.

```
1   extern int32_t pca9685_setPwmFreq(int32_t fildes, float freq)
```

La rutina pca9685_setPwm() configura el duty-cycle del canal (servomotor) que se requiera.

```
1   extern int32_t pca9685_setPwm(int32_t fildes, int16_t servoNum, int16_t
    timeOn, int16_t timeOff
```

La rutina pca9685_write() escribe un byte en el registro especificado (reg).

```
1   extern int32_t pca9685_write(int32_t fildes, int16_t reg, char data)
```

La rutina pca9685_read() lee un byte del registro especificado (reg).

```
1   extern int32_t pca9685_read(int32_t fildes,int16_t reg, char * data)
```

9.2 METODOLOGÍA

La primera parte del trabajo final de grado 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 explerimental multipropósito, 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 finales del conjunto obteniéndose las conclusiones finales del trabajo.

9.3 ACTIVIDADES REALIZADAS

En ANEXO 1 – Planning de la página 167, se encuentra el diagrama de Gantt con el detalle de las actividades.

9.4 RESULTADOS ALCANZADOS

9.4.1 ESQUEMATICOS

Los esquemáticos de todos los circuitos que se desarrollaron se encuentran en el ANEXO 2 – Circuito esquemático.

9.4.2 PONCHO CIAAPILOT

En las Figura 85 - Diseño de circuito impreso - Capa Top y Figura 86 - Diseño de circuito impreso - Capa Bottom, se muestran las capturas de pantalla del diseño del circuito impreso realizado en Altuim Designer. Como resultado del

proceso se muestra en la Figura 87, Figura 88 y Figura 89, el circuito impreso fabricado previo al ensamblaje final. Como se puede ver, 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 y se rompa el circuito.

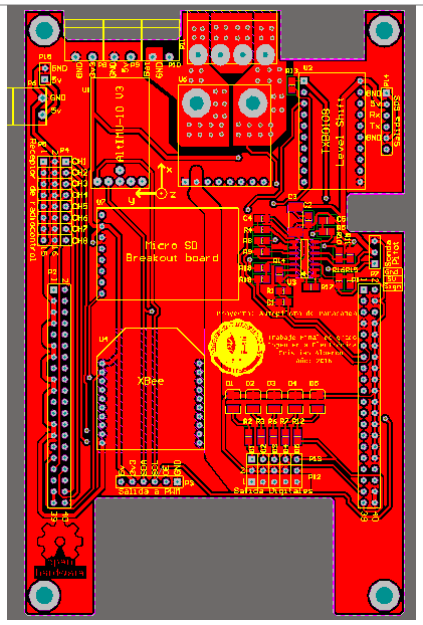


Figura 85 - Diseño de circuito impreso - Capa Top

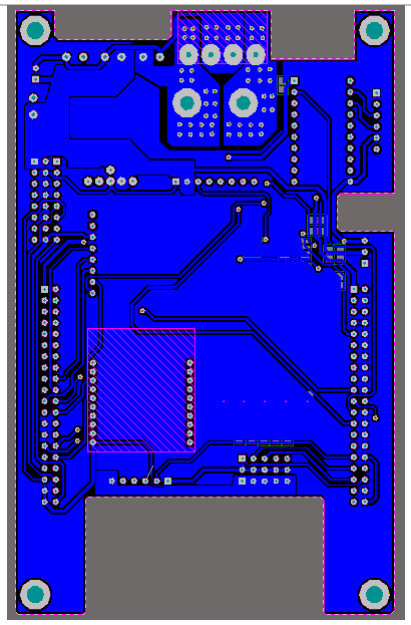


Figura 86 - Diseño de circuito impreso - Capa Bottom

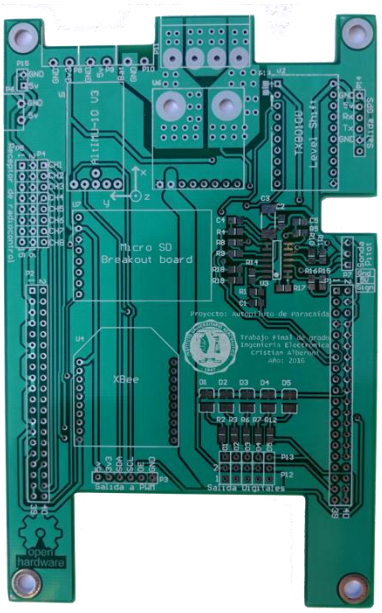


Figura 87 - PCB fabricada - Capa Top

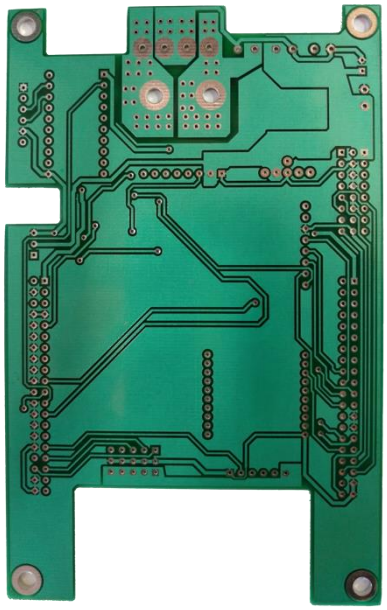


Figura 88 - PCB fabricada - Capa Bottom

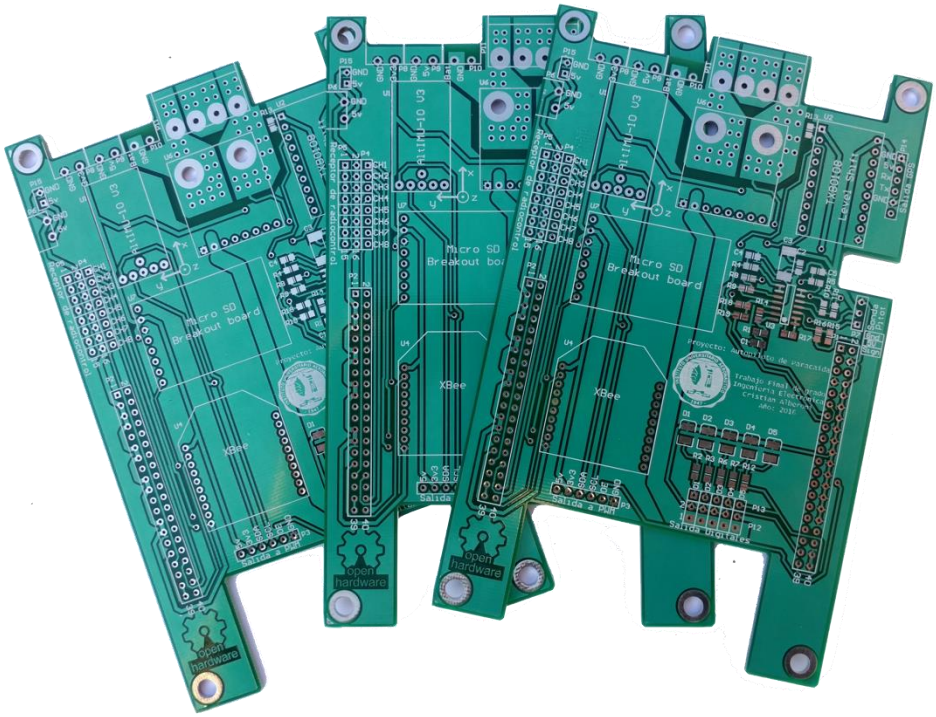


Figura 89 - PCB CIAAPilot

9.4.3 MEDICIÓN DE CONSUMO DE CORRIENTE

Se realizó una medición del consumo de corriente del circuito, este comprende a la fuente switching, la plataforma EDU-CIAA y el poncho CIAAPILOT montado sin actuadores (servomotores o motores brushless).

Instrumentos utilizados:

- Fuente de alimentación variable – Marca: V&a, Modelo: Hy3003d-2
- Multímetro digital – Marca: Unit , Modelo: UT30

Resultados:

Gráfico de variación de consumo de corriente [mA] en función a la tensión de entrada.

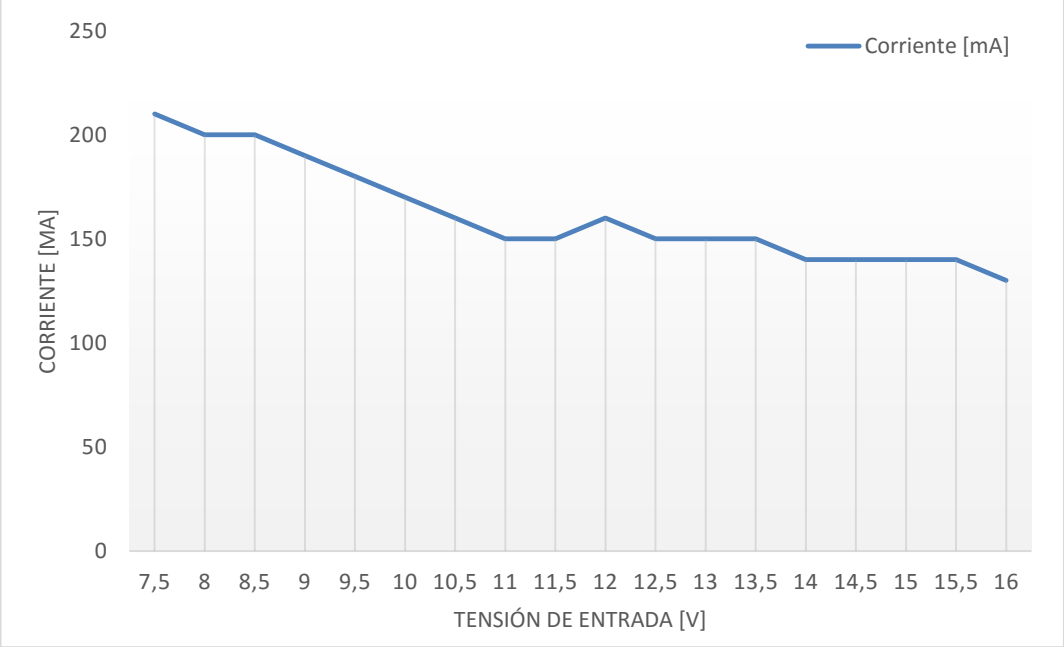


Tabla 6 - Resultado medición corriente consumida

Tensión de entrada [V]	Corriente [mA]

7,5	210
8	200
8,5	200
9	190
9,5	180
10	170
10,5	160
11	150
11,5	150
12	160
12,5	150
13	150
13,5	150
14	140
14,5	140
15	140
15,5	140
16	130

El circuito presenta un consumo de corriente media de 160 mA y un pico de corriente 210mA en 7,5V de tensión de entrada.

9.4.4 MEDICIÓN DE FILTRO COMPLEMENTARIO – ACELERACIONES Y VELOCIDAD ANGULAR

Como se puede observar en la Figura 90 - Cálculo de ángulo con filtro complementario Vs sin filtrado, se muestra los resultados obtenidos del ángulo calculado con el acelerómetro, luego con el girómetro y finalmente aplicando el filtro complementario.

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 pertenecienta a las mediciones de ángulo aplicando el filtro, se ve suavisada y no presenta deriva, tal como se esperaba.

El script realizado en Matlab que se programó para realizar las mediciones, se adjunta en el ANEXO 3 – Script Matlab para adquisición de filtro complementario.

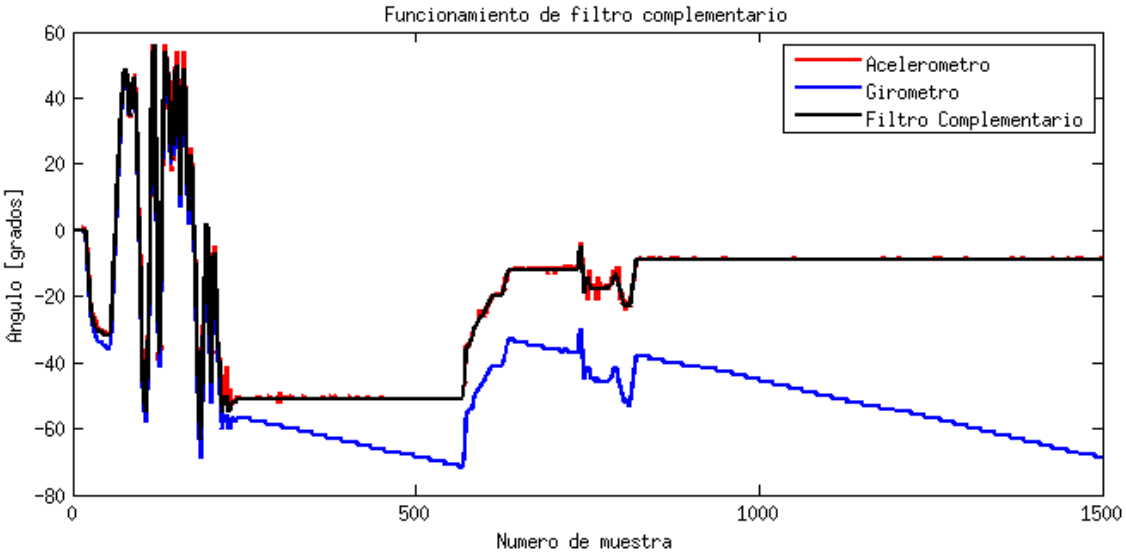


Figura 90 - Cálculo de ángulo con filtro complementario Vs sin filtrado

9.4.5 MEDICIÓN DE ÁNGULOS (YAW, PITCH Y ROLL)

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. En la Figura 91 - Dispositivo para verificación de ángulos, se puede observar que la base de fijación que contiene al autopiloto, rota sobre el eje permitiendo solo un grado de libertad. La medición del ángulo se realiza a traves del transportador solidario al eje.

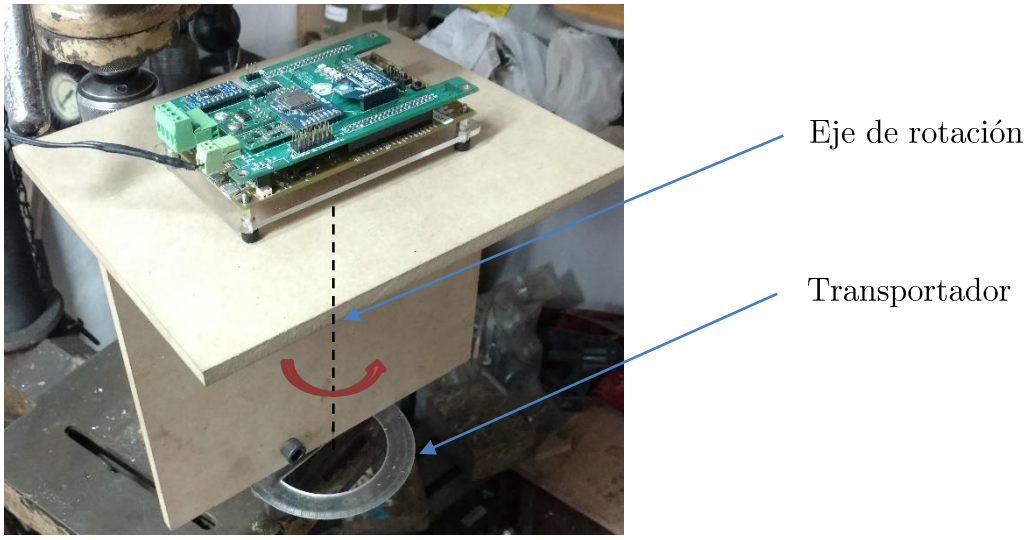


Figura 91 - Dispositivo para verificación de ángulos

A continuación se muestra el gráfico de linealidad de las mediciones de los ángulos de roll, pitch y yaw (según Tabla 7 - Medición Yaw, Tabla 8 - Medición Pitch y Tabla 9 - Medición Roll)

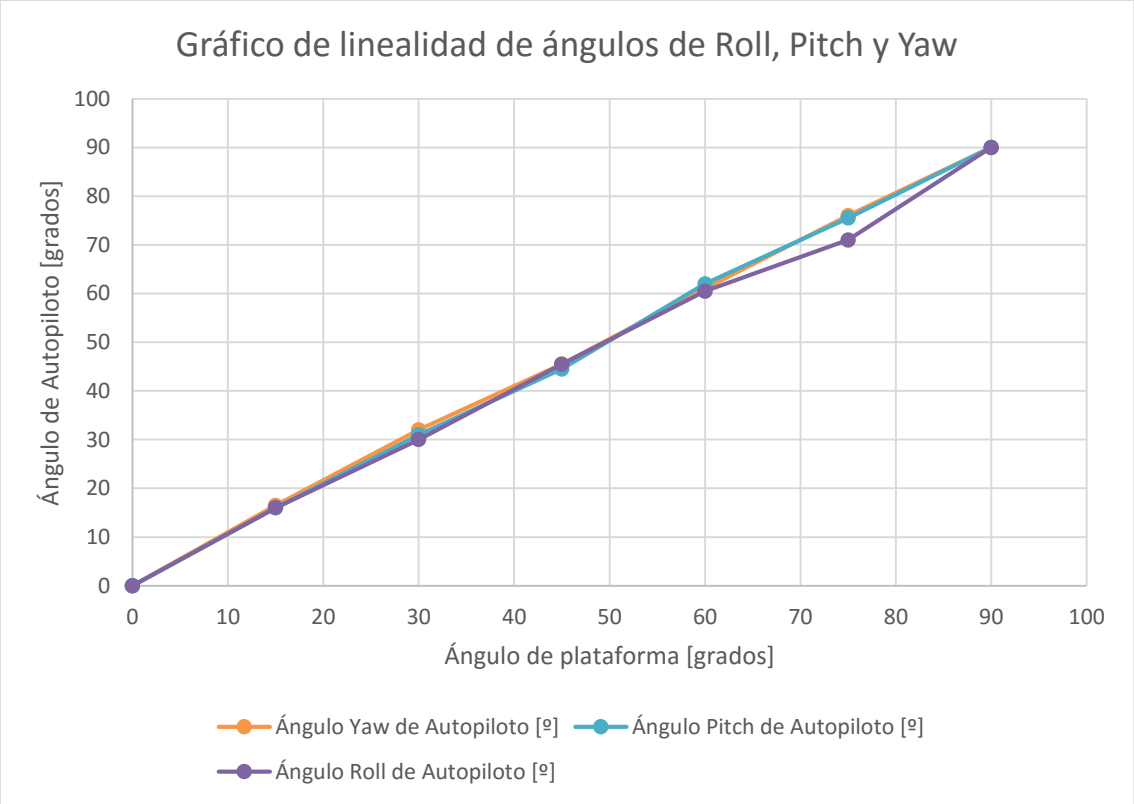


Tabla 7 - Medición Yaw

Ángulo Yaw de plataforma [º]	Ángulo Yaw de Autopiloto [º]
0	0
15	16,5
30	32
45	45,5
60	61
75	76
90	90

Tabla 8 - Medición Pitch

Ángulo Pitch de plataforma [º]	Ángulo Pitch de Autopiloto [º]
0	0

15	16
30	31
45	44,5
60	62
75	75,5
90	90

Tabla 9 - Medición Roll

Ángulo Roll de plataforma [º]	**Ángulo Roll de Autopiloto [º]**
0	0
15	16
30	30
45	45,5
60	60,5
75	71
90	90

9.4.6 MEDICIÓN DE PWM

Se mide que la señal de salida entregada por el circuito integrado PCA9685 es la configurada, es decir su frecuencia, duty cycle son las requeridas.

En la Tabla 10 - Medición de señal PWM salida, se verifica la frecuencia y duty cycle de la señal de PWM de salida respecto de la configurada digitalmente al PCA9685. Se seleccionan los valores de 50Hz y 60Hz, ya que son los típicos para el trabajo en actuadores como servomotores y controladores de velocidad de motores BLDC.

Tabla 10 - Medición de señal PWM salida

Frecuencia configurada [Hz]	**Frecuencia de salida [Hz]**	**Duty Cycle configurado [%]**	**Duty Cycle de salida [%]**
50	51,125	0	0
50	51,125	25	25
50	51,125	50	50
50	51,125	75	75
50	51,125	100	100
60	61,340	0	0
60	61,340	25	25
60	61,340	50	50
60	61,340	75	75
60	61,340	100	100

9.4.7 MEDICIÓN DE ALTURA SENSOR PRESIÓN

Para realizar la comprobación de las mediciones de altura calculadas, se propone utilizar una camara de vacio con un vacuómetro, adecuado para esta tarea, que permita constrastar los valores tomados.

Rango de medición: presión atmosférica hasta 126Kpa a 26KPa.

En el presente trabajo se decide dejar pendiente este ensayo debido a que no se dispone del equipamiento necesario. Cabe aclara que esto no impide verificar el correcto funcionamiento del firmware desarrollado.

9.4.8 FOTOS DE SISTEMA COMPLETO

En la Figura 92 y en la Figura 93 se puede apreciar la EDU-CIAA y el Poncho CIAAPilot ensamblado.

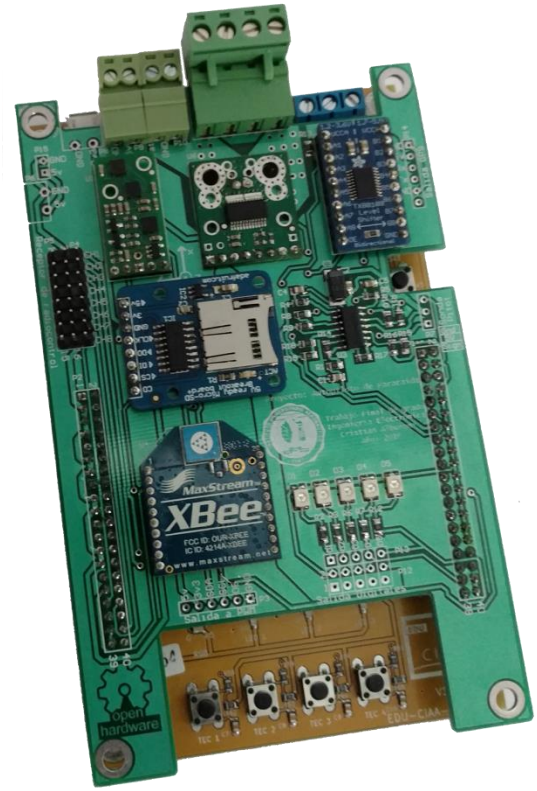


Figura 92 - Foto sistema completo - Frontal

Figura 93 - Foto sistema completo - Lateral

9.5 CONTROL DE COSTOS

A continuación se detallan los materiales empleados y los costos de estos.

Como se puede observar solo se hace una descripción de los materiales requeridos para el armado del prototipo de desarrollo o Target de desarrollo, despreciando los costos de mano de obra de desarrollo, instalaciones y equipos de medición que

fueron provistos por el Departamento de Eléctronica y Telecomunicaciones, como así también por el Laboratorio de Electrónica del IUA.

Prototipo de Desarrollo (PD)

Ítem	Material	Descripción	Proveedor	Cantidad	Costo Unitario	Costo total
1	EDU-CIAA	Computadora Industrial Abierta Argentina	IUA	1	$ 550,00	$ 550,00
2	IMU	AltIMU-10 V3	OpenHacks	2	$ 758,00	$ 1.516,00
3	Sensor de corriente	Sensor efecto Hall ACS709	OpenHacks	1	$ 205,00	$ 205,00
4	Adaptador MicroSD	Cod. D000254	OpenHacks	1	$ 307,00	$ 307,00
5	Sensor presion dinámica	MPXV7002	IUA	1	$ 700,00	$ 700,00
6	Convertidor lógico	TXB0108 - convertidor bidireccional lógico	OpenHacks	1	$ 188,00	$ 188,00
7	Receptor GPS	-	IUA	1	$ 900,00	$ 900,00
8	Transceptor Xbee	Módulo RF XBee 802.15.4 series 1 DE 1 Mw	Stock	1	$ 600,00	$ 600,00
9	Componentes discretos	Resistencias, capacitores, diodos, etc	Stock	1	$ 30,00	$ 30,00
10	Placa de propósito general	-	Electrocomponentes	1	$ 100,00	$ 100,00
11	Amp. OP	-	Stock	2	$ 20,00	$ 40,00
12	Batería de LiPo	Bat. LiPo 5000mAh - 11,1V	IUA	1	$ 1.800,00	$ 1.800,00
13	Otros	Cables, conectores y caja de protección	Stock	1	$ 300,00	$ 300,00
14	PCB	Fabricación de PCB prototipo Poncho	Ruben Estevan	3	$ 900,00	$ 2700,00
15	Fuente Switching	Fuente switching de 5v 3A	OpenHacks	1	$ 244,00	$ 244,00

Total	$ 10.180,00

9.6 DIFICULTADES QUE SE HAN PRESENTADO

Al momento de comenzado el desarrollo del proyecto, se decidió utilizar un sistema de control de versiones como se mensiona en el punto 9.2 y "congelar" la versión beta del RTOS que se disponía en ese momento, por lo tanto muchas de las características no se encuentran disponibles o encontraban disponibles al momento del inicio del proyecto.

Es importante destacar que no se disponía de ninguna implementación de los sensores para el RTOS y para esta plataforma.

El desarrollo del proyecto significó un reto desde el punto de vista académico ya que se necesitó estudio e investigación sobre tecnología de la cual se desconocía como protocolos de comunicación, sensores, arquitectura ARM, etc. A su vez, se establecieron tiempos acotados para llevar a cabo cada tarea, lo que puso un punto de complejidad adicional en la gestión del proceso de diseño y desarrollo.

A los puntos anteriores se suma a las dificultadas existentes, la carencia de documentación clara u organizada y en algunos casos inexistentes, sobre el CIAA Firmware. Cabe destacar que la comunidad de desarrolladores se encuentra muy

activa y brinda ayuda al instante ante consultas realizadas en el foro de Embebidos32[20].

10.TRABAJO A FUTURO

A continuación se plantean 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.

[20] https://groups.google.com/forum/#!forum/embebidos32

11.CONCLUSIONES

A lo largo de este trabajo se presentó el desarrollo de un piloto automático para un sistema de paracaída comandado autónomo, para la entrega de cargas en ejercicios militares, situaciones de aislamiento producidas por inundaciones, lucha contra el fuego y ayuda humanitaria, entre otros.

En los capítulos 1 al 8, se expone la problemática de la entrega de carga con paracaídas, desde una introducción teórica hasta el estudio técnico de antecedentes y posible solución. Se analizan los requerimientos y se determina el alcance del proyecto.

En el capitulo 9 se desarrolla la implementación en su totalidad, detallando los cálculos obtenidos, metodología empleada, planificación de las tareas, dificultades en el desarrollo y resultados obtenidos. Para esto, se analizan las variables a obtener y cómo medirlas.

Cabe destacar en este punto, que la elección de la plataforma CIAA fue utilizada porque provee una base tecnológica de Hardware y Firmware de uso profesional capaz de acortar el tiempo de desarrollo y contener la flexibilidad que requiere este proyecto.

En el punto 9.2 se desarrolla la metodología empleada. Como parte de esta, se realizaron diagramas de caso de uso y a partir de estos, se dividió el proyecto en módulos de desarrollo permitiendo el desarrollo en paralelo de tanto de hardware como de firmware. Una vez fijados los módulos, se procede con la asignación de los tiempos y responsabilidades que cada tarea necesitará.

Paralelamente al desarrollo del firmware, se decidió montar un target de desarrollo en una placa experimental multipropósito, de manera de trabajar en simultáneo con el desarrollo de hardware del Poncho. El Poncho desarrollado (llamado CIAAPilot) es la versión de target final a utilizar por el equipo de desarrollo de firmware y en las pruebas iniciales de campo.

Luego se procedió 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 todos los módulos para obtener el sistema final.

En el punto 9.3 se muestra el diagrama de Gantt de proyecto que se obtuvo como resultado de la estimación de tiempos que cada tarea conlleva. Esto fue escencial ya que el desarrollo de un piloto automático aplicado al guiado de un sistema de paracaídas a medida, con diseño de hardware propio y firmware propio, permite tener total control del sistema. Es por esto que, desarrollar la electrónica desde cero, como se hizo en este proyecto, implica un tiempo de ejecución del proceso mucho mayor y si no se cuenta con una gestión del proyecto acorde a los plazos, estos pueden llegar a ser excesivos. Es proyecto se realizó en los tiempos que se planificaron.

En los puntos 9.4.1 y 9.4.2 se muestran los circuitos esquemáticos y el circuito impreso fabricado del Poncho CIAAPilot, corroborándose de esta manera la correcta disposición dimensional de todos los elementos, como zócalos de conexión hacia la EDU-CIAA y borneras de uso externo.

En el punto 9.4.3, los ensayos determinaron que el consumo de corriente del piloto automático sin tener en cuenta al equipo de radio transceptor que realiza la telemetría, fue de 210mA a 7,5V en su peor condición. Esto demuestra que es perfectamente comparable con el consumo de corriente de los piloto automáticos profesionales disponibles en el mercado y acorde a un sistemas de alimentado a baterías.

En el punto 9.4.4 se midió el filtro complementario implementado, demostrando su correcto funcionamiento para conseguir una medición de ángulo corregida proveniente de un acelerómetro y girómetro sin sus respectivos ruidos y deriva. Las mediciones de ángulo aplicando el filtro, suavisando la señal y no presentando deriva, tal como se espera.

En el punto 9.4.5 se realizaron las mediciones para ensayar, caracterizar y poner a punto el sistema. Fue necesario el uso de distintos dispositivos como una bomba de vacío para la simulación de altura y en otro caso se requirió construir específicamente una base móvil con el objeto de corroborar las mediciones de los ángulos de navegación y los errores.

Los resultados obtenidos de los ensayos se detallan en profundidad en el punto 9.4 y a continuación se resumen los características principales del sistema en como función de las pruebas realizadas:

La Tabla 11 contiene las características físicas del sistema electrónico montado. Como se puede apreciar, las dimensiones y peso sin su gabinete es inferior a los sistemas comerciales profesionales.

Tabla 11 - Especificaciones mecánicas

Especificaciones mecánicas	
Ancho	85,80 [mm]
Largo	137,00 [mm]
Alto	40,00 [mm]
Peso	200 [g]

La Tabla 12 contiene las características de potencia consumida y rango de temperaturas de operación. Al ser un sistema de prototipo para ser usado dentro de laboratorio en el desarrollo del firmware, los rangos de temperatura se encuentran acotados . Esto no impide que posteriormente sea montado en un gabinete que permita tener extender este rango de temperatura y ser ensayado en tal sentido.

Tabla 12 - Alimentación y medio ambiente

Alimentación y Medio ambiente	
Temperatura de operación	0º C a 40º C
Fuente de energía	VIN: 8 a 32V
	Potencia: 3W

La Tabla 13 detallan las características generales como entradas y salidas digitales, entradas analógicas y características de sensores.

Tabla 13 - Caraterísticas generales

Generales	
Interfaces RS232	3
Interfaces digitales	Salidas de propositos generales: 5
	Entradas de propositos generales: 8
	I2C: 1
	SPI: 1
	Led de estados: 6
	Switches: 3
Salidas PMW	16
Interfaces analógicas	Entradas ADC: 3
	Resolución: 10bit
Sensor de presión	Barométrica: 26KPa a 126KPa
	Diferencial: -2KPa a 2KPa
Sensores inerciales	Acelerómetro 3 ejes hasta 16g
	Girómetro 3 ejes hasta 2000 dps
	Magnetómetro 3 ejes hasta 12 gauss

En el punto 9.5 se detalla el control de los costos para la construcción del prototipo y a pesar de que no se contaba con este dato como objetivo claro, se decidió mantener un control de estos y minimizar los mismo. Es por esto que el prototipo se pudo finalizar con un costo de $ 10.180,00 en moneda local. Esto implica aproximadamente un 3% del valor del Autopiloto marca Piccolo sin tener en cuenta su estación de control en tierra.

En resumen, puede decirse que este proyecto prueba que mediante el desarrollo de un hardware y firmware propios basados en la tecnología que otorga la plataforma CIAA, el autopiloto diseñado pudo ser implementado en el 10 meses, cumpliendo con la planificación. Las pruebas de campo exeden a este trabajo final, pero se dejan previsto todo lo necesario para su ejecución, de manera de poder ensayar el sistema cuando se dispongan de todos las partes componentes.

12.REFERENCIAS

Aeroquad. (2016). Recuperado el 17 de Junio de 2016, de http://aeroquad.com/showthread.php?6006-AeroQuad-32-Flight-Control-Board

Catsoulis, J. (2005). *Designing Embedded Hardware.* Sebastopol, California: O'Reilly Media.

Cerdeiro, M. (2015). *Introducción a OSEK-OS - El Sistema Operativo del CIAA-Firmware.* Buenos Aires: ACSE. Recuperado el 22 de Junio de 2016

EMLID Limited. (24 de Octubre de 2014). *Navios 2*. Recuperado el 17 de Junio de 2016, de https://docs.emlid.com/navio2/

Massa, M. B. (2006). *Programming Embedded Systems: With C and GNU Development Tools, 2nd Edition.* Sebastopol, California: O'Reilly Media.

Paparazzi UAV wiki. (s.f.). Recuperado el 17 de Junio de 2016, de https://wiki.paparazziuav.org/wiki/Main_Page

Pernia, E. (24 de Abril de 2016). *pinout EDU-CIAA*. Recuperado el 19 de Junio de 2016, de Sitio web de proyecto CIAA: http://www.proyecto-ciaa.com.ar/devwiki/lib/exe/fetch.php?media=desarrollo:edu-ciaa:edu-ciaa-nxp_pinout_a4_v4r2_es.pdf

Stringham, G. (December 1, 2009). *Hardware/Firmware Interface Design: Best Practices for Improving Embedded Systems Development.* Newnes.

13.ANEXO 1 – Planning

14.ANEXO 2 – Circuito esquemático

15.ANEXO 3 – Script Matlab para adquisición de filtro complementario

FiltroComple.m

```
% Captura n muestras de angulos calculado con acelerometro
% girómetro y filtro complementario. Grafica resultado de
% las tres mediciones

clear
numCapturas = 1500;
ejeX = [1:numCapturas];
ang = ciaaCapture(numCapturas);

angFiltro = ang(1:end,1,1);
angFiltro = angFiltro.';
angAcel = ang(1:end,2,1);
angAcel = angAcel.';
angGyro = ang(1:end,3,1);
angGyro = angGyro.';


plot(ejeX,angAcel,'-r','LineWidth',2);
hold on
plot(ejeX,angGyro,'-b','LineWidth',2);
plot(ejeX,angFiltro,'-k','LineWidth',2);
hold off
legenda = legend ('Acelerometro','Girometro','Filtro Complementario');
xlabel('Numero de muestra')
ylabel('Angulo [grados]')
title('Funcionamiento de filtro complementario');
```

ciaaCapture.m

```
function angulos = ciaaCapture (cant_muestras)

%Borra datos anteriores y vuelve a declarar todo

delete(instrfind({'port'},{'/dev/ttyUSB1'}));
ciaaPort=serial('/dev/ttyUSB1');
ciaaPort.BaudRate=115200;

fopen(ciaaPort);%abre el puerto a utilizar
cont = 1;
while cont<=cant_muestras
    captura=fscanf(ciaaPort,'%s'); %Toma el valor recibido por el
puerto y lo guarda en la variable
    angulos(cont,:) = sscanf(captura, '%*4c%3d%*1s%3d%*1s%3d',inf)
    cont = cont +1;
```

```
15  end
16  %cierra y borra el puerto utilizado, borra todas las variables
    utilizadas
17  fclose(ciaaPort);
18  delete(ciaaPort);
19  end
```