Arquitectura multinucleo asimétrica para registrador de variables de vuelo de cohetes experimentales

Larosa, F. S. , Fernández, M, Castelucci Vidal, I. , Mignone, M. N.
Departamento de Ingeniería Electrónica
Universidad Tecnológica Nacional
Facultad Regional Haedo
Email: facundolarosa@gmail.com, martingabrielfernandez@gmail.com, icvidal@frh.utn.edu.ar, mnmignone@gmail.com

***Resumen*—El objetivo de este trabajo es presentar la implementación multinúcleo de un registrador de vuelo de variables provenientes de diferentes sensores (giróscopo, acelerómetro, magnetómetro de tres ejes, barómetro, sensor de temperatura y módulo GPS) para cohetes experimentales. Para ello, se propone la utilización de una arquitectura multinúcleo asimétrica basada en el microcontrolador LPC4337. La utilización de una arquitectura multinúcleo permite distribuir convenientemente las responsabilidades de procesamiento entre los núcleos de la plataforma mejorando la performance del sistema en general. El diseño de software propuesto se basa en técnicas que favorecen la modularidad y la abstracción de forma tal de facilitar la integración a futuro de nuevos sensores y nuevas funcionalidades de procesamiento.**

***Keywords* - Multinúcleo, Asimétrico, Microcontrolador, FreeRTOS, Registrador, Vuelo, Sensores.**

I. Introducción

Como consecuencia de las mejoras en los procesos de fabricación de semiconductores se hace posible la integración de sistemas microcontrolados cada vez más complejos. De esta manera, se han incorporado a los microcontroladores gran variedad de periféricos y, en muchos casos, varios núcleos que pueden utilizarse para distribuir de forma más conveniente la responsabilidad de procesamiento del sistema a diseñar.

Existen dos variantes de arquitecturas: multiprocesamiento simétrico (SMP, del inglés *Symmetric Multi-Processing*) y multiprocesamiento asimétrico (AMP, del inglés *Asymmetric Multi-processing*). En ambas arquitecturas se dispone de dos o más núcleos los cuales pueden o no tener individualmente la misma microarquitectura [1], aunque es más común en el caso de AMP que los núcleos difieran. La diferencia entre ambos enfoques radica en que mientras en el enfoque simétrico se utiliza normalmente un sistema operativo que gestiona los núcleos de forma tal de distribuir las responsabilidades uniformemente, en el enfoque asimétrico se dividen a priori las responsabilidades entre los núcleos de forma distintiva. De esta manera, los núcleos asumen tareas que no son intercambiables, sino que cada uno lleva adelante funcionalidades únicas para el sistema que se desarrolla. Este es el caso, por ejemplo, cuando uno de los núcleos posee extensiones de hardware de uso específico (tal como instrucciones para el procesamiento digital de señales).

En este trabajo, se presenta la implementación de un sistema multinúcleo asimétrico, donde cada procesador posee una arquitectura distinta y realizan funcionalidades no intercambiables. En la sección *Arquitectura multinúcleo del LPC4337* se explican detalles funcionales de la arquitectura del microcontrolador LPC4337. Luego, en la sección *Arquitectura de hardware del registrador* se describe brevemente la plataforma de hardware utilizada. En la sección *Arquitectura de software* se desarrolla como se distribuyen las tareas entre los núcleos y las técnicas utilizadas para intercomunicarlos. Finalmente en las secciones *Verificación y ensayos* y *Resultados* se muestran algunas mediciones realizadas conjuntamente con el procesamiento de las mismas y se plantean los próximos pasos del desarrollo.

II. Arquitectura multinúcleo en el LPC4337

El microcontrolador LPC4337 [2], de la firma NXP, posee como unidades centrales de procesamiento dos núcleos de la familia Cortex M. Por un lado, un Cortex M4 y por otro, un Cortex M0. Ambos núcleos están incluídos dentro del mismo encapsulado junto con una amplia variedad de periféricos integrados (controladores SPI, I2C, UART, CAN, ADC, DAC, etc.) con la intención de posibilitar el desarrollo y ejecución de aplicaciones multinúcleo.

El procesador Cortex M4 es un procesador de 32 bits, diseñado por la empresa ARM el cual posee como características distintivas [3]:

- Arquitectura Harvard: Posee buses separados para memoria de datos e instrucciones, permitiendo el acceso simultáneo a ambos
- Pipeline de tres etapas: *fetch, execute, decode*
- Arquitectura RISC (del inglés, *Reduced Instruction Set Computer*
- Extensiones de hardware para el procesamiento de señales digitales a través de instrucciones SIMD (del inglés *Single Instruction Multiple Data*). Estas instrucciones permiten operar sobre un conjunto de datos vectorialmente, por ejemplo realizando operaciones aritméticas o lógicas sobre un juego de variables a la vez. De esta manera, se reducen los tiempos de procesamiento para operaciones comúnmente utilizadas en sistemas digitales

(tales como la convolución o la correlación por nombrar algunas).

- Unidad de punto flotante: Permite realizar operaciones por hardware sobre tipos de datos flotantes de precisión simple según el estándar IEEE 754. De esta forma, se acelera el procesamiento aritmético de este tipo de datos.

El procesador Cortex M0 es un procesador de 32 bits, diseñado por la empresa ARM para competir en el segmento de precio y prestaciones con procesadores de bajo costo de 8 bits. Posee como características [4]:

- Arquitectura Von Neumann: Posee un único bus para datos e instrucciones
- Pipeline de tres etapas: *fetch, execute, decode*
- Arquitectura RISC
- Optimizado para bajo consumo

De lo anterior, se ponen de manifiesto las diferencias entre ambos núcleos: mientras el M4 está diseñado para proveer funcionalidades de procesamiento intensivo (a través de las extensiones SIMD y la unidad de punto flotante), el M0 tiene una arquitectura de menores prestaciones, pero de menor consumo. Así, cada núcleo es idóneo para realizar determinados tipos de tareas. Se puede plantear el uso de dos esquemas de operación. Por un lado, si se considera como núcleo principal al M4, el M0 podría utilizarse como un controlador de entradas/salidas programable capaz de configurar, interrogar, recibir y empaquetar los datos provenientes de diferentes dispositivos de entrada con el objeto de unificar el flujo de información que recibe el M4. De esta manera, se aligera la carga de responsabilidades del M4, el cual podría enfocarse en el procesamiento de datos. Por el otro, si se toma como núcleo principal al M0, éste podría encargarse del funcionamiento general del sistema, reservando al M4 la función de coprocesador matemático, el cual podría activarse a demanda.

Figura 1. Diagrama en bloques de la arquitectura del LPC4337.

Ambos procesadores comparten la posibilidad de funcionar con un reloj configurable hasta la frecuencia máxima de 204 MHz y cuentan con una interfaz de conexión matricial a numerosos periféricos *on-chip*, como se indica en la figura 1 .

III. Arquitectura de hardware

El sistema diseñado posee diferentes fuentes de datos las cuales reportan magnitudes físicas a partir de los cuales pueden obtenerse datos de interés asociados a la dinámica y cinemática del vuelo. Como plataforma de hardware se utilizó un dispositivo diseñado y fabricado por este grupo de trabajo [5] el cual comprende de un microcontrolador NXP LPC 4337, memoria de datos SD extraíble, fuente de alimentación y las siguientes unidades de mediciones:

- Unidad de medidas inerciales: compuesta por un acelerómetro y un giróscopo en tres ejes
- Magnetómetro en tres ejes
- Receptor GPS
- Altímetro: permite obtener la altura del vehículo a partir de la presión relativa y la temperatura

El sistema es capaz de adquirir los datos provenientes de las diferentes fuentes a una tasa de muestreo configurable (entre 100Hz y 1000Hz), operar sobre los mismos (realizar ajustes de escala, conversión, filtrado de datos entre otras), organizarlos y empaquetarlos para su posterior almacenamiento en la memoria interna. Además, la memoria interna puede accederse a través de una interfaz serie de forma tal que el sistema podría integrarse dentro de un cuerpo de resina rígida para brindarle mayor rigidez estructural y proteccion frente a factores ambientales.

En la figura 2, se muestra un diagrama en bloques del sistema y en la figura 3 se observa una fotografía del dispositivo.

IV. Arquitectura de software

IV-A. Arquitectura de software general

Respecto de la implementación de software, se utilizaron dos instancias del sistema operativo embebido FreeRTOS [6] que corren separadamente en cada núcleo. En la figura 4, se muestra de forma esquemática la separación de responsabilidades entre cada uno de los núcleos.

La adquisición de las mediciones provenientes de todas las fuentes de datos fue asignada al M0, debido a que estas tareas si bien comprenden una gran cantidad de microoperaciones conllevan relativamente poco procesamiento de datos. El M0 se encarga de dar formato y empaquetar todas las mediciones en una única estructura de manera tal de entregar al M4 los datos para un eventual procesamiento y registro en la memoria externa.

IV-B. Arquitectura de software M0

Tal como se indicó en el apartado anterior, el M0 tiene como responsabilidad principal la recolección, formateo y empaquetamiento de datos que luego serán enviados al M4 por medio

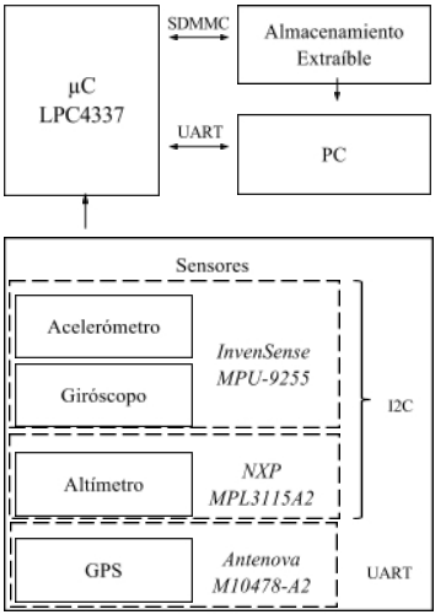

Figura 2. Diagrama en bloques del sistema utilizado

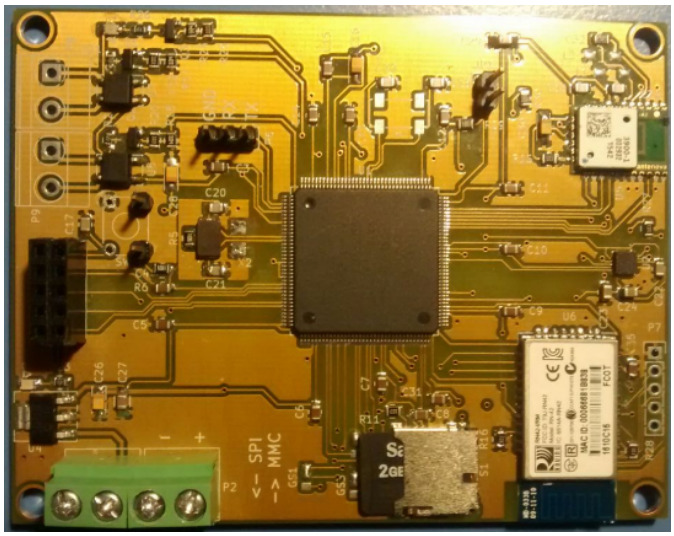

Figura 3. Fotografía del dispositivo de registro de datos

del mecanismo de comunicación entre procesos diseñado al efecto (IPC, del inglés *InterProcess Communication*). En la figura 5 se muestra un esquema en bloques de las diferentes tareas implementadas del sistema operativo y su interrelación.

Las tareas *UartTask* e *I2CTask* son tareas de menor nivel cuya función es interactuar directamente con el hardware, abstrayendo a las tareas de mayor jerarquía, según se indica en la figura 5. *UartTask* está relacionada con uno de los puertos de comunicación serie (UART, del inglés *Universal Asynchronous Receiver/Transmitter*) a través de interrupciones. Implementa dos colas de datos (una para la transmisión y otra para la recepción) las cuales centralizan todo el tráfico de datos del periférico y pueden ser accedidas por las otras tareas del sistema.

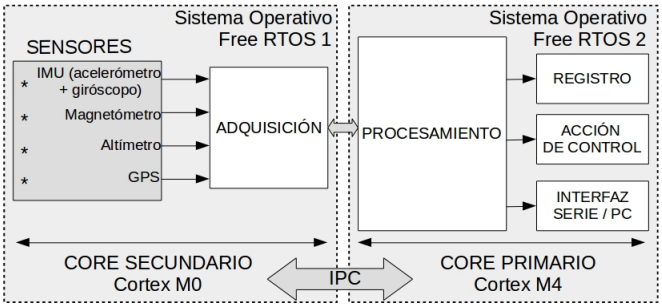

Figura 4. Diagrama en bloques del firmware.

Las tareas *I2CTask* se comunican con las interrupciones de los periféricos I2C y son procesos de bajo nivel que abstraen a las tareas de orden superior de las particularidades del hardware, dando por resultado las mediciones finales. Dado que tanto la unidad de medidas inerciales (IMU, del inglés *Inertial Measurement Unit*) como el barómetro comparten interfaces I2C, se requieren dos instancias de la misma tarea (*I2CTask[0]* e *I2CTask[1]*). La tarea *GPS* tiene por función interpretar los datos compartidos por la tarea *UartTask*, obteniendo las variables de interés a partir de la trama NMEA (del inglés *National Marine Electronics Association*) reportada por el receptor GPS para luego almacenarlos en una estructura de datos de tamaño fijo. Estos datos se incorporan a una única estructura de datos que integra las mediciones provenientes del resto de los sensores (IMU, magnetómetro y altímetro). La tarea *GPS Data Ready* verifica si existen nuevos datos provenientes del receptor GPS y despierta a la tarea *Empaquetador*. Ésta se ejecuta periódicamente cada 1 s y su función es recolectar los datos de los distintos sensores y almacenarlos en una cola binaria. Además, calcula una suma de verificación (*checksum*) que se utiliza para evaluar la integridad de los datos. Se implementaron dos colas binarias intercambiables, de forma tal que en todo momento una de ellas se utiliza para la escritura de datos (cola activa) y al completarse cambia de estado (cola lista) pasando al estado de 'cola activa' la de reserva. En esta circunstancia, el M0 envía un mensaje al M4, conjuntamente con un puntero a la 'cola lista'. Posteriormente, el M4 se encargará de procesar esta información y el proceso vuelve a repetirse indefinidamente, evitando por este mecanismo la concurrencia en el acceso a datos. Finalmente, la tarea *IPCTask* tiene por objeto manejar los mensajes IPC entre los núcleos operando a través del uso de las interrupciones cruzadas tal como se describirá en la subsección *Mecanismo de comunicación internúcleo*. En particular, en el M0, esta tarea se ocupa de enviar el mensaje de que una cola está lista para ser procesada por el M4.

IV-C. Arquitectura de software M4

El objeto de la arquitectura propuesta es aliviar la carga de procesamiento del M4 (núcleo con extensiones para procesamiento de señales), traspasando el mayor número de tareas

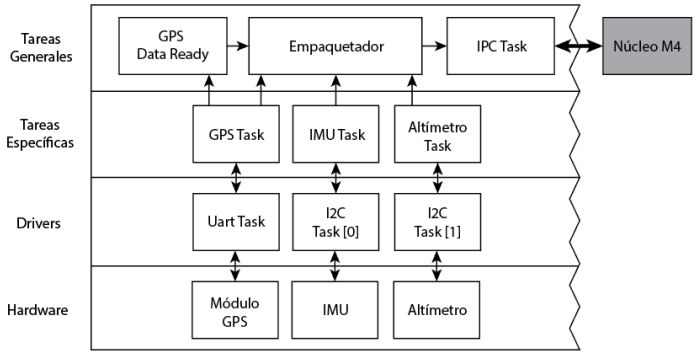

Figura 5. Diagrama en bloques de la arquitectura de software del M0

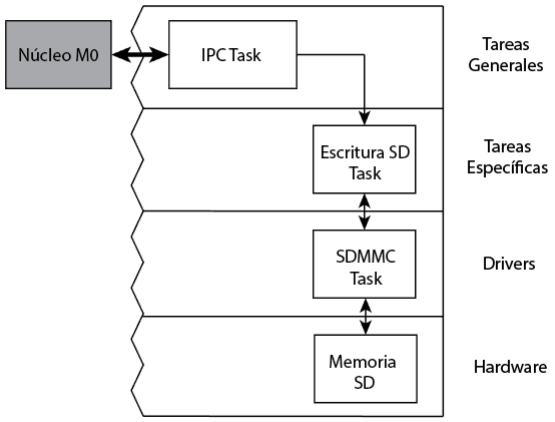

Figura 6. Diagrama en bloques de la arquitectura de software del M4

posibles al M0 (núcleo optimizado para menor consumo) con vistas a realizar procesamiento en tiempo real de los datos en el M4. Por ello, en el M4 sólo se implementan una pequeña cantidad de tareas. En la figura 6 se muestra un esquema en bloques de las diferentes tareas implementadas del sistema operativo y su interrelación.

La tarea *IPCTask* es una tarea espejo de la tarea homónima descrita para el núcleo M0. Se encarga de manejar los mensajes IPC enviados por el núcleo M0 e interpretarlos a partir de su código de operación (*opcode*). Particularmente, esta tarea recibe el mensaje enviado por el M0 indicando que hay una cola lista para procesar y un puntero a la misma.

La tarea *Escritura SD* recibe el puntero y la longitud de datos a almacenar e interacciona con la tarea *SDMMC Task*. Esta última se encarga de escribir la memoria SD utilizando la interfaz SDMMC (del inglés, *Secure Digital/MultiMedia Card*) del microcontrolador para bajar las estructuras de datos en forma de archivo binario en formato FAT [7].

Eventualmente, en este núcleo se implementarán los algoritmos de procesamiento de datos que permitirán filtrar las señales de entrada, reconstruir la trayectoria del vehículo y generar acciones de control según lo que se indica en la sección *Verificación y ensayos*.

IV-D. Mecanismo de comunicación internúcleo

En el NXP LPC 4337 los núcleos están relacionados por medio de dos fuentes de interrupción cruzadas tal como se muestra en la figura 7. Cuando alguno de los núcleos desea realizar alguna operación, pedido o señal sobre el otro activa esta interrupción a través de una instrucción específica (denominada SEV), es decir, por software. El otro núcleo captura la interrupción a través de una rutina dedicada de la misma manera que gestiona las fuentes de interrupción por hardware. La implementación del mecanismo de comunicación es responsabilidad del programador del sistema a través del uso de una sección de memoria compartida.

El LPC 4337 posee varios bancos de memoria RAM los cuales pueden ser utilizados indistintamente por cada núcleo. En el presente trabajo se decidió distribuir los bancos en tres grupos, como se indica en la figura 8. Un banco de

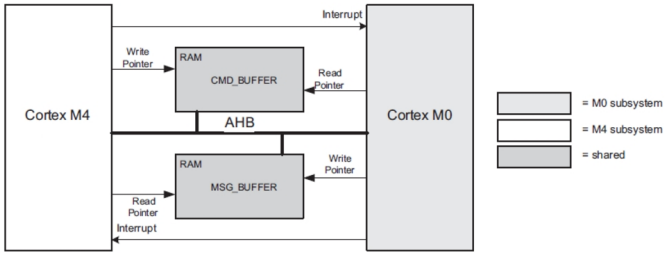

Figura 7. Mecanismo de comunicación por hardware

RAM de 32 Kb (RAMLOC32) se destinó exclusivamente al núcleo M0, un banco de RAM de 40 Kb (RAMLOC40) se destinó exclusivamente al núcleo M4, mientras que los bancos restantes (RAMAHB32, RAMAHB16 y RAMAHB_ETB16) se asignaron de forma tal que puedan compartirse entre ambos núcleos. Dentro de la sección de 16 Kb denominada 'RAMAHB16' se definieron dos secciones de memoria donde se alojarán sendas colas destinadas a comunicar los núcleos. Una de ellas sirve de *buffer* para los datos generados por el M0 destinados al M4 y viceversa.

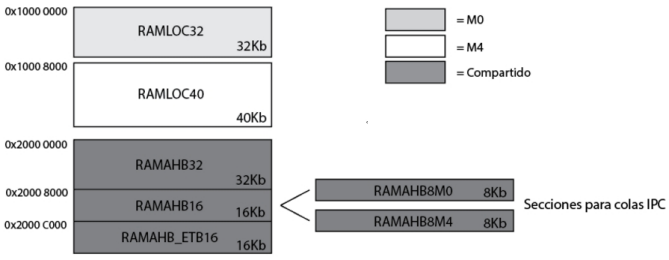

Figura 8. Distribución de los bancos de memoria

La estructura de los mensajes se muestra en la figura 9. Cada mensaje está formado por un encabezado el cual consta de información sobre el identificador del proceso que lo emite (*processid_t pid*) y el núcleo en el cual se encuentra (*coreid_t*

cid) y el comando que desea enviar indicando: código de operación (*opcode_t opcode*) y una serie de operandos (*operand_t operand[N]*).

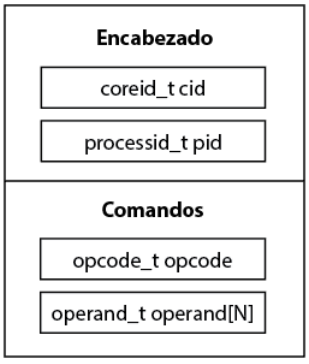

Figura 9. Estructura del mensaje internúcleo

En el destino, la tarea *IPCTask* se encarga de interpretar el código de operación y activar la funcionalidad predefinida de acuerdo a los operandos recibidos.

V. Verificación y ensayos

Con el objeto de verificar el correcto funcionamiento del sistema se procedió a recrear trayectorias predefinidas para luego comparar las mediciones obtenidas con las esperadas. En la figura 10 se muestra el resultado obtenido para una de esas mediciones.

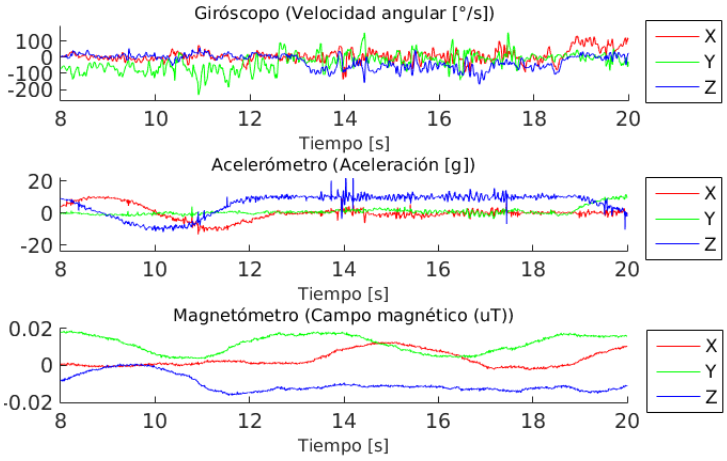

Figura 10. Mediciones del giróscopo, acelerómetro y magnetómetro para una trayectoria conocida

Posteriormente, se utilizó el algoritmo de Madgwick [8] para reconstruir en base a las mediciones tomadas la actitud del dispositivo, lo cual pudo realizarse correctamente. En la figura 11 se muestra la actitud para los diferentes ángulos de navegación del sistema (comúnmente denominados en inglés: *yaw*, *pitch*, *roll*) para la medición indicada en la figura 10. Las discontinuidades en los ángulos obedecen a que los ángulos están limitados en el intervalo $[-180°, +180°]$.

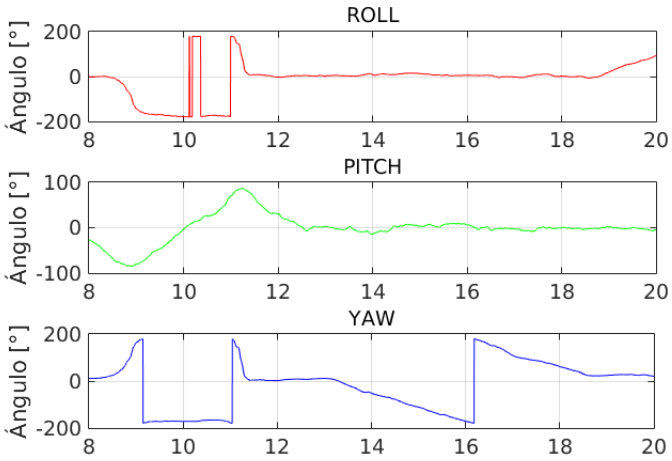

Figura 11. Reconstrucción de la actitud del dispositivo (*yaw*, *pitch*, *roll*) por medio del algoritmo de Madgwick

VI. Conclusiones

Se diseñó e implementó un registrador de variables para cohetes experimentales utilizando un esquema de software multinúcleo. Se cumplió el objetivo de distribuir las responsabilidades del sistema con el objeto de aliviar al núcleo principal (M4) e implementar a futuro algoritmos que permitan reconstruir la actitud, la trayectoria y generar eventualmente acciones de control.

Referencias

[1] C. Walls, "Asymmetric Multi-Processing (AMP) vs. Symmetric Multi-Processing (SMP)" Available: http://scitechconnect.elsevier.com/asymmetric-multi-processing-amp-vs-symmetric-multi-processing-smp/

[2] NXP, "LPC43xx/LPC43Sxx ARM Cortex-M4/M0 multi-core microcontroller - User Manual" Available: http://www.nxp.com/docs/en/user-guide/UM10503.pdf

[3] J. Yiu, "Technical Overview" in *The Definitive Guide to ARM Cortex-M3 and Cortex-M4 Processors*, Newnes, 2013, ch. 3, pp. 57-73

[4] J. Yiu, "Technical Overview" in *The Definitive Guide to ARM Cortex-M0 and Cortex-M0+ Processors*, Newnes, 2015, ch. 2, pp. 27-53

[5] F. Larosa, M. Mignone, I. Castellucci, M. Fernández, "Dispositivo de adquisición y registro de datos para cohetes experimentales", Congreso Argentino de Tecnología Espacial (CATE) 2017

[6] Real Time Engineers Ltd., "The FreeRTOS Reference Manual 9.0". Available: https://goo.gl/kZLAtx

[7] Microsoft Corp., "Microsoft Extensible Firmware Initiative FAT32 File System Specification "Available: https://staff.washington.edu/dittrich/misc/fatgen103.pdf

[8] S. Madgwick, "An efficient orientation filter for inertial and inertial/magnetic sensor arrays", Available: https://www.samba.org/tridge/UAV/madgwick_internal_report.pdf