CEA Comité Español de Automática
XLVII JORNADAS DE AUTOMÁTICA CÓRDOBA 2026
UNIVERSIDADE DA CORUÑA

Jornadas de Automática

# Arquitectura y modelo de ejecución para control de un cuadricóptero

Fontalba, M.[a,*], Ortiz, L.[a], Guasque, A.[a], Simó, J.[a]

[a] *Instituto de Automática e Informática Industrial, Universitat Politècnica de València, Camino de Vera, s/n, 46022, Valencia, España.*

**To cite this article:** Fontalba, M, Ortiz, L, Guasque, A, Simó, J. 2026. Architecture and execution model for quad-copter control. Jornadas de Automática, 47. https://doi.org/10.17979/ja-cea.2026.47.13790

**Resumen**

En este trabajo se propone un modelo temporal para la ejecución de la carga computacional de un vehículo aéreo no tripulado (VANT) de tipo cuadricóptero, considerado como un sistema de tiempo real particionado de criticidad mixta sobre una plataforma multiprocesador. El análisis se lleva a cabo mediante diagramas *End-to-End Flows* (ETEF), lo que permite caracterizar la interacción entre tareas y modelar el flujo de información de los bucles funcionales con un alto nivel de abstracción. El sistema considerado se estructura mediante particiones software que agrupan funcionalidades específicas y sus tareas asociadas, siguiendo principios de arquitecturas particionadas inspiradas en enfoques IMA y estándares como ARINC 653. El enfoque propuesto evita el análisis exhaustivo a nivel de tarea individual, centrándose en la representación de relaciones de precedencia y comunicación mediante flujos temporales. Como resultado, se obtiene una descripción estructurada del comportamiento del sistema basada en dependencias entre tareas y en la organización de los distintos bucles funcionales. Este modelo proporciona una base sólida para el análisis temporal posterior del sistema.

*Palabras clave:* Vehículos autónomos, Aviónica y equipos embarcados, Sistemas ciberfísicos, Arquitectura de software de control, Robótica aérea y espacial

**Architecture and execution model for quadcopter control**

**Abstract**

This paper proposes a temporal model for executing the computational load of a quadcopter-type unmanned aerial vehicle (UAV), treated as a partitioned real-time system of mixed criticality running on a multiprocessor platform. The analysis is carried out using *End-to-End Flows* (ETEF) diagrams, which allow the interaction between tasks to be characterised and the information flow of the functional loops to be modelled at a high level of abstraction. The system under consideration is structured using software partitions that group specific functionalities and their associated tasks, following principles of partitioned architectures inspired by IMA approaches and standards such as ARINC 653. The proposed approach avoids exhaustive analysis at the individual task level, focusing instead on the representation of precedence and communication relationships through temporal flows. As a result, a structured description of the system's behaviour is obtained, based on dependencies between tasks and the organisation of the various functional loops. This model provides a solid basis for subsequent temporal analysis of the system.

*Keywords:* Autonomous vehicles, Avionics and on-board equipments, Cyber physical systems, Control software architecture, Aerial and space robotics.

## 1. Introducción

Los sistemas de tiempo real son aquellos cuyo correcto funcionamiento está determinado no únicamente por el resultado lógico de sus operaciones, sino también por el cumplimiento de una serie de plazos temporales definidos (Davis and Burns, 2011). El incumplimiento de dichos plazos comprome-

*Autor para correspondencia: mfonroc@ai2.upv.es

te la validez de la salida del sistema, pudiendo ocasionar desde una degradación de esta hasta la pérdida completa de su valor, o incluso poner en peligro la integridad del propio sistema. En función de la criticidad de la tarea, las consecuencias del incumplimiento son mayores, pudiendo llegar a provocar, en los casos más críticos, grandes pérdidas materiales o de vidas humanas. Sin embargo, es habitual encontrar sistemas que combinan más de un nivel de criticidad, dando lugar a los denominados sistemas de criticidad mixta (Burns and Davis, 2026).

También encontramos los sistemas monoprocesador y multiprocesador, clasificados según el número de procesadores que utilizan, siendo estos últimos aquellos que presentan varias unidades de procesamiento, ya sean homogéneas o heterogéneas. Estas se caracterizan por su mayor potencia de cómputo al tener un mayor número de procesadores; sin embargo, su análisis e implementación presentan una mayor complejidad. Por otro lado, los sistemas particionados son aquellos que dividen la carga computacional en diversas particiones sobre un mismo hardware compartido. Para llevar a cabo esta división se hace uso de hipervisores, con el objetivo de gestionar la ejecución de las distintas particiones. Este tipo de sistemas se caracteriza por su robustez, derivada del aislamiento espacial y temporal de cada una de sus unidades lógicas, lo que evita la propagación de errores. Esta tecnología ha tenido un gran impacto en la evolución de los sistemas aeronáuticos, dando lugar al paradigma *Integrated Modular Avionics* (IMA), en el que la virtualización es un elemento clave que ha permitido facilitar procesos como la certificación del software, así como la optimización de recursos hardware.

Por otro lado, los vehículos aéreos no tripulados (VANT) multirrotor de tipo cuadricóptero han experimentado un notable auge en las últimas décadas, impulsado tanto por su dinámica relativamente simple como por el abaratamiento de componentes propiciado por el desarrollo de la electrónica en áreas como la sensorización y la actuación. Esto los convierte en vehículos de bajo coste, fácilmente controlables y con una gran variedad de campos de aplicación gracias a su versatilidad. Además, por sus características, se enmarcan en la clasificación de sistemas ciberfísicos (Simó et al., 2021), dado que integran los tres elementos definitorios de este tipo de sistemas: la comunicación con la estación de tierra, la interacción con el entorno físico mediante sus actuadores y el cómputo embebido para el procesamiento de la información sensorial, requisitos de tiempo real relacionados con sus bucles de control.

El desarrollo de modelos de cuadricópteros se ha centrado principalmente en el estudio de su dinámica y cinética, como ilustra el trabajo presentado en (Wang et al., 2016). Sin embargo, el modelado temporal y el análisis de su ejecución no han alcanzado el mismo nivel de desarrollo. En este contexto, resulta de especial interés la formulación de un modelo temporal que permita abordar estos aspectos de forma más estructurada.

Los cuadricópteros reúnen de forma natural las características propias de los sistemas particionados de tiempo real: presentan restricciones temporales estrictas en sus lazos de control, integran una amplia variedad de subsistemas con funcionalidades diferenciadas que interactúan entre sí y agrupan tareas de distinta criticidad, lo que los sitúa dentro del ámbito de los sistemas de criticidad mixta. Todo ello los convierte en un caso de estudio especialmente representativo para el análisis de su comportamiento temporal mediante diagramas *End-to-End Flow* (ETEF), los cuales permiten abstraer y representar las relaciones temporales existentes entre las distintas tareas del sistema.

El resto del artículo se estructura de la siguiente manera: en primer lugar, se describe la carga computacional objeto de estudio, así como su estructuración en particiones; a continuación, se presentan las tareas identificadas junto con sus características temporales; y, finalmente, se realiza el análisis mediante diagramas ETEF.

## 2. Descripción del sistema

El cuadricóptero objeto de estudio se ha modelado como un sistema particionado, en el que cada partición representa un subsistema funcional de la aeronave. La definición de estas particiones se ha realizado atendiendo a cuatro criterios principales: la criticidad de las tareas asociadas a cada subsistema, los requisitos específicos de hardware —incluyendo los buses de comunicación necesarios para su funcionamiento—, la cohesión funcional entre las tareas que lo componen y la necesidad de aislar posibles fallos para limitar su propagación al resto del sistema. En particular, la criticidad se ha clasificado mediante cuatro niveles, de A a D, donde las tareas de nivel A son las más críticas y aquellas cuyo fallo puede comprometer en mayor medida la integridad y el funcionamiento seguro del sistema, mientras que los niveles inferiores representan tareas con un impacto progresivamente menor. La aplicación de estos criterios ha permitido obtener una arquitectura software modular y bien delimitada, cuya organización se muestra en la Figura 1.

Debido al aislamiento estricto entre particiones, la comunicación entre subsistemas se realiza únicamente mediante los mecanismos del hipervisor, siguiendo el estándar ARINC 653. Se utilizan *sampling ports* para los subsistemas del bucle de control y *queuing ports* para el intercambio de eventos e información entre el resto de particiones.

- Subsistema de navegación inercial (**INS**): Encargado de acceder y manejar los sensores, además de estimar la posición de la aeronave mediante la aplicación de filtros sobre los valores leídos.

- Subsistema de evitación de colisiones (**CAS**): Encargado de generar las referencias para el resto de controladores en caso de detectar obstáculos. Mediante las mediciones del LIDAR y algoritmos de comportamiento reactivo, determina cuando es necesario modificar la alimentación de referencias en el FNS Y FCS.

- Subsistema de navegación de vuelo (**FNS**): Encargado del seguimiento de trayectorias preestablecidas en una lista de puntos de camino. Además, recibe comandos del GCS a través del OBDH para modificar la ruta en mitad del vuelo.

- Subsistema de control de vuelo (**FCS**): Encargado de asegurar la sustentación de la aeronave y garantizar los movimientos solicitados por el FNS.

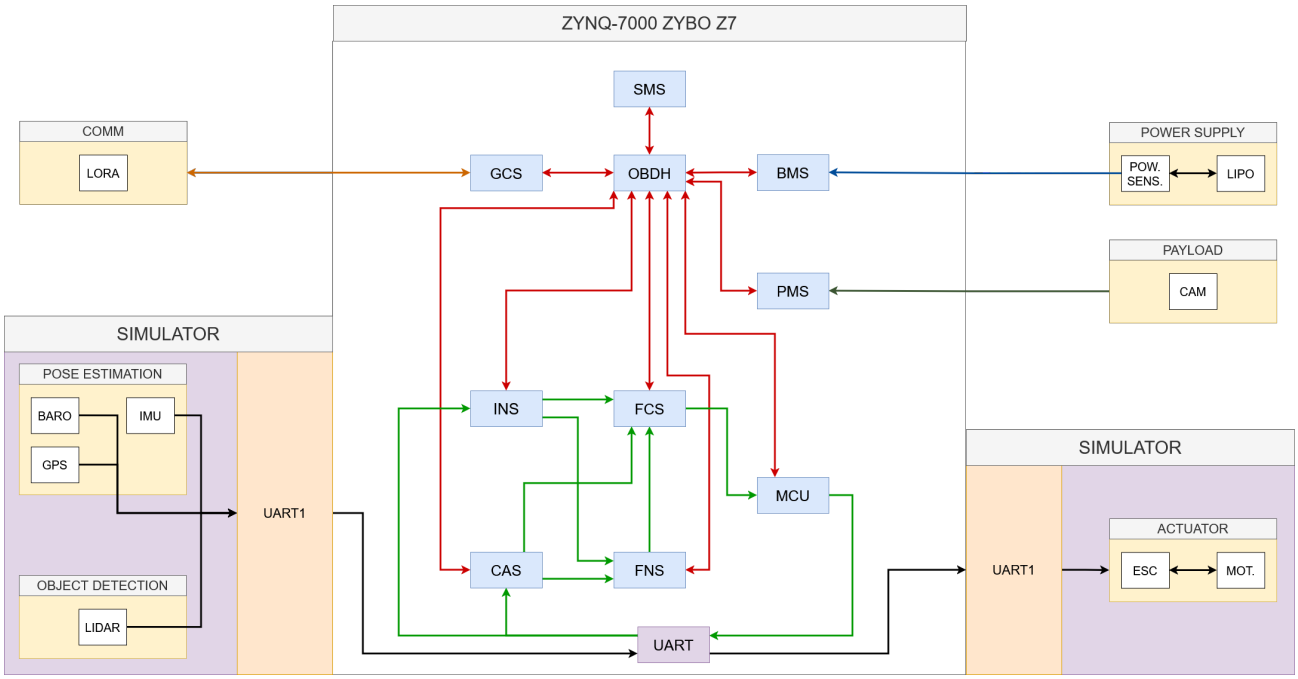


Figura 1: Arquitectura software del cuadricóptero.

- Subsistema de control de motores (**MCS**): Encargado de comunicarse con los controladores electrónicos de velocidad (ESC).
- Subsistema de gestión de baterías (**BMS**): Encargado de monitorizar y alertar al SMS del estado de la batería.
- Subsistema de gestión de carga útil (**PMS**): Encargado de gestionar el cumplimiento de la misión, maneja la cámara y emplea algoritmos sobre las imágenes para dar avisos al sistema respecto al objetivo de la misión.
- Subsistema de comunicaciones con tierra (**GCS**): Encargado de comunicar mediante LORA la aeronave con la estación de tierra. Envía información del estado de la aeronave y recibe comandos desde la base.
- Subsistema de gestión de datos a bordo (OBDH): Encargado de manejar la distribución de información por el resto de sistemas de la aeronave.
- Subsistema de supervisión (**SMS**): Encargado de manejar todo el sistema, organiza la ejecución, cambios de modo, estrategias de recuperación y mantenimiento de un registro de vuelo.

## 3. Modelo de tareas

Tras definir la arquitectura del software, se han extraído las tareas a ejecutar en el sistema, siguiendo el criterio de descomponer la funcionalidad de cada subsistema en operaciones concretas y cohesivas. Por ejemplo, en el subsistema INS se han diferenciado tareas como el acceso al bus para la consulta del acelerómetro, el giróscopo y el magnetómetro, el filtrado de dichas medidas y el envío de los datos a otros subsistemas a través de los puertos de comunicación entre particiones.

Las medidas de cada tarea se han obtenido mediante pruebas experimentales sobre un SoC Xilinx ZYBO Z7-10, equipado con un procesador ARM Cortex-A9 de doble núcleo a 667 MHz y una FPGA Artix-7 integrada, bajo la ejecución del hipervisor de tipo 1 Xtratum, desarrollado en (Masmano et al., 2009),(Poggi et al., 2018), (Wessman et al., 2021). Asimismo, la caracterización temporal se ha realizado empleando sensores y actuadores reales integrados en la plataforma experimental, con el objetivo de obtener medidas representativas del comportamiento del sistema en condiciones de operación reales.

Las tareas identificadas y caracterizadas experimentalmente se han modelado empleando una extensión del modelo temporal propuesto en (Liu and Layland, 1973). La caracterización experimental se ha llevado a cabo mediante la ejecución del código sobre una plataforma hardware y la medición de los tiempos de ejecución e interacción con sensores y actuadores reales. Las tareas $\tau_i$ se definen mediante el siguiente conjunto de parámetros:

$$\tau_i = \{C_i, D_i, T_i, Cr_i, P_i\}$$

donde $C_i$ representa el peor tiempo de ejecución (*Worst-Case Execution Time*, WCET), $D_i$ el plazo de la tarea, $T_i$ el período de activación, $Cr_i$ la criticidad, $P_i$ la partición a la que pertenece la tarea.

La Tabla 1 recoge las 70 tareas identificadas, mostrando los parámetros asociados a cada una de ellas junto con un identificador del subsistema correspondiente. Las tareas cuyo período y plazo relativo aparecen indicados mediante un guion corresponden a tareas de inicialización; por tanto, únicamente se ejecutan una vez durante el modo de inicialización del sistema y no forman parte de la ejecución periódica en modo nominal.

Tabla 1: Tabla de tareas en tiempo real con su ID (partición y número de tarea), WCET, período y plazo.

| **id** | WCET(us) | Periodo(ms) | Deadline(ms) | Crit | **id** | WCET(us) | Periodo(ms) | Deadline(ms) | Crit |
|---|---|---|---|---|---|---|---|---|---|
| INS1 | 6880 | - | - | A | MCS1 | 1156 | - | - | A |
| INS2 | 2940 | - | - | A | MCS2 | 11000 | - | - | A |
| INS3 | 2820 | - | - | A | MCS3 | 121 | 20 | 20 | A |
| INS4 | 850 | 20 | 20 | A | MCS4 | 132 | 20 | 20 | A |
| INS5 | 875 | 20 | 20 | A | MCS5 | 43 | 20 | 20 | B |
| INS6 | 1130 | 20 | 20 | A | BMS1 | 2254 | - | - | A |
| INS7 | 1060 | 20 | 20 | A | BMS2 | 2120 | 200 | 200 | A |
| INS8 | 2130 | 1000 | 1000 | A | BMS3 | 30 | 200 | 200 | B |
| INS9 | 510 | 1000 | 1000 | A | BMS4 | 43 | 200 | 200 | A |
| INS10 | 215 | 20 | 20 | A | GCS1 | 1248 | - | - | A |
| INS11 | 23 | 20 | 20 | A | GCS2 | 142 | 40 | 40 | A |
| INS12 | 42 | 20 | 20 | A | GCS3 | 46 | 40 | 40 | A |
| INS13 | 43 | 20 | 20 | A | GCS4 | 1412 | 40 | 40 | A |
| INS14 | 42 | 20 | 20 | A | GCS5 | 142 | 40 | 40 | B |
| INS15 | 43 | 20 | 20 | A | GCS6 | 1174 | 40 | 40 | B |
| CAS1 | 1250 | - | - | A | GCS7 | 238 | 40 | 40 | B |
| CAS2 | 380 | 20 | 20 | A | GCS8 | 48 | 40 | 40 | B |
| CAS3 | 480 | 20 | 20 | A | PMS1 | 3520 | - | - | C |
| CAS4 | 33 | 20 | 20 | A | PMS2 | 30000 | 2000 | 2000 | C |
| CAS5 | 45 | 20 | 20 | A | PMS3 | 20000 | 2000 | 2000 | C |
| CAS6 | 45 | 20 | 20 | B | PMS4 | 1480000 | 2000 | 2000 | C |
| FNS1 | 1230 | - | - | B | PMS5 | 42 | 2000 | 2000 | D |
| FNS2 | 52 | 200 | 200 | B | PMS6 | 43 | 2000 | 2000 | C |
| FNS3 | 53 | 200 | 200 | B | OBDH1 | 2246 | - | - | A |
| FNS4 | 145 | 200 | 200 | B | OBDH2 | 198 | - | - | C |
| FNS5 | 22 | 200 | 200 | B | OBDH3 | 310 | 20 | 20 | C |
| FNS6 | 43 | 200 | 200 | B | OBDH4 | 146 | 200 | 200 | C |
| FNS7 | 43 | 200 | 200 | B | OBDH5 | 133 | 40 | 40 | D |
| FCS1 | 670 | - | - | A | OBDH6 | 170 | 200 | 200 | C |
| FCS2 | 110 | 20 | 20 | A | OBDH7 | 154 | 200 | 200 | A |
| FCS3 | 47 | 20 | 20 | A | OBDH8 | 3700 | 20 | 20 | A |
| FCS4 | 52 | 20 | 20 | A | SMS1 | 1470 | - | - | A |
| FCS5 | 430 | 20 | 20 | A | SMS2 | 43 | 40 | 40 | A |
| FCS6 | 42 | 20 | 20 | A | SMS3 | 310 | 40 | 40 | A |
| FCS7 | 43 | 20 | 20 | B | SMS4 | 148 | 40 | 40 | C |

## 4. Modelo temporal

El modelo temporal se ha construido a partir de las tareas identificadas en el apartado anterior y describe el comportamiento del cuadricóptero en condiciones nominales de operación, asumiendo que la fase de inicialización se ha completado correctamente y que todos los subsistemas se encuentran plenamente operativos.

Para analizar las relaciones temporales y de dependencia entre tareas se han empleado diagramas de flujo *End-to-End* (ETEF). Estos diagramas permiten representar el flujo de ejecución del sistema mediante relaciones de precedencia entre tareas, de forma que una tarea no puede iniciar su ejecución hasta que todas aquellas de las que depende hayan finalizado la suya. Asimismo, tareas independientes pueden ejecutarse sin restricciones de orden entre ellas.

El empleo de diagramas ETEF facilita la identificación de las cadenas críticas de ejecución y el análisis de las propiedades temporales del sistema, proporcionando una representación abstracta y estructurada del comportamiento temporal del cuadricóptero. El ETEF general del sistema se muestra en la Figura 2. Debido a su representatividad, se presenta un desarrollo detallado de los principales bucles de control.

### *4.1. Bucle de control de estabilidad*

Dentro de los distintos bucles que componen el modelo temporal, el de mayor criticidad es el de control de estabilidad, que integra tareas de los subsistemas INS, FCS y MCS. Este bucle abarca desde la adquisición y procesamiento de los datos del INS hasta el control de los motores por parte del MCS, pasando por el cálculo de la acción de control en el FCS. Este bucle es el más exigente del sistema desde dos perspectivas independientes: por un lado, es el responsable directo de la sustentación de la aeronave, lo que le confiere la mayor criticidad; por otro, debe ejecutarse a la frecuencia de control más alta del sistema, imponiendo las restricciones temporales más estrictas.

El bucle se inicia en la partición INS con dos ramas principales. La primera, correspondiente a la adquisición de datos

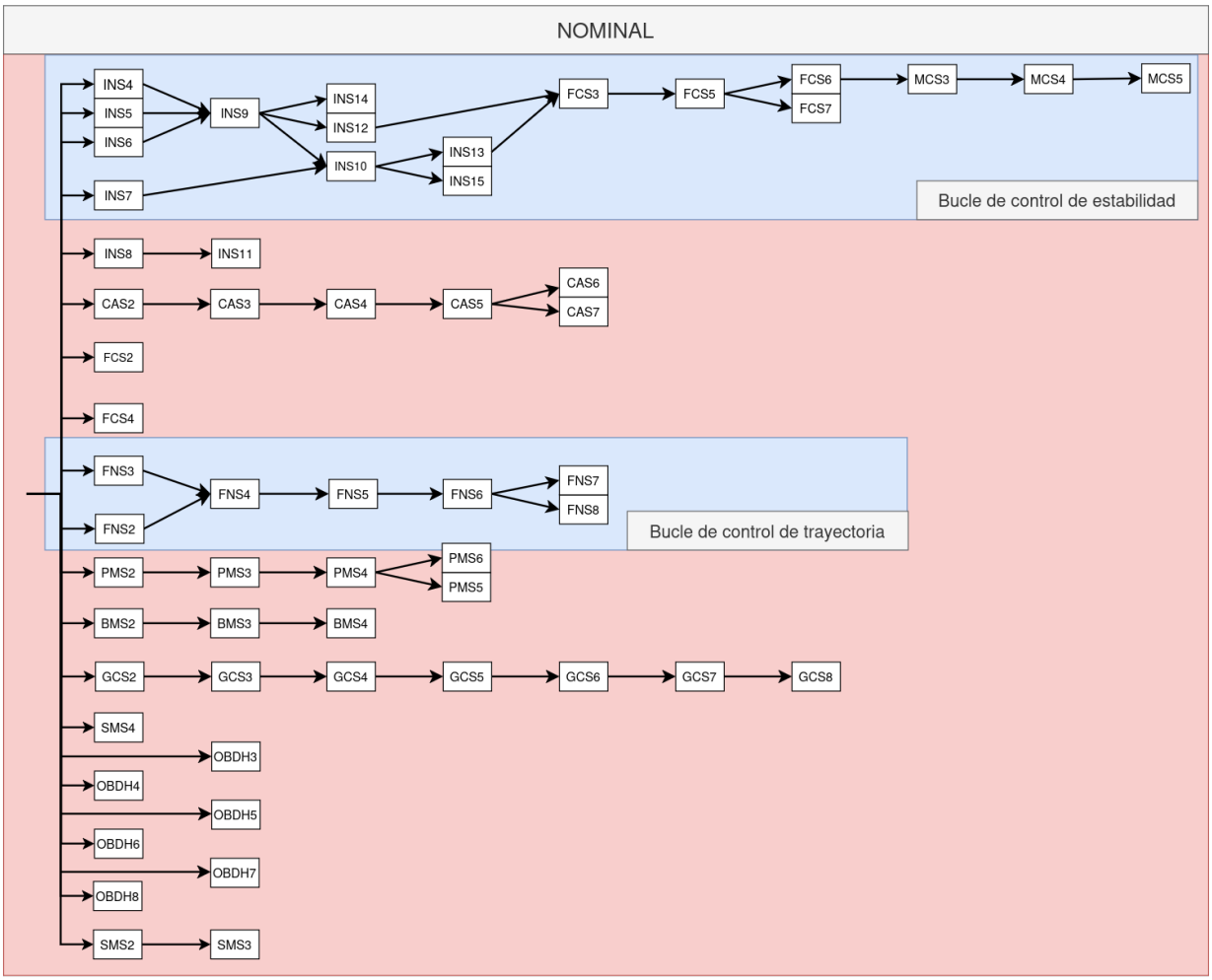


Figura 2: Arquitectura software del cuadricóptero.

de la IMU, se compone a su vez de tres ramas paralelas sin precedencia entre ellas: INS4, INS5 e INS6, encargadas de la lectura del acelerómetro, giroscopio y magnetómetro respectivamente, que convergen en INS9 para estimar la posición angular mediante un filtro. La segunda, correspondiente a la posición traslacional, se compone de una rama paralela a la anterior, inicia en INS7, encargada de la lectura del barómetro, que convergen con INS9 en INS10 para estimar la posición XYZ. Tanto INS9 como INS10 finalizan con las tareas de comunicación hacia la partición FCS: INS14 transmite la posición angular e INS13 la posición XYZ.

La siguiente fase del bucle corresponde al cálculo de la acción de control en la partición FCS. La tarea FCS3 lee los datos recibidos de ambas ramas de la partición INS, constituyendo el punto de convergencia del bucle. A partir de esta información, FCS5 calcula la acción de control, que es posteriormente transmitida a la partición MCS mediante FCS6 y FCS7. En la última fase, la partición MCS recibe la acción de control a través de MCS3, actualiza los valores de las señales PWM de los motores en MCS4 y notifica el estado del sistema a la partición OBDH mediante MCS5.

### *4.2. Bucle de control de trayectoria*

El siguiente bucle de control en relevancia es el encargado del seguimiento de trayectoria, el cual complementa el control de bajo nivel gestionado por el FCS. Aunque este bucle depende de la información proporcionada por los subsistemas INS y CAS, opera a una frecuencia inferior, lo que permite desacoplarlo del bucle de estabilidad desde el punto de vista de la precedencia de ejecución, manteniendo únicamente dependencias de datos entre ambos.

El flujo comienza con la lectura del estado actual del cuadricóptero, estimado en la partición INS, así como de la información relativa a los obstáculos detectados en la partición CAS, correspondientes a las tareas FNS2 y FNS3, respectivamente. Posteriormente, en la tarea FNS4, se selecciona la estrategia de control más adecuada en función de la información recibida, decidiendo entre un comportamiento de evitación de obstáculos o el seguimiento convencional de la trayectoria.

A continuación, la tarea FNS5 calcula la distancia al siguiente punto de la trayectoria o, en caso de evasión de obstáculos, a un punto auxiliar generado dinámicamente. Finalmente, en la tarea FNS6 se obtiene la acción de control asociada al seguimiento de trayectoria, la cual es enviada en FNS7 al bucle interno responsable de ejecutar el movimiento de la aeronave.

La Figura 2 muestra en detalle el ETEF correspondiente al flujo descrito.

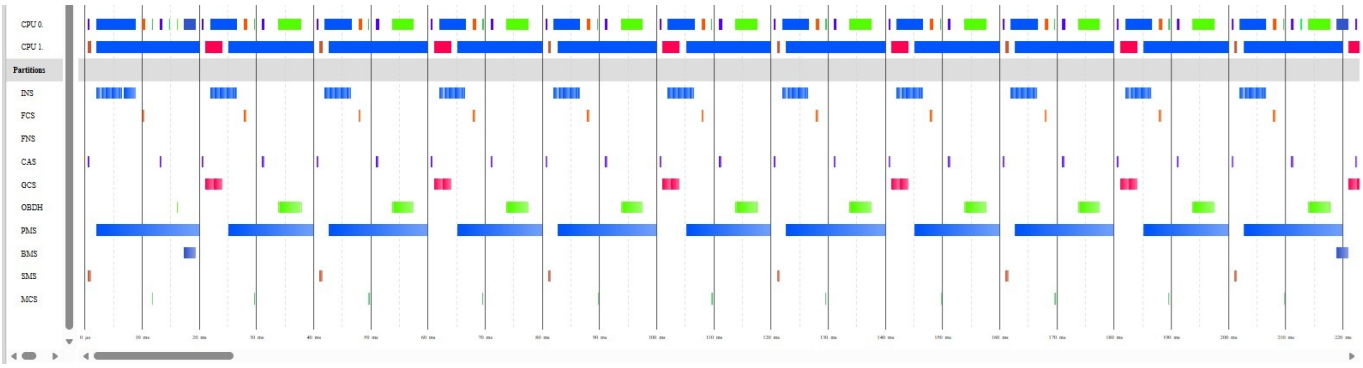


Figura 3: Fragmento de la planificación del modelo generada mediante la herramienta Xconcrete.

### *4.3. Sistema restante*

El resto de los bucles identificados no intervienen directamente en el control del movimiento de la aeronave, sino que implementan funcionalidades auxiliares del sistema, tales como el cumplimiento de la misión, la monitorización del estado interno y las comunicaciones con la estación de tierra. Desde el punto de vista estructural, estos flujos presentan una complejidad significativamente menor que los bucles de control descritos anteriormente, reduciéndose en muchos casos a secuencias de pocas tareas con relaciones de precedencia simples o incluso a la ejecución aislada de una única tarea.

Entre estos bucles destacan el flujo asociado a la lectura del LiDAR, comprendido entre las tareas CAS1 y CAS6, y el flujo de adquisición del GPS, correspondiente a la relación INS8 → INS11. En particular, las transiciones CAS3 → CAS4 e INS8 → INS11 presentan un comportamiento temporal relevante, ya que requieren la introducción de intervalos del orden de decenas de milisegundos antes de poder ejecutar la tarea sucesora. Este retardo se debe a la necesidad de que los sensores completen determinadas operaciones internas de procesamiento y adquisición antes de que la información pueda ser transmitida al sistema.

## 5. Planificación

El análisis temporal del sistema descrito en las secciones previas se ha realizado mediante el uso de la herramienta Xconcret (Brocal et al., 2010). Esta herramienta permite generar y analizar la planificación de los ETEF identificados en el sistema, además de modelar con precisión la sobrecarga asociada al hipervisor y al sistema operativo de la partición. La planificación resultante se observa en la Figura 3, la utilización real del sistema contemplando las sobrecargas alcanza una utilización del 94 % y 95 % en sus dos núcleos sin perder ningún plazo.

## 6. Conclusiones

Este artículo presenta un primer acercamiento al modelado temporal de un cuadricóptero bajo un enfoque de sistema particionado de tiempo real, con el objetivo de servir como base para el análisis de este tipo de sistemas en aplicaciones de tiempo real. El modelo recoge las tareas identificadas, sus parámetros temporales y sus dependencias, constituyendo una referencia estructurada para futuros estudios de planificabilidad. Asimismo, se plantea como un caso de uso para la aplicación de algoritmos de planificación y asignación de tareas. Finalmente, se propone la validación del modelo mediante un enfoque Hardware-in-the-Loop, utilizando un simulador para contrastar el comportamiento temporal con la ejecución real del sistema.

## Agradecimientos

Este trabajo ha sido realizado gracias al proyecto PID2024-155230OB-C42 (Re-InITS) financiado por MICIU/AEI/10.13039/501100011033 y el FEDER/UE; PREP2024-003187 financiado por MICIU/AEI/10.13039 /501100011033 y el FSE+.

## Referencias

Brocal, V., Masmano, M., Ripoll, I., Crespo, A., Balbastre, P., Metge, J.-J., May 2010. Xoncrete: a scheduling tool for partitioned real-time systems. In: ERTS 2010 proceedings. Toulouse, France.
URL: https://hal.science/hal-02264388

Burns, A., Davis, R. I., February 2026. Mixed criticality systems - a review (14th and final edition). Tech. rep., University of York.
URL: https://whiterose.ac.uk

Davis, R. I., Burns, A., 2011. A survey of hard real-time scheduling for multiprocessor systems. ACM Comput. Surv. 43 (4).

Liu, C. L., Layland, J. W., 1973. Scheduling algorithms for multiprogramming in a hard-real-time environment. J. ACM 20 (1), 46–61.

Masmano, M., Ripoll, I., Crespo, A., Metge, J.-J., September 2009. Xtratum: a hypervisor for safety critical embedded systems. In: Proceedings of the 11th Real-Time Linux Workshop. Dresden, Germany, pp. 263–272.

Poggi, T., Onaindia, P., Azkarate-askatsua, M., Grüttner, K., Fakih, M., Peiró, S., Balbastre, P., 2018. A hypervisor architecture for low-power real-time embedded systems. In: 2018 21st Euromicro Conference on Digital System Design (DSD). pp. 252–259.
DOI: 10.1109/DSD.2018.00054

Simó, J., Balbastre, P., Blanes, J. F., Poza-Luján, J.-L., Guasque, A., 2021. The role of mixed criticality technology in industry 4.0. Electronics 10 (3).

Wang, P., Man, Z., Cao, Z., Zheng, J., Zhao, Y., 2016. Dynamics modelling and linear control of quadcopter. In: 2016 International Conference on Advanced Mechatronic Systems (ICAMechS). pp. 498–503.
DOI: 10.1109/ICAMechS.2016.7813499

Wessman, N.-J., Malatesta, F., Andersson, J., Gomez, P., Masmano, M., Nicolau, V., Rhun, J. L., Cabo, G., Bas, F., Lorenzo, R., Sala, O., Trilla, D., Abella, J., 2021. De-risc: the first risc-v space-grade platform for safety-critical systems. In: 2021 IEEE Space Computing Conference (SCC). pp. 17–26.
DOI: 10.1109/SCC49971.2021.00010