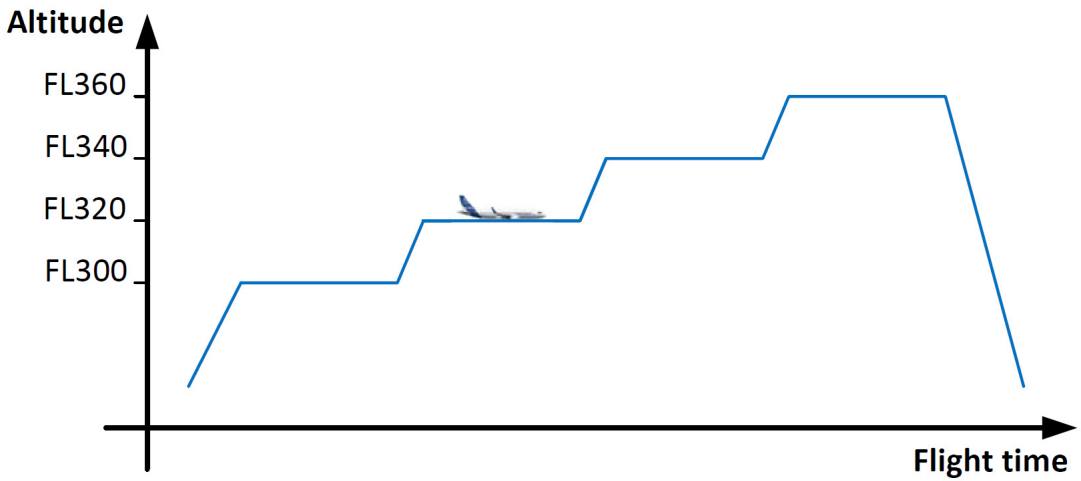
# Desarrollo de software para sistemas empotrados

## Examen Febrero 2021

# Introducción

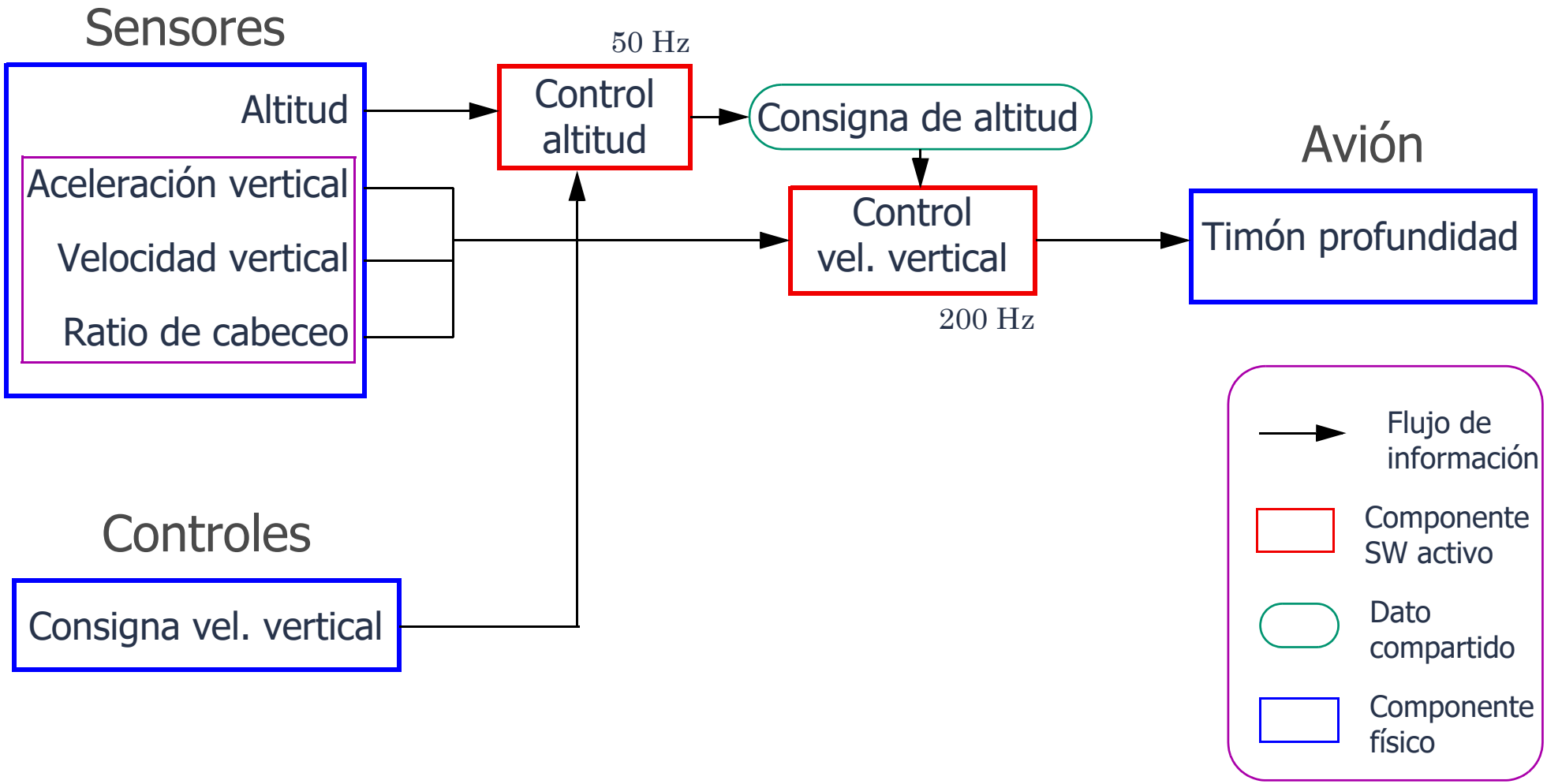
El objetivo de este ejercicio es realizar el análisis y diseño arquitectónico de una parte de un sistema de control de vuelo que se ejecuta en un sistema multiprocesador con dos CPUs.

El sistema se diseña para la fase de vuelo en ruta. Para el despegue, aterrizaje u otras fases se seguiría un diseño diferente. Esta fase tiene dos subfases:




- crucero: el piloto automático mantiene la altitud y velocidad relativa al aire
- cambio de nivel: el piloto ordena un ritmo de ascenso o descenso y una altitud objetivo; al llegar al objetivo se cambia a la subfase crucero

# Componentes del sistema: subsistema de control del timón de profundidad



# Software funcional

La funcionalidad del software está ya desarrollada en forma de 4 módulos:

- `Sensores`: Contiene dos funciones para leer datos de los sensores. Son independientes, es decir, no necesitan exclusión mutua. Son:
  - `lee_altitud()`: retorna la altitud real
  - `lee_vertical()`: retorna la aceleración y velocidad verticales así como la ratio de cabeceo reales
- `Controles`: Contiene una función para leer los mandos del piloto. No necesita exclusión mutua. Es:
  - `lee_consigna_vel_vertical()`: retorna la consigna de velocidad vertical
- `Avión`: Contiene una función para actuar sobre el avión. Es independiente, es decir, no necesita exclusión mutua. Es:
  - `actua_timon_profundidad()`: actúa sobre el timón de profundidad para cambiar el ángulo de ataque (cabeceo) del avión

# Software funcional (cont.)

- `Tareas`: Contiene dos funciones que deberán ser invocadas periódicamente para realizar el control de vuelo:
  - `control_altitud()`: obtiene datos invocando a `lee_altitud()` y a `lee_consigna_vel_vertical()`, hace cálculos y deposita el resultado invocando a `escribe_consigna_altitud()`
  - `control_vel_vertical()`: obtiene datos invocando a `lee_vertical()` y a `lee_consigna_altitud()`, hace cálculos y envía el resultado al timón de profundidad invocando a `actua_timon_profundidad()`
- Asimismo, el módulo de `Tareas` contiene un objeto compartido que almacena la consigna de altitud y sirve como comunicación entre las dos funciones indicadas arriba. También contiene dos operaciones:
  - `escribe_consigna_altitud()`: toma el mutex, escribe el dato compartido y libera el mutex
  - `lee_consigna_altitud()`: toma el mutex, lee el dato compartido y libera el mutex

# Requisitos funcionales

1. El sistema debe invocar a `control_altitud()` periódicamente, con una frecuencia de 50 Hz
2. El sistema debe invocar a `control_vel_vertical()` periódicamente, con una frecuencia de 200 Hz

# Requisitos no funcionales

3. Las actividades periódicas tienen plazos iguales a los periodos.
4. Los tiempos de ejecución de peor y mejor caso (C y $C^b$) medidos para las funciones software ya disponibles son los siguientes:

| función | C (ms) | $C^b$(ms) |
|---|---|---|
| lee_altitud() | 0.1 | 0.05 |
| lee_vertical() | 0.1 | 0.05 |
| lee_consigna_vel_vertical() | 0.1 | 0.05 |
| actua_timon_profundidad() | 0.3 | 0.1 |
| escribe_consigna_altitud() | 0.05 | 0.03 |
| lee_consigna_altitud() | 0.05 | 0.03 |
| control_altitud() | 3 | 2.9 |
| control_vel_vertical() | 2 | 1.6 |

5. Las actividades periódicas se implementan mediante dos threads alojados cada uno en una CPU de forma estática

# Requisitos no funcionales

6. Los requisitos temporales deben validarse con un análisis de planificabilidad.
7. Los modelos *temporal* y *arquitectónico* solo tendrán en cuenta los elementos software. En cambio, el modelo de requisitos tendrá en cuenta también los componentes físicos.

# Otros requisitos no funcionales

El dato compartido usa una política de techo inmediato de prioridad

Los desarrollos y el software básico estarán basados en una plataforma que dispone de dos CPUs de memoria compartida con un sistema operativo gobernado por eventos, con prioridades fijas y las siguientes características

| Propiedad | Valores |
|---|---|
| **Rango de prioridades** | 1...32 |
| **Rango de prioridades de interrupción[a]** | 32...32 |
| **Tiempo de cambio de contexto (ms)** | 0.002...0.003 |
| **Overhead de las interrupciones** | 0.002...0.002 |
| **Tipo de Temporizador** | Alarm Clock |
| **Overhead del temporizador (ms)** | 0.001...0.002 |

a. En todo caso, observar que en este sistema no hay interrupciones

# Ejercicios

1. Dentro del proceso de análisis de requisitos, generar uno o varios diagramas UCM para los requisitos del sistema
2. Modelar con AADL una arquitectura para este sistema, con 2 flujos de eventos:
   - control de altitud
   - control de velocidad vertical
3. Modelar con MAST la arquitectura del sistema para realizar un análisis de planificabilidad inicial
   - ¿Es planificable el sistema?
   - ¿Cuánto podríamos aumentar los tiempos de ejecución y seguir manteniendo el sistema planificable?
   - ¿Cuáles son los tiempos de bloqueo?
     - justifica los valores obtenidos

# Entregar 4 ficheros

Un informe en pdf con:

- diagrama(s) UCM
- diagrama AADL de nivel de sistema, con el máximo nivel de detalle
- una captura de pantalla de los resultados de MAST
- las respuestas a las preguntas planteadas

Workspace de UCMNav comprimido

Workspace de OSATE comprimido

Ficheros del modelo MAST en un archivo comprimido