CUANDO EL TIEMPO SE HACE REAL !!!

MARGA MARCOS
Grupo de Control e Integración de Sistemas
Departamento de Ingeniería de sistemas y Automática
ETSI Bilbao, UPV/EHU

Marga Marcos
marga.marcos@ehu.eus
GCIS Group, ETSI Bilbao – University of the Basque Country

Ingeniaritza Goi Eskola Teknikoa
Escuela Técnica Superior de Ingeniería
Bilbao

Universidad del País Vasco
Euskal Herriko Unibertsitatea

Indice

- ☑ **Sistemas Empotrados y de Tiempo Real**
- ❑ Principales características
- ❑ Mecanismos para sistemas de tiempo real
- ❑ ¿Qué es imprescindible recordar?

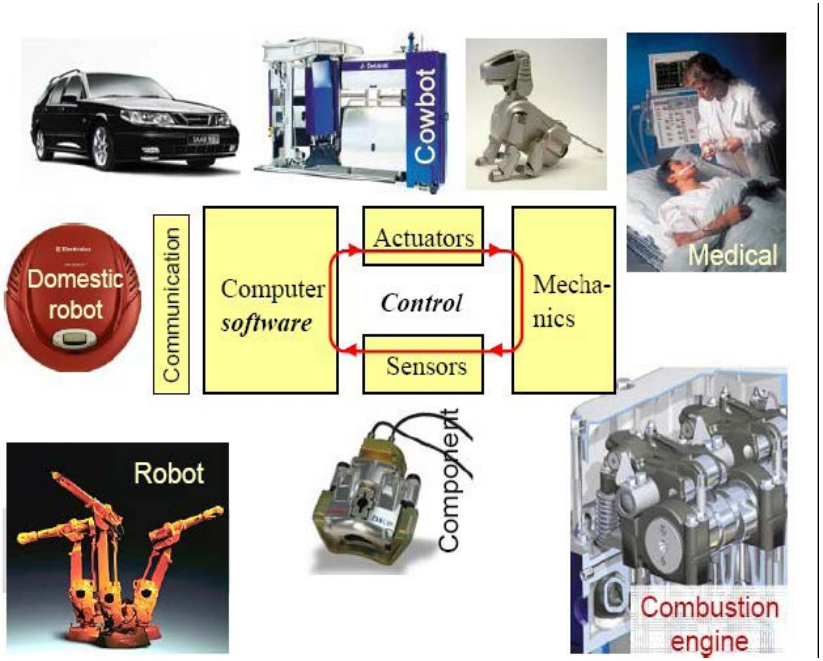
Sistemas empotrados y de tiempo real

- Un **sistema empotrado** es un sistema basado en computador que forma parte de otro sistema mayor, realizando algunas de las funciones del sistema.

Sistemas empotrados y de tiempo real

Muchos sistemas de uso común en la industria, el transporte, las comunicaciones y el hogar contienen **sistemas empotrados**:

- aviones
- Trenes
- Coches
- Teléfonos móviles
- Televisores
- …



Los **sistemas empotrados realizan funciones de control de otros sistemas**

Sistemas empotrados y de tiempo real

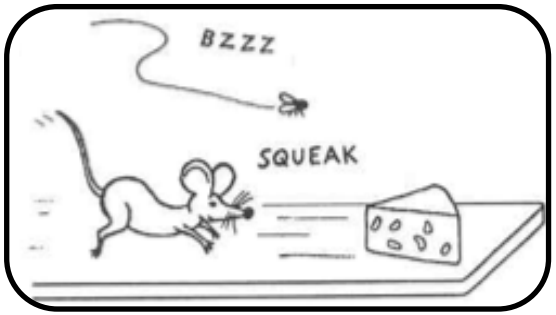
- Los **sistemas de tiempo real** son sistemas cuya corrección depende no sólo del **resultado lógico** del procesamiento que realizan si no también del **tiempo** en que lo producen.
- Es tan importante que el resultado sea correcto como que lo proporcionen a tiempo.
  - ✓ Las restricciones de tiempo vienen normalmente impuestas por el entorno.

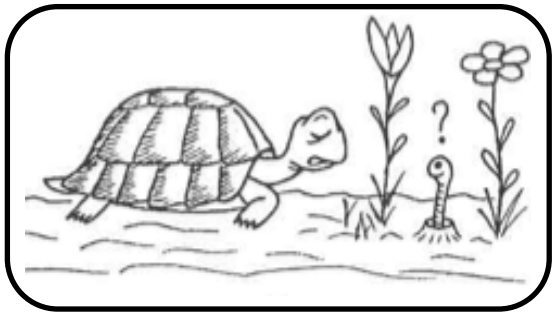
Sistemas empotrados y de tiempo real

❑ Tiempo real vs rápido

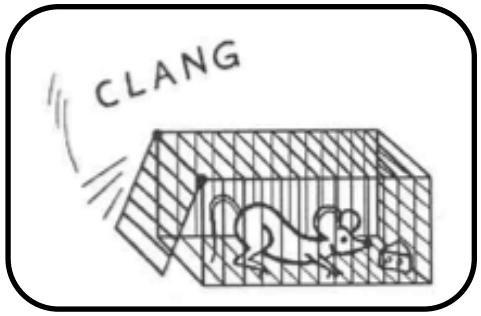
El tiempo es relativo al entorno

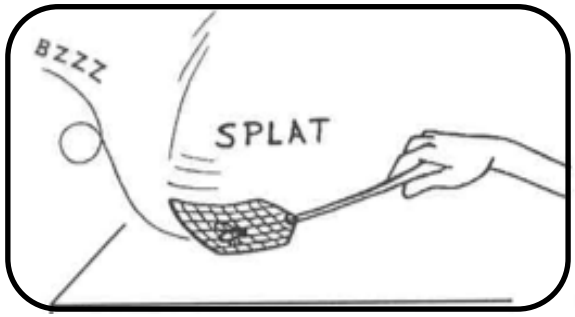
- ✓ Tanto el ratón como la tortuga viven en **tiempo real** con respecto a su habitat natural.





- ✓ Sin embargo, la supervivencia de animales rápidos como el ratón o la mosca puede ponerse en peligro debido a eventos más rápidos que su capacidad de reacción.





Sistemas empotrados y de tiempo real

❑ ¿Has llegado alguna vez tarde a coger un tren?

Normalmente (!), lo pierdes!!
Y esto puede tener consecuencias **REALES**!

Por tanto, el **tiempo** de retraso fue **REAL**

En general,

Si llegar tarde tiene un impacto en el curso de una acción o, en otras palabras, cuando el resultado deseado impone restricciones de tiempo, entonces se trata de una **situación de tiempo real**.

Sistemas empotrados y de tiempo real

❑ ¿Quién impone las restricciones de tiempo?

✓ La dinámica del entorno (**proceso físico**)

- Control de sistemas dinámicos
- Procesamiento en línea de video, audio...

  Particularmente en casos interactivos (llamadas, video conferencias…)
- Limitaciones de sensores/actuadores:

  Mínimo tiempo entre operaciones.

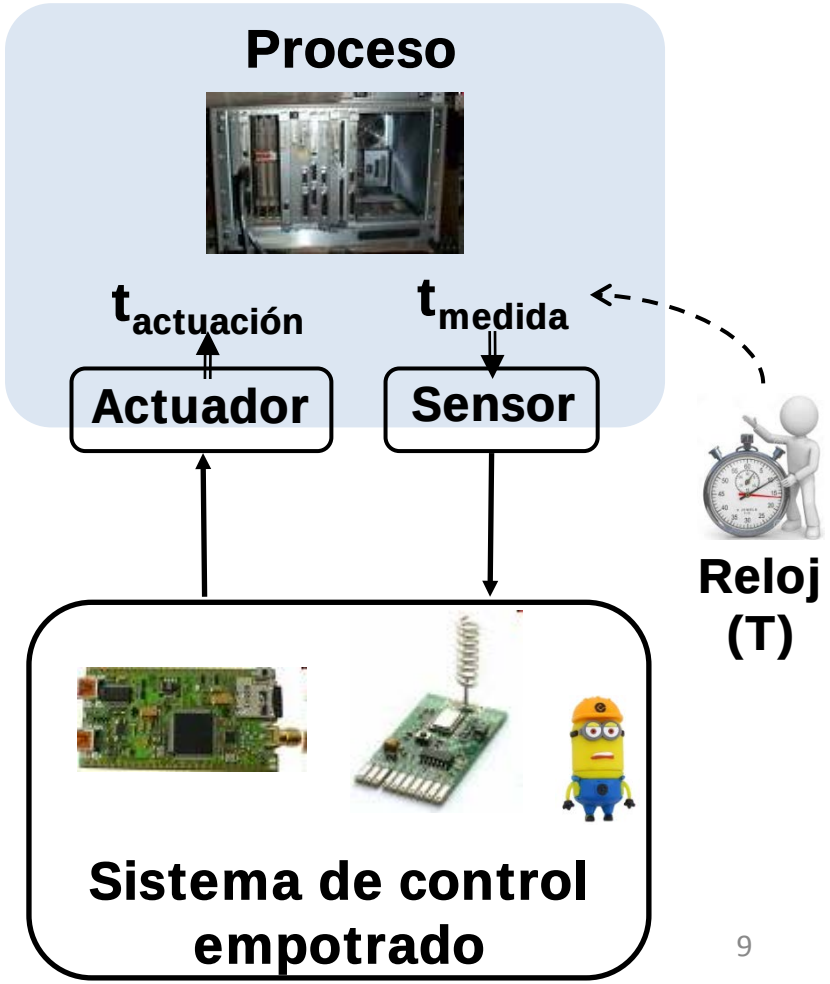
✓ Restricciones asociadas al **proceso lógico**

- Protocolos de communicación
- Sistemas de realidad virtual (Simuladores de procesos físicos)

Sistemas empotrados y de tiempo real

❑ Ejemplo: sistema de control empotrado

```
Control_loop
    repetir
        Leer_sensor;
        Calcular_accion_control;
        Enviar_accion_control;
    cada T
```



$\forall k, (t_{k\_actuación} - t_{k\_medida}) < \delta$
$\forall k, (t_{k+1medida} - t_{kmedida}) \cong T$

Restricciones de tiempo
externas al sistema de control

Sistemas empotrados y de tiempo real

❑ ¿Cómo de malo es llegar tarde?

1 Sistemas **críticos - Hard Real Time Systems (HRT)**

- ✓ Fallos de tiempo no admisibles
- ✓ Se asocian con aplicaciones críticas en seguridad.
- ✓ Un fallo (por ejemplo de tiempo) puede llevar a la pérdida de vidas humanas, daños al medio-ambiente o severas pérdidas económicas.

Campos de aplicación:

Transporte, espacio, producción de energía, equipamiento médico …

Sistemas empotrados y de tiempo real

❑ ¿Cómo de malo es llegar tarde?

2 Sistemas **flexibles** - **Soft Real Time**

- ✓ Fallos de tiempo no deseados pero no críticos. Degradación de la calidad de servicio (QoS)
- ✓ El valor de su respuesta depende del tiempo en que está disponible:
  - Un resultado imperfecto a tiempo es a veces mejor que un resultado exacto pero tardío.
  - Cuanto más resultados dé antes de un plazo mejor.
  - Existe relación directa entre el tiempo de entrega y el valor de la respuesta.

Sistemas empotrados y de tiempo real

❑ ¿Cómo de malo es llegar tarde?

2 Sistemas **flexibles** - **Soft Real Time**

- ✓ Fallos de tiempo no deseados pero no críticos. Degradación de la calidad de servicio (QoS)
- ✓ El valor de su respuesta depende del tiempo en que está disponible

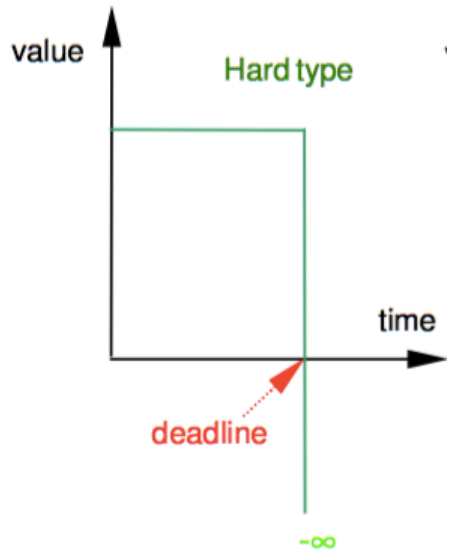
Campos de aplicación:

Multimedia, simuladores, comunicaciones…

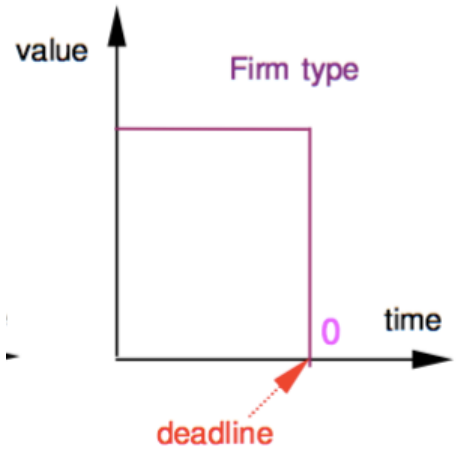
Sistemas empotrados y de tiempo real

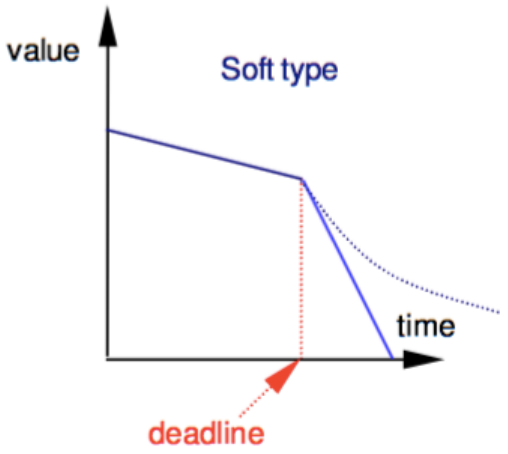
❑ ¿Cómo de malo es llegar tarde?

1 Sistemas **críticos**



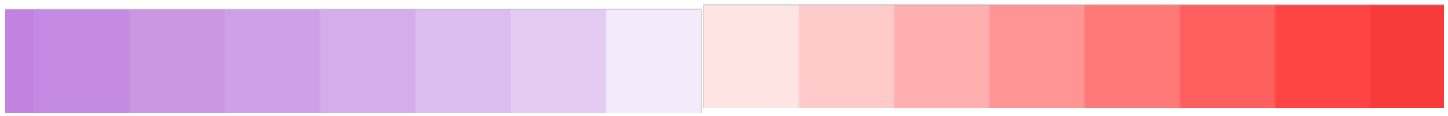
2 Sistemas **no críticos**





Sistemas empotrados y de tiempo real

❑ Espectro de los sistemas de tiempo real



No TR ⟵ TR flexible ⟶ TR crítico

Simulación por computador

Interfaces de usuario

Video

audio por internet

Control de crucero

telecomunicaciones

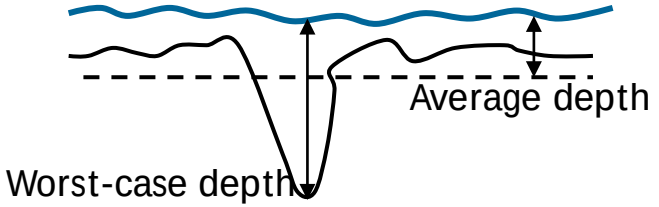
Control de vuelo

Tren de Alta Velocidad, …

Sistemas empotrados y de tiempo real

❑ Peor-caso versus caso-medio

Un estadístico se ahogó cruzando un río con una profundidad media de 20 cm!!

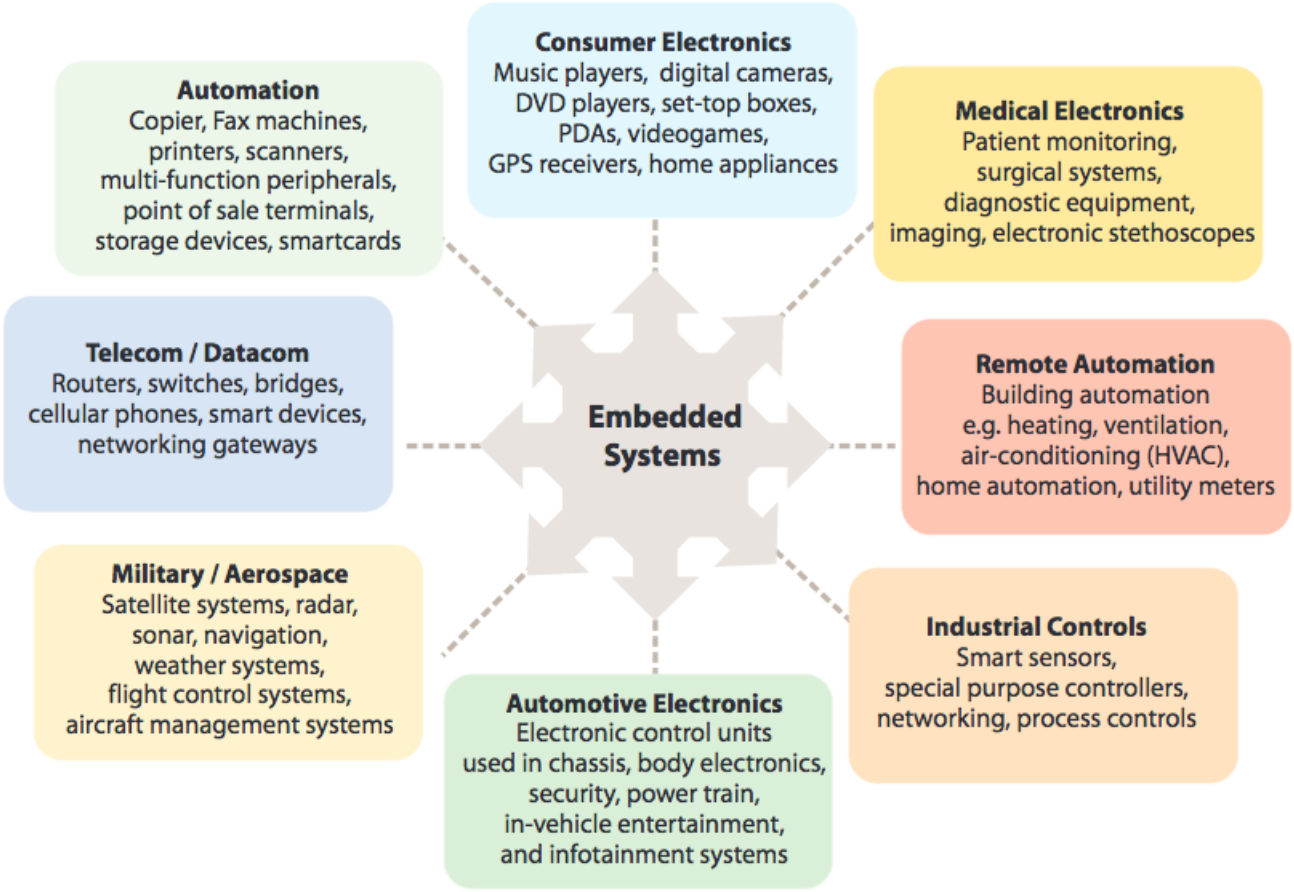


Los **tiempos de respuesta medios a eventos** pueden ser cortos pero una secuencia de eventos tratados tarde puede llevar al sistema a una situación inmanejable y provocar un fallo crítico.

- ❖ Rápido: tiempo medio bajo
- ❖ Tiempo real: predecible, tiempo de peor caso acotado

Sistemas empotrados y de tiempo real

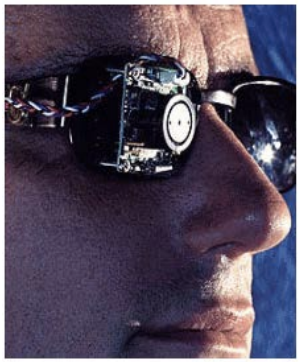
❑ Ejemplos de aplicación



Sistemas empotrados y de tiempo real

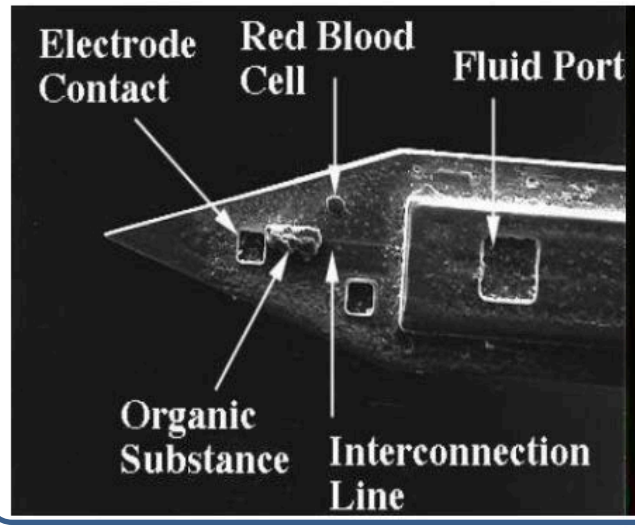
❑ Ejemplos de aplicación

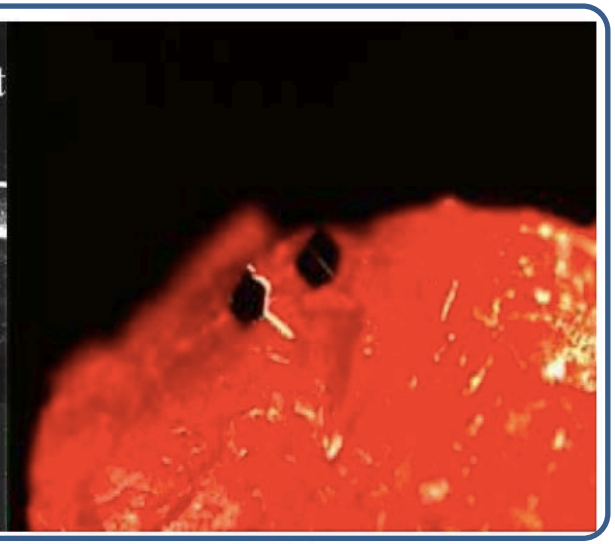
Sistemas médicos



El ojo artificial

Micro agujas

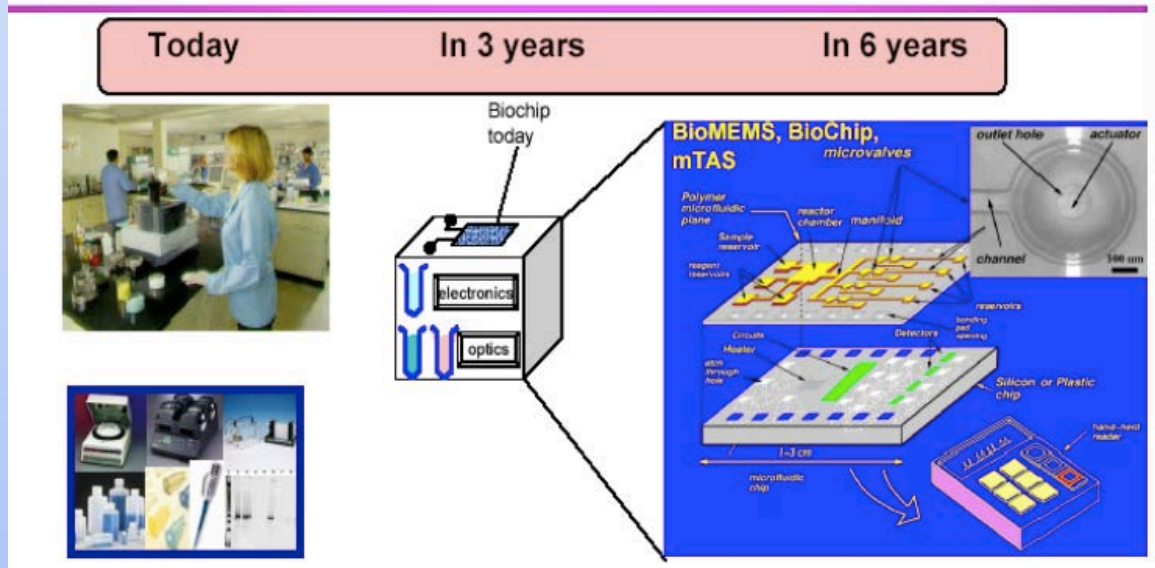




Sistemas empotrados y de tiempo real

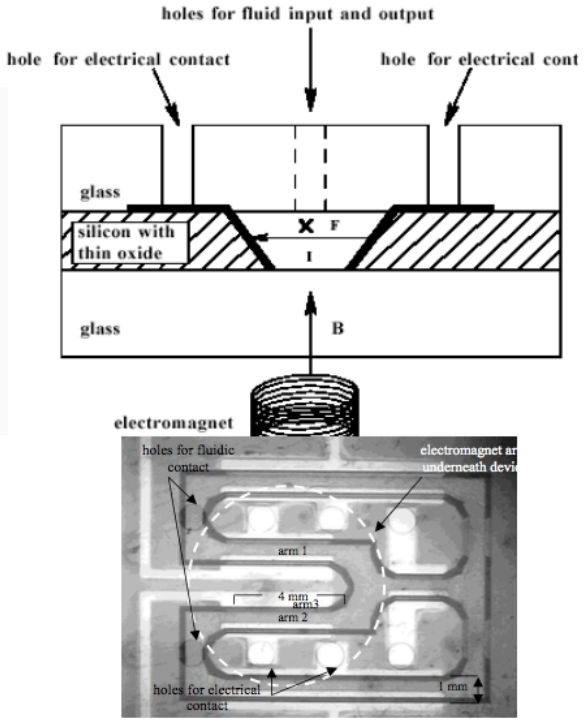
❑ Ejemplos de aplicación

Química en un chip



Horas-días 1 hora segundos

tiempo desde la muestras hasta la respuesta



Sistemas empotrados y de tiempo real

❑ Ejemplos de aplicación

Podómetro

Trabajo obvio:

- Contar pasos,
- Guardar el tiempo,
- Hacer la media,…

Trabajo duro:

- Identificar cuándo se da un paso,
- El sensor mide el movimiento del dispositivo no del pie

Sistemas empotrados y de tiempo real

❑ Ejemplos de aplicación

Dentro de un ordenador personal

Procesadores personalizados:
- Gráficos, sonido

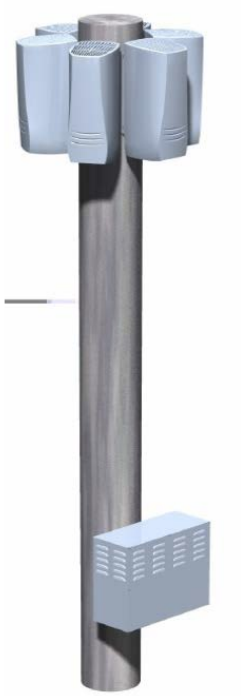
Procesadores de 32-bit
- IR, Bluetooth
- Red, WLAN
- Disco duro
- Controladores RAID

Procesadores de 8-bit
- USB
- Teclado, ratón

Sistemas empotrados y de tiempo real

❑ Ejemplos de aplicación

Estación base de telefonía móvil



Procesamiento de señal masivo

- Varias tareas por móvil conectado

Basado en DSPs

- Estándar o personalizado
- Cientos de procesadores

❑ Ejemplos de aplicación

Máquina de soldadura inteligente

Control electrónico de tensión y de la velocidad de alimentación del hilo.
Ajustes de operador:

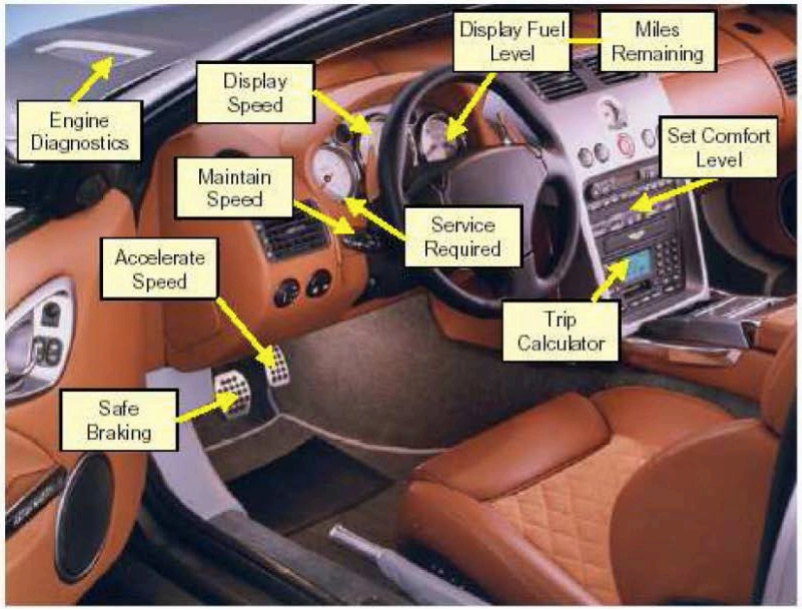
- Frecuencia de muestreo (KHz)
- Miles de decisiones por segundo

Soldadura perfecta incluso con operadores torpes.
Producto fácil de usar pero el sistema no es trivial.

Sistemas empotrados y de tiempo real

❑ Ejemplos de aplicación

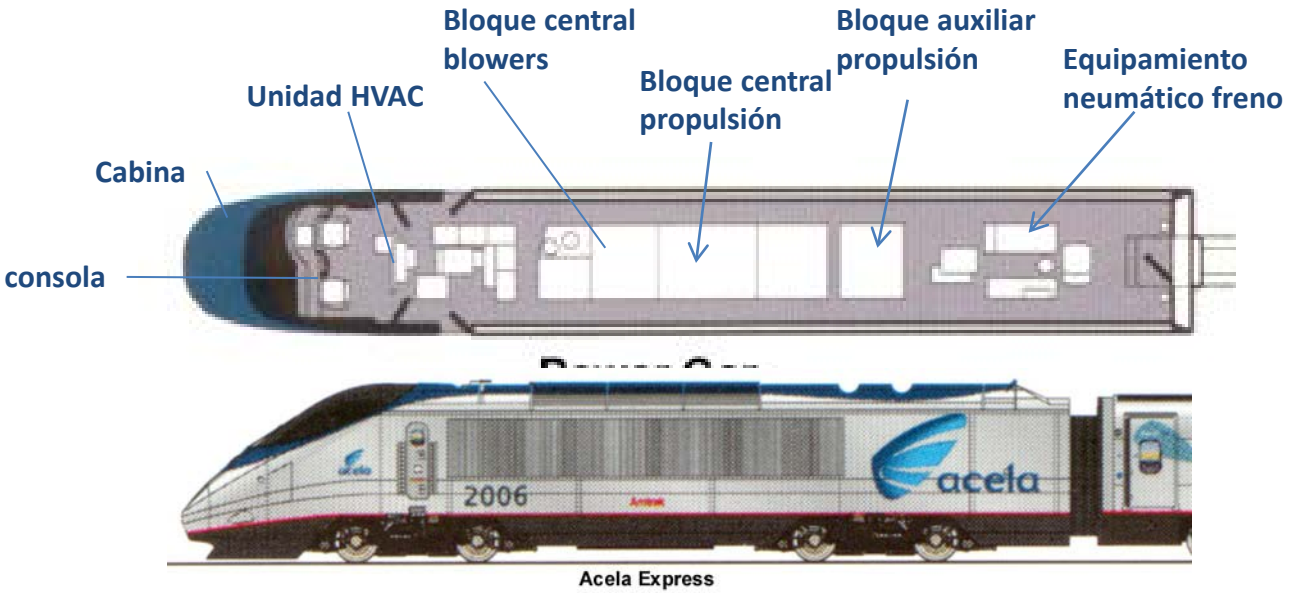
Funciones de control en el automóvil



Sistemas empotrados y de tiempo real

❑ Ejemplos de aplicación

Tren de alta velocidad (ACELA)



Sistemas empotrados y de tiempo real

❑ Ejemplos de aplicación

Automatización de edificios

Coeur Défense, Paris

- 182.000 m2
- 2 torres de 180m y 3 de 8 pisos
- 15000 sistemas de control empotrados

Sistemas empotrados y de tiempo real

❑ Ejemplos de requisitos temporales

Sistema de navegación de un avión

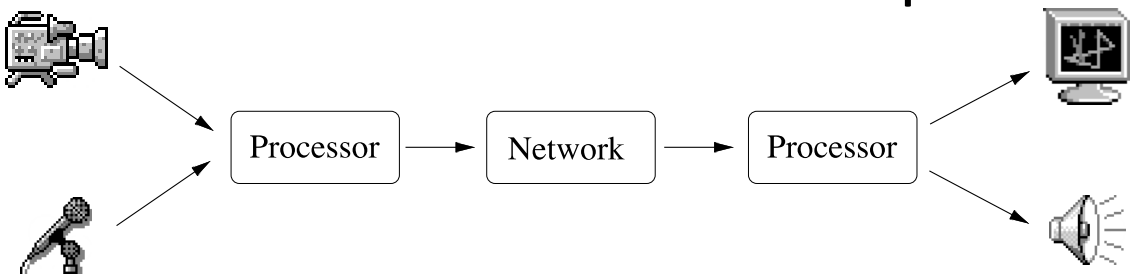
- Requisitos temporales muy exigentes (máximo tiempo de respuesta, plazo,…)
- Debe asegurar que dispone de los recursos necesarios incluso en la peor situación.
- Redundancia hardware y software (uso dedicado de componentes)
  - SOTR (e.g. RTEMS), procesador embebido (e.g. Leon Sparc).
  - Entradas: Medidas aceleración x,y,z (5 ms), Ángulo de inclinación, giro y cabeceo (40 ms), Temperatura (1 s)
  - Salidas: Cálculo de la velocidad real (cada 40 ms), visualización velocidad (cada 1 s).

Los procesos son concurrentes y de diferente temporización

Sistemas empotrados y de tiempo real

❑ Ejemplos de requisitos temporales

Aplicaciones multimedia



Restricciones temporales:

- Tiempo de respuesta de peor caso
- Retardo extremo a extremo
- Sincronización del flujo y entre flujos.

Entorno de ejecución de propósito general (PCs y S.O. Windows).

Requisitos de recursos difíciles de especificar:

- Nº de flujos, ancho de banda de cada uno
- No se pueden reservar los recursos necesarios en peor caso.

Los procesos comparten recursos

Sistemas empotrados y de tiempo real

❑ Ejemplos de requisitos temporales

Sistema de monitorización de una planta nuclear

Evento disparado por el valor de una señal

- Diferentes niveles de seguridad)
  → respuesta en 1s

Señales críticas (sobre temperatura del núcleo)

- Respuesta en 1 ms

Los procesos tienen diferente prioridad y criticidad

Sistemas empotrados y de tiempo real

❑ Ejemplos de requisitos temporales

Sistema de reserva de billetes de una aerolínea

Sistema distribuido

- diferentes agentes pueden utilizar el sistema concurrentemente.

Tiempo de entrega

- menor de 15 s

Sin overbooking

Los procesos comparten recursos

Sistemas empotrados y de tiempo real

❑ Ejemplos tristemente célebres

¿¿QUÉ PUEDE IR MAL????

Importancia de las consecuencias

Sistemas empotrados y de tiempo real

❑ Ejemplos tristemente célebres

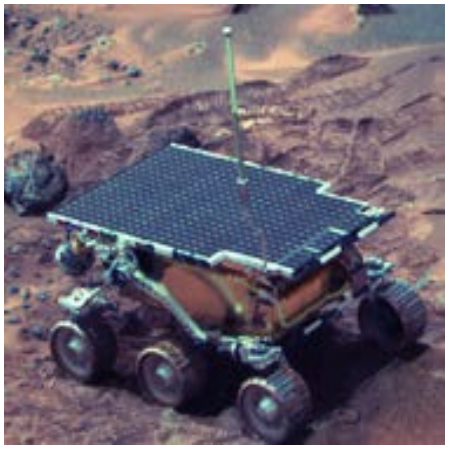
- ✓ Explotó 40s después del despegue (1996)
- ✓ Desintegración provocada por desviaciones en todos los motores (comandados en base a datos transmitidos por un computador de referencia inercial).
- ✓ Excepción de coma flotante no tratada. El manejo de excepciones estaba desactivado para ajustar la utilización de CPU .
- ✓ Habían transcurrido 37 segundos desde el encendido del motor principal Vulcain, el cohete se autodestruyó a 3.400 metros de altura sobre la base espacial de Kourou.

Sistemas empotrados y de tiempo real

❑ Ejemplos tristemente célebres

- ✓ Nave espacial no tripulada, aterrizó en Marte en 1997.
- ✓ Interbloqueos frecuentes -> reinicios, pérdida de tiempo.
- ✓ Provocado por el clásico problema de inversión de prioridades (en el acceso a datos compartidos).

Mars Pathfinder



Sistemas empotrados y de tiempo real

❑ Ejemplos tristemente célebres

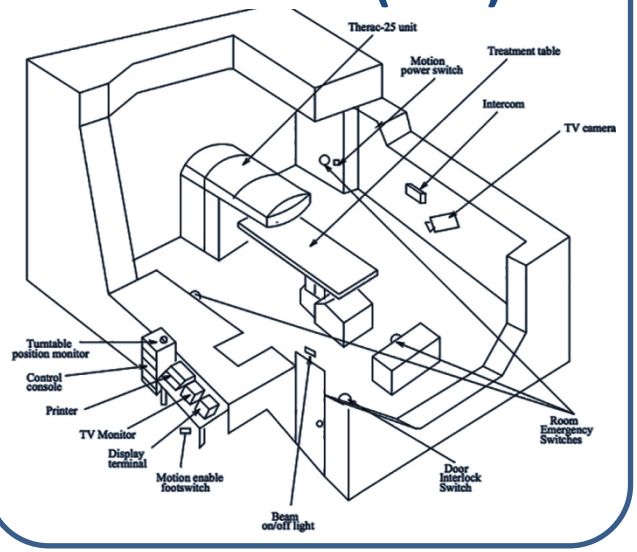
Apollo 11

- ✓ Primer aterrizaje en la Luna (20-7-1969).
- ✓ Fallo software durante el descenso casi se aborta el aterrizaje.
- ✓ Nave espacial equipada con un computador con el sistema de navegación y guiado. Sistema de control saturado (computador lento para manejar todas las tareas concurrentes).
- ✓ Saltaban alarmas y no se ejecutaron trabajos de baja prioridad (no críticos).
- ✓ Los ingenieros responsables decidieron ignorar el problema (posteriormente recibieron la misma medalla que los astronautas).

Sistemas empotrados y de tiempo real

❑ Ejemplos tristemente célebres

Therac-25 (80's)



- ✓ Máquina de radiación terapeútica
- ✓ 6 muertos y daños graves (sobredosis de radiación masiva).
- ✓ Provocado por condición de carrera (violación de acceso en exclusión mutua).
- ✓ Operaba en 2 modos: electrón (baja energía) y rayos X (alta energía). El operador pasó erróneamente a modo rayos X. Se dio cuenta y volvió a modo electrón (baja energía). En menos de 8 s.

En esa ventana de tiempo:

- ✓ La tarea de fase de tratamiento ignoró las órdenes de teclado
- ✓ Otras tareas sí registraron el cambio
- ✓ Radiación de alta energía desprotegida sin indicación al operador

Sistemas empotrados y de tiempo real

❑ Ejemplos tristemente célebres

Misil Patriot

- ✓ Sistema usado para proteger a Arabia Saudí durante la Guerra del Golfo.
- ✓ Detecta objetos volantes y realiza predicción. La trayectoria encaja con la predicción → misil lanzado.
- ✓ 25-2-1991: misil Scud cae sobre Dhrahran, clasificado como falsa alarma.
- ✓ Error software: reloj de tiempo real acumulando un retraso de 57 µs por minuto, 100 h → 343 ms.

Sistemas empotrados y de tiempo real

❑ Lecciones a aprender:

- Si algo puede ir mal, irá mal.
- El argumento: “funciona ahora” no es válido en un sistema de tiempo real.
- Las pruebas pueden detectar muchos errores, pero nunca son garantía de corrección.
- La corrección debería idealmente ser verificada formalmente.

Sistemas empotrados y de tiempo real

❑ Errores generales en programación:

- INTEL: no más de 80-90 errores en el Pentium.
- Software estándar:

  25 errores por cada 1000 líneas de código.
- Software bueno : 2 errores por cada 1000 líneas.
- Software de transporte espacial:

  < 1 error por cada 10000 líneas.
- Windows-95: 10 Millones de líneas; hasta 200.000 errores.

Indice

- ❑ Sistemas Empotrados y de Tiempo Real
- ☑ **Principales características**
- ❑ Mecanismos para garantizar el comportamiento temporal
- ❑ ¿Qué es imprescindible recordar?

Principales características

❑ Tamaño y complejidad

- ✓ El número de líneas de código es un síntoma.
- ✓ La funcionalidad diversa incrementa la complejidad de sistemas relativamente pequeños.

❑ Concurrencia

- ✓ Los dispositivos físicos (el entorno) deben ser controlados de forma simultánea.
- ✓ Las acciones de control actúan concurrentemente.

❑ Dispositivos de entrada/salida específicos

- ✓ Los manejadores de dispositivos forman parte del software de la aplicación.

Principales características

❑ Determinismo temporal

- ✓ Deben realizar sus acciones en instantes de tiempo determinados.
- ✓ El comportamiento temporal debe ser determinista o al menos predecible.
  - Lo que no significa que deba ser eficiente.
  - El sistema debe responder correctamente a cualquier situación.
  - Si el sistema es de tiempo real crítico, es necesario asegurar que los requisitos temporales se cumplen incluso en el peor caso.

Principales características

❑ Confiabilidad (Dependability)

- ✓ Sistemas críticos: los fallos conducen a situaciones catastróficas.
  - Pérdida de vidas humanas, pérdidas económicas, daños medio-ambientales,…

❑ Fiabilidad (Reliability)

- ✓ Probabilidad de suministrar el servicio especificado
  - Métrica: **M**edium **T**ime **T**o **f**ailure (MTTF)
  - Sistemas con MTTF > 10^9 se denominan **sistemas ultrafiables**

Principales características

❑ Seguridad (Safety)

- ✓ Los fallos pueden ser malignos y benignos
- ✓ El coste de un fallo maligno es mucho mayor que la utilidad del sistema.

Sistemas críticos

- ✓ Deben ser **ultrafiables** respecto a fallos malignos.
- ✓ En muchos casos se requiere una certificación emitida por un agente independiente.

Principales características

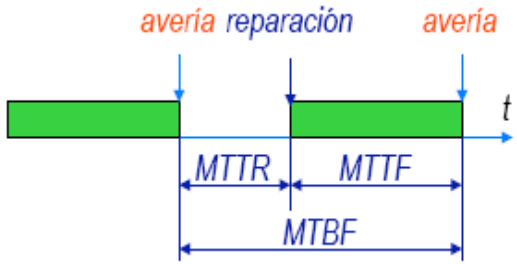
❑ Mantenibilidad (Maintainability)

- ✓ Medium Time To Repair (MTTR).
- ✓ Tiempo necesario para reparar un fallo benigno.
- ✓ Conflicto: los elementos fáciles de mantener son menos fiables.

❑ Disponibilidad (Availability)

- ✓ Intervalo de tiempo durante el que el sistema está disponible.

$$A = \frac{MTTF}{MTTF + MTTR} = \frac{MTTF}{MTBF}$$



Principales características

❑ Tipos de sistemas de tiempo real

1 Sistemas **críticos (HRT)**

- plazo estricto
- comportamiento temporal impuesto por el entorno
- respuesta predecible ante situaciones de sobrecarga
- requisitos de seguridad críticos
- redundancia activa
- volúmenes de datos reducidos

2 Sistemas **flexibles(SRT)**

- plazo flexible
- comportamiento temporal impuesto por el sistema de cómputo
- comportamiento degradado a situaciones de sobrecarga
- recuperación de fallos
- volumen de datos alto

Se distinguen por sus requisitos de tiempo y seguridad

Principales características

❑ Tipos de sistemas de tiempo real

3. sistemas con parada segura (**fail-safe**)
   - Paran en un estado seguro
   - Alta probabilidad de detección de fallos

4. sistemas con degradación aceptable (**fail-soft**)
   - Operan con pérdida parcial de funcionalidad o prestaciones
   - Existen sistemas con tolerancia a fallos completa (**fail-operational)**

Se distinguen por sus comportamiento a fallos

Principales características

❑ Tipos de sistemas de tiempo real

5 sistemas con sistemas con **respuesta garantizada**

- Comportamiento temporal garantizado analíticamente
- Es necesario definir la carga y los fallos de forma precisa

6 sistemas que hacen lo que pueden (**best effort**)

- La carga y los fallos no se caracterizan precisamente.
- Sólo resultan útiles en sistemas de tiempo real flexible.

Se distinguen por su comportamiento temporal

Principales características

❑ Tipos de sistemas de tiempo real

7 sistemas con sistemas con **recursos adecuados**

- Se diseñan con recursos suficientes para garantizar el comportamiento temporal bajo máxima carga y fallos.

8 sistemas con **recursos inadecuados**

- Se diseñan con recursos "razonables" desde un punto de vista económico
- Sólo son útiles para sistemas de tiempo real flexible.

Se distinguen por la cantidad de recursos disponibles

Principales características

❑ Tipos de sistemas de tiempo real

7 sistemas conducidos por eventos (**event-triggered**)

- La activación se produce por un evento o cambio de estado.
- Mecanismo básico: interrupciones.

8 sistemas conducidos por tiempo (**time-triggered**)

- La activación se produce en determinados instantes de tiempo.
- Mecanismo básico: reloj.

Se distinguen por la forma en que comienzan sus actividades

Indice

- ❑ Sistemas Empotrados y de Tiempo Real
- ❑ Principales características
- ☑ **Mecanismos para tiempo real**
- ❑ ¿Qué es imprescindible recordar?

Mecanismos para tiempo real

¿Cómo se puede garantizar el determinismo temporal?

- ✓ **Ejecución concurrente** de tareas
  - *Threads* (procesos ligeros) o mecanismos similares
- ✓ **Medida del tiempo**: relojes
  - Tareas periódicas y esporádicas
- ✓ **Método de planificación** del procesador que permita analizar el comportamiento temporal del sistema.
  - Uso de **prioridad** de las tareas
  - Otros métodos de planificación
- ✓ **Análisis del comportamiento temporal** del sistema antes de implementarlo.
  - Las pruebas no siempre garantizan el peor caso

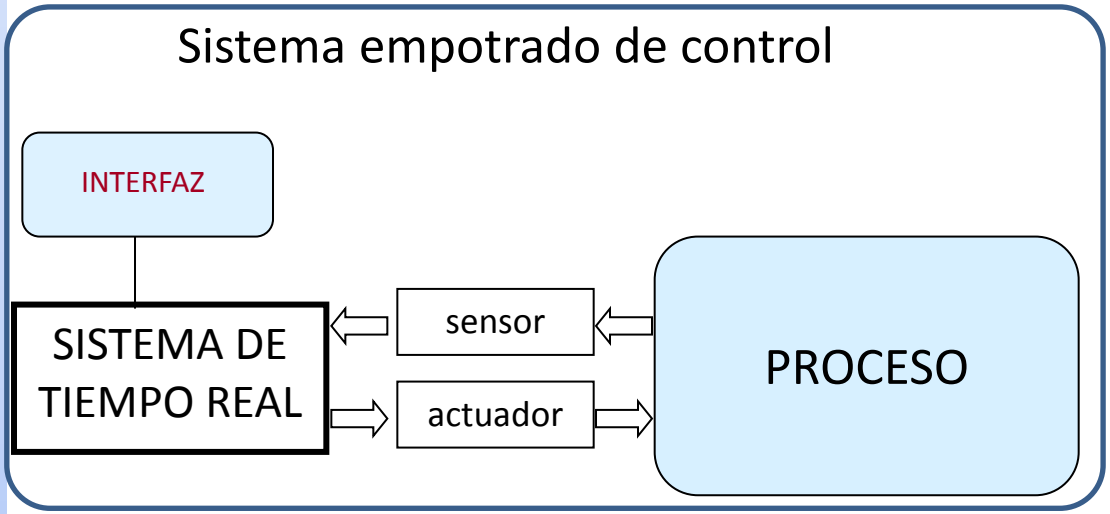
Mecanismos para tiempo real

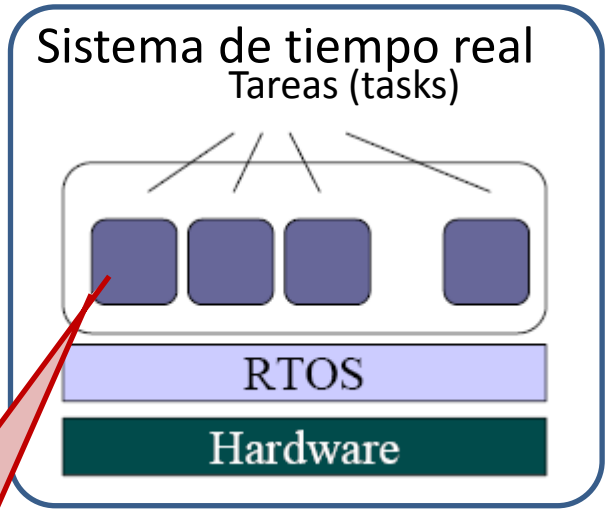
Concurrencia

- ✓ Por razones de ***eficiencia***, la mayoría de los RTOS soportan:
  - 1 proceso con un conjunto de flujos de control (tarea o ***thread***).
  - Todos los *threads* comparten memoria (cambio de contexto más eficiente).
  - Ejemplos: RTAI, RT-Linux, Shark, VxWorks, QNX, etc.
- ✓ El planificador (del S.O. o del núcleo en ejecución) es el encargado de seleccionar el siguiente proceso a ejecutar.
  - En sistemas de tiempo real se usa la **prioridad** de las tareas.

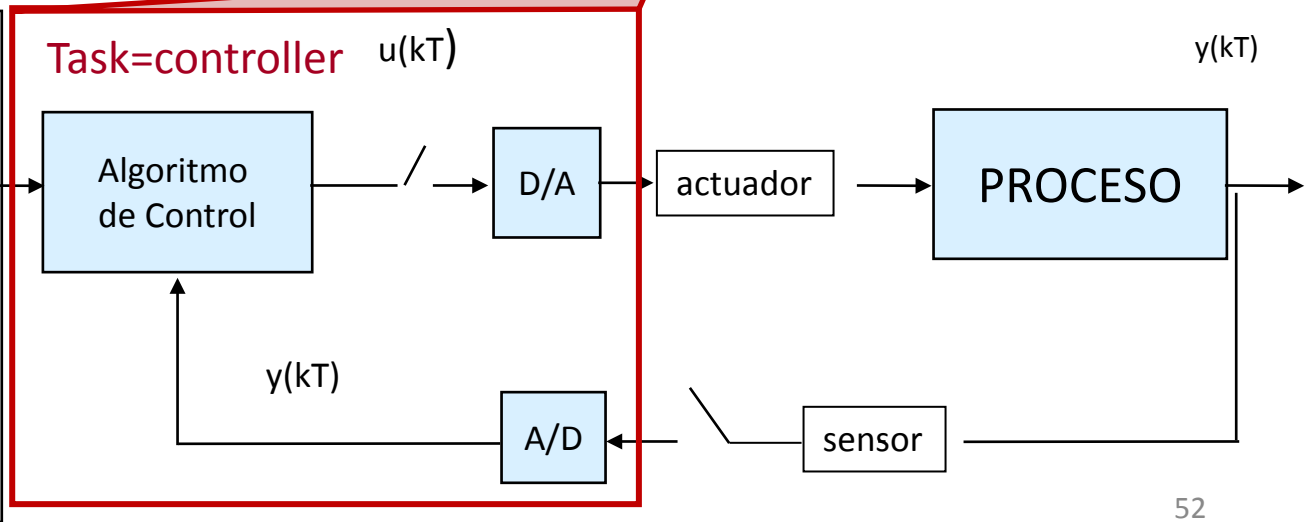
Mecanismos para tiempo real

❑ Concurrencia





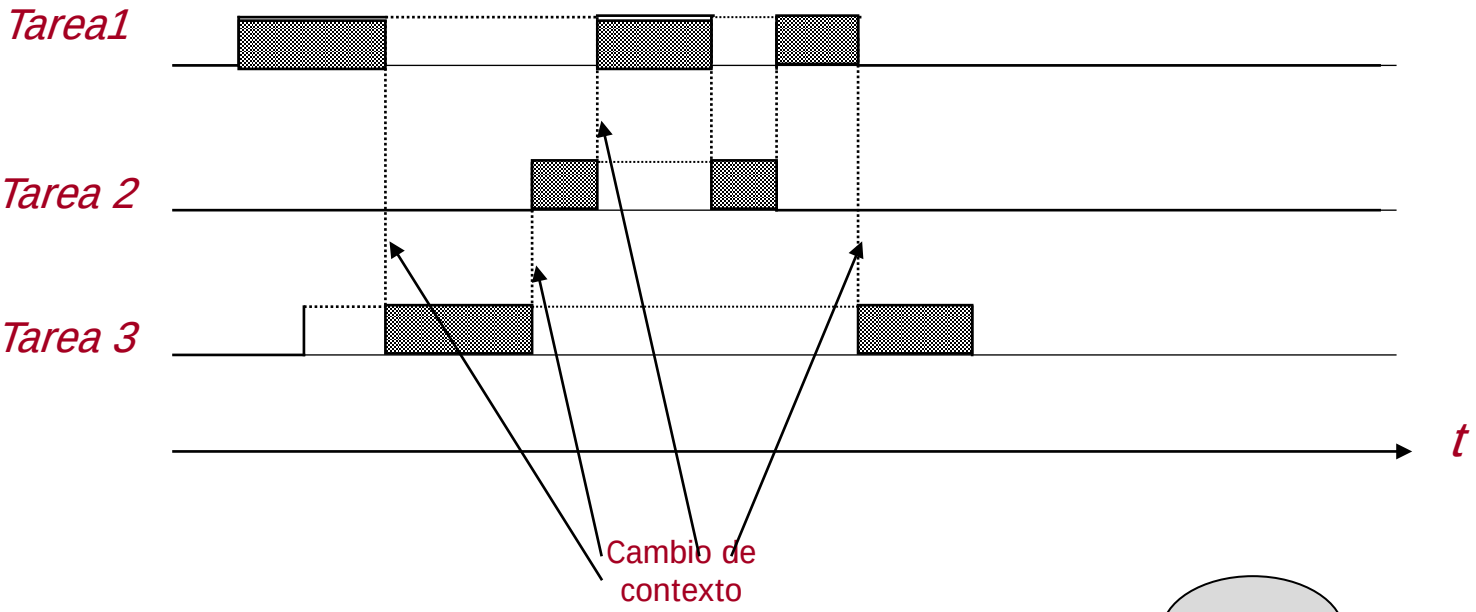

```
task controller
 each sample do
  read_sensor(y);
 compute_control_action(u);
 send_control:action(u):
 update:variables(e,y,u,…)
 end do;
end task;
```

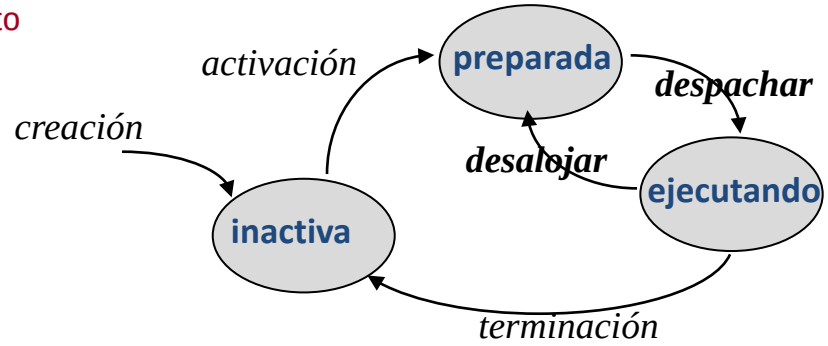


Mecanismos para tiempo real

Concurrencia

- ✓ Multiprogramación:





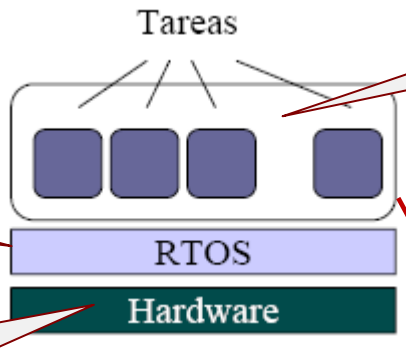
Las acciones de *despachar* y *desalojar* las realiza el **planificador**.

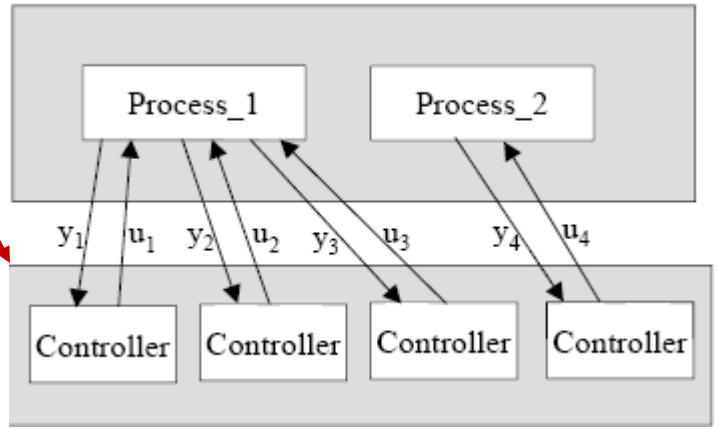
Mecanismos para tiempo real

❑ Concurrencia

RTOS:
gestiona las tareas concurrentes y la comunicación

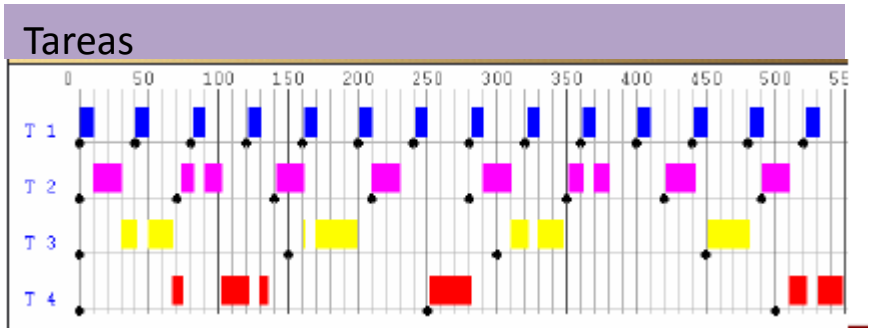
La aplicación se organiza en tareas concurrentes





Microprocesadores y periferia

Planificador basado en prioridades

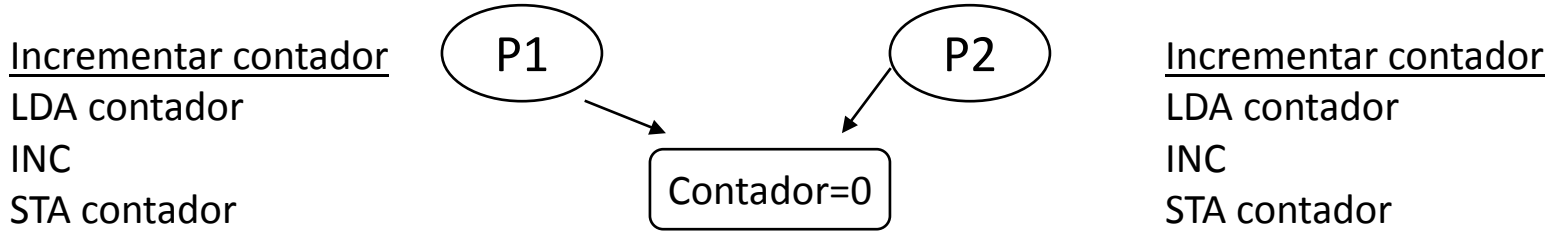


Mecanismos para tiempo real

❑ Concurrencia

- ✓ Acceso a recursos compartidos de uso exclusivo:

Ejemplo 1: dos tareas accediendo a una variable



No es una operación indivisible: carga, incremento, almacenamiento.
El resultado final puede ser 1 o 2 (**condición de carrera**, el resultado depende del orden en que se ejecutan los procesos).

Ejemplo 2: la tarea1 escribe un mensaje en el puerto serie.
La tarea 2 la interrumpe y escribe otro.

```
Task 1 ( )                Task 2 ( )
{...                      {...
printf (“*Task1*”);        printf (“*Task2*”);
...                       ...
}                         }
```

Output in the serial line:

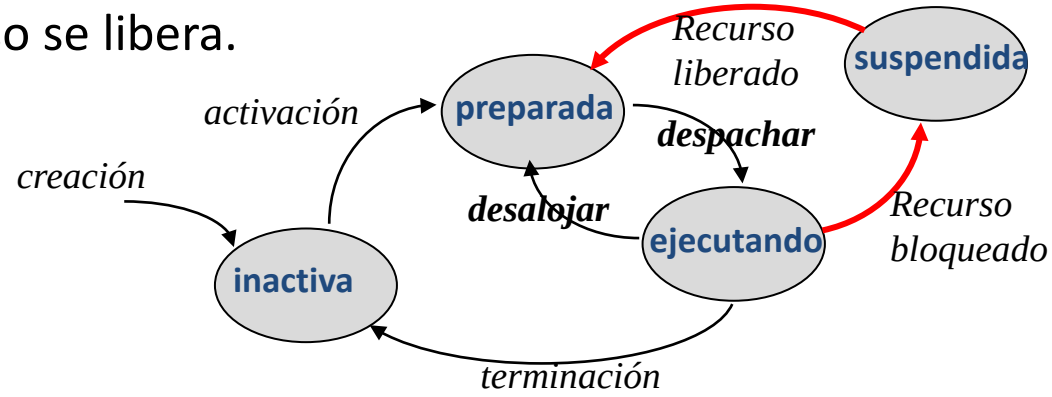
Tas*Task2*k1

Transmission time

Mecanismos para tiempo real

❑ Concurrencia

- ✓ Acceso a recursos compartidos de uso exclusivo:
  - Para evitar condiciones de carrera se debe asegurar que las operaciones sobre recursos de acceso exclusivo se realiza de **forma atómica**.
  - Una secuencia de instrucciones que se ejecuta de forma indivisible se denomina **sección crítica**.
  - Cuando una tarea en ejecución trata de acceder a un recurso compartido que está siendo utilizado por otra tarea de menor prioridad, se **bloquea** (pasa a estado de suspendida) hasta que el recurso se libera.



Mecanismos para tiempo real

❑ Concurrencia

- ✓ Acceso a recursos compartidos de uso exclusivo:
  - Para evitar condiciones de carrera se debe asegurar que las operaciones sobre recursos de acceso exclusivo se realiza de **forma atómica**.
  - Una secuencia de instrucciones que se ejecuta de forma indivisible se denomina **sección crítica**.
  - Cuando una tarea en ejecución trata de acceder a un recurso compartido que está siendo utilizado por otra tarea de menor prioridad, se **bloquea** (pasa a estado de suspendida) hasta que el recurso se libera.
  - El mecanismo más básico para proteger el acceso exclusivo a recursos compartidos es el **semáforo.**

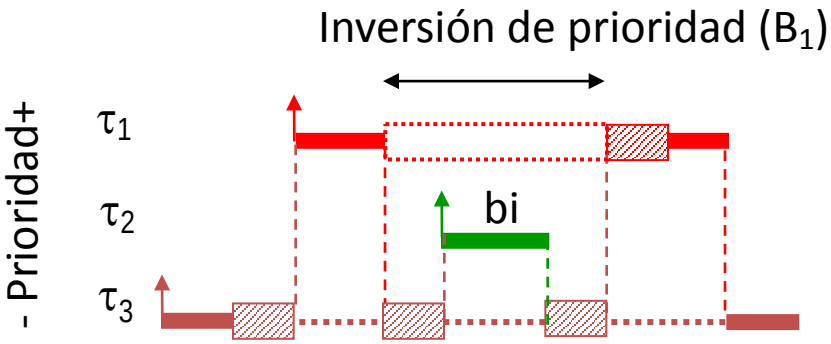
Mecanismos para tiempo real

❑ Concurrencia

- ✓ Acceso a recursos compartidos de uso exclusivo:

Inversión de prioridad:

- Cuando una tarea se bloquea, la tarea bloqueante es de menor prioridad. Cuando la tarea bloqueante se ejecuta (y cualquier otra de prioridad intermedia) existe **inversión de prioridad**.



- La inversión de prioridad es un fenómeno inevitable en presencia de acceso a recursos compartidos en exclusión mutua.
- EL bloqueo directo se puede acotar. El bloqueo indirecto no está acotado y se debe evitar.

Mecanismos para tiempo real

❑ Medida del tiempo

Sistemas de **referencia de tiempo**

- ✓ El tiempo es una magnitud física fundamental cuya unidad en el SI es el **segundo**.
- ✓ Se necesitan dos tipos de medidas:
  - Intervalos de tiempo
  - Tiempo absoluto
- ✓ Para medir valores absolutos de tiempo hace falta un **sistema de referencia**.
  - Un sistema de referencia se basa en una escala de tiempo cuyo origen se denomina **época**.

Mecanismos para tiempo real

❑ Medida del tiempo

Sistemas de **referencia estándar**

- ✓ Locales (tiempo desde el arranque del sistema).
- ✓ Astronómicos:
  - Tiempo universal (UT0)
  - Tiempo solar medio en el meridiano de Greenwich (definido en 1884 (GMT- Greenwich Mean Time) y oficial hasta 1955). Se determina mediante observaciones astronómicas.
    - o1s=1/(24 x 3600)=1/86400 del día solar medio
  - Correcciones del Tiempo Universal (UT0)
    - oUT1: corrección a UT0 debida al movimiento polar
    - oUT2: corrección a UT1 debida a la variación en la velocidad de rotación de la tierra

Mecanismos para tiempo real

❑ Medida del tiempo

Sistemas de **referencia estándar**

TAI: combina la salida de aprox 400 relojes atómicos
de alta precisión distribuidos por el mundo y suministra la velocidad exacta con la que tienen que dar el tick nuestros relojes.

- ✓ Atómicos:
  - Proporcionan medidas estables y precisas
  - Definición oficial (SI) desde 1967
  - Precisión del orden de 10^{-13} (1s en 300.000 años)
- ✓ Tiempo atómico internacional (TAI)
  - Definido en 1970, basado en reloj atómico de cesio
  - Se mide en segundos con origen a las 0 horas UT del 1-1-1958
  - 1 s = 91.922.631.770 periodos de la radiación asociada a la transición hiperfina del estado base del átomo de cesio-133
  - TAI es ***cronoscópico***, es una escala de tiempo sin discontinuidades

Mecanismos para tiempo real

❑ Medida del tiempo

Sistemas de **referencia estándar**

✓ El TAI Se aparta lentamente del UT

- Debido a que la duración del día solar medio (UT0) va en aumento
- Actualmente la diferencia es de aproximadamente 37 s

✓UTC (**Universal Time Coordinated**)

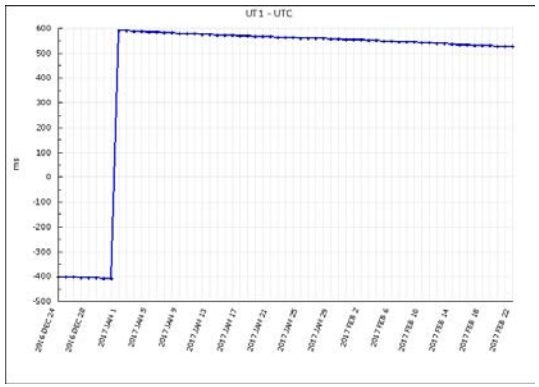
- UTC está basado en el TAI de forma que su valor difiere en menos de 0,9 segundos del UT1 (mediante la adición de ticks ocasionales)
- UTC = TAI + H, H se elige de forma que $|UT1-UTC| \leq 0{,}9$ s
- Después de añadir el último el 31-12-2016 la diferencia es de 37 s (el último minuto del 2016 tuvo 61 s).
- Esto no quiere decir que el día sea 36s más largo, significa que los días en que se añade el **segundo intercalar** tienen 86401s

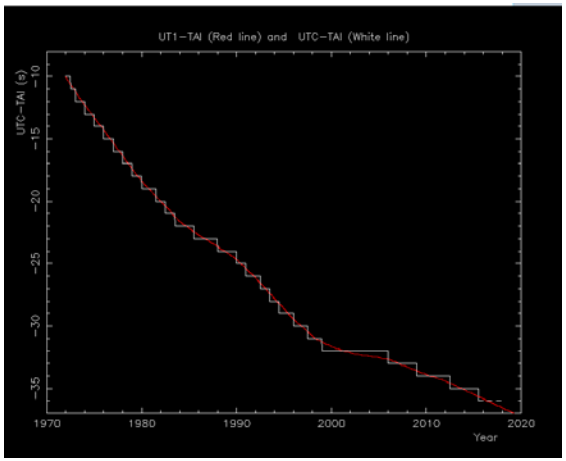
Mecanismos para tiempo real

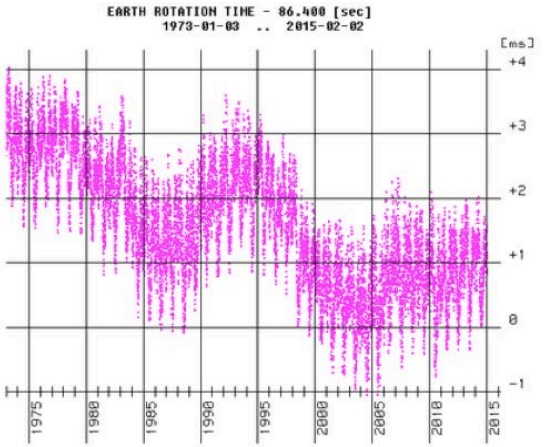
❑ Medida del tiempo

Sistemas de **referencia estándar**

✓Diferencia UT1 – UTC – TAI







Mecanismos para tiempo real

❑ Medida del tiempo

Sistemas de **referencia estándar**

✓UTC (Universal Time Coordinated)

- Se añade un segundo intercalar al UTC cuando es necesario (30 de junio o 31 de diciembre a las 24:00) (UTC no es cronoscópico)
- La hora oficial (OT o TO) en cada país se basa en el UTC corregido al uso horario y retrasos solares oficiales. TO = UTC + Z + C
- Desde 1972 se ha añadido un segundo intercalar al UTC 37 veces (la última el 31 de diciembre de 2016). Se anuncia en http://hpiers.obspm.fr/eop-pc/
- Próximo segundo intercalar: no planificado

Mecanismos para tiempo real

❑ Medida del tiempo

Sistemas de **referencia estándar**

- ✓El UTC es adecuado para la comunicación entre personas, pero presenta saltos que lo hacen inadecuado para medidas de tiempo precisas.
- ✓El TAI es una referencia más adecuada para controlar la ejecución de tareas de tiempo real
- ✓Lo que hace falta es una referencia de tiempo:
  - Estable: sin grandes variaciones a lo largo del tiempo
  - Exacta: diferencia con el TAI acotada
    - oExcepto una constante: puede tener un época (origen) cualquiera
  - Precisa: la diferencia entre dos lecturas sucesivas está acotada
  - Monótona no decreciente: sin saltos hacia atrás

Mecanismos para tiempo real

❑ Medida del tiempo

Medida del tiempo y relojes

✓La introducción de la noción del tiempo en un lenguaje de programación (o S.O.) se puede describir en términos de cuatro requisitos que se apoyan unos en otros:

- Acceso a un reloj, de forma que se pueda medir el paso del tiempo.
- Mecanismos para retardar un proceso, de forma que quede suspendido hasta un tiempo futuro.
- Programación de límites de espera, de forma que se pueda detectar y actuar en consecuencia, la no ocurrencia de algún evento en un periodo de tiempo especificado.
- Mecanismos para planificar la ejecución de los procesos (por ejemplo, prioridades).

Mecanismos para tiempo real

❑ Medida del tiempo

Medida del tiempo y relojes

Características:

✓**Estáticas** (representación del tiempo):

- **Resolución**. Mínima diferencia entre dos valores de tiempo distintos. Depende de la frecuencia del oscilador y de la forma de contabilizar impulsos.
- **Intervalos de valores**. Valores de tiempo mínimo y máximo que puede representar. Depende de la resolución y de la capacidad del contador.

✓**Dinámicas**:

- **Granularidad.** Mínimo tiempo durante el que el valor proporcionado se mantiene constante.
- **Exactitud**. Diferencia con el valor de un sistema de referencia.
- **Estabilidad**. Un reloj es estable si su frecuencia permanece constante.

Mecanismos para tiempo real

Medida del tiempo

Medida del tiempo y relojes

Ejemplos:

POSIX 1b (p.e VxWorks)

- nº entero de segundos (32 bits)
- nº entero de nanosegundos (32 bits)
- resolución: 1 ns, granularidad: <20 ms
- época: 0 h UT 1/1/1970
- intervalo: 2^{32} segundos ≈ 136 años

NTP

- coma fija con 64 bits (32+32)
- resolución: 2^{-32} segundos ≈ 232 ps
- época: 0 h UT 1/1/1990
- intervalo: 2^{32} segundos ≈ 136 años

Mecanismos para tiempo real

❑ Medida del tiempo

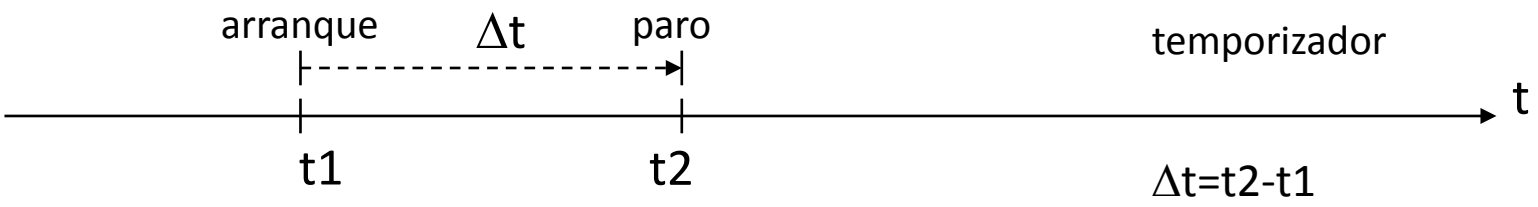
Formas de medir el tiempo en un computador

✓ Relojes

- Medida de tiempo absoluto
- Funcionan de forma continua

✓ Temporizadores

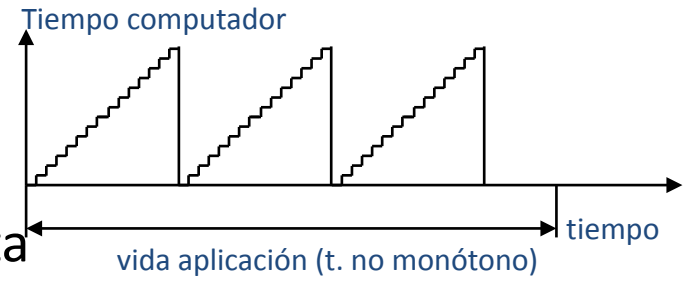
- Medida de intervalos temporales
- Arranque y paro



Mecanismos para tiempo real

Medida del tiempo

Formas de medir el tiempo en un computador



✓Unidad de tiempo: tick

✓Contador de capacidad limitada

- Desbordamiento => reinicio de la cuenta

Tiempo monótono

- El tiempo de vida de la aplicación es menor que el tiempo de desbordamiento
- Sin saltos (cronoscópico)

Tiempo no monótono

- La aplicación vive por encima del tiempo de desbordamiento
- No se dispone de tiempo absoluto (por ejemplo, no se puede mantener fecha y hora).
- No se pueden utilizar intervalos temporales mayores que el tiempo de desbordamiento del contador del reloj.

Mecanismos para tiempo real

❑ Medida del tiempo

Retardos

✓Un retardo suspende la ejecución de un proceso durante un cierto tiempo

✓Existen dos tipos de retardos:

- Retardo relativo
  - oLa ejecución se suspende durante un intervalo de tiempo relativo al instante actual
- Retardo absoluto
  - oLa ejecución se suspende hasta que se llegue a un instante determinado de tiempo absoluto.

Mecanismos para tiempo real

❑ Medida del tiempo

Retardos y activación de tareas

✓ Activación por retardo

- Los retardos permiten controlar el tiempo de activación de las tareas.
- Esta técnica no depende del hardware (hardware de propósito general, portabilidad).
- Permiten realizar **procesos periódicos**.
  - Se activan regularmente en instantes de tiempo separados por un periodo de tiempo:

```
proceso periódico
  cada T repetir
     actividad;
  fin repetir;
fin periódico;
```

Mecanismos para tiempo real

❑ Medida del tiempo

Retardos y activación de tareas

✓ Activación por retardo relativo: suspensión de la tarea

```
tarea_periodica
   begin
      while (TRUE)
         Event_Processing ();
         TaskDelay (periodo);
      end while;
    end;
```

Problemas:

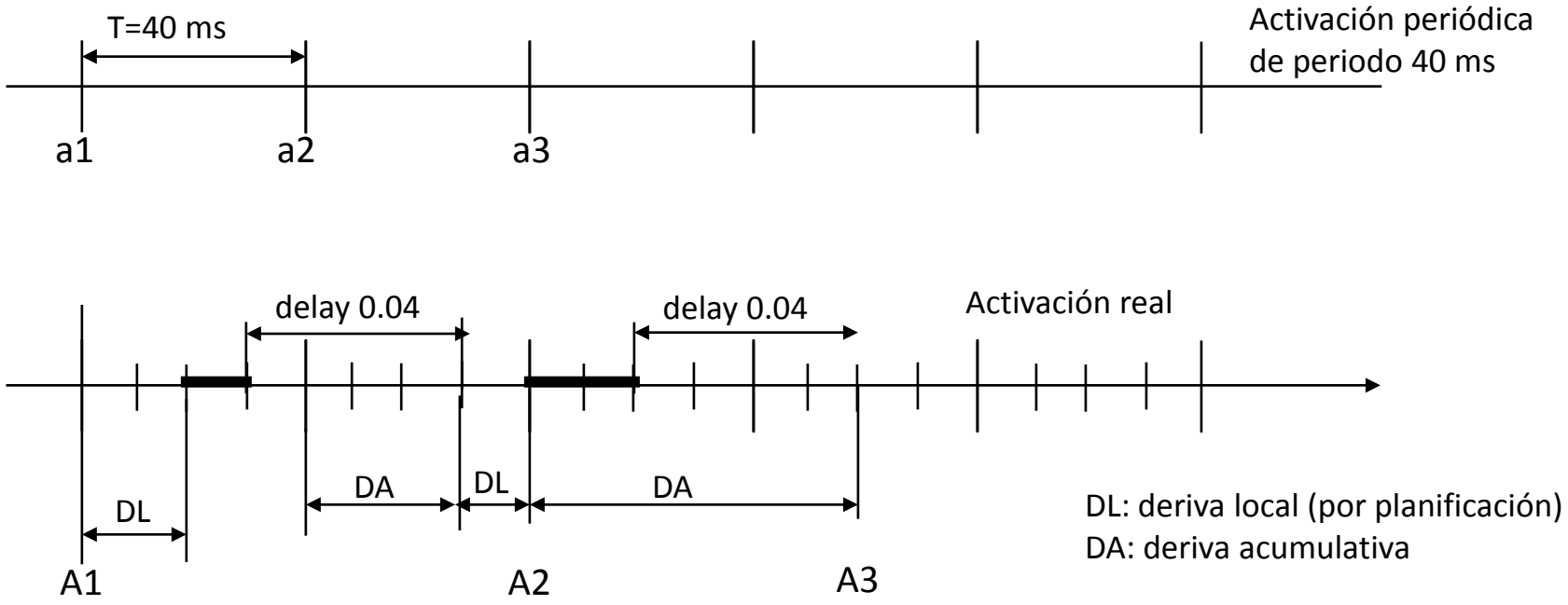
puede existir **deriva local** y **acumulativa** (esta última se puede evitar).

Mecanismos para tiempo real

❑ Medida del tiempo

Retardos y activación de tareas

✓ Deriva local y acumulativa



Mecanismos para tiempo real

❑ Medida del tiempo

Retardos y activación de tareas

✓ Retardo absoluto

```
tarea_periodica
   begin
      while (TRUE)
         Event_Processing ();
         Siguiente_Activación  = Siguiente_Activación + periodo ;
         retardo = Siguiente_Activación - Hora_Actual ();
         TaskDelay (retardo);
      end while;
   end;
```

- Elimina la deriva acumulativa (si el cálculo de retardo fuera atómico). Como no lo es, puede sufrir retrasos intermitentes (si es desalojada antes de la instrucción de retardo).
- Es mejor una instrucción de retardo absoluto (p.e. delay until de Ada) elimina la deriva acumulativa y los retrasos intermitentes del retardo relativo.

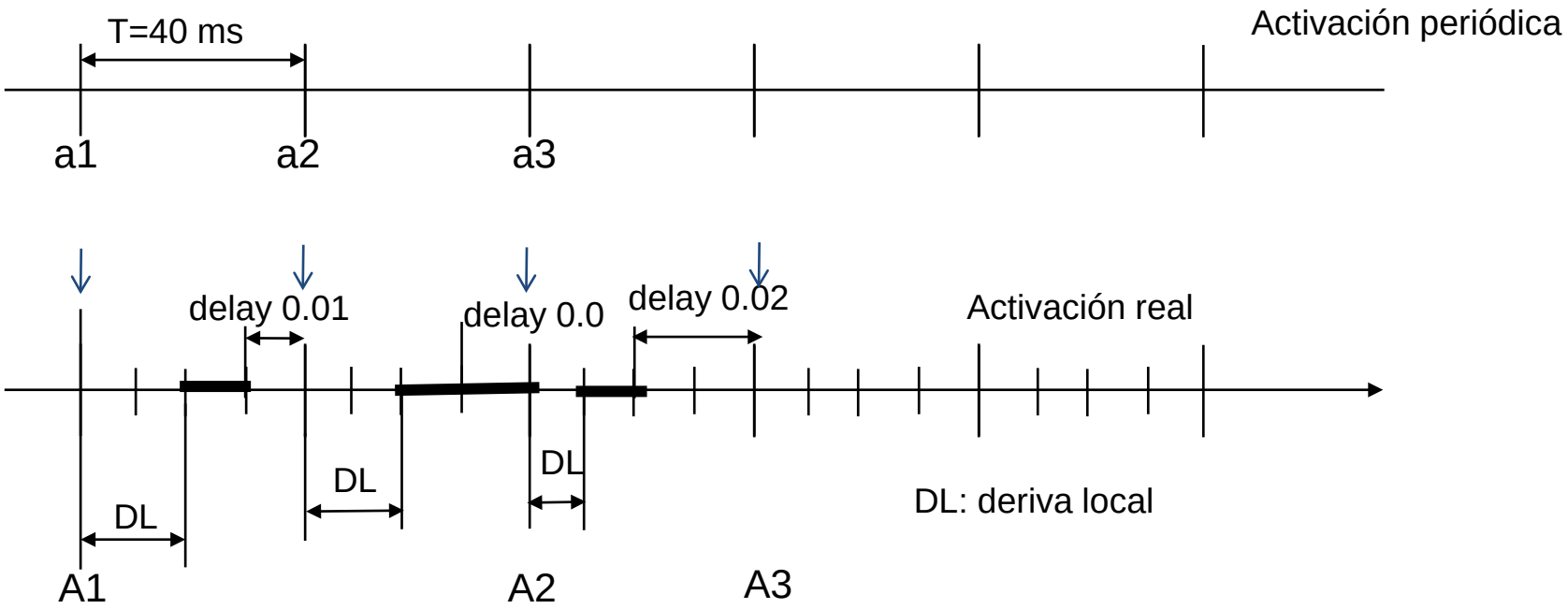
Mecanismos para tiempo real

❑ Medida del tiempo

Retardos y activación de tareas

✓ Retardo absoluto

- Pueden existir retrasos intermitentes debido a que el retraso no es atómico.

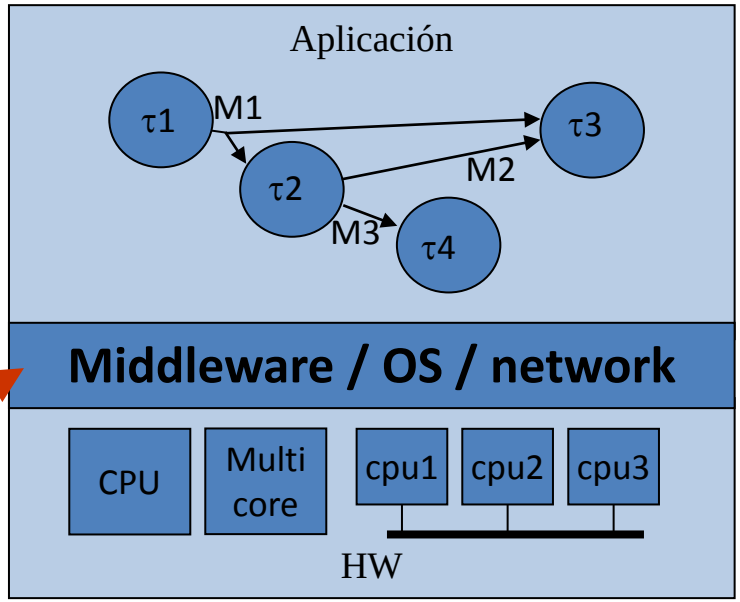


Mecanismos para tiempo real

❑ Planificación de tareas

Una aplicación es un conjunto de **tareas concurrentes** que se comunican y se sincronizan para conseguir los objetivos del sistema.

Técnicas RT



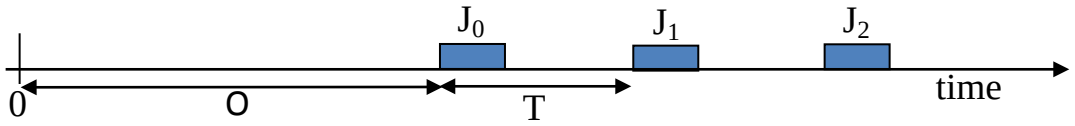
Mecanismos para tiempo real

❑ Planificación de tareas

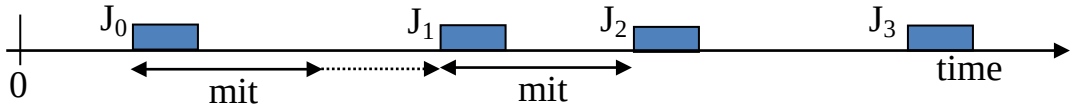
Requisitos temporales de las tareas

✓Cada tarea consume **C (WCET)** unidades de tiempo de CPU **recurrentemente** (secuencia infinita de ejecuciones).

- **Periódica:** demanda una nueva ejecución (activación) cada **T ut**



- **Esporádica:** demanda una nueva ejecución al menos **mit ut** después



- **Aperiódica:** tiempo entre demandas de ejecución estocástico

- Cada tarea tiene un **tiempo de Respuesta (R)**
- Debe acabar en su **plazo (D- Deadline)**

Mecanismos para tiempo real

❑ Planificación de tareas

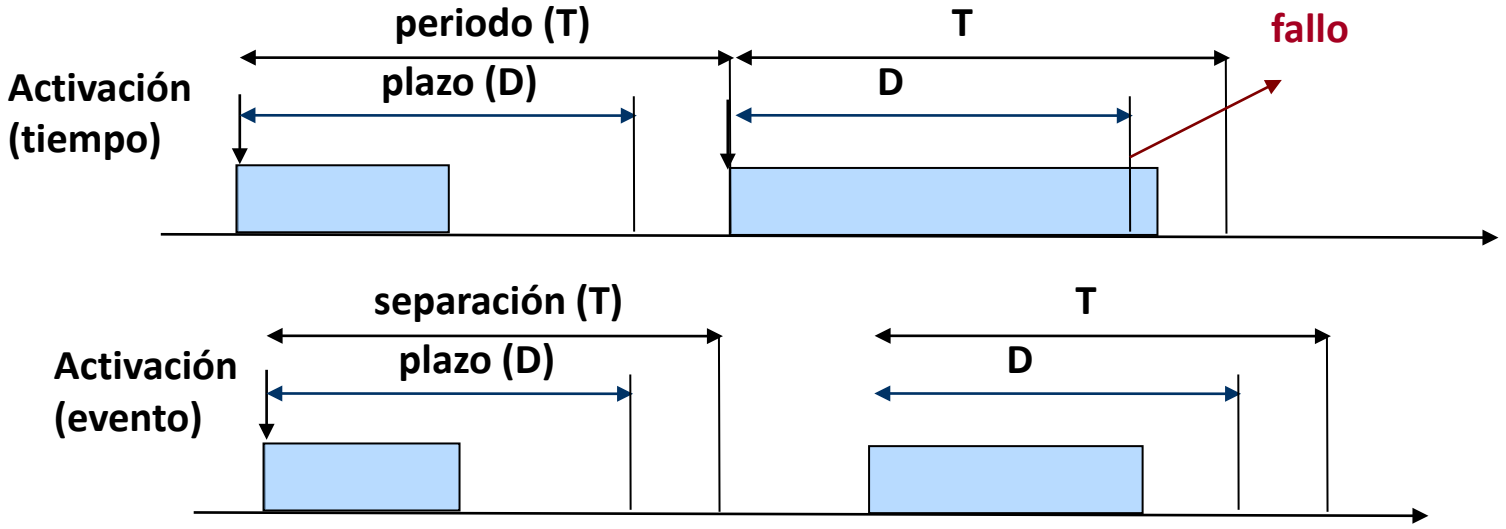
Requisitos temporales de las tareas

✓Esquema de activación:

- Tareas **periódicas**: se ejecutan a intervalos regulares.
- Tareas **esporádicas**: se ejecutan cuando ocurren determinados sucesos. Por tanto, en instantes de tiempo distribuidos irregularmente.

✓Intervalo de ejecución

- Suele venir determinado por un **plazo** relativo al instante de activación.

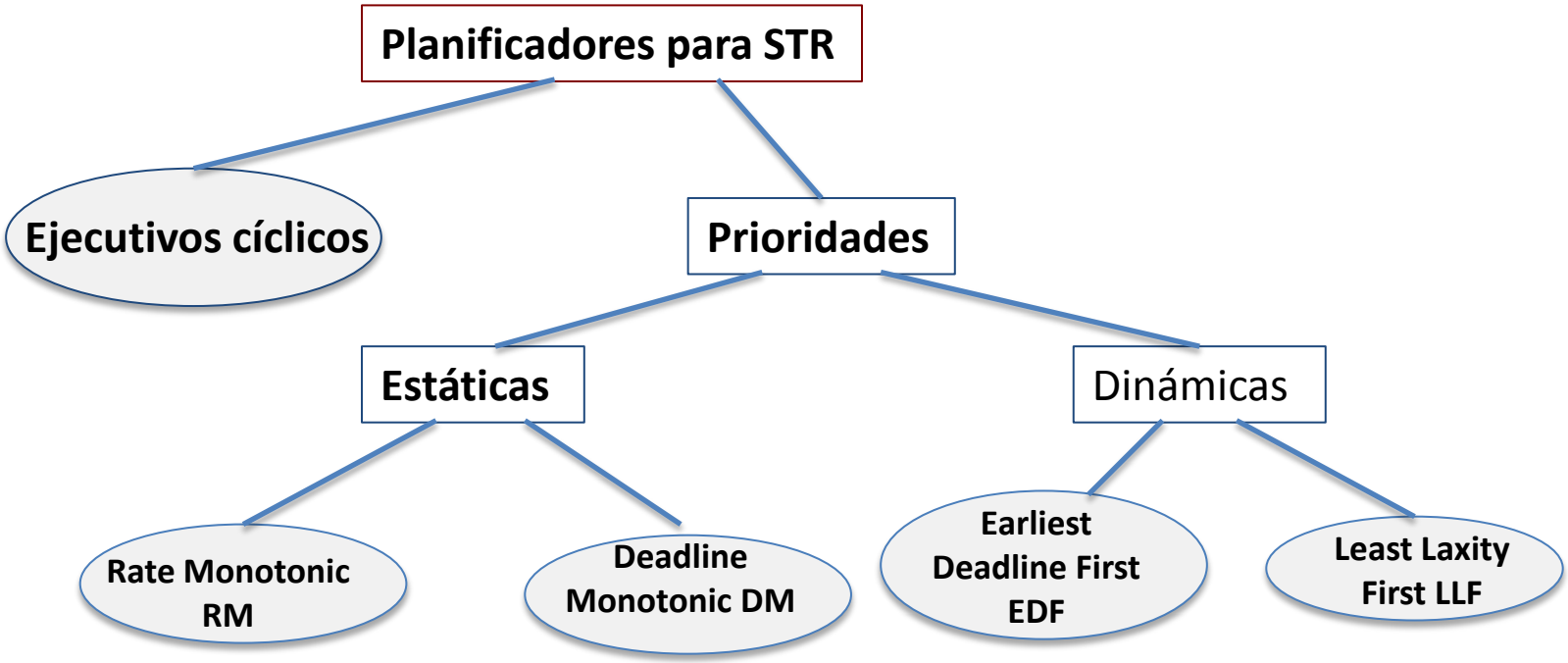


Mecanismos para tiempo real

❑ Planificación de tareas

Planificación estática y dinámica

✓ Un método muy utilizado es el de planificación estática basada en **prioridades fijas con desalojo**.

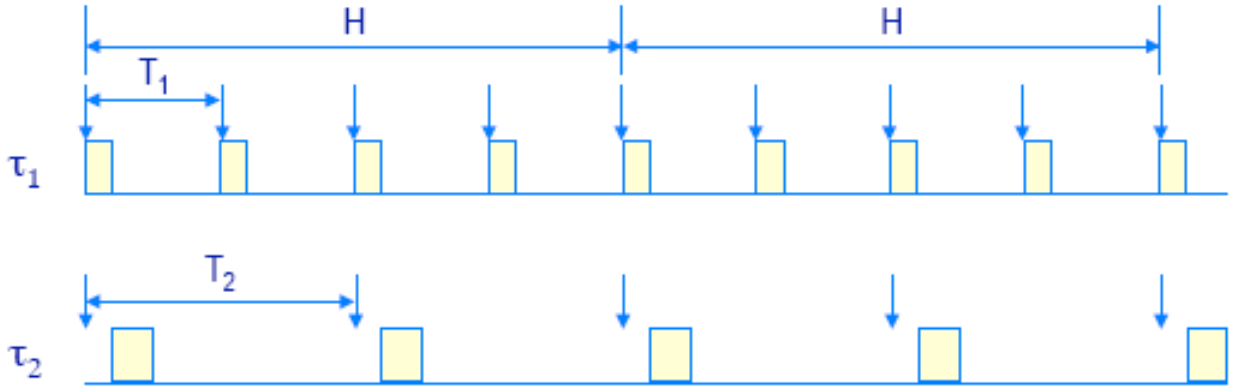


Mecanismos para tiempo real

❑ Métodos de planificación

Planificación estática cíclica

- ✓ Si todas las taras son periódicas, se puede confeccionar un plan de ejecución fijo.
- ✓ El esquema se repite cada hiperperiodo o ciclo principal.
- ✓ El ciclo principal se divide en ciclos secundarios de periodo T_s, siendo $T_M = K\,T_s$.
- ✓ En cada ciclo secundario se ejecutan las actividades correspondientes a determinadas tareas.



Mecanismos para tiempo real

❑ Métodos de planificación

Planificación estática cíclica

- ✓El plan se calcula fuera de línea.
- ✓El plan es temporalmente correcto por construcción (se comprueba que se cumplen los plazos sobre el plan de ejecución).
- ✓No tiene por qué existir concurrencia.
- ✓El ciclo secundario debe cumplir unas restricciones:

$$\begin{cases} T_s \geq \max C_i \\ \forall i : \frac{T_i}{T_s} - \left\lfloor \frac{T_s}{T_i} \right\rfloor = 0 \\ \forall i : 2T_s - mcd(T_s, T_i) \leq D_i \end{cases}$$

Desventajas:

- Pequeñas variaciones en el conjunto de tareas puede implicar un re-diseño completo.
- A veces es necesario segmentar las tareas (restricciones si comparten datos).
- No es adecuado para tareas Esporádicas/Aperiódicas

Mecanismos para tiempo real

Métodos de planificación

Planificación estática cíclica

```
procedure Cyclic_Executive is
  type Frame is mod 1;
  Index: Frame := 0;
begin
  Set_Timer (Periodic, 0.020);
  loop
    Wait_clock_Interrupt;              -- every 20 ms
    case  Index is
         when  0 => A; B;
         when  1 => A;
    end case ;
    Index := Index + 1;
  end loop ;
end Cyclic_Executive;
```

Mecanismos para tiempo real

❑ Métodos de planificación

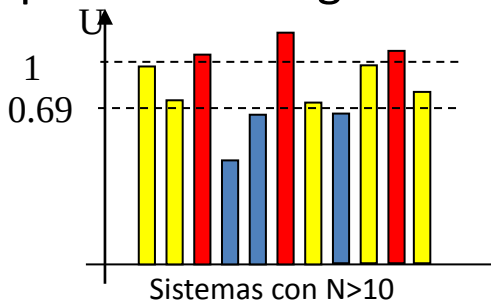
Planificación basada en prioridades fijas

- ✓ Es más adecuado cuando existen tareas esporádicas
- ✓ Garantía de plazos basada en U (utilización del sistema)
  - Si todas las tareas son periódicas, Ti=Di y no comparten datos, la asignación óptima de prioridades es RM (monótona en frecuencia).

$$U = \sum_{i=1}^{N} \frac{C_i}{T_i} \leq N\left(2^{\frac{1}{N}} - 1\right)$$

FACTOR DE UTILIZACIÓN MÍNIMA GARANTIZADA

  - Es una condición suficiente. Sistemas que no cumple la utilización mínima garantizada pueden estar garantizados.

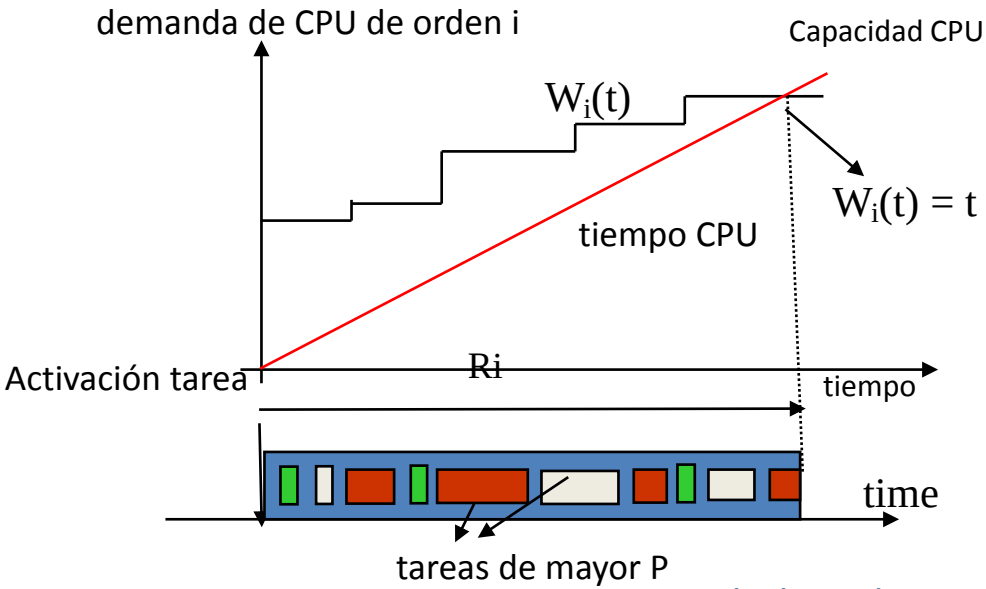


Mecanismos para tiempo real

Métodos de planificación

Planificación basada en prioridades fijas

- ✓ Garantía de plazos basada en el tiempo de respuesta
  - El tiempo de respuesta máximo de una tarea es la suma de su tiempo de cómputo y la interferencia que le producen la ejecución de tareas más prioritarias (tiempo de desalojo).
  - R_i será la solución mínima de:



$$W_i(t) = C_i + \sum_{j \varepsilon hp(i)} \left\lceil \frac{t}{T_j} \right\rceil C_j$$

donde:

$$I_i(t) = \sum_{j \varepsilon hp(i)} \left\lceil \frac{t}{T_j} \right\rceil C_j$$

es la interferencia producida por las tareas más prioritarias en el intervalo [0,t).

Mecanismos para tiempo real

❑ Métodos de planificación

Planificación basada en prioridades fijas

- ✓ La ecuación de tiempo de respuesta para la tarea i no es ni continua ni lineal.
- ✓ Se puede resolver mediante la relación de recurrencia:

$$W_i^{n+1} = C_i + \sum_{j \varepsilon hp(i)} \left\lceil \frac{W_i^n}{T_j} \right\rceil C_j$$

La sucesión de pasos es monótona no decreciente (aumenta de forma discreta cuando se activa una de mayor prioridad que i).

- ✓ La iteración siempre termina, bien cuando $W_i^{n+1} = W_i^n$ (es decir, no se ha activado ninguna de mayor P y entonces la tarea i ha acabado $R_i = W_i^n$) o bien cuando y el plazo de la tarea no está garantizado. $W_i^{n+1} > T_i = D_i$
- ✓ Converge siempre que U<1.
- ✓ La ecuación de tiempo de respuesta sigue siendo válida aunque existan tareas esporádicas (el peor caso de una esporádica es que se active con su separación mínima).
- ✓ Funciona con cualquier asignación de prioridades (si $D_i \leq T_i$ la asignación de prioridades óptima es DM, monótona en plazos).

Mecanismos para tiempo real

Métodos de planificación

Ejemplo: Sistema embarcado en un automóvil

Cuatro tareas con requisitos temporales independientes

Hiperperiodo: mcm(20,40,80)=80

tarea	C	T
Medida de velocidad: V	4	20
Control de frenado: ABS	10	40
Control de inyección I	40	80

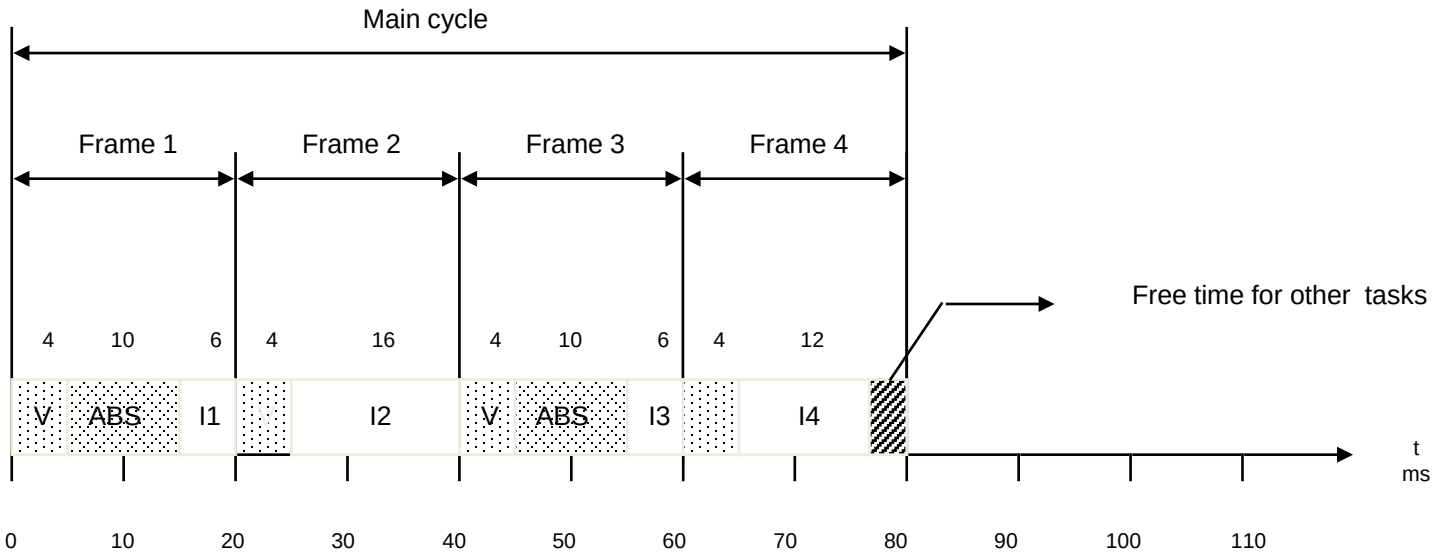
Mecanismos para tiempo real

❑ Métodos de planificación

Ejemplo: Sistema embarcado en un automóvil

✓ **Sistema síncrono (ejecutivo cíclico)**

T_M=80 Ts=20



Mecanismos para tiempo real

❑ Métodos de planificación

Ejemplo: Sistema embarcado en un automóvil

✓ **Sistema síncrono (ejecutivo cíclico)**

Implementación:

```
procedure On_board_computer is
  type Indice is range 1 .. 4;
  Frame: Indice := 1;
begin
  loop
    Wait_for_clock_tick;              -- every 20 ms
    case  Frame is
         when  1 => Speed_M; Brake_C; Injection_C1 ;

         when  2 => Speed_M; Injection_C2 ;
         when  3 => Speed_M; Brake_C; Injection_C3 ;
         when  4 => Speed_M; Injection_C4;
    end case ;
    Frame := ( Frame mod 4) + ;
  end loop ;
end On_board_computer;
```

Mecanismos para tiempo real

❑ Métodos de planificación

Ejemplo: Sistema embarcado en un automóvil

✓ **Sistema asíncrono**

Implementación:

task Medida_Velocidad

```
loop
  Medir_Velocidad;
  siguiente := siguiente + 0.02;
  delay until siguiente;
end loop;
```

task Control_Frenado

```
loop
  Controlar_Frenado;
  siguiente := siguiente+0.04;
  delay until siguiente;
end loop;
```

task Control_Inyeccion

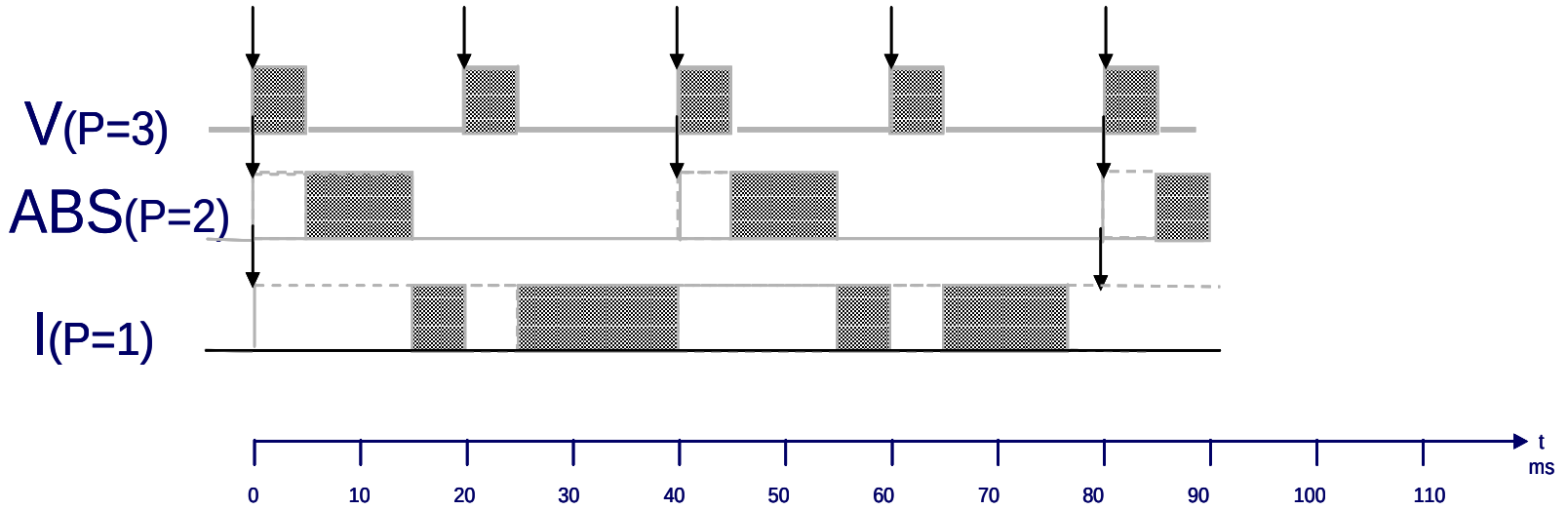
```
loop
  Controlarl_Inyeccion;
  siguiente := siguiente + 0.08;
  delay until siguiente;
end loop;
```

Mecanismos para tiempo real

❑ Métodos de planificación

Ejemplo: Sistema embarcado en un automóvil

✓ **Sistema asíncrono** Implementación



- Cada tarea tiene una prioridad fija
- El planificador multiplexa la CPU entre las tareas
- En cada instante se ejecuta la tarea de mayor prioridad

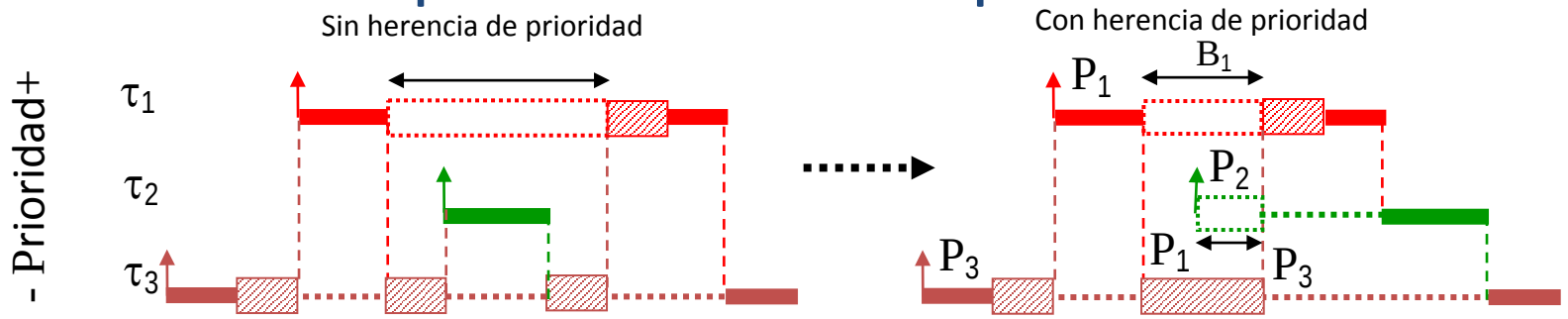
Mecanismos para tiempo real

❑ Métodos de planificación

Planificación basada en prioridades fijas

✓ Inversión de prioridad:

- Cuando las tareas comparten recursos, se produce el fenómeno de **inversión de prioridad**.
- Sin embargo, es fundamental limitar y cuantificar su impacto en el caso peor de forma que se pueda analizar la garantía de plazos del conjunto de tareas.
- Para ello son necesarias primitivas de sincronización que permitan acotar el máximo bloqueo que cada tarea puede sufrir, por ejemplo **semáforos con protocolo de herencia de prioridad**.



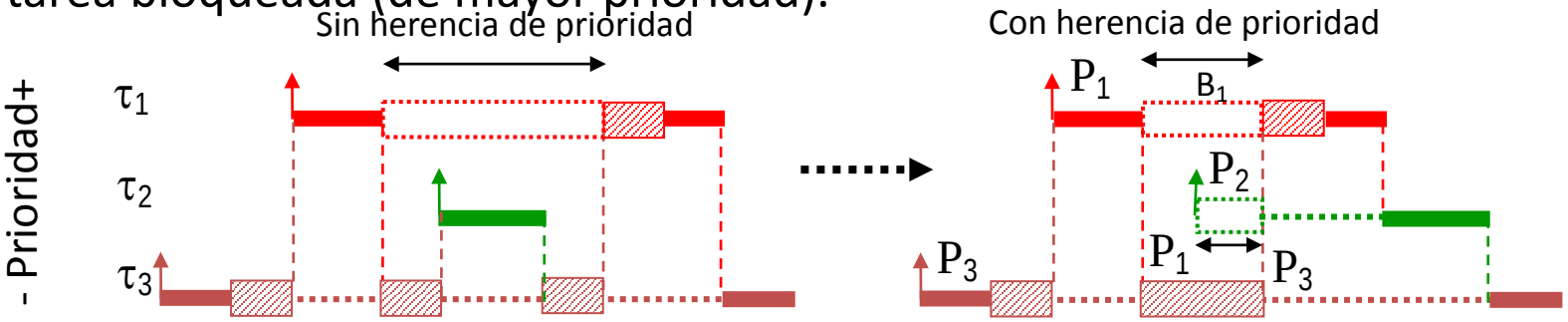
Mecanismos para tiempo real

❑ Métodos de planificación

Planificación basada en prioridades fijas

✓ Inversión de prioridad:

- La tarea bloqueante (de menor prioridad) hereda la prioridad de la tarea bloqueada (de mayor prioridad).



- Una tarea puede ser bloqueada por cualquier tarea de menor prioridad:
  - Con la que comparte un recurso
  - Que puede bloquear a una tarea de mayor prioridad.
  - Cada tarea puede bloquear a otra sólo una vez.
  - Cada tarea puede bloquearse sólo una vez en cada semáforo.

Mecanismos para tiempo real

❑ Métodos de planificación

Planificación basada en prioridades fijas

✓ Inversión de prioridad:

- Cálculo del bloqueo:

$$B_i = \sum_{k=1}^{K} utilización\ (k,i)\ C(k)$$

- K es el número de recursos
- Utilización(k,i)= 1 si el recurso k es bloqueante. Un recurso es bloqueante para la tarea i si se utiliza por:
- al menos una tarea de prioridad menor que la de la tarea i.
- al menos una tarea de prioridad mayor o igual que la de la tarea i.
- Utilización(k,i)= 0 en otro caso
- C(k): peor caso de acceso al recurso K (por tareas de menor P)

Mecanismos para tiempo real

Métodos de planificación

Cálculo del tiempo de respuesta

- ✓ Si pueden existir bloqueos, la ecuación que verifica el tiempo de respuesta es:

$$t = C_i + I_i(t) + B_i = W_i(t)$$

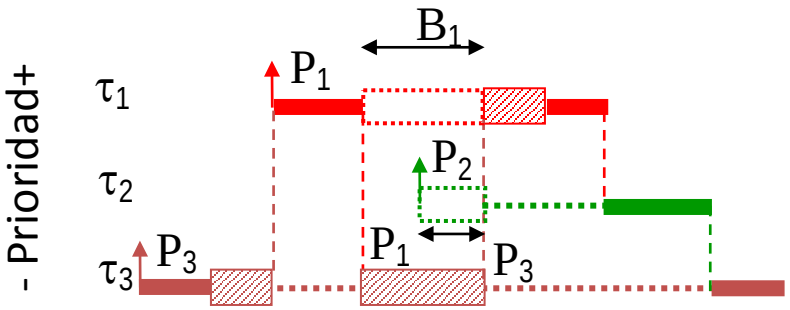
- ✓ $W_i(t)$: Carga o demanda aumentada de orden i de tiempo de procesador en el intervalo [0,t)
- ✓ La solución se obtiene mediante la relación de recurrencia:

$$W_i^{n+1} = C_i + B_i + \sum_{j \varepsilon\, hp(i)} \left\lceil \frac{W_i^n}{T_j} \right\rceil . C_j$$

Mecanismos para tiempo real

Métodos de planificación

Ejemplo: Sistema de tareas que comparten 3 recursos



e.g	C_i	T_i	B_i
τ_1	5	30	17
τ_2	15	60	13
τ_3	20	80	6
τ_4	20	100	0

←

	S_1	S_2	S_3
τ_1	1	2	0
τ_2	0	9	3
τ_3	8	7	0
τ_4	6	5	4

Indice

- ❑ Sistemas Empotrados y de Tiempo Real
- ❑ Principales características
- ❑ Mecanismos para tiempo real
- ☑ **¿Qué es imprescindible recordar?**

Qué es imprescindible recordar

❑ Los sistemas de tiempo real se caracterizan por:

- ✓ **Interactuar** repetidamente **con el entorno**, bien en instantes de tiempo definidos por un reloj o en respuesta a eventos.
- ✓ Responder a eventos que ocurren **simultáneamente**.
- ✓ El tiempo es **significativo**: Además de dar respuestas lógicamente correctas, deben llegar a tiempo, definido por el **plazo**.
- ✓ Si el sistema es **crítico** no es admisible perder plazos de respuesta.

Qué es imprescindible recordar

❑ Mecanismos en los que se apoyan los STR:

- ✓ **Concurrencia** (*threads* y mecanismos de acceso en exclusión mutua)
- ✓ Mecanismos para **gestionar el tiempo**:
  - Medida del tiempo mediante relojes, retardos, limitación del tiempo de espera, especificación de los requisitos temporales:
    - o Esquema de activación (periódico, esporádico)
    - o Plazo de terminación
    - o Latencia de activación
    - o Tiempo de cómputo máximo

Qué es imprescindible recordar

❑ Mecanismos en los que se apoyan los STR:

- ✓ Mecanismos para **planificar la ejecución de las tareas**:
  - Planificadores estáticos:
    - Arquitectura síncrona (ejecutivos cíclicos):
      - Poco flexible, de bajo nivel aunque temporalmente correcto por construcción.
    - Arquitectura asíncrona.
      - Un método muy utilizado es la planificación basada en prioridades fijas.
      - Las prioridades se asignan por orden de periodos, plazos o de forma arbitraria.
      - Se pueden analizar tareas con interacción si se usa un protocolo de herencia o techo de prioridad.

CUANDO EL TIEMPO SE HACE REAL !!!

En estas transparencias se ha utilizado material de colegas y otras universidades

- Transparencias del Profesor Juan Antonio de la Puente (UPM)
- Transparencias del Profesor Alfons Crespo (UPV)
- Transparencias del Profesor Luis Almeida (Univ. De Porto)
- De otras universidades
- Capítulo 15 del libro:

  A. Burns, A. Wellings “Real-Time Systems and Programming Languages” 4ª edición 2009

Gracias !!