

Sistemas gobernados por tiempo (TDS) Conceptos

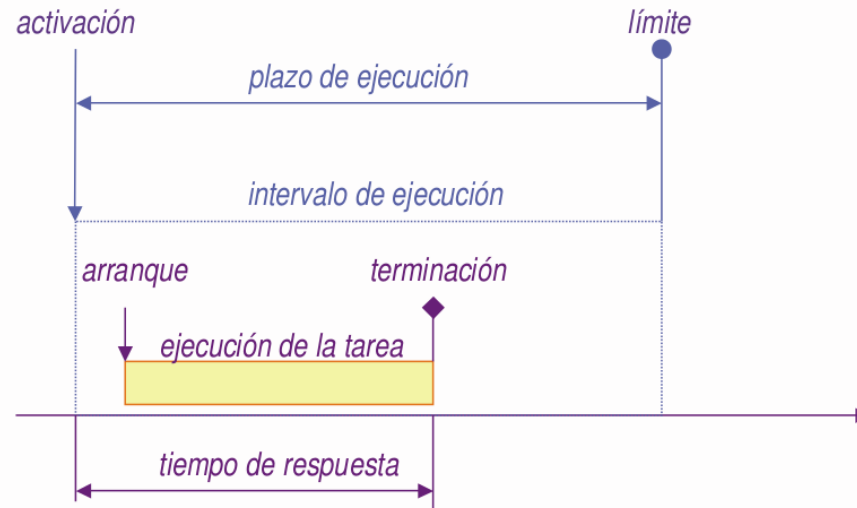
Agenda.

- Sistemas en tiempo real.
- Sistemas embebidos Confiables.
- Arquitectura TDS vs. EDS.
- Ticks de Sistema.
- Planificador TDS.
- Modos de sistema.
- Chequeos en tiempo de ejecución.
- Modelo de tareas cíclico.
- Hiperperíodo.
- Planificación estática.
- Factor de utilización.

Sistemas en tiempo real.

- Un sistema en tiempo real se puede definir como aquel en el que el resultado de una operación tiene que ser lógicamente correcta y dada en un tiempo acotado y predecible .

Necesitamos que el tiempo de respuesta $<$ plazo de ejecución.
Si no lo logro esa ejecución es invalida y el sistema falla.



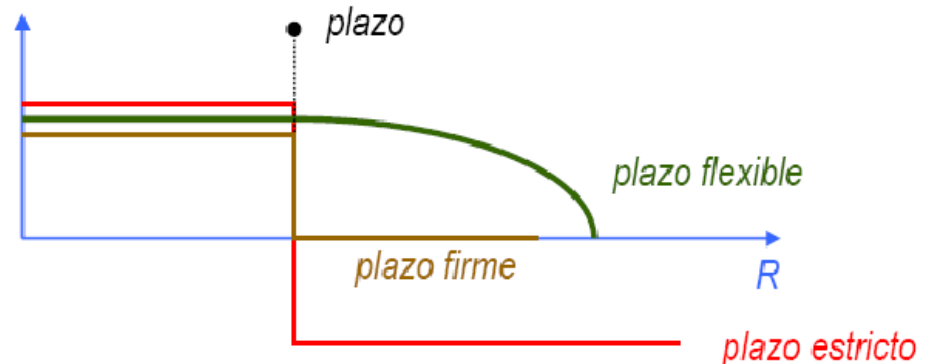
Sistemas en tiempo real.

- Se definen tres tipos de sistemas de tiempo real:

- **Estrictos (hard real time).** Si no se alcanza la respuesta en el tiempo determinado se considera que el sistema **falla**.
valor

- **Flexible (soft real time).** Pueden no alcanzar el tiempo de generación de la salida ocasionalmente. El valor de la respuesta decrece con el tiempo (adquisición de datos).

- **Firme (firm real time).** Se puede no alcanzar el tiempo de generación de salida ocasionalmente. Una respuesta tardía no tiene valor (ej: sistema multimedia)



Características de los sistemas en tiempo real.

■ Seguridad y fiabilidad

□ Sistemas críticos: fallos con consecuencias graves

- pérdida de vidas humanas
- pérdidas económicas
- daños medioambientales

activacion de airbag,
sistemas nucleares,ect.

■ Determinismo temporal

□ Acciones en intervalos de tiempo determinados

□ Es fundamental que el comportamiento temporal sea determinista

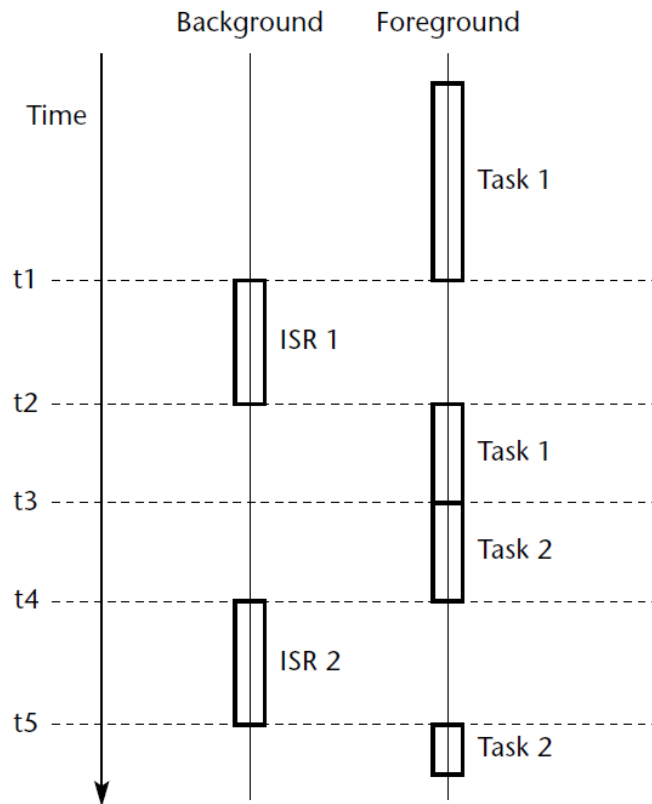
- No hay que confundirlo con la necesidad de que sea eficiente
- El sistema debe responder correctamente en todas las situaciones
- Hay que prever el comportamiento en el peor caso posible

necesito siempre saber como responde mi sistema

Sistemas de tiempo real. Resumen.

- Los sistemas de tiempo real interaccionan con su entorno y ejecutan sus acciones dentro de intervalos de tiempo determinados
- Tienen requisitos muy exigentes
 - Tamaño y complejidad
 - Concurrencia resolver varias cosas a la vez
 - Interfaces de hardware específicas
 - Fiabilidad y seguridad
 - Determinismo temporal
- Hay dos clases principales de requisitos de tiempo real
 - Tiempo real estricto (hard real-time)
 - Tiempo real flexible (soft real-time)

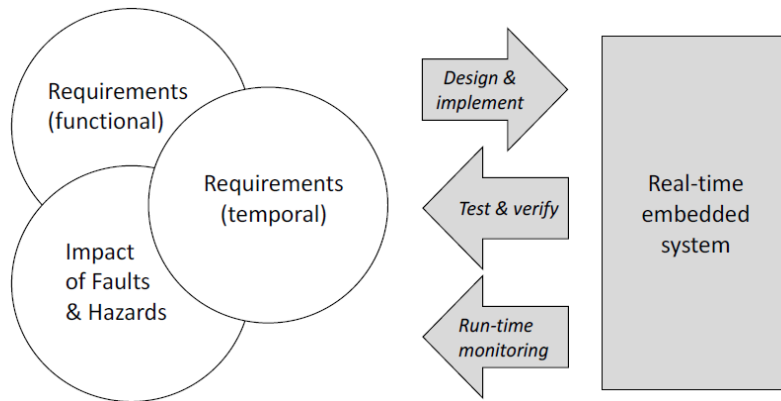
Sistemas embebidos gobernados por eventos (EDS).



- Los EDS son sistemas embebidos a los que cada evento a manejar (teclas, periféricos, temporizaciones, etc). Son manejados por interrupciones.
- Los EDS son sistemas de mínima latencia pero de previsibilidad compleja ya que las interrupciones pueden aparecer en cualquier momento (por definición las interrupciones son y se consideran asincrónicas).

se busca minimizar la latencia entre que hay una interrupción y que se la atiende

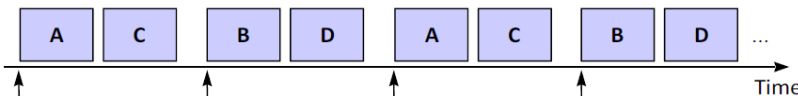
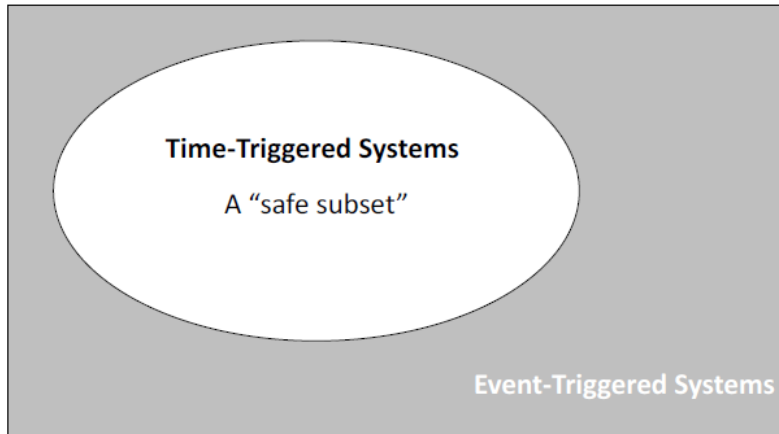
Sistemas embebidos confiables



podemos estimar el peor tiempo de respuesta y este peor caso debe cumplir los requisitos.

- La ingeniería de un sistema confiable requiere la definición, verificación y validación de **requerimientos funcionales y temporales y el impacto de las fallas del sistema.**
- En este tipo de sistemas es de vital importancia el **procesamiento determinístico** que implica que se pueda garantizar que una actividad particular del sistema siempre se completará dentro del mismo intervalo de tiempo.
- Un sistema confiable que no cumple los requerimientos (por ejemplo que responda más lentamente de lo especificado) **no es útil para esa función.**

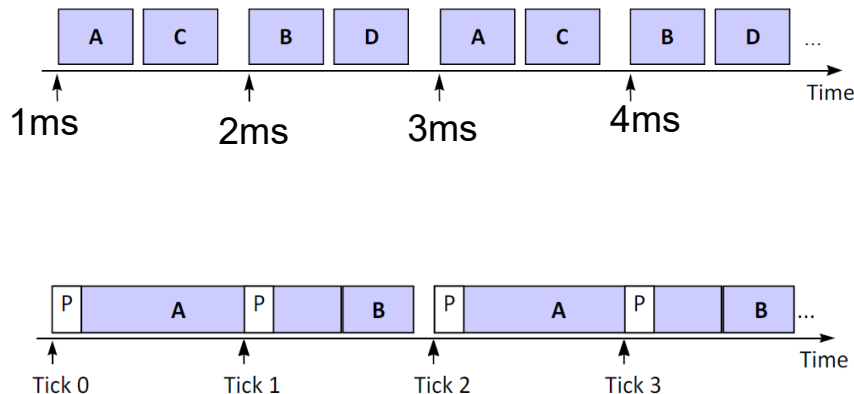
Sistemas embebidos gobernados por tiempo (TDS)



- Los sistemas gobernados por tiempo sólo utilizan una única interrupción (de temporización) para ejecutar un planificador.
- El planificador suele ser estático y cooperativo (las tareas están definidas por el/la desarrollador/a y no se interrumpen entre ellas)
- Estas condiciones hacen que sean sistemas de **alta previsibilidad temporal y por lo tanto verificables y validables**.
- Debido a las restricciones impuestas por los sistemas TDS **es posible modelar el comportamiento del sistema de manera precisa y determinar si se cumplen los requerimientos temporales**.
- Su aplicación típica es en sistemas que requieren alta confiabilidad (safety systems)

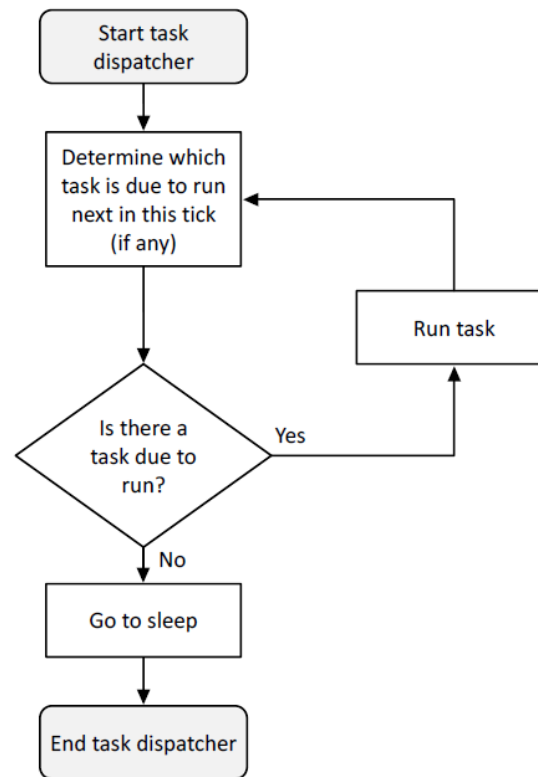
Sistemas TDS

si usamos un tick de 1ms:



- Un sistema TDS ejecuta una **lista de tareas predeterminadas**.
- El sistema TDS se va a regir por el “tick de sistema” que se implementará con un timer del microcontrolador. Valores usuales de este “tick” pueden ser:
 - ☐ 1ms
 - ☐ 100μs
 - ☐ 25ms.
- La elección del tick de sistema va a ser una relación de compromiso entre el tiempo de respuesta y el tiempo que tarda el planificador en cambiar de una tarea a otra.

Planificador TDS Cooperativo.



Modos del sistema principales

Modo normal

- Es el modo por defecto donde el sistema está completamente operativo y se alcanzan todos los requerimientos esperados sin que el sistema cause daños o riesgos.

Modo de falla controlada.

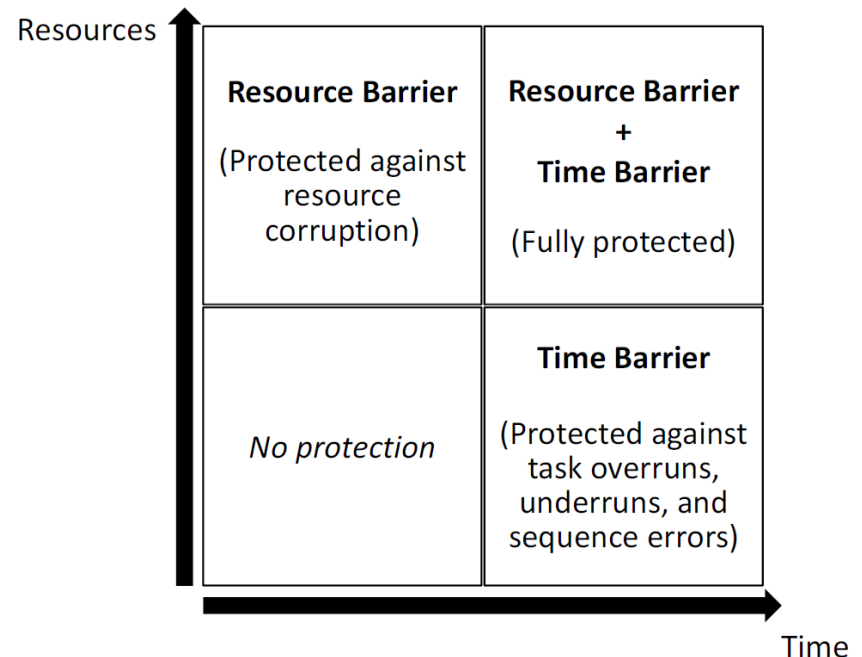
- Es el modo en el que el sistema detecta alguna condición anómala y hay que apagar el sistema llevándolo a una condición de que minimice el daño o no lo produzca.

yo de antemano se cuanto va a tardar cada tarea

Chequeos en tiempo de ejecución

- Construir un sistema manejado por tiempo involucra:

- Modelar el sistema como una lista de tareas y su verificación.
- Seleccionar el tipo de planificador para la aplicación (vamos a plantear cooperativo)
- Generar chequeos en tiempo real para comprobar que el hardware funcione de acuerdo a requerimientos



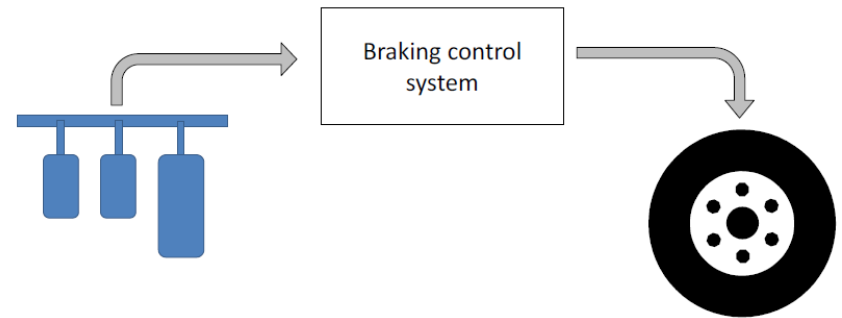
Chequeos en tiempo de ejecución.

- Los chequeos en tiempo de ejecución son fundamentales para los sistemas de alta disponibilidad o confiabilidad.
 - **Chequeos en el arranque.** (*Power On Self Test*). El fin de este chequeo es verificar la correcta configuración del sistema.
 - **Temporizador de Watchdog.** Es un temporizador que al alcanzar su conteo máximo resetea al procesador. Se usa reseteándolo periódicamente. En caso de que haya algún problema este temporizador reinicia el sistema.
 - **Monitor de tareas.** El monitor es un temporizador (puede ser uno de uso general) que verifica que las tareas se ejecuten al menos un tiempo mínimo (BCET) y menos de un tiempo máximo estipulado (WCET).

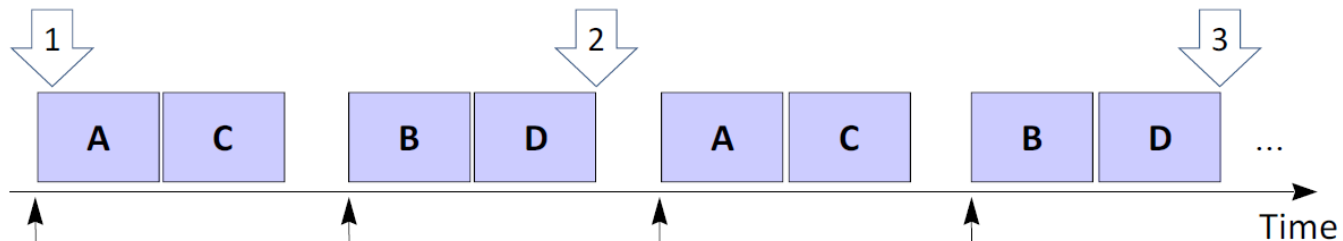
Es un periférico. cuando el temporizador del WDG se agota se resetea el procesador. Si una tarea se traba el WDG reseteará el procesador y arranca de vuelta.

Tiempo de respuesta

- Independientemente de la arquitectura interna del sistema todos los sistemas en tiempo real necesitan responder a eventos.
- Una consideración muy importante es el tiempo que tarda el sistema desde que sucede un evento hasta que responde al mismo.

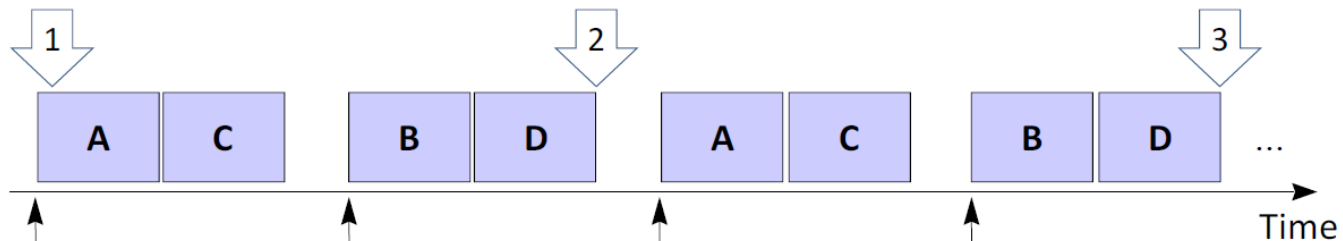


Tiempo de respuesta



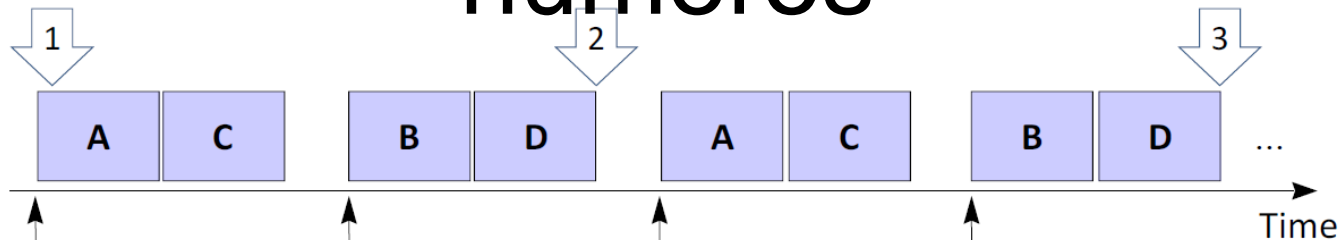
- Para este ejemplo, vamos a considerar que la tarea “A” monitorea el pedal y la tarea “D” actúa sobre el pedal de freno.
- Suponemos que en 1 se detecta que se presionó el pedal y que 2 ya se actuó el freno.
- Generalmente se suele tomar el caso de que el evento se detecta en el período siguiente.

Tiempo de respuesta



- Para sistemas estrictamente periódicos, se va a considerar que el tiempo de que va a tardar la tarea “A” en detectar un evento es:
 - $\text{Periodo}(A) + \text{WCET}(A)$.
- Para este sistema y suponiendo que la de la figura es la lista completa de tareas:
 - $\text{Periodo}(A) + \text{WCET}(A) + \text{Tick} + \text{WCET}(B) + \text{WCET}(D)$

Tiempo de respuesta. Poniendo números



- Si suponemos que se requiere un tiempo de respuesta de 50ms con los siguientes datos:
 - ☐ Tick de sistema 10ms
 - ☐ WCET(A) = 4ms
 - ☐ WCET(B) = 4ms
 - ☐ WCET(D) = 5ms
- ¿Cumple este sistema con el tiempo requerido?

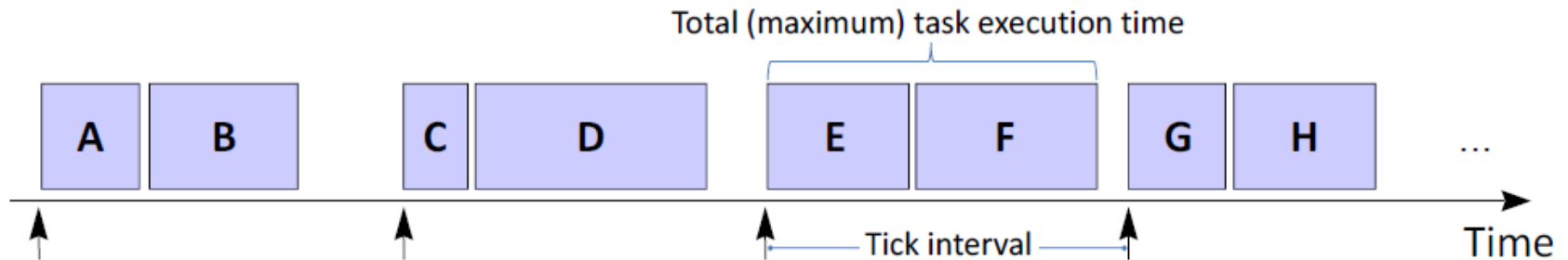
Modelando la carga de CPU

Task	Period (ms)	WCET (ms)	CPU load (%)
A	5	2	40%
B	10	4	40%
C	20	3	15%

95%

- La carga de CPU es un parámetro fundamental para cualquier embebido, ya que describe el uso del tiempo de sistema que es un recurso que va a determinar fuertemente la selección de un procesador.
- La carga de CPU que impone una tarea será el cociente entre el peor tiempo de ejecución de cada tarea (WCET) y el período de la misma.
- La suma de la carga de todas las tareas es la carga del sistema que para poder ser planificable debe ser menor al 100%.
- Este simple chequeo, lo podemos considerar un chequeo “preliminar”, pero no da detalle por tick de sistema.

Carga de CPU



- Para tener más información de la carga de CPU es necesario analizar cada tick del hiperperíodo.
- $\text{Carga de CPU}_{(\text{por tick})} = \text{WCET} / \text{tiempo de Tick}$

Carga de CPU. Ejemplo.

[Tick 0]	[Tick 7]	[Tick 12]
Task A	Task C	Task A
Task B	Task C	Task B
Task C	[Tick 8]	Task C
[Tick 1]	Task A	[Tick 13]
[Tick 2]	Task B	[Tick 14]
Task B	[Tick 9]	Task B
[Tick 3]	Task C	Task C
Task C	[Tick 10]	Task C
[Tick 4]	Task B	[Tick 15]
Task A	Task C	[Tick 16]
Task B	Task C	Task A
[Tick 5]	Task C	Task B
Task C	[Tick 11]	[Tick 17]
[Tick 6]		Task C
Task B		[Tick 18]
		Task B
		[Tick 19]

- Dada la lista de tareas de la izquierda, determine la máxima carga de CPU y la carga media de CPU.
- Suponga:
 - $WCET_A = 9ms$
 - $WCET_B = 3ms$
 - $WCET_C = 5ms$
 - Tick = 20ms
 - Carga del planificador 2%

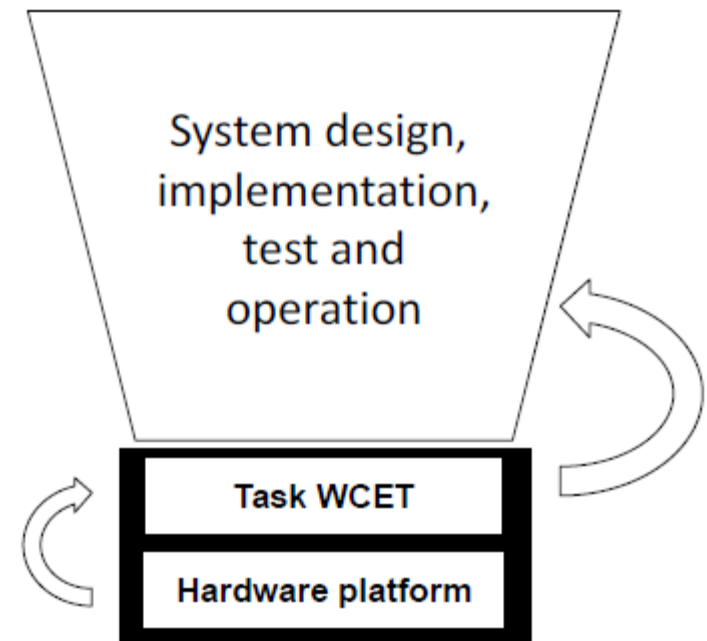
la suma de todos los WCET de cada tick tiene que ser menor a 20ms

Por ejemplo en el tick 10 tengo 18ms que es el 90% de carga del CPU, a eso le sumo un 2% del planificador.

si no cumpla el tiempo no se va a romper pero nos va a correr el tiempo de ejecución del próximo tick

¿Cómo conocer el WCET y BCET?

- El peor caso de tiempo de ejecución (WCET) de cada tarea es fundamental para el modelado de un sistema.
- Conocer el mejor caso de tiempo de ejecución (BCET) también ayuda a monitorear el sistema. Un BCET de 0 probablemente indique que una tarea no esté haciendo nada.
- Los tiempos de ejecución es complicado modelarlos (ya que implica un conocimiento detallado del compilador y el hardware) por lo que para nuestra aproximación los mediremos



DWT

- Los Cortex M3 tienen una unidad llamada Data Watchpoint y Trace que tiene hardware dedicado para el “profiling” o análisis de rendimiento del sistema implementado.
- La unidad DWT tiene cuatro comparadores y el contador de ciclos de programa CYCCNT que es un contador ascendente de 32bits que se incrementa cada ciclo de reloj.
- El CYCCNT se puede leer y escribir en cualquier momento y se habilita a través del registro DWT_CTRL

```
//Inicializo el Contador de Ciclos de reloj  
DWT->CTRL |=DWT_CTRL_CYCCNTENA_Msk;  
DWT->CYCCNT=0;
```

voy a iniciar una tarea y pongo este contador en 0. Cuando termina esta tarea leo el contador y eso me da la cantidad de ciclos de reloj que pasaron desde que inicie.

Manos a la obra

- Tomar el ejemplo de un led parpadeante (puede ser https://gitlab.frba.utn.edu.ar/jalarcon/led_tds_td2.git o cualquier otro). Medir el WCET y la carga del procesador.
- Una vez calculado, cambiar la interrupción del SysTick a 100 μ s, luego 10 μ s y recalcular.

Modelos de tareas

Un modelo de tareas especifica las características de las tareas de un sistema de tiempo real

Se restringen para poder analizar el sistema y garantizar los requisitos temporales

Ejemplos:

Sólo tareas periódicas independientes

Tareas periódicas y esporádicas independientes

Tareas con comunicación y sincronización

Tareas estáticas o dinámicas

Comenzar con modelos sencillos: tareas periódicas independientes

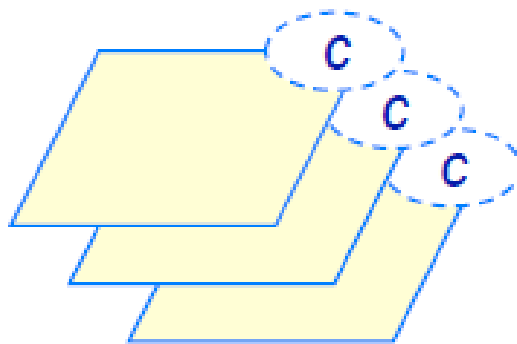
Modelo de Tareas Cíclico

Hay muchos sistemas de tiempo real que sólo tienen tareas periódicas

Son más fáciles de construir

Su comportamiento está completamente determinado

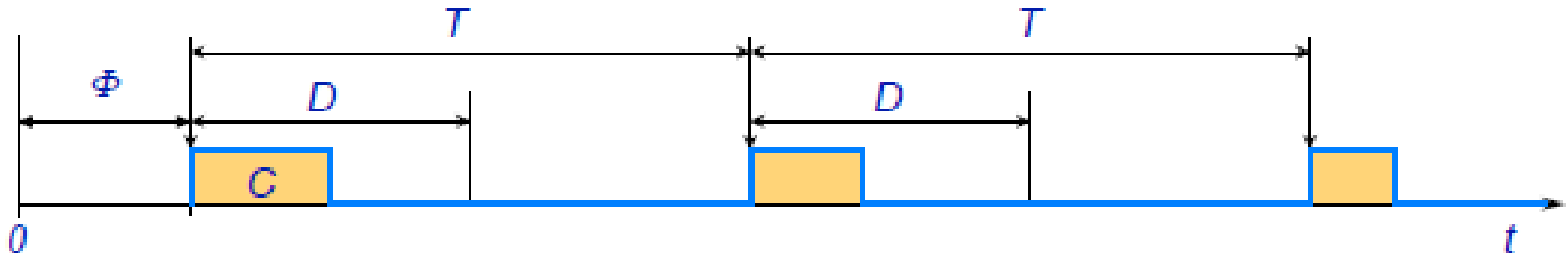
Inicialmente consideramos que no hay comunicación entre tareas (tareas independientes)



Parámetros Temporales

Una tarea periódica se define por sus parámetros (Φ , T , C , D)

- Φ es la fase
- T es el período de activación de la tarea (Minimum time between process releases - process period)
- C es su tiempo de procesamiento en el peor caso (Worst-case computation time -WCET- of the process)
- D es el plazo de respuesta relativo a la activación (Deadline of the process)



Elegir el tick de sistema

Task ID	Period (ms)	WCET (μ s)	Offset (ticks)
A	1	200	0
B	2	300	0
C	2	200	1
D	2	150	1
E	3	200	11
F	5	100	13
G	7	50	25

- Dada una lista de tareas el tick de sistema lo va a dar el mayor divisor común entre los períodos de las diferentes tareas.
- En la tabla de la izquierda es de 1ms.

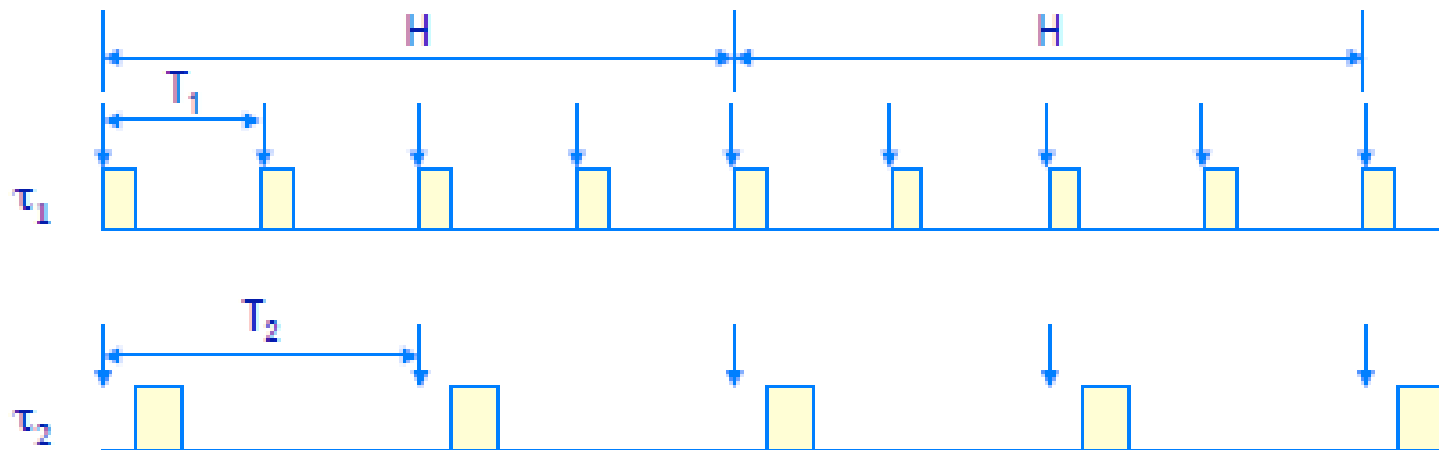
Hiperperíodo

o ciclo principal

En un sistema formado únicamente por tareas periódicas con períodos T_i , $i = 1 \dots N$, el comportamiento global se repite con un período:

$$H = \text{mcm}(T_i)$$

H es el **hiperperíodo** del sistema



Planificación estática

Si todas las tareas son periódicas, se puede confeccionar un **plan de ejecución** fijo

Se trata de un esquema que se repite cada:

$$\mathbf{TM} = \text{mcm} (\mathbf{Ti}) \text{ ciclo principal (hiperperíodo del sistema)}$$

El ciclo principal se divide en **ciclos secundarios**, con período:

$$\mathbf{TS} (\mathbf{TM} = K \mathbf{TS})$$

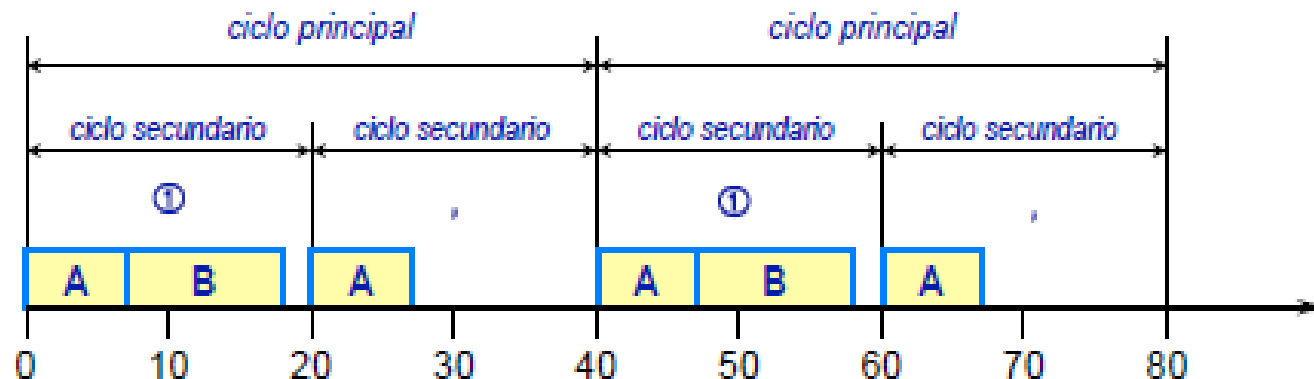
En cada ciclo secundario se ejecutan las actividades correspondientes a determinadas tareas

Planificación Estática: Ejemplo 1



la tarea A aparece cada 20 ms y la B cada 40 ms y hay un tick cada 20 ms

Plan cíclico: $T_M = 40$ ms; $T_S = 20$ ms



Planificación cíclica estática

- La principal ventaja de los sistemas gobernados por tiempo es que se puede determinar la planificación de las tareas del sistema de manera determinística. Para todas las condiciones del sistema.

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

$$T_M = mcm(T_i)$$

$$T_S = mcd(T_i)$$

- Para generar la planificación del sistema se necesita:
 - El período de activación de cada tarea (asumiendo un plazo de terminación igual al período)
 - El peor tiempo de ejecución de cada tarea.
- Se calcula el factor de carga de CPU (U), debe ser menor a uno para que el sistema sea planificable.
- El tick de sistema se va a calcular como el **máximo común divisor de todos los períodos de las tareas**.
- El hiperperíodo se va a calcular como **el mínimo común múltiplo de todos los períodos de las tareas**.
- Se arma la lista de períodos secundarios donde se debe respetar el factor de carga menor a uno para cada ciclo secundario.

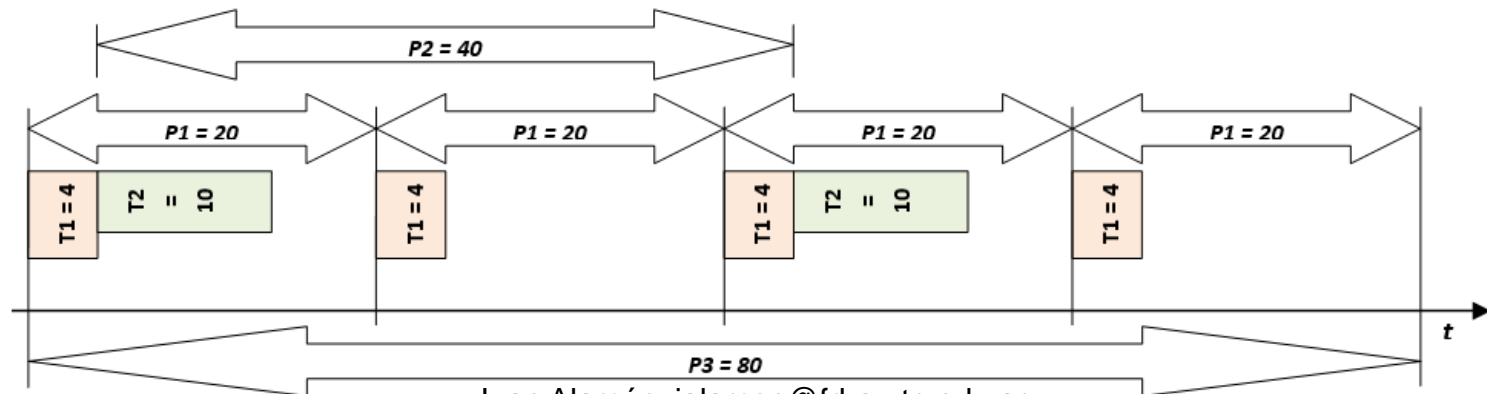
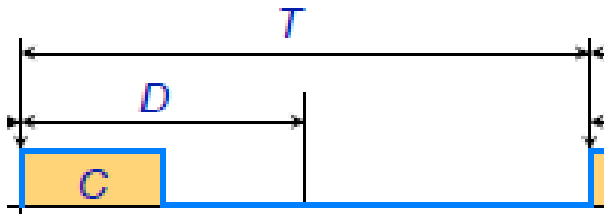
Planificación cíclica estática.

tiempo que
tarda cada
tarea

Ejemplo

Tarea	T_i (ms)	C_i (ms)	D_i (ms)
T1	20	4	20
T2	40	10	40
T3	80	40	80

- $U = 0,95$
- $T_M = \text{mcm}(T_i) = 80$
- $T_S = \text{mcd}(T_i) = 20$
- Para poder planificar todas las tramas se necesita un tick de 20ms pero el peor tiempo de tarea 3 es de 40ms.
- **No se puede planificar el sistema**



Segmentación de Tareas/Problemas

A veces no es posible confeccionar un plan cíclico que garantice los plazos

Si $U \leq 1$, es posible planificar la ejecución segmentando una o más tareas (los segmentos son secuencias de instrucciones de la tarea con un tiempo de cómputo conocido)

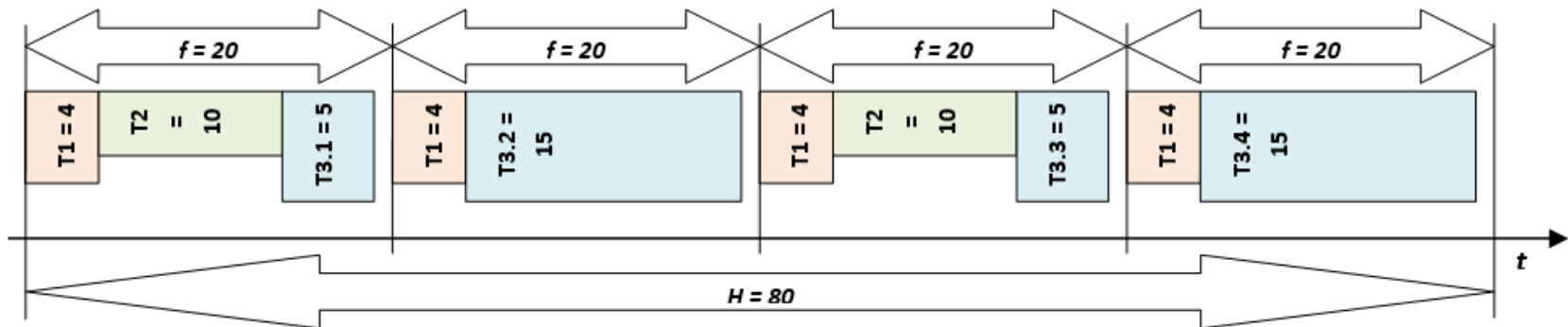
A veces es difícil ajustar el tiempo de cómputo de los segmentos

Si hay recursos compartidos, cada sección crítica debe estar incluida en un solo segmento

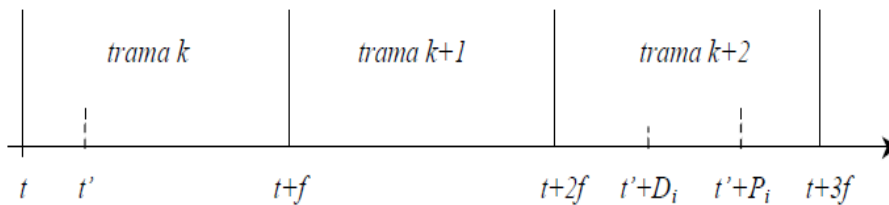
Si se modifica una sola tarea hay que rehacer la planificación completa (y posiblemente volver a segmentar de otra manera)

Planificación de la tarea

Tarea	T_i (ms)	C_i (ms)	D_i (ms)
T1	20	4	20
T2	40	10	40
T3.1	80	5	80
T3.2	80	15	80
T3.3	80	5	80
T3.4	80	15	80



Planificación cíclica estática (2)



1. $T_s \geq \max(C_i)$
2. $U = \sum_1^N \left(\frac{C_i}{T_i} \right) \leq 1$
3. $T_M = \text{mcm}(T_i)$
4. $\exists i: \frac{P_i}{T_s} - \left\lfloor \frac{P_i}{T_s} \right\rfloor = 0$
5. $\forall i: 2T_s - \text{mcd}(T_s, P_i) \leq D_i$

- La ecuación 1 intenta que toda tarea se pueda terminar en un único período secundario.
- La ecuación 2 determina el factor de carga de la CPU al igual que en la planificación anterior.
- La ecuación 3 nos determina el hiperperíodo.
- La ecuación 4 nos dice que **debe existir al menos una tarea que tenga período que sea múltiplo entero del período secundario**.
- La ecuación 5 determina que una tarea tiene que terminar antes de un nuevo período de activación.

Planificación cíclica estática.

Ejemplo

Tarea	T_i (ms)	C_i (ms)	D_i (ms)
T1	20	4	20
T2	40	10	40
T3	80	40	80

1. $T_s \geq \max(C_i)$
2. $U = \sum_1^N \left(\frac{C_i}{T_i} \right) \leq 1$
3. $T_M = \text{mcm}(T_i)$
4. $\exists i: \frac{P_i}{T_s} - \left\lfloor \frac{P_i}{T_s} \right\rfloor = 0$
5. $\forall i: 2T_s - \text{mcd}(T_s, P_i) \leq D_i$

1. $T_s \geq 40$ ms
2. $U = 0,95$
3. $T_M = 80$ ms
4. Se cumple para T2 y T3
5. Condición de fin
 - **T1: 60 ms \leq 20 ms**
 - T2: 40 ms \leq 40 ms
 - T3: 40 ms \leq 80 ms
 - **No se puede planificar**

Planificación cíclica estática. Ejemplo

Tarea	T_i (ms)	C_i (ms)	D_i (ms)
T1	20	4	20
T2	40	10	40
T3.1	80	5	80
T3.2	80	15	80
T3.3	80	5	80
T3.4	80	15	80

1. $T_s \geq 20$ ms
2. $U = 0,95$
3. $T_M = 80$ ms
4. Se cumple para todas las tareas
5. Condición de fin
 - T1 : 20 ms < 20 ms
 - T2 : 40 ms < 40 ms
 - T3.1: 20 ms < 80 ms
 - T3.2: 20 ms < 80 ms
 - T3.3: 20 ms < 80 ms
 - T3.4: 20 ms < 80 ms
 - ***Es planificable.***

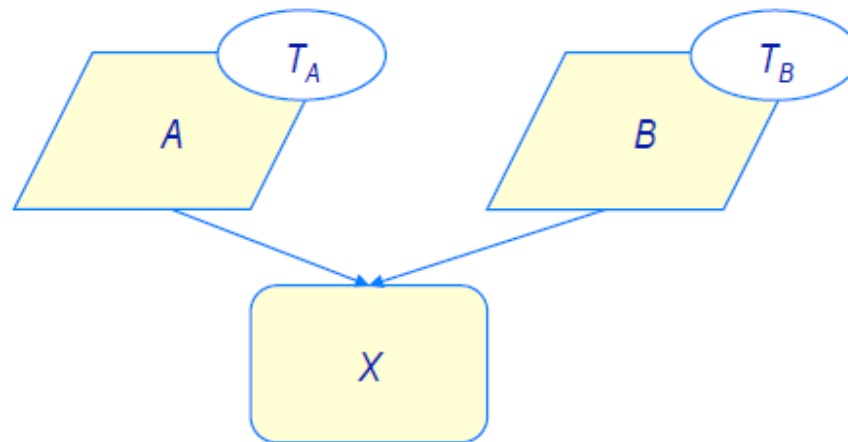
1. $T_s \geq \max(C_i)$
2. $U = \sum_1^N \left(\frac{C_i}{T_i} \right) \leq 1$
3. $T_M = mcm(T_i)$
4. $\exists i: \frac{P_i}{T_s} - \left\lfloor \frac{P_i}{T_s} \right\rfloor = 0$
5. $\forall i: 2T_s - mcd(T_s, P_i) \leq D_i$

Recursos Compartidos

Una tarea (o segmento) se ejecuta sin interrupción hasta que termina

No es necesario proteger los recursos compartidos
La exclusión mutua es automática

Es mas facil que en FRTOS
porque se el orden de
ejecucion de las tareas



Bibliografía.

- Sistemas Operativos. Diseño e Implementación. Andrew S. Tanenbaum.
- Sistemas de Tiempo Real y Lenguajes de Programación. Alan Burns, Andy Wellings. Tercera Edición. Addison Wesley.
- The Engineering of Reliable Embedded Systems. Michael J. Pont. SafeTTY Systems. este es el mejor libro