

# Tema 2

## Introducción a los Sistemas de Tiempo Real



# Objetivos

1. Comprender qué es un Sistema de Tiempo Real (STR) y saber diferenciarlos de otros sistemas informáticos
2. Conocer las principales características de un STR y cómo clasificarlos
3. Conocer las distintas arquitecturas software para la programación de STR



# Índice

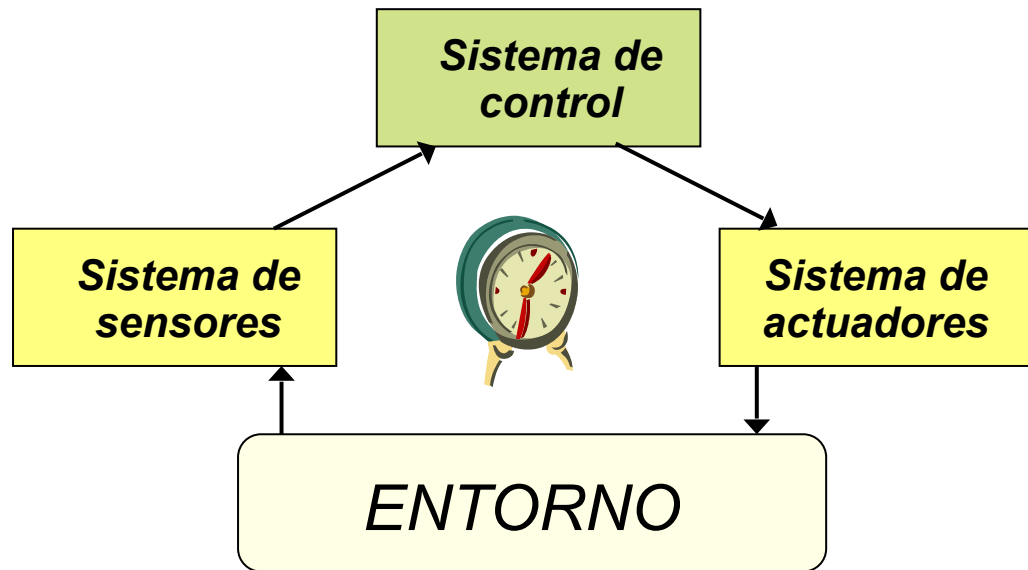
1. **Definición de un Sistema de Tiempo Real (STR)**
2. Características de un STR
3. Tipos de STR
4. Arquitecturas software en STR
5. Lenguajes de programación de tiempo real



## 1. Definición de un Sistema de Tiempo Real

### Definición de un STR

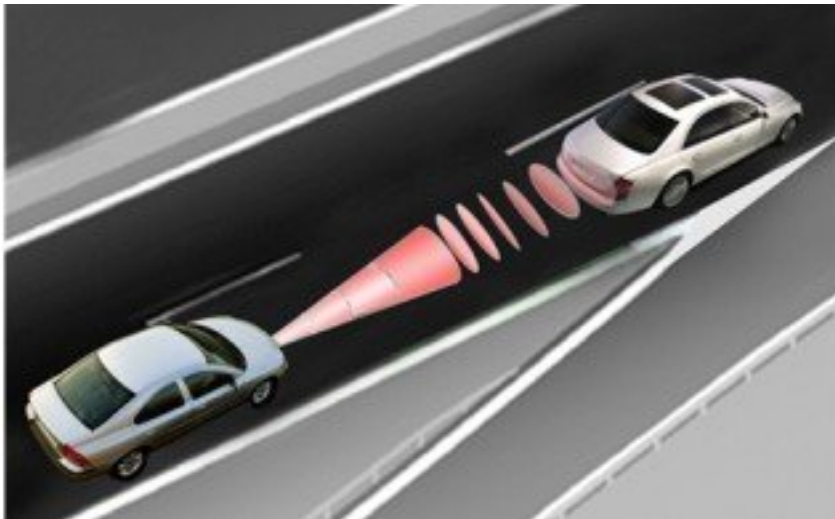
- Un **Sistema de Tiempo Real** es un sistema informático que:
  - **interacciona** repetidamente con su entorno físico
  - responde a los estímulos que recibe de dicho entorno en un **plazo de tiempo** determinado



## 1. Definición de un Sistema de Tiempo Real

### Significado de Tiempo Real

- Programa correcto lógica y **temporalmente**
- STR **distinto de** sistema rápido
- **Deadline**



**¿De cuánto tiempo disponemos para evitar la colisión? ...**

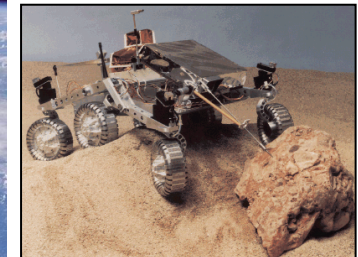
- ¿a qué velocidad van los vehículos?
- ¿a qué distancia están?
- ¿cuál es el estado de los frenos y de los neumáticos?
- ¿cuál es el estado de la carretera?
- etc.



## 1. Definición de un Sistema de Tiempo Real

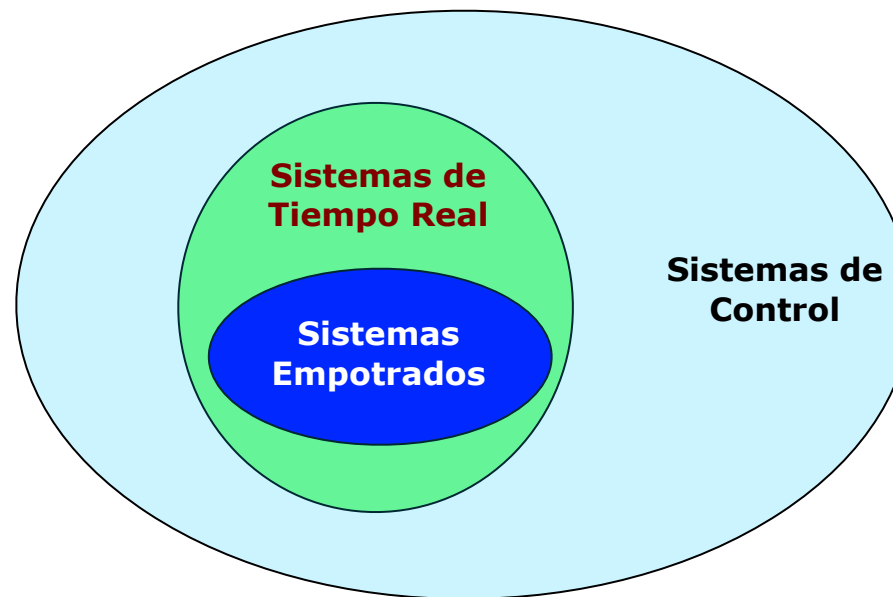
# Algunos ejemplos

- Transporte: ayudas a la conducción, control de tráfico aéreo, navegación autónoma ...
- Aplicaciones médicas: monitorización de pacientes, regulación presión sanguínea ...
- Industria militar: guiado de misiles ...
- Industria aeroespacial: satélites, cohetes ...
- Programación de electrodomésticos, sistemas domóticos ...
- Sistemas de control y fabricación de automóviles



## 1. Definición de un Sistema de Tiempo Real

### Conceptos relacionados



## 1. Definición de un Sistema de Tiempo Real

### Sistemas empotrados

- Por lo general, es un STR que forma parte de otro sistema físico en el que realiza operaciones de control (**embedded system**)
- Recursos limitados (procesador, memoria, pantalla, etc.)
- Dispositivos de entrada y salida especiales
- Suele ser no visible desde el exterior y es generalmente inaccesible al usuario
- La aplicación se ejecuta desde ROM





## 1. Definición de un Sistema de Tiempo Real

### Algunos requerimientos temporales de un controlador de vuelo

- Hacer cada ciclo de  $1/180$  segundos:
  - Validar datos de sensores y seleccionar fuente de datos; si error, reconfigurar el sistema
  - Realizar los cálculos de control a 30Hz, una vez cada 6 ciclos, de los circuitos externos de inclinación, balanceo y guiñada
  - Realizar los cálculos de control a 90Hz una vez cada 2 ciclos, de los circuitos internos de inclinación y balanceo usando como entrada las salidas del paso anterior
  - Realizar los cálculos de los circuitos internos de guiñada, usando las salidas del paso anterior
  - Dar como salida los comandos
  - Ejecutar los comandos y esperar al principio del siguiente ciclo



# Índice

1. Definición de un Sistema de Tiempo Real (STR)
2. **Características de un STR**
3. Tipos de STR
4. Arquitecturas software en STR
5. Lenguajes de programación de tiempo real



## 2. Características de un STR

# Principales características de un STR

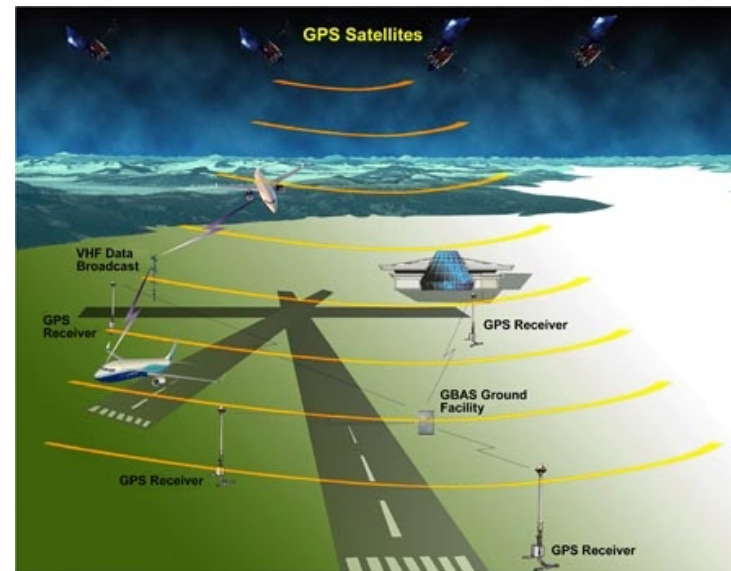
- Concurrencia
- Dependencia del tiempo
- Fiabilidad y seguridad



## 2. Características de un STR

# Concurrencia

- Modelar el paralelismo en el mundo real
- Los STR son inherentemente concurrentes



## 2. Características de un STR

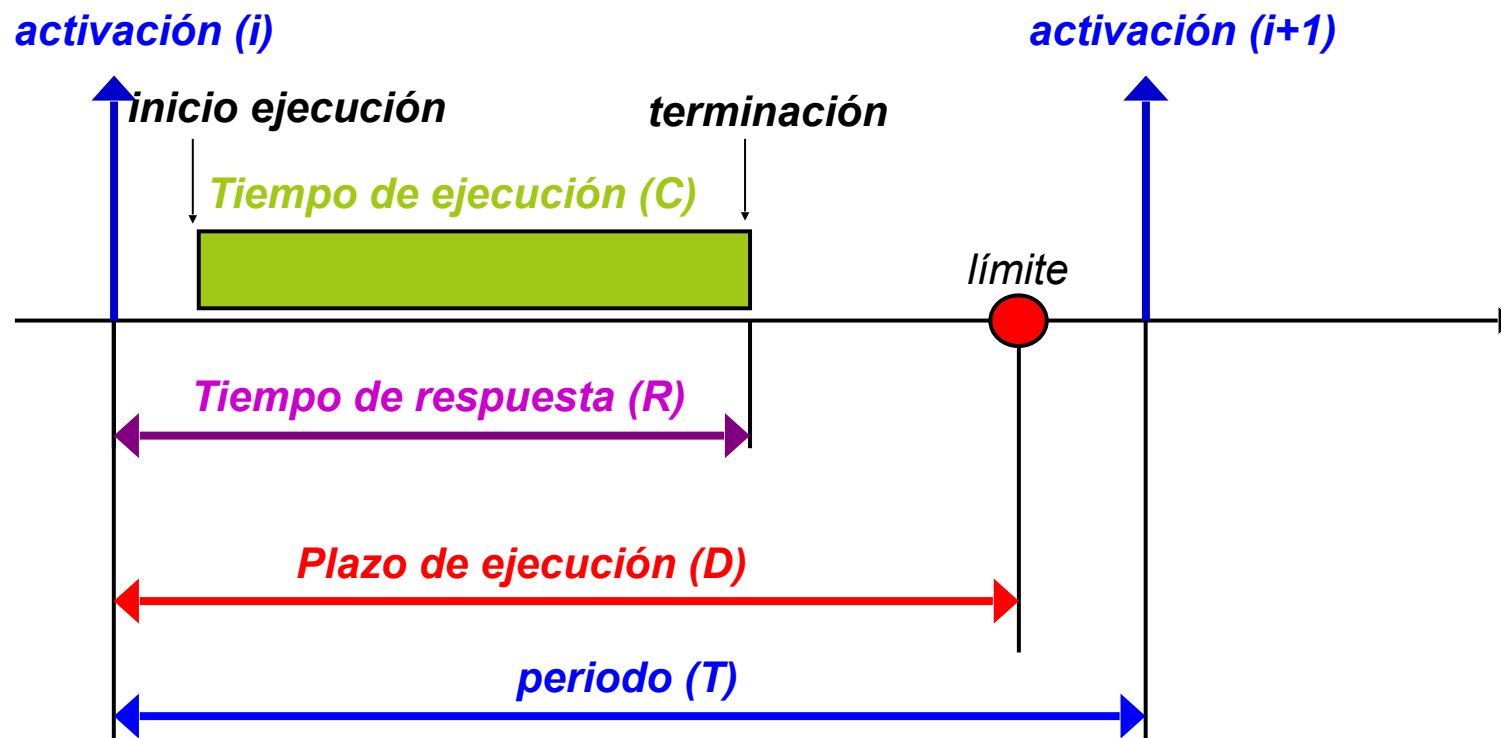
### Dependencia del tiempo

- ▣ Garantizar que todas las tareas terminan antes de su **plazo (deadline)**
- ▣ El comportamiento general de un programa concurrente es **no determinista**
- ▣ Todas las secuenciaciones posibles de las tareas pueden no cumplir las **restricciones temporales** del sistema
- ▣ Un STR necesita restringir el no determinismo de un sistema concurrente => **PREDECIBLE** => **SCHEDULING**
- ▣ El **scheduling** proporciona:
  - ▣ un **PLANIFICADOR**
  - ▣ un **TEST DE PLANIFICABILIDAD**



## 2. Características de un STR

# Tiempos en la ejecución de una tarea



## 2. Características de un STR

### Fiabilidad y Seguridad

- **Fiabilidad** = Probabilidad de proporcionar el servicio requerido
  - Corrección
    - Lógica
    - Temporal
- **Seguridad** = Probabilidad de recuperación frente a un fallo



## 2. Características de un STR

### Caso de estudio: fallo del misil Patriot

- Cálculo predictivo de la posición del misil Scud a interceptar
- Reloj interno = contador de décimas de segundos desde el encendido
- Cálculo erróneo de la trayectoria
  - Barrido del radar cada aproximadamente  $\frac{1}{2}$  segundo
  - Convertir los valores del reloj a segundos => Multiplicar por 0.1
  - El número real 0.1 es periódico en binario
  - En un procesador de 24 bits se almacena con un error de 0,000000095
  - Después de 100 horas de funcionamiento, se acumula un error de 0.34 segundos
  - Un misil Scud viaja a más de 1600 m/s
  - En 0.34 segundos, el misil Scud viaja aproximadamente 600 metros
  - Por tanto, después de 100h, el misil Patriot no fue capaz de interceptar al misil enemigo





## 2. Características de un STR

### Caso de estudio: accidente del Ariane 5

- A los 36,7 segundos el software de guiado inercial produce overflow al convertir un número representado en 64 bits (float) a 16 bits (int)
- Caída del sistema de backup, a los 0.05 segundos caída del sistema principal
- El sistema de control principal recibe datos de diagnóstico que interpreta como datos de vuelo (ha cambiado la posición del cohete)
- Fuerte reacción para corregir la trayectoria. Desintegración por la fuerza aerodinámica. Autodestrucción
- Causas: El sistema de guiado inercial es el mismo que el del Ariane 4 y la velocidad horizontal de Ariane 5 es cinco veces superior a la de su antecesor, lo que provoca este overflow.
- Las especificaciones y pruebas realizadas no lo tuvieron en cuenta.



## 2. Características de un STR

### Otras características de un STR

- Interacción con dispositivos físicos
- Gran tamaño y complejidad
- Cálculos con números reales
- Datos volátiles I/O
- Pruebas (*Testing*)



# Índice

1. Definición de un Sistema de Tiempo Real (STR)
2. Características de un STR
3. **Tipos de STR**
4. Arquitecturas software en STR
5. Lenguajes de programación de tiempo real



### 3. Tipos de STR

## Tipos de STR

#### ■ Según su **Criticidad**

- Tiempo real estricto/**críticos** (*hard real time*)
- Tiempo real flexible/**acríticos** (*soft real time*)
- Tiempo real **firme** (*firm real time*)

#### ■ Según el instante de **activación** de las tareas

- Sistemas **dirigidos por tiempo** (*time-triggered systems*)
- Sistemas **dirigidos por eventos** (*event-triggered systems*)



### 3. Tipos de STR

## Según su criticidad

- Tiempo real estricto/**críticos** (*hard real time*)
  - Todas las acciones deben ocurrir dentro del plazo especificado
  - Una sola respuesta tardía puede tener consecuencias fatales
  
- Tiempo real flexible/**acríticos** (*soft real time*)
  - Se pueden perder plazos ocasionalmente
  - El valor o utilidad de la respuesta decrece con el tiempo
  
- Tiempo real **firme** (*firm real time*)
  - Se pueden perder plazos ocasionalmente
  - Una respuesta tardía no tiene valor

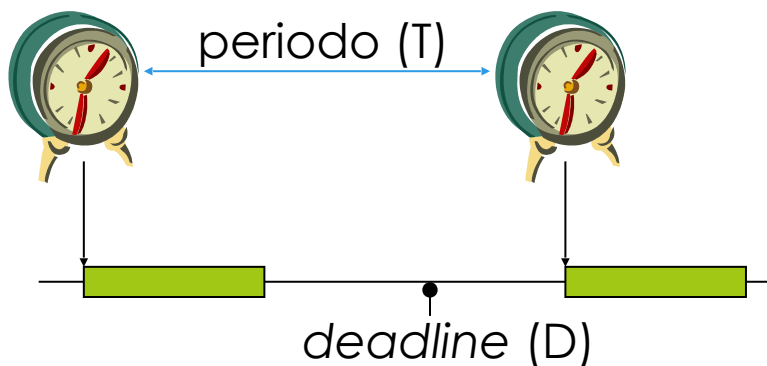


### 3. Tipos de STR

## Según el instante de activación de las tareas

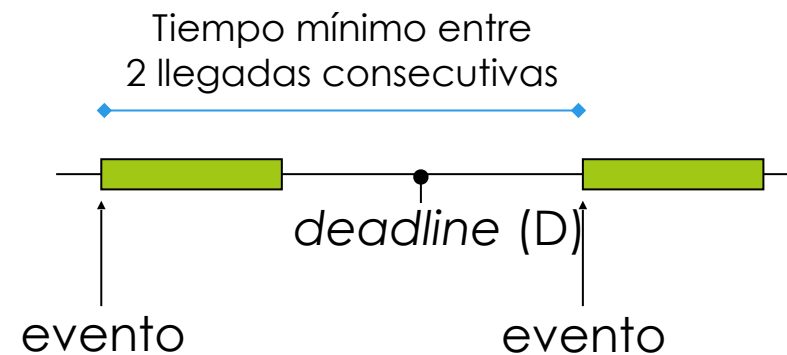
Sistemas **dirigidos por tiempo**  
(*time-triggered systems*)

- Inicio en instantes predeterminados
- Mecanismo básico: reloj



Sistemas **dirigidos por eventos**  
(*event-triggered systems*)

- Inicio cuando se produce un suceso de cambio de estado
- Mecanismo básico: interrupciones



### 3. Tipos de STR

## Esquemas de activación

#### ■ Periódica

- Se ejecuta regularmente, con un periodo bien definido (ciclo)

#### ■ Aperiódica

- Se ejecuta de forma irregular, en respuesta a un suceso del entorno o del propio sistema

#### ■ Esporádica

- Se exige una separación mínima entre dos sucesos consecutivos



# Índice

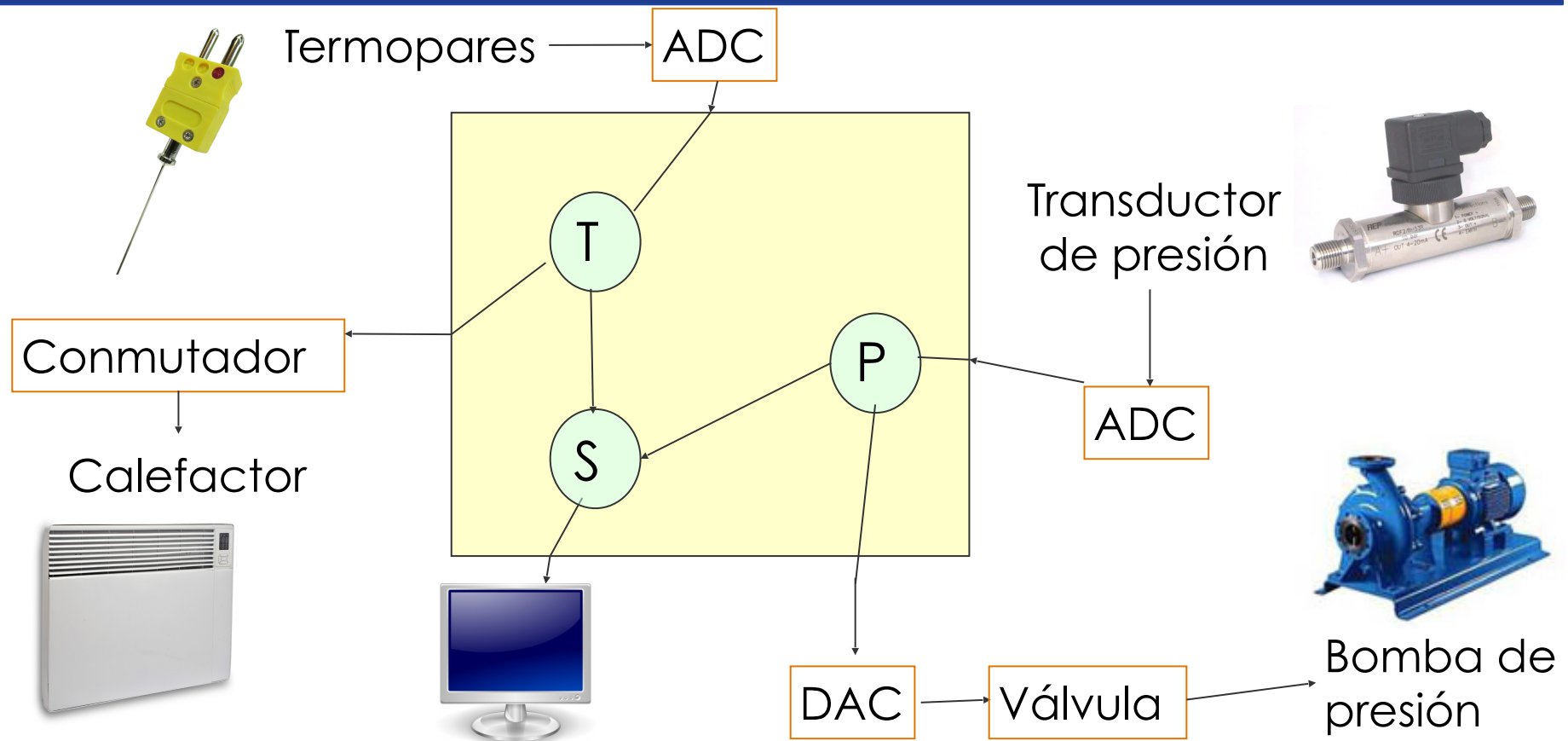
1. Definición de un Sistema de Tiempo Real (STR)
2. Características de un STR
3. Tipos de STR
4. **Arquitecturas software en STR**
5. Lenguajes de programación de tiempo real





## 4. Arquitecturas software en STR

### Un ejemplo de STR sencillo



El objetivo del sistema es mantener la temperatura y presión de un determinado proceso químico dentro de unos límites definidos



## 4. Arquitecturas software en STR

### Arquitecturas software posibles

#### ■ Ejecutivo cíclico

- Un único **programa secuencial** con un bucle de control, ignorando la evidente concurrencia de T, P y S.

#### ■ Sistema Operativo de Tiempo Real

- T, P y S se escriben en un lenguaje de programación secuencial (como **programas separados** o como **procedimientos** distintos de un mismo programa)

#### ■ Lenguaje de programación de Tiempo Real

- Un **único programa concurrente** que mantiene la estructura lógica de T, P y S.



¿Cuál es la mejor alternativa?



## 4. Arquitecturas software en STR

### Ejecutivo Cíclico: solución secuencial

```
procedure Controller is
  TR : Temp_Reading;      -- 10..500
  PR : Pressure_Reading;  -- 0..750
  HS : Heater_Setting;    -- (on, off)
  PS : Pressure_Setting;  -- (0..9)
begin
  loop
    Read(TR);      -- del ADC de los termopares
    Temp_Convert(TR,HS); -- convertir lectura en actualización
    Write(HS);     -- al conmutador
    Write(TR);     -- a la pantalla
    -- Ahora hacemos lo mismo pero para la presión
    Read(PR);      -- del ADC del transductor de presión
    Pressure_Convert(PR,PS); -- convertir lectura en actualización
    Write(PS);     -- a la válvula
    Write(PR);     -- a la pantalla
  end loop; -- bucle infinito, común en software empotrado
end Controller;
```



# Desventajas de la solución secuencial

- Las lecturas de temperatura y presión se toman con la misma frecuencia
- El uso de contadores y sentencias *if* pueden mejorar la situación
- Pero todavía sería necesario entremezclar acciones de secciones computacionalmente intensivas (`Temp_Convert` y `Pressure_Convert`), para que se cumpla el balance de trabajo requerido
- Mientras se espera a leer la temperatura no se presta atención a la presión (y viceversa)
- Si no se devuelve el control desde la lectura de la temperatura, entonces no se realizaría ninguna lectura adicional de la presión



## 4. Arquitecturas software en STR

# Una mejora de la solución secuencial

```
procedure Controller is
  TR : Temp_Reading;
  PR : Pressure_Reading;
  HS : Heater_Setting;
  PS : Pressure_Setting;
  Ready_Temp, Ready_Pres : Boolean;
begin
  loop
    if Ready_Temp then
      Read(TR);
      Temp_Convert(TR,HS);
      Write(HS);
      Write(TR);
    end if;
    if Ready_Pres then
      Read(PR);
      Pressure_Convert(PR,PS);
      Write(PS);
      Write(PR);
    end if;
  end loop;
end Controller;
```



¿Cuál es ahora el problema?



#### 4. Arquitecturas software en STR

### Problema: espera ocupada

- El programa consume una gran proporción de su tiempo en un **bucle ocupado**, comprobando si los dispositivos de entrada están disponibles
- Las esperas ocupadas son **ineficientes**

La principal crítica al programa secuencial es que no se reconoce el hecho de que los ciclos de temperatura y presión son subsistemas completamente independientes



## 4. Arquitecturas software en STR

### Desventajas del Ejecutivo Cíclico

- Generalmente son difíciles de diseñar y complican la labor del programador
- Programas poco claros y poco elegantes
- Dificultad para probar la corrección del programa
- Dificultad para lograr la ejecución en paralelo del programa en un sistema multiprocesador
- Dificultad de ubicar el código que trata los fallos
- Mal encaje de tareas aperiódicas
- Limitaciones en los periodos



## 4. Arquitecturas software en STR

# Sistema operativo de tiempo real: usando primitivas del S.O.

```
package OSI is  
  type Thread_ID is private;  
  type Thread is access procedure;  
  
  function Create_Thread(Code : Thread) return Thread_ID;  
  -- otros subprogramas  
  procedure Start(ID : Thread_ID);  
private  
  type Thread_ID is ...;  
end OSI;
```





## 4. Arquitecturas software en STR

### Usando primitivas del S.O.

```
package body Processes is  
  procedure Temp_C is  
    TR : Temp_Reading;  
    HS : Heater_Setting;  
  begin  
    loop  
      Read(TR);  
      Temp_Convert(TR,HS);  
      Write(HS);  
      Write(TR);  
    end loop;  
  end Temp_C;  
  procedure Pressure_C is  
    PR : Pressure_Reading;  
    PS : Pressure_Setting;  
  begin  
    loop  
      Read(PR);  
      Pressure_Convert(PR,PS);  
      Write(PS);  
      Write(PR);  
    end loop;  
  end Pressure_C;  
end Processes;
```



## 4. Arquitecturas software en STR

### Usando primitivas del S.O.

```
procedure Controller is  
    TC, PC : Thread_ID;  
begin  
    TC := Create_Thread(Temp_C'Access);  
    PC := Create_Thread(Pressure_C'Access);  
    Start(TC);  
    Start(PC);  
end Controller;
```

- **Mejora la solución secuencial**
- **Pero en sistemas complejos la interfaz procedural oscurece la estructura del programa : no está claro qué procedimientos son realmente procedimientos y cuáles se pretende que sean actividades concurrentes**



## 4. Arquitecturas software en STR

# Usando un lenguaje de programación de tiempo real

```
procedure Controller is
```

```
task Temp_Controller;  
task body Temp_Controller is  
    TR : Temp_Reading;  
    HS : Heater_Setting;  
    begin  
        loop  
            Read(TR);  
            Temp_Convert(TR,HS);  
            Write(HS);  
            Write(TR);  
        end loop;  
    end Temp_Controller;
```

```
task Pressure_Controller;  
task body Pressure_Controller is  
    PR : Pressure_Reading;  
    PS : Pressure_Setting;  
    begin  
        loop  
            Read(PR);  
            Pressure_Convert(PR,PS);  
            Write(PS);  
            Write(PR);  
        end loop;  
    end Pressure_Controller;
```

```
begin  
    null;    -- ha comenzado la ejecución de ambas tareas  
end Controller;
```



# Solución con un lenguaje de tiempo real

### ■ **Ventajas:**

- Cada tarea define su propio ciclo de control
- Mientras una tarea está suspendida, la otra puede estar ejecutándose
- La ejecución concurrente de las tareas expresa el paralelismo inherente del dominio de aplicación

### ■ **Inconvenientes:**

- Control de recursos:
  - Las tareas acceden a recursos que requieren exclusión mutua
  - Comunicación y sincronización entre tareas



# Índice

1. Definición de un Sistema de Tiempo Real (STR)
2. Características de un STR
3. Tipos de STR
4. Arquitecturas software en STR
5. **Lenguajes de programación de tiempo real**



## 5. Lenguajes de programación de tiempo real

### Tipos de Lenguajes para un STR

#### □ **Ensamblador**

- Flexible y eficiente, pero costoso y poco fiable

#### □ **Secuencial** (Fortran, C, C++)

- Necesitan un S.O. para concurrencia y tiempo real

#### □ **Concurrente** (Ada, Modula-2, Java)

- Concurrencia y tiempo real incluidos en el lenguaje



## 5. Lenguajes de programación de tiempo real

### Criterios de diseño

Características	Lenguaje de TR	Lenguaje Ada
Concurrencia	Procesos simultáneos	Tareas, objetos protegidos, citas extendidas
Dependencia del tiempo	Especificación y análisis	Librería de paquetes: Calendar y Real_Time
Fiabilidad y seguridad	Legibilidad y mecanismos de recuperación de errores	Fuertemente tipado, manejadores de excepciones
Interacción con el hardware	Control de interrupciones y drivers	Manejador de interrupciones
Tamaño y complejidad	Modularidad y portabilidad	Paquetes y unidades genéricas



# Conclusiones

- Los STR trabajan en un **entorno concurrente** y con **restricciones temporales**
- Amplio **dominio de aplicación**
- Sus particulares **características** los hacen diferentes de otros tipos de sistemas informáticos
- Requieren de una **tecnología software** apropiada
- La noción de **proceso** proporciona mayor expresividad y facilidad de uso del lenguaje
- Sin concurrencia, el software tiene que ser construido con un único **bucle de control**





## Bibliografía Recomendada

Sistemas de tiempo real y lenguajes de programación (**3ª edición**)

Alan Burns and Andy Wellings

Addison Wesley (2002)

☐ Capítulo 1 (Completo)

☐ Capítulo 2 (Apartado 2.5)

☐ Capítulo 7 (Completo, excepto lo referente a otros lenguajes)



# Bibliografía Complementaria

Real-time systems  
Jane W. S. Liu  
Prentice Hall (2000)

☒ Capítulo 1 (Completo)  
☐ Ejemplos de aplicaciones de tiempo real

