

Sistemas de tiempo real

Objetivos

Conocer la problemática del desarrollo de los sistemas de tiempo real

Saber desarrollar aplicaciones con lenguajes apropiados.

Conocer las características de los lenguajes de programación

Conocer la problemática del sistema operativo y su influencia

Conocer y saber analizar un sistema de tiempo real.

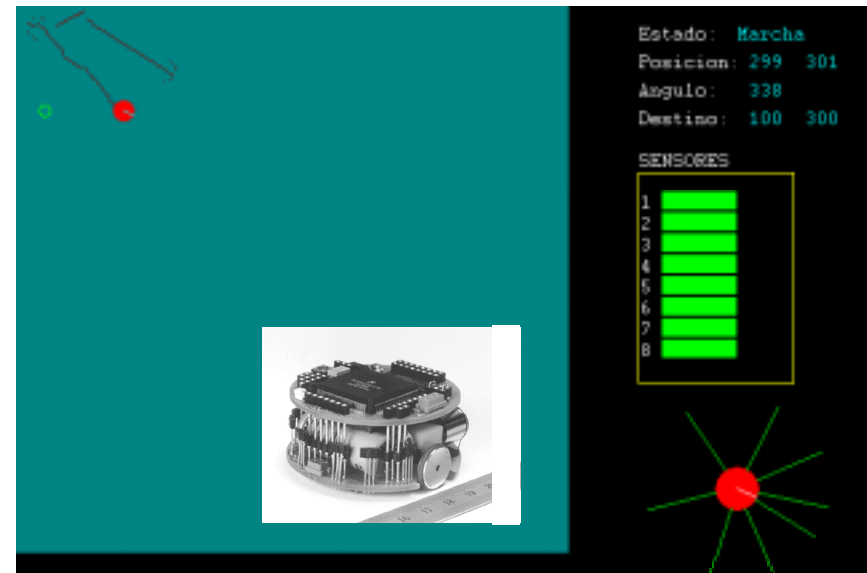
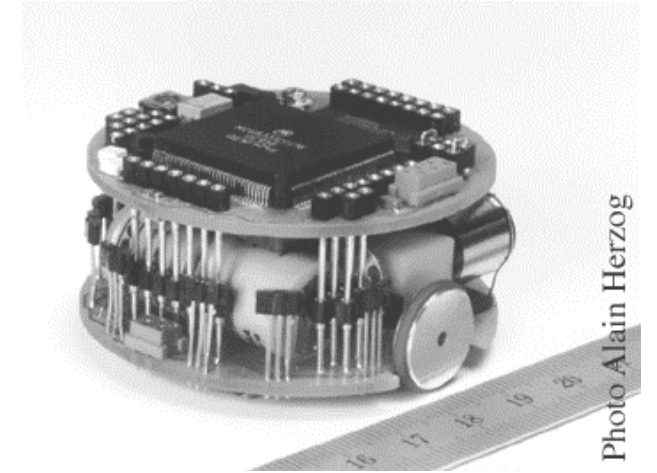
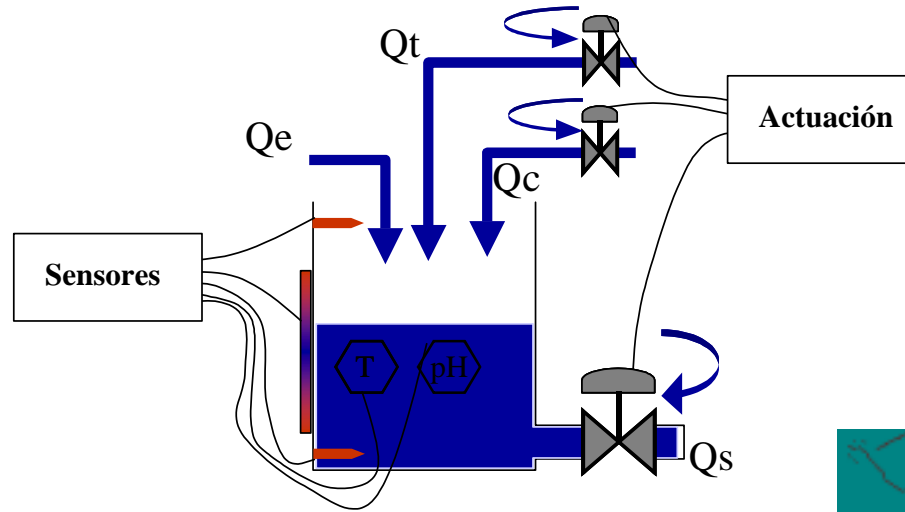
1. Introducción
2. Introducción a Lenguajes
3. Programación concurrente
 - Planificación
 - Datos compartidos
 - Mecanismos de control
4. Programación de sistemas de t.r.
 - Lenguajes de programación
 - Ejemplos de programación
 - Sistemas operativos de t.r.
5. Análisis de la planificabilidad de los s.t.r.
 - Modelo de tareas
 - Planificación estática
 - Planificación dinámica
6. Sistemas operativos de tiempo real
 - Requisitos
 - Sotr disponibles
 - RTLlinux

Lenguajes:

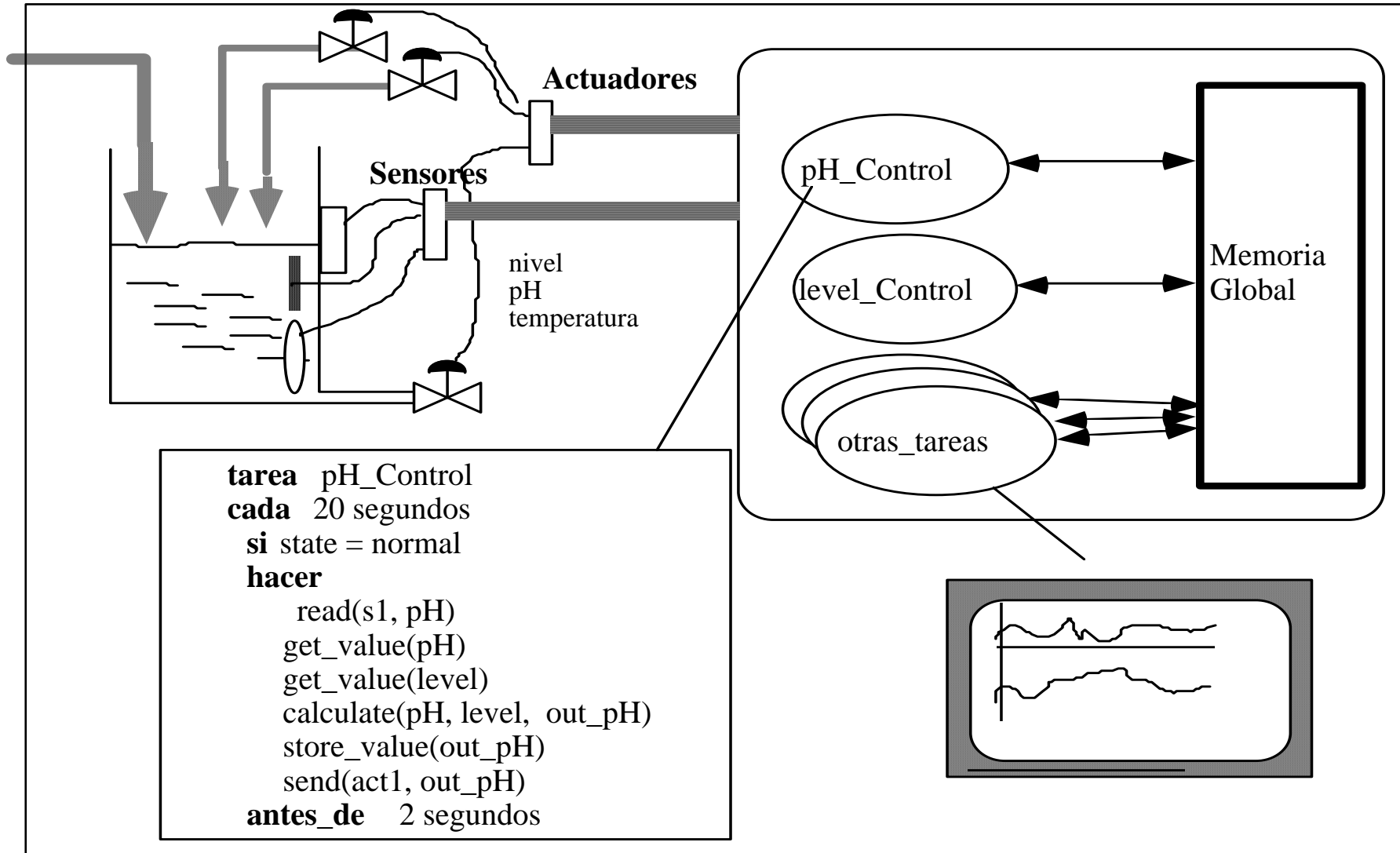
**Sistema de control
simulado**

Sistema de control real

Robot industrial



Ejemplo



Definiciones

Un sistema de tiempo real es un sistema informático en el que es significativo el tiempo en el que se producen sus acciones.

No es suficiente que las acciones del sistema sean correctas lógicamente, sino que, además, deben producirse *dentro de un intervalo de tiempo determinado*

Esto es debido a que el sistema está conectado a un proceso externo del que recibe estímulos a los que debe responder con suficiente rapidez para evitar que evolucione a un estado indeseable.

Los estímulos externos aparecen en instantes de tiempo no predecibles. Tiene que resolver distintas cosas a la vez (*concurrentemente*)

Tipos de sistemas

Se pueden distinguir cuatro tipos de sistemas:

- **Críticos:** (hard real time systems) son aquellos en los que el tiempo de respuesta debe garantizarse a toda costa. Una respuesta tardía puede tener consecuencias fatales.
- **Esenciales:** (soft real time systems) representan aquellos sistemas con restricciones de tiempo en las que una respuesta tardía no produce graves daños pero sí un deterioro del funcionamiento global.
- **Incrementales:** la calidad de la respuesta obtenida depende del tiempo disponible para su cálculo. Si se les da más tiempo la respuesta mejora.
- **No esenciales;** corresponden con las tareas sin restricciones temporales.

El computador en el bucle de control

task body Controlador **is**

Periodo : Time_Span; -- Periodo de la tare

Proxima_Iteración : Time; -- tiempo absolutc

begin

Proxima_Iteración := Clock;

loop

convertir_sensor_analógico_digital(y);

calcular_accion_control(u);

enviar_accion_control_convertida(u);

actualizar_variables_internas(e,y,u,...);

Proxima_Iteración:=Proxima_Iteración + Periodo;

delay until Proxima_Iteración;

end loop;

end Controlador;

task Controlador

cada Tmues **hacer**

convertir_sensor_analógico_digital (y);

calcular_accion_control(u);

enviar_accion_control_convertida(u);

actualizar_variables_internas(e,y,u,...);

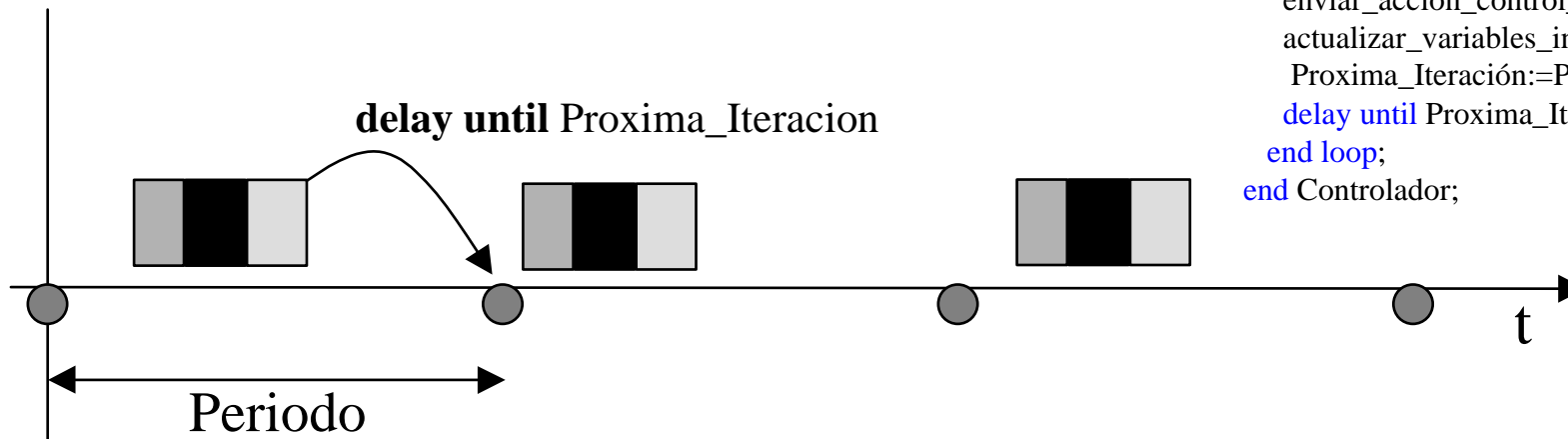
fin hacer




fin tarea

El computador en el bucle de control

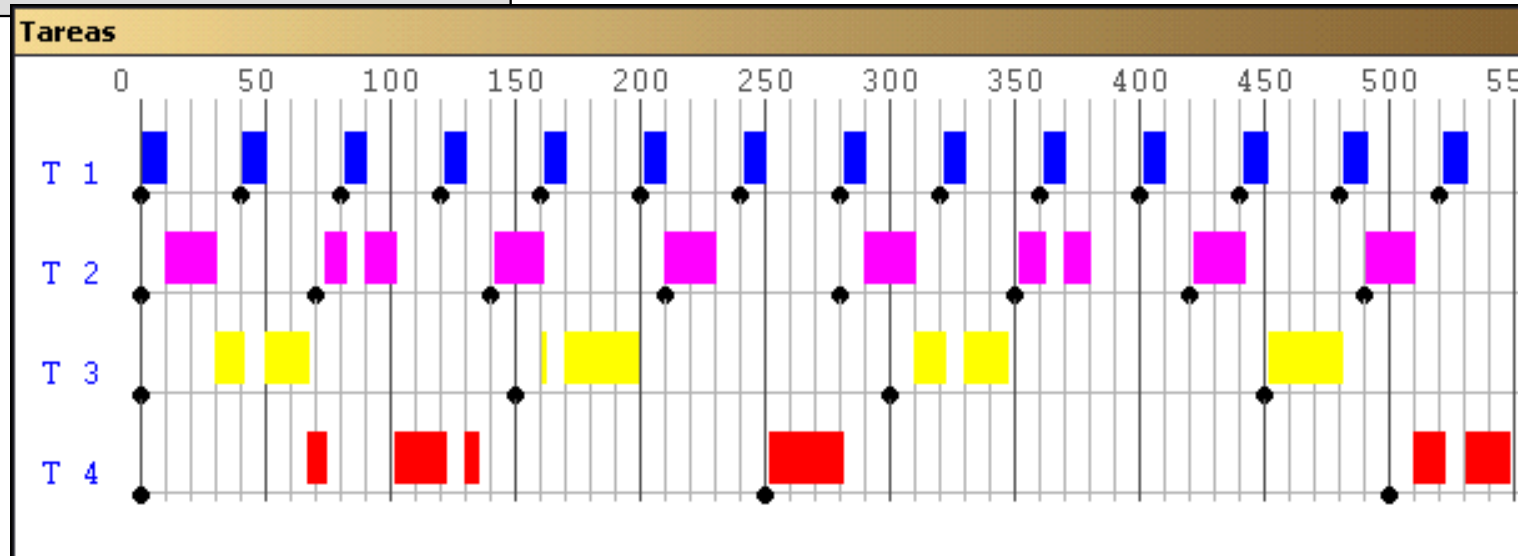
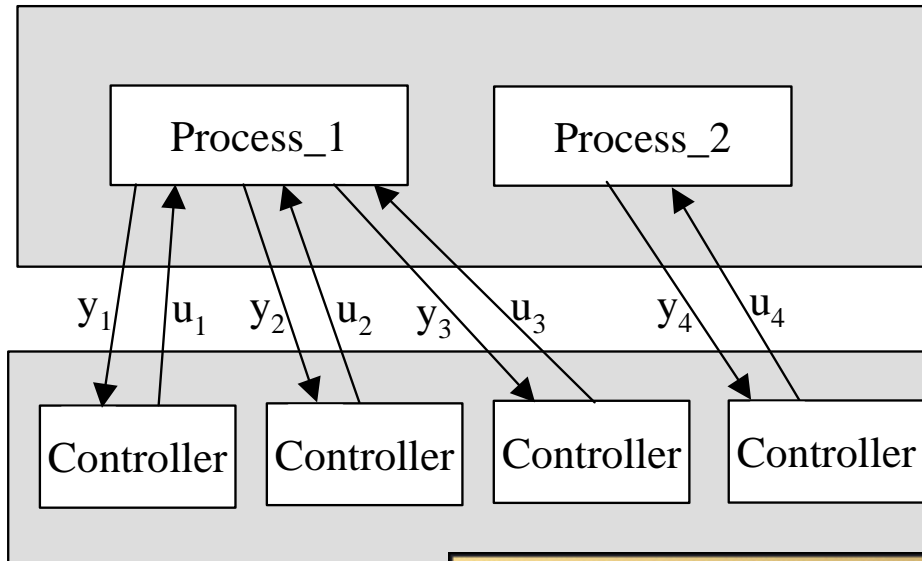
```

task body Controlador is
    Periodo      : Time_Span; -- Periodo de la tarea
    Proxima_Iteración : Time;  -- tiempo absoluto
begin
    Proxima_Iteración := Clock;
loop
    convertir_sensor_analógico_digital(y);
    calcular_accion_control(u);
    enviar_accion_control_convertida(u);
    actualizar_variables_internas(e,y,u,...);
    Proxima_Iteración:=Proxima_Iteración + Periodo;
    delay until Proxima_Iteración;
end loop;
end Controlador;
    
```

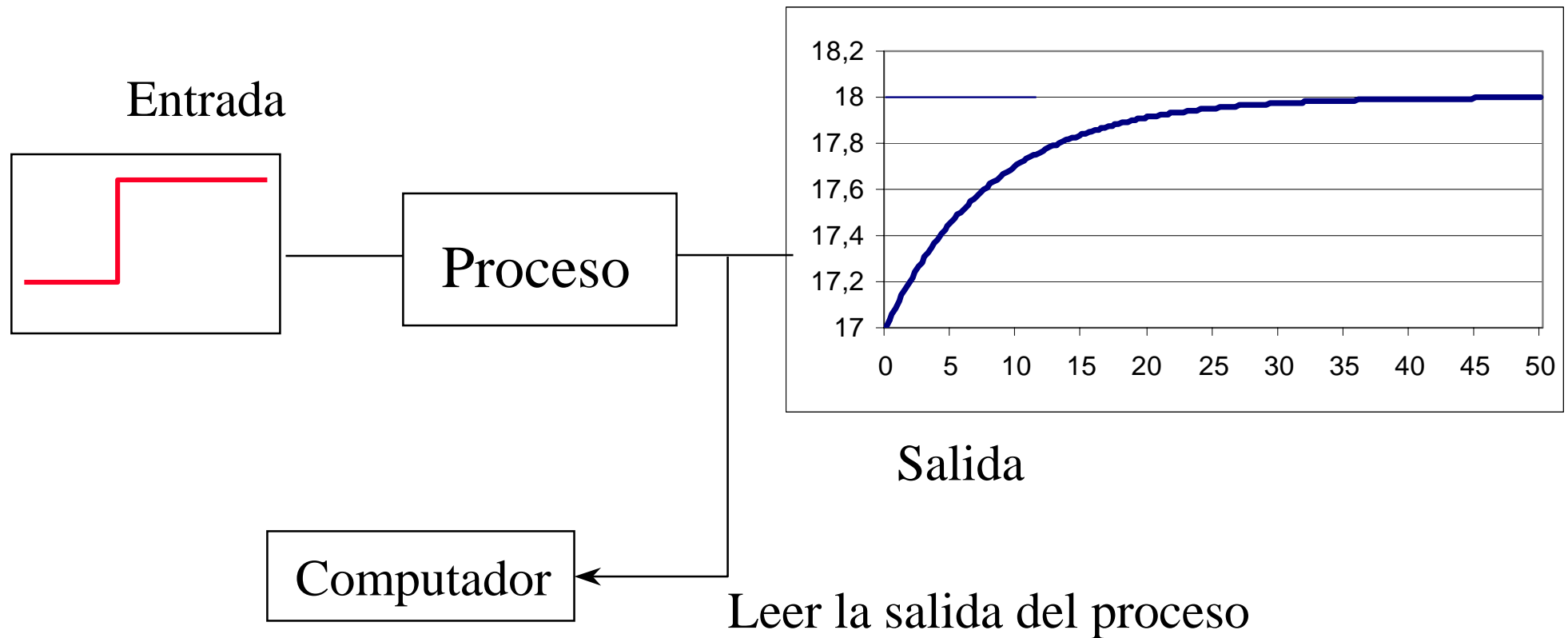


-  convertir_sensor_analogico_digital(y);
-  calcular_accion_control(u);
-  enviar_accion_control_convertida(u);

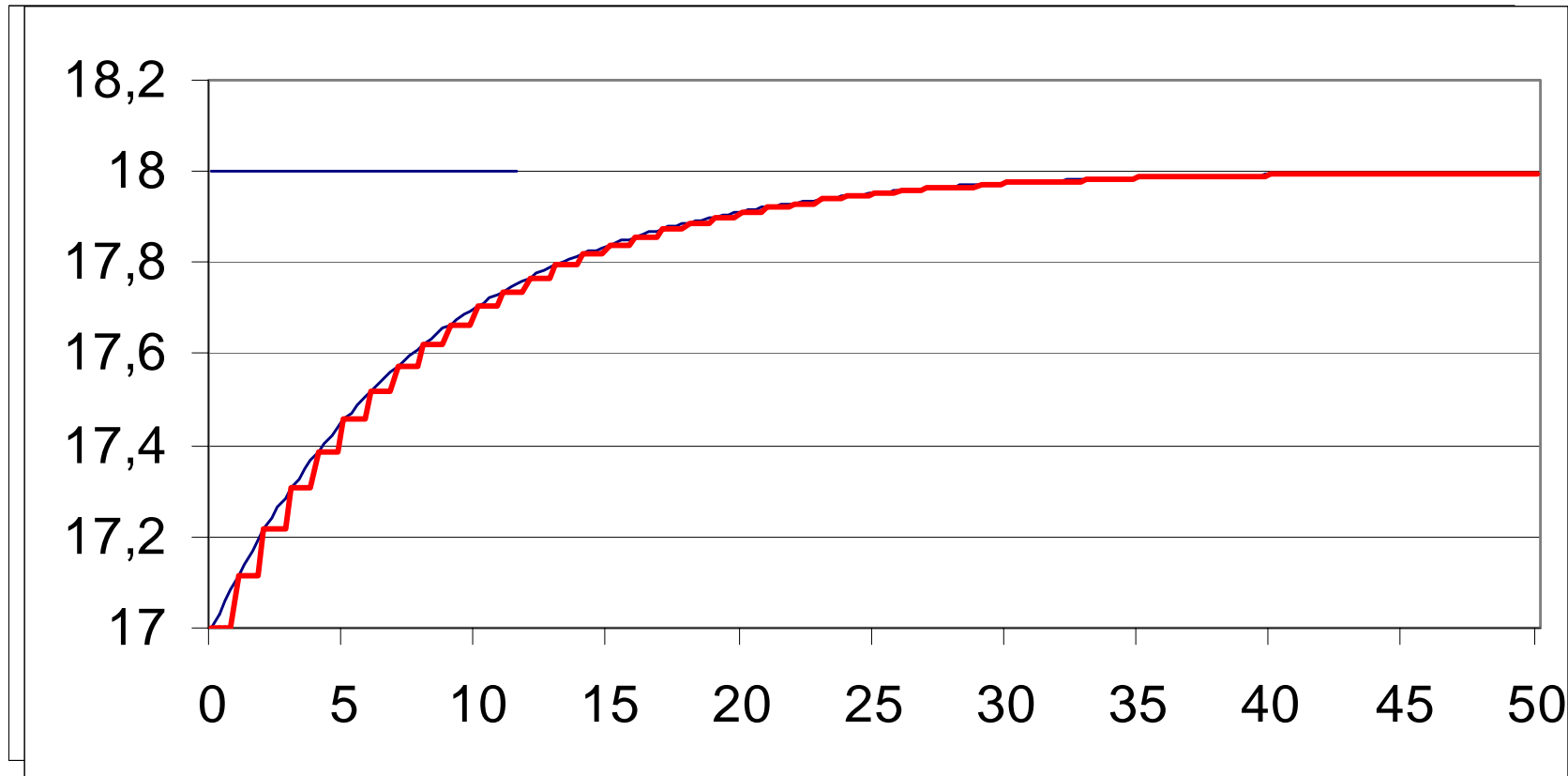
El computador en el bucle de control



Periodo de muestreo



Periodo de muestreo



Periodo de muestreo

En el diseño del regulador se define una frecuencia mínima (teorema de Shanon-Nyquist).

A frecuencia mayores mejor funcionamiento.

Pero: Sistemas no lineales => nuevo regulador dependiendo de la frecuencia

En la práctica: $[P_{\min}, P_{\max}]$

Modelo de tareas

Los sistemas de control de basan en la realización de actividades:

Periodicas: Las acciones se realizan a intervalos regulares.

El modelo de tarea periódico es un modelo muy conocido y para el cual se han desarrollado varias extensiones caracterizando distintas funcionalidades de los sistemas de tiempo real. Hay métodos y herramientas para el diseño, análisis y validación de sistemas de tiempo real caracterizados pro este modelo.

Esporádicas: que son tareas que responden a la llegada de eventos aperiódicos.

Modelo de tareas

Periodicas: Características

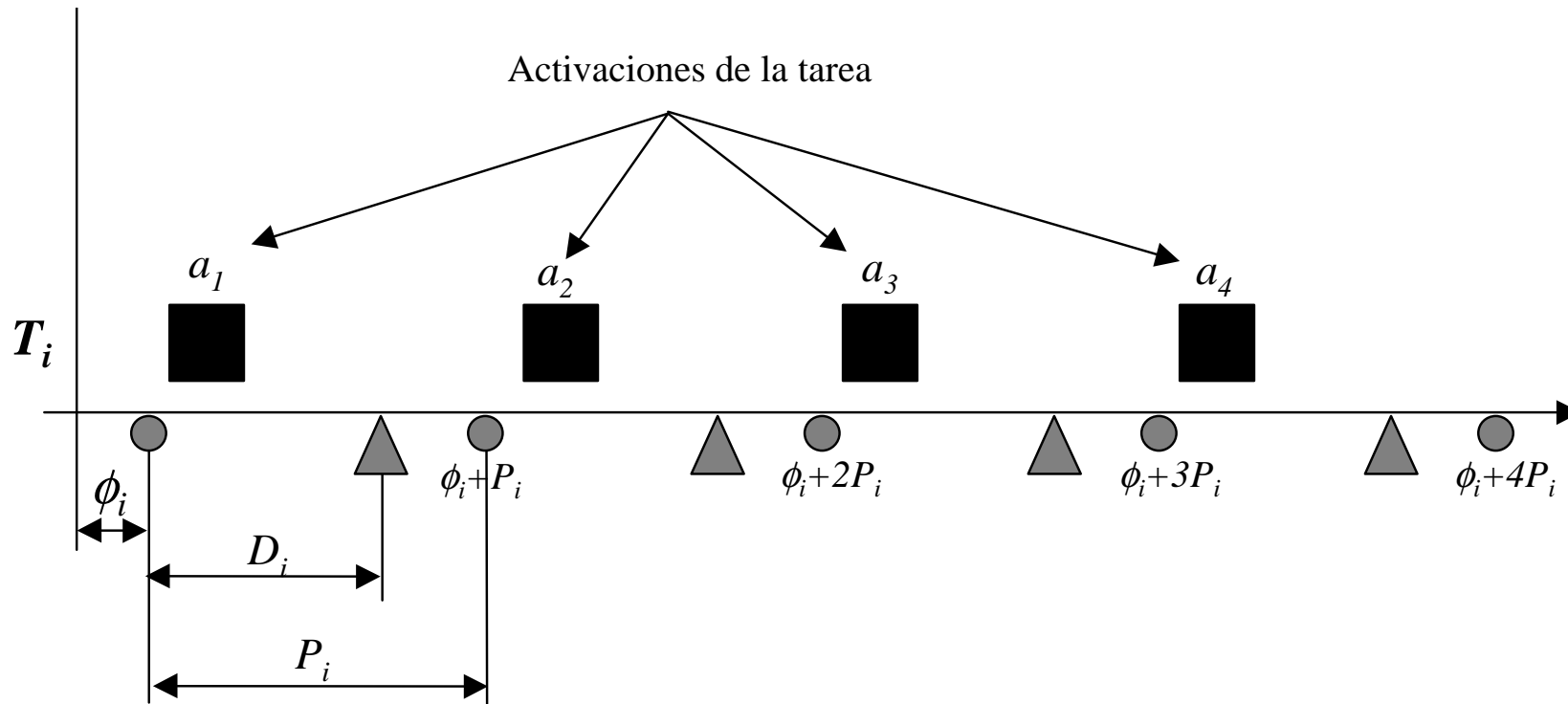
Periodo: El cómputo es ejecutado de forma regular cada intervalo de tiempo. Específicamente, cada periodo de la tarea T_i , denotado por P_i es una secuencia de activaciones, $a_1, a_2..a_n$, cada una de ellas en un intervalo de tiempo denotado por los intervalos $[(k-1) \cdot P_i, k \cdot P_i]$.

Plazo de entrega: El plazo de entrega (D_i) de una tarea T_i es el instante en el cual cualquier activación de la tarea tiene que haber finalizado. Normalmente, el plazo de entrega de una tarea será igual al periodo. Esto quiere decir que cada activación de la tarea debe finalizar su ejecución antes del siguiente periodo. En algunos casos, dependiendo de la restricciones de usuario, el plazo de entrega será menor o mayor que el periodo.

Fase Inicial: La fase inicial de una tarea T_i denota el defase que tiene el inicio de la primera activación de la tarea respecto al tiempo de inicio y se denota como ϕ_i .

Modelo de tareas

Periodicas: Características



Modelo de tareas

Periodicas: Características

Tiempo de cómputo: El tiempo de cómputo de una tarea T_i es el tiempo de CPU necesario C_i para completar su ejecución en cada una de las activaciones. Este tiempo depende de la complejidad del algoritmo de control y la velocidad del procesador. No depende en absoluto de la forma en que esta tarea es planificado. El tiempo de cómputo puede variar atendiendo a distintos aspectos: código con ejecución condicional que pueden consumir distinto tiempo, el subsistema de ejecución (memoria cache y pipeline). Esto hace que la ejecución de una activación cualquiera de la tarea T_i pueda variar entre dos límites $[e_{i-}, e_{i+}]$, en el que e_{i-} es el mínimo tiempo de cómputo y e_{i+} corresponde con el máximo tiempo de cómputo. Con el fin de considerar la peor situación posible, en el análisis del sistema se considerará siempre los tiempos máximos y se denotará como tiempo de peor caso (WCET, *worst case execution time*) y se asumirá que el tiempo de cómputo de la tarea T_i es C_i que corresponde al tiempo de peor caso.

Modelo de tareas

Periodicas: Características

Tiempo de cómputo: C_i

Retardo de inicio: Una actividad de una tarea T_i tiene un retardo de inicio que corresponde con el tiempo en el cual la actividad empieza a ser ejecutada. Este retraso viene influido por la granularidad del reloj para reconocer el inicio del periodo, la ejecución del planificador, y la decisión del planificador sobre que tarea debe ejecutarse en cada instante. Adicionalmente, influye la disponibilidad de los datos leídos de los sensores externos cuando la tarea de control lee datos que pueden tener retrasos. Mientras que los factores que dependen de la ejecución del núcleo del sistema operativo de tiempo real son predecibles y despreciables, los debidos a la ejecución de otras tareas y la llegada de los datos externos pueden ser importantes y no predecibles. Este retardo se puede modelar mediante un intervalo $[r_i^-, r_i^+]$ que delimita el retardo. Este término recibe en la literatura el nombre de *jitter* de entrada.

Modelo de tareas

Periodicas: Características

Tiempo de cómputo: C_i

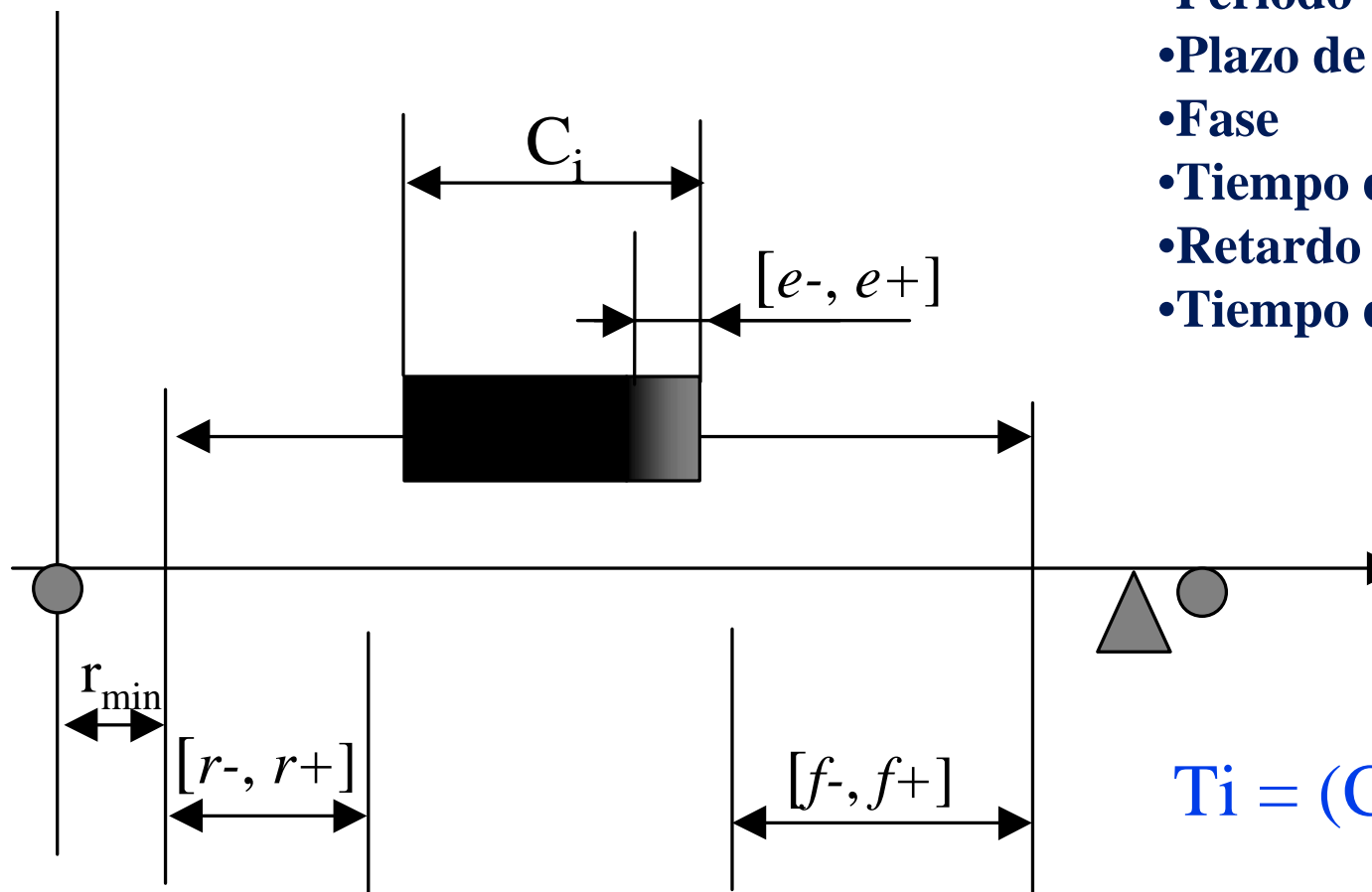
Retardo de inicio: R_i

Tiempo de finalización: El tiempo de finalización es el tiempo en el cual una actividad de una tarea T_i finaliza su ejecución. Este tiempo depende de en qué momento empezó la ejecución de la tarea, el tiempo de cómputo necesario, los retrasos que la ejecución de la actividad ha podido sufrir al acceder a datos compartidos y las posibles ejecuciones de otras tareas decididas por el planificador. El tiempo de finalización de una actividad se modela mediante un intervalo $[f_i^-, f_i^+]$ que delimita el retardo de finalización.

Modelo de tareas

Periodicas: Características

- Periodo
- Plazo de entrega
- Fase
- Tiempo de cómputo
- Retardo de inicio
- Tiempo de finalización



$$T_i = (C_i, D_i, P_i, \phi_i)$$

CARACTERISTICAS

Tamaño y complejidad

- Afecta al software
- La complejidad no sólo depende del tamaño
- Posibilidad de realizar modificaciones
- Descomposición en subsistemas pequeños y sencillos

Fiabilidad y seguridad

- Un fallo puede tener repercusiones muy importantes: pérdidas humanas, económicas, etc.
- Es necesario que si el sistema falla se le conduzca a estados seguros que sean previsibles, y se tengan los medios para solucionarlo.

Concurrencia

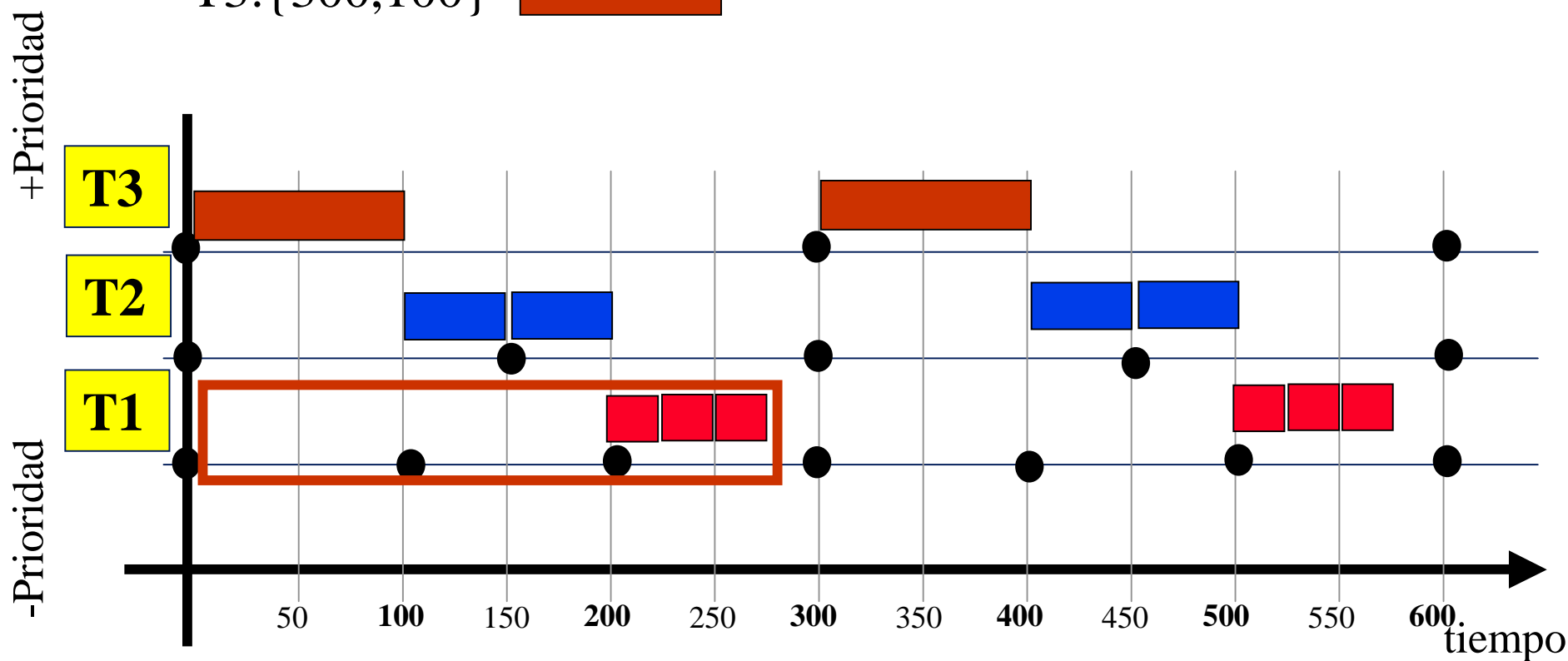
- Programación concurrente
- Distintas tareas accediendo a recursos comunes
- Restricciones temporales
- Planificación de tareas atendiendo a las restricciones temporales

Dificiles de especificar y validar

Ejecución de tareas

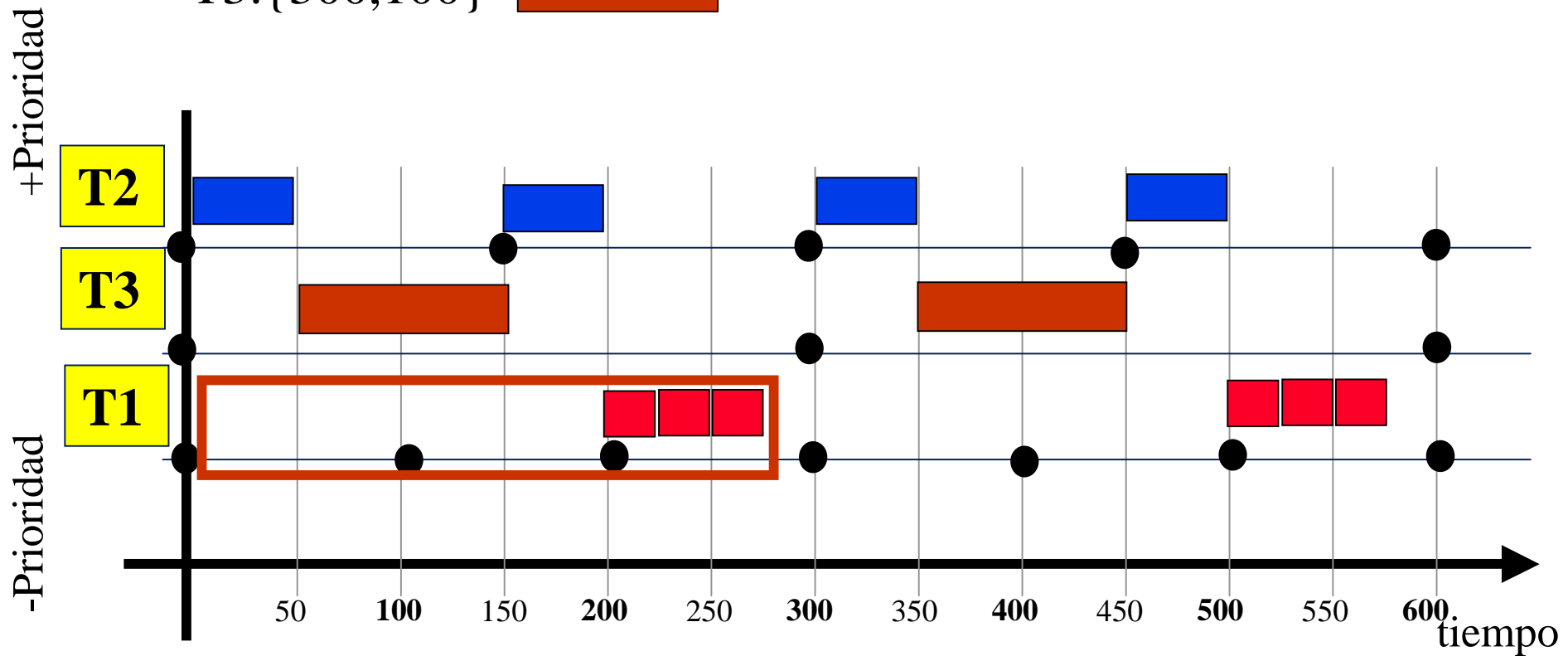
- T1: { 100, 20 } ■
- T2: { 150, 40 } ■
- T3: { 300, 100 } ■

Planificación basada en prioridades

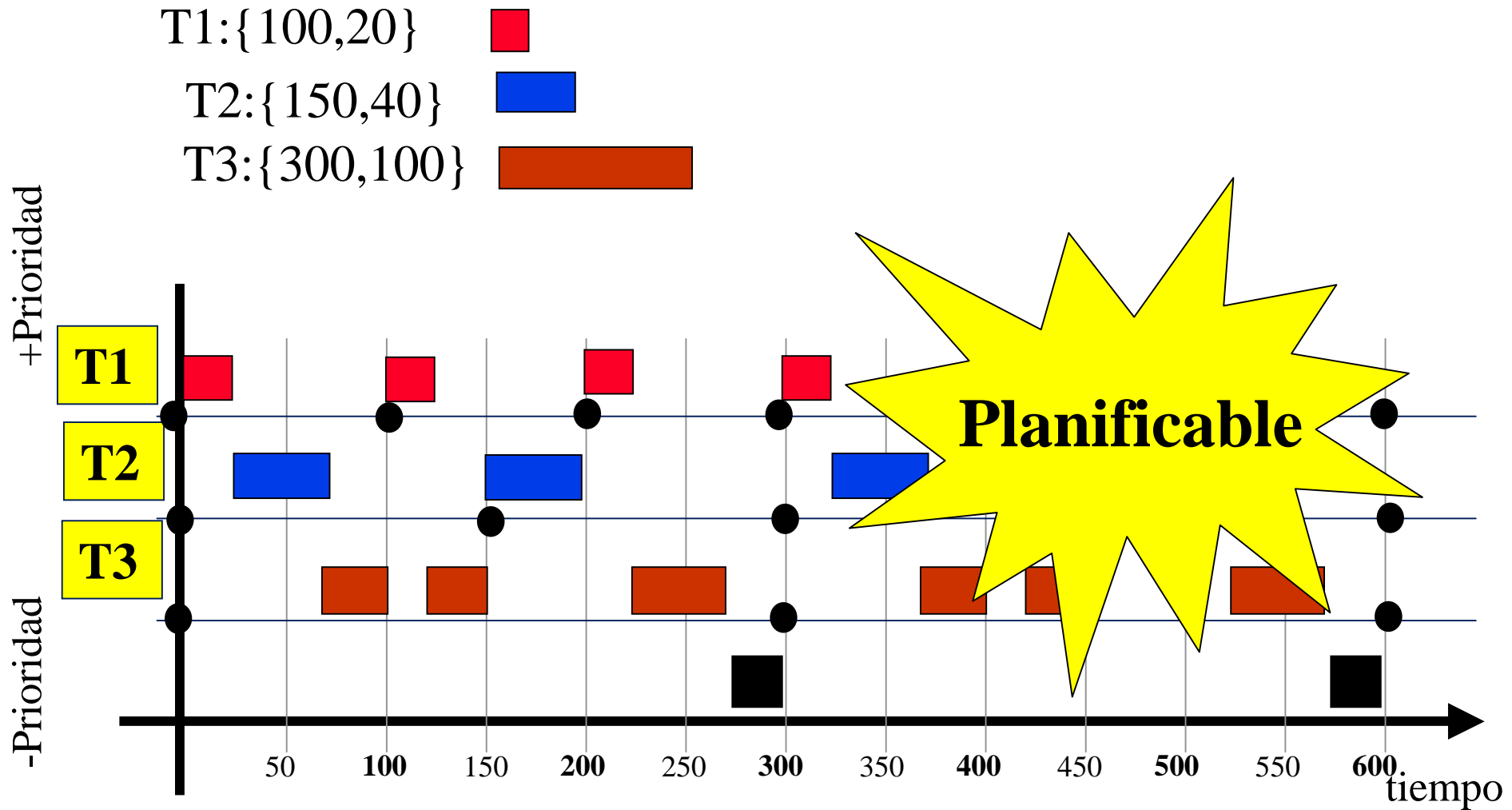


Ejecución de tareas

T1: { 100, 20 } ■
 T2: { 150, 40 } ■
 T3: { 300, 100 } ■



Ejecución de tareas



LENGUAJES DE PROGRAMACION

Características necesarias:

- Concurrencia: abstracciones para la creación de tareas
- Mecanismos de comunicación y sincronización entre tareas
- Mecanismos para gestionar recursos compartidos.
- Acceso a bajo nivel
- Gestión de dispositivos
- Tolerancia a fallos: Manejo de situaciones excepcionales y erróneas
- Transportabilidad: independencia del sistema operativo
- Eficiencia

LENGUAJES DE PROGRAMACION

Lenguajes ensambladores

- Eficiente
- Costoso
- Difícil de mantener
- No transportable

LENGUAJES DE PROGRAMACION

Lenguajes secuenciales (Fortran, Pascal, C, PL/M,...)

- Un único flujo de control
- Buenos compiladores
- Eficientes
- Requieren el soporte del s.o. para desarrollar aplicaciones concurrentes

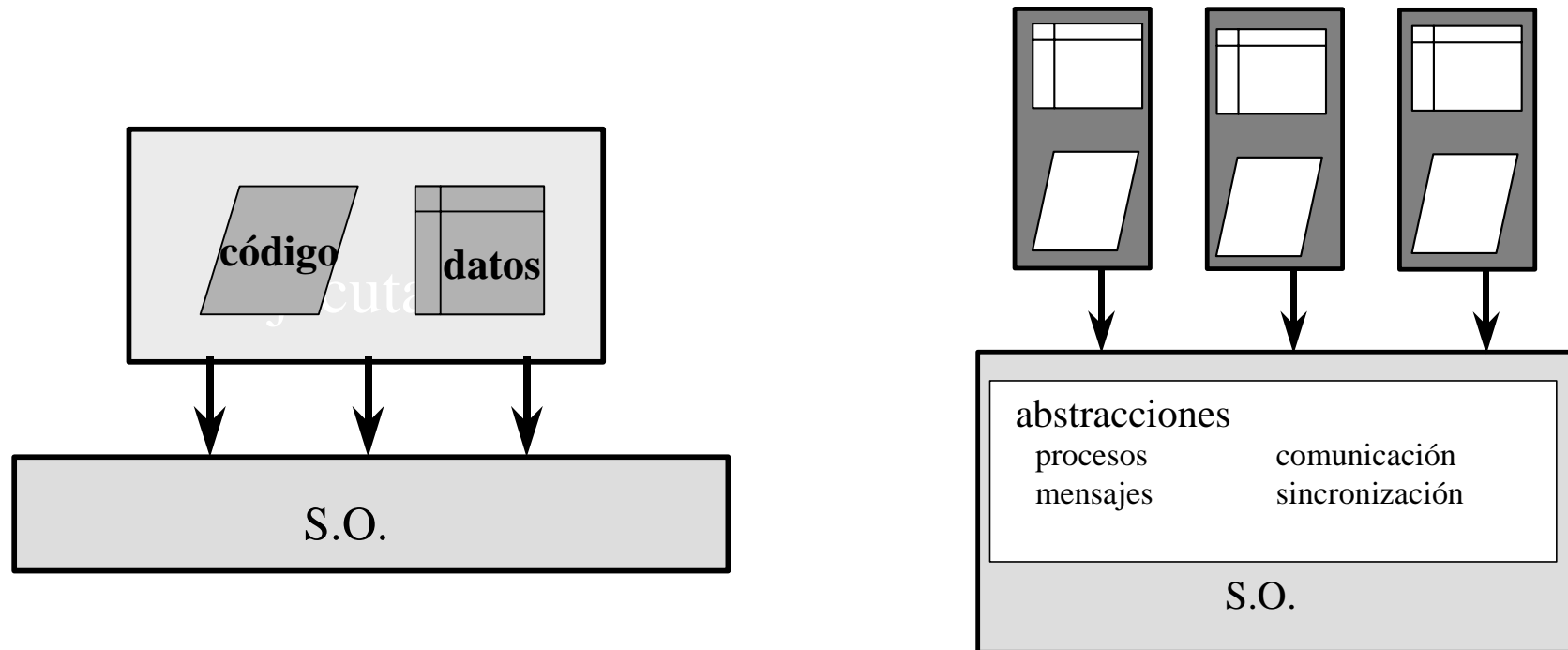
LENGUAJES DE PROGRAMACION

Lenguajes secuenciales (C)

- Lenguaje estructurado y muy flexible
- No es fuertemente tipado (poco seguro)
- Baja legibilidad
- Eficiencia
- Disponibilidad de un gran número de compiladores y herramientas para el desarrollo de programas.
- No ofrece concurrencia ni tiempo real.
- Puede integrarse con un sistema operativo que dé los servicios

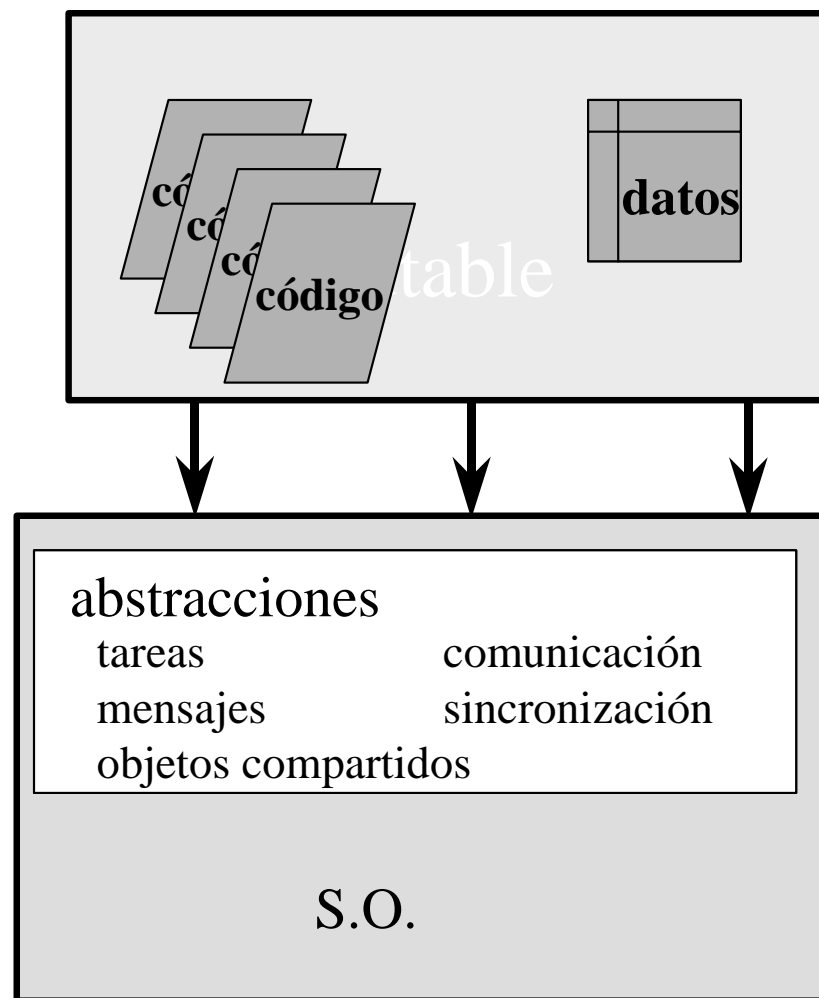
LENGUAJES DE PROGRAMACION

Lenguajes secuenciales (C)



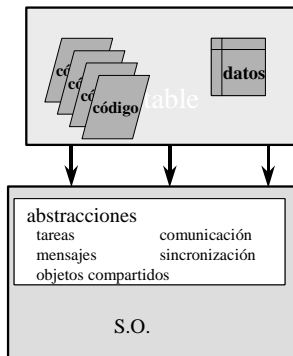
LENGUAJES DE PROGRAMACION

Lenguajes secuenciales (C)



LENGUAJES DE PROGRAMACION

Lenguajes secuenciales (C)



```
#include <pthread. h>
....
pthread_mutex_t lock;
.....
void *productor(void *arg){
    int num = (int) arg;
    int i;
    for (i= 0; i<num; i++)
        Produce(i);
    pthread_exit( 0);
}

void *consumidor(void *arg){
    int num = (int) arg;
    int i;

    for (i= 0; i<num; i++)
        Consumir(i);
    pthread_exit(0);
}

main()
{
    pthread_mutex_init (& lock, NULL);
    .....
    pthread_create(&th_a, NULL, productor, N_Elem);
    pthread_create(&th_a, NULL, consumidor, N_Elem);
    .....
    exit( 0);
}
```

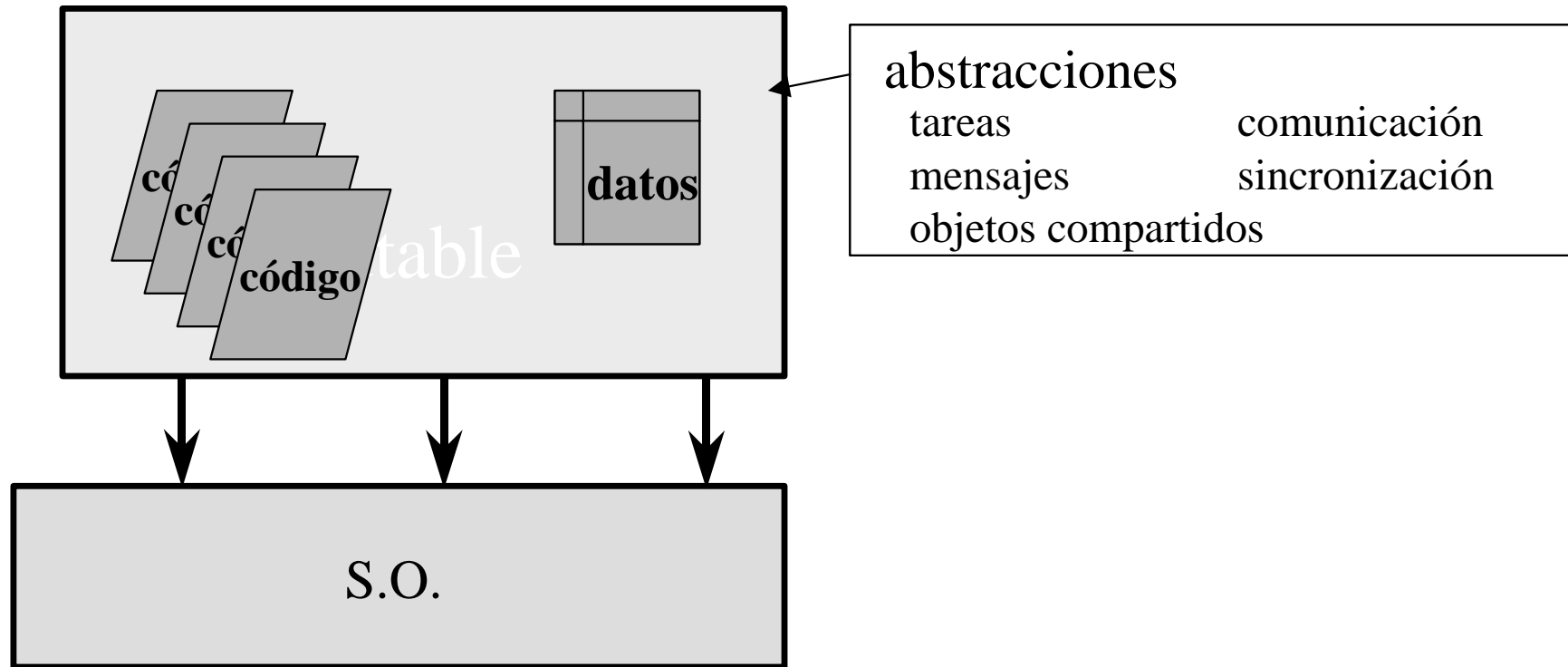
LENGUAJES DE PROGRAMACION

Lenguajes concurrentes (Ada, Modula-2, Euclid, Pearl, Java, RTJava ...)

- Incorporan las abstracciones necesarias
 - gestión de tareas
 - comunicación y sincronización
 - recursos compartidos

LENGUAJES DE PROGRAMACION

Lenguajes concurrentes (Ada, Modula-2, Euclid, Pearl, Java, RTJava ...)



LENGUAJES DE PROGRAMACION

Lenguajes concurrentes (Ada)

Diseñado para sistemas de tiempo real empotrados

Ofrece gestión del tiempo

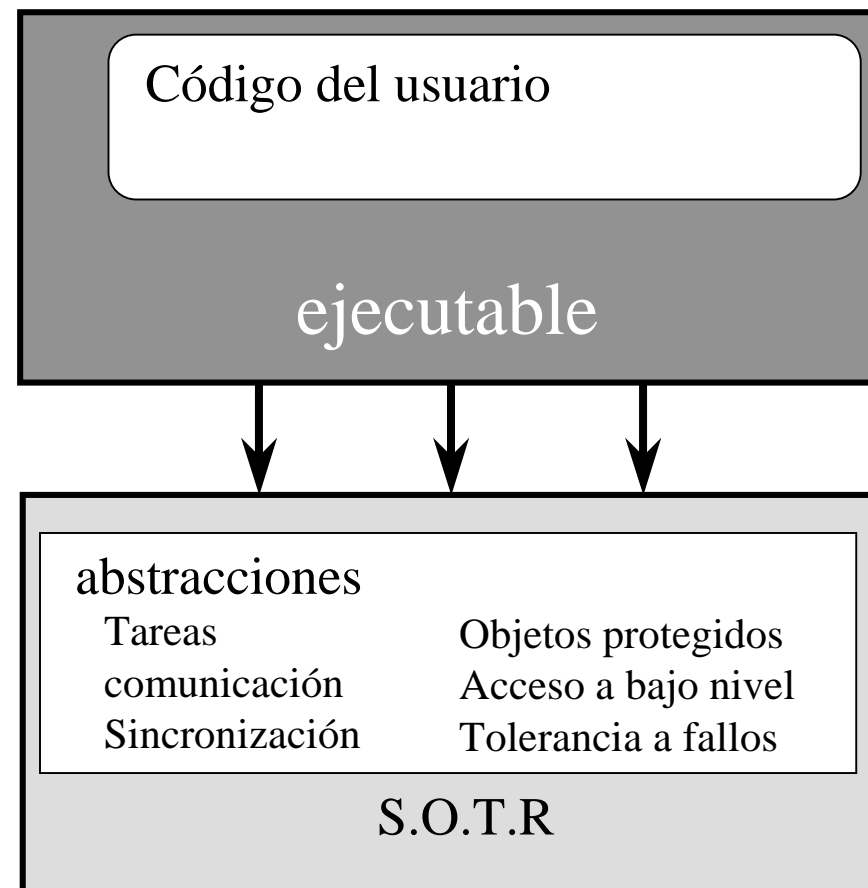
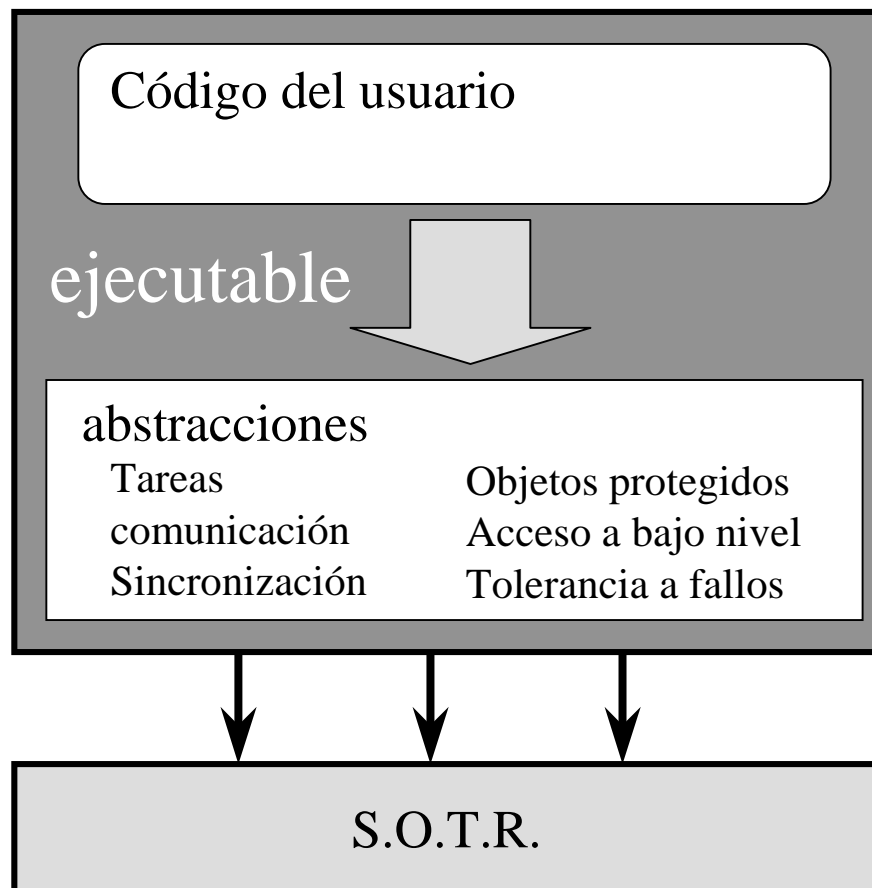
Mecanismos para hacer sw fiable y robusto

Planificación expulsiva basada en prioridades fijas

Protocolos de gestión de recursos compartidos

Acceso a bajo nivel

Secuenciales/concurrentes



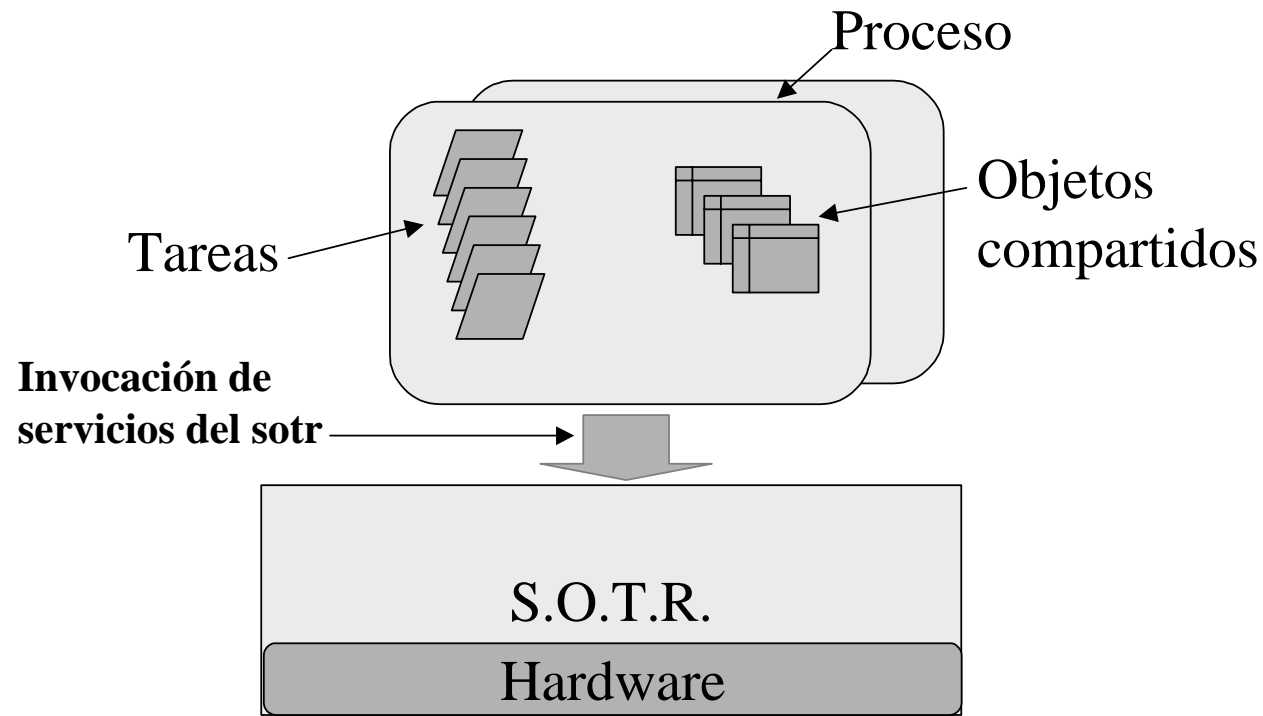
Sistema de ejecución

Ha de realizarse sobre sistemas operativos de tiempo real

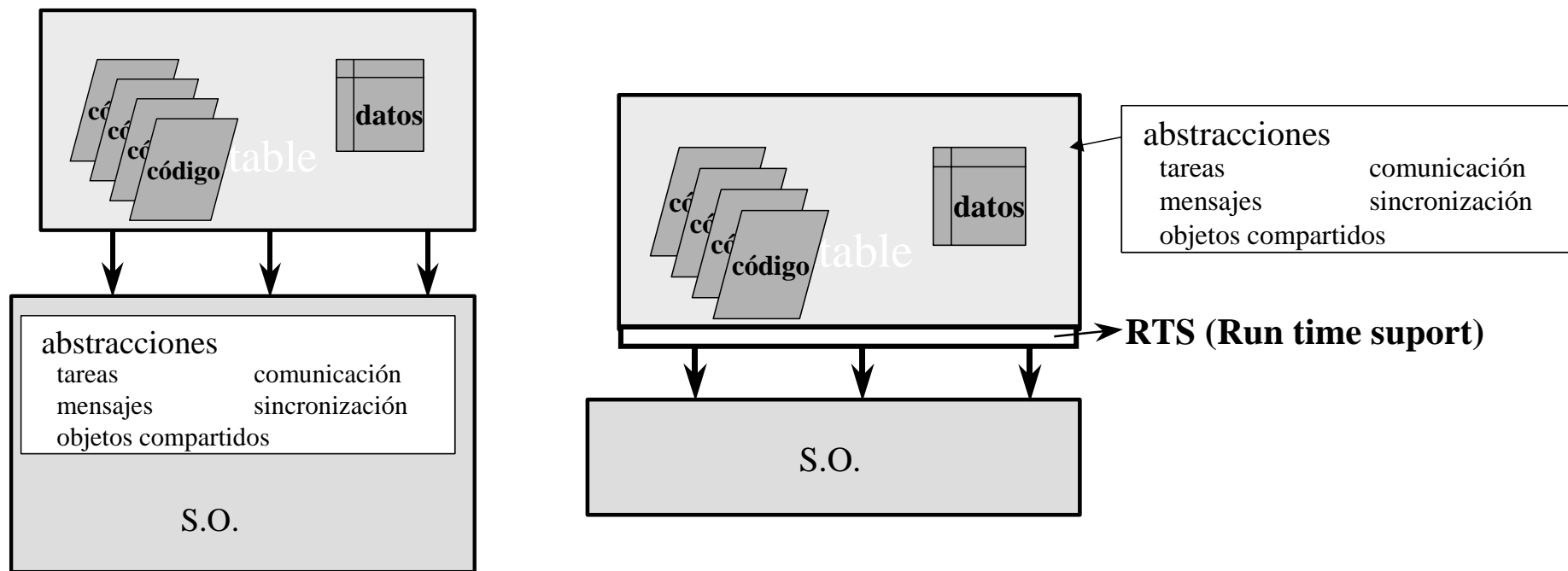
Los s.o. de propósito general no ofrecen predecibilidad

Eficientes y configurables

Sistema de ejecución



Sistema de ejecución

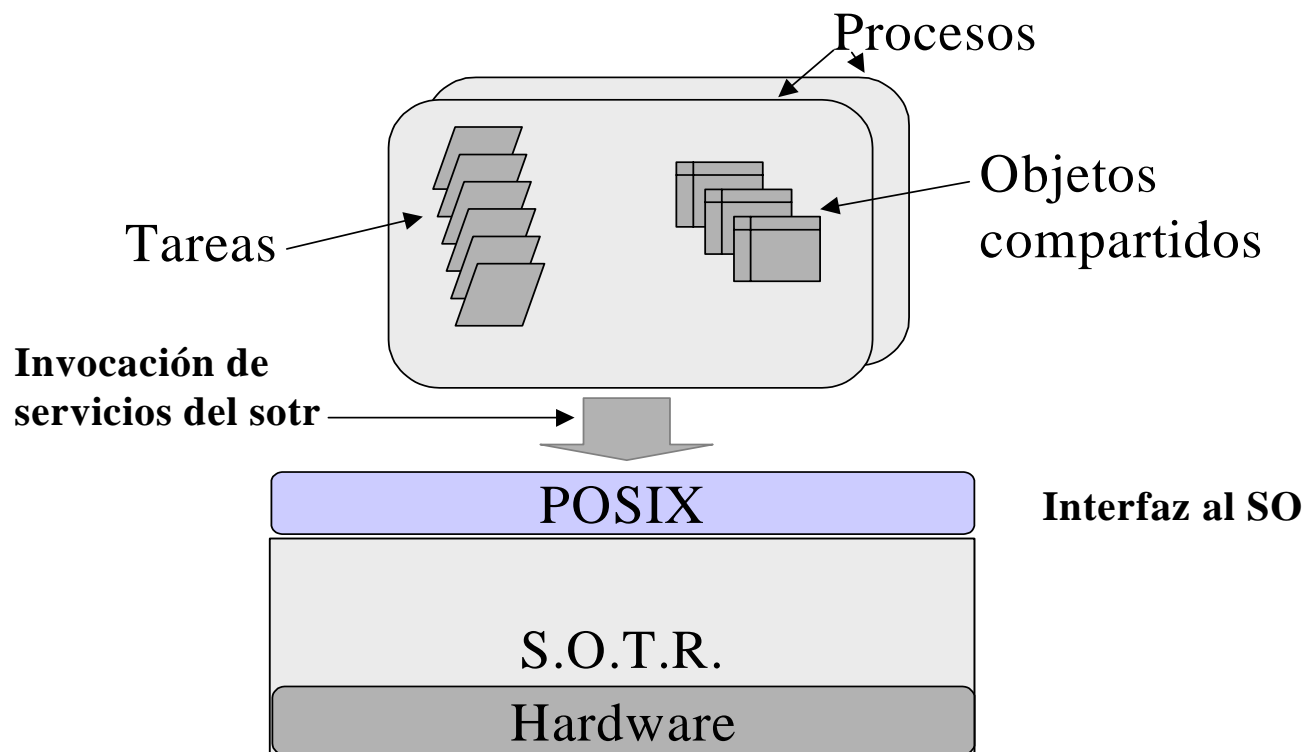


Lenguajes secuenciales

Lenguajes concurrentes

Sistema de ejecución

POSIX (estándar 1003, Portable Operating System Interface basado en unIX) es un conjunto de normas IEEE/ ISO que definen interfaces de sistemas operativos y permiten desarrollar software portátil y reutilizable



Sistema de ejecución

- POSIX. 1a Unix básico.
- POSIX. 1b, 1d, 1o, 1j Extensiones de tiempo real
- POSIX. 1c Extensiones de tareas (threads)
- POSIX. 5 Enlaces con el lenguaje Ada

El objetivo de las extensiones de tiempo real es añadir a POSIX básico los servicios que se necesitan para conseguir un comportamiento predecible y facilitar la programación concurrente. Estos servicios permiten:

- Sincronización entre procesos, Compartición de memoria, Gestión de la memoria virtual, Señales para tiempo real, Definición de relojes y temporizadores, Colas de mensajes, Entrada/Salida síncrona y asíncrona.

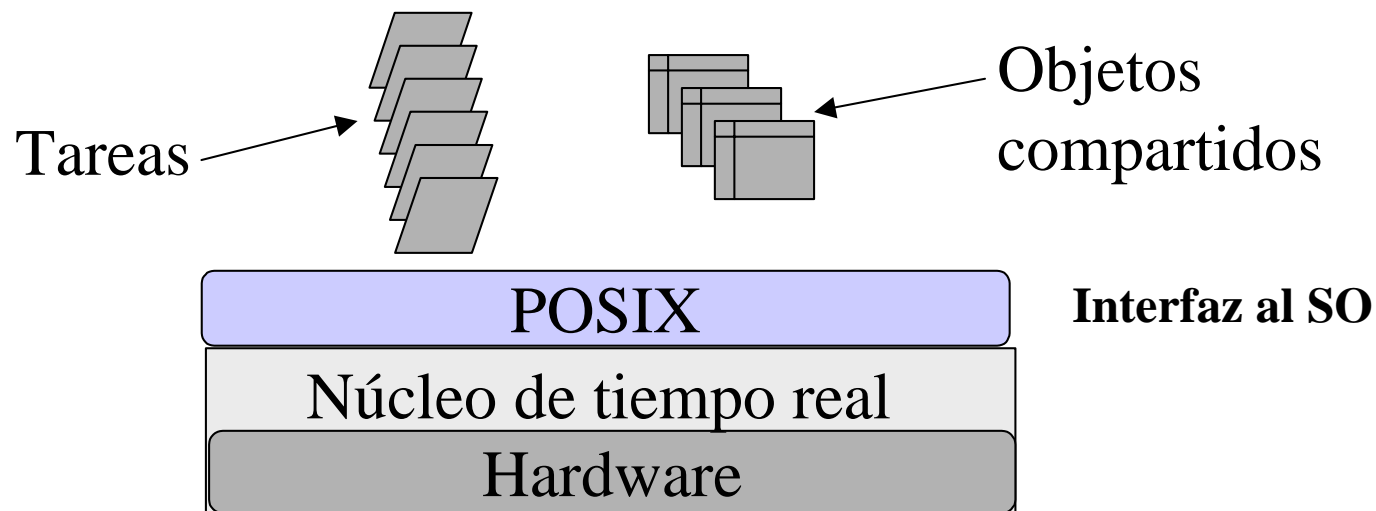
Perfiles

- Controlador de tiempo real: tiene sistema de ficheros y terminal
- Sistema de tiempo real dedicado: tiene gestión de memoria y procesos pesados
- Sistema de tiempo real generalizado: sistema completo con todo tipo de servicios

Sistemas empotrados

Un sistema empotrado es un componente basado en un computador que incorpora un sistema operativo mínimo y la aplicación.

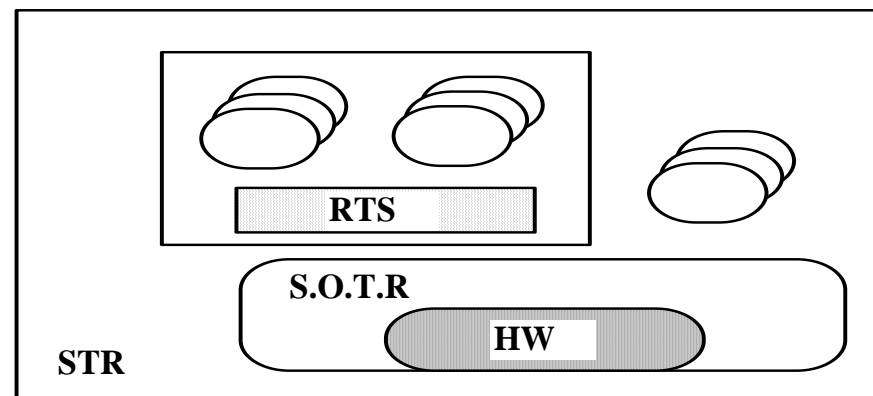
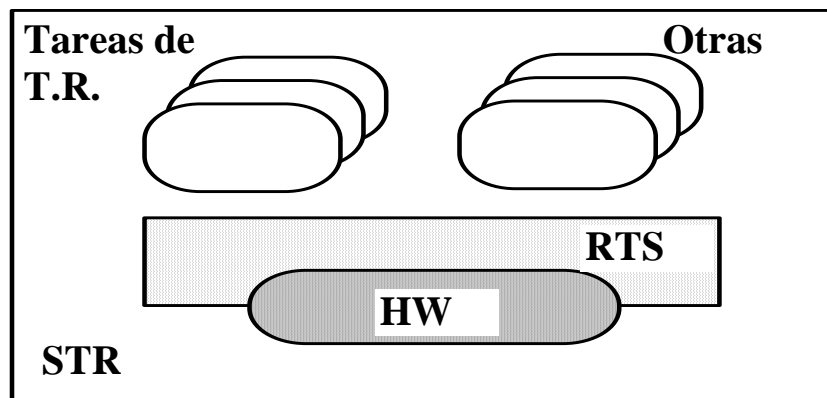
- Automóviles
- Electrónica de consumo: teléfonos, radios, televisores
- Electrodomésticos
- Periféricos de computador



Sistemas empotrados

Soporte de ejecución específico

Los lenguajes que permiten concurrencia pueden crear su propio soporte de ejecución (en inglés *run time support*) que sustituye (o puede sustituir) al sistema operativo.



Núcleos y SOTR

Allegro.

Helios (Transputer,C40,ARM).

INtime for Windows NT by Radisys.

iRMX by Radisys.

Lynx OS (newsgroup).

OS/9 and OS-9000: Users Group in Chicago,

pSOS+.

QNX-RTOS

QNX/Neutrino real-time POSIX kernel.

RTEMS is FREE! (Real-Time Executive for Military Systems).

RTX. (RTXDOS-16 (for embedded PCs) RTXDOS-32 (WIN32 API)

TXS-32

Real-time Extension (Windows NT for real-time and embedded computing).

VRTX/OS 3.0.

VxWorks (newsgroup).

Conclusiones

- Qué es un sistema de tiempo real
- Características
- Tipos de actividades
- Organización
- Lenguajes de programación de str
- Formas de obtener un s.t.r.