Control Digital - Maestría en Sistemas Embebidos

Sistemas de tiempo real - Introducción

20 de marzo 2023

Índice

Definiciones

Sistemas de Tiempo Real

Un sistema en tiempo real es aquel en el cuál la exactitud del sistema depende no solamente en los resultados lógicos de computación si no también del tiempo en el cuál esos resultados son producidos.

En un sistema en tiempo real *hard*, es absolutamente imperativo que la respuesta ocurra dentro del límite temporal requerido (*e.g.* apliucaciones *safety-critical* en aeroespacial, automóviles, etc.).

Un sistema en tiempo real *soft* es aquel en el cual los tiempos límites son importantes, pero el sistema puede seguir funcionando si ocasionalmente se supera ese tiempo (*e.g.* sistemas multimedia, interfaces de usuario, etc.)

Hard Real-time Systems

- La mayoría de los *hard real-time systems* son sistemas de control en tiempo real.
- La mayoría de los sistemas de control en tiempo real **no** son *hard real-time systems*.
- Muchos *hard real-time systems* son *safety-critical*.

Tiempo real no indica que sea necesario realizar computaciones en alta velocidad. Los sistemas en tiempo real se ejecutan a la velocidad que les permite cumplir con los requerimientos de tiempo.

Sistemas de control en tiempo real (RTCS)

Dos tipos de sistemas de control en tiempo real:

- Sistemas Embebidos
  - sistema de control dedicado.
  - microprocesadores, real-time kernels, RTOS.
  - aeroespacial, robots industriales/domésticos, sistemas vehiculares, sistemas médicos.
- Sistemas de control industrial
  - Sistemas de Control Distribuidos (DCS).
  - Lógica de Control Programable (PLC).
  - organizados jerárquicamente.
  - industria de procesos, industria manufacturera.

Características

Características del control embebido

- Tienen recursos de cómputo y comunicación limitados:
  - usualmente aplicados a productos de consumo masivo, como los autos.
  - Limitaciones en tiempo de cómputo, ancho de banda en la comunicación, energía y memoria.
- Operación autónoma:
  - no hay un operador humano en el lazo de control.
  - necesitan especificaciones formales.

Recursos limitados $\Rightarrow$ Exige eficiencia en tamaño del código, tiempo de ejecución, consumo de energía, costo.

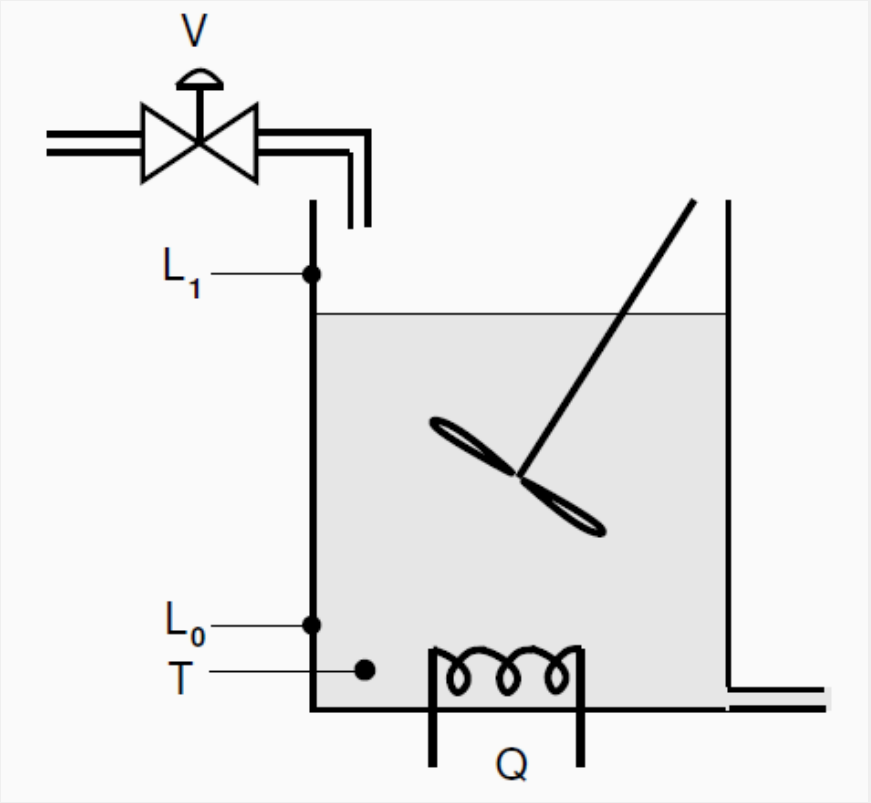
Operación autónoma $\Rightarrow$ Exige confianza - Fiabilidad, Disponibilidad, Seguridad, Mantenibilidad.

Ejemplo: Tanques de Inercia

Mezclar y calentar un fluido

Objetivos de control:

- Control de nivel: Abrir *V* cuando el nivel está por debajo de L_0, mantener la válvula abierta hasta que el nivel llegue a L_1.
- Regular la velocidad de giro para la paleta de mezclado.
- Controlar la temperatura.



Características típicas

- Actividades paralelas.
- Requerimientos de tiempo: más o menos duros.

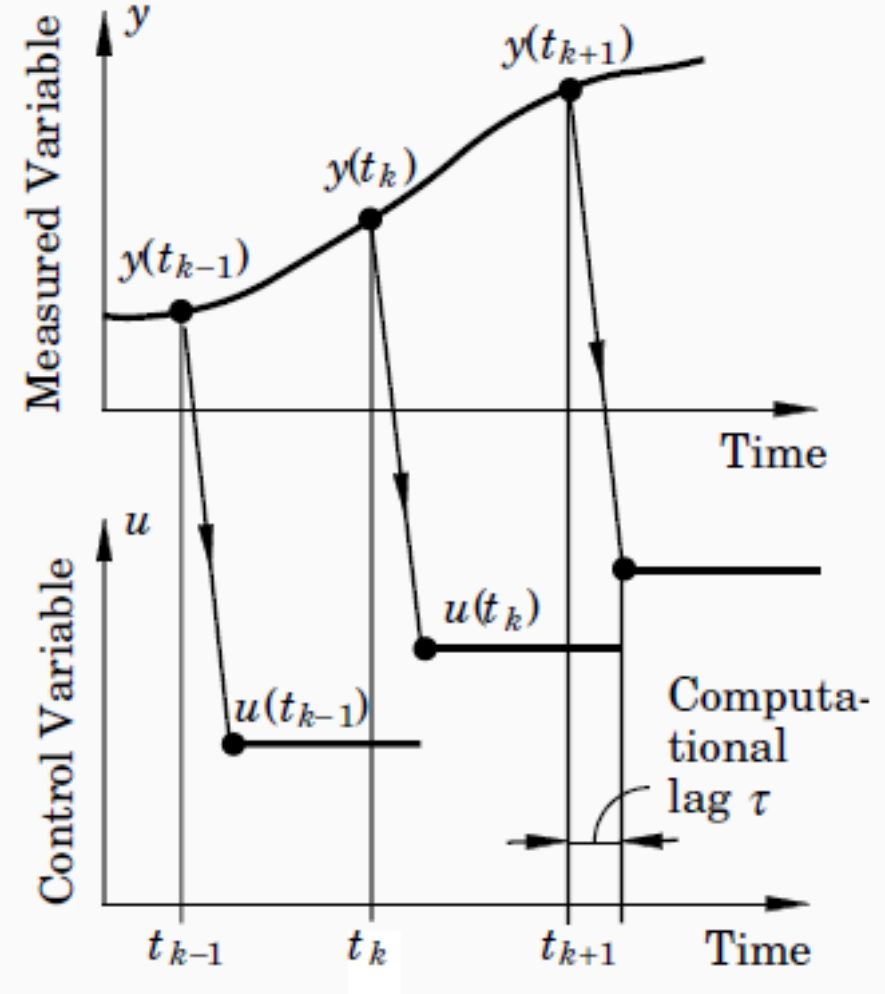
Cómo llevar adelante el control?

Muy frecuentemente siguiendo los pasos de muestreo, control, actuación:

- Muestreo de la señal medida $y(t)$.
- Cálculo de la señal de control (algoritmo de software).
- Actuación de la acción de control calculada $u[k]$.

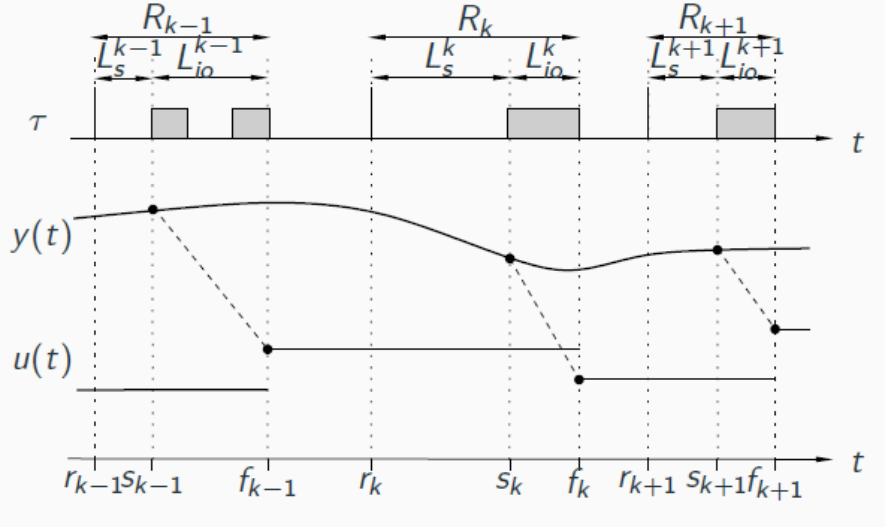
En casi todos los casos de manera periódica, *i.e.* dependientes del *clock*

Sincronización ideal del control



- La salida $y(t)$ muestreada periódicamente en instantes de tiempo $t_k = kh$,
- Control $u(t)$ generado luego de un retardo de tiempo τ.

Sincronización real del controlador



- La tarea de control τ es iniciada periódicamente en los instantes de tiempo $r_k = kh$,
- La salida $y(t)$ se realiza luego de una *latencia de muestreo*, L_s, variante en el tiempo,
- La señal de control $u(t)$ se genera luego de una *latencia de entrada-salida*, L_{io}.

Se produce una sincronización no determinística.
¿Cómo podemos minimizar ese no-determinismo?

Control por eventos discretos

Metodología de trabajo con eventos:

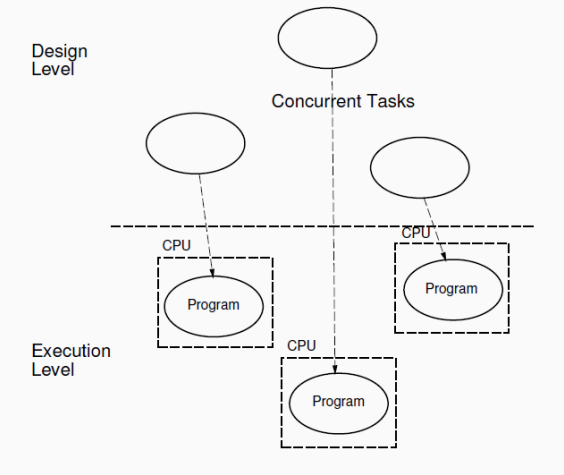
- Esperar que una condición sea verdadera o a que ocurra un evento,
- realizar algunas acciones,
- esperar nuevas condiciones.

El evento puede ser un tick del clock.
Los sistemas en tiempo real deben responder a eventos, muchos eventos pueden ocurrir al mismo tiempo (paralelismo).

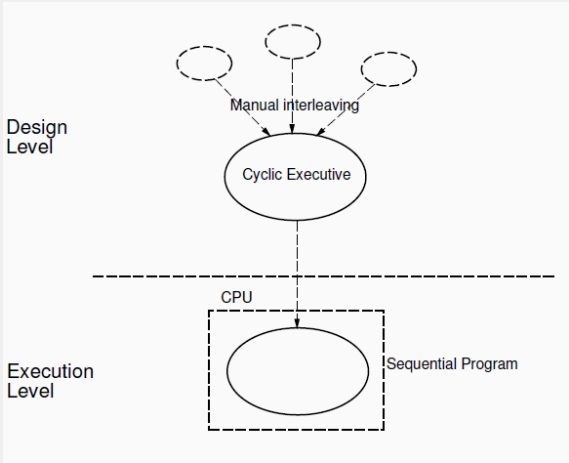
Paradigmas

Paradigmas de Programación

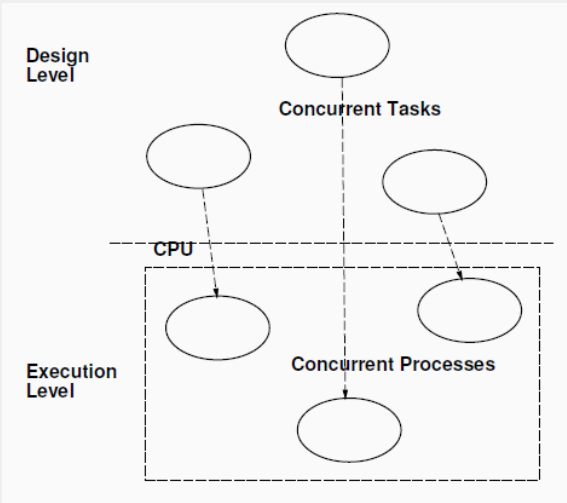
Programación multi-core paralela



Programación secuencial



Programación concurrente

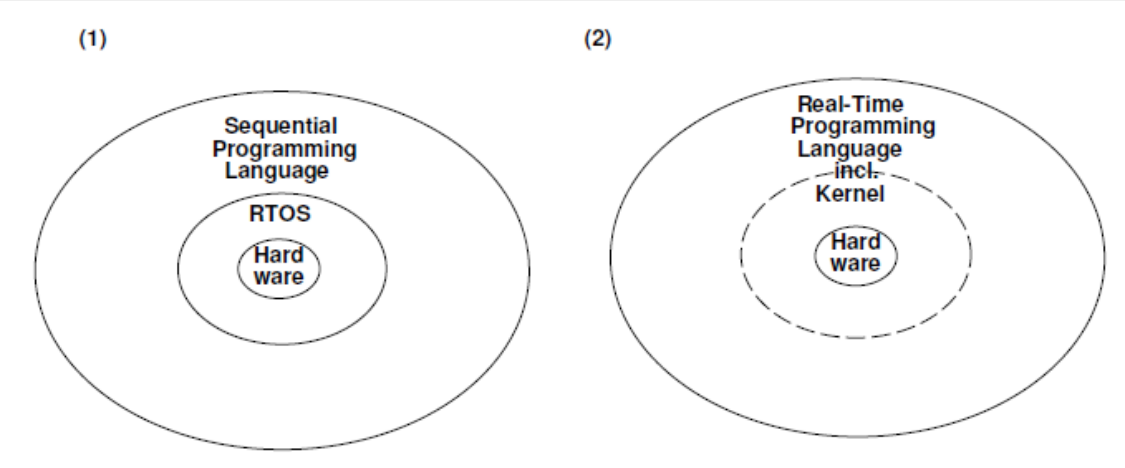


¿Cuál paradigma de control elegirían para los RTCS?

Pogramación concurrente

La programación concurrente puede ser realizada de diferentes maneras:

- Utilizando un Real-Time Operating System (RTOS). En este caso los procesos son implementados en un lenguaje de programación secuencial como C. La funcionalidad de tiempo real puede ser alcanzada a partir de llamadas a las primitivas de tiempo real provistas por el sistema operativo. La parte del RTOS que es responsable del manejo del proceso es el Real-time Kernel.
- Utilizando un sistema de programación en tiempo real. En este caso el lenguaje en sí mismo o el sistema donde corren proveen el real-time kernel. Este enfoque es utilizado por ejemplo en Ada.



Referencias*

Para esta clase me basé mucho en
- El libro: Arzén, Karl-Erik, (2015), *Real-time Control Systems*, Department of Automatic Control, Lund University, Lund, Sweden.
- Las clases asociadas al curso que se da en la universidad de Lund: Real-time Systems