INGENIERÍA DE SISTEMAS Y AUTOMÁTICA

Universidad de Oviedo

Control por Computador

Ignacio Alvarez García

Octubre - 2011

Indice

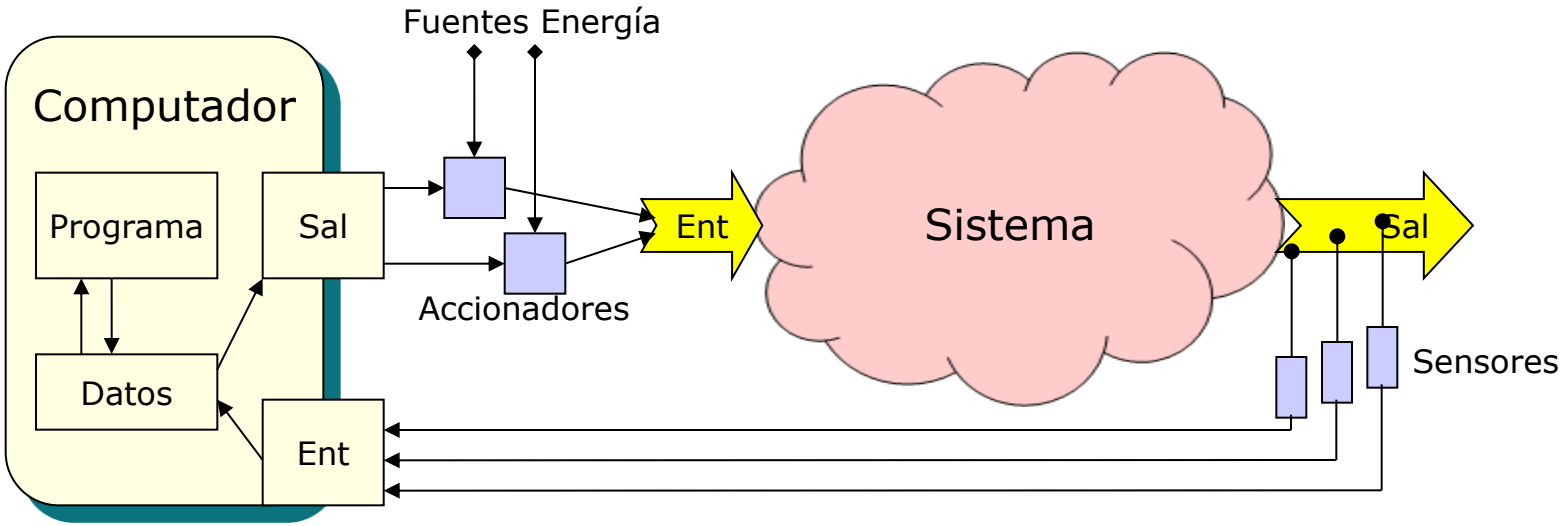
- Introducción control de procesos por computador
- Las matemáticas del control
- Programación del lazo de control
- Programación en lenguaje C
- Implantación del control en el computador
- El control secuencial

Indice

- **Introducción control de procesos por computador**
- Las matemáticas del control
- Programación del lazo de control
- Programación en lenguaje C
- Implantación del control en el computador
- El control secuencial

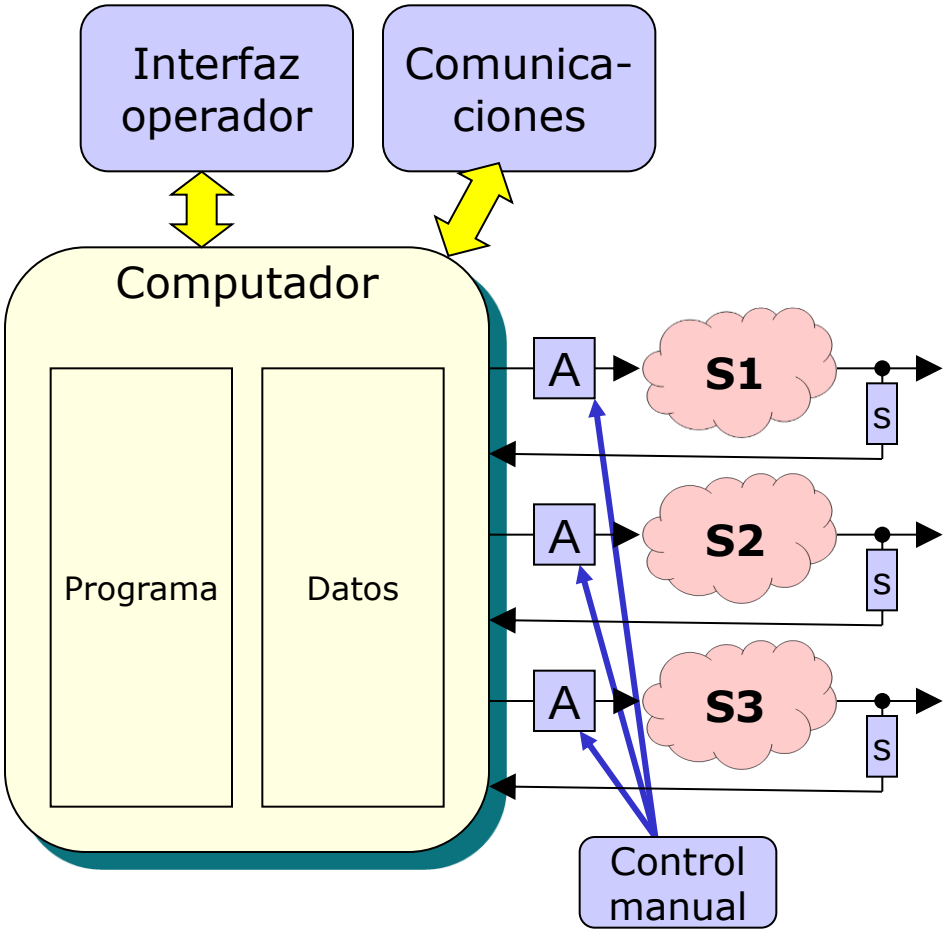
El Control de Procesos por Computador

- Proceso o sistema
  - Conjunto de elementos físicos que cumplen un cometido común
  - **Salidas del sistema**: valores medibles cuya variación en el tiempo se desea controlar
  - **Entradas al sistema**: acciones que se pueden realizar para modificar los valores de las salidas
- Control de proceso por computador
  - Un **Computador** lee las Salidas del sistema, realiza cálculos de control, y modifica las Entradas del sistema de acuerdo con estos cálculos
  - Operación en **Tiempo Real**: es importante que la acción de control sea correcta y realizada en tiempo



El Control de Procesos por Computador

- Elementos de un equipo de control



- **Sistemas:**
  - Conjuntos de elementos físicos a controlar
- **Computador:**
  - El “cerebro” del control
  - Programa y datos alojados en memoria
- **Sensores:**
  - Dan al computador información del estado de los sistemas, midiendo sus salidas.
- **Accionadores:**
  - Permiten al computador generar cambios sobre los sistemas, modulando la energía entregada por fuentes externas
- **Interfaz operador:**
  - Panel(es) que permite(n) al humano intervenir en el control, generando órdenes y recibiendo informaciones
- **Comunicaciones:**
  - Interconexión con otros computadores para intercambio de informaciones y comandos
- **Control manual:**
  - Reemplaza las salidas del computador en caso de error o alarma

Ejemplos de sistemas

Control de temperatura en horno-túnel

Robot de encolado automático

Teléfono móvil

Control de vuelo de un avión

Línea de fabricación de botes

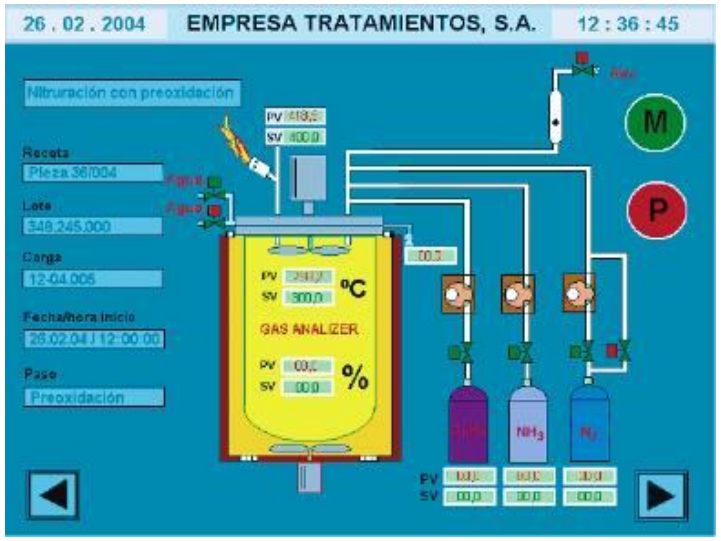
Sensores

- Convierten magnitudes físicas en valores de pequeña tensión/corriente.
- Clasificaciones:
  - Según la magnitud medida:
    - De posición (lineal/angular)
    - De velocidad (lineal/angular)
    - De temperatura
    - De presión
    - De presencia
    - …
  - Según el formato del valor entregado:
    - Analógicos
    - Digitales todo/nada
    - De tren de pulsos
    - Numéricos
- Para obtener el valor medido:
  - Adaptar señal del sensor (electrónica)
  - ValorLeido = LeerDispositivoDeEntrada() ;
  - MagnitudFísica = Cálculo (ValorLeido) ;

Accionadores

- Permiten el comando del sistema mediante la energía procedente de una fuente externa, modulada por un valor de pequeña tensión/ corriente.
- Clasificaciones:
  - Según la energía:
    - Eléctricos
    - Neumáticos
    - Hidráulicos
    - …
  - Según la acción:
    - Desplazamiento
    - Giro
    - Calentamiento
    - Compresión
    - …
  - Según la modulación:
    - Analógicos
    - Digitales todo/nada
    - PWM
    - Numéricos
- Para entregar la energía deseada:
  - ValorAEscribir = Cálculo (ValorDeseadoEnergía) ;
  - EscribirSalida (ValorAEscribir) ;
  - Adaptar/amplificar salida (electrónica)

Interfaz con el operador



Interfaz tipo SCADA

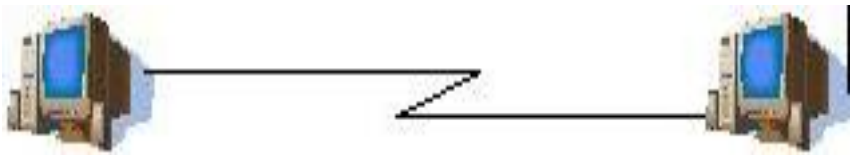
Pupitre / botonera

Display gráfico personal

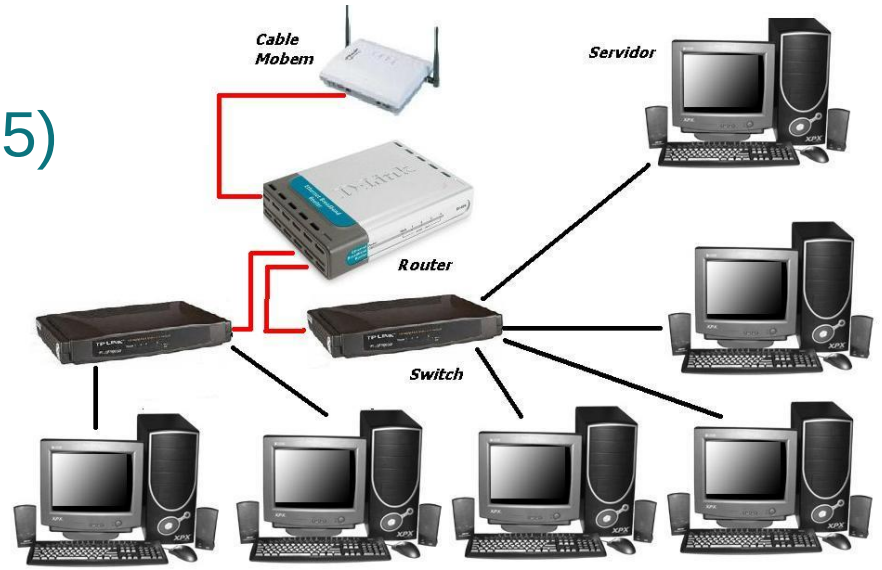
- Dispositivos salida interfaz humana:
  - Indicador luminoso (LED)
  - Display (7 segmentos, LCD,…)
  - Pantalla (texto/gráficos)
  - Impresora
  - …
- Dispositivos entrada interfaz humana:
  - Botón / pulsador
  - Selector rotativo
  - Teclado
  - Ratón / Puntero / Joystick
  - …

Comunicaciones

- Punto a punto:
  - Serie (RS-232,USB)
  - Paralelo
  - Inalámbrica (Bluetooth)

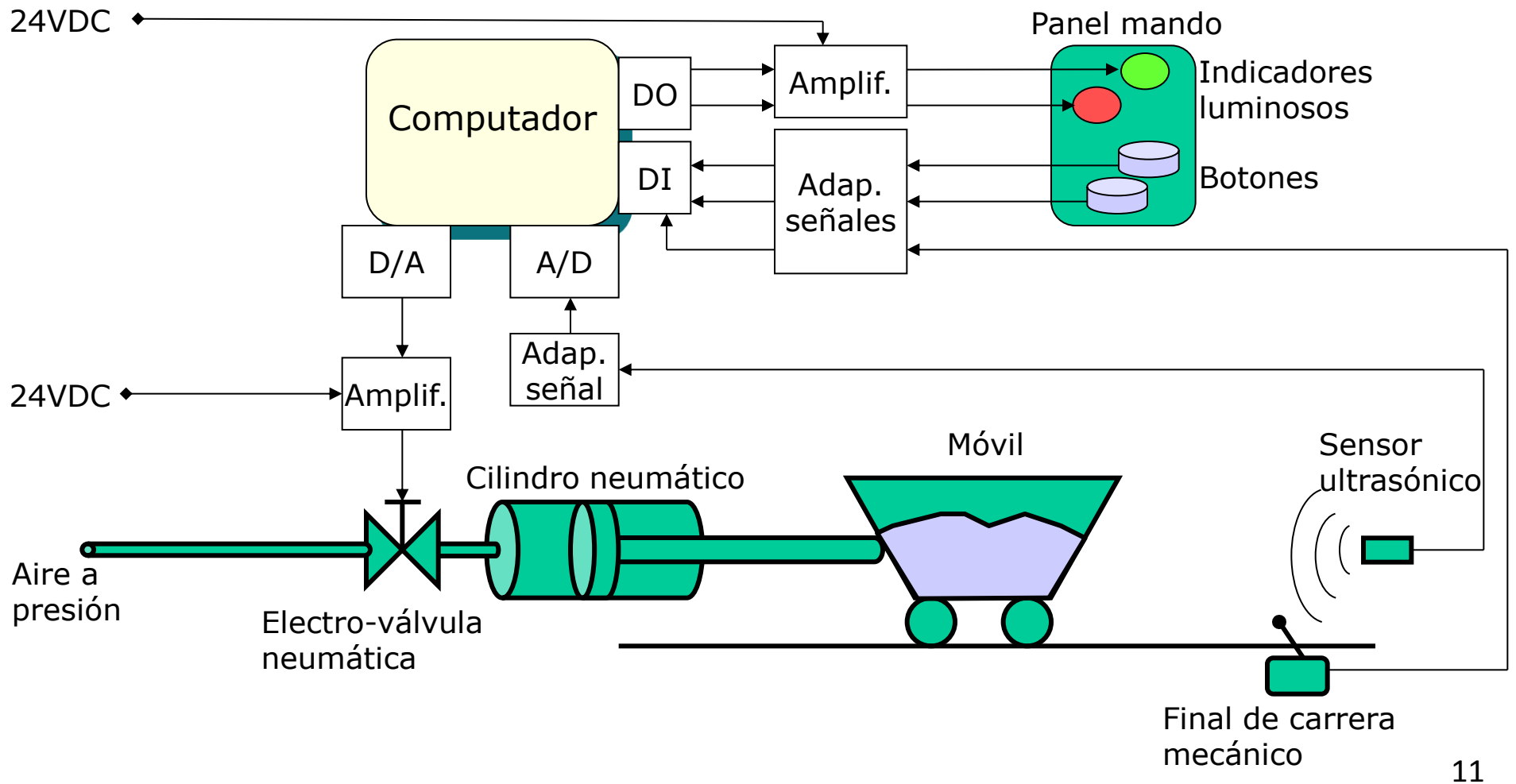


- Multipunto:
  - Serie (I2C, CAN, RS-485)
  - Red (Ethernet)
  - Inalámbrica (Wifi)



Ejemplo de sistema de control

- Control del posición de una carretilla



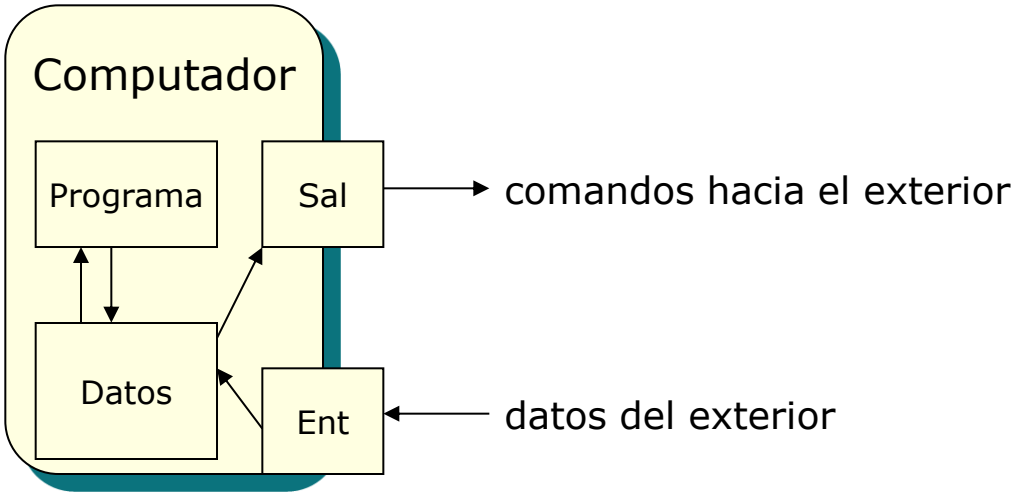
Niveles de control

- Control secuencial:
  - Genera la secuencia de acontecimientos deseada
  - Determina los valores deseados de las variables a controlar
  - Diagrama de estados / transiciones
  - Cambio de estados por:
    - Tiempo máximo
    - Combinación de valores de entradas

- Lazo de control:
  - Asegura que las variables a controlar alcancen sus valores con precisión y velocidad
  - Compara la **variable a controlar** con una **consigna**, y modifica la **acción de control** para que se igualen

Computadores para control

- El computador es capaz de:
  - Aceptar información digital procedente del exterior a través de sus entradas.
  - Almacenar la información (datos) en su memoria.
  - Procesar los datos de acuerdo a un programa (secuencia de instrucciones sencillas) instalado en su memoria.
  - Generar comandos hacia el exterior a través de sus salidas.



Computadores para control

- Microcontrolador

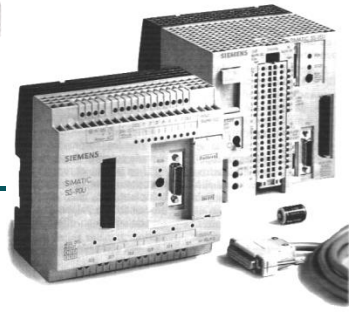
- Procesador Digital de Señal (DSP)

- Dispositivos Electrónicos Programables (FPGA, PLD)

- Computador embebido

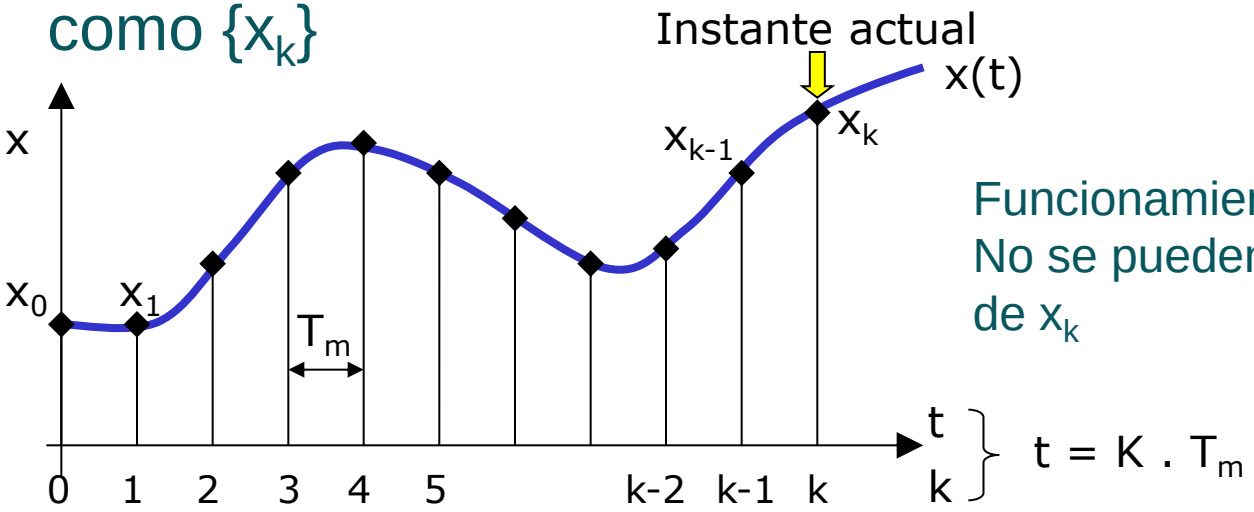
- Ordenador Industrial

- Autómata Programable (PLC)



Variables continuas y discretas

- En el mundo real, las variables son continuas y analógicas.
- En el computador, las variables son discretas y digitales.
- Se representa una señal analógica como x(t)
- Se representa una secuencia de valores discretos como $\{x_k\}$

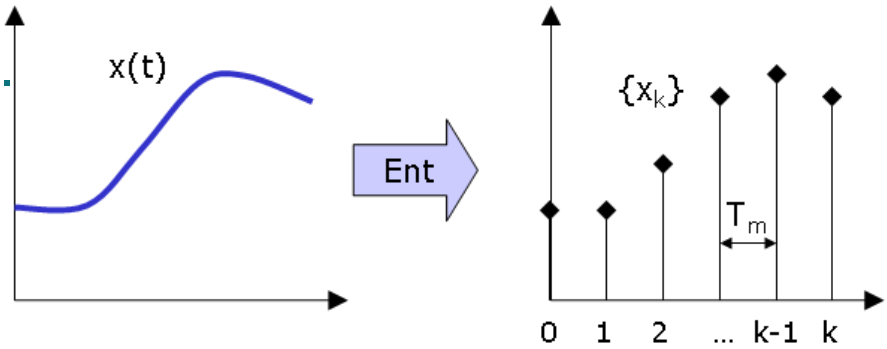


$t = K \cdot T_m$

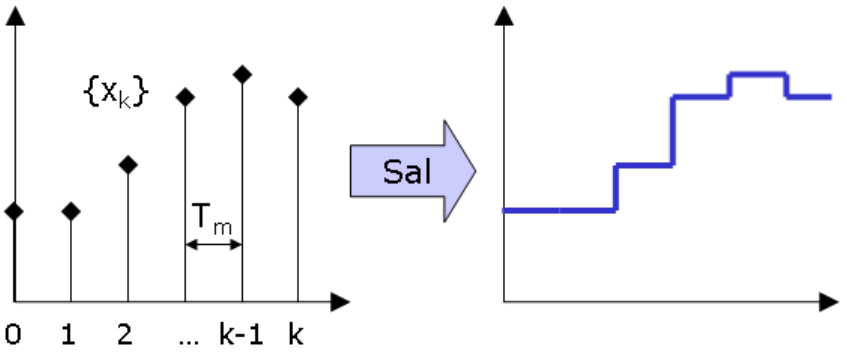
Funcionamiento hasta $t \rightarrow \infty$
No se pueden guardar ∞ valores de x_k

Computadores para control

- Entradas del computador: convierten valores continuos y analógicos (exterior) en valores discretos y digitales (computador):
  - Entradas digitales (todo/nada).
  - Conversores A/D
  - Contadores de pulsos
  - Entradas numéricas

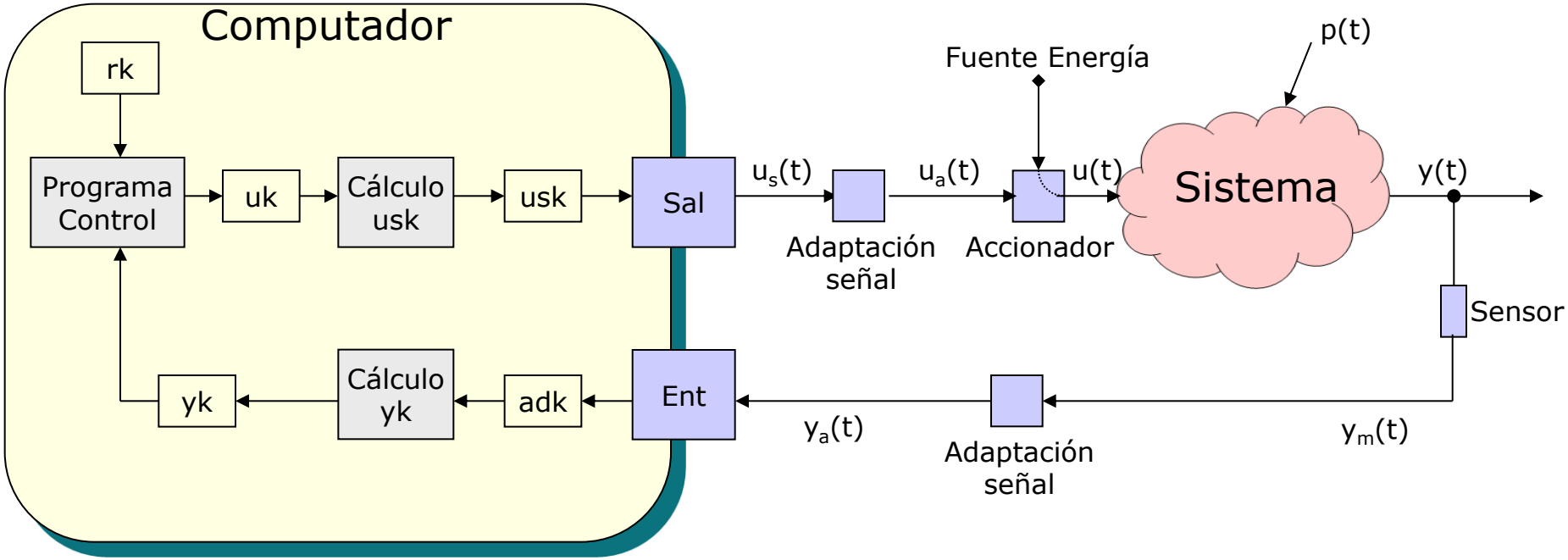


- Salidas del computador: convierten valores discretos y digitales (computador) en señales continuas y analógicas (exterior):
  - Salidas digitales (todo/nada)
  - Conversores D/A
  - Accionadores PWM
  - Salidas numéricas



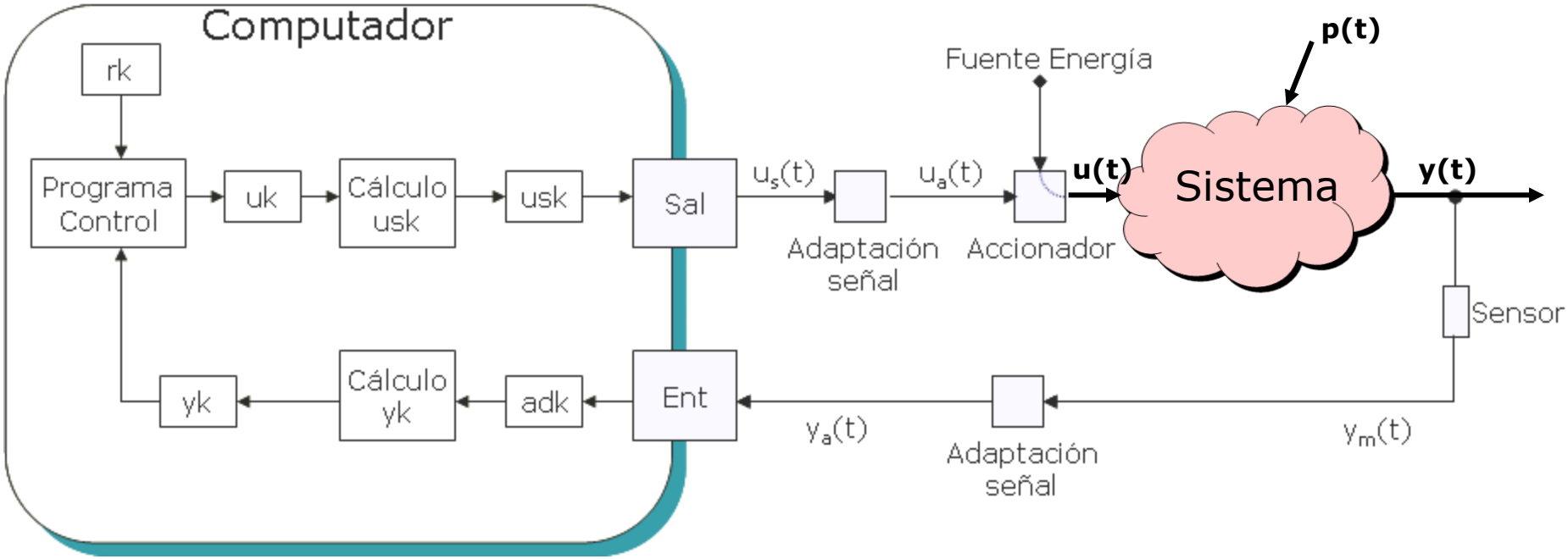
El lazo de control

- Esquema típico del lazo de control por computador (1 ent, 1 sal):



El lazo de control

- Esquema típico del lazo de control por computador (1 ent, 1 sal):

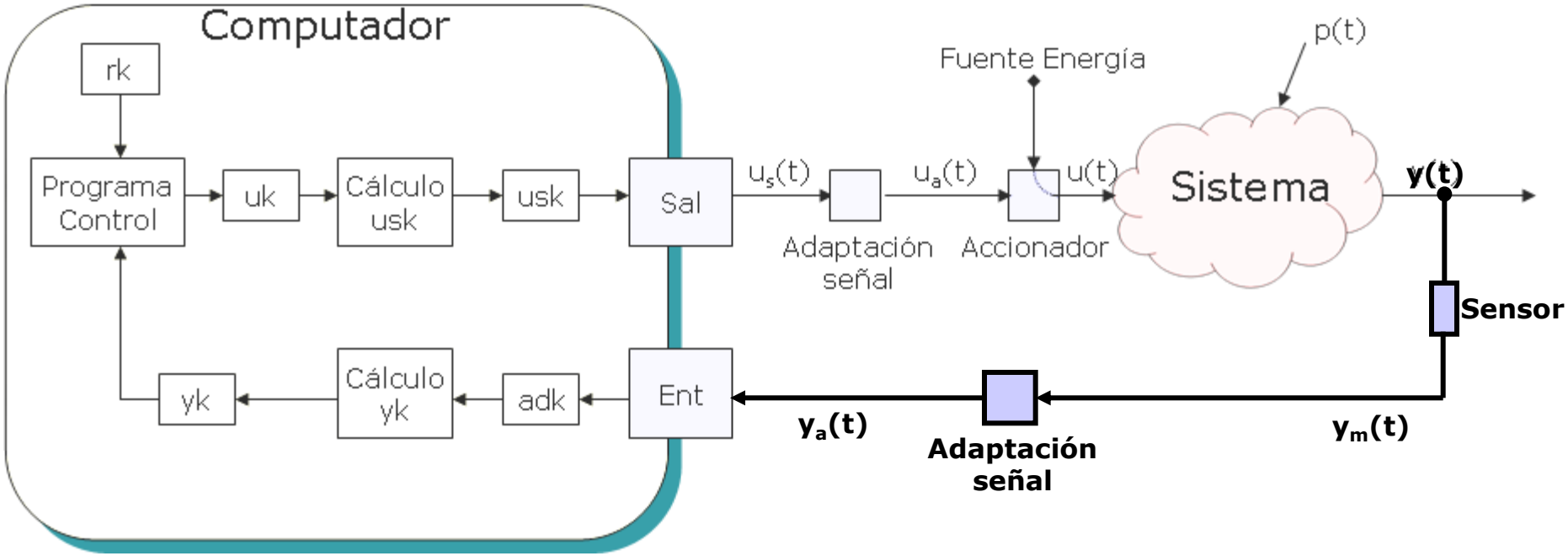


El sistema reacciona ante cambios en su entrada, modificando el valor de su salida.

y(t) = Salida a controlar del sistema
u(t) = Acción mediante la que se gobierna la salida
La relación y(t)/u(t) no es constante, sino dinámica. Se expresa mediante la T. Laplace G(s)=Y(s)/U(s)
p(t) = Perturbaciones que modifican la relación u(t)→y(t)

El lazo de control

- Esquema típico del lazo de control por computador (1 ent, 1 sal):

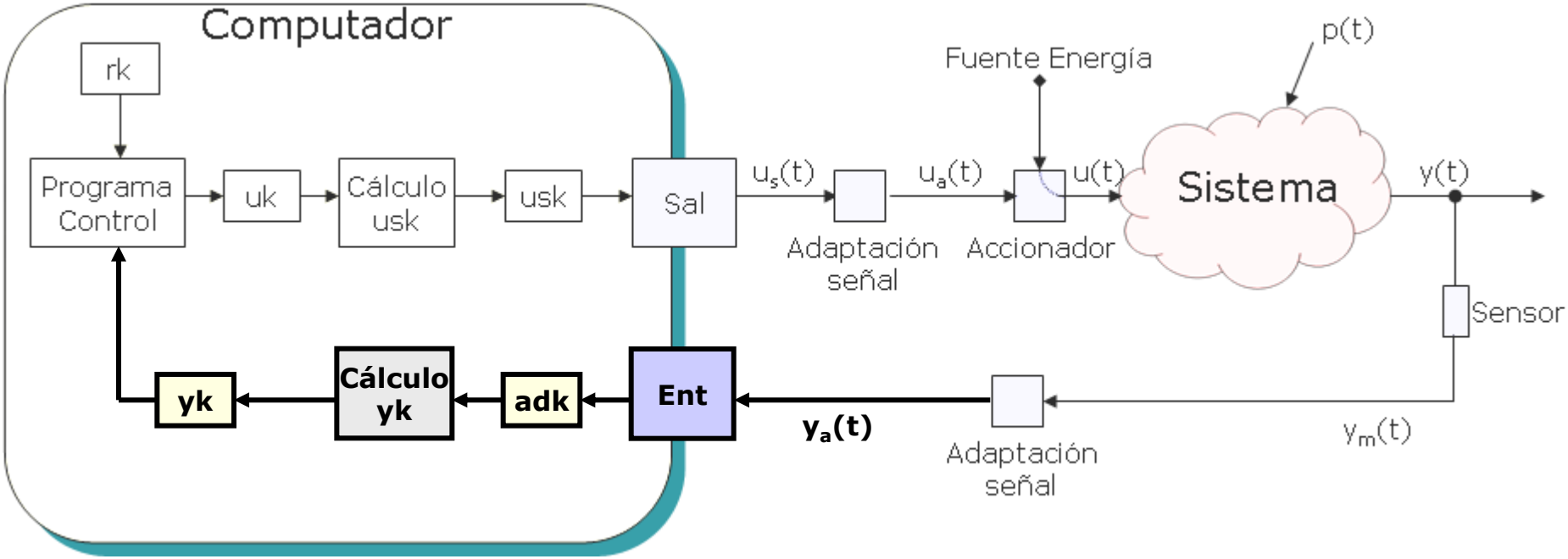


Mediante un sensor se mide la salida del sistema. La señal del sensor debe ser adaptada a los niveles aceptados por el computador

y(t) = Salida a controlar del sistema
$y_m(t)$ = Valor medido mediante un sensor (pequeña tensión o corriente)
$y_a(t)$ = Valor adaptado a la tensión o corriente requerida por la entrada del computador

El lazo de control

- Esquema típico del lazo de control por computador (1 ent, 1 sal):



La entrada del computador digitaliza el valor analógico, obteniendo un valor digital equivalente. Con ese valor, se puede calcular la salida del sistema.

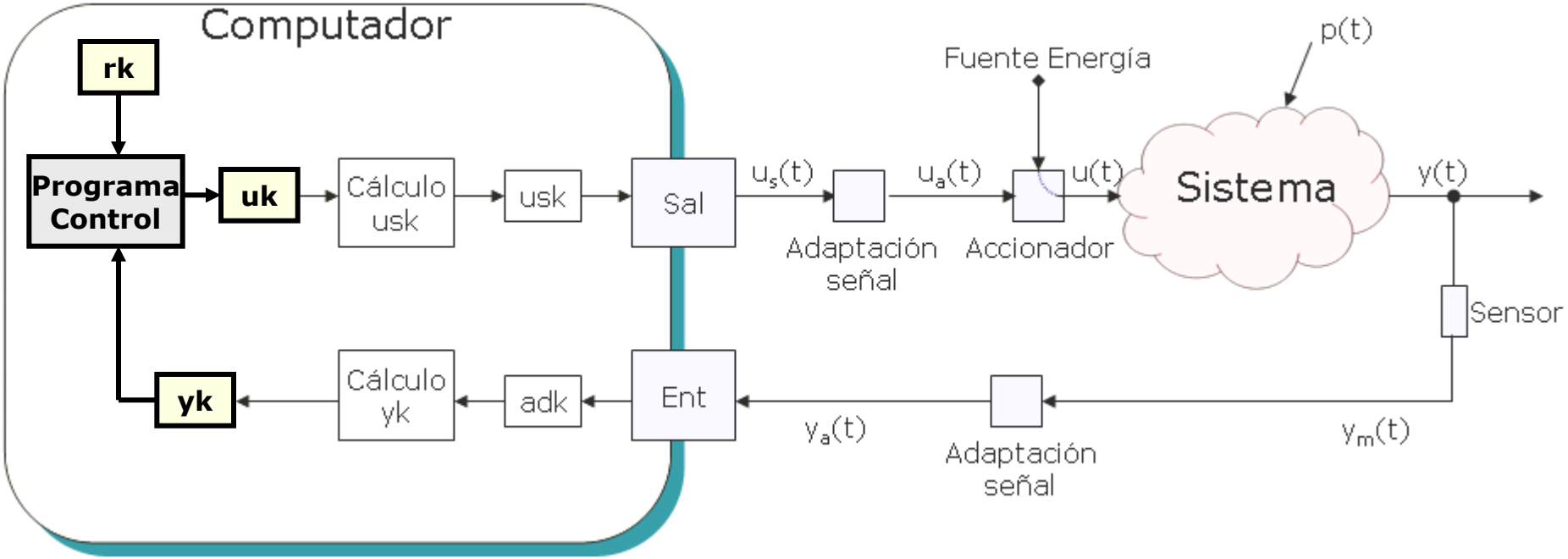
$y_a(t)$ = Valor adaptado a la tensión o corriente requerida por la entrada del computador

adk = Valor digitalizado de la entrada del computador

yk = Valor digitalizado de y(t), utilizable para cálculos de control

El lazo de control

- Esquema típico del lazo de control por computador (1 ent, 1 sal):

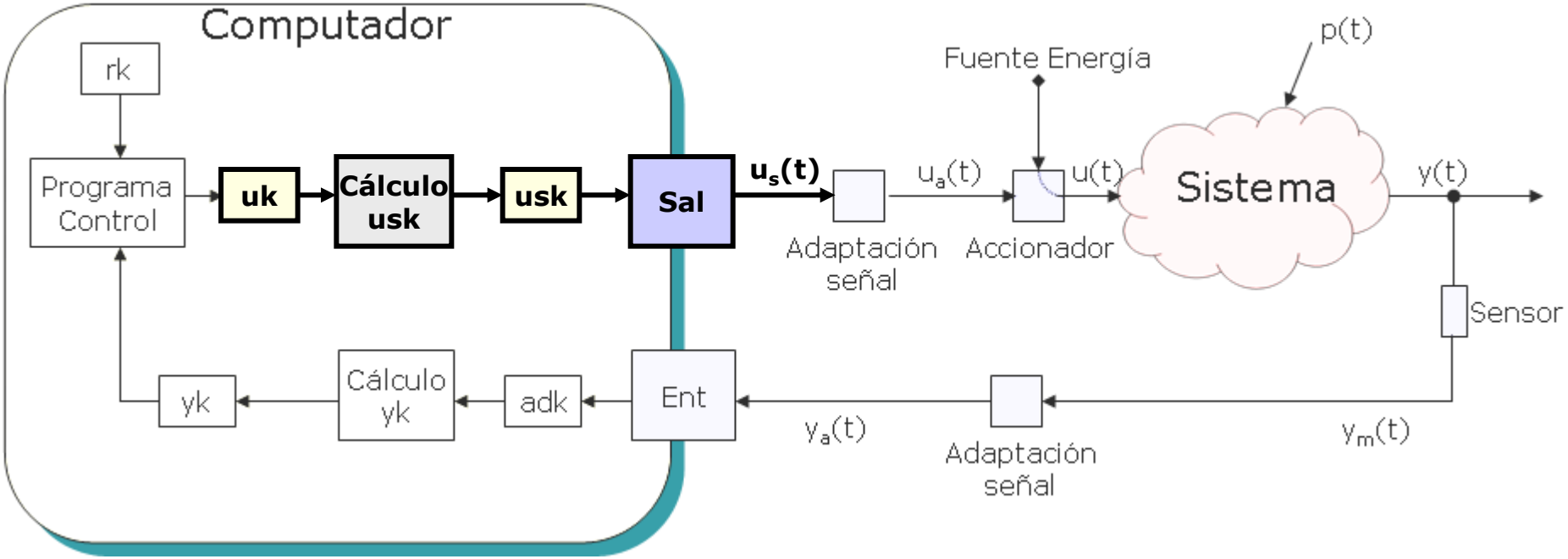


El programa de control compara la salida del sistema con una referencia (valor deseado), calculando una acción de control que corrija el error observado.

yk = Valor digitalizado de y(t)
rk = Valor digitalizado de la consigna (valor deseado para y(t))
uk = Valor calculado de la acción de control (valor deseado para u(t))

El lazo de control

- Esquema típico del lazo de control por computador (1 ent, 1 sal):



Se debe calcular el valor digital a colocar en la salida del computador para que, tras ser amplificada, se convierta en la entrada u(t) deseada.

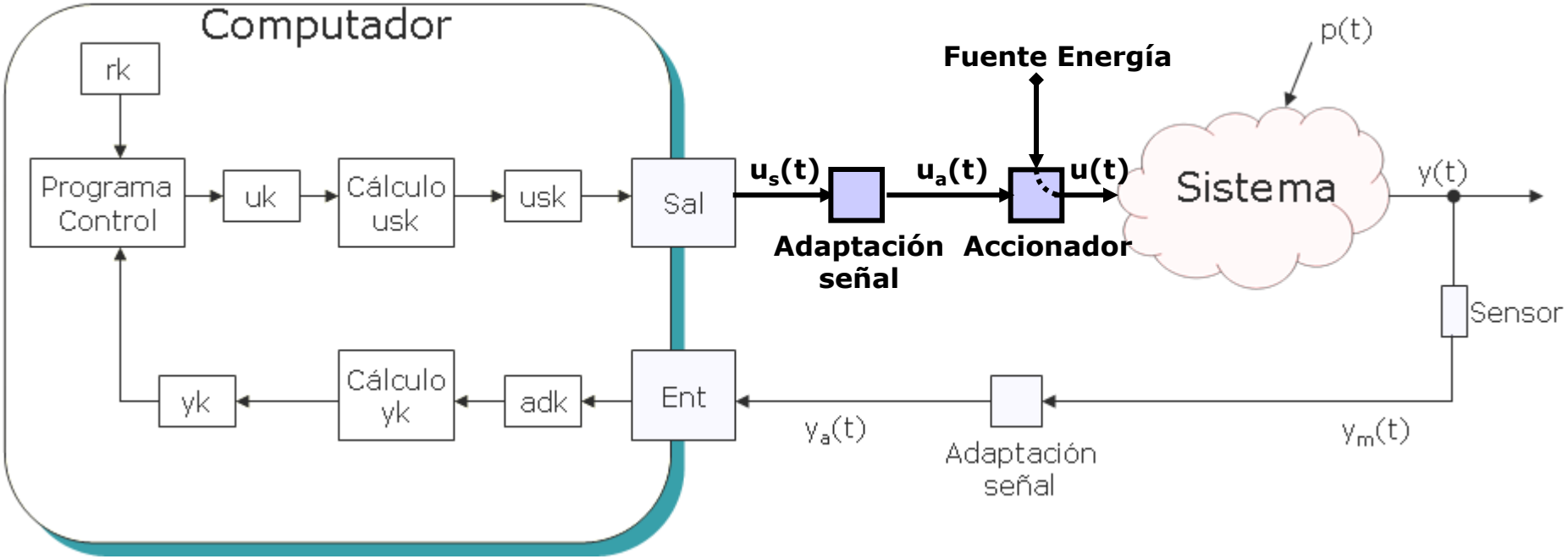
uk = Valor calculado de la acción de control (valor deseado para u(t))

usk = Valor digitalizado de la salida necesaria del computador

$u_s(t)$ = Valor analógico a la salida del computador

El lazo de control

- Esquema típico del lazo de control por computador (1 ent, 1 sal):



La salida del computador es adaptada a los niveles de entrada del accionador, que deja pasar la cantidad de Energía indicada de una fuente externa hacia el sistema.

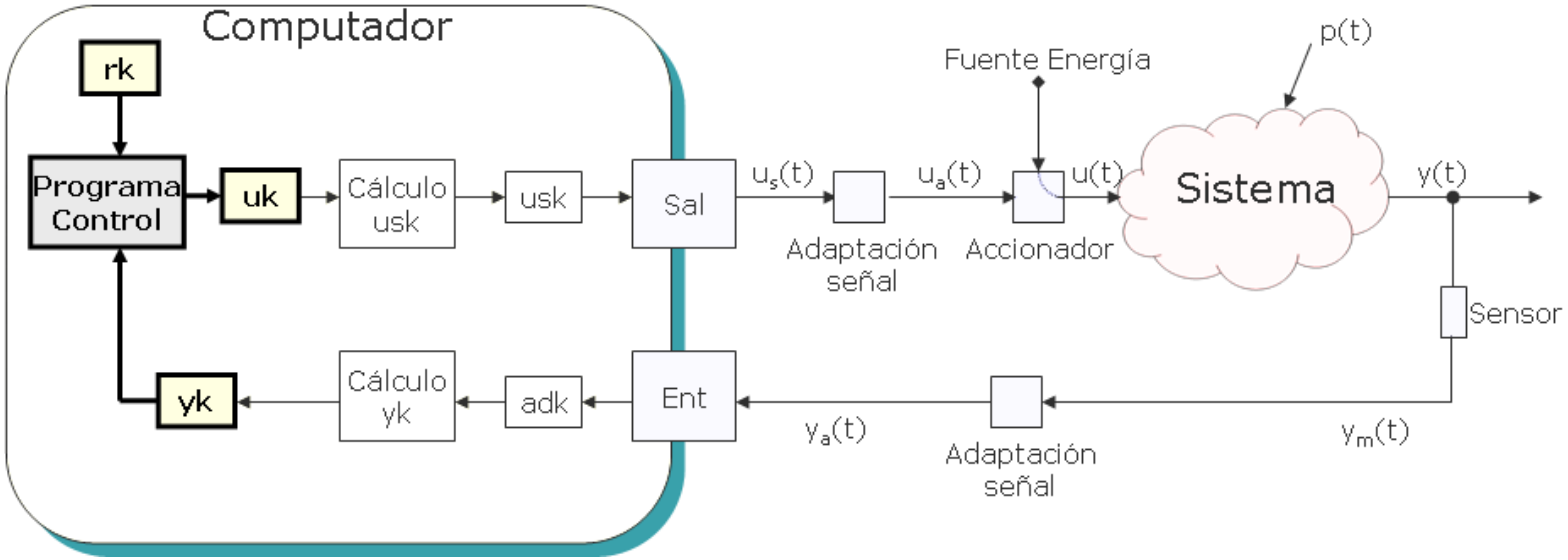
$u_s(t)$ = Valor analógico a la salida del computador
$u_a(t)$ = Señal adaptada para la entrada del accionador
u(t) = Acción que se aplica al sistema (igual a la uk calculada por el programa de control)

Indice

- Introducción control de procesos por computador
- **Las matemáticas del control**
- Programación del lazo de control
- Programación en lenguaje C
- Implantación del control en el computador
- El control secuencial

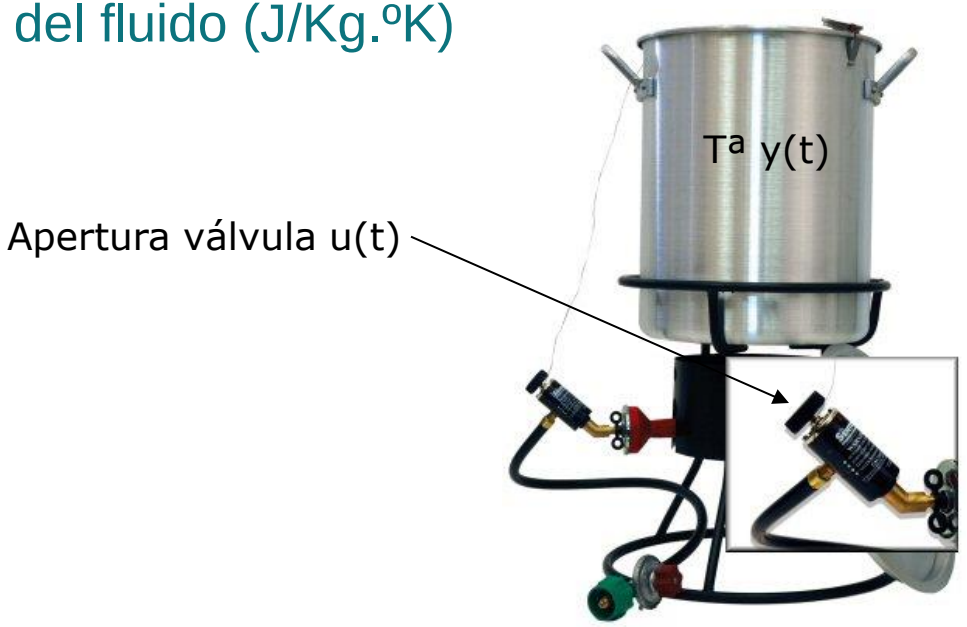
Las matemáticas del control

- El cálculo óptimo de uk=Fn(yk,rk) es crucial para satisfacer las necesidades del control:
  - **Precisión**: la salida debe seguir fielmente a la referencia.
  - **Velocidad**: la salida debe seguir rápidamente las variaciones de la referencia.
  - **Robustez**: las perturbaciones, ruidos, o variaciones del sistema no deben modificar el comportamiento.



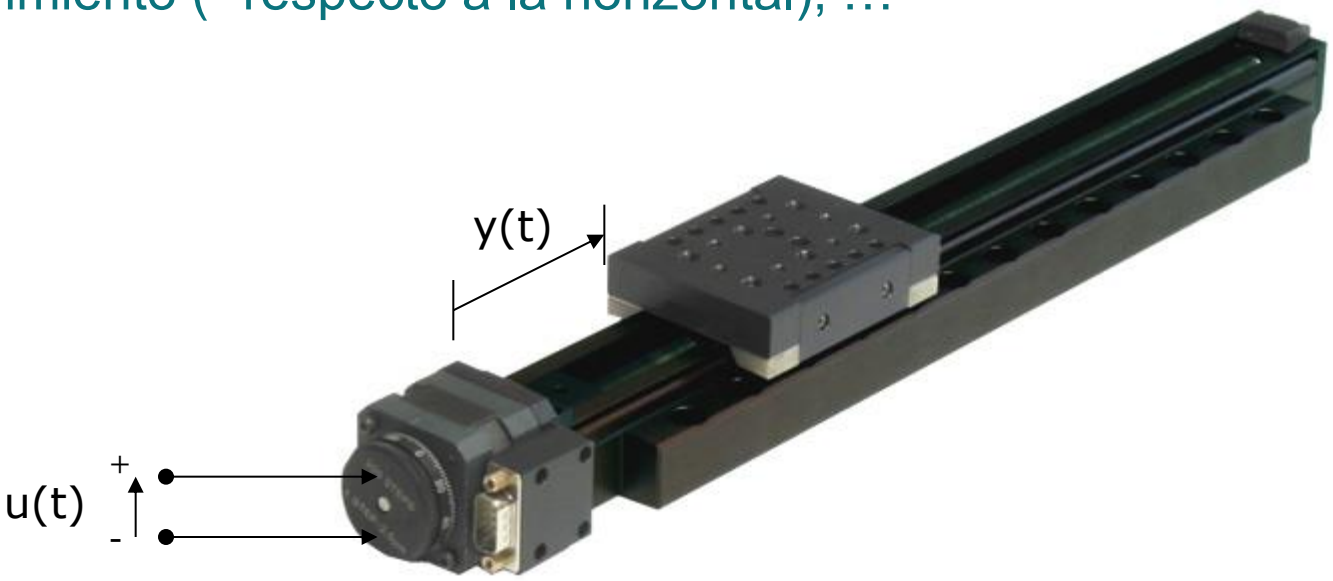
Las matemáticas del control

- Ejemplo 1: control de temperatura de un fluido mediante una llama de gas.
  - **Referencia**: temperatura deseada del fluido (ºC)
  - **Salida del sistema**: temperatura medida del fluido (ºC)
  - **Acción de control**: apertura de la válvula de gas (0%-100%)
  - **Perturbaciones:** temperatura exterior (ºC), cantidad de fluido (Kg), calor específico del fluido (J/Kg.ºK)



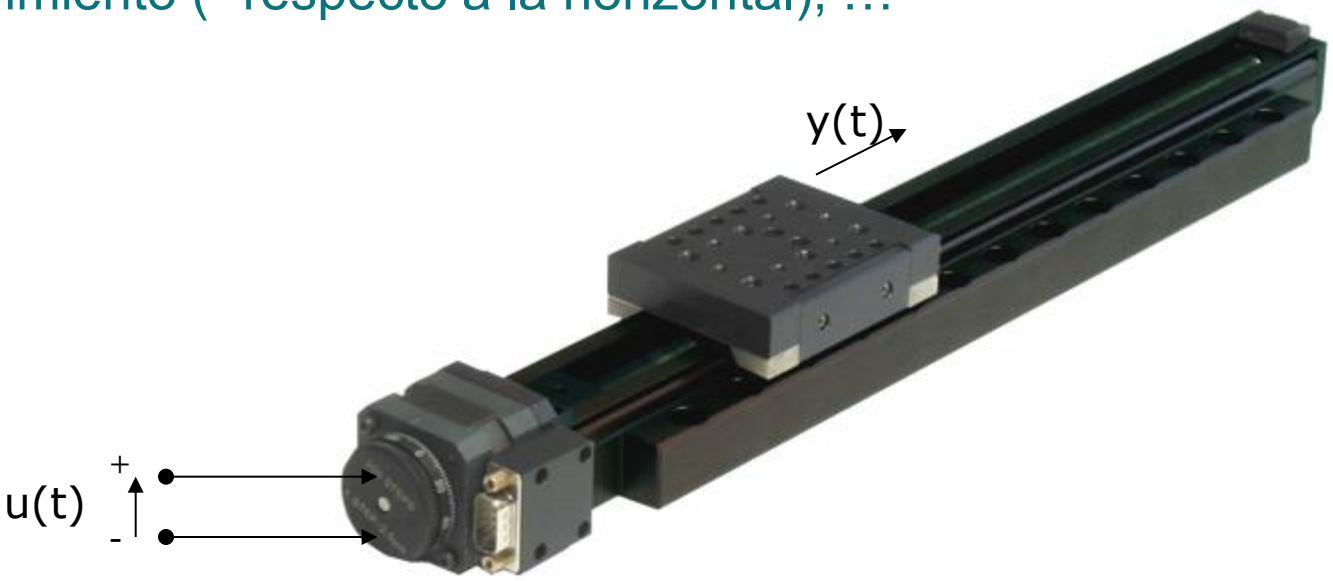
Las matemáticas del control

- Ejemplo 2: control de posición de un móvil mediante un motor DC.
  - **Referencia**: posición deseada del móvil (mm).
  - **Salida del sistema**: posición medida del móvil (mm).
  - **Acción de control**: tensión aplicada al motor (V).
  - **Perturbaciones:** peso del móvil (Kg), dirección del movimiento (º respecto a la horizontal), …



Las matemáticas del control

- Ejemplo 3: control de velocidad de un móvil mediante un motor DC.
  - **Referencia**: velocidad deseada del móvil (mm/s).
  - **Salida del sistema**: velocidad medida del móvil (mm/s).
  - **Acción de control**: tensión aplicada al motor (V).
  - **Perturbaciones:** peso del móvil (Kg), dirección del movimiento (º respecto a la horizontal), …



Las matemáticas del control

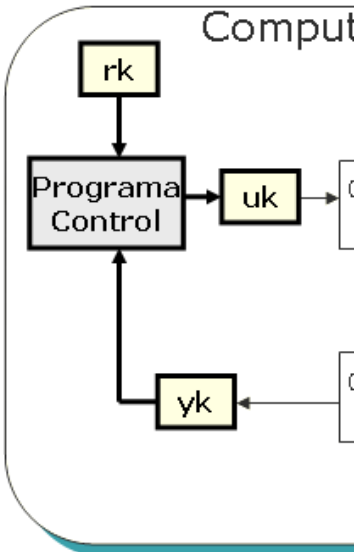
- Un cálculo sencillo: control todo/nada

  Si yk < rk → uk = cte1

  Si yk > rk → uk = cte2

  Si yk = rk → no modificar uk

  - Se acciona el sistema con un valor constante, que depende del signo del error.
  - Cálculo excesivamente simple: la acción de control no depende del valor absoluto del error.



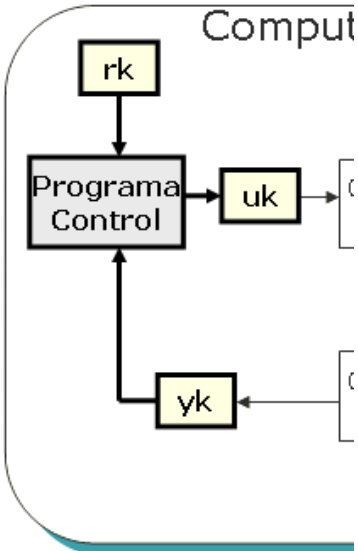
Ejemplo 1:
- Se abre la válvula de gas (100%) si la Tª es menor que la referencia.
- Se cierra la válvula (0%) si la Tª es mayor o igual que la referencia.
- La apertura es la misma para error=0.1ºC ó error=100ºC
- Excesivas conmutaciones cuando yk ≈ rk (se suele usar histéresis)

Las matemáticas del control

- Un cálculo sencillo: control proporcional

```
ek = rk - yk
uk = cte . ek
```

  - Cuando el error es 0, la salida es 0.
  - La salida es mayor (en valor absoluto) cuanto más grande sea el error (en valor absoluto), y tiene el mismo signo que el error.
  - El cálculo es muy simple: no tiene en cuenta la evolución que llevaba el sistema anteriormente.



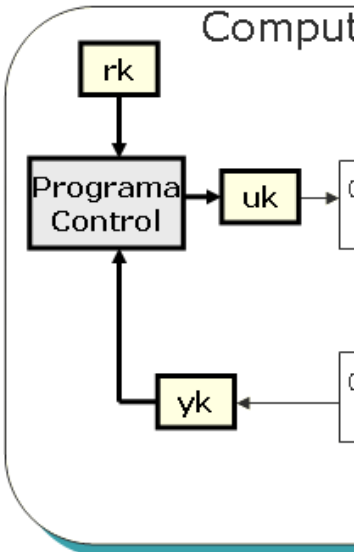
Ejemplo 1:
- Se abre más la válvula de gas cuanto mayor sea el error.
- Se cierra la válvula (0%) si la Tª iguala a la referencia.
- No se tiene en cuenta si la Tª estaba subiendo o bajando previamente.

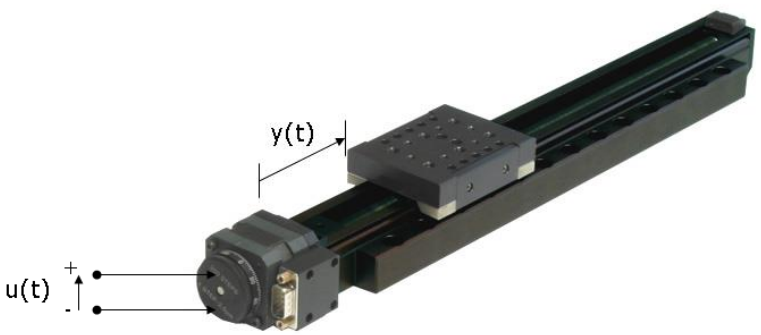
Las matemáticas del control

- Un cálculo sencillo: control proporcional

```
ek = rk - yk
uk = cte . ek
```

  - Cuando el error es 0, la salida es 0.
  - La salida es mayor (en valor absoluto) cuanto más grande sea el error (en valor absoluto), y tiene el mismo signo que el error.
  - El cálculo es muy simple: no tiene en cuenta la evolución que llevaba el sistema anteriormente.





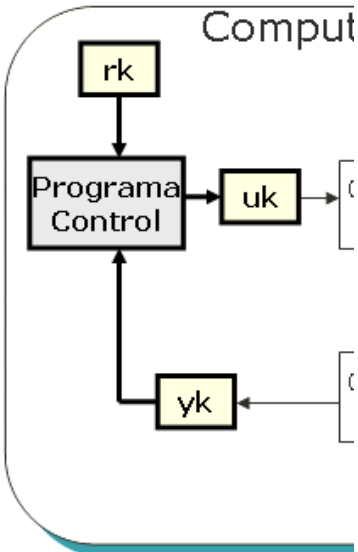
Ejemplo 2:
- Se da más tensión al motor cuanto mayor sea la distancia al destino.
- Se para el motor (tensión 0) cuando alcanza el destino (error 0).
- No se tiene en cuenta si el móvil se está acercando o alejando del destino.

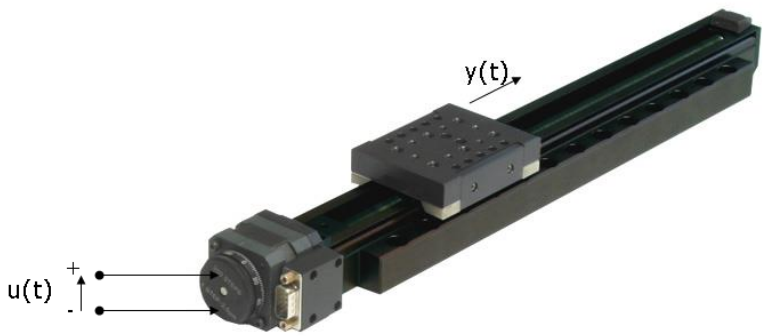
Las matemáticas del control

- Un cálculo sencillo: control proporcional

```
ek = rk - yk
uk = cte . ek
```

  - Cuando el error es 0, la salida es 0.
  - La salida es mayor (en valor absoluto) cuanto más grande sea el error (en valor absoluto), y tiene el mismo signo que el error.
  - En ciertos sistemas, no se puede llegar a error 0.





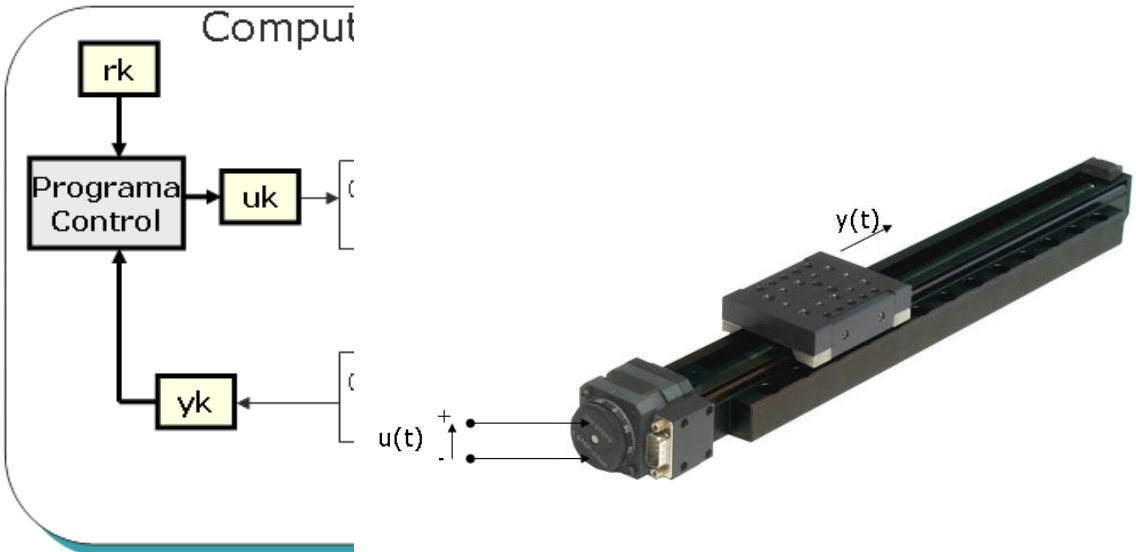
Ejemplo 3:
- Se aumenta la tensión del motor si la velocidad es menor que la deseada
- Si la velocidad fuese igual a la deseada, la tensión aplicada sería 0V, por tanto el motor tendería a pararse.
- Es necesario mantener un error para que el sistema se estabilice.

Las matemáticas del control

- Un cálculo sencillo: control proporcional-integral

```
ek = rk - yk
uk = uk_anterior + cte . ek
```

  - Cuando el error es 0, la salida se mantiene.
  - La salida se incrementa (en valor absoluto) tanto más cuanto más grande sea el error (en valor absoluto), y el incremento tiene el mismo signo que el error.
  - El cálculo necesita recordar un valor anterior de uk.



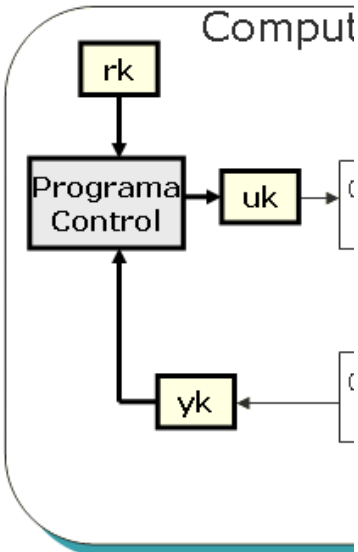
Ejemplo 3:
- Se aumenta la tensión del motor si la velocidad es menor que la deseada
- Si la velocidad es igual a la deseada, se mantiene la tensión aplicada.
- El error llega a 0.

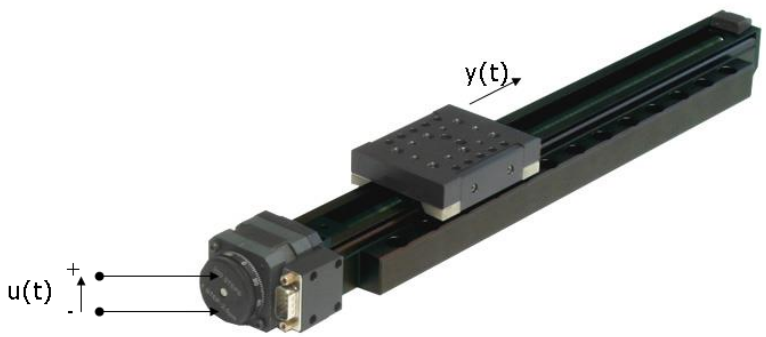
Las matemáticas del control

- Un cálculo sencillo: control proporc.-integral-diferencial

```
ek = rk - yk
uk = uk_anterior + cte1 . ek - cte2 . (ek-ek_anterior)
```

  - Cuando el error se mantiene en 0, la salida se mantiene.
  - La salida se incrementa tanto más cuanto mayor sea el error, y cuanto el error vaya más en sentido contrario al deseado.
  - El cálculo necesita recordar un valor anterior de uk y ek.





Ejemplo 3:
- Se aumenta la tensión del motor si la velocidad es menor que la deseada
- Si la velocidad es igual a la deseada, se mantiene la tensión aplicada.
- El error llega a 0 más rápidamente, al tenerse en cuenta su derivada.

Las matemáticas del control

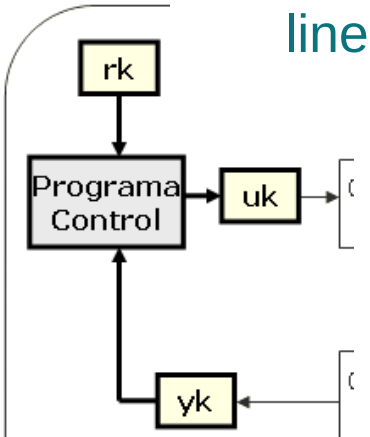
- Cálculo general:

$e_k = r_k - y_k$

$u_k = Fn(e_k, e_{k-1}, e_{k-2}, \ldots, e_{k-m}, u_{k-1}, u_{k-2}, \ldots, u_{k-n})$

  - La función realiza un cálculo con los valores actual y anteriores (hasta retraso m) del error, y los valores anteriores (hasta retraso n) de la acción de control calculada.
  - Se pueden diseñar múltiples funciones, pero sería adecuado disponer de una teoría unificada para calcularlas.
  - La teoría unificada es más sencilla si se usan funciones lineales, ya que se puede aplicar el principio de superposición:

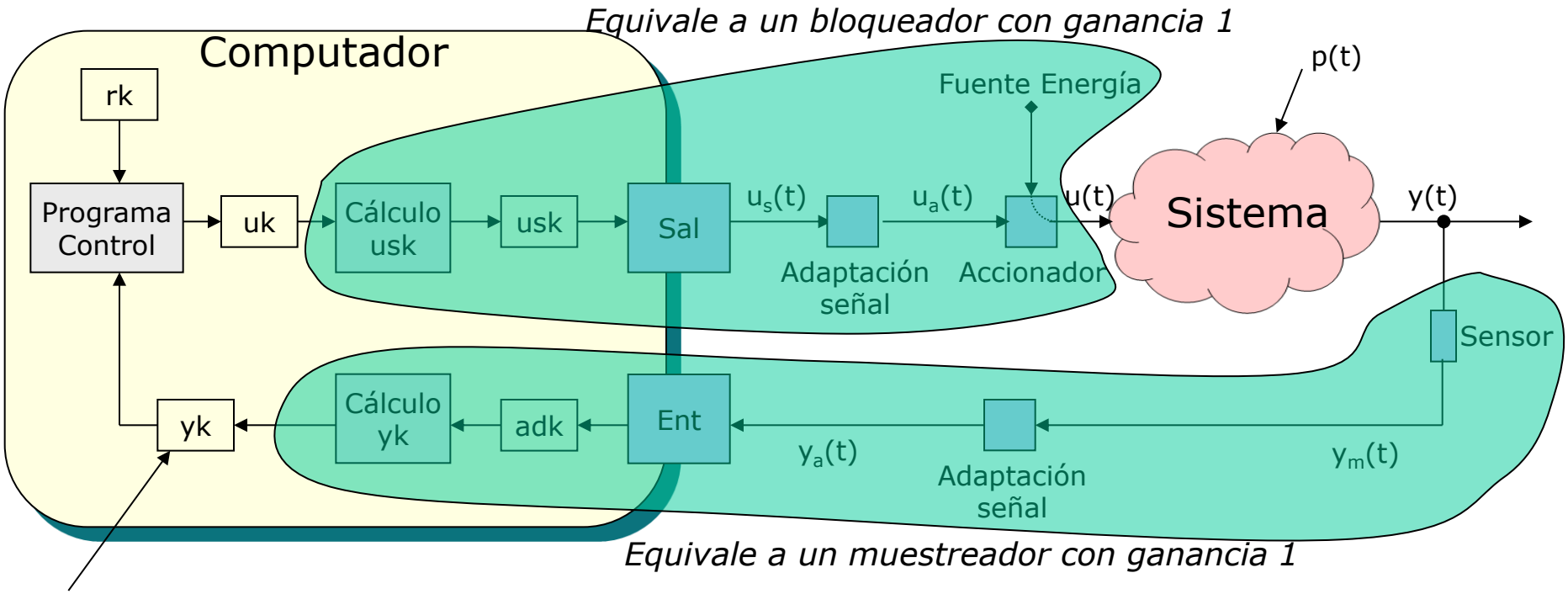
$$u_k = b_0 \cdot e_k + b_1 \cdot e_{k-1} + \ldots + b_m \cdot e_{k-m} - a_1 \cdot u_{k-1} - a_2 \cdot u_{k-2} - \ldots - a_n \cdot u_{k-n}$$



- Control proporcional:
  - » m=0, n=0, b_0=cte
- Control proporcional-integral:
  - » m=0, n=1, b_0=cte, a_1=-1
- Control proporcional-integral-diferencial:
  - » m=1, n=1, b_0=cte1-cte2, b_1=cte2, a_1=-1

Las matemáticas del control

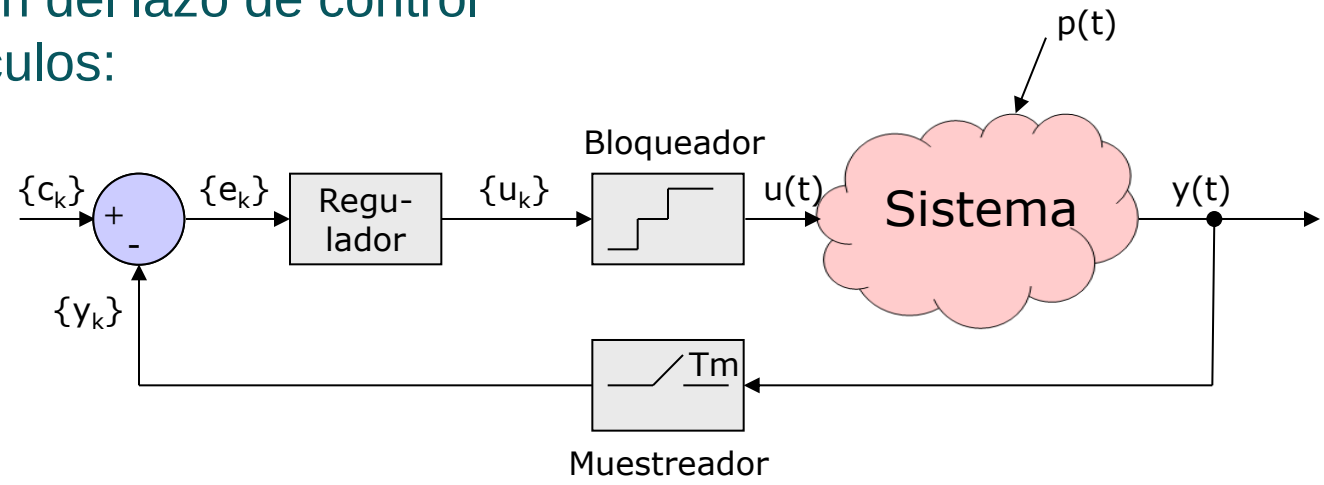
- Simplificación del lazo de control para los cálculos:



La representación matemática de los valores de las variables discretas a lo largo del tiempo es una secuencia: $\{y_k\}$

Las matemáticas del control

- Simplificación del lazo de control para los cálculos:



- La teoría más sencilla se obtiene:
  - Calculando la Transformada de Laplace de las señales continuas

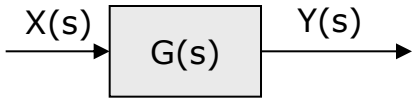
$$X(s) = \mathscr{L}[x(t)] = \int_{\tau=0}^{t} x(\tau).e^{-s\tau}.d\tau$$

  - Calculando la Transformada en Z de las señales discretas

$$X(z) = \mathscr{Z}[\{x_k\}] = \sum_{i=0}^{k} x_{k-i}.z^{-i}$$

Las matemáticas del control

- Transformada de Laplace, interpretación básica:



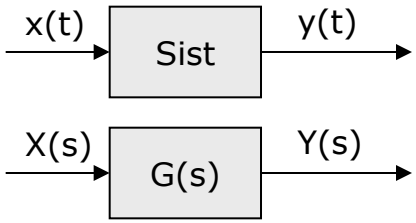
  - G(s)=Y(s)/X(s) para sistemas lineales dinámicos continuos.
  - G(s) es un cociente de polinomios, grado den ≥ grado num

$$G(s) = \frac{Y(s)}{X(s)} = \frac{b_o.s^m + b_1.s^{m-1} + ... + b_{m-1}.s + b_m}{s^n + a_1.s^{n-1} + ... + a_{n-1}.s + a_n} \quad , n >= m$$

  - Las raíces de los polinomios denominador (polos) y numerador (ceros) definen el comportamiento básico del sistema.
  - Sistema estable: todos los polos con parte real negativa. En este caso:
    - Régimen permanente: G(0)=ganancia estática (con unidades).
    - Respuesta dinámica (transitorio) dependiente de la posición de polos y ceros.

Las matemáticas del control

- Transformada de Laplace, relación entre transformada y señales temporales:



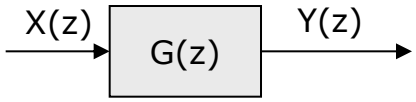
$$G(s) = \frac{Y(s)}{X(s)} = \frac{b_o.s^m + b_1.s^{m-1} + ... + b_{m-1}.s + b_m}{s^n + a_1.s^{n-1} + ... + a_{n-1}.s + a_n} \quad , n >= m$$

$$\overset{...(n}{y}(t) = b_o.\ \overset{...(m}{x}(t) + b_1.\ \overset{...(m-1}{x}(t) + ... + b_{m-1}.\dot{x}(t) + b_m.x(t) - \left(a_1.\overset{...(n-1}{y}(t) + ... + a_{n-1}.\dot{y}(t) + a_n.y(t) \right)$$

- Ecuación diferencial del sistema

Las matemáticas del control

- Transformada en Z, interpretación básica:



  - G(z)=Y(z)/X(z) para sistemas lineales dinámicos discretos.
  - G(z) es un cociente de polinomios, grado den ≥ grado num.

$$G(z)=\frac{Y(z)}{X(z)}=\frac{b_o.z^m+b_1.z^{m-1}+...+b_{m-1}.z+b_m}{z^n+a_1.z^{n-1}+...+a_{n-1}.z+a_n}\ ,\ n>=m$$

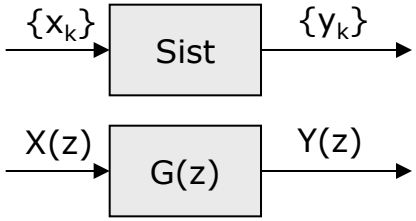
  - Es habitual presentar G(z) dividiendo entre z^n:

$$G(z)=\frac{Y(z)}{X(z)}=\frac{b_o+b_1.z^{-1}+...+b_{m-1}.z^{-(m-1)}+b_m.z^{-m}}{1+a_1.z^{-1}+...+a_{n-1}.z^{-(n-1)}+a_n.z^{-n}}$$

  - Las raíces de los polinomios denominador (polos) y numerador (ceros) definen el comportamiento básico del sistema.
  - Sistema estable: todos los polos con módulo < 1. En este caso:
    - Régimen permanente: G(1)=ganancia estática (con unidades).
    - Respuesta dinámica (transitorio) según posición de polos y ceros.

Las matemáticas del control

- Transformada de en Z, relación entre transformada y señales temporales:



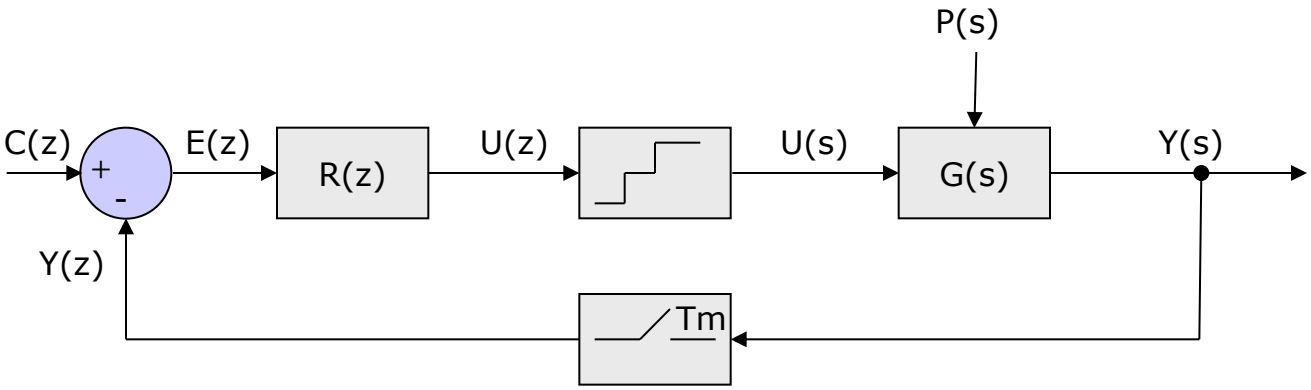
$$G(z) = \frac{Y(z)}{X(z)} = \frac{b_o + b_1.z^{-1} + \ldots + b_{m-1}.z^{-(m-1)} + b_m.z^{-m}}{1 + a_1.z^{-1} + \ldots + a_{n-1}.z^{-(n-1)} + a_n.z^{-n}}$$

$$y_k = b_0.x_k + b_1.x_{k-1} + \ldots + b_{m-1}.x_{k-(m-1)} + b_m.x_{k-m} - (a_1.y_{k-1} + \ldots + a_{n-1}.y_{k-(n-1)} + a_n.y_{k-n})$$

- Ecuación en diferencias del sistema

Las matemáticas del control

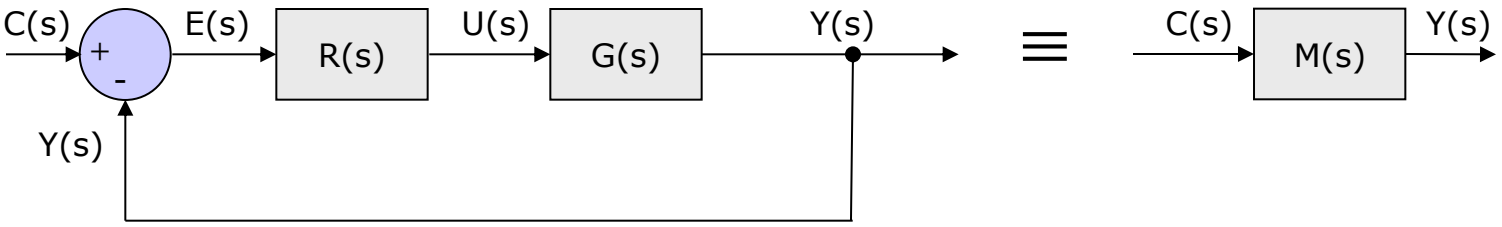
- El lazo de control en forma matemática:



- En el lado continuo:
  - G(s)=Y(s)/U(s) G(s) : función de transferencia del sistema continuo
- En la parte discreta:
  - E(z)=C(z)-Y(z)
  - R(z)=U(z)/E(z) R(z) : función de transferencia del sistema discreto
- Lo más sencillo es convertir todo a continuo o discreto para los cálculos. Existen fórmulas para esta conversión.

Las matemáticas del control

- El lazo de control en forma continua:



- A partir de la Tª de sistemas:

$$\frac{Y(s)}{C(s)} = M(s) = \frac{R(s).G(s)}{1 + R(s).G(s)}$$

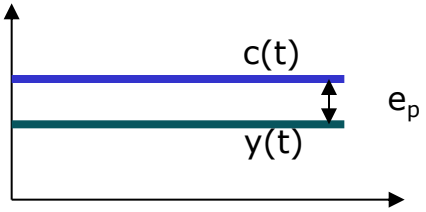
- El óptimo sería: M(s) = 1
- El óptimo no es posible porque los sistemas tienen comportamiento dinámico.

Las matemáticas del control

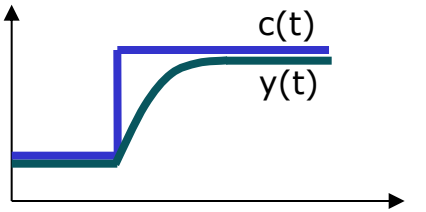
- Optimización del regulador R(s):

$$\frac{Y(s)}{C(s)} = M(s) = \frac{R(s).G(s)}{1 + R(s).G(s)}$$

  - **Régimen permanente**: ante consigna estable, la salida debe igualarse a la consigna.
    - Óptimo: Ganancia[M(s)] $\rightarrow$ 1



  - **Régimen transitorio**: la salida debe seguir variaciones en la consigna lo más rápido posible.
    - Óptimo: Retardo[M(s)] $\rightarrow$ 0



Las matemáticas del control

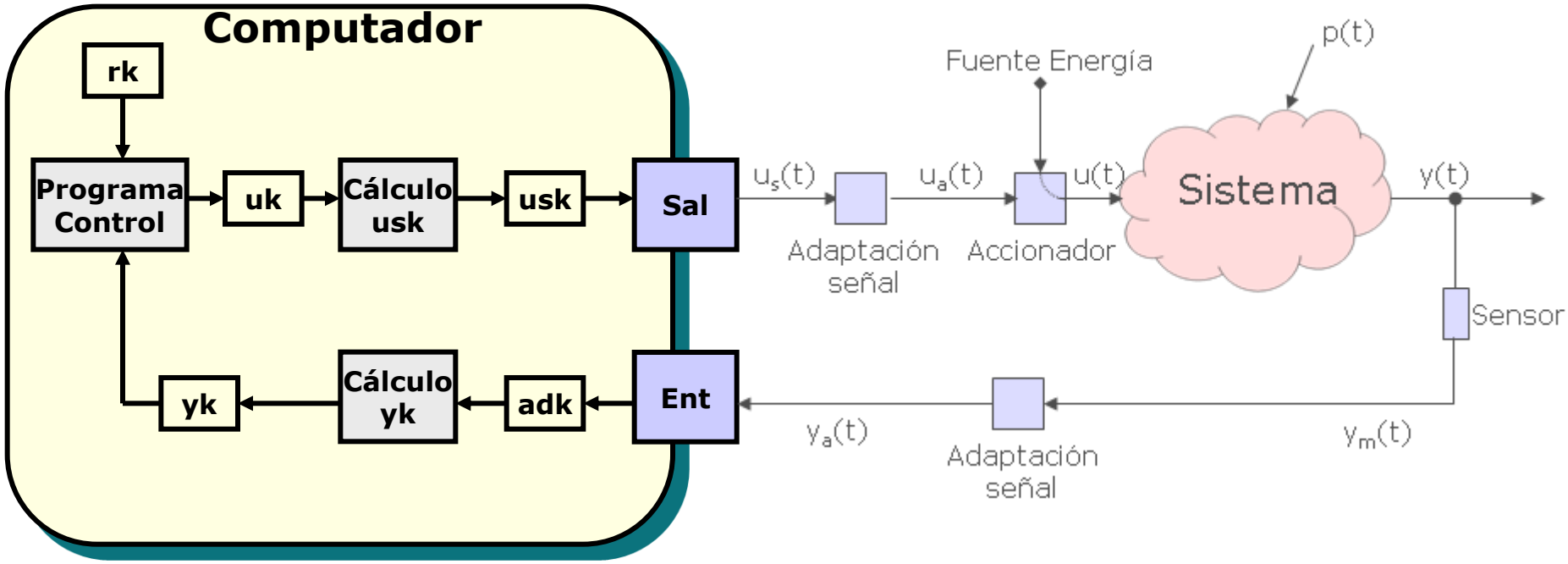
- A partir de aquí (materia de las asignaturas de Regulación Automática):
  - Tª de errores en régimen permanente
  - Lugar de las raíces para régimen transitorio
  - Robustez ante perturbaciones
  - Tª de control frecuencial
  - Discretización de reguladores
  - Aliasing y filtros analógicos

Indice

- Introducción control de procesos por computador
- Las matemáticas del control
- **Programación del lazo de control**
- Programación en lenguaje C
- Implantación del control en el computador
- El control secuencial

Programación del lazo de control

- Programación del lazo de control:



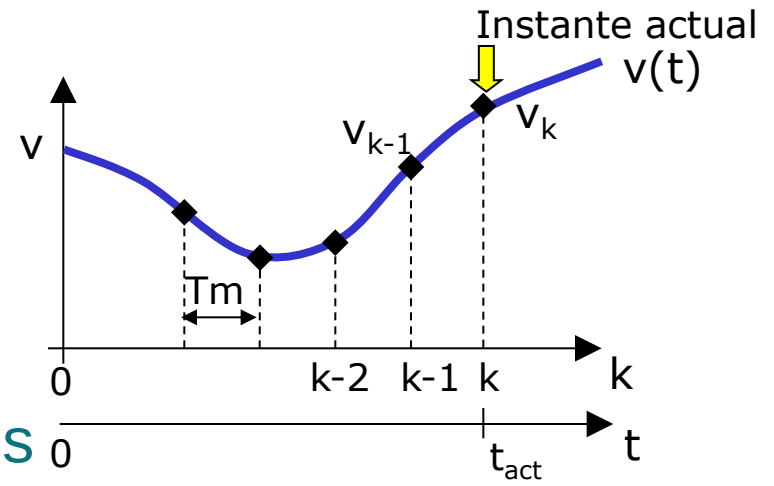
Programación del lazo de control

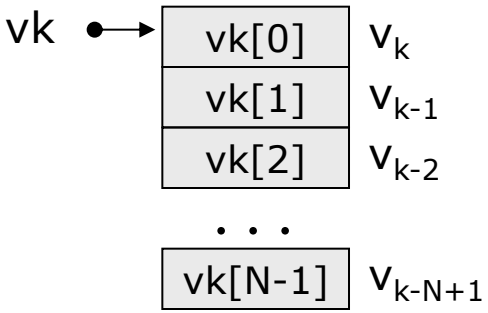
- Programación del lazo de control:
  - Cada Tm (periodo de muestreo) se debe repetir el mismo algoritmo.
  - El algoritmo debe poder ejecutarse hasta un tiempo indefinido.
- Ejecución de un paso del lazo de control:
  - Adquisición de datos
  - Cálculo de valores adquiridos en sus unidades
  - Actualización de tablas temporales
  - Cálculo de la acción de control (regulador)
  - Cálculo de los valores de salida
  - Escritura de los dispositivos de salida
  - Espera al siguiente Tm

Programación del lazo de control

- Variables para almacenar secuencias temporales:
  - No es posible almacenar todos los valores desde el inicio del tiempo.
  - Son necesarios el valor actual y algunos retrasados (nº finito) para los cálculos.
  - Solución. Tabla de los N valores más recientes:

```
float vk[N];
// En cada instante:
// vk[i] ≡ v_{k-i}
```

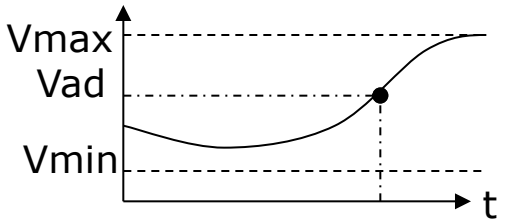


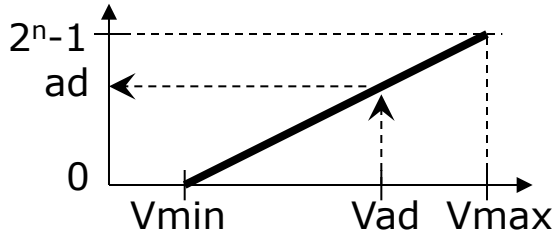


Programación del lazo de control

- Programación del lazo de control:
  - **Adquisición de datos**
  - Cálculo de valores adquiridos en sus unidades
  - Actualización de tablas temporales
  - Cálculo de la acción de control (regulador)
  - Cálculo de los valores de salida
  - Escritura de los dispositivos de salida
  - Espera al siguiente Tm

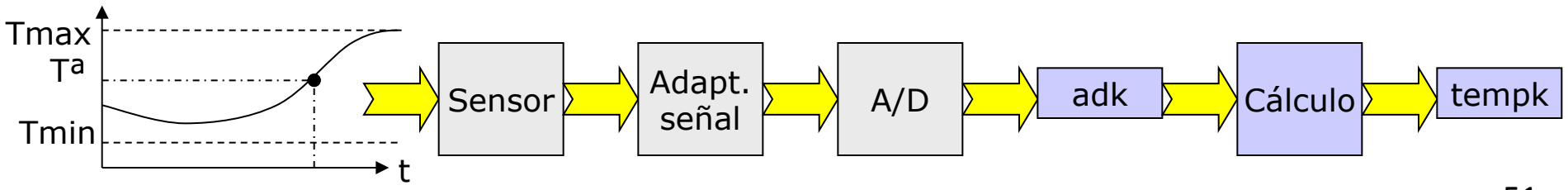
- Adquisición de datos:
  - Las salidas del sistema se adquieren mediante dispositivos de entrada:
    - Conversores A/D, entradas digitales, entradas de pulso, entradas numéricas.
  - Los dispositivos de entrada proporcionan un valor (normalmente entero) relacionado con el valor de la variable medida.
- Ejemplo. Conversor A/D:





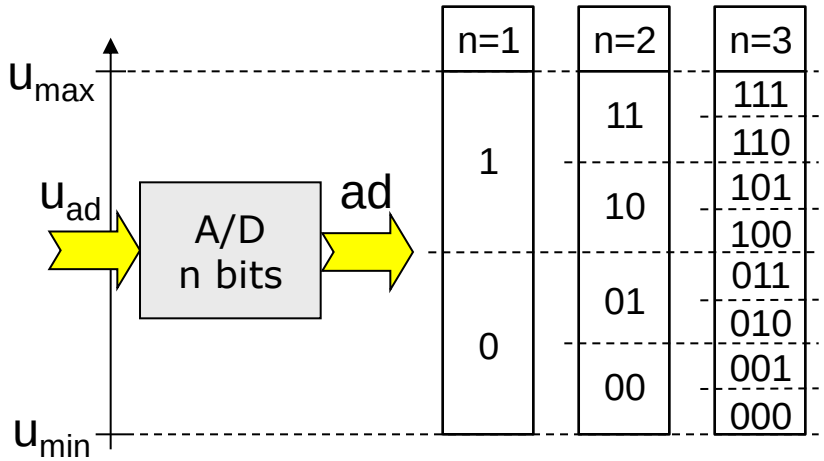
Programación del lazo de control

- Programación del lazo de control:
  - Adquisición de datos
  - **Cálculo de valores adquiridos en sus unidades**
  - Actualización de tablas temporales
  - Cálculo de la acción de control (regulador)
  - Cálculo de los valores de salida
  - Escritura de los dispositivos de salida
  - Espera al siguiente Tm
- Cálculo de valores adquiridos en sus unidades:
  - Se deben convertir los valores adquiridos (enteros en unidades del dispositivo de entrada) en valores del sistema (reales en unidades del sistema).
  - El cálculo típicamente es una función lineal.
- Ejemplo. Adquisición y cálculo temperatura leída:



Programación del lazo de control

- Programación del lazo de control:
  - **Adquisición de datos**
  - **Cálculo de valores adquiridos en sus unidades**
  - …
- Ejemplo: conversor A/D
  - Funcionamiento del conversor A/D de n bits:
    - Divide un rango de tensión de entrada en 2^n intervalos, asociando a cada uno de ellos un valor numérico ($0…2^{n-1}$).
    - El resultado de la conversión es un valor entero utilizable por el programa.
  - Valores típicos de n: 8, 10, 12, 16, 20, 24, 32.

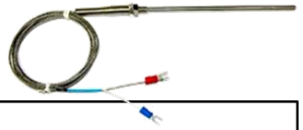


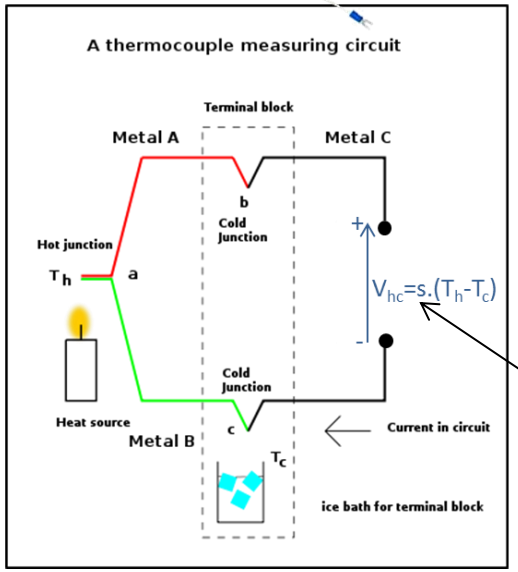
Programación del lazo de control

- Programación del lazo de control:
  - **Adquisición de datos**
  - **Cálculo de valores adquiridos en sus unidades**
  - …

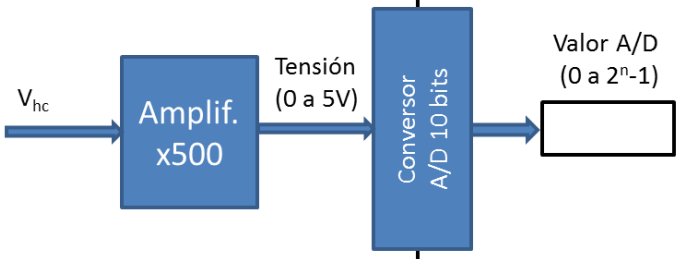
- Ejemplo: adquisición de Tª medida por termopar con conversor A/D de 10 bits y límites de tensión 0-5V

Termopar:





s=43 μV/ºC (termopar tipo T)



¡¡ Usar #define para todas las constantes !!

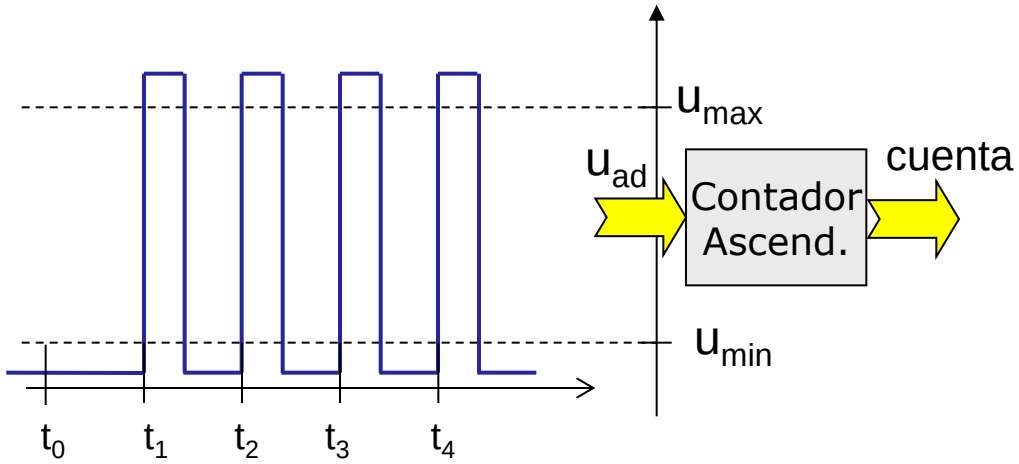
Programa:

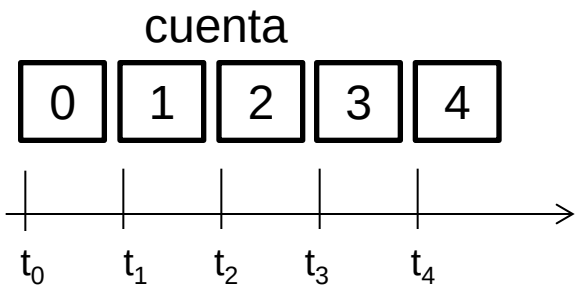
```
int adk;
float Vad_k,Vhc_k,Th_k;
float s=43e-6f; // V/°C
float Tc=0;

adk=LeerAD();
Vad_k=adk*5.0f/1023.0f;
Vhc_k=Vad_k/500.0f;
Th_k=Vhc_k/s-Tc;
```

Programación del lazo de control

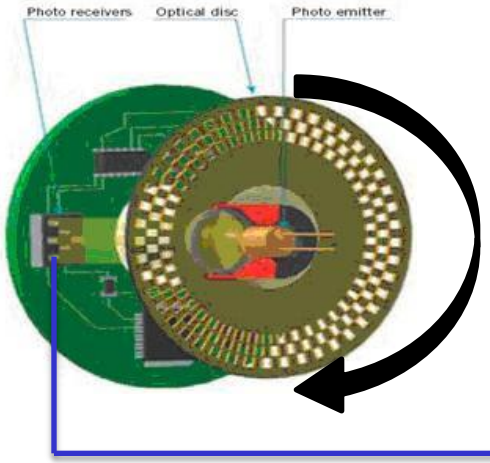
- Programación del lazo de control:
  - **Adquisición de datos**
  - **Cálculo de valores adquiridos en sus unidades**
  - …
- Ejemplo: entrada de pulsos
  - Funcionamiento de una entrada de pulsos
    - Incrementa un contador (entero de n bits) cada vez que se produce un pulso (de subida, de bajada, ambos) en la señal de entrada (todo/nada).
    - El valor del contador es un valor entero utilizable por el programa.
    - Se puede reiniciar el contador a 0 por programa.
    - ¡¡ Ojo con desbordamientos del contador !!

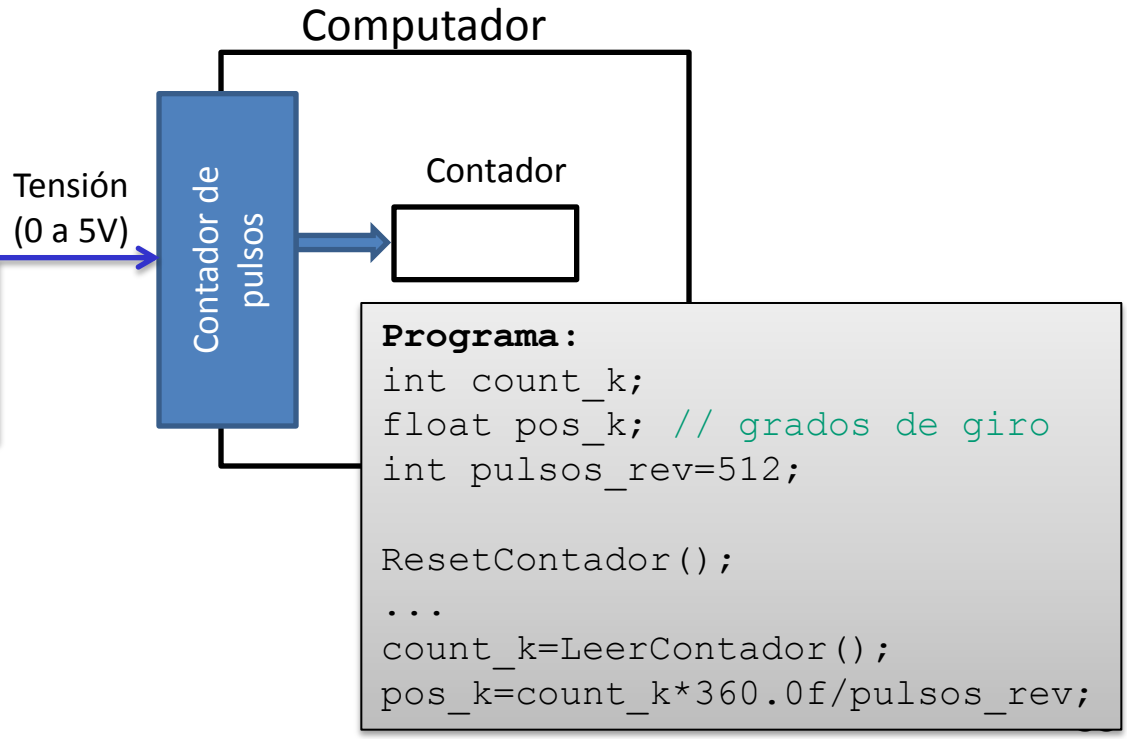




Programación del lazo de control

- Programación del lazo de control:
  - **Adquisición de datos**
  - **Cálculo de valores adquiridos en sus unidades**
  - …
- Ejemplo: entrada de pulsos de un encóder óptico:





Programa:

```
int count_k;
float pos_k; // grados de giro
int pulsos_rev=512;

ResetContador();
...
count_k=LeerContador();
pos_k=count_k*360.0f/pulsos_rev;
```

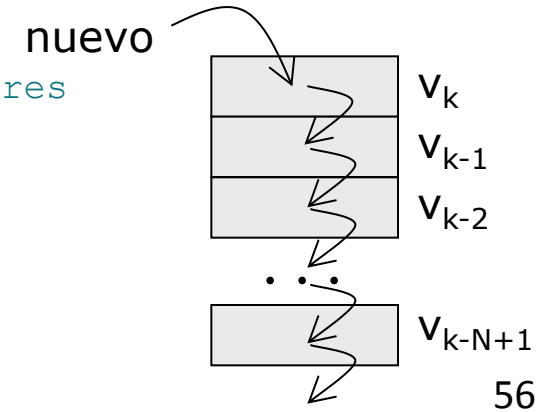
Programación del lazo de control

- Programación del lazo de control:
  - Adquisición de datos
  - Cálculo de valores adquiridos en sus unidades
  - **Actualización de tablas temporales**
  - Cálculo de la acción de control (regulador)
  - Cálculo de los valores de salida
  - Escritura de los dispositivos de salida
  - Espera al siguiente Tm

- Actualización de tablas temporales:
  - Los valores anteriores de las tablas que almacenan señales temporales pasan a ser una muestra más antiguos.
  - Los nuevos valores obtenidos se insertan en el elemento 0 de las tablas.

Función Desplaza()

```
/* Cada nueva temporización: todos los valores
     están 1 instante más retrasados */
int i;
for (i=N-1;i>0;i--)
     vk[i]=vk[i-1];
/* Ahora se pone el nuevo valor */
vk[0]=Valor del nuevo instante;
```



Programación del lazo de control

- Programación del lazo de control:
  - Adquisición de datos
  - Cálculo de valores adquiridos en sus unidades
  - Actualización de tablas temporales
  - **Cálculo de la acción de control (regulador)**
  - Cálculo de los valores de salida
  - Escritura de los dispositivos de salida
  - Espera al siguiente Tm

- Cálculo de la acción de control (regulador):
  - El regulador habrá sido calculado como una FdeT discreta:

$$R(z)=\frac{U(z)}{E(z)}=\frac{b_0+b_1.z^{-1}+...+b_m.z^{-m}}{1+a_1.z^{-1}+...+a_n.z^{-n}}$$

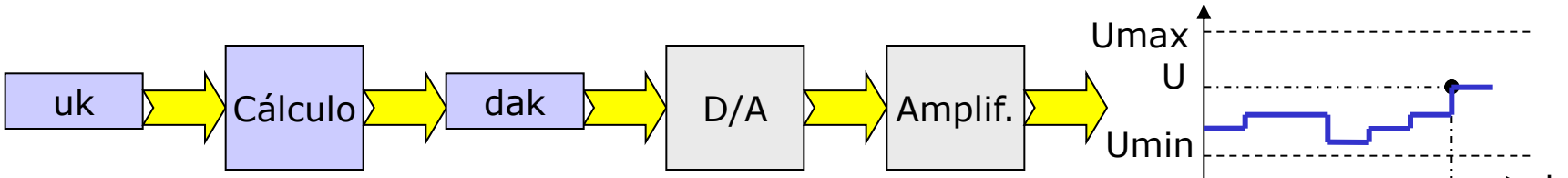
  - La FdeT discreta se corresponde con la ecuación en diferencias a implementar:

$$u_k=\underbrace{b_0.e_k+b_1.e_{k-1}+...+b_m.e_{k-m}}_{\text{ProdEscalar(b,ek,m+1)}}-\underbrace{(a_1.u_{k-1}+...+a_n.u_{k-n})}_{\text{ProdEscalar(a+1,uk+1,n)}}$$

Programación del lazo de control

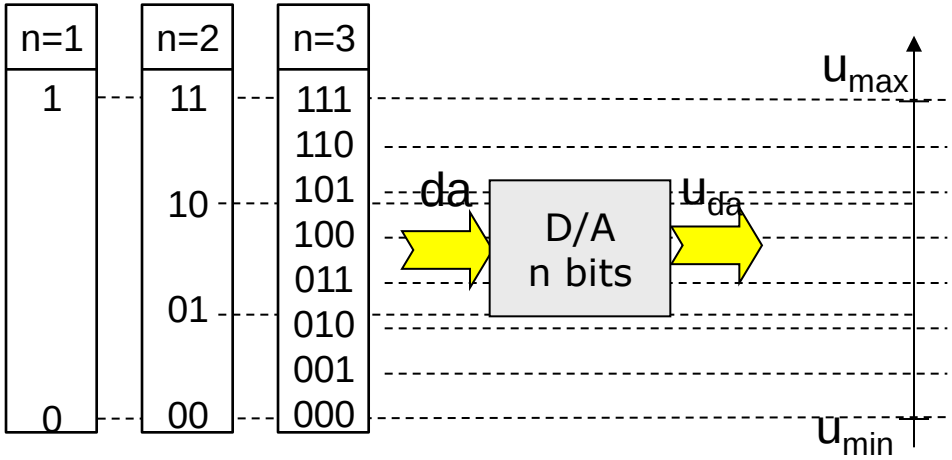
- Programación del lazo de control:
  - Adquisición de datos
  - Cálculo de valores adquiridos en sus unidades
  - Actualización de tablas temporales
  - Cálculo de la acción de control (regulador)
  - **Cálculo de los valores de salida**
  - **Escritura de los dispositivos de salida**
  - Espera al siguiente Tm

- Cálculo y escritura de los valores de salida:
  - El valor de la acción de control es un real en unidades del sistema (entrada).
  - Para actualizar la entrada al sistema se debe escribir un nuevo valor sobre un dispositivo de salida (conversor D/A, salida PWM, salida digital, …).
  - Al igual que los dispositivos de entrada, los dispositivos de salida manejan valores numéricos típicamente enteros.
  - La entrada final al sistema está directamente relacionada con el valor escrito en el dispositivo de salida.



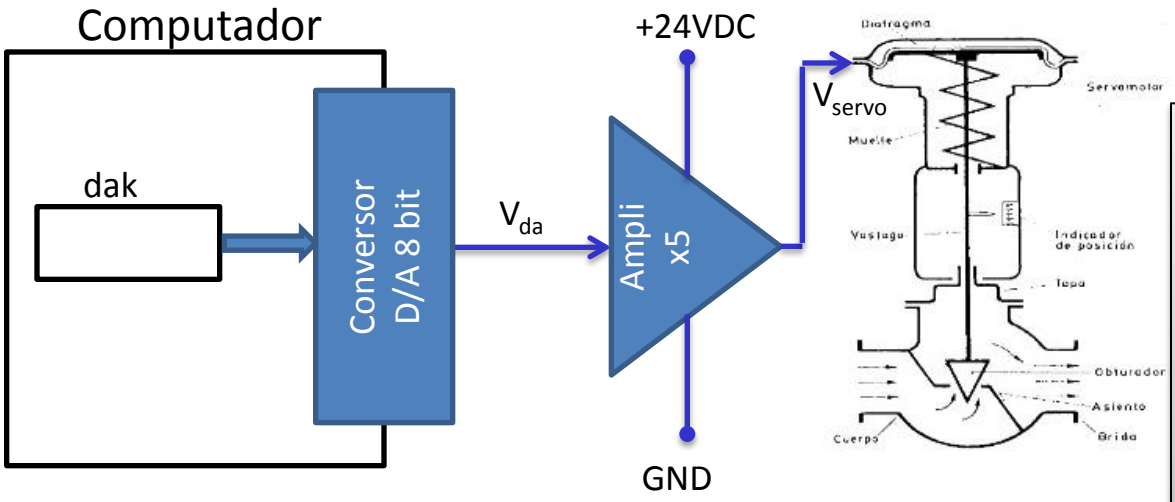
Programación del lazo de control

- Programación del lazo de control:
  - …
  - **Cálculo de los valores de salida**
  - **Escritura de los dispositivos de salida**
  - …
- Ejemplo: conversor D/A
  - Funcionamiento del conversor D/A de n bits:
    - Genera una tensión de salida en función de un valor numérico ($0…2^{n-1}$) generado por el programa.
  - Valores típicos de n: 8, 10, 12, 16, 20, 24, 32.



Programación del lazo de control

- Programación del lazo de control:
  - **Adquisición de datos**
  - **Cálculo de valores adquiridos en sus unidades**
  - …
- Ejemplo: accionamiento de una electroválvula proporcional de 24VDC, mediante conversor DA de 8 bits y límites de tensión 0-5V

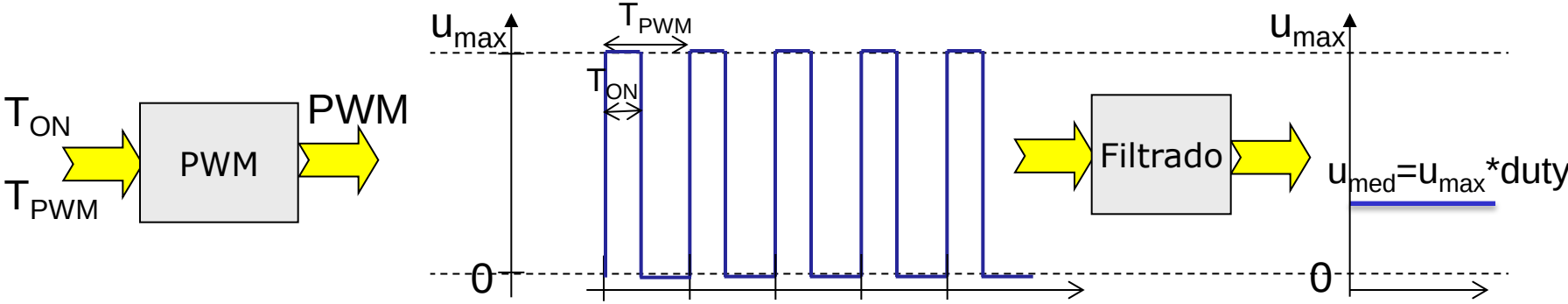



```
Programa:
float pct_k;
float Vservo_k,Vda_k;
int dak;

pct_k=apertura deseada(%);
Vservo_k=pct_k/100.0f*24.0f;
Vda_k=Vservo_k/5.0f;
dak=Vda_k/5.0f*255;
EscribirDA(dak);
```

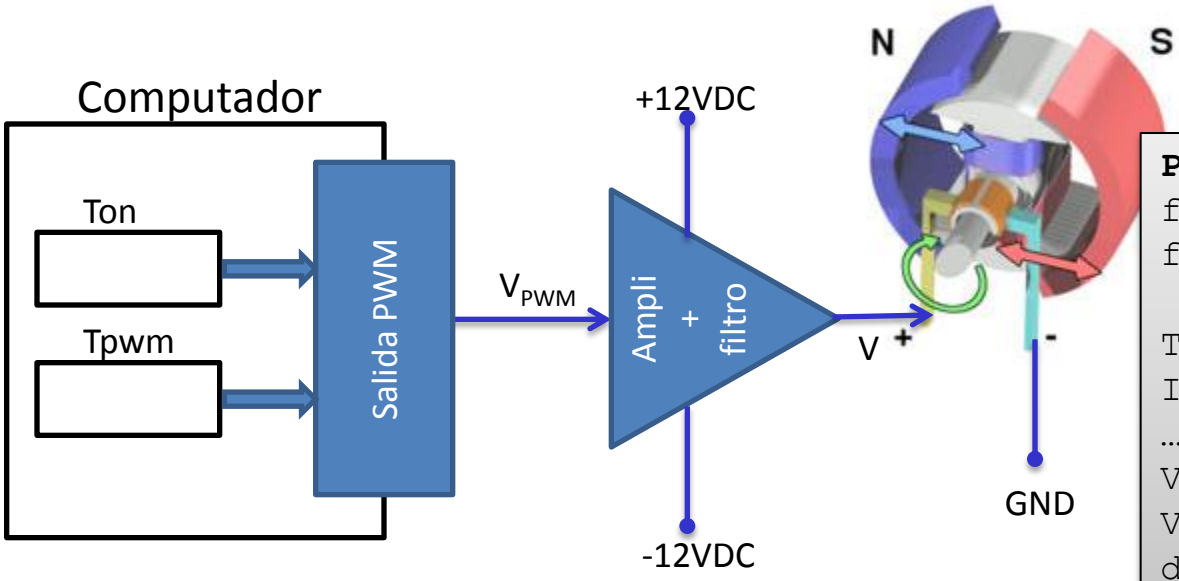
Programación del lazo de control

- Programación del lazo de control:
  - …
  - **Cálculo de los valores de salida**
  - **Escritura de los dispositivos de salida**
  - …
- Ejemplo: generador PWM
  - Funcionamiento del conversor PWM:
    - Genera una onda tensión de salida alternativamente ON/OFF, siendo programable el tiempo de onda (T_{PWM}) y la duración de encendido (T_{ON}).
    - La relación T_{ON} / T_{PWM} se conoce como ciclo duty.
    - El resultado, una vez filtrado, es un valor analógico constante proporcional al valor del ciclo duty.
    - T_{PWM} suele ser fijo y función del filtro utilizado.



Programación del lazo de control

- Programación del lazo de control:
  - **Adquisición de datos**
  - **Cálculo de valores adquiridos en sus unidades**
  - …
- Ejemplo: accionamiento de un motor DC de 12V mediante un accionamiento PWM de 5V.



Programa:

```
float V_k,Vpwm_k;
float duty_k,Tpwm,Ton_k;

Tpwm=valor constante;
InicializarPWM(Tpwm);
…
V_k=tensión deseada (Volt);
Vpwm_k=(V_k+12.0f)/24.0f*5.0f;
duty_k=Vpwm_k/5.0f;
Ton_k=(int) (duty_k*Tpwm);
EscribirPWM(Ton_k);
```

Programación del lazo de control

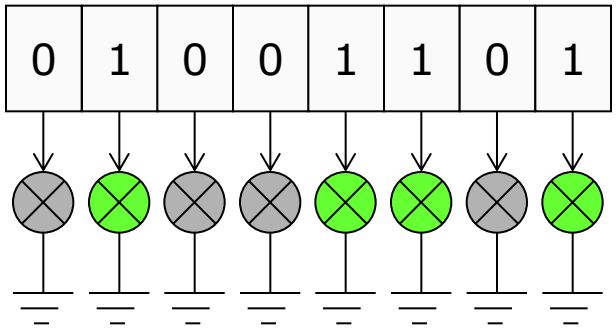
- Programación del lazo de control:
  - Adquisición de datos
  - Cálculo de valores adquiridos en sus unidades
  - Actualización de tablas temporales
  - Cálculo de la acción de control (regulador)
  - Cálculo de los valores de salida
  - Escritura de los dispositivos de salida
  - **Espera al siguiente Tm**

- Espera al siguiente periodo de muestreo:
  - El programa de control no necesita realizar ninguna acción adicional hasta el siguiente periodo de muestreo.
  - El tiempo disponible se puede utilizar para tareas auxiliares:
    - Actualización de interfaz de usuario.
    - Comunicaciones.
    - Atención a entradas del usuario.
    - Almacenamiento en bases de datos.
    - …

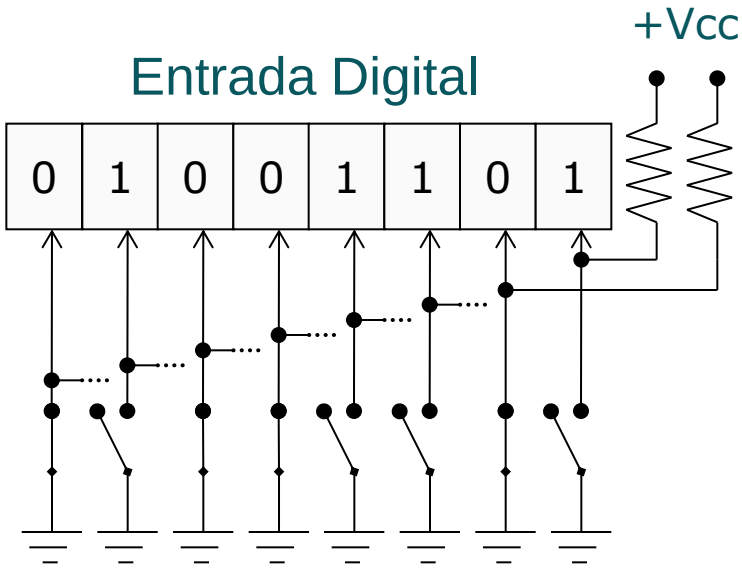
Programación de E/S digital

- El interfaz con muchos dispositivos se realiza en formato todo/nada (1/0).
- Con estos dispositivos, se utilizan E/S digitales.
- Las E/S digitales están formadas por valores enteros, de los cuales cada bit está conectado a un dispositivo externo.

Salida Digital



Entrada Digital



Operaciones con variables

- Operaciones con el bit de peso P de la variable entera X:
  - En la mayoría de ocasiones, de debe operar con un solo bit (E ó S) sin alterar el resto.

```
int x,p;
int mascara;
…Dar valores a p y x…
mascara=1<<p;
```

  - ¿Está activo el bit de peso P de la variable X?

```
if (x & mascara)
  …
```

  - Poner a 1 el bit de peso P de la variable X:

```
x=x | mascara;
```

  - Poner a 0 el bit de peso P de la variable X:

```
x=x & ~mascara;
```

  - Cambiar el valor del bit de peso P de la variable X:

```
x=x ^ mascara;
```

Indice

- Introducción control de procesos por computador
- Las matemáticas del control
- Programación del lazo de control
- **Programación en lenguaje C**
- Implantación del control en el computador
- El control secuencial

Programación en lenguaje C

- Ver presentación específica de lenguaje C

Indice

- Introducción control de procesos por computador
- Las matemáticas del control
- Programación del lazo de control
- Programación en lenguaje C
- **Implantación del control en el computador**
- El control secuencial

Computadores para control

- Microcontrolador

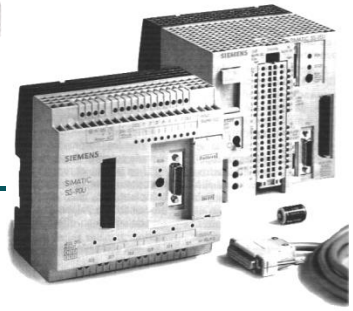
- Procesador Digital de Señal (DSP)

- Dispositivos Electrónicos Programables (FPGA, PLD)

- Computador embebido

- Ordenador Industrial

- Autómata Programable (PLC)



Implantación del control

- **Variables**: almacenan valores actuales y anteriores de entradas, salidas, estados y parámetros.
- **Funciones**: realizan cálculos habituales (media, rz, gráfico, …).
- **Funcionamiento temporal**: los valores evolucionan en el tiempo.
- **Acceso a E/S**: es necesario intercambiar datos con el mundo real.
- **Tiempo-real**: importan el valor de la respuesta y su plazo.
- **Multi-tarea**: varias cosas que hacer 'a la vez'.
- **Programación orientada a eventos**: la secuencia de acontecimientos es desconocida a priori
- **Ejemplos de eventos**:
  - Vencimiento temporización
  - Cambio de estado de entrada(s)
  - Entrada de datos por teclado
  - …

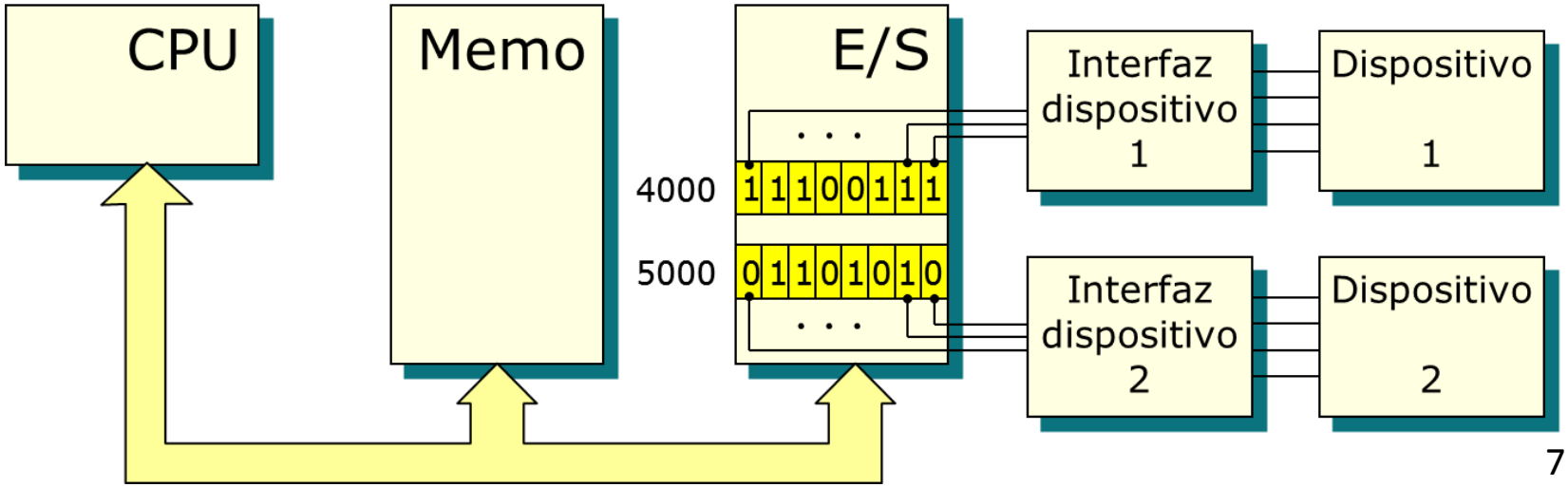
  } **Interrupciones**

- **Prioridades:** algunos eventos son más importantes que otros

Interrupciones:
Señales binarias externas al computador que, cuando se activan, provocan que la CPU deje temporalmente de ejecutar el programa en curso, para ejecutar el código de tratamiento de la interrupción (ISR).
Una vez terminado el código de la ISR, la CPU sigue ejecutando el programa.

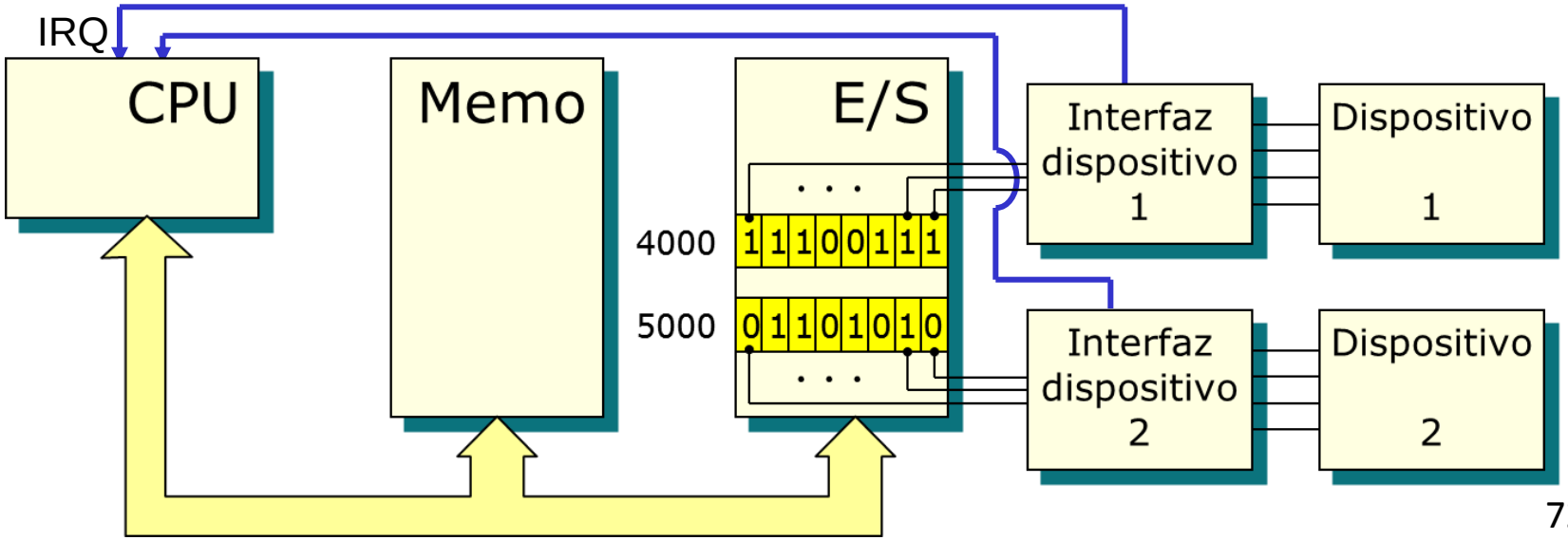
Implantación del control

- Programación de E/S:
  - El intercambio de datos con el mundo exterior se realiza escribiendo y leyendo puertos de E/S: direcciones “especiales” cuyo contenido provoca cambios o es modificado por los dispositivos de E/S.
  - Los puertos están organizados en direcciones.
  - Es necesario conocer las direcciones de E/S de cada dispositivo, y el significado de cada bit de los puertos.



Programación de E/S

- Servicio de eventos:
  - Los dispositivos de E/S envían una interrupción a la CPU cuando necesitan ser atendidos.
  - La interrupción provoca que la CPU abandone temporalmente el programa que estaba ejecutando, pase a ejecutar el código de la ISR, retornando al programa cuando ésta termina.
  - Ejemplo animado disponible en: http://isa.uniovi.es/~ialvarez/Curso/descargas/Funcionamiento%20Computador.pps



Implantación del control

- Sistemas ‘pequeños’:
  - Sin soporte de Sistema Operativo
  - El programador accede directamente al hardware
    - Programación de puertos de E/S
    - Funciones para servicio de interrupciones
  - El programador debe realizar la planificación de la ejecución temporal de las diferentes tareas:
    - Round-robin: chequeo ordenado de las tareas a realizar en cada momento, sin uso de interrupciones.
    - Round-robin con interrupción: las tareas más prioritarias o puntuales son lanzadas por interrupciones hardware y servidas en Rutinas de Servicio de Interrupción (ISR).
    - Function-queue-scheduling: las ISR encolan las tareas a realizar con sus prioridades. Un planificador round-robin comprueba la cola y va sirviendo por orden de prioridad

Implantación del control

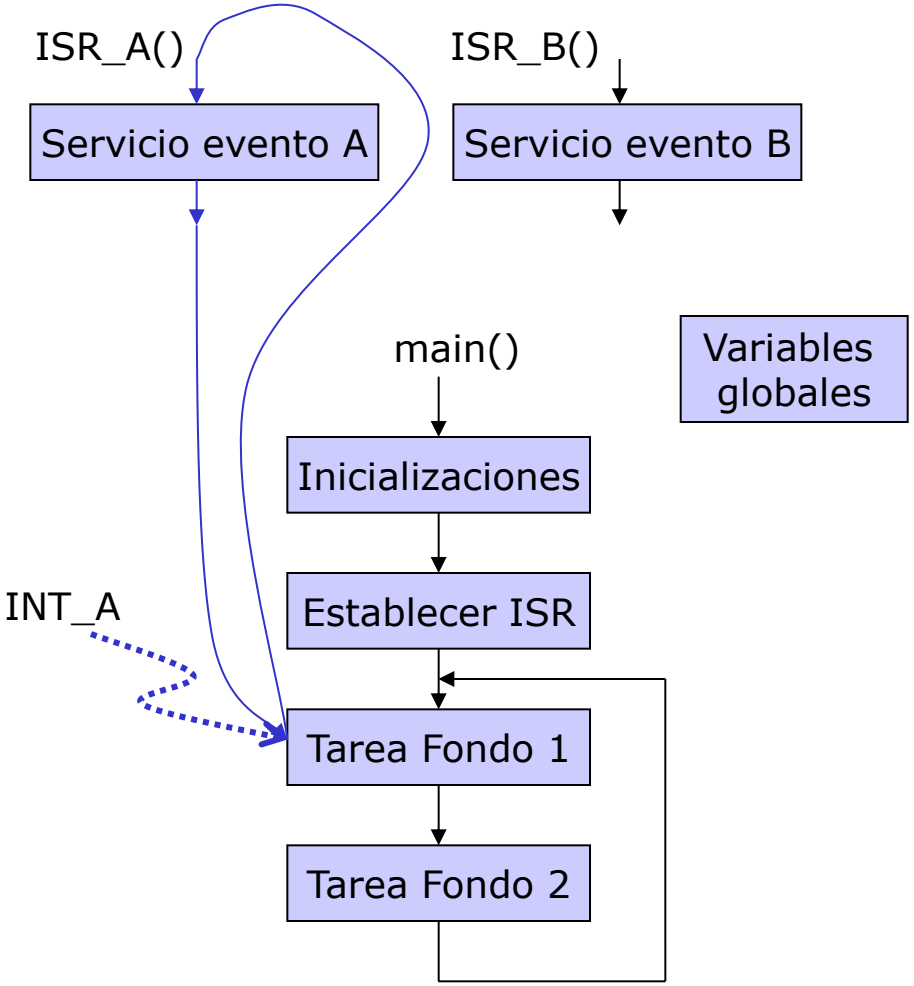
- Sistemas ‘grandes’:
  - Una tarea especial Sistema Operativo (S.O.) se encarga de dar servicio a:
  - Acceso al hardware (modo síncrono y asíncrono):
    - Nivel de S.O.
    - Nivel de driver
  - Planificación de tareas:
    - Cada vez que sucede un ‘evento’, el S.O. toma el control, actualiza estado de tareas, y pasa a ejecutar la más prioritaria.
  - Gestión de memoria
  - El programador debe realizar llamadas a funciones del S.O. para todos los servicios
  - El S.O. puede llamar a funciones del programador (callback) para rutinas de servicio asíncrono

Implantación del control

- Sistemas “pequeños” vs sistemas “grandes”:
  - La tarea S.O. requiere recursos (memoria y tiempo de ejecución), que pueden no estar disponibles en sistemas pequeños.
  - La tarea S.O. requiere ejecutarse en modo privilegiado: la CPU debe disponer de 2 modos de funcionamiento (usuario y privilegiado).
  - La tarea S.O. facilita operaciones que pueden ser requeridas en sistemas grandes:
    - La programación de E/S (no hay que conocer todos los puertos e interrupciones)
    - La gestión de tareas (prioridades, sincronización, memoria)
    - Las protecciones de seguridad basadas en privilegios y claves.
    - Las comunicaciones

Sistemas “pequeños”

- Organización del programa:
  - Un bucle principal con la(s) tarea(s) de fondo.
  - Funciones de Servicio de Interrupción (ISR) para el servicio de eventos.
  - Las ISR detienen a la tarea de fondo o a otras ISR: ejecución rápida y sin esperas.
  - Gestión de prioridades de las ISR.
  - Las variables compartidas por el programa y las ISR deben ser globales.



Sistemas “pequeños”

- Variables globales:
  - Se declaran al principio del programa, fuera de todas las funciones.
  - Existen durante todo el programa, y son accesibles por todas las funciones (incluidas las ISR).
  - Por ello, permiten el intercambio de datos entre funciones que no se llaman directamente (código principal e ISR).
  - **¡¡¡ ATENCION !!!**
    - Es recomendable usar un nombre adecuado para distinguir las variables globales: comenzando por _ (ej: `int _v1;`)
    - No abusar de las variables globales: utilizarlas sólo cuando sea estrictamente necesario.
    - El acceso “concurrente” a variables globales puede dar lugar a problemas de sincronización.

Sistemas “pequeños”

- Ejemplo con interrupciones:
  - Realización de un control todo/nada temporizado, con entrada de consigna por teclado.

```
#define TM_MS         100
...
float _ck;    // La consigna es global para que
              // sea accesible por main() y la ISR

void ISR_Control()
// Se ejecuta por interrupción cada TM_ms:
    // (1) Debe ejecutarse rápido y sin esperas
    // (2) Sus variables locales se crean y
    //     destruyen en cada llamada
    // (3) Se comunica con main() mediante
    //     la variable global _ck
{
    int adk;
    float yk;

    adk=LeerCanalAD(...);
    yk=ConvertirValor(adk,...);
    if (_ck - yk > 0)
        ActivarSalidaControl(...);
    else
        DesactivarSalidaControl(...);
}
```

```
main()
{
    ...
    // Programa el dispositivo de E/S
    // temporizador para generar una interrupción
    // cada TM_ms, y establece que la
    // función de tratamiento será ISR_Control
    EstablecerISRTemporiz(TM_ms,ISR_Control);

    while (1)
    {
        // Este código será interrumpido cada
        // TM_ms para hacer un paso del control,
        // retornando después al mismo código.
        printf("Introduzca consigna:");
        scanf("%d",&_ck);
    }
}
```

Sistemas “pequeños”

- Ejemplo con interrupciones:
  - Realización de un control R(z) con entrada de consigna por teclado.

```
#define TM_MS          100
#define M              2
#define N              3
...
float _ck;
float _a[N+1],_b[M+1],_uk[N+1],_ek[M+1];

void ISR_Control()
{
     int adk,dak;
     float yk;

     DesplazaTabla(_uk,N+1);
     DesplazaTabla(_ek,M+1);
     adk=LeerCanalAD(...);
     yk=ConvertirValor(adk,...);
     _ek[0]=_ck - yk;
     _uk[0]=  ProdEsc(_b,_ek,M+1)-
              ProdEsc(_a+1,_uk+1,N);
     dak=ConvertirValor(_uk[0],...);
     ActivarSalidaAnalogica(dak,...);
}
```

```
main()
{
     ...
     InicializacionVariables(...);
     EstablecerISRTemporiz(TM_ms,ISR_Control);

     while (1)
     {
          printf("Introduzca consigna:");
          scanf("%d",&_ck);
     }
}
```

Son globales porque necesitan mantener su valor entre una llamada y otra de la función (además, normalmente será main quien les dé los valores iniciales)

Son locales porque NO necesitan mantener su valor entre una llamada y otra de la función