INGENIERÍA DE SISTEMAS Y AUTOMÁTICA

Universidad de Oviedo

Control por Computador

Ignacio Alvarez García

Octubre - 2018

Indice

- Introducción al control de procesos por computador
- Conceptos básicos para programación de control
- Interfaz del computador con el exterior
- Las matemáticas del control
- Programación del lazo de control
- Programación en lenguaje C
- Implantación del control en el computador
- El control secuencial

Indice

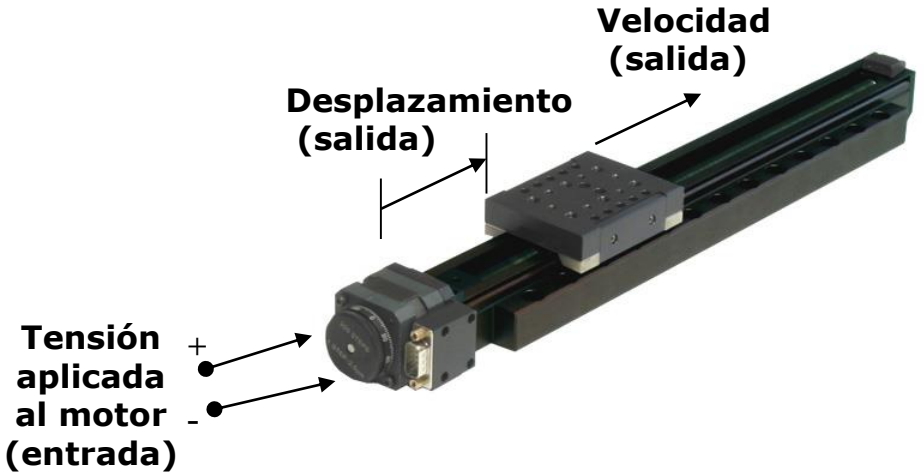
- **Introducción al control de procesos por computador**
- Conceptos básicos para programación de control
- Interfaz del computador con el exterior
- Las matemáticas del control
- Programación del lazo de control
- Programación en lenguaje C
- Implantación del control en el computador
- El control secuencial

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

Ejemplo 1: sistema térmico (calentador de fluido por gas)

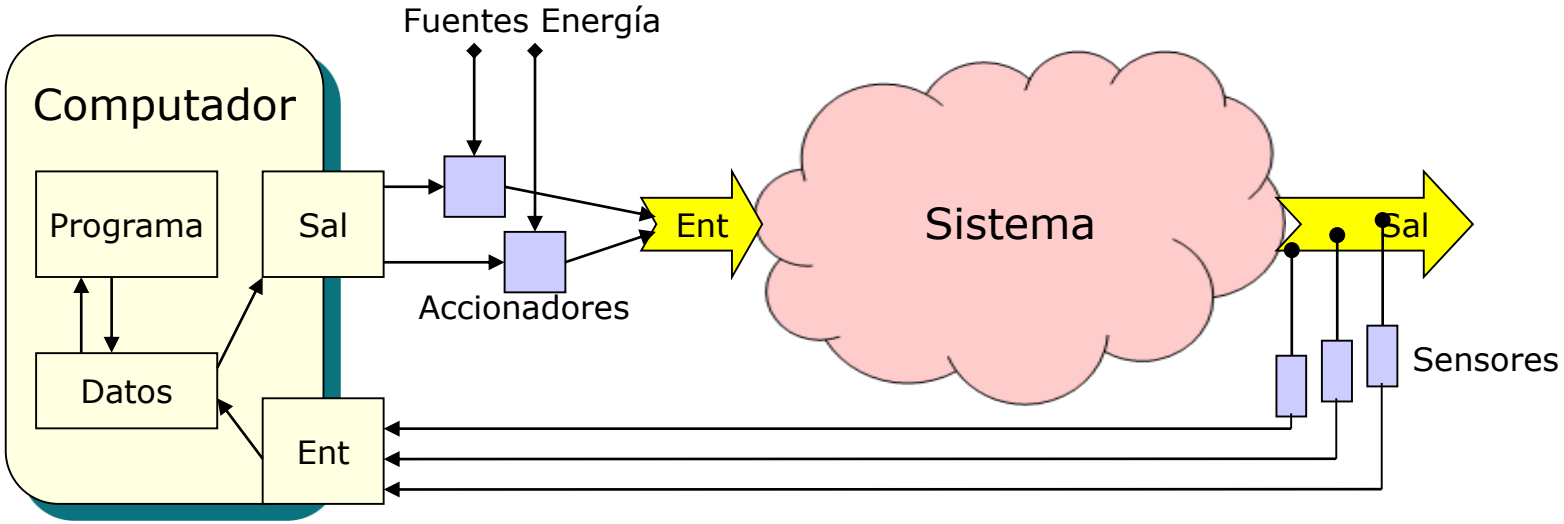
Ejemplo 2: sistema mecánico (motor CC + guía lineal)



El Control de Procesos por Computador

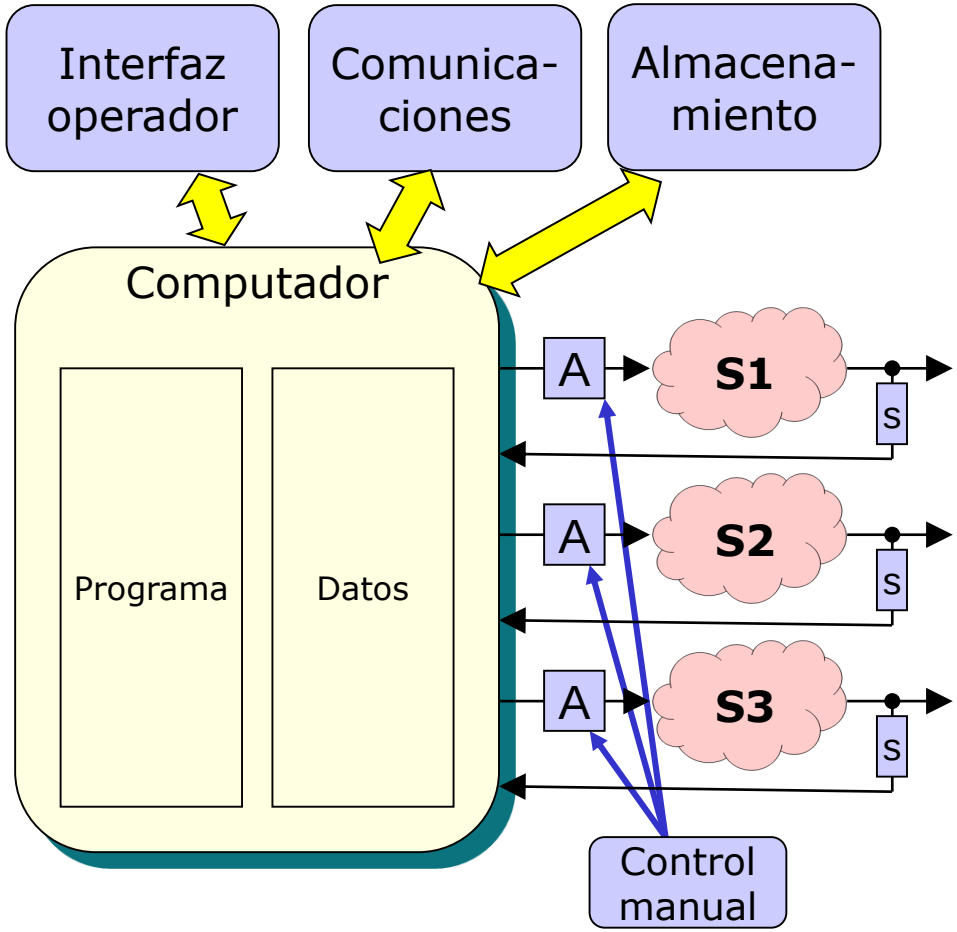
- Control de un proceso o sistema
  - Variables principales involucradas:
    - **Referencia o consigna**: valor deseado para la salida en cada momento
    - **Error**: Valor referencia – Valor salida
    - **Acción de control**: valor a aplicar en la entrada para conseguir (idealmente) error cero
  - Todas las variables evolucionan en el tiempo

- 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



- **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
- **Almacenamiento:**
  - De configuraciones y resultados
- **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

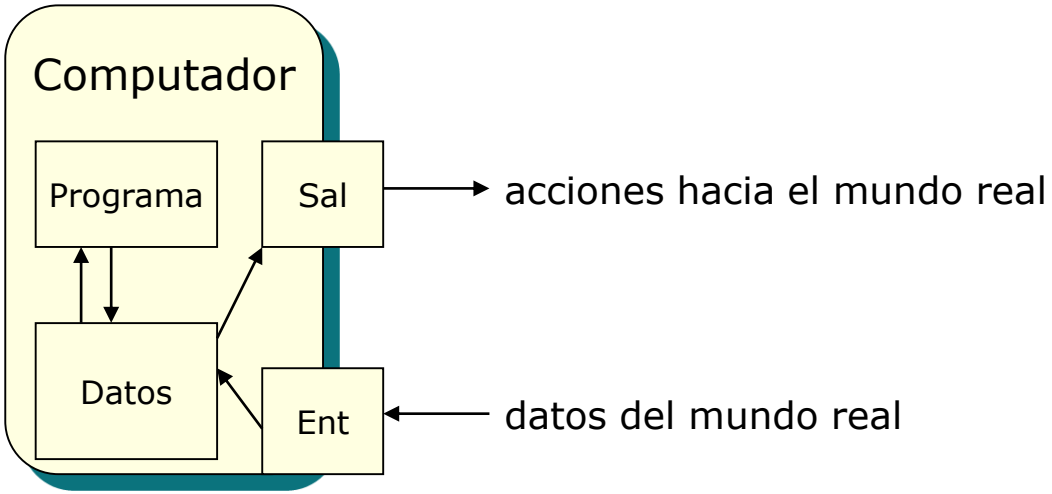
Teléfono móvil

Control de vuelo de un avión

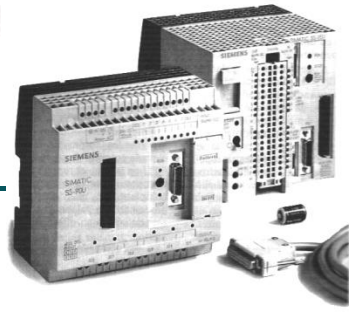
Línea de fabricación de botes

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 (variables).
  - 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) 

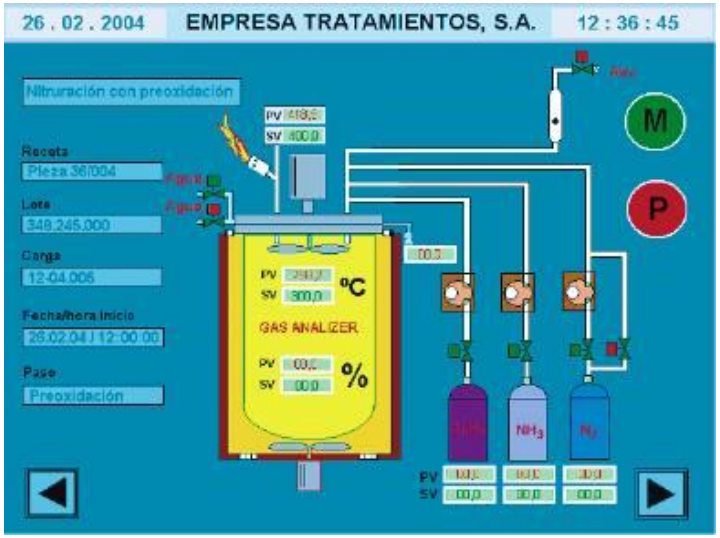
Sensores

- 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

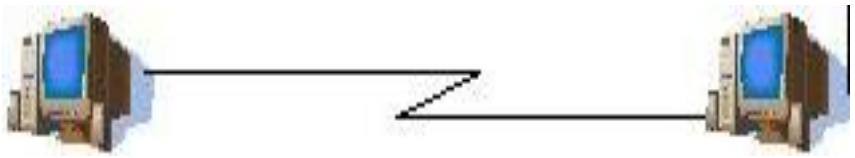
Pupitre / botonera

Display gráfico personal

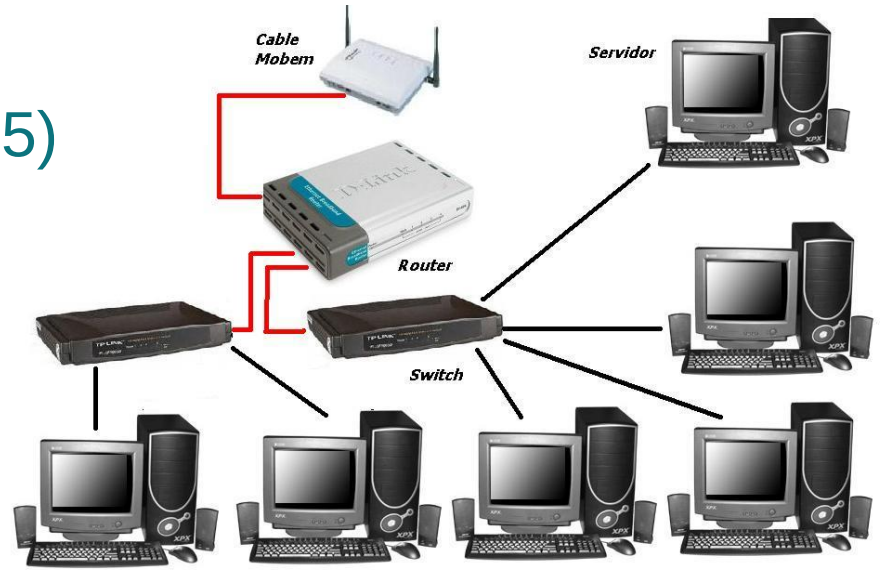
- Dispositivos salida interfaz humana:
  - Indicador luminoso (LED)
  - Display (7 segmentos, LCD,…)
  - Pantalla (texto/gráficos)
  - Impresora
  - …
- Dispositivos entrada interfaz humana:
  - Botón / pulsador
  - Selector rotativo
  - Teclado
  - Ratón / Puntero / Joystick
  - …

Comunicaciones

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



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

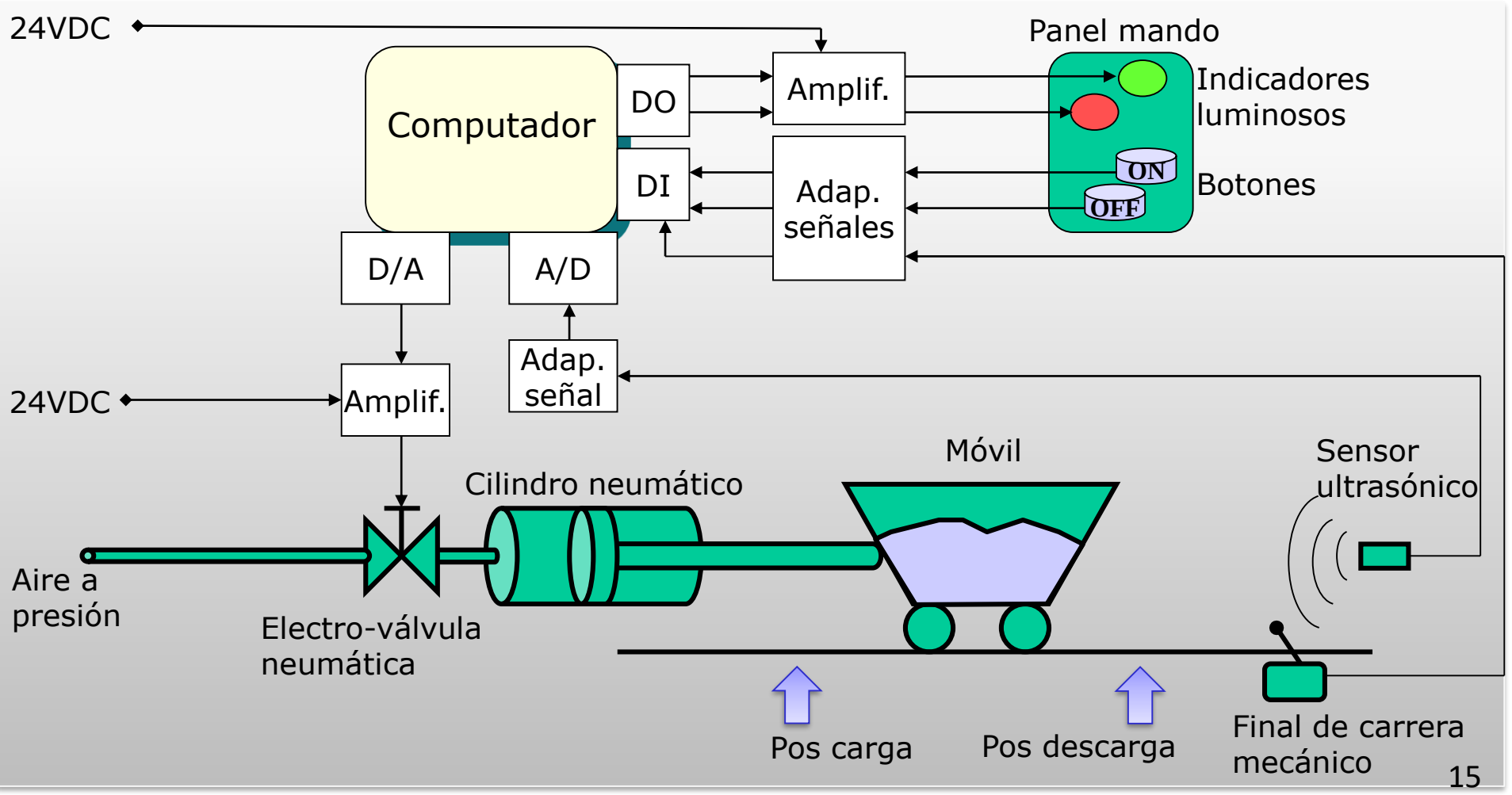


Niveles de control

- Control secuencial:
  - Genera la secuencia de acontecimientos deseada
  - Determina los valores deseados (consignas) de las variables a controlar
  - Diagrama de estados / transiciones
  - Cambio de estados por:
    - Tiempo máximo
    - Combinación de valores de entradas
    - Acciones del operador o de otros computadores
- Lazo de control:
  - Asegura que las variables a controlar alcancen el valor de sus consignas con precisión y velocidad
  - Compara la **variable a controlar** con la **consigna**, y modifica la **acción de control** para que se igualen

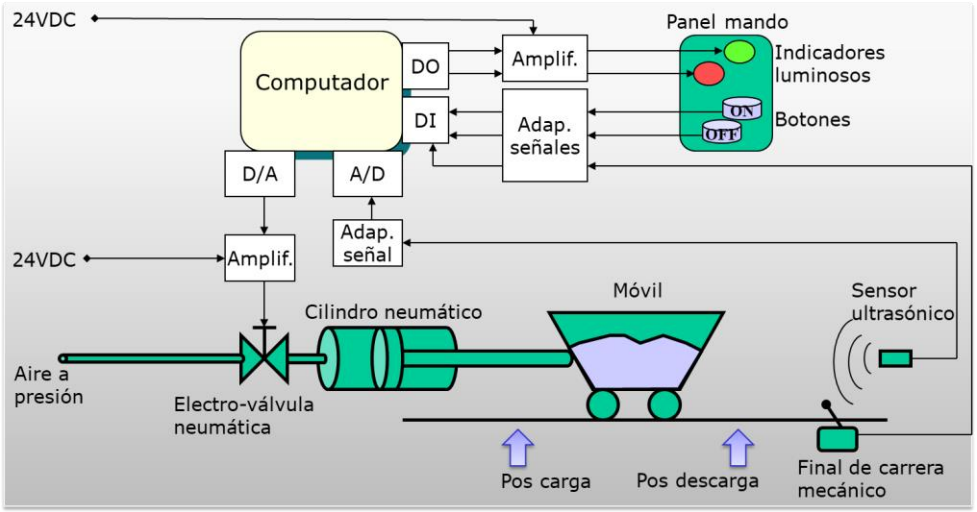
Ejemplo de sistema de control

- Control del posición de una carretilla



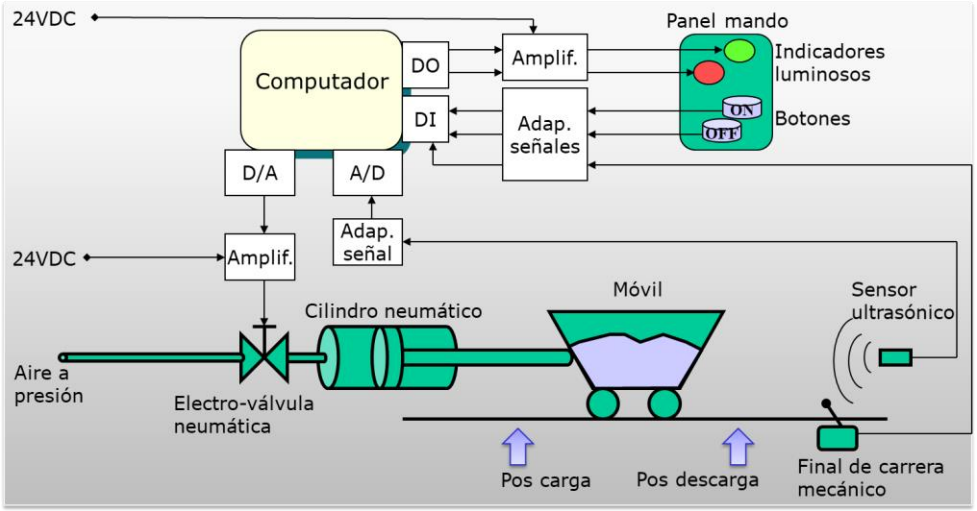
Niveles de control

- Ejemplo: Control secuencial
  - Cuando se pulsa el botón ON, la carretilla debe repetir el ciclo siguiente:
    - Moverse a la posición de carga
    - Permanecer 5 seg
    - Moverse a la posición de descarga
    - Permanecer 2 seg
  - Si se alcanza el final de carrera mecánico, activar alarma y detener ciclo
  - Si el operador pulsa el botón OFF, terminar último ciclo y no repetir



Niveles de control

- Ejemplo: Lazo de control
  - Cuando ordena el movimiento de la carretilla a una posición, se debe alcanzar la misma con la precisión y velocidad requeridas
  - Repetir hasta llegar al destino:
    - Leer posición actual del sensor ultrasónico
    - Calcular aire a inyectar para moverse hacia posición destino
    - Abrir electro-válvula para inyectar el aire deseado

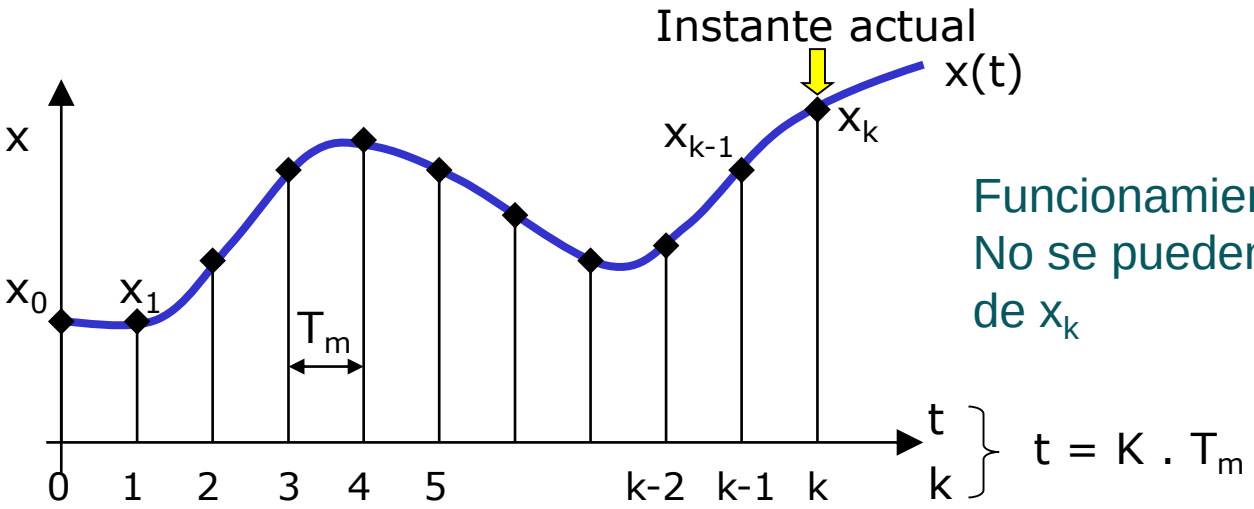


Indice

- Introducción al control de procesos por computador
- **Conceptos básicos para programación de control**
- Interfaz del computador con el exterior
- Las matemáticas del control
- Programación del lazo de control
- Programación en lenguaje C
- Implantación del control en el computador
- El control secuencial

Variables continuas y discretas

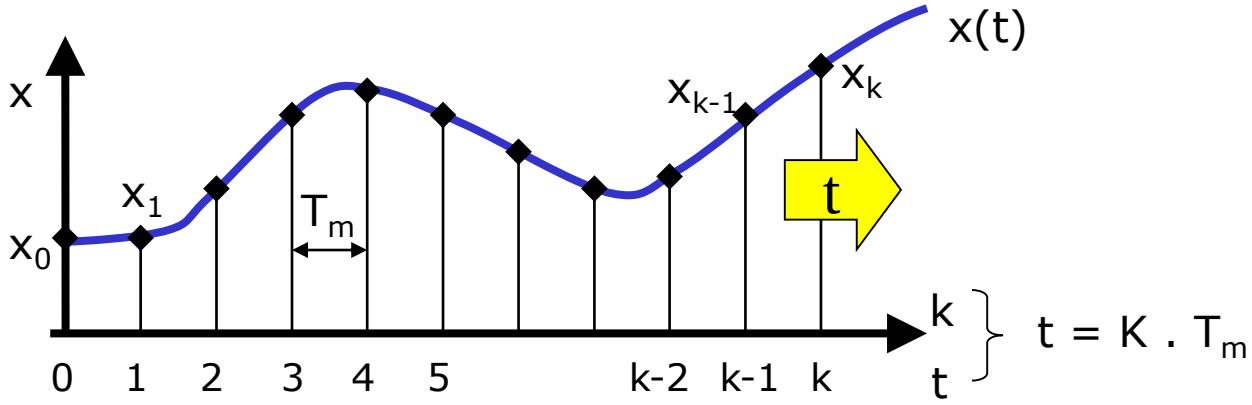
- En el mundo real, las variables son continuas y analógicas.
- En el computador, las variables son discretas y digitales.
- Se representa una señal analógica como x(t)
- Se representa una secuencia de valores discretos como $\{x_k\}$
- En el computador sólo se dispone de valores discretos, obtenidos en instantes separados Tm (periodo de muestreo).



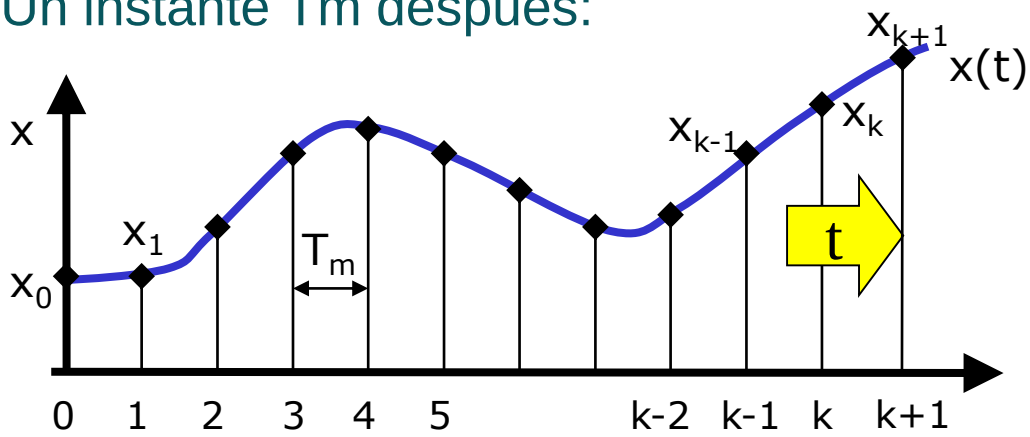
Funcionamiento hasta $t \rightarrow \infty$
No se pueden guardar ∞ valores de x_k

Variables temporales discretas

- En el mundo real, el tiempo avanza…



- Un instante Tm después:

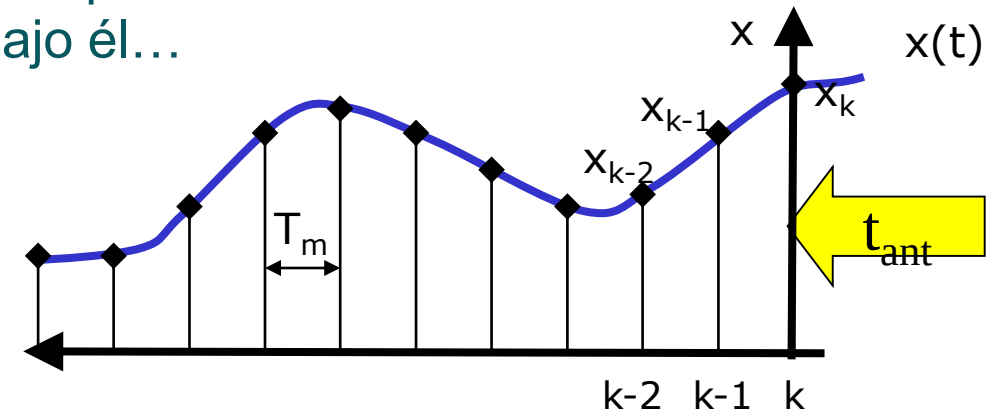


El nuevo valor de este instante es x_{k+1}

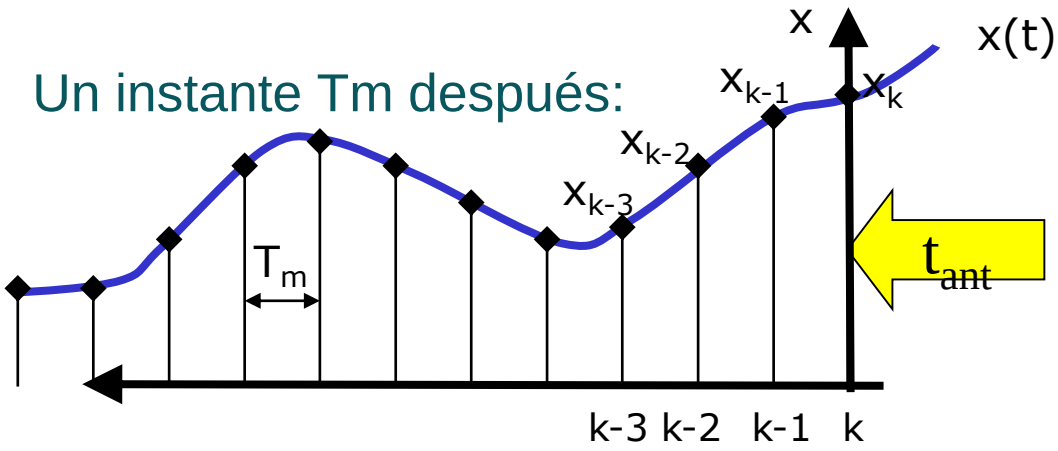
Los valores anteriores se mantienen

Variables temporales discretas

- Desde el punto de vista del programa de control, consideramos siempre el momento actual como el instante **k**, el tiempo pasa bajo él…



- Un instante Tm después:



El nuevo valor de este instante es ahora x_k

El anterior x_k ahora es x_{k-1} (se ha retrasado 1 Tm respecto al instante actual)

El anterior x_{k-1} ahora es x_{k-2}

...

Programa básico de control

- Un programa básico de control tendrá el aspecto siguiente:

```
#include ...

int main()
{
   // Declaración de variables necesarias, al menos:
   float ref_k;   // Valor de la consigna actual (instante k)
   float y_k;     // Valor de la salida actual
   float u_k;     // Valor de la acción de control a aplicar

   // Inicializaciones necesarias
   ...

   // Bucle de control
   while (1)
   {
      ref_k=ValorDeLaReferencia(); // De operador, cálculo, medida, etc.
      y_k=ValorDeLaSalida(); // Medida mediante sensor
      u_k=CalculoAccionDeControl(y_k,ref_k); // Según algoritmo deseado
      AplicarAccionDeControl(uk); // Enviar a accionador
      Sleep(Tm);  // No ejecutar código hasta el siguiente Tm
   }
}
```

Programa básico de control

- Si el programa necesita manejar valores actuales y anteriores de una señal (ejemplo, acción de control depende de crecimiento de y_k : necesita y_k e y_{k-1})

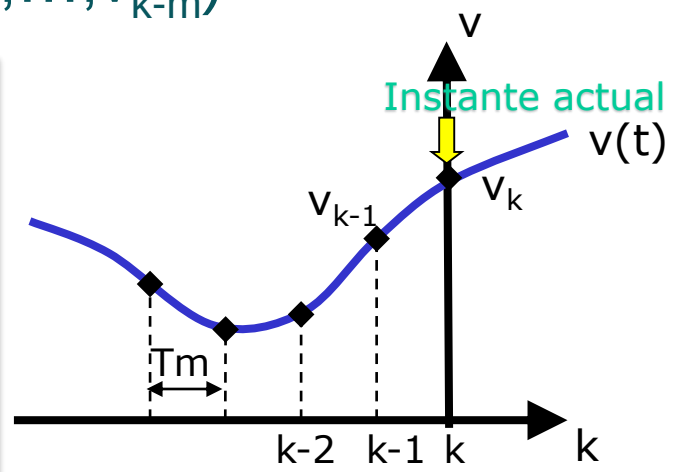
```
#include ...

int main()
{
   float ref_k;
   float y_k,y_k_menos_1;      // Valor de la salida actual y la anterior
   float u_k;

   // Inicializaciones necesarias
   . . .
   // Bucle de control
   while (1)
   {
      y_k_menos_1=yk;  // La última yk (del instante anterior) es ahora yk-1
      ref_k=ValorDeLaReferencia();
      y_k=ValorDeLaSalida();
      u_k=CalculoAccionDeControl(y_k,y_k_menos_1,ref_k);
      AplicarAccionDeControl(uk);
      Sleep(Tm);
   }
}
```

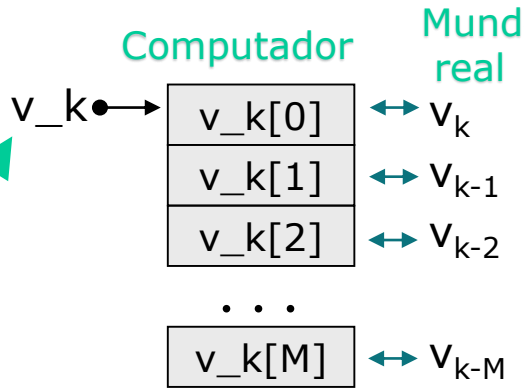
Programa básico de control

- Generalizando: el programa necesita el valor actual y 'm' anteriores de la señal v (v_k,v_{k-1},…,v_{k-m})

```
#include ...
#define M  6

int main()
{
   float v_k[M+1];  // Desde 0 a m incluido

// Inicializaciones
   ...
// Bucle de control
   while (1)
   {
      v_k[M]=v_k[M-1];
      v_k[M-1]=v_k[M-2];
      ...
      v_k[2]=v_k[1];
      v_k[1]=v_k[0];
      v_k[0]=nuevo_valor();
      // Usar tabla v_k con la interpretación

      Sleep(Tm);
   }
}
```

Mejor función:
Desplaza(v_k,M+1);



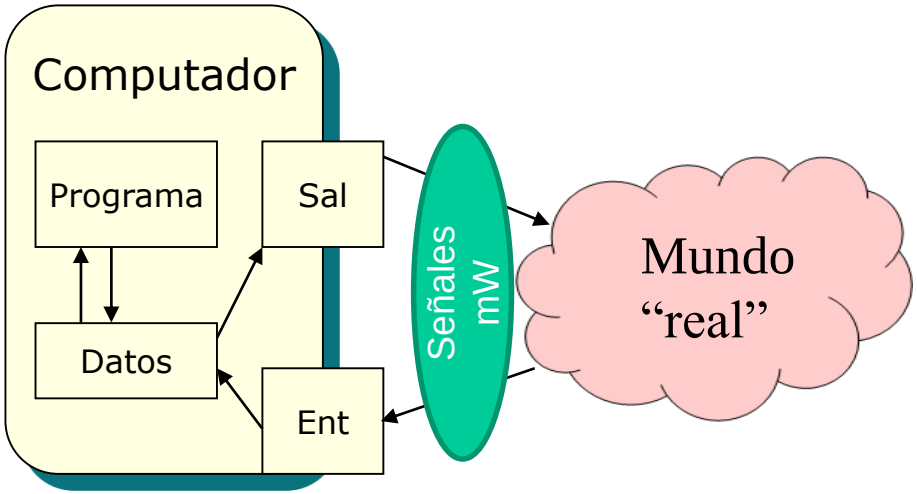


Indice

- Introducción al control de procesos por computador
- Conceptos básicos para programación de control
- **Interfaz del computador con el exterior**
- Las matemáticas del control
- Programación del lazo de control
- Programación en lenguaje C
- Implantación del control en el computador
- El control secuencial

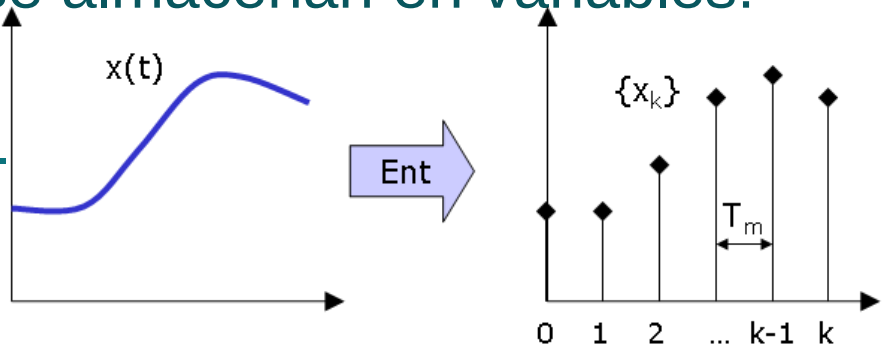
Interfaz del computador con el exterior

- El programa del computador maneja datos que son variables alojadas en su memoria
- Los datos manejados por el programa de control proceden o tienen como destino el mundo “real”
- Los computadores disponen de subsistemas electrónicos llamados de Entrada y Salida (E/S ó I/O) para hacer el intercambio de información entre mundo exterior y variables internas.
- El interfaz se realiza mediante pequeñas tensiones y corrientes (potencia de mW)

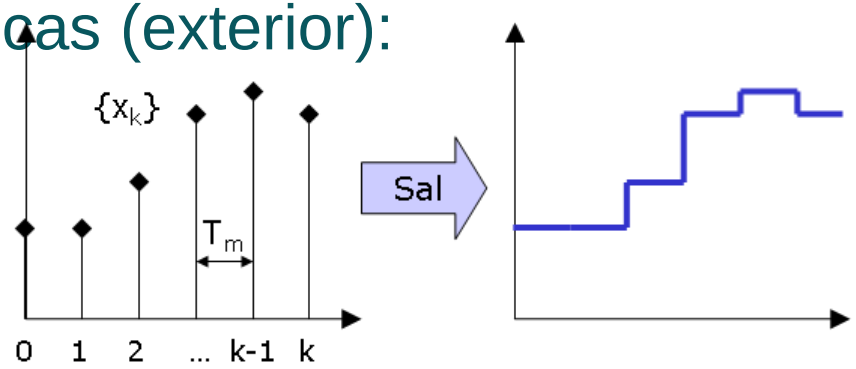


Interfaz con el “mundo real”

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



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

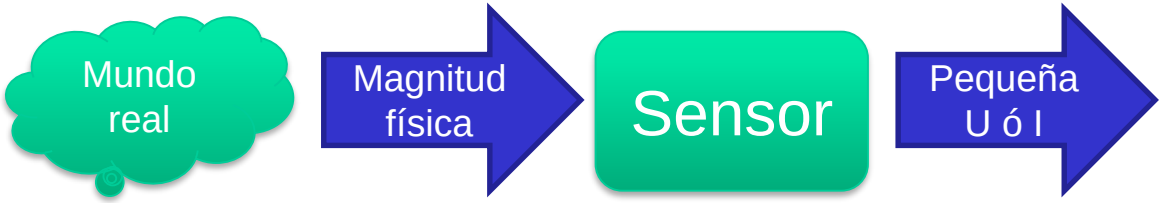


Interfaz con el “mundo real”

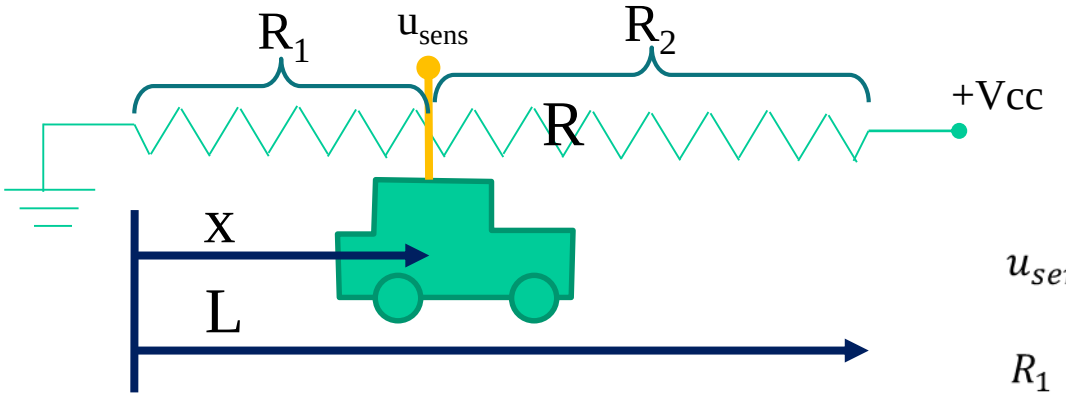
- Entradas del computador: diversos dispositivos electrónicos en función de la naturaleza del sensor y la información a extraer de él:
  - Conversor A/D (ADC) :
    - Para sensores que generan una tensión analógica
    - Proporcionan al programa una variable de tipo entero proporcional a la tensión a su entrada
  - Entrada digital (DI):
    - Para sensores que generan una tensión con únicamente 2 estados (0/1)
    - Proporcionan al programa un bit de una variable de tipo entero
  - Contador de pulsos (counter):
    - Para sensores que generan una tensión con únicamente 2 estados (0/1), de la que no interesa el estado 0 ó 1, sino el número de transiciones 0→1 y/o 1→0
    - Proporcionan al programa una variable de tipo entero con el valor de la cuenta

Obtención de valores analógicos

- Obtención de valores analógicos: conversores A/D
  - Sensor: genera una pequeña tensión o corriente directamente relacionada con una magnitud física



  - Ejemplo: potenciómetro para medida de desplazamiento



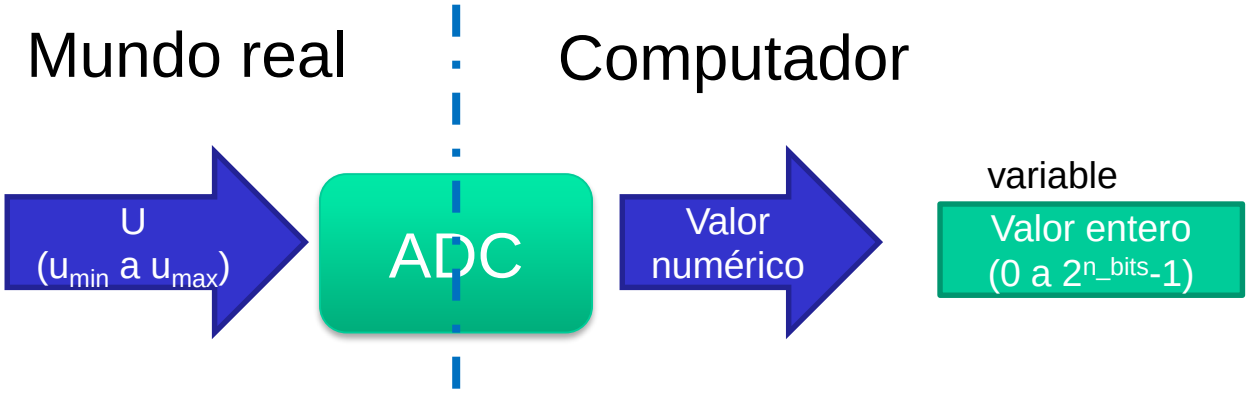
$$u_{sens} = v_{cc}\frac{R_1}{R}$$

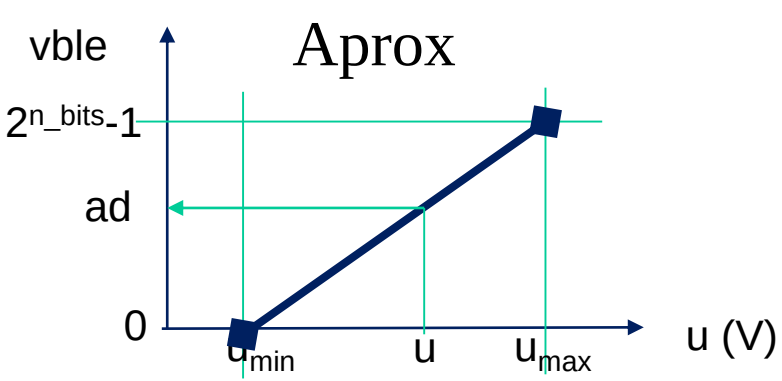
$$R_1 = R\frac{x}{L}$$

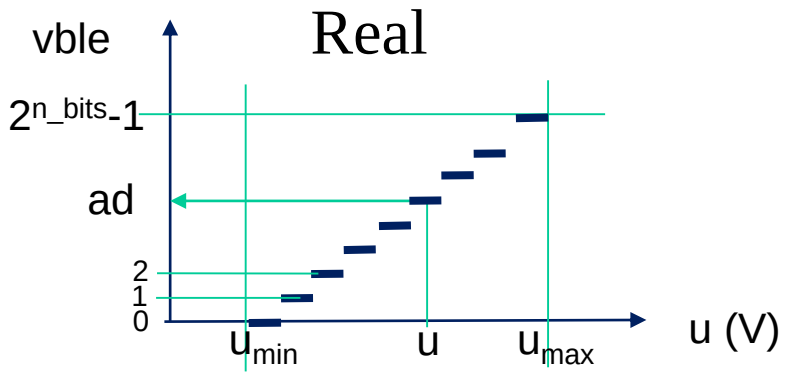
$$\Rightarrow u_{sens} = v_{cc}\frac{x}{L}$$

Obtención de valores analógicos

- Obtención de valores analógicos: conversor A/D
  - Conversor A/D: genera un valor numérico (variable de tipo **entero**) relacionada con la tensión a su entrada
  - Características: u_{min}, u_{max}, n_bits

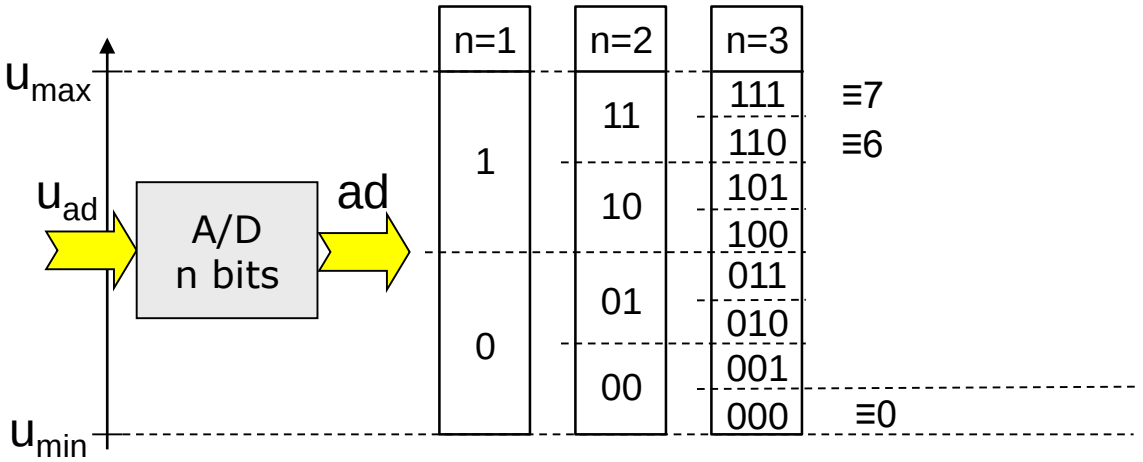






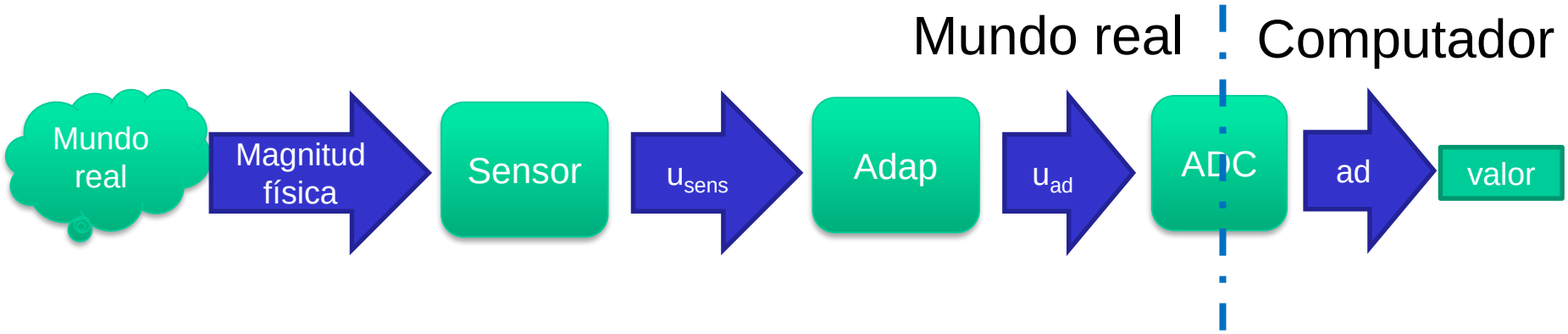
Obtención de valores analógicos

- 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 ($0 \ldots 2^{n-1}$).
    - Se realiza con electrónica mediante comparaciones sucesivas
    - El resultado de la conversión es un valor entero utilizable por el programa.
  - Valores típicos de n: 8, 10, 12, 16, 20, 24, 32.
  - Tipos: flash, aproximaciones sucesivas, etc (http://www.onmyphd.com/?p=analog.digital.converter)



Obtención de valores analógicos

- Si el rango de tensiones generado por el sensor no concuerda con el rango de tensiones del conversor:
  - Se precisa una electrónica de adaptación intermedia (Amplificadores Operacionales)

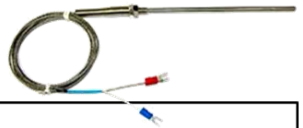


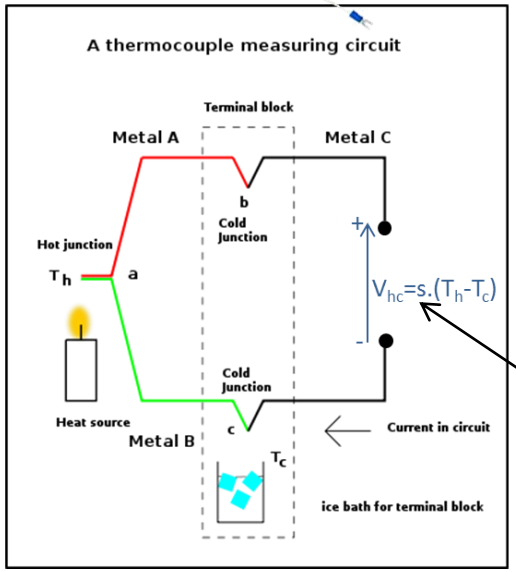
- El programa del computador sólo puede leer el valor del conversor, que está relacionado (**pero no es**) con la magnitud medida
  - El programa debe calcular la magnitud a partir de las especificaciones de la cadena de medida (sensor, adap, ADC)

Obtención de valores analógicos

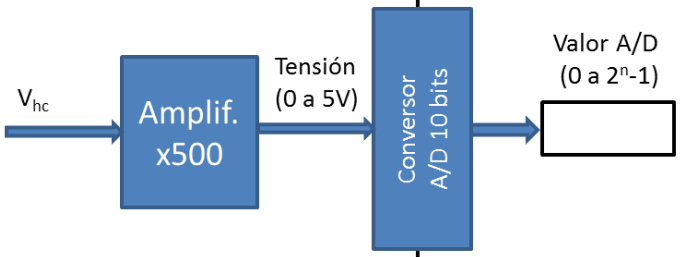
- Ejemplo: adquisición de Tª medida por termopar con conversor A/D de 10 bits y límites de tensión 0-5V

Termopar:





s=43 μV/ºC (termopar tipo T)



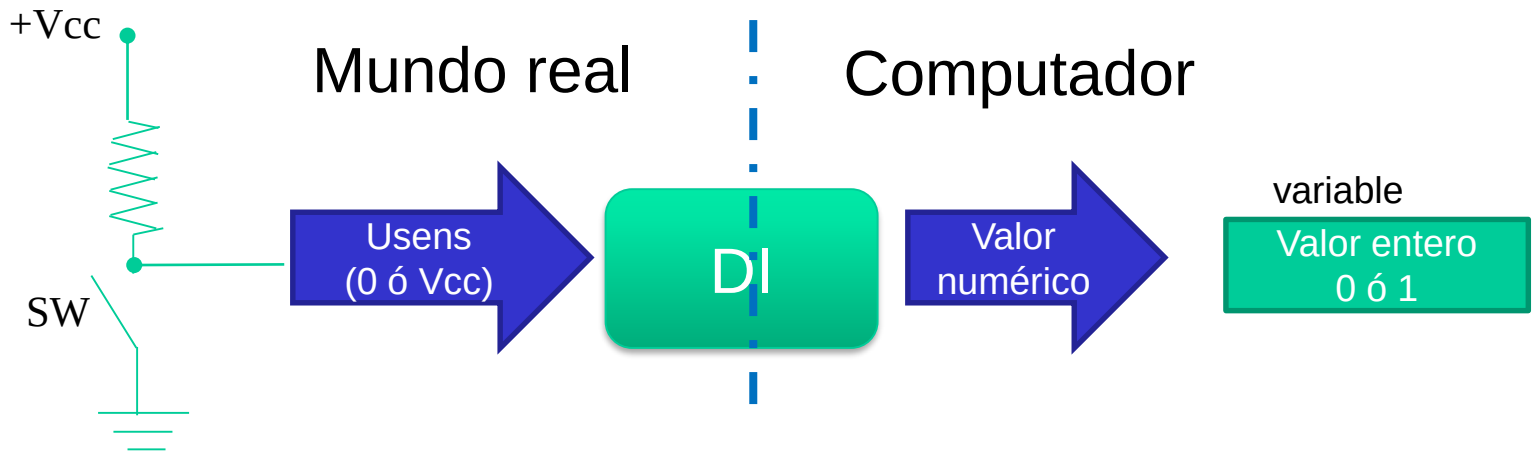
¡¡ Usar #define para todas las constantes !!

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;
Vhc_k=Vad_k/500.0f;
Th_k=Vhc_k/s-Tc;
```

Entradas digitales

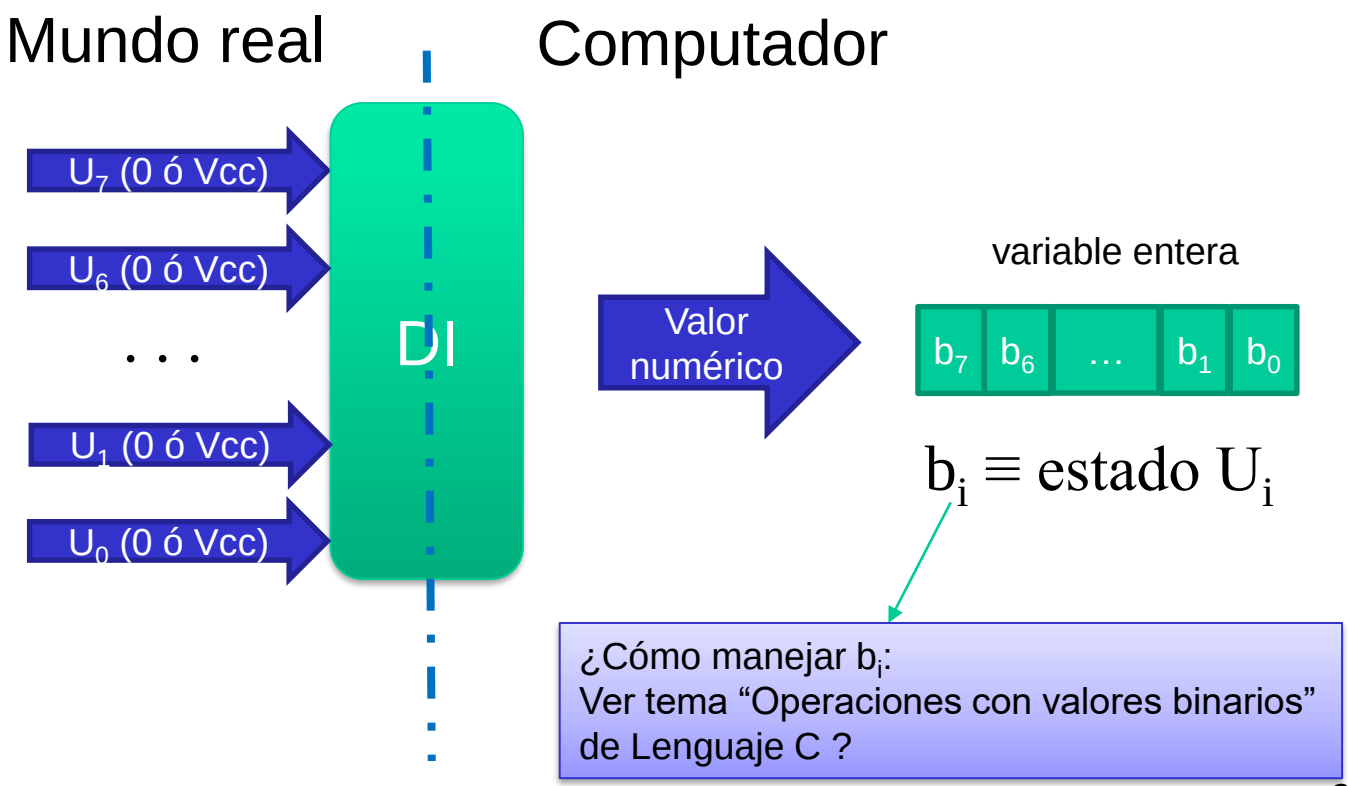
- Algunos sensores generan información en forma de todo/nada (true/false , 1/0), que serán capturadas mediante entradas digitales (DI) del computador
  - Ejemplo: interruptor (switch)



- El hardware electrónico es de la máxima sencillez

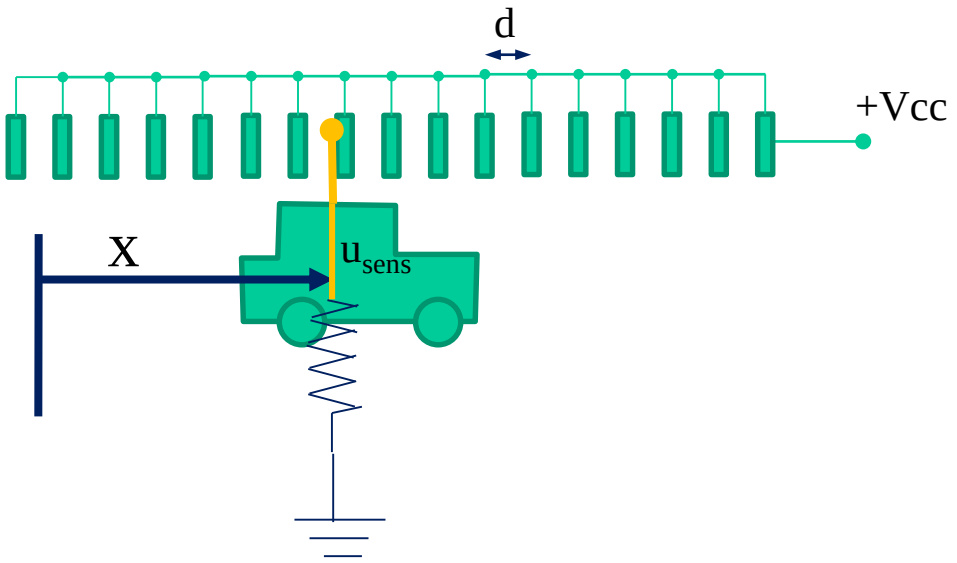
Entradas digitales

- Es habitual que varias entradas digitales se agrupen en bits diferentes de una sola variable
  - Ejemplo: interruptor (switch)



Entradas tipo contador

- Algunos sensores generan información en forma de todo/nada (true/false , 1/0), de la que no interesa el estado (1/0) sino la cuenta de transiciones (1→0 y/o 0 →1)
- Estos sensores serán capturados mediante entradas tipo contador
  - Ejemplo: encoder para medida de desplazamiento

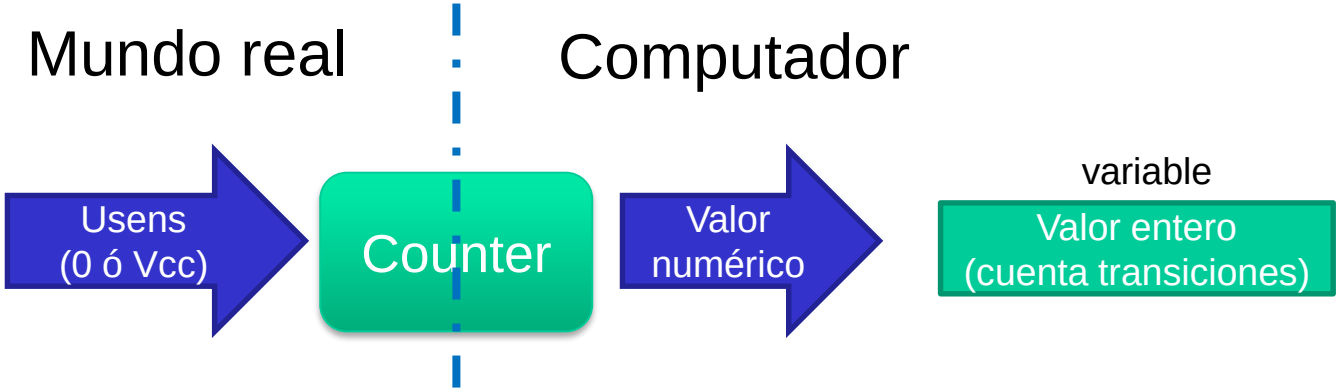


$$u_{sens} = \begin{cases} +Vcc \text{ si la escobilla toca conductor} \\ 0 \text{ si la escobilla no toca conductor} \end{cases}$$

x=d*(cuenta transiciones 0 →Vcc)

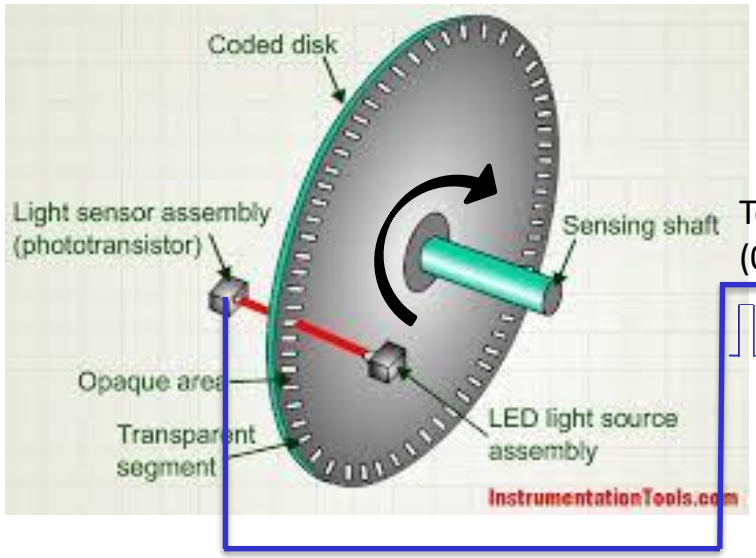
Entradas tipo contador

- Entrada contador:
  - 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.
  - Se necesita una entrada digital auxiliar para definir el 0 absoluto.
  - ¡¡ Ojo con desbordamientos del contador !!



Entradas tipo contador

- Ejemplo: entrada de pulsos de un encóder óptico:



Tensión
(0 a 5V)

Computador

Contador de pulsos

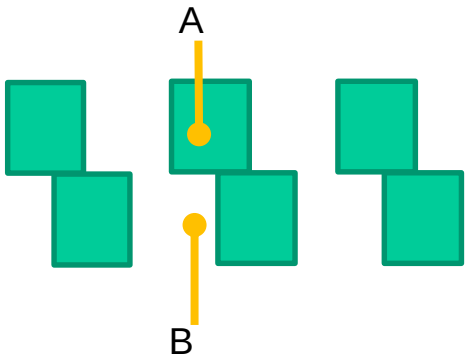
Contador

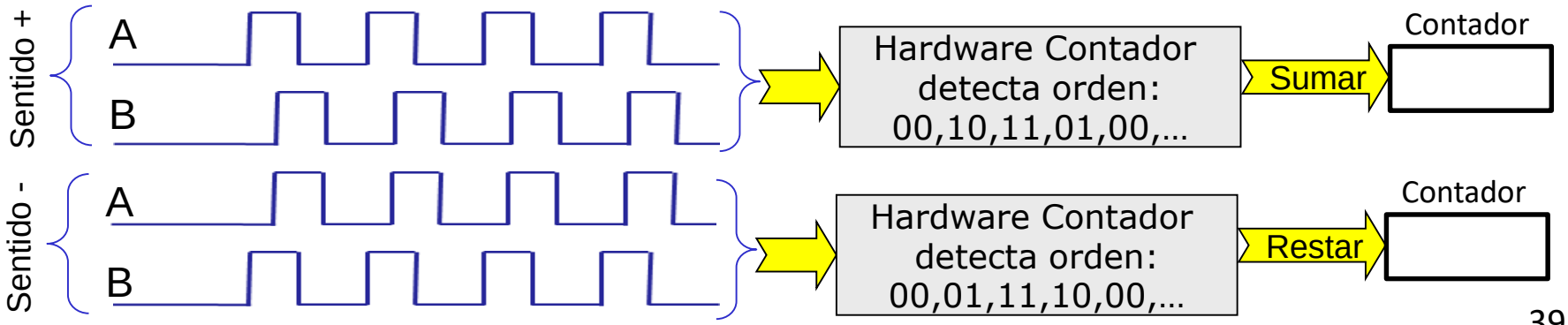
Programa:

```
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;
```

Entradas tipo contador

- Entrada de pulsos: algunas consideraciones
  - Necesidad de referencia “home” para medidas absolutas
    - El sensor sólo mide variación respecto a la posición original. Se necesita una referencia absoluta (mediante entrada digital), y mover posición a esa referencia al comienzo
  - Señal en cuadratura para detección de sentido
    - Con una sola señal no se puede distinguir el sentido del movimiento
    - Con dos señales en cuadratura se puede distinguir (habitualmente se integra en el hardware del contador)



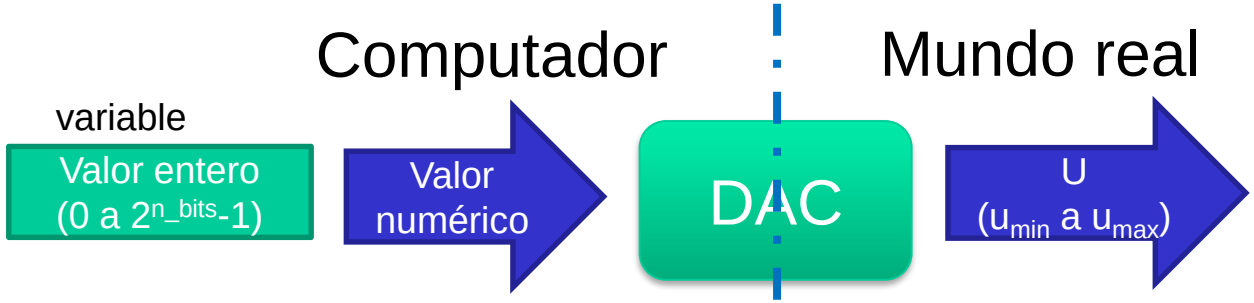


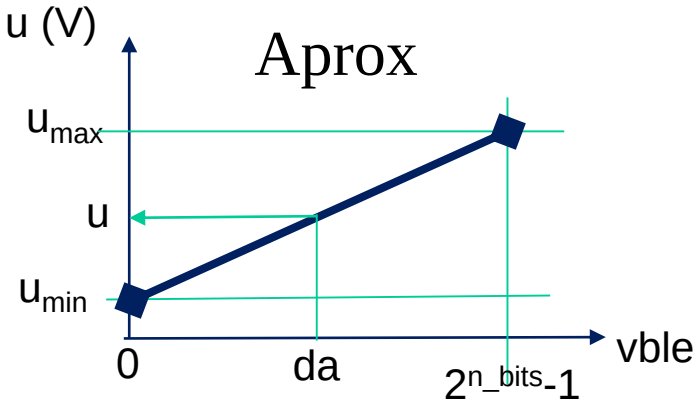
Interfaz con el “mundo real”

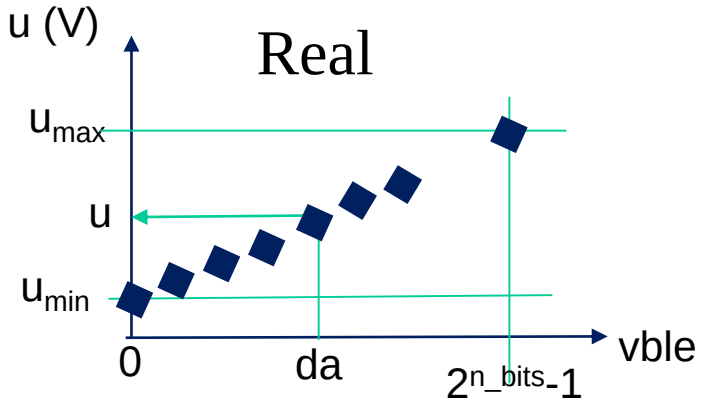
- Salidas del computador: diversos dispositivos electrónicos en función de la naturaleza de la acción a generar:
  - Conversor D/A (DAC) :
    - Generan una tensión analógica proporcional a un valor entero proporcionado por el programa
  - Salida digital (DO):
    - Generan una tensión con únicamente 2 estados (0/1 ≡ u_{min}/u_{max}) a partir de un bit de una variable de tipo entero proporcionado por el programa
  - Salida modulada en ancho de pulso (PWM):
    - Generan una tensión en forma de tren de pulsos 0/1 cuyo valor promedio proporcional a un valor entero proporcionado por el programa
- Para todos ellos, la salida es de poca potencia (mW):
  - Es necesario un accionador que obtenga la energía de una fuente externa al computador
  - El accionador es modulado por la salida del computador

Generación de valores analógicos

- Generación de valores analógicos: conversor D/A (DAC)
  - Conversor D/A: genera una tensión a su salida relacionada con el valor numérico (variable de tipo **entero**) que se aplica a su entrada
  - Características: u_{min}, u_{max}, n_bits

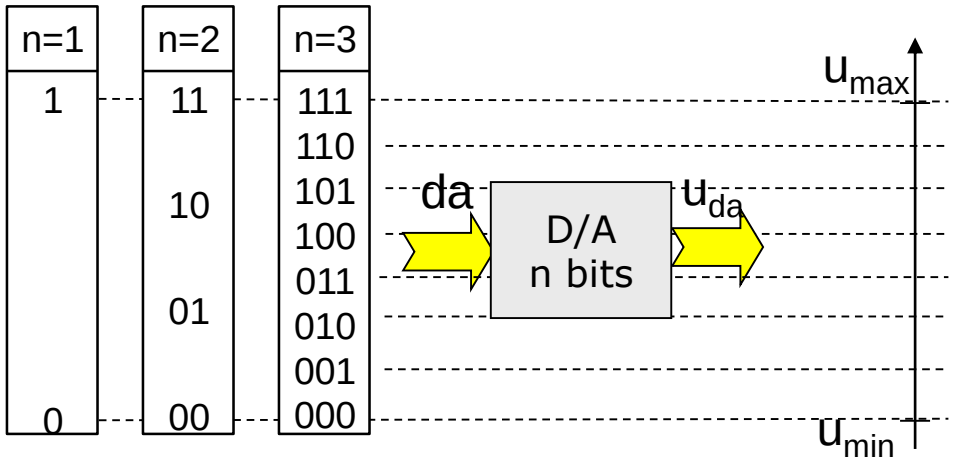






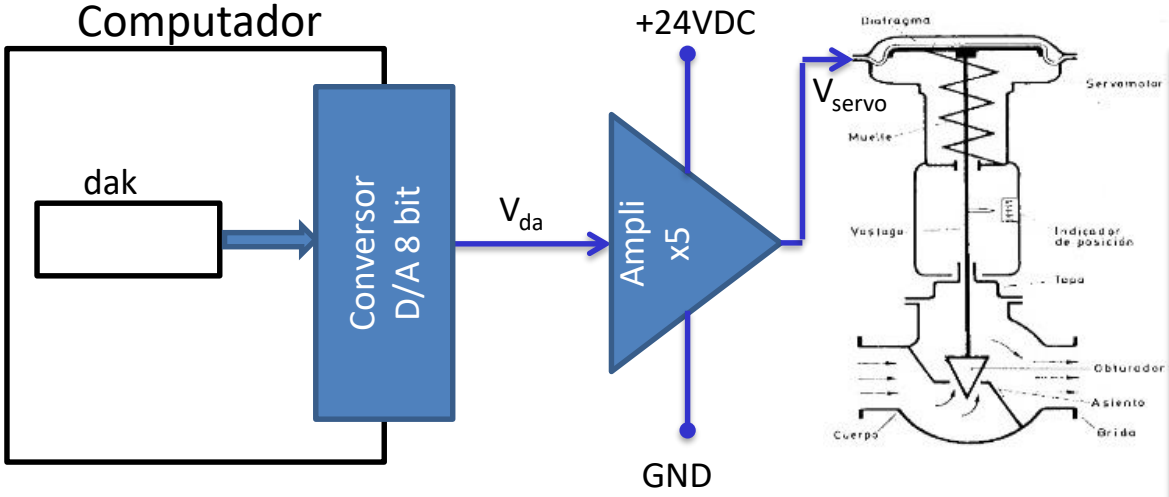
Generación de valores analógicos

- 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.
    - Se realiza con electrónica mediante adiciones sucesivas
  - Valores típicos de n: 8, 10, 12, 16, 20, 24, 32.
  - Tipos: oversampling, ladder, … (http://www.onmyphd.com/?p=digital.analog.converter)



Generación de valores analógicos

- Conversor D/A
  - La salida es una pequeña tensión con baja potencia (mW) que puede servir para modular energía externa de un accionador analógico
- Ejemplo: accionamiento de una electroválvula proporcional de 24VDC, mediante conversor DA de 8 bits y límites de tensión 0-5V

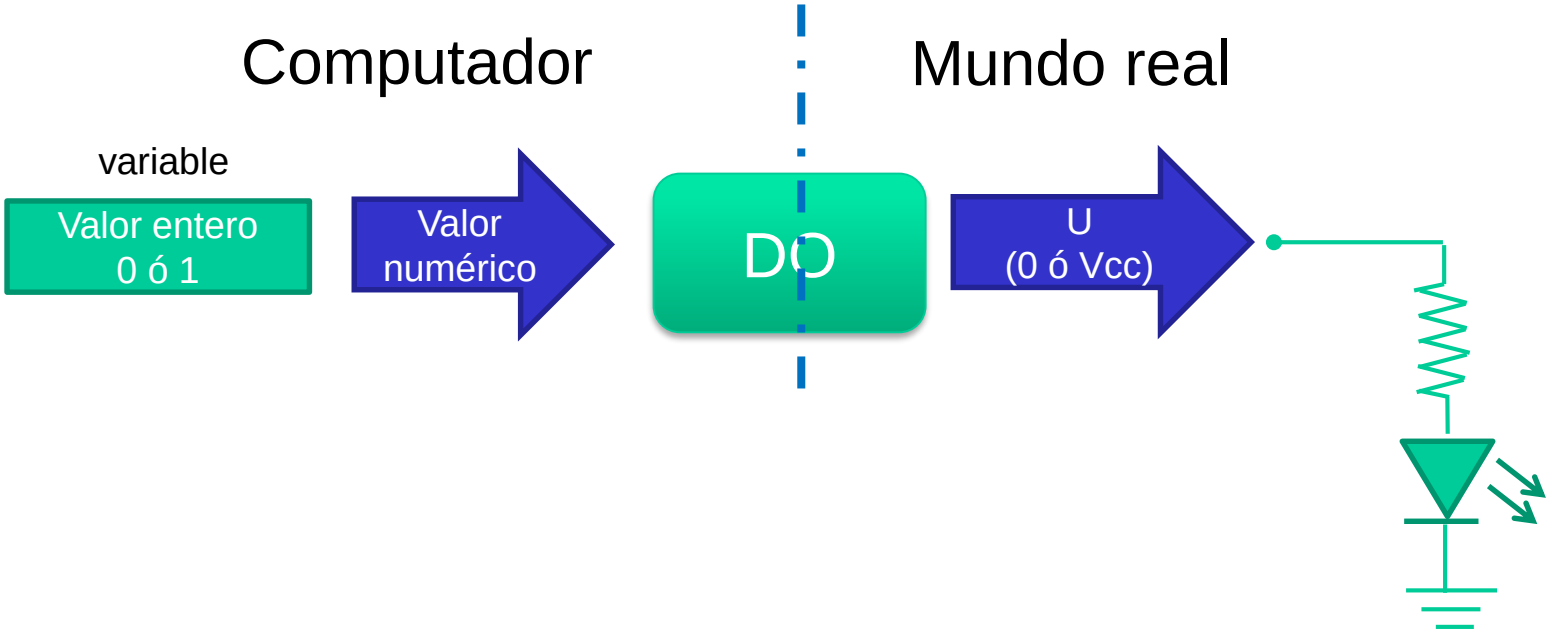


Programa:

```
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);
```

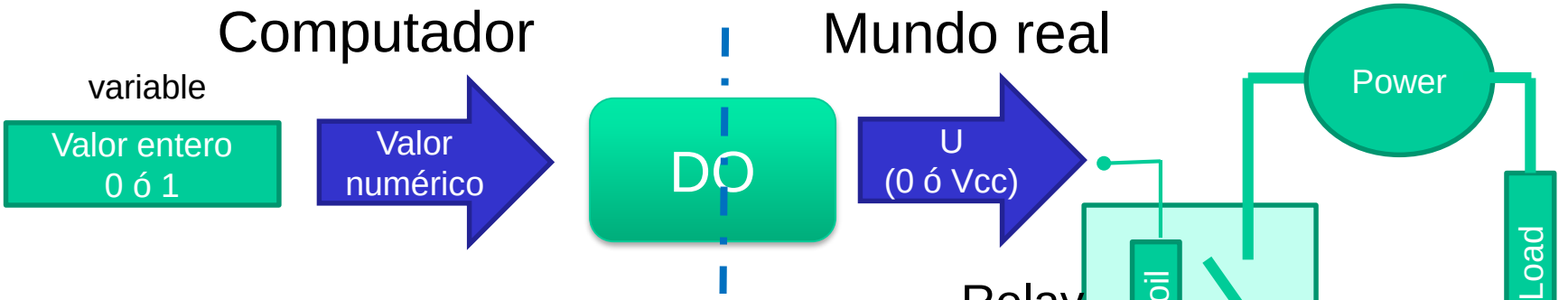
Salidas digitales

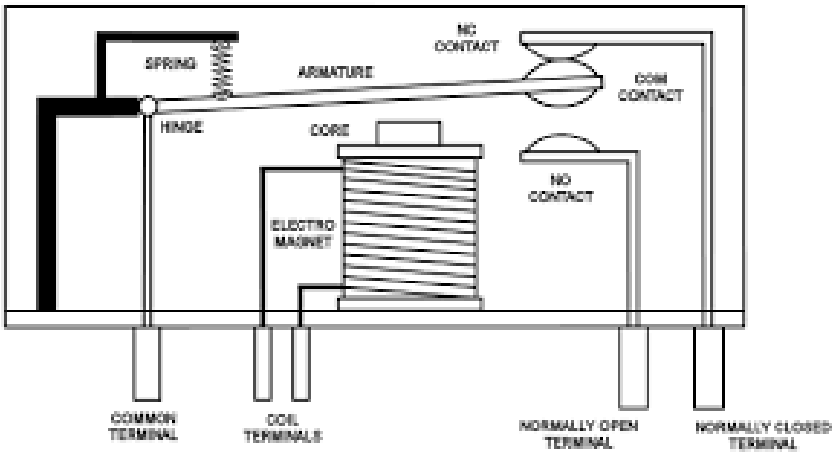
- Algunos accionadores sólo tienen dos acciones posibles: todo/nada (true/false , 1/0)
- Para ellos se utilizan salidas digitales (DO)
  - Ejemplo: lámpara LED (mW)



Salidas digitales

- Si es necesaria mayor potencia, lo usual es utilizar un relé (relay):
  - Electromagnético (bobina+interruptor)
  - De estado sólido (amplificador electrónico)

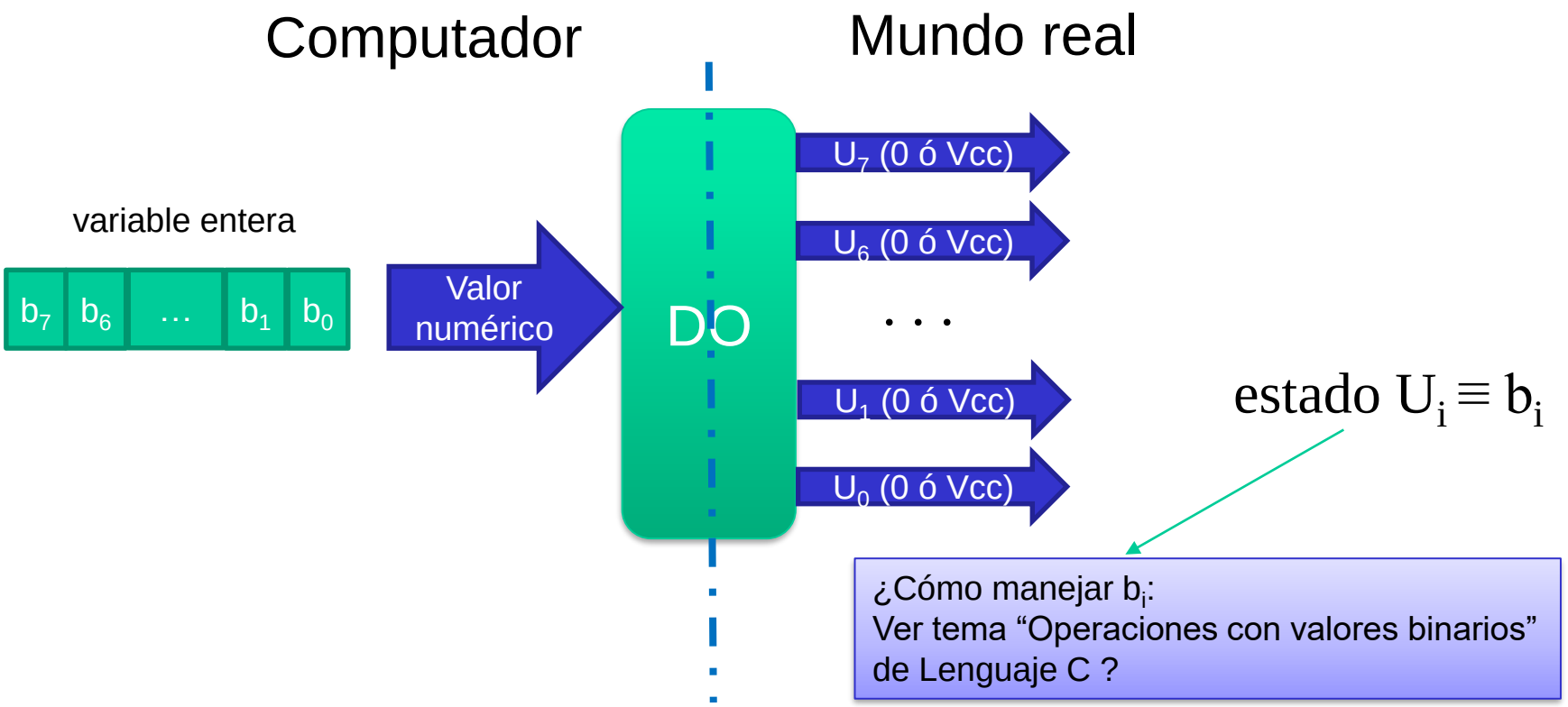




Circuito bobina: baja potencia
Circuito potencia: según necesidades

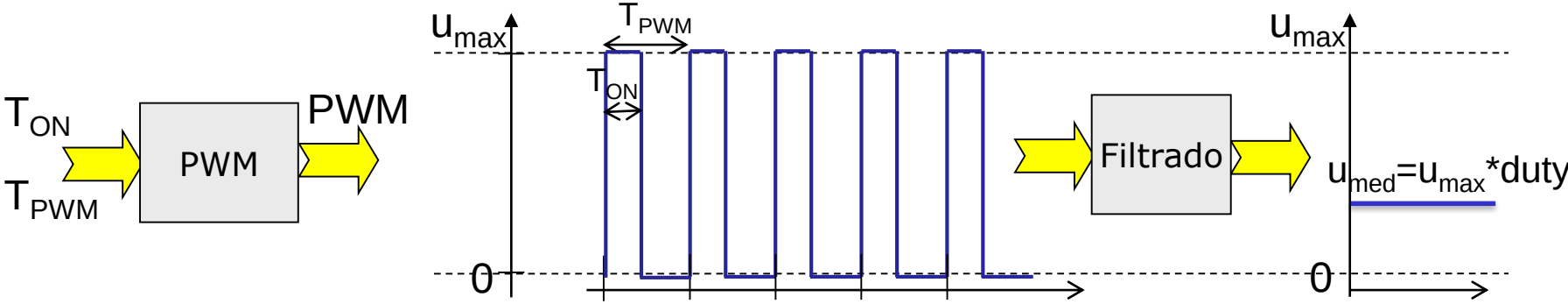
Salidas digitales

- Es habitual que varias salidas digitales se agrupen en bits diferentes de una sola variable



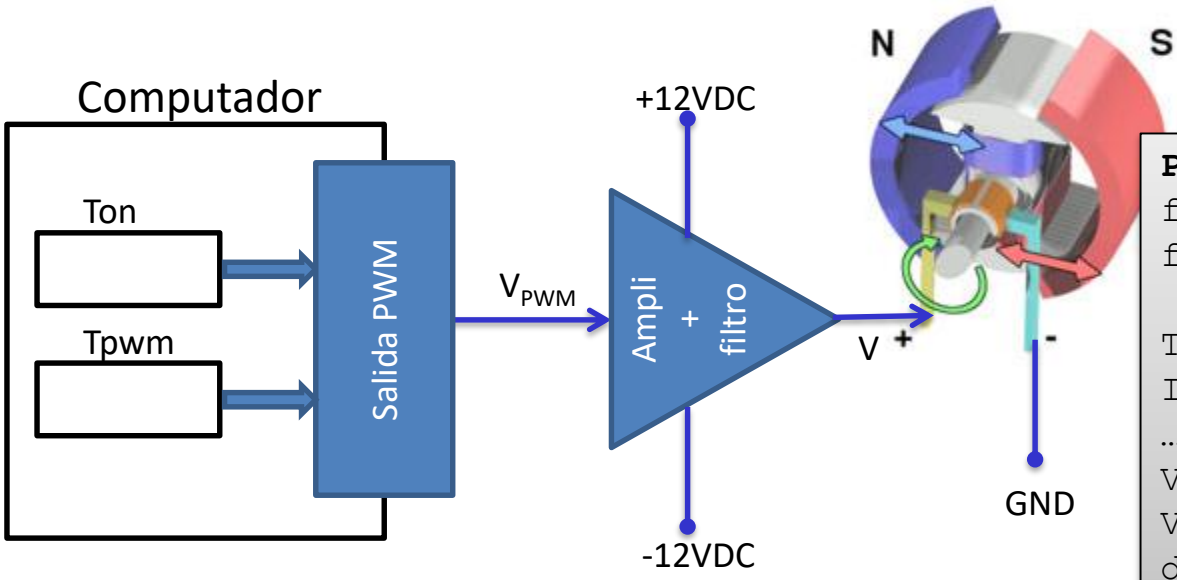
Salidas PWM

- PWM: Pulse Width Modulated
- Funcionamiento del generador 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.
  - En muchas ocasiones actúa como filtro 'natural' la propia dinámica del sistema sobre el que se aplica la onda
  - T_{PWM} suele ser fijo y función del filtro utilizado.



Salidas PWM

- Ejemplo: accionamiento de un motor DC de 12V mediante un generador PWM de 5V y un amplificador.



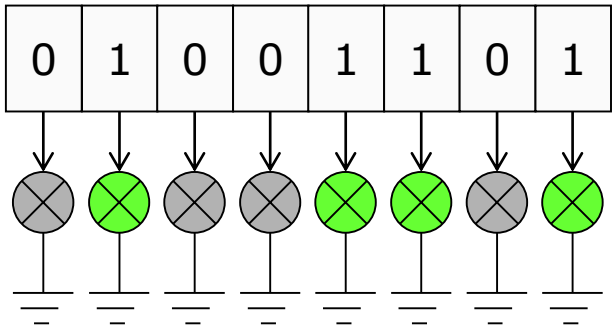
Programa:

```
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);
```

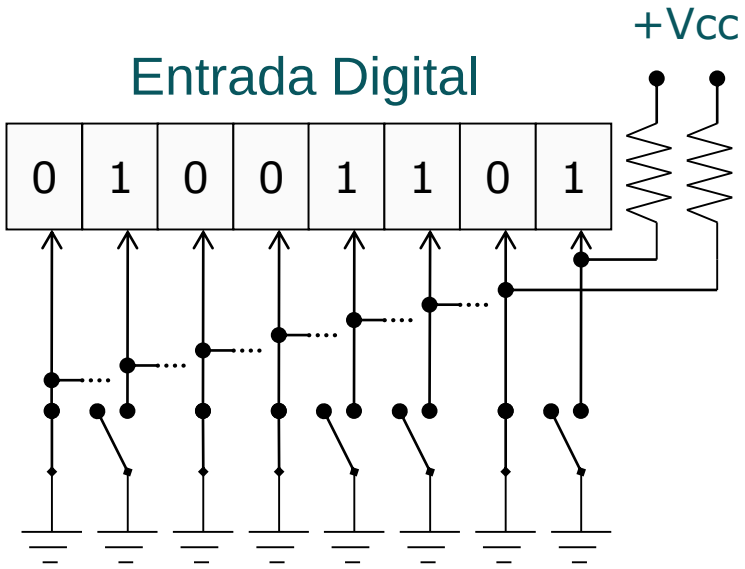
Programación de E/S digital

- 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:

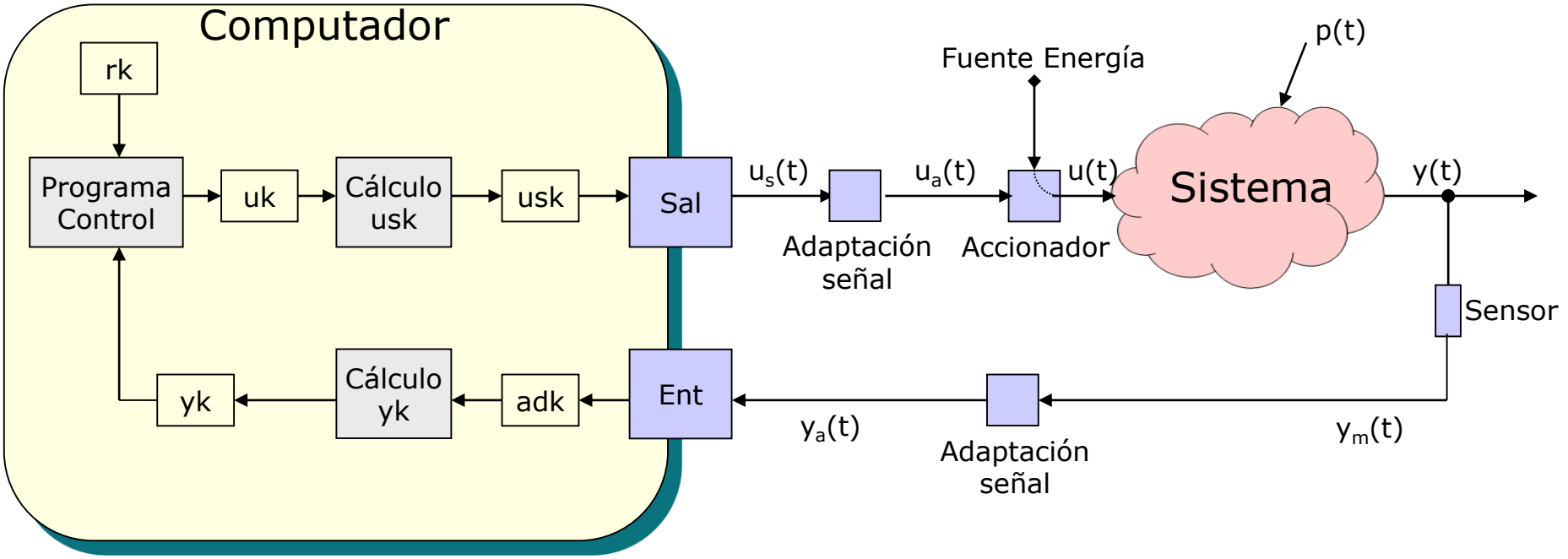
```
x=x ^ mascara;
```

Indice

- Introducción control de procesos por computador
- Conceptos básicos para programación de control
- Interfaz del computador con el exterior
- **Las matemáticas del control**
- Programación del lazo de control
- Programación en lenguaje C
- Implantación del control en el computador
- El control secuencial

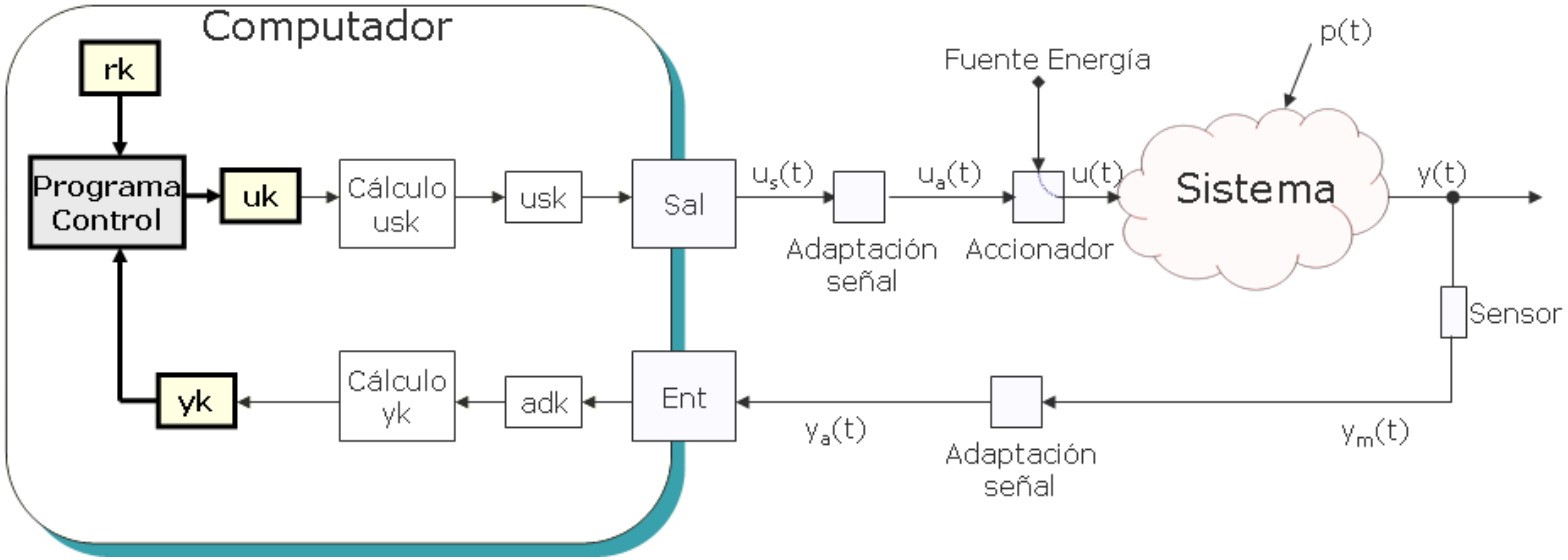
El lazo de control

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



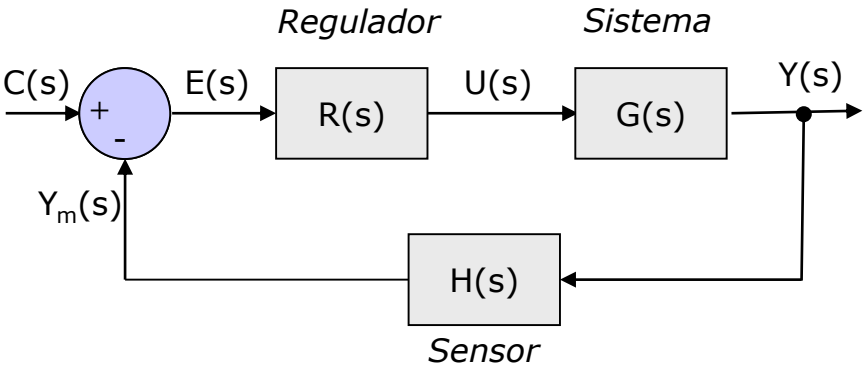
Las matemáticas del control

- 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

- En las asignaturas de control se estudia la acción de control a aplicar de forma matemática mediante tranformadas de Laplace :



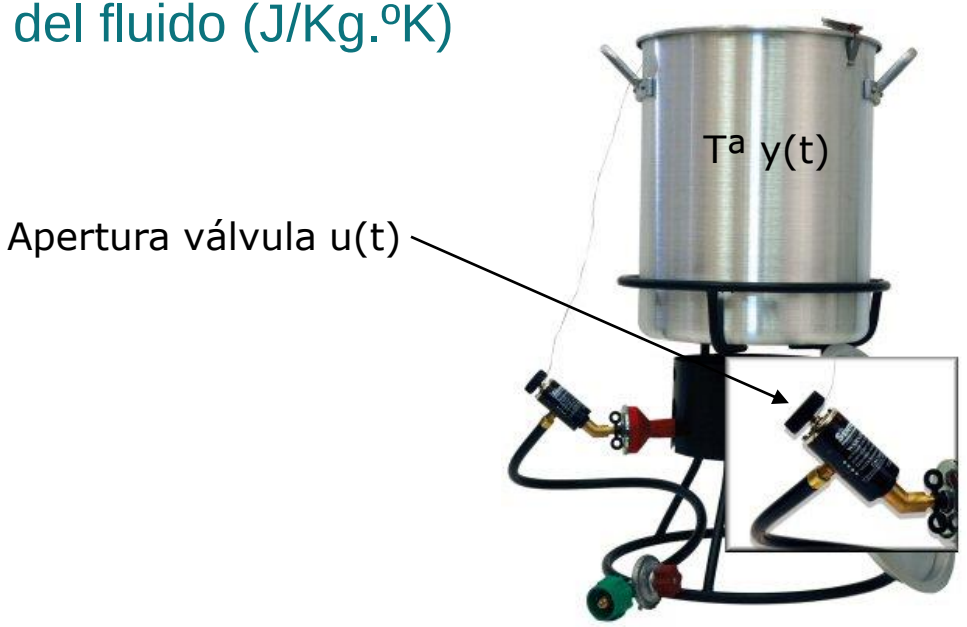
- Pero en el control por computador:
  - Sólo pueden realizar cálculos cada Tm (sistema discreto), por lo que no es aplicable la transformada de Laplace (requiere sistema continuo)
  - En el sistema real aparecen muchos más elementos que en el lazo de control “matemático”

El lazo de control

- Una aproximación intuitiva a los reguladores programados:
  - Mediante ejemplos se verán las diferentes acciones P-I-D y la forma de programarlas en el computador
  - Se llega a un formato general de cálculo para cualquier tipo de regulador
  - Por último, se relacionan los conocimientos previos de teoría de control analógica con el cálculo a realizar

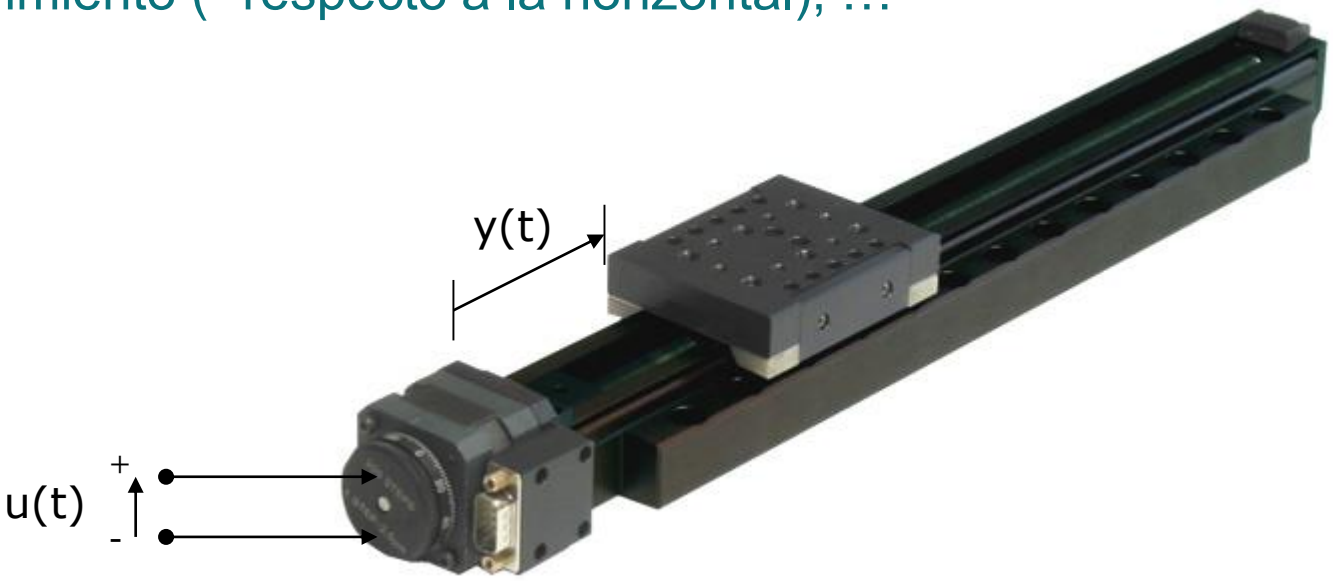
Las matemáticas del control

- 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)



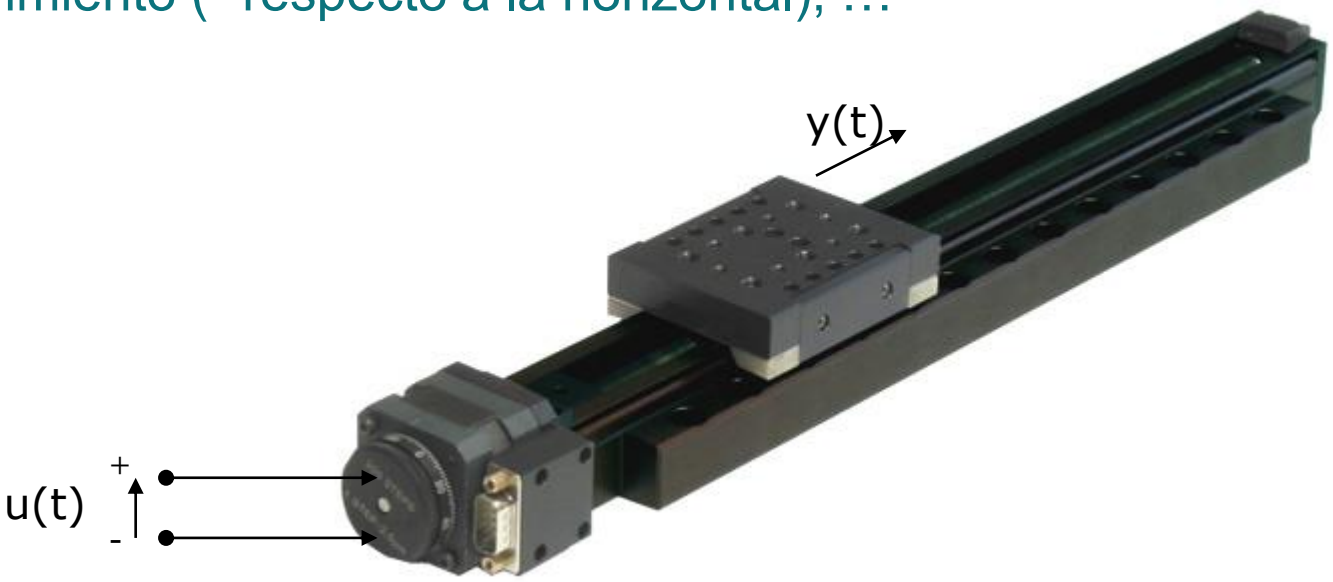
Las matemáticas del control

- 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

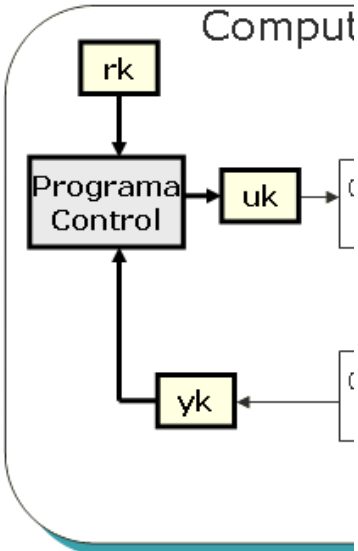
- Un cálculo sencillo: control todo/nada

  `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.



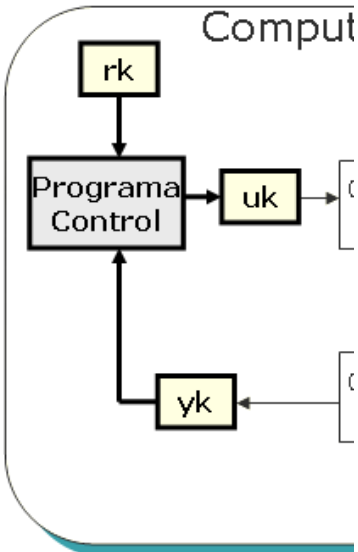
Ejemplo 1:
- Se abre la válvula de gas (100%) si la Tª es menor que la referencia.
- Se cierra la válvula (0%) si la Tª es mayor o igual que la referencia.
- La apertura es la misma para error=0.1ºC ó error=100ºC
- Excesivas conmutaciones cuando yk ≈ rk (se suele usar histéresis)

Las matemáticas del control

- Un cálculo sencillo: control proporcional

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

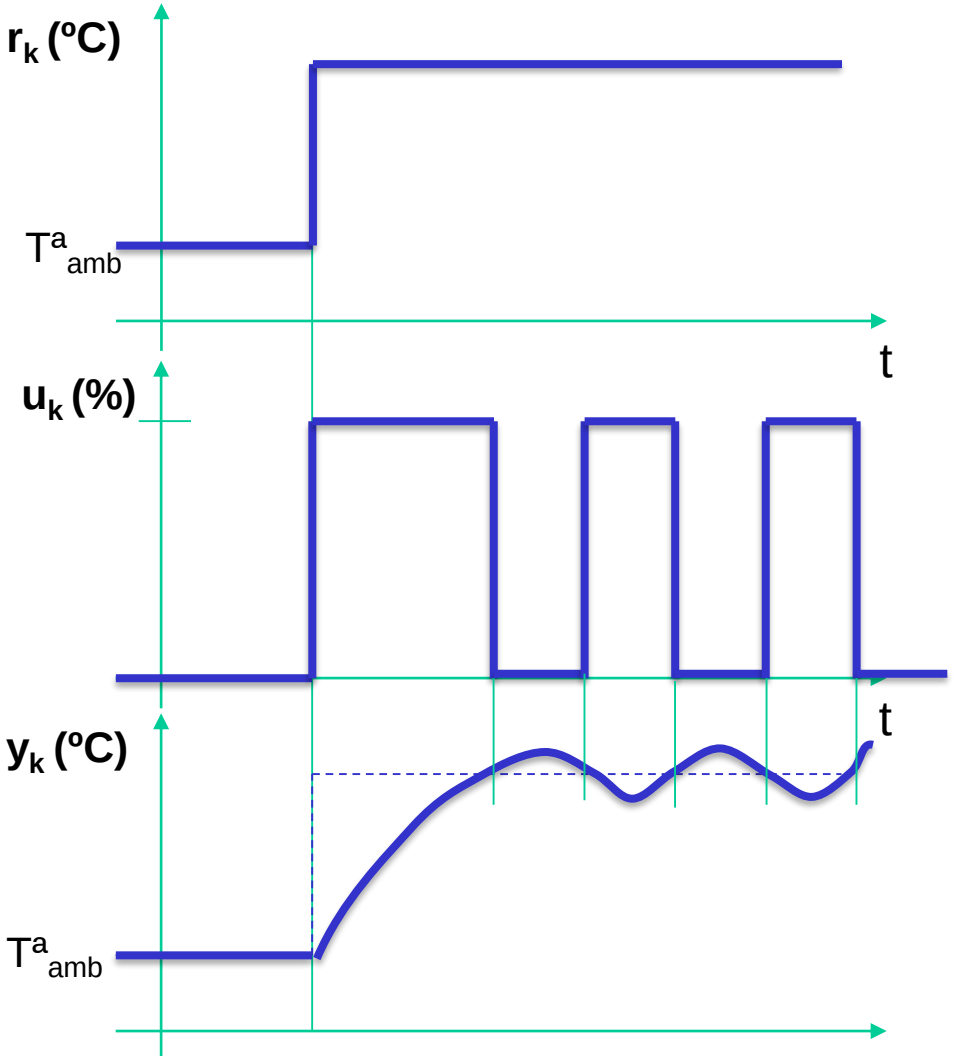
  - Cuando el error es 0, la acción de control es 0.
  - La acción de control es mayor (en valor absoluto) cuanto más grande sea el error (en valor absoluto), y tiene el mismo signo que el error.
  - El cálculo es muy simple



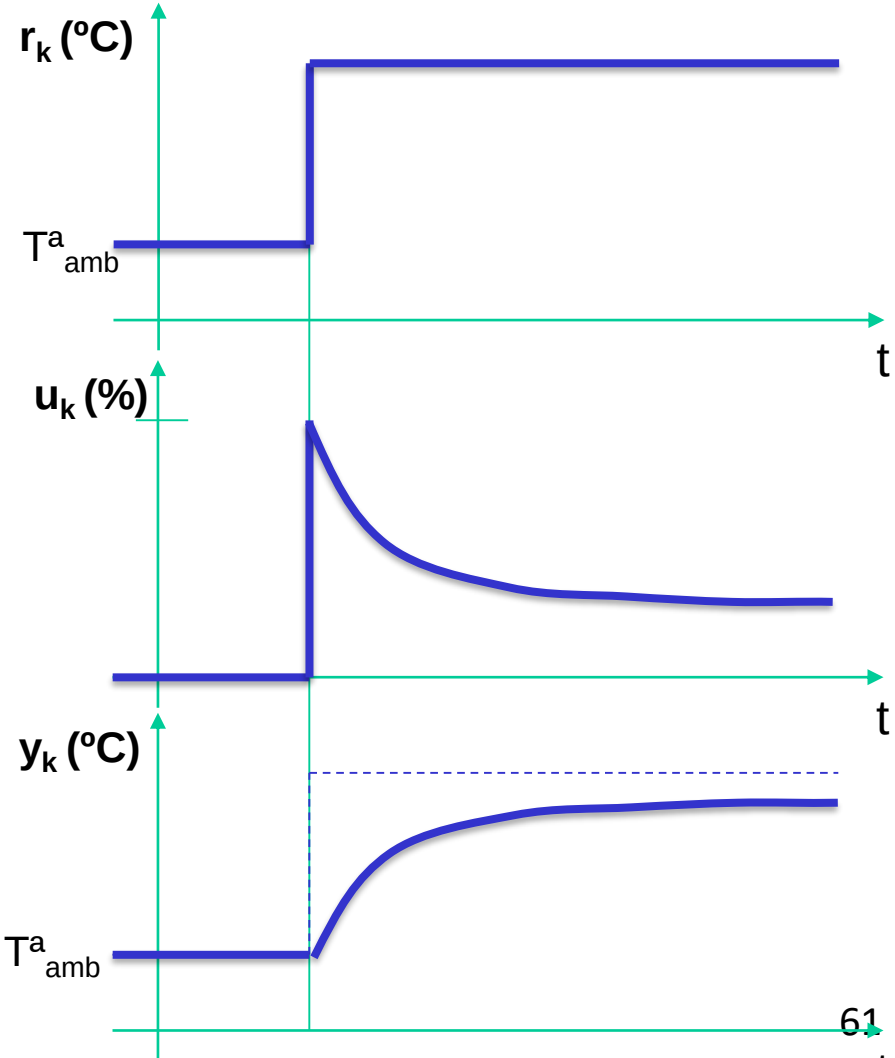
Ejemplo 1:
- Se abre más la válvula de gas cuanto mayor sea el error.
- Se cierra la válvula (0%) si la Tª iguala a la referencia.
- No se tiene en cuenta si la Tª estaba subiendo o bajando previamente.

Las matemáticas del control

Control Todo/Nada



Control P

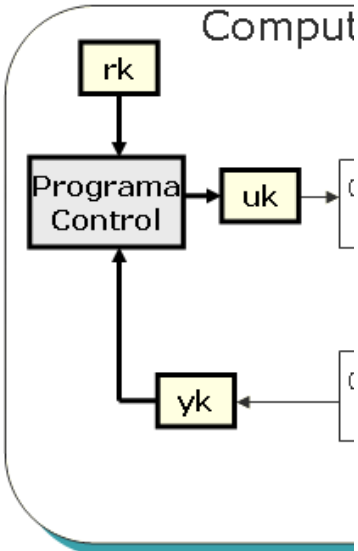


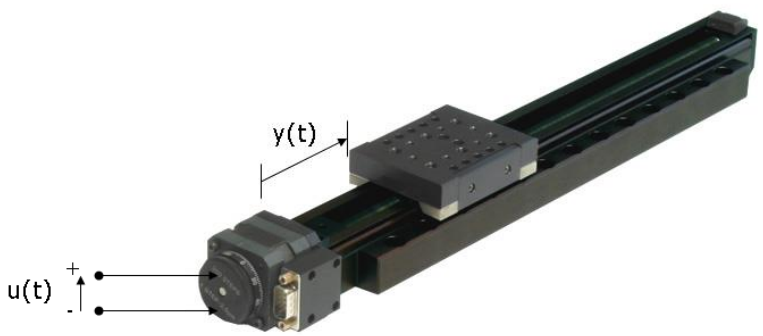
Las matemáticas del control

- 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.





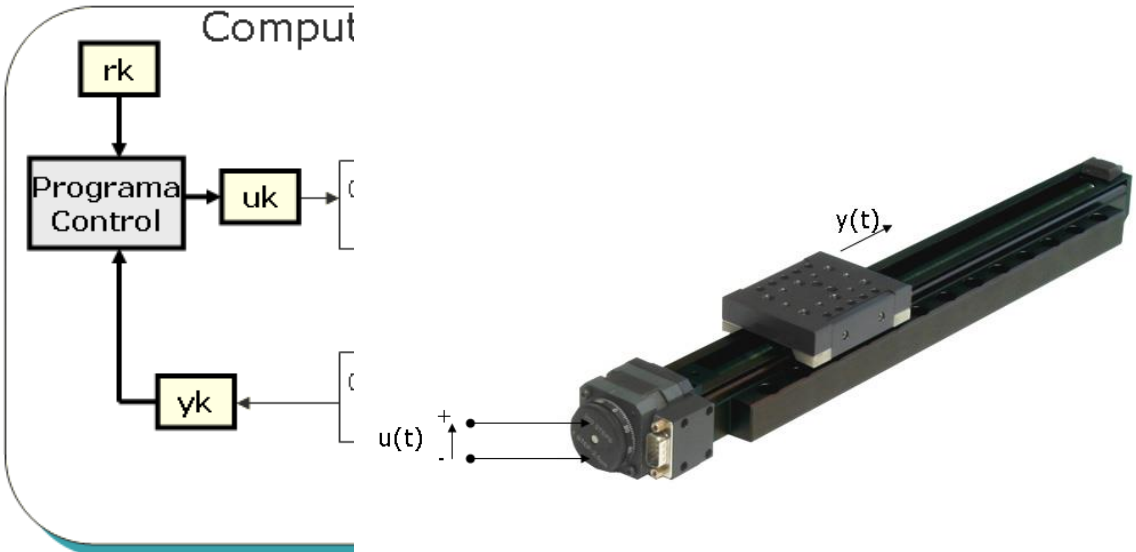
Ejemplo 2:
- Se da más tensión al motor cuanto mayor sea la distancia al destino.
- Se para el motor (tensión 0) cuando alcanza el destino (error 0).
- No se tiene en cuenta si el móvil se está acercando o alejando del destino.

Las matemáticas del control

- Un cálculo sencillo: control proporcional

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

  - Cuando el error es 0, la acción de control es 0.
  - La acción de control 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.



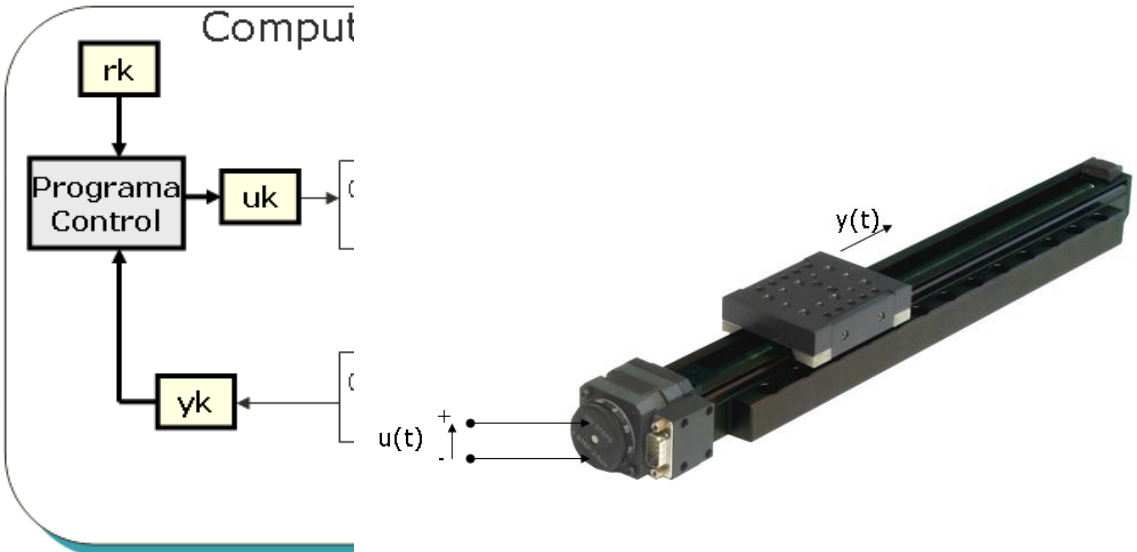
Ejemplo 3:
- Se aumenta la tensión del motor si la velocidad es menor que la deseada
- Si la velocidad fuese igual a la deseada, la tensión aplicada sería 0V, por tanto el motor tendería a pararse.
- Es necesario mantener un error para que el sistema se estabilice.

Las matemáticas del control

- Un cálculo sencillo: control proporcional-integral

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

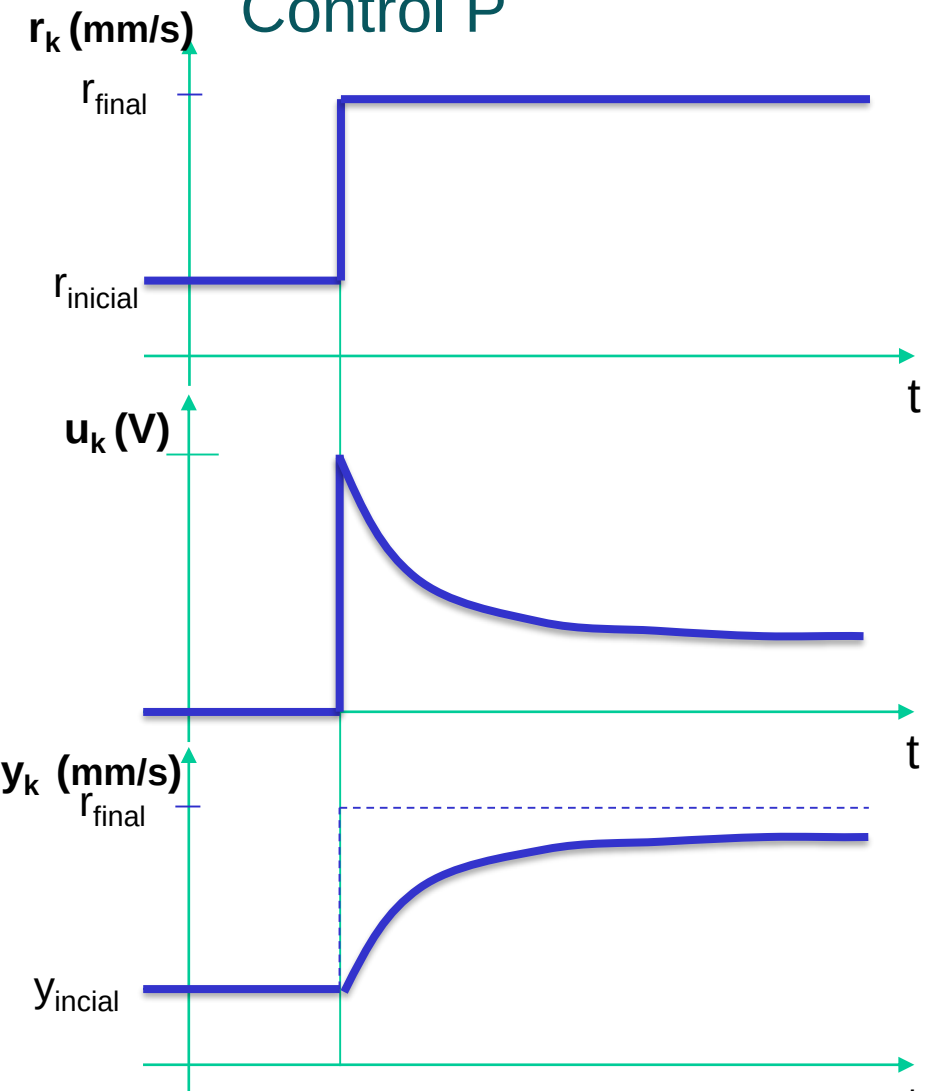
  - Cuando el error es 0, la acción de control se mantiene.
  - La acción de control se incrementa (en valor absoluto) tanto más cuanto más grande sea el error (en valor absoluto), y el incremento tiene el mismo signo que el error.
  - El cálculo necesita recordar un valor anterior de uk.



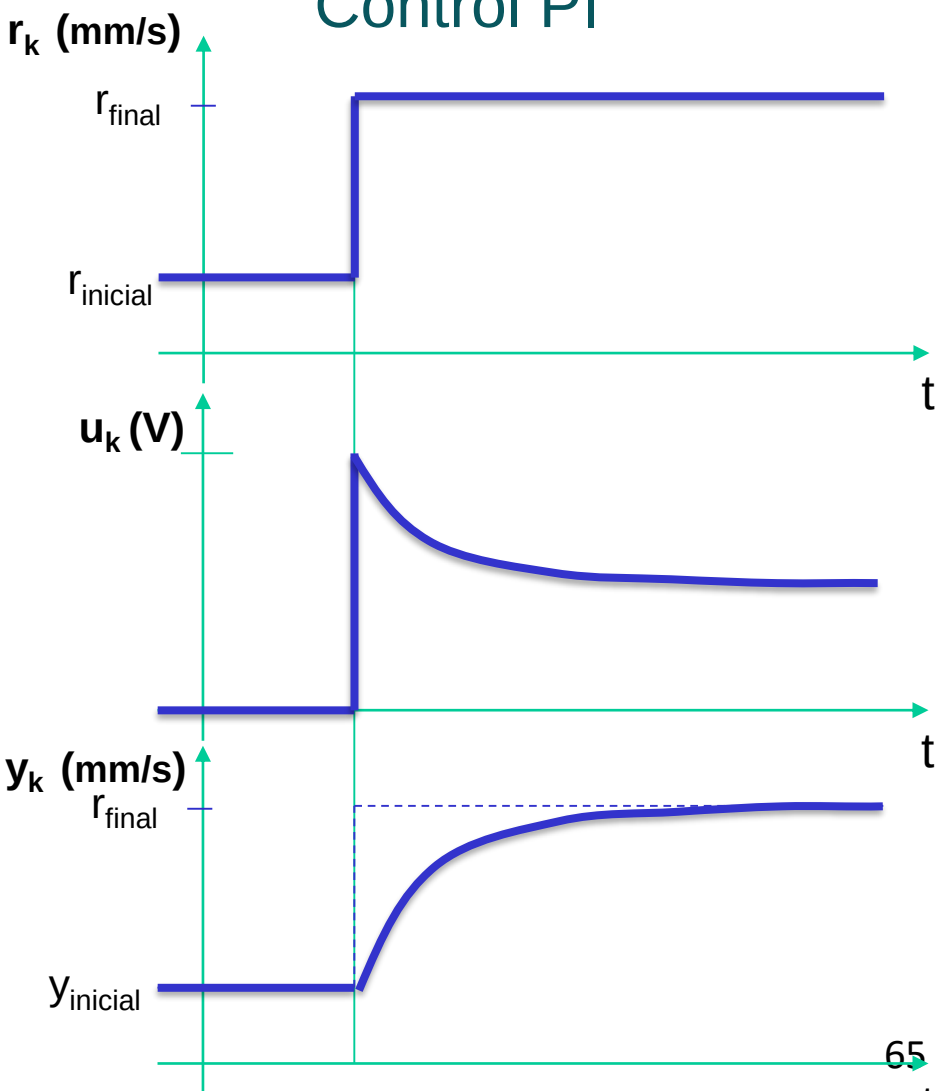
Ejemplo 3:
- Se aumenta la tensión del motor si la velocidad es menor que la deseada
- Si la velocidad es igual a la deseada, se mantiene la tensión aplicada.
- El error llega a 0.

Las matemáticas del control

Control P



Control PI

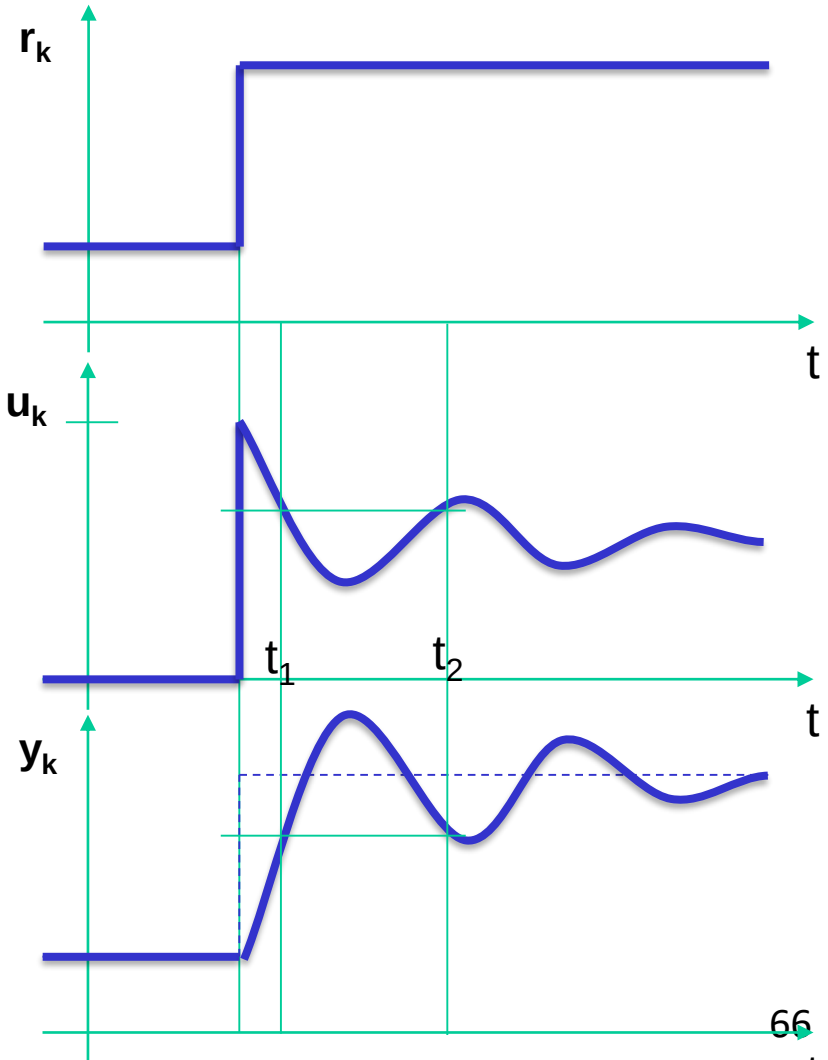


Las matemáticas del control

Control PI:

- En $t=t_1$ y $t=t_2$, la acción de control es la misma (mismo e_k), aunque las situaciones son distintas:
  - En $t=t_1$ el error está disminuyendo
  - En $t=t_2$ el error está aumentando

- Solución: Añadir acción dependiente de la derivada del error

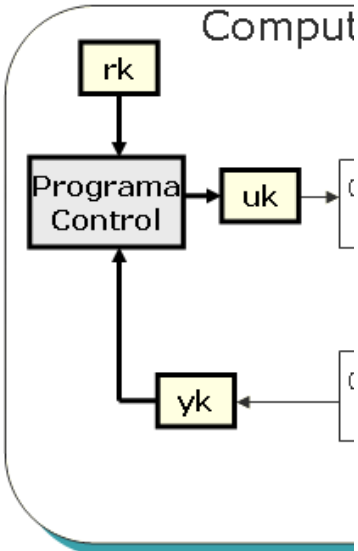


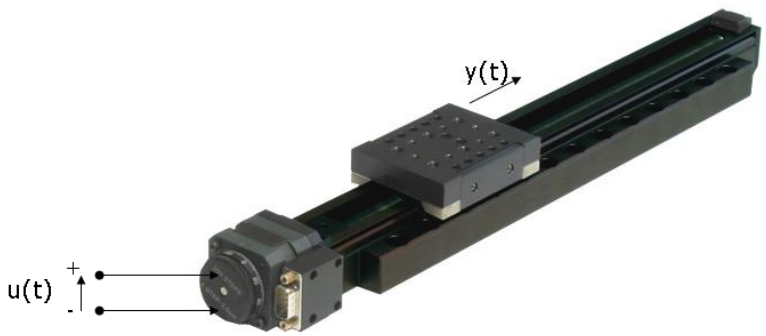
Las matemáticas del control

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

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

  - Cuando el error se mantiene en 0, la salida se mantiene.
  - La acción de control 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.

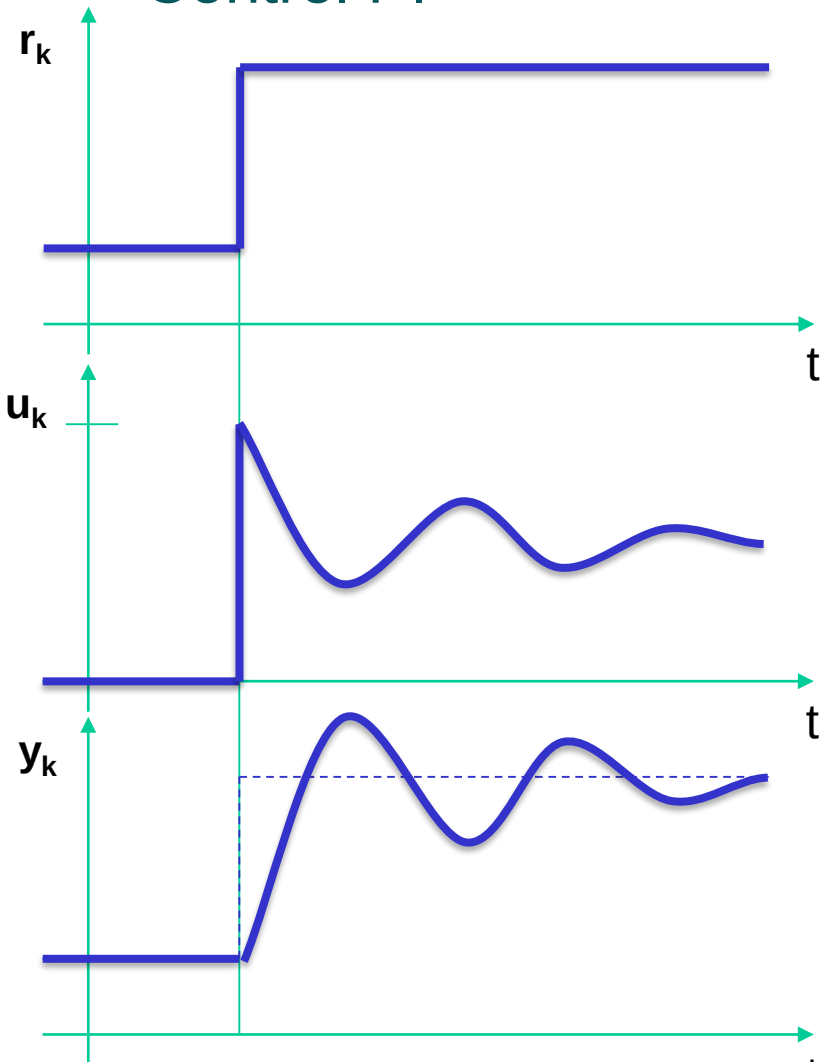




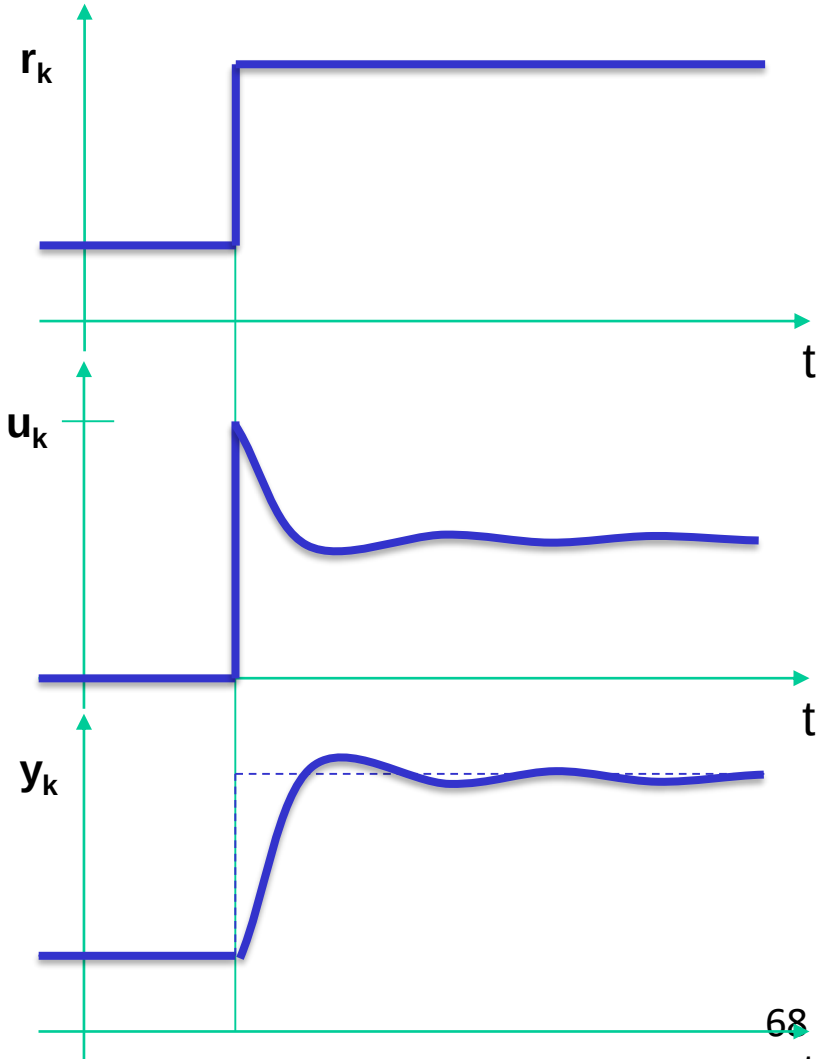
Ejemplo 3:
- Se aumenta la tensión del motor si la velocidad es menor que la deseada
- Si la velocidad es igual a la deseada, se mantiene la tensión aplicada.
- El error llega a 0 más rápidamente, al tenerse en cuenta su derivada.

Las matemáticas del control

Control PI



Control PID



Las matemáticas del control

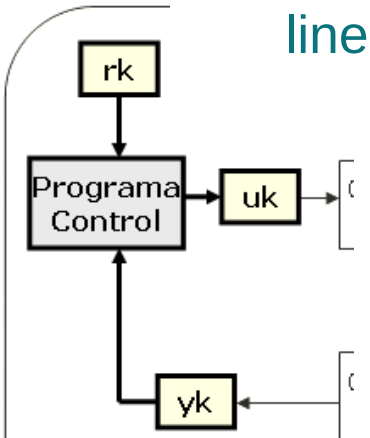
- Cálculo general:

$e_k = r_k - y_k$

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

  - 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 es 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:

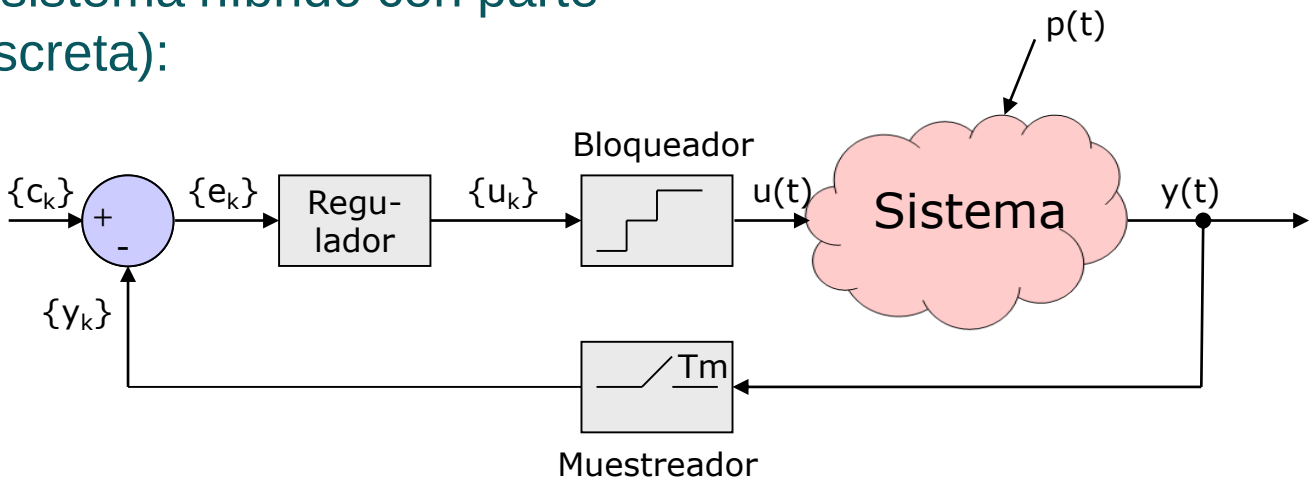
$$u_k = b_0 \cdot e_k + b_1 \cdot e_{k-1} + \ldots + b_m \cdot e_{k-m} - a_1 \cdot u_{k-1} - a_2 \cdot u_{k-2} - \ldots - a_n \cdot u_{k-n}$$



- 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 (sistema híbrido con parte continua y discreta):



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

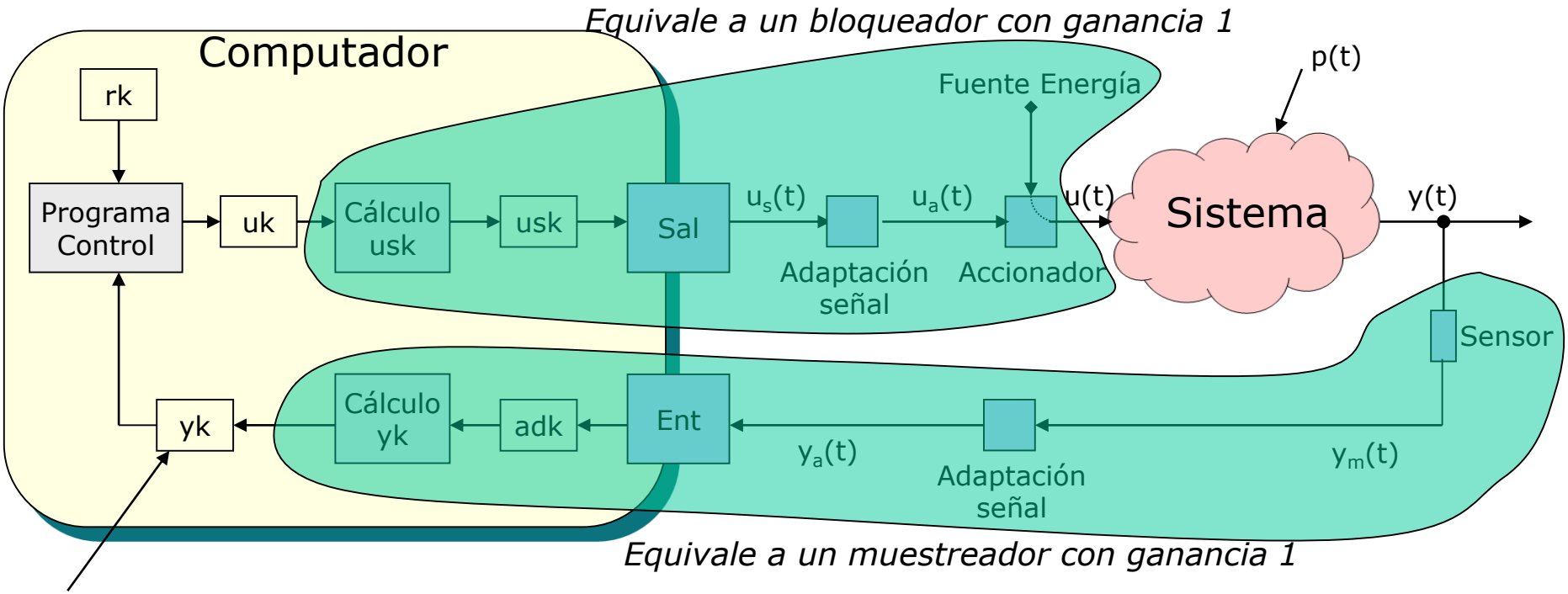
$$X(s) = \mathcal{L}[x(t)] = \int_{\tau=0}^{t} x(\tau).e^{-s\tau}.d\tau$$

  - Calculando la Transformada en Z de las señales discretas

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

Las matemáticas del control

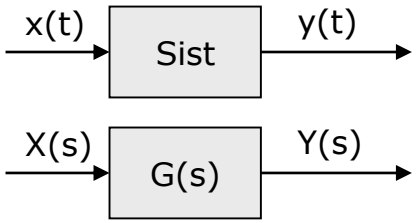
- Simplificación del lazo de control para los cálculos:



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

Las matemáticas del control

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



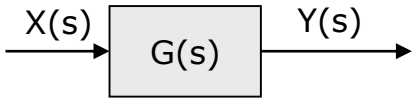
$$G(s) = \frac{Y(s)}{X(s)} = \frac{b_o.s^m + b_1.s^{m-1} + ... + b_{m-1}.s + b_m}{s^n + a_1.s^{n-1} + ... + a_{n-1}.s + a_n} \quad , n >= m$$

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

- Ecuación diferencial del sistema

Las matemáticas del control

- Transformada de Laplace, interpretación básica:



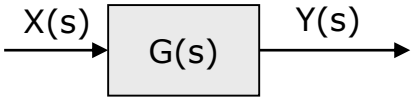
  - G(s)=Y(s)/X(s) para sistemas lineales dinámicos continuos.
  - G(s) es un cociente de polinomios, grado den ≥ grado num

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

  - 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, interpretación:

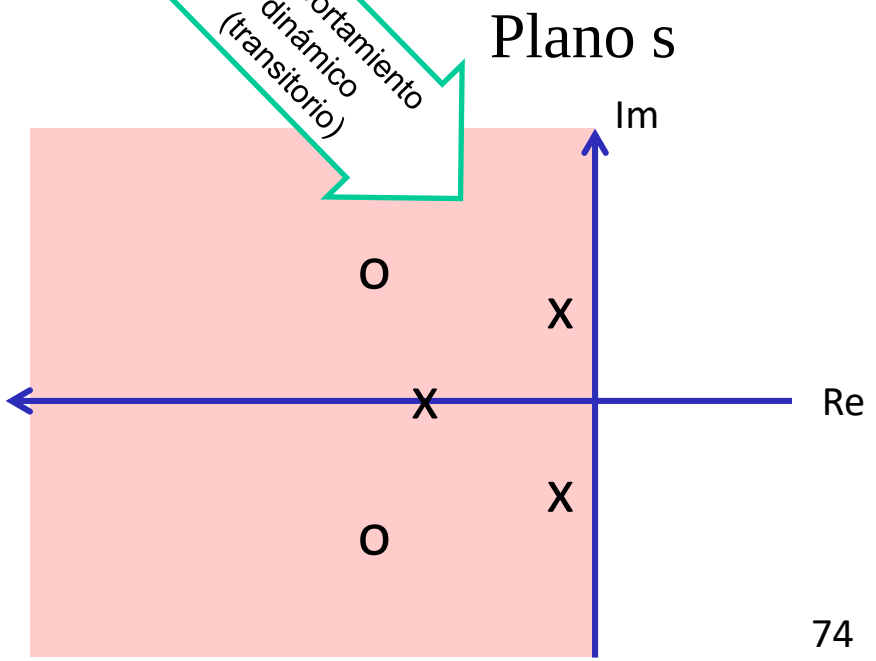


$$G(s) = \frac{Y(s)}{X(s)} = \frac{b_o.s^m + b_1.s^{m-1} + ... + b_{m-1}.s + b_m}{s^n + a_1.s^{n-1} + ... + a_{n-1}.s + a_n} \quad , n >= m$$

Comportamiento estático (régimen permanente)

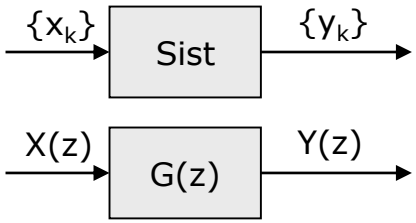
Comportamiento dinámico (transitorio)

$$K_p = \lim_{t\to\infty} \frac{y(t)}{x(t)} = \lim_{s\to 0} G(s)$$



Las matemáticas del control

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



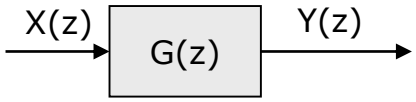
$$G(z) = \frac{Y(z)}{X(z)} = \frac{b_o + b_1.z^{-1} + ... + b_{m-1}.z^{-(m-1)} + b_m.z^{-m}}{1 + a_1.z^{-1} + ... + a_{n-1}.z^{-(n-1)} + a_n.z^{-n}}$$

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

  - Ecuación en diferencias 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.

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

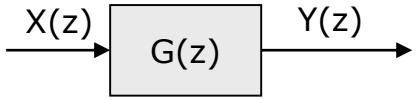
  - Es habitual presentar G(z) dividiendo entre z^n:

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

  - 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 en Z, interpretación:

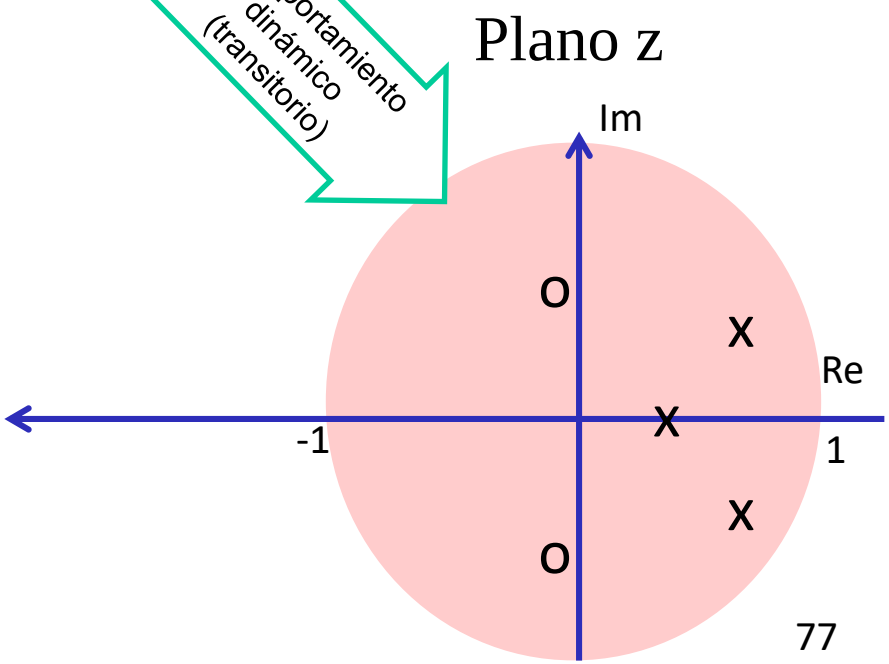


$$G(z) = \frac{Y(z)}{X(z)} = \frac{b_o.z^m + b_1.z^{m-1} + ... + b_{m-1}.z + b_m}{z^n + a_1.z^{n-1} + ... + a_{n-1}.z + a_n} \quad , n >= m$$

Comportamiento estático (régimen permanente)

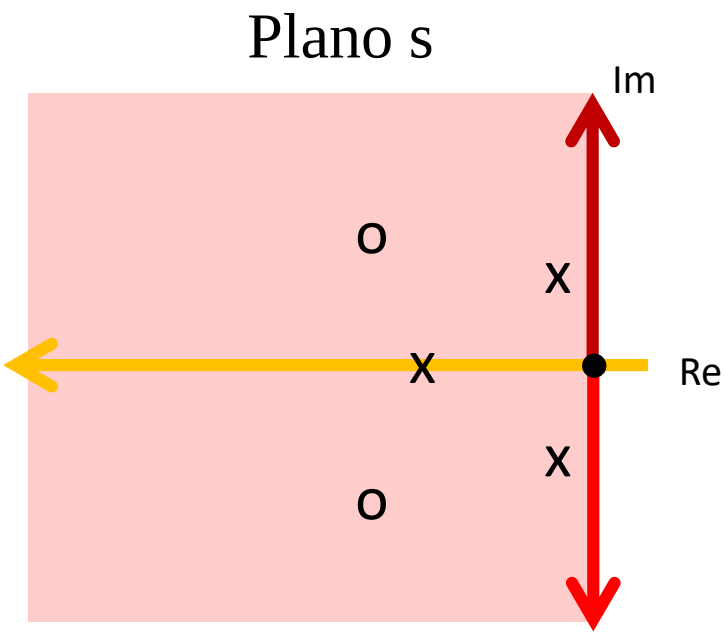
Comportamiento dinámico (transitorio)

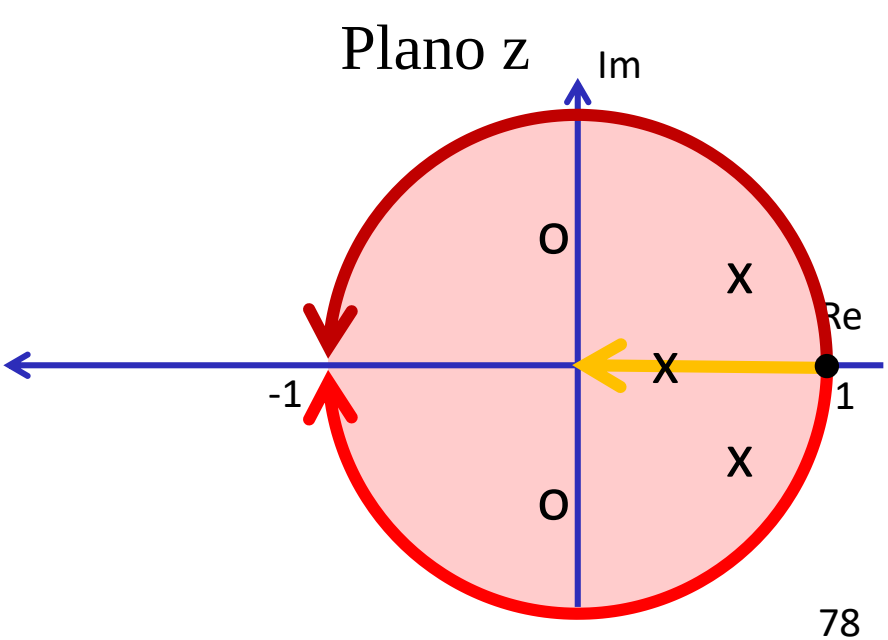
$$K_p = \lim_{t\to\infty} \frac{\{y_k\}}{\{x_k\}} = \lim_{z\to 1} G(z)$$



Las matemáticas del control

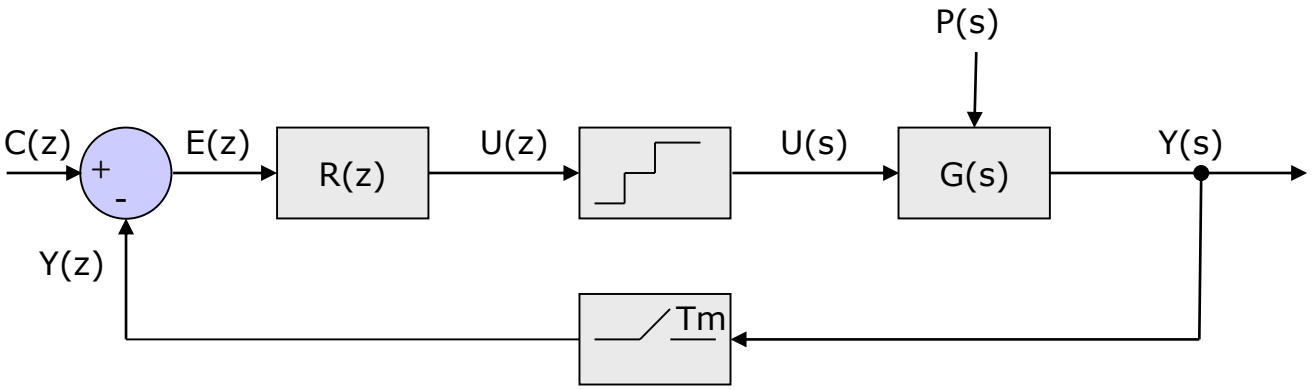
- Sistema continuo vs sistema discreto:
  - El semiplano de estabilidad Re(s)<0 se convierte en el interior de la circunferencia de radio unidad |z| < 1
  - s=0 ≡ z=1
  - s=∞ ≡ z=0





Las matemáticas del control

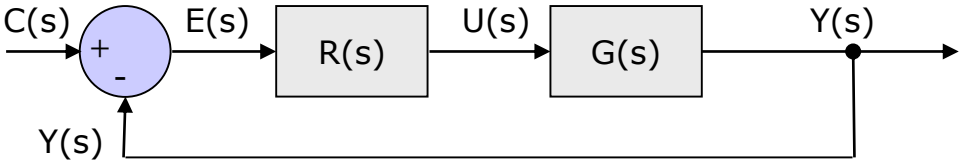
- El lazo de control híbrido en forma matemática:



- 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
- Elección de Tm:
  - Suficientemente pequeño para que en el sistema realimentado se puedan 'seguir' las variaciones: < 0.1 * $\tau_{sistema\_realimentado}$

Las matemáticas del control

- Dos aproximaciones:
  - Suponer sistema completo continuo, calcular regulador continuo R(s) y discretizar:



R(s) = resultado aplicación teoría Regulación Automática

R(z) = R(s=$\frac{2}{T_m}\frac{z-1}{z+1}$) (Aproximación de Tustin)

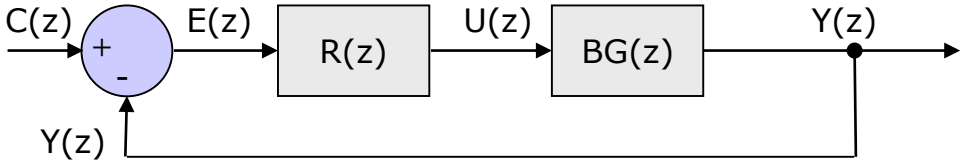
(En Matlab)

```
>> rs=tf([coefs numerador rs], [coefs denominador rs])
>> rz=c2d(rs,Tm,'tustin')
```

  - Suponer sistema completo discreto y calcular directamente regulador discreto

Las matemáticas del control

- Dos aproximaciones:
  - Suponer sistema completo continuo, calcular regulador continuo R(s) y discretizar.
  - Suponer sistema completo discreto y calcular directamente regulador discreto



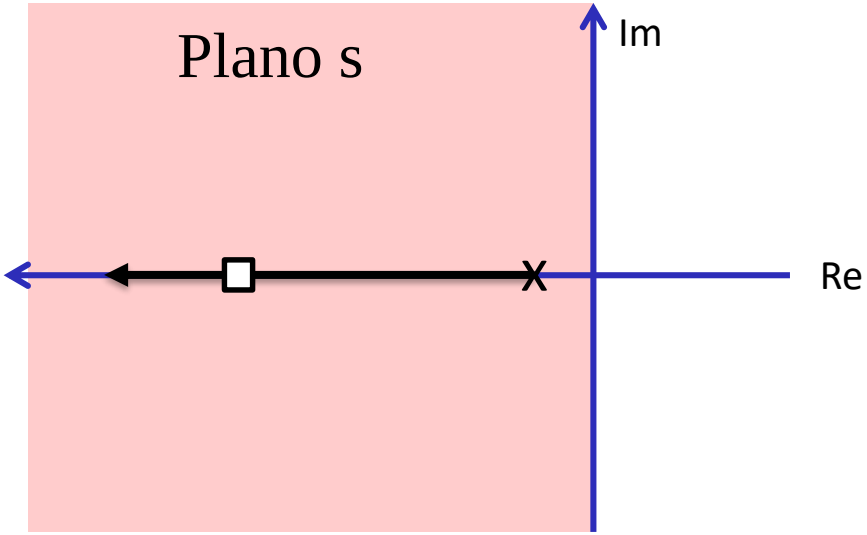
BG(z) = discretización del conjunto bloqueador + sistema

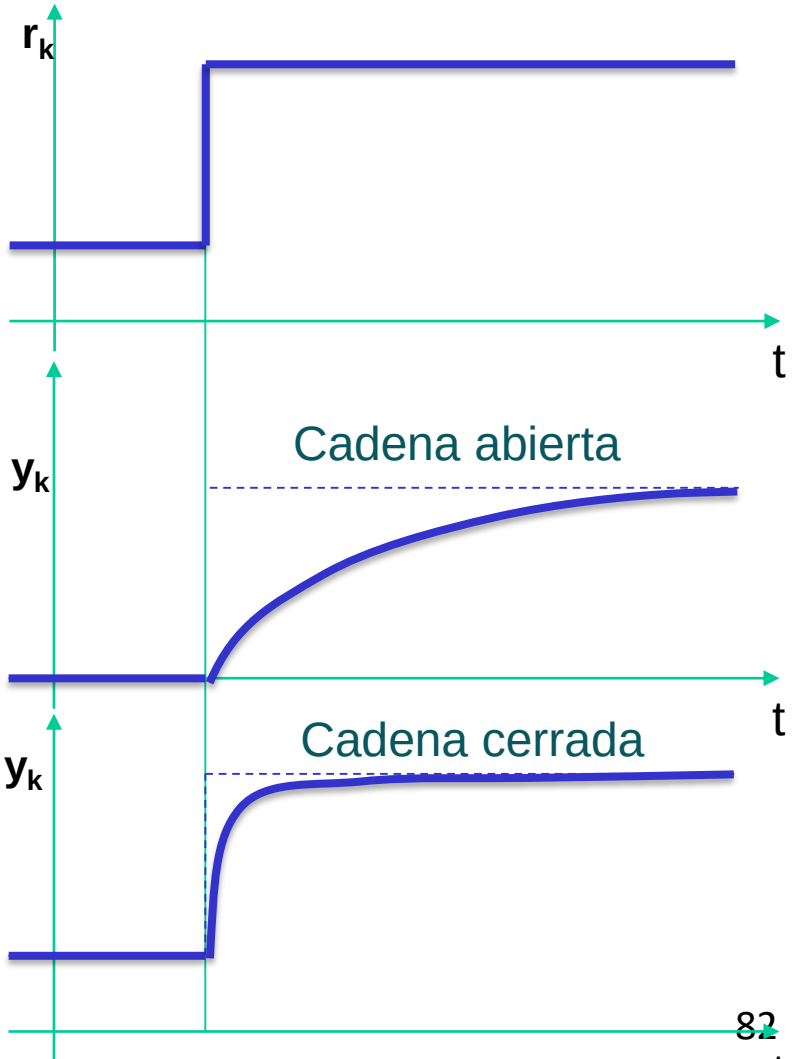
(En Matlab)
>> gs=tf([coefs numerador gs], [coefs denominador gs])
>> bgz=c2d(gs,Tm,'zoh')

R(z) = cálculo utilizando teoría de sistemas discretos

Las matemáticas del control

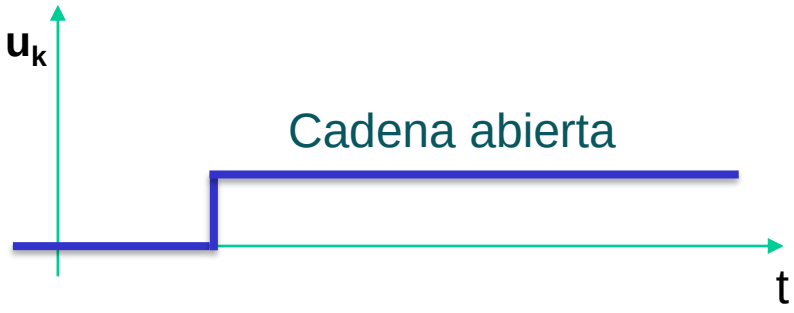
- ¡¡¡¡ ATENCION !!!! Las matemáticas “admiten todo”, la realidad no. Ejemplo :
  - En un sistema con un solo polo, se puede realimentar con Kp grande y conseguir una respuesta mucho más rápida que en cadena abierta.

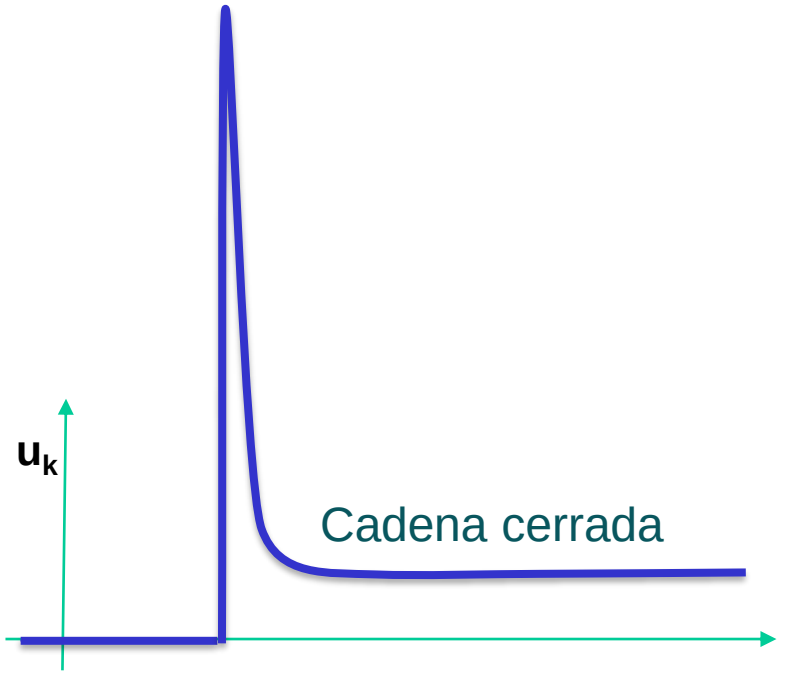




Las matemáticas del control

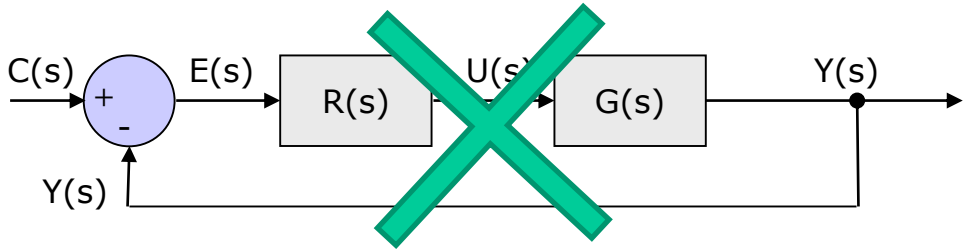
- ¡¡¡¡ ATENCION !!!! Las matemáticas “admiten todo”, la realidad no:
  - Para conseguir mayor velocidad, el controlador calcula acciones de control muy grandes al inicio, que son imposibles de generar en el sistema físico real

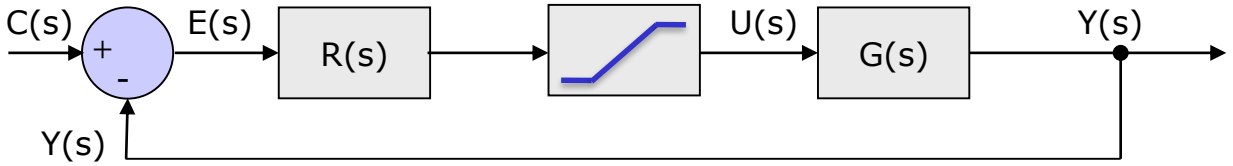




Las matemáticas del control

- Si se desea simular el comportamiento real, hay que añadir una saturación en el modelo que refleje los límites del accionador (comportamiento no-lineal)



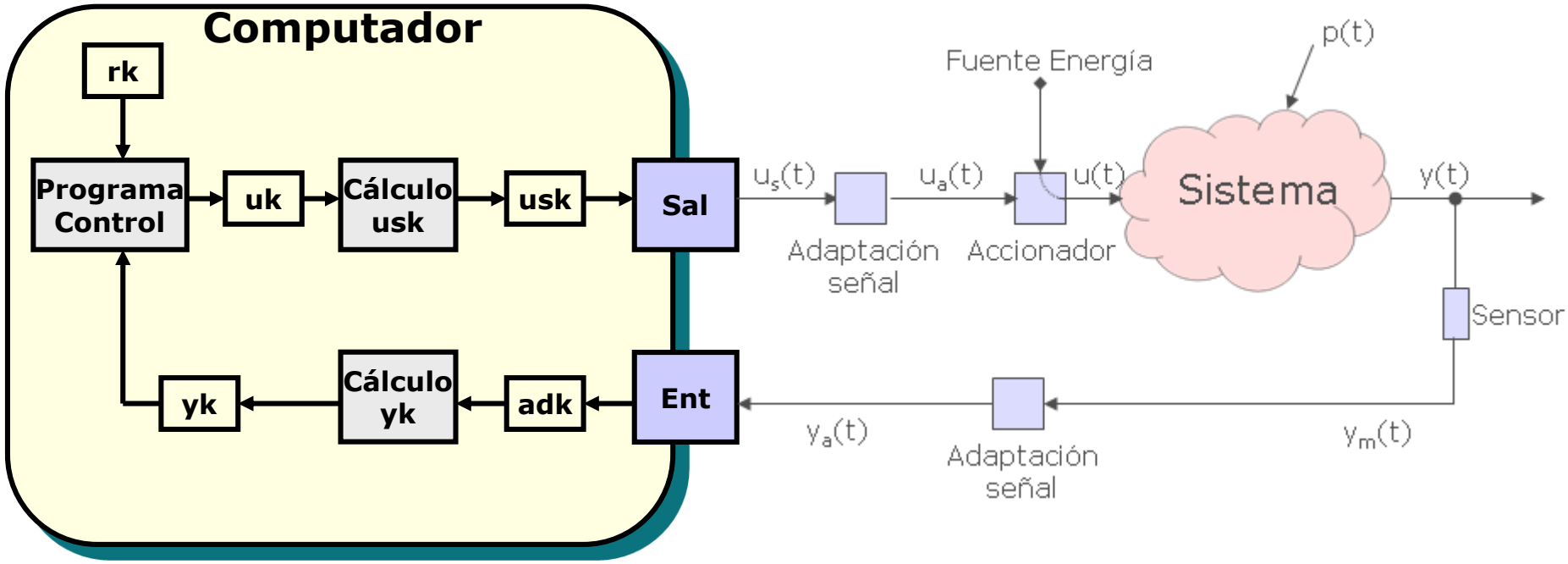


Indice

- Introducción control de procesos por computador
- Conceptos básicos para programación de control
- Interfaz del computador con el exterior
- 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:



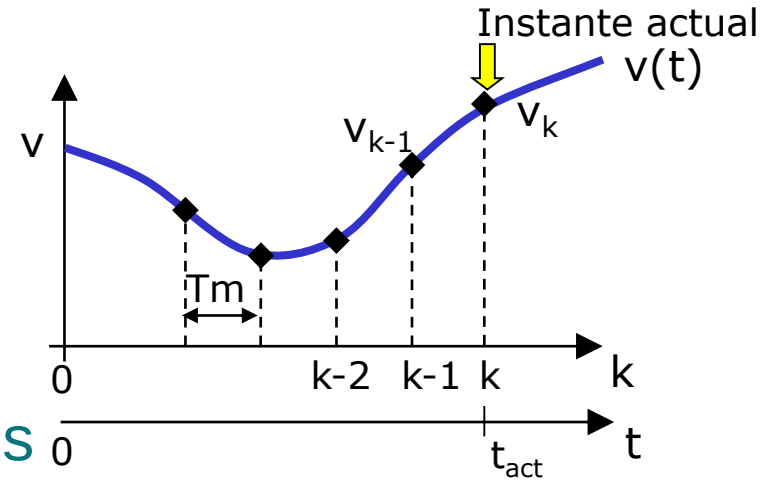
Programación del lazo de control

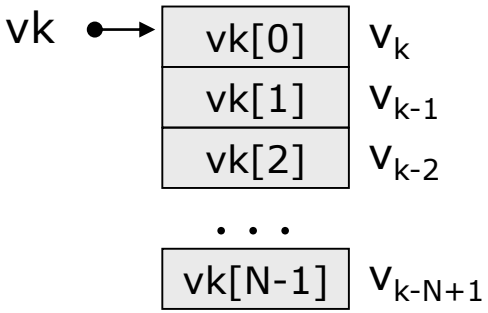
- Programación del lazo de control:
  - Cada Tm (periodo de muestreo) se debe repetir el mismo algoritmo.
  - El algoritmo debe poder ejecutarse hasta un tiempo indefinido.
- Ejecución de un paso del lazo de control:
  - Actualización de tablas temporales
  - Adquisición de datos
  - Cálculo de valores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

- Aspecto general de un programa de control

```
...

main()
{
   float c_k,y_k,u_k[N_MAX+1],e_k[M_MAX+1];
   otras variables necesarias (ej: m,n, tablas b[i],a[i])

   inicializaciones: m,n,b[i],a[i],e_k[i]=0,u_k[i]=0, etc.

   while (1)
   {
       Desplazar Tablas de variables temporales (al menos u_k, e_k)
       c_k=Nuevo valor consigna
       y_k=Valor leído de sensor
       e_k[0]=c_k-y_k;
       u_k[0]=CalculoAccionRegulador(e_k,u_k,...);
       AplicarAccionDeControl(u_k[0]);
       Otras operaciones auxiliares (comu operador, bb.dd., etc.)
       Sleep(Tm);
   }
}
```

Programación del lazo de control

- Aspecto general de un programa de control

```
...

main()
{
   float c_k,y_k,u_k[N_MAX+1],e_k[M_MAX+1];
   otras variables necesarias (ej: m,n, tablas b[i],a[i])

   inicializaciones: m,n,b[i],a[i],e_k[i]=0,u_k[i]=0, etc.

   while (1)
   {
       Desplazar Tablas de variables temporales (al menos u_k, e_k)
       c_k=Nuevo valor consigna
       y_k=Valor leído de sensor
       e_k[0]=c_k-y_k;
       u_k[0]=CalculoAccionRegulador(e_k,u_k,...);
       AplicarAccionDeControl(u_k[0]);
       Otras operaciones auxiliares (comu operador, bb.dd., etc.)
       Sleep(Tm);
   }
}
```

Variables temporales (secuencias):
- Variable 'normal' si sólo es necesario valor actual
- Array si son necesarios valores anteriores

Variables atemporales

Programación del lazo de control

- Aspecto general de un programa de control

```
...

main()
{
   float c_k,y_k,u_k[N_MAX+1],e_k[M_MAX+1];
   otras variables necesarias (ej: m,n, tablas b[i],a[i])

   inicializaciones: m,n,b[i],a[i],e_k[i]=0,u_k[i]=0, etc.

   while (1)
   {
       Desplazar Tablas de variables temporales (al menos u_k, e_k)
       c_k=Nuevo valor consigna
       y_k=Valor leído de sensor
       e_k[0]=c_k-y_k;
       u_k[0]=CalculoAccionRegulador(e_k,u_k,...);
       AplicarAccionDeControl(u_k[0]);
       Otras operaciones auxiliares (comu operador, bb.dd., etc.)
       Sleep(Tm);
   }
}
```

Dar valores iniciales a todas las variables antes de iniciar el bucle (para secuencias, típicamente 0)

Programación del lazo de control

- Aspecto general de un programa de control

```
...

main()
{
   float c_k,y_k,u_k[N_MAX+1],e_k[M_MAX+1];
   otras variables necesarias (ej: m,n, tablas b[i],a[i])

   inicializaciones: m,n,b[i],a[i],e_k[i]=0,u_k[i]=0, etc.

   while (1)
   {
       Desplazar Tablas de variables temporales (al menos u_k, e_k)
       c_k=Nuevo valor consigna
       y_k=Valor leído de sensor
       e_k[0]=c_k-y_k;
       u_k[0]=CalculoAccionRegulador(e_k,u_k,...);
       AplicarAccionDeControl(u_k[0]);
       Otras operaciones auxiliares (comu operador, bb.dd., etc.)
       Sleep(Tm);
   }
}
```

Se ejecuta una pasada del bucle cada Tm: cada ejecución del bucle corresponde a un nuevo instante

Programación del lazo de control

- Aspecto general de un programa de control

```
...

main()
{
   float c_k,y_k,u_k[N_MAX+1],e_k[M_MAX+1];
   otras variables necesarias (ej: m,n, tablas b[i],a[i])

   inicializaciones: m,n,b[i],a[i],e_k[i]=0,u_k[i]=0, etc.

   while (1)
   {
       Desplazar Tablas de variables temporales (al menos u_k, e_k)
       c_k=Nuevo valor consigna
       y_k=Valor leído de sensor
       e_k[0]=c_k-y_k;
       u_k[0]=CalculoAccionRegulador(e_k,u_k,...);
       AplicarAccionDeControl(u_k[0]);
       Otras operaciones auxiliares (comu operador, bb.dd., etc.)
       Sleep(Tm);
   }
}
```

Si cada ejecución del bucle corresponde a un nuevo instante, los valores de las señales temporales están ahora retrasados 1 Tm: actualizar

Programación del lazo de control

- Aspecto general de un programa de control

```
...

main()
{
   float c_k,y_k,u_k[N_MAX+1],e_k[M_MAX+1];
   otras variables necesarias (ej: m,n, tablas b[i],a[i])

   inicializaciones: m,n,b[i],a[i],e_k[i]=0,u_k[i]=0, etc.

   while (1)
   {
       Desplazar Tablas de variables temporales (al menos u_k, e_k)
       c_k=Nuevo valor consigna
       y_k=Valor leído de sensor
       e_k[0]=c_k-y_k;
       u_k[0]=CalculoAccionRegulador(e_k,u_k,...);
       AplicarAccionDeControl(u_k[0]);
       Otras operaciones auxiliares (comu operador, bb.dd., etc.)
       Sleep(Tm);
   }
}
```

Obtener los valores actuales (del nuevo instante) de las diferentes señales

Programación del lazo de control

- Aspecto general de un programa de control

```
...

main()
{
   float c_k,y_k,u_k[N_MAX+1],e_k[M_MAX+1];
   otras variables necesarias (ej: m,n, tablas b[i],a[i])

   inicializaciones: m,n,b[i],a[i],e_k[i]=0,u_k[i]=0, etc.

   while (1)
   {
       Desplazar Tablas de variables temporales (al menos u_k, e_k)
       c_k=Nuevo valor consigna
       y_k=Valor leído de sensor
       e_k[0]=c_k-y_k;
       u_k[0]=CalculoAccionRegulador(e_k,u_k,...);
       AplicarAccionDeControl(u_k[0]);
       Otras operaciones auxiliares (comu operador, bb.dd., etc.)
       Sleep(Tm);
   }
}
```

Con los valores actualizados, calcular la nueva acción de control, y aplicarla al exterior

Programación del lazo de control

- Aspecto general de un programa de control

```
...

main()
{
   float c_k,y_k,u_k[N_MAX+1],e_k[M_MAX+1];
   otras variables necesarias (ej: m,n, tablas b[i],a[i])

   inicializaciones: m,n,b[i],a[i],e_k[i]=0,u_k[i]=0, etc.

   while (1)
   {
       Desplazar Tablas de variables temporales (al menos u_k, e_k)
       c_k=Nuevo valor consigna
       y_k=Valor leído de sensor
       e_k[0]=c_k-y_k;
       u_k[0]=CalculoAccionRegulador(e_k,u_k,...);
       AplicarAccionDeControl(u_k[0]);
       Otras operaciones auxiliares (comu operador, bb.dd., etc.)
       Sleep(Tm);
   }
}
```

Ahora puede haber tiempo para 'ocuparse' de tareas 'menos importantes'

Indice

- Introducción control de procesos por computador
- Conceptos básicos para programación de control
- Interfaz del computador con el exterior
- 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
- Conceptos básicos para programación de control
- Interfaz del computador con el exterior
- 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

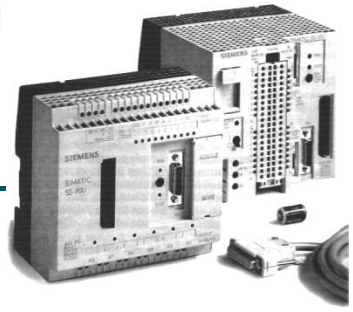
- Procesador Digital de Señal (DSP)

- 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
  - …

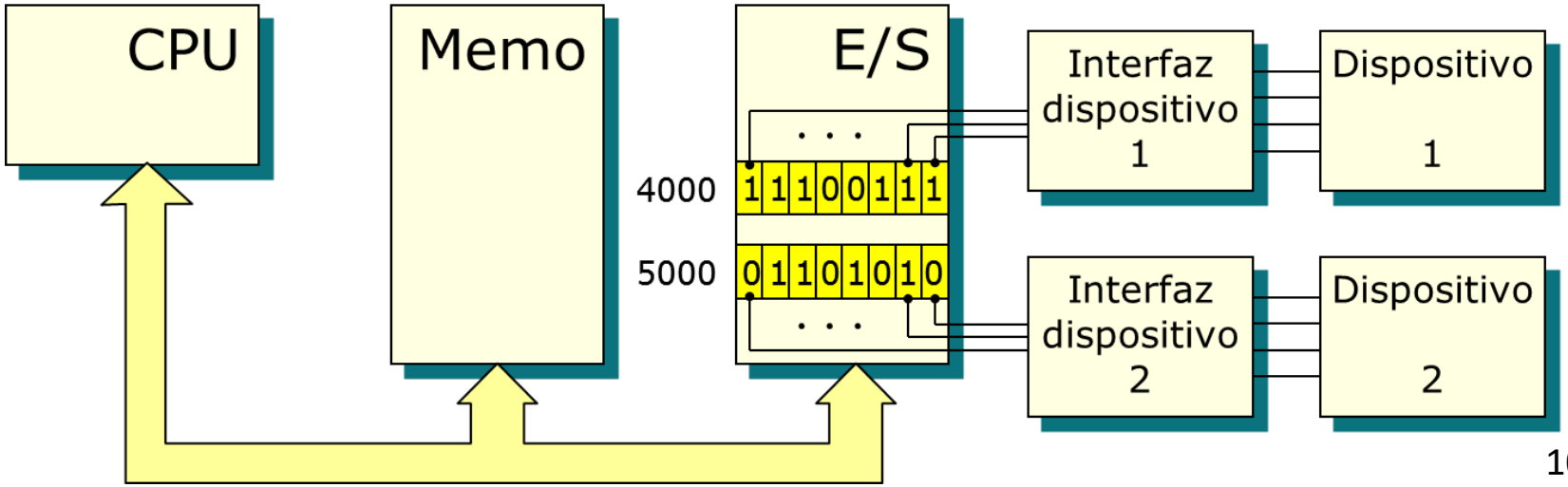
  } **Interrupciones**

- **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.

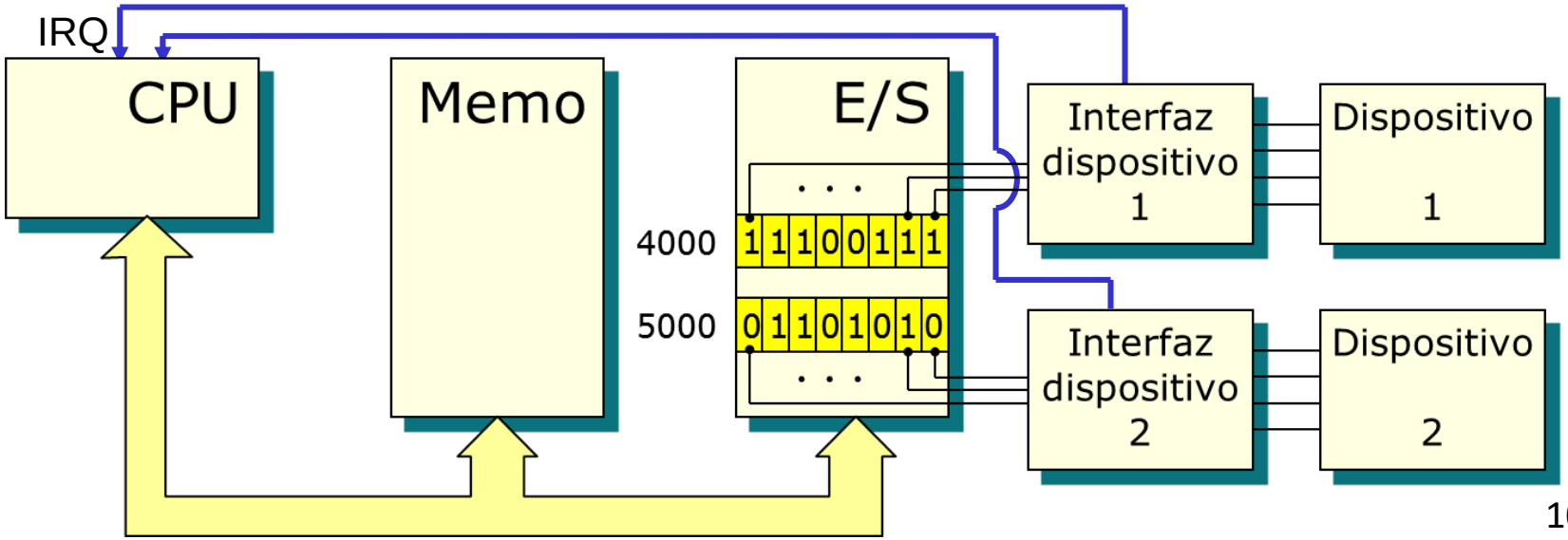
Implantación del control

- 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

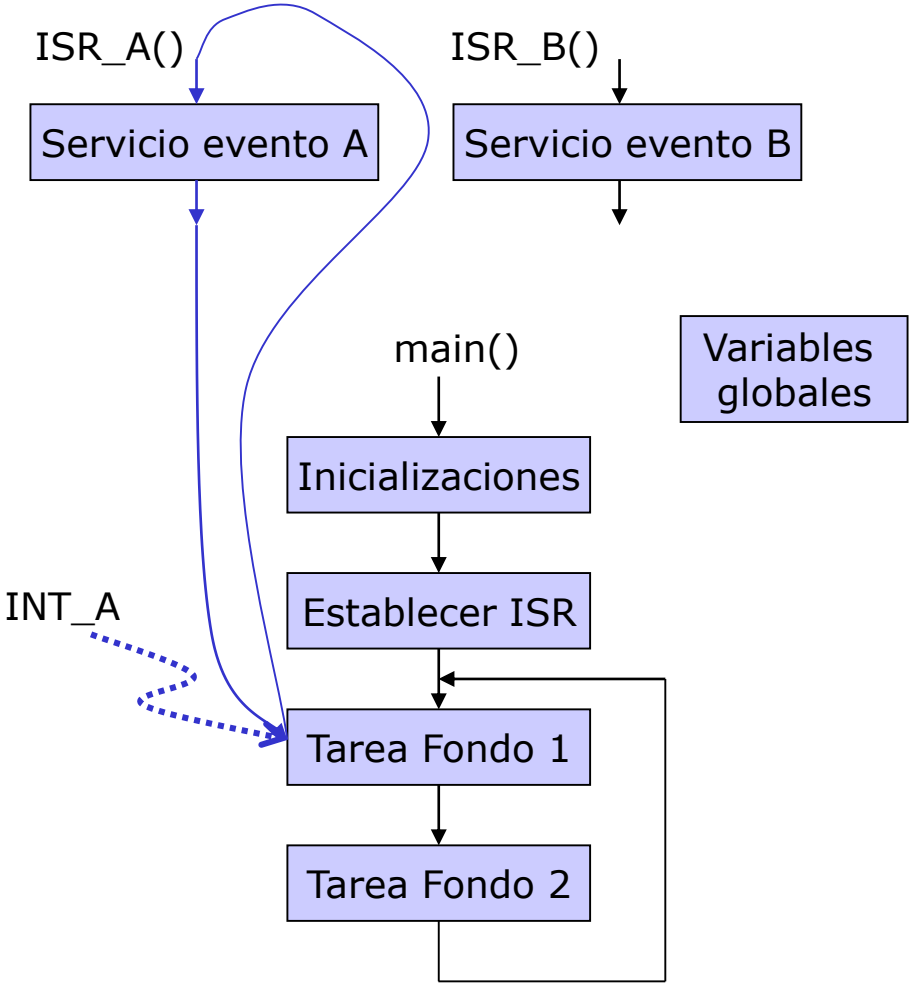
- Sistemas ‘grandes’:
  - Una tarea especial Sistema Operativo (S.O.) se encarga de dar servicio a:
  - Acceso al hardware (modo síncrono y asíncrono):
    - Nivel de S.O.
    - Nivel de driver
  - Planificación de tareas:
    - Cada vez que sucede un ‘evento’, el S.O. toma el control, actualiza estado 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,...);
}
```

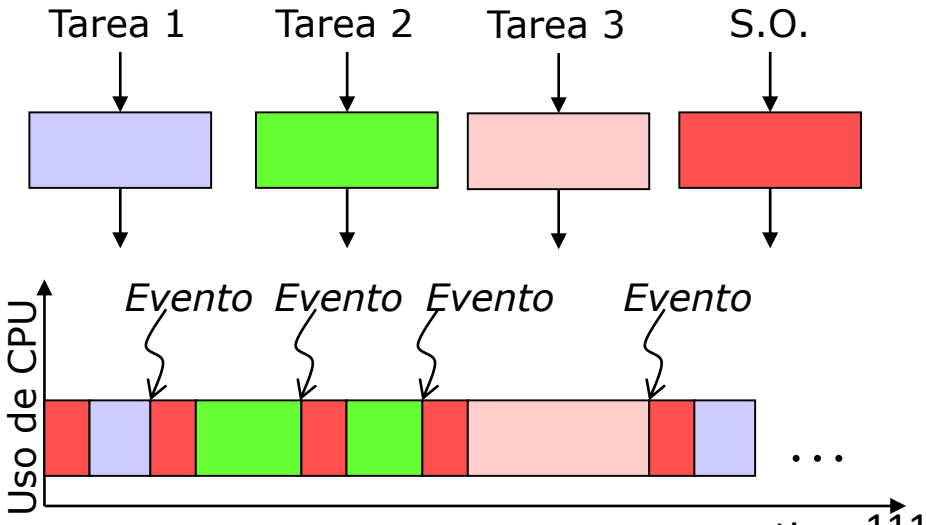
```
main()
{
     ...
     InicializacionVariables(...);
     EstablecerISRTemporiz(TM_ms,ISR_Control);

     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

Sistemas “grandes”

- El programador organiza su código en tareas, que se ejecutan “en paralelo” o, más exactamente, “en concurrencia”.
- Las tareas no siempre quieren ejecutar su código, están en muchas ocasiones en estado de espera (por teclado, por temporización, por aviso de otra tarea, …)
- Una tarea especial, el Sistema Operativo, se encarga de:
  - Gestión de puertos E/S mediante pequeños programas (drivers)
  - Gestión de eventos y temporizaciones
  - Planificación de tareas
  - Gestión de memoria



Ante cualquier evento:
- El S.O.toma el control (la CPU pasa a ejecutar su código).
- El evento puede provocar el cambio de estado de alguna tarea.
- El S.O. revisa las tareas pendientes y cede el control a la más prioritaria que necesite ejecutarse, hasta el siguiente evento.

Sistemas “grandes”

- Variedad de Sistemas Operativos:
  - Windows, Unix, Linux, Mac OS, Android, VX-Works, …
- Tipos de Sistemas Operativos:
  - De tiempo compartido:
    - El retardo en ejecutar una tarea no es importante.
    - El S.O. distribuye tiempos de ejecución entre todas las tareas.
  - De tiempo real:
    - El retardo en ejecutar una tarea puede ser muy importante.
    - El S.O. ejecuta siempre la tarea más prioritaria.
- POSIX: Portable Operating System Interface:
  - Basado en Unix
  - Mismo interfaz con las tareas (system calls) para diferentes Sistemas Operativos.