

Máquinas de Estados

Informática II - R2004
2021

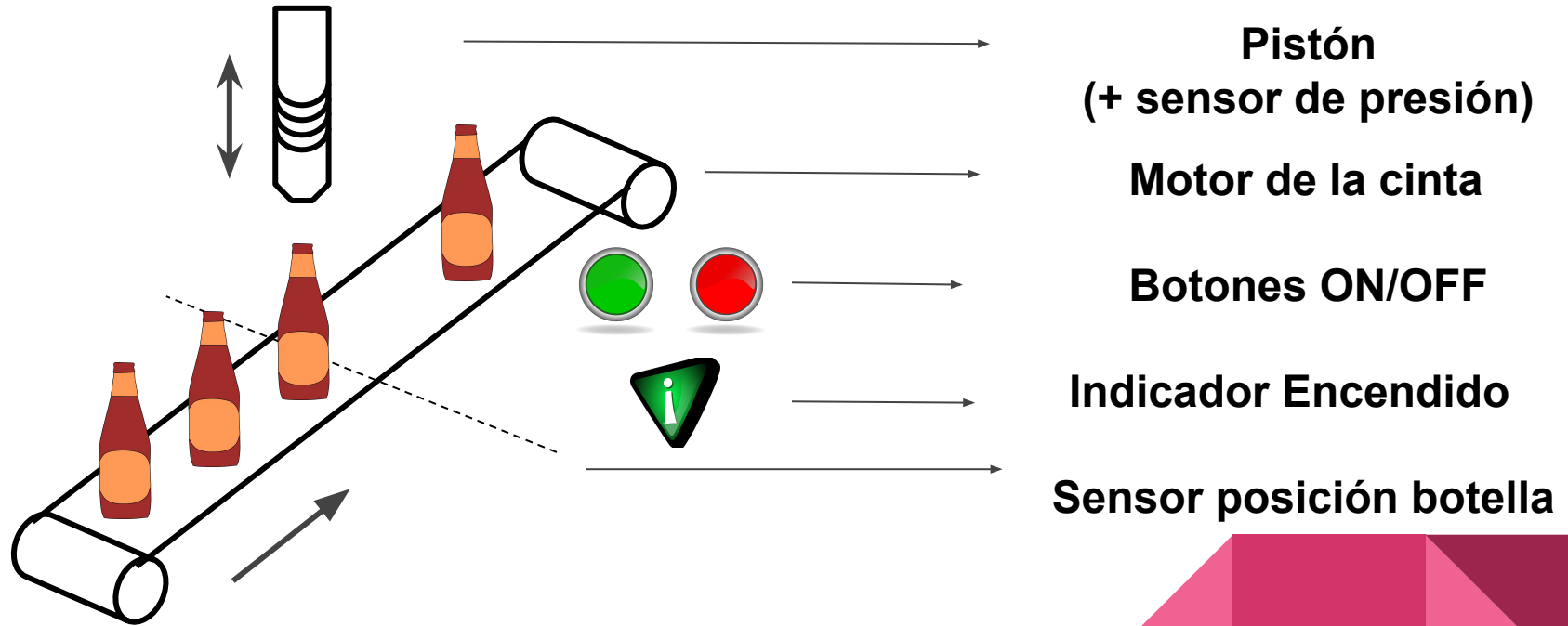
¿Qué tipo de sistemas estamos programando?

Como ejemplo de un sistema que podríamos implementar, vamos a trabajar sobre una máquina taponadora de botellas de cerveza:

<https://drive.google.com/open?id=0B6Ko0eS6KO5OU1dPN1B3Vjhialk>

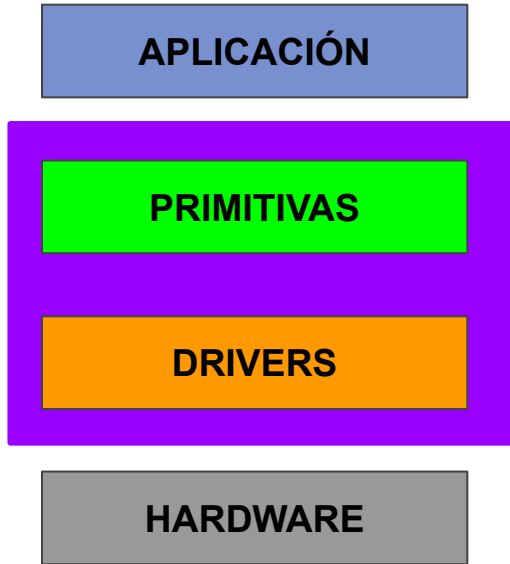


¿Qué tipo de sistemas estamos programando?



Cualquier sistema puede ser diagramado en base a sus entradas y salidas

¿Cómo vamos a programar un sistema complejo?



Entradas:

<code>bool Encendido()</code>	<code>//Botón de encendido</code>
<code>bool Parada()</code>	<code>//Botón de parada</code>
<code>bool Presencia()</code>	<code>//Sensor de presencia de botellas</code>
<code>bool Presion()</code>	<code>//Sensor de presión del pistón</code>

Salidas:

<code>CintaOn() / CintaOff()</code>	<code>//Motor de la cinta</code>
<code>PistonOn() / PistonOff()</code>	<code>//Pistón neumático</code>
<code>IndicadorOn() / IndicadorOff()</code>	<code>//Luz encendido</code>

LIBRERÍA INFO 2

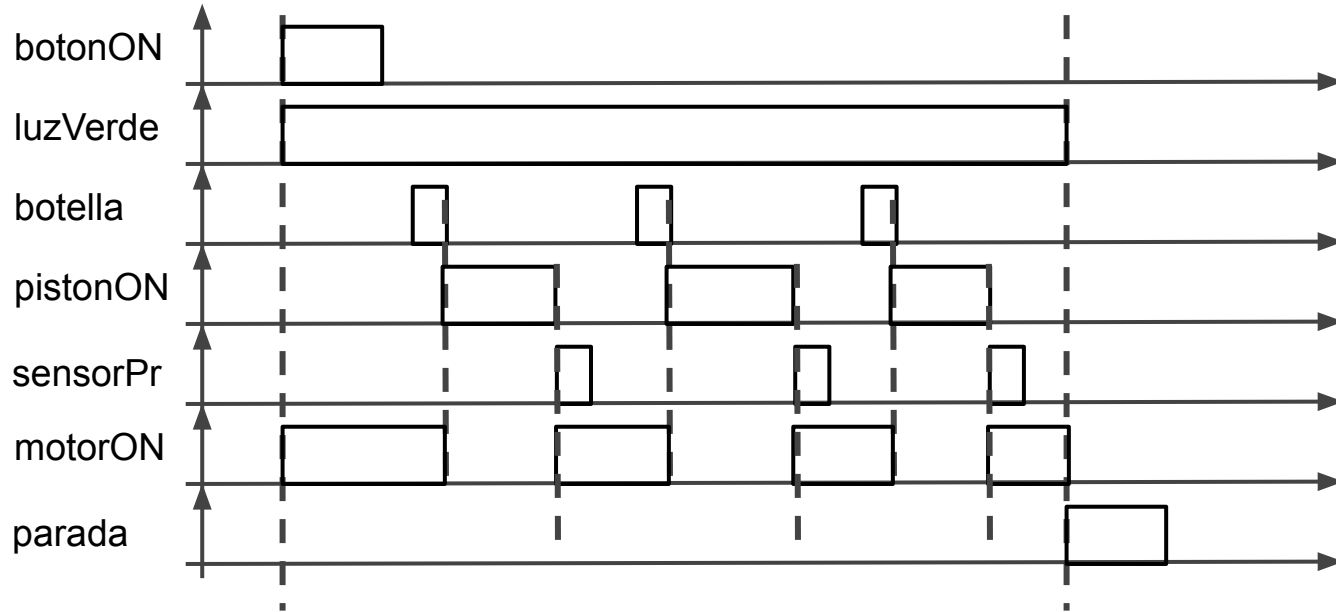
¿Cómo vamos a programar un sistema complejo?

APLICACIÓN

La APLICACIÓN es la lógica que hace funcionar nuestro programa. Es esencialmente el algoritmo que hace que el mismo se comporte de acuerdo a la lógica que nosotros diseñamos, siguiendo los pasos que pensamos cuando hacemos el análisis de nuestro sistema.



¿Cómo podemos diagramar el funcionamiento de un sistema?

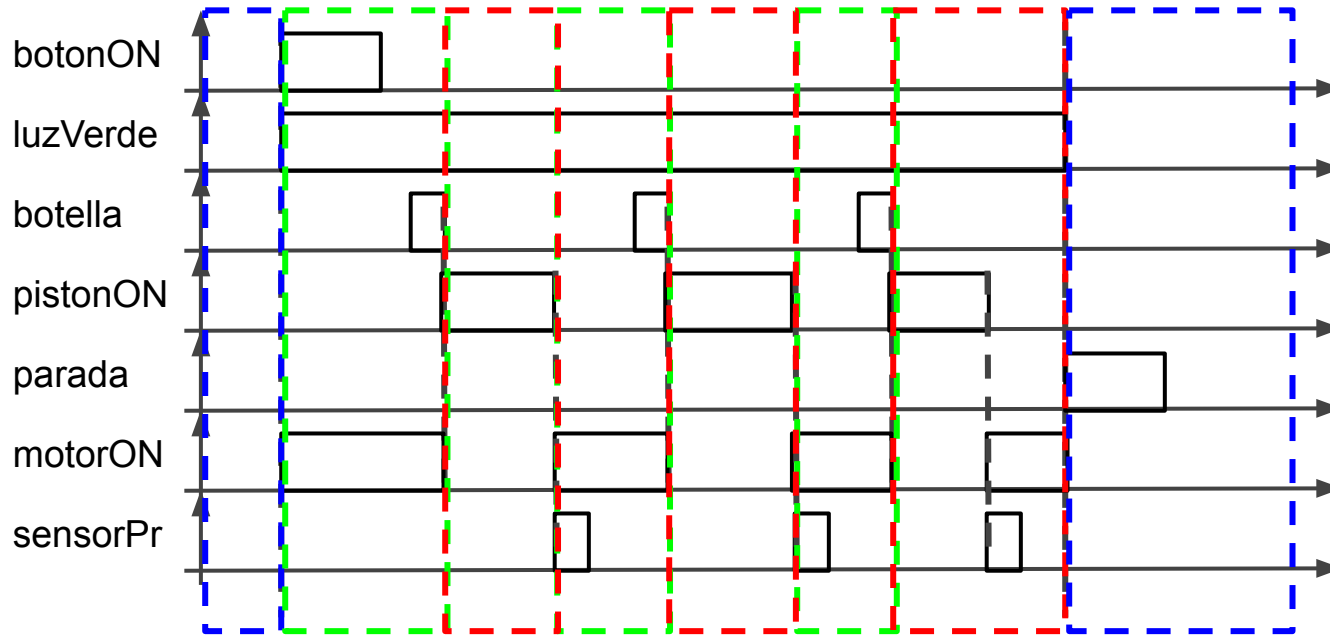


Cada máquina va a tener una lógica de control que va a ser más o menos compleja de acuerdo al sistema por el que esté compuesta

¿Cómo organizo la gran cantidad de E/S de mi sistema en un solo programa?

- En lugar de focalizar mi atención en las E/S en sí mismas, vamos a intentar buscar los ESTADOS ESTACIONARIOS de mi sistema
- Estos estados son aquellos en los que mi máquina tiene todas sus E/S “estables”. Intentamos focalizarnos en la “estabilidad” del sistema más que en el valor constante de las entradas y salidas. Por ejemplo, un estado estable puede ser uno en el cual un led de estado está parpadeando (su salida cambia constantemente, pero sin embargo el estado del sistema en general permanece “estable”. Un sistema en modo alarma puede prender y apagar una sirena en forma intermitente, pero no por esto cambio de estado.

Identificando los estados de mi sistema



Estado 1
Máquina
PARADA

Estado 2
Cinta
Encendida

Estado 3
Piston
Activo

Haciendo un diagrama más sencillo...

Estado 1
Máquina
PARADA

Estado 2
Cinta
Encendido

Estado 3
Piston
Activo

Cuando tocan el botón de encendido
(prendo el motor y el indicador verde)

Cuando presionan la parada
(apago la cinta y el indicador verde)

Cuando presionan la parada
(desactivo el pistón y la luz verde)

Cuando detecto una botella
(apago la cinta y activo el pistón)

Cuando se activa el sensor de presión
(prendo la cinta y desactivo el pistón)

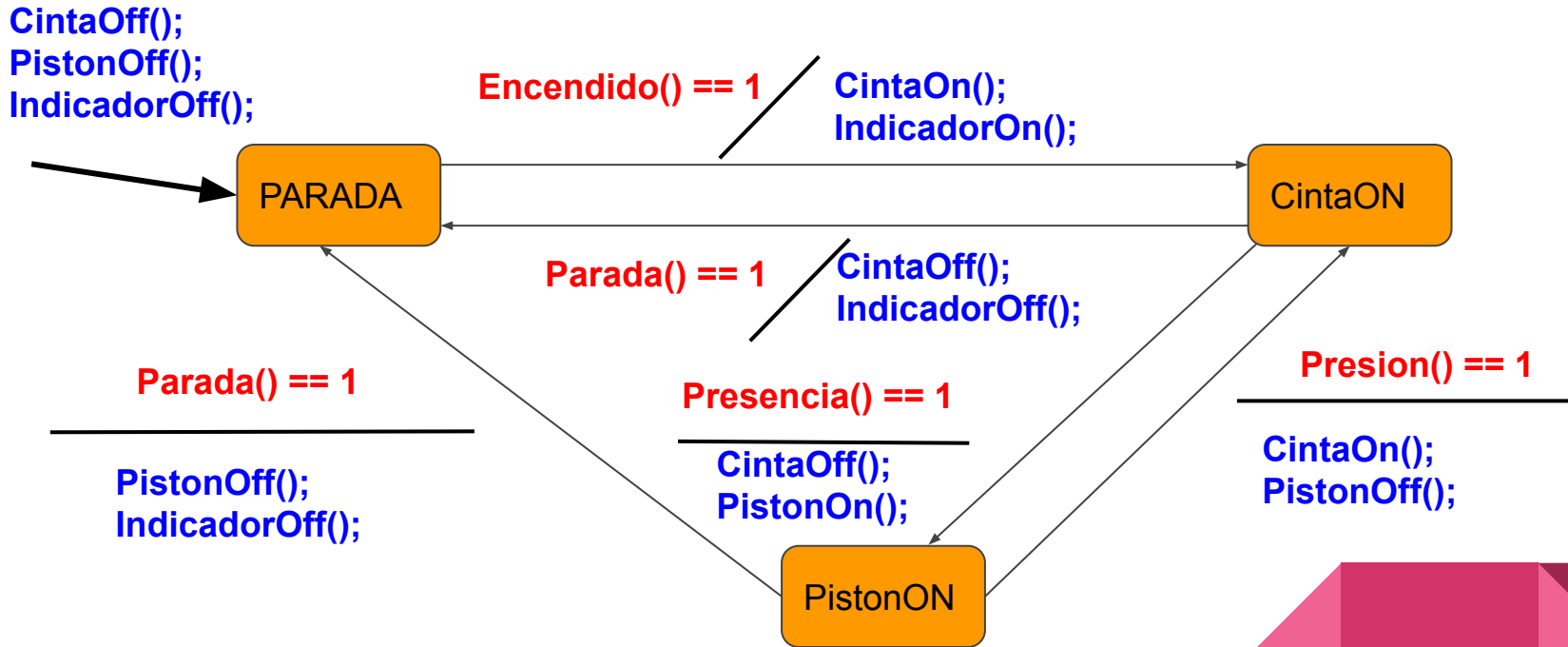
Organización de los estados en un método gráfico

Para simplificar el diagrama de un sistema, voy a organizar los estados del mismo con un método gráfico. En el mismo debemos identificar:

- Los ESTADOS de mi sistema
- Las TRANSICIONES posibles entre los estados
- Los EVENTOS que generan dichas transiciones
- Las ACCIONES que traen aparejados los cambios de estado



Nuestro primer diagrama de estados



SIEMPRE DEBEMOS IDENTIFICAR NUESTRO ESTADO DE “RESET”

Codificación de nuestro diagrama de estados

Recordando el Diseño de un algoritmo para SE...



Diseño de un algoritmo para un sistema embebido

2. Atención “inmediata” de las entradas y salidas

¿Cómo hacemos para que el microcontrolador esté siempre pendiente de lo que está pasando?

```
while ( 1 )  
{  
    while ( LeerBoton1() == 1 ) {  
        PrenderLampara1();  
    }  
    ApagarLampara1();  
  
    while ( LeerBoton2() == 1 ) {  
        PrenderLampara2();  
    }  
    ApagarLampara2();  
}
```



¿Funciona?



¿Sigue
Funcionando?

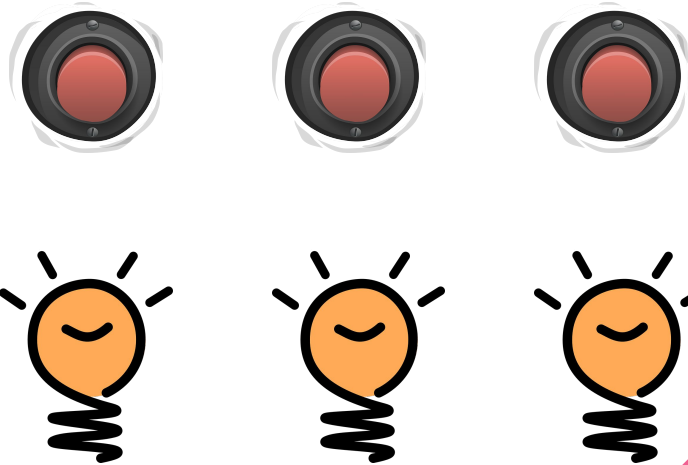
¡¡EN INFO2 NO PODEMOS USAR LOOPS BLOQUEANTES!!

Diseño de un algoritmo para un sistema embebido

2. Atención “inmediata” de las entradas y salidas

¿Cómo hacemos para que el microcontrolador esté siempre pendiente de lo que está pasando?

```
while ( 1 )  
{  
    if (LeerBoton1() == 1)  
        EncenderLampara1();  
    else  
        ApagarLampara1();  
  
    if (LeerBoton2() == 1)  
        EncenderLampara2();  
    else  
        ApagarLampara2();  
}
```



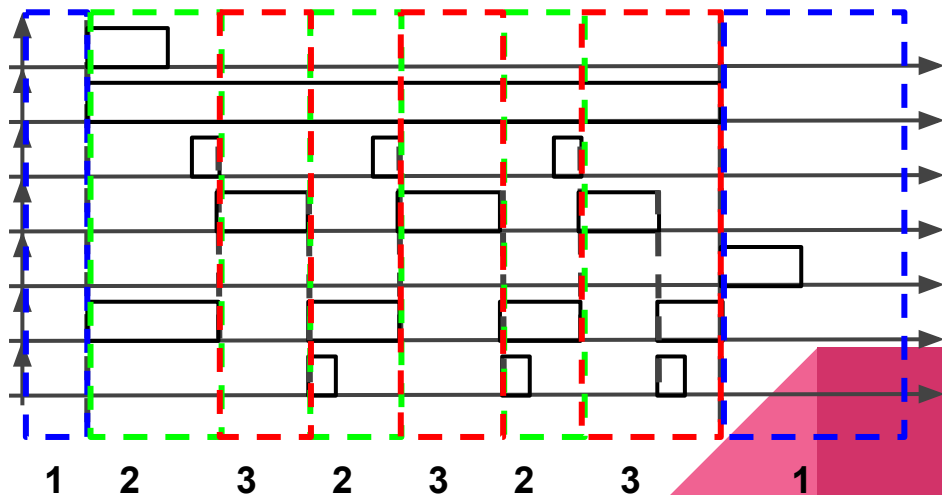
Introducimos la “variable de estado”

Una variable de estado será una variable que nos permita saber en que estado de los diseños se encuentra el sistema. Intentaremos codificar el programa de manera que sea lo más entendible que podamos por terceros.

De esta manera:

```
#define PARADA 1
#define CINTA_ON 2
#define PISTON_ON 3
```

unsigned char estado



Codificación: switch - case

```
switch ( estado ) {
```

```
case PARADA:
```

```
    if ( Encendido()==1 )  
    {
```

```
        CintaOn();  
        IndicadorOn();
```

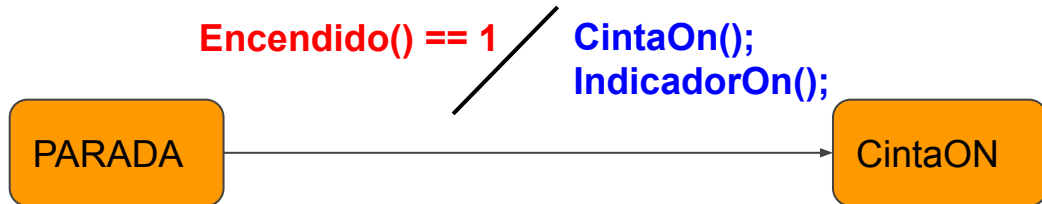
```
        estado = CINTA_ON;
```

```
    }  
    break;
```

```
}
```

```
case CINTA_ON:
```

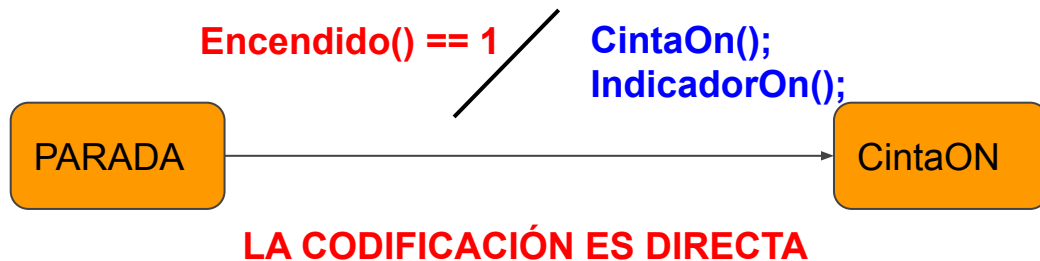
```
case PISTON_ON:
```



LA CODIFICACIÓN ES DIRECTA

Codificación: cadena de if

```
if ( estado == PARADA ){  
    if ( Encendido() )  
    {  
        CintaOn();  
        IndicadorOn();  
        estado = CINTA_ON;  
    }  
}  
  
else if ( estado == PISTON_ON ){  
  
}  
  
else if ( estado == CINTA_ON ){  
  
}
```



Precauciones

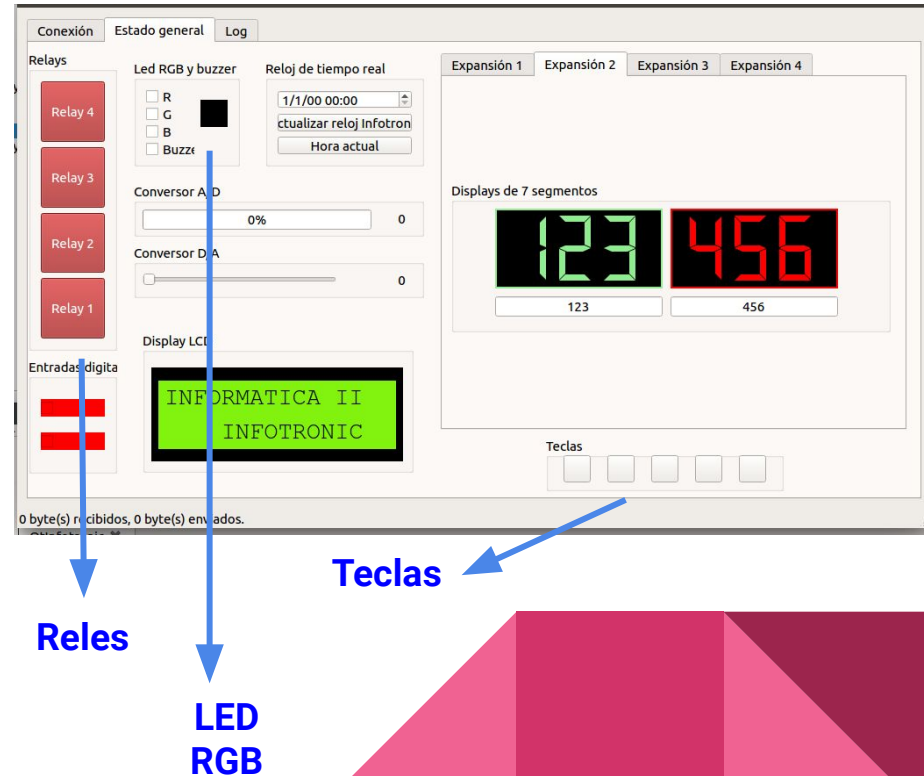
En cualquiera de los casos, la pérdida del valor de la variable estado haría que perdamos el control del sistema, y por ende que la máquina deje de responder al operador. Se pueden tomar algunos recaudos para que esto no pase:

```
switch (estado) {  
    case PARADA:  
        ...  
    case CINTA_ON:  
        ...  
    case PISTON_ON:  
        ...  
    default:  
        CintaOff();  
        IndicadorOff();  
        PistonOff();  
        estado = PARADA;  
}
```

```
if ( estado == PARADA )  
{  
    ...  
}  
else if ( estado == CINTA_ON )  
{  
    ...  
}  
else if ( estado == PISTON_ON )  
{  
    ...  
}  
else {  
    CintaOff();  
    IndicadorOff();  
    PistonOff();  
    estado = PARADA;  
}
```

Simulación en el kit y en el MicroModel

Si bien un sistema embebido real que controle esta máquina tendrá salidas dedicadas para hacer cada una de las acciones del ejemplo, en nuestro caso probaremos todos los ejercicios en un kit de desarrollo. El mismo cuenta con algunas entradas y salidas básicas (4 relés, un led RGB, 5 teclas, algunas entradas digitales externas), por lo que cada una de las salidas de nuestro sistema la simularemos con alguna de estas entradas y salidas que tenemos disponible



Veamos esta máquina de estados en umf

ENTRADAS

Tecla0 : Encendido --- Presencia --- Presion

Tecla1 : Parada

SALIDAS

Led Azul : PrenderMotor / ApagarMotor

Led Verde : PrenderPistón / ApagarPistón


Led Rojo : PrenderLedRojo / ApagarLedRojo



Temporizaciones en las máquinas de estado

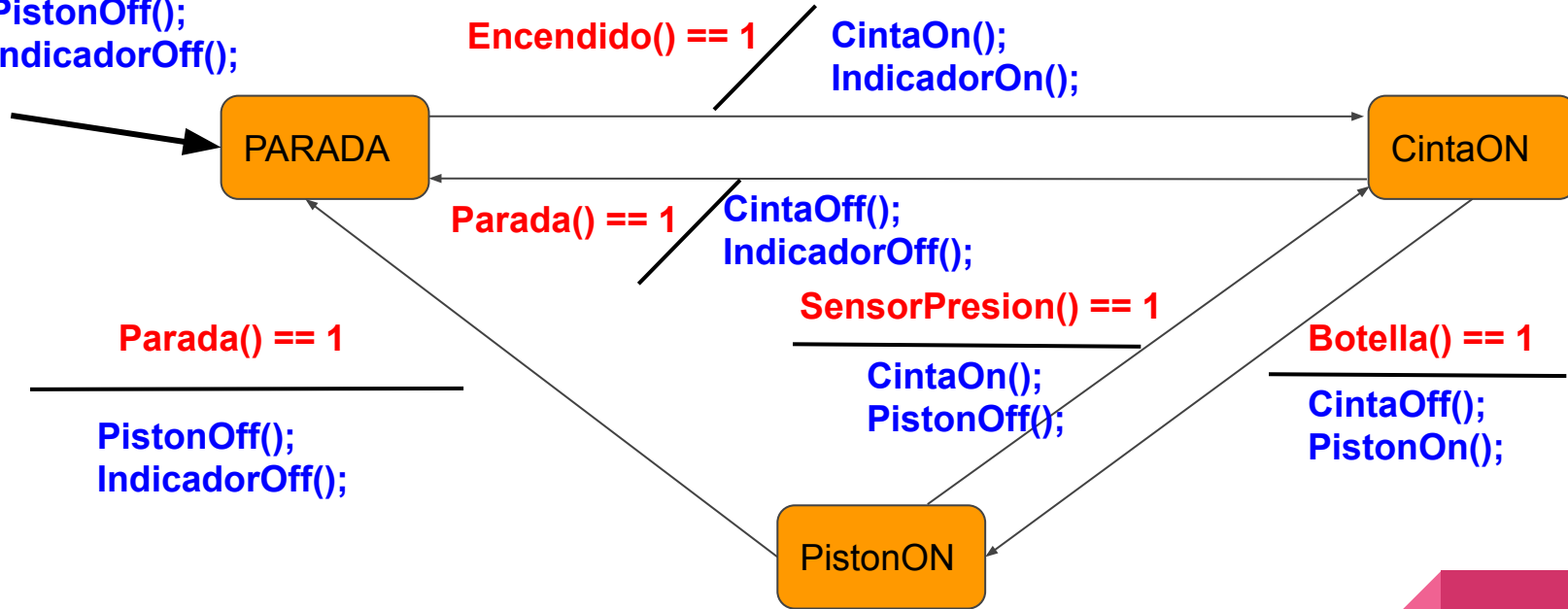
De la misma manera que una variación en una E/S del sistema puede generar una transición en una máquina de estados, el vencimiento de un tiempo puede tener el mismo efecto.

Para que haya una temporización en nuestro diagrama, deberemos considerar:

1. Un momento en el que disparamos el temporizador
 - *¿Cuándo empiezo a contar el tiempo?*
 2. Un momento en el que el temporizador finaliza
 - *¿Qué pasa cuando se venció el tiempo que estaba contando?*
 3. Si existiesen eventualidades que hacen que el temporizador deba detenerse
 - *¿Qué hago si mientras contaba tiempo pasó otra cosa?*
- 

Ejemplo... Nuestra taponadora de cervezas

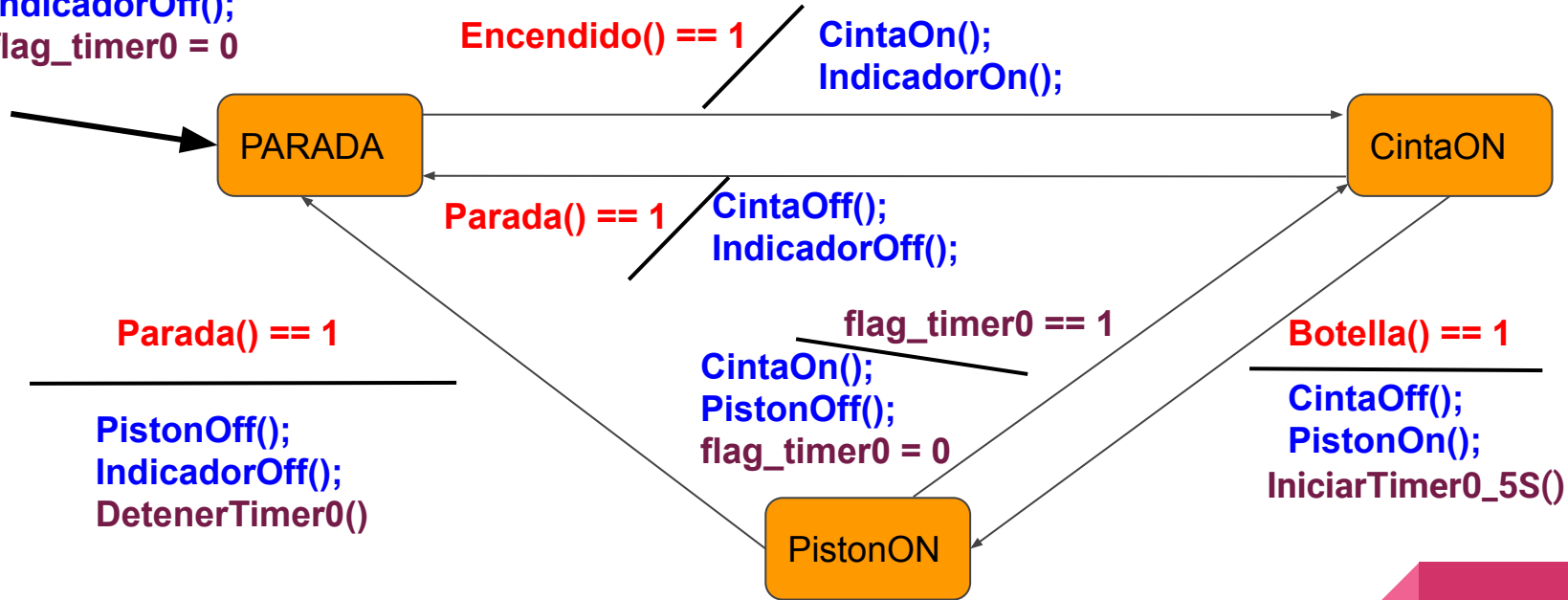
CintaOff();
PistonOff();
IndicadorOff();



¿Cómo se modificaría si en lugar de un presostato el pistón se activara por 5 segundos?

Ejemplo... Nuestra taponadora de cervezas

CintaOff();
PistonOff();
IndicadorOff();
flag_timer0 = 0



Funciones de la biblioteca para temporizaciones

Para disparar un temporizador:

```
void TimerStart(uint8_t event, uint32_t time, void (*handler)(void) handler , uint8_t base);
```

Parámetros

event: Número de evento entre 0 y 31

t: Tiempo del evento. Dependiente de la base de tiempos

handler: Callback del evento

base: base de tiempo del time posibles valores: DEC, SEG, MIN

Número de timer, cada uno de ellos puede llevar un tiempo. Por si quiero hacer varias temporizaciones en paralelo. Tengo 32 disponibles.

Tiempo que temporiza. Hay que tener en cuenta la base de tiempo que ponemos. Por ejemplo si queremos 5 segundos puede ir **t** en 5 y **base** en SEG o **t** en 50 y **base** en DEC

Función que se ejecuta cuando pasa (vence) el tiempo que seteamos. Desde esa función podemos levantar un flag y en la máquina de estados estar viendo ese flag. La función tiene que recibir y devolver void.

Funciones de la biblioteca para temporizaciones

Detener un temporizador:

```
void TimerStop( uint8_t event );
```

Parámetros

event: Número de evento entre 0 y 31

Miremos la documentación de la librería para ver qué otras funciones hay.



¿Cómo se completa el ejemplo?

```
uint8_t flag_timer0 = 0;
```

```
void IniciarTimer0_5S( void )  
{  
    TimerStart( 0, 5, Timer0Vencido , SEG);  
}
```

IniciarTimer0_5S()

```
void Timer0Vencido( void )  
{  
    flag_timer0 = 1;  
}
```

```
void DetenerTimer0( void )  
{  
    TimerStop(0);  
}
```

DetenerTimer0()

Llamar en el while(1) TimerEvent()


¿Qué tipo de datos tengo en el micro?

Como parte de las librerías de la cátedra nos encontraremos con la redefinición de 3 tipos de datos (char, int y short) para hacer mención específica al tamaño de las mismas en memoria:

```
typedef unsigned char  uint8_t  
typedef unsigned int   uint32_t  
typedef unsigned short int uint16_t
```

Y de la misma manera:

```
typedef char  int8_t  
typedef int   int32_t  
typedef short int int16_t
```

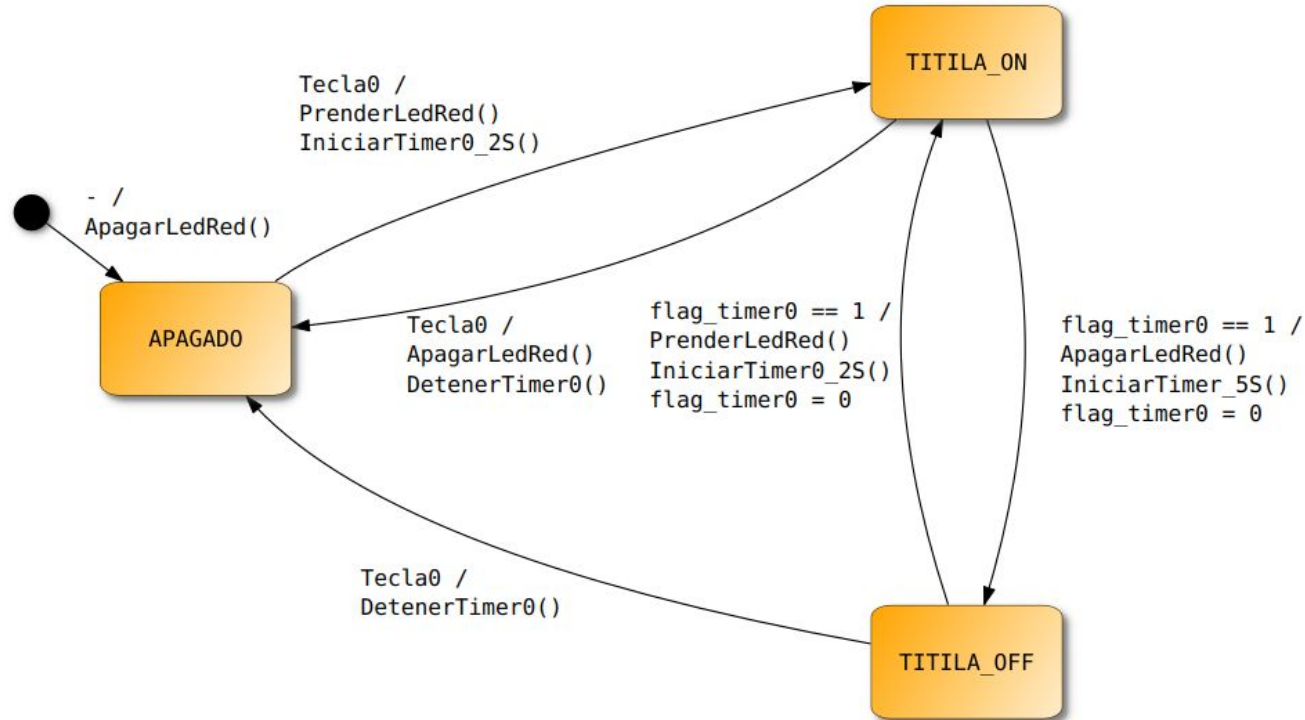


Hagamos un ejemplo sencillo con tiempos

Realizar una aplicación que mediante la opresión de una tecla prenda o apague un led y al estar prendido dicho led debe titilar (2 Segundos prendido / 5 segundos apagado).



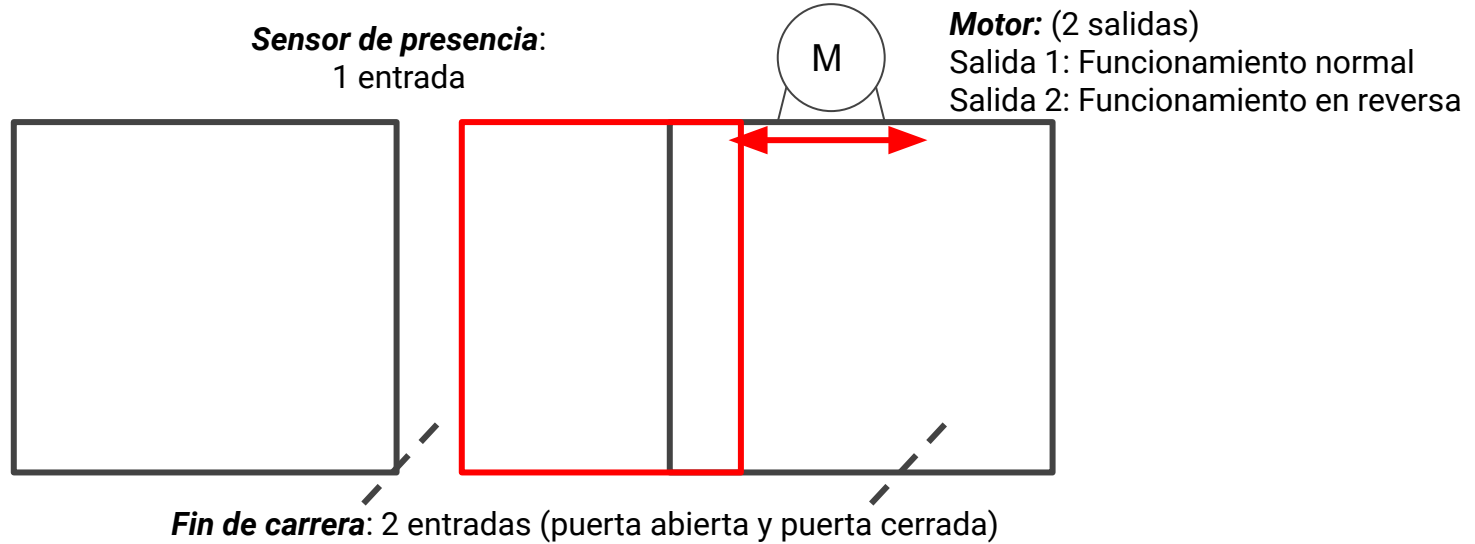
Hagamos un ejemplo sencillo con tiempos - MdE



**Escribamos el código en
el MCU**

Ejercicio:

Diseñar el diagrama de estados (y su correspondiente codificación en C) de un sistema compuesto por una puerta (con un motor para abrirla o cerrarla), un sensor de presencia y dos sensores de fin de carrera (que permiten saber si la puerta se abrió o cerró en su totalidad). La puerta deberá abrirse cuando el sensor de presencia detecta una persona, deberá permanecer abierta completamente por 5 segundos, y volver a cerrarse. La aplicación deberá controlar la apertura o cierre de la puerta, en función de las entradas descritas.



¿Cómo modifico el ejemplo para agregar la condición de seguridad que si la puerta no se cierra o no se abre completamente en 10 segundos desde que se da la orden se apaga el motor y se espera a que se presione un botón de servicio, que vuelve a abrir la puerta y vuelve el sistema a su funcionamiento normal?