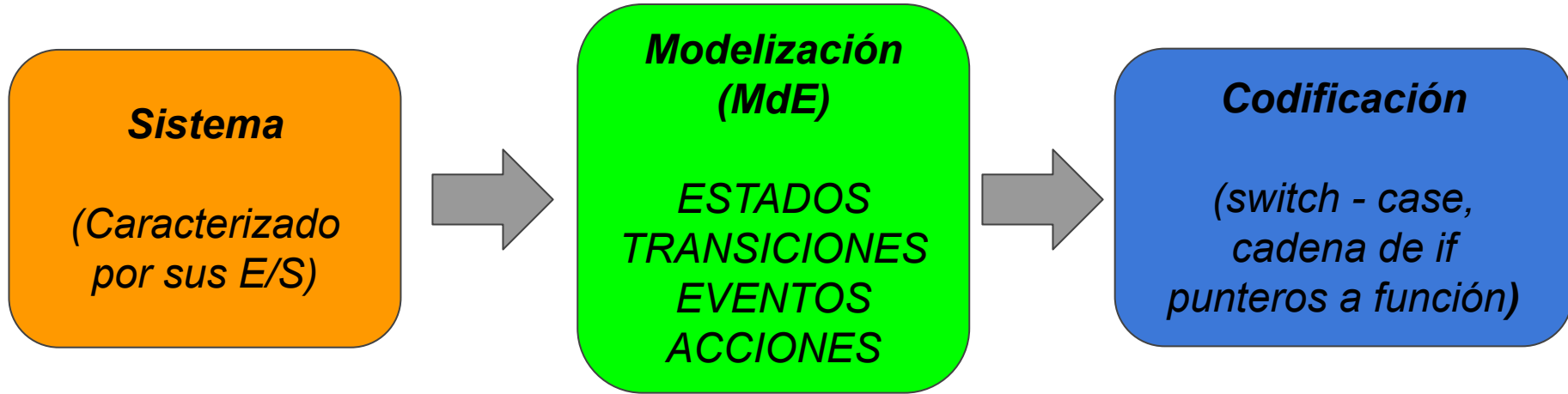


Máquinas de estado - Temporizaciones y MdE en paralelo

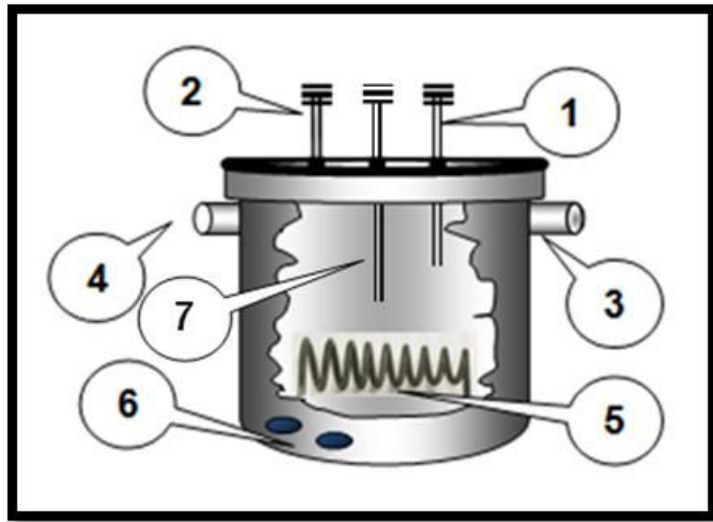
Informática II
R2004 - 2021

Recordando... ¿Para qué utilizábamos las máquinas de estado?



¿Cualquier sistema es modelizable con una máquina de estados?

Vamos a partir de un sistema que se nos presente complejo, para pensar cómo podemos modelizarlo en estados estables de manera sencilla. El siguiente es un ejemplo de la caldera de un edificio, que se basa en un recipiente que debe estar constantemente lleno de agua, y cuenta con una serpentina que debe en todo momento estar calentando para suplir de agua caliente al edificio.

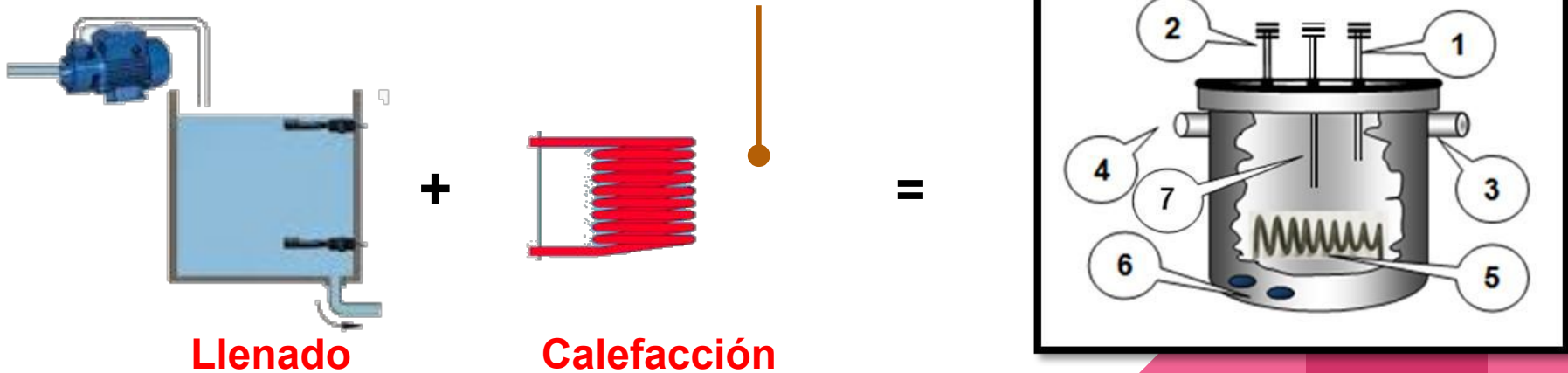


Índice	Descripción
1	Sensor de nivel de Máxima : Varilla de acero inoxidable
2	Varilla asociada a la carcasa de la caldera. La carcasa es de acero inoxidable y oficia de tierra.
3	Solenoide de paso de agua
4	Salida de vapor
5	Calefactor
6	Bornes de conexión del calefactor
7	Sensor de nivel de Mínima : Varilla de acero inoxidable

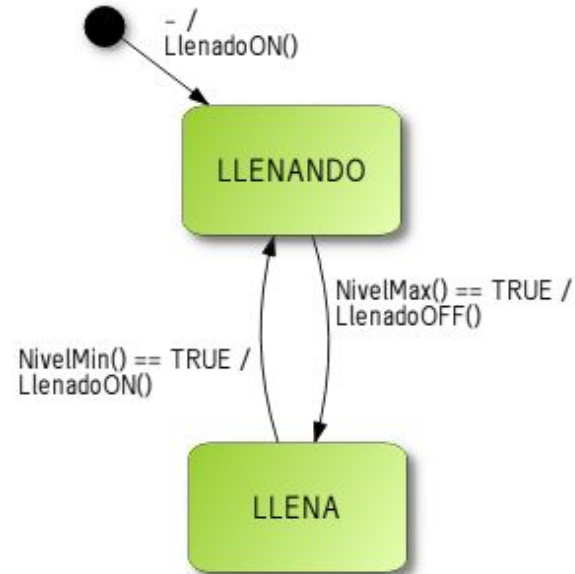
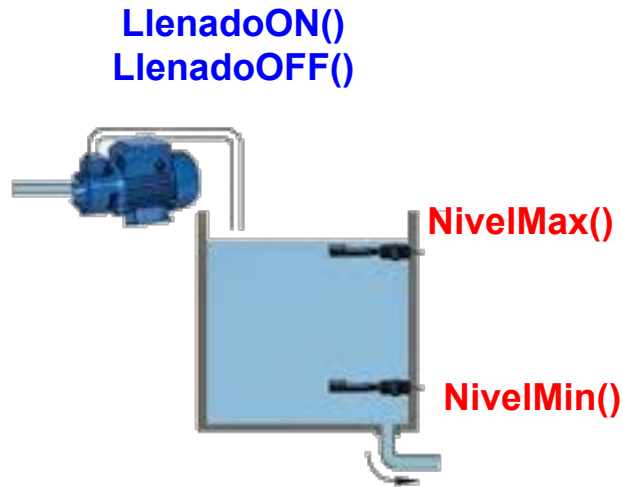
Combinación de máquinas de estado

Si bien los sistemas pueden ser modelizados con máquinas de estado lo suficientemente abarcativas, los sistemas complejos muchas veces pueden ser modelizados por **varias** máquinas de estado sencillas.

Para esto tenemos que poder reconocer los “subsistemas” dentro de mi sistema principal



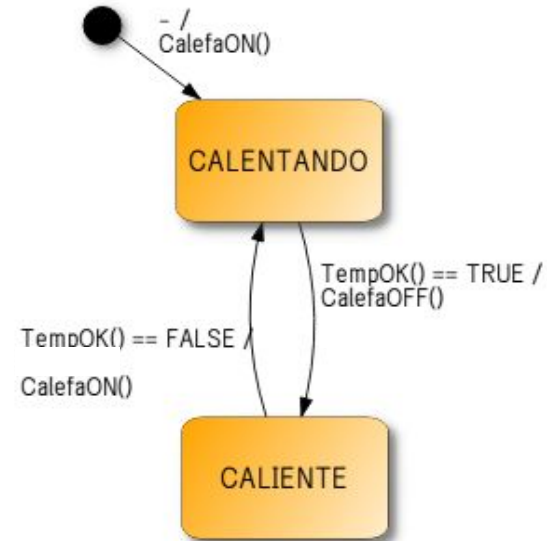
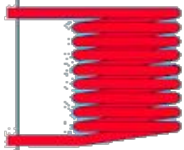
Modelizando los sistemas...



Modelizando los sistemas...

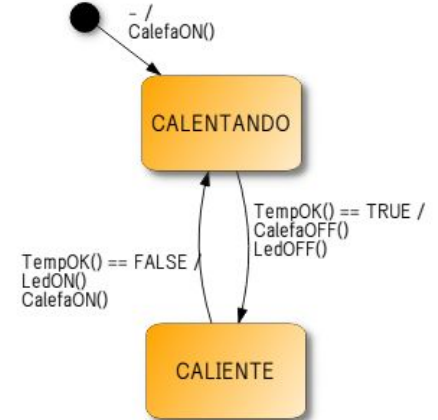
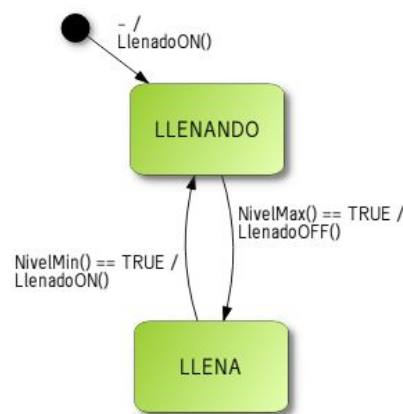
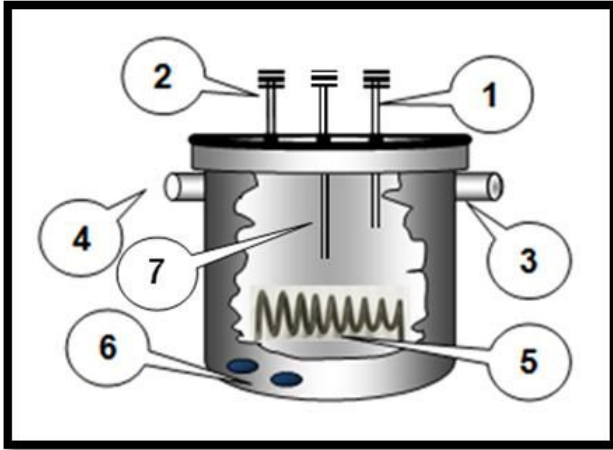
TempOK()

CalefaON()
CalefaOFF()



Un sistema como un conjunto de subsistemas...

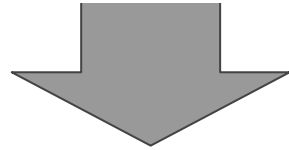
Y... ¿Cómo hacemos para que estos dos sistemas corran al mismo tiempo?



“Encapsulando” las máquinas de estado (I)

```
void main (void)
{
    Inicializar ();

    while (1)
    {
        ControlLlenado();
        ControlCalefa();
    }
}
```



Recordando...

Si **evitamos** el uso de **loops bloqueantes** dentro de las funciones, el tiempo en que el procesador está dedicado a cada una de las funciones es **MUY** pequeño (microsegundos), por lo que desde el punto de vista de los sistemas, estamos controlando los dos procesos **AL MISMO TIEMPO**.

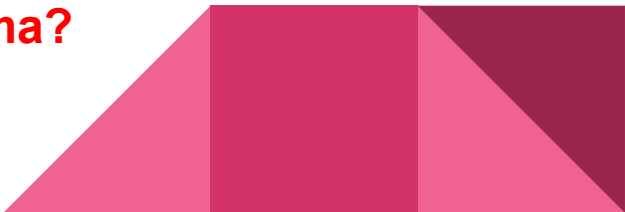
Si “encapsulamos” las máquinas de estado poniendo todo su código en una función, el programa resulta más legible

“Encapsulando” las máquinas de estado (II)

Al ampliar la complejidad de mi sistema (y por ende de mi programa), tengo que tomar los resguardos necesarios para que la máquina nunca se escape de mi control.

Esto implica en la práctica que la variable de **estado** de cada una de mis máquinas de estado nunca puede tomar un valor distinto de los diseñados

¿Cómo puedo proteger una variable dentro de mi programa, para evitar accesos indeseados a la misma?



“Encapsulando” las máquinas de estado (III)

```
void ControlLlenado ( void )
{
    static unsigned char estado = ESTADO_INICIAL_LLENADO;
    switch ( estado )
    {
        case LLENANDO:
            if ( NivelMax() == TRUE ) {
                LlenadoOFF();
                estado = LLENA;
            }
            break;

        case LLENA:
            if ( NivelMin() == TRUE ) {
                LlenadoON();
                estado = LLENANDO;
            }
            break;

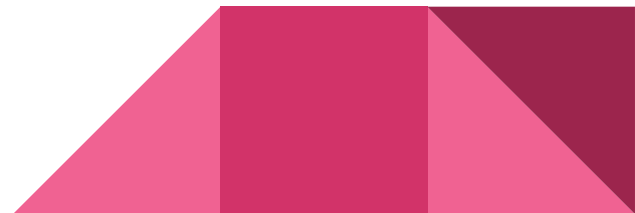
        default:
            estado = LLENA;
            LlenadoOFF();
            break;
    }
}
```

Modificador “**static**” en variables locales:

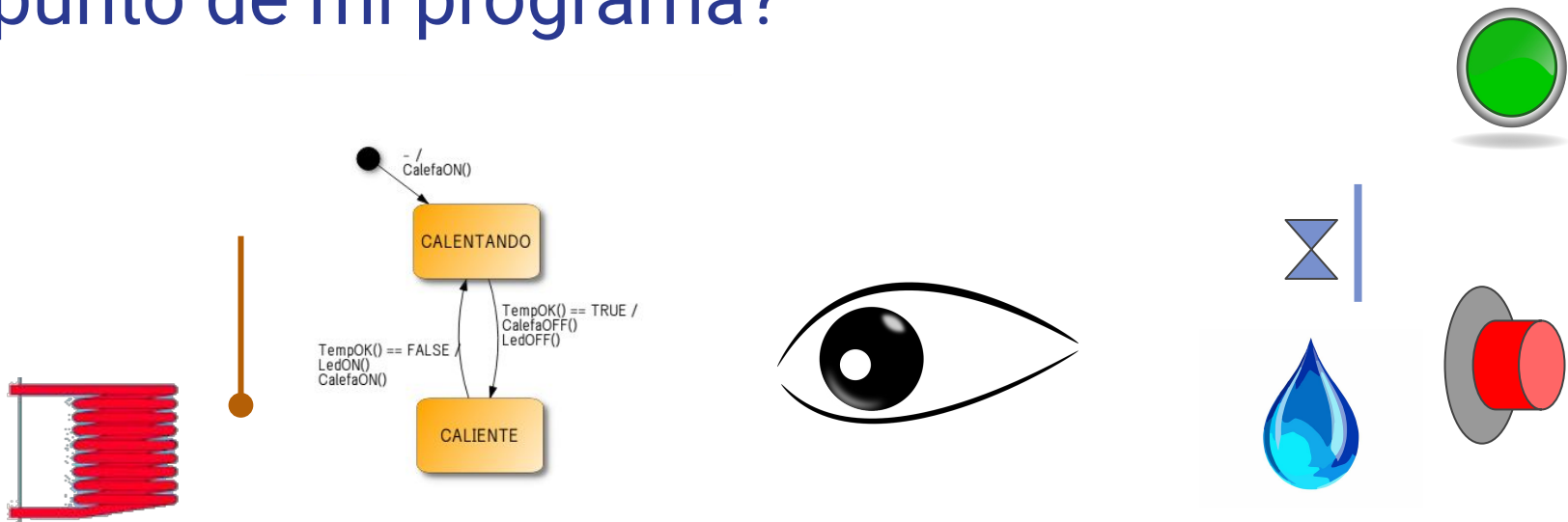
¿Cómo modifica el comportamiento de mi variable dentro de una función?

y...

¿Cuál es la visibilidad de la variable FUERA de la función?

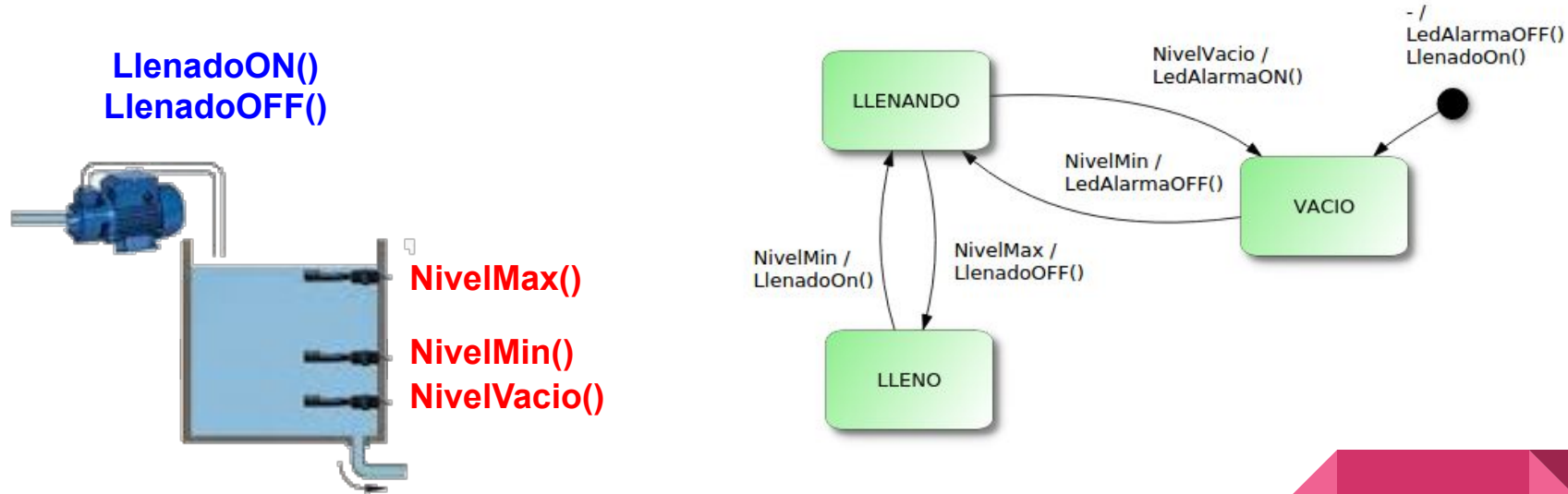


¿Qué hago si necesito ver la variable desde otro punto de mi programa?



Por ejemplo: Quiero habilitar una salida de agua caliente (una electrovalvula) solo cuando la temperatura supere un determinado valor y haya suficiente agua

¿Cómo me doy cuenta si tengo suficiente agua? Agregando estados “de emergencia”



**AGREGAR UN ESTADO DE EMERGENCIA A UNA MÁQUINA YA
DISEÑADA NO DEBERÍA GENERAR MUCHAS VARIACIONES**

Protegiendo y compartiendo variables...

```
static int estado = VACIO;
```

```
void maquina_MdeLlenado()  
{  
    switch(estado)  
    {  
        case LLENANDO:  
            if( NivelMax( ) ) {...}  
            if( NivelVacio( ) ) {...}  
            break;  
        case LLENO:  
            if( NivelMin( ) ) {...}  
            break;  
        case VACIO:  
            if( NivelMin( ) ) {...}  
            break;  
        default: estado = VACIO;  
    }  
}  
  
unsigned char HayAgua ( void )  
{  
    return estado != VACIO;  
}
```

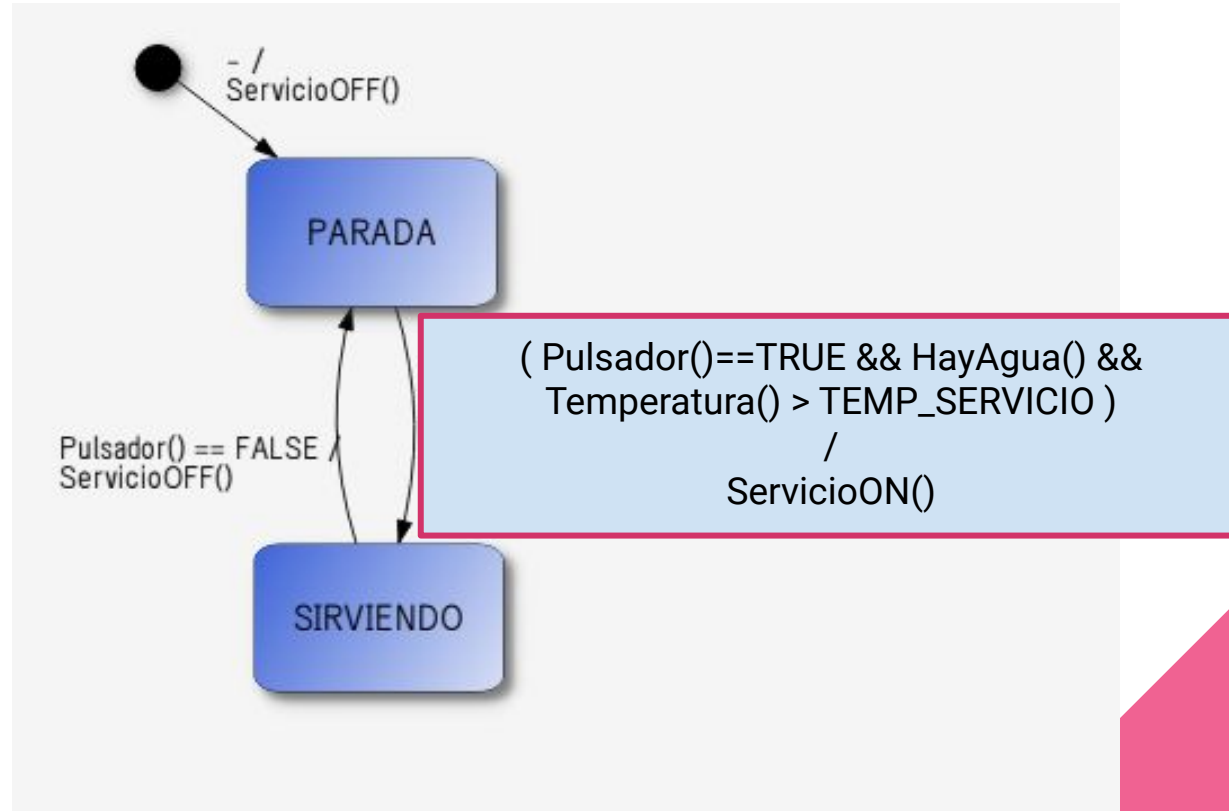
Modificador **“static”** en variables globales:

¿Cómo modifica la visibilidad de mi variable?
Cuando uso el modificador static en una variable global, esta variable solo es visible en el ámbito del ARCHIVO en donde fue creada (no la puedo ver desde otros archivos)

“Interfaz de comunicación”:

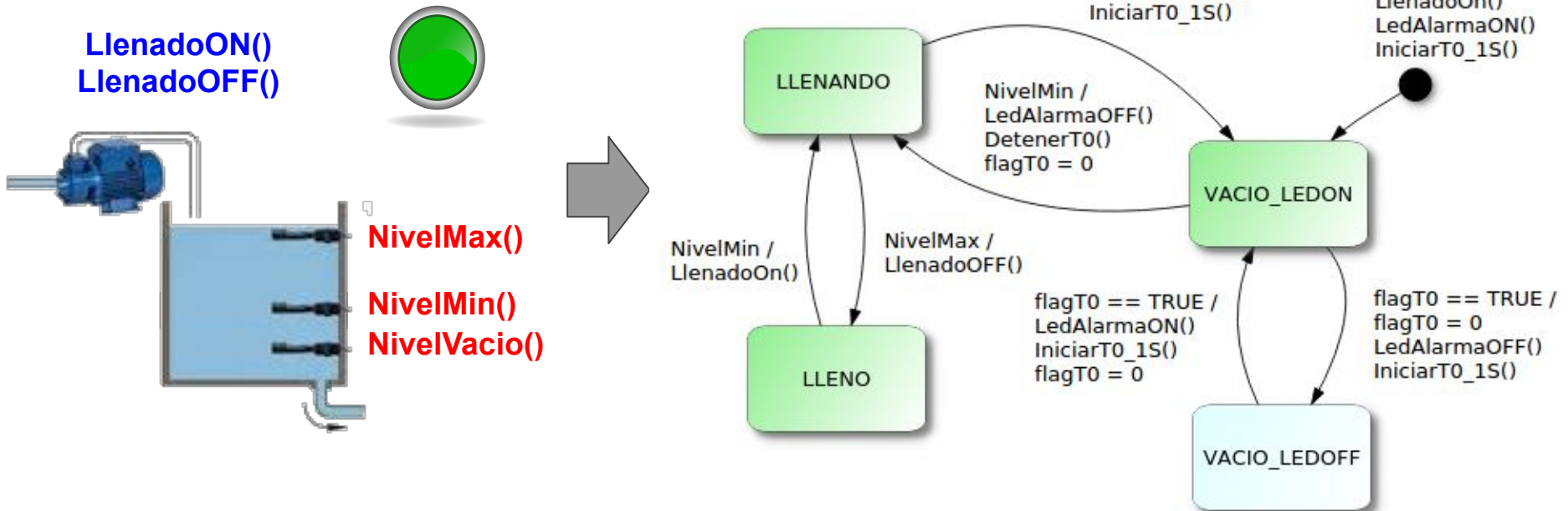
Doy visibilidad SOLO a los aspectos que yo quiero de mi variable

Variables del programa como entradas de mi MdE

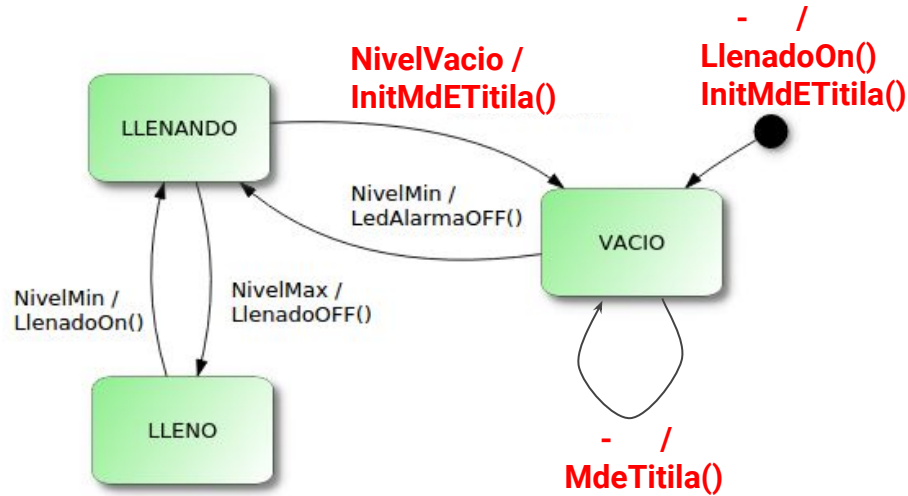


Otras estrategias

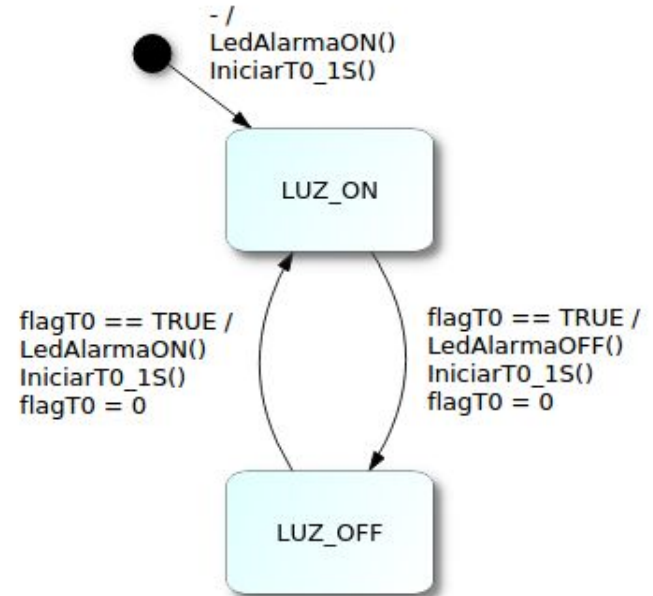
¿Qué pasaría ahora si tengo una luz que titila cuando no hay agua en el tanque?



Simplificando...



MdETitila:



No hay una sola forma de resolver las MDE

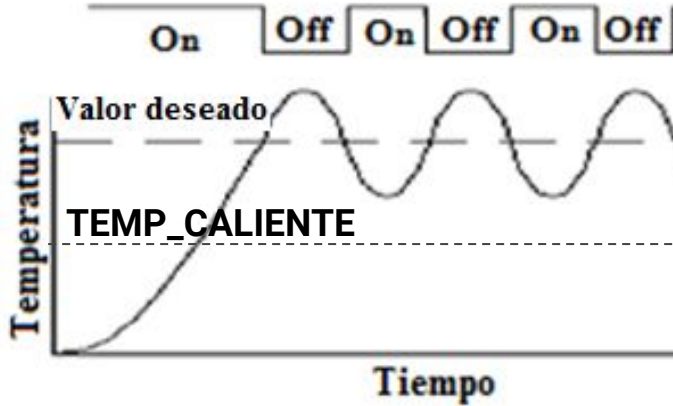
En el código...

```
switch(estado)
{
    case LLENANDO:
        ...
        if( NivelVacio( ) )
        {
            InitMdETitila();
            estado = VACIO;
        }
        break;
```

```
case VACIO:
    MdETitila();
    ...
    break;
```



Ejemplo 2: Puerto Serie



Se posee un sistema de calefacción ON/OFF, similar al que realizamos anteriormente, el cual se enciende con la opresión de una tecla (y se apaga con la misma). Cuando la temperatura supera un valor de TEMP_CALIENTE, un led debe titilar indicando el caso, y cuando se llega a la TEMP_OPERACIÓN, el calefactor debe apagarse, para volver a iniciarse al momento que la temperatura desciende de este valor.

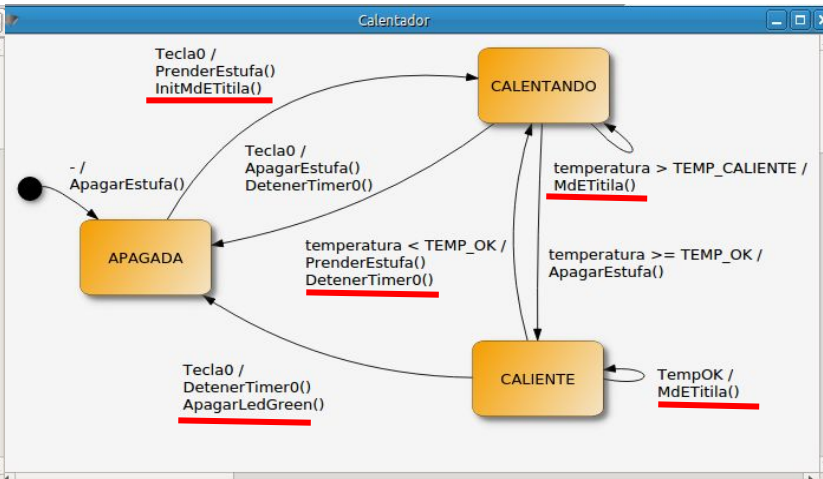
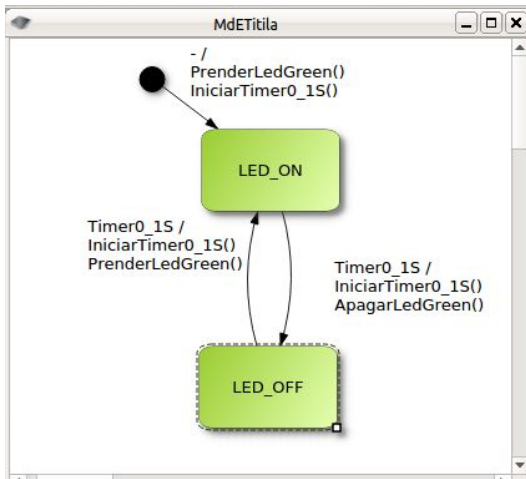
En este caso, sin embargo, el valor de la temperatura actual nos llegará por medio de un sensor que se encuentra conectado a nuestro microcontrolador por medio del puerto serie.

Mirando la hoja de datos del sensor, observamos que la información llega codificada de la siguiente manera:

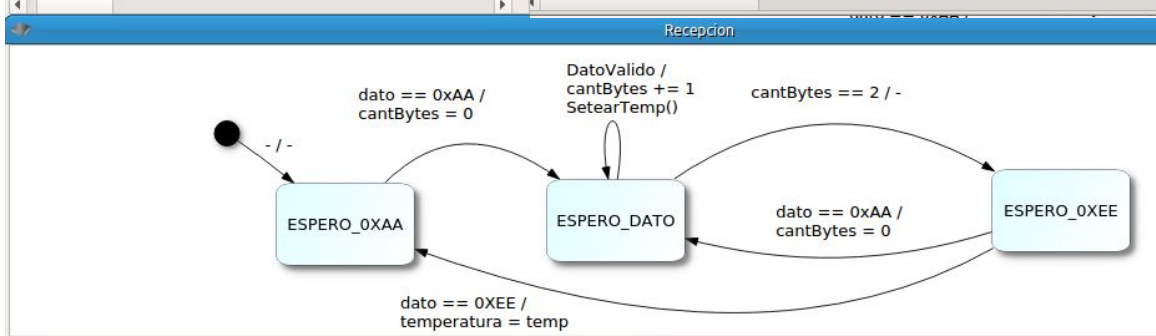


En donde cada bloque es un byte que llega desde el puerto serie. Los bytes 0xAA y 0xEE son el encabezador y la finalización de la trama, que verifican que el dato llegue correctamente y sin errores. Los datos pueden llegar en cualquier momento, y para su lectura se cuenta con la función RecibirSerie(), que devuelve 0xFF en caso de que no haya un dato nuevo, o el dato (en formato de un byte) en caso de que haya un dato nuevo para leer en el puerto

Realizar las máquinas de estado que resuelvan dicho sistema, e implementarlas en un programa que las invoque de manera que su ejecución se realice en paralelo



Dentro de los estados CALENTANDO y CALIENTE se llama a la MdETitila(), hay que verificar que cuando se pase a este estado se hagan las acciones correspondientes al RESET de esta Máquina de Estados



```

void main ( void )
{
    Inicializar();
    while ( 1 )
    {
        Calentador();
        dato = RecibirSerie();
        if ( dato != -1 )
        {
            Recepcion ( dato );
        }
        TimerEvent();
    }
}
  
```

A la MdE Recepción solo se la llama si hay un dato válido, y en caso de que se cumpla con la correcta recepción de la trama, puede cargar un valor global temperatura con el valor recibido, o devolverlo a una variable local