RETO 2

SISTEMA DE ALARMA DOMÉSTICA

ISEL 2020/2021

Alfonso Botas Rodríguez

Luis De Pablo Beltrán

Belén Vega Castrillo

ÍNDICE

1. INTRODUCCIÓN

El segundo reto de la asignatura consiste en la implementación de un sistema de alarma doméstica que, mediante un sensor de movimiento, detecte presencia y genere una señal de alarma.

Además de dicho sensor, el sistema contará con un pulsador para introducir un código que debe coincidir con uno predefinido por el usuario previamente, además de unas luces que se activan al detectar presencia.

Para ello, utilizaremos la metodología propuesta para el desarrollo de soluciones, realizando los pasos necesarios del diseño orientado a modelos:

- Especificación en LTL.
- Modelado de los comportamientos con máquinas de estados.
- Verificación formal del modelo.
- Implementación semiautomática.
- Análisis de tiempos

2. ESPECIFICACIÓN LTL

Para la especificación LTL, hemos creado tres ficheros .pml que se dividen en los distintos pasos a seguir en el reto: Alarma básica, código y luces.

2.1. Alarma básica

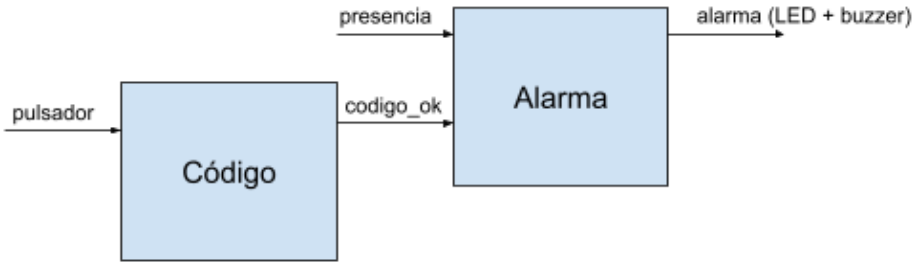
En primer lugar, implementamos la alarma básica, la cual tiene como entradas la presencia que puede detectar el sensor de infrarrojos (PIR) y un pulsador para activar y/o desactivar a alarma, como única salida tiene la alarma (activada o desactivada) la cual será un LED.

Las propiedades que hemos tenido en cuenta a la hora de la especificación son las siguientes:

- Si la alarma está activa y se detecta presencia, la alarma suena.
- Si la alarma no está activa, la alarma no suena.
- Si la alarma no está activa y se presiona el pulsador, la alarma se activa.
- Si la alarma está activa y se presiona el pulsador, la alarma se desactiva.

2.2. Código con un pulsador

A continuación, utilizando el pulsador como entrada para introducir el código, evaluamos la correcta introducción de este para activar o desactivar la alarma. Como salida tiene la alarma (activada o desactivada) la cual será un LED.

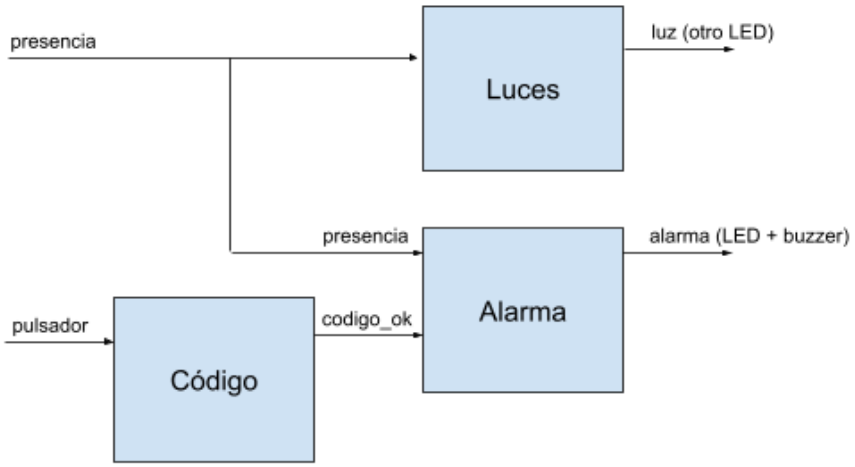


Las propiedades que hemos tenido en cuenta a la hora de la especificación son las siguientes:

- Cada nueva pulsación del código incrementa el dígito actual.
- Una espera de más de 1 segundo desde la última pulsación en ese dígito avanza al siguiente dígito.
- Una espera de más de 10 segundos vuelve al estado inicial.
- Un cero se introduce con 10 pulsaciones.
- Cuando se han introducido todos los dígitos, si el código corresponde al de activación, se activa la señal codigo_ok
- En cualquier momento, si espero 10 segundos sin pulsar e introduzco el código correcto, la señal codigo_ok se activa.
- En cualquier momento, si espero 10 segundos sin pulsar e introduzco el código incorrecto, la señal codigo_ok no se activa.

2.3. Control de luces

Por último, añadimos un control automático de luces el cual utilizará el sensor de movimiento como entrada y como salida el led (activado o desactivado)



Las propiedades que hemos tenido en cuenta a la hora de la especificación son las siguientes:

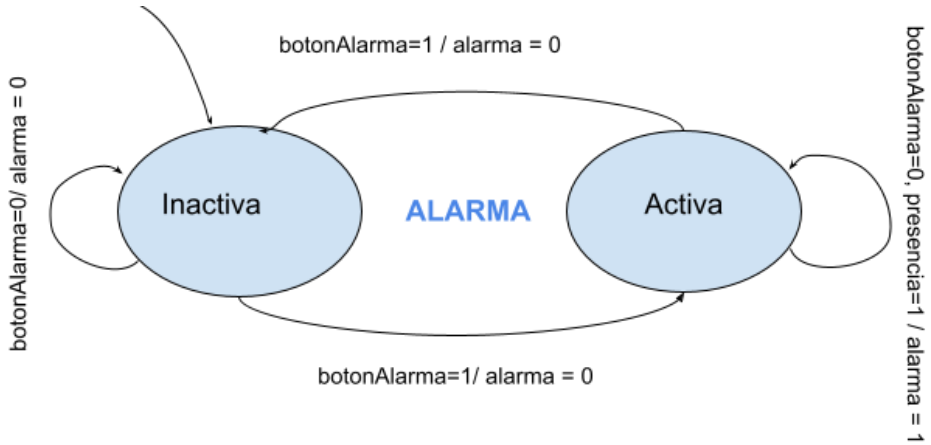
- Si hay presencia, la luz se enciende.
- Si han pasado 30 segundos desde la última vez que se detectó presencia, la luz se apaga.
- En cualquier momento, si pasan 30 segundos sin presencia, la luz se apaga.
- En cualquier momento, si se detecta presencia, la luz se enciende.

3. MODELADO

Adjuntamos a continuación as máquinas de estado Mealy que hemos creado y que forman el sistema.

3.1. Alarma Básica

- Entradas
  - botonAlarma: Se trata de un pulsador que activa o desactiva la alarma
  - presencia: Sensor para detectar presencia y se active la señal de alarma
- Salidas
  - alarma: señal de alarma, LED

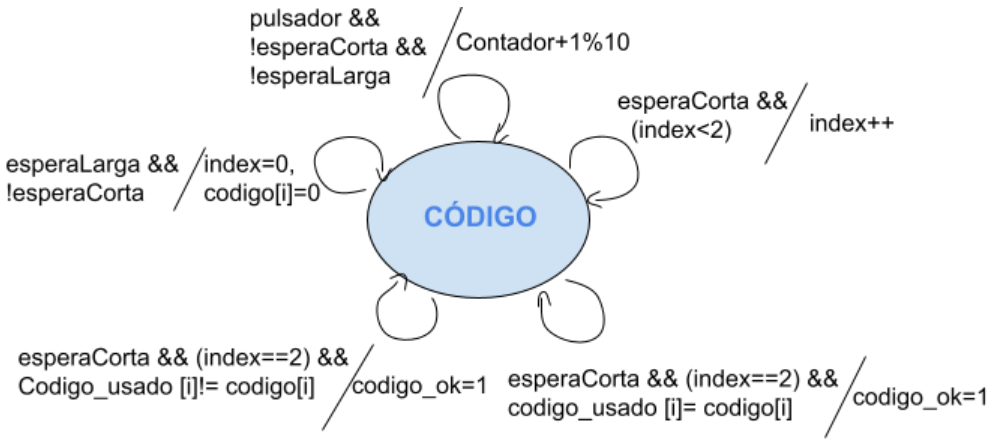


3.2. Código con un pulsador

Máquina de estados que modela el funcionamiento del manejo del código para activar y desactivar la alarma. Consiste en una máquina de estados extendida, lo que nos facilita su desarrollo y posterior implementación.

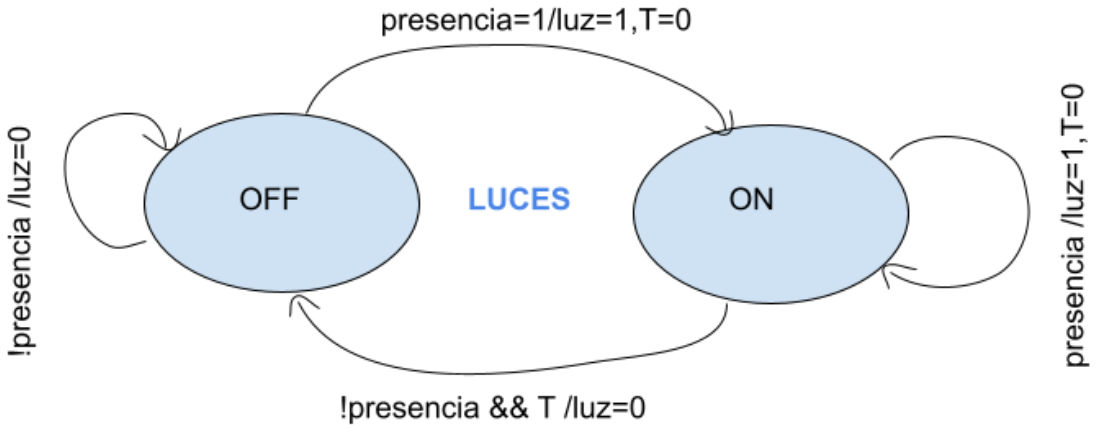
- Entradas
  - pulsador: aumenta el valor del dígito que se quiere cambiar
  - esperaCorta: tiempo de espera para cambiar de dígito, si ya se han escrito los 3 dígitos, tiempo de espera para comprobar el código
  - esperaLarga: tiempo de espera para resetear el código
- Salidas
  - codigo_ok: Señal que indica si el código puesto es correcto o no

- Variables
    - Contador: aumenta cada vez que se pulsa el pulsador en el dígito en el que está
    - Index: indica en qué digito nos encontramos



3.3. Control de luces

- Entradas:
    - Presencia: la misma que en Alarma Básica
- Salidas
    - Luz: se enciende o apaga según haya presencia o no
- Variable
    - T: Tiempo de espera para apagar la luz si no hay presencia



4. IMPLEMENTACIÓN

Usaremos el ESP32 para

El periodo elegido es de 50ms, es suficientemente grande para que se pueda implementar, pero lo hemos ajustado de tal manera que no parezca un programa lento.

Hemos implementado el modelo de tres maneras distintas:

- Ejecutivo cíclico (main_ec.c)
- Thread (main_thread.c)
- Reactor (main_reactor.c)

La asignación de prioridades la hemos hecho de manera que el control de luces tenga la máxima prioridad para que si hay una presencia se activen las luces lo más rápido posible. La será la segunda más prioritaria porque interesa que la alarma actué también rápido. Por ultimo, la gestión del código de la alarma será la tarea menos prioritaria, ya que las pulsaciones del botón suelen ser mas lentas.

Tenemos dos recursos compartidos, la entrada “presencia” que lo comparten las tareas de alarma y luces, y la señal “codigo_ok” que lo comparten las tareas de alarma y código.

Usaremos prioridades fijas con desalojo y herencia de prioridad. No usamos techo de prioridad, ya que usamos un ESP32 y no dispone de dicho protocolo.

5. VERIFICACIÓN

Hemos pasado la especificación LTL y el modelado de las máquinas de estados a Promela. Después, lo hemos verificado con Spin y comprobado que funciona correctamente.

El lenguaje de las máquinas de estados está contenido en el lenguaje Promela, ya que estamos usando máquinas de estados deterministas. Por lo tanto, la segunda verificación no es necesaria.

6. ANÁLISIS DE TIEMPO

En este apartado vamos a analizar la planificabilidad de cada una de las implementaciones que hemos mencionado. En todas las tareas hemos seleccionado un periodo de 50 ms, ya que es lo suficientemente grande para que todas las tareas se puedan ejecutar y es lo suficientemente pequeño para que el sistema responda rápidamente.

Para calcular los tiempos de ejecución máximos, hemos medido el tiempo justo antes y justo después de cada tarea. Guardamos ese valor y lo imprimimos por el monitor serie. Hemos obtenido el tiempo máximo de 1000 pruebas, quedándonos con la mayor. En todas las iteraciones ese tiempo ha sido 0, lo que equivale a 0 Ticks. Eso es porque la tarea se ejecuta en el mismo Tick. Por lo tanto, nos quedamos con un valor más pesimista que es 1 Tick, lo que equivale en el ESP32 a 10 ms.

6.1. Ejecutivo Cíclico

Para hacer el análisis se ha tenido en cuenta el caso peor, que es cuando la activación de todas las tareas ocurre en el mismo tiempo, en nuestro caso en t=0.

La duración máxima de las tareas es como se ha dicho antes Ci=10ms, con Di=Ti=50ms. Todo esto se muestra en la siguiente tabla:

tarea		C	T	D	P
t1	Luces	10	50	50	3
t2	Alarma	10	50	50	2
t3	Código	10	50	50	1

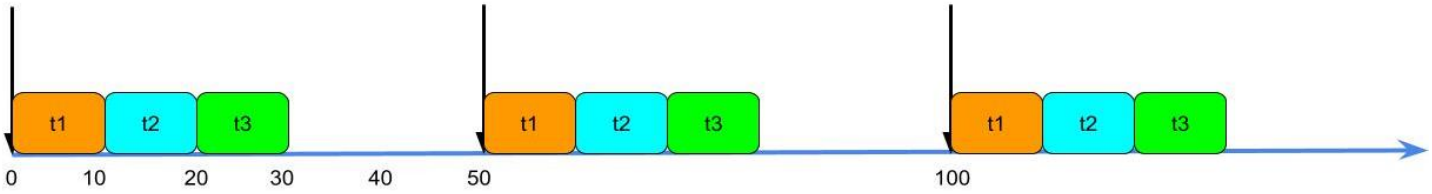
Las tareas corresponden a las siguientes maquinas de estados.

- t1: tarea de control de las luces de la alarma
- t2: tarea de control de la alarma
- t3: tarea de control del código de la alarma

Se puede comprobar que tanto el hiperperiodo como los ciclos secundarios son iguales:

$$T_M = T_S = 50\ ms$$

Para hacer el análisis de tiempos hemos dibujado el diagrama de ejecución del ejecutivo cíclico.



Como puede apreciarse, se cumplen los plazos en todas las tareas, por lo tanto, con esta implementación el sistema es **planificable**.

6.2. Thread

Para hacer el análisis hemos tenido en cuenta el caso peor, donde la duración de los recursos compartidos es la misma que el tiempo de ejecución de la tarea.
La duración máxima de las tareas es como se ha dicho antes Ci=10ms, con Di=Ti=50ms. Todo esto se muestra en la siguiente tabla:

tarea	C	T	D	P	B	R	I	r1	r2

t1	Luces	10	50	50	3	20	30	0	10	-
t2	Alarma	10	50	50	2	10	30	10	10	10
t3	Código	10	50	50	1	0	30	20	-	10
									3	2

Las tareas corresponden a las siguientes maquinas de estados.

- t1: tarea de control de las luces de la alarma
- t2: tarea de control de la alarma
- t3: tarea de control del código de la alarma

Los recursos compartidos entre las FSM son:

- r1: es la entrada del sistema "presencia" que comparten la tarea 1 y la tarea 2
- r2: es la entrada del sistema "codigo_ok" que comparten la tarea 2 y la tarea 3

Para hacer el análisis de tiempos calculamos el tiempo máximo de respuesta con bloqueos y calculamos la interferencia que sufre cada una.

$$W_i^{n+1} = C_i + B_i + \sum_{j \in hp(i)} \left\lceil \frac{W_i^n}{T_j} \right\rceil * C_j \;\; y \; I_i = R_i - C_i - B_i$$

$$W_1^0 = C_1 + B_1 = 10 + 20 = 30 = R_1$$

$$I_1 = R_1 - C_1 - B_1 \; = 30 - 10 - 20 = 0$$

$$W_2^0 = C_2 + B_2 + 1 * C_1 = 10 + 10 + 10 = 30$$

$$W_2^1 = C_2 + B_2 + \left\lceil \frac{W_2^0}{T_1} \right\rceil * C_1 = 10 + 10 + 10 = 30 = R_2$$

$$I_2 = R_2 - C_2 - B_2 = 30 - 10 - 10 = 10$$

$$W_3^0 = C_3 + B_3 + 1 * C_1 + 1 * C_2 = 10 + 0 + 10 + 10 = 30$$

$$W_3^1 = C_3 + B_3 + \left\lceil \frac{W_3^0}{T_1} \right\rceil * C_1 + \left\lceil \frac{W_3^0}{T_2} \right\rceil * C_2 = 10 + 0 + 10 + 10 = 30 = R_3$$

$$I_3 = R_3 - C_3 - B_3 = 30 - 10 - 0 = 20$$

Como se puede comprobar los tiempos de respuesta de todas las tareas son menores a su plazo, por lo tanto, con esta implementación el sistema es **planificable**.

6.3. Reactor

Para hacer el análisis hemos tenido en cuenta el caso peor, donde la duración de los recursos compartidos es la misma que el tiempo de ejecución de la tarea.

La duración máxima de las tareas es como se ha dicho antes Ci=10ms, con Di=Ti=50ms. Todo esto se muestra en la siguiente tabla:

tarea		C	T	D	P	B	R	I	r1	r2
t1	Luces	10	50	50	3	20	30	0	10	-
t2	Alarma	10	50	50	2	10	30	10	10	10
t3	Codigo	10	50	50	1	0	30	20	-	10
									3	2

Las tareas corresponden a las siguientes maquinas de estados.

- t1: tarea de control de las luces de la alarma
- t2: tarea de control de la alarma
- t3: tarea de control del código de la alarma

Los recursos compartidos entre las FSM son:

- r1: es la entrada del sistema “presencia” que comparten la tarea 1 y la tarea 2
- r2: es la entrada del sistema “código_ok” que comparten la tarea 2 y la tarea 3

Para hacer el análisis de tiempos calculamos el tiempo máximo de respuesta con bloqueos y calculamos la interferencia que sufre cada una.

$R_i = W_i + F_i$

$W_i^{n+1} = (C_i - F_i) + B_i + \sum_{j \in hp(i)} \left\lceil \frac{W_i^n}{T_j} \right\rceil * C_j \;\; y \; I_i = R_i - C_i - B_i$

$W_1^0 = (C_1 - F_1) + B_1 = 0 + 20 = 20 = W_1$

$R_1 = W_1 + F_1 = 20 + 10 = 30$

$I_1 = R_1 - C_1 - B_1 \; = 30 - 10 - 20 = 0$

$W_2^0 = (C_2 - F_2) + B_2 + 1 * C_1 = 0 + 10 + 10 = 20$

$W_2^1 = (C_2 - F_2) + B_2 + \left\lceil \frac{W_2^0}{T_1} \right\rceil * C_1 = 0 + 10 + 10 = 20 = W_2$

$R_2 = W_2 + F_2 = 20 + 10 = 30$

$I_2 = R_2 - C_2 - B_2 = 30 - 10 - 10 = 10$

$W_3^0 = (C_3 - F_3) + B_3 + 1 * C_1 + 1 * C_2 = 0 + 0 + 10 + 10 = 20$

$W_3^1 = (C_3 - F_3) + B_3 + \left\lceil \frac{W_3^0}{T_1} \right\rceil * C_1 + \left\lceil \frac{W_3^0}{T_2} \right\rceil * C_2 = 0 + 0 + 10 + 10 = 20 = W_3$

$R_3 = W_3 + F_3 = 20 + 10 = 30$

$I_3 = R_3 - C_3 - B_3 = 30 - 10 - 0 = 20$

Como se puede comprobar los tiempos de respuesta de todas las tareas son menores a sus plazos, por lo tanto, con esta implementación el sistema es **planificable**.