Sistema de Alarma Domiciliaria Esquema Sw de Aplicación

Sistemas Embebidos Avanzados
Eduardo Daniel Cohen
dcohen@herrera.unt.edu.ar
http://www.microprocesadores.unt.edu.ar/embebidosavanzados

Temario

- Matriz de Percepción.
- Interacción entre Actividades.
- Esquema de una Actividad.
  - Hilos.
  - Timers

Matriz de Percepción.

- Matriz de 8x8. MP.
  - Enmascara las entradas.
  - Con operación AND.
  - El sistema consulta la matriz de transición resultante – cuando pasa de 1→2 o 2→3.
- Estado 1.
  - Ningún sensor válido. Teclas F. Forzar-P o Armar-P no.
  - Teclas BORRAR y Teclas numéricas.
- Estado 2.
  - Sensores de fuego y desarme.
  - Todas las teclas habilitadas.
- Estado 3.
  - No ver sensores temporales por un tiempo.
  - Luego ver todos los sensores.
    - Correspondientes al modo de armado.
  - No ver teclas Forzar-P, Armar-P, F.

Armado de Percepción

- En ROM matrices de percepción:
  - Matriz Estado 0.
  - Matriz Estado 1.
  - Matriz Estado 2.
  - Matriz solo con los sensores transitorios (MST)
  - Matriz temporal Estado 3 ausente (MT3A).
  - Matriz temporal Estado 3 presente.(MT3P)
  - **¿y Matriz temporal forzada?**
    - Ya se vió en tema anterior.
  - Cuando hay un cambio de estado se genera MP correspondiente.
  - Ej: Armado Ausente.
    - MP = MT3A. Disparar timer.
    - Con timer: MP = MT3A OR MST
  - Lo mismo para Armado Presente.

Percepción, continuación.

- Cuando se pasa de estado 2 al 3.
  - Armar matriz de percepción sin sensores retardados.
  - Al cabo de un tiempo se pasa de esta matriz de percepción temporal a una definitiva, según el tipo de armado que sea.
  - Revisar antes que no haya sensores activos a percibirse en el estado 3.

Pase de Estado 3 al 2.

- Cuando se pasa de estado 3 a 2.
  - No se cambia la matriz de percepción hasta que ingrese la clave correcta.
  - Se dispara un timer (timer-R).
  - Si no se teclea la clave antes de que salte el timer, sonará la alarma.
  - Si salta otro sensor retardado no se debe disparar el timer de nuevo.
  - Cada sensor temporalque salta antes de timer, provoca que se guarde su número.
  - Si salta sensor permanente guarde núm. → alarma, abortar timer.

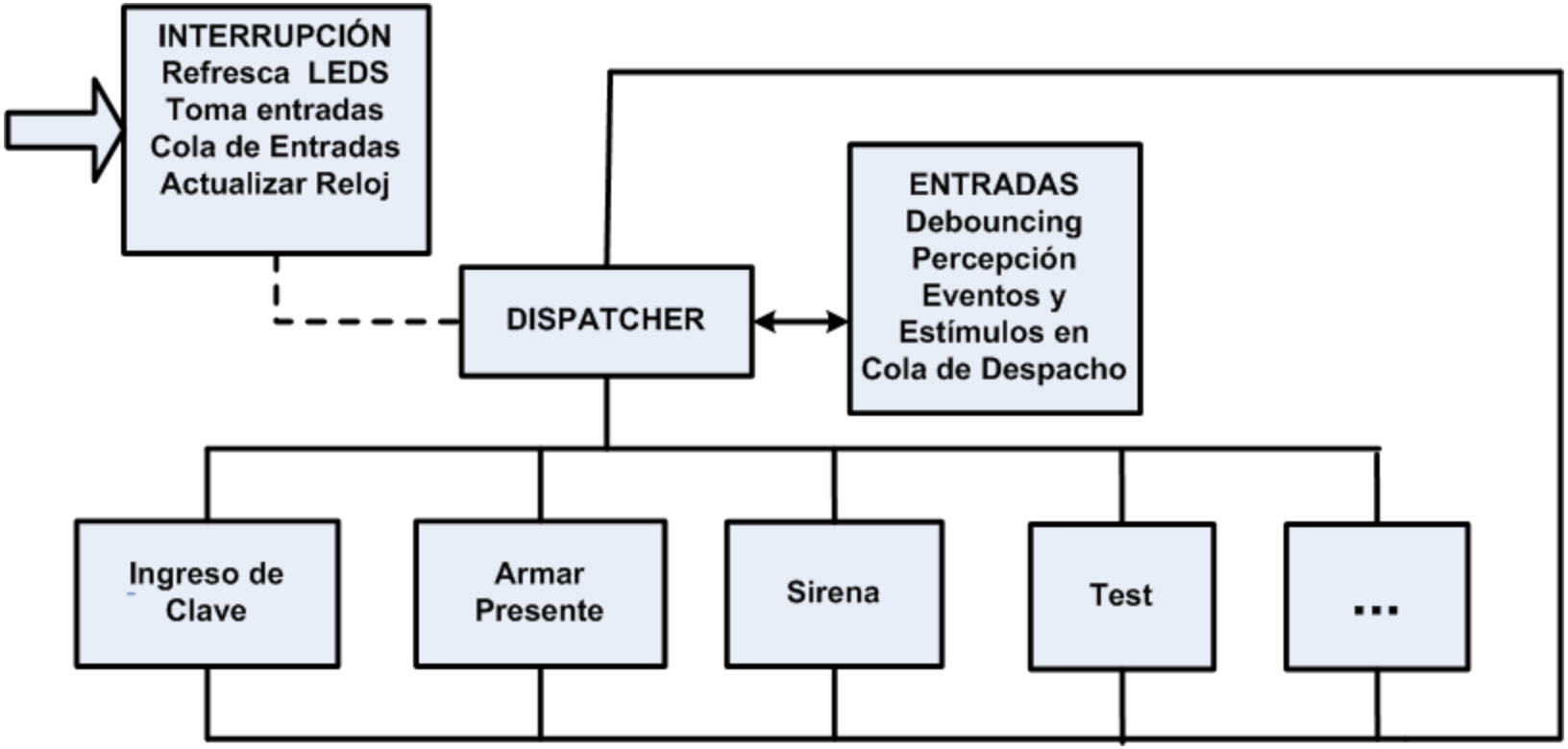
Pase de 3 a 2 (cont)

- Si salta el timer.
  - Suena la alarma.
  - Se copian los num sensores guardados a matriz U-ESTADO, sin borrar lo que hubiera allí de antes.
  - Todo sensor transitorio se trata como definitivo
- Si se marca la clave correcta
  - Se cambia a matriz percepción estado 2.
  - Se borran los sensores retardados guardados.
  - Abortar timer.

ACTIVIDADES

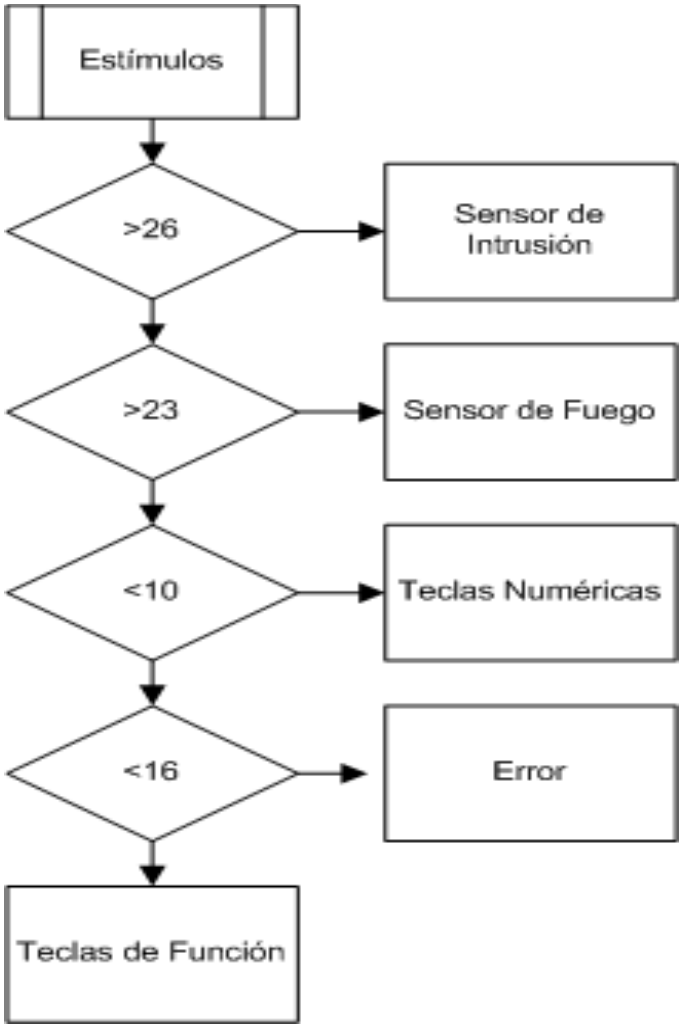
- A cada actividad incorporar revisión de interacción otras actividades.
  - Recordar Tabla de Interacción.
  - Primero controlar si puede iniciarse.
  - Luego abortar a las que correspondan.
- Sw de Base.
  - Provee rutinas:
    - ESTADO(id_tarea, st): st=0 wait, st=1 sleep.
    - ABORTAR(id_tarea).
    - ESTADO SENSORES(es, modo): es=1 hay al menos uno activo (para controlar que el sistema se pueda armar).
    - Funciones para cambiar las salidas.
    - PROCESAR – para debouncing, conversión, etc.
  - Cada tarea
    - Devuelve su propio indicador de estado a Dispatcher.
    - Lleva su propia rutina para abortarla. ¿Por qué?

Esquema Gral del Hw.: Hilos.



En Dispatcher hay un salto múltiple, esto es solo un esquema Parcial para uso didáctico.. Falta inicialización, por supuesto. Para los eventos se salta primero a la tarea y allí al evento por id

Esquema Tabla de Saltos para Estímulos



Es un esquema general
El salto es directo, no por preguntas.

Esquema Respuesta a Eventos

- ENTRADA encola mensajes:
  - Variable lógica que indica que es evento.
  - ID Actividad.
  - ID timer dentro de la actividad.
- Tabla de Saltos a actividades.
- Al comenzar actividad, si es timer
  - Tabla de saltos a puntos de continuación.
- Cada actividad fija sus timers contando desde 0. También su tabla

Tabla de Saltos

1. Una entrada por cada estímulo (64)
   - Dirección procedimiento por cada una.
   - En ROM.

Pregunta...

- Tecla Función:
  - Solo cuando se pasa a estado 3.
- Desde el estado 1.
  - **¿Hay que validar el estado para tomarla?**
- Armado Forzado, al tomar la /MEL enmascarar las entradas de teclado
  - Para que no quede una tecla activa forzada por error.

Tareas

- Las tareas se implementarán como funciones.
- La idea es que sean autocontenidas
  - Con variables locales salvo pocas excepciones
- Se la programa como una función separada que se llama desde el despachador o desde otra tarea.

Ejemplo: algunas funciones.

- Titilar un Led.
- Testear Leds.
- Estado.
- Todos están relacionados con timers.
- Se activan desde la lógica de la aplicación.
- Y también en respuesta a evento (timer).
  - Cambian de estado (titilan).
  - Se apagan (caso test y estado).

Funciones del Sistema

- El Software de base (ENTRADA) provee servicios a través de funciones
- Para aislar las variables del sistema de las tareas.
  - Para lograr mayor simplicidad en tareas..
  - También, como una futura mejora
    - Verificar que las funciones sean llamadas solo por algunas tareas y no otras.
    - Protección.

Funciones para timers

- setTimer(id,i,t)
  - La tarea “id” incorpora un timer “i” a la cola de eventos para que se dispare dentro de “t” décimas de segundos.
  - Revisa previamente si este timer de esa tarea está en cola de eventos o despacho.
    - En ese caso lo aborta y luego lo inserta con el valor adecuado y devuelve un código de respuesta 1.
    - Caso contrario entrega un código de respuesta 0.
- abortTimer(id,i)
  - Si se encuentra el timer en la cola lo elimina.
    - Tanto en **cola de eventos como de despacho**.
    - Código de respuesta es 1.
  - Caso contrario
    - Código de respuesta es 0.

Funciones Leds (Sw Base)

- putLed(k,st) ‘ cambia estado del led.
  - st=1 → led k se enciende
  - st=0 → led k se apaga.
- getLed(k,st) ‘ estado led k.
  - 1 si está encendido.
  - 0 si está apagado.
- putModo(k,st) ‘modo del led (solo informativa)
  - 1 titilar, 0 no titilar.
- getModo(k,st)
  - 1 titila, 0 no titila.
- Los estados del led requieren 2 bytes por led.
  - Correspondencia de k, 1 a 1 con matriz de salidas.

Ej: Titilar led k.

- Los leds se ubican en filas 3 a 7 en matriz de salida.
- Hay 6 leds indicadores de función.
  - Podrían Titilar.
  - Leds 16 a 21.
    - Led 16 corresponde a k0 y así sucesivamente.
    - Si se cablea diferente, el sistema realiza el mapeo
      - De lógico a físico.
- titilarLed(16,50,i,id) 'led 16
  - Debe titilar el led 16 durante 50 dec. de seg.
  - Emplear el timer i
  - Llamado desde la tarea id.

Ejemplo: titilarLed(k,50,i,id)

1. setTimer(id,i+1,50) ‘5 seg se apaga
2. getLed(k,st) ‘entrada evento i y llamado
3. st=/st
4. setTimer(id, i, 5) ‘0,5 seg. para cambio
5. putLed(k,st)
6. Return

- Cuando llega timer i, entra en 2.
- **¿Por qué el paso 2?**
- Antes de llamar, en la tarea:

  putModo(k,1), get(k,st1) ‘estado previo led

noTitilar(k,st,id,i)

1. abortTimer(id, i+1) ‘convocada por tarea
2. abortTimer(id, i) ‘pasaron 0,5 seg
3. setModo(k,0) ‘no titila
4. setLed(k,st) ‘estado previo
5. Return

- Cuando expira el timer retorna a un punto en la tarea en que se convoca a noTitilar(k,st1,id,i)
- NoTitilar puede usarse también cuando se desee que deje de titilar.

Cuidado

- No emplear variables globales.
  - El procedimiento se puede usar para varios leds al mismo tiempo y se pierde la variable global.
- El único dato que requiere lo toma directamente con getLed.
- Un flag no hubiera funcionado (local se pierde, global también).

Alternativa para Titilar

- ¡Se trata de una función de I/O!
- ¿por qué la debe manejar la aplicación?
  - Que la maneje la interrupción.
  - Cada 0,5 seg que prenda o apague.
  - Cuando venza el tiempo de titilar, la tarea llama a no-titilar.
- Conclusión:
  - aplicación más sencilla.
  - En este caso no se complica casi nada la interrupción. Caso contrario pensar de nuevo

Manejo de Leds (1)

- Si un led puede estar en 3 estados, se necesitan 2 matrices de salida.
- Matriz 1 – estado actual.
- Matriz 2 – próximo estado.
- Cuando el led k está apagado, su posición está en 0 en las dos matrices.
- Si está prendido, pasa lo contrario.
- Si titila, las posiciones están en 1 y 0.

Manejo de Leds (2)

- setLed(k,valor) – Redefinimos la función.
  - Valor:
    - 0 - 0 en ambas matrices.
    - 1 – 1 en la primera, 0 en la segunda
    - 2 – 0 en la primera, 1 en la segunda.
    - 3 – 1 en ambas matrices
- Cada 0,5 seg la interrupción alterna entre matriz1 y matriz2.
- Valores 1 y 2 dan leds que titilan en contrafase.
- Para cambiar funciones de led 31.
  - setLed(31,0) apaga el led 31.
  - setLed(31,1) titila el led 31.

Funciones para titilar

Titilar(k,t,id,i)

1. setLed(k,1)
2. setTimer(id,i,t) ‘t segundos titila
3. Return

NoTitilar(k,id,st,i) ‘st estado previo de led

1. abortTimer(id,i) ‘llama tarea
2. setLed(k,st) ‘evento i
3. Fin

Ej: Llamado

getLed(30,st)

Call Titilar(30,600,id,i) ‘un minuto

….

Call noTitilar(30,id,st,i)

…

- st es el estado en que estaba antes de titilar, lo conoce la tarea que llama.

Otras funciones del Sistema

- getMELP(k) – entrega MELP a partir del puntero k.
- Sirve para tener una copia de la matriz.
- callTarea(k1,k2,msg) –
  - Tarea k1 llama a tarea k2.
- getOutput(k) –matrices de salida en k.
- putOutput(k) – pone en matrices de salida la que están a partir de k.
- Entregan matrices lógicas de salida

Ej. PRUEBA

- Chequear tabla interacción de actividades.
  - Revisar primero si puede arrancar.
  - Luego si hay tareas a abortar y hacerlo.
- No se debe perder el estado de los leds de función.
- Se los respalda previamente
- Se usarán variable globales (sin problema pues hay una sola tecla para esta función).
- Las rutinas PRUEBA y ESTADO
  - Responden a la tecla correspondiente.
  - Y al timer correspondiente.

Ejemplo: pruebaLeds

Test inicialmente 0.

1. test = /test (tecla toggle) ‘variable global
2. If test = 1
   1. getOutput(k) ‘backup a partir de k
   2. Poner 1 en todos los leds ‘2 matrices desde k1.
      - putOutput(k1)
   3. setTimer(id_tarea, 3, 600). ’1 min. Timer 3.
3. Else ‘se oprime la tecla de nuevo (toggle)
   1. putOutput(k) ‘recupera estado anterior salida
   2. abortTimer(id_tarea, 3).
4. FIN.

Terminó el timer

1. putOutput(k)
2. test=0
3. Fin

- Para abortar esta tarea **¿cuándo?**:

1. putOutput(k)
2. test=0
3. abortTimer(id_tarea,3)
4. Fin

Pregunta

- ¿En que estado queda la tarea cuando se encola el evento y retorna?
- ¿Y en qué estado cuando responde al evento?
- ¿Cuándo retorna y no queda ningún evento?

Ejercicio

- Seudoalgoritmo: la percepción de una tecla oprimida invierte el led "presente" por 1 segundo.
  - Cuál es la tarea que debe convocar a esta función?
  - Se usará el led "presente", denominado k.
  - Se supone que el led no está titilando.

Invertir Led con tecla.

Desde el programa. ‘ if Tecla=0 then call InvertLed

invertLed:

1. setTimer(id, 6, 5) ’ 0,5 seg.
2. getLed(k,st)
3. If st=0 then st=3 else st=0
4. putLed(k,st)
5. Tecla=/Tecla
6. Fin ‘retorna a la tarea y continúa la misma.

Con el evento, entrar en la misma rutina desde (2).

¿Si se oprime otra tecla antes del cambio?

Ejemplo: Estado

Se asigna un timer, ej: 4. Inicialmente Estado=0

1. Estado = /Estado (tecla toggle)
2. If Estado = 1
   1. getMEL(k)
   2. getOutput(k1)
   3. Copiar matrices k1 en k2.
   4. Pasar últimas filas de sensores de MEL en k a últimas de SAL en k2 (son dos matrices en k2).
   5. putOutput(k2)
   6. setTimer(id,4,600) ‘1 minuto y pasa a sleep.
3. Else
   1. abortTimer(id,4) ‘oprime ESTADO de nuevo
   2. putOurput(k1) ‘retorna la salida al estado anterior.
4. FIN.

- Al llamarlo, Estado=0 (valor inicial).

Estado (respuesta a timer 4)

1. Estado= 0
1. putOutput(k1) (leds en estado anterior).
2. Fin.

Pero hay un problema: Estado

- Cuidado, porque si alguien entra, eso no se reflejará automáticamente en la salida…
- Alternativas
  1. Si está armado, un sensor dispara alarma y llama a Estado.
  2. Poner otro timer menor que refresque la salida de acuerdo al estado.

¿Qué alternativa es mejor?

Ejemplo BORRAR

- Chequear con otras tareas y actuar según tabla de transición
- ct (cant teclas numéricas) = 0.
- CL=1 (indica que se oprimió la tecla).
- P=0, FZ=0 (ni presente ni forzado).
- F=0 – inicializa tecla de función.
- Apaga led de fallas.
- Lanza timer 1.
- No se repite timer 1, ya que es una tarea distinta.

Evento 1 (expira timer1)

- CL=0. "Todo como era antes de BORRAR
- Prende led de fallas por un tiempo corto.
- Fin

- Para chequear y abortar otras tareas:
  - stateTask(i) – devuelve 1 Task sleep, 0 wait.
  - abortTask(i) – 1 abortada, 0 no abortada

Actividad: Ingreso de Claves

Variables

- CLAVE – 5 números en ROM.
- CLAVEI – clave para salir de estado inactivo, también en ROM.
- S – estado del sistema.
- CL = 1 – indica si se oprimió Borrar.
- CI(1:5) – vector que guardará la clave ingresada
- CT – cantidad de teclas numéricas ingresadas (inicializada en 0).
- NT valor tecla numérica ingresada
  - Concuerda con posición del estímulo.

Ej. Ingreso de claves

- Si CL=0, FALLA, fin.
- Si CT=0 ‘(solo en S=2, en el resto no se ve Fción).
  - Si Fción → F=1, fin.
  - Si F=1
    - Si NT=0 then P=1, Fin (se armará presente).
    - Si NT=9 then FZ=1, Fin (será forzado).
    - Led de FALLA, CL=0, Fin.
- CT=CT+1
- CI(CT)=NT ‘CI=CI(1)…CI(5) clave ingresada
- Si CT=5
  - Si S=3 and CI=CLAVE then DESARMAR.
  - Si S=2
    - If CI=CLAVE then ARMAR(P,Fz), fin.
    - If CI=CLAVEI then pasarInactivo, fin.
  - Si S=1 and CI = CLAVEI then DESARMAR1.
  - Else FALLA ‘led de fallas y CL=0
- Fin

Pregunta

- ¿Si en medio del ingreso de teclas numéricas, se oprime Fción? ¿Cómo reacciona?
- Toma el valor del contacto de Fcion como un número.
  - Al marcar la quinta tecla (CT=5) marca falla, clave incorrecta.

Comentarios

- Armar (de S=1 a 2) y Desarmar1 (de S=0 a 1)
  - Aumentan el nivel de seguridad.
  - Revisan sensores, sino falla y no cambian de estado.
- Falla – enciende led de fallas, mal la clave o el ingreso.
- Falla1 – titila led de fallas, un sensor activo impide la actividad.
- Cambiar percepción con cada cambio de estado.
  - A partir de ese momento solo entran los estimulos de ese estado.
  - put_MP(k) cambia matriz de percepción.

Sensor Retardado

1. Guardar Num Sensor en una tabla.
2. Si SR=0 then ‘primer sensor retard.
    1. setTimer(task_id,1, 200) ´Timer1 nueva tarea, 200= 20 seg.
    2. SR=1
3. Si SR=2 then Alarma(Id,t)
4. Return.

Timer1

1. Pasar Num Sensores a U-ESTADO.
2. Limpiar tabla num sensores.
3. SR=2
4. Alarma(Id, t).
5. FIN.

- Alarma – función compartida con sensor normal.
  - Si estaba sonando la aborta y la reemplaza con con la misma duración original.
  - Tabla interrelación – reemplaza – es abortar y lanzar de nuevo.
  - También consigo misma.
  - Cuando suena es lo mismo que sensor normal, todos los sensores en este caso se igualan.

El Sw de base debe proveer estado de los sensores para revisar al cambiar de estado. Sin filtro percepción.

- getMEL(k) 'es otra matriz y aplicación empleará matriz de percepción para ver

A mayor detalle:

- Cuidado en ARMAR.
  - Revisar sensores comunes antes, y en ese caso no armar, sino mostrar falla titilando y terminar.
  - Si es presente o ausente
    - Usar percepción correspondiente.
  - Forzado: crear percepción (P o A)
  - Cambiar Percepción retardos luego de un tiempo (con un timer).

- En todo cambio
  - Ej: ARMAR, DESARMAR, INACTIVO
  - Cambiar percepción.
- …

Programación

- Se dará nombres a todos los sensores, mediante EQUs.
- Las tareas se programan con variables locales, con excepciones justificadas.
- El Ing. Volentini publicará las convenciones a seguir

Mejoras futuras

- Uso de recursos.
- Numerar los recursos.
- Cada uno tiene una variable que dice si se usa o no. putUso(k), getUso(k).
- Antes de usar un recurso revisar si se está usando, en ese caso hay un error de interacción de tareas o de programación.
- Realizar Simulación.

Grupos

Pregunta

- ¿Cómo se incorpora un pulsador de pánico?