CIRCUITOS DIGITALES Y MICROCONTROLADORES 2022

Facultad de Ingeniería UNLP

Ejemplos de MEF y su implementación en Software

Ing. José Juárez

Abril 2021

Implementación: MEF en software

6) Definir procedimiento para actualizar la MEF

```
Actualizar_MEF(){
  Leer(&entradas);
  estado=tabla[estado][entradas];
  Act_Salidas(estado,entradas);
}
```

Puede ser Bloqueante o No-bloqueante

Implementación por tablas

7) Definir procedimientos para ejecutar la MEF

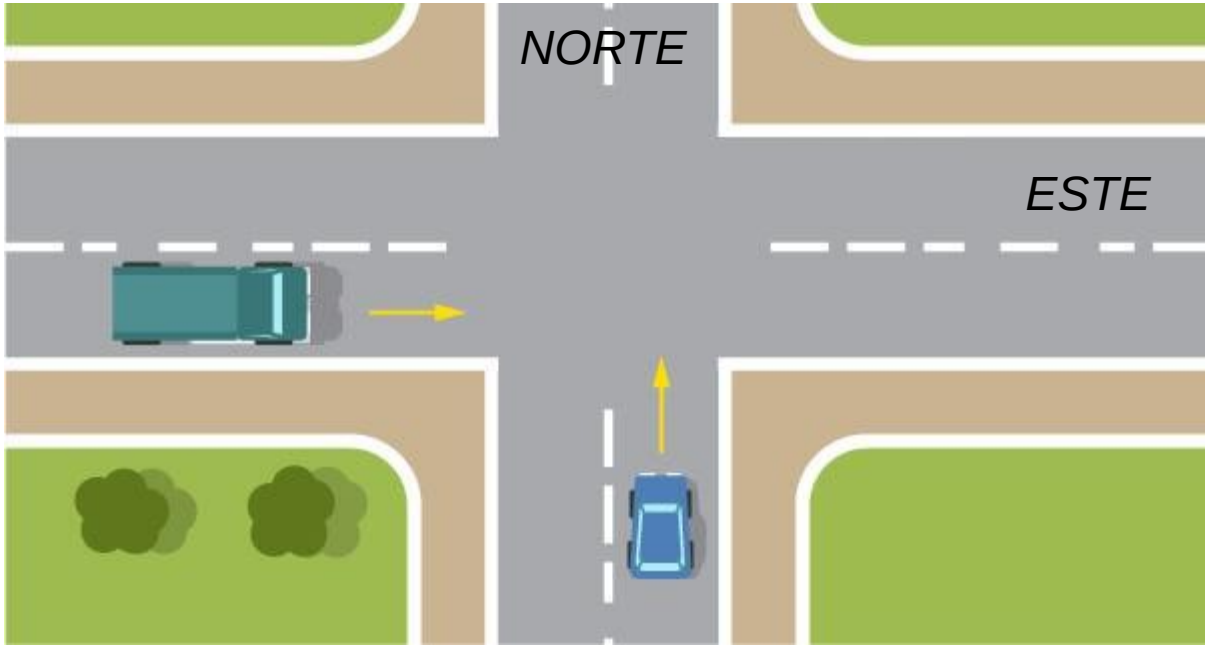
a)
```
Ejecutar_MEF{

  Iniciar_MEF();
  repetir siempre{
        Actualizar_MEF();
  }
}
```

Sin temporizar

b)
```
Ejecutar_MEF{

  Iniciar_MEF();
  repetir cada T segundos{
        Actualizar_MEF();
  }
}
```

temporizada

Control de tráfico



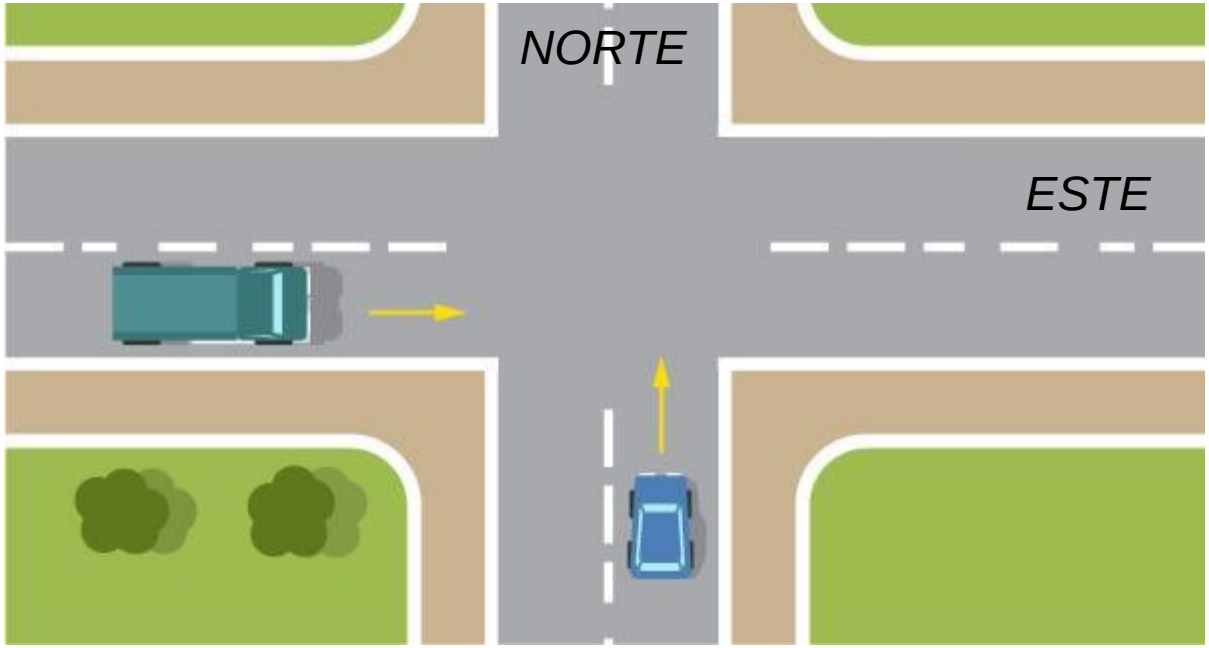
Requerimientos:

1-Si no hay autos permanece en verde
2-Para cambiar de verde a rojo pasar por amarillo 5seg
3-El verde debe durar al menos 30seg
4-Si hay tráfico en una sola dirección el verde queda en esa dirección
5-Si el trafico es en ambas direcciones rotamos el verde

Entradas:
00:calles vacias
01:autos hacia el este
10:autos hacia el norte
11:autos en ambas dir.

Estados y salidas (Moore)
goN,Out0:verde hacia el norte & rojo hacia el este
waitN,Out1:amarillo hacia el norte & rojo hacia el este
goE,Out2:verde hacia el este & rojo hacia el norte
waitE,Out3:amarillo hacia el este & rojo hacia el norte

Ejemplo: Control de tráfico



Variable Out

B0: Verde hacia el norte
B1: Amarillo hacia el norte
B2: Rojo hacia el norte
B3: Verde hacia el este
B4: Amarillo hacia el este
B5: Rojo hacia el este
B6: X
B7: X

salidas (Moore)

Out0:verde hacia el norte & rojo hacia el este
Out1:amarillo hacia el norte & rojo hacia el este
Out2:verde hacia el este & rojo hacia el norte
Out3:amarillo hacia el este & rojo hacia el norte

Por ejemplo Out0:

Out= 0b00100001

Control de tráfico

- Tabla de transición de estados:

Estados\Entradas	00	01	10	11
goN, Out0, 30seg	goN	waitN	goN	waitN
waitN, Out1, 5seg	goE	goE	goE	goE
goE, Out2, 30seg	goE	goE	waitE	waitE
waitE, Out3, 5seg	goN	goN	goN	goN

Las entradas digitales serán encuestadas luego de vencido el plazo de tiempo en un estado
Este tipo de MEF no-periódica se llama Event-driven.
Con lo cuál es cambio de estado en el diagrama será en función de la entrada pero solo cuando el evento tiempo (delay) se cumpla. Esto debe especificarse en diagrama de alguna forma.

Ejemplo: Control de tráfico

Estados\Entradas	00	01	10	11
goN, Out0, 30seg	goN	waitN	goN	waitN
waitN, Out1, 5seg	goE	goE	goE	goE
goE, Out2, 30seg	goE	goE	waitE	waitE
waitE, Out3, 5seg	goN	goN	goN	goN

- Implementación:

```
//Estados posibles
typedef enum {goN,waitN,goE,waitE} state_name;
//estructura de un estado (1 fila de la tabla)
typedef struct state_row {uint8_t Out;
                          uint16_t Time;
                          state_name Next[4];
                         };
//Tabla completa
state_row MEF[4]={
                  {Out0,30,{goN,waitN,goN,waitN}},
                  {Out1, 5,{goE,goE,goE,goE}},
                  {Out2,30,{goE,goE,waitE,waitE}},
                  {Out3, 5,{goN,goN,goN,goN}}
                  };
```

Definición y Declaración de Variables de estado

Definición de la MEF

la tabla es un arrego de estructuras y no un matriz de elementos del mismo tipo.

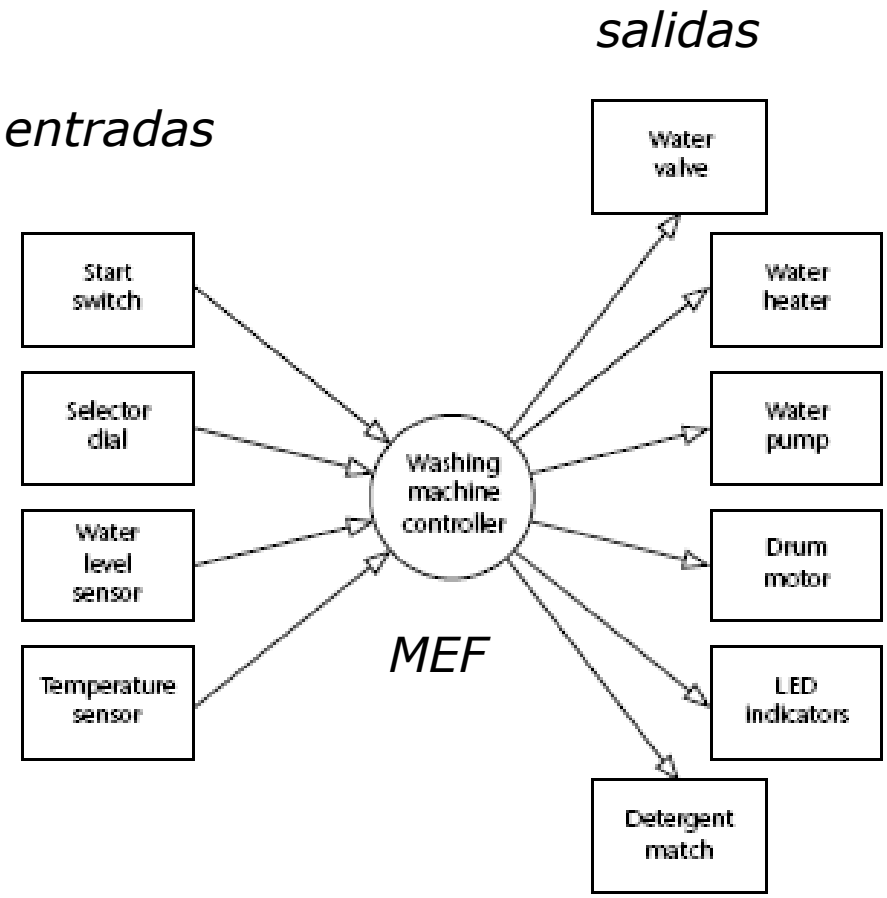
```
main()
{
 uint8_t inputs;
 state_name state;

    Init_MEF();
    while(1){
        inputs=readSensors();
        state=MEF[state].Next[inputs];
        LightsUpdate(MEF[state].Out);
        Delay(MEF[state].Time);
    }
```

Ejecución de la MEF

Método de Inicialización

No tiene temporización periódica o CLK
Los delay son distintos para cada estado.
Es arquitectura event-driven

Control de lavadora



Control de lavadora

Modelo Conceptual:

1. El usuario selecciona el programa de lavado
2. El usuario presiona START
3. Se activa la traba de seguridad
4. Se abre la válvula de agua
5. Si el programa de lavado requiere jabón, se abre la compuerta de jabón y luego se cierra
6. La válvula de agua se cierra cuando se alcanza el nivel
7. Si el programa requiere agua caliente, se encienden los calentadores hasta que se alcance la temperatura especificada
8. Se enciende el motor y se comienza la secuencia de movimientos adecuadas para el programa elegido
9. Al finalizar, la bomba de desagote se enciende para vaciar el tambor y se apaga cuando se detecte que este esta vacío
10. Se desactiva la traba de seguridad

Tarea: hacer el diagrama de estados

Control de lavadora

Posibles Estados:

INIT, START, FILL_DRUM, HEAT_WATER, WASH_01, WASH_02, …, PUMP, ERROR

Funciones a implementar para las entradas:

```
Read_Selector_Dial()        //Leer selector de programa
Read_Start_Switch()         //leer pulsador de arranque
Read_Water_Level()          //leer sensor de nivel de agua
Read_Water_Temperature()    //lleer sensor de temperatura
```

Funciones a implementar para las salidas:

```
Control_Detergent_Hatch()   //controlar dispenser de jabon
Control_Door_Lock()         //controlar traba de seguridad
Control_Motor()             //accionar motor
Control_Pump()              //controlar bomba de desagote
Control_Water_Heater()      //controlar calentadores de agua
Control_Water_Valve()       //controlar válvula de agua
```

Control de lavadora

```
/*-------------------------------------------*-
Washer.C
-*-------------------------------------------*/
#include "Washer.H“

// ------ Private data type declarations -------
typedef enum {INIT, START, FILL_DRUM,
   HEAT_WATER, WASH_01, WASH_02, ERROR}
   eSystem_state;

// ------ Private function prototypes ----------
char WASHER_Read_Selector_Dial(void);
char WASHER_Read_Start_Switch(void);
char WASHER_Read_Water_Level(void);
char WASHER_Read_Water_Temperature(void);
void WASHER_Control_Detergent_Hatch(bit);
void WASHER_Control_Door_Lock(bit);
void WASHER_Control_Motor(bit);
void WASHER_Control_Pump(bit);
void WASHER_Control_Water_Heater(bit);
void WASHER_Control_Water_Valve(bit);
```

```
// ------ Private constants --------------------------
#define OFF 0
#define ON 1
#define MAX_FILL_DURATION            1000L
#define MAX_WATER_HEAT_DURATION 1000L
#define WASH_01_DURATION             30000L

// ------ Private variables --------------------------
static eSystem_state System_state;
static long Time_in_state;
static char Program;
// 10 programas diferentes

// cada uno con uso o no de detergente
static char Detergent[10] = {1,1,1,0,0,1,0,1,1,0};

// cada uno usa o no agua caliente
static char Hot_Water[10] = {1,1,1,0,0,1,0,1,1,0};

/* ---------------------------------------------*/
void WASHER_Init(void)
{
   System_state = INIT;
}
/* ---------------------------------------------*/
```

Control de lavadora: MEF

```
void WASHER_Update(void)
{
   switch (System_state)
   {
     case INIT:
       //código …
       break;
     case START:
      //código
      break;
     case FILL_DRUM:
      //código
      break;
    case HEAT_WATER:
     //código
     break;
   case WASH_01:
   //código
    break;
   //ídem WASH_02:  … etc
   case ERROR:
    //código
    break;
}
```

MEF temporizada con int. Periódica

Usaremos interrupción de Timer para ejecutar periódicamente la MEF

Control de lavadora

- Actualizaremos la MEF con:

void WASHER_Update(void)

- Para temporizar la ejecución de la MEF sabemos hacer:

```
/* ------------------------------------------------------------ */
void main(void)
{
        // Inicializar board y tareas
        WASHER_Init();

        while(1)  {  // Super Loop


           WASHER_Update();
           delay(1000);
         }
} /* ------------------------------------------------------------ */
```

- Pero vamos a utilizar una metodología que veremos en las próximas clases

Control de lavadora

```
/* ------------------------------------------- */
void main(void)
{
        // Inicializar MCU, RTC, y tareas

        while (1)  {  // Super Loop

           if (Flag_MEF) {
                WASHER_Update();
                Flag_MEF =0;
            }

        }
} /* ------------------------------------------- */
```

// MEF se ejecuta por ejemplo
1 vez por segundo

```
/* ------------------------------------------------------- */
    ISR TIMER : ocurre cada 10 ms
/* ------------------------------------------------------- */
ISR (TIMER0_CompA_vect)
{
   if (++cont_MEF==100) {
        Flag_MEF=1;              //ejecutar cada  1s
        cont_MEF=0;
}
```

Control de lavadora

```
case INIT:
// Motor off
  WASHER_Control_Motor(OFF);
  // Pump off
  WASHER_Control_Pump(OFF);
  // Heater off
  WASHER_Control_Water_Heater(OFF);
  // Valve closed
  WASHER_Control_Water_Valve(OFF);
  // Esperar hasta que se presione START
  If (WASHER_Read_Start_Switch() == 1)
  {
    // Start presionado...
    // Leer selector dial
    Program = WASHER_Read_Selector_Dial();
    // cambiar estado
    System_state= START;
}
break;
```

Actualizar salidas

*Lee entrada asociada al estado *.*

Estado siguiente

Ejemplo: lectura del pulsador (No bloqueante)

```
char WASHER_Read_Start_Switch(void)
{
  //polling periódico
  if (Start_pin == 0)
  {
     // Start switch presionado
    // puede tener verificación
    // NO tiene retardo anti-rebote
     return 1;
  }
  else
  {
     return 0;
  }
}
```

** En este estado no afectan el resto de las entradas*

Control de lavadora

```
case START:
  // Trabar puerta
  WASHER_Control_Door_Lock(ON);
  // abrir válvula agua
  WASHER_Control_Water_Valve(ON);
  // abrir válvula de detergente (si
   corresponde)
  if (Detergent [Program] == 1)
  {
    WASHER_Control_Detergent_Hatch(ON);
   }

  // ir al siguiente estado
  System_state = FILL_DRUM;
  Time_in_state = 0;
  break;
```

```
case FILL_DRUM:
  // esperar hasta que se llene el tanque
  if (++Time_in_state >= MAX_FILL_DURATION)
      // incluye Timeout
      System_state= ERROR; // hay problemas...
  if (WASHER_Read_Water_Level() == 1)
  {   // tanque lleno
     // se requiere agua caliente?
     if (Hot_Water  [Program] == 1)
     {
        WASHER_Control_Water_Heater(ON);
        // ir a siguiente estado
        System_state= HEAT_WATER;
        Time_in_state = 0;
     } else    {
       // caso agua fría - comenzar programa
       System_state = WASH_01;
       Time_in_state = 0;
     }
  }
  break;
```

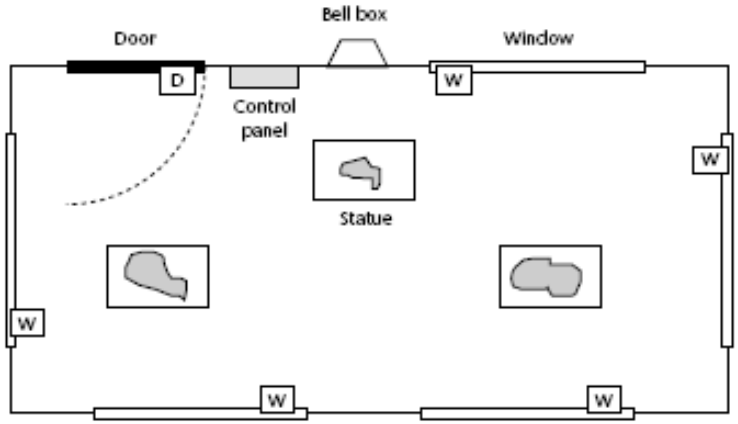
Control de lavadora

```
case HEAT_WATER:
  // esperar se caliente el agua

  if (++Time_in_state >= MAX_WATER_HEAT_DURATION)
  {
    // Timeout...
    System_state= ERROR;
   }
  // verificar temperatura del agua
  if (WASHER_Read_Water_Temperature() == 1)
  {
    // agua a temperatura deseada
    // próx estado
    System_state = WASH_01;
    Time_in_state = 0;
   }
  break;
```

```
case WASH_01:
  // WASH_01
  WASHER_Control_Motor(ON);
  if (++Time_in_state >=WASH_01_DURATION)
  {
    System_state = WASH_02;
    Time_in_state = 0;
  }
  break;


//demás estados omitidos

case ERROR:
  // activar alarma
break;
```

// FIN EJEMPLO

Implementación de un alarma domiciliaria



Las ventanas y las puertas tienen sensores magnéticos.

En el exterior se encuentra la sirena.

En el interior se encuentra

El teclado, el display y un buzzer

El usuario arma la alarma colocando la clave de 4 dígitos, y cuenta con un tiempo de 60 seg para salir.

De la misma manera al entrar tiene 60 seg para desactivarla colocando la password, de lo contrario la alarma sonará

Modelado y especificación: MEF

- Inicialmente el sistema esta en el estado *Desarmado*, hasta que se ingrese la password correcta.
- A partir de aquí entramos al modo *Armando* que espera 60 seg.
- Luego el sistema esta *Armado* monitoreando los sensores. Si un sensor de ventana se activa, se pasa al modo *Intruso*, en cambio si el sensor de puerta se activa se pasa a modo *Desarmando.*
- En el estado *Desarmando* se espera 60seg para que el usuario autorizado pueda ingresar la clave. Si se ingresa la clave correcta el sistema para a *Desarmado* Caso contrario pasa a estado *Intruso.*
- En el estado *Intruso* la alarma suena indefinidamente hasta que se ingrese la clave correcta.
- Con esta información definimos el “diagrama de estados”

Tarea: hacer el diagrama de estados completo

Arquitectura del Software

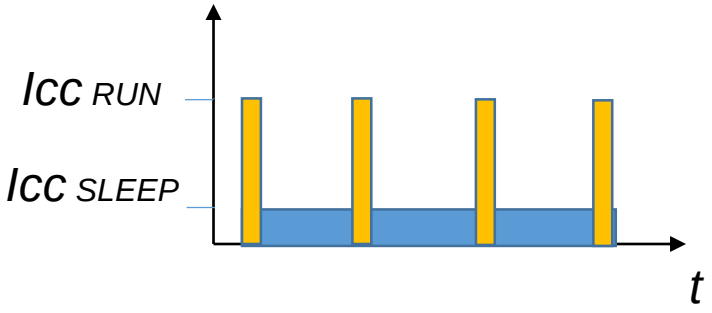
- Temporización de Tareas:
  - La temporización será por ISR de Timer (Tick del sistema)
  - La interrupción de Timer será cada 100 ms
  - La maquina de estados se actualizará cada 100ms
  - Se utilizarán Funciones no bloqueantes para el manejo de entradas-salidas
- Módulos del software:
  - Main.c, main.h
    - Inicialización del sistema y super loop
  - sEOS.c, sEOS.h
    - Planificador y Despachador cooperativo multitarea
  - Intruder.c, Intruder.h
    - Implementación MEF de la alarma
  - Keypad.c, Keypad.h
    - Biblioteca de funciones para leer el teclado
  - Lcd.c, Lcd.h
    - Biblioteca de funciones para manejo de LCD

Implementación: main()

```
void main(void)
{
      MCU_Init();
      // Inicializar LCD
      LCD_Init();
      // inicializar teclado
      KEYPAD_Init();
      // inicializar maq de estados y buffers
      INTRUDER_Init();
      // configurar timer  (100ms tick)
      sEOS_Init_Timer(100);

      while(1)
      {
            sEOS_Dispatch_Tasks(); // ejecutar tareas
            sEOS_Go_To_Sleep(); // modo idle bajo consumo
      }
}
```

```
void INTRUDER_Init(void)
{
  System_state = DISARMED;
  LCD_Write_String("\nDisarmed");
  // Alarma apagada
  Sounder_pin = 0;
  State_call_count = 0;
}
```

La ISR despierta al MCU



Implementación: planificación y despacho

```
ISR (Timer0_CompA_vect) //  cada 100 ms
{
      // actualizar MEF cada 100 ms
      MEF_flag=1;
}
```

La función sEOS_Init_Timer(100) Inicializa el timer0 por ejemplo para que interrumpa periódicamente cada 100ms

```
void sEOS_Dispatch_Tasks(void) {

      // actualizar MEF cada 100 ms
      if ( MEF_flag) {
            MEF_flag = 0;
            INTRUDER_Update(); // MEF
      }
}
```

Implementación: MEF

MOORE

```
void INTRUDER_Update(void)
// llamada cada 100 ms
{
      // Contar numero de interrupciones
      State_call_count++;

      switch (System_state)
      {
            case DISARMED:
                  if (New_state) // si se cambia de estado hay que informarlo
                  {
                        LCD_Write_String ("\nDisarmed");
                        New_state = 0;
                  }
                  // Alarma apagada
                  Sounder_pin = 0;
                  // esperar por clave correcta ...
                  if (INTRUDER_Get_Password() == 1)
                  {
                        System_state = ARMING; //pasar a armando...
                        New_state = 1;
                        State_call_count = 0;
                        break;
                   }
            break;
```

Actualiza salidas

Condición de entrada al estado. Única vez

Lee entrada

Estado siguiente

Implementación: MEF

MEALY

```
void INTRUDER_Update(void)
{
      // Contar numero de interrupciones para calcular el tiempo en cada estado
      State_call_count++;

      switch (System_state)
      {
            case DISARMED:
                  // esperar por clave correcta ...
                  if (INTRUDER_Get_Password() == 1)
                  {
                        System_state = ARMING; //pasar a armando...
                        LCD_Write_String("\nArming...");
                        State_call_count = 0;
                        break;
                   }
            break;
```

Leer entrada relevante para este estado

Siguiente estado

Actualiza salida (Mealy)

Continuamos con MEALY

Implementación: MEF

```
case ARMING:

        // Esperar 60 segundos (600 ticks)
        if (++State_call_count > 600)
        {
                System_state = ARMED;
                LCD_Write_String("\nArmed");
                State_call_count = 0;

        }
        break;
```

Implementación: MEF

```
case ARMED:

    if (INTRUDER_Check_Window_Sensors() == 1) // Chequear sensores de ventanas
    {
        // Intruso detectado
        System_state = INTRUDER;
        LCD_Write_String("\n** INTRUDER! **");
        //Activar alarma
        Sounder_pin = 1;
        State_call_count = 0;
    }
    if (INTRUDER_Check_Door_Sensor() == 1) // chequear sensor de puerta
    {
        System_state = DISARMING; // Puede ser un usuario autorizado
        LCD_Write_String("\nDisarming...");
        State_call_count = 0;
    }
    if (INTRUDER_Get_Password() == 1) // Si hay clave correcta desarmar
    {
        System_state = DISARMED;
        LCD_Write_String("\nDisarmed");
        // Alarma apagada
        Sounder_pin = 0;
        State_call_count = 0;
    }
break;
```

Podría implementarse
un switch-case dentro del estado
para evaluar los eventos de entrada

Implementación: MEF

```
case DISARMING:
        // Esperar 60 seg para que ingrese la clave correcta, sino ALARMA
        if (++State_call_count > 1200)
        {
                System_state = INTRUDER;
                LCD_Write_String("\n** INTRUDER! **");
                //Activar alarma
                Sounder_pin = 1;
                State_call_count = 0;
                break;
        }
        // mientras tanto chequear sensores de ventana
        if (INTRUDER_Check_Window_Sensors() == 1)
        {
                // intruso detectado
                System_state = INTRUDER;
                LCD_Write_String("\n** INTRUDER! **");
                //Activar alarma
                Sounder_pin = 1;
                State_call_count = 0;
                break;
        }
        if (INTRUDER_Get_Password() == 1) // si hay clave correcta desarmar
                {
                System_state = DISARMED;
                LCD_Write_String("\nDisarmed");
                // Alarma apagada
                Sounder_pin = 0;
                State_call_count = 0;
                break;
```

Implementación: MEF

```
case INTRUDER:


        // Esperar sonando hasta que se ingrese la clave correcta
        if (INTRUDER_Get_Password() == 1)
        {
                System_state = DISARMED;
                LCD_Write_String("\nDisarmed");
                // Alarma apagada
                Sounder_pin = 0;
                State_call_count = 0;
        }
break;
```

Implementación: funciones de verificación

```
//Las funciones devuelven 0 si no hay evento y 1 si lo hay

char INTRUDER_Check_Window_Sensors(void)
{
      // un solo sensor en el ejemplo
      if (Window_sensor_pin == 0)
      {
      // Intruso detectado...
      return 1;
      }
      // Default
      return 0;
}
```

No bloqueantes, sin retardos ni condiciones de espera.

```
char INTRUDER_Check_Door_Sensor(void)
{
      // Sensor de puerta
      if (Door_sensor_pin == 0)
      {
      // Puerta abierta
      return 1;
      }
      // Default
      return 0;
}
```

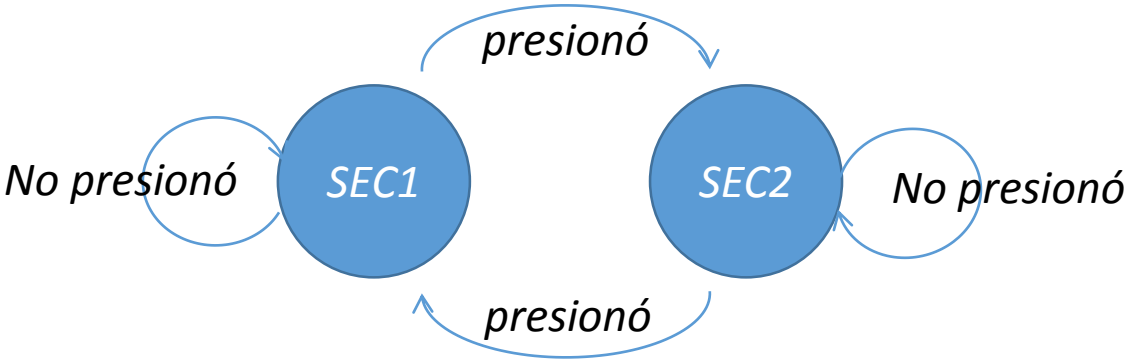
Implementación: funciones de verificación

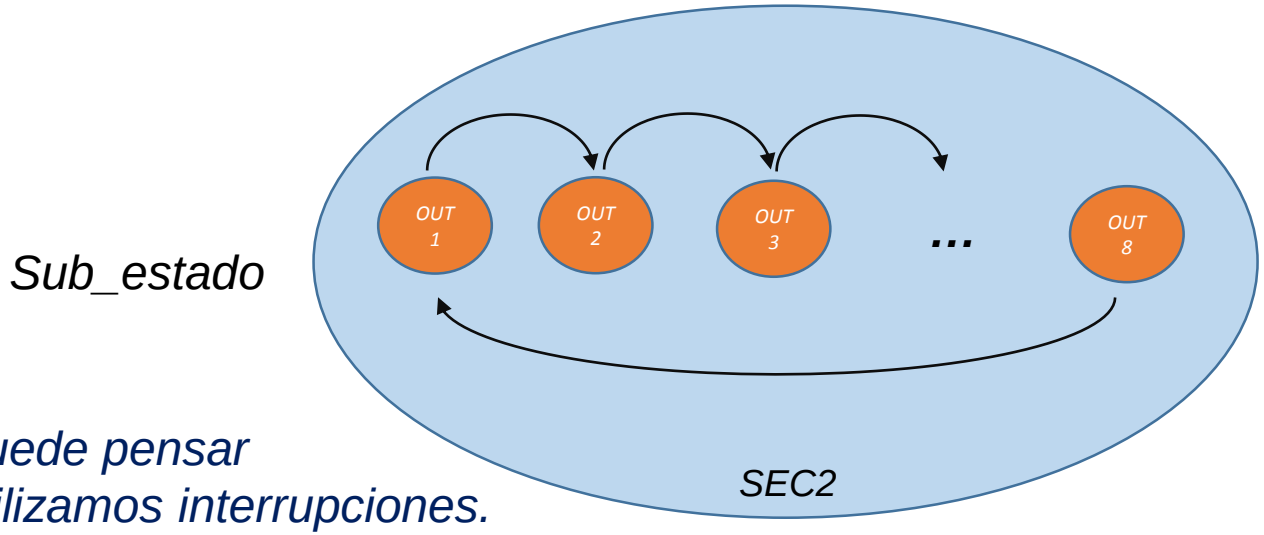
```
char INTRUDER_Get_Password(void)
{
        // Hay nuevo dato en el buffer de teclado?
        if (KEYPAD_scan(&Key) == 0)
        {
                return 0; // No Hay dato nuevo y por lo tanto no hay clave correcta
        }
        // Se ha apretado una tecla, se espera 50seg mas para leer la clave completa
        if (State_call_count > 500)
        {
                // timeout, no se completo el ingreso de la clave, reiniciar
                State_call_count = 0;
                Tecla=0;
                return 0;
        }
        //Escribo y guardo tecla presionada
        LCD_Write_Char(Key);
        Input [Tecla++] = Key;
        // tengo las 4 teclas?
        if(Tecla==4) {
                Tecla=0;
                return validar_clave(); //clave correcta retorna 1, incorrecta 0
         } else
                return 0; //no tengo las cuatro teclas aún
}
```

No bloqueante, sin retardos ni condiciones de espera.

TP2

- **Ejercicio No 12: Planificación de tareas con RTI**
  a) Realizar el ejercicio entregado en el TP1 utilizando una interrupción periódica de TIMER0 para planificar y temporizar las tareas involucradas (no deberá usar más los retardos bloqueantes).





Esta MEF también se puede pensar
Como event-driven si utilizamos interrupciones.

Bibliografía

Los ejemplos fueron extraídos de la siguiente bibliografía:

- *"Embedded Microcomputer System, Real-Time Interfacing".* J. Valvano 2nd Ed. Thomson 2007.

- *"Patterns for time-triggered Embedded Systems"*, Michael J. Pont. 2014

Published by SafeTTy Systems. pdf disponible en la web en https://www.safetty.net/publications/pttes (descarga gratis en la web de autor)

Introducción a los sistemas reactivos

Sistemas Reactivos

- El término **reactivo** se aplica a sistemas que responden dinámicamente a los **eventos** de interés que reciben y cuyo comportamiento lo define el orden de llegada de esos eventos.
- Prácticamente todos los sistemas embebidos son reactivos por naturaleza, lo que significa que su trabajo principal es reaccionar a eventos, como presionar botones, tocar una pantalla, esperar por alarmas o llegadas de paquetes de datos.
- Ejemplos: los cajeros automáticos, los sistemas de reservas de vuelos, los sistemas de control, los sistemas embebidos en los automóviles.
- En consecuencia, la mayoría de las veces, un sistema está esperando eventos y solo después de reconocer un evento reacciona realizando el cálculo apropiado.
- Los principales desafíos de un sistema reactivo son:
  - realizar el cálculo correcto
  - y realizarlo de manera oportuna (nos vamos acercando a la definición de RTOS)

La definición de sistemas reactivos abarca además a grandes sistemas de computo, sistemas distribuidos y sistemas ciber-físicos:
https://www.reactivemanifesto.org/

Introducción a los sistemas reactivos

Sistemas embebidos reactivos:

- La mayoría de los sistemas embebidos se programan de manera secuencial, donde un programa espera de manera explícita los eventos mediante sondeo (polling) o bloqueo en un retardo de tiempo (delay), un semáforo u otro mecanismo similar de un sistema operativo tradicional en tiempo real (RTOS). Ejemplo del código secuencial básico es la implementación tradicional "Blinky LED“.
- La mayoría de los sistemas de la vida real no son secuenciales, lo que significa que el sistema debe manejar múltiples eventos simultáneamente. El problema fundamental es que mientras un programa secuencial está esperando un tipo de evento (por ejemplo, que se cumpla el delay) no está haciendo nada más y no responde a otros eventos (por ejemplo, presionar un botón).
- El paradigma secuencial no atenta contra la abstracción de sistemas reactivos. El problema para reaccionar a tiempo a múltiples eventos parece ser el bloqueo y la secuencialidad (no-concurrencia).
- Por esta y otras razones, hay que tener mucho cuidado con los diversos mecanismos de bloqueo (utilicemos o no RTOS), porque a menudo conducen a programas que no responden, son difíciles de entender, impredecibles e inseguros.

Introducción a los sistemas reactivos

Programación Reactiva o basada en eventos (Event-driven)

- Es una estrategia moderna de diseño (un paradigma de programación) para combinar múltiples tareas de manera concurrente y se basa en los siguientes principios (abstracción):

  1-Mantener los datos aislados y vinculados a las tareas. Los objetos deben ocultar (encapsular) sus datos privados y otros recursos, y no compartirlos con el resto del sistema.

  2-Mantener la comunicación entre tareas de forma asincrónica a través de mensajes (más abstracción). El uso de eventos asincrónicos (objetos reactivos) mantiene las tareas funcionando de manera verdaderamente independiente, sin bloquearse entre sí.

  3-Las tareas deben pasar toda su vida respondiendo a los eventos entrantes, por lo que su línea principal debe consistir en un bucle que maneje los eventos uno a la vez hasta su finalización, evitando así cualquier peligro de concurrencia dentro de una tarea.

Referencias:

Quantum Leaps: Modern Embedded Software

RKH framework para Statecharts

Introducción a los sistemas reactivos

- Para modelar el comportamiento de los objetos activos (encapsulados, asincrónicos, sin bloqueo y controlados por eventos) se utilizan los Statecharts.

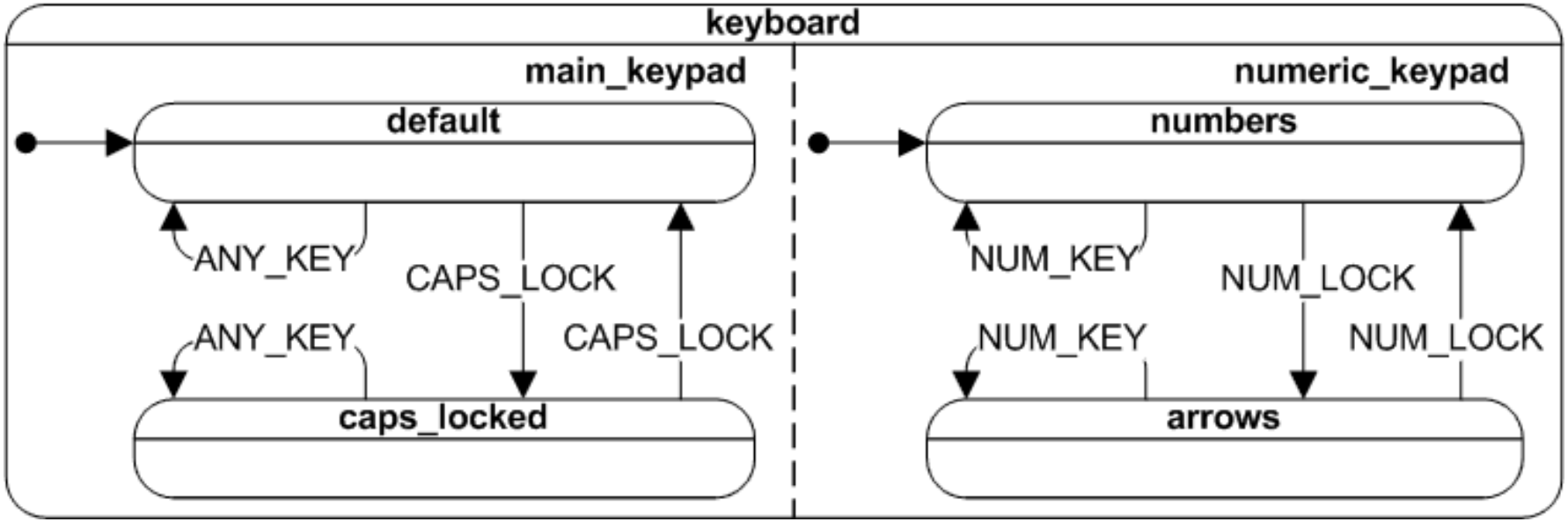
- Surgen a mediados de la década de 1980, cuando David Harel propuso una amplia extensión al formalismo convencional de las máquinas y diagramas de estados

- El objetivo principal del nuevo formalismo gráfico es modelizar o describir **sistemas reactivos complejos**.

- Los statecharts extienden los diagramas de transición de estados convencionales con tres elementos principales: jerarquía, concurrencia y comunicación. Se encuentran en la especificación UML (Unified Modeling Lenguage).

Introducción a los sistemas reactivos

- Ejemplo de un modelo Statechart



Implementación: MEF en software

c)

```
Ejecutar_MEF{

  Iniciar_MEF();
  repetir siempre{
        sleep();
        Actualizar_MEF();

  }
}
ISR(IRQ1_request) {
        Leer_Main_Keypad();
}

 ISR(IRQ2_request) {
         Leer_Num_Keypad();
 }
```

Posible implementación MEF Event-driven