Ejercicios Complementarios

Máquina de Estados

Realización de funciones para resolución de necesidades básicas.

Ejercicios generales de Máquina de Estados que se pueden desarrollar utilizando uModelFactory.

Para su codificación supondremos el siguiente escenario:

- La aplicación correrá sobre un Microcontrolador (μC) de 32 bits, sin Sistema Operativo.
- Para acceder al HW del μC y a utilidades para el manejo de tiempo, se utilizará una biblioteca de funciones.
- El nombre de las funciones puede variar dependiendo de la biblioteca utilizada. Al final de este documento se anexan nombres de funciones a modo referencial.
- Tener en cuenta que al trabajar sobre un μC sin sistema operativo:
    - No se puede capturar los recursos de procesamiento.
    - En general, la aplicación nunca finaliza (solo termina al apagar el equipo)

<u>Ejercicio 1</u>: Túnel vehicular

Se requiere implementar un sistema para señalizar el acceso a un túnel vehicular en el cual, por cuestiones de seguridad debe pasar solo un vehículo a la vez (un solo carril, un solo sentido de circulación).

El sistema cuenta con:

- Dos sensores para detectar el ingreso y egreso de vehículos al túnel (INGRESO / EGRESO).
- Dos luces en el acceso al túnel como señalación visual de lo que ocurre (VERDE / ROJA).
- Una sirena en el acceso al túnel como señalización sonora para informar ante un eventual (SIRENA).
- Un pulsador de pared para desactivar la sirena (DESACTIVAR).

El sistema debe comportarse así:

- La luz verde debe permanecer encendida si no hay vehículos en el túnel.
- Al ingresar un vehículo al túnel, debe encender la luz roja y apagar la verde.
- Si un vehículo permanece dentro del túnel más de 10S:
    - La luz roja debe parpadear con una cadencia de 2 segundos ON y 1s OFF.
    - La sirena sonará cada 20s, de la siguiente manera 1s ON / 0,5s OFF, 3 veces.
- Para desactivar el parpadeo y la sirena (de modo inmediato) se debe presionar el pulsador de pared.
- Si en situación de “alarma”, el vehículo sale del túnel antes de que se oprima el pulsador de pared, todo el sistema vuelve a su estado de reposo.

Se pide que:

- ⇒ Realice el diagrama de estados para cumplir con el comportamiento enunciado. La solución debe contemplar que al iniciar el sistema podría haber un vehículo en el túnel.
- ⇒ Defina el diagrama de conexiones – recursos a utilizar (a su criterio)
- ⇒ Implemente la aplicación (usando la librería de la cátedra)

Ejercicio 2: Túnel vehicular - Configurable

Se pide modificar el ejercicio anterior para poder configurar/modificar los tiempos definidos para:

- Permanencia dentro del túnel – original: 10 s - configurable entre 8 y 60 segundos
- Periodo de parpadeo de la luz roja – original: 10 s - configurable entre 3s y 9s
- Tiempo para que la sirena vuelva a sonar – original: 20 s - configurable entre 6s y 30s

La configuración se recibe por tramas en formato ASCII a través del Puerto Serie, Estas tramas poseen el siguiente formato:

Inic	Com	V1	V2	Flin

En donde

- **Inic:** byte de inicio de trama – su valor es siempre el símbolo asterisco (*)
- **Com:** Comando o parámetro a configurar:
  - **T**: Tiempo máximo de permanencia en Túnel
  - **R**: parpadeo de luz roja
  - **S**: Cadencia de la sirena
- **V1** y **V2:** Dos bytes con el valor a configurar en formato decimal-ASCII
- **Fin:** byte de fin de trama – su valor es siempre es el símbolo ($)

La trama será considerada válida, si se respeta con toda la trama y si los valores recibidos están acordes con los requerimientos.

***Nota**: período de parpadeo de luz roja incluye el tiempo en ON + el Tiempo en OFF. La aplicación debe tener en cuenta que siempre el tiempo en ON debe ser el doble del tiempo de OFF.*

Se pide que:

- ⇒ Realice el diagrama de estados para poder recibir las tramas indicadas.
- ⇒ Establezca los mecanismos de comunicación entre ambas MdE (ésta y la del ejercicio anterior)
- ⇒ Implemente la aplicación completa (usando la librería de la cátedra)

Ejercicio 3: Túnel vehicular - Final

Modificar el ejercicio anterior de manera que:

- Cuando un vehículo se encuentra dentro del túnel, se encienda la iluminación interna del túnel (LUZ)
- Informe por puerto serie:
  - Cuando Inicializa el sistema
  - Cada vez que se reciba una trama válida, indicando valores y comando recibido.
  - Cada vez que el túnel entra en alarma y sale de ella, indicando modo de salida del estado
  - Cada 30 minutos, la cantidad de vehículos que pasaron por el túnel

Se pide que:

- ⇒ Realice las modificaciones necesarias sobre las MdE.
- ⇒ Defina las tramas que utilizará para enviar los datos por Puerto Serie
- ⇒ Implemente la aplicación completa en base a sus definiciones (usando la librería de la cátedra)

Ejercicio 4: Control de temperatura ON-OFF

Se desea controlar la temperatura de un horno que tiene un sensor conectado al ADC de nuestro sistema microcontrolado, de modo tal que la misma oscile entre 25 y 35 °C. Se debe considerar una histéresis de 2°C a los efectos de suavizar la conexión y desconexión de los contactores de la estufa y el ventilador que posee el sistema a tal efecto.

Funcionamiento

- Si la temperatura medida disminuye por debajo de la TEM_MIN, se debe activar la estufa. En cuanto alcance un valor igual a TEM_MAX-K (siendo K el valor establecido de la histéresis) debe apagarse.
- Del mismo modo, si la temperatura supera la TEM_MAX, debe activarse el ventilador, hasta que la temperatura alcance un valor de TEM_MIN+K.
- Mientras la temperatura se encuentre entre TEM_MIN+K y TEM_MAX-K el sistema mantiene apagados tanto a la estufa como el ventilador.

Se pide que:

- ⇒ Realice el diagrama de estados para cumplir con el comportamiento enunciado.
- ⇒ Defina el diagrama de conexiones – recursos a utilizar (a su criterio)
- ⇒ Implemente la aplicación (usando la librería de la cátedra)

Ejercicio 5: Control de temperatura ON-OFF - Configurable

Se pide modificar el ejercicio anterior para poder configurar/modificar las temperaturas definidas:

- Temp. inferior – original: 25°C - configurable entre 12 y 32 grados
- Temp. superior – original: 35°C - configurable entre 20 y 50 grados

La configuración se recibe por tramas en formato ASCII a través del Puerto Serie, Estas tramas poseen el siguiente formato:

Inic	Ti1	Ti2	Ts1	Ts2	TX	Flin

En donde

- **Inic:** byte de inicio de trama – su valor es siempre el símbolo asterisco (*)
- **Ti1** y **Ti2:** Dos bytes con el valor a configurar de la temperatura límite inferior en formato decimal-ASCII
- **Ts1** y **Ts2:** Dos bytes con el valor a configurar de la temperatura límite superior en formato decimal-ASCII
- **TX:** Byte de verificación
- **Fin:** byte de fin de trama – su valor es siempre es el símbolo numeral (#)

Byte de verificación: su valor es igual al dígito menos significativo que resulte de todos los dígitos de temperatura recibido

Por ejemplo:

*	'1'	'8'	'3'	'9'	TX	#

De esta trama se extrae:

- Temperatura mínima: 18°C
- Temperatura máxima: 39°C
- TX: debería ser: 1 + 8 + 3 + 9 = 21 +> el dígito menos significativo 1, Es decir que TX debe ser 1
    - Si TX==1 se entiende que la trama no tuvo errores de comunicación

*	'1'	'8'	'3'	'9'	'1'	#

    - Si TX <> 1 la trama es inválida

La trama será considerada válida, si:

- Se respeta con toda la trama y el valor de TX cumple con el protocolo de verificación establecido
- Si la temperatura inferior es al menos 6° C menor que la temperatura máxima considerando el valor de histéresis.

Se pide que:

⇒ Realice el diagrama de estados para poder recibir las tramas indicadas.
⇒ Establezca los mecanismos de comunicación entre ambas MdE (ésta y la del ejercicio anterior)
⇒ Implemente la aplicación completa (usando la librería de la cátedra)

Ejercicio 6: Control de temperatura ON-OFF – con alarma

Modificar el punto anterior, de manera que, si la temperatura se va de los límites prestablecidos, se encienda una alarma o sirena (ALARMA).

Si la salida de los límites de temperatura, es producto de un cambio en la configuración del sistema, antes de encender la alarma, se debe dar TT minutos para que el sistema reestablezca su temperatura. Si dentro de TT minutos el sistema no entro en régimen, se debe encender la sirena.

La solución debe contemplar la histéresis térmica.

La alarma se debe apagar automáticamente cuando la temperatura entra dentro del rango de temperatura configurado.

TT se calcula como: la diferencia entre la temperatura del sistema al momento del cambio de configuración y la temperatura de debe alcanzar (1°C equivale a 1 minuto), más 2 minutos.

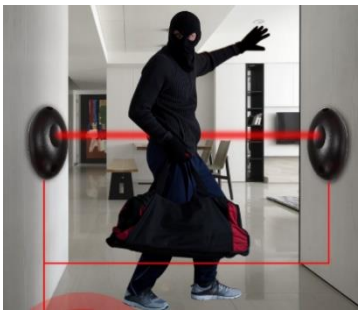
<u>Ejercicio 7</u>: Alarma para Museo

Se desea desarrollar un sistema de monitoreo y control para las obras de arte de un museo. Para ello, se busca desarrollar unidades autónomas que permitan detectar diferentes niveles de amenaza sobre las obras.

Cada unidad se encuentra compuesta por:

- Un sistema basado en una barrera infrarroja que permite la detección en caso de que se intente tocar el objeto.
- Una balanza que permite detectar pequeñas variaciones sobre el peso del objeto, por ejemplo, si se lo intenta remover o reemplazar por otro elemento.
- Una cámara de vídeo IP que permite el registro en caso de alerta.





Funcionamiento

El equipo arranca en estado de “armado” a la espera de la ocurrencia de uno de los 2 eventos posibles: activación de la barrera infrarroja o activación del sensor de peso.

En este estado, si se detecta un acercamiento a la obra (detección mediante la barrera), se deberá activar una alarma silenciosa. Frente a esta situación el guardia se acercará a la zona y mediante un pulsador externo oculto la podrá apagar (si corresponde). Si la alarma no es apagada dentro de los 10 segundos se deberá activar la alarma sonora. El apagado de esta alarma es mediante el mismo pulsador externo.

Si en cualquier condición se detecta una variación de peso se deberá activar, además de la alarma sonora, el monitoreo mediante la cámara. Una vez activada esta alarma, la única forma de apagarlo es mediante el pulsador externo.

Una vez apagada la alarma, el equipo queda nuevamente en modo armado.

Nota: Para simular el peso se puede utilizar el potenciómetro asociado al ADC del kit. Queda a criterio del desarrollador cual es la variación de este potenciómetro (“cuentas”) que signifique que se está moviendo la obra de arte.

Se pide que:

⇒ Realice el diagrama de estados para cumplir con el comportamiento enunciado.
⇒ Defina el diagrama de conexiones – recursos a utilizar (a su criterio)
⇒ Implemente la aplicación (usando la librería de la cátedra)

<u>Ejercicio 8</u>: Simón

Se pide implementar una versión simplificada del exitoso juego electrónico de los ochenta Pocket Simón.
El juego posee 4 teclas grandes, cada una de un color: rojo, azul, verde y amarillo; Debajo de cada tecla hay una luz del mismo color. Adicionalmente posee una tecla "Start" para dar comienzo al juego, un Buzzer para emitir el sonido de si gano o perdió el juego.
A continuación, se explica cómo jugar y luego se dará una especificación de los tiempos que deben respetarse.

- Presione "Start". Pocket Simón encenderá una de las cuatro luces.
- Esta es la primera señal de la secuencia.
- Presione la tecla del color correspondiente.
- Pocket Simón repetirá la primera señal y agregará una más.
- Presione las dos teclas, secuencialmente y en el orden correcto.
- Pocket Simón repetirá las dos primeras señales y agregará una más.
- Presione las tres teclas, secuencialmente y en el orden correcto.
- Esto continuará hasta alcanzar el máximo de repeticiones que es 12.
- Si el jugador presiona una tecla en orden incorrecto o demora demasiado tiempo perderá el juego.

Mientras el usuario mantenga una tecla presionada, se encenderá la luz de la tecla y sólo se apagará cuando la suelte.
Si no es la última tecla de la secuencia, dispone de 3 segundos para presionar la próxima, desde que presionó esta última.
Si se alcanza el máximo de repeticiones, se emitirá un sonido especial por haber ganado y ante cualquier equivocación se emitirá un sonido especial por haber perdido.
Si una vez terminada la jugada se presiona "Start" se vuelve a comenzar una nueva jugada.

Notas: ¿Cómo puede generar un valor aleatorio si no cuenta con la función rand()? (necesario para la secuencia de las luces)

Se pide que:

⇒ Realice los diagramas de estados para cumplir con el comportamiento enunciado.
⇒ Defina el diagrama de conexiones – recursos a utilizar (a su criterio)
⇒ Implemente la aplicación (usando la librería de la cátedra)

ANEXO
Ejemplo de manual de primitivas

El nombre de las funciones puede variar según la versión de Librería en uso. Si los va a hacer correr sobre la placa base, utilice los nombres de las funciones de la librería que este en uso.

- Configuración y Background
- Tipos de Datos
- Entradas y Salidas digitales
- Entradas analógicas
- Temporización
- Puerto Serie
- Salidas de texto

Configuración y Background

Configuración:

```
void Inicializacion(void);
```

Configura e inicializa el HW a utilizar.
Parámetros y valor de retorno: **void**

background:

```
void TimerEvent (void);
```

Vincula a las aplicaciones que corren en segundo plano con nuestra aplicación.
Parámetros y valor de retorno: **void**

Ejemplo de uso:

```
void main ()
{
      ⋮
Inicializacion();
      ⋮
   while(1)
   {
      TimerEvent();
            ⋮
   }
      ⋮
```

Tipos de datos:

```
typedef unsigned int uint32_t;
typedef short unsigned int uint16_t;
typedef unsigned char uint8_t ;
typedef int int32_t;
typedef short int int16_t;
typedef char int8_t;
typedef void (*Timer_Handler)(void);
```

Entradas y Salidas digitales

Lectura del Teclado (4 teclas):

```
uint8_t GetKey (void)
```

Trae del buffer de teclado el código de la tecla pulsada.
Parámetros: **void**
Retorno: Código de tecla (0 a 3) o NO_KEY (0xFF) *(#define NO_KEY 0xFF)*

Entradas Digitales (ED):

```
uint8_t Entradas (uint8_t nEntrada)
```

Lee el valor de la ED solicitada.
Parámetro:
nEntrada: Número de entrada (valores 0, 1 y 2)
Retorno: valor de la entrada (0 o 1) – revisar si es bajo o alto

Salidas Digitales (Relays):

```
void Relays (uint8_t nRelay, uint8_t estado)
```

Activa o desactiva un relay.
Parámetros:
nRelay: Numero de relay (0 a 3)
estado: ON (1) u OFF(0)
Retorno: **void**

Salidas Digitales (RGB):

```
void LedsRGB (uint8_t led, uint8_t estado)
```

Activa o desactiva uno de los leds del RGB.
Parámetros:
led: usar las siguientes macros
- ROJO: 0
- VERDE: 1
- AZUL: 2

estado: ON (1) u OFF(0)
Retorno: **void**

Temporización

Iniciar un timer

```
void TimerStart (uint8_t event, timer_t t, void (*handler)(void), uint8_t base)
```

*Inicia el timer identificado por **event** y al transcurrir el tiempo especificado por **t** (según **base**) se llama a la función indicada por **handler**.*
Parámetros:
***event**: Número de timer asociado al evento. Valor entre 0 y 31*
***t**: Tiempo asignado al evento. Dependiente de la base de tiempos (base)*
***handler**: Rutina que atiende el evento a su vencimiento (no puede ser NULL).*
***base**: Base de tiempo elegida (DEC=decimas - SEG=segundos - MIN=minutos)*
Retorno: **void**

Nota: Al vencer un temporizador, éste no se reinicia. Si se requiere seguir contando se debe volver a iniciar

Reiniciar (o reconfigura) el tiempo de un timer

```
void SetTimer ( uint8_t event , timer_t t )
```

*Reinicia el timer identificado por **event** y al transcurrir el tiempo especificado por **t** (según **base**) se llama a la función indicada por **handler** (utilizada en TimerStart).*

Parámetros:

***event**: Número de timer asociado al evento. Valor entre 0 y 31*

***t**: Tiempo reasignado al evento. (depende de la base de tiempos asignada en TimerStart)*

Retorno: **void**

Obtener valor de un timer

```
uint32_t GetTimer ( uint8_t event )
```

*Retorna el valor actual del del timer asociado al evento **event**.*

Parámetro: ***event**: Numero de evento entre 0 y 31*

Retorno: valor del timer (según base)

Dormir/despertar un timer

```
void StandByTimer ( uint8_t event , uint8_t accion)
```

Suspende/continua el timer. NO lo reinicia, solo pone o lo saca del estado de stand by

Parámetros:

***event**: Numero de evento entre 0 y 31*

***accion**: RUN lo arranca, PAUSE lo pone en stand by*

Retorno: **void**

Detener un timer

```
void Timer_Stop (uint8_t event)
```

*Detiene el timer identificado por **evento.** Si el timer está apagado no tiene efecto*

Parámetro: ***event**: Numero de evento entre 0 y 31*

Retorno: **void**

Detener todos los timer

```
void Timer_Close (void)
```

Detiene TODOS los timers activos.

Parámetros: **void**

Retorno: **void**

Entradas Analógicas

lectura del termistor

```
int16_t Temperatura ( void )
```

Retorna el valor de temperatura del termistor

Parámetros: **void**

Retorno: temperatura (rango -50 a 120)

ADC Externa

```
int16_t ADC_Externa ( void )
```

Retorna el valor de tensión asociado a la bornera.

Parámetro: **void**

Retorno: Lectura de cuentas (rango 0 a 50v - *número de cuentas de 12bits).*

Comunicación – Puerto Serie (UART)

Configurado a 9600 8,N,1 sin control de flujo

Transmisión vía PS

```
int16_t Transmitir ( uint8_t com, const void* datos, uint8_t cant)
```

Envía una trama de datos a través del Puerto Serie, según configuración prestablecida

Parámetros:

***com**: Puerto que será utilizado [UART0 o UART1]*
***datos**: puntero a los datos a transmitir*
***cant**: cantidad de datos a transmitir*

Retorno:

0 por éxito
-1 por Error (datos excedidos)

Recepción vía PS

```
int16_t UART_PopRX(uint8_t com);
```

Recibe un carácter de la UART definida

Parámetro:

***com**: Puerto a leer [UART0 o UART1]*

Retorno:

>=0 Carácter recibido
-1 no hay datos para leer

Salidas de Texto

LCD:

```
void LCD_Display (const char *str, uint8_t line, uint8_t pos)
```

Muestra una string en un LCD de 2 x 16.

Parámetros:

***str**: dirección de comienzo de la string*
***line**: número de renglón del Display (DSP0: renglón superior, DSP1: renglón inferior)*
***pos** posición relativa dentro del renglón*

Retorno: **void**

Display7seg:

```
void Display (unsigned int val, unsigned char dsp)
```

Muestra el valor que recibe a través de val en el display indicado por dsp.
Cada display está constituido por 3 dígitos (DSP0 = 0 y DSP1 = 1)

Parámetros

***val:** Valor a mostrar en el display elegido*
***dsp:** Display elegido para mostrar el valor Val*

Retorno: **void**