SISTEMAS EMPOTRADOS Y UBICUOS

PRACTICA 4

EDUARDO MORA GONZÁLEZ

Ejercicio 1

En este ejercicio usamos diversos componentes, los componentes que tienen código especifico en el fichero Events.c son:

- **TimerInt:** Al tener que hacer distintas acciones (una cada segundo y otra cada 0.4 segundos) se ha definido un *TimerInt* que genere una interrupción cada milisegundo y depende en el milisegundo que se encuentre se asigna un valor a la variable opción.

```
int ms = 0;
int opcion = 0;
void Timer1_OnInterrupt(void)
{
    ms++;

    if (ms % 4 == 0) opcion = 1;

    if (ms % 10 == 0) opcion = 2;

    if (ms == 20) ms = 0;
}
```

- **ADC:** para la lectura del dato del potenciómetro se ha creado una bandera para que cada vez que se lea el dato se avise al main.c de su finalización:

```
volatile bool AD_finished;
void POTENCIOMETRO_OnEnd(void) {
    AD_finished = TRUE;
}
```

La lógica del fichero main.c es la siguiente:

- **Declarar las clases y variables externas**: la clase para realizar la normalización de los datos y las variables son las usadas en el Events.c

```
#include "UART.h"
extern volatile bool AD_finished;
extern int opcion;
```

- **Definir las variables donde almacenar los datos, y definir los mensajes que se enviarán:**

```
word valor_potenciometro;
word valor_temperatura;

char mensajePotenciometro[] = "Potenciometro 1: ";
char mensajeTemperatura[] = "Temperatura: ";
```

- **Si la opción es la primera:**
    - Se pone la bandera de confirmación de lectura a false.
    - Se pone la variable opción a la por defecto para que no entre otra vez hasta que le vuelva a tocar.
    - Esperamos a que se lea el dato.
    - Obtenemos el valor.
    - Enviamos el mensaje.
    - Normalizamos el dato y lo enviamos.
    - Enviamos fin de línea y salto de línea.

```
if (opcion == 1){

    AD_finished = FALSE;

    opcion = 0;

    POTENCIOMETRO_Measure(FALSE);

    while (!AD_finished) {
    }

    POTENCIOMETRO_GetValue(&valor_potenciometro);

    for (int i = 0; i < sizeof(mensajePotenciometro); i++) while (AS1_SendChar((byte) mensajePotenciometro[i]) != ERR_OK) {}

    UART_Write_Numero_Int(valor_potenciometro);

    while(AS1_SendChar(10) != ERR_OK) {}
    while(AS1_SendChar(13) != ERR_OK) {}

}
```

- **Si la opción es la primera:**
    - Se pone la variable opción a la por defecto para que no entre otra vez hasta que le vuelva a tocar.
    - Esperamos a que se lea el dato.
    - Obtenemos el valor.
    - Enviamos el mensaje.
    - Normalizamos el dato y lo enviamos.
    - Enviamos fin de línea y salto de línea.

```
if (opcion == 2){

    opcion = 0;

    TEMPERATURA_Measure(TRUE);

    TEMPERATURA_GetValue(&valor_temperatura);

    for (int i = 0; i < sizeof(mensajeTemperatura); i++) while (AS1_SendChar((byte) mensajeTemperatura[i]) != ERR_OK) {

    UART_Write_Numero_Int(valor_temperatura);

    while(AS1_SendChar(10) != ERR_OK) {}
    while(AS1_SendChar(13) != ERR_OK) {}
}
```

¿Qué problemas pueden aparecer en este programa?

Según he observado los problemas que pueden ocurrir son:

- Se paralice el programa ya que el evento de finalización de lectura del potenciómetro falle.
- Igual que en la practica anterior, exista un retardo en el envío de los datos por el AsynchroSerial.
- No se esté leyendo los datos de los canales correctos

Ejercicio 2

Este ejercicio es similar que el anterior, pero se ha añadido el componente PWM y se ha usado de la siguiente forma:

```
POTENCIOMETRO_GetValue(&valor_potenciometro);

PWM1_SetRatio16(valor_potenciometro);

 TEMPERATURA_GetValue(&valor_temperatura);

 PWM1_SetRatio16(valor_temperatura);
```

Cada vez que se obtiene el dato, se modifica con el valor le intensidad del Led.