PRACTICA 2: Implementación de la arquitectura SW Round-Robin con interrupciones

En esta práctica se implementará el sistema propuesto utilizando una arquitectura software del tipo Round-Robin con interrupciones. En esta arquitectura todas las acciones se programan dentro de un único lazo donde se atiende a los eventos por consulta. Las subrutinas de atención a las interrupciones activan un "flag" que indicará que se ejecute una determinada acción dentro del lazo principal. Recordad que es responsabilidad del programador volver a desactivar este "flag" una vez se realice la acción correspondiente.

Para realizar la programación se utilizarán las siguientes interrupciones:

- Una interrupción para el Timer que se activará cada 10ms (100Hz)
- Una interrupción para la UART que únicamente se activará cuando se reciban datos.

Se utilizará la técnica Round-Robin de programación mediante interrupciones minimizando el código que se tenga que ejecutar dentro de la interrupción, sólo las tareas críticas, y se activará el FLAG correspondiente para poder realizar el resto de tareas en el programa principal.

Al tener que trabajar con interrupciones tendremos que programar utilizando los comandos del ESP-IDF Programming Guide, ya que los comandos de Arduino no gestionan bien las interrupciones del ESP32.

Consulta de la última versión del ESP-IDF, está disponible on-line y en versión PDF:

Versión on-line: https://docs.espressif.com/projects/esp-idf/en/latest/

Versión pdf: https://readthedocs.com/projects/espressif-esp-idf/downloads/pdf/latest/

Estructura del software

A continuación, se muestra las acciones que tiene que realizar el software. Se recomienda que se siga esta organización y que se implementen como funciones aquellas acciones que requieran de varias líneas de código para su implementación.

ISR_Timer{
- Capturar muestra con el ADC y guardarla en una variable global
- Activar un flag_timer

}

ISR_Uart{
- Leer dato recibido por la UART y guardarlo en una variable global
- Activar un flag_uart
- Limpiar el estado de petición de interrupción

}

setup{
- Inicializar M5Stack
- Inicializar puerto serie (A)
- Configurar el ADC

- Calibrar el sensor de gas
- Desactivar puerto serie con Serial.end
- Configurar UART con IDF
- Configurar Timer
- Habilitar interrupciones del timer
- Habilitar interrupciones de la UART

}

loop{

- Si está activado el flag_Timer:
  - Convertir la muestra a voltios
  - Procesar las últimas 200 muestras
  - Si se han procesado 100 muestras activar flag_LCD
  - Desactivar flag_Timer
- Si se han realizado 100 capturas (flag_LCD):
  - Mostrar los datos por la pantalla LCD del M5Stack
  - Desactivar flag_LCD
- Si está activado el flag_UART
  - Si se ha recibido el carácter "a" por puerto serie:
    - Mostrar los datos en el monitor serie
  - Desactivar flag_UART

}

Se programará utilizando interrupciones para los siguientes eventos:

- Interrupción periódica con el Timer para indicar al sistema que tiene que realizar una nueva medida del sensor. Se utilizará la función **timerAlarmWrite()** programada para que genere la interrupción cada 10ms.
- Interrupción de la UART. Programe la UART0 para que reciba los datos mediante interrupción. Tenga en cuenta que la UART0 es la que utiliza el programa monitor para comunicarse con el IDE de Arduino. Para que funcione correctamente tiene que desactivar el Serial de Arduino mediante la función **Serial.end()**

NOTA1: Para facilitar la programación no desactive el Serial de Arduino hasta no completar la calibración del sensor, así puede utilizar las funciones **Serial.print()** para el envío de la información de la calibración. Una vez se completa la calibración, y envíados los valores de la misma por el Serial del monitor de Arduino ya puede desactivarlo y configurar la UART0 mediante comandos IDF para que funcione por interrupciones.

NOTA2: Tenga en cuenta que cuando configure la UART0 mediante IDF, sólo se pueden enviar caracteres con la función **uart_write_bytes()**, y por tanto, tendrá que utilizar su propia función para enviar los datos numéricos en forma de carácter o string. Pruebe a utilizar la función **itoa(variable, buf, 10)** donde **variable** es la variable entera (uint16_t) a mostrar, **buf** es la variable array char donde se obtiene la salida y **10** indica que se utiliza base 10 (decimal) para hacer la conversión.

Para los datos en formato **float** y **double** necesitará otra función diferente ya que el formato en el que se guarda la información es completamente diferente al de los enteros. Pruebe a realizar su propia función o utilizar una de las funciones del sistema, como es **snprintf (buf, sizeof(buf), "%f", variable)** donde **variable** es la variable a mostrar, en este caso float, **buf** es el puntero donde está el array char donde se obtiene la salida, **sizeof(buf)** es el tamaño del array a mostrar y **"%f"** es el formato con el que queremos que se muestre el número, en los enlaces siguientes puede ampliar el formato a utilizar. (Esta función también serviría para mostrar las variables enteras)

Puede encontrar más información en la página:

http://www.nongnu.org/avr-libc/user-manual/group__avr__stdlib.html

http://www.cplusplus.com/reference/cstdio/snprintf/

http://www.cplusplus.com/reference/cstdio/printf/#compatibility

NOTA3: Para habilitar la interrupción de la UART deberá realizar las siguientes tareas:

1. Configurar los parámetros de la UART
2. Asignación del pin correspondiente
3. Instalar el driver definiendo el buffer adecuado para la aplicación
4. Realizar una escritura por el puerto
5. Liberar el controlador de la subrutina de interrupción
6. Registrar la subrutina de interrupción
7. Habilitar las interrupciones en recepción del puerto serie

NOTA4: Para habilitar la interrupción del Timer sólo deberá realizar las siguientes tareas:

1. Definir el puntero al objeto "timer" igualado a NULL.
2. Asignar y arrancar el timer
3. Asignar la subrutina de interrupción al timer
4. Programar la alarma del timer
5. Habilitar la alarma del timer

Documentación adicional:

- Funciones para micros:
  - http://www.nongnu.org/avr-libc/user-manual/group__avr__stdlib.html
  - http://www.cplusplus.com/reference/cstdio/snprintf/