Trabajo Práctico Final
RTOS 1

Autor: Luis Mariano Campos

Contextualización

Objetivo de la aplicación:

- La tarea recibe comandos por la UART.
- Estos comandos son interpretados para que realice las siguientes acciones :
  - Habilita/Deshabilita un motor paso a paso.
  - Establece el sentido de giro del eje del motor : Derecha / Izquierda.
  - Envía un tren de pulsos para girar una cierta cantidad de pasos, cada pasos corresponde a 7,5°

Implementación

- La aplicación recibe comandos mediante el empleo de una rutina de servicio de interrupción por recepción de un carácter por la UART y lo envía a una cola (xQueueSendFromISR).

```
void onRx(void *noUsado) {
    BaseType_t xHigherpriorityTaskWoken = pdFALSE;
    char data = uartRxRead(UART_USB);
    printf("Recibimos un dato:%c por la UART\r\n", data);
    xQueueSendFromISR(cola_rec, &data, &xHigherpriorityTaskWoken);

    if (xHigherpriorityTaskWoken == pdTRUE) {
        portYIELD_FROM_ISR(xHigherpriorityTaskWoken);
    }

}
```

Implementación

- Existe una tarea que se bloquea al empezar su bucle sin fin a la espera de que llegue algún carácter a la cola.
- Cuando la función xQueueReceive retorne debido a la llegada de un carácter a la cola, éste se añadirá a la cadena usada para guardar el mensaje y si éste es el avance de línea se añadirá el terminador nulo a la cadena del mensaje.

Implementación

Tarea que almacena en un buffer los datos que va recibiendo de la cola

```
void processRxSerieTask(void * taskParmPtr) {

    static uint8_t index = 0;
    char car_rec;
    while (TRUE) {
        if (xQueueReceive(cola_rec, &car_rec, portMAX_DELAY) == pdTRUE) {
            message[index] = car_rec;
            if (message[index] == '\n') {
                message[index + 1] = '\0';
                index = 0;
                xSemaphoreGive(sinsynchronizeTasks);
            } else {
                index++;
            }

        }
    }

}
```

Implementación

- Se debe tener en cuenta que es imprescindible verificar que la llamada a la función xQueueReceive ha devuelto pdTRUE por la llegada de un carácter antes de procesarlo. Si se ha desbloqueado por un timeout, la función devolverá pdFALSE y no se hace nada, volviendo a esperar la llegada de un carácter a la cola.
- Una vez detectado el avance de línea se hará empleo de un semáforo para sincronizar con una tarea la cual se encargará de interpretar o procesar el comando recibido y llamar a las funciones del driver ya implementadas en una biblioteca.

```
void processCommndTask(void * taskParmPtr)
```

Implementación

- Para generar un tren de pulsos se creó un temporizador de software para que se ejecute una determinada función periódicamente.
- Esta función periódica pondrá en un estado alto o bajo a un GPIO según corresponda la secuencia.
- Es importante aclarar que una vez que se determine la cantidad de pulsos se inicia la ejecución de dicha función periódica con xTimerStart.
- Se detendrá cuando se haya terminado de enviar el número de pulsos solicitados con xTimerStop.

Implementación

Temporizador de Software - Tarea Periódica

```
static TimerHandle_t xAutoReloadTimer;
```

```
static void pvrAutoReloadTimerCallback(TimerHandle_t xTimer) {
    static uint32_t i = 0;
    static bool_t flag = TRUE;
    if (i < pulse) {
        if (flag == TRUE) {
            gpioWrite(motor.pulsePin, TRUE);
            flag = FALSE;

        } else {
            gpioWrite(motor.pulsePin, FALSE);
            flag = TRUE;
            i++;
        }
    } else {
        i = 0;
        xTimerStop(xTimer, 0);
    }

}
```

Formato de Comandos

La aplicación identifica como comando hasta que detecta un '\n'.

Convención de la disposición que deben tener los caracteres que conforman los comandos

1° Caracter: Es M en representación a Motor

2° Caracter: Es E: Enable o D: Disable

Ejemplos:

ME\n : habilita el motor

MD\n :deshabilita el motor

Formato de Comandos

El segundo tipo de comando es el siguiente:

1° Caracter: Es M en representación a Motor

2° Caracter: Corresponde al sentido de giro del eje del motor que pueden ser L: Left R:Right

3° Caracter: Indica que los 4 próximos caracteres corresponden a la cantidad de pulsos que se debe generar o bien el angulo que debe girar el eje del motor.

4° al 7° Caracter : cantidad de pulsos o el ángulo de giro.

Formato de Comandos

Ejemplo:

MLP0010: El eje del motor gira a la izquierda 10 pasos.

MRP0015: El eje del motor gira a la derecha 10 pasos.

MLA0045: El eje del motor gira a la izquierda 45°.

MRA0090: El eje del motor gira a la derecha 90°.