# Ejercicio 1: Manejo de Prioridades. Resuelto

Se desea implementar un juguete que tenga una señal luminosa multicolor, es decir, se pretende usar la combinación de 3 color primarios RGB (Red = Rojo, Green = Verde y Blue = Azul) para crear cada color. Para la paleta de colores de esta señal luminosa, es suficiente con que pueda representar hasta 7 colores, más la posibilidad de no mostrar ningún color, equivalente a estar apagado.

La primera versión de este juguete, una vez encendido el equipo, el mismo estará continuamente mostrando un color distinto, sin necesidad de intervención del usuario. La secuencia no es aleatoria y debe respetar la siguiente tabla.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | Color |  | Orden de encendido | Tiempo de permanencia |
| R | G | B |
| Apagado | Apagado | Prendido | 1 | 500 ms |
| Apagado | Prendido | Apagado | 2 | 500 ms |
| Apagado | Prendido | Prendido | 3 | 500 ms |
| Prendido | Apagado | Apagado | 4 | 500 ms |
| Prendido | Apagado | Prendido | 5 | 500 ms |
| Prendido | Prendido | Apagado | 6 | 500 ms |
| Prendido | Prendido | Prendido | 7 | 500 ms |

Al mismo tiempo, el juguete, necesita realizar otras actividades, entre las cuales se encuentra una tarea de diagnóstico. Del departamento de ingeniería simplemente nos envían el siguiente código, que es la tarea de diagnóstico que debemos compilar con el resto de nuestro proyecto:

**void** **Diagnostico**(**void** \*pvParameters)

{

vPrintString("Comienza el diagnostico\n");

//Loop infinito

**while**(1)

{

//IMPORTANTE:

//Este loop pretende simular un conjunto de funciones que

//insumen un tiempo para realizar el diagnostico del equipo //Pero NO es aceptable tener al micro dentro de esta tarea //haciendo nada, esto es solo un ejemplo.

**if**(!tiempo\_de\_diagnostico)

{

vPrintString("Termina el diagnostico\n"); vTaskDelay(TIEMPO\_DE\_DIAGNOSTICO\_ms/portTICK\_RATE\_MS); tiempo\_de\_diagnostico = TIEMPO\_DE\_DIAGNOSTICO\_ms;

vPrintString("Comienza el diagnostico\n");

}

}

}

Las condiciones que nos pasa el departamento de ingeniería son las siguientes:

1. La tarea debe ejecutarse cada 2 segundos, cuando el micro no esté atendiendo otras tareas de mayor prioridad.
2. La tarea, una vez iniciada, insume 2 segundos. Recuerde que el juguete debe seguir cumpliendo con sus tareas de mayor prioridad mientras se hace el diagnóstico.
3. Para contabilizar los 2 segundos de ejecución, deberá utilizarse el tick de sistema del FreeRTOS, haciéndose necesaria la implementación del recurso de FreeRTOS siguiente:

**void** **vApplicationTickHook** ( **void** )

{

**if**(tiempo\_de\_diagnostico)

tiempo\_de\_diagnostico--;

}

**Nota: La tarea de diagnóstico no hace falta hacerla, hay que hacer copypaste del código de este enunciado.**

Se pide:

1. Diseñar el circuito necesario, contando con un LED RGB con número de parte AAAF3528QBFSEJ3ZGW y transistores BC817-25. (pueden usar la placa de info2 o el stick nuevo)
2. Utilizando FreeRTOS, asignar la estrategia y prioridades necesarias a las tareas que correrán en el sistema.
3. Realizar la/s tarea/s que logren el objetivo de hacer la secuencia continua de colores.

# Ejercicio 2: Manejo de Queue. Resuelto

Al juguete desarrollado en el ejercicio anterior, quieren hacerle una modificación de modo que la secuencia de colores no sea ejecutada en forma continua. La nueva versión va necesitar un pulsador a través del cual el usuario le dá la orden para ir cambiando el color.

El departamento de ingeniería ha puesto las siguientes condiciones condiciones:

1. Se trata de un pulsador que no va a ser apretado con mucha frecuencia, posiblemente un niño pueda apretarlo 2 veces en 1 segundo.
2. No se ha resuelto el antirrobote por hardware y deberá realizarse por software.
3. Para futuras implementaciones, se desea contar con la información de cuánto tiempo se presionó el pulsador. Este tiempo es suficiente con expresarlo en unidades de ms.
4. No está permitido utilizar variables globales para almacenar la información del pulsador y tiempo que estuvo apretado.

Se pide:

* 1. Modificar la tarea de manejo de los colores del ejercicio 1, para que sea capaz de reaccionar ante la presión de una tecla (la información del tiempo deberá estar disponible, aunque no se use por ahora)
  2. Utilizar Yakindu SCT para realizar el diagrama de estados, generar el software asociado y completar.
  3. Realizar la tarea que sea capaz de escanear el pulsador y avisar el tiempo que estuvo apretado. Especificar en qué se basa para determinar el tiempo de escaneo.

# Ejercicio 3: Manejo de tres pulsadores y tres leds. TP2 a entregar

Al juguete desarrollado en el ejercicio anterior, quieren hacerle una modificación de modo que la secuencia de colores no sea ejecutada en forma continua. La nueva versión va necesitar un pulsador a través del cual el usuario le dá la orden de invertir la secuencia, un pulsador para detener la secuencia y uno para arrancar la secuencia.

Secuencia normal:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | Color |  | Orden de encendido | Tiempo de permanencia |
| R | G | B |
| Apagado | Apagado | Prendido | 1 | 700 ms |
| Apagado | Prendido | Apagado | 2 | 700 ms |
| Prendido | Apagado | Apagado | 3 | 700 ms |
| Prendido | Apagado | Prendido | 4 | 700 ms |
| Apagado | Prendido | Prendido | 5 | 700 ms |
| Prendido | Prendido | Apagado | 6 | 700 ms |
| Prendido | Prendido | Prendido | 7 | 700 ms |

El departamento de ingeniería ha puesto las siguientes condiciones condiciones:

1. Los pulsadores no van a ser apretados con mucha frecuencia, posiblemente un niño pueda apretarlo 2 veces en 1 segundo.
2. No se ha resuelto el antirrobote por hardware y deberá realizarse por software.
3. No está permitido utilizar variables globales para almacenar la información de los pulsadores.

Se pide:

* 1. Utilizar Yakindu SCT para realizar el diagrama de estados, generar el software asociado y completar.