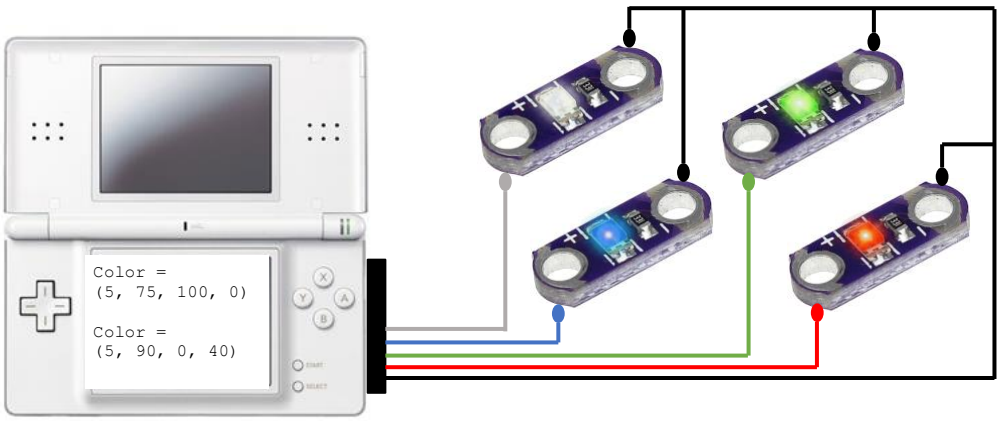
Problema 34: Combinaciones de color (Ex. 2ª Conv. 2020-21)

Se propone conectar a la NDS un conjunto de minileds de la serie Arduino LilyPad® (Sparkfun), para generar un color determinado a partir de la combinación de intensidades de los distintos colores base de los leds. Cada led tiene un color base determinado (blanco, azul, verde, rojo, etc.), pero si se juntan varios leds y se regula la intensidad de luz de cada tipo, se puede formar un color combinado, por ejemplo, un verde Esmeralda, que se podría conseguir fijando las siguientes intensidades: blanco → 5%, azul → 75%, verde → 100%, rojo → 0%.



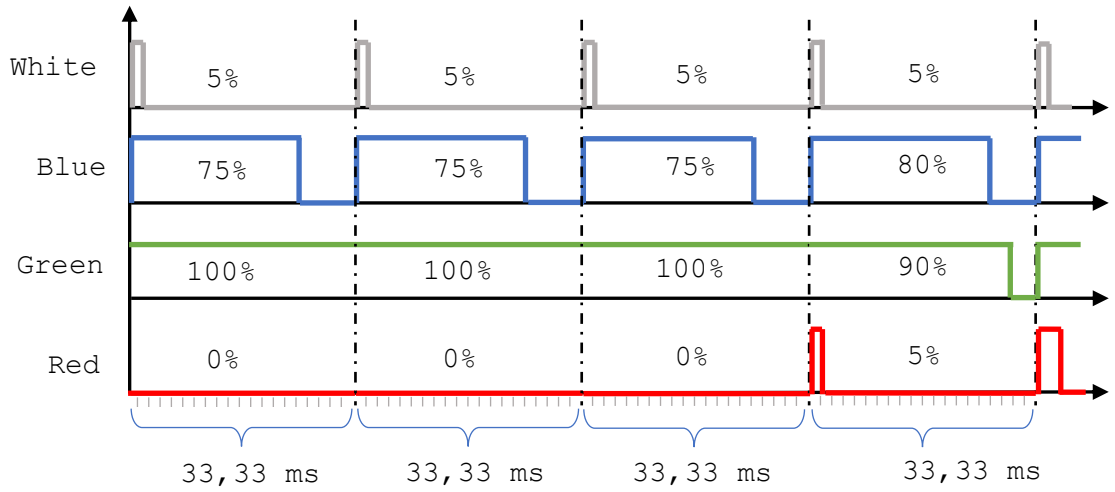
El esquema anterior muestra una conexión de la NDS con cuatro tipos de leds. Sin embargo, existe toda una gamma de colores base para los leds, de modo que el programa de control deberá estar preparado para manejar un número predeterminado de tipos de led, que podrá variar entre 1 y 6 tipos.

Hay un cable de masa común (negro) para todos los leds, y un cable de señal por cada tipo de led. Cada cable de señal solo puede valer 0 (apagado) o 1 (encendido). Los cables de señal estarán conectados con los bits de menor peso de un registro de entrada/salida de 16 bits, de nombre simbólico REG_IO. Por ejemplo, se puede utilizar el bit 0 para el rojo, el bit 1 para el verde, el bit 2 para el azul y el bit 3 para el blanco. Obviamente, el número de bits utilizados dependerá del número de tipos distintos de led a controlar. El resto de bits del registro pueden tener otras funciones (para tareas independientes), de modo que no se deberán modificar.

Al encender un led, éste emite luz a su máxima intensidad. Para regular su nivel de intensidad podemos usar la técnica de la modulación por ancho de pulso (PWM: *Pulse Width Modulation*), que consiste en encender el led durante una fracción del tiempo, según el porcentaje de la intensidad que se requiera.

Para que esta técnica funcione, el proceso debe ser repetido a intervalos muy breves de tiempo para el ser humano. En el contexto del problema actual, se establece que los intervalos serán de 33,33 milisegundos, aproximadamente. Cada uno de estos intervalos estará dividido en 20 subintervalos, de modo que se podrán generar 21 niveles de intensidad entre el 0% y el 100%, en incrementos de 5%.

El siguiente cronograma muestra cuatro intervalos con diferentes ejemplos de intensidad para 4 tipos de led. El valor del bit de control de cada tipo de led se indica con un nivel bajo (0) o alto (1) de la señal, para diferentes momentos en el tiempo. Los 3 primeros intervalos repiten los mismos valores de intensidad, mientras que el cuarto intervalo introduce pequeñas variaciones. Estos ejemplos muestran diferentes formas de onda (anchos de pulso) para cada tipo de led.



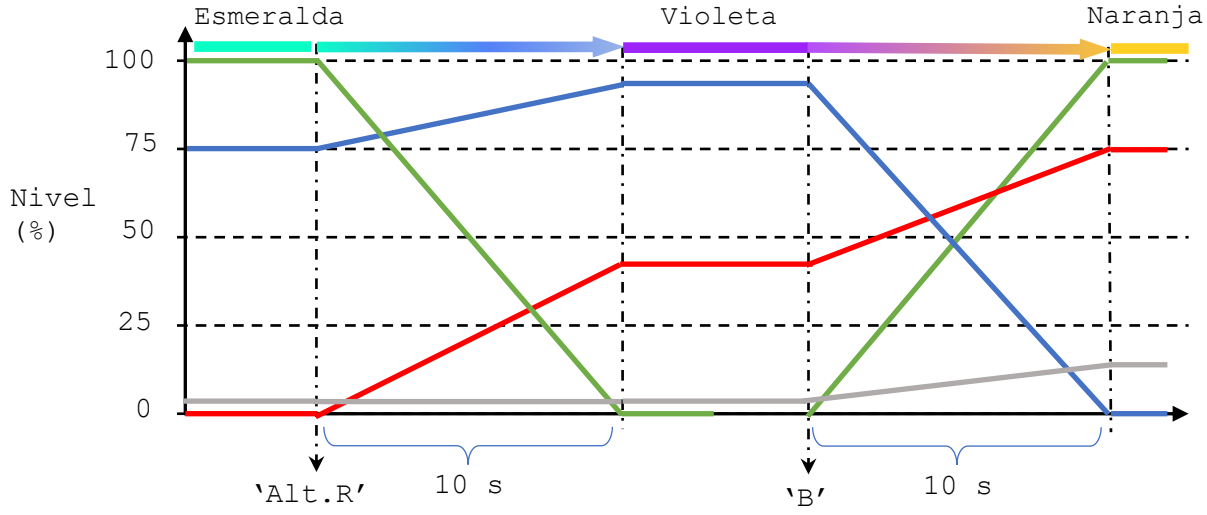
El programa a realizar debe controlar periódicamente el valor de los bits de cada tipo de led, utilizando para ello la rutina de servicio de interrupción del *timer* 0, que se programará a la frecuencia adecuada.

El programa también permitirá al usuario cambiar el color generado. Para ello se definirá un vector con diez combinaciones de niveles de intensidad. Las siguientes líneas de código fuente en C son un ejemplo de definición de este vector:

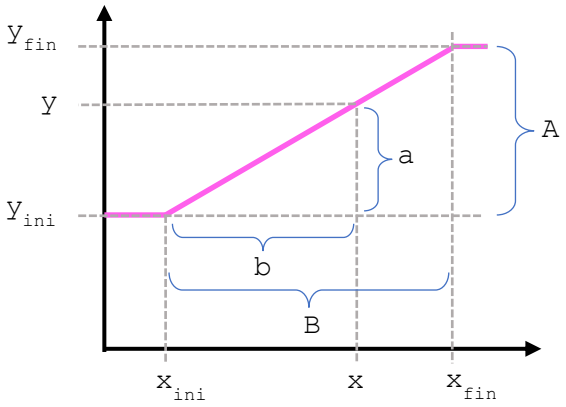
```
#define NUM_LEDS      4                 // número de leds
                // vector colores preestablecidos (R, G, B, W)
unsigned char colores[10][NUM_LEDS]={ {100,  0,  0,  0},       // Rojo
                                      {75, 100,  0, 15},       // Naranja
                                      {100, 100,  0,  0},      // Amarillo
                                      {15,  65,  0, 20},       // Caqui
                                      {0,  100,  0,  0},       // Verde
                                      {0,  100, 75,  5},       // Esmeralda
                                      {0,  15,  70, 30},       // Azul claro
                                      {0,  0,  100,  0},       // Azul
                                      {40,  0,  90,  5},       // Violeta
                                      {0,  0,  0,  100}        // Blanco
                                    };
```

El usuario podrá cambiar de combinación pulsando alguno de los botones de la NDS. Al pulsar un botón, el programa deberá pasar del color actual al solicitado por el usuario de manera gradual, durante un periodo de tiempo perceptible, por ejemplo, 10 segundos.

El siguiente esquema muestra dos ejemplos de variaciones de color, al pulsar dos botones (‘`Alt. Right`’, ‘`B`’) en diferentes momentos:



Para conseguir generar los niveles intermedios durante el tiempo que dure la transición de colores, hay que usar interpolación. Para calcular el valor interpolado de la intensidad de cada tipo de led, se puede usar la propiedad de conservación de la ratio entre dos lados homólogos de dos triángulos semejantes. El siguiente esquema muestra cómo identificar estos dos triángulos en un gráfico de interpolación, junto con la fórmula de la propiedad:



$$\frac{A}{B} = \frac{a}{b}$$

El valor a calcular es `y` (nivel de intensidad) respecto de `x` (tiempo). Los niveles inicial y final se representan como y_{ini} e y_{fin}. Los valores concretos de tiempo x_{ini} y x_{fin} no son relevantes, puesto que lo importante es contar el tiempo relativo (`b`) desde que se ha pulsado el botón. Para contar el tiempo, se sugiere utilizar un contador de los intervalos usados en la generación de los pulsos de control PWM. Por ejemplo, para medir el paso de 10 segundos (aprox.) se deberán contar 300 intervalos.

Se recomienda que el control de la gradación se realice desde el programa principal. Sin embargo, será necesario que exista una sincronización con la RSI del *timer* 0, puesto que el programa principal **no** debe cambiar los niveles de intensidad en medio de un intervalo de generación de pulsos PWM. Para realizar esta sincronización de forma eficiente, se puede

usar la rutina `swiWaitForIRQ()`, que pone el procesador en reposo hasta que se produzca una interrupción cualquiera.

Después de cada proceso de cambio de color, por la pantalla inferior de la NDS se deberán mostrar los nuevos niveles de intensidad de cada tipo de led, con el siguiente formato:

```
color = (P_{n-1}, P_{n-2}, ..., P_0)
```

donde P_i es el porcentaje de intensidad del led *i*-ésimo. También hay que dejar una línea vacía respecto a la línea del color anterior. A modo de ejemplo, a continuación se muestran dos escrituras consecutivas, correspondientes a los colores Esmeralda y Violeta:

```
color = (5, 75, 100, 0)

color = (5, 90, 0, 40)
```

Por último, el programa debe ir entrelazando la generación de los colores con la ejecución de unas tareas independientes. Sin embargo, mientras se esté realizando la gradación de colores, no será necesario invocar a las tareas independientes.

Se dispone de las siguientes rutinas ya implementadas:

Rutina	*Descripción*
`inicializaciones()`	Inicializa el *hardware* (pantalla, interrupciones, etc.)
`activar_timer0(int frec)`	Activa el funcionamiento del *timer* 0 a la frecuencia indicada por parámetro (en Hz)
`tareas_independientes()`	Tareas que no dependen de la gestión de los colores, por ejemplo, cálculo de la humedad relativa del aire (tiempo de ejecución < 100 ms)
`scanKeys()`	Captura el estado actual de los botones de la NDS
`keysDown()`	Devuelve un patrón de bits (`int`) con los botones activos
`indice_boton(int keys)`	Devuelve un índice entre 0 y 9 (`unsigned char`) de botón, según un patrón de bits `keys` que indica los botones activos; si no hay ningún botón activo, devuelve 255.
`swiWaitForIRQ()`	Espera hasta la próxima interrupción
`swiWaitForVBlank()`	Espera hasta el próximo retroceso vertical
`printf(char *format,...)`	Escribe un mensaje en la pantalla inferior de la NDS

Además, se propone el uso de las siguientes constantes y variables globales (junto con las sugeridas anteriormente):

```
#define BITS_LEDS     0xF             // bits de los leds a 1 (4 bits)
#define NUM_INTERPOL  300             // número de interpolaciones

unsigned char vectPWM[NUM_LEDS];      // vector de porcentajes PWM
```

La definición `NUM_INTERPOL` corresponde al número de intervalos PWM necesarios para realizar todas las interpolaciones de una gradación de colores.

El vector `vectPWM[NUM_LEDS]` permitirá almacenar los niveles de intensidad actuales de los distintos tipos de led. Hay que recordar que `NUM_LEDS` es una constante (*define*) que se puede cambiar antes de cada compilación. Esto significa que el programa debe estar diseñado para poder ser recompilado con cualquier número de tipos de led (no tiene por qué ser 4).

Se pide:

Programa principal y variables adicionales en C; RSI del *timer* 0 en ensamblador.