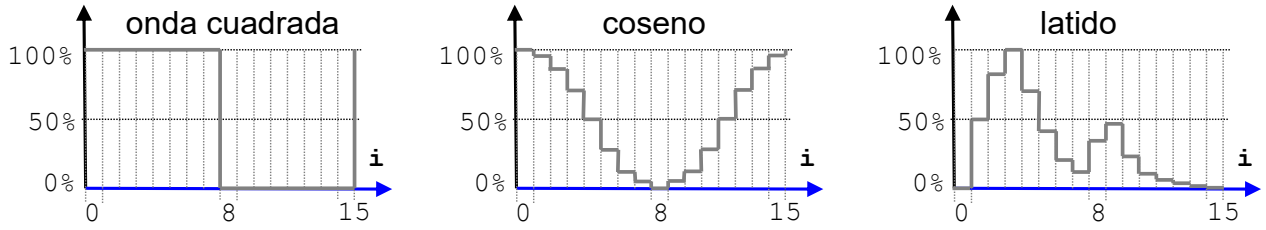
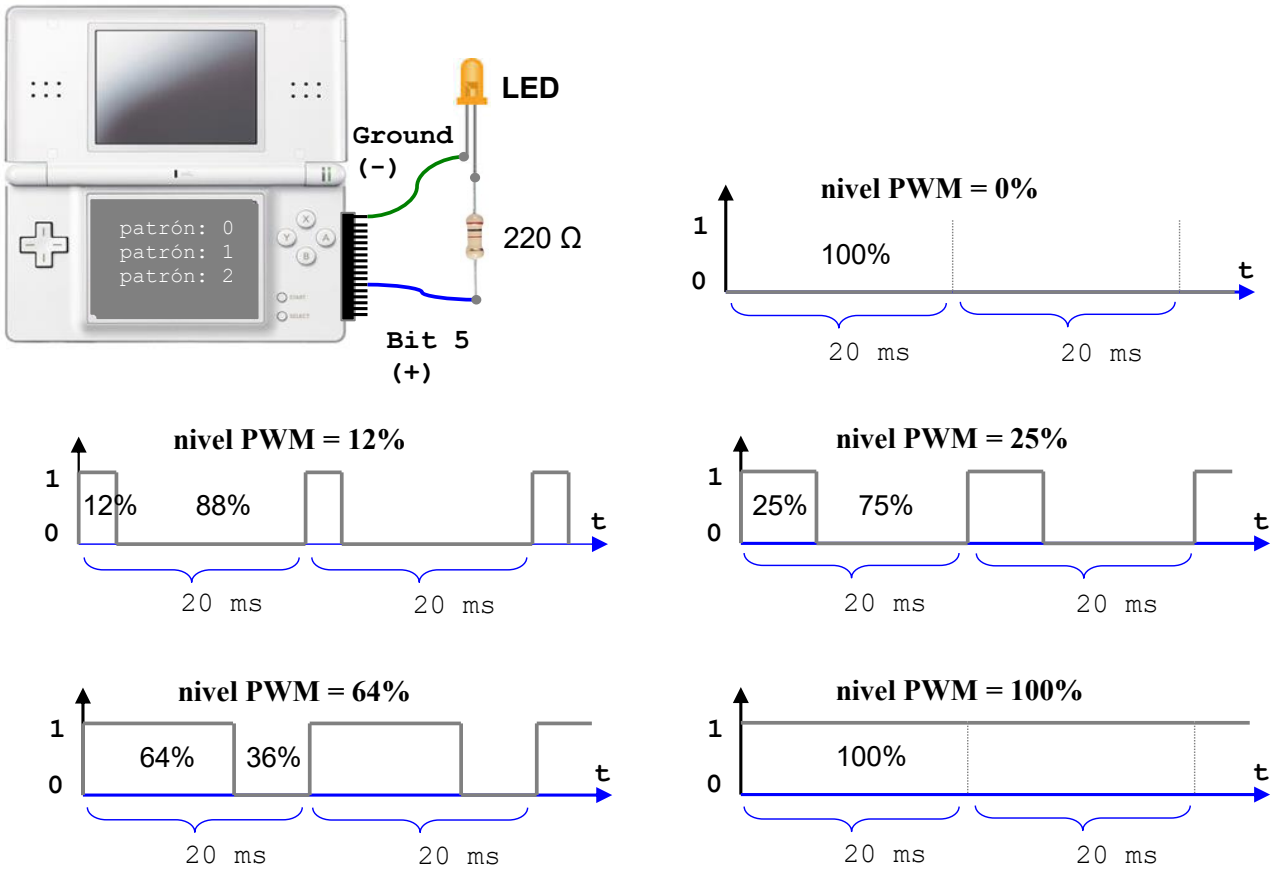
Problema 18: Luz LED regulada por PWM (Ex. 2ª Conv. 2015-16)

Se propone controlar la intensidad (o brillo) de una luz LED con la NDS, para que emita, repetidamente, una determinada secuencia de niveles del brillo. El programa a realizar permitirá emitir diferentes secuencias, que llamaremos *patrones*. A continuación se muestran tres patrones diferentes, con 16 valores de brillo por cada patrón (la coordenada horizontal indica el índice de cada valor):



Los niveles de brillo se codificarán como el tanto por ciento de la intensidad máxima que emite el LED. A continuación se muestra un esquema de la estructura del sistema, junto con unos cronogramas de ejemplo del control de brillo (nivel PWM):



Conectados a la NDS del esquema, se han representado los pins de salida de un dispositivo

que permite generar 16 voltajes digitales, es decir, 0 o 5 voltios en cada pin, más un pin extra de masa (*Ground*). El voltaje de los pins se controlará con el valor de los bits de un registro de salida de 16 bits, etiquetado como REG_DOUT, donde el valor del bit 0 indicará el voltaje el pin inferior, y el resto de bits se asignarán consecutivamente a los siguientes pines (hacia arriba). El pin superior es para conectar la masa de los circuitos.

El ánodo del LED (+) se ha conectado a una resistencia de 220 ohmios, y esta resistencia se ha conectado al pin del bit 5, mientras que el cátodo del LED (-) se ha conectado a masa. Sin embargo, con esta configuración solo podemos apagar el LED (bit 5 = 0) o encenderlo a su máxima intensidad (bit 5 = 1). Es decir, no es posible indicar directamente un porcentaje del brillo del LED usando únicamente el estado del bit 5 del registro de salida.

Para solucionar este problema se propone aplicar la técnica de modulación por ancho de pulso (PWM = *Pulse Width Modulation*), que consiste en encender y apagar el LED repetidamente, aplicando el valor 1 durante un porcentaje determinado del período del pulso, y el valor 0 durante el resto del período.

Los gráficos anteriores muestran diversas formas del pulso, con el ancho del valor 1 modulado según el nivel PWM (% de brillo) requerido. Si dicha forma del pulso se repite a una frecuencia alta, por ejemplo, 50 Hz, el ojo humano no percibe los cambios entre estados de encendido/apagado, sino que percibe diferentes intensidades de luz según el porcentaje del tiempo en que se está emitiendo luz.

El programa a implementar, además de controlar el ancho de pulso correspondiente al nivel de brillo actual del patrón seleccionado, tendrá que actualizar dicho nivel de brillo periódica y cíclicamente, es decir, debe pasar al siguiente valor del patrón cada cierto tiempo y, cuando llegue al último valor, volver a empezar por el primero. También se requiere que el usuario pueda seleccionar el patrón de entre un conjunto predefinido de patrones, pulsando el botón 'A' de la NDS.

Se dispone de las siguientes rutinas, ya implementadas:

Rutina	*Descripción*
`inicializaciones()`	Inicializa el *hardware* (pantalla, interrupciones, *timers*, etc.)
`tareas_independientes()`	Tareas que no interfieren con la gestión del bit 5 del registro de salida, aunque pueden estar modificando algunos de los otros bits del mismo registro (ej. activar un altavoz externo); t. ejecución < 10 ms

`scanKeys()`	Captura el estado actual de los botones de la NDS
`int keysDown()`	Devuelve un patrón de bits con los botones activos
`swiWaitForVBlank()`	Espera hasta el próximo retroceso vertical
`printf(char *format,...)`	Escribe un mensaje en la pantalla inferior de la NDS

Para generar la forma del pulso correspondiente al nivel PWM requerido, se pide utilizar la RSI del *timer* 0, que se programará (por la rutina de inicializaciones) para realizar 5.000 interrupciones por segundo (5 KHz). A esta frecuencia, se podrá controlar el % del periodo del pulso (20 ms) para cambiar los estados 1 y 0 del bit 5 del registro de salida, ya que se producirá una interrupción a cada centésima de dicho periodo (cada 200 μs).

Para cambiar el valor actual de brillo de cada patrón, se pide utilizar la RSI del *timer* 1, que se programará (por la rutina de inicializaciones) para realizar 16,6667 interrupciones por segundo (periodo de 60 ms). A esta frecuencia, cada valor de brillo estará visible durante 3 ciclos de PWM (aprox.). Para simplificar el diseño, no se exige que las dos RSIs estén sincronizadas.

Para realizar la gestión de los patrones se propone usar las siguientes estructuras de datos:

```
#define NUM_PATTERNS = 5    // número de patrones de brillo
#define NUM_VALUES = 16     // número valores de brillo por patrón

unsigned char patterns[NUM_PATTERNS][NUM_VALUES] = {
     //patrón cuadrado
     {100,100,100,100,100,100,100,100,0,0,0,0,0,0,0,0},
     //patrón triangular
     {100,88,75,63,50,38,25,13,0,13,25,38,50,63,75,88},
     //patrón rampa
     {0,6,13,19,25,31,37,44,50,56,63,69,75,81,88,94},
     //patrón coseno
     {100,96,85,69,50,31,15,4,0,4,15,31,50,69,85,96},
     //patrón latido
     {0,50,80,100,70,40,21,15,37,48,25,13,8,4,2,1}
};
unsigned char curr_pattern = 0; // índice del patrón actual
unsigned char curr_value = 0;   // índice del valor actual del pat.
unsigned char brightness = 0;   // nivel de brillo actual
```

En este ejemplo se han definido cinco patrones de 16 valores cada uno (no hace falta copiar los valores de ejemplo en la solución del examen).

La variable `curr_pattern` almacenará el índice del patrón seleccionado actualmente. Cada vez que se pulsa el botón 'A', esta variable debe actualizarse y mostrar su nuevo valor por la pantalla inferior de la NDS.

La variable `curr_value` almacenará el indice del nivel de brillo actual del patrón. Cada vez que cambie (por la RSI del *timer* 1), se copiará el contenido de la posición de la matriz `patterns[][]`, correspondiente a los índices de patrón y nivel actual, dentro de la variable `brightness`, que es la que consultará la RSI del *timer* 0 para gestionar la modulación del brillo del LED por ancho de pulso.

Los símbolos `NUM_PATTERNS` y `NUM_VALUES` también estarán disponibles para el código fuente en ensamblador, definidos con las siguientes lineas:

```
NUM_PATTERNS = 5          @; número de patrones de brillo
NUM_VALUES = 16           @; número valores de brillo por patrón
```

Se pide:

Programa principal y variables globales en C, RSI del *timer* 0 y RSI del *timer* 1 en ensamblador.