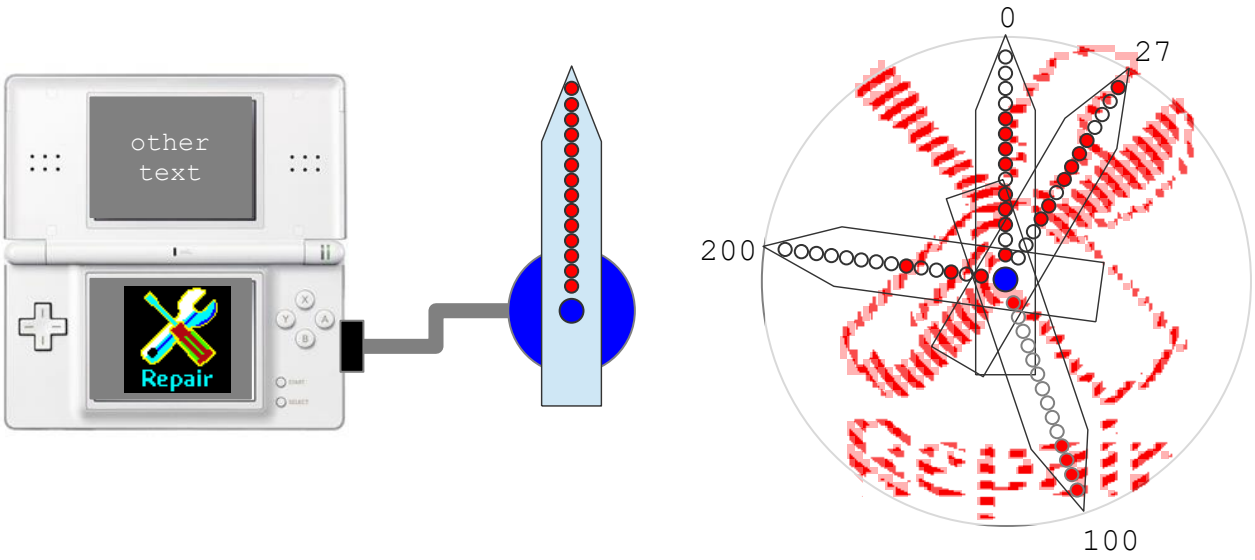
Problema 16: Propeller display (Ex. 1ª Conv. 2015-16)

Se propone controlar un display de rotación con la NDS. Este dispositivo es un circuito impreso con una línea de LEDs, montado sobre un motor que lo hace girar a una velocidad más o menos constante. El programa de control debe encender y apagar los LEDs según el ángulo de giro de la línea, de modo que se visualicen los puntos de una imagen correspondientes a dicho ángulo. Si el motor gira lo suficientemente rápido (>25 rev./s), se podrá observar la imagen "en el aire", gracias a la persistencia de la luz en la retina del ojo humano. A continuación se muestra un esquema de la estructura del dispositivo y su funcionamiento:



En el esquema anterior se ha representado un circuito impreso con 14 LEDs, y la activación de dichos LEDs en cuatro ángulos diferentes, así como una simulación del efecto del display "en el aire" para la imagen de ejemplo, la cual se muestra en la pantalla inferior de la NDS.

El sistema real a controlar dispone de 32 LEDs. Además, la circunferencia se divide en 256 fracciones, de modo que el rango del valor del ángulo será de 0 a 255. En la figura anterior se muestra la posición de la línea en los ángulos 0, 27, 100 y 200.

El dispositivo dispone de dos registros de Entrada/Salida:

- RDISP_STATUS: registro de 16 bits, del cual solo se utilizará el bit 0, que el dispositivo activará durante un pulso de 156 µs cada vez que el motor pase por el ángulo 0,
- RDISP_DATA: registro de 32 bits, que el programa debe fijar para indicar el estado de cada uno de los 32 LEDs (0: apagado, 1: encendido), donde el LED más exterior corresponde al bit de más peso.

El programa de control debe recibir imágenes "rectangulares" (64x64 píxeles) desde el

interfaz wifi de la NDS, convertir cada imagen recibida a su correspondiente imagen "circular", y transferir continuamente el estado de los LEDs al display de rotación, según el color de los *áxeles* de la imagen circular (*áxel = angular pixel*) y el ángulo de la línea de LEDs en cada instante, además de ajustar a cero el ángulo actual según el bit 0 del registro de estado del dispositivo.

Se dispone de las siguientes rutinas, ya implementadas:

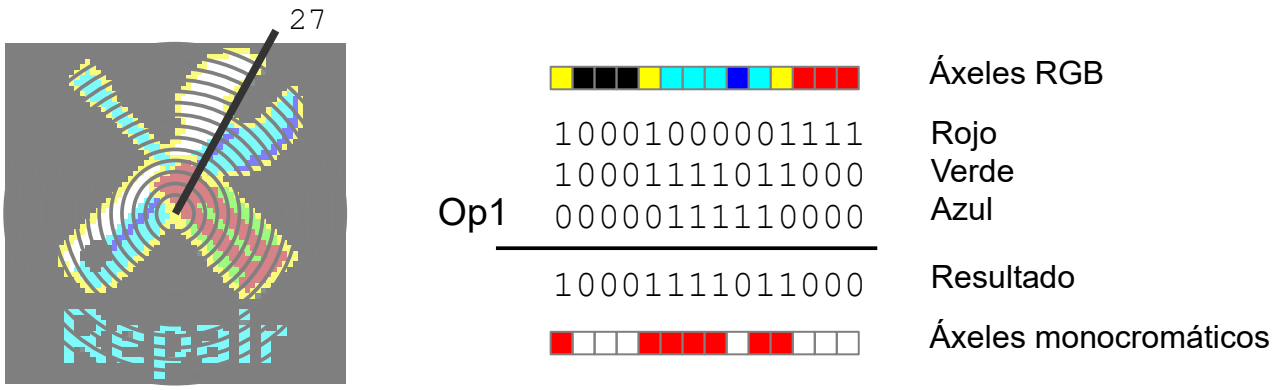
Rutina	*Descripción*
`inicializaciones()`	Inicializa el *hardware* (pantalla, wifi, interrupciones, *timers*, etc.)
`int wifiReceiveImage(unsigned char rimg[])`	Rutina de recepción de una imagen rectangular por la interfaz wifi de la NDS; si hay una nueva imagen (desde la última llamada), copiará los píxeles de dicha imagen en el vector que se pasa por referencia y devolverá 1 (t. ejecución ≈ 150 ms); si no hay nueva imagen, devolverá 0 (t. ejecución ≈ 5 μs)
`convertirImagen(unsigned char rimg[], unsigned int cimg[])`	Rutina de conversión de una imagen rectangular a la correspondiente imagen circular, determinando el color de los 32 *áxeles* de los 256 ángulos (tarda unos 30 ms)
`swiWaitForVBlank()`	Espera hasta el próximo retroceso vertical
`mostrarImagen(unsigned char rimg[])`	Muestra la imagen rectangular que se pasa por parámetro en la pantalla inferior de la NDS

Para poder representar la imagen circular actual de forma concurrente con la recepción de nuevas imágenes, se pide utilizar la RSI del *timer* 0, que se programará (por la rutina de inicializaciones) para realizar 6.400 (25*256) interrupciones por segundo.

En esta RSI hay que controlar el barrido de los 256 radios (ángulos) de la circunferencia, leyendo los colores de los 32 *áxeles* de cada radio. Hay que tener en cuenta que una imagen circular, obtenida con la rutina `convertirImagen()`, almacenará el color de cada radio como tres *words* consecutivos, que contendrán el estado (binario) de los tres canales básicos de color (rojo, verde, azul, por este orden), donde el bit de más peso de cada *word* corresponderá al LED más exterior. La información de todos los 256 radios se almacenará consecutivamente en la memoria reservada para la imagen circular, a partir del radio cero.

La RSI deberá llamar a una rutina auxiliar que se encargará de convertir los tres bits de color de cada *áxel* en un bit para cada LED, ya que el dispositivo a controlar solo dispone de LEDs monocromáticos. Además de realizar la conversión, la RSI también debe transferir los bits

resultantes al dispositivo. La siguiente figura muestra un ejemplo de conversión del radio 27, aunque aquí solo se representa el color de 14 *áxeles* (para simplificar):



La conversión de los bits de color a bits monocromáticos se debe realizar dentro de la siguiente rutina:

```
void transferir_radio(unsigned int cim[], int num_radio);
```

Esta rutina recibe la dirección de memoria inicial de la imagen circular, así como el número de radio (ángulo) que se tiene que procesar y enviar al display de rotación. Además, esta rutina debe ser capaz de realizar dos tipos de procesado, según el estado del botón 'SELECT' de la NDS (bit 2 del registro REG_KEYINPUT):

- SELECT = 1 (soltado): cada bit monocromático valdrá 1 si el bit correspondiente del canal verde está a 1 y uno de los bits de los canales rojo y azul también está a 1, pero no los dos a la vez,
- SELECT = 0 (pulsado): cada bit monocromático valdrá 1 si alguno de los bits correspondientes de los tres canales de color vale 1.

El cálculo de los 32 bits monocromáticos se debe realizar aplicando las operaciones lógicas que correspondan (`and`, `or`, `xor`, `not`).

Es importante también realizar la puesta a cero del ángulo actual cada vez que se detecte el pulso de ángulo cero emitido por el display de rotación, puesto que la velocidad de rotación del motor puede tener pequeñas variaciones momentáneas de velocidad (± 3%).

Además, hay que tener en cuenta que por la wifi podremos recibir hasta 25 imágenes rectangulares por segundo, aunque pueden ser menos, incluso puede que se envíe una única imagen para toda la sesión. En cualquier caso, por el display de rotación se deberá mostrar continuamente la última imagen circular recibida (imagen actual), mientras que, concurrentemente, puede que se reciban nuevas imágenes por la wifi.

Para el almacenamiento de imágenes se propone usar las siguientes estructuras de datos:

```
unsigned char rect_img[64*64];
unsigned int circ_img[2][256*3];
```

El espacio para 2 imágenes circulares en `circ_img[2][256*3]` permitirá aplicar la técnica del doble buffer, de modo que la recepción y conversión de las nuevas imágenes rectangulares enviadas por wifi no interfiera con la visualización de la imagen circular actual.

Por último, se puede considerar que, inicialmente, el contenido de los *buffers* de imagen están a cero, de modo que la visualización en el display de rotación de dicho contenido no activará ningún LED, aunque el motor girará a partir del primer instante que se encienda todo el sistema (NDS y dispositivo).

Se pide:

Programa principal y variables globales en C, RSI del *timer* 0 y rutina `transferir_radio()` en ensamblador.