1 ACTIVIDADES

Software para Robots

Cristian González García: gonzalezcristian@uniovi.es
Basado en el material original de Jordán Pascual Espada

Actividades obligatorias

El total de las actividades tienen un valor de 0,6 punto obligatorio y 0,6 puntos optativos dentro del bloque 1.

(Dado1.1) Dado electrónico (0,2 puntos)

Diseña un sistema con 3 LEDs y un botón. Cada vez que se pulse el botón haz que se genere un número aleatorio entre 1 y 3. El número obtenido se notificará al usuario encendiendo esa misma cantidad de LEDs.

Los LEDs se deben situar en una fila, y deberán encenderse siempre de forma ordenada: primero los encendidos y luego los apagados, es decir, no mezclados.

- Número 2: Encendido, Encendido, Apagado -> Correcto
- Número 2: Encendido, Apagado, Encendido -> Incorrecto, ejercicio mal

(Memoria1.2) Juego de memoria (0,4 puntos)

Se requerirá tenerlo implementado para el opcional 1.3

El objetivo de esta actividad es desarrollar una versión simple del juego de memoria «Simón». El sistema debe tener 2 LEDs de colores y 2 pulsadores.

Cuando iniciamos la aplicación se debe generar una secuencia aleatoria de tres parpadeos en los LEDs, por ejemplo: Rojo, Rojo, Verde. (Entre parpadeo y parpadeo de cada led se debe esperar al menos un segundo).

El usuario ahora debe reproducir esa secuencia utilizando los pulsadores. Un pulsador se corresponderá a un color y el otro a otro. Es decir, para «acertar» la secuencia Rojo, Rojo, Verde el usuario debe pulsar el pulsador asociado al rojo dos veces y luego el pulsador asociado al verde. **Cuando se pulse un pulsador, el LED asociado a ese pulsador se debe iluminar.**

Si el usuario realiza la acción correctamente, el juego sube de nivel. Entonces, el juego debe añadir un nuevo elemento aleatorio a la secuencia de parpadeos en los leds de colores. Cada vez que el usuario acierta la combinación se vuelve a repetir la secuencia anterior añadiendo un parpadeo adicional (Para ir aumentando la dificultad). **¡Ojo! Se añade un nuevo color a la secuencia previa, no hay que generar una nueva secuencia entera.**

En el momento en el que el usuario realiza alguna pulsación incorrecta, el juego finaliza y se inicia desde el principio. Se debe volver a generar una secuencia aleatoria inicial.

Para generar un aleatorio hay una guía sencilla en la página de Arduino: https://www.arduino.cc/reference/en/language/functions/random-numbers/random/. Básicamente, consiste en leer un pin analógico vacío y usar la función random, que puede recibir un parámetro (0 a máximo+1) o dos (mínimo y máximo+1).

Para esta actividad será necesario utilizar alguna estructura de datos y generación de números aleatorios. Se debe dividir la funcionalidad del sistema en funciones.

- Si toda la lógica se encuentra en el método loop, se restará la mitad de puntos, aunque el ejercicio esté bien

Los pulsadores necesitan de una resistencia, así que hay que añadirla, pues, en el caso de que quedaran conectadas directamente las corrientes positivas y negativa se podría quemar la placa.

Atención: cuando se grabe el video, hay que mostrar una partida en la que funcione bien y también una partida en la que se falle, para ver cómo se reinicia el juego. Sino, se descontarán puntos.

Actividades Opcionales

(Zumbador1.3) Ampliar Simón con el Zumbador y LED RGB (0,6 puntos)

Requiere tener implementado el ejercicio obligatorio 1.2

Hay que ampliar el ejercicio anterior y agregar un sonido característico a cada uno de los LEDs utilizando el zumbador y sustituir los LEDs por un LED RGB.

El sonido debe reproducirse cuando los leds parpadeen o cuando el usuario pulse el botón asociado a cada led.

Zumbador: capaz de producir varios sonidos. Se suele utilizar en alarmas y sonidos característicos del sistema.

Tiene una patilla positiva **VCC**, una negativa **GND** y una patilla **I/O** que se conecta a un pin PWM, como el ~11, capaz de emular escrituras analógicas (de esta forma especificaremos los tonos). Dependiendo de la señal que le enviemos por el pin emitirá un tono u otro.

```
int pinbuzzer = 11;

int song[] = {261, 349, 392, 440, 392, 330, -10, 261, 349, 392, 440,
392, -10, -10, 261, 349, 392, 440, 392, 330, -10, 330, 349, 330, 261,
261};

void setup()
{
  Serial.begin(9600);
  pinMode(pinbuzzer, OUTPUT);
}
```

```
void loop()
{
  for (int i = 0; i < sizeof(song)/sizeof(int); i++)
  {
    analogWrite(pinbuzzer, song[i]);
    delay(500);
  }
}
```

El api de Arduino provee de dos métodos específicos `tone & notone` que podemos utilizar para manejar el zumbador:

https://www.arduino.cc/en/pmwiki.php?n=Reference/Tone

```
void loop()
{
  for (int i = 0; i < sizeof(song)/sizeof(song[0]); i++)
  {
    tone(pinbuzzer, song[i]);
    delay(500);
  }

   noTone(pinbuzzer);
   delay(5000);
}
```

En lugar de utilizar 2 LEDs normales hay que utilizar un LED RGB. El LED RGB no requiere utilizar una resistencia externa, pues la lleva integrada. Dispone de una patilla negativa (que se debe conectar al GND) y de tres patillas que deben ser conectadas a pines digitales, cada una de las patillas se corresponde con un color R, G, B. El color se iluminará si enviamos voltaje como salida por el pin correspondiente. Podemos utilizar solo una patilla de color R, G, B o varias. Al enviar voltaje de forma simultanea por varias patillas podemos realizar combinaciones de colores.

Luz RGB	Placa
R	Pin digital
G	Pin digital
B	Pin digital
-	GND

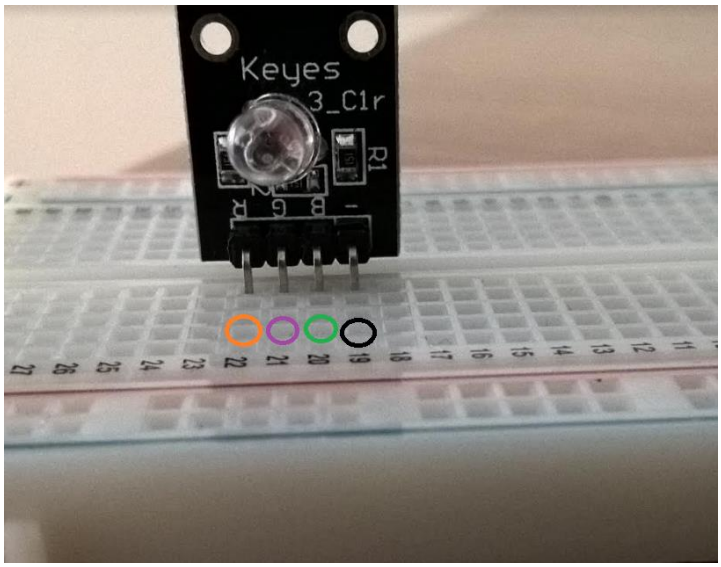

Ilustración 1 Conexión. VCC a 5V (Rojo) GND a GND (Negro) Out a un pin digital (Verde)

Podemos utilizar la función digitalWrite(<pin>, LOW/HIGH) si queremos **alumbrar al máximo** de la capacidad del LED o apagarlo.

Si quisiéramos hacer **combinaciones** más complejas **de colores**, por ejemplo, un poco de rojo y mucho verde, podemos utilizar el analogWrite(<pin>, [0-255]), pero en este caso tenemos que estar seguros de **usar pines PWM (los señalados con ~).**

Si por ejemplo solo queremos utilizar el LED como rojo y verde, basta con conectar los pines digitales a la patilla R y G, podemos dejar la B sin conexión.