MÓDULO 4: PROJECTO 8 "Reloj de Arena"

CURSO PROGRAMACIÓN DE PLACAS ROBÓTICAS

Introducción

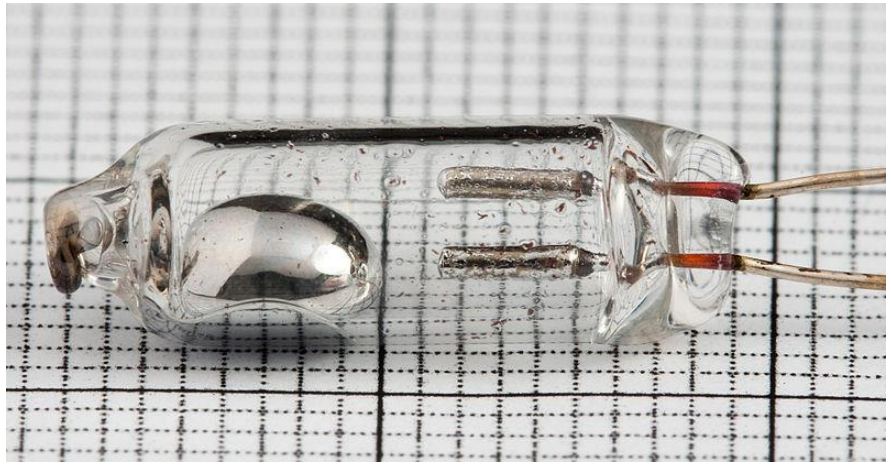
- En este proyecto, se va a construir un reloj que cambia un LED en cada diez minutos. El reloj utiliza el Arduino Uno de temporizador. Reinicie el reloj de la misma forma que lo haría para reiniciar un reloj de arena: variando su inclinación
- aprenderemos a utilizar la función **millis()** la cual, nos permite contabilizar el tiempo que transcurre.
- Hasta ahora, cuando queríamos que algo ocurriera al cabo de un intervalo específico de tiempo, recurríamos a la función **delay()**. La función **millis()** contabiliza en milisegundos el tiempo que nuestro Arduino está en marcha.

Introducción

- Hemos creado variables de tipo **int**, es un número de 16 bits, que comprende los valores en un rango que va desde el -32.768 hasta el 32.767.
- Por contra, el tipo de dato **long** contiene números de 32 bits, comprendidos en un rango que va desde el -2.147.483.648 hasta el 2.147.483.647.
- la variable que usemos para almacenar el tiempo de la función **millis()** se denomina **unsigned long** (long sin signo). Cuando un tipo de variable es llamada *unsigned*, se la considera solamente positiva. Un **long** sin signo puede contar hasta 4.294.967.295, esto es suficiente para que la función **millis()** almacene el tiempo transcurrido durante 50 días.
- Cuando le demos la vuelta a nuestro reloj de arena digital, un sensor de inclinación (tilt switch en inglés) cambiará su estado, haciendo que los leds que indican el tiempo transcurrido se apaguen para volver a contar un nuevo periodo de tiempo.

Introducción

- Un sensor de inclinación funciona igual que un interruptor, ya que tan solo es un componente con dos estados, es por ello que lo usaremos como elemento de entrada. Existen dos modelos de este componente, los que están basados en una pequeña bola metálica y los que funcionan con mercurio.



Montando el circuito

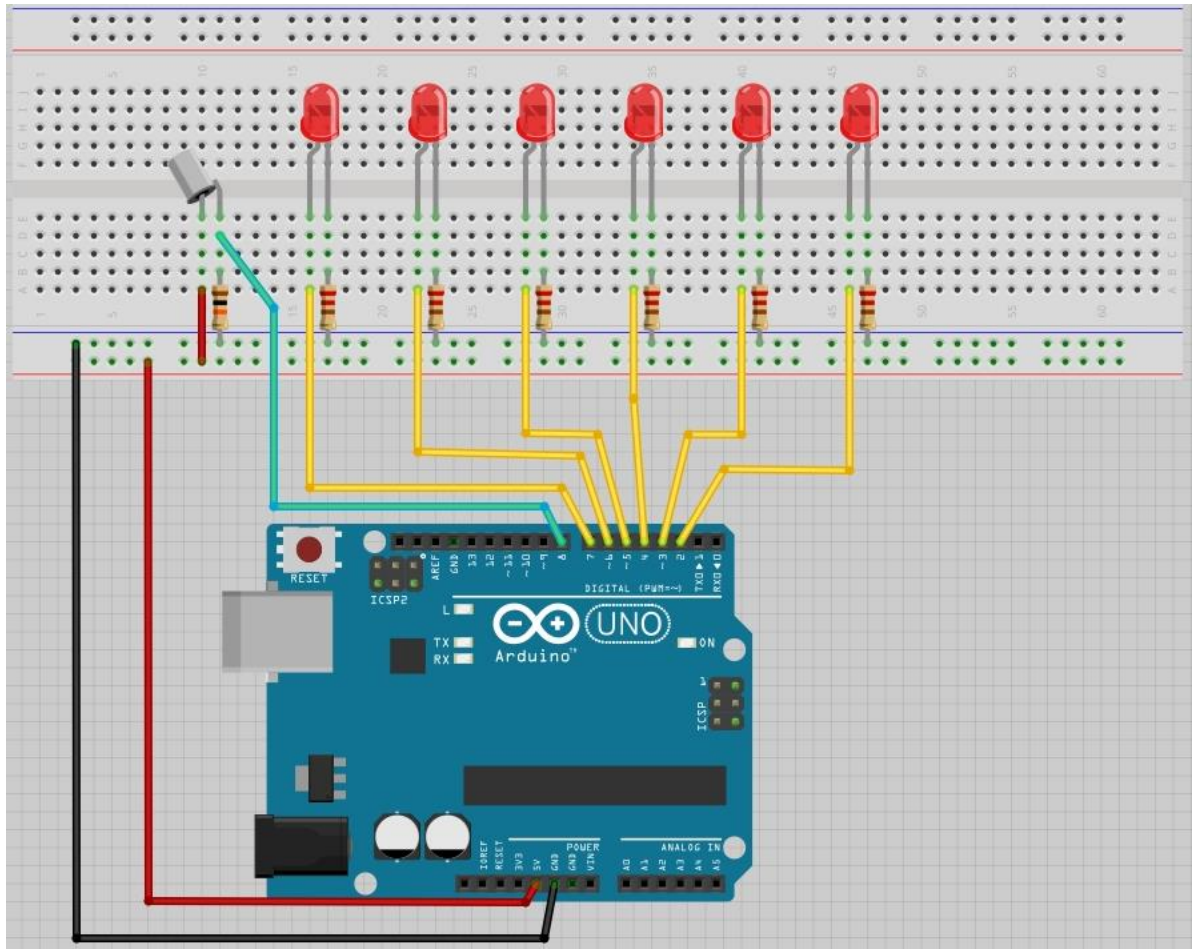
1. Tal y como hemos hecho en los proyectos anteriores, conectamos la alimentación y la tierra en nuestra protoboard.
2. Unimos el ánodo de cada uno de los seis leds a los pines del 2 al 7 del Arduino. El cátodo de los leds lo conectaremos a tierra mediante las resistencias de 220Ω.
3. Ahora cogemos y unimos una de las patillas del sensor a la alimentación. La otra irá a tierra a través de la resistencia de 10KΩ.
4. Finalmente colocaremos un cable que vaya desde el pin número 8 a la unión entre el sensor y la resistencia de 10KΩ.

Montando el circuito

5. Seguidamente, cogemos los tres trozos de papel o plástico de colores y colocamos uno sobre cada fotorresistencia. Ponemos el de color rojo sobre la fotorresistencia conectada al pin A0, el verde sobre la que está conectada al pin A1 y el azul, sobre la fotorresistencia conectada al pin A2. Cada uno de estos plásticos solo deja pasar la luz de una determinada longitud de onda. El rojo solo deja pasar el rojo, el de color verde la luz verde y el azul solo la luz azul. De esta manera, cada fotorresistencia solo detectará un determinado tipo de luz.
6. El siguiente componente que vamos a utilizar es el led RGB. Este tipo de led tiene cuatro patillas: una para el color rojo, una para el color verde, una para el color azul y otra que sirve de tierra para las otras tres (el cátodo). Cada una de las patillas estará conectada a una resistencia de 220Ω (a excepción del cátodo).

Montando el circuito

Componentes	
Sensor de inclinación	1
LED Rojos	6
Resistencias 220 Ω	6
Resistencias 10 K Ω	1



Comentario del código

- Vamos a necesitar unas cuantas variables globales en nuestro programa para que todo funcione. Así que para empezar, crearemos una constante llamada switchPin.

- A continuación, creamos una variable de tipo **unsigned long**, la cual, almacenará el tiempo transcurrido desde la última vez que cambió el estado de un led.

- Después, crearemos una variable para el estado actual del sensor y otra para su estado anterior. Seguidamente, creamos una variable llamada led. Ésta nos servirá para contabilizar que led será el próximo en encenderse. El primero de ellos será el conectado al pin 2.

Comentario del código

- La última variable en ser creada será aquella que almacene el valor del intervalo de tiempo entre el encendido de un led y el siguiente. Dicha variable será de tipo **long**. El tiempo transcurrido entre el encendido de un led y el siguiente será de 10 minutos, o lo que es lo mismo, 600.000 milisegundos.

- En la función **setup()**. En esta función debemos declarar los pins del 2 al 7 (los correspondientes a los leds) como salidas. Para realizar esto usaremos un bucle **for()** y declaramos switchPin como entrada.

- Cuando la función **loop()** empieza, leeremos mediante la función **millis()** la cantidad de tiempo que ha lleva Arduino encendido y almacenaremos dicho valor en una variable local llamada **currentTime**.

Comentario del código

- utilizaremos la sentencia **if()** para comprobar si ha transcurrido tiempo suficiente para activar un led. Restaremos el tiempo actual (**currentTime**) del tiempo anterior (**previousTime**) y comprobamos si el resultado de dicha resta es mayor que el intervalo establecido anteriormente. Si han transcurrido 600.000 milisegundos (10 minutos), almacenaremos el valor de **currentTime** en la variable **previousTime**.

- Una vez que hayamos establecido el valor de **previousTime**, encenderemos un led e incrementaremos el valor de la variable led. La próxima vez que haya transcurrido el intervalo establecido, se encenderá el led siguiente.

- añadiremos una sentencia **if()** más en el programa con el fin de determinar si el led conectado al pin 7 está activado. Si lo está, es que ha transcurrido una hora. En este momento, si dicha sentencia es cierta, no haremos nada.

Comentario del código

- Hemos comprobado cuanto tiempo ha pasado, miraremos si el sensor ha cambiado su estado. Para ello leeremos el estado de dicho sensor y almacenaremos su valor en la variable **switchState**.

- Mediante una sentencia **if()** comprobaremos si el estado actual del sensor difiere del estado anterior. El símbolo **!=** sirve para evaluar si el valor de **switchState** no es igual que el valor de **prevSwitchState**.

- Si tienen valores diferentes, apagaremos los leds, le daremos a la variable led el valor del primer pin y resetearemos el temporizador de los leds dándole a la variable **previousTime** el valor de **currentTime**.

- Para acabar, guardaremos el estado del sensor en la variable **prevSwitchState**, comprobando dicho estado con el que obtendremos en la variable **switchState** de la siguiente iteración de la función **loop()**.