08

INGREDIENTES

RELOJ DE ARENA DIGITAL

EN ESTE PROYECTO, SE VA A MONTAR UN RELOJ DIGITAL DE ARENA QUE ENCIENDE UN DIODO LED CADA 10 SEGUNDOS. DE ESTA MANERA AL USAR UN TEMPORIZADOR CONSTRUIDO EN ARDUINO SE PODRÁ SABER CUANDO TIEMPO SE TRABAJA EN UN PROYECTO

Descubra: datos de tipo largo, creación de un temporizador

Tiempo: **30 MINUTOS**

Nivel: **medio**

Proyectos en los que se basa: **1,2,3,4**

Hasta ahora, cuando se ha querido que suceda algo al pasar un intervalo de tiempo específico con Arduino se ha usado la instrucción delay(), la cual es útil pero un tanto limitada. Cuando se ejecuta delay() Arduino se paraliza hasta que se termine el tiempo especificado dentro de esta instrucción. Esto significa que no es posible trabajar con las señales de entrada y salida mientras está paralizado. Delay tampoco es muy útil para llevar un control del tiempo transcurrido. Resulta un tanto engorroso hacer algo cada 10 segundos utilizando para ello delay junto con este tiempo de retraso.

La función **millis()** ayuda a resolver estos problemas. Realiza un seguimiento del tiempo que Arduino ha estado funcionando en mili segundos. Se ha utiliza esta función en el proyecto número 6 (página 70 Theremin controlado por luz) para establecer un tiempo de 5 segundos que permita calibrar los niveles de iluminación del sensor.

Hasta ahora se han declarado variables del tipo **int**. Una variable **int** (entero) es un número de 16 bit, esto significa que puede guardar números decimales entre -32768 y 32767. Estos valores numéricos pueden parecer muy grandes, pero si Arduino cuenta 1000 veces por segundo usando la función **mills()**, llegará a superar el rango de valores entre -32768 y 32767 (65536 números) en 65.5 segundos. Los datos de tipo largo o **long** pueden guardar números de 32 bits (entre -2147483648 y 2147483647). Ya que no es posible contar el tiempo hacia atrás usando números negativos, la variable para guardar el tiempo que hay que usar en la función **mills()** se llama **unsigned long**. Cuando un tipo de datos es llamado *unsigned* (sin signo), solo trabaja con números positivos. Esto posibilita realizar cuentas mucho mayores. Una variable del tipo unsigned long puede contar más de 4294 millones de números. Son bastantes números para ser contados con **mills()** ya que le llevaría hacerlo más de 50 días. Al poder comparar el tiempo que cuenta **millis()** con un tiempo en concreto, es posible determinar la cantidad de tiempo que ha transcurrido con respecto a ese tiempo.

Cuando se gira el reloj de arena sobre si mismo, el interruptor de inclinación cambia de estado, y comenzará un nuevo ciclo de encendido de los diodos LED.

El interruptor de inclinación trabaja igual que un interruptor normal, pero en esta aplicación se comporta como un sensor de encendido/apagado. Aquí se usará como una entrada digital, ya que proporciona dos niveles lógicos diferentes, “0” y “1”. Los interruptores de inclinación son únicos a la hora de detectar la orientación o inclinación de un objeto. En su interior disponen de una pequeña cavidad con una bola de metal. Cuando el interruptor se gira la bola de metal se mueve en su interior rodando hasta uno de los extremos de la cavidad, haciendo que dos terminales se conecten entre sí de forma que se cierra el circuito que está conectado a la placa de pruebas. En ese momento el reloj de arena digital comenzará a contar un tiempo de 60 segundos encendiendo un LED, de los 6 de que dispone, cada 10 segundos.

MONTANDO EL CIRCUITO

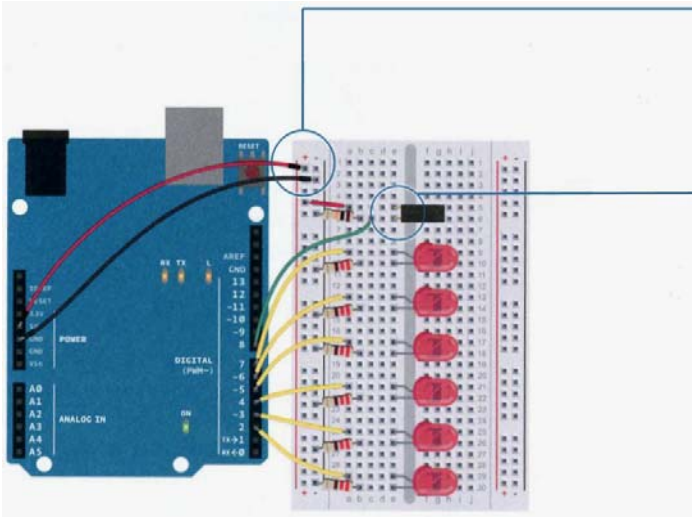

Figura 1

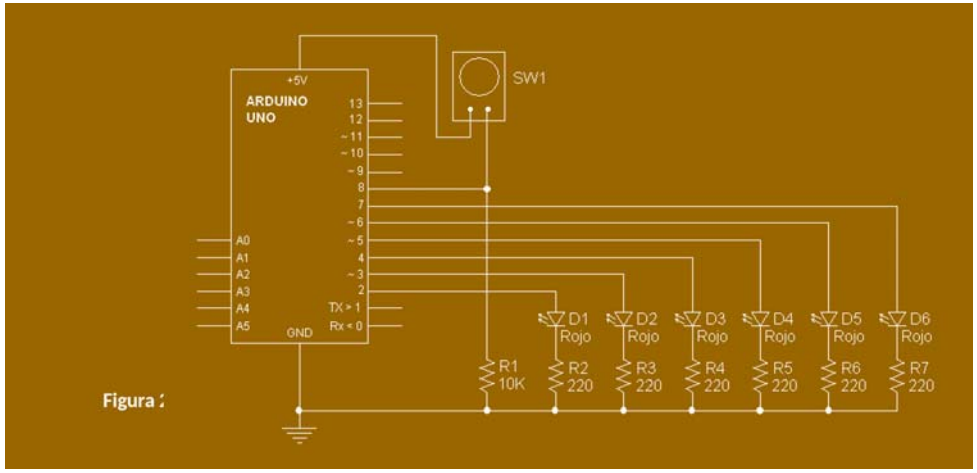

Figura 2

1. Conectar el cable de alimentación positivo y negativo (cables rojo y negro) a la placa de pruebas
2. Conectar el ánodo de cada uno de los seis LEDs a los pins digitales de 2 a 7. Conectar el otro pin de los LED a masa y a través de una resistencia de 220 ohmios .
3. Conectar un pin del interruptor de inclinación al positivo de alimentación +5V de Arduino. Conectar el otro pin a masa usando una resistencia de 10 kilo ohmios. Conectar el punto de unión de esta resistencia con el terminal del interruptor al pin digital número 8.

No es necesario mantener este proyecto conectado al ordenador una vez ha sido cargado el programa en la placa de Arduino. Se podría intentar construir una pequeña caja de cartón para colocar el montaje en su interior junto con la batería. También se puede dibujar en una de las caras de esta caja unos indicadores numéricos para los diodos led, de manera que vaya indicando el tiempo transcurrido después de haber girado la caja.

Los interruptores de inclinación son geniales, componentes de bajo coste para determinar la inclinación de algún objeto. ***Los acelerómetros*** son otro tipo de sensores que proporcionan mucha más información. También son significativamente más caros. Para saber si un objeto se gira cuando se mueve la utilización de un sensor de inclinación es la opción más adecuada.

EL CÓDIGO

Declarar variable de tipo constante

Es necesario definir varias variables del tipo global para conseguir que el programa funcione. Para comenzar, se crea una variable del tipo constante llamada **PinInterruptor**. Este será el nombre del pin del sensor de inclinación que detecta el giro del circuito.

```
1  const int PinInterruptor = 8;
```

Crear una variable para guardar el tiempo

Crear una variable del tipo **unsigned long**. Se utiliza para almacenar el tiempo en que cambia de estado (apagado-encendido) cualquiera de los seis LEDs.

```
2  unsigned long TiempoPrevio = 0;
```

Definir variables para las salidas y la entrada

Crear una variable que almacene el estado del sensor de inclinación y otra que guarde el estado anterior del mismo. Se usarán estas dos variables para comparar la posición (cerrado o abierto) del interruptor del sensor de un ciclo completo al siguiente.

Crear una variable llamada Led. Se usará para contar qué LED será el próximo en encenderse. Comienza por la salida digital de pin 2 (LED D1).

```
3  int EstadodelInterruptor = 0;
4  int EstadoPreviodelInterruptor = 0;

5  int Led = 2;
```

Crear una variable para establecer el intervalo entre los encendidos

La última variable que se va a crear guarda el intervalo de tiempo de encendido entre cada LED. Será una variable del tipo **long**. Para establecer un tiempo de 10 segundos (el tiempo que pasa entre el encendido de un LED y el siguiente) el valor de esta variable valdrá 10.000 mili segundos. Para aumentar o disminuir este tiempo simplemente hay que cambiar este valor, por ejemplo, para un tiempo de 1 minuto poner 60000.

```
6  long TiempoIntervalocadaLed = 10000;
```

Establecer como trabajan los pins digitales

Dentro de la función **setup()** se configuran los pins que van a trabajar como salidas para encender los diodos LED, en este caso los pins 2 al 7. Se utiliza la instrucción **for()** para crear un bucle que define en solo tres líneas de código estos seis pins como salidas **OUTPUT**. También aquí dentro se define el pin (8 dentro de PinInterruptor) que se conecta al sensor de inclinación como una entrada **INPUT**.

```
7  void setup() {
8    for(int x = 2;x<8;x++){
9     pinMode(x, OUTPUT);
10   }

11   pinMode(PinInterruptor, INPUT);
12 }
```

Averiguar el tiempo que el programa lleva funcionando

Cuando se ejecuta la parte principal de este programa dentro del **loop()**, es necesario saber el tiempo que ha transcurrido desde que Arduino está encendido usando la instrucción **mills()**, y a continuación guardar este dato dentro de una variable local (porque se localiza dentro del bucle) llamada **TiempoActual**.

```
13 void loop() {
14   unsigned long TiempoActual = millis();
```

Evaluar la cantidad de tiempo transcurrido desde el bucle anterior

Utilizando una instrucción **if()**, se verificará si se ha pasado el intervalo establecido para encender un diodo LED. Restando el valor de la variable **TiempoActual** del valor de la variable **TiempoPrevio** se comprueba si este resultado es mayor que el valor de la variable **TiempoIntervalocadaLed**. Si han pasado 10.000 mili segundos (10 segundos), la variable **TiempoPrevio** toma el valor de la variable **TiempoActual**.

```
15   if(TiempoActual - TiempoPrevio > TiempoIntervalocadaLed){
16    TiempoPrevio = TiempoActual;
```

Encender un LED y prepararse para encender el siguiente

La variable **TiempoPrevio** indica la última vez que un LED fue encendido. Una vez se ha configurado **TiempoPrevio,** se enciende un LED y se incrementa el valor de la variable Led. Una vez que transcurra el tiempo indicado en la variable **TiempoIntervalocadaLed**, se encenderá el siguiente LED.

Comprobar si todos los LEDs están encedidos

Se añade otra instrucción **if()** en el programa para comprobar si el LED conectado al pin número 7 está encendido. En caso de que se cumpla esta condición no se hace nada. Se podrá decidir más adelante que sucede cuando ha transcurrido este tiempo de 1 minuto.

Leer el estado del sensor

Ahora que se ha comprobado el tiempo, también se comprueba si el sensor de inclinación ha cambiado de estado. Se lee el valor del interruptor dentro de la variable **EstadodelInterruptor**.

Poner a cero los valores de las variables si es necesario

Con otra instrucción **if()** se comprueba si el interruptor del sensor de inclinación ha cambiado de posición con respecto a otro estado anterior. La operación != comprueba si la variable **EstadodelInterruptor** no es igual a la variable **EstadoPreviodelInterruptor**. Si son diferentes, se apagan todos los diodos LEDs, y la variable LED toma el valor del primer pin de salida de Arduino (2), además de poner a cero el temporizador de los LEDs al igualar la variable **TiempoPrevio** a **TiempoActual**.

Establecer el estado actual al estado anterior

Al final del bucle **loop()**, se guarda el estado actual del interruptor del sensor de inclinación dentro de la variable **EstadoPreviodelInterruptor**, de esta manera se puede comparar con el valor que se guarde en **EstadodelInterruptor** en la siguiente ejecución del programa o bucle **loop()**.

COMO SE UTILIZA

Una vez que se ha programado la tarjeta Arduino, comprobar el tiempo de un minuto usando un reloj. Después de que hayan pasado 10 segundos se debe de encender el primer diodo LED. Cada 10 segundos debe de encenderse un LED. Al final de los 60 segundos todos los diodos LEDs deberán de estar encendidos. Si se mueve el circuito en cualquier momento y se consigue que el sensor de inclinación cambie de estado, todos los diodos LEDs se apagarán y comenzará uno nuevo ciclo de encendido de los LEDs cada 10 segundos.


```
17  digitalWrite(Led, HIGH);
18    Led++;
```

```
19    if(Led == 7){
20    }
21   }
```

```
22  EstadodelInterruptor = digitalRead(PinInterruptor);
```

```
23   if(EstadodelInterruptor != EstadoPreviodelInterruptor){
24    for(int x = 2;x<8;x++){
25     digitalWrite(x, LOW);
26    }

27    Led = 2;
28    TiempoPrevio = TiempoActual;
29   }
```

```
30   EstadoPreviodelInterruptor = EstadodelInterruptor;
31  }
```

Cuando el reloj alcanza los 60 segundos se encienden los seis diodos LEDs y permanecen en este estado. ¿Puede pensar como conseguir que el circuito llame la atención cuando se alcance este tiempo? Son buenos indicadores para hacer esto generar un sonido o hacer que los diodos LEDs se pongan en intermitencia. La variable Led se puede comprobar para ver si todos los LEDs están encendidos, ese es un buen lugar para colocar varias instrucciones en el programa para que se llame la atención al finalizar el tiempo. A menos que el reloj se rellene con arena, las luces podrían ir encendiéndose hacia arriba o hacia abajo dependiendo de la orientación del sensor de inclinación. Se puede pensar como usar la variable que indica el estado del interruptor del sensor para indicar la dirección de encendido de los LEDs.

Para medir una cantidad de tiempo entre eventos, se utiliza la función millis(). Los números que genera esta función son más grandes que los que se puede guardar usando el tipo de variable int, por eso es necesario usar el tipo de variable unsigned long para guardar los valores que genera mills()