06

INGREDIENTES

THEREMIN CONTROLADO POR LUZ

¡HA LLEGADO EL MOMENTO DE HACER ALGO DE RUIDO! USANDO UNA FOTO RESISTENCIA Y UN ZUMBADOR, VAMOS A MONTAR UN THEREMIN CONTROLADO POR LA LUZ

Descubra: generar sonido con la función tone(), calibrar sensores analógicos

Tiempo: **45 MINUTOS**

Nivel: **bajo-alto**

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

Un Theremin es un instrumento que genera sonidos mediante el movimiento de las manos de un músico alrededor de este instrumento. Probablemente habrá oído alguno de ellos en películas de terror. El Theremin detecta donde se produce el movimiento de la mano del artista en relación a dos antenas al poder leer la variación de la carga capacitiva de estas antenas. Las antenas se conectan a un circuito analógico que crea los sonidos. Una de las antenas controla la frecuencia del sonido y la otra controla el volumen. Aunque Arduino no puede producir de una forma exacta los misteriosos sonidos de este instrumento, si que es posible emularlos usando la función **tone()**. La figura 1 muestra los diferentes tonos que se producen al usar **analogwrite()** y **tone()**. Esto posibilita que un transductor como pueda ser un altavoz o un zumbador se mueva hacia delante y hacia atrás a diferentes velocidades.

Observa como este tono está en estado bajo la mayoría del tiempo, pero la frecuencia es la misma que en el siguiente tono, PWM200.

Observa como en este tono la tensión está en estado alto (+5V) durante la mayoría del tiempo, pero la frecuencia sigue siendo la misma que para el tono anterior, PWM50.

La relación cíclica de este tono es del 50% (la mitad del tiempo esta en estado alto y la otra mitad en estado bajo) y la frecuencia es la misma.

En este tono la relación cíclica es igual que para el tono de 440 pero la frecuencia vale el doble

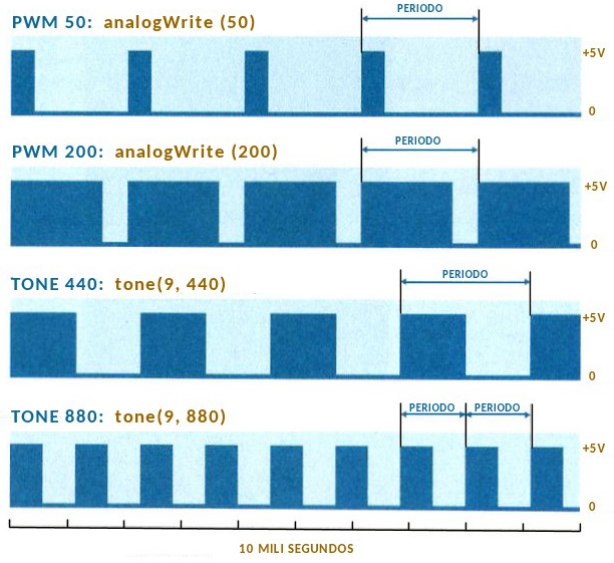


Figura 1

En lugar de usar sensores capacitivos con Arduino, se utilizará una foto resistencia LDR para detectar la cantidad de luz. Al mover las manos sobre el sensor, se variará la cantidad de luz que llega a la superficie de la LDR, como se hizo en el proyecto número 4 (lámpara de mezcla de colores). El cambio de tensión en el pin de una entrada analógica de Arduino (A0) determinará la frecuencia de la nota que se va a escuchar.
La foto resistencia LDR se conectará a Arduino usando un divisor de tensión tal y como se hizo en el proyecto número 4. Probablemente se habrá dado cuenta de que en el proyecto 4 la lectura que se obtenía al usar **analogRead()** no cubría todo el rango de 0 a 1024. La resistencia fija conectada a masa limita la parte baja del rango y el brillo de la luz sobre la LDR limita la parte superior de este rango. En lugar de configurar el circuito para limitar el rango, se calibran las lecturas del sensor entre los valores alto y bajo, convirtiendo estos valores a frecuencias sonoras usando la función **map()** y de esta forma conseguir el mayor rango posible del Theremin.

Un zumbador piezoeléctrico es un componente electrónico que vibra cuando recibe electricidad. Cuando se mueve desplaza el aire a su alrededor, creando ondas de sonido.

MONTANDO EL CIRCUITO

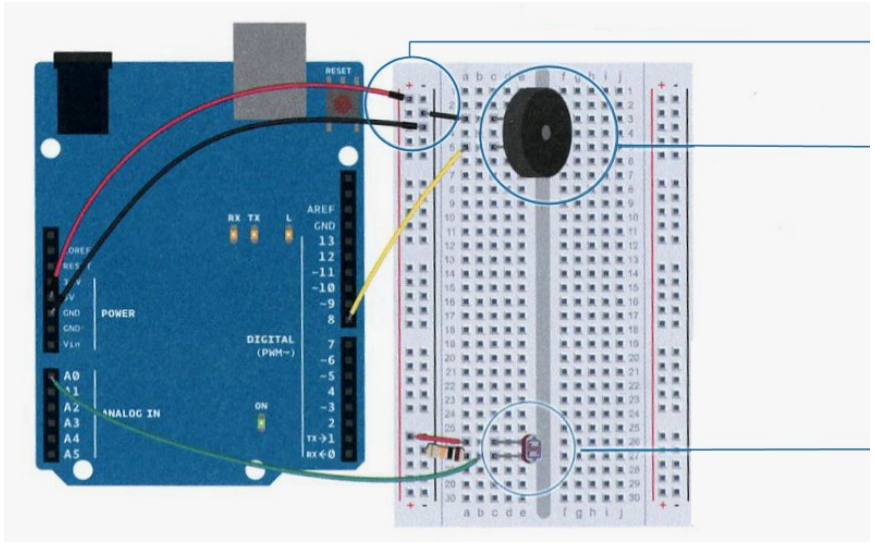


Figura 2

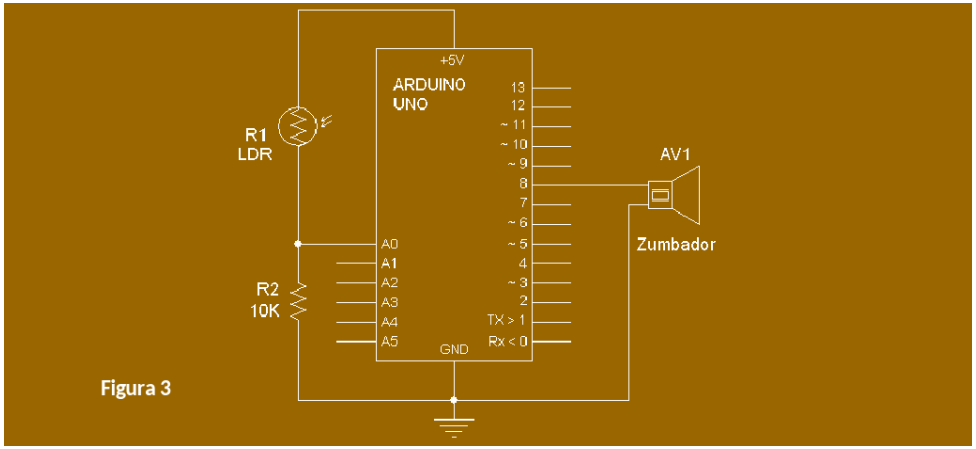
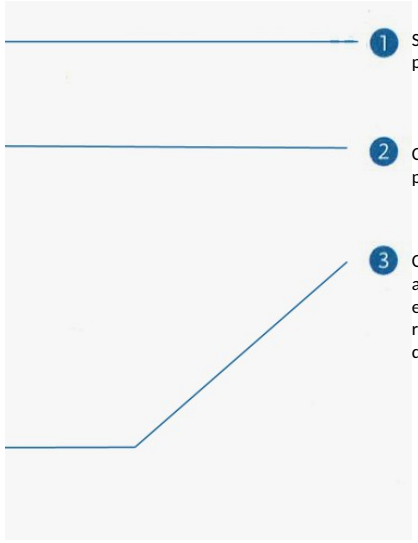


Figura 3

Tradicionalmente el Theremin puede controlar la frecuencia y el volumen del sonido. En este proyecto solamente se podrá controlar la frecuencia. Ya que no se puede controlar el volumen usando Arduino, es posible cambiar manualmente el nivel de tensión que le llega al zumbador. ¿Qué sucede si se conecta un potenciómetro en serie con el zumbador y conectado al pin 8? ¿Y si se colocase otra foto resistencia?



1. Sobre la placa de pruebas, conectar los cables de alimentación y masa desde la placa Arduino.

2. Coger el zumbador y conectar uno de sus pins a masa, el otro pin conectarlo al pin digital 8 de Arduino.

3. Colocar la foto resistencia LDR en la placa de pruebas, conectar uno de sus pins a la columna de alimentación de +5V a través de un cable rojo. Conectar el otro extremo de la LDR al pin analógico A0 de Arduino y a masa a través de una resistencia de 10 Kilo ohmios. Este circuito es el mismo que el divisor de tensión del proyecto número 4.

EL CÓDIGO

Crear variables para calibrar el sensor

Crear una variable (ValordelSensor) que guarde la lectura del valor de la foto resistencia LDR usando la instrucción **analogRead()**. A continuación, crear dos variables, una para almacenar el valor máximo y otra para guardar el valor mínimo (ValorMaximodelSensor y ValoMinimodelSensor). Se establece un valor inicial de 1023 para el valor mínimo y un valor de 0 para el valor máximo. Cuando se ejecute el programa por primera vez, se comparan estos números con los valores obtenidos en la LDR, para encontrar de esta forma los valores máximo y mínimo del rango y poder así calibrar el sensor.

```
1  int ValordelSensor;
2  int ValorMinimoSensor = 1023;
3  int ValorMaximoSensor = 0;
```

Crear una constante para el indicador de calibración

Crear una constante llamada **PinLed**. Se utilizará para indicar que el sensor está siendo calibrado mientras el diodo LED de la placa Arduino permanezca encendido. Este diodo LED integrado se conecta al pin digital 13 de Arduino.

```
4  const int PinLed = 13;
```

Establecer el número del pin digital y ponerlo en estado alto

Dentro de la función **setup(),** y usando la instrucción **pinMode(),** definir el pin del led (PinLed) como una salida (**OUTPUT**) y encender este diodo LED.

```
5  void setup() {

6   pinMode(PinLed,13);
7   digitalWrite(PinLed,HIGH);
```

Usar la instrucción While() para realizar la calibración

Los siguientes pasos calibrarán los valores máximo y mínimo del sensor. Se utiliza la instrucción **while()** para ejecutar otras instrucciones varias veces durante 5 segundos. **while()** ejecuta las instrucciones que contiene durante este tiempo hasta que ciertas condiciones se cumplen. Junto con la instrucción **while()** se utiliza otra instrucción, **mills()**, para saber el tiempo actual. Esta instrucción informa del tiempo que Arduino lleva funcionando desde que se ha sido encendido o desde que se hizo un reset.

```
8   while(millis() < 5000) {
```

Comparar los valores del sensor para realizar la calibración

Dentro de la función **loop()**, se realiza la lectura del valor del sensor; si este valor es mayor que el valor que almacena la variable ValorMaximoSensor (inicialmente 0), se actualiza el valor de esta variable con el valor leído del sensor. Al igual ocurrirá con la variable que almacena el valor minimo (ValorMinimoSensor, inicialmente con un valor de 1024), si el valor leído del sensor es menor esta variable almacenará este valor.

```
9       ValordelSensor = analogRead(A0);
10      if(ValordelSensor > ValorMaximoSensor) {
11       ValorMaximoSensor = ValordelSensor;
12      }
13      if(ValordelSensor < ValorMinimoSensor) {
14       ValorMinimoSensor = ValordelSensor;
15      }
16     }
```

Indicar que la calibración ha finalizado

Cuando los 5 segundos han pasado, la ejecución en bucle de la instrucción while() llegará a su fin. El diodo LED conectado al pin 13 de Arduino dejará de alumbrar, indicando que la calibración ha finalizado. Se utilizarán los valores máximo y mínimo que se acaban de guardar para establecer el rango de frecuencias en la parte principal del programa. Estas frecuencias son las que se van a oír a través del zumbador.

```
17   digitalWrite(PinLed,LOW);
18  }
```


while()
arduino.cc/while

Leer y guardar el valor del sensor

Dentro de la función **loop()**, se procede a la lectura del sensor conectado al terminal A0 de Arduino para almacenarlo dentro de la variable **ValordelSensor**.

Convertir el valor del sensor a frecuencia

Crear una variable llamada **tono**. El valor que se almacena dentro de esta variable **tono** se obtiene de cambiar de rango el valor leído del sensor y que está guardado dentro de la variable **ValordelSensor**. Usar las variables **ValorMaximoSensor** y **ValorMinimoSensor** para establecer los límites del rango que se va a cambiar de escala usando la función **map()**. Los valores iniciales de salida de la nueva escala están comprendidos entre 50 y 4000. Estos números representan el rango de frecuencias que Arduino va a generar a través del zumbador, es decir, producirá frecuencias comprendidas entre 50 y 4000 ciclos por segundo.

Reproducir la frecuencia (tono)

A continuación, se ejecuta la función **tone()** la cual produce un sonido. Esta función necesita de tres argumentos: el número de pin de salida que producirá el sonido (en este caso el pin 8) que frecuencia se va a generar (determinada por la variable tono) y durante cuanto tiempo va a sonar esa nota (probar con 20 mili segundos para comenzar).

Después de ejecutar la función anterior se introduce un retraso de 10 mili segundos para que le de tiempo a Arduino a generar la nota. Se realiza mediante la instrucción **delay(10)**.

COMO SE UTILIZA

Nada mas alimentar la placa Arduino se genera una ventana de 5 segundos para calibrar el sensor; esto se indica por que el diodo LED amarillo de la placa se enciende durante todo este tiempo. En este momento acercar la mano a la foto resistencia, a continuación alejar la mano para no hacer sombra a dicha foto resistencia, de esta manera el programa establece los límites del rango de trabajo en función de la iluminación ambiente. Los movimientos de la mano en el momento de realizar la calibración deberán ser lo más parecidos a los movimientos que se van a realizar después de la calibración para generar los sonidos.

Después de 5 segundos, la calibración estará completada, y el diodo LED de la placa Arduino se apagará. Cuando esto suceda, ¡se debe de escuchar un sonido a través del zumbador piezoeléctrico! Ahora si se mueva la mano sobre la foto resistencia deberá de variar la frecuencia de la nota que se escucha y de esta forma se podrán generar notas con diferentes frecuencias. El efecto final será parecido a la música de fondo que se suele escuchar en las películas de miedo.


```
19 void loop() {
20   ValordelSensor = analogRead(A0);

21   int tono =
       map(ValordelSensor,ValorMinimoSensor,ValorMaximoSensor,50,4000);

22    tone(8,tono,20);

23    delay(10);
24  }
```

El rango que se define dentro de la función **map()** es bastante amplio, se puede intentar modificar estos valores para conseguir que las frecuencias que se generan se ajusten más a su gusto sobre un estilo musical.

La función **tone()** trabaja de una forma muy parecida a PWM cuando se usa con la instrucción **analogWrite(),** pero con la diferencia que al usar esta instrucción la frecuencia de los impulsos generados no cambia: es posible cambiar el ratio de los impulsos durante este periodo de tiempo y así poder variar la relación cíclica sin variar la frecuencia (ver figura 1, PWM 50 y PWM200). Con **tone()** también se envían impulsos pero además se puede cambiar la frecuencia de estos impulsos. La función **tone()** siempre produce impulsos con una relación cíclica del 50%, (la mitad del tiempo la señal está en estado alto y la otra mitad está en estado bajo), puede ver un ejemplo en la figura 1 con los tonos de 440 y 880 (página 71). Con **tone()** no se puede variar la relación cíclica de una señal.

La función tone() ofrece la posibilidad de generar diferentes frecuencias cuando son enviadas a un altavoz o a un zumbador. Cuando se utilizan sensores en un circuito divisor de tensión, no es posible conseguir que los valores varíen todo el rango entre 0 y 1024. Al calibrar los sensores, es posible cambiar estos valores de rango para que se puedan usar.