Parte 2

Lección 6

Zumbador Pasivo

Resumen

En esta lección, usted aprenderá cómo utilizar una señal acústica pasiva.

El propósito del experimento es generar ocho sonidos, cada sonido dura 0,5 segundos: de Alto Do (523Hz), Re (587 hertzios), Mi (659Hz), Fa (698Hz), tan (784Hz), La (880Hz), Si (988Hz) hacer agudos (1047Hz).

Componente necesario:

(1) x Elegoo Uno R3

(1) x zumbador pasiva de

(2) x F M cables (cables de hembra a macho DuPont)

Introducción del componente

Zumbador pasiva:

- **El** principio de funcionamiento del zumbador pasiva está utilizando PWM generar audio para hacer el aire para que vibre. Debidamente cambiado tanto como la frecuencia de vibración, puede generar diferentes sonidos. Por ejemplo, enviando un pulso de 523Hz, puede generar Alto, pulso de 587Hz, puede generar el rango medio, pulso de 659Hz, que puede producir Mi de rango medio. Por el timbre, puede reproducir una canción.
- **Nosotros** debemos tener cuidado de no utilizar la función de () escritura analógica Kit UNO R3 para generar un pulso el timbre, porque la salida de pulso de analógico (de escritura) se fija (500 Hz).

Conexión Esquema

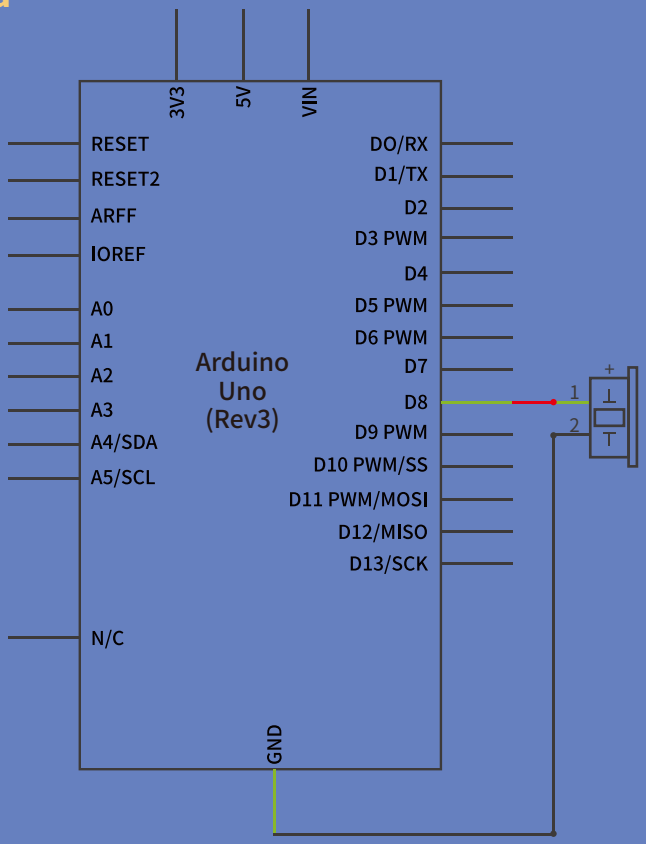
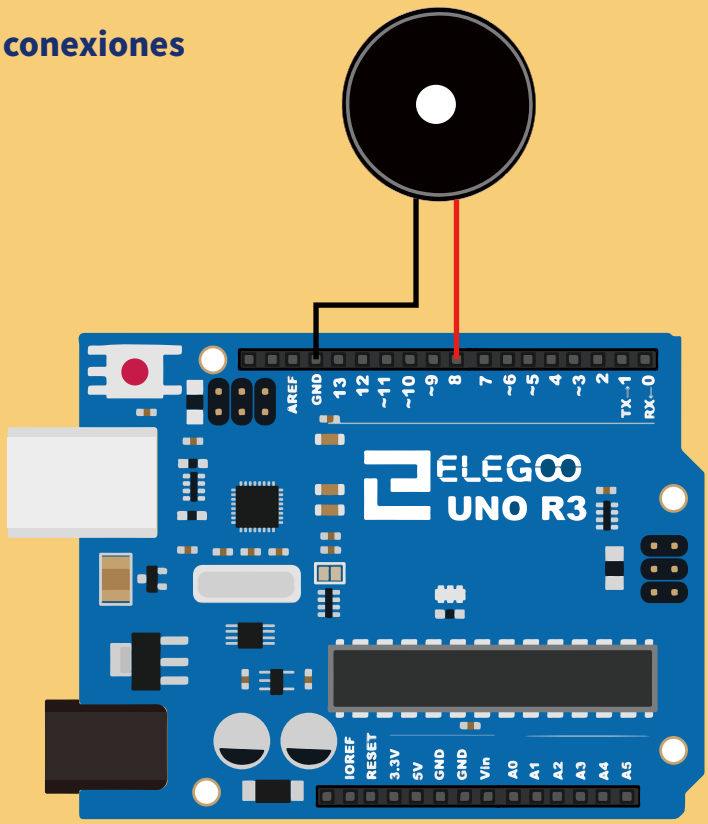


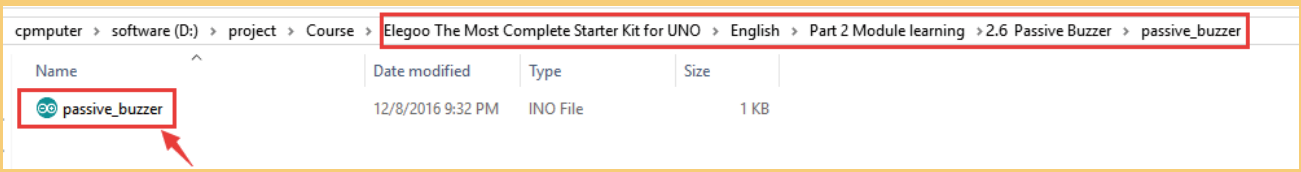
Diagrama de conexiones



Cableado el zumbador conectado a la placa UNO R3, rojo (positivo) que el pin8, cable negro (negativo) a la tierra.

Código

- Abra el programa:



- Para poder ejecutar esto, asegúrese antes de haber instalado la librería <pitches> o vuelva a instalarla, si es necesario. De lo contrario, su código no funcionará.

- Para obtener más información sobre carga el archivo de biblioteca, ver Lección 5 parte1.

```
int melody [] = { NOTE_C5, NOTE_D5, NOTE_E5, NOTE_F5, NOTE_G5, NOTE_A5, NOTE_B5, NOTE_C6};
int melody [] = {....}   : is a array.
```

array
[Tipos de datos]
Descripción

- Una matriz es una colección de variables a las que se accede con un número de índice. Las matrices en el lenguaje de programación C ++ en los que se escriben los programas de Arduino pueden ser complicadas, pero el uso de matrices simples es relativamente sencillo.

Creando (declarando) una matriz

- Todos los métodos a continuación son formas válidas de crear (declarar) una matriz.
Por ejemplo:

```
int myInts[6];
int myPins[] = {2, 4, 8, 3, 6};
int mySensVals[6] = {2, 4, -8, 3, 2};
char message[6] = "hello";
```

- Puede declarar una matriz sin inicializarla como en “myInts”.

- En “myPins” declaramos una matriz sin elegir explícitamente un tamaño. El compilador cuenta los elementos y crea una matriz del tamaño apropiado.

- Finalmente, puede inicializar y dimensionar su matriz, como en “mySensVals”. Tenga en cuenta que al declarar una matriz de tipo “char”, se requiere un elemento más que los especificados en su inicialización para contener el carácter nulo requerido.

Accediendo a una matriz

- Las matrices están indexadas a cero, es decir, en referencia a la inicialización de la matriz anterior, el primer elemento de la matriz está en el índice 0, por lo tanto
- mySensVals [0] == 2, mySensVals [1] == 4, y así sucesivamente.
- Esto también significa que en una matriz con diez elementos, el índice nueve es el último elemento. Por ejemplo:

```
int myArray[10]={9, 3, 2, 4, 3, 2, 7, 8, 9, 11};
// myArray[9]   contains 11
// myArray[10]   is invalid and contains random information (other memory address)
```

- Por esta razón, debe tener cuidado al acceder a las matrices. Acceder más allá del final de una matriz (usando un número de índice mayor que el tamaño de matriz declarado - 1) es leer de la memoria que está en uso para otros fines. Leer en estas posiciones, probablemente no sirva sino para generar datos no válidos. Escribir en ubicaciones de memoria aleatorias es definitivamente una mala idea y a menudo puede conducir a resultados erróneos, como bloqueos o mal funcionamiento del programa. Esto también puede ser un error difícil de rastrear.
- A diferencia de BASIC o JAVA, el compilador de C ++ no verifica si el acceso a la matriz está dentro de los límites permitidos según el tamaño de la matriz que ha declarado.

Para asignar un valor a una matriz:

- mySensVals[0] = 10;

Para recuperar un valor de una matriz:

- x = mySensVals[4];

```
tone(8 , melody[thisNote] , duration);
```

tone()

- **[E/S Avanzado]**

Descripción

Genera una onda cuadrada de la frecuencia especificada (y 50% de ciclo de trabajo) en un pin. Se puede especificar una duración; de lo contrario, la onda continúa hasta que se llame a noTone (). El pin se puede conectar a un zumbador u otro altavoz para reproducir tonos.

- Solo se puede generar un tono a la vez. Si ya se está reproduciendo un tono en un pin diferente, la llamada al tono () no tendrá efecto. Si el tono se reproduce en el mismo pin, la llamada establecerá su frecuencia.
- El uso de la función tone () interferirá con la salida PWM en los pines 3 y 11 (en placas que no sean Mega).
- No es posible generar tonos inferiores a 31Hz.

Syntax

```
tone(pin, frequency)
tone(pin, frequency, duration)
```

Parámetros

pin: el pin de Arduino en el que se generará el tono.
frequency: la frecuencia del tono en hercios. Tipos de datos permitidos: UNSIGNED INT.
duration: la duración del tono en milisegundos (opcional). Tipos de datos permitidos: unsigned long.

Devuelve

- Nada

Notas y advertencias

- Si desea ejecutar diferentes tonos en múltiples pines, debe llamar a noTone() en un pin antes de llamar a tone() en el siguiente pin.