BUZZER

QUE ES

Un zumbador o buzzer son considerados dispositivos electrónicos que actúan como un transductor* y se encargan de transformar energía eléctrica en acústica, es decir, permiten convertir una señal eléctrica en una onda de sonido. Son actuadores polarizados, no disponen de electrónica interna, por ello seremos nosotros quienes proporcionaremos una señal eléctrica para lograr el sonido deseado.

*instrumento o dispositivo capaz de transformar la energía disponible en una magnitud física dada en otra magnitud)

Estos dispositivos disponen de dos terminales; una positiva donde se envía la señal eléctrica y otra negativa que por lo general puesta a tierra. Comúnmente estas terminales son indicadas a través de colores: rojo (terminal positiva) y negro (terminal puesta a tierra).

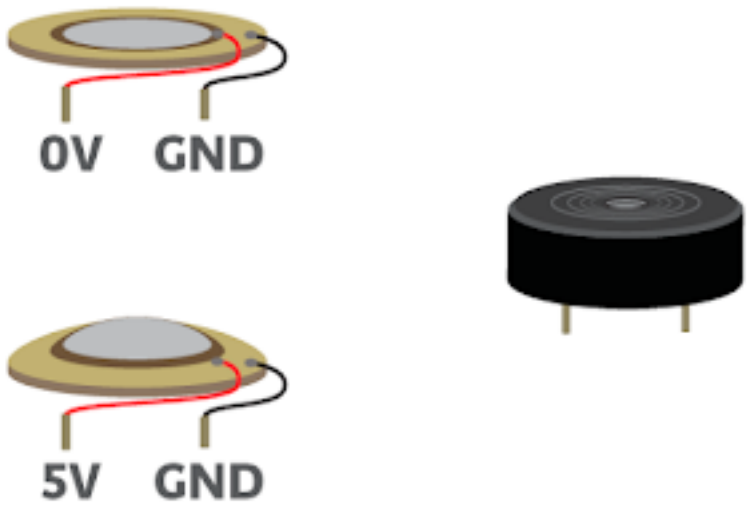
Se dividen en dos grupos:

El **zumbador piezoeléctrico** en la que está pegado un disco cerámico piezoeléctrico sobre una placa vibrante. Una tensión alternativa aplicada sobre este elemento causará una contracción o una expansión diametral del elemento. Esta característica se utiliza para hacer plegar una placa vibrante, lo que genera los sonidos.

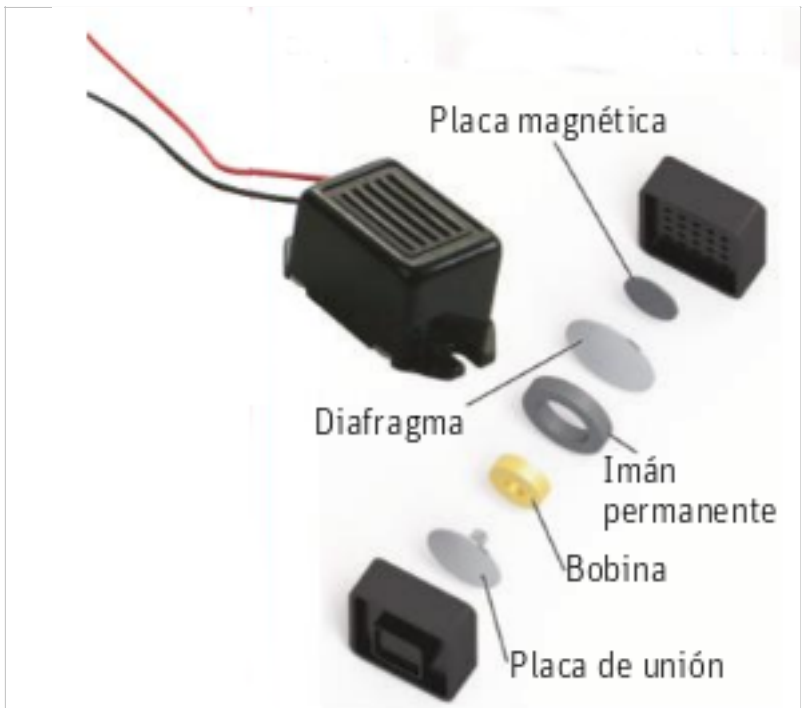
Dentro de los piezoeléctricos podemos encontrar 2 tipos:

- **BUZZER ACTIVO O CON OSCILADOR:** disponen de un multi-vibrador u oscilador interno, el externo solo necesita proporcionar voltaje para enviar un sonido de frecuencia fija. No reproduce tonos o música.

- **BUZZER PASIVO O SIN OSCILADOR:** no disponen de una fuente de oscilador interno, por ello la necesidad de la existencia de circuitos externos para proporcionar una cierta frecuencia de la señal de accionamiento. En la mayoría de casos, se acompaña de una placa; que incorpora un transistor y resistencias permitiendo la funcionalidad del zumbador.



El **buzzer electromagnético** ensaque que intercambia los materiales por un imán permanente y una bobina. por la que circula una corriente que hace vibrar una pequeña chapa de acero que produce un sonido característico

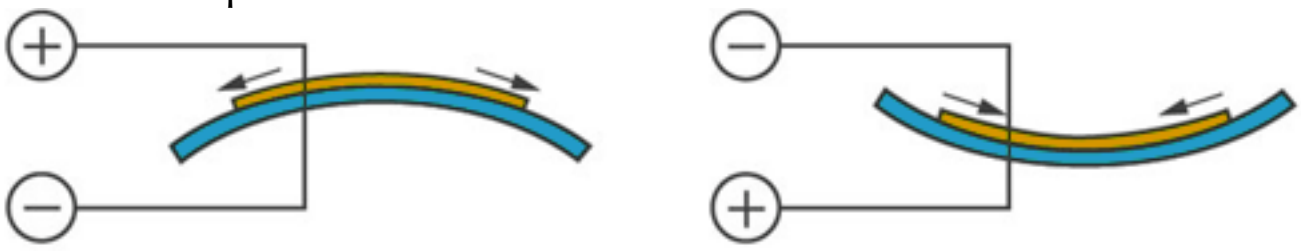


Constitución de un zumbador

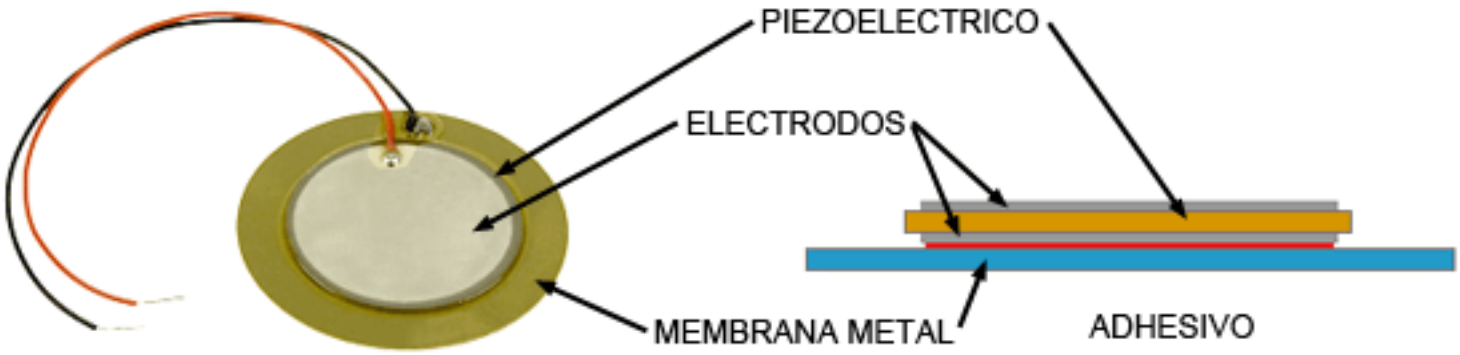

www.ingmecafenix.com

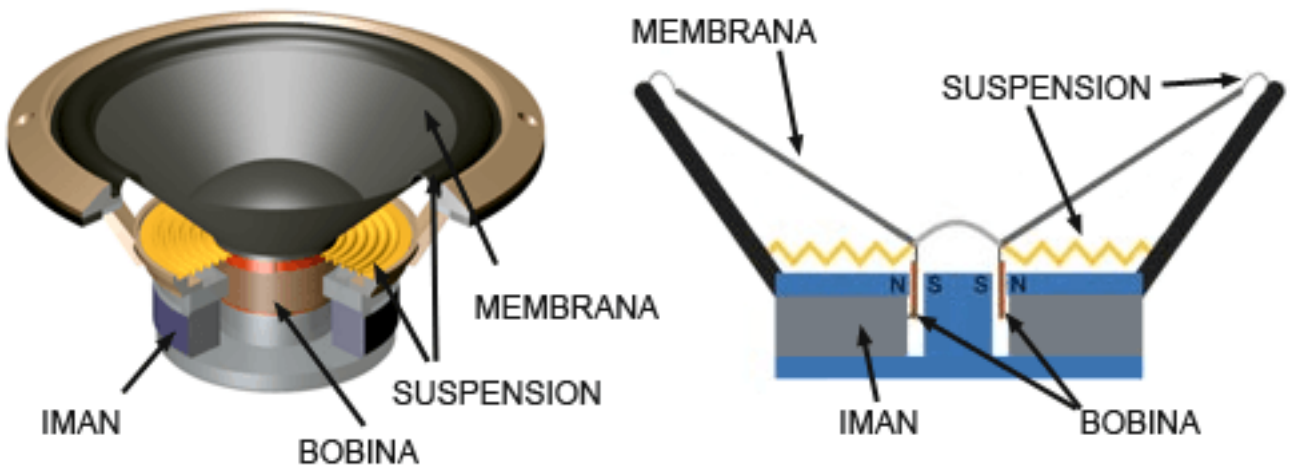
FUNCIONAMIENTO BUZZERS PIEZOELÉCTRICOS

Los materiales piezoeléctricos tiene la propiedad especial de variar su volumen al ser atravesados por corrientes eléctricas.



Un buzzer aprovecha este fenómeno para hacer vibrar una membrana al atravesar el material piezoeléctrico con una señal eléctrica



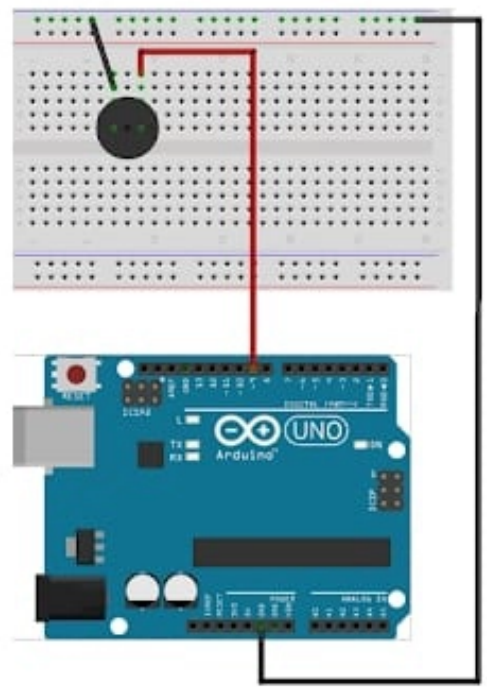


Al hacer circular una corriente por la bobina el campo magnético resultante genera una atracción con el imán, haciendo vibrar la membrana.
Los altavoces, por presentan una mejor calidad de sonido pero, en general, necesitan mayor potencia y es necesario disponer de dispositivos de amplificación para su uso.

Para generar un sonido de un tono, es decir de una sola frecuencia, basta con aplicar una onda cuadrada a la terminal positiva del buzzer o zumbador para que éste transforme la onda cuadrada a un sonido con su correspondiente frecuencia.

INTEGRACION POR ARDUINO:

Montaje, Códigos y Pines



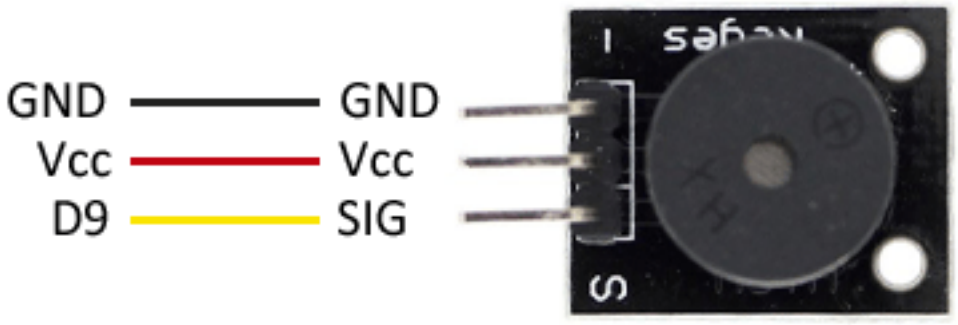
Su **integración con Arduino** no puede ser más sencilla, tanto si compras un buzzer normal como un módulo pasivo para Arduino se puede conectar muy fácilmente y el código que tienes que escribir en Arduino IDE es verdaderamente simple también (la base, luego dependerá de lo que quieras agregar tú).

En cuanto a un **ejemplo de código simple**, podría ser el siguiente, en el que el buzzer emite sonido 1 segundo y se detiene, vuelve a producir 1 segundo, y así:

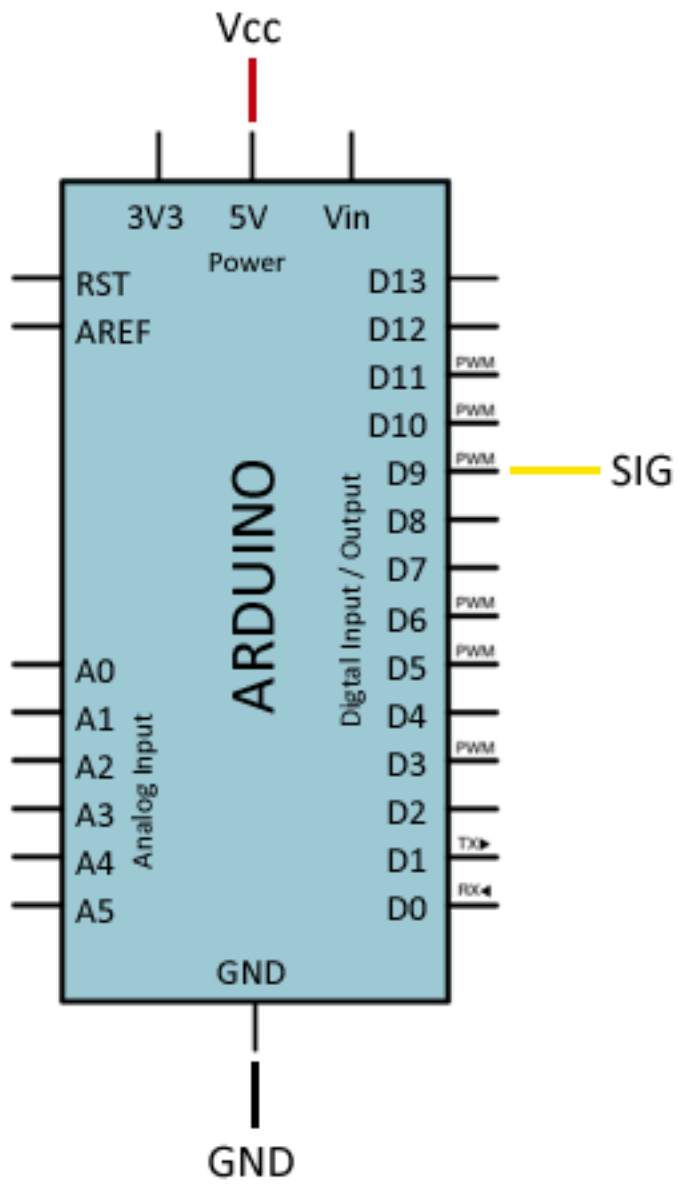
```
/* Programa simple para emitir pitidos de 1 segundo
intermitentes */

const int buzzer = 9; //El pin al que se conecta el buzzer es el
9


void setup(){

  pinMode(buzzer, OUTPUT); // Pin 9 declarado como salida

}

void loop(){

  tone(buzzer, 50); // Envía señal de 1Khz al zumbador
  delay(1000);
  noTone(buzzer);     // Detiene el zumbador
  delay(1000);        //Espera un segundo y vuelve a repetir el
bucle

}
```

ESQUEMA DE MONTAJE

Si usamos una de las placas comerciales para pequeños proyectos y hobbies, que incorporan la electrónica y terminales necesarios, la conexión con Arduino es realmente sencilla. Simplemente alimentamos el módulo conectando Vcc y GND a Arduino, y la entrada de señal a cualquier salida digital de Arduino.
El esquema de conexión visto desde el componente sería el siguiente



Mientras que el esquema de conexión visto desde Arduino quedaría así



EJEMPLOS DE CÓDIGO

Arduino dispone de dos funciones que nos permiten generar fácilmente señales eléctricas para convertir en sonido, usando cualquiera de las salidas digitales disponibles.
Estas funciones son tone() y noTone() y, como su nombre indican, permiten generar o detener la señal del tono en un pin.

```
tone(pin, frecuencia); //activa un tono de frecuencia determinada en un pin dado
noTone(pin);           //detiene el tono en el pin
```

La función tone() también permite especificar la duración del sonido generado.

```
tone(pin, frequencia, duracion);  //activa un tono de frecuencia y duracion determinados en un pin dado
```

Pese a su sencillez, al usar las funciones para la generación de tone tenemos que asumir importantes limitaciones.

- La función Tone emplea el Timer 2, por lo que mientras este funcionando no podremos usar las salidas PWM en los pines 3 y 11 en Arduino Nano y Uno (pines 9 y 10 en Arduino Mega).
- No podemos usar la función tone() en dos pines de forma simultánea. Deberemos apagar el tono con la funcion noTone() antes de poder usarlo en otro pin.
- Los rangos de la función tone son de 31 Hz a 65535 Hz.

El siguiente código muestra el uso de estas funciones estas funciones en un ejemplo simple, en el que empleamos el buzzer o altavoz conectado en el Pin9 para generar una función de 440Hz durante un segundo, pararlo durante 500ms, y finalmente un tono de 523Hz durante 300ms, para repetir el programa tras una pausa de 500ms.
const int pinBuzzer = 9;

```
void setup()
{
}

void loop()
{
  //generar tono de 440Hz durante 1000 ms
  tone(pinBuzzer, 440);
  delay(1000);

  //detener tono durante 500ms
  noTone(pinBuzzer);
  delay(500);

  //generar tono de 523Hz durante 500ms, y detenerlo durante 500ms.
```

```
  tone(pinBuzzer, 523, 300);
  delay(500);
}
```

El siguiente, también muy básico, emplea un array con frecuencias que recorremos secuencialmente para realizar un barrido que aproxima las distintas notas musicales.

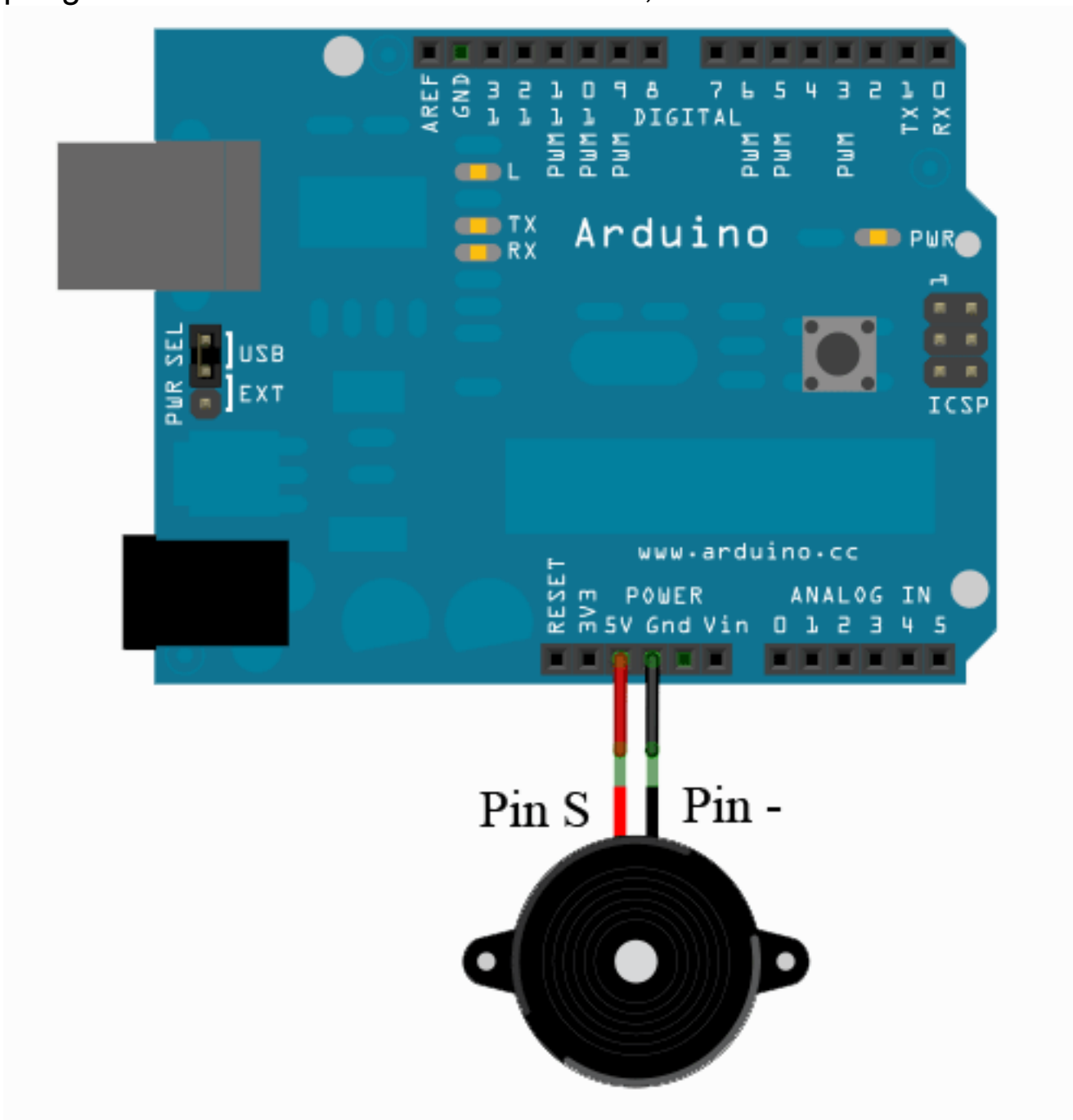
```
const int pinBuzzer = 9;

const int tonos[] = {261, 277, 294, 311, 330, 349, 370, 392, 415,
440, 466, 494};
const int countTonos = 10;

void setup()
{
}

void loop()
{
  for (int iTono = 0; iTono < countTonos; iTono++)
  {
   tone(pinBuzzer, tonos[iTono]);
   delay(1000);
  }
  noTone(pinBuzzer);
}
```

PINES

Arduino Compatible Active Speaker Buzzer Module - BlacK

Tiene tres terminales (pines), pero **solo utiliza dos** de ellos el pin **- (GND)** y el pin **S (Señal)**. (En otro modelo: **IN**)
El pin intermedio está destinado a la alimentación Vcc de 5 V, pero en este módulo de la izquierda no está conectado, así que solo utilizamos el **pin - y el pin S.**
- Si conectamos
el **Pin - a GND** del Arduino y
el **Pin S a 5 V** del Arduino, oiremos un sonido constante.
El Pin central no lo conectamos.

Podríamos conectar el Pin S a un pin de salida, por ejemplo el Pin 3, y según pongamos esa salida en HIGH o en LOW, se oirá el sonido o no.

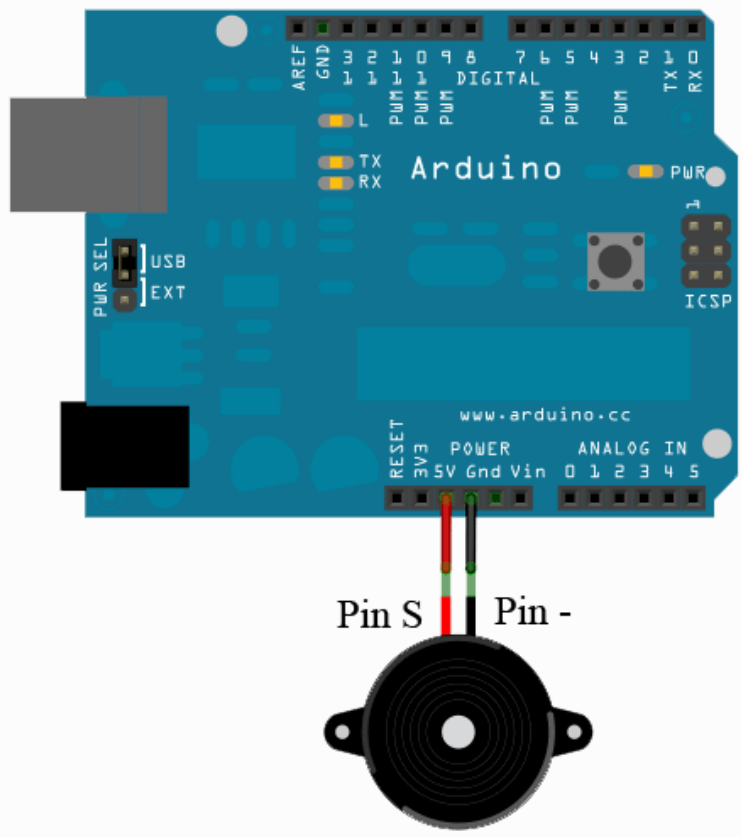


En internet podemos encontrar muchos programas de ejemplos.

1.- Conectamos el **Pin S del Buzzer al Pin Digital 2** del Arduino. Sonará un pitido contínuo.

- Si conectamos
el **Pin - a GND** del Arduino y
el **Pin S a 5 V** del Arduino, oiremos un sonido constante.
El Pin central no lo conectamos.

Podríamos conectar el Pin S a un pin de salida, por ejemplo el Pin 3, y según pongamos esa salida en HIGH o en LOW, se oirá el sonido o no.



Programa para el Arduino.

```
int buzzPin = 2; // Connect Buzzer on Digital Pin2
void setup()
{
pinMode(buzzPin, OUTPUT);
}
void loop()
{
digitalWrite(buzzPin, HIGH);
delayMicroseconds(50);
digitalWrite(buzzPin, LOW);
delayMicroseconds(50);
}
```

Programa para el Arduino.

```
// Buzzer example function for the CEM-1203 buzzer (Sparkfun's part #COM-07950).
// by Rob Faludi
// http://www.faludi.com

void setup() {
pinMode(9, OUTPUT); // set a pin for buzzer output
}

void loop() {
buzz(9, 2500, 500); // buzz the buzzer on pin 4 at 2500Hz for 500 milliseconds
delay(100); // wait a bit between buzzes
}


void buzz(int targetPin, long frequency, long length) {
long delayValue = 1000000/frequency/2; // calculate the delay value between
transitions
//// 1 second's worth of microseconds, divided by the frequency, then split in
half since
//// there are two phases to each cycle
long numCycles = frequency * length/ 1000; // calculate the number of cycles for
proper timing
//// multiply frequency, which is really cycles per second, by the number of
seconds to
//// get the total number of cycles to produce
for (long i=0; i < numCycles; i++){ // for the calculated length of time...
digitalWrite(targetPin,HIGH); // write the buzzer pin high to push out the
diaphram
delayMicroseconds(delayValue); // wait for the calculated delay value
digitalWrite(targetPin,LOW); // write the buzzer pin low to pull back the diaphram
delayMicroseconds(delayValue); // wait againf or the calculated delay value
}
}
```

Programa para el Arduino.

```
// Buzzer example function for the CEM-1203 buzzer (Sparkfun's part #COM-07950).
// by Rob Faludi
// http://www.faludi.com

// Additions by Christopher Stevens
// http://www.christopherstevens.cc

//referenced from http://www.phy.mtu.edu/~suits/notefreqs.html
//starting with F noteFreqArr[1]
int noteFreqArr[] = {
49.4, 52.3, 55.4, 58.7, 62.2, 65.9, 69.9, 74, 78.4, 83.1, 88, 93.2,
98.8, 105, 111, 117, 124, 132, 140, 148, 157, 166, 176, 186,
198, 209, 222, 235, 249, 264, 279, 296, 314, 332, 352, 373,
395, 419, 444, 470, 498, 527, 559, 592, 627, 665, 704, 746,
790, 837, 887, 940, 996, 1050, 1110, 1180, 1250, 1320, 1400, 1490,
1580, 1670, 1770, 1870, 1990, 2100};

void setup() {
pinMode(4, OUTPUT); // set a pin for buzzer output
}

void playNote(int noteInt, long length, long breath = 20) {
length = length - breath;
buzz(4, noteFreqArr[noteInt], length);
if(breath > 0) { //take a short pause or 'breath' if specified
delay(breath);
}
}

void loop() {
//main course
playNote(24,500);

playNote(17,1000);
playNote(19,250);
playNote(22,250);
playNote(21,250);
playNote(19,250);
playNote(24,500);
playNote(24,500);
playNote(24,250);
playNote(26,250);
playNote(21,250);
playNote(22,250);
playNote(19,500);
playNote(19,500);
playNote(19,250);
playNote(22,250);
playNote(21,250);
playNote(19,250);
playNote(17,250);
playNote(29,250);
playNote(28,250);
playNote(26,250);
playNote(24,250);
playNote(22,250);
playNote(21,250);
playNote(19,250);

playNote(17,1000);
```

Programa para el Arduino.

```
/*
Piezo

This example shows how to run a Piezo Buzzer on pin 9
using the analogWrite() function.

It beeps 3 times fast at startup, waits a second then beeps continuously
at a slower pace

*/

void setup() {
// declare pin 9 to be an output:
pinMode(9, OUTPUT);
beep(50);
beep(50);
beep(50);
delay(1000);
}

void loop() {
beep(200);
}

void beep(unsigned char delayms){
analogWrite(9, 20); // Almost any value can be used except 0 and 255
// experiment to get the best tone
delay(delayms); // wait for a delayms ms
analogWrite(9, 0); // 0 turns it off
delay(delayms); // wait for a delayms ms
}
```

Programa para el Arduino.

```
int Pin = 9;

void setup() {
// nothing happens in setup
}

void loop() {
analogWrite(Pin, 250); // Cambiamos este valor para oir distintos sonidos.
}
```

Otra manifestación del efecto piezo eléctrico consiste en lo contrario, es decir **aplicamos una tensión alterna** en los extremos de un elemento piezo eléctrico y éste vibra.
Este es el efecto que utilizan estos **buzzer piezoeléctrico**, aplicamos una tensión variable a sus extremos y el elemento vibra según la frecuencia aplicada, esta vibración se acopla a una membrana que emite un sonido debido a su movimiento vibrante.

Lista de materiales:

- 1 Arduino Uno
- 1 Protoboard
- 1 Buzzer pasivo
- Cables dupont

Lo primero que debes hacer es armar el siguiente circuito imagen 5.0, una vez armado el circuito se vería como en la imagen 5.1

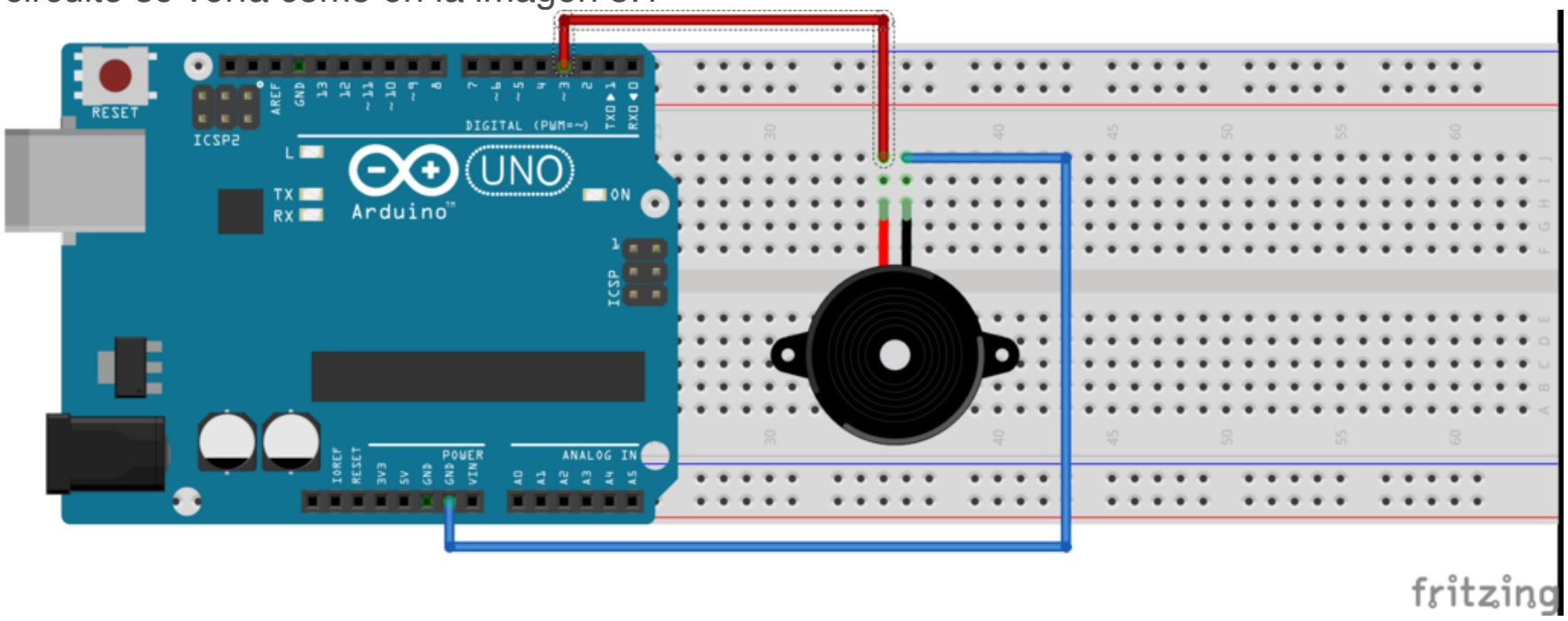


Imagen 5.0

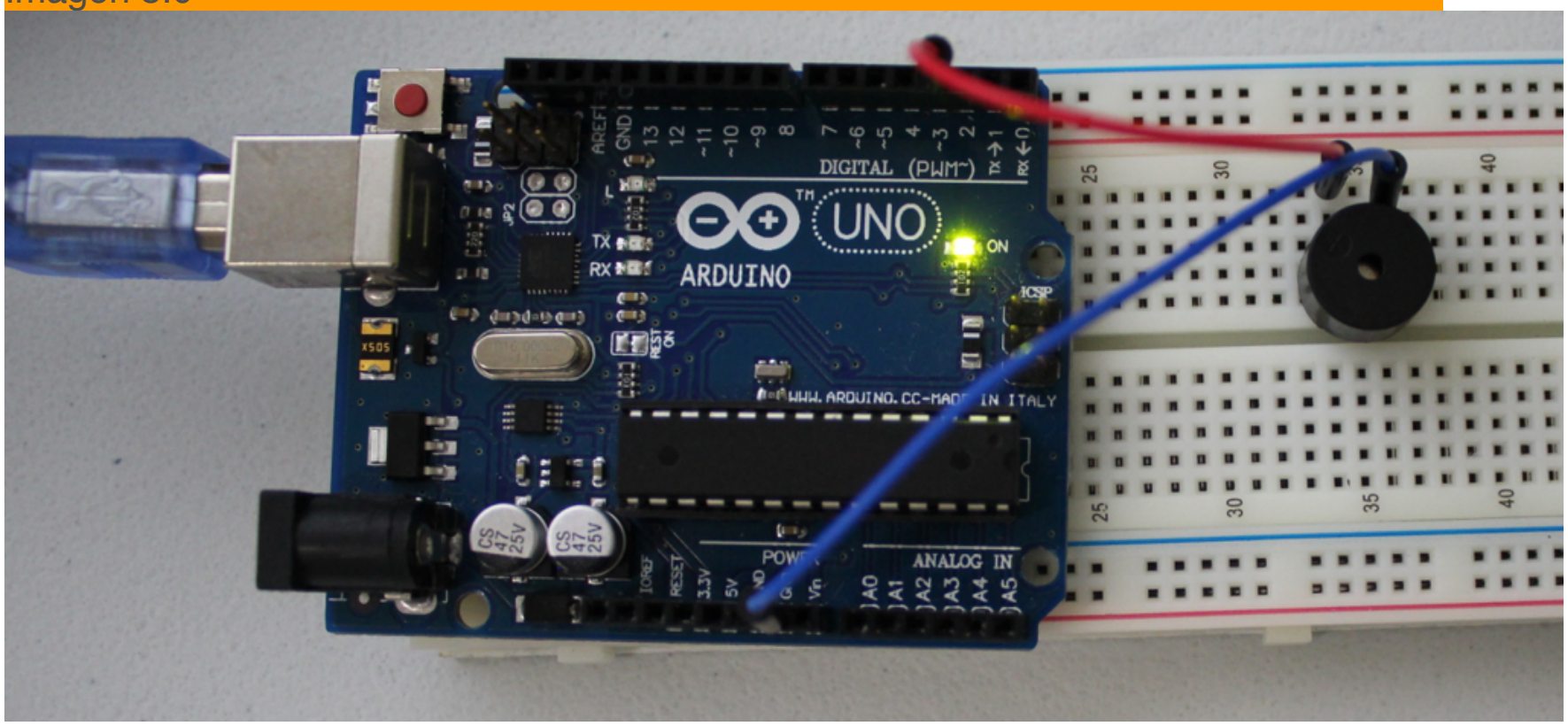


Imagen 5.1

Arduino dispone de 2 funciones que nos permiten generar fácilmente señales eléctricas para convertir en sonido, usando cualquiera de las salidas digitales disponibles.
Estas funciones son tone() y noTone() imagen 5.2 y, como su nombre indica, permiten generar o detener la señal del tono en un pin.

```
//Activa un tono de frecuencia determinada en un pin dado
tone(pin, frecuencia);
//Detiene el tono en el pin
noTone(pin);
```

Imagen 5.2

Para probar el buzzer pasivo usaremos la dos funciones mencionadas, el codigo queria como el de la imagen 5.3, guarda tu propia copia y subelo a la placa arduino

```
const int pinBuzzer = 3;

void setup()
{
}

void loop()
{
  //generar tono de 440Hz durante 1000 ms
  tone(pinBuzzer, 440);
  delay(1000);

  //detener tono durante 500ms
  noTone(pinBuzzer);
  delay(500);

  //generar tono de 523Hz durante 500ms, y detenerlo durante 500ms.
  tone(pinBuzzer, 523, 300);
  delay(500);
}

//Activa un tono de frecuencia determinada en un pin dado
tone(pin, frecuencia);
//Detiene el tono en el pin
noTone(pin);
```