HARDWARE YOU WILL NEED:

- Arduino board, I am using an Arduino Uno [1]
- Solderless breadboard [1]
- Jumper wire [2]
- 100 ohm resistor [1]
- Piezo speaker (aka piezo buzzer) [1]
- At least 12 cm dental floss, with a thick wax coating (mint flavored works best) [1]

A QUICK INTRO TO PIEZO SPEAKERS (AKA PIEZO BUZZERS)

Ahh, noise….

Birds make it, kids make it – it can be music to our ears or pure torture.

We are going to use a piezo buzzer to make some noise with Arduino.

A piezo buzzer is pretty sweet. It's not like a regular speaker that you might think of. It uses a material that's *piezoelectric*, it actually changes shape when you apply electricity to it. By adhering a piezo-electric disc to a thin metal plate, and then applying electricity, we can bend the metal back and forth, which in turn creates noise.

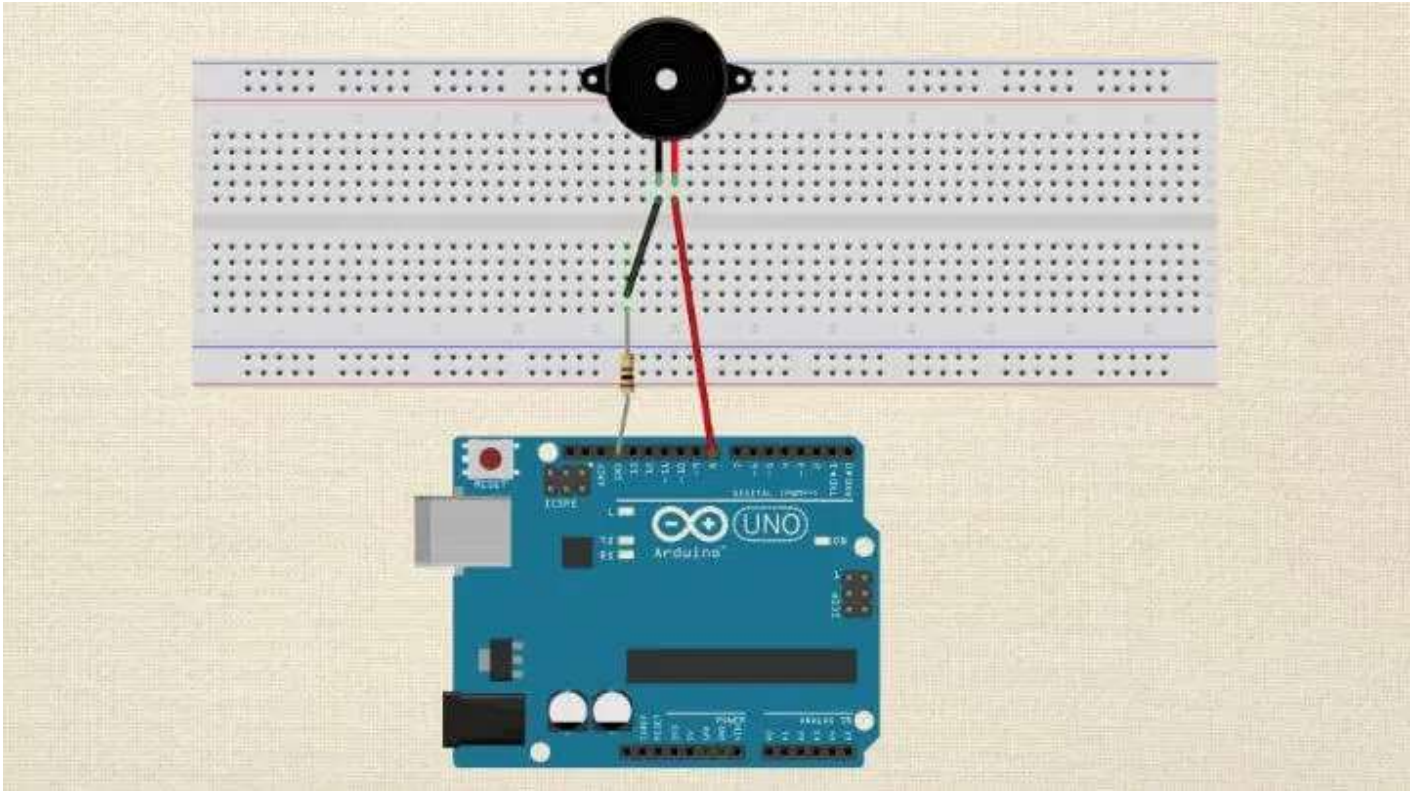
The faster you bend the material, the higher the pitch of the noise that's produced. This rate is called frequency. Again, *the higher the frequency, the higher the pitch of the noise we hear*.

So basically, by shocking the plate over and over really fast, we can make noise. I don't know who comes up with this stuff, but they're friggin' mean.

Now, let's set up this circuit.

HOW TO SET UP A SIMPLE PIEZO SPEAKER CIRCUIT USING ARDUINO

It’s painfully easy to set up a simple piezo speaker circuit with an Arduino.



1. Place the piezo buzzer into the breadboard, so that the two leads are on two separate rows.
2. Using jumper wires, connect the positive lead to Arduino digital pin 8. The case of the buzzer may have a positive sign (+) on it to indicate the positive lead (if not, then the red wire usually indicates the positive lead).
3. Connect the other lead to the 100 ohm resistor, and then to ground.

That’s it.

Let’s go ahead and jump into the Arduino sketch.

THE BASICS AND MORE OF USING THE TONE() FUNCTION

The tone() function works with two arguments, but can take up to three arguments. Let’s address the two required items first:

```
tone( pin number, frequency in hertz);
```

1. The pin number that you will use on the Arduino.
2. The frequency specified in hertz. Hertz are cycles per second.

The frequency is an unsigned integer and can take a value up to 65,535 – but if you are trying to make tones for the human ear, then values between 2,000 and 5,000 are where our ears are most tuned.

Here is a simple sketch demonstrating the tone() function:

```
//A sketch to demonstrate the tone() function

//Specify digital pin on the Arduino that the positive lead of piezo buzzer is attached.
int piezoPin = 8;

void setup() {

}//close setup

void loop() {

  /*Tone needs 2 arguments, but can take three
    1) Pin#
    2) Frequency - this is in hertz (cycles per second) which determines the pitch of the noise m
    3) Duration - how long teh tone plays
  */
  tone(piezoPin, 1000, 500);

  //tone(piezoPin, 1000, 500);
  //delay(1000);

}
```

As an experiment, try changing the second argument in tone() to 100, 1000, 10000, 650000 and listen to the effect it has on the audio signal.

You will notice that the higher the number, the higher the pitch that is created.

HOW TO SEPARATE THE NOISE – AKA MAKE A BEAT

What if we want to add some space between the noise, so we can get a beat?

We can try adding a delay(1000) after the tone(), but if you test this out, you will find it doesn’t get you anywhere.

This is because the tone() function uses one of the built in timers on the Arduino's micro-contoller. **tone() works independently of the delay() function.** You can start a tone and do other stuff – while the tone is playing in the background.

But, back to that question, *how can we separate the noise a little*?

Let's talk about that third parameter we can pass to the tone() function – it is the duration of the tone in milliseconds.

```
tone( pin number, frequency in hertz, duration in milliseconds);
```

If you want to generate distinct beats, and you want to do this with the delay() function, then you need to keep in mind what we just said, that the tone() function uses one of the built in timers on the Arduino board.

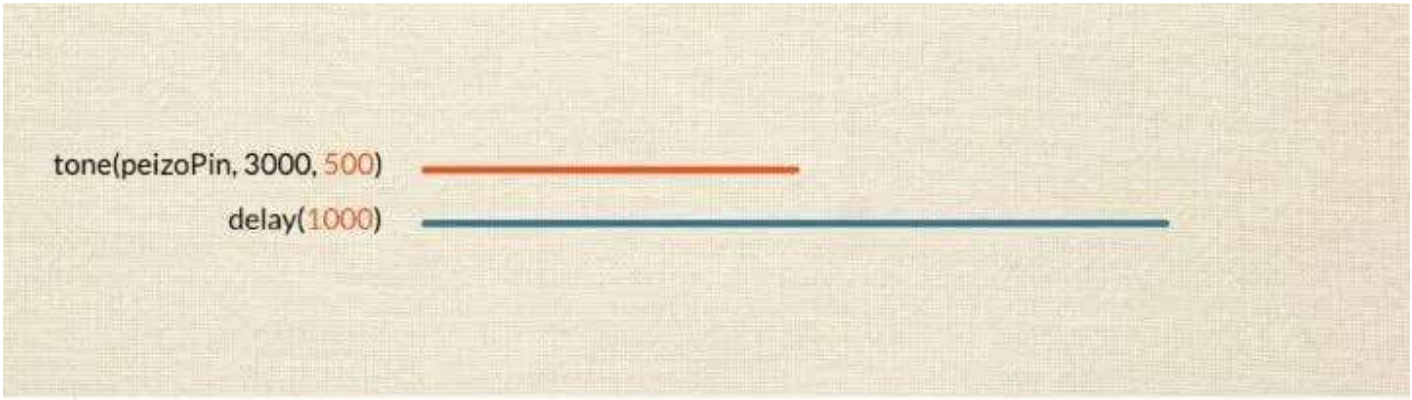
Therefore, if you use 500 milliseconds as the third argument in tone(), and follow that by a delay of 1000 milliseconds, you will only be creating a "quiet time" of 500 milliseconds.

```
//This code only generates a delay of 500 milliseconds between the tone

  tone( 8, 2000, 500);

  delay(1000);
```

Instead of the time being added together – as in 500 millisecond of noise, and then 1000 milliseconds of delay – the delay and the tone start at about the exact same time, so we get 500 millisecond of tone separated by 500 milliseconds of quiet.




```
16
17    tone(peizoPin, 3000, 500);
18
19    delay(1000);
20
21  }
```

It's a little quirky to wrap your mind around, but if you play around with it a little, you'll pick up on the intricacies.

THE LIMITS YOU SHOULD KNOW WHEN USING TONE()

Like everything else in the world, the tone() function has some limitations of which you should be aware. Let's touch on them here:

1. You can't use tone() while also using analogWrite() on pins 3 or 11. If you do – you get some whacky results – neither will work like you expect. That's because the tone() function uses the same built in timer that analogWrite() does for pins 3 and 11. It's worth trying just hear the weird noises.
2. You cannot generate a tone lower than 31 HZ. You can pass values 31 and less to the tone() function, but it doesn't mean you will get a good representation of it.
3. The tone() function cannot be used by two separate pins at the same time. Let's say you have two separate piezo speakers, each on a different pin. You can't have them both play at the same time. One has to be on, and then the other. Furthermore, before you can have the other pin use the tone() function, you must call the noTone() function and "turn off" the tone from the previous pin.

REVIEW THE LESSON

Here is a quick breakdown of what we just covered:

1. A piezo speaker use piezo-electric material to bend a metal diaphragm which makes noise.
2. The tone() function needs at least two arguments to operate (pin#, frequency in hertz), but can take a third argument, duration – which is in milliseconds.
3. The tone() function works independently of the delay() function. The duration of the tone() will be continuous if you don't use the third parameter.
4. The limitations of the tone() function include:
    1. Not being able to use PWM on pins 3 and 11 while you use tone()
    2. You can't go lower than 31 hertz.
    3. When using tone() on different pins, you have to turn off the tone on the current pin with noTone() before using tone() on a different pin.

I find it can facilitate learning to review what you cover after you learn about it. *The best teacher is practice though* – so get a piezo speaker and start making some mad jams with your Arduino! I would love to hear about them in the comments.