



### The Big Idea:

This lesson is about enabling an Arduino™ to play musical tones. This capability will be used in Lesson 17 to turn a common television remote control into a sort of musical instrument that produces different sounds when its buttons are pushed. That lesson, in turn, provides the technical base for using remotes for more sophisticated things and, eventually, creating devices that can generate their own messages.

### Background:

A **tone** is nothing more than the turning on and then off of something that moves the air. A violin string, for example, moves back and forth rapidly, moving the surrounding air. The musical note *A* is heard when a violin string moves 440 times per second. The number of vibrations per second is called the **frequency**.

A **speaker** is a device with a coil of wire suspended near a permanent magnet. The wire is glued to a paper cone called a diaphragm. When an electrical current moves through the wire, a magnetic field is set up that pushes against that of the magnet causing the cone to move.

10

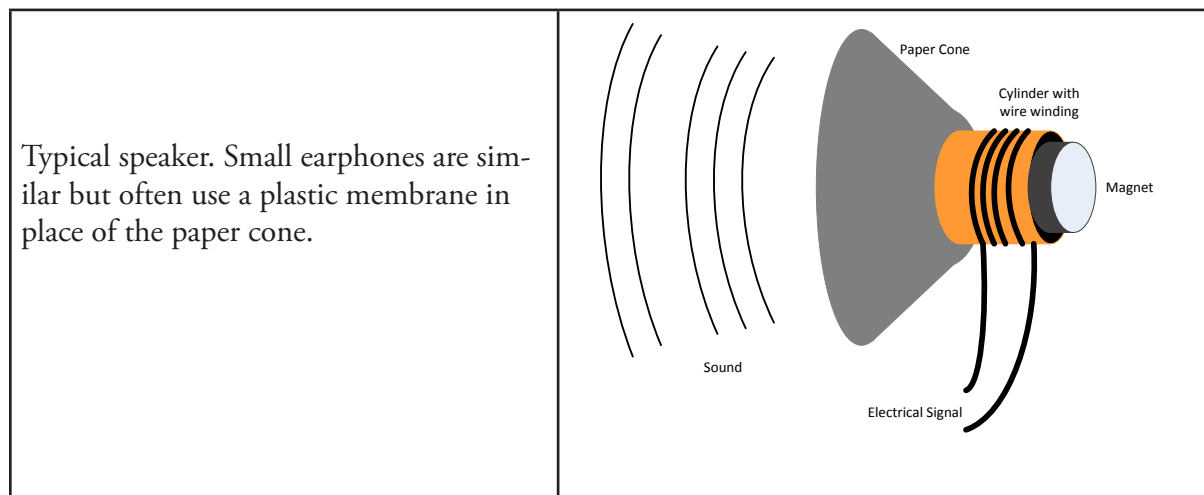


Figure 10-1. Speaker

If a speaker is connected to a digital port on an Arduino™ and that port is turned on and off, the speaker diaphragm will move in and out. As with the violin string, if the port turns on and off 440 times per second, the musical note *A* will be heard coming from the speaker.

As it happens, there is a C-language method that does precisely this. It is `tone()`. The C-language method `noTone()` stops a tone from playing.

Table 10-1. Vocabulary

Term	Definition
<b>coupling capacitor</b>	An electronic component that passes on current that is increasing or decreasing in magnitude but does not allow a steady flow of electrons. For audio it allows only the sound produced by the Arduino™ to reach the speaker.
<b>frequency</b>	The ratio of the number of times some repeating event occurs per unit of time. For audio the most common ratio is cycles per second. The unit name for this is the Hertz.
<b>Hertz</b>	The name of the unit of measure for cycles per second or frequency. Abbreviated as Hz. 1 Hertz = 1 cycle / second.
<b>period</b>	The duration of one cycle. For a frequency of 1000 Hz the period is 1/1000 second, or $1 \times 10^{-3}$ seconds.
<b>sine wave</b>	A repeating event the magnitude of which is sinusoidal. In electronics a sine wave of audio frequency is a pure tone.
<b>speaker</b>	A mechanical device that translates electrical current into the movement of air. Examples include loudspeakers and ear buds. Many technologies are used to perform this translation, but electromagnetism and piezo crystal are the most common.
<b>square wave</b>	A repeating event in which the resulting voltage is a square wave, high for one-half the period and low for one-half the period. The sound produced is not pure, but it is suitable for steady tones, sound effects, and simple melodies.
<b>tone</b>	The turning on and then off of something that moves the air.

### Description:

A "pure" tone is a *sine wave*. But an Arduino™ is a digital device. It simply turns on and off. The resulting voltage is then a *square wave*, high for one-half the period and low for one-half the period. The duration of one cycle is called the *period*. The speaker, being a mechanical device that takes time to respond to changes in voltage, is most sensitive to the output frequency, but other frequencies are present as well. As a result, the sound of a square wave is not pure, but the pitch is obvious to the listener. Square wave audio is suitable for steady tones, sound effects, and simple melodies.

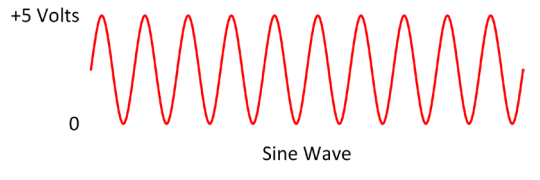
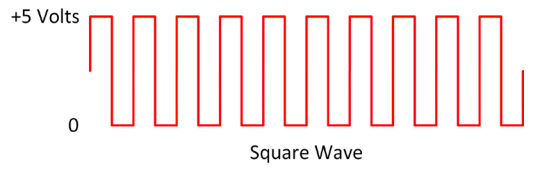
	<p>A sine wave in audio frequencies produces a pure, clean, and undistorted tone. The Arduino™ cannot produce a sine wave without added components.</p>
	<p>A square wave in audio frequencies produces a clear pitch that is dominant but not pure. Many other frequencies are present in lower volumes. The Arduino™ excels at producing square waves.</p>

Figure 10-2. Sine and square waves

The musical note *A* has a frequency of 440 cycles per second. The unit of frequency is the *Hertz*. So we say an *A* has a frequency of 440 Hertz. The duration of a single pulse, up and down, is the inverse of the frequency, as shown in Figure 10-3.

$$\text{duration} = 1 / \text{frequency in Hertz} \Rightarrow \text{duration of an } A = 1 / 440$$

$$\text{duration of an } A = 2.28 \text{ milliseconds.}$$

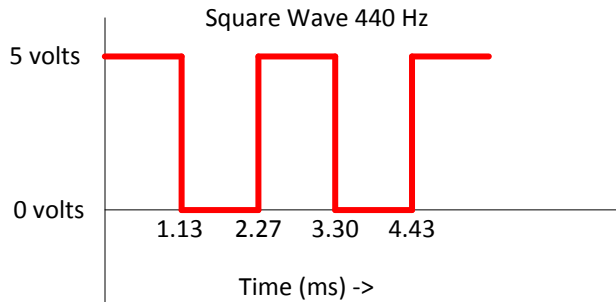


Figure 10-3. Portion of a square wave that produces the musical note *A*

To make an Arduino™ produce a sound, all that needs to be done is to alternately set a pin HIGH then LOW for the duration of half a cycle of the desired frequency. Fortunately, the Arduino™ method `tone()` does precisely that.

Table 10-2. Tone-related Arduino™ methods

Method Signature	Description
<code>tone( pin, frequency)</code>	<p>Sends a pulse with the specified frequency to the specified pin. The pulse is repeated until the <code>noTone()</code> method is called or the <code>tone()</code> method is called again.</p> <p><code>pin</code><sup>1</sup>: Pin number of the digital port.</p> <p><code>frequency</code>: Frequency of the tone.</p>
<code>tone( pin, frequency, duration)</code>	<p>Sends a pulse with the specified frequency to the specified pin. The pulse is repeated for the time specified in "duration."</p> <p><code>pin</code>: Pin number of the digital output port.</p> <p><code>frequency</code>: desired frequency of tone.</p> <p><code>duration</code><sup>2</sup>: duration of tone in milliseconds.</p>
<code>noTone(pin)</code>	<p>Stops a tone playing.</p> <p><code>pin</code>: specifies pin where note is to be stopped.</p>

Notes:

1. Even though `noTone` specifies a pin number, the Arduino™ will not play more than one tone at a time. If a tone is started on pin 3 and then another on pin 7, the tone on pin 3 will stop immediately when the tone on pin 7 begins.
2. The duration is observed only if no other tone is started and `noTone` is not called. In the code shown in Example 10-1, the 440 Hertz tone will not be heard because the 1000 Hertz tone begins immediately after the 440 tone was started.

**Example 10-1.**







```
tone(3, 440, 500); // won't be heard
tone(3, 1000, 500); // might be heard
```

The code shown in Example 10-2 will, however, play the 440 tone followed one-half second later by the 1000 Hertz tone.

*Example 10-2.*

```
tone(3, 440, 500); // will be heard  
delay(500);  
tone(3, 1000, 500); // will be heard  
delay(500);
```

**Materials:**

Quantity	Part	Image	Notes	Catalog Number
1	Arduino™ Uno		Single-board computer. This board is delicate and should be handled with care. When you are not using it, keep it in a box or plastic bag.	3102
1	USB Cable		This is a standard USB adapter cable with a flat connector on one end and a square connector on the other.	2301
1	Computer with at least one USB port and access to the Arduino™ website, <a href="http://www.arduino.cc">http://www.arduino.cc</a> .	---	The operating system of this computer must be Windows, Macintosh OS/X, or Linux.	---
1	Bread-board		Used for prototyping.	3104
As req'd	Jumper wires		Used with bread-boards for wiring the components.	3105
1	Capacitor		2.2 mfd.	0205
1	Speaker		Magnetic speaker 4, 8, or 16 ohm.	3119

## Procedure:

1. Connect a speaker to the Arduino™ as shown in Figure 10-4.

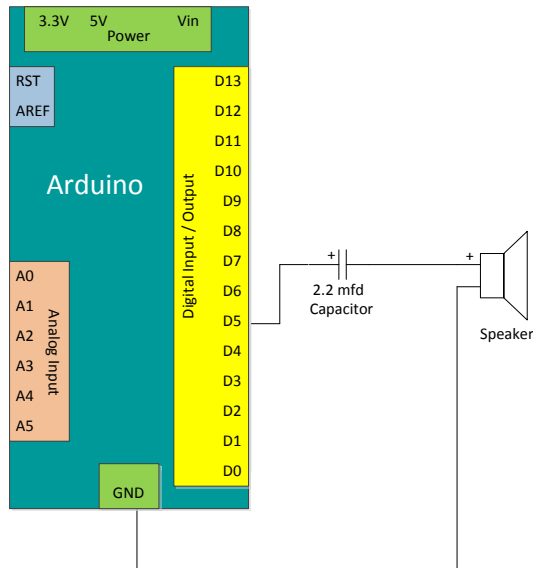


Figure 10-4. Schematic diagram of connection between Arduino™ and speaker

2. Create a new Arduino™ sketch. Name it Lesson10SimpleTones. Enter the test program as shown in Snippet 10-1.

### Snippet 10-1. Lesson10SimpleTones sketch

```
// Lesson10SimpleTones.ino
// <author>
// <date>

#include "pitches.h"

#define soundPin 5 // speaker pin

void setup(){ }

void loop(){
  tone(soundPin, NOTE_A4, 50);
  delay(600);
  tone(soundPin, NOTE_E4, 50);
  delay(600);
  tone(soundPin, NOTE_C4, 50);
  delay(600);
}
```

```
tone(soundPin, NOTE_E4, 50);  
delay(600);  
}
```

3. Save the sketch as `Lesson10SimpleTones`.
4. Close the Arduino™ IDE.
5. Download the file `itches.h`. Save it to the same folder that contains the `Lesson10SimpleTones.ino` file.
6. Open the Arduino™ IDE.
7. Using File -> Sketchbook open `Lesson10SimpleTones`. Two tabs should appear. One should be labeled `Lesson10SimpleTones`, the other `itches.h`.
8. Upload the program and listen for the sounds.
9. Experiment with the program until you have some feel for how to use `tone()` and `delay()`.

### Exercises:

#### *Exercise 10-1. Create sketch to play a recognizable melody*

Create a new Arduino™ sketch, `Lesson10Exercise1.ino`, that plays a recognizable melody.

#### *Exercise 10-2. Create sketch to generate sound effects*

Create a new Arduino™ sketch, `Lesson10Exercise2.ino`, that uses `for` loops to generate sound effects. Experiment with changing durations of sounds.

#### *Exercise 10-3. Amplify sounds*

Use "How to Make Arduino™ Tones Louder" to increase the volume of the tones coming from your Arduino.™

