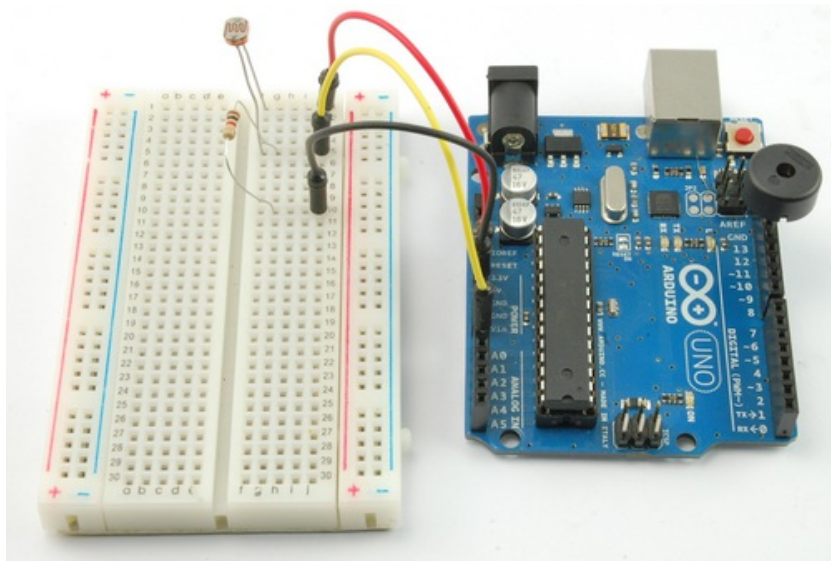


## Arduino Lesson 10. Making Sounds

Created by Simon Monk



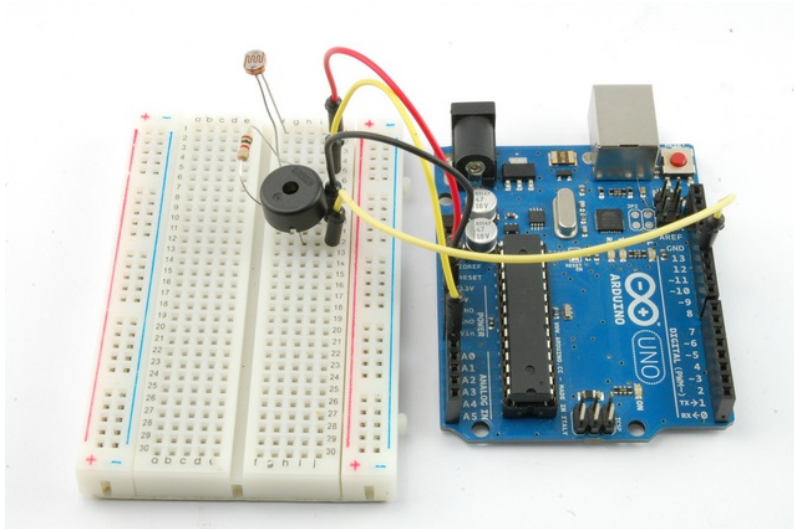
## Guide Contents

Guide Contents	2
Overview	3
Parts	4
Part	4
Qty	4
Playing a Scale	6
Sound	8
Pseudo-Theremin	9
Arduino Code	10
Other Things to Do	11


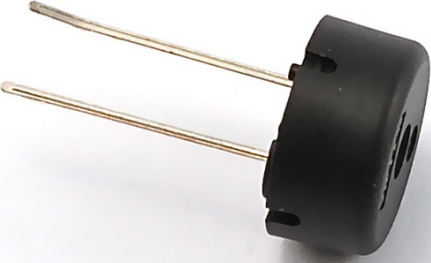
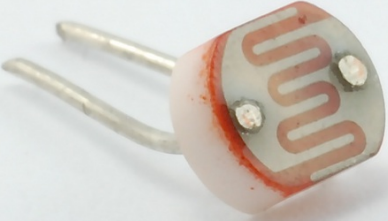
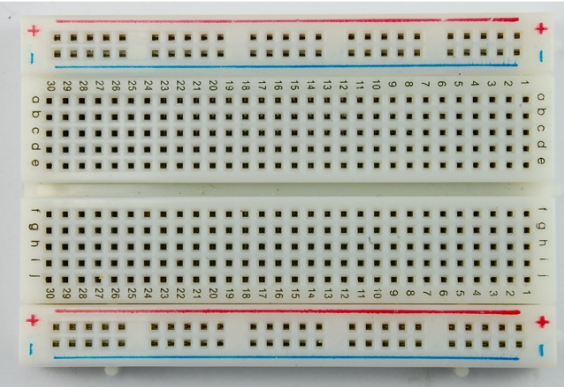
## Overview

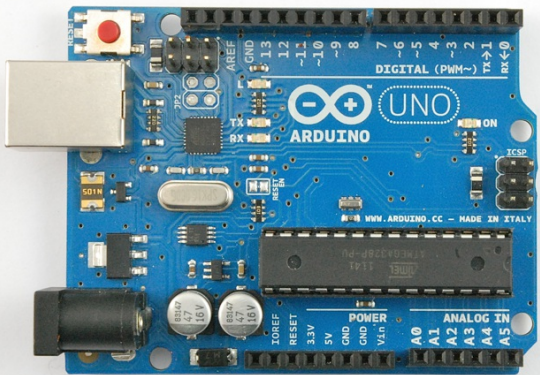
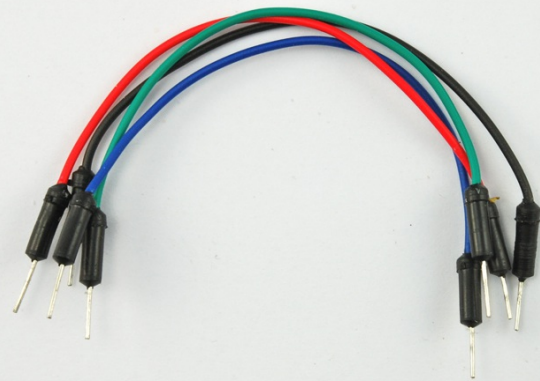
---

In this lesson, you will learn how to make sounds with your Arduino. First you will make the Arduino play a 'musical' scale and then combine this with a photocell, to make a Theremin-like instrument that changes the pitch played as you wave your hand over the photocell.



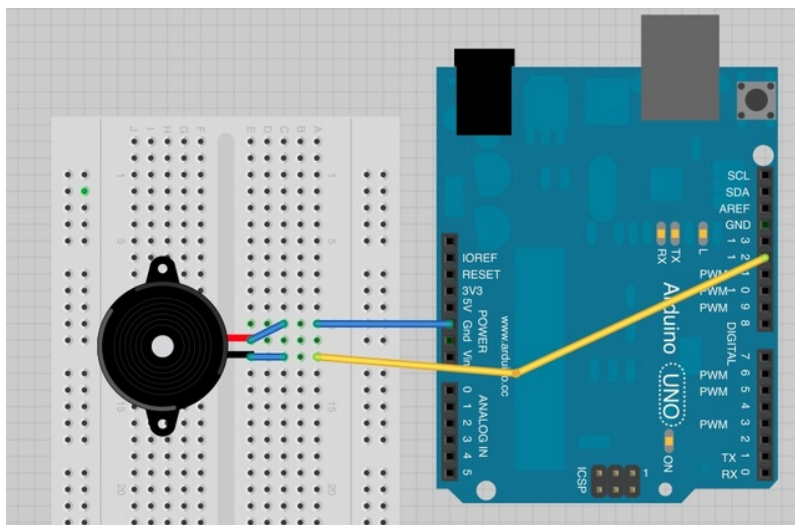
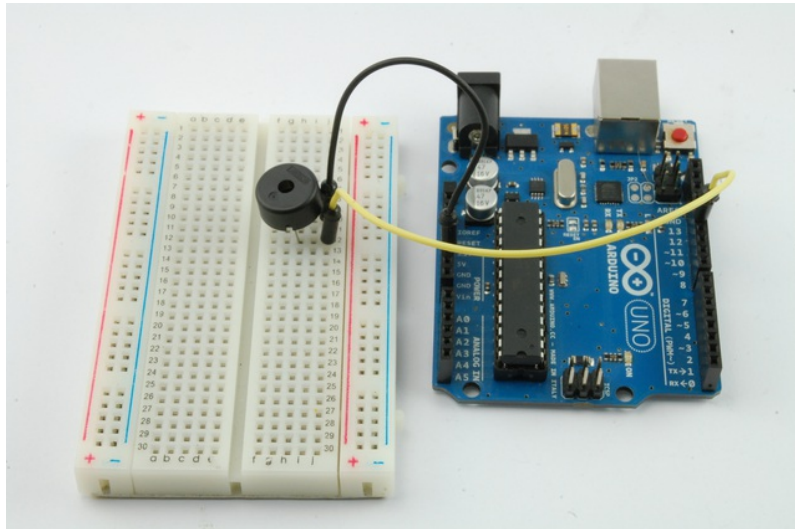
## Parts

	Part	Qty
	1 kΩ Resistor (brown, black, red stripes)	1
	Piezo sounder	1
	Photocell (Light Dependent Resistor)	1
	Half-size Breadboard	1

	<p>Arduino Uno R3</p>	<p>1</p>
	<p>Jumper wire pack</p>	<p>1</p>

## Playing a Scale

For the first part of this lesson, the only thing on the breadboard is the Piezo buzzer. One pin of the piezo sounder goes to GND connection and the other to digital pin 12.



Program your Arduino with the following sketch:

```
/*
Adafruit Arduino - Lesson 10. Simple Sounds
*/
```

```

int speakerPin = 12;

int numTones = 10;
int tones[] = {261, 277, 294, 311, 330, 349, 370, 392, 415, 440};
//          mid C  C#  D   D#  E   F   F#  G   G#  A

void setup()
{
  for (int i = 0; i < numTones; i++)
  {
    tone(speakerPin, tones[i]);
    delay(500);
  }
  noTone(speakerPin);
}

void loop()
{
}

```

To play a note of a particular pitch, you specify the frequency. See the following section on sound. The different frequencies for each note are kept in an array. An array is like a list. So, a scale can be played by playing each of the notes in the list in turn.

The 'for' loop will count from 0 to 9 using the variable 'i'. To get the frequency of the note to play at each step, we use 'tones[i]'. This means, the value in the 'tones' array at position 'i'. So, for example, 'tones[0]' is 261, 'tones[1]' is 277 etc.

The Arduino command 'tone' takes two parameters, the first is the pin to play the tone on and the second is the frequency of the tone to play.

When all the notes have been played, the 'noTone' command stops that pin playing any tone.

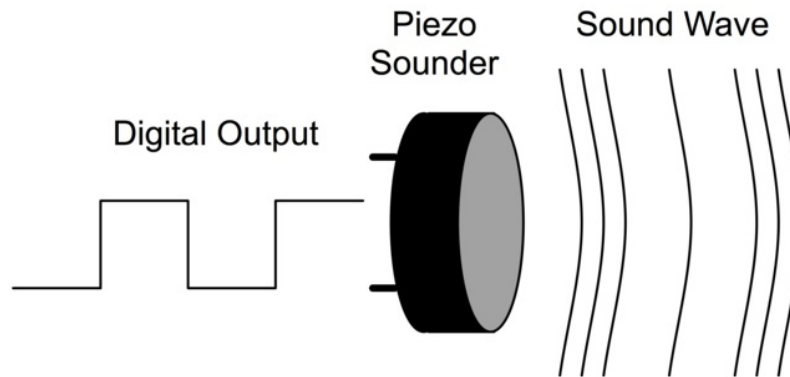
We could have put the tone playing code into 'loop' rather than 'setup', but frankly the same scale being played over and over again gets a bit tiresome. So in this case, 'loop' is empty.

To play the tune again, just press the reset button.

## Sound

---

Sound waves are vibrations in the air pressure. The speed of the vibrations (cycles per second or Hertz) is what makes the pitch of the sound. The higher the frequency of the vibration, the higher the pitch.



Middle C is usually defined as a frequency of 277 Hz. If you turn a digital output on and off again 277 times every second then that output will be middle C.

To hear the output, we need to attach something that will convert the electrical signal into sound waves. This can be done with a loudspeaker or as we have used here a piezo sounder.

Piezo sounders use a special crystal that expands and contracts as an electrical signal passes through it. This will generate a tone that we can hear.

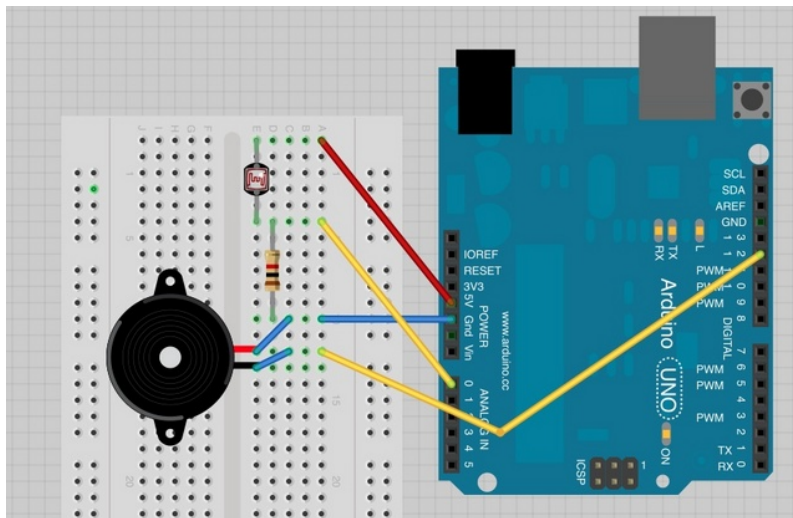


## Pseudo-Theremin

The Theramin (<http://en.wikipedia.org/wiki/Theremin> (<http://adafru.it/aTB>)) is a musical instrument that makes spooky synthesized sounds as you wave your hands in front of it. It was used in the theme music for the original Star Trek series.

We are going to make a similar instrument, albeit a lot less musical, but it will change the pitch of the note as you wave your hand in front of it.

We can leave the piezo sounder where it is, attached directly to the Arduino, but we will need the breadboard for the photocell and resistor that are going to control the pitch.



## Arduino Code

---

Upload the following code on to your Arduino board.

```
/*  
Adafruit Arduino - Lesson 10. Pseudo Thermin  
*/  
  
int speakerPin = 12;  
int photocellPin = 0;  
  
void setup()  
{  
}  
  
void loop()  
{  
  int reading = analogRead(photocellPin);  
  int pitch = 200 + reading / 4;  
  tone(speakerPin, pitch);  
}
```

The sketch is actually really straightforward. We simply take an analog reading from A0, to measure the light intensity. This value will be in the range of something like 0 to 700.

We add 200 to this raw value, to make 200 Hz the lowest frequency and simply add the reading divided by 4 to this value, to give us a range of around 200Hz to 370Hz.

## Other Things to Do

---

Try changing the value 4 in the line below to lower and higher values.

```
int pitch = 200 + reading / 4;
```



This will expand or restrict the range of frequencies.

Returning to the first sketch, try and modify it to play a tune. Hint, you can just change the values in the 'tones' array. Note that if you change the number of notes from 10 notes, then you will need to change 'numTones' accordingly.

### About the Author

Simon Monk is author of a number of books relating to Open Source Hardware. The following books written by Simon are available from Adafruit: [Programming Arduino](http://adafru.it/1019) (<http://adafru.it/1019>), [30 Arduino Projects for the Evil Genius](http://adafru.it/868) (<http://adafru.it/868>) and [Programming the Raspberry Pi](http://adafru.it/aM5) (<http://adafru.it/aM5>).