DIGITAL

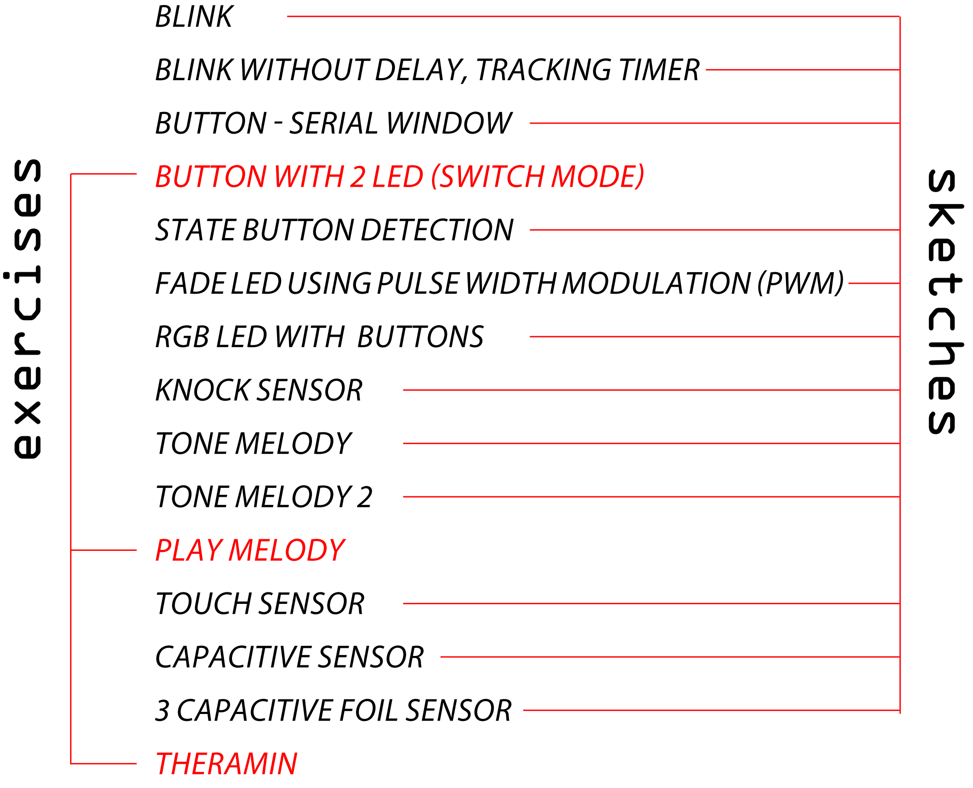
ARDUINO COOKBOOK

by Angelos Floros

CIRCUITS

physical
computing
meals
01

APPERTIZER



exercises

sketches

BLINK

BLINK WITHOUT DELAY, TRACKING TIMER

BUTTON - SERIAL WINDOW

BUTTON WITH 2 LED (SWITCH MODE)

STATE BUTTON DETECTION

FADE LED USING PULSE WIDTH MODULATION (PWM)

RGB LED WITH BUTTONS

KNOCK SENSOR

TONE MELODY

TONE MELODY 2

PLAY MELODY

TOUCH SENSOR

CAPACITIVE SENSOR

3 CAPACITIVE FOIL SENSOR

THERAMIN

Blink

http://arduino.cc/en/Tutorial/Blink

HARDWARE REQUIRED

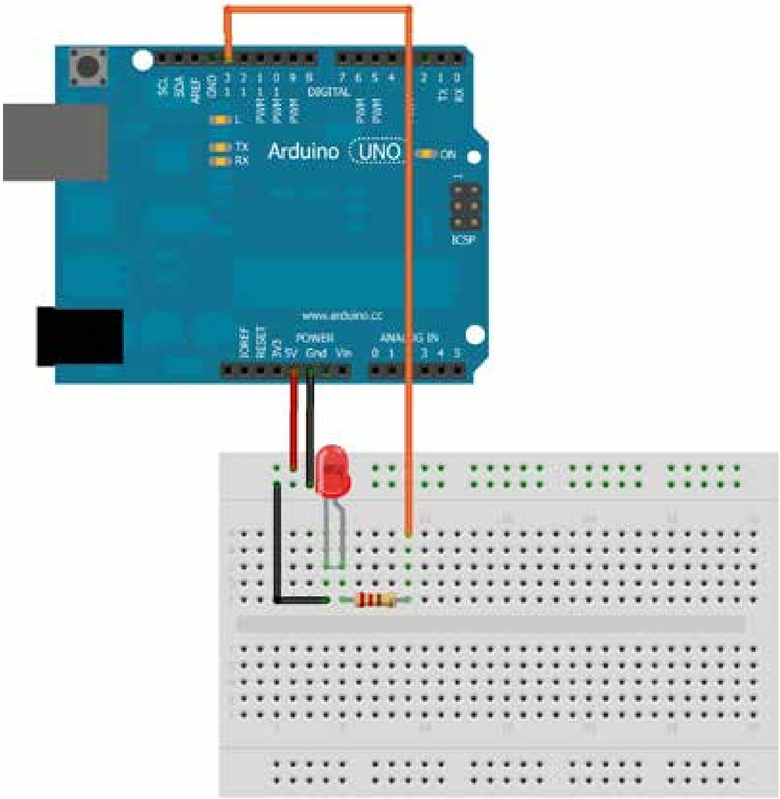
Arduino Board
1 LED
1 Resistor 220 ohm
breadboard
hook-up wires

CIRCUIT

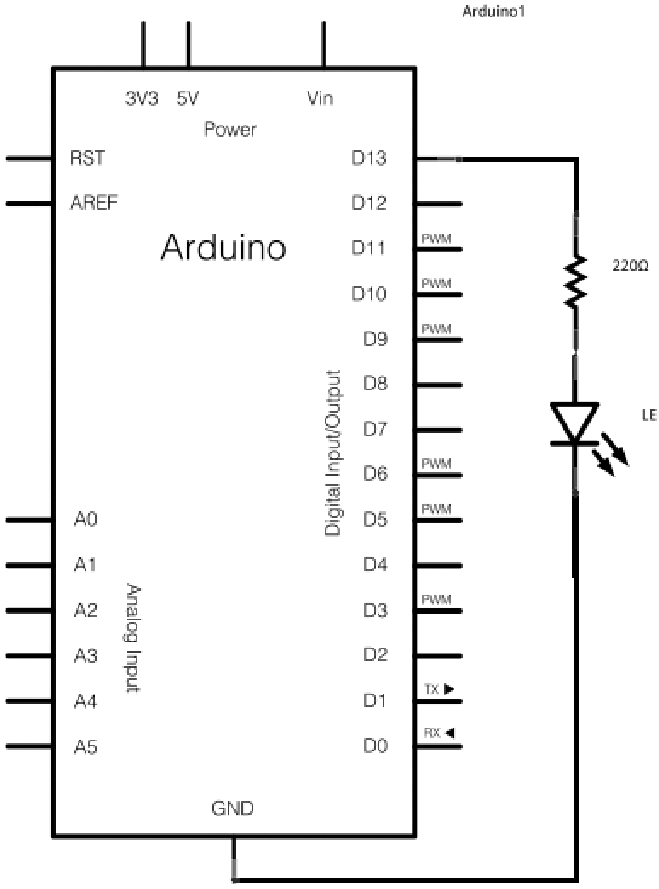
To build the circuit, attach a 220-ohm resistor to pin 13. Then attach the long leg of an LED (the positive leg, called the anode) to the resistor. Attach the short leg (the negative leg, called the cathode) to ground. Then plug your Arduino board into your computer, start the Arduino program, and enter the code below.

Most Arduino boards already have an LED attached to pin 13 on the board itself. If you run this example with no hardware attached, you should see that LED blink.

IMAGE



SCHEMATIC



CODE

```
/*
  Blink
  Turns on an LED on for one second, then off for one second, repeatedly.

  This example code is in the public domain.
 */

// Pin 13 has an LED connected on most Arduino boards.
// give it a name:
int led = 13;

// the setup routine runs once when you press reset:
void setup() {
  // initialize the digital pin as an output.
  pinMode(led, OUTPUT);
}

// the loop routine runs over and over again forever:
void loop() {

  digitalWrite(led, HIGH);   // turn the LED on (HIGH is the voltage level)
  delay(1000);               // wait for a second

 digitalWrite(led, LOW);    // turn the LED off by making the voltage LOW
 delay(1000);             // wait for a second

}
```

Blink without Delay, Tracking Timer

http://www.arduino.cc/en/Tutorial/BlinkWithoutDelay

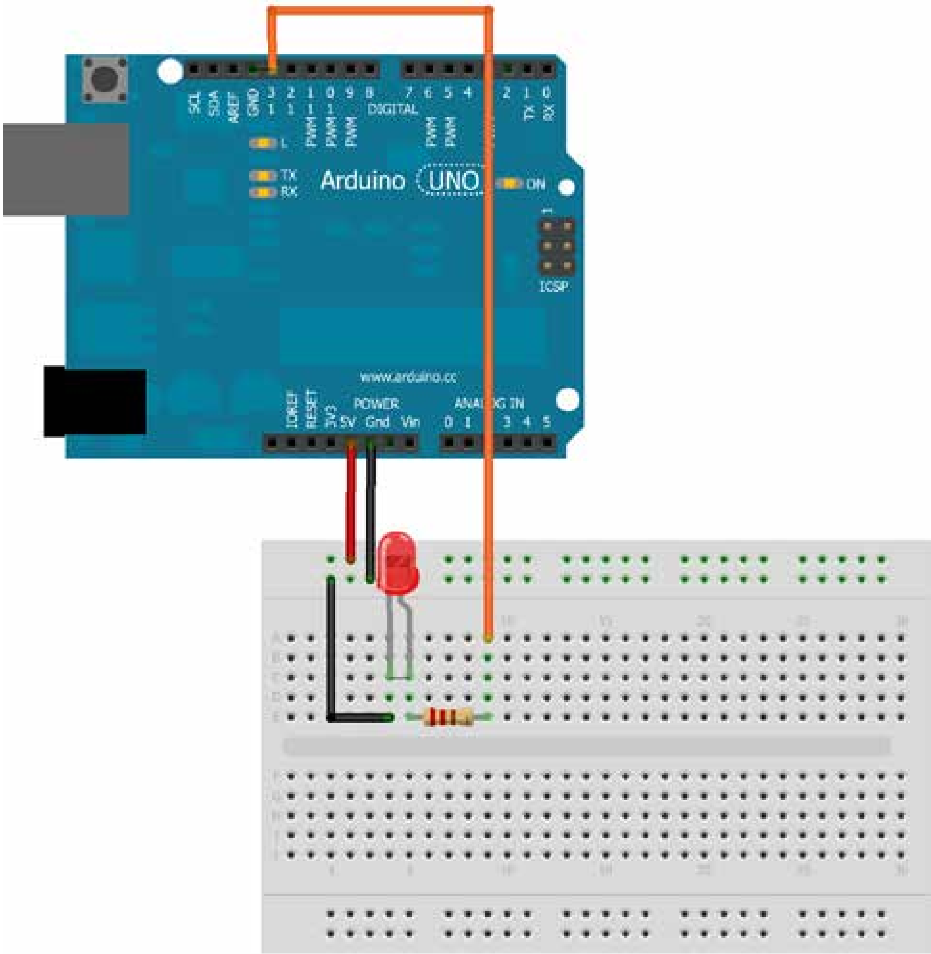
HARDWARE REQUIRED

Arduino Board
1 LED
1 Resistor 220 ohm
breadboard
hook-up wires

CIRCUIT

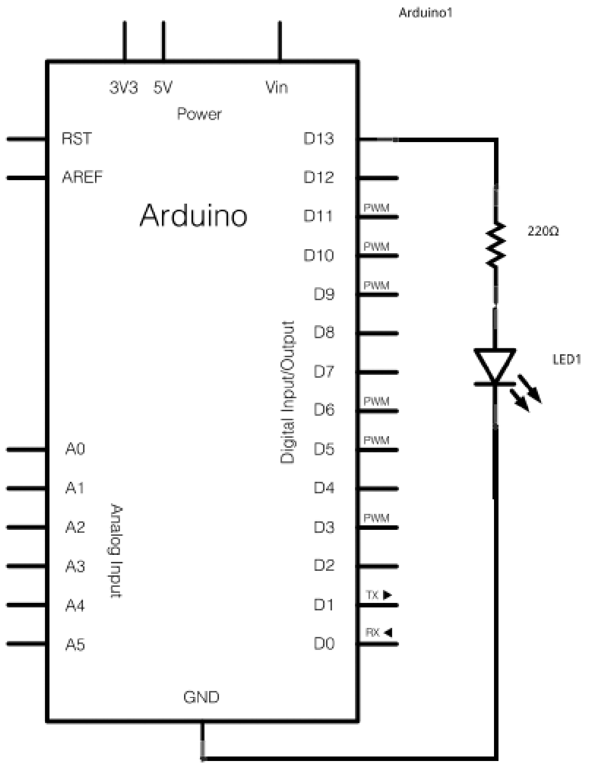
Sometimes you need to do two things at once. For example you might want to blink an LED (or some other time-sensitive function) while reading a button press or other input. In this case, you can't use delay(), or you'd stop everything else the program while the LED blinked. The program might miss the button press if it happens during the delay(). This sketch demonstrates how to blink the LED without using delay(). It keeps track of the last time the Arduino turned the LED on or off. Then, each time through loop(), it checks if a long enough interval has passed. If it has, it toggles the LED on or off.

IMAGE



ARDUINO
COOKBOOK
by Angelos Floros

SCHEMATIC



CODE

```
/* Blink without Delay

 Turns on and off a light emitting diode(LED) connected to a digital
 pin, without using the delay() function.  This means that other code
 can run at the same time without being interrupted by the LED code.

 The circuit:
 * LED attached from pin 13 to ground.
 * Note: on most Arduinos, there is already an LED on the board
 that's attached to pin 13, so no hardware is needed for this example.


 created 2005
 by David A. Mellis
 modified 8 Feb 2010
 by Paul Stoffregen

 This example code is in the public domain.


 http://www.arduino.cc/en/Tutorial/BlinkWithoutDelay
 */
```

```
// constants won't change. Used here to
// set pin numbers:
const int ledPin =  13;      // the number of the LED pin

// Variables will change:
int ledState = LOW;             // ledState used to set the LED
long previousMillis = 0;        // will store last time LED was updated

// the follow variables is a long because the time, measured in miliseconds,
// will quickly become a bigger number than can be stored in an int.
long interval = 1000;           // interval at which to blink (milliseconds)

void setup() {
  // set the digital pin as output:
  pinMode(ledPin, OUTPUT);
}

void loop()
{
  // here is where you'd put code that needs to be running all the time.

  // check to see if it's time to blink the LED; that is, if the
  // difference between the current time and last time you blinked
  // the LED is bigger than the interval at which you want to
  // blink the LED.
  unsigned long currentMillis = millis();

  if(currentMillis - previousMillis > interval) {
    // save the last time you blinked the LED
    previousMillis = currentMillis;

    // if the LED is off turn it on and vice-versa:
    if (ledState == LOW)
      ledState = HIGH;
    else
      ledState = LOW;

    // set the LED with the ledState of the variable:
    digitalWrite(ledPin, ledState);
  }
}
```

Button / Serial Window

http://www.arduino.cc/en/Tutorial/Button

HARDWARE REQUIRED

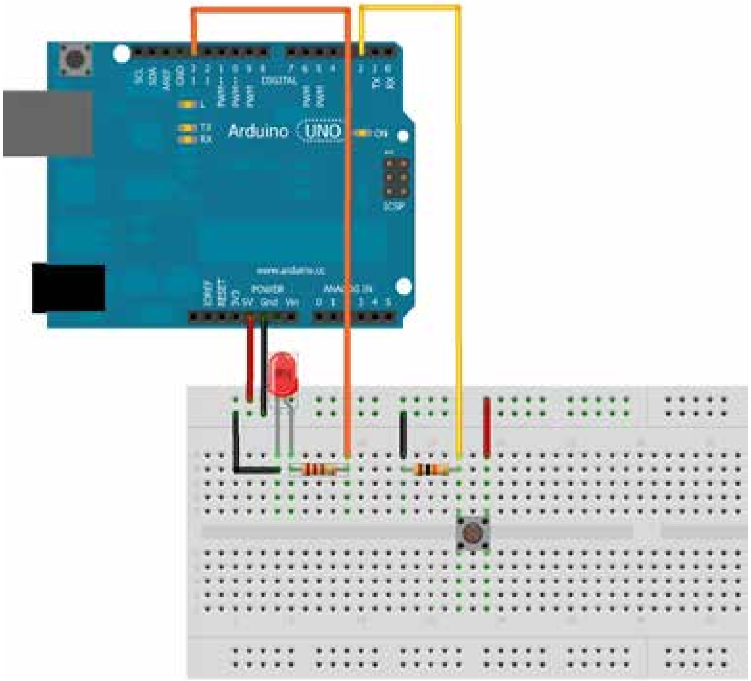
Arduino Board
1 momentary button or switch
1 10K ohm resistor
1 LED
220ohm Resistor
breadboard
hook-up wires

CIRCUIT

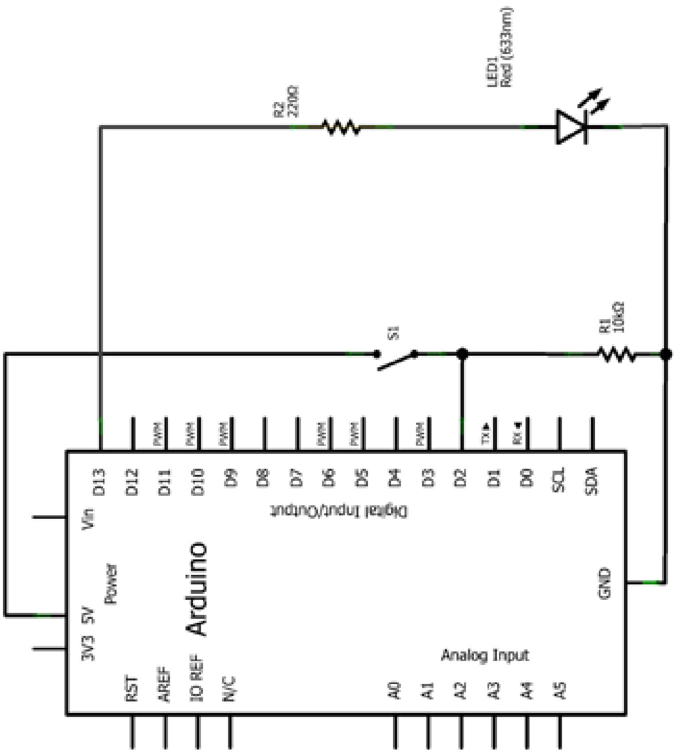
Connect three wires to the Arduino board. The first two, red and black, connect to the two long vertical rows on the side of the breadboard to provide access to the 5 volt supply and ground. The third wire goes from digital pin 2 to one leg of the pushbutton. That same leg of the button connects through a pull-down resistor (here 10 KOhms) to ground. The other leg of the button connects to the 5 volt supply.

When the pushbutton is open (unpressed) there is no connection between the two legs of the pushbutton, so the pin is connected to ground (through the pull-down resistor) and we read a LOW. When the button is closed (pressed), it makes a connection between its two legs, connecting the pin to 5 volts, so that we read a HIGH.
You can also wire this circuit the opposite way, with a pullup resistor keeping the input HIGH, and going LOW when the button is pressed. If so, the behavior of the sketch will be reversed, with the LED normally on and turning off when you press the button.
If you disconnect the digital i/o pin from everything, the LED may blink erratically. This is because the input is «floating» - that is, it will randomly return either HIGH or LOW. That's why you need a pull-up or pull-down resistor in the circuit.

IMAGE



SCHEMATIC



CODE

```
/*
  Button
*/

// constants won’t change. They’re used here to
// set pin numbers:
const int buttonPin = 2;     // the number of the pushbutton pin
const int ledPin =  13;      // the number of the LED pin

// variables will change:
int buttonState = 0;         // variable for reading the pushbutton status

void setup() {
  // initialize serial communication:
  Serial.begin(9600);
  // initialize the LED pin as an output:
  pinMode(ledPin, OUTPUT);
  // initialize the pushbutton pin as an input:
  pinMode(buttonPin, INPUT);
}

void loop(){
  // read the state of the pushbutton value:
  buttonState = digitalRead(buttonPin);

  // check if the pushbutton is pressed.
  // if it is, the buttonState is HIGH:

  if (buttonState == HIGH) {

    // turn LED on:
    digitalWrite(ledPin, HIGH);
    // Print to Seiral
    Serial.println(“HIGH”);
   }
  else {
    // turn LED off:
    digitalWrite(ledPin, LOW);
    // Print to Seiral
    Serial.println(“LOW”);
  }
}
```

Turns on and off a light emitting diode(LED) connected to digital pin 13, when pressing a pushbutton attached to pin 2.

The circuit:
* LED attached from pin 13 to ground
* pushbutton attached to pin 2 from +5V
* 10K resistor attached to pin 2 from ground
* 220Ω resistor attached to pin 13 from LED anode

* Note: on most Arduinos there is already an LED on the board attached to pin 13.

created 2005
by DojoDave <http://www.0j0.org>
modified 30 Aug 2011
by Tom Igoe
modified by Angelos Floros

This example code is in the public domain.
http://www.arduino.cc/en/Tutorial/Button

Button with 2 LEDs (Switch Mode)

EXERCISE

Try to turn on and off two LEDs using the state of one button.

HARDWARE REQUIRED

Arduino Board
momentary button or switch
10K ohm resistor
2 LEDs
2 220Ω Resistors
breadboard
hook-up wires

SCHEMATIC

State Button Detection

http://arduino.cc/en/Tutorial/ButtonStateChange

HARDWARE REQUIRED

Arduino Board
1 momentary button or switch
1 10K ohm resistor
1 LED
1 220ohm Resistor
breadboard
hook-up wires

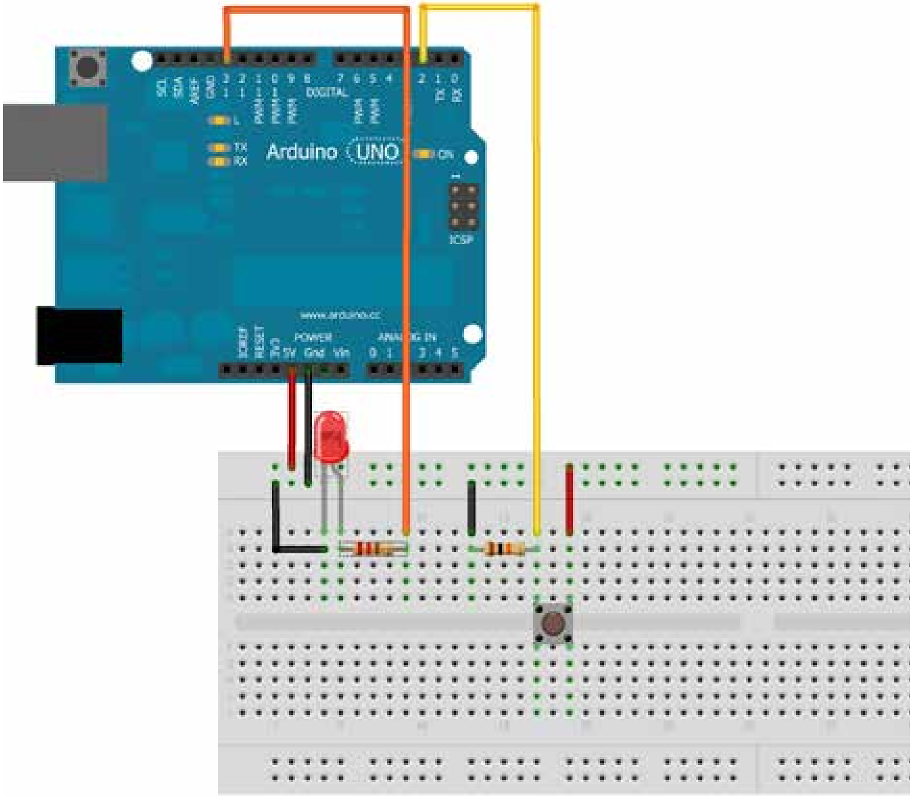
CIRCUIT

Once you've got a pushbutton working, you often want to do some action based on how many times the button is pushed. To do this, you need to know when the button changes state from off to on, and count how many times this change of state happens. This is called state change detection or edge detection.

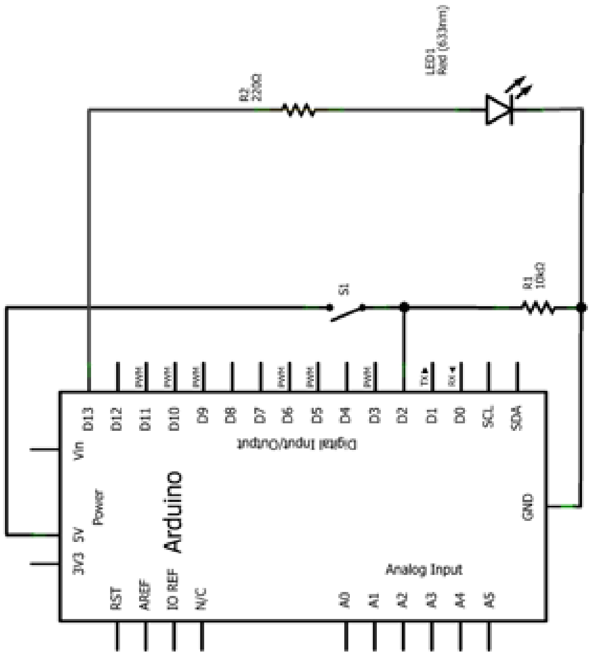
When the pushbutton is open (unpressed) there is no connection between the two legs of the pushbutton, so the pin is connected to ground (through the pull-down resistor) and we read a LOW. When the button is closed (pressed), it makes a connection between its two legs, connecting the pin to voltage, so that we read a HIGH. (The pin is still connected to ground, but the resistor resists the flow of current, so the path of least resistance is to +5V.)

If you disconnect the digital i/o pin from everything, the LED may blink erratically. This is because the input is «floating» - that is, not connected to either voltage or ground. It will more or less randomly return either HIGH or LOW. That's why you need a pull-down resistor in the circuit.

IMAGE



SCHEMATIC



CODE

```
/*
  State change detection (edge detection)

 Often, you don’t need to know the state of a digital input all the time,
 but you just need to know when the input changes from one state to another.
 For example, you want to know when a button goes from OFF to ON.  This is called
 state change detection, or edge detection.

 This example shows how to detect when a button or button changes from off to on
 and on to off.

 The circuit:
 * pushbutton attached to pin 2 from +5V
 * 10K resistor attached to pin 2 from ground
 * LED attached from pin 13 to ground (or use the built-in LED on
   most Arduino boards)

 created  27 Sep 2005
 modified 30 Aug 2011
 by Tom Igoe

This example code is in the public domain.

 http://arduino.cc/en/Tutorial/ButtonStateChange

 */
```

```
// this constant won’t change:
const int  buttonPin = 2;    // the pin that the pushbutton is attached to
const int ledPin = 13;       // the pin that the LED is attached to

// Variables will change:
int buttonPushCounter = 0;   // counter for the number of button presses
int buttonState = 0;         // current state of the button
int lastButtonState = 0;     // previous state of the button

void setup() {
  // initialize the button pin as a input:
  pinMode(buttonPin, INPUT);
  // initialize the LED as an output:
  pinMode(ledPin, OUTPUT);
  // initialize serial communication:
  Serial.begin(9600);
}

void loop() {
  // read the pushbutton input pin:
  buttonState = digitalRead(buttonPin);

  // compare the buttonState to its previous state
  if (buttonState != lastButtonState) {
    // if the state has changed, increment the counter
    if (buttonState == HIGH) {
      // if the current state is HIGH then the button wend from off to on:
      buttonPushCounter++;
      Serial.println(“on”);
      Serial.print(“number of button pushes:  “);
      Serial.println(buttonPushCounter);
    } else {
      // if the current state is LOW then the button wend from on to off:
      Serial.println(“off”);
    }
  }
  // turns on the LED every four button pushes by
  // checking the modulo of the button push counter.
  // the modulo function gives you the remainder of the division of two numbers:
  if (buttonPushCounter % 4 == 0) {
    digitalWrite(ledPin, HIGH);
  } else {
   digitalWrite(ledPin, LOW);
  }
 // save the current state as the last state, for next time through the loop
 lastButtonState = buttonState;
}
```

Fade LED using Pulse Width Modulation (PWM)

http://arduino.cc/en/Tutorial/Fade

Connect the anode (the longer, positive leg) of your LED to digital output pin 9 on your Arduino through a 220-ohm resistor. Connect the cathode (the shorter, negative leg) directly to ground.

This projects uses Pulse Width Modulation (PWM) pins.
Arduino Uno Rev3 PWM pins: 3, 5, 6, 9, 10, 11

http://arduino.cc/en/Tutorial/PWM

HARDWARE REQUIRED

Arduino board
1 LED
1 220 ohm resistor
breadboard
hook-up wires

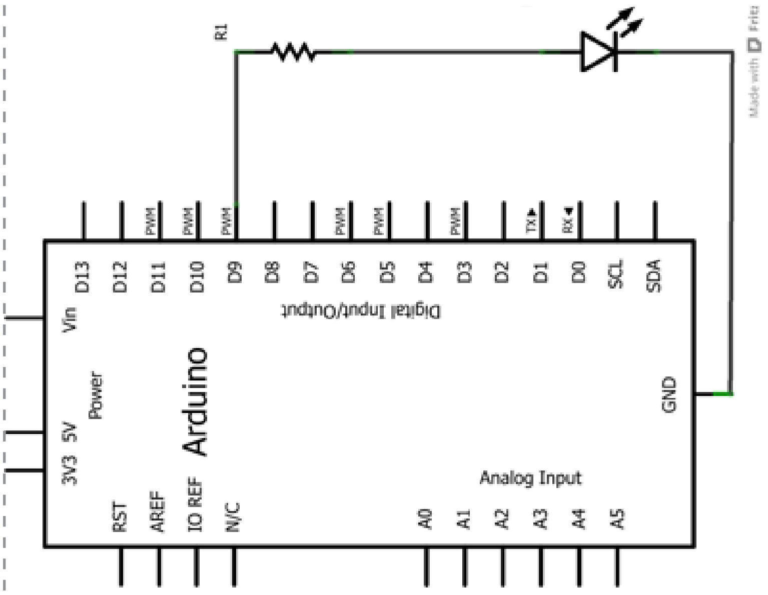
CIRCUIT

After declaring pin 9 to be your ledPin, there is nothing to do in the setup() function of your code.

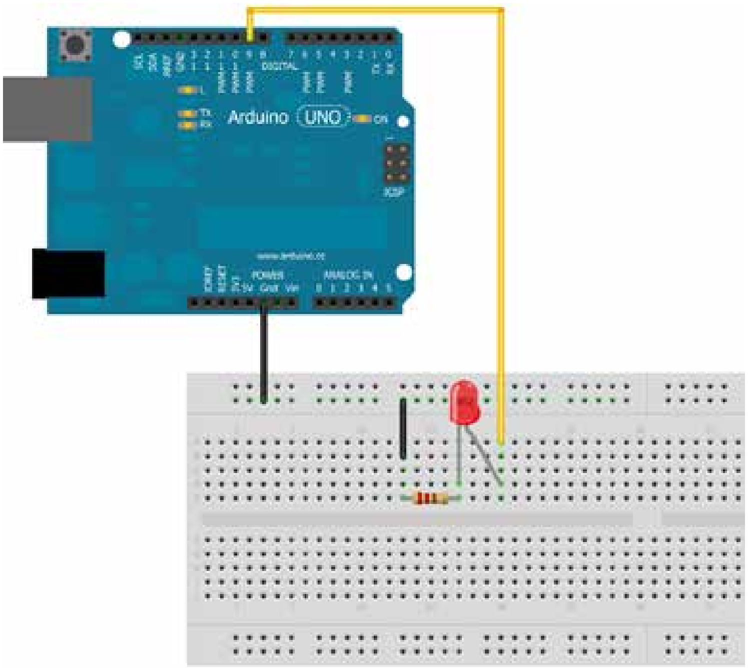
The analogWrite() function that you will be using in the main loop of your code requires two arguments: One telling the function which pin to write to, and one indicating what PWM value to write. In order to fade your LED off and on, gradually increase your PWM value from 0 (all the way off) to 255 (all the way on), and then back to 0 once again to complete the cycle. In the sketch below, the PWM value is set using a variable called brightness. Each time through the loop, it increases by the value of the variable fadeAmount. If brightness is at either extreme of its value (either 0 or 255), then fadeAmount is changed to its negative. In other words, if fadeAmount is 5, then it is set to -5. If it's 55, then it's set to 5. The next time through the loop, this change causes brightness to change direction as well.

analogWrite() can change the PWM value very fast, so the delay at the end of the sketch controls the speed of the fade. Try changing the value of the delay and see how it changes the program.

SCHEMATIC



IMAGE



CODE

```
int led = 9;           // the pin that the LED is attached to
int brightness = 0;    // how bright the LED is
int fadeAmount = 5;    // how many points to fade the LED by

// the setup routine runs once when you press reset:
// it prints on Serial Window the brightness value
// it requires to open the Serial Window from Arduino Software

void setup()  {
  // declare pin 9 to be an output:
  pinMode(led, OUTPUT);
  Serial.begin(9600);
}

// the loop routine runs over and over again forever:
void loop()  {
  // set the brightness of pin 9:
  analogWrite(led, brightness);

  // change the brightness for next time through the loop:
  brightness = brightness + fadeAmount;

  // reverse the direction of the fading at the ends of the fade:
  if (brightness == 0 || brightness == 255) {
    fadeAmount = -fadeAmount ;
  }

  Serial.println(brightness, DEC);
  // wait for 30 milliseconds to see the dimming effect
  delay(30);
}
```

```
/*
 Fade

 This example shows how to fade an LED on pin 9
 using the analogWrite() function.

 This example code is in the public domain.
 */
```

RGB LED with Buttons

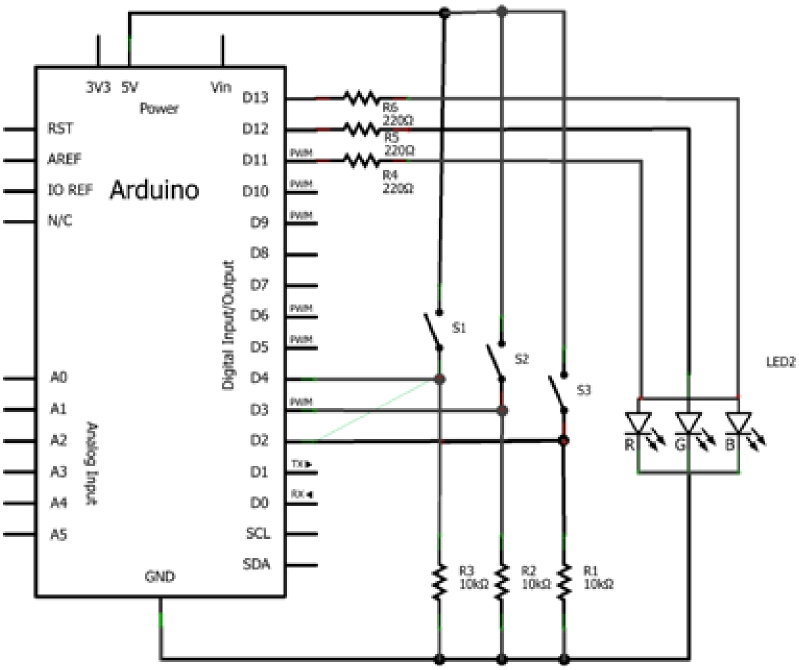
HARDWARE REQUIRED

Arduino board
1 RGB LED
3 buttons
3 220 ohm resistors
3 10KΩ resistors
breadboard
hook-up wires

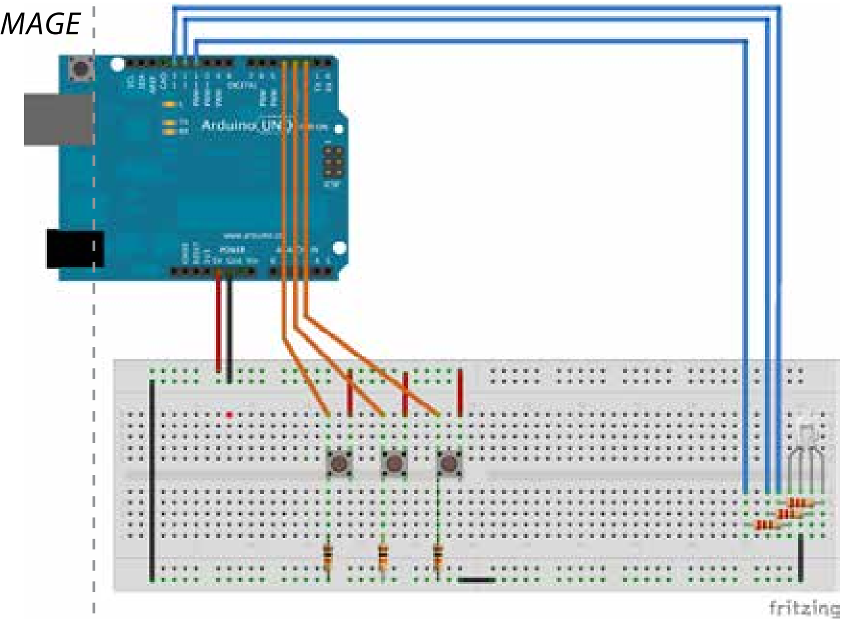
CIRCUIT

With this example you can mix the Red, Green and Blue lights using and RGB LED and three buttons.

SCHEMATIC



IMAGE



CODE

```
/*
  RGB LED with Buttons

 Use three buttons to control RBG LED color
 Turns on and off a light emitting diode(LED)
connected to digital   pin 13, when pressing a
pushbutton attached to pin 2.

 The circuit:
 * RED light attached to pin 13
 * GREEN light attached to pin 12
 * BLUE light attached to pin 11
 * The cathode of the RBG LED is attached to
ground
 * RED pushbutton attached to pin 4 from +5V
 * GREEN pushbutton attached to pin 3 from +5V
 * BLUE pushbutton attached to pin 2 from +5V
 * 10K resistor attached to pin 2 from ground
 * 10K resistor attached to pin 3 from ground
 * 10K resistor attached to pin 4 from ground
 * 220Ω resistor attached to pin 13 from Red pin
 * 220Ω resistor attached to pin 12 from Green pin
 * 220Ω resistor attached to pin 11 from Blue pin
 * Note: on most Arduinos there is already an LED
on the board  attached to pin 13.

 created 2013  by Angelos Floros

  */
```

```
// constants won't change. They're used here to
// set pin numbers:

const int buttonPinR = 2;     // the number of the RED pushbutton pin
const int buttonPinG = 3;     // the number of the GREEN pushbutton pin
const int buttonPinB = 4;     // the number of the BLUE pushbutton pin
const int ledPinR =  13;    // the number of the RED light of RGB LED pin
const int ledPinG =  12;  // the number of the GREEN light of RGB LED pin
const int ledPinB =  11;    // the number of the BLUE light of RGB LED pin

// variables will change:
int buttonStateR = 0;         // variable for reading the pushbutton status
int buttonStateG = 0;         // variable for reading the pushbutton status
int buttonStateB = 0;         // variable for reading the pushbutton status

void setup() {
  // initialize the LED pin as an output:
  pinMode(ledPinR, OUTPUT);
  pinMode(ledPinG, OUTPUT);
  pinMode(ledPinB, OUTPUT);

  // initialize the pushbutton pin as an input:
  pinMode(buttonPinR, INPUT);
  pinMode(buttonPinG, INPUT);
  pinMode(buttonPinB, INPUT);
}

void loop(){
  // read the state of the pushbutton value:
  buttonStateR = digitalRead(buttonPinR);
  buttonStateG = digitalRead(buttonPinG);
  buttonStateB = digitalRead(buttonPinB);

  // check if the pushbutton is pressed.
  // if it is, the buttonState is HIGH:
  if (buttonStateR == HIGH) {
    // turn RED light on:
    digitalWrite(ledPinR, HIGH);
  }
  else {
    // turn RED light off:
    digitalWrite(ledPinR, LOW);
  }
   if (buttonStateG == HIGH) {
    // turn GREEN light on:
    digitalWrite(ledPinG, HIGH);
  }
  else {
    // turn GREEN light off:
    digitalWrite(ledPinG, LOW);
  }
    if (buttonStateB == HIGH) {
    // turn BLUE light on:
    digitalWrite(ledPinB, HIGH);
  }
  else {
    // turn BLUE light off:
    digitalWrite(ledPinB, LOW);
  }
}
```

Knock Sensor

http://arduino.cc/en/Tutorial/KnockSensor

This tutorial shows you how to use a Piezo element to detect vibration, in this case, a knock on a door, table, or other solid surface.
A piezo is an electronic device that generates a voltage when it's physically deformed by a vibration, sound wave, or mechanical strain. Similarly, when you put a voltage across a piezo, it vibrates and creates a tone. Piezos can be used both to play tones and to detect tones.
The sketch reads the piezos output using the analogRead() command, encoding the voltage range from 0 to 5 volts to a numerical range from 0 to 1023 in a process referred to as analog-to-digital conversion, or ADC.
If the sensors output is stronger than a certain threshold, your Arduino will send the string "Knock!" to the computer over the serial port.

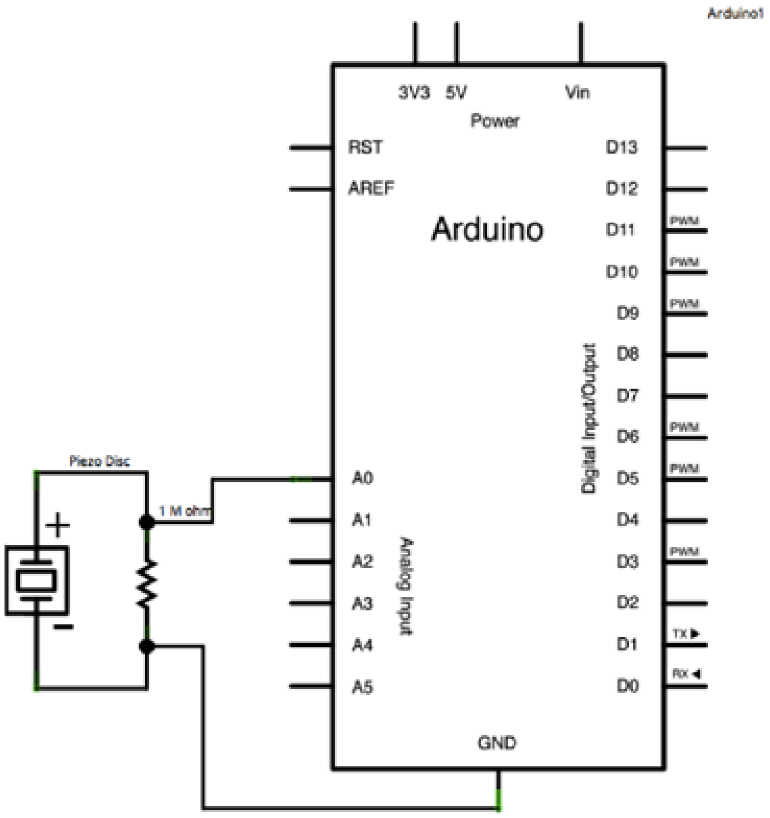
HARDWARE REQUIRED

Arduino Board
1 Piezo electric disc
1 Megohm resistor
breadboard
hook-up wires

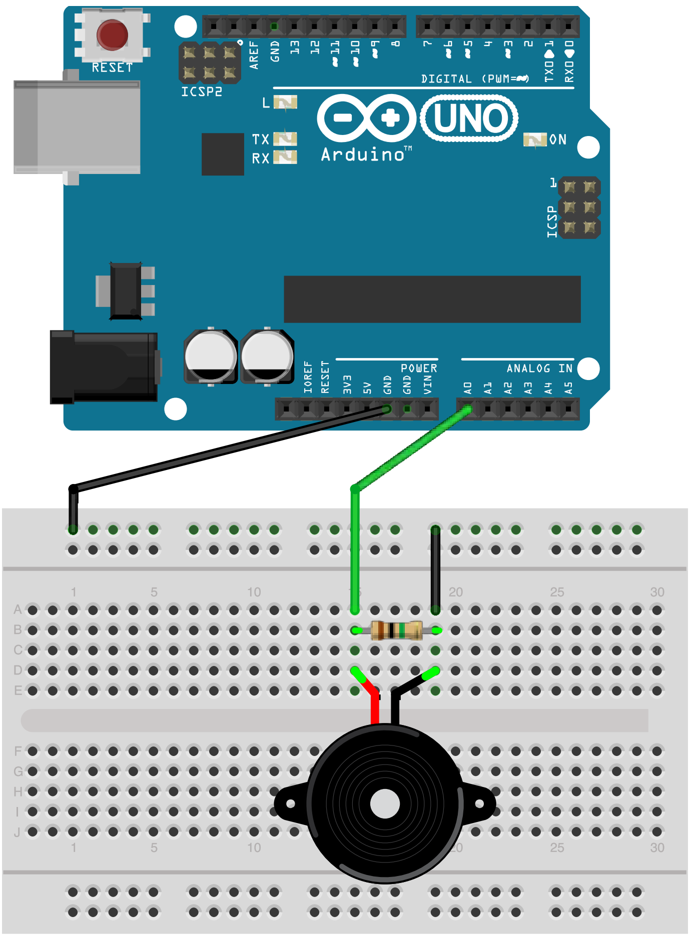
CIRCUIT

Piezos are polarized, meaning that voltage passes through them (or out of them) in a specific direction. Connect the black wire (the lower voltage) to ground and the red wire (the higher voltage) to analog pin 0. Additionally, connect a 1-megohm resistor in parallel to the Piezo element to limit the voltage and current produced by the piezo and to protect the analog input.
It is possible to acquire piezo elements without a plastic housing. These will look like a metallic disc, and are easier to use as input sensors. Plezo sensors work best when firmly pressed against, taped, or glued their sensing surface.

SCHEMATIC



IMAGE



CODE

```
/* Knock Sensor

  This sketch reads a piezo element to detect a knocking sound.
  It reads an analog pin and compares the result to a set threshold.
  If the result is greater than the threshold, it writes
  «knock» to the serial port, and toggles the LED on pin 13.

  The circuit:
   * + connection of the piezo attached to analog in 0
   * - connection of the piezo attached to ground
   * 1-megohm resistor attached from analog in 0 to ground

  http://www.arduino.cc/en/Tutorial/Knock

  created 25 Mar 2007
  by David Cuartielles <http://www.0j0.org>
  modified 30 Aug 2011
  by Tom Igoe

  This example code is in the public domain.

 */
```

```
// these constants won’t change:
const int ledPin = 13;      // ONBOARD LED (digital pin 13)
const int knockSensor = A0; // the piezo is connected to analog pin 0
const int threshold = 100;  // threshold value to decide when the detected
sound is a knock or not


// these variables will change:
int sensorReading = 0;    // variable to store the value read from the sensor pin
int ledState = LOW;       // variable used to store the last LED status, to toggle the light

void setup() {
 pinMode(ledPin, OUTPUT); // declare the ledPin as as OUTPUT
 Serial.begin(9600);      // use the serial port
}

void loop() {
  // read the sensor and store it in the variable sensorReading:
  sensorReading = analogRead(knockSensor);

  // if the sensor reading is greater than the threshold:
  if (sensorReading >= threshold) {
    // toggle the status of the ledPin:
    ledState = !ledState;
    // update the LED pin itself:
    digitalWrite(ledPin, ledState);
    // send the string “Knock!” back to the computer, followed by newline
    Serial.println(“Knock!”);
  }
  delay(100);  // delay to avoid overloading the serial port buffer
}
```

Tone Melody

http://arduino.cc/en/Tutorial/Tone

This example shows how to use the tone() command to generate notes. It plays a little melody you may have heard before.

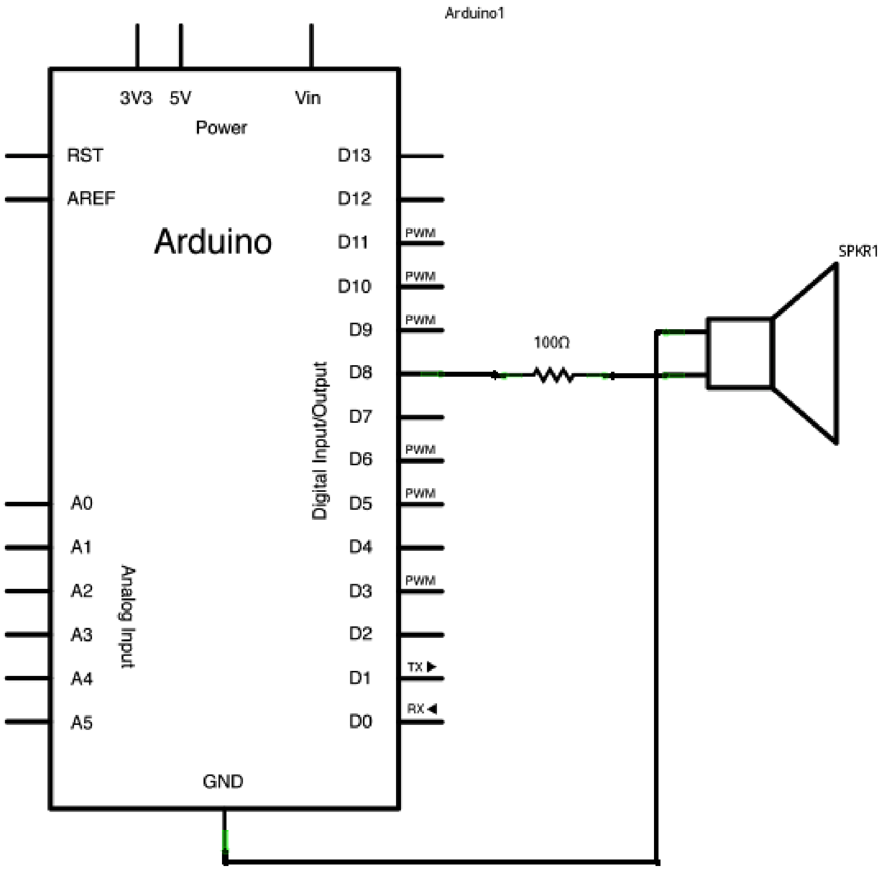
HARDWARE REQUIRED

Arduino Board
1 Piezo Buzzer
1 100 ohm resistor
breadboard
hook-up wires

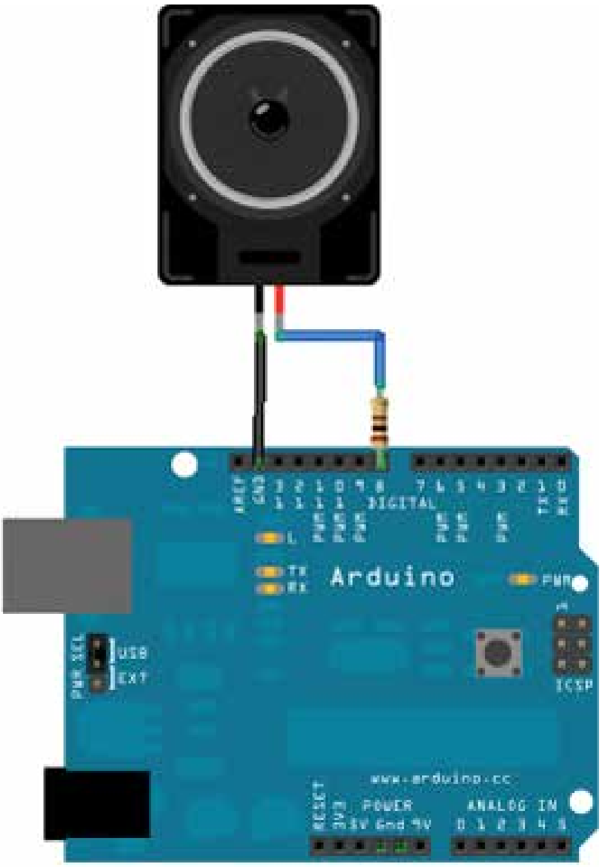
CIRCUIT

The code below uses an extra file, pitches.h. This file contains all the pitch values for typical notes. For example, NOTE_C4 is middle C. NOTE_FS4 is F sharp, and so forth. This note table was originally written by Brett Hagman, on whose work the tone() command was based. You may find it useful for whenever you want to make musical notes.

SCHEMATIC



IMAGE



CODE

```
/*
  Melody

 Plays a melody

 circuit:
 * 8-ohm speaker on digital pin 8

 created 21 Jan 2010
 modified 30 Aug 2011
 by Tom Igoe

This example code is in the public domain.

 http://arduino.cc/en/Tutorial/Tone

 */
```

```
// notes in the melody:
int melody[] = {
  NOTE_C4, NOTE_G3,NOTE_G3, NOTE_A3, NOTE_G3,0, NOTE_B3,
NOTE_C4};

// note durations: 4 = quarter note, 8 = eighth note, etc.:
int noteDurations[] = {
  4, 8, 8, 4,4,4,4,4 };

void setup() {
  // iterate over the notes of the melody:
  for (int thisNote = 0; thisNote < 8; thisNote++) {

    // to calculate the note duration, take one second
    // divided by the note type.
    //e.g. quarter note = 1000 / 4, eighth note = 1000/8, etc.
    int noteDuration = 1000/noteDurations[thisNote];

   // use pin8 to output sound
    tone(8, melody[thisNote],noteDuration);

    // to distinguish the notes, set a minimum time between them.
    // the note's duration + 30% seems to work well:
    int pauseBetweenNotes = noteDuration * 1.30;
    delay(pauseBetweenNotes);
    // stop the tone playing:
    noTone(8);
  }
}

void loop() {
  // no need to repeat the melody.
}
```

Tone Melody 2

http://arduino.cc/en/Tutorial/Melody

This example uses a piezo speaker to play melodies. It sends a square wave of the appropriate frequency to the piezo, generating the corresponding tone.
The calculation of the tones is made following the mathematical operation:

timeHigh = period / 2 = 1 / (2 * toneFrequency)

where the different tones are described as in the table:

note	frequency	period	timeHigh
c	261 Hz	3830	1915
d	294 Hz	3400	1700
e	329 Hz	3038	1519
f	349 Hz	2864	1432
g	392 Hz	2550	1275
a	440 Hz	2272	1136
b	493 Hz	2028	1014
C	523 Hz	1912	956

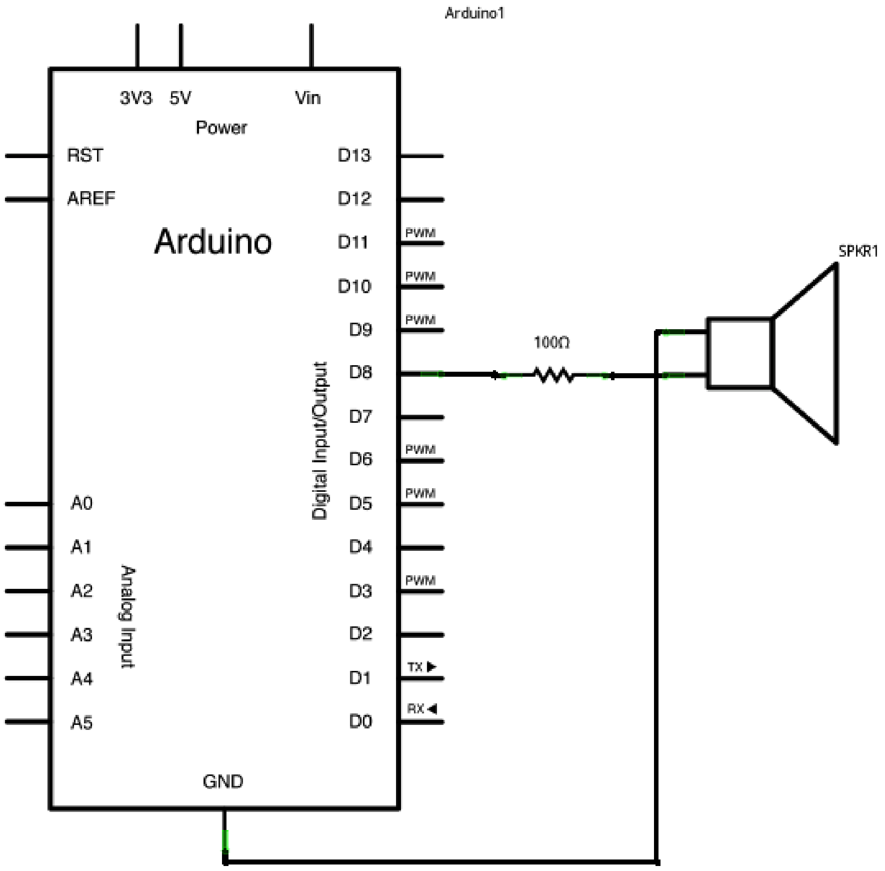
HARDWARE REQUIRED

Arduino Board
Breadboard
1 Piezo Buzzer
1 100 ohm resistor

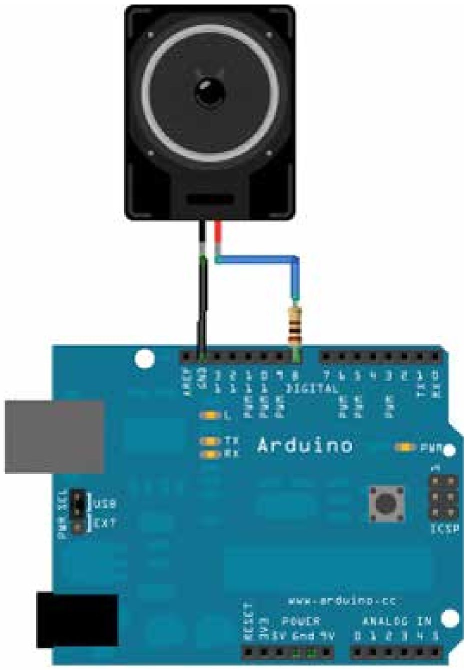
CIRCUIT

Piezos have polarity. Commercial devices are usually have a red (positive) and a black (negative). Connect the red wire digital pin 9 and the black wire to ground. Sometimes it is possible to acquire Piezo elements without a plastic housing, then they will just look like a metallic disc.

SCHEMATIC



IMAGE



CODE

```
int speakerPin = 8;

int length = 15; // the number of notes
char notes[] = “ccggaagffeeddc “; // a space represents a rest
int beats[] = { 1, 1, 1, 1, 1, 1, 2, 1, 1, 1, 1, 1, 1, 2, 4 };
int tempo = 300;

void playTone(int tone, int duration) {
  for (long i = 0; i < duration * 1000L; i += tone * 2) {
    digitalWrite(speakerPin, HIGH);
    delayMicroseconds(tone);
    digitalWrite(speakerPin, LOW);
    delayMicroseconds(tone);
  }
}

void playNote(char note, int duration) {
  char names[] = { ‘c’, ‘d’, ‘e’, ‘f’, ‘g’, ‘a’, ‘b’, ‘C’ };
  int tones[] = { 1915, 1700, 1519, 1432, 1275, 1136, 1014, 956
};

  // play the tone corresponding to the note name
  for (int i = 0; i < 8; i++) {
    if (names[i] == note) {
      playTone(tones[i], duration);
    }
  }
}

void setup() {
  pinMode(speakerPin, OUTPUT);
}

void loop() {
  for (int i = 0; i < length; i++) {
    if (notes[i] == ‘ ‘) {
      delay(beats[i] * tempo); // rest
    } else {
      playNote(notes[i], beats[i] * tempo);
    }

    // pause between notes
    delay(tempo / 2);
  }
}
```

```
/*
 Melody

 Plays a melody

 circuit:
 * 8-ohm speaker on digital pin 8

 created 21 Jan 2010
 modified 30 Aug 2011
 by Tom Igoe

This example code is in the public domain.

 http://arduino.cc/en/Tutorial/Tone

 */
```

Play Melody

EXERCISE

Play melody using buttons and 2 buzzers.

HARDWARE REQUIRED

Arduino Board
momentary buttons or switch
10K ohm resistors
LEDs (optional)
220Ω Resistors (optional)
buzzers
breadboard
hook-up wire

SCHEMATIC

Touch Sensor

http://playground.arduino.cc/Main/CapacitiveSensor

The capacitiveSensor library turns two or more Arduino pins into a capacitive sensor, which can sense the electrical capacitance of the human body. All the sensor setup requires is a medium to high value resistor and a piece of wire and a small (to large) piece of aluminum foil on the end. At its most sensitive, the sensor will start to sense a hand or body inches away from the sensor.

Version 04 adds support for Arduino 1.0, and fixes an obscure possible race condition with Tone, Servo and other libraries that perform I/O in interrupt context.

Version 03 has been updated to C++ and supports multiple inputs. It also includes some utility functions to make it convenient to change timeout values.

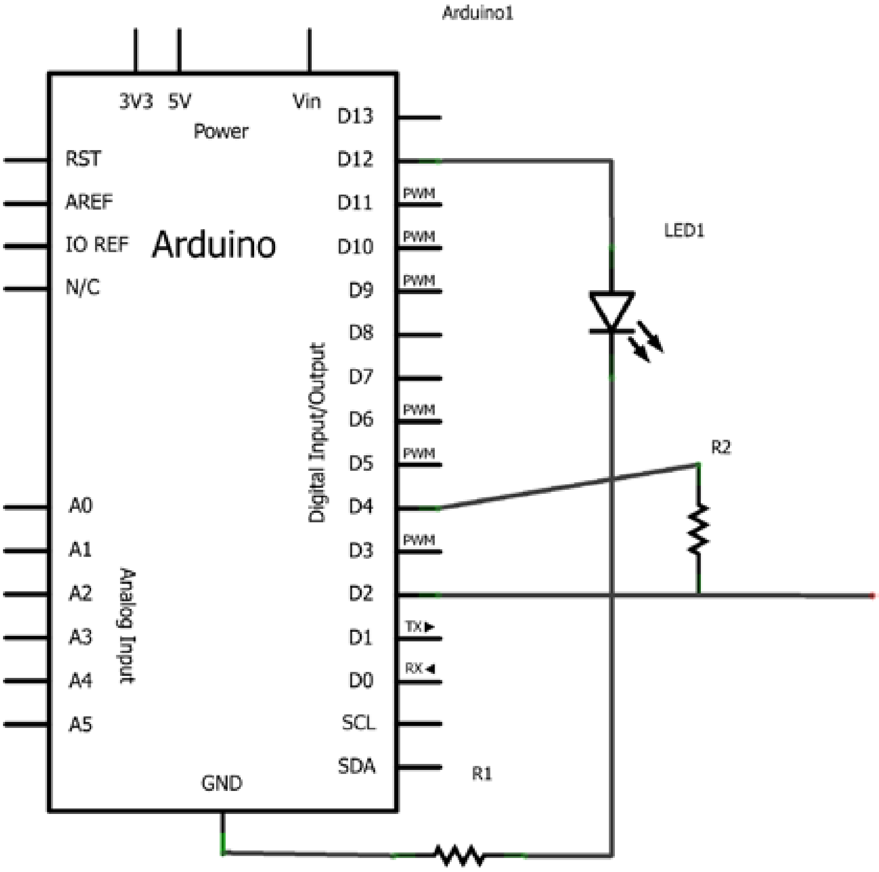
HARDWARE REQUIRED

Arduino Board
1 LED
1 470 Ωhm Resistor
1 1MΩhm Resistor
breadboard
hook-up wire

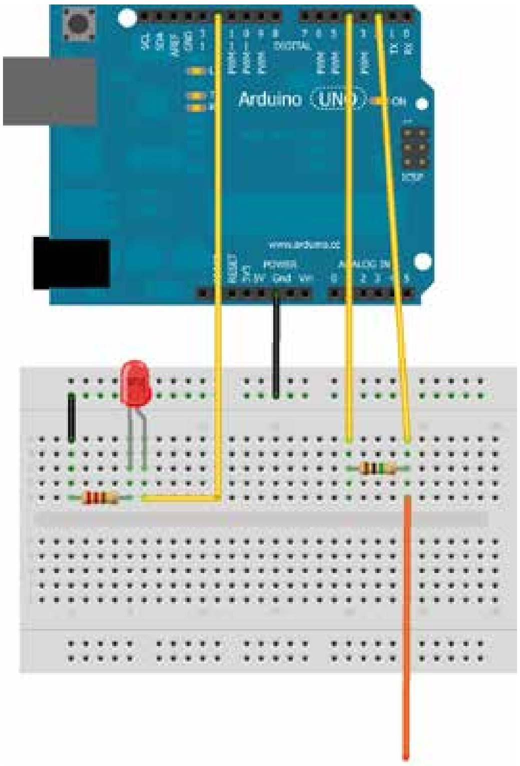
CIRCUIT

Here are some guidelines for resistors but be sure to experiment for a desired response. Use a 1 megohm resistor (or less maybe) for absolute touch to activate.

SCHEMATIC



IMAGE



CODE

```
// import the library (must be located in the
// Arduino/libraries directory)
#include <CapacitiveSensor.h>

// create an instance of the library
// pin 4 sends electrical energy
// pin 2 senses senses a change
CapacitiveSensor capSensor = CapacitiveSensor(4,2);

// threshold for turning the lamp on
int threshold = 1000;

// pin the LED is connected to
const int ledPin = 12;


void setup() {
  // open a serial connection
  Serial.begin(9600);
  // set the LED pin as an output
  pinMode(ledPin, OUTPUT);
}

void loop() {
  // store the value reported by the sensor in a variable
  int sensorValue = capSensor.capacitiveSensor(30);

  // print out the sensor value
  Serial.println(sensorValue);

  // if the value is greater than the threshold
  if(sensorValue > threshold) {
    // turn the LED on
    digitalWrite(ledPin, HIGH);
  }
  // if it's lower than the threshold
  else {
    // turn the LED off
    digitalWrite(ledPin, LOW);
  }

  delay(10);
}
```

```
/*
  Arduino Starter Kit example
 Project 13  - Touch Sensor Lamp

 This sketch is written to accompany Project 13
in the
 Arduino Starter Kit

 Parts required:
 1 Megohm resistor
 metal foil or copper mesh
 220 ohm resistor
 LED

 Software required :
 CapacitiveSensor library by Paul Badger
 http://arduino.cc/playground/Main/
CapacitiveSensor

 Created 18 September 2012
 by Scott Fitzgerald

 http://arduino.cc/starterKit

 This example code is part of the public domain
 */
```

Capacitive Sensor

http://playground.arduino.cc/Main/CapacitiveSensor

The capacitiveSensor library turns two or more Arduino pins into a capacitive sensor, which can sense the electrical capacitance of the human body. All the sensor setup requires is a medium to high value resistor and a piece of wire and a small (to large) piece of aluminum foil on the end. At its most sensitive, the sensor will start to sense a hand or body inches away from the sensor.

Have a look also a Pencil Based Capacitive Sensor
http://www.bareconductive.com/capacitance-sensor

Tutorial about touch sensors features
http://www.instructables.com/id/Touche-for-Arduino-Advanced-touch-sensing/
https://www.youtube.com/watch?v=ikD_3Vemkf0

HARDWARE REQUIRED

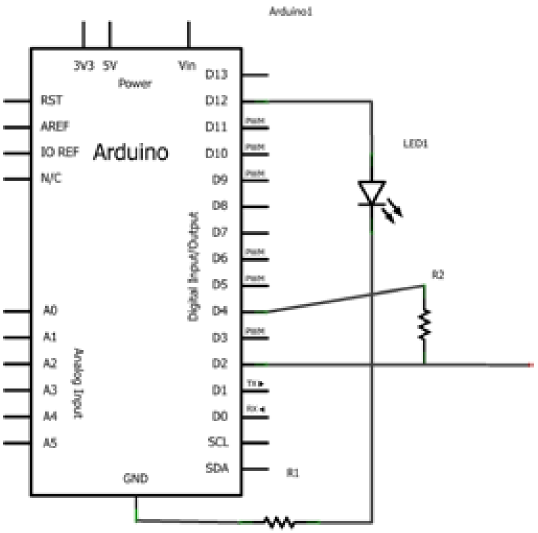
Arduino Board
1 LED
1 470 Ωhm Resistor
1 1MΩhm Resistor (Test the values using 5MΩ up to 50MΩ resistors)
breadboard
hook-up wire

CIRCUIT

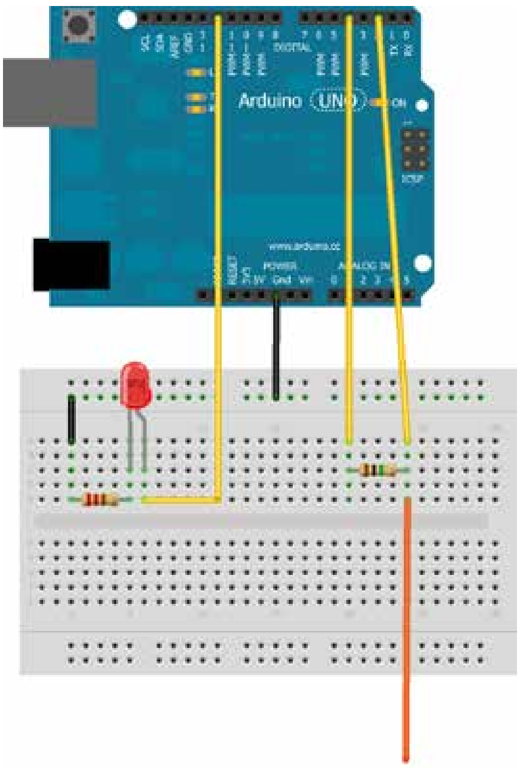
Resistor Choice

Here are some guidelines for resistors but be sure to experiment for a desired response. Use a 1 megohm resistor (or less maybe) for absolute touch to activate.
With a 10 megohm resistor the sensor will start to respond 4-6 inches away.
With a 40 megohm resistor the sensor will start to respond 12-24 inches away (dependent on the foil size). Common resistor sizes usually end at 10 megohm so you may have to solder four 10 megohm resistors end to end.
One tradeoff with larger resistors is that the sensor's increased sensitivity means that it is slower. Also if the sensor is exposed metal, it is possible that the send pin will never be able to force a change in the receive (sensor) pin, and the sensor will timeout.
Also experiment with small capacitors (100 pF - .01 uF) to ground, on the sense pin. They improve stability of the sensor.
Note that the hardware can be set up with one sPin and several resistors and rPin's for calls to various capacitive sensors. See the example sketch.

SCHEMATIC



IMAGE



CODE

```
/*
 Arduino Starter Kit example
 Project 13  - Touch Sensor Lamp

 This sketch is written to accompany Project 13
in the
 Arduino Starter Kit

 Parts required:
 1 Megohm resistor
 metal foil or copper mesh
 220 ohm resistor
 LED

 Software required :
 CapacitiveSensor library by Paul Badger
 http://arduino.cc/playground/Main/
CapacitiveSensor

 Created 18 September 2012
 by Scott Fitzgerald

 http://arduino.cc/starterKit

 This example code is part of the public domain
 */
```

```
// import the library (must be located in the
// Arduino/libraries directory)
#include <CapacitiveSensor.h>

// create an instance of the library
// pin 4 sends electrical energy
// pin 2 senses senses a change
CapacitiveSensor capSensor = CapacitiveSensor(4,2);

// threshold for turning the lamp on
int threshold = 1000;

// pin the LED is connected to
const int ledPin = 12;


void setup() {
  // open a serial connection
  Serial.begin(9600);
  // set the LED pin as an output
  pinMode(ledPin, OUTPUT);
}

void loop() {
  // store the value reported by the sensor in a variable
  int sensorValue = capSensor.capacitiveSensor(30);

  // print out the sensor value
  Serial.println(sensorValue);

  // if the value is greater than the threshold
  if(sensorValue > threshold) {
    // turn the LED on
    digitalWrite(ledPin, HIGH);
  }
  // if it’s lower than the threshold
  else {
    // turn the LED off
    digitalWrite(ledPin, LOW);
  }

  delay(50);
}
```

3 Capacitive Foil Sensors

http://playground.arduino.cc/Main/CapacitiveSensor

The capacitiveSensor library turns two or more Arduino pins into a capacitive sensor, which can sense the electrical capacitance of the human body. All the sensor setup requires is a medium to high value resistor and a piece of wire and a small (to large) piece of aluminum foil on the end. At its most sensitive, the sensor will start to sense a hand or body inches away from the sensor.

Have a look also a Pencil Based Capacitive Sensor

http://www.bareconductive.com/capacitance-sensor

Tutorial about touch sensors features

http://www.instructables.com/id/Touche-for-Arduino-Advanced-touch-sensing/

HARDWARE REQUIRED

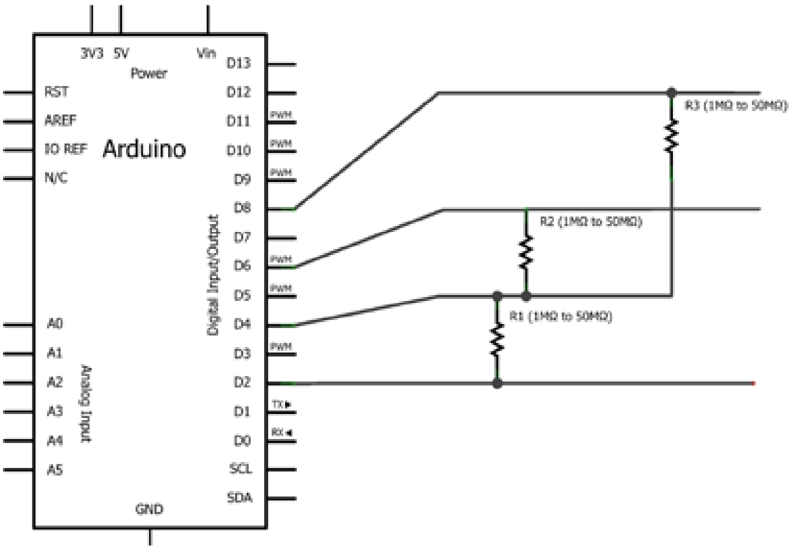
Arduino Board
3 10MΩhm Resistor (Test the values using 5MΩ up to 50MΩ resistors)
3 Pieces of Foil (Connect the foils to wires)
breadboard
hook-up wires

CIRCUIT

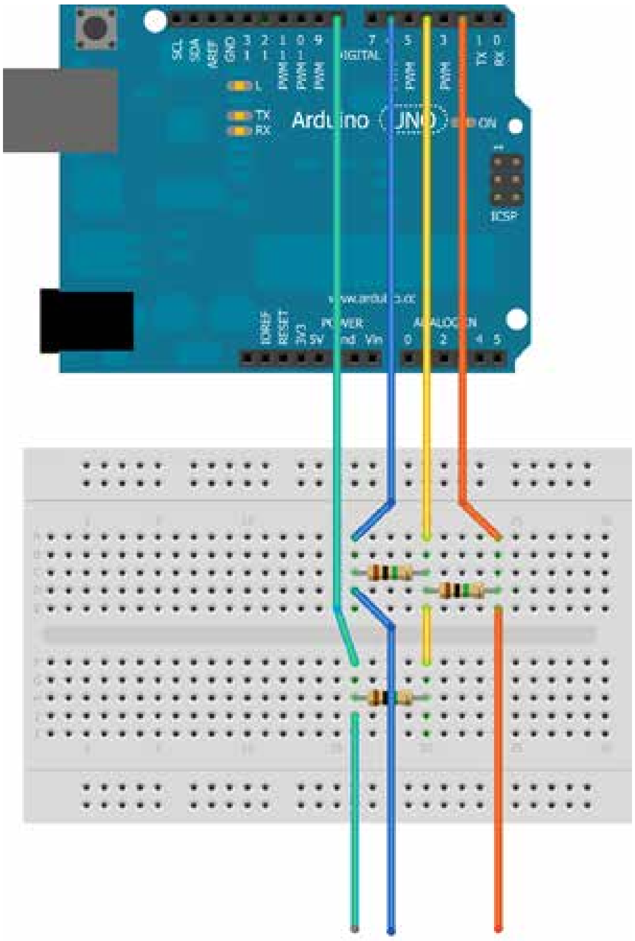
Resistor Choice

Here are some guidelines for resistors but be sure to experiment for a desired response. Use a 1 megohm resistor (or less maybe) for absolute touch to activate.
With a 10 megohm resistor the sensor will start to respond 4-6 inches away.
With a 40 megohm resistor the sensor will start to respond 12-24 inches away (dependent on the foil size). Common resistor sizes usually end at 10 megohm so you may have to solder four 10 megohm resistors end to end.
One tradeoff with larger resistors is that the sensor's increased sensitivity means that it is slower. Also if the sensor is exposed metal, it is possible that the send pin will never be able to force a change in the receive (sensor) pin, and the sensor will timeout.
Also experiment with small capacitors (100 pF - .01 uF) to ground, on the sense pin. They improve stability of the sensor.
Note that the hardware can be set up with one sPin and several resistors and rPin's for calls to various capacitive sensors. See the example sketch.

SCHEMATIC



IMAGE



CODE

```
#include <CapacitiveSensor.h>

// 10M resistor between pins 4 & 2, pin 2 is sensor pin, add a wire and or foil if
desired
CapacitiveSensor   cs_4_2 = CapacitiveSensor(4,2);
 // 10M resistor between pins 4 & 6, pin 6 is sensor pin, add a wire and or foil
CapacitiveSensor   cs_4_6 = CapacitiveSensor(4,6);
 // 10M resistor between pins 4 & 8, pin 8 is sensor pin, add a wire and or foil
CapacitiveSensor   cs_4_8 = CapacitiveSensor(4,8);

void setup()
{
   // turn off autocalibrate on channel 1 - just as an example
   cs_4_2.set_CS_AutocaL_Millis(0xFFFFFFFF);
   Serial.begin(9600);
}

void loop()
{
    long start = millis();
    long total1 =  cs_4_2.capacitiveSensor(30);
    long total2 =  cs_4_6.capacitiveSensor(30);
    long total3 =  cs_4_8.capacitiveSensor(30);

    Serial.print(millis() - start);      // check on performance in milliseconds
    Serial.print(“\t”);                  // tab character for debug windown spacing

    Serial.print(total1);                // print sensor output 1
    Serial.print(“\t”);
    Serial.print(total2);                // print sensor output 2
    Serial.print(“\t”);
    Serial.println(total3);              // print sensor output 3

    delay(50);                           // arbitrary delay to limit data to serial port
}
```

```
/*
 * CapitiveSense Library Demo Sketch
 * Paul Badger 2008
 * Uses a high value resistor e.g. 10M between send pin and receive pin
 * Resistor effects sensitivity, experiment with values, 50K - 50M. Larger resistor values yield larger sensor values.
 * Receive pin is the sensor pin - try different amounts of foil/metal on this pin
 */
```

Theramin

EXERCISE

Play melody using Capasitive Sensor features. Use a digital circuit as an analogue device to produce sound. Buttons are also desirable.

HARDWARE REQUIRED

Arduino Board
momentary buttons or switch (optional)
10K ohm resistors (optional)
LEDs (optional)
220Ω Resistors (optional)
MΩ Resistors
Metal foils or metal objects
buzzers or 8Ω Speakers
breadboard
hook-up wire

SCHEMATIC