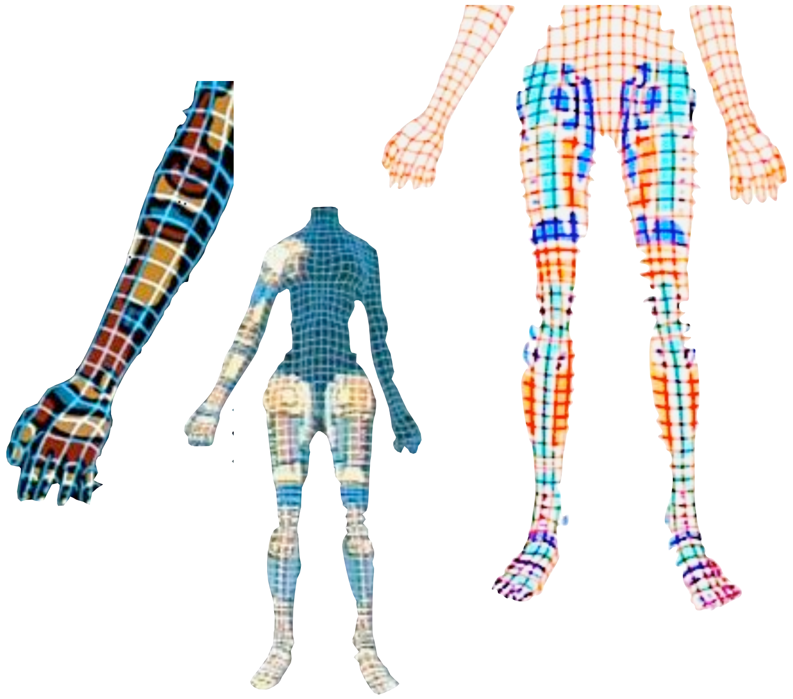
Bionic Arduino

Introduction to Microcontrollers with Arduino

Class 3



18 Nov 2007 - machineproject - Tod E. Kurt

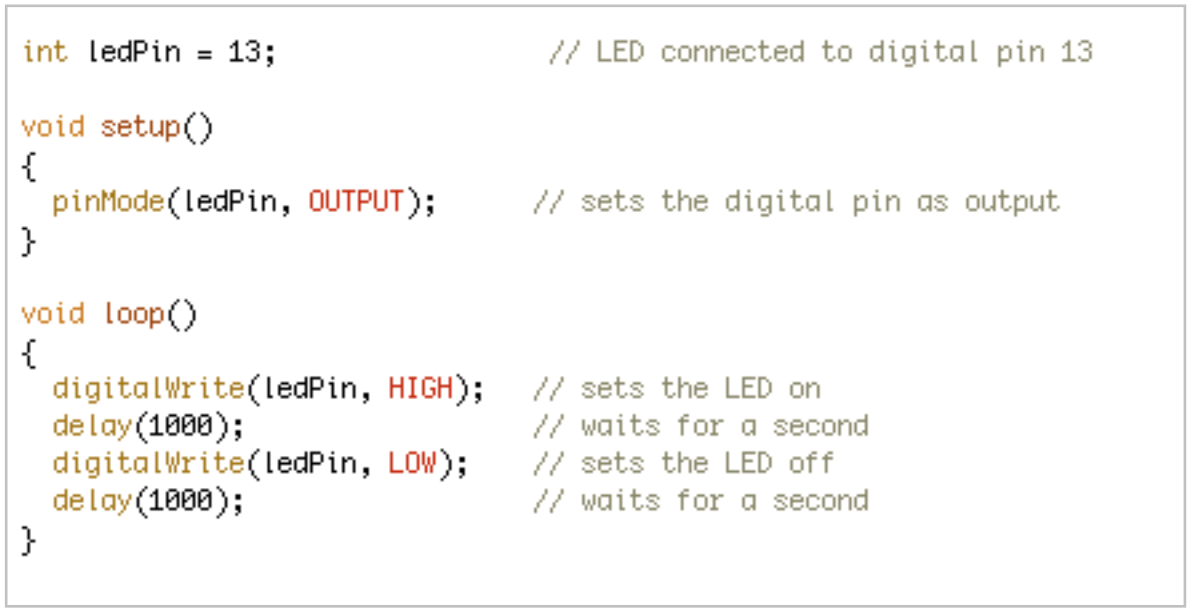
What’s for Today

- About DC motors
- Transistors as switches
- Controlling DC motors
- Introduction to Processing
- Controlling your computer with Arduino
- Piezo buzzers as sensors

In the handout thumbdrives, be sure to copy the Processing zip or dmg file for your OS.

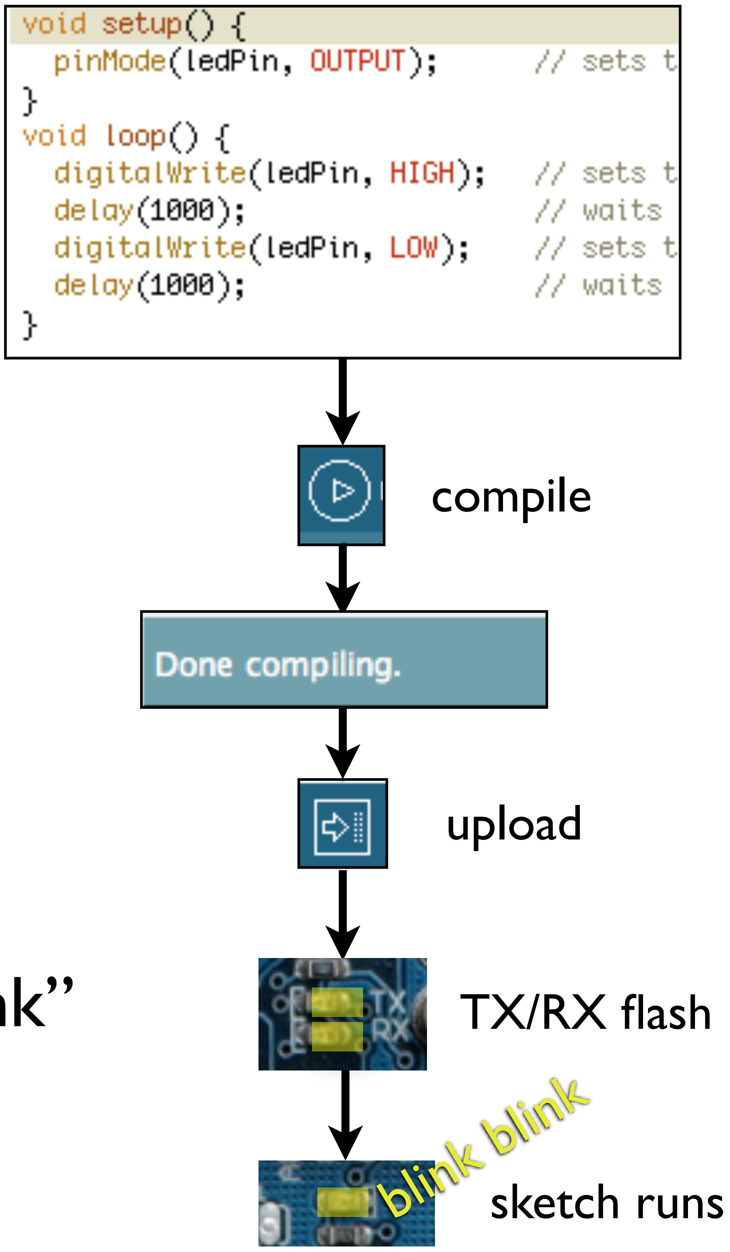
Recap: Blinky LED

Make sure things still work



```
int ledPin = 13;                // LED connected to digital pin 13

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

void loop()
{
  digitalWrite(ledPin, HIGH);   // sets the LED on
  delay(1000);                  // waits for a second
  digitalWrite(ledPin, LOW);    // sets the LED off
  delay(1000);                  // waits for a second
}
```

Load "File/Sketchbook/Examples/Digital/Blink"



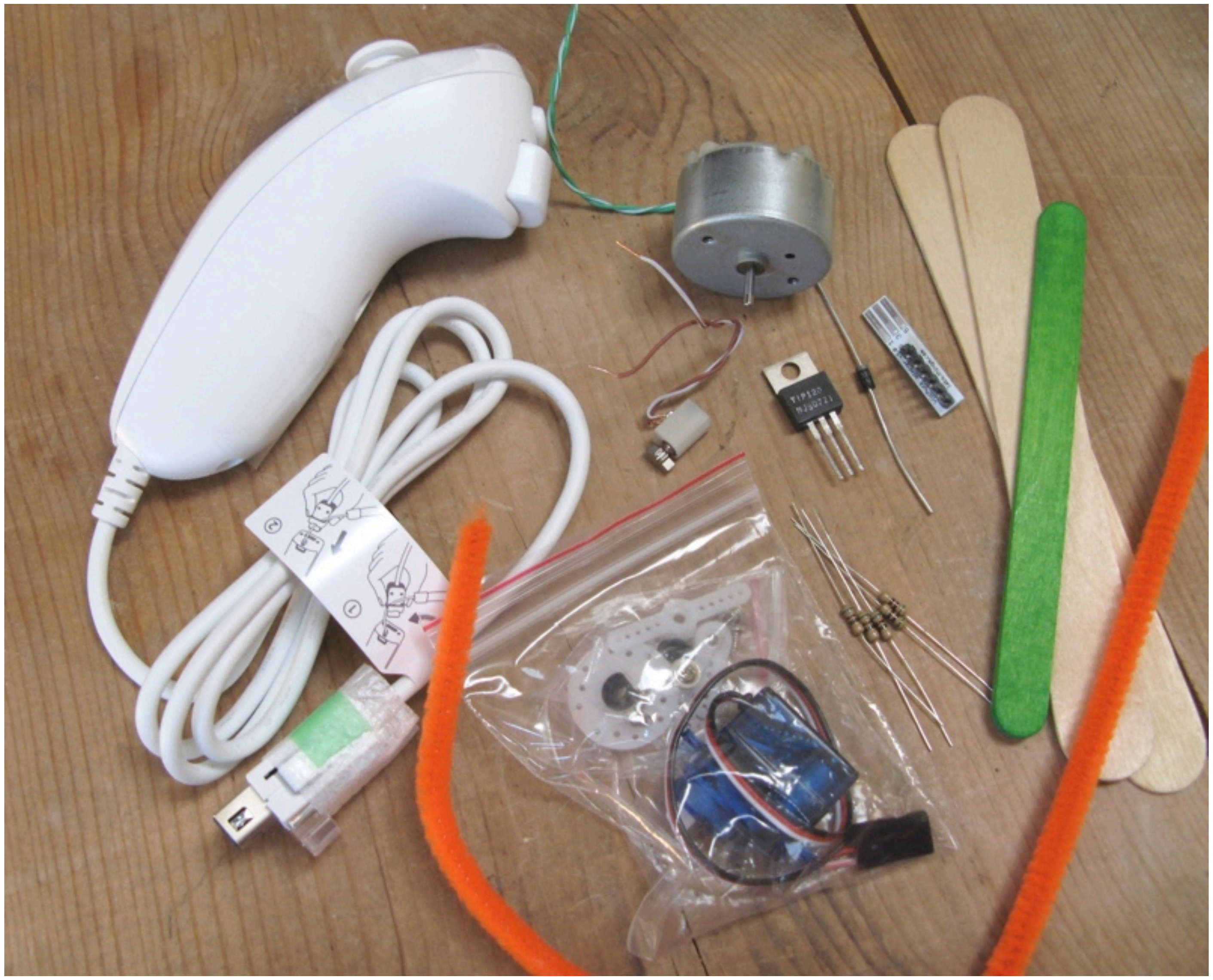
```
void setup() {
  pinMode(ledPin, OUTPUT);      // sets t
}
void loop() {
  digitalWrite(ledPin, HIGH);   // sets t
  delay(1000);                  // waits
  digitalWrite(ledPin, LOW);    // sets t
  delay(1000);                  // waits
}
```

Done compiling.

Change the "delay()" values to change blink rate

Class Kit 2 Contents

"motors & motion"



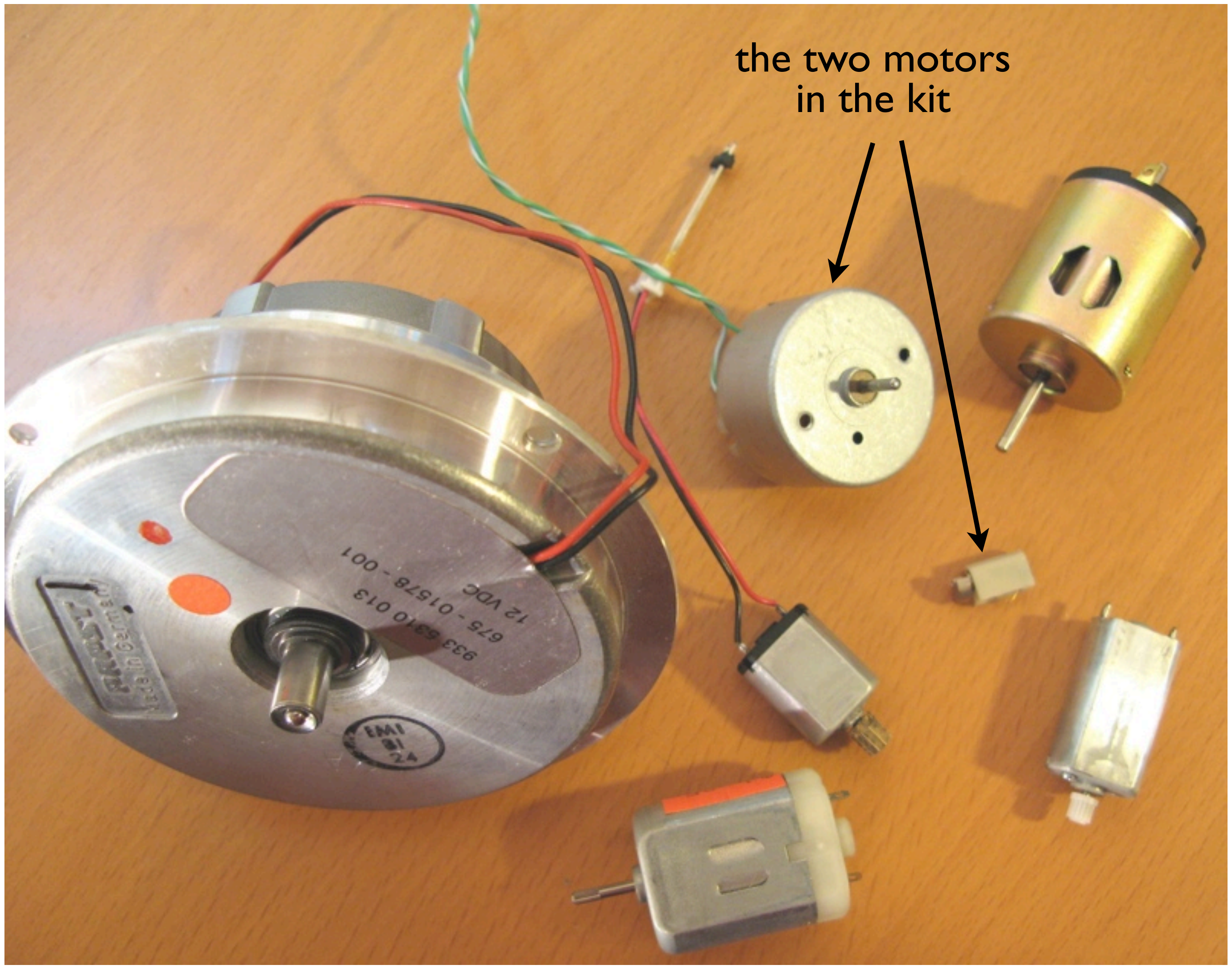
Class Kit 2 Manifest

"motors & motion"

- Nintendo Wii Nunchuck
- Wii Nunchuck Adapter
- Large DC motor
- Small DC motor
- Small servo motor
- TIP120 power transistor
- 1N4001 power diode
- Several 500 ohm resistors (green-brown-brown)
- Couple of popsicle sticks
- Colorful pipe cleaners

DC Motors

come in all shapes and sizes



You probably have 3-4 on you right now

(cell vibrate, laptop fan, laptop dvd drive)

When motors first came out, people thought we'd just have one for the house. The household motor. Various attachments for vacuuming, meat grinding, ceiling fan were available, and some houses had intricate mazes of belts and gears routed through the house to supply this rotational power.

DC Motors

A dizzying array of parameters specify a motor

- direct-drive vs. gearhead – built-in gears or not
- voltage – what voltage it best operates at
- current (efficiency) – how much current it needs to spin
- speed – how fast it spins
- torque – how strong it spins
- oh, and also: size, shaft diameter, shaft length,etc.

The two motors you have are small direct-drive, high-efficiency motors that work at 5 volts

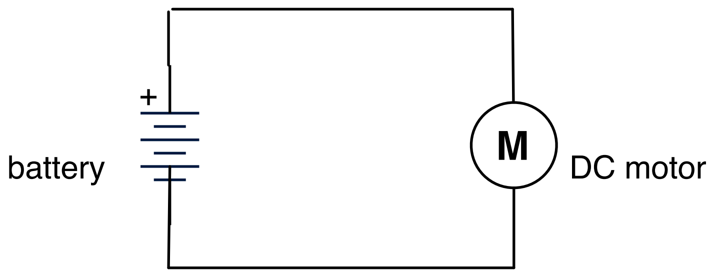
Gearhead motors are the best.

DC Motors Characteristics

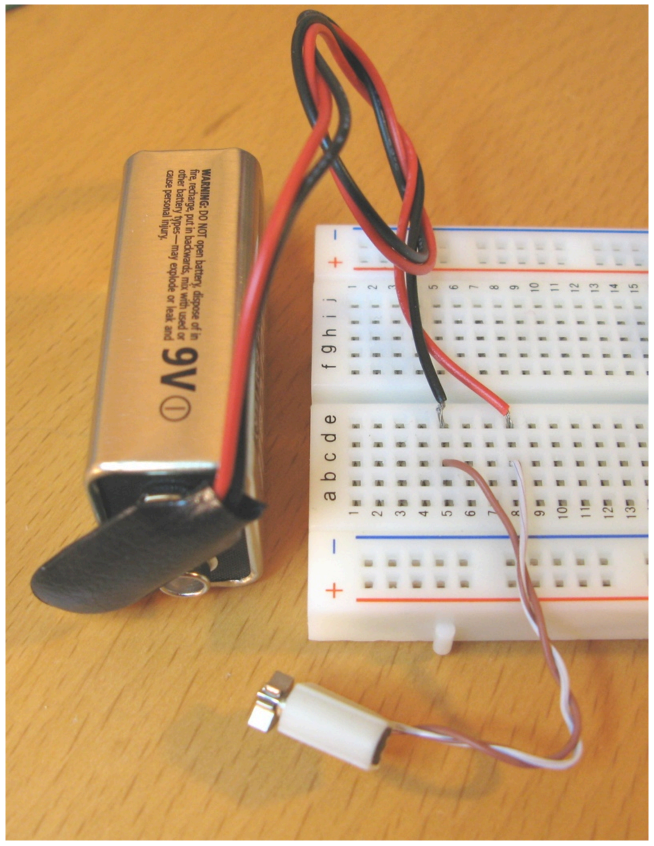
- When the first start up, they draw a *lot* more current, up to 10x more.
- If you "stall" them (make it so they can't turn), they also draw a lot of current
- They can operate in either direction, by switching voltage polarity
- Usually spin very fast: >1000 RPM
- To get slower spinning, need gearing

DC Motors

To drive them, apply a voltage
The higher the voltage, the faster the spinning



polarity determines which way it rotates

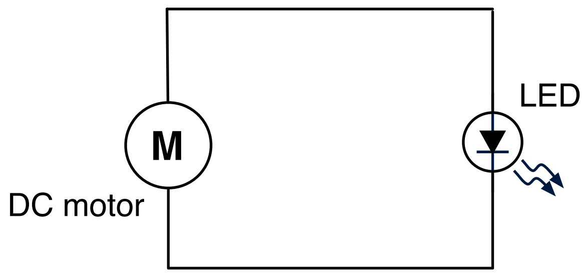


Try this out real quick.
Then swap polarity

Don’t let it go to long. These motors will work at 9V for awhile, but aren’t made to continuously run at that voltage.

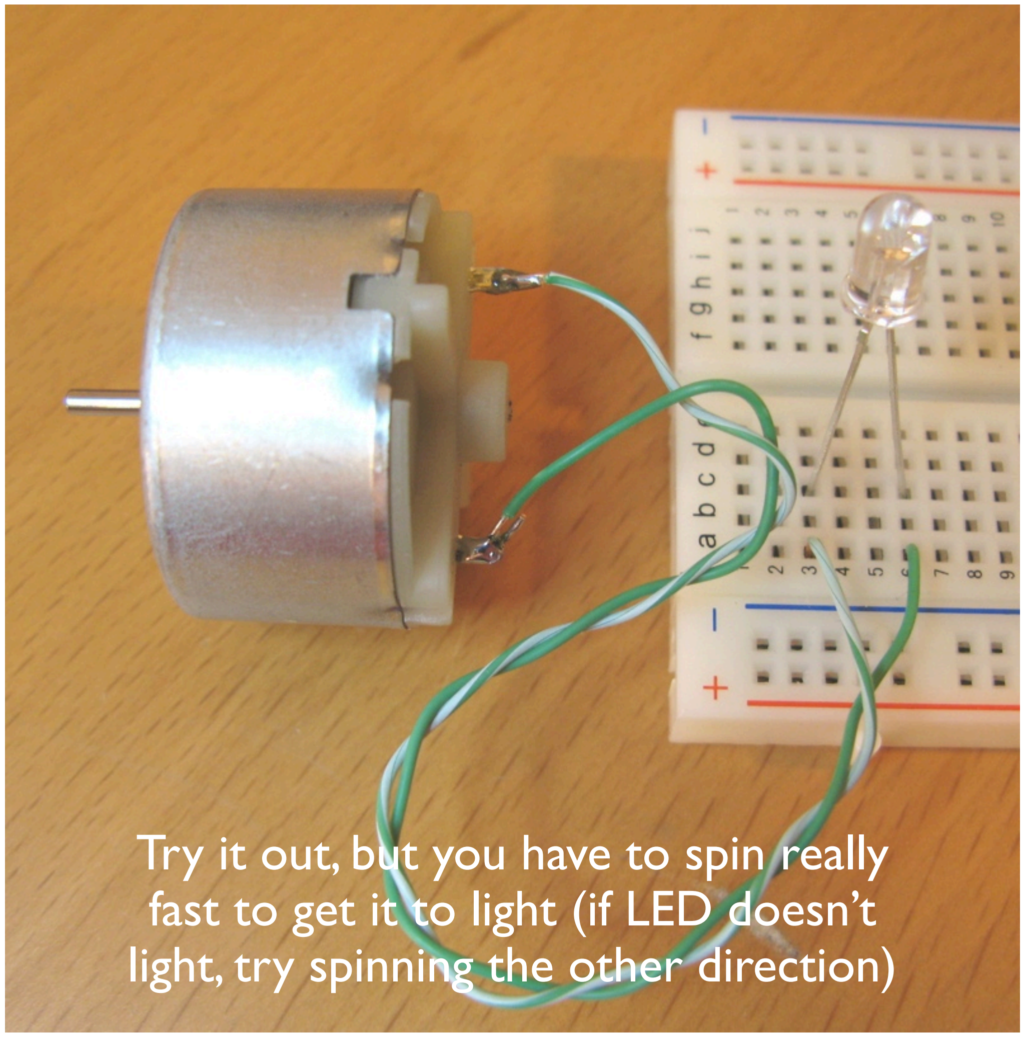
DC Motors as Generators

Just as voltage causes rotation...



...rotation causes voltage

This is used for "regenerative braking" in electric & hybrid cars



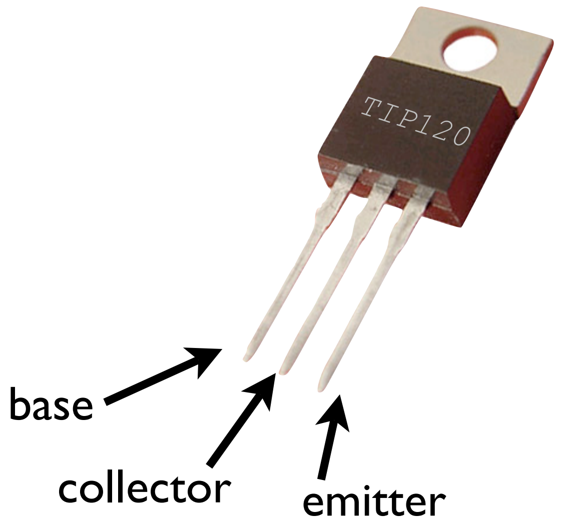
Try it out, but you have to spin really fast to get it to light (if LED doesn't light, try spinning the other direction)

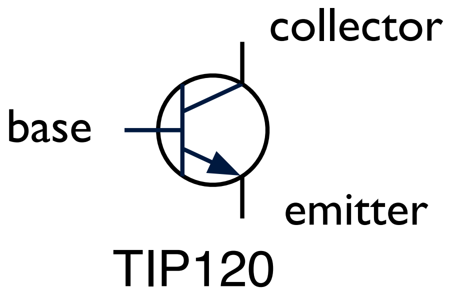
These high-efficiency motors I gave you don't generate much current (because they don't use much current). I have a cheapy motor that lights LEDs better that I can show you.

Transistors

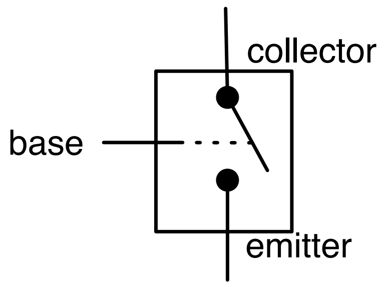
Act like switches

electricity flicks the switch instead of your finger





schematic symbol



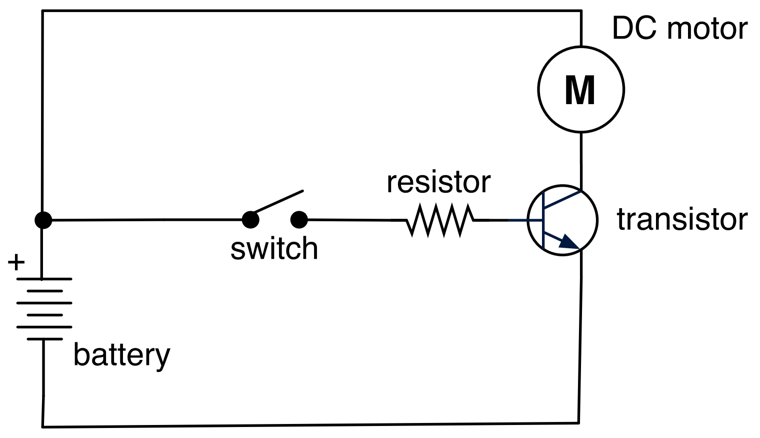
how it kind of works

Turning on the "base" connects the "collector" & "emitter" together

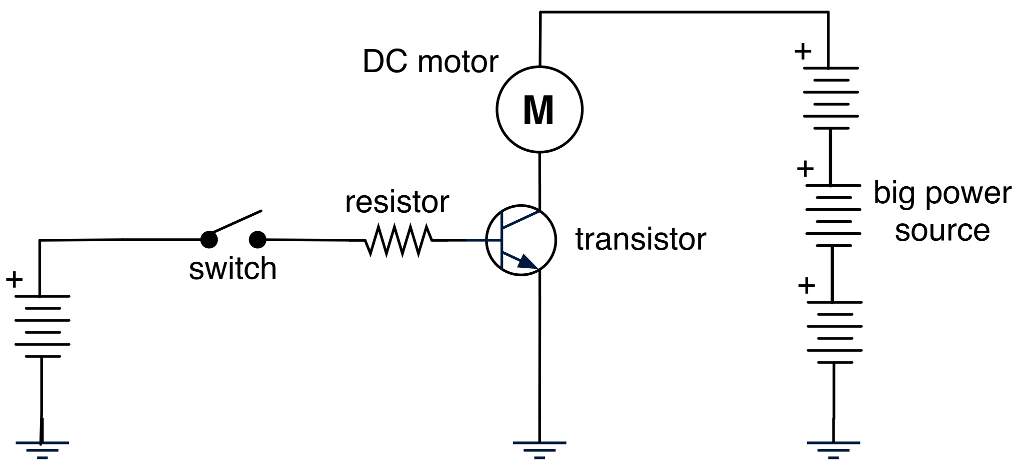
The differences between the pins are very important. The names aren't that important, but their functions are. The "base" is the input that you use to open and close the "switch" across the "collector" and "emitter". On this type of transistor (called an NPN), you need to make sure the collector is always more positive than the emitter. Generally you do this by connecting the emitter to ground.

Switching Motors with Transistors

little motor



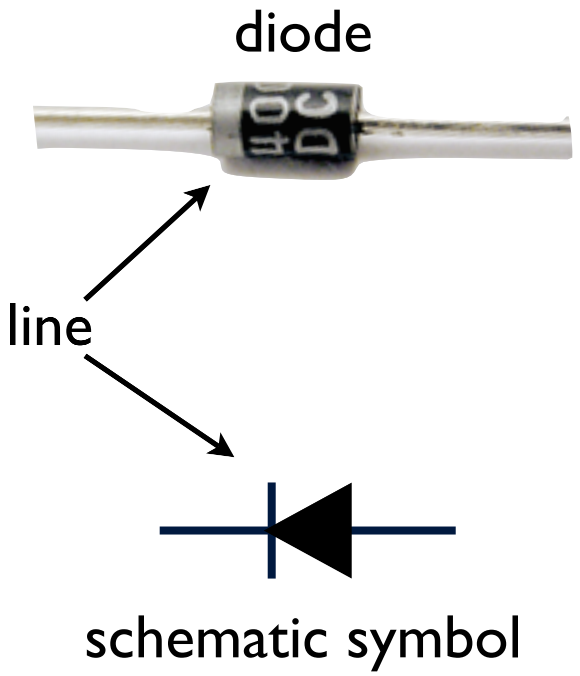
big motor

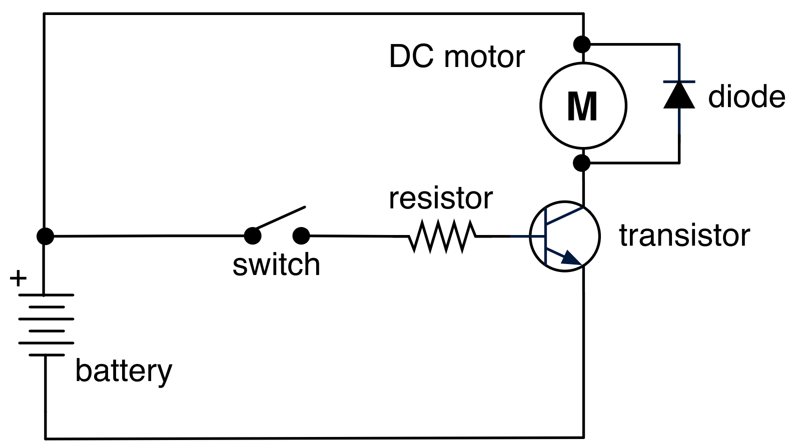


switching a different power source

transistors switch big signals with little signals

Need a "Kickback" Diode



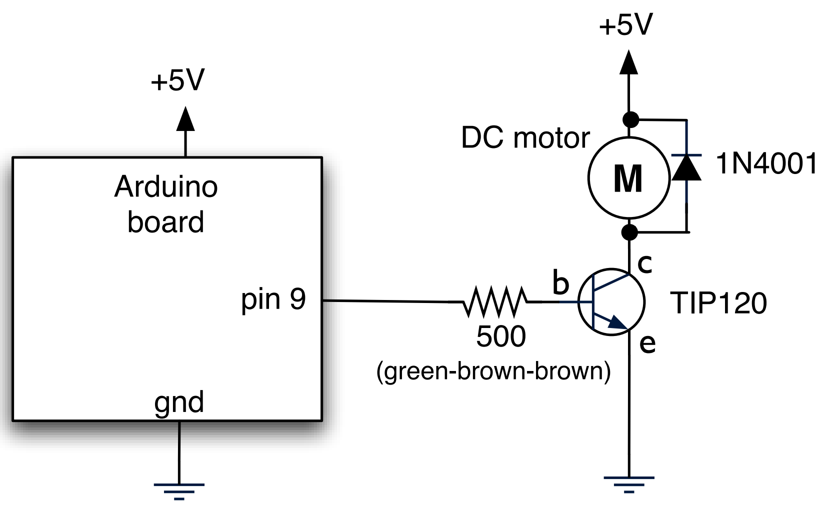


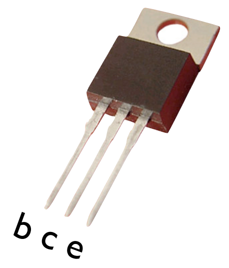
since motors can act like generators,
need to prevent them from generating "kickback" into the circuit

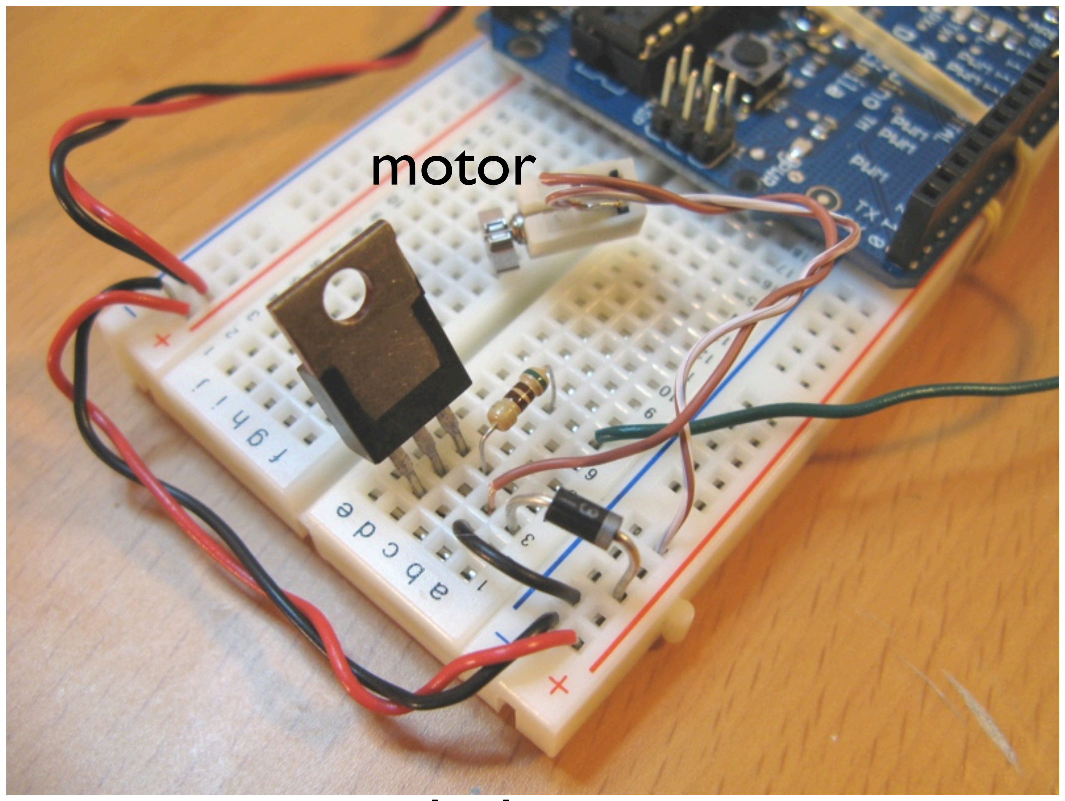
Once a motor starts spinning, its inertia keeps it spinning, this turns it into a generator and thus can generate a "kickback" voltage. The kickback diode routes that voltage harmlessly back into the motor so it can't damage the rest of the circuit.

Kickback is also called "back EMF" (EMF == electromotive force == voltage)

Controlling a Motor





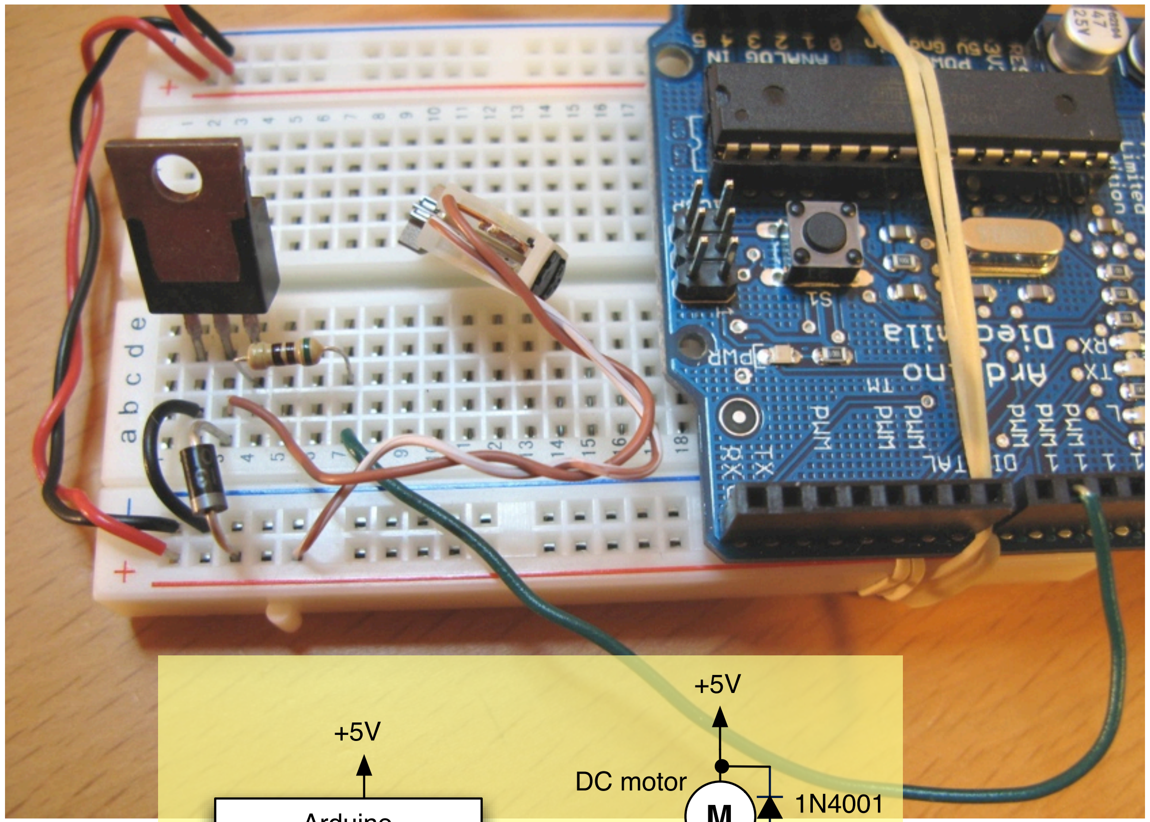


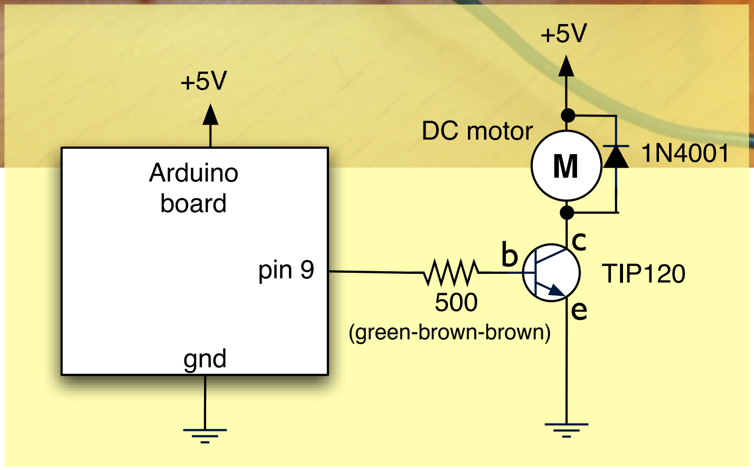
start with the tiny motor

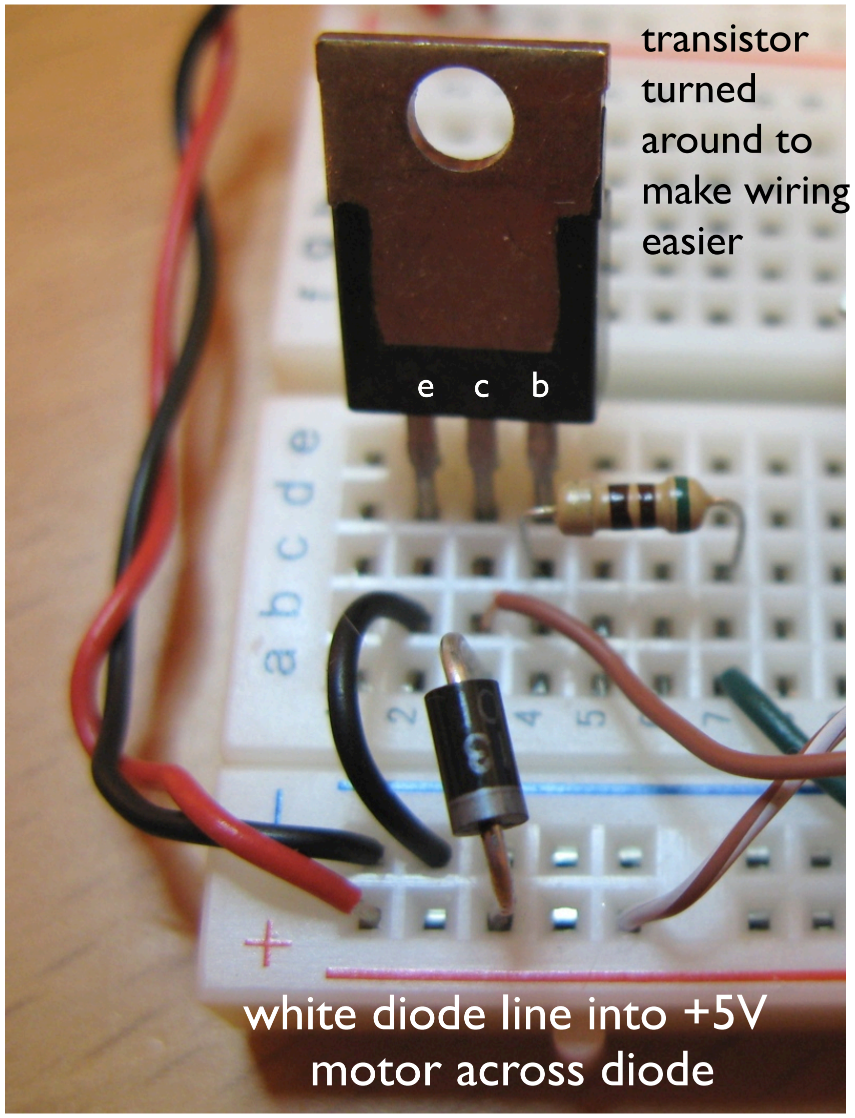
Can control speed of motor with `analogWrite()`
just like controlling brightness of LED

Why 500 ohms? Because I have a lot of 500 ohm resistors. Typically you see 1k ohms. Anything 1k or below will work. The lower the value, the more current you're "wasting" to turn on the transistor.

Wiring up Motor Circuit







transistor turned around to make wiring easier

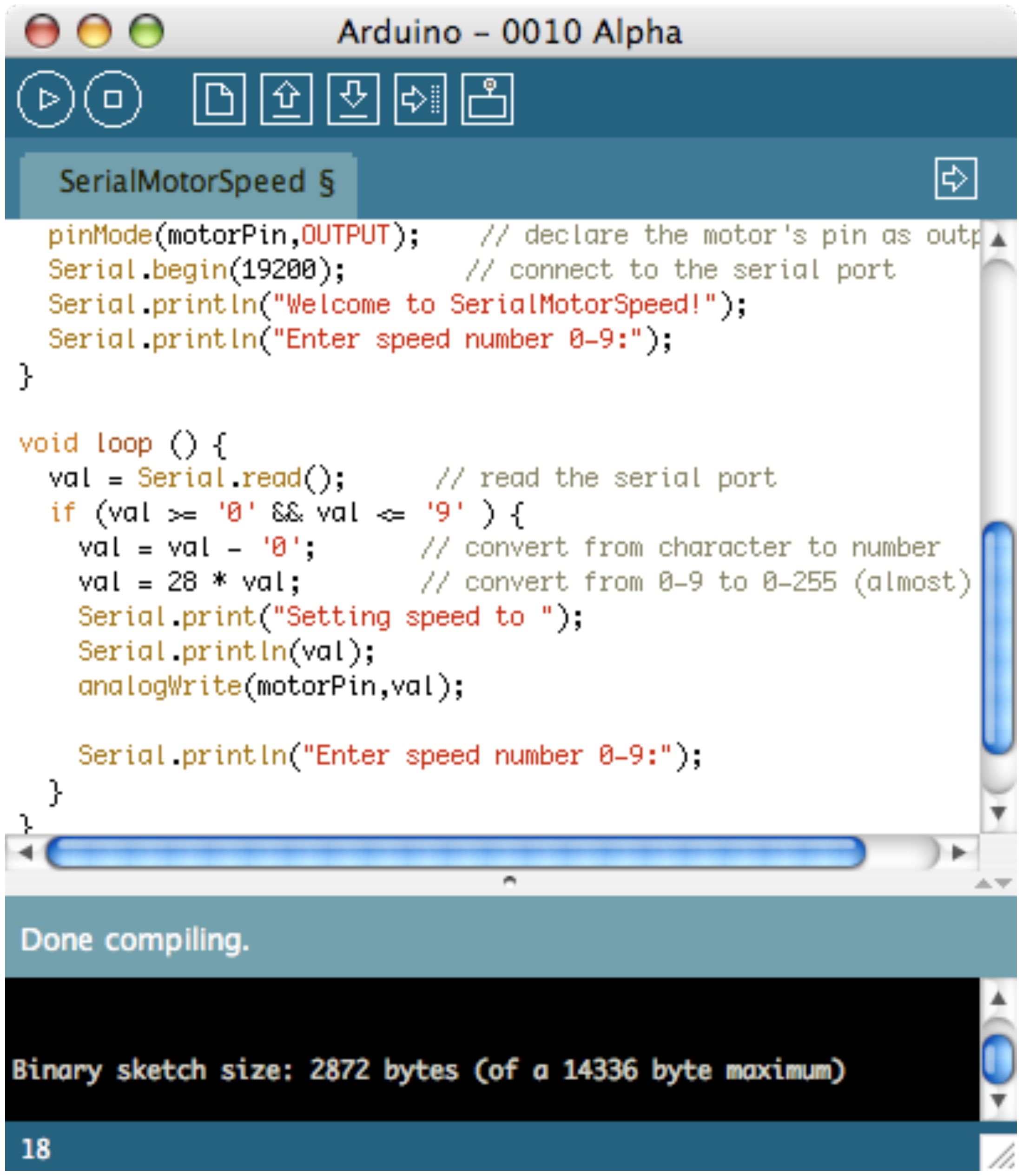
white diode line into +5V
motor across diode

Sketch

“SerialMotorSpeed”

Type a number 0-9 in Serial Monitor to control the speed of the motor

How would you change this to control the motor speed with the potentiometer?



```
  pinMode(motorPin,OUTPUT);     // declare the motor's pin as outp
  Serial.begin(19200);         // connect to the serial port
  Serial.println("Welcome to SerialMotorSpeed!");
  Serial.println("Enter speed number 0-9:");
}

void loop () {
  val = Serial.read();      // read the serial port
  if (val >= '0' && val <= '9' ) {
    val = val - '0';       // convert from character to number
    val = 28 * val;        // convert from 0-9 to 0-255 (almost)
    Serial.print("Setting speed to ");
    Serial.println(val);
    analogWrite(motorPin,val);

    Serial.println("Enter speed number 0-9:");
  }
}
```

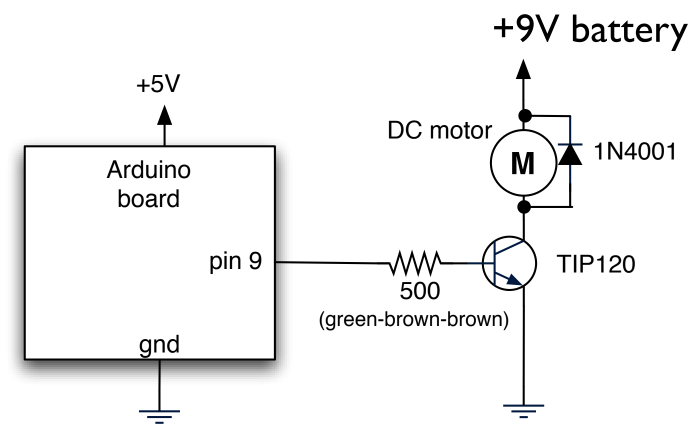
Done compiling.

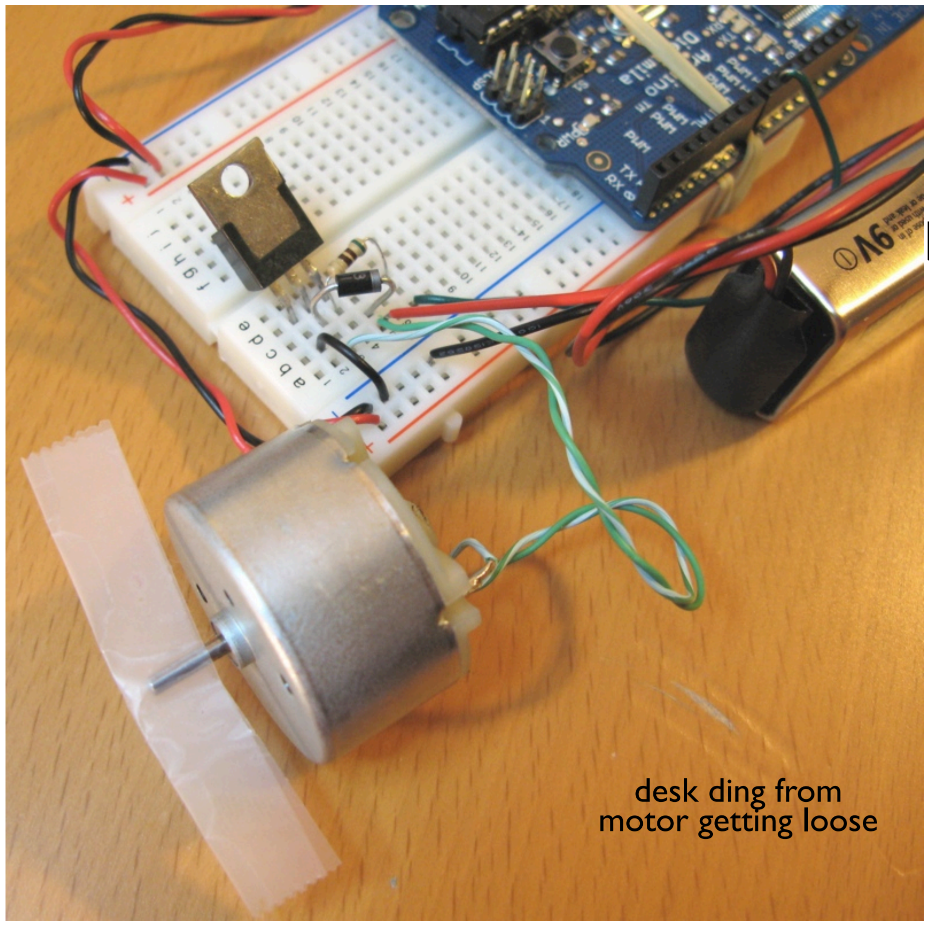
```
Binary sketch size: 2872 bytes (of a 14336 byte maximum)
```

18

Controlling a Bigger Motor

Same circuit as before, different voltage source

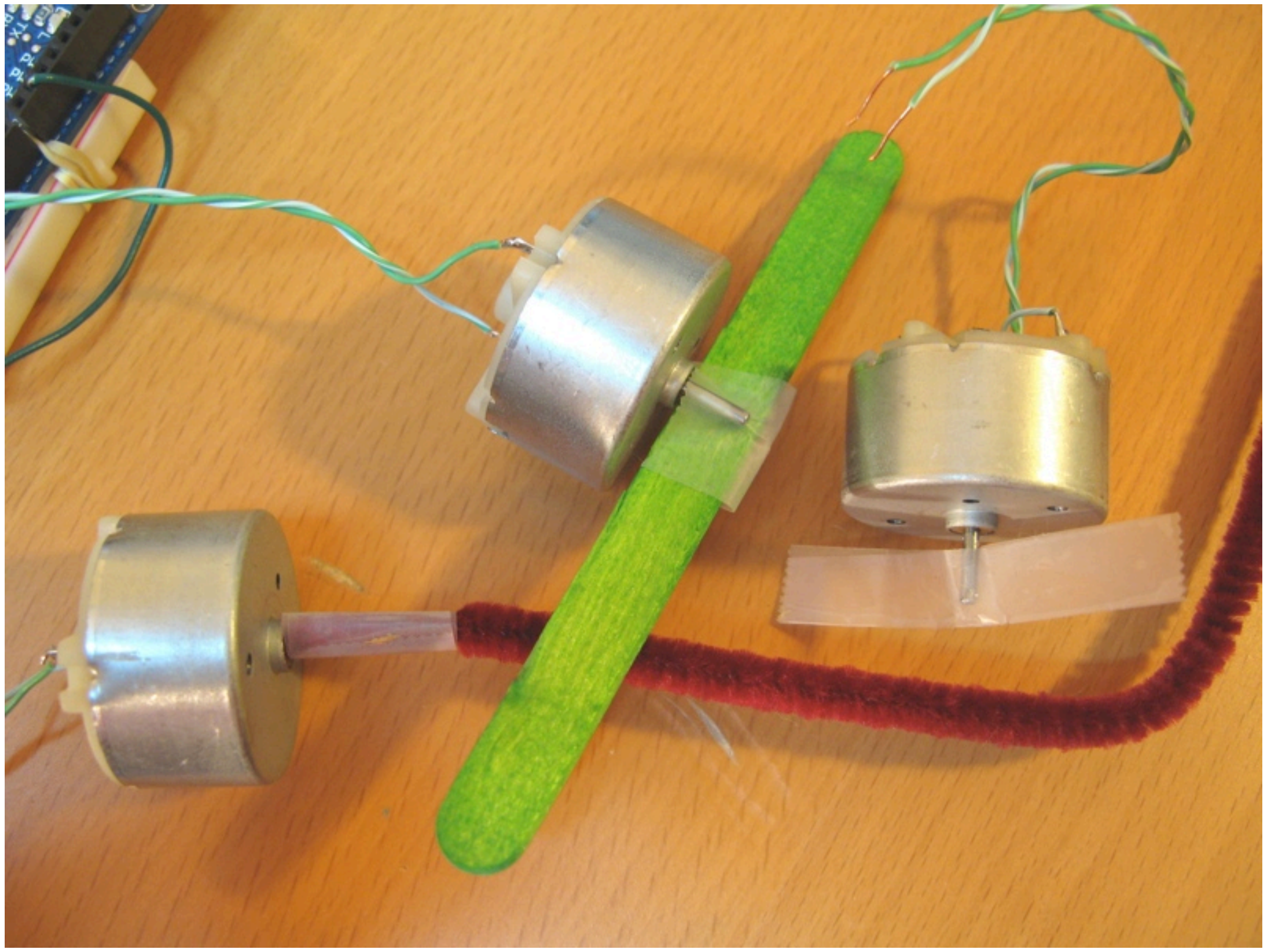




Motor will spin faster for a given `analogWrite()` value

Actually with both of the motors you have, you can run off the Arduino power supply. But many motors cannot because they either draw too much current or they need a voltage higher than 5 volts.

Fun Motor Attachments



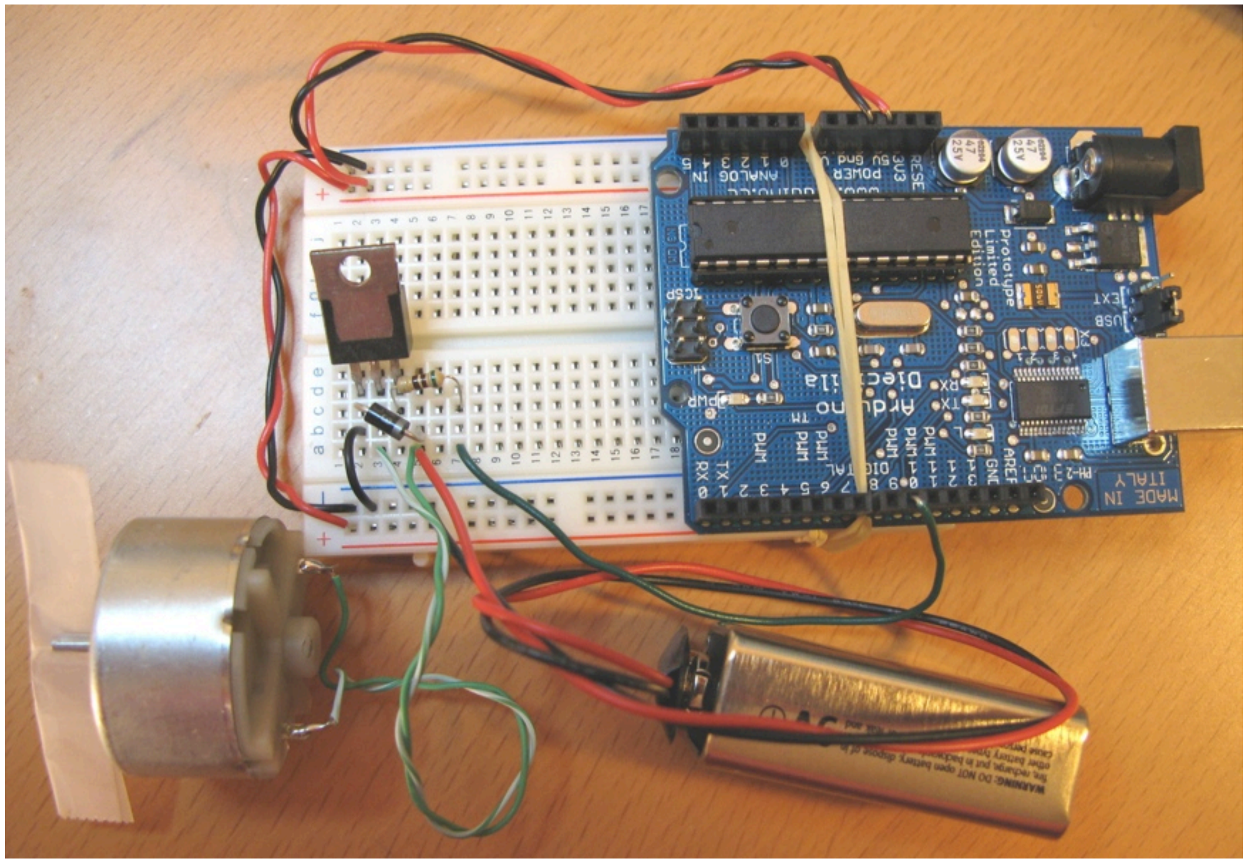
pipe cleaner squiggler

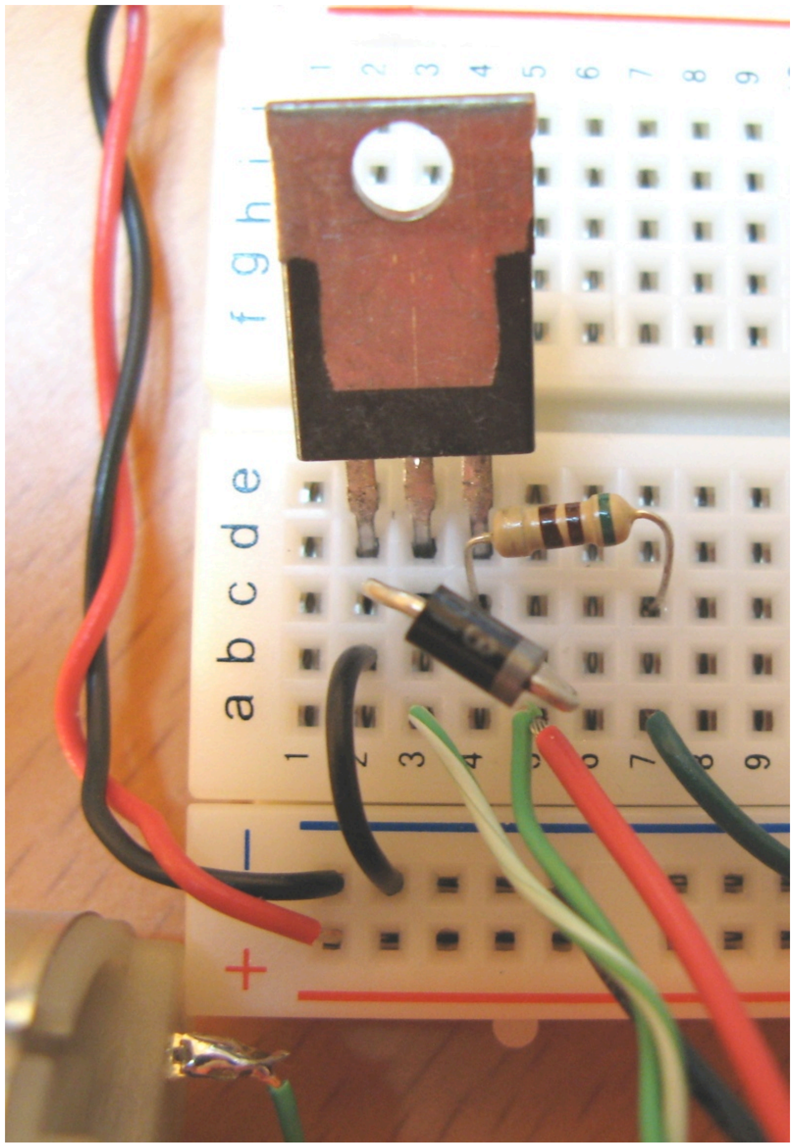
tape propeller

popsicle stick beater

I'm terrible at mechanical engineering. If anyone has good ways of mounting things to motors, let me know. :-)

Wiring Up Bigger Motor

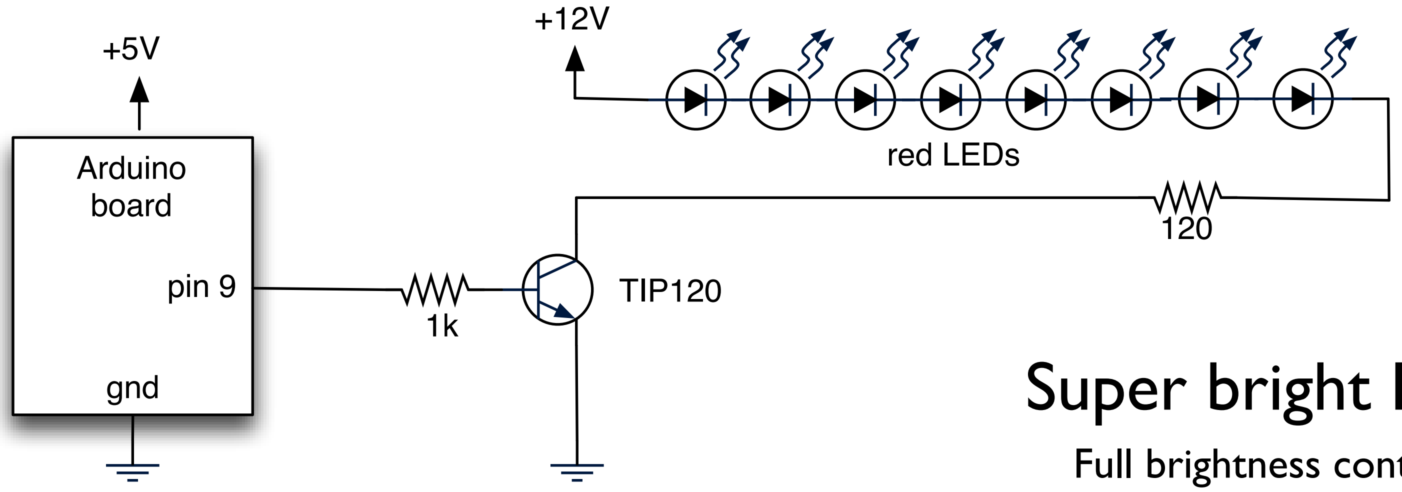




Don’t just add 9V to +5v bus!
Move the diode from +5 to another row
Add red 9V wire to that row,
Add black 9V wire to Gnd

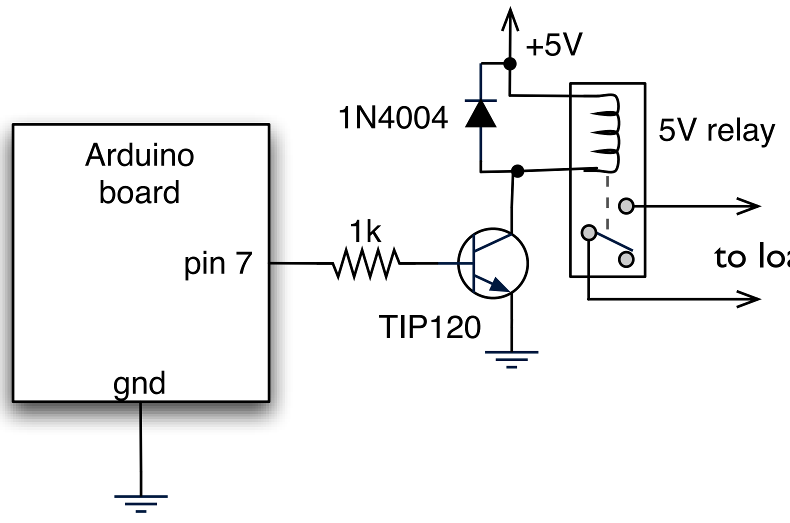
You might find it easier to push the red 9V wire in with the motor wire.

Can Switch Anything*



Super bright LED light

Full brightness control with PWM



to load: light bulb, car ignition, washing machine, etc.

Relay switcher

Just on/off, and a relay needs a diode too

*Anything up to about 1 amp. Need a bigger transistor or a relay after that

Piezo Buzzer as Sensor

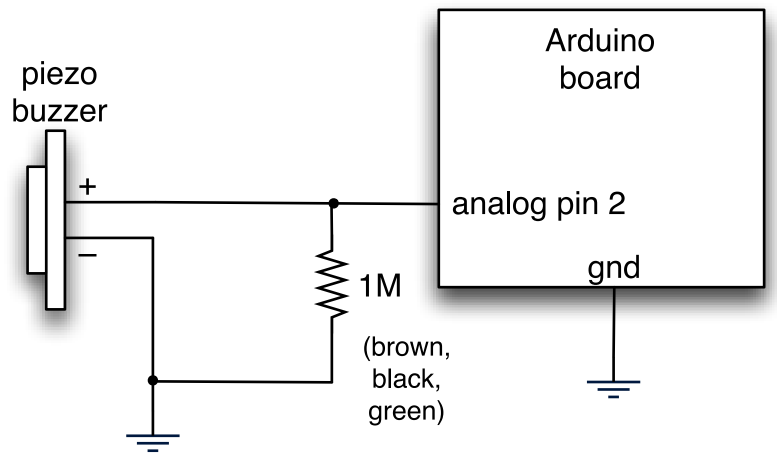
- Piezo buzzers exhibit the *reverse* piezoelectric effect.
- The normal piezoelectric effect is generating electricity from squeezing a crystal.
- Can get several thousand volts, makes a spark
- You probably have seen a big example of this already:

fireplace lighter

I have a demo piezo igniter from one of these lighters. It’s fun to shock yourself.
Puts out several thousand volts. (ionization voltage of air =~ 30kV/cm)

Piezo Knock Sensor

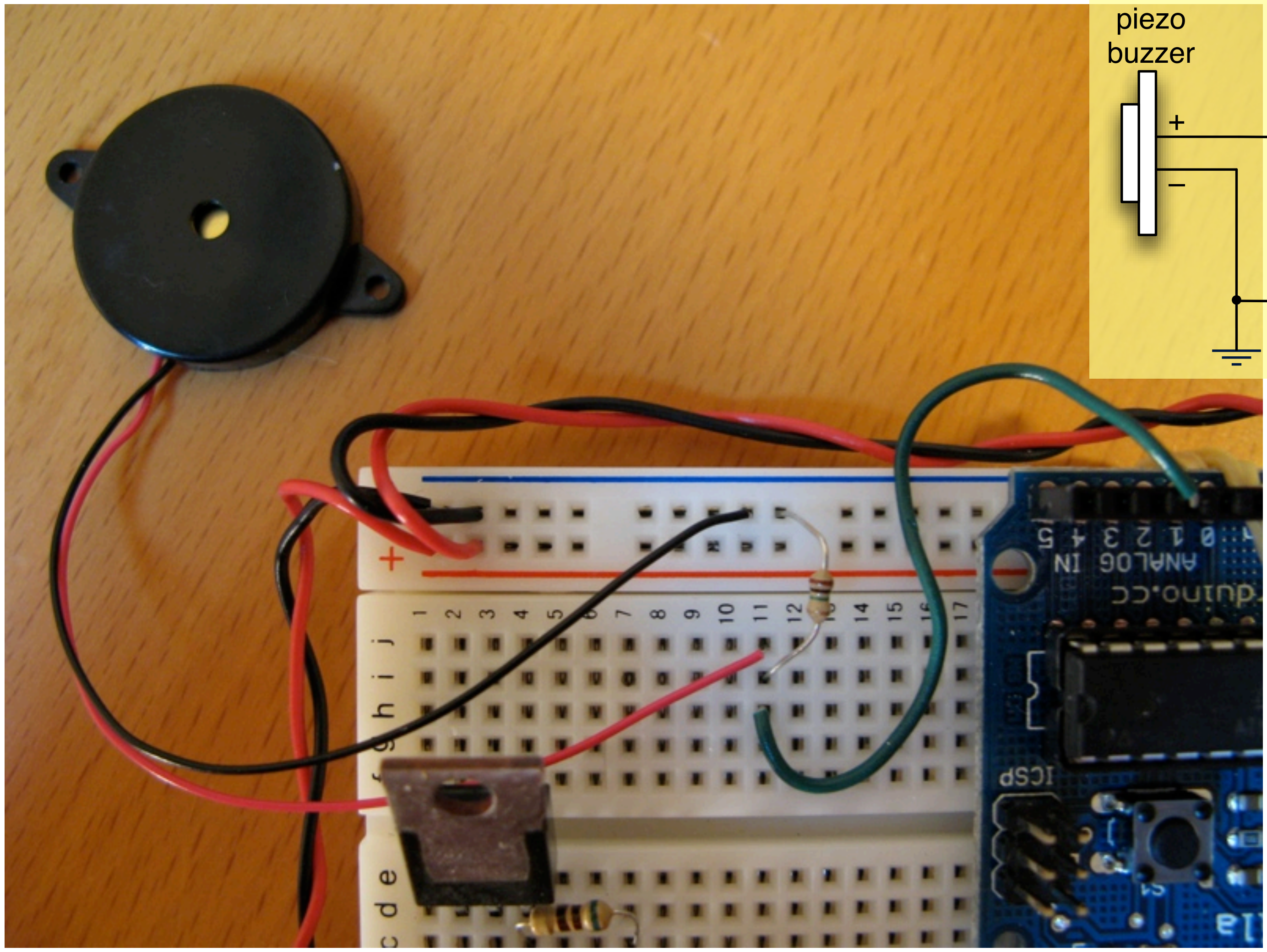
- To read a piezo you can just hook it into an analog input, but:
- You need to drain off any voltage with a resistor, or it just builds up
- The protection diodes inside the AVR chip protect against the high voltage

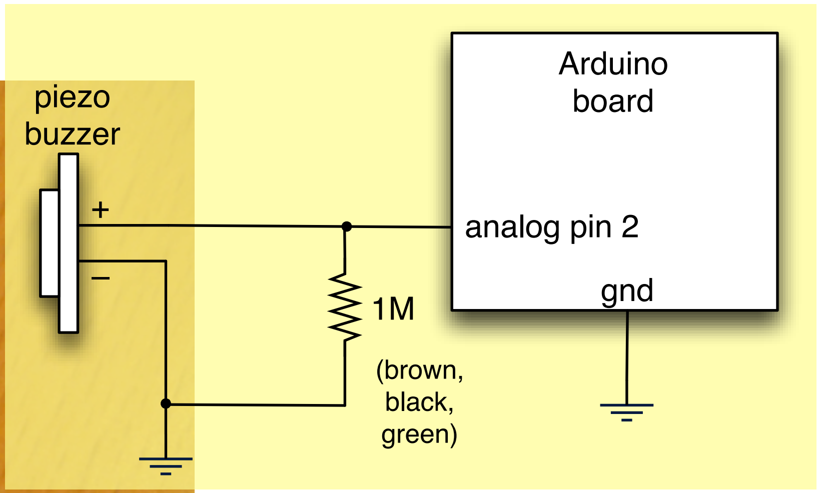


piezo input schematic

Note polarity of piezo still matters.
If you're doing this for real, you'd probably want to add an external protection diode, called a "zener diode". It acts invisible until the voltage gets over its designed value (like 5 volts in this case), then it acts like a short circuit.

Wiring up Piezo Sensor





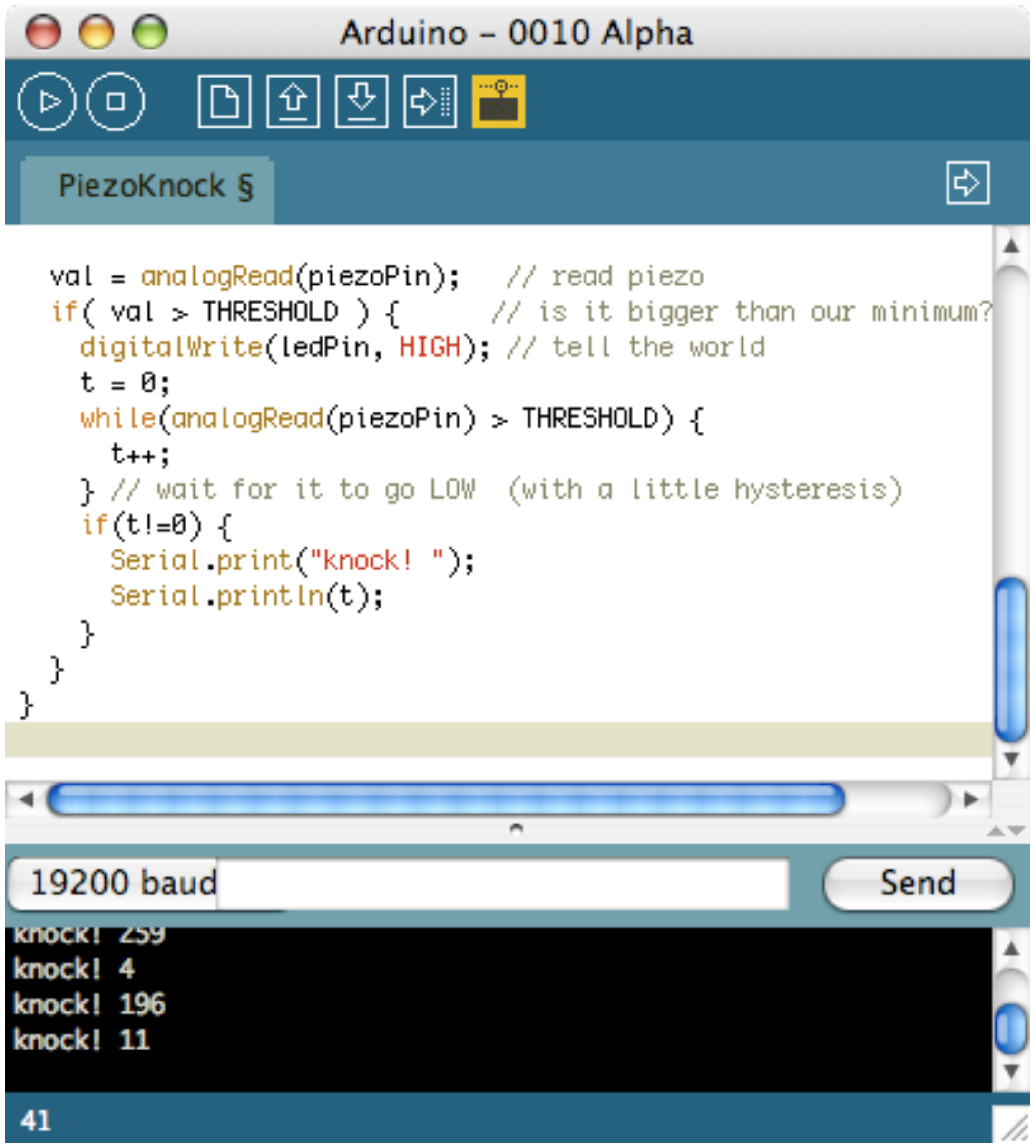
Could also plug it directly into the Arduino, might be easier because of those thin little wires on the piezo.

Piezo Knock

"PiezoKnock"

Whack the piezo to print out a number based on force of whack

Waits for input to go over threshold, then to drop below threshold



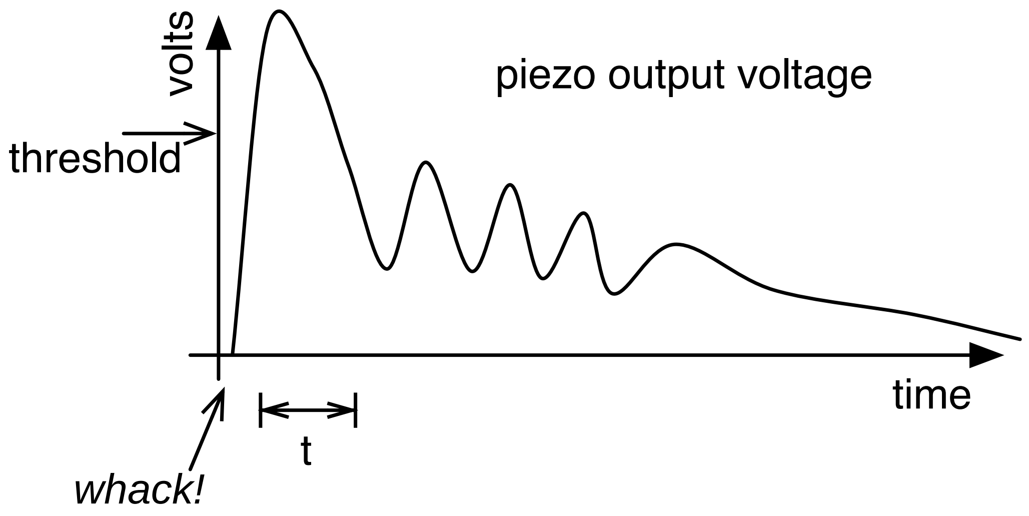
```
  val = analogRead(piezoPin);    // read piezo
  if( val > THRESHOLD ) {        // is it bigger than our minimum?
    digitalWrite(ledPin, HIGH); // tell the world
    t = 0;
    while(analogRead(piezoPin) > THRESHOLD) {
      t++;
    } // wait for it to go LOW  (with a little hysteresis)
    if(t!=0) {
      Serial.print("knock! ");
      Serial.println(t);
    }
  }
}
```

```
knock! 259
knock! 4
knock! 196
knock! 11
```

Number is "t", the number of times it looped waiting for the value to drop below THRESHOLD.
Notice how it doesn't work quite right.

How Does that Work?

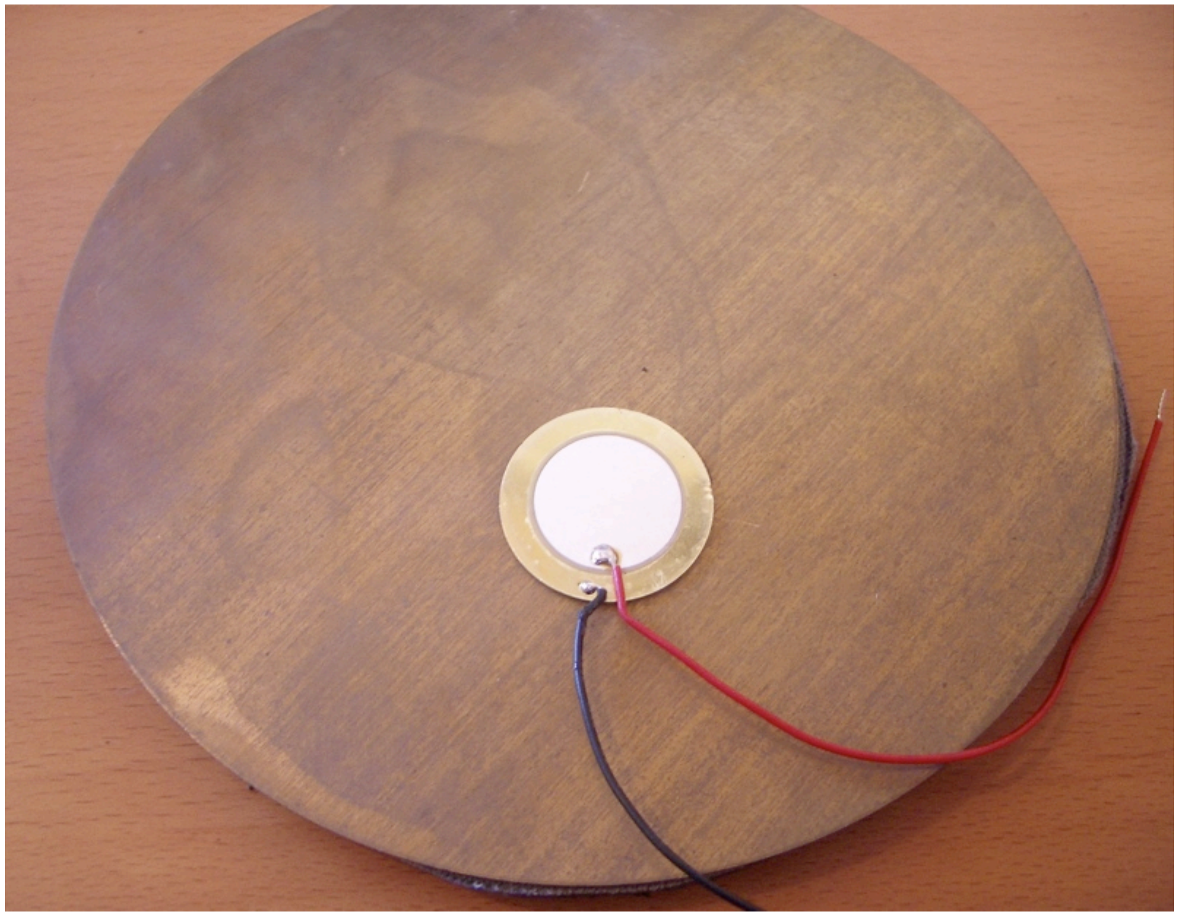
- When a piezo is struck, it “rings” like a bell
- But instead of sound, it outputs voltage
- The sketch measures *time* above a certain voltage, hoping to catch largest ring



Depending on how fast you can watch the input, this technique works either really well or not that well. There are much faster ways of watching inputs that loops with analogRead()
But for now it works okay

Custom Piezo Sensors

Can mount the element on anything
(under rugs, floor mat, door, your body, etc.)



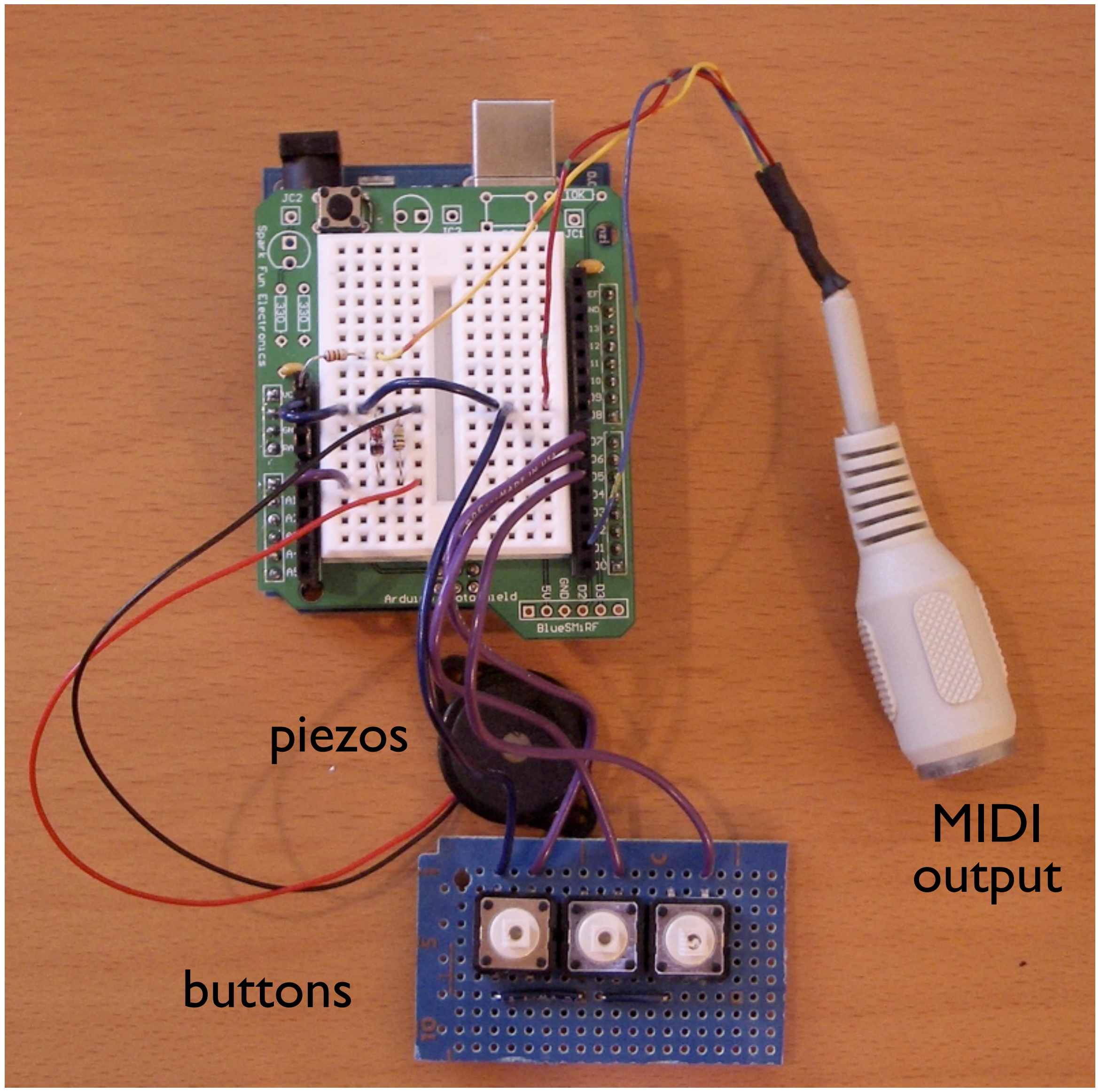
Here's one glued to a larger brass disc for a drum trigger

You can get bare piezo buzzers (not in a black plastic enclosure) that you can mount on whatever you want.

Could make a MIDI Trigger

Uses piezos & buttons to send MIDI messages

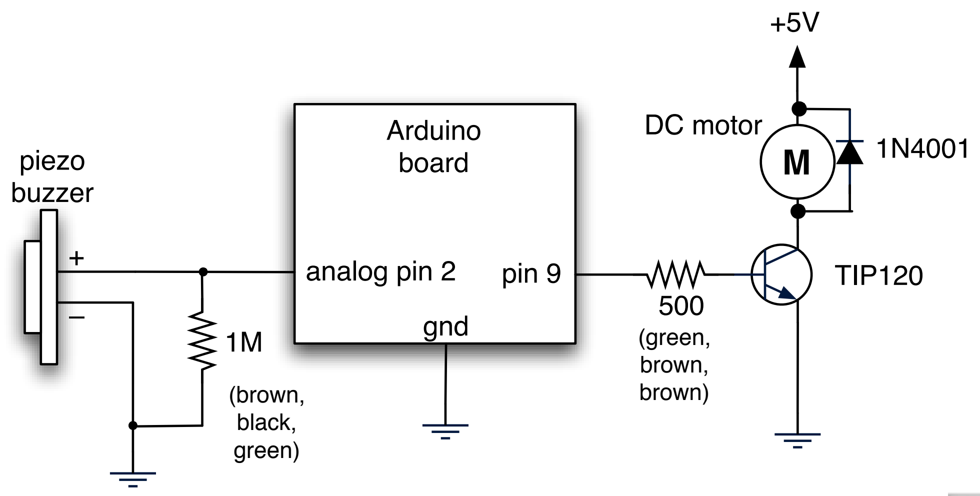
Can trigger drum sounds or arbitrary sound samples



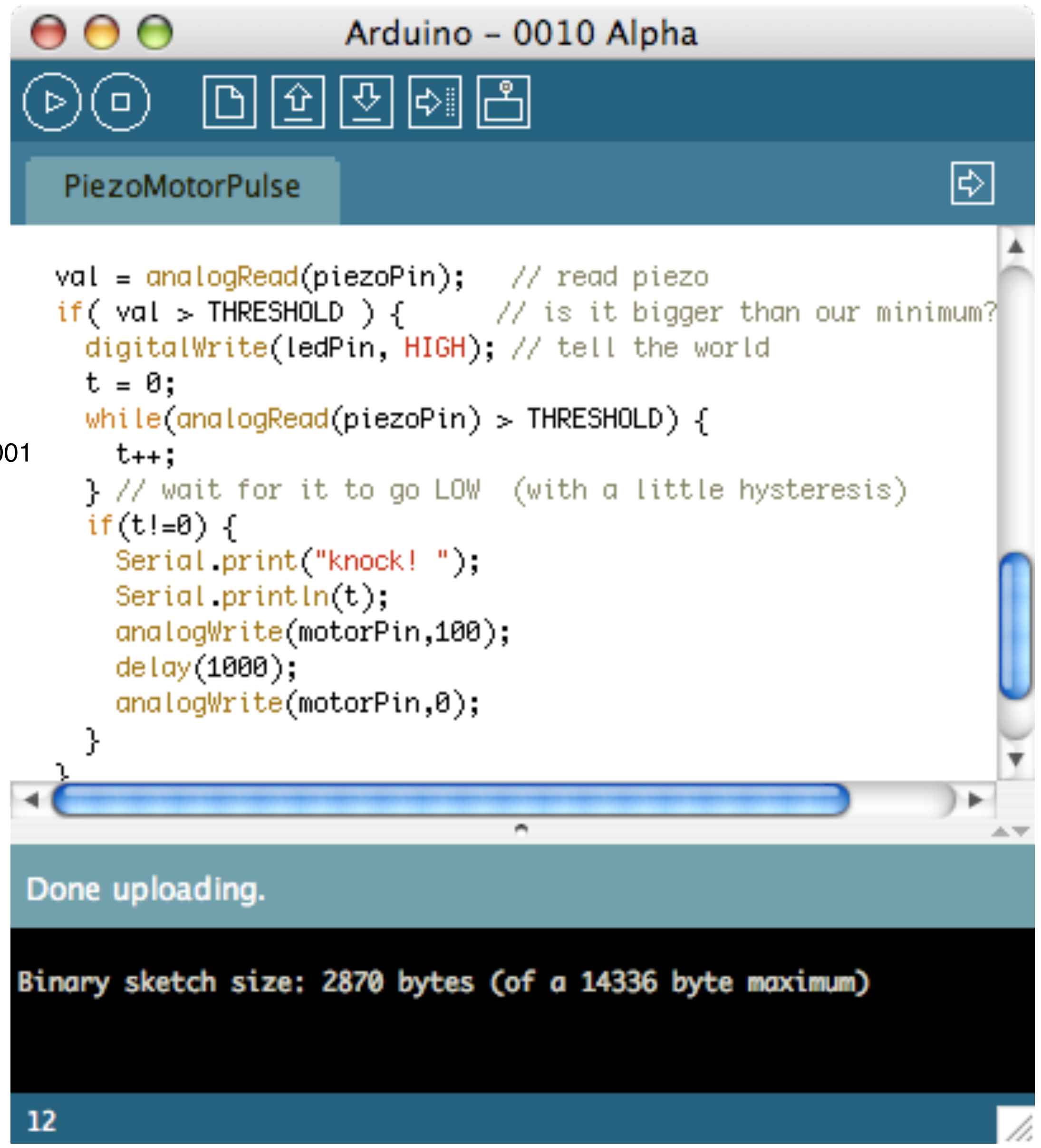
I used this during Halloween a few years ago to trigger scary sounds.

Or Trigger Actuators

"PiezoMotorPulse"



If you still have your motor wired up

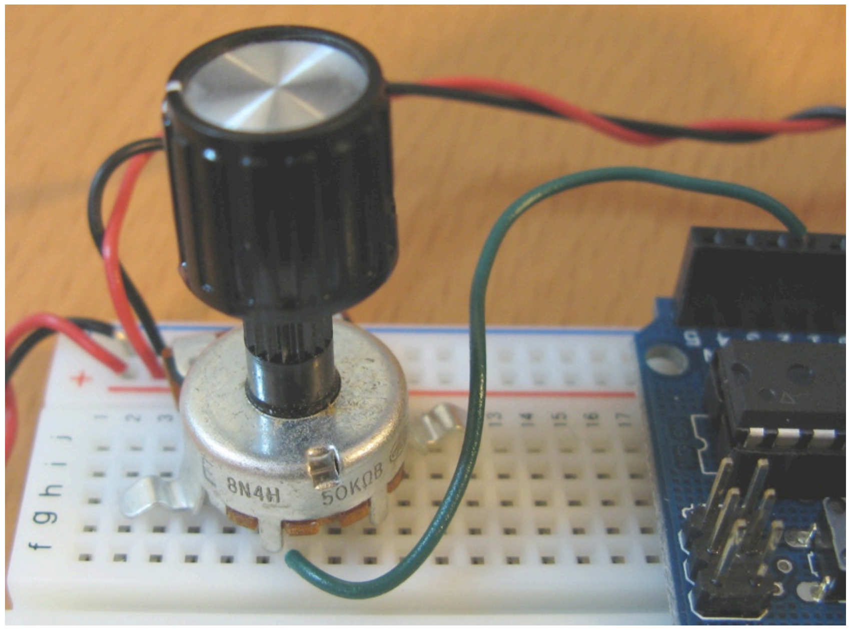


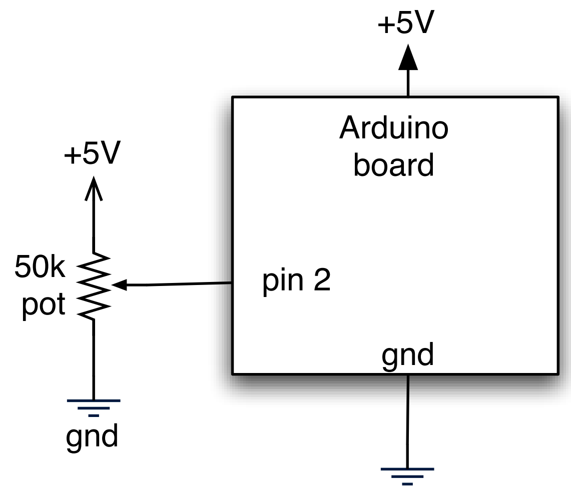
```
val = analogRead(piezoPin);   // read piezo
if( val > THRESHOLD ) {       // is it bigger than our minimum?
  digitalWrite(ledPin, HIGH); // tell the world
  t = 0;
  while(analogRead(piezoPin) > THRESHOLD) {
    t++;
  } // wait for it to go LOW  (with a little hysteresis)
  if(t!=0) {
    Serial.print("knock! ");
    Serial.println(t);
    analogWrite(motorPin,100);
    delay(1000);
    analogWrite(motorPin,0);
  }
}
```

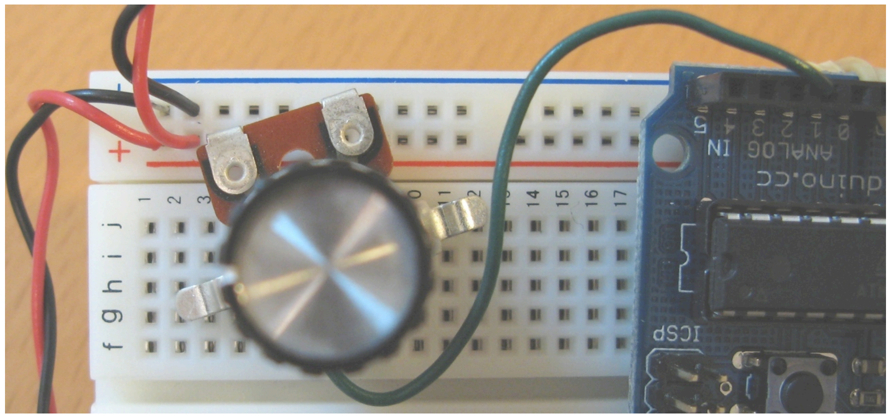
```
Done uploading.

Binary sketch size: 2870 bytes (of a 14336 byte maximum)

12
```

Take a Break

Getting the Board Set Up





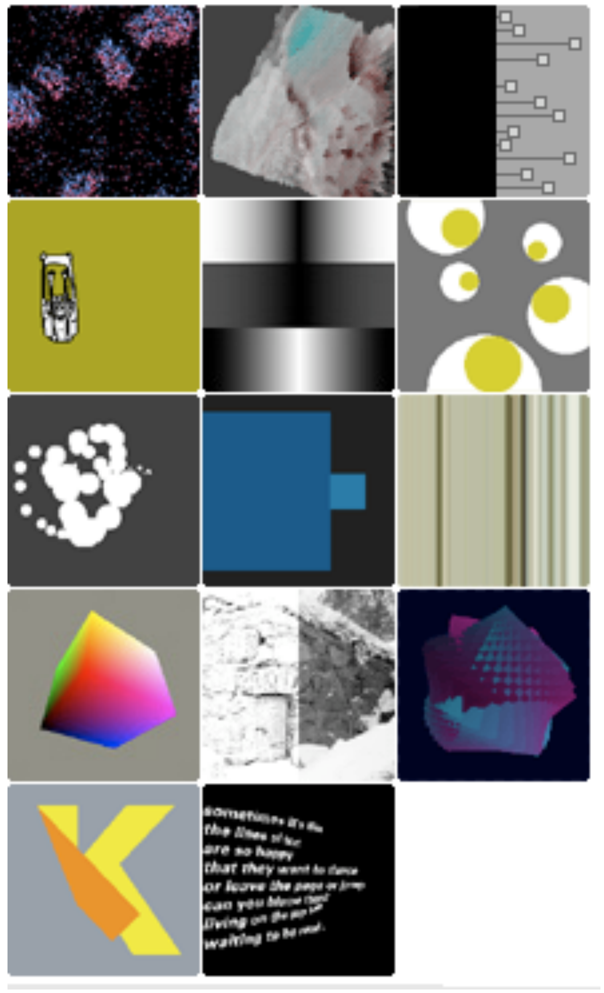


Wire up the potentiometer like from last week

And if you wire up an LED to pin 9, you can try out the "PotDimmer" sketch again to make sure things are wired up right.

Processing

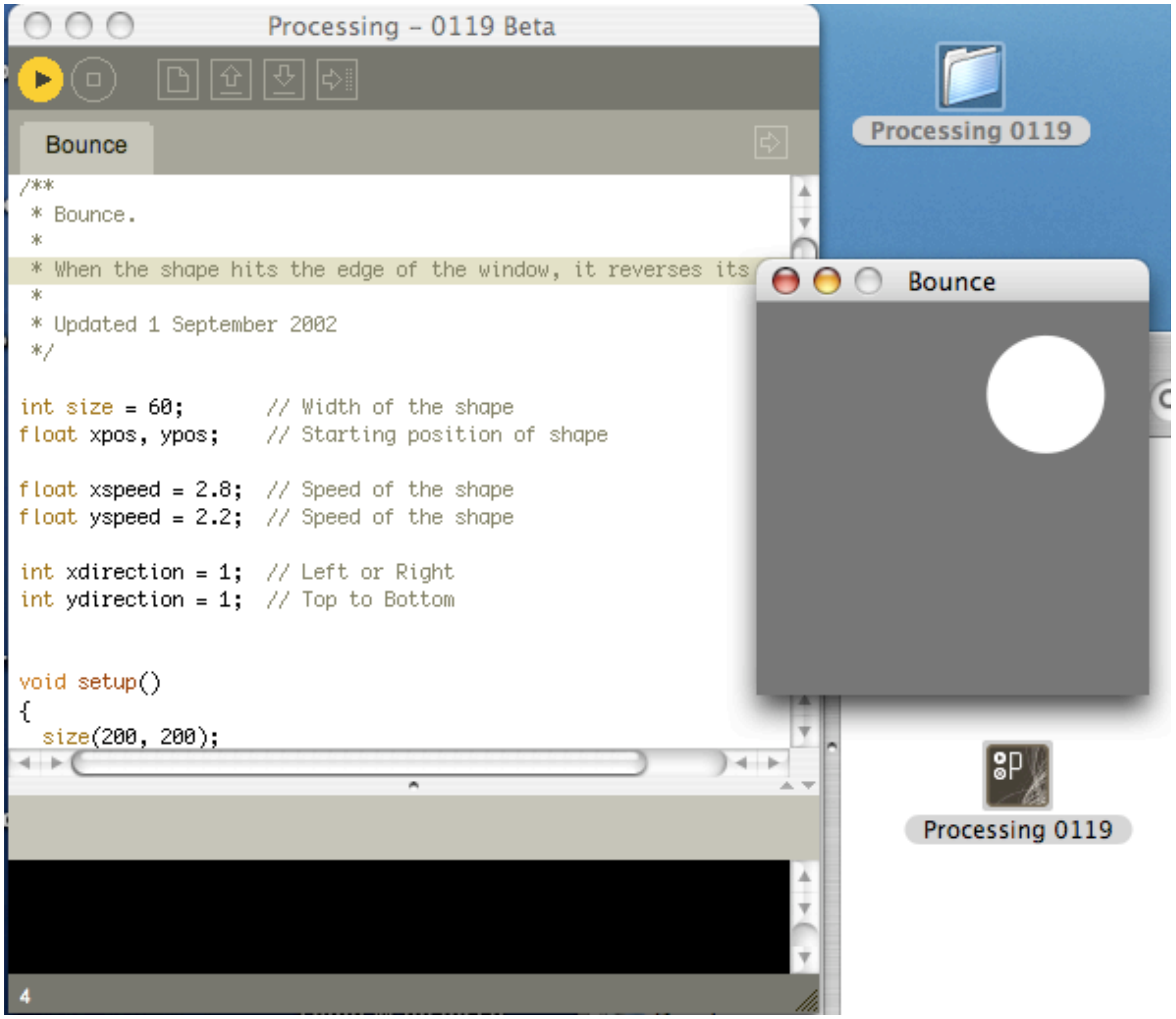
- Processing makes Java programming as fun & easy as Arduino makes AVR programming
- Started as a tool to make generative art
- Is also often used to interface to devices like Arduino
- Think of it as a free Max/MSP



And it’s totally open source like Arduino.
Processing GUI and Arduino GUI are from the same code, which is why it looks & acts similar.

Using Processing

- First, "install" Processing
- Load up "Examples » Topics » Motion » Bounce"
- Press "Run" button
- You just made a Java applet



```
/**
 * Bounce.
 *
 * When the shape hits the edge of the window, it reverses its
 *
 * Updated 1 September 2002
 */

int size = 60;       // Width of the shape
float xpos, ypos;    // Starting position of shape

float xspeed = 2.8;  // Speed of the shape
float yspeed = 2.2;  // Speed of the shape

int xdirection = 1;  // Left or Right
int ydirection = 1;  // Top to Bottom


void setup()
{
  size(200, 200);
```

The Processing application folders are in the handout, no installation is needed.
Also try Examples » Topics » Motion » Collision. It's a lot of fun.
Notice how "Run" launches a new window containing the sketch.
The black area at the bottom is a status window, just like in Arduino.

About Processing

- Processing sketches have very similar structure to Arduino sketches
  - `setup()` – set up sketch, like size, framerate
  - `draw()` – like `loop()`, called repeatedly
- Other functions can exist when using libraries

Processing & Arduino

serial communications

- Processing and Arduino both talk to “serial” devices like the Arduino board
- Only one program per serial port
  - So turn off Arduino’s Serial Monitor when connecting via Processing and vice-versa.
- Processing has a “Serial” library to talk to Arduino. E.g.:

```
port = new Serial(..,“my_port_name”,19200)
port.read(), port.write(), port.available(), etc.
serialEvent() { }
```

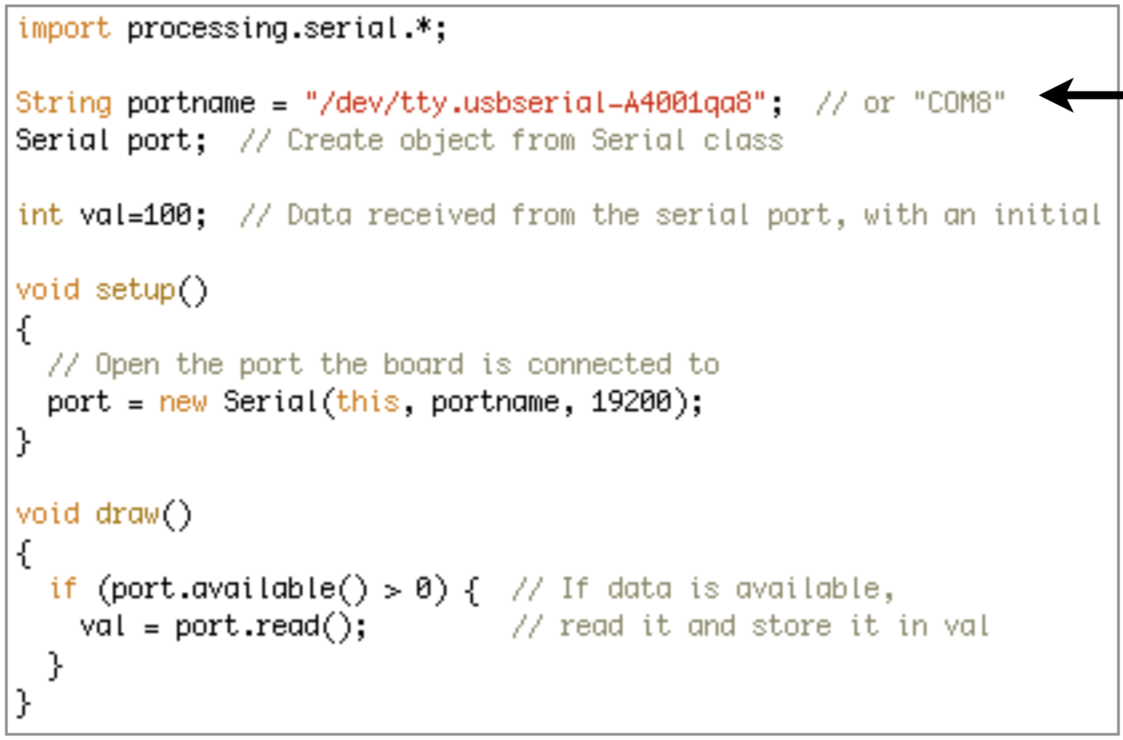
The built-in serial library adds a new function you can use to your sketch: serialEvent()
The serialEvent() function will get called whenever serial data is available.
Or you can poll with port.available().

Processing Serial

common Processing serial use

<u>four steps</u>

1. load library
2. set portname
3. open port
4. read/write port



```
import processing.serial.*;

String portname = "/dev/tty.usbserial-A4001qa8";  // or "COM8"
Serial port;  // Create object from Serial class

int val=100;  // Data received from the serial port, with an initial

void setup()
{
  // Open the port the board is connected to
  port = new Serial(this, portname, 19200);
}

void draw()
{
  if (port.available() > 0) {  // If data is available,
    val = port.read();         // read it and store it in val
  }
}
```

be sure to set to the same as "Serial Port" in Arduino GUI

All you need to do talk to Arduino in Processing.
The import statement says you want to do serial stuff.
The "new Serial" creates a serial port object within Processing
Then you can that object (or used the passed in one) to read from in the "serialEvent()" function

Arduino Talking to Processing

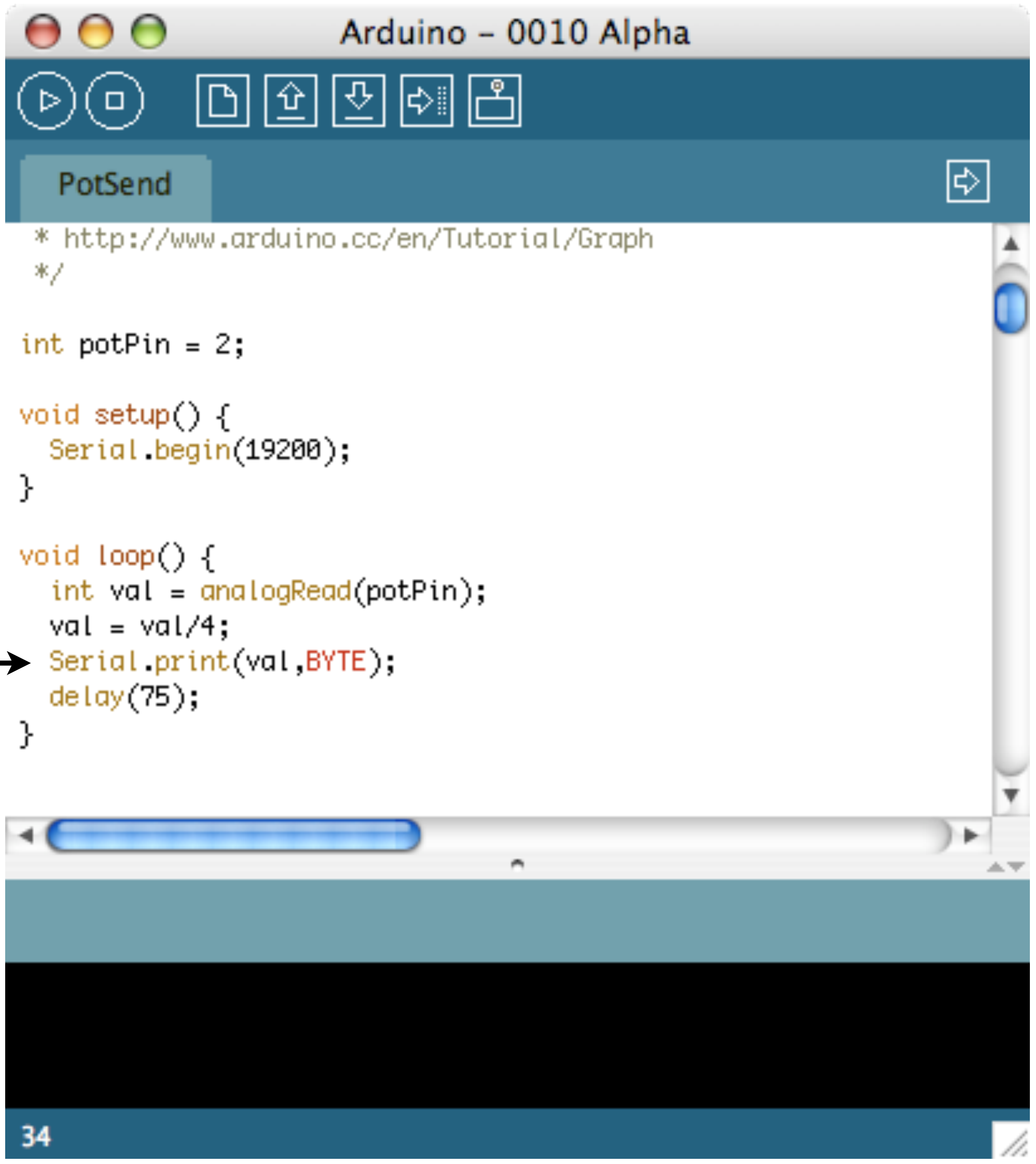
"PotSend"

Read knob,
send it's value

Note: doesn't send the value as ASCII text, but as a binary byte

(BYTEs are easier to parse in Processing than other formats)

You can have 6 knobs total because there are 6 Analog In pins



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

int potPin = 2;

void setup() {
  Serial.begin(19200);
}

void loop() {
  int val = analogRead(potPin);
  val = val/4;
  Serial.print(val,BYTE);
  delay(75);
}
```

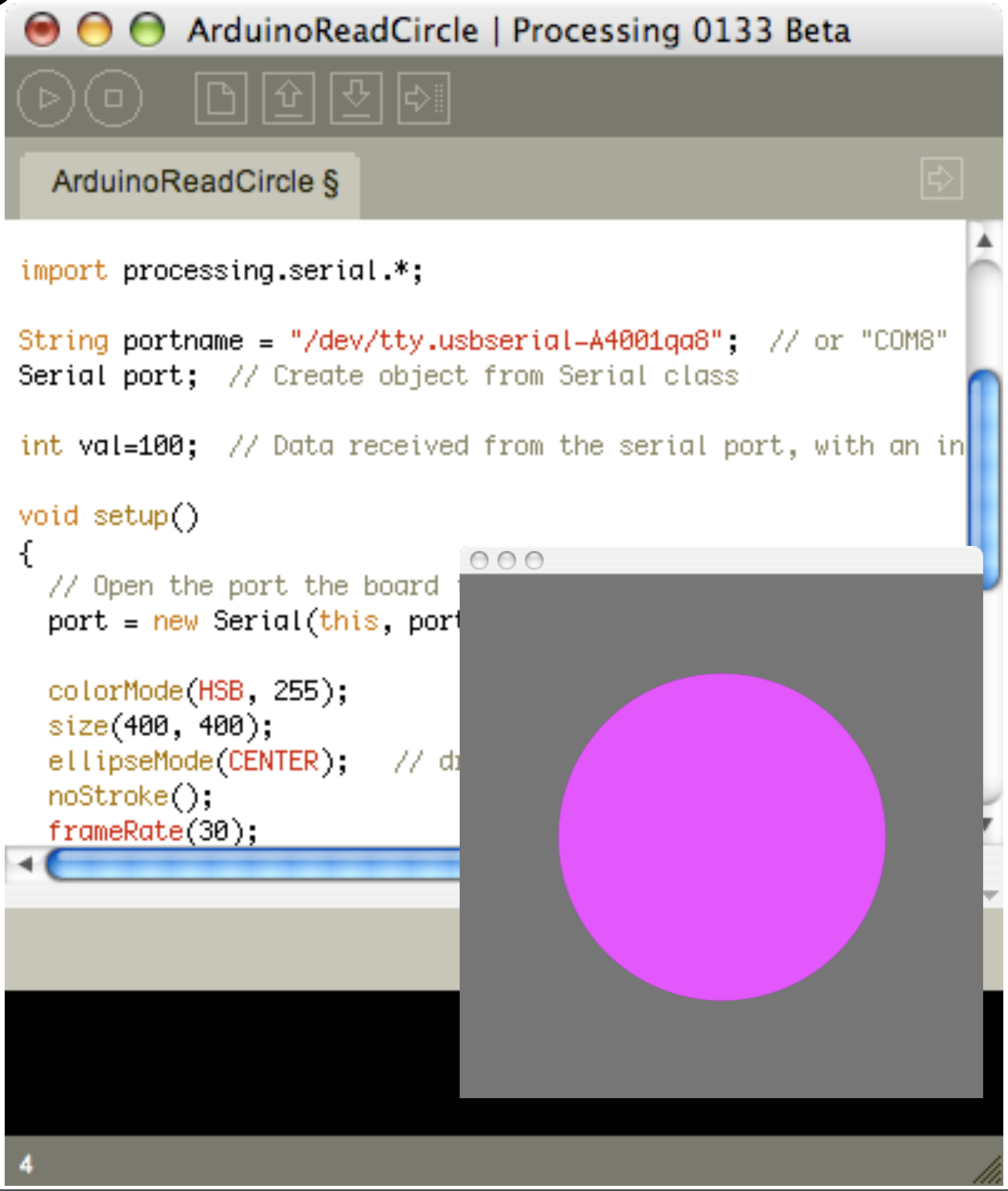
Meanwhile, back in Arduino, load up this sketch we'll use with Processing

Processing + Arduino

“ArduinoReadCircle”

The pot controls the hue of the onscreen circle

Arduino is running “PotSend”, repeatedly sending a number from 0-255 indicating knob position



```
import processing.serial.*;

String portname = "/dev/tty.usbserial-A4001qa8";  // or "COM8"
Serial port;  // Create object from Serial class

int val=100;  // Data received from the serial port, with an in

void setup()
{
  // Open the port the board
  port = new Serial(this, port

  colorMode(HSB, 255);
  size(400, 400);
  ellipseMode(CENTER);   // d
  noStroke();
  frameRate(30);
```

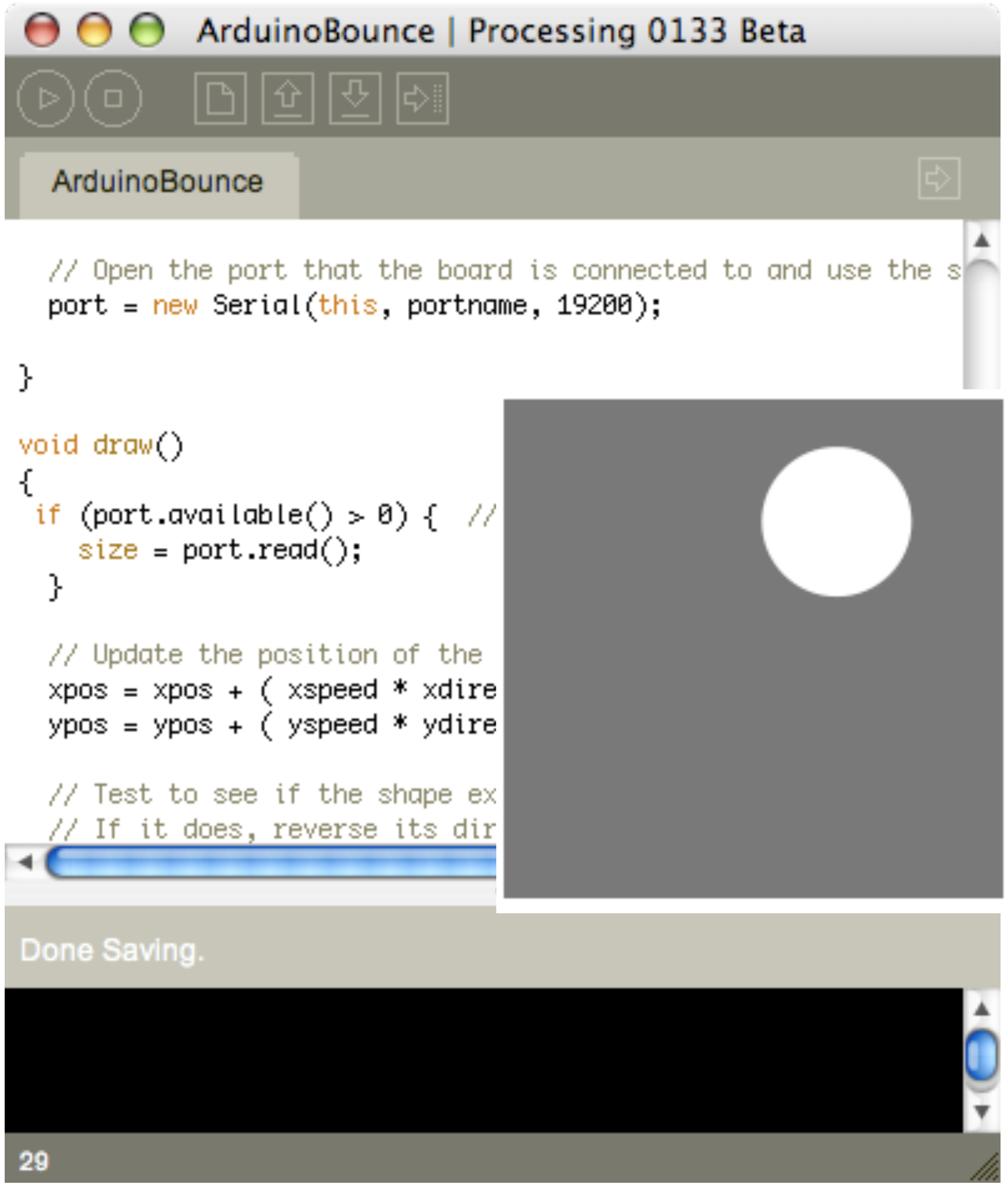
This sketch is in the handout, under “processing_sketches”.

Another One

“`ArduinoBounce`”

Every time a byte is received via the serial port, it alters the size of the ball to match.

Comment out the “`background(102)`” line to get trails
Uncomment the “`fill()`” line to get color trails



```
  // Open the port that the board is connected to and use the s
  port = new Serial(this, portname, 19200);

}

void draw()
{
  if (port.available() > 0) {  //
    size = port.read();
  }

  // Update the position of the
  xpos = xpos + ( xspeed * xdire
  ypos = ypos + ( yspeed * ydire

  // Test to see if the shape ex
  // If it does, reverse its dir
```

```
Done Saving.
```

```
29
```

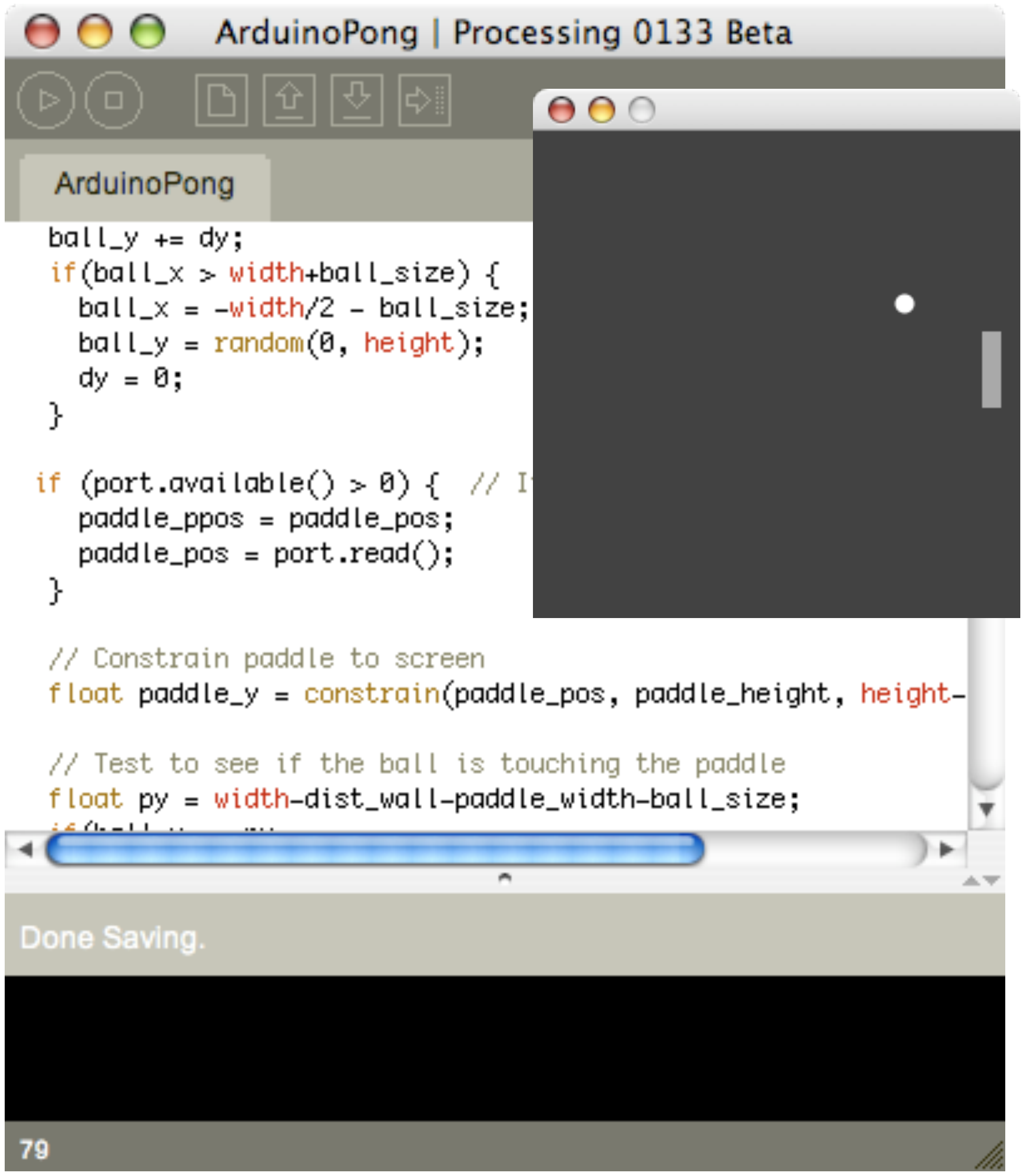
Notice the bug that happens when you change the size near a border.

And Another One

"ArduinoPong"

The basics of a pong game. The pot controls paddle position

Add another pot and a little more game logic and you have a 2-player game



```
  ball_y += dy;
  if(ball_x > width+ball_size) {
    ball_x = -width/2 - ball_size;
    ball_y = random(0, height);
    dy = 0;
  }

 if (port.available() > 0) {  // I
    paddle_ppos = paddle_pos;
    paddle_pos = port.read();
  }

  // Constrain paddle to screen
  float paddle_y = constrain(paddle_pos, paddle_height, height-

  // Test to see if the ball is touching the paddle
  float py = width-dist_wall-paddle_width-ball_size;
```

```
Done Saving.
```

79

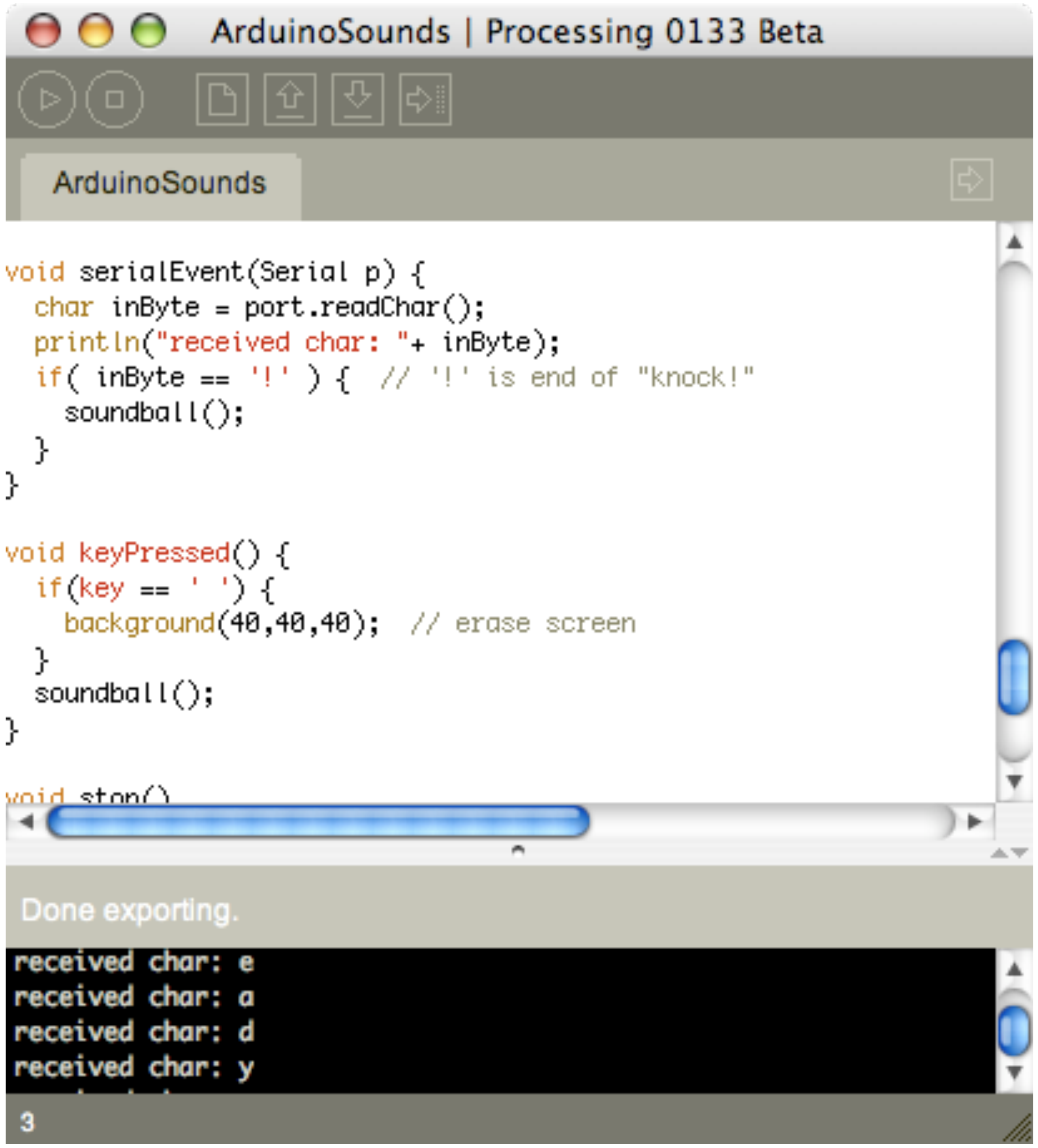
These are all very minorly-modified examples of standard Processing sketches.

Triggering Sounds

"ArduinoSounds"

Every time the piezo is knocked... a sound plays and a red disc appears onscreen

This sketch needs the "minim" sound library.



```
void serialEvent(Serial p) {
  char inByte = port.readChar();
  println("received char: "+ inByte);
  if( inByte == '!' ) {  // '!' is end of "knock!"
    soundball();
  }
}

void keyPressed() {
  if(key == ' ') {
    background(40,40,40);  // erase screen
  }
  soundball();
}

void stop()
```

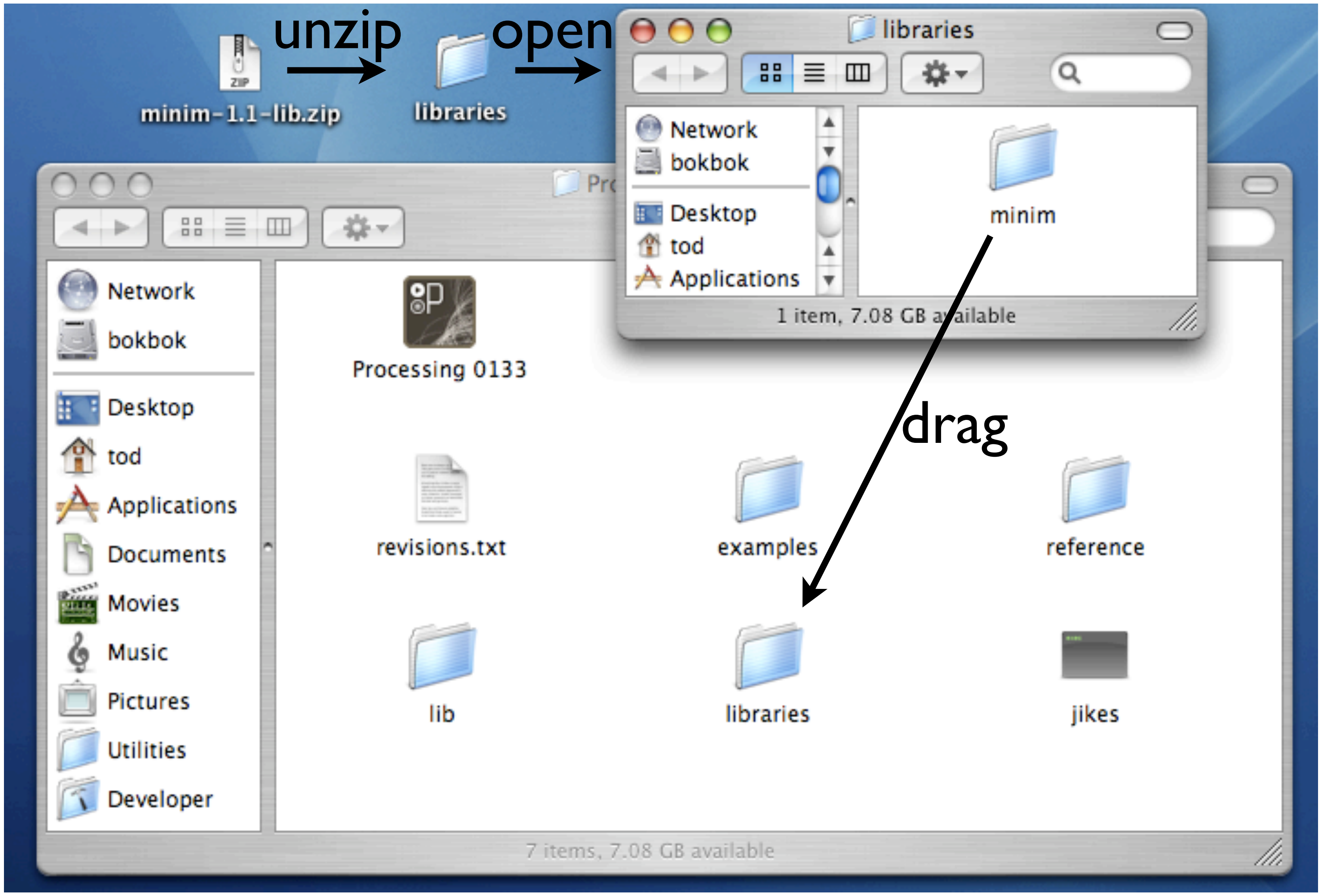
```
Done exporting.
received char: e
received char: a
received char: d
received char: y
```

You can add your own sounds (WAV or MP3)
Hook a piezo up to your front door, and plug your computer into your stereo.
Every time someone knocks on your door, a sound is played: a custom doorbell!

The zipfile for the "minim" library is in the handout, called "minim-1.1-lib.zip".
Unzip it and place the "minim" folder in the "Processing 0133/libraries" folder.

Adding Processing Libraries

Unzip, drop into "libraries" folder



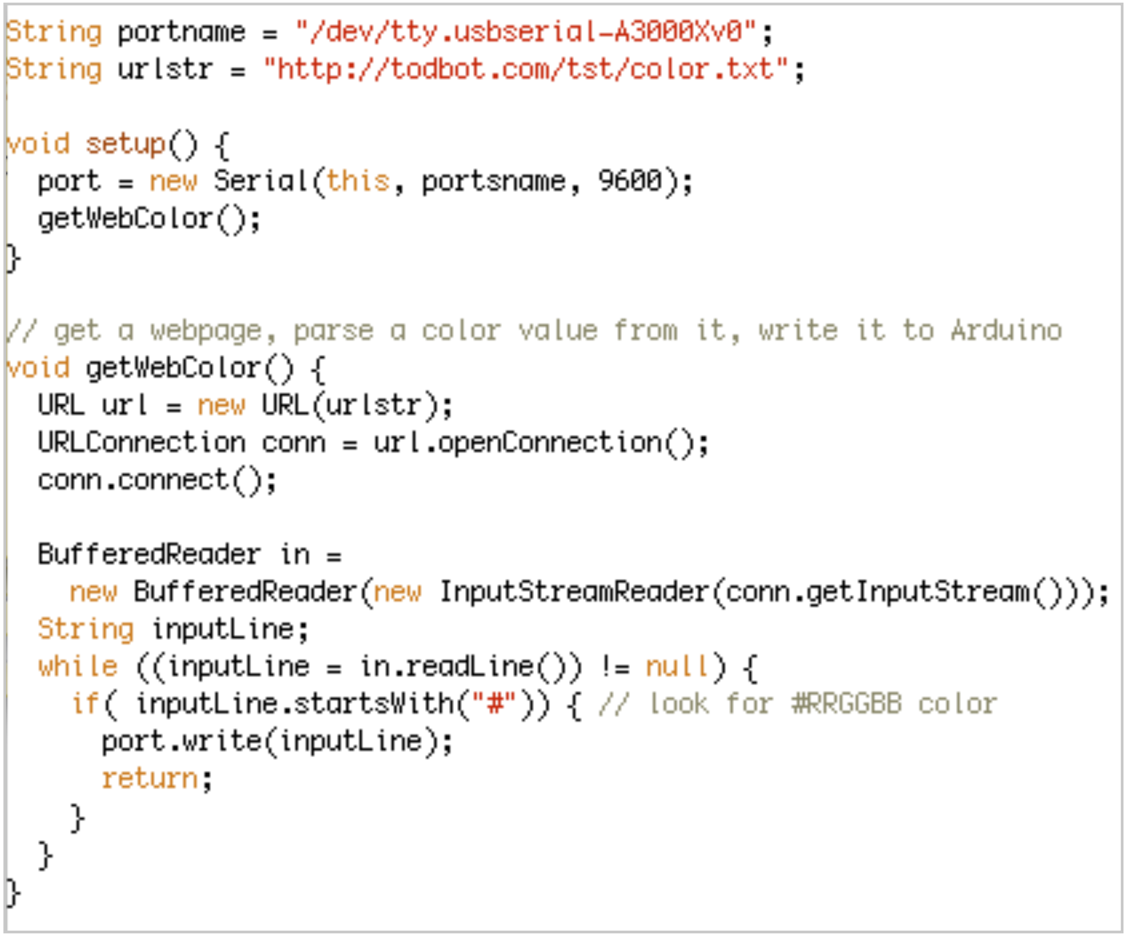
Same for Windows and Mac OS X. Mac OS X shown.

Processing to Arduino

real quick

“http_rgb_led”

Fetch a web page, get a color value from it, send the color to Arduino with RGB LED



```
String portname = "/dev/tty.usbserial-A3000Xv0";
String urlstr = "http://todbot.com/tst/color.txt";

void setup() {
  port = new Serial(this, portsname, 9600);
  getWebColor();
}

// get a webpage, parse a color value from it, write it to Arduino
void getWebColor() {
  URL url = new URL(urlstr);
  URLConnection conn = url.openConnection();
  conn.connect();

  BufferedReader in =
    new BufferedReader(new InputStreamReader(conn.getInputStream()));
  String inputLine;
  while ((inputLine = in.readLine()) != null) {
    if( inputLine.startsWith("#")) { // look for #RRGGBB color
      port.write(inputLine);
      return;
    }
  }
}
```

This is not to build, just quickly cover. It’s not in the handout, but, full details at: http://todbot.com/blog/2006/10/23/diy-ambient-orb-with-arduino-update/

Going Further

- DC motors
  - Get some gearhead motors for serious torque or slower RPM
  - Use Lego, Erector, Meccano to build mechanical linkages for motors
  - Oh and you can now build a robot

Going Further

- Transistor switches
  - Anytime you need to switch a signal more powerful than what Arduino can use
  - These transistors switch up to 1 amp of DC voltage. For AC household currents, use transistor to switch a relay
  - Can control just about anything in your house

Going Further

- Processing & Serial communications
  - Processing can talk to the Net. It’s an Internet-to-Arduino gateway
  - It can also talk to many computer peripherals, like video cameras
  - Maybe: Arduino controls the motors, laptop controls the cameras of your robot

END Class 3

http://todbot.com/blog/bionicarduino/

Tod E. Kurt

tod@todbot.com

Feel free to email me if you have any questions.