Spooky Projects

Introduction to Microcontrollers with Arduino

Class 4

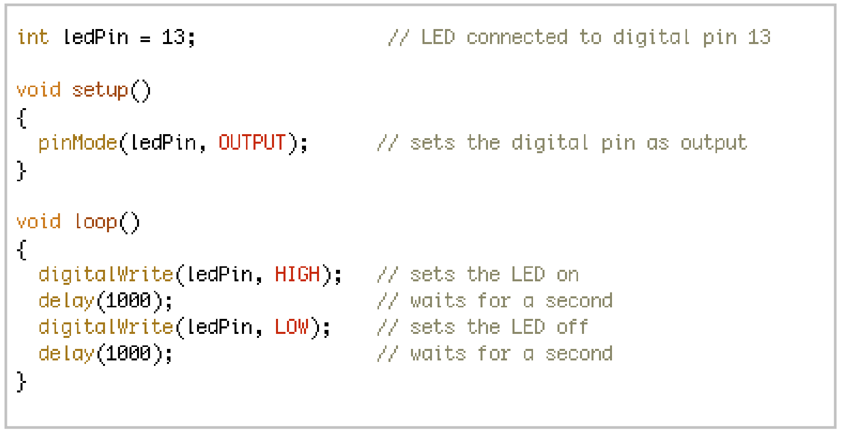
28 Oct 2006 - machineproject - Tod E. Kurt

What’s For Today

- Switches without Resistors
- All about piezos
- Building a melody player
- Using piezos as pressure & knock sensors
- Using Processing with Arduino
- Stand-alone Arduino

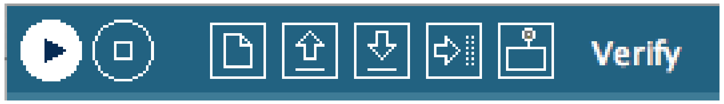
Recap: Programming

Edit

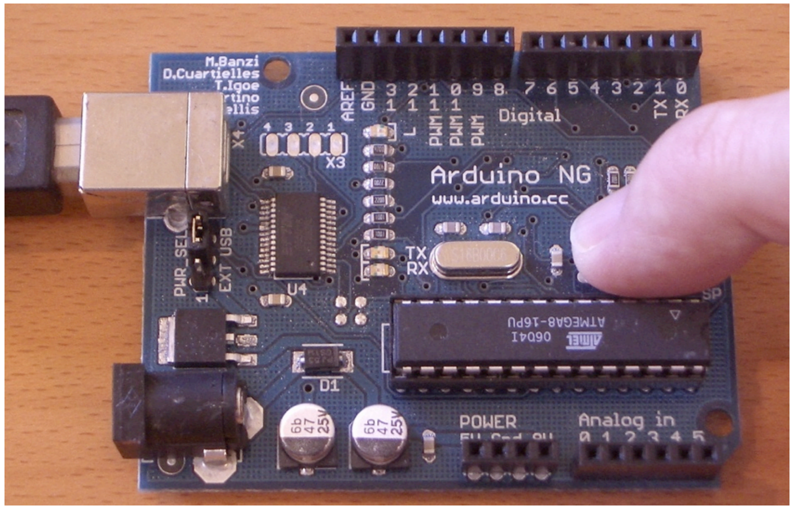


```
int ledPin = 13;                 // LED connected to digital pin 13

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
}
```

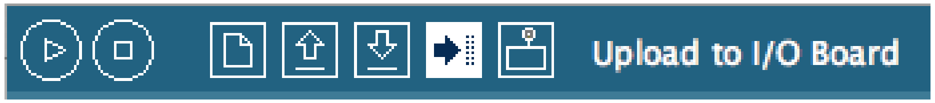
Compile



Reset



Upload

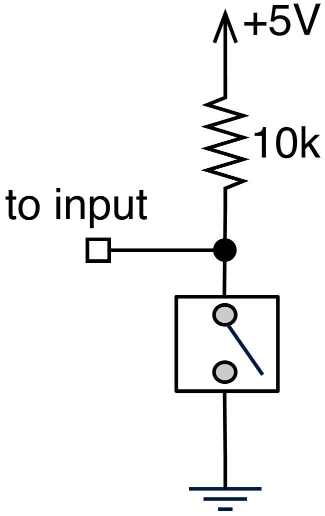


Like always, just make sure. Make “led_blink” come alive again. Do it. Trust me.

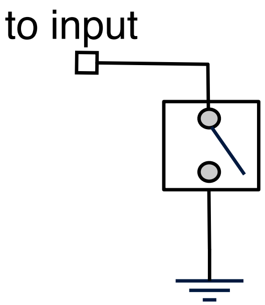
Switches w/o Resistors

AVR chip has internal "pull-up" resistors

Instead of this:



You can just do this:

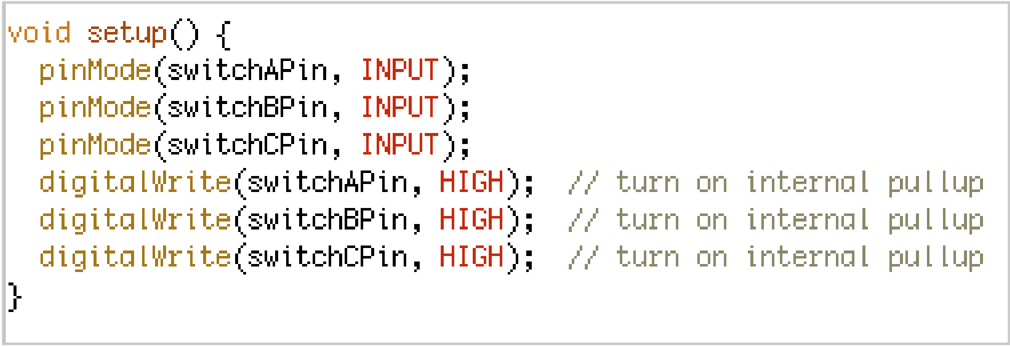


But how do you turn on these internal pull-ups?

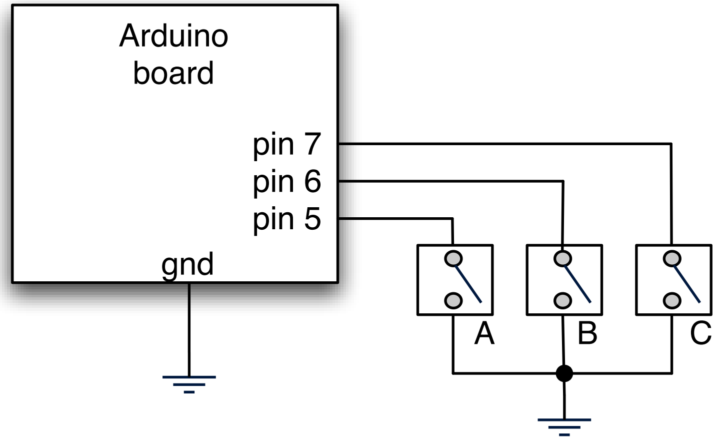
This is sort of an aside, but it saves a lot of wiring.

Switches w/o Resistors

Answer: use `digitalWrite(pin,HIGH)` on the input



```
void setup() {
  pinMode(switchAPin, INPUT);
  pinMode(switchBPin, INPUT);
  pinMode(switchCPin, INPUT);
  digitalWrite(switchAPin, HIGH);  // turn on internal pullup
  digitalWrite(switchBPin, HIGH);  // turn on internal pullup
  digitalWrite(switchCPin, HIGH);  // turn on internal pullup
}
```



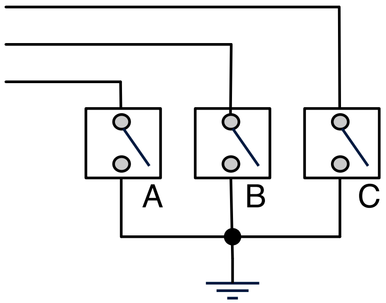
Seems a little counter-intuitive,
think of it as setting the default value of the input

but note, it doesn't work the other way: you can't set it to LOW then wire the switch to +5V.

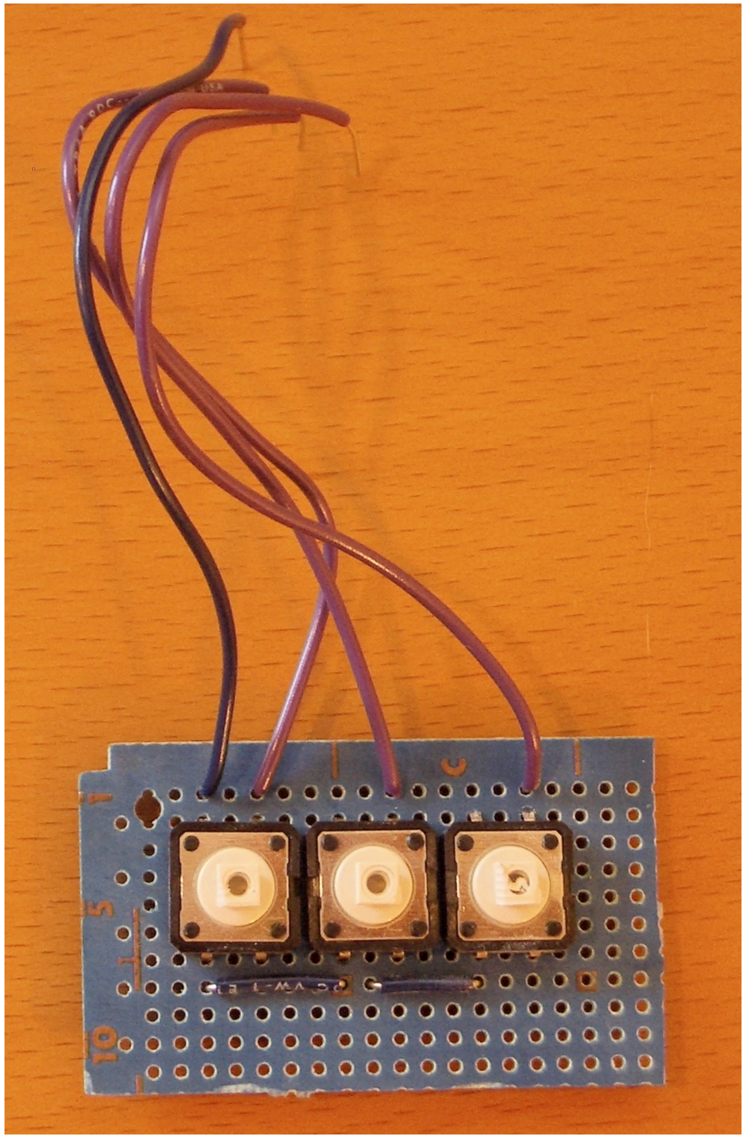
Switches w/o Resistors

Can make a button box easily
if no resistors are needed

Plugs right into Arduino board



=

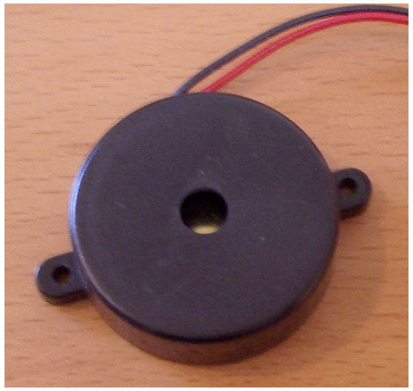


Piezoelectrics

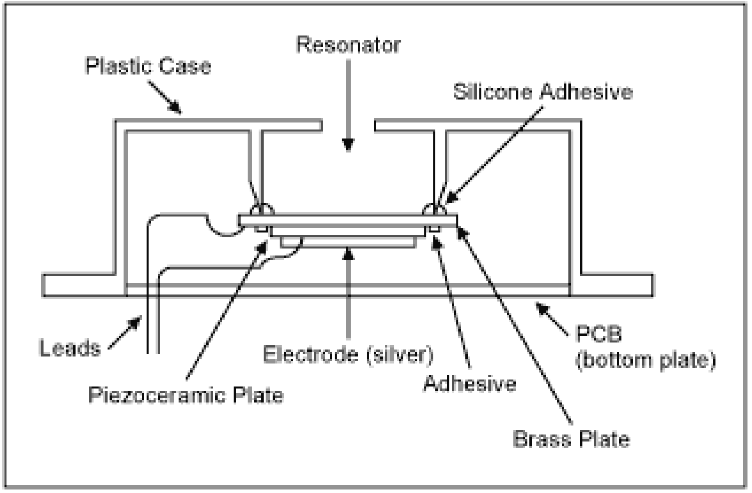
- Big word – *piezein* is greek for "squeeze"
- Some crystals, when squeezed, make a spark
- Turns out the process goes the other way too
- Spark a quartz crystal, and it flexes
- Piezo buzzers use this to make sound
  (flex something back and forth, it moves air)

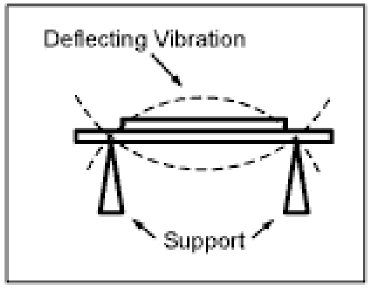
Piezo buzzers don't have quartz crystals, but instead a kind of ceramic that also exhibits piezoelectric properties.
I pronounce it "pie-zoh". Or sometimes "pee-ay-zoh".

Piezo Buzzers



- Two wires, red & black.
  Polarity matters: black=ground
- Apply an oscillating voltage to make a noise
- The buzzer case supports the piezo element and has resonant cavity for sound





Oscillating voltage alternately squeezes and releases the piezo element.
Must apply flucuating voltage, a steady HIGH or LOW won't work.

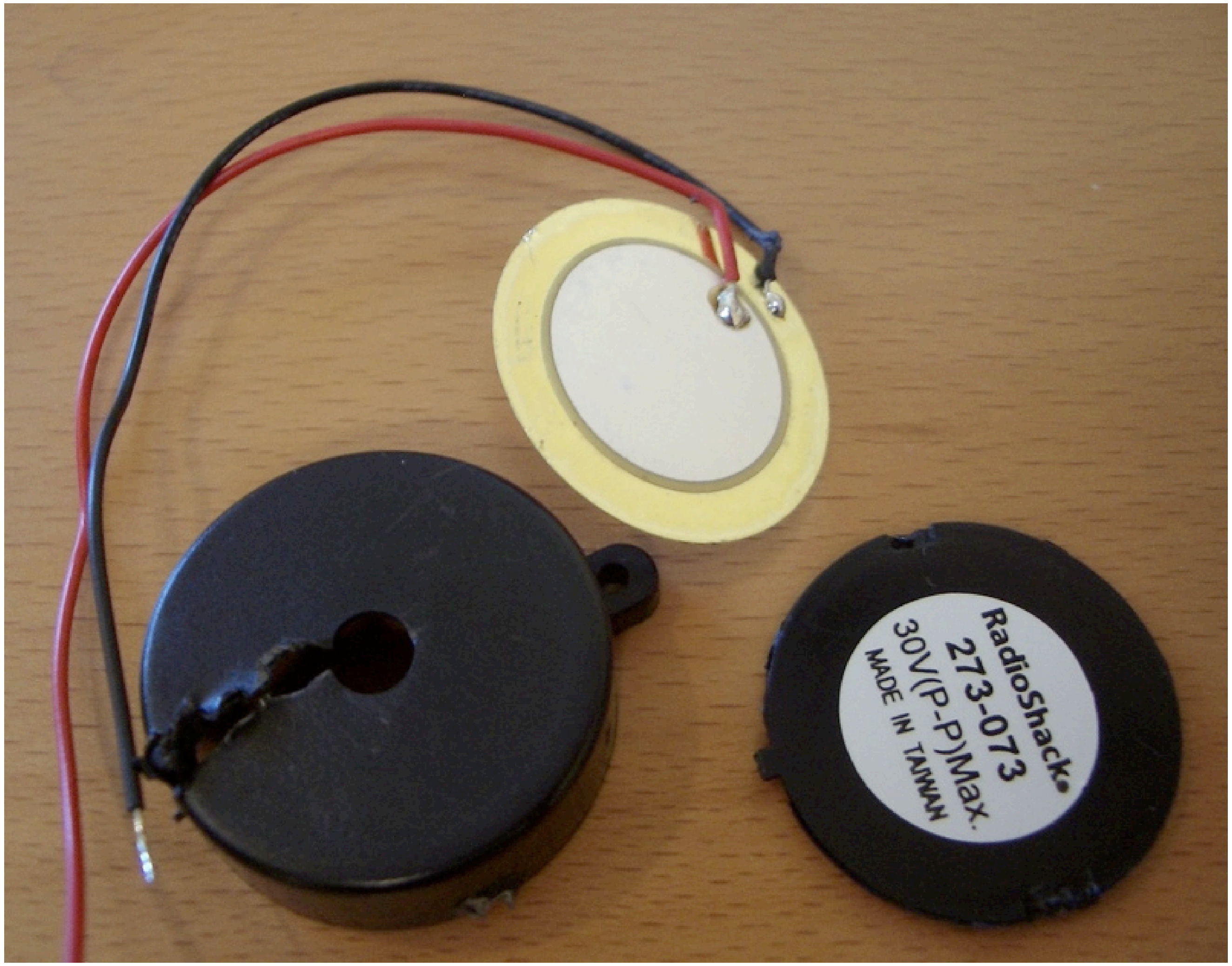
diagrams from: http://www.maxim-ic.com/appnotes.cfm/appnote_number/988

What's in a Piezo Buzzer?

You can get at the piezo element pretty easily.

Be careful not to crack the white disc that is the actual piezo

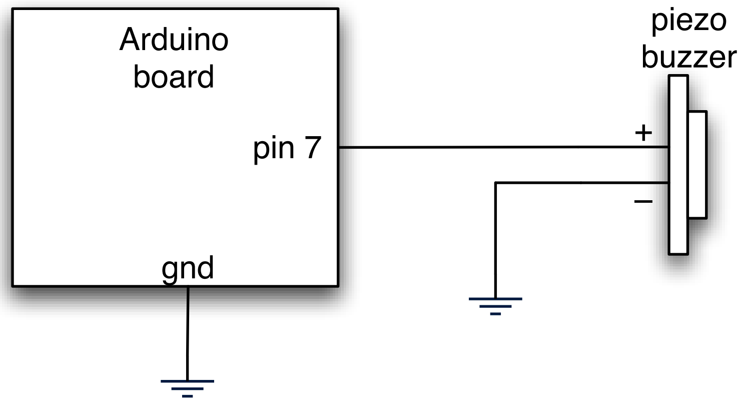
Only take it out of its case to use it as a sensor

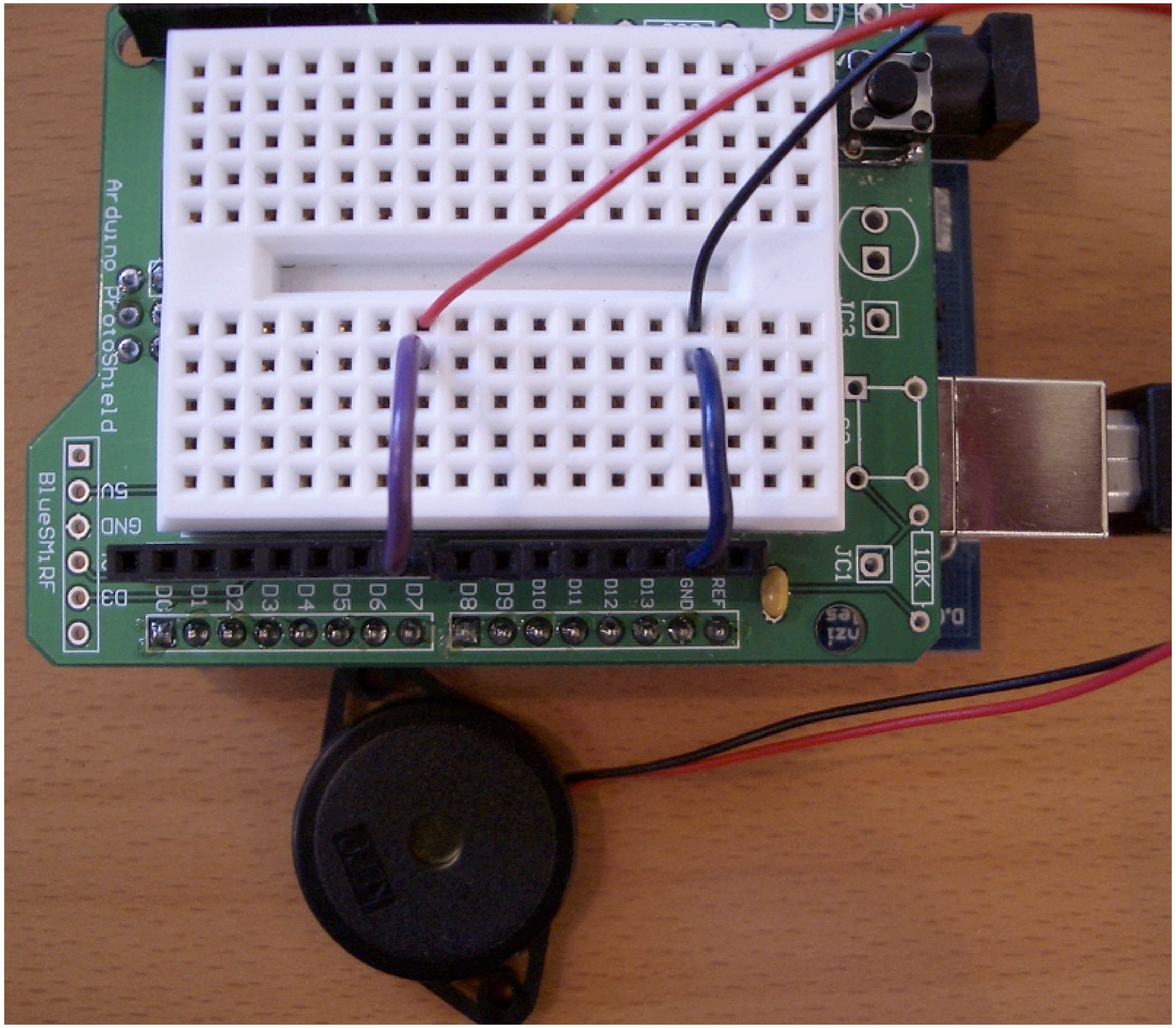


another $1.99 I won't be getting back from Radio Shack

Of course, you usually destroy the enclosure to get at the element.
And it's the enclosure that has the proper support and resonant cavity to make a loud sound

Piezo Buzzer





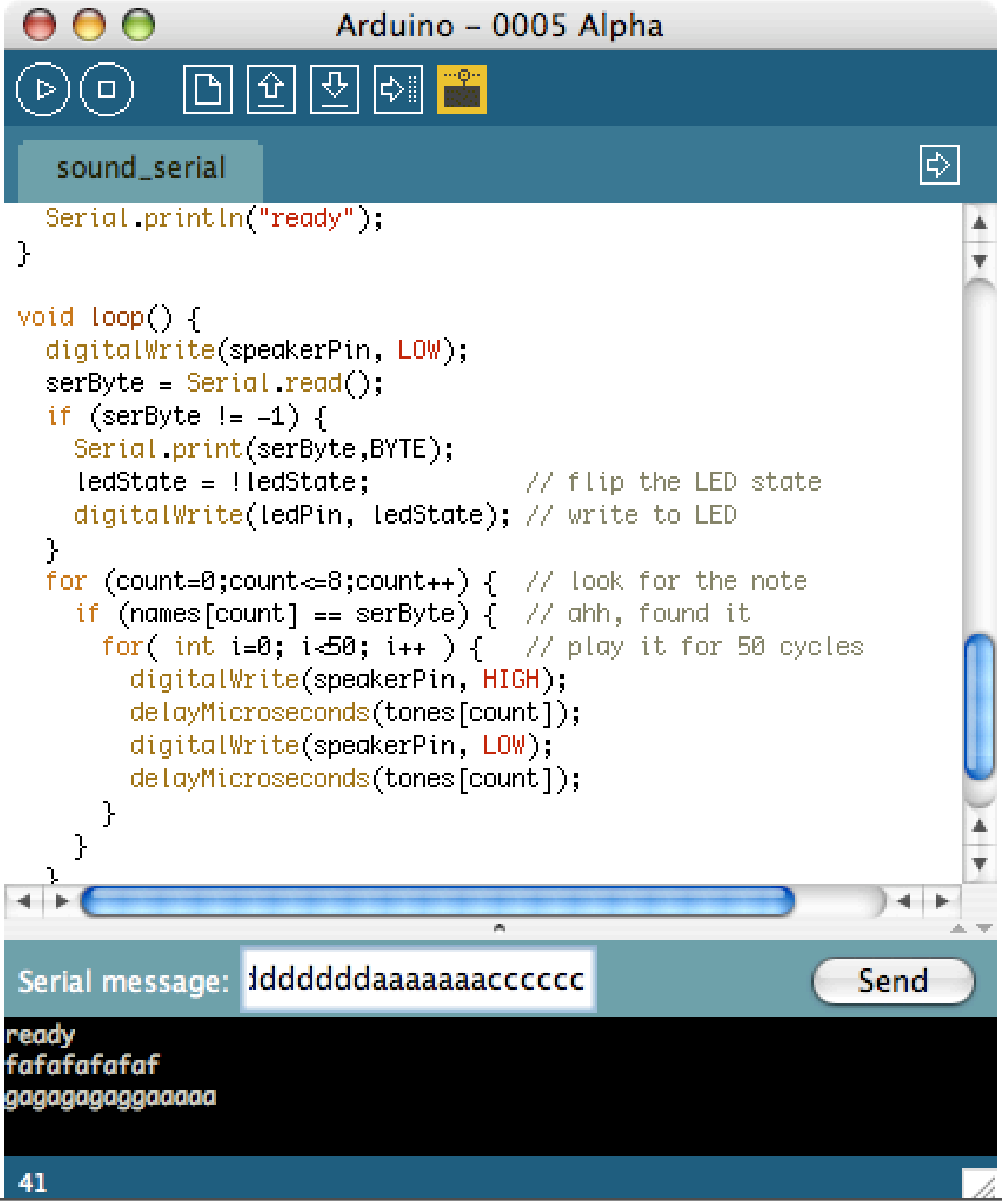
Piezo leads are very thin. The breadboard holes grab them better than the header sockets, which is why the jumper leads are used.

Play a Melody

“`sound_serial`”

Play the piezo beeper with the Serial Monitor

Type multiple letters from “`cdefgabC`” to make melodies



```
  Serial.println("ready");
}

void loop() {
  digitalWrite(speakerPin, LOW);
  serByte = Serial.read();
  if (serByte != -1) {
    Serial.print(serByte,BYTE);
    ledState = !ledState;           // flip the LED state
    digitalWrite(ledPin, ledState); // write to LED
  }
  for (count=0;count<=8;count++) {  // look for the note
    if (names[count] == serByte) {  // ahh, found it
      for( int i=0; i<50; i++ ) {   // play it for 50 cycles
        digitalWrite(speakerPin, HIGH);
        delayMicroseconds(tones[count]);
        digitalWrite(speakerPin, LOW);
        delayMicroseconds(tones[count]);
      }
    }
  }
```

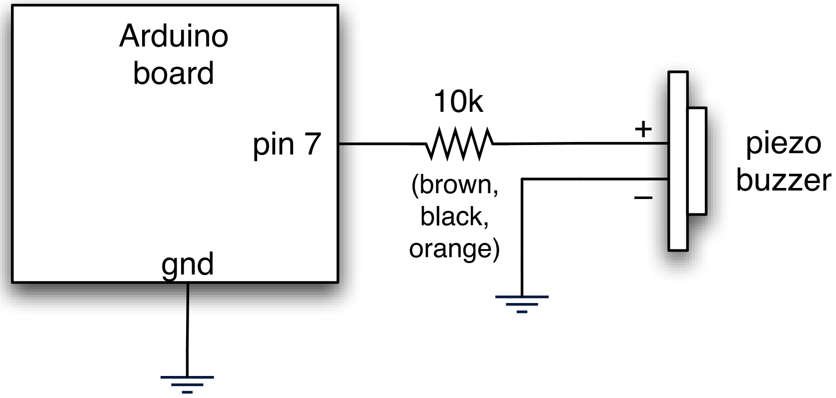
Serial message: dddddddaaaaaaacccccc

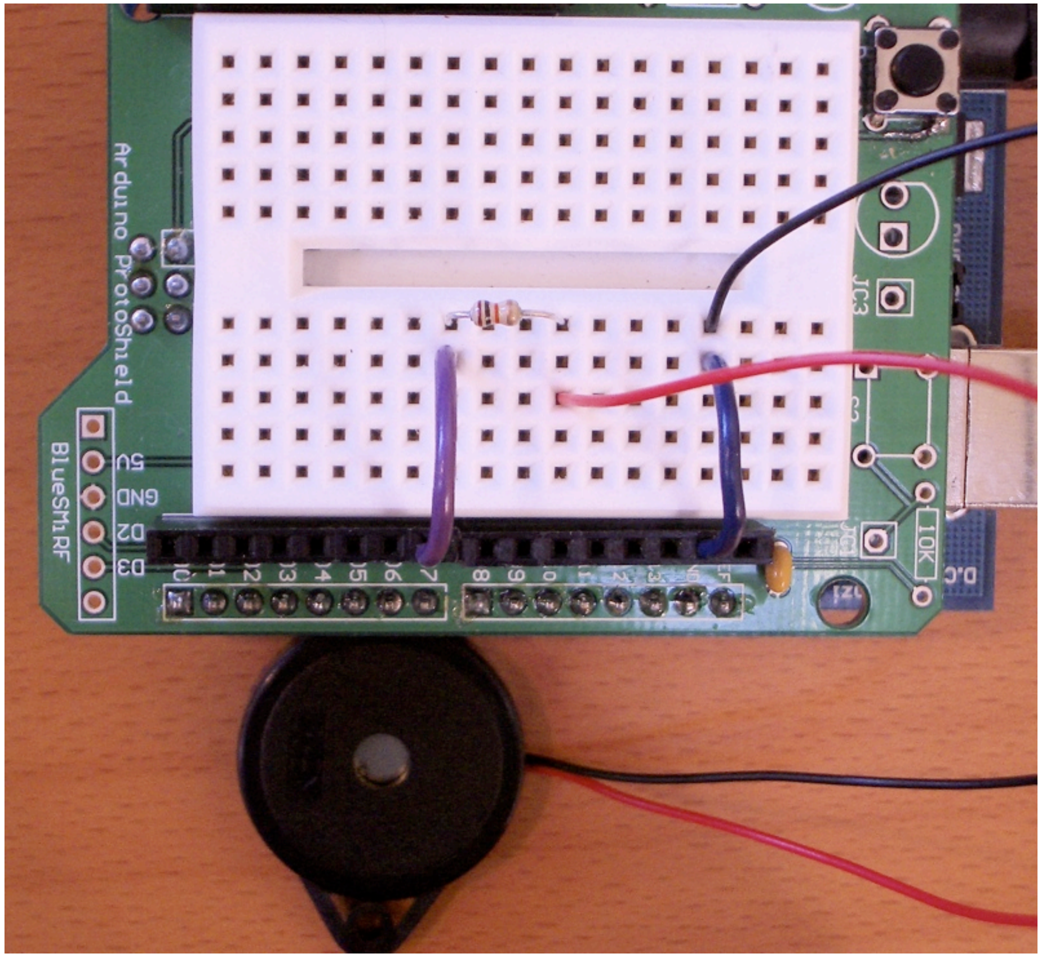
```
ready
fafafafafaf
gagagagaggaaaaa
```

This sketch is in the handout, and is based on “Examples/pwm_sound/keyboard_serial”
Notice the problem with this sketch?
Different notes play for different amounts of time.
50 cycles of low C isn’t the same amount of time as 50 cycles of high B

Making it Quieter

Easiest way: add a resistor



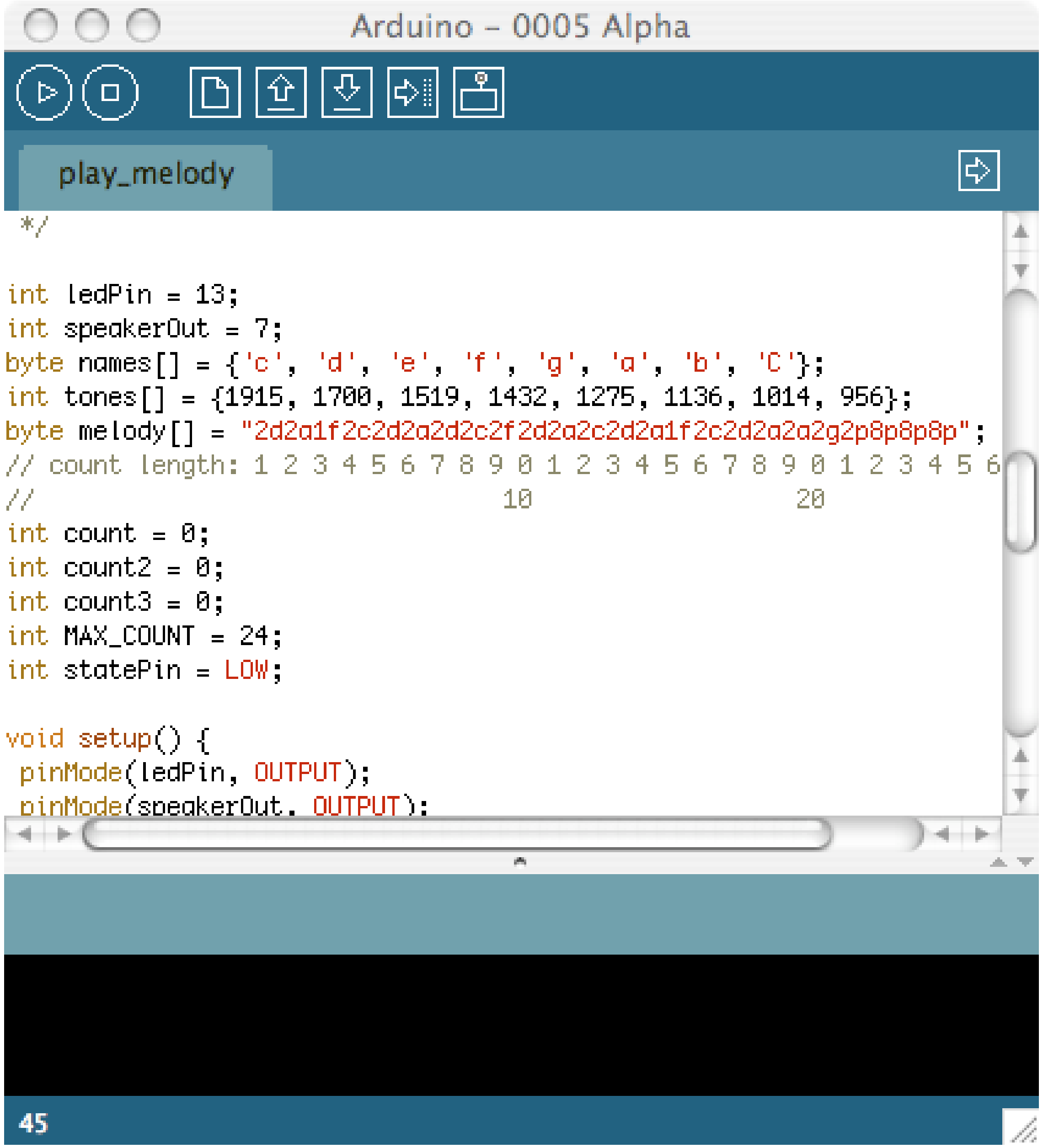


Like most things in electronics, if you want less of something, add a resistor.
A better value would probably be 1k, but we don’t have that on hand.
This may not seem important now, but wait for the next project.

Play a Stored Melody

“play_melody”

Plays a melody stored in the Arduino



```
 */

int ledPin = 13;
int speakerOut = 7;
byte names[] = {'c', 'd', 'e', 'f', 'g', 'a', 'b', 'C'};
int tones[] = {1915, 1700, 1519, 1432, 1275, 1136, 1014, 956};
byte melody[] = "2d2a1f2c2d2a2d2c2f2d2a2c2d2a1f2c2d2a2a2g2p8p8p8p";
// count length: 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6
//                                10                  20
int count = 0;
int count2 = 0;
int count3 = 0;
int MAX_COUNT = 24;
int statePin = LOW;

void setup() {
 pinMode(ledPin, OUTPUT);
 pinMode(speakerOut, OUTPUT);
```

This is in the handout, but is also in “Examples/pwm_sound/play_melody” (pin changed)
Melody definition is sort of like the old cell ringtone style
Melody playing logic is hard to follow.

Make a Theremin

"ooo-weee-ooooo"

The original spooky sound machine

Works by measuring your body's electric field

No touching needed!

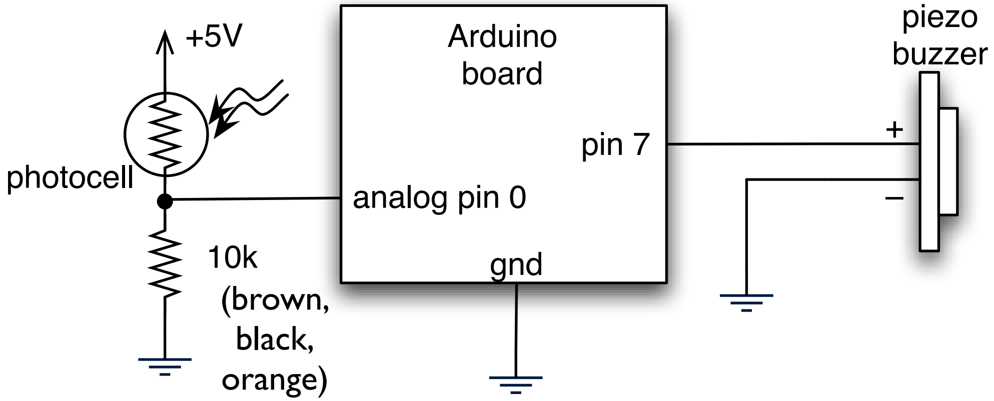
We'll use light in lieu of RF

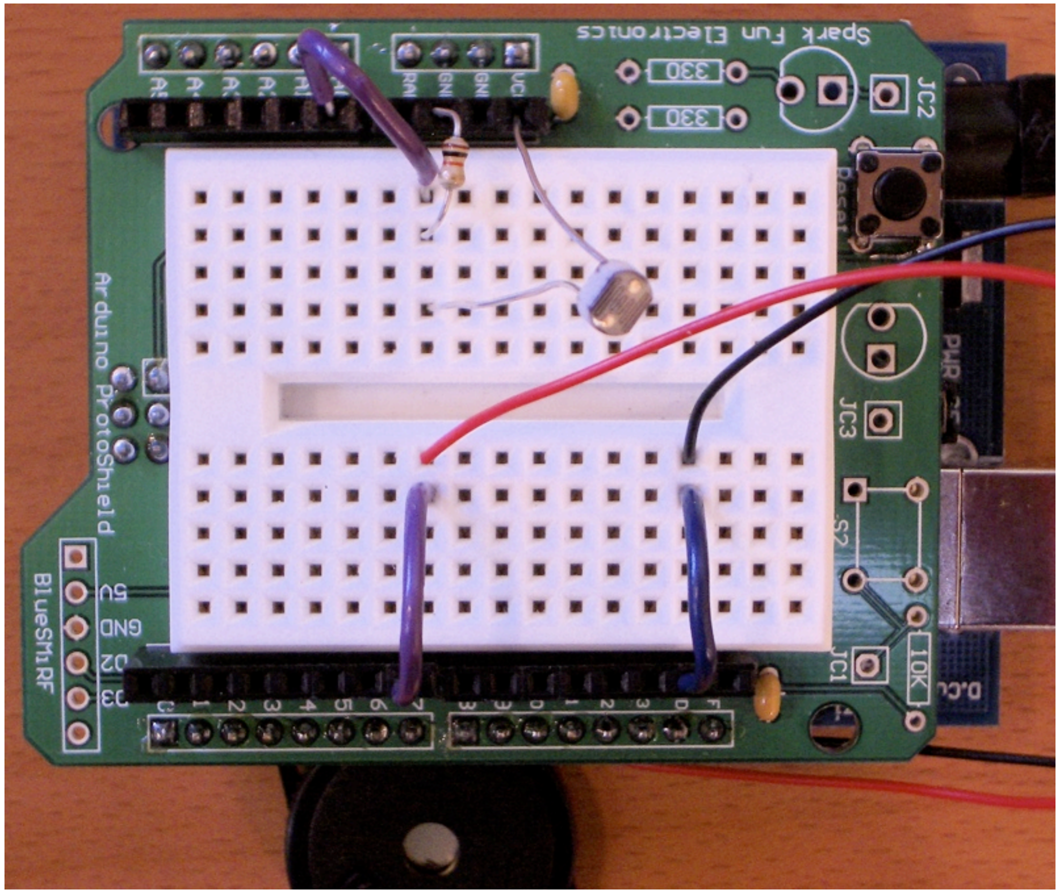
Leon Theremin

As heard on Star Trek, Beach Boys, horror movies, Mars Attacks!, and bad New Age songs.
Works sorta like those touch switches, but no touching here.
That is, your body becomes a variable capacitor.

Make a Theremin

Take photocell circuit from before, bolt it on





This is a light-to-sound converter, if you will.

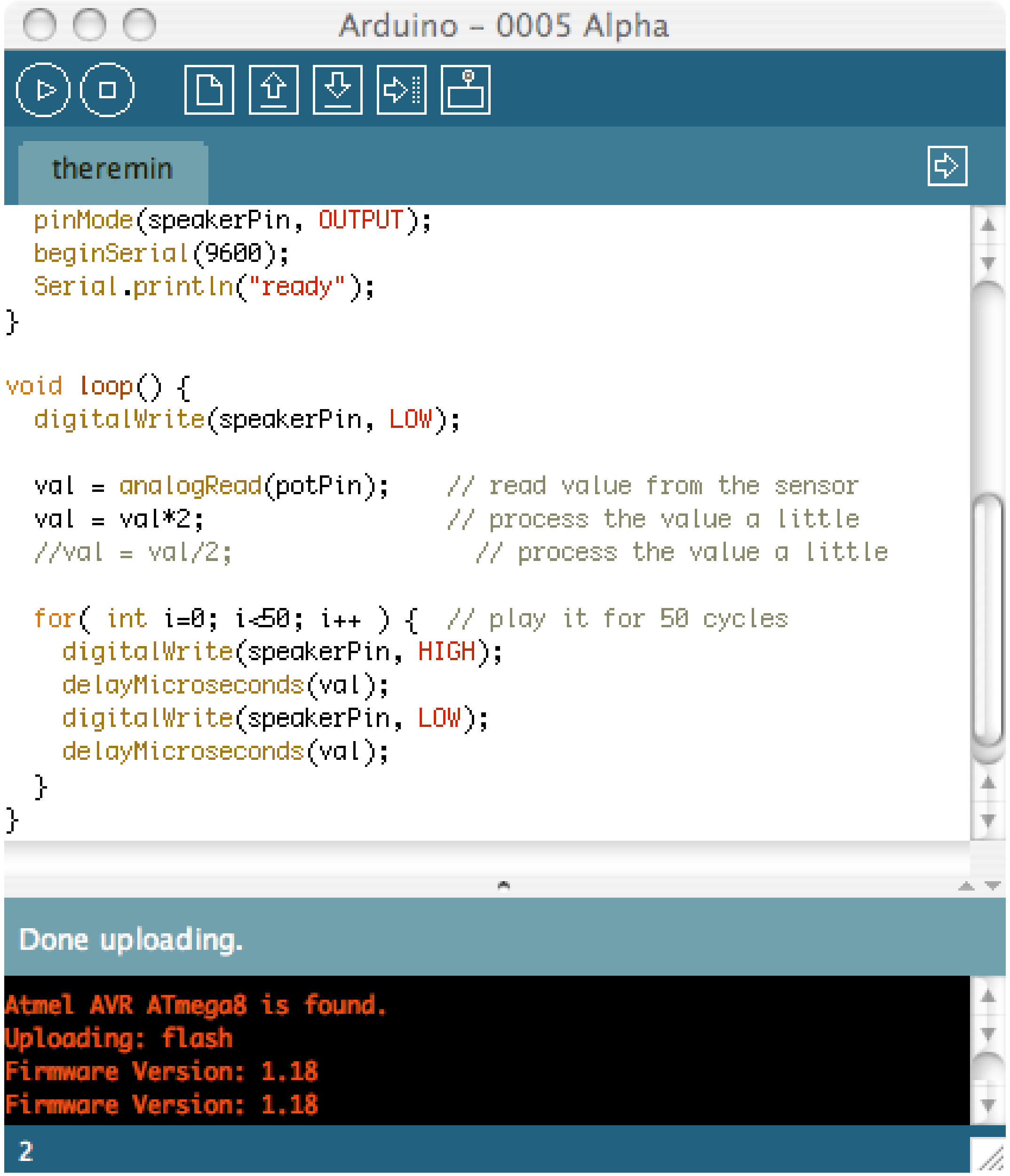
Make a Theremin

“theremin”

Move hand over photocell to change pitch

Play with val processing & cycles count to alter sensitivity, pitch and timbre

This is *frequency modulation,* since you’re changing the frequency



```
  pinMode(speakerPin, OUTPUT);
  beginSerial(9600);
  Serial.println("ready");
}

void loop() {
  digitalWrite(speakerPin, LOW);

  val = analogRead(potPin);    // read value from the sensor
  val = val*2;                 // process the value a little
  //val = val/2;                 // process the value a little

  for( int i=0; i<50; i++ ) {  // play it for 50 cycles
    digitalWrite(speakerPin, HIGH);
    delayMicroseconds(val);
    digitalWrite(speakerPin, LOW);
    delayMicroseconds(val);
  }
}
```

Done uploading.

```
Atmel AVR ATmega8 is found.
Uploading: flash
Firmware Version: 1.18
Firmware Version: 1.18
```

2

Okay so maybe it sounds more like a bad video game than a spooky movie
The glitchy sound is cause because of the time it takes to read the sensor
There are ways around such stuff, but requires more complex programming using timers & interrupts
The sound can get annoying quick

Piezo Buzzer as Sensor

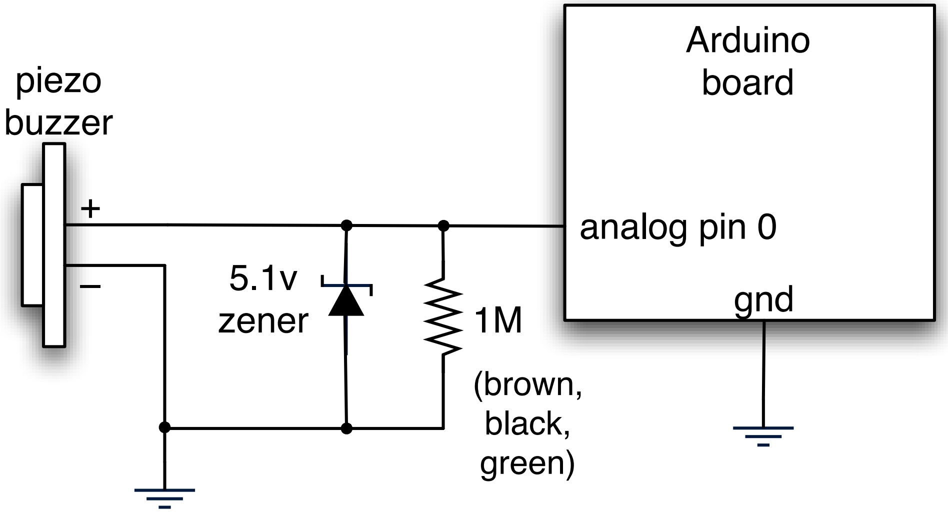
- Piezo buzzers exhibit the *reverse* piezoelectric effect.
- The normal piezoelectric effect is generating electricity from squeezing a crystal.
- Can get several thousand volts, makes a spark
- You probably have seen a big example of this already:

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 Read

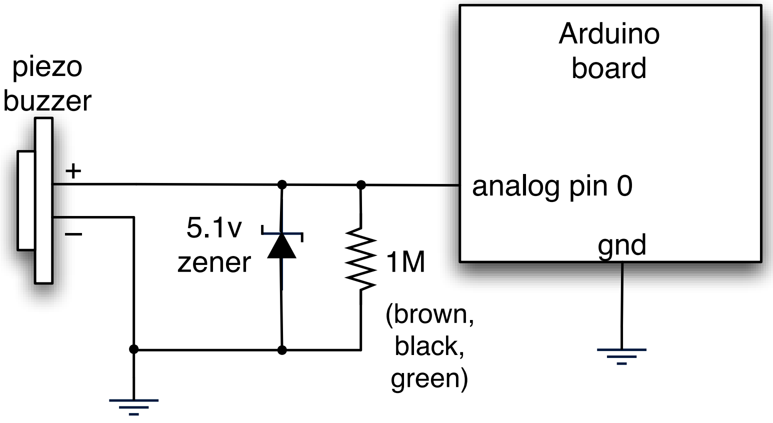
- To read a piezo you can just hook it into an analog input, but:
- You need to drain off any voltage with a resistor, or it just builds up
- You should have a protection diode to limit big voltages, else fry your inputs

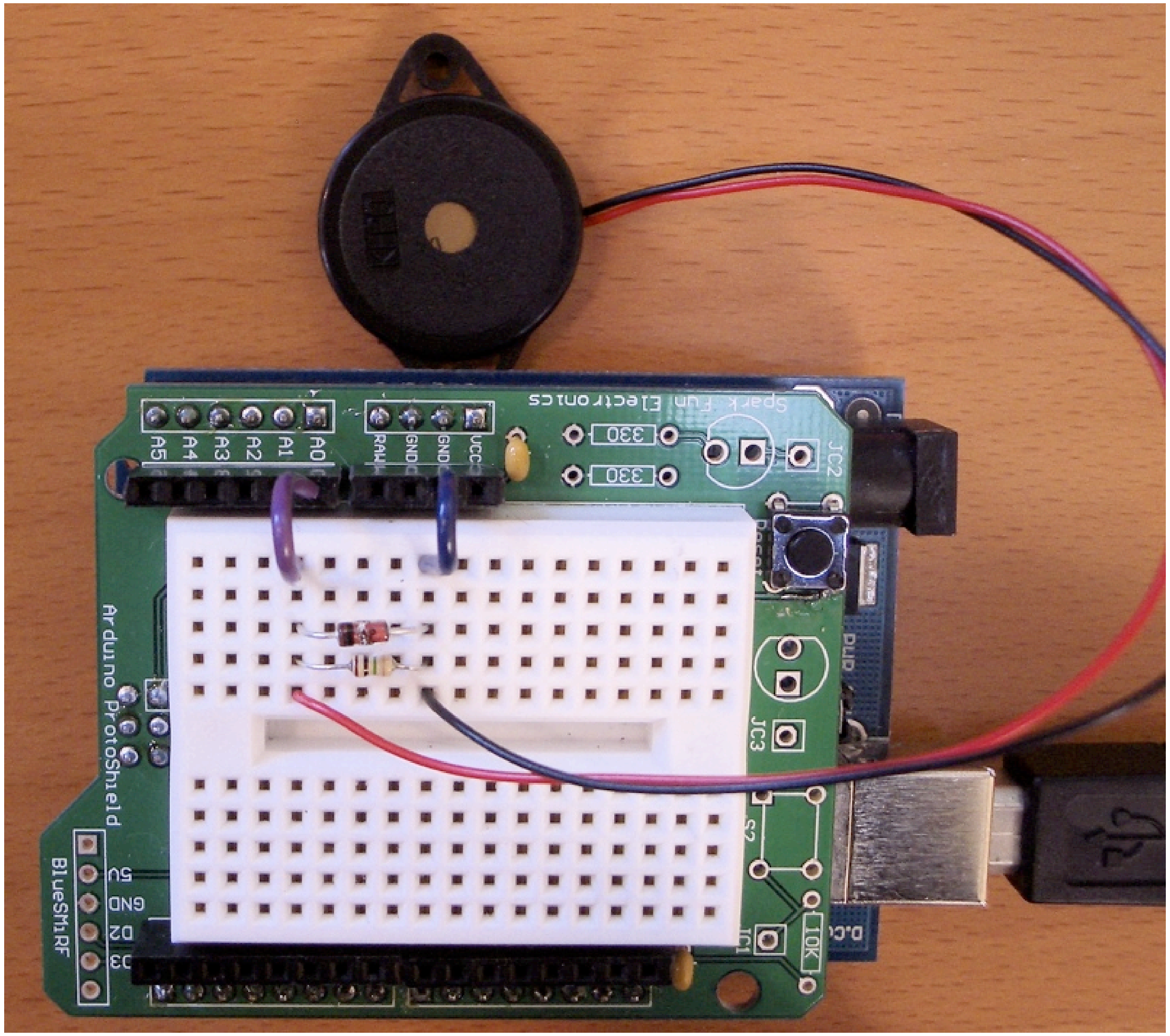


piezo input schematic

Note polarity of piezo still matters.
The protection diode is a special kind of diode called a “zener diode”. It acts invisible until the voltage gets over its designed value (5.1 volts in this case), then it acts like a short circuit.

Piezo Read





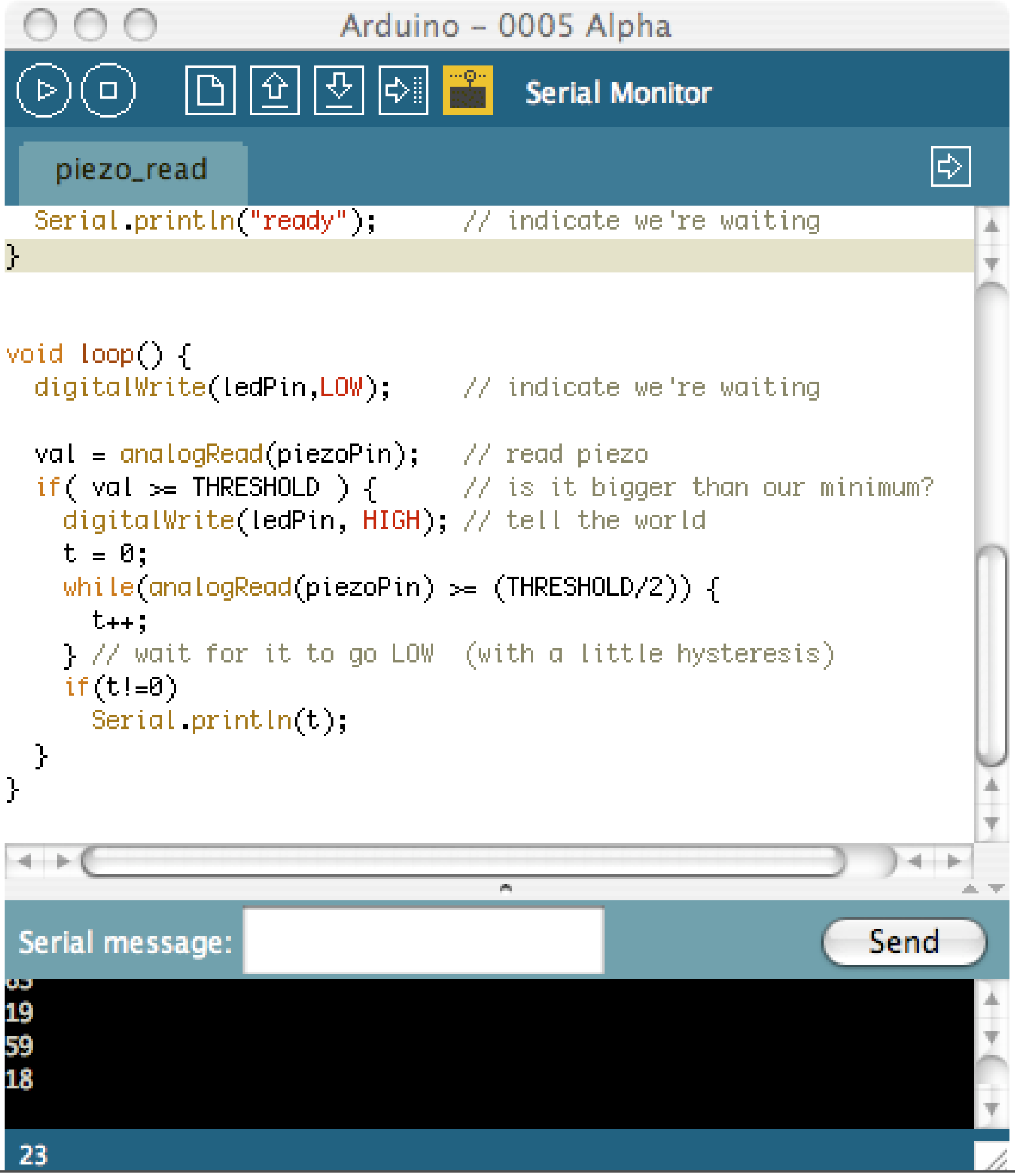
Create two little busses for GND and A0, and hook components across it.
Black bar on diode indicates “bar” of diode.

Piezo Read

"`piezo_read`"

Whack the piezo to generate a number based on force of whack

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



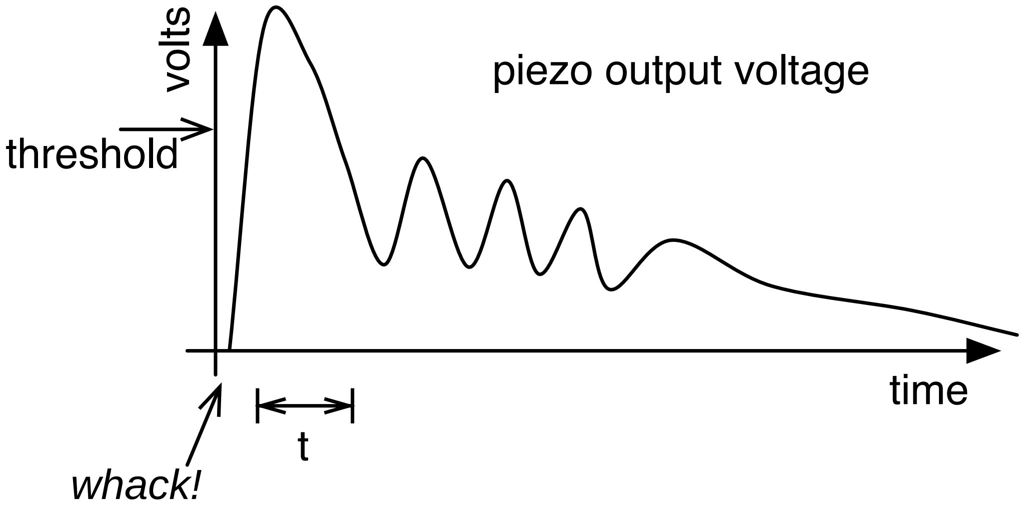
```
  Serial.println("ready");      // indicate we're waiting
}


void loop() {
  digitalWrite(ledPin,LOW);     // indicate we're waiting

  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/2)) {
      t++;
    } // wait for it to go LOW  (with a little hysteresis)
    if(t!=0)
      Serial.println(t);
  }
}
```

```
19
59
18
```

Number is "t", the number of times it looped waiting for the value to drop below THRESHOLD/2.

How Does that Work?

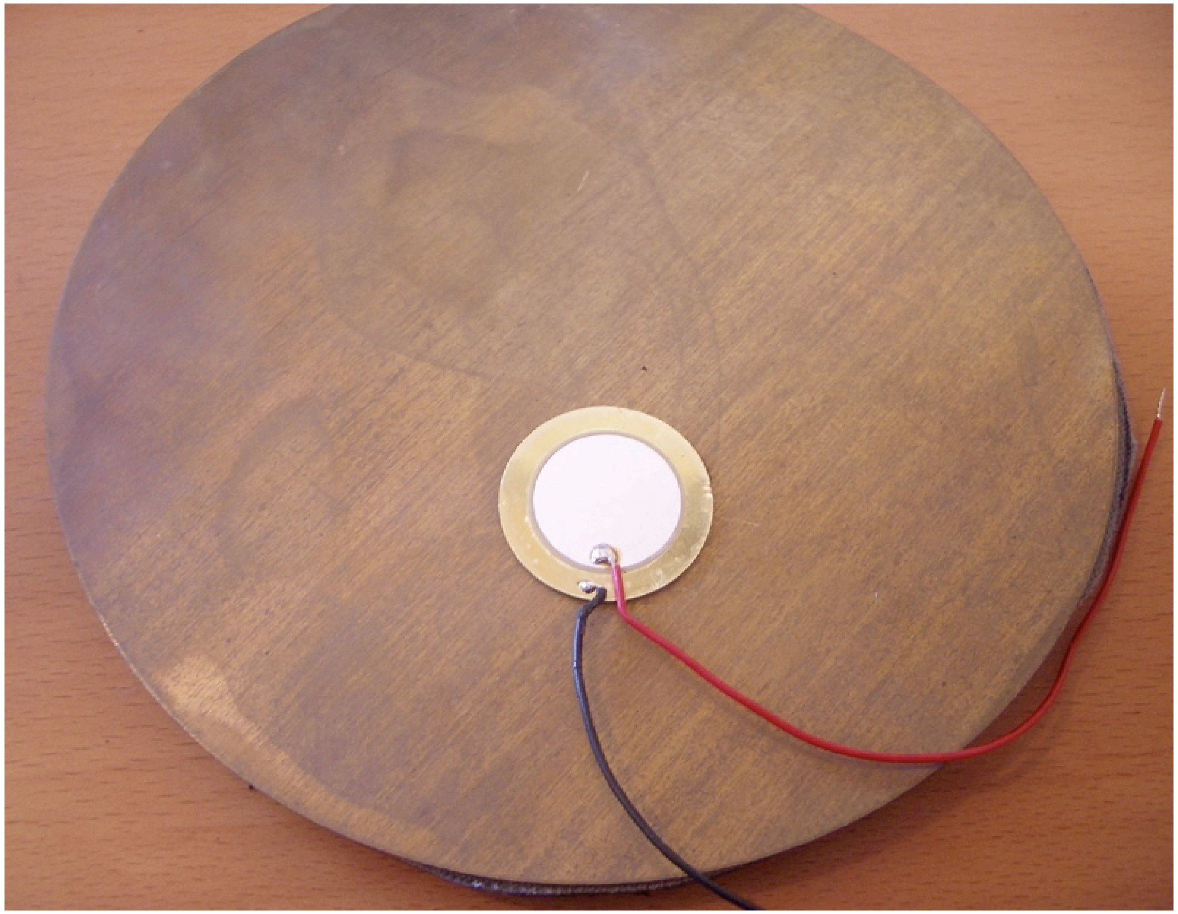
- When a piezo is struck, it "rings" like a bell
- But instead of sound, it outputs voltage
- The sketch measures *time* above a certain voltage, hoping to catch largest ring



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
(floor mat, door, your body, etc.)



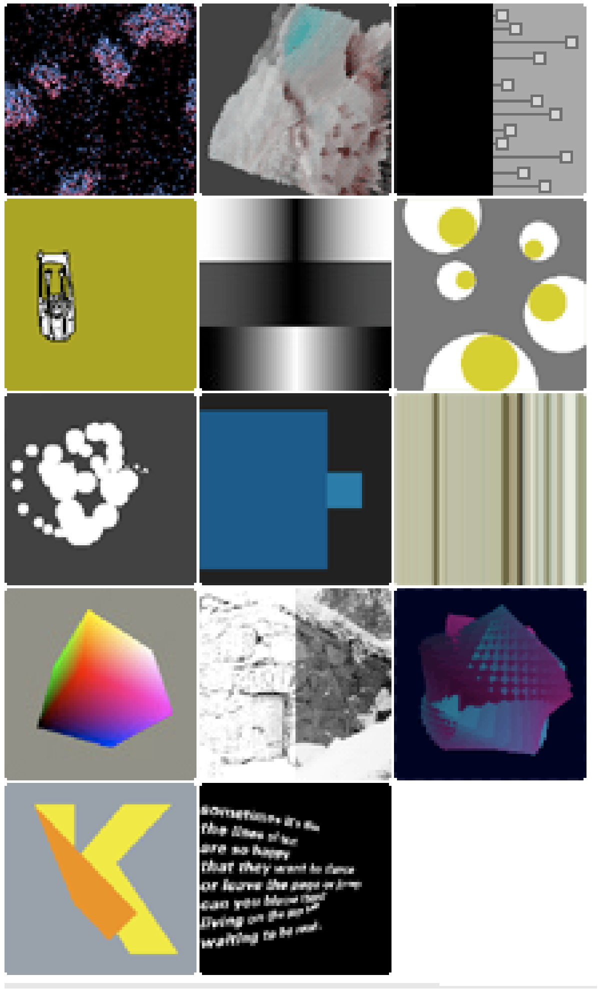
Here’s one glued to a larger brass disc for a drum trigger

Take a Break

(see Craft magazine!)

Processing

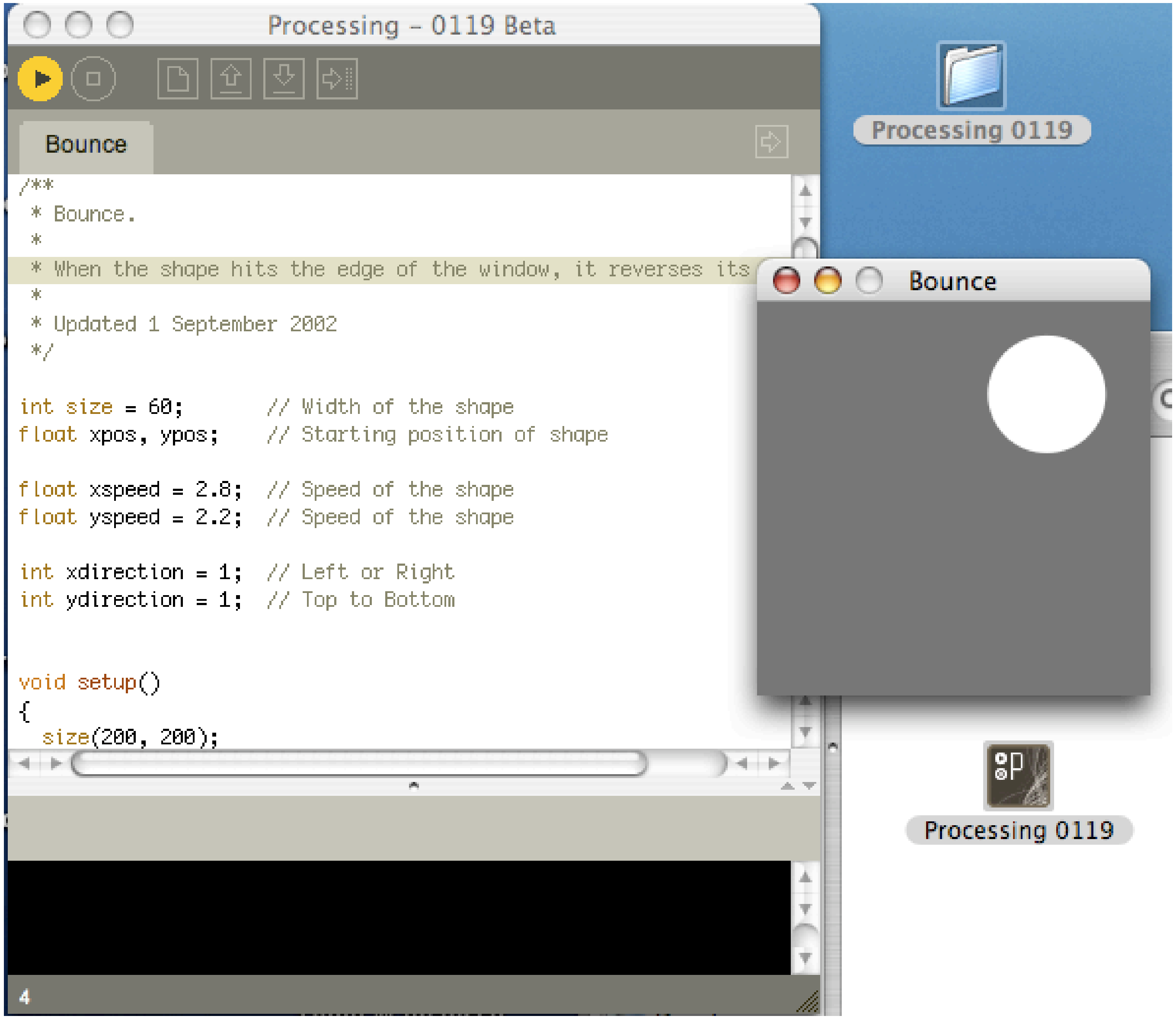
- Processing makes Java programming as fun & easy as Arduino makes AVR programming
- Started as a tool to make generative art
- Is also often used to interface to devices like Arduino



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 "Sketchbook » Examples » 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 » 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”,9600)
port.read(), port.write(), etc.
serialEvent() { }
```

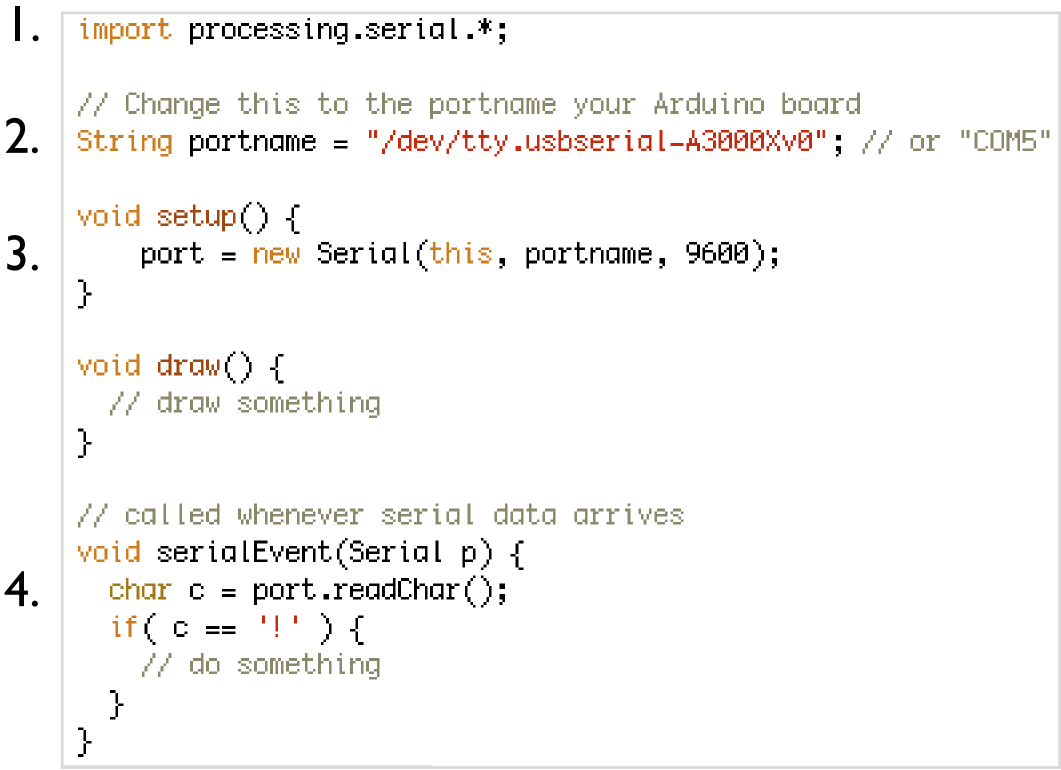
Using the serial library adds a new function you can use to your sketch: serialEvent()
The serialEvent() function will get called whenever serial data is available.

Processing Serial

common Processing serial use

four steps

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



```
import processing.serial.*;

// Change this to the portname your Arduino board
String portname = "/dev/tty.usbserial-A3000Xv0"; // or "COM5"

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

void draw() {
  // draw something
}

// called whenever serial data arrives
void serialEvent(Serial p) {
  char c = port.readChar();
  if( c == '!' ) {
    // do something
  }
}
```

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

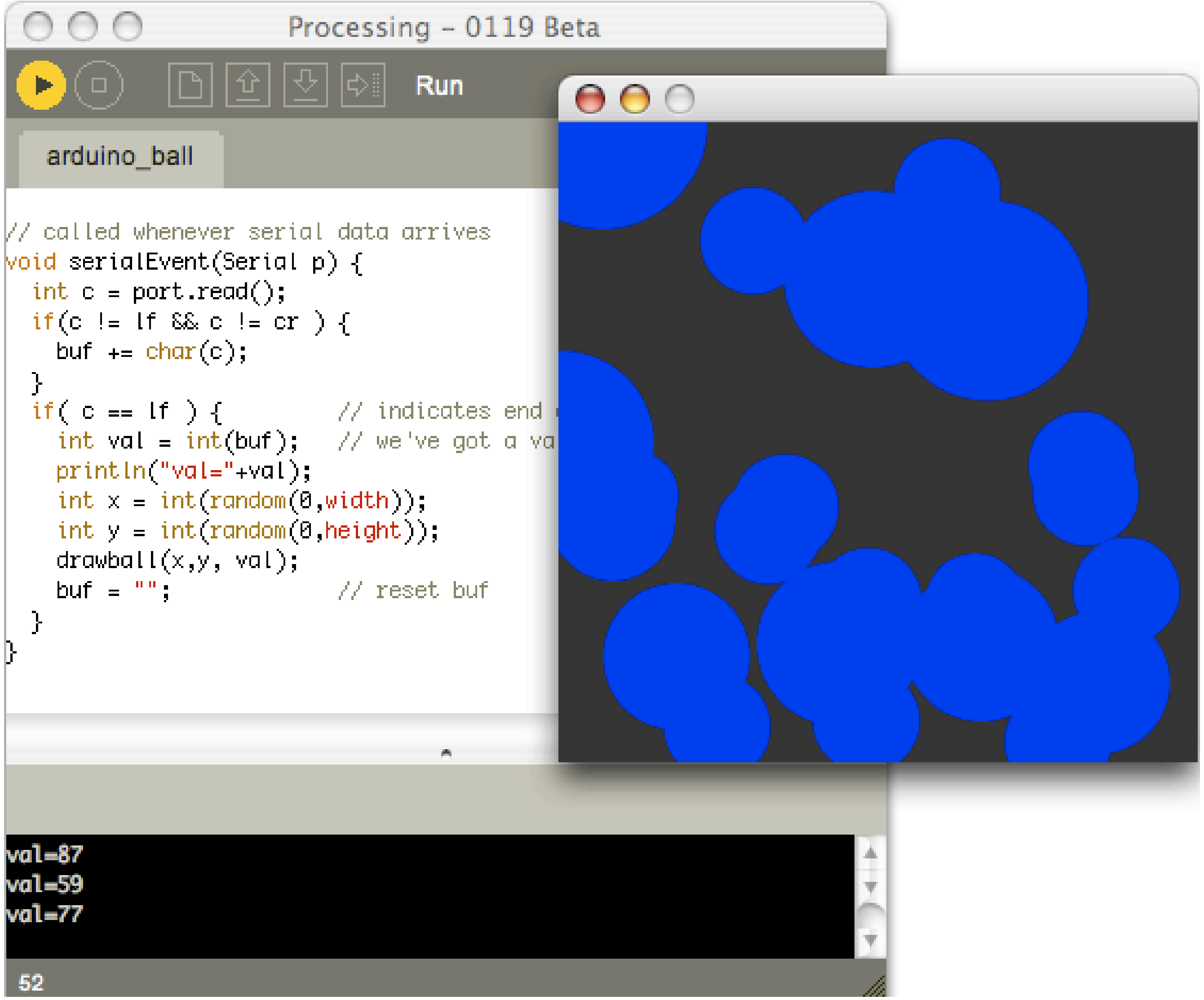
All you need to do talk to Arduino in Processing.
The import statement says you want to do serial stuff.
The “new Serial” creates a serial port object within Processing
Then you can that object (or used the passed in one) to read from in the “serialEvent()” function

Processing & Arduino

“arduino_ball”

Every time a number is received via the serial port, it draws a ball that size.

Use “piezo_read” Arduino sketch from before



```
// called whenever serial data arrives
void serialEvent(Serial p) {
  int c = port.read();
  if(c != lf && c != cr ) {
    buf += char(c);
  }
  if( c == lf ) {         // indicates end 
    int val = int(buf);   // we've got a va
    println("val="+val);
    int x = int(random(0,width));
    int y = int(random(0,height));
    drawball(x,y, val);
    buf = "";             // reset buf
  }
}
```

```
val=87
val=59
val=77
```

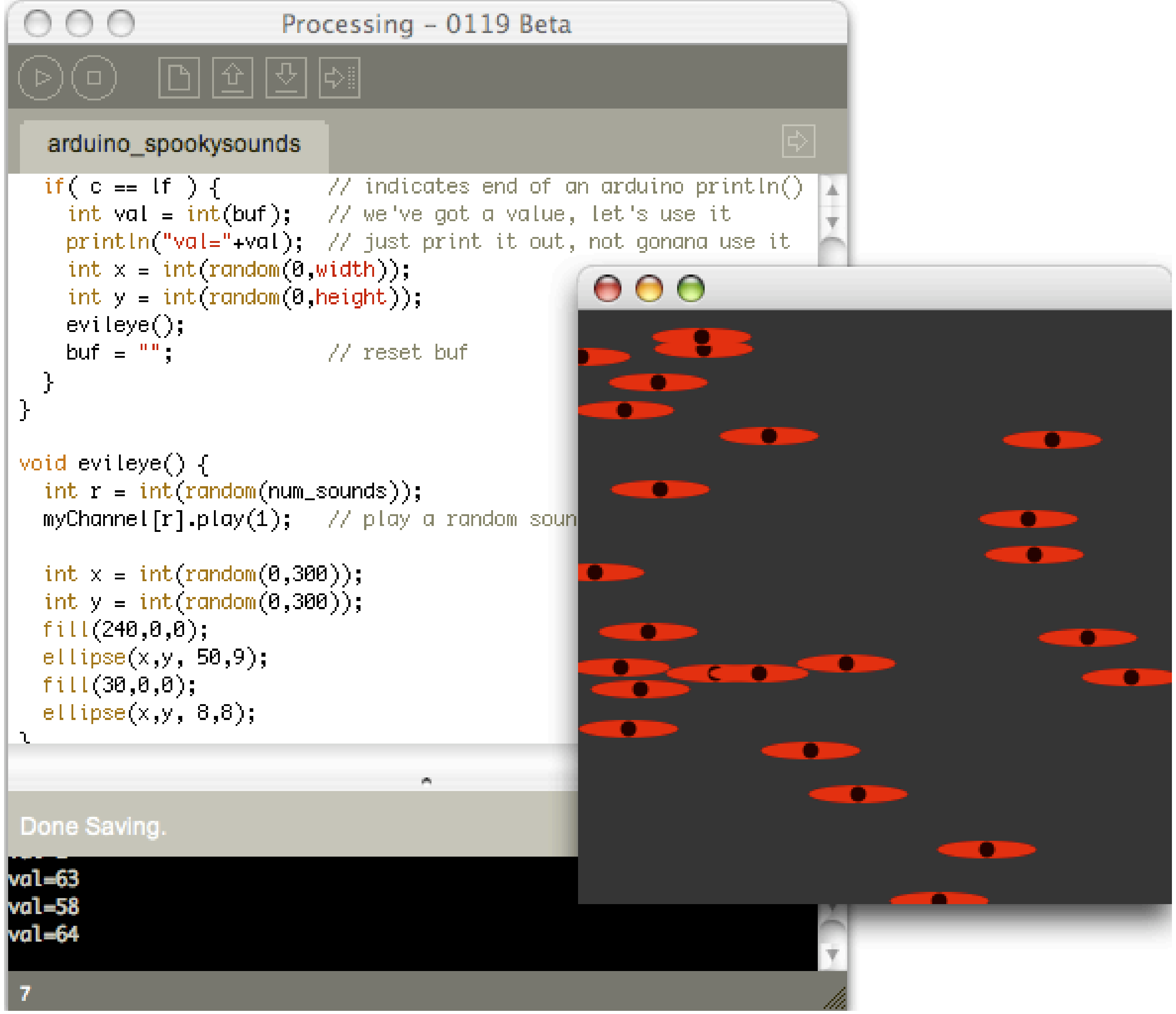
This sketch is in the handout.
Uses “serialEvent()” and “read()” to build up a string and then parse it into a number with “int()”

Spookier, Please

“arduino_ spookysounds”

Every time the piezo is knocked... a scary eye opens and a spooky sound plays

piezo val is printed, but not used: just its existance is



```
    if( c == lf ) {         // indicates end of an arduino println()
      int val = int(buf);   // we've got a value, let's use it
      println("val="+val);  // just print it out, not gonana use it
      int x = int(random(0,width));
      int y = int(random(0,height));
      evileye();
      buf = "";             // reset buf
    }
  }

  void evileye() {
    int r = int(random(num_sounds));
    myChannel[r].play(1);   // play a random soun

    int x = int(random(0,300));
    int y = int(random(0,300));
    fill(240,0,0);
    ellipse(x,y, 50,9);
    fill(30,0,0);
    ellipse(x,y, 8,8);
  }
```

```
val=63
val=58
val=64
```

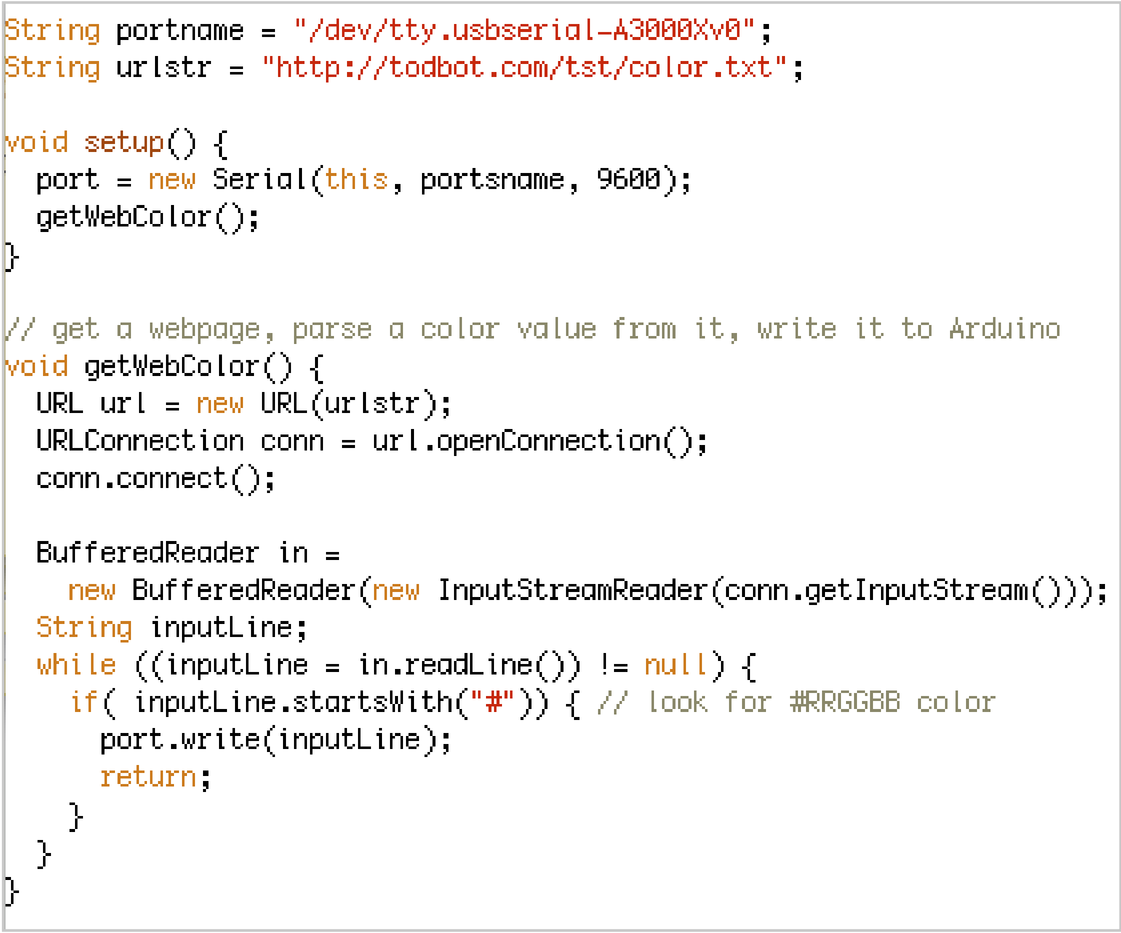
This sketch is in the handout.
You can add your own sounds (must be 16-bit WAV or AIFF).
Hook a piezo up to your front door, and plug your computer into your stereo.
Every time someone knocks on your door, a scary sound is played

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/

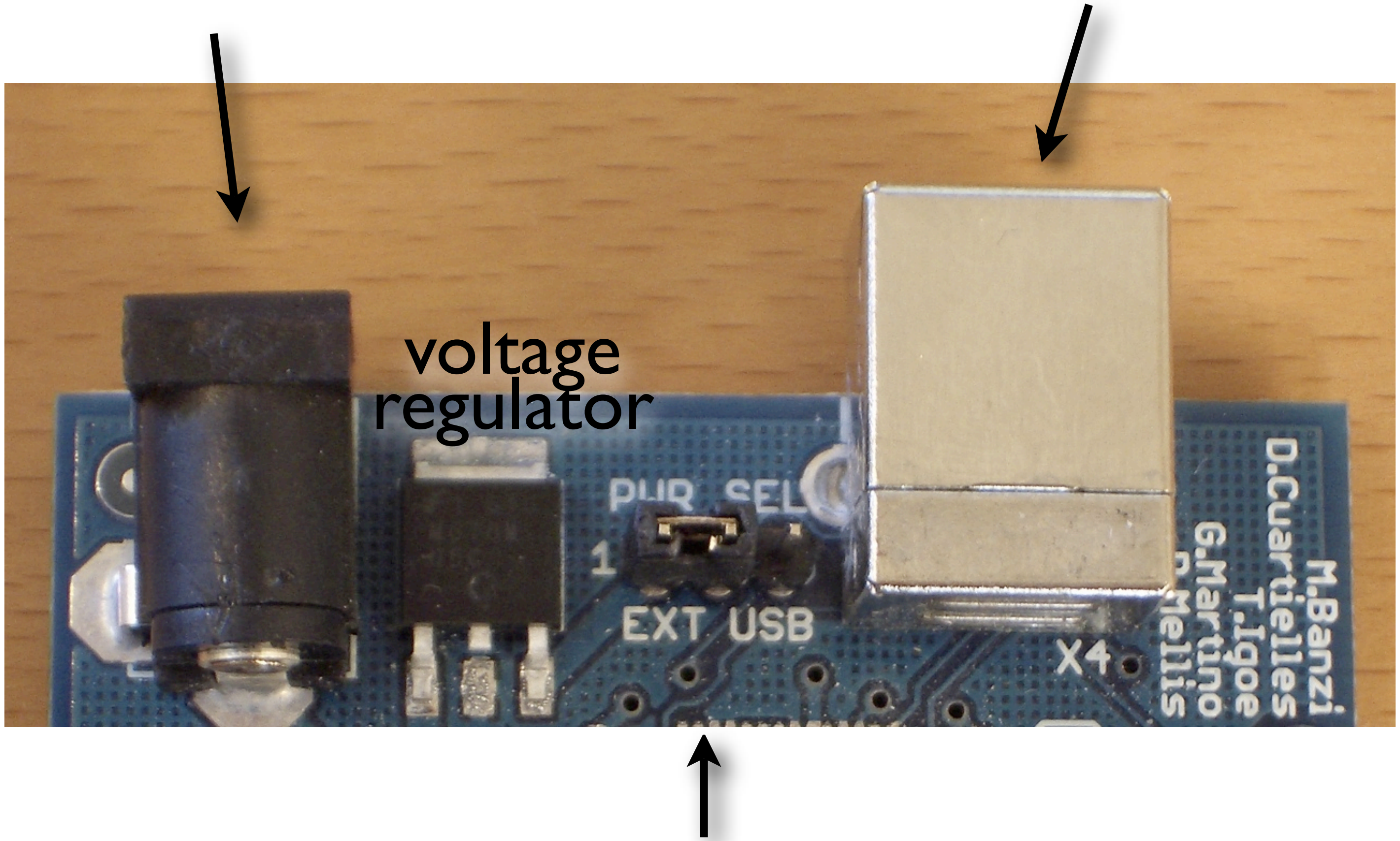
Fun Uses

External Power

Arduino can run off USB power or external power

External power connector

USB connector



voltage regulator

jumper switch to choose power source

External Power

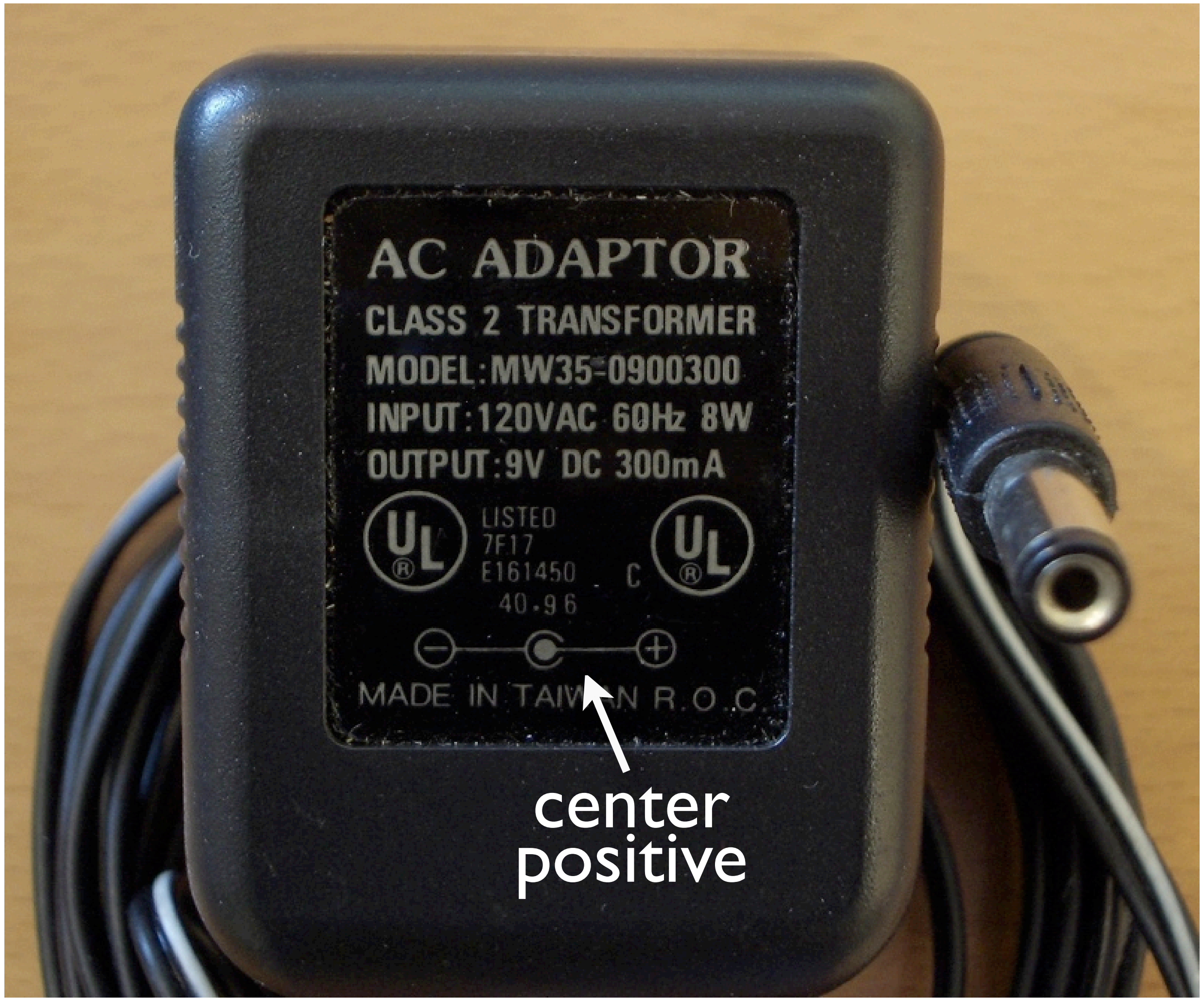
You can use an AC adpater

Connector is standard barrel connector

Make sure it's "center positive"

Voltage can be 9-15V DC

Amps is > 200mA

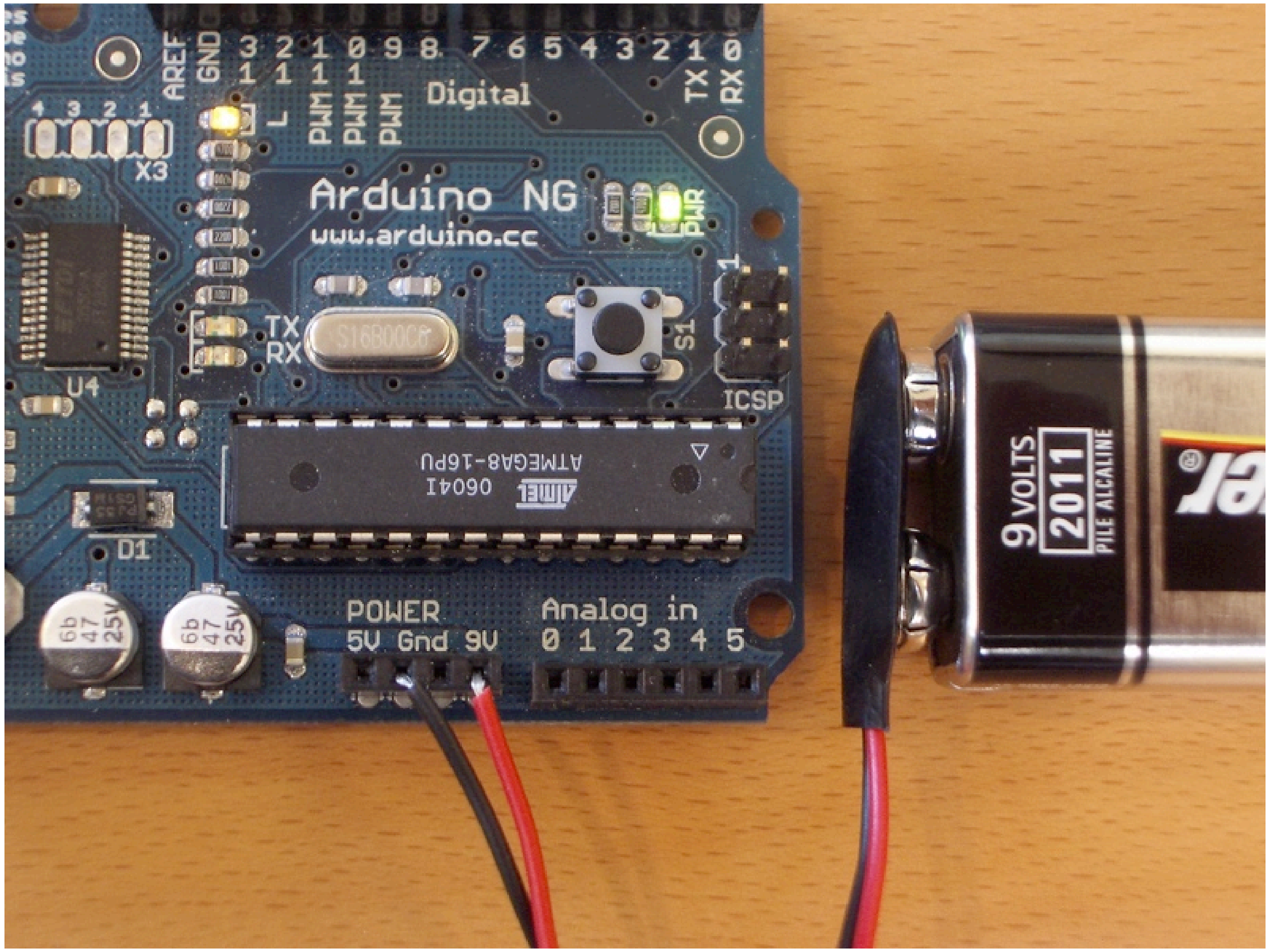


center positive

Actually input voltage can be from like 7.5V to 35V, but don't go over 15V so the voltage regulator doesn't have to work so hard.

External Power

Or you can use a battery

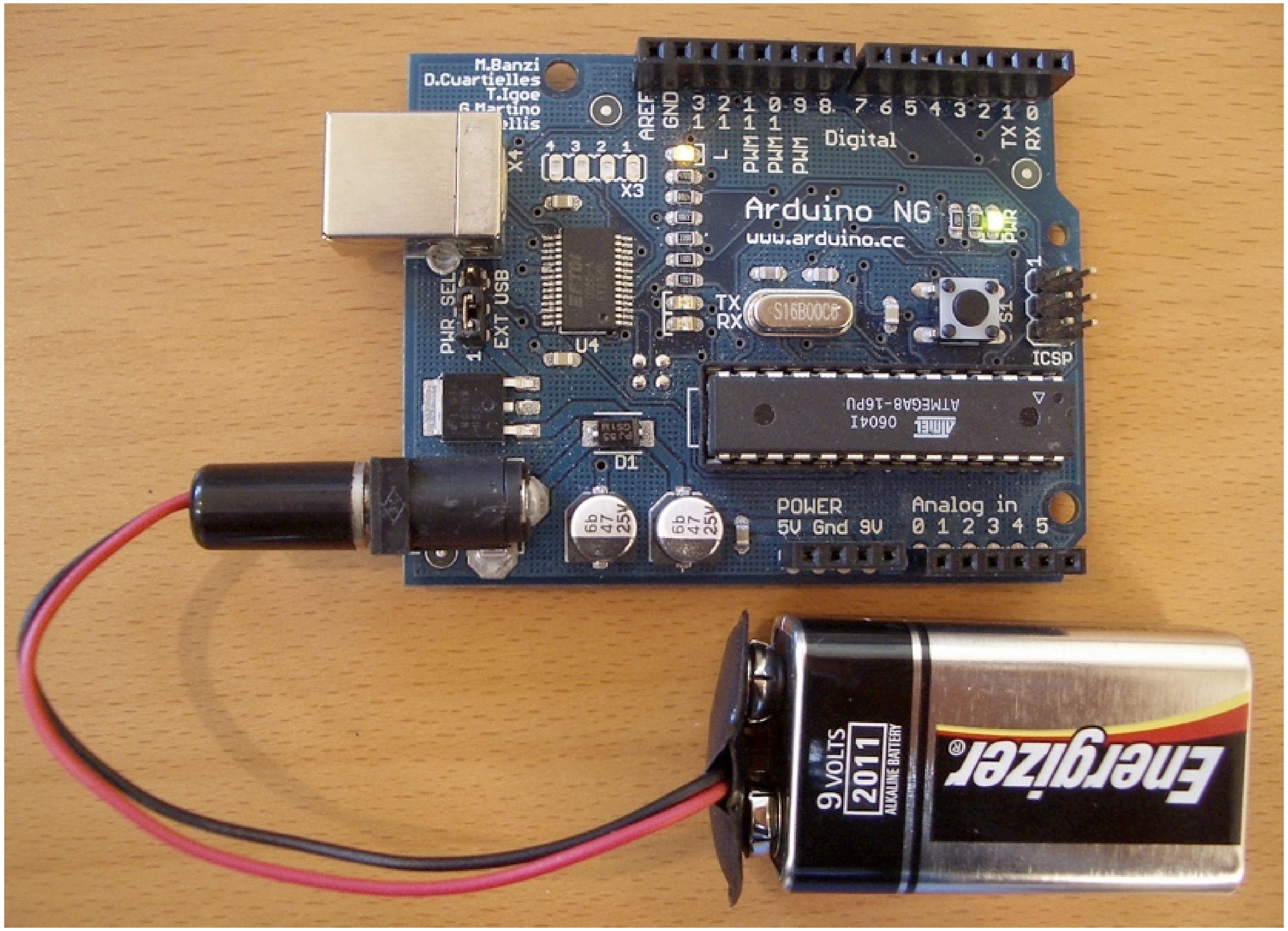


Be careful about polarity! And shorts!

On the prototyping shield you plug in on top, the “9V” socket is called “raw”

External Power

An easier way to connect a battery



also solves polarity concerns

Power connector input has protection diode.
Also it’s easier with the connector

External Power

Battery life

How long does Arduino last on 9V battery?

- Arduino board draws about 40 mA by itself
- Each LED adds about 20mA when on
- Each servo maybe 100 mA when running
- Switches, pots, etc. are effectively zero

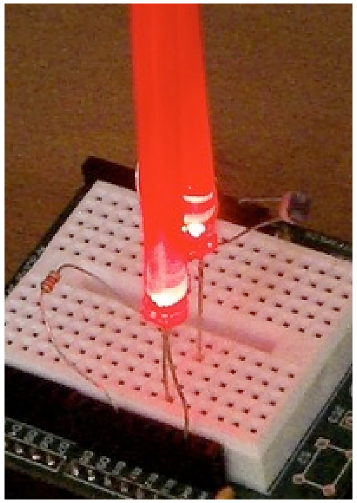
- Battery capacity rated in milliamp-hours (mAh)
- 9V batteries have about 400 mAh capacity

Thus, Arduino by itself lasts 400/40 = 10 hours

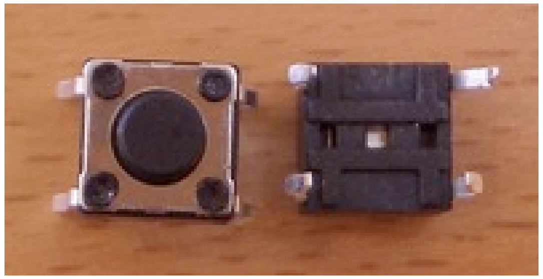
Take all your power, add it up, divide it into your battery capacity to get time in hours.
There are techniques to make an AVR chip go into sleep mode, and draw microamps (1/1000 mA), but those techniques don’t have nice Arduino-style wrappers yet.
For more on batteries and their capacities: http://en.wikipedia.org/wiki/List_of_battery_sizes

Summary

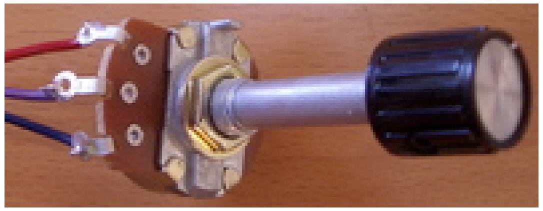
You've learned many different physical building blocks



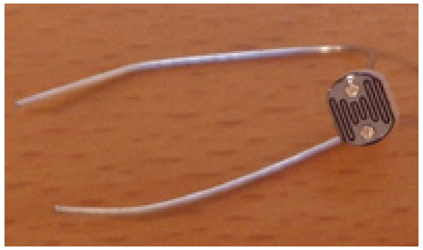
LEDs

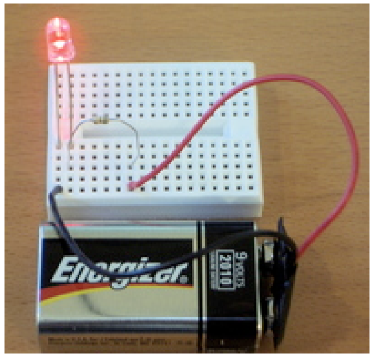


switches/buttons

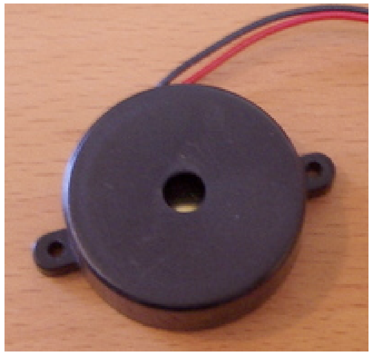


resistive sensors

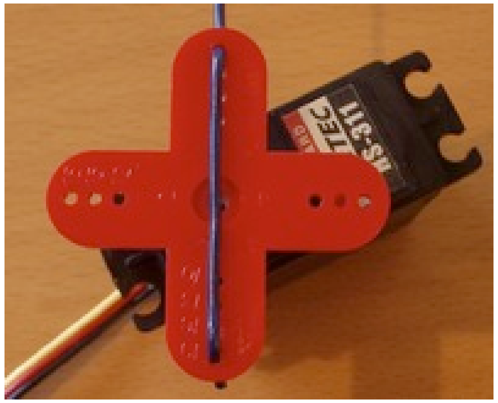




fast prototyping



piezos



servos

Summary

And you've learned many software building blocks

serial communication

pulse width modulation

analog I/O

digital I/O

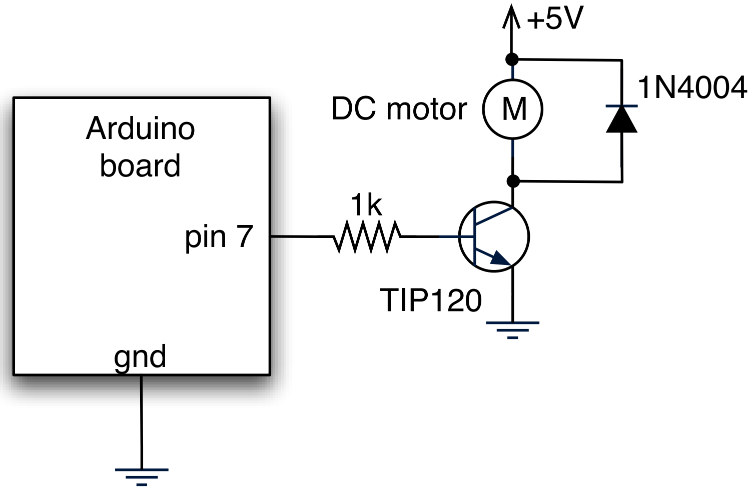
data driven code

frequency modulation

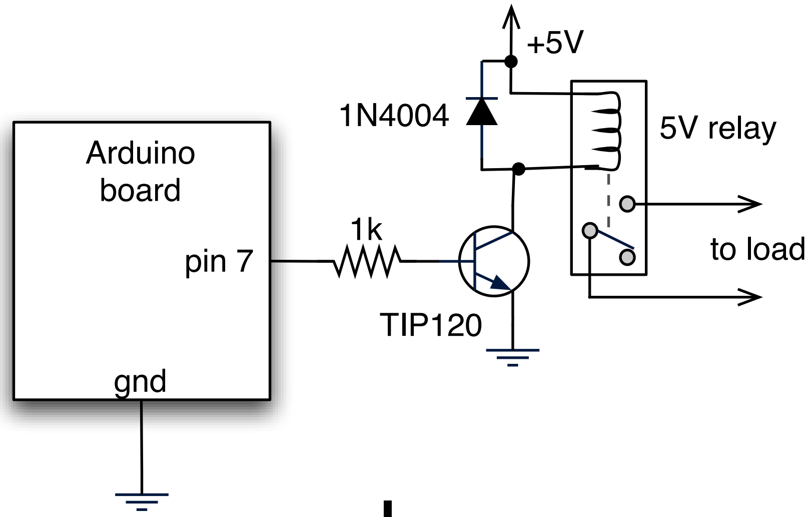
multiple tasks

Summary

Some things we didn't cover, like:



motors



relays

But they use concepts you know

Summary

Hope you had fun and learned something

Feel free to contact me to chat about this stuff

END Class 4

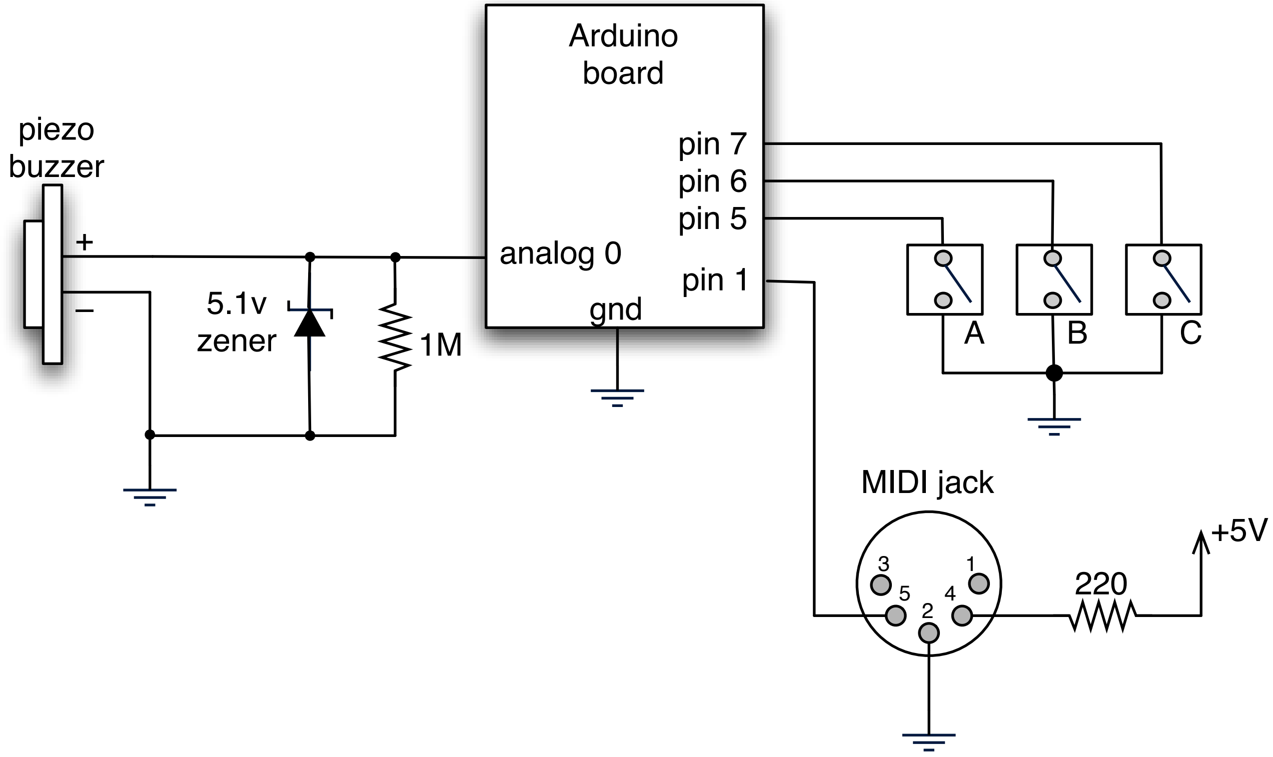
http://todbot.com/blog/spookyarduino

Tod E. Kurt

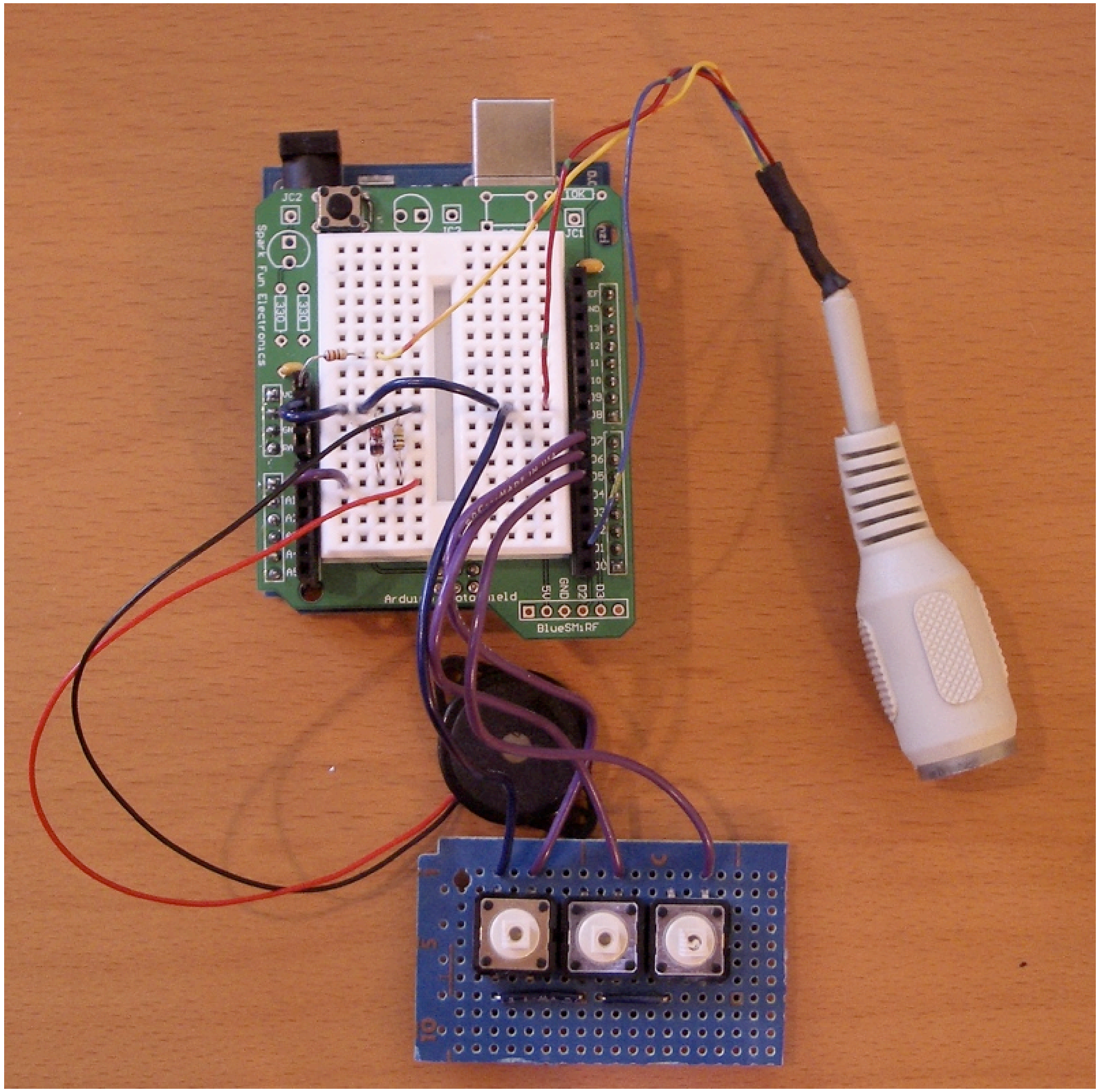
tod@todbot.com

A little extra: MIDI

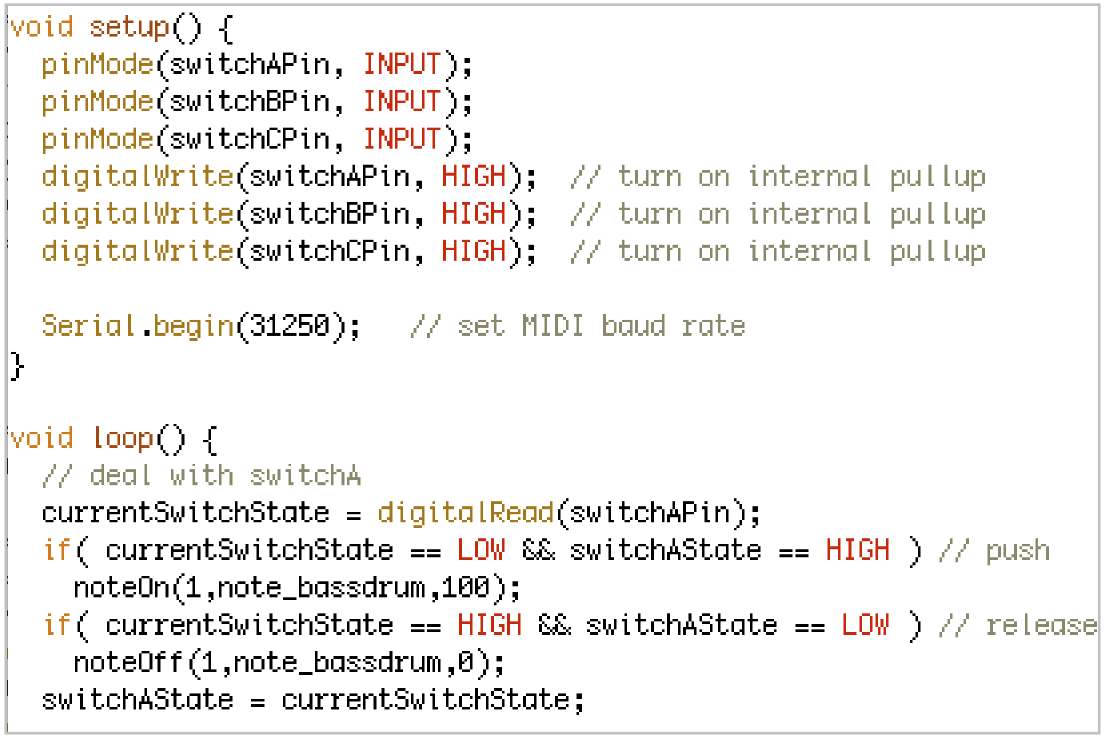
Combine everything, add a MIDI jack



A little extra: MIDI

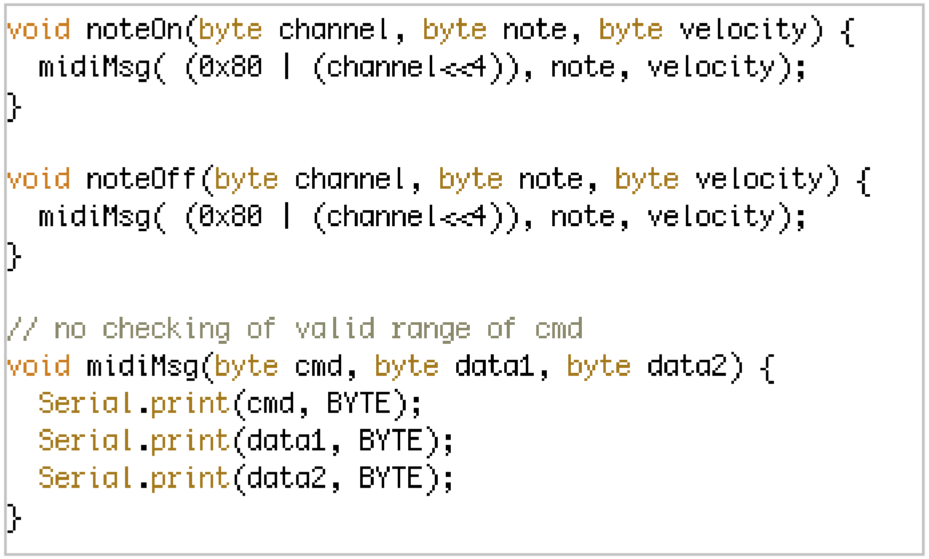


A little extra: MIDI



```
void setup() {
  pinMode(switchAPin, INPUT);
  pinMode(switchBPin, INPUT);
  pinMode(switchCPin, INPUT);
  digitalWrite(switchAPin, HIGH);  // turn on internal pullup
  digitalWrite(switchBPin, HIGH);  // turn on internal pullup
  digitalWrite(switchCPin, HIGH);  // turn on internal pullup

  Serial.begin(31250);   // set MIDI baud rate
}

void loop() {
  // deal with switchA
  currentSwitchState = digitalRead(switchAPin);
  if( currentSwitchState == LOW && switchAState == HIGH ) // push
    noteOn(1,note_bassdrum,100);
  if( currentSwitchState == HIGH && switchAState == LOW ) // release
    noteOff(1,note_bassdrum,0);
  switchAState = currentSwitchState;
```

MIDI is just serial at 31250 baud
buttons are drum triggers



```
void noteOn(byte channel, byte note, byte velocity) {
  midiMsg( (0x80 | (channel<<4)), note, velocity);
}

void noteOff(byte channel, byte note, byte velocity) {
  midiMsg( (0x80 | (channel<<4)), note, velocity);
}

// no checking of valid range of cmd
void midiMsg(byte cmd, byte data1, byte data2) {
  Serial.print(cmd, BYTE);
  Serial.print(data1, BYTE);
  Serial.print(data2, BYTE);
}
```

sends MIDI note-on & note-off messages

end