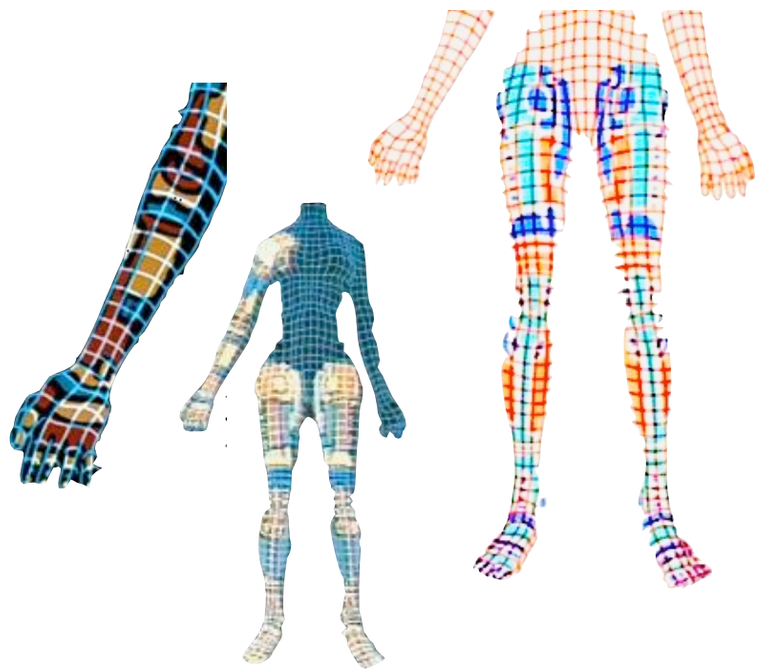
Bionic Arduino

Introduction to Microcontrollers with Arduino



Class 2

13 Nov 2007 - machineproject - Tod E. Kurt

What's for Today

- Random Behavior
- RGB LEDs
- Color mixing
- Analog input with variable resistors
- Potentiometers & photocells
- Basic serial input & output
- Playing sound with piezo buzzers

This is a lot of stuff, let's see how far we get.

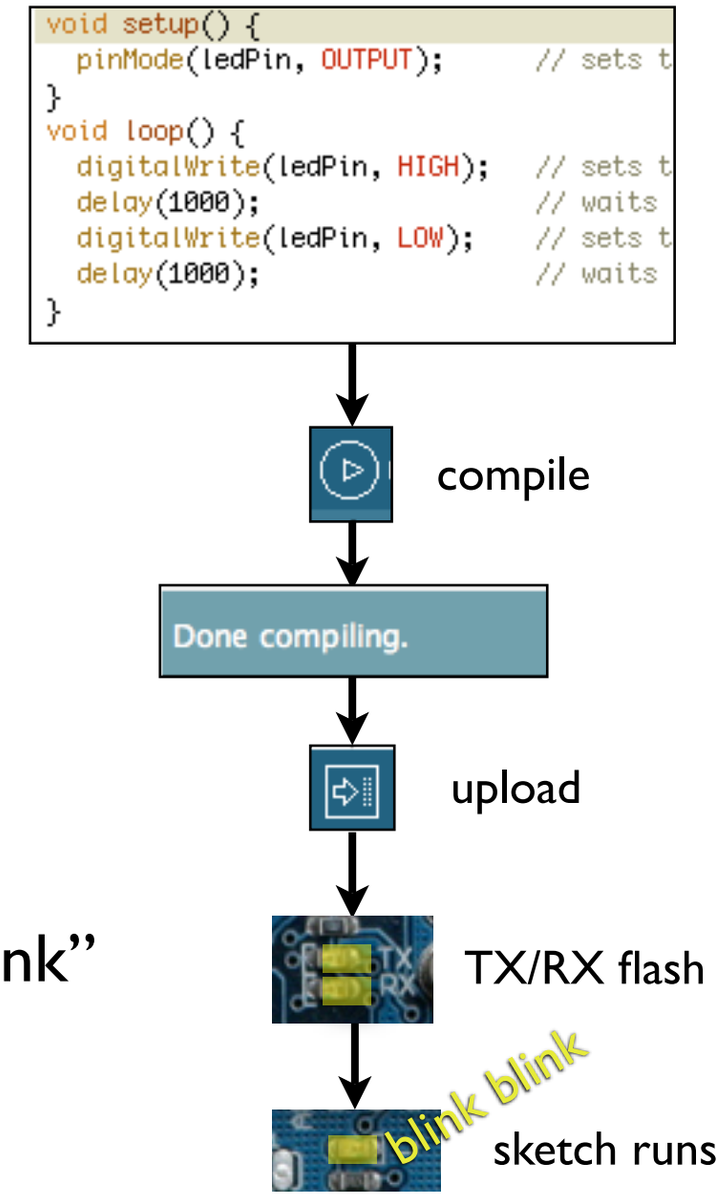
Recap: Blinky LED

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”

Change the “delay()” values to change blink rate

Known Good Configuration

Rule #1 of experimenting:

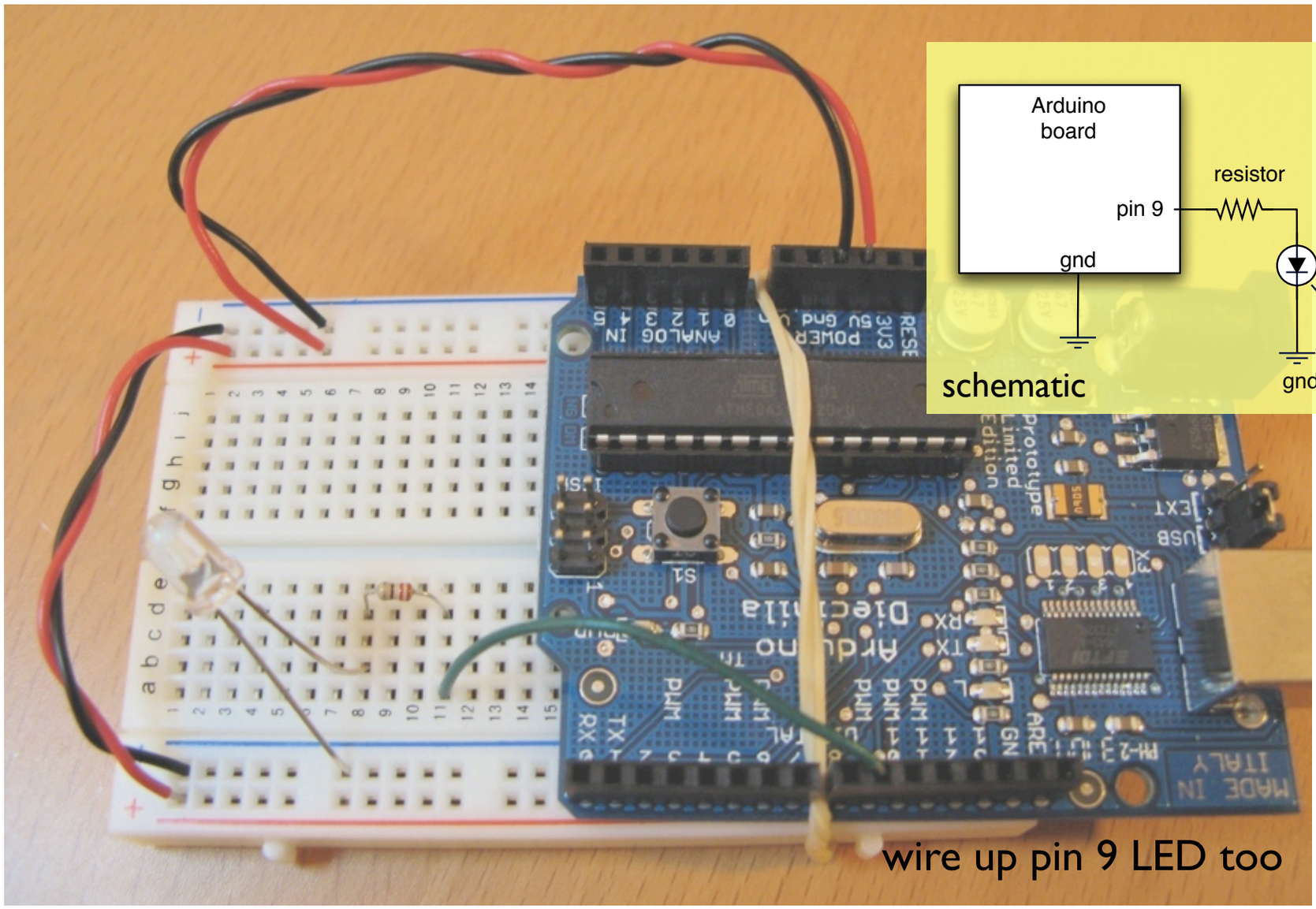
Before trying anything new,

Get back to a known working state

So spend a few minutes & get "Blink" working again

Get your entire edit->compile->upload->run working
Even if it becomes so second nature to you that you feel you shouldn't need to, do it anyway.
Especially when mysterious problems arise, revert to a known state

Getting the Board Set Up

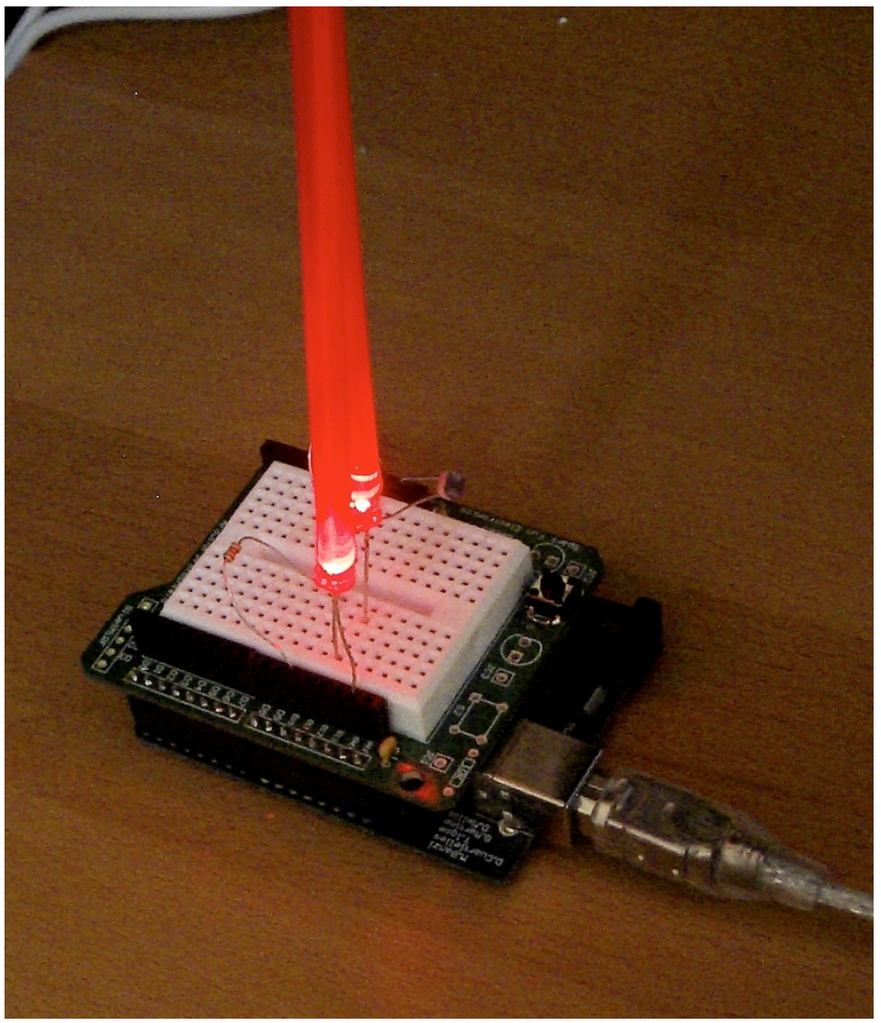


Questions / Review

Any questions, comments, or problems?

Aside: LED Light Tubes

Snug-fit straws on the end of your LEDs to make them glow more visibly



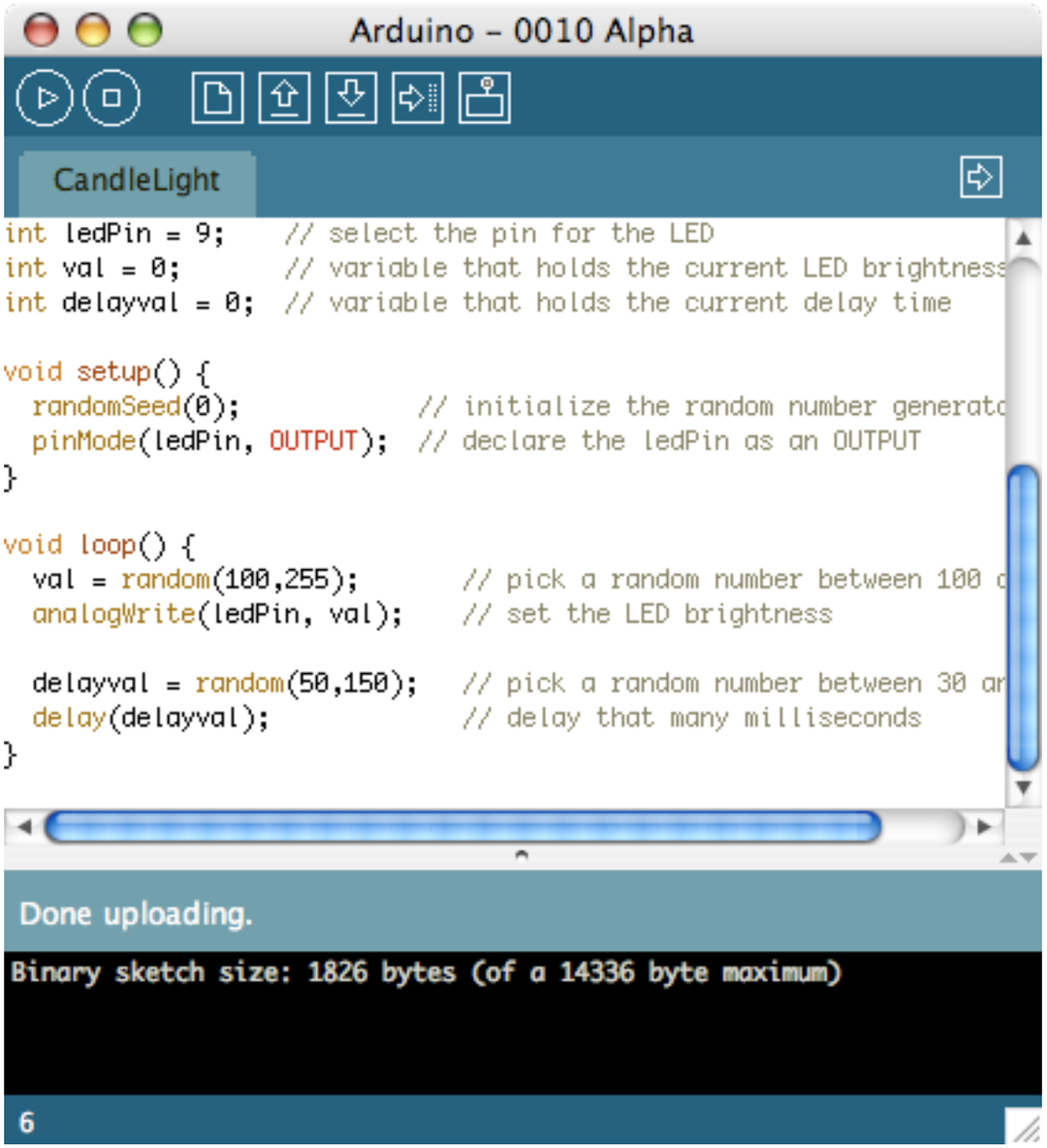
I have a box of multi-colored straws for whatever color LED you like

Random Behavior

"CandleLight"

Uses simple pseudo random number generator to mimic flame

Use `random(min,max)` to pick a number between min & max.



This sketch is in the handout.
Can also use random numbers to make random decisions.
Note: not truly random, but good enough for most purposes.

Analog Input

To computers, analog is chunky

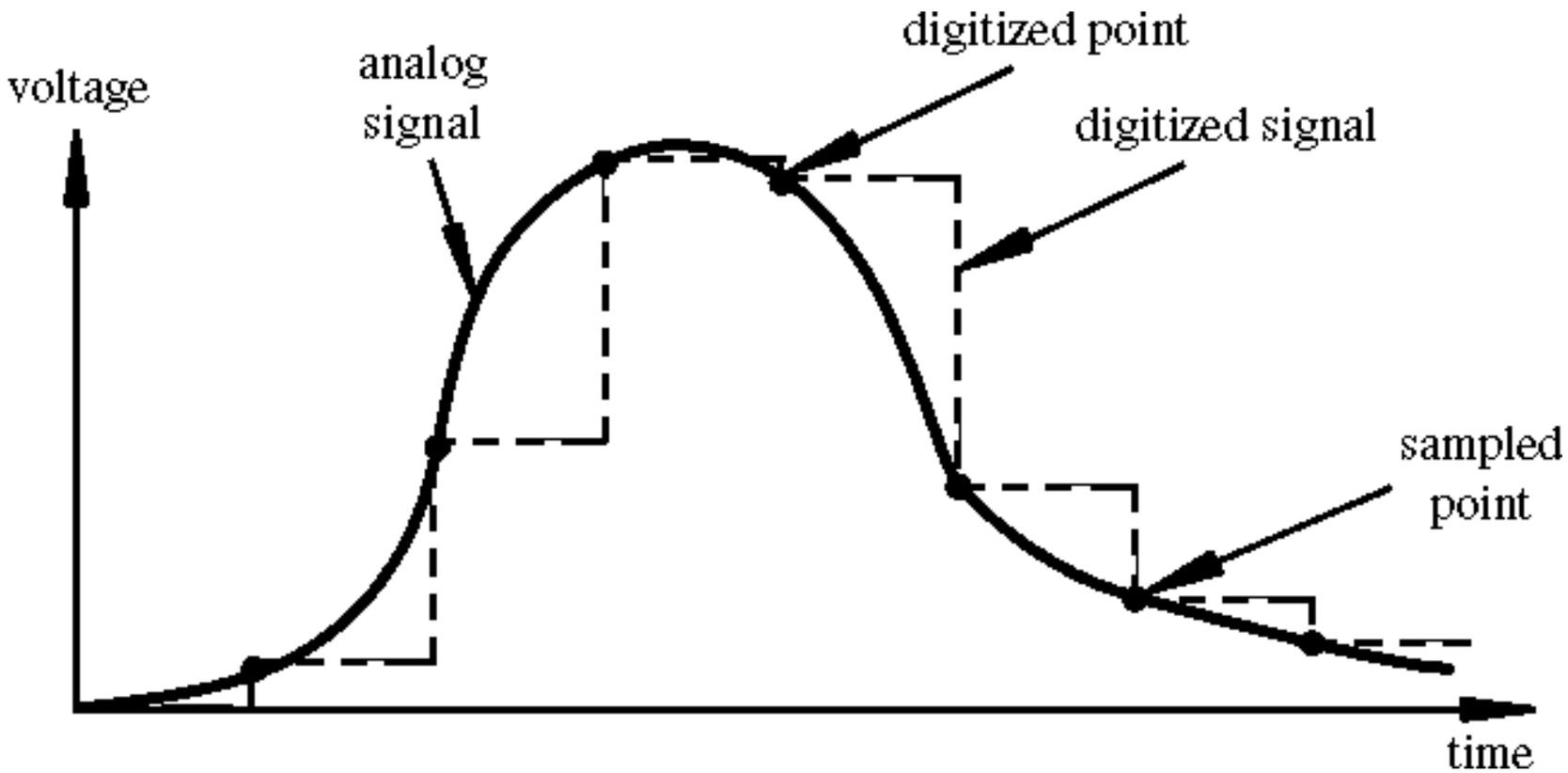
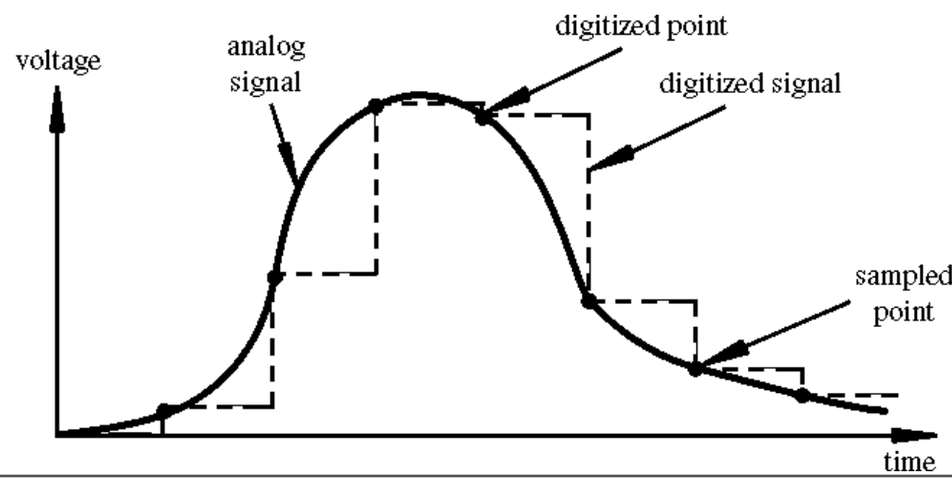


image from: http://www.engr.colostate.edu/~dga/me307/lectures.html

Analog Input

- Many states, not just two (HIGH/LOW)
- Number of states (or values, or "bins") is *resolution*
- Common computer resolutions:
  - 8-bit = 256 values
  - 16-bit = 65,536 values
  - 32-bit = 4,294,967,296 values



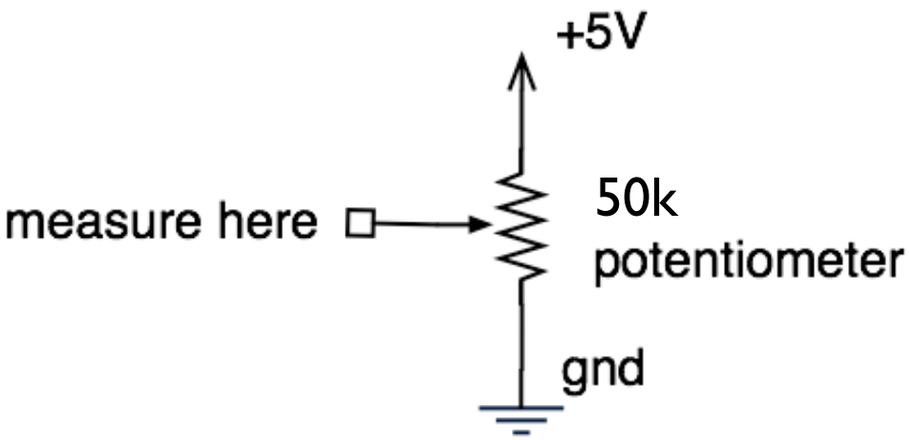
Analog Input

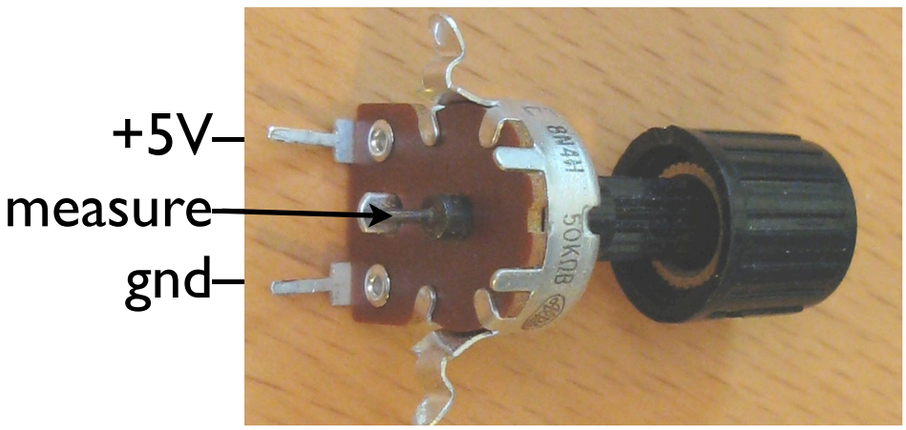
- Arduino (ATmega168) has six ADC inputs
- (ADC = Analog to Digital Converter)
- Reads voltage between 0 to 5 volts
- Resolution is 10-bit (1024 values)
- In other words, 5/1024 = 4.8 mV smallest voltage change you can measure

Analog Input

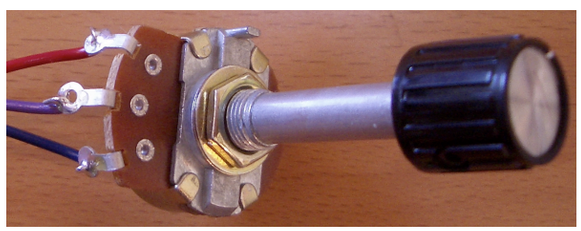
Sure sure, but how to make a varying voltage?

With a *potentiometer*. Or just *pot*.





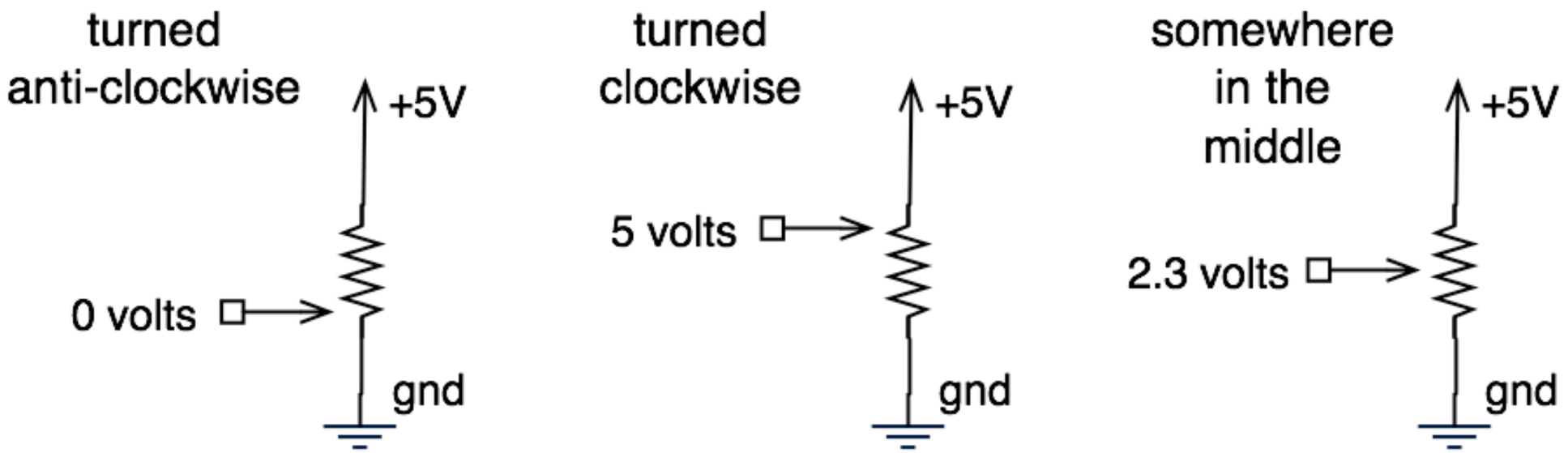
The pot you have



pots also look like this

Potentiometers

Moving the knob is like moving
where the arrow taps the voltage on the resistor



When a resistor goes across a voltage difference, like +5V to Gnd, the voltage measured at any point along a resistor's length is proportional to the distance from one side.

If you take apart a pot, there's a little wiper just like in the schematic symbol.
But I might have the directions reversed (clockwise vs. anti-clockwise).

What good are pots?

- Anytime you need a ranged input
  - (we're used to knobs)
- Measure rotational position
  - steering wheel, robotic joint, etc.

- But more importantly for us, potentiometers are a good example of a *resistive sensor*

There are many kinds of resistive sensors

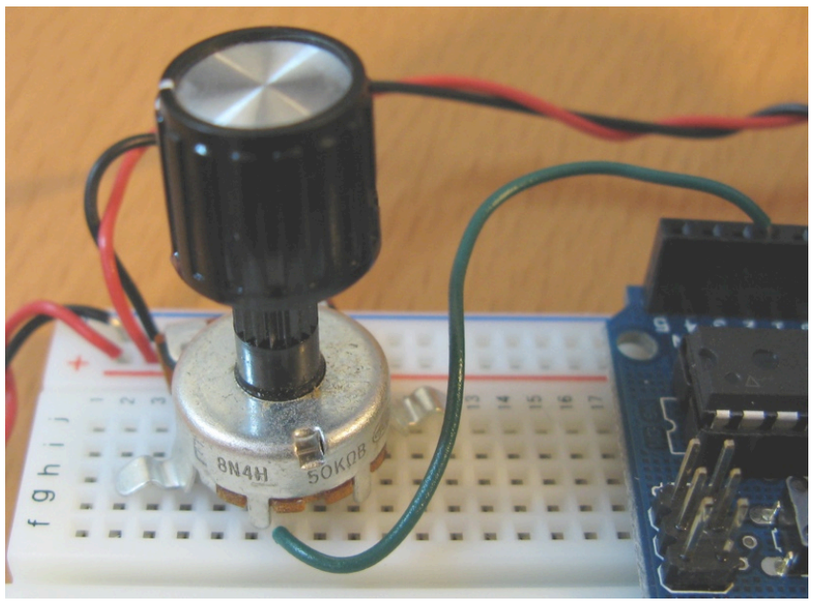
Arduino Analog Input

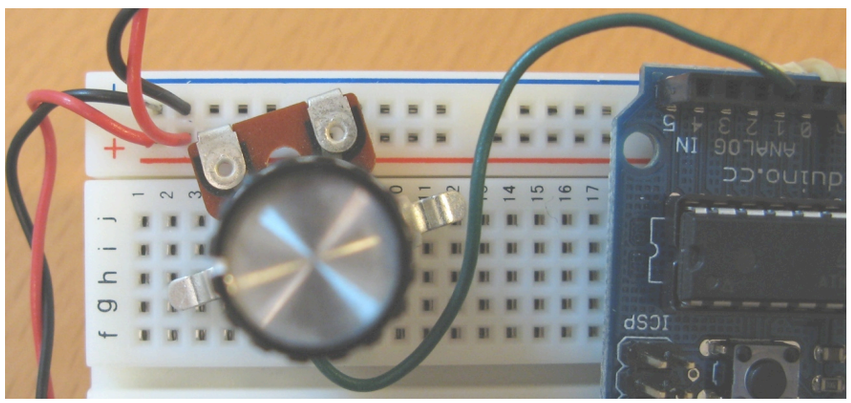
Plug pot directly into breadboard

Two "legs" plug into +5V & Gnd (red + & blue -) buses

Middle "post" plugs into a row (row 7 here)

Run a wire from that row to Analog In 2

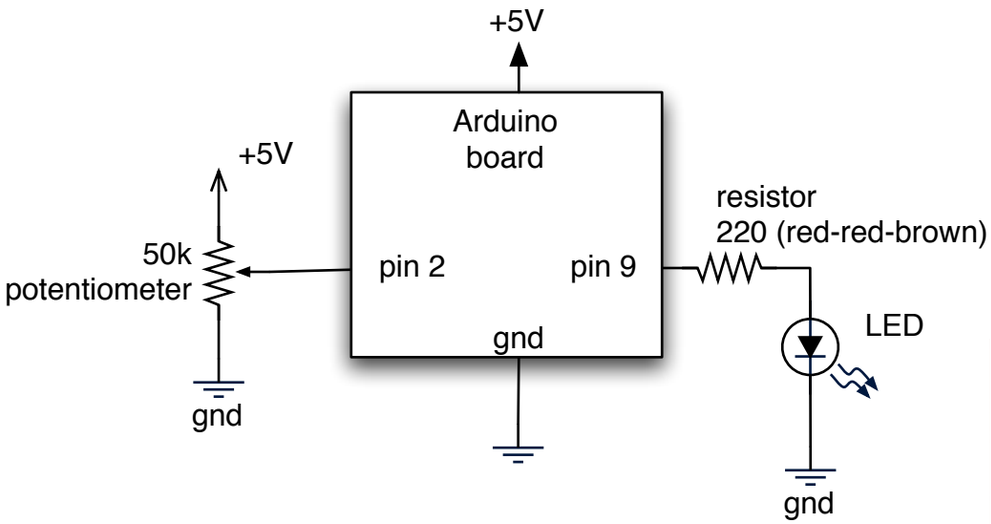


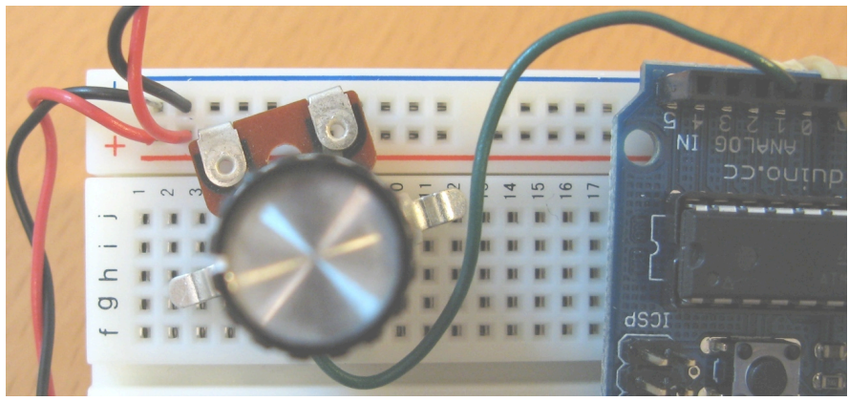


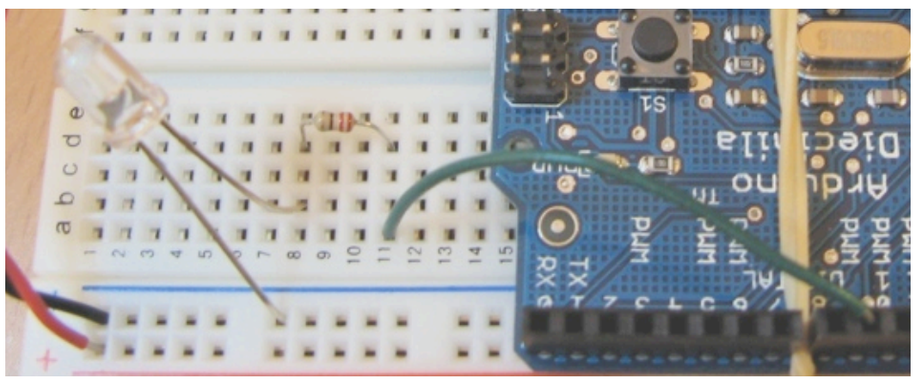
Why are we using Analog In 2? Because it's in the middle. There's no reason, any of the 6 analog inputs would work the same.

Pot & LED Circuit

This is what your board should have on it now







In schematics, inputs are usually on the left, outputs on the right
Also, more positive voltages are on the top, more negative on the bottom

Varying Brightness by Hand

"PotDimmer"

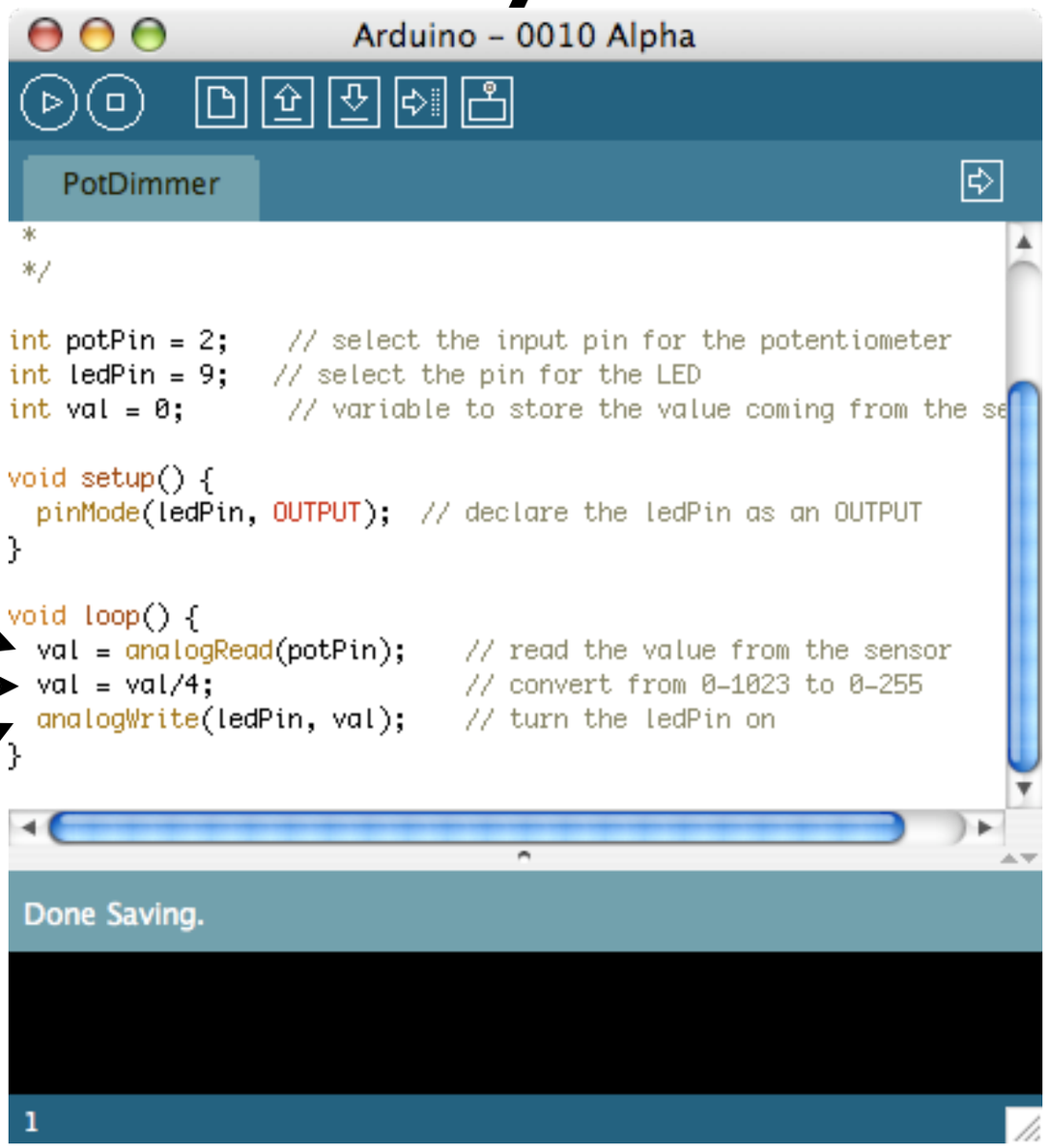
Turn the knob to change LED brightness

input

process the input data

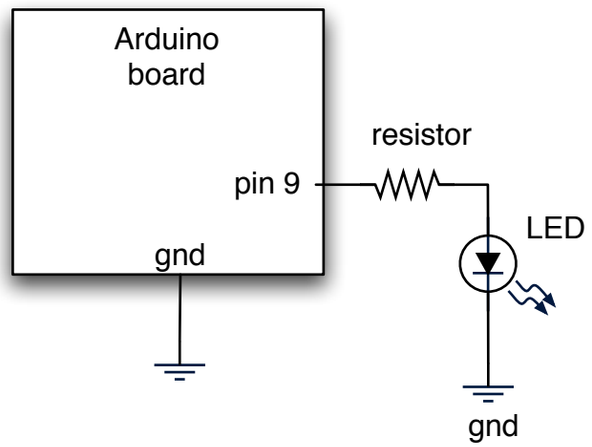
output

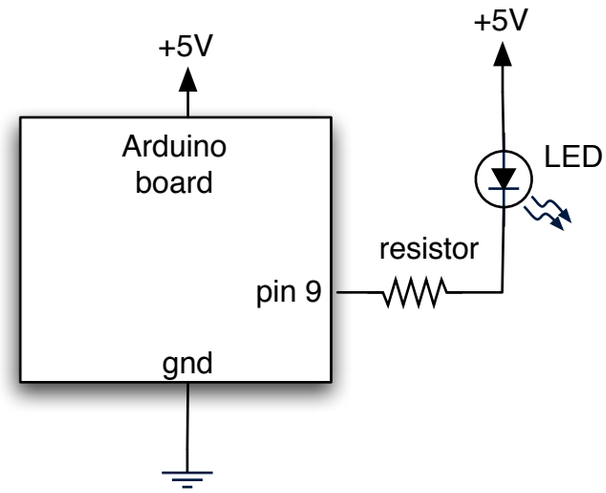
Most all embedded systems have a input→process→output loop



Sketch available in handout

Two Ways to Hook up LEDs





To turn ON: `digitalWrite(9,HIGH)`

To turn OFF: `digitalWrite(9,LOW)`

To set brightness: `analogWrite(9,val)`

To turn ON: `digitalWrite(9,LOW)`

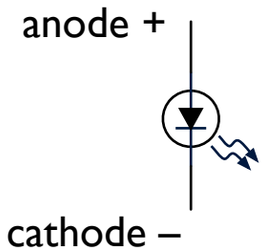
To turn OFF: `digitalWrite(9,HIGH)`

To set brightness: `analogWrite(9,255-val)`

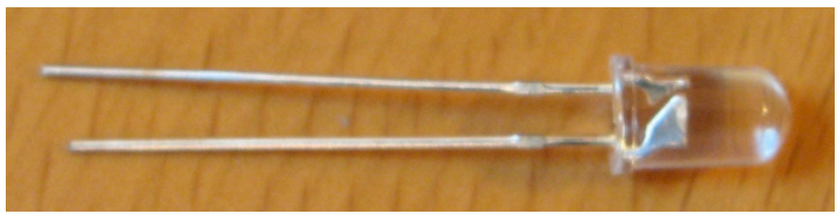
We've been using the one on the left because it makes more sense.
But you'll see the method on the right as well.
The reason for this is that some circuits can switch to Gnd better than they can switch to +5V.

RGB LEDs

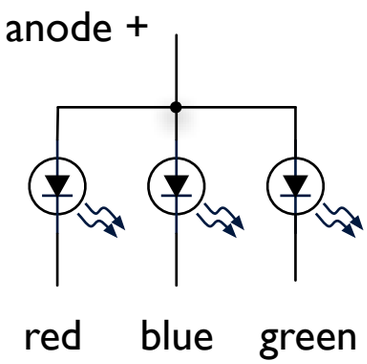
Normal LED



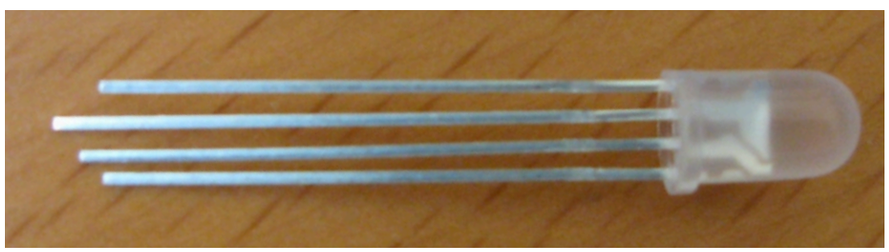
anode +
cathode –



RGB LED



red cathode –
anode +
blue cathode –
green cathode –

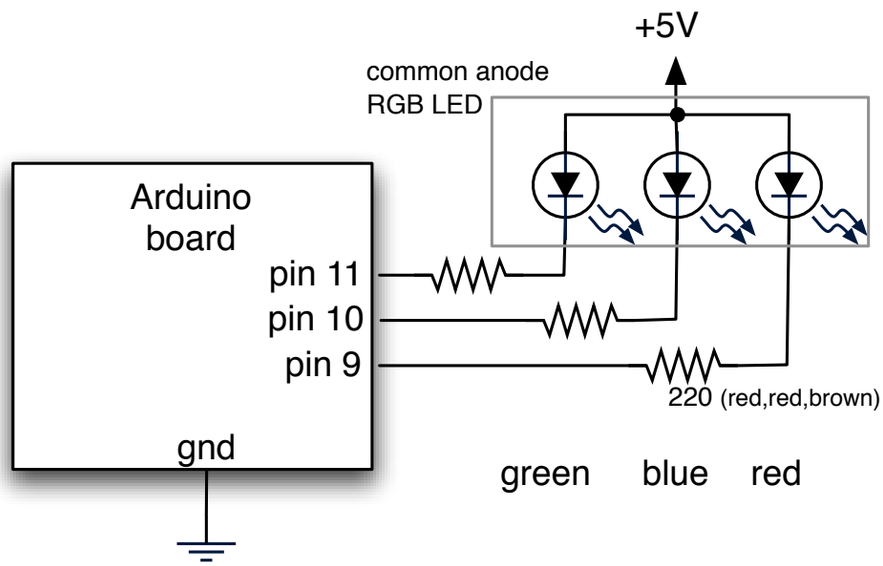


actually 3 LEDs in one package

RGB LED, aka “tri-color LED”
Common-anode RGB LEDs are much more available than common-cathode.
This is why we’re changing around the logic.

Color Mixing

With just 3 LEDs you can make any* color



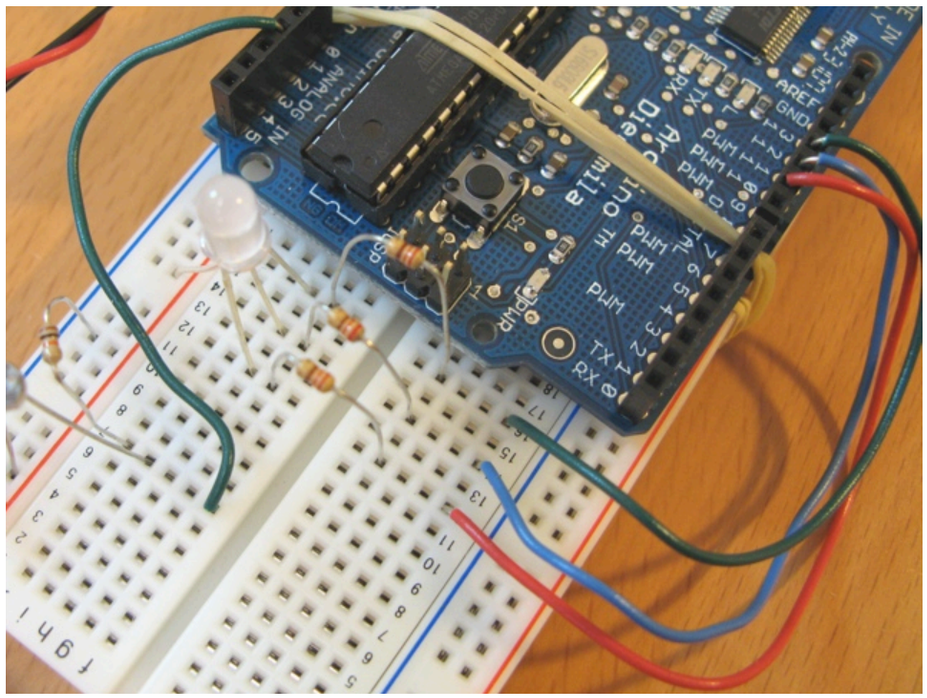
With RGB you can make any color
(except black)

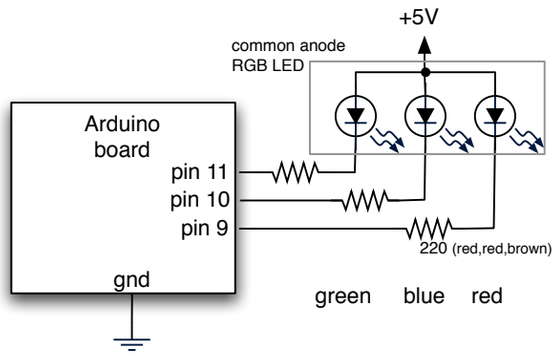
Mixing light is the additive color model

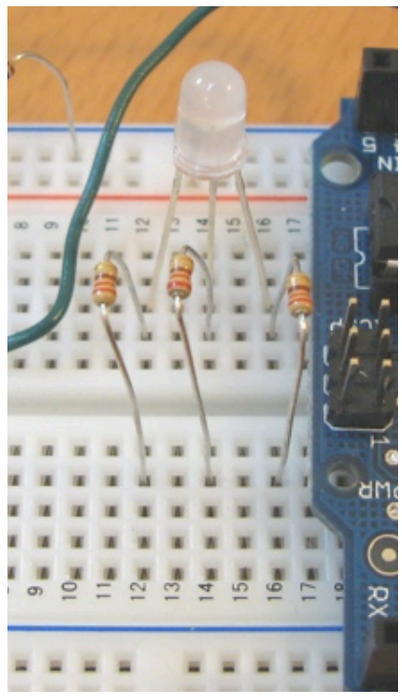
(paint is subtractive color, and can give you brown)

*besides the additive/substractive color different, it's hard to get the mix to be just right for a variety of annoying reasons:
- the physics of LEDs mean that different color LEDs put out different amounts of light
- our eyes respond non-linearly across the spectrum, i.e. we're more sensitive to green than red
- the lenses in most RGB LEDs don't focus each color to the same spot

Laying out RGB LED Circuit







slightly bend the longest lead and plug it into the +5v (red) bus

plug remaining leads into rows (12,14,&16 here)

connect 220 (red-red-brown) resistors across middle to matching rows

run wires from resistors to pins 9,10,11 of Arduino, can color-code if you want

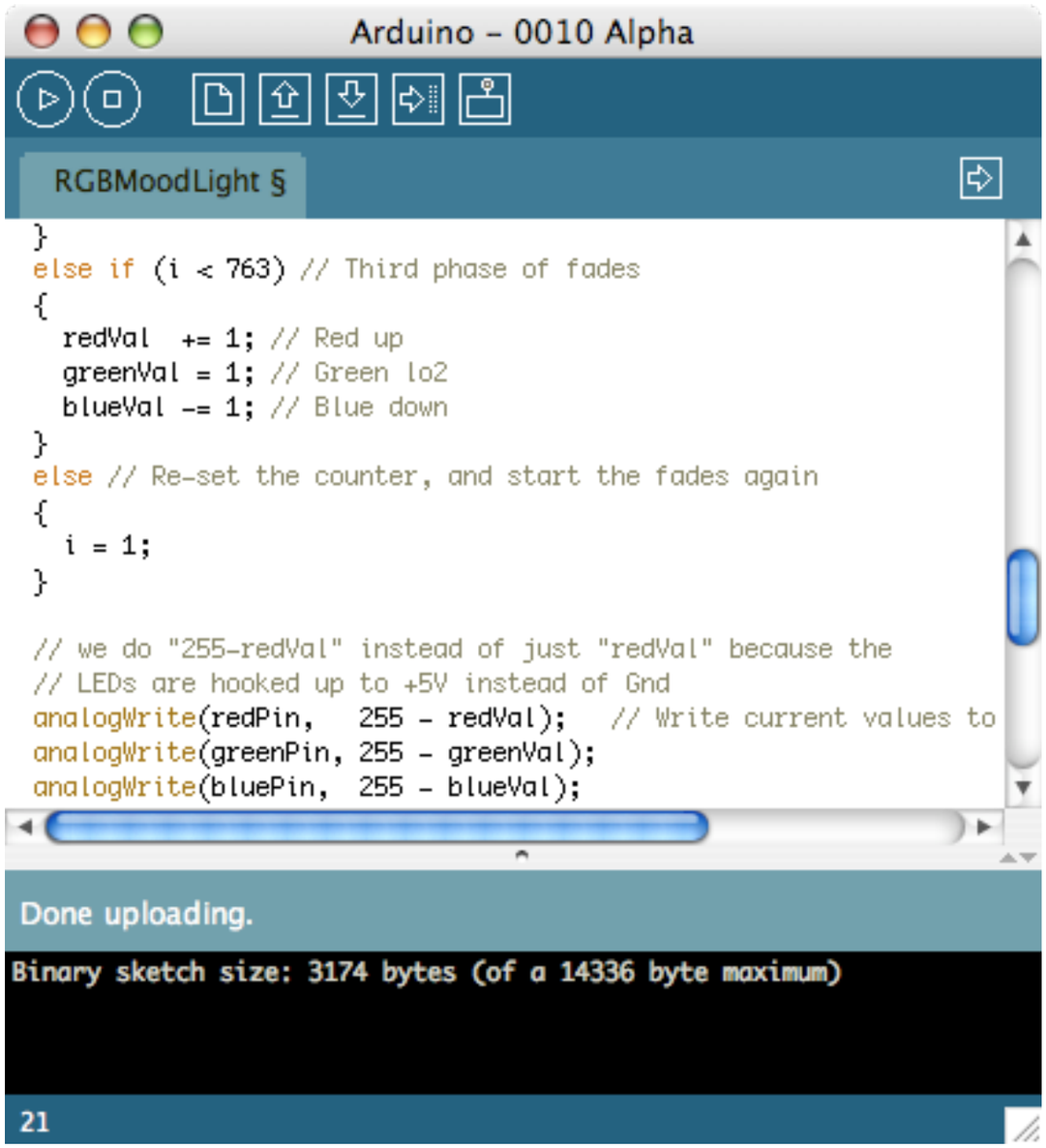
Ignore the green wire in the pictures, that's another circuit.
Keep the pot from last circuit if you can.

RGB Color Fading

"RGBMoodLight"

Slow color fading
and mixing

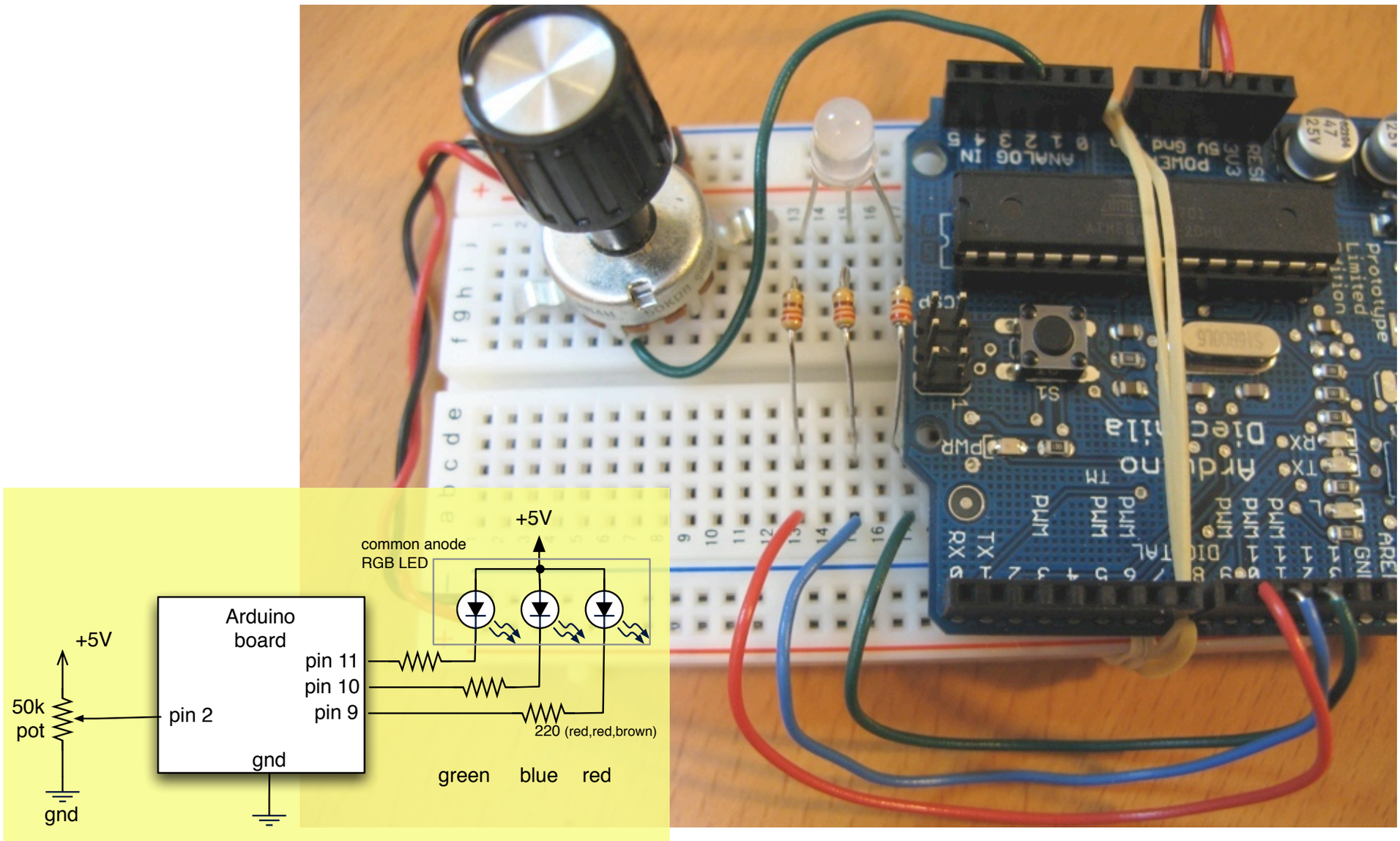
Also outputs the current
color values to the serial port



This sketch is located in the handout.
We'll get to the serial port stuff in a minute.

It just ramps up and down the red,green,& blue color values and writes them with analogWrite()
from http://www.arduino.cc/en/Tutorial/DimmingLEDs

Pot-controlled RGB

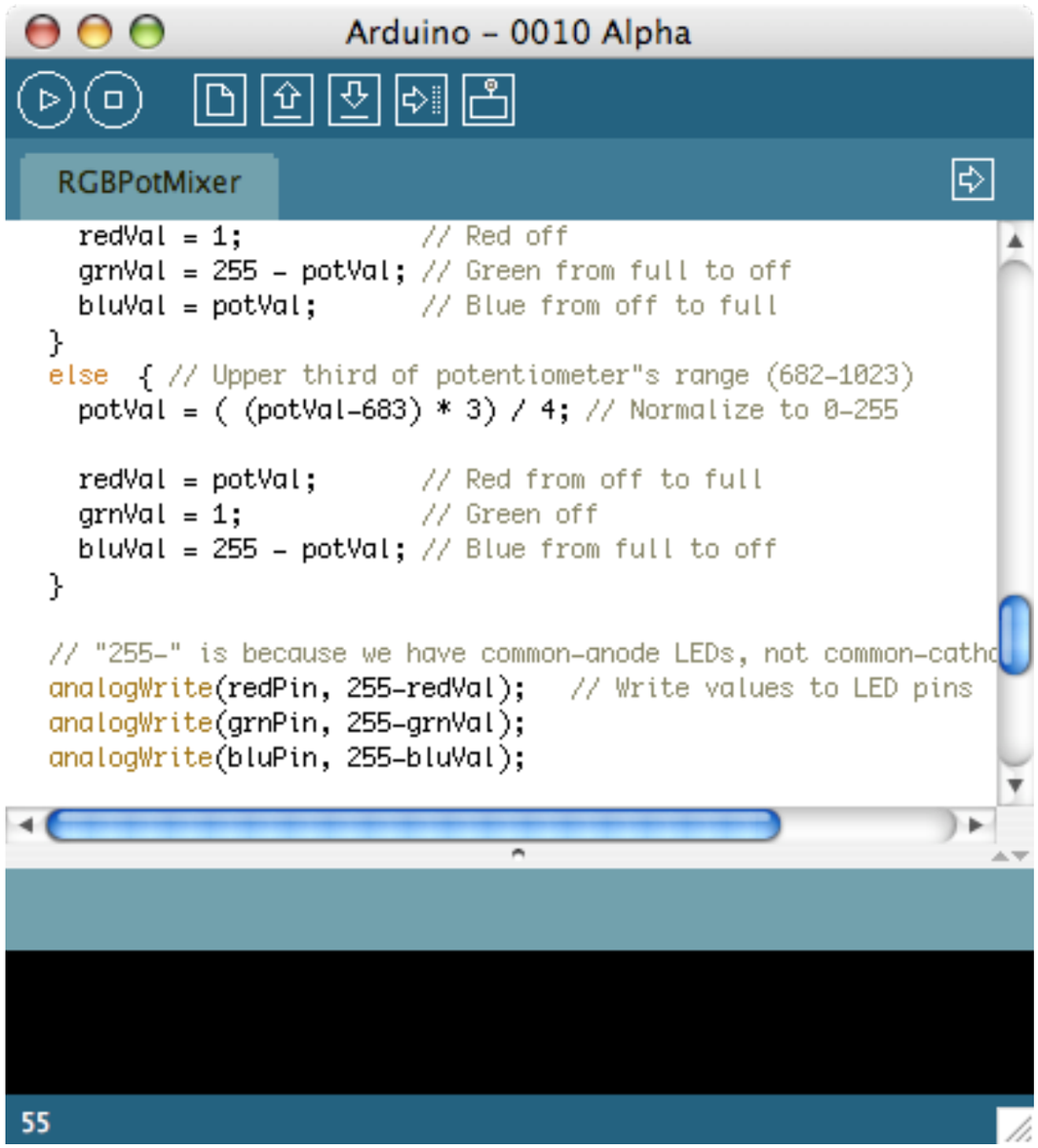


Pot-controlled RGB

"RGBPotMixer"

Use the pot from before to control the color mix

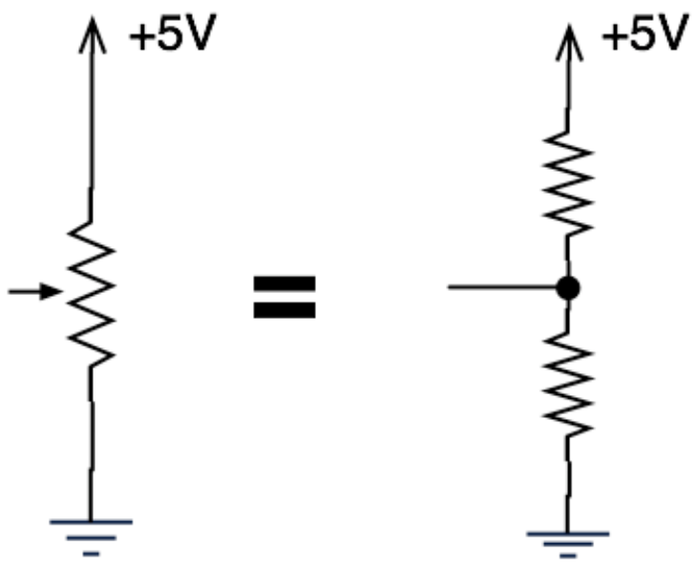
The code turns the single ranged input value into "sectors" where each sector is a color



Also see "RGBPotMixer2" for a variation.
How would you change it to adjust brightness?

Sensing the Dark

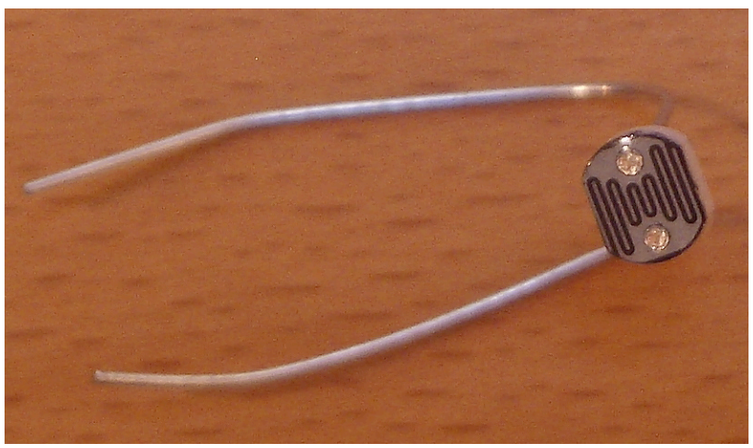
- Pots are example of a *voltage divider*
- Voltage divider splits a voltage in two
- Same as two resistors, but you can vary them



Sensing the Dark: Photocells

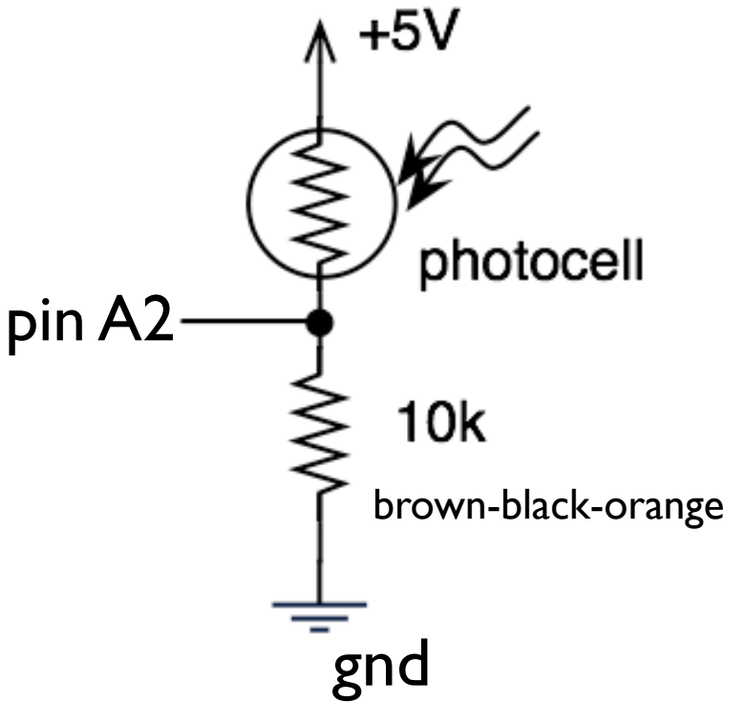
- aka. photoresistor, light-dependent resistor
- A *variable* resistor
- Brighter light == lower resistance
- Photocells you have range approx. 0-10k-1M

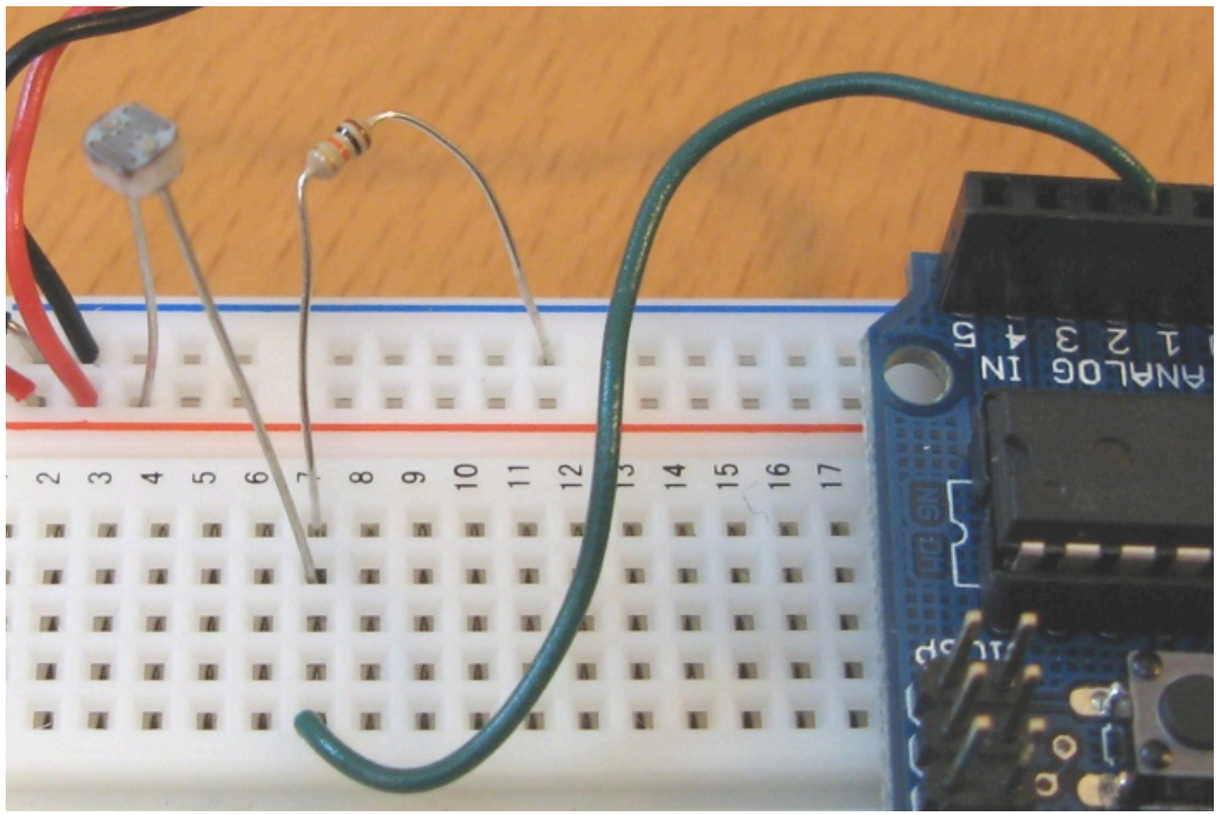
schematic symbol



Pretty cheap too. Can get a grab bag of 100 misc from Jameco for $20

Photocell Circuit



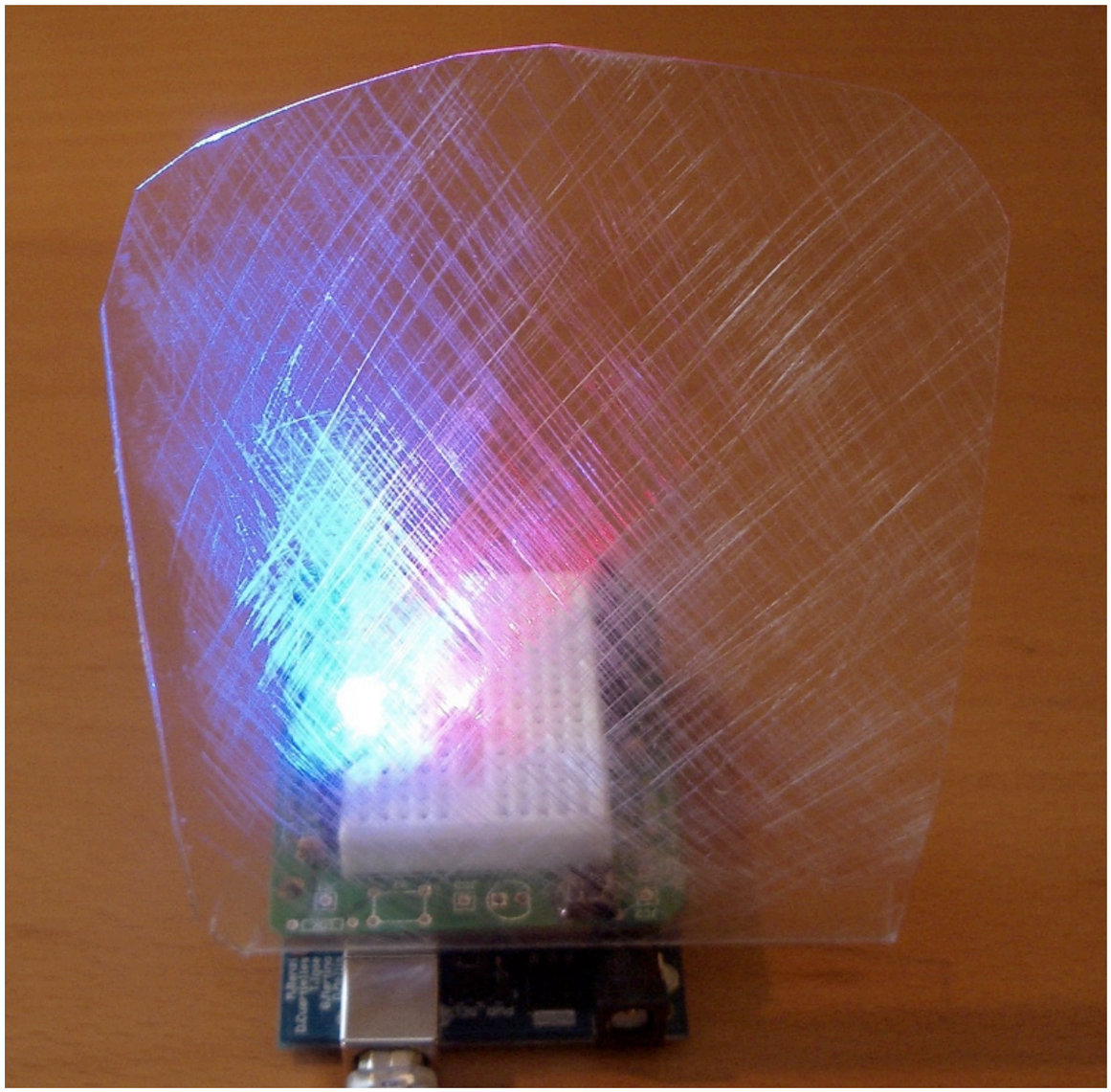


Try it with RGBPotMixer from before

Looks a lot like the pot circuit, doesn't it?

Mood Light

Diffuser made from piece of plastic scratched with sandpaper

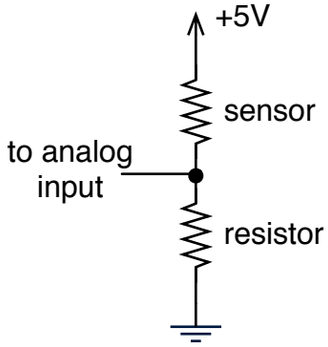


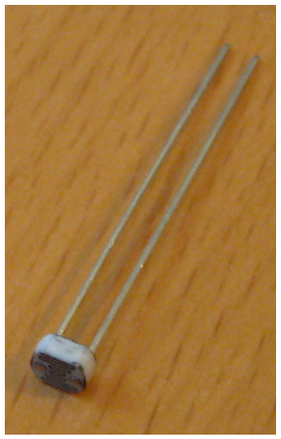
Also, can use plastic wrap scrunched up to make an interesting diffuser.

Resistive sensors

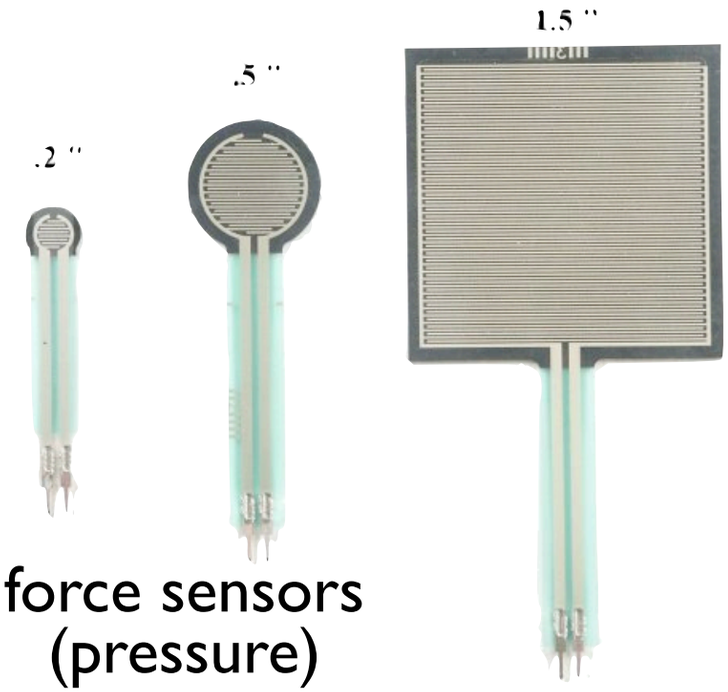
thermistor
(temperature)

circuit is the same
for all these

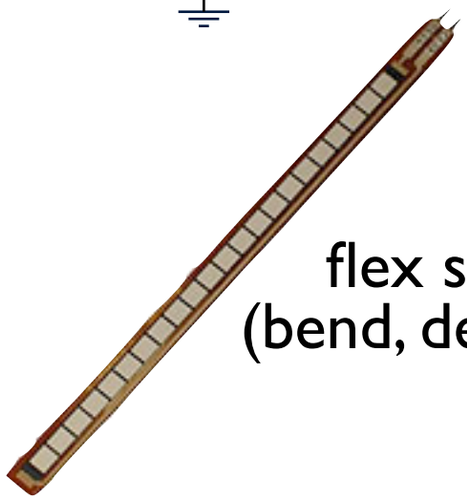




photocell
(light)



force sensors
(pressure)



flex sensor
(bend, deflection)

also air pressure
and others

Thermistor image from: http://www.facstaff.bucknell.edu/mastascu/elessonsHTML/Sensors/TempR.html
Also see: http://www.ladyada.net/make/midisense/makesensor.html

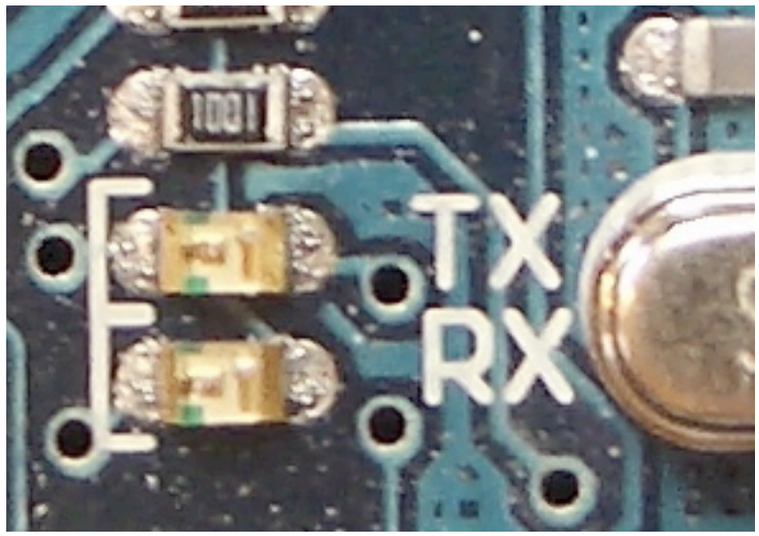
Communicating with Others

- Arduino can use same USB cable for programming and to talk with computers
- Talking to other devices uses the "`Serial`" commands
  - `Serial.begin()` – prepare to use serial
  - `Serial.print()` – send data to computer
  - `Serial.read()` – read data from computer

Can talk to not just computers.
Most things more complex than simple sensors/actuators speak serial.

Watch the TX/RX LEDS

- TX – sending to PC
- RX – receiving from PC
- Used when programming or communicating



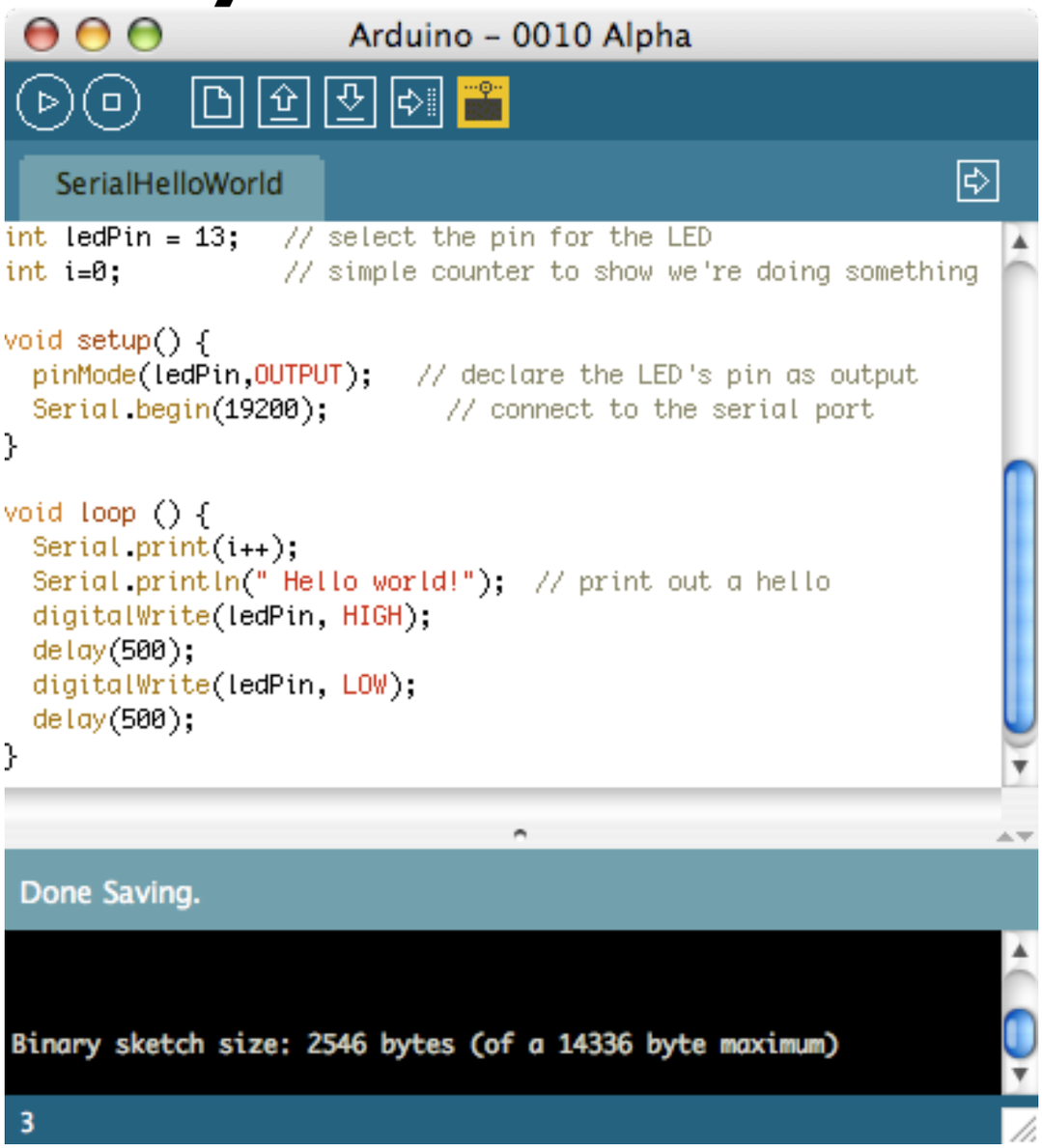
Arduino Says "Hi"

"SerialHelloWorld"

Sends "Hello world!" to your computer

Click on "Serial Monitor" button to see output

Watch TX LED compared to pin 13 LED



This sketch is located in the handout, but it's pretty short.
Use on-board pin 13 LED, no need to wire anything up.

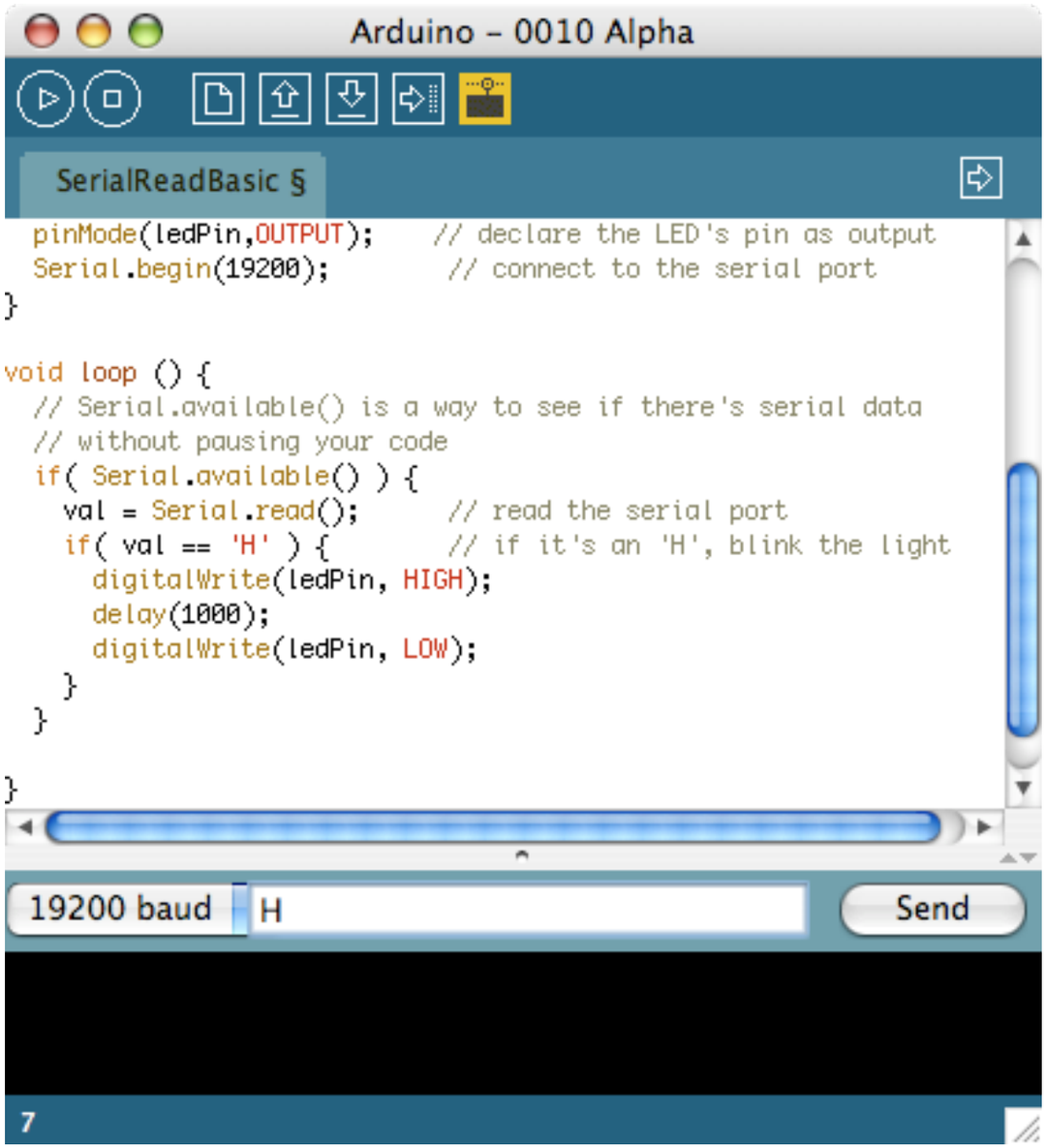
Telling Arduino What To Do

"SerialReadBasic"

You type "H", LED blinks

In "Serial Monitor", type "H", press Send

`Serial.available()` tells you if data present to read



This sketch is in the handout
Always check Serial.available() or if Serial.read() != -1 to determine if there's actual data to read.

Can modify it to print "hello world" after it receives something, but before it checks for 'H'.
This way you can verify it's actually receiving something.

Arduino Communications

is just serial communications

- Psst, Arduino doesn't really do USB
- It really is "serial", like old RS-232 serial
- All microcontrollers can do serial
- Not many can do USB
- Serial is easy, USB is hard

serial terminal from the olde days

Serial Communications

- "Serial" because data is broken down into bits, each sent one after the other down a single wire.
- The single ASCII character 'B' is sent as:

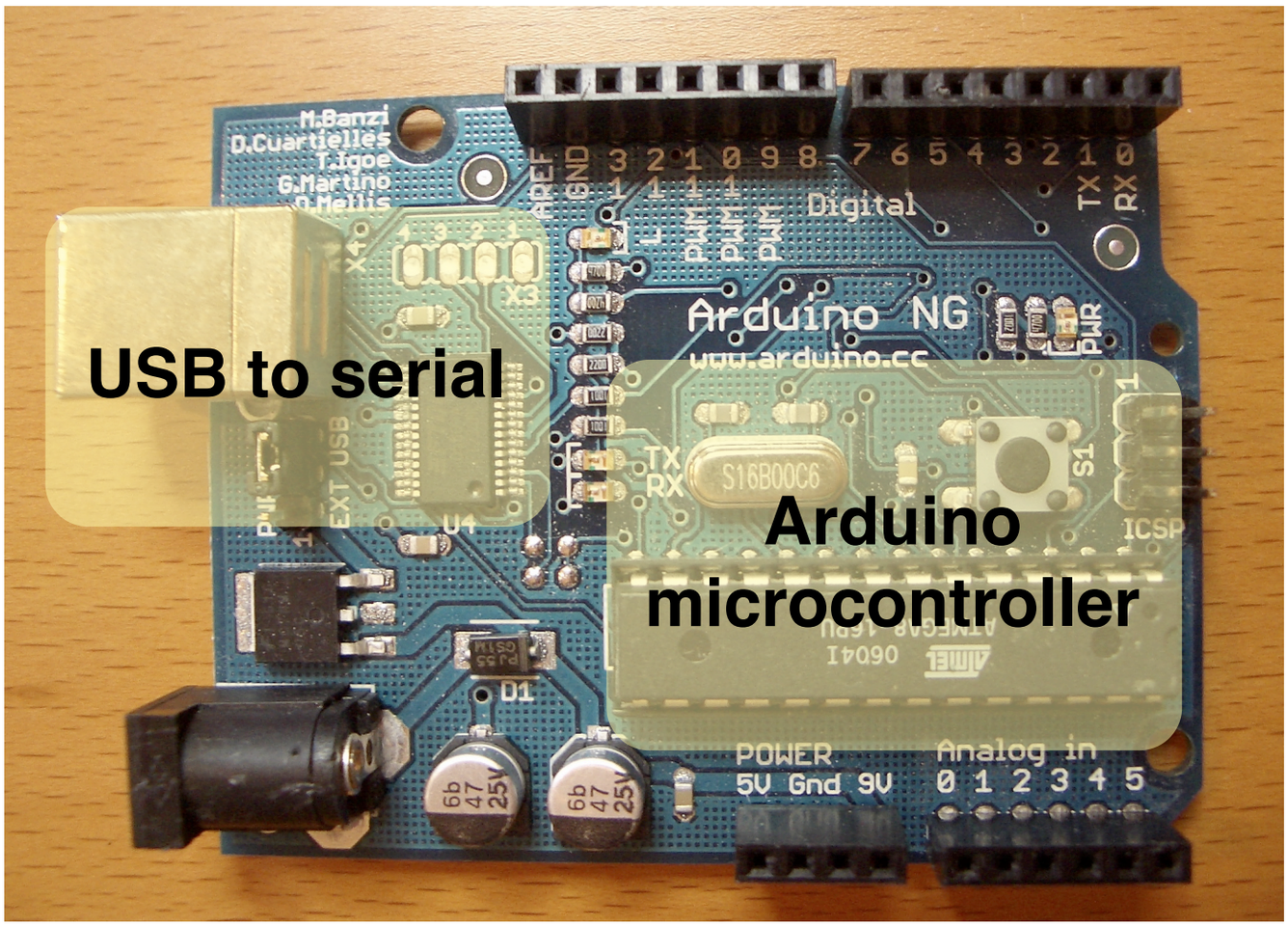
```
'B' =  0 1 0 0 0 0 1 0
    =  L H L L L L H L
    =
```

- Toggle a pin to send data, just like blinking an LED
- You could implement sending serial data with `digitalWrite()` and `delay()`
- A single data wire needed to send data. One other to receive.

Note, a single data wire. You still need a ground wire.

Arduino & USB-to-serial

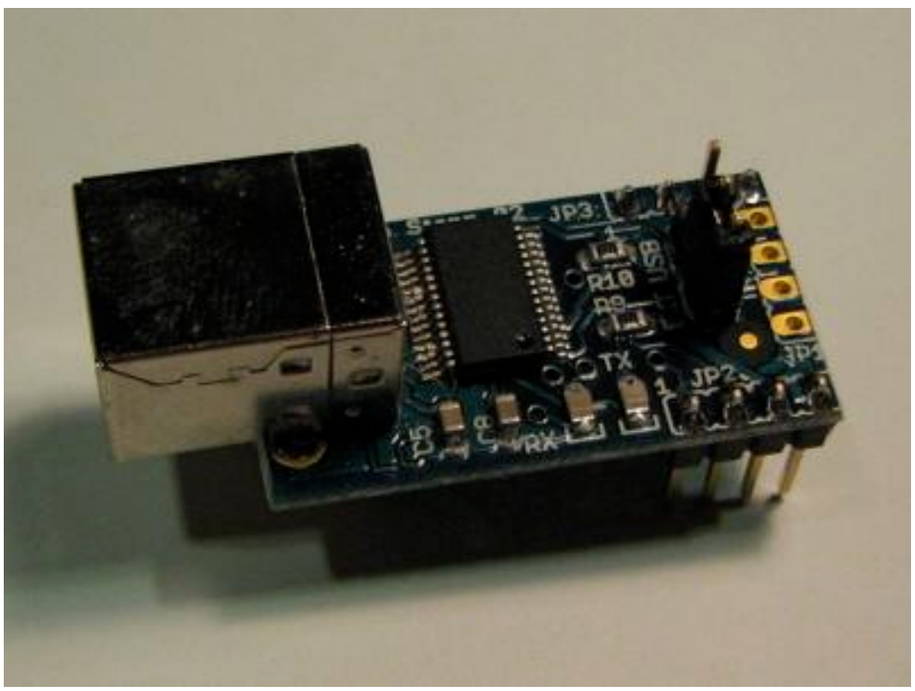
Arduino board is really two circuits



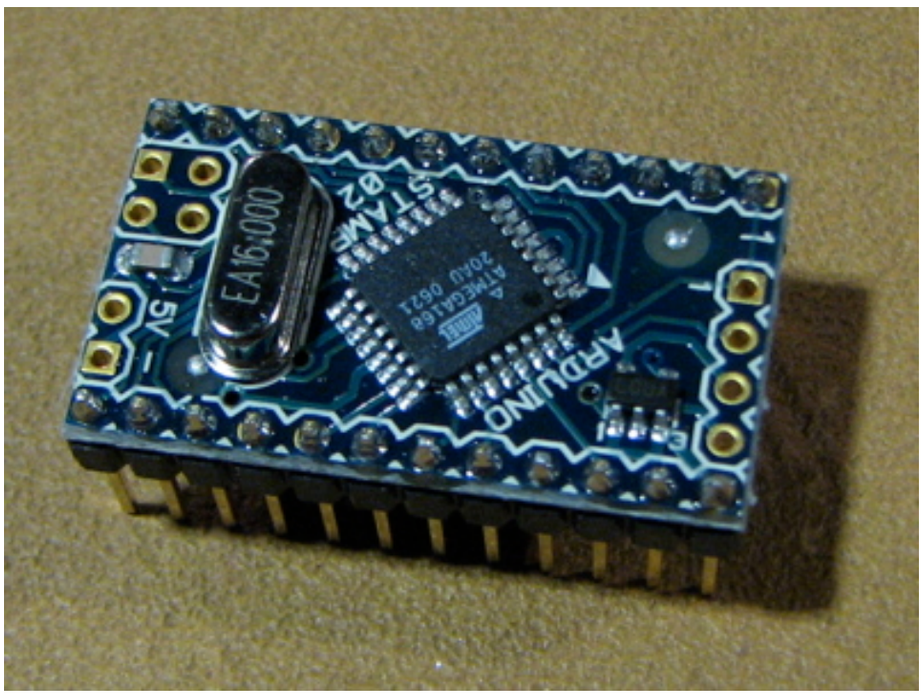
Original Arduino boards were RS-232 serial, not USB.

Arduino Mini

Arduino Mini separates the two circuits



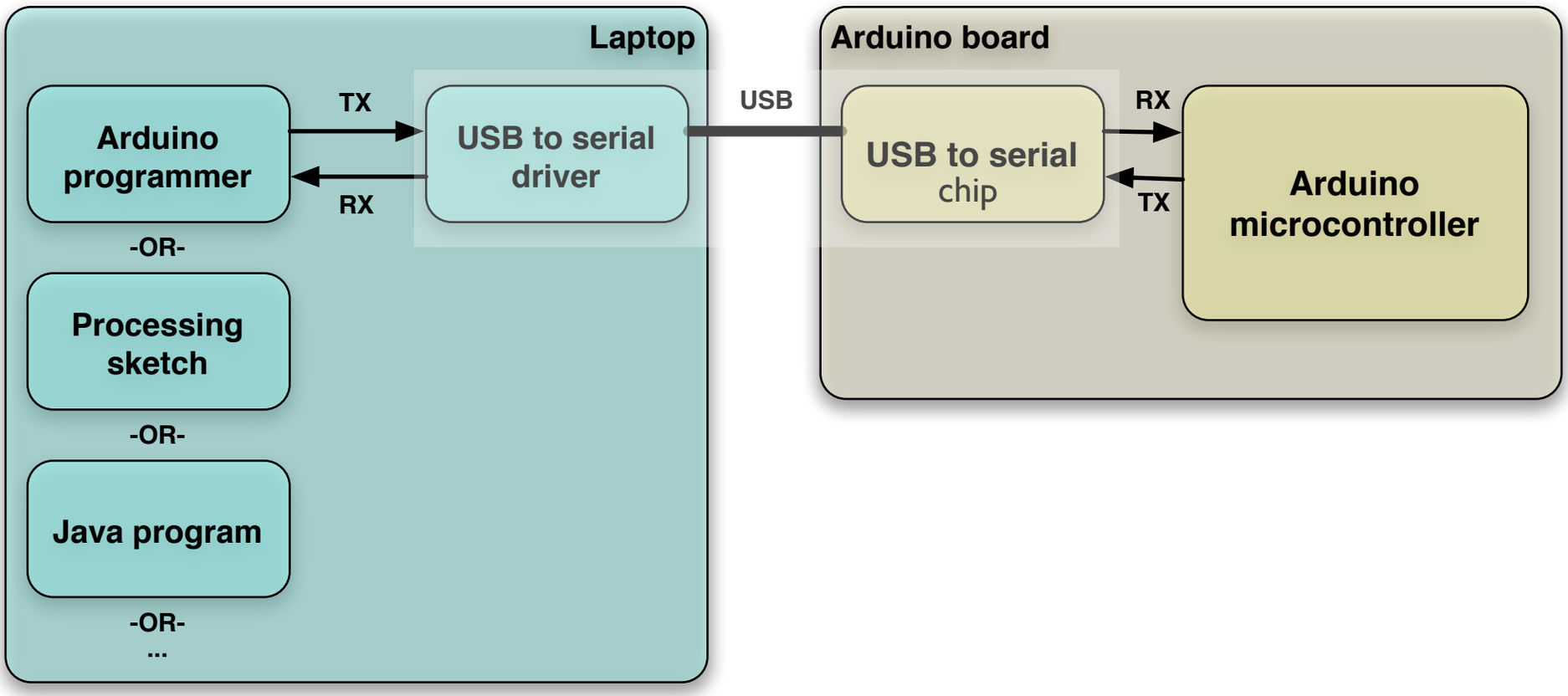
Arduino Mini USB adapter



Arduino Mini

aka. “Arduino Stamp”
If you don’t talk with a computer, the USB-to-serial functionality is superfluous.

Arduino to Computer



USB is totally optional for Arduino
But it makes things easier

Original Arduino boards were RS-232 serial, not USB.
All programs that talk to Arduino (even the Arduino IDE) think that they're talking via a serial port.

Arduino & USB

- Since Arduino is all about serial
- And not USB,
- Interfacing to things like USB flash drives, USB hard disks, USB webcams, etc. is *not* possible

Also, USB is a host/peripheral protocol. Being a USB "host" means needing a lot of processing power and software, not something for a tiny 8kB microcontroller.
It can be a peripheral. In fact, there is an open project called "AVR-USB" that allows AVR chips like used in Arduino to be proper USB peripherals. See: http://www.obdev.at/products/avrusb/

Controlling the Computer

- Can send sensor data from Arduino to computer with `Serial.print()`
- There are many different variations to suite your needs:

```
int val = 123;
Serial.print(val);      // sends 3 ASCII chars "123"
Serial.print(val,DEC);  // same as above
Serial.print(val,HEX);  // sends 2 ASCII chars "7B"
Serial.print(val,BIN);  // sends 8 ASCII chars "01111011"
Serial.print(val,BYTE); // sends 1 byte, the verbatim value
```

Controlling the Computer

You write one program on Arduino, one on the computer

In Arduino: read sensor, send data as byte

```
void loop() {
  val = analogRead(analogInput);   // read the value on analog input
  Serial.print(val/4,BYTE);        // print a byte value out
  delay(50);                       // wait a bit to not overload the port
}
```

In Processing: read the byte, do something with it

```
import processing.serial.*;

Serial myPort;  // The serial port

void setup() {
  String portname = "/dev/tty.usbserial-A3000XV0";
  myPort = new Serial(this, myPort, 9600);
}

void draw() {
  while (myPort.available() > 0) {
    int inByte = myPort.read();
    println(inByte);
  }
}
```

But writing Processing programs is for later

Controlling the Computer

- Receiving program on the computer can be in any language that knows about serial ports
  - C/C++, Perl, PHP, Java, Max/MSP, Python, Visual Basic, etc.
- Pick your favorite one, write some code for Arduino to control

If interested, I can give details on just about every language above.

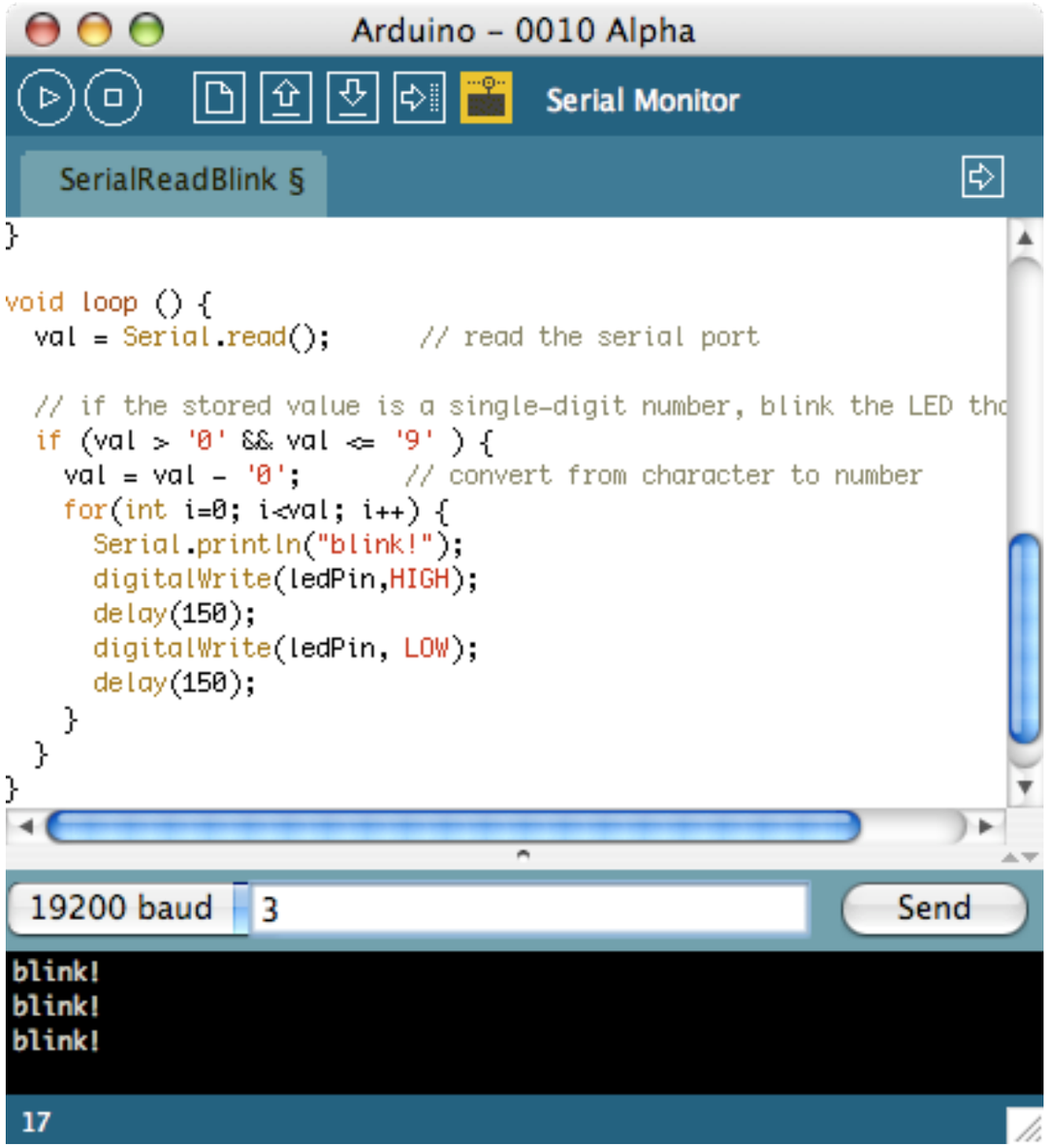
Controlling Arduino, Again

"SerialReadBlink"

Type a number 1-9 and LED blinks that many times

Converts typed ASCII value into usable number

Most control issues are data conversion issues



This sketch is also in the handout

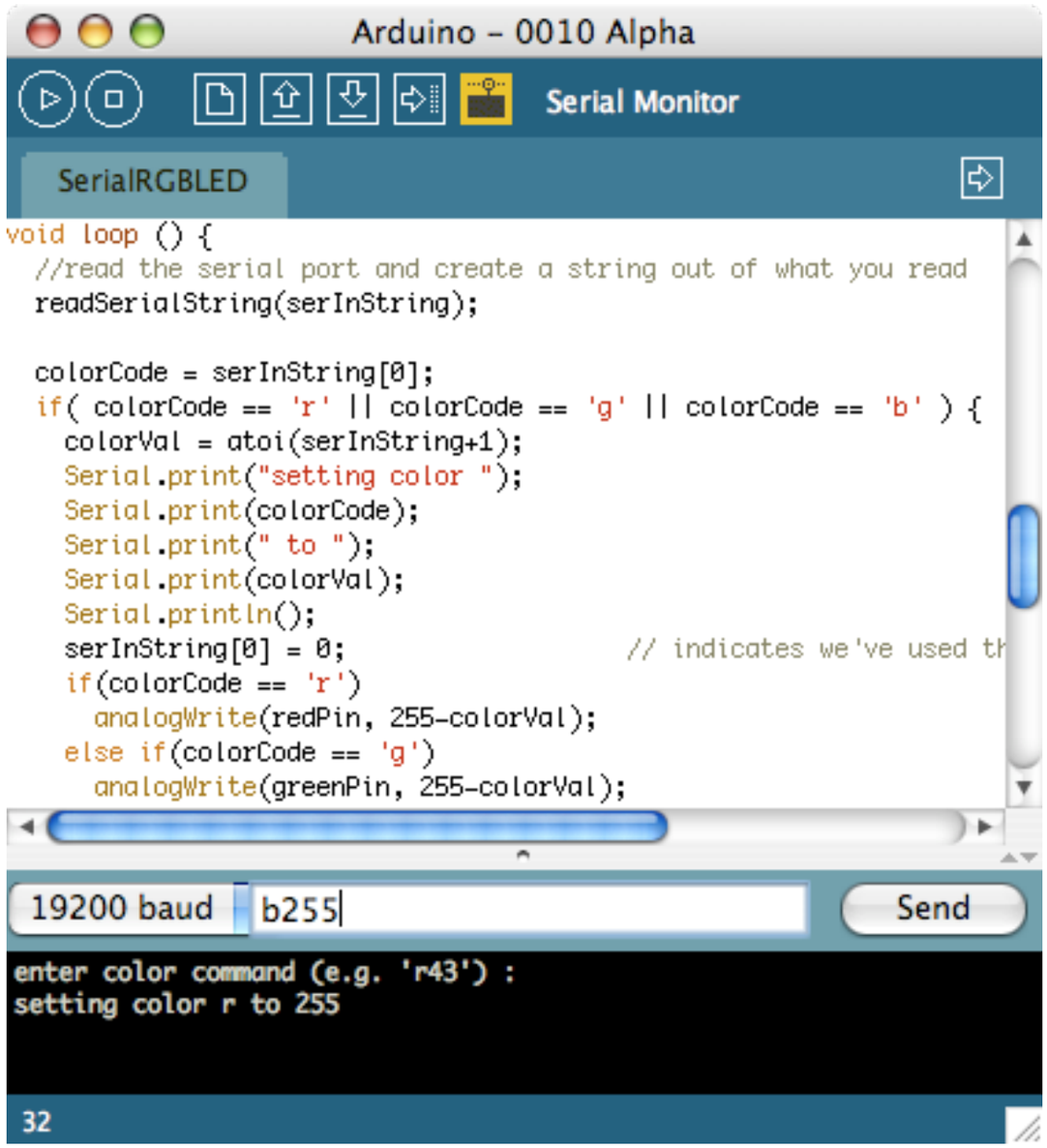
Serial-controlled RGB

"SerialRGBLED"

Send color commands to Arduino

e.g."r200","g50","b0"

Sketch parses what you type, changes LEDs



This sketch is in the handout.
Color command is two parts: colorCode and colorValue
colorCode is a character, 'r', 'g', or 'b'.
colorValue is a number between 0-255.
Sketch shows rudimentary character string processing in Arduino.
This is still one of the hardest tasks, unfortunately.

Reading Serial Strings

- The function "`Serial.available()`" makes reading strings easier
- Can use it to read all available serial data from computer
- The "`readSerialString()`" function at right takes a character string and sticks available serial data into it

```
//read a string from the serial and store it in an array
//you must supply the array variable
void readSerialString (char *strArray) {
  int i = 0;
  if(!Serial.available()) {
    return;
  }
  while (Serial.available()) {
    strArray[i] = Serial.read();
    i++;
  }
  strArray[i] = 0;  // indicate end of read string
}
```

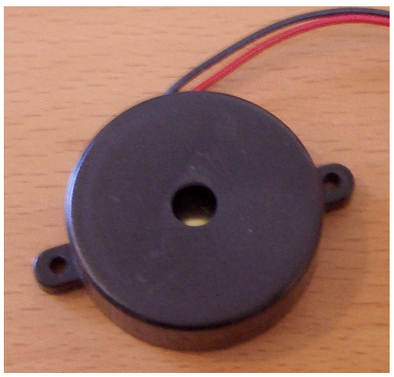
Pay no attention to the pointer symbol ("*")
Must be careful about calling readSerialString() too often or you'll read partial strings

Piezoelectrics

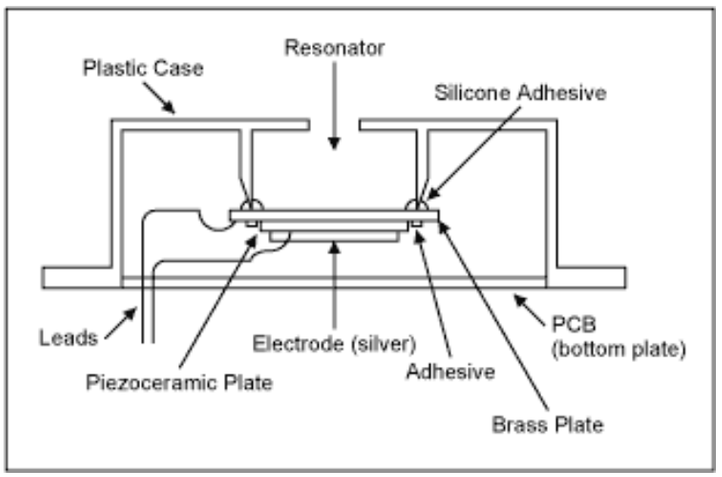
- Big word – *piezein* is greek for "squeeze"
- Some crystals, when squeezed, make a spark
- Turns out the process goes the other way too
- Spark a quartz crystal, and it flexes
- Piezo buzzers use this to make sound
  (flex something back and forth, it moves air)

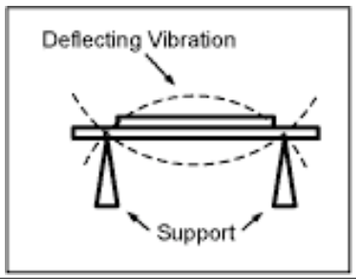
Piezo buzzers don't have quartz crystals, but instead a kind of ceramic that also exhibits piezoelectric properties.
I pronounce it "pie-zoh". Or sometimes "pee-ay-zoh".

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.

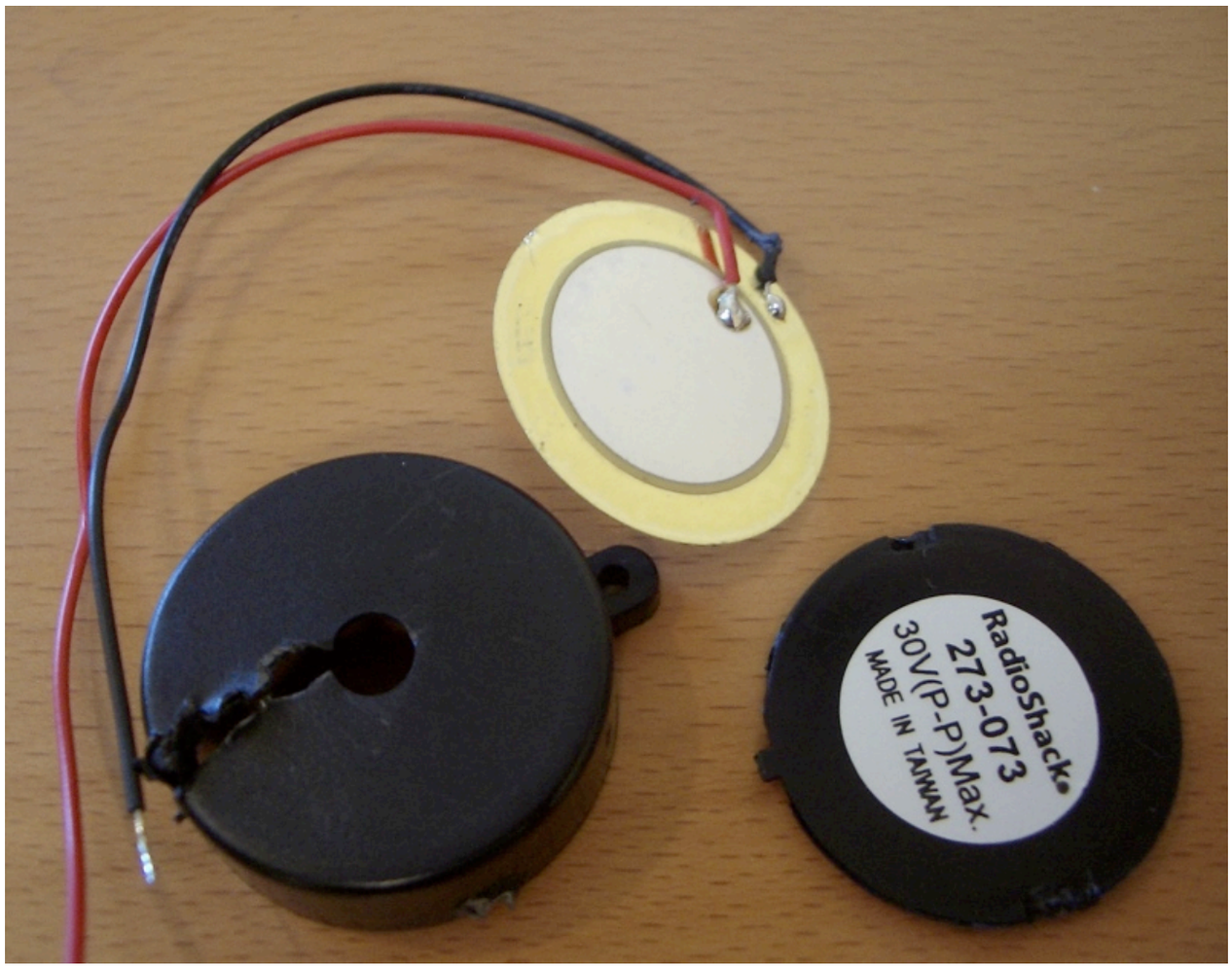
diagrams from: http://www.maxim-ic.com/appnotes.cfm/appnote_number/988

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

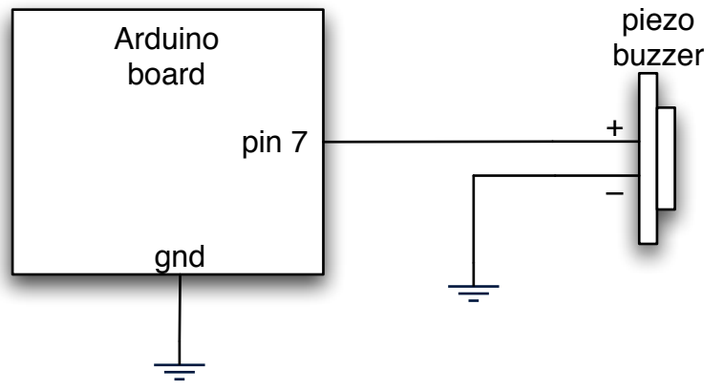
Only take it out of its case to use it as a sensor

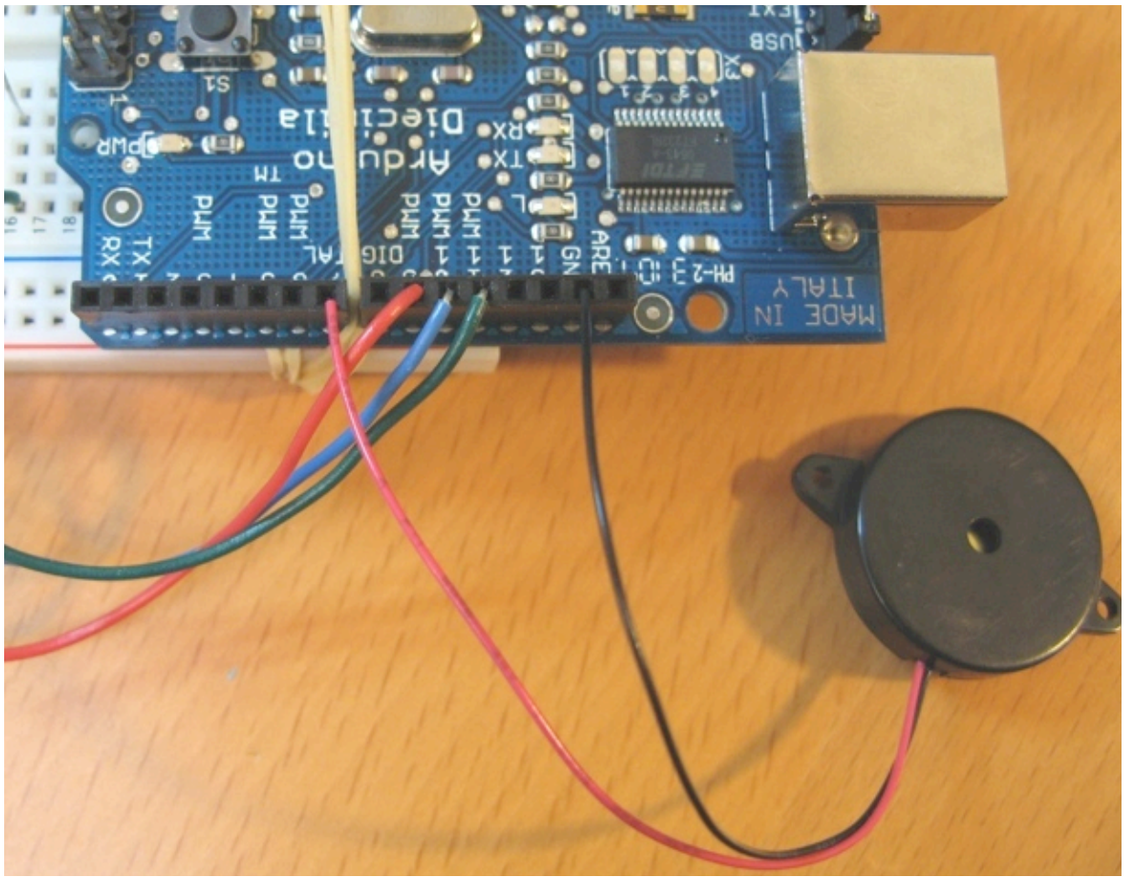


another $1.99 I won't be getting back from Radio Shack

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.
Or you can jam a jumper wire in the holes to hold in the piezo leads.

Play a Melody

"SoundSerial"

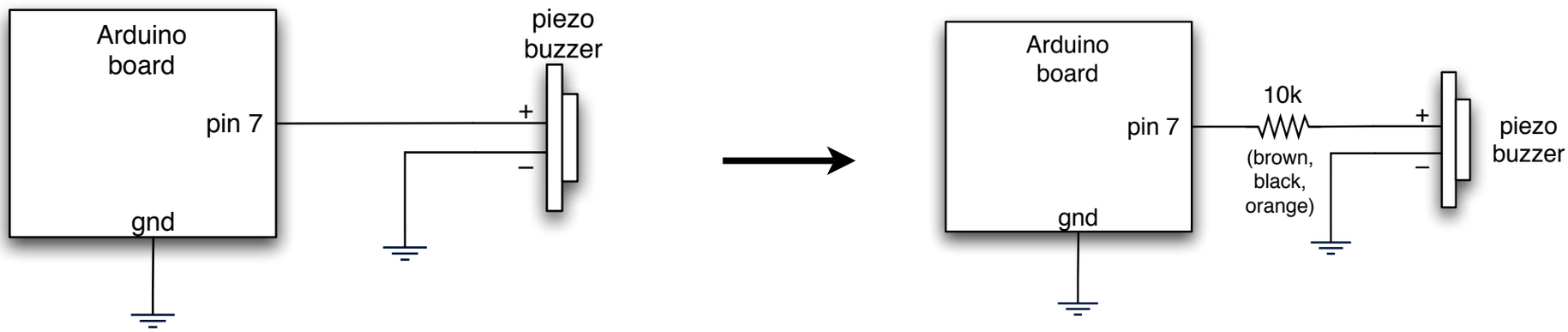
Play the piezo beeper with the Serial Monitor

Type multiple letters from "cdefgabC" to make melodies

This sketch is in the handout,
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



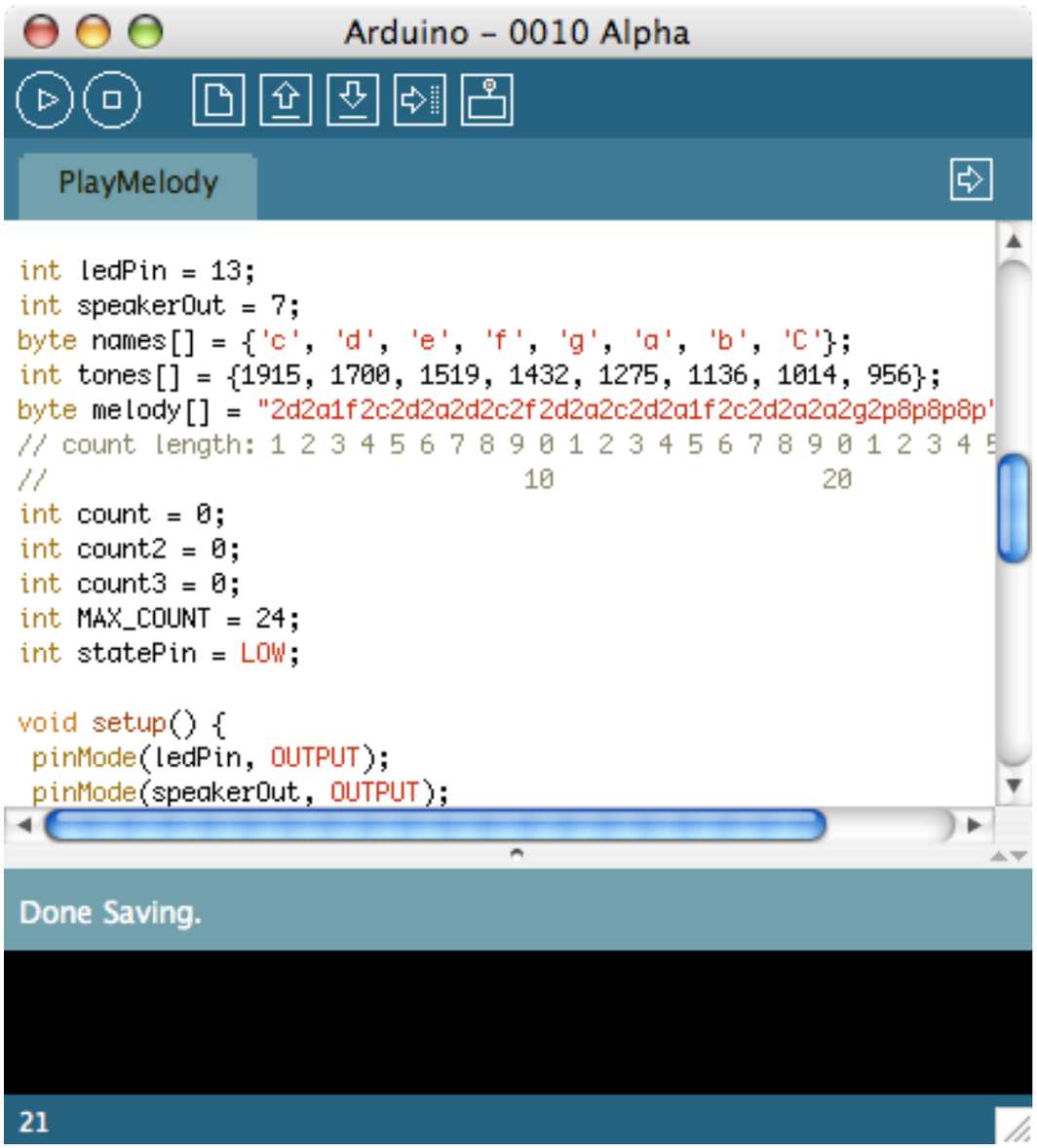
Like most things in electronics, if you want less of something, add a resistor.
A better value would probably be 1k, but we don't have that on hand.
This may not seem important now, but wait for the next project.

Play a Stored Melody

"PlayMelody"

Plays a melody stored in the Arduino

Could be battery-powered, play melody on button trigger, control playback speed with photocell, etc.



Melody definition is sort of like the old cell ringtone style
Melody playing logic is a little hard to follow, since it is timing critical.

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.

Light Theremin

"Theremin"

Move hand over photocell to change pitch

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

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

Other Serial Devices

to Wi-Fi to Ethernet

to graphic LCD

to 8-servo controller

Lantronix Wi-Port and Lantronix Xport http://lantronix.com/
Seetron Serial Graphic display and Mini SSC http://www.seetron.com/slcds.htm http://www.seetron.com/ssc.htm

Serial Examples

to Roomba

You've already seen this. :)
http://hackingroomba.com/

Going Further

- Piezo buzzers
  - Can hook up multiple buzzers for polyphonic sound
  - Can play waves other than just square waves using PWM techniques
  - Can also be used as input devices (we'll cover that later)

Going Further

- Serial communications
  - Not just for computer-to-Arduino communications
  - Many other devices speak serial
  - Older keyboards & mice speak are serial (good for sensors!)
  - Interface boards (graphic LCDs, servo drivers, RFID readers, Ethernet, Wi-Fi)

Going Further

- RGB LEDS
  - You can pretty easily replicate the Ambient Orb ($150) functionality
  - Make a status display for your computer
  - Computer-controlled accent lighting (a wash of color against the walls)

Ambient Orb doesn't connect to computer though. Uses the pager network.
Ambient Devices: http://www.ambientdevices.com/

END Class 2

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

Tod E. Kurt

tod@todbot.com

Feel free to email me if you have any questions.