AS220
LABS

SOUND
PIXEL GRIDS
MOVING PIXELS
FOLK ELECTRONICS

LAB 120

Introduction to Arduino and Electronics

Class 2

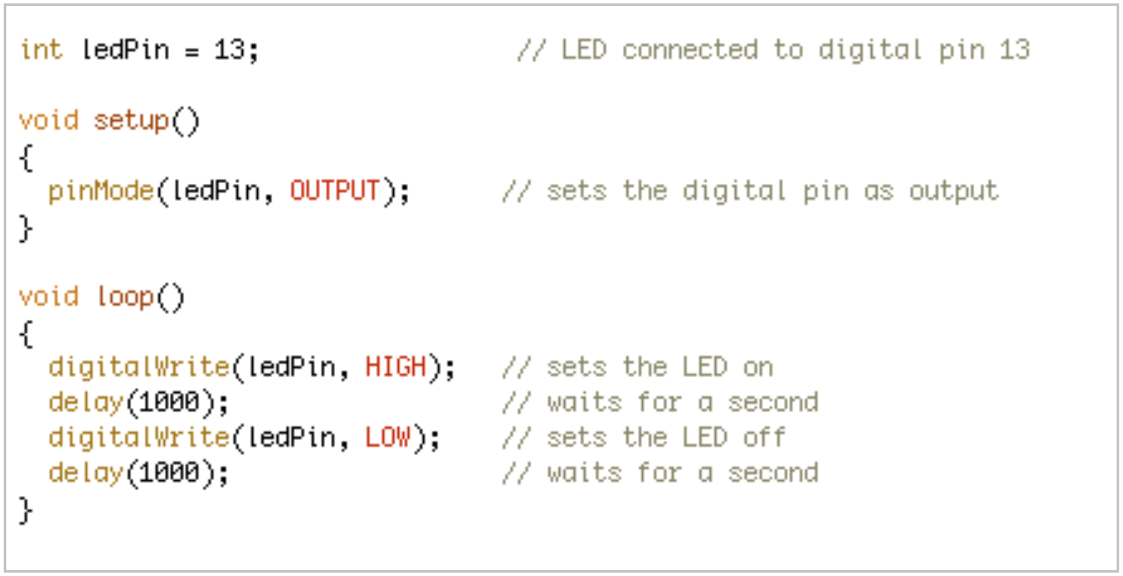
16 June 2009 - AS220 Labs - John Duksta

What’s for Today

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

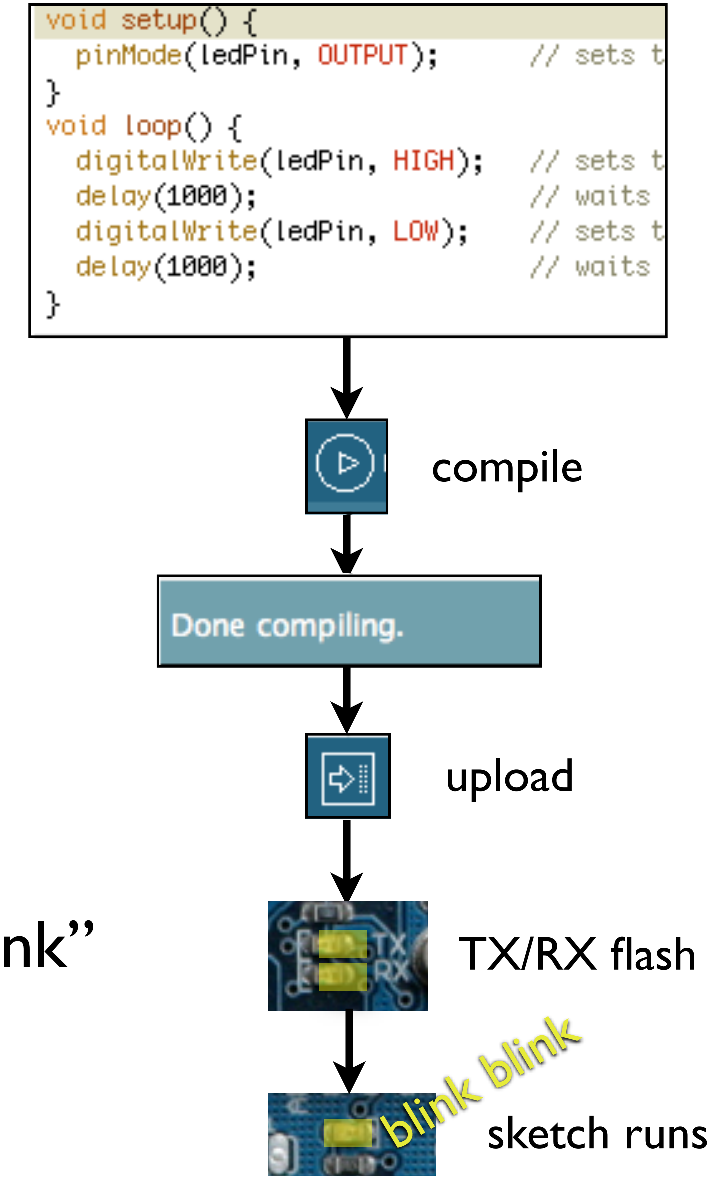
Recap: Blinky LED

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
}
```

compile

Done compiling.

upload

TX/RX flash

sketch runs

Known Good Configuration

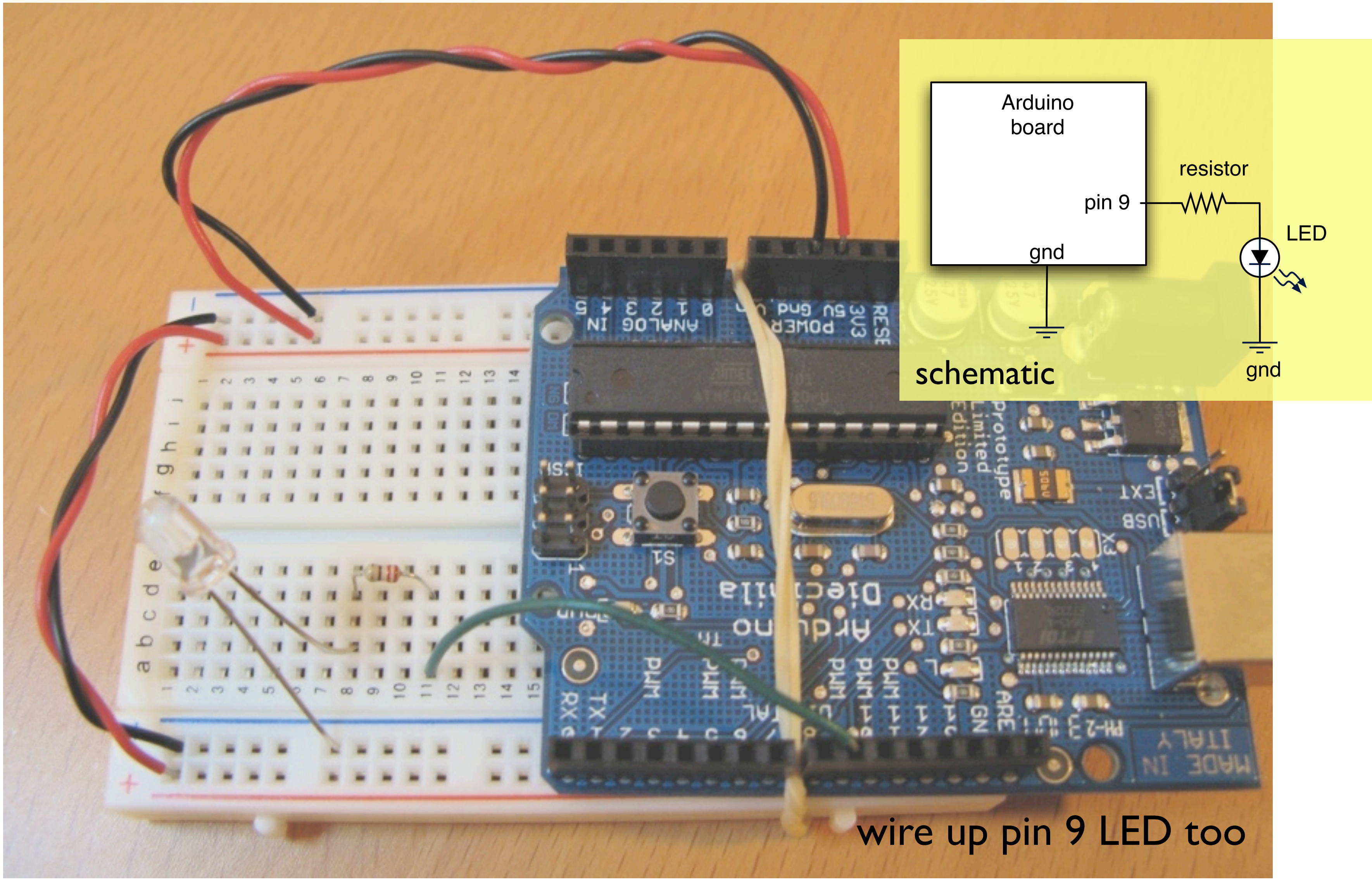
Rule #1 of experimenting:

Before trying anything new,

Get back to a known working state

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

Getting the Board Set Up



schematic

Arduino board

pin 9

gnd

resistor

LED

gnd

wire up pin 9 LED too

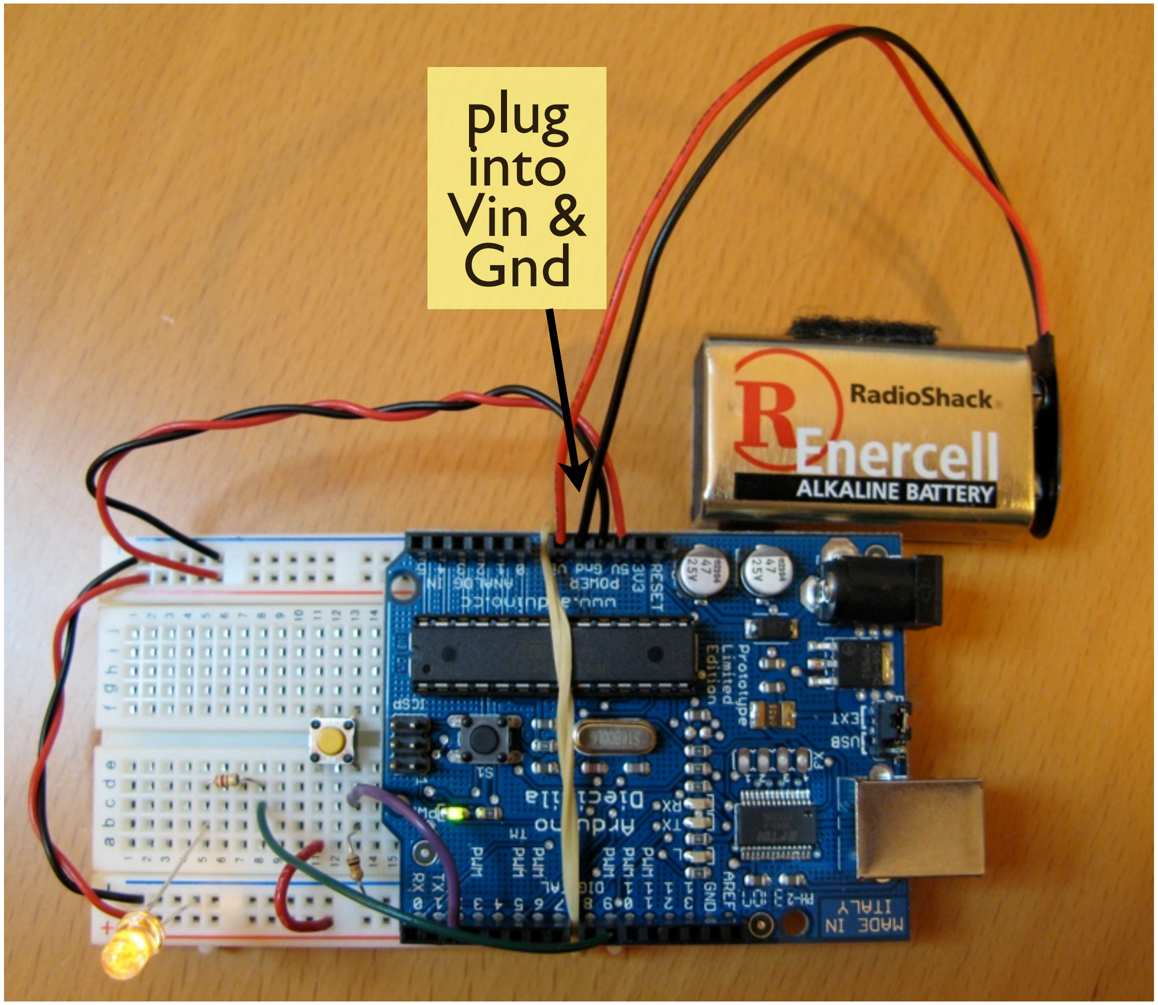
Questions / Review

Any questions, comments, or problems?

Battery Power

Arduino can work totally stand-alone. It’s easy

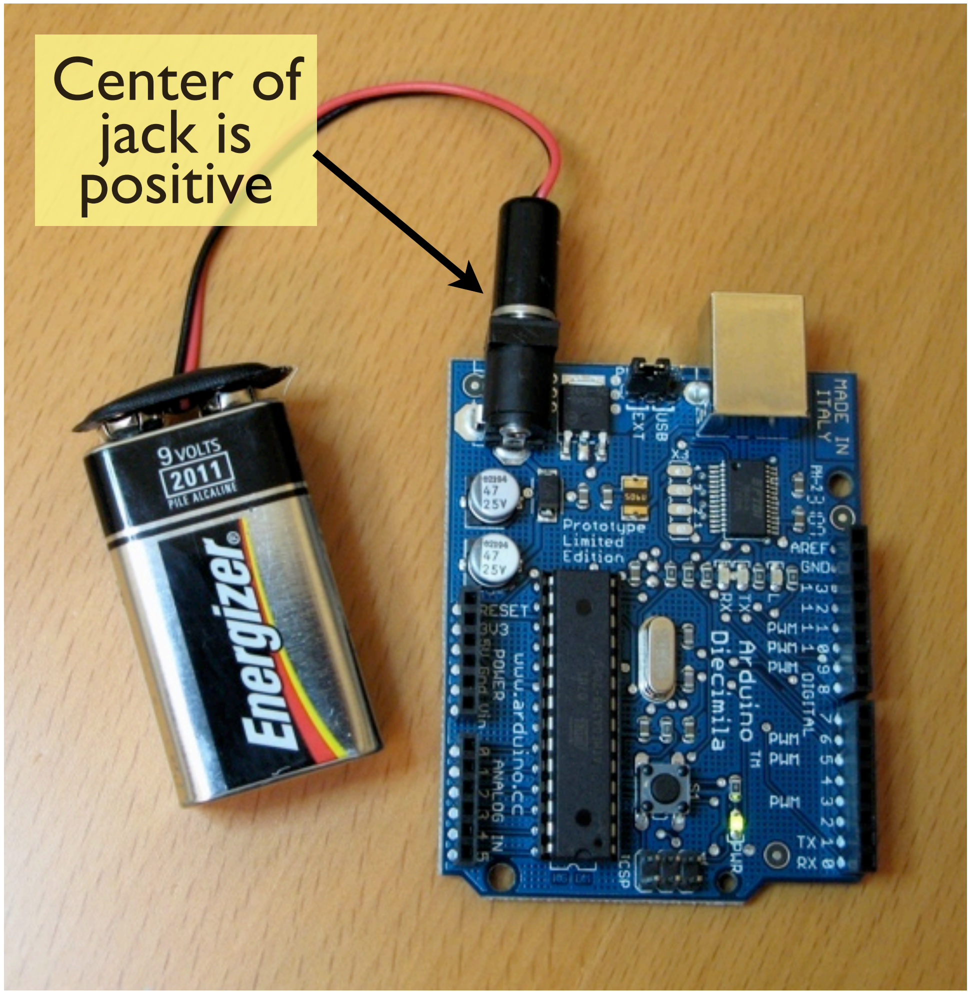
- First, program sketch into Arduino
- Unplug USB cable
- Plug in power (7-12VDC)
- Power LED lights up. It works!
- Reverse steps to reprogram



plug into Vin & Gnd

Battery Power

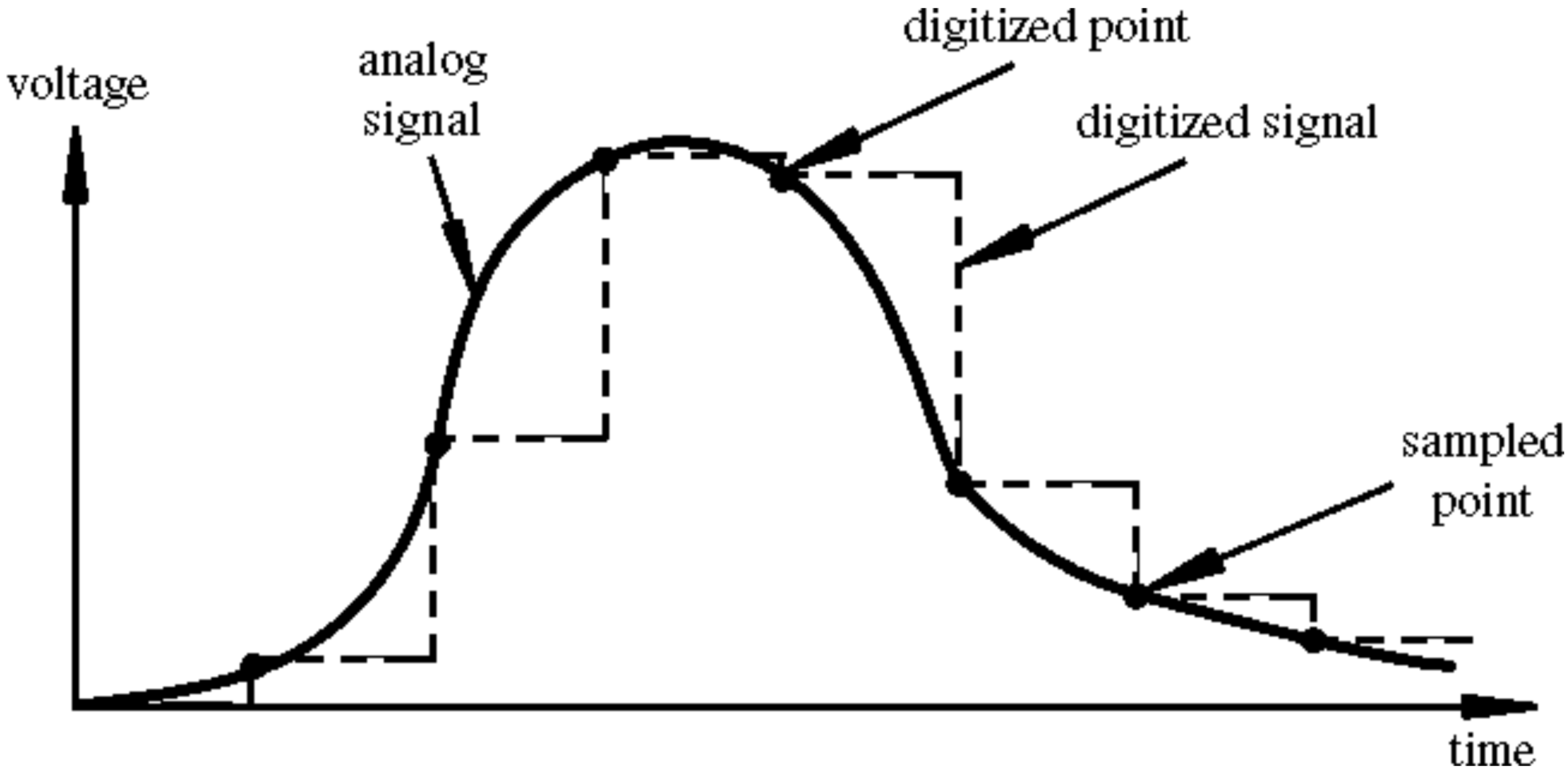
- Plugging into the sockets is kind of fiddly
- Better to plug into the power jack
- Works great, but requires a little soldering



Center of jack is positive

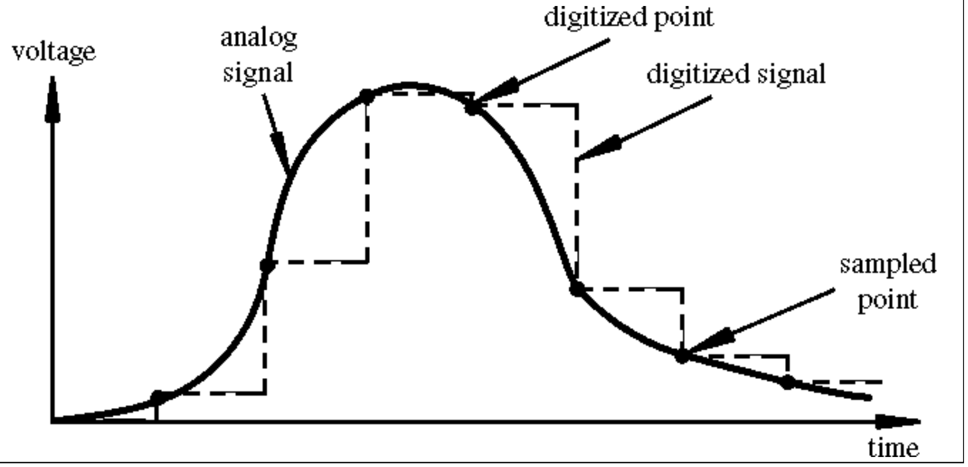
Analog Input

To computers, analog is chunky



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



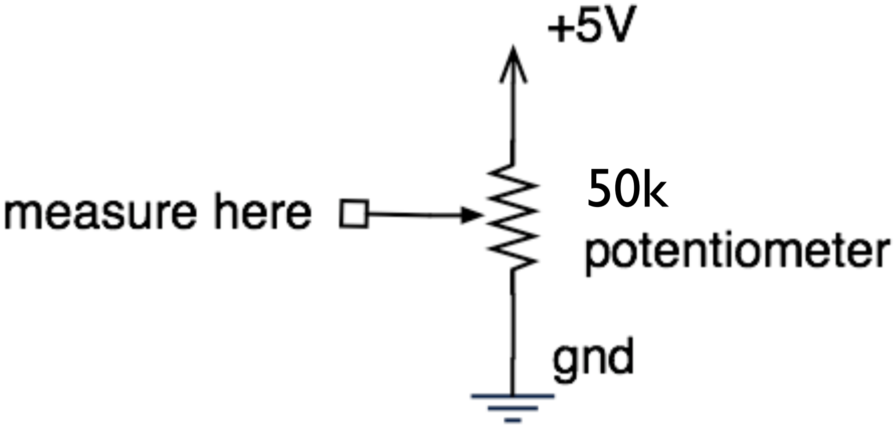
Analog Input

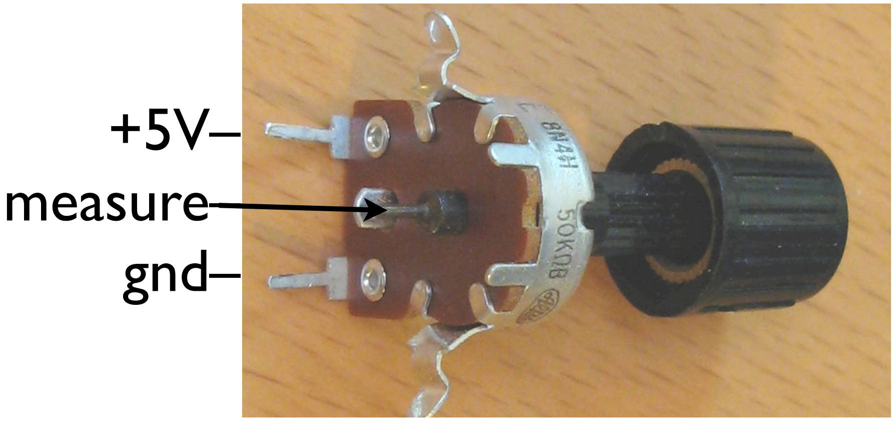
- Arduino (ATmega328) has six ADC inputs
- (ADC = Analog to Digital Converter)
- Reads voltage between 0 to 5 volts
- Resolution is 10-bit (1024 values)
- In other words, 5/1024 = 4.8 mV smallest voltage change you can measure

Analog Input

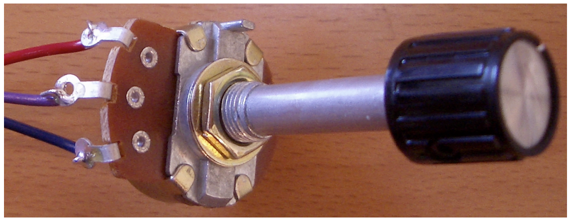
Sure sure, but how to make a varying voltage?

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





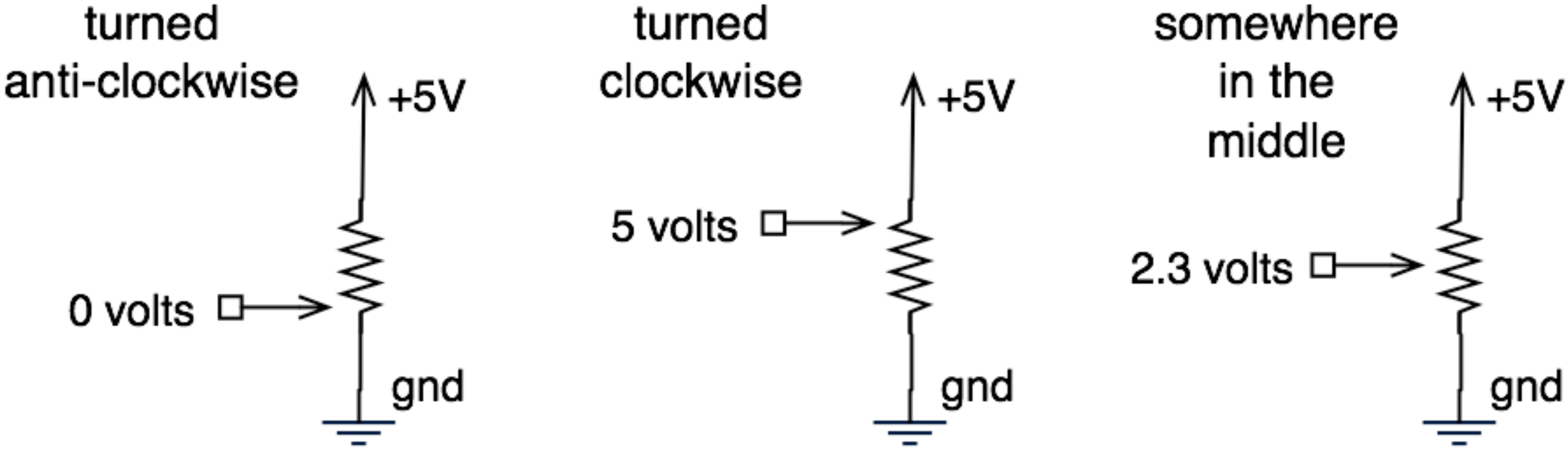
The pot you have



pots also look like this

Potentiometers

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



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*

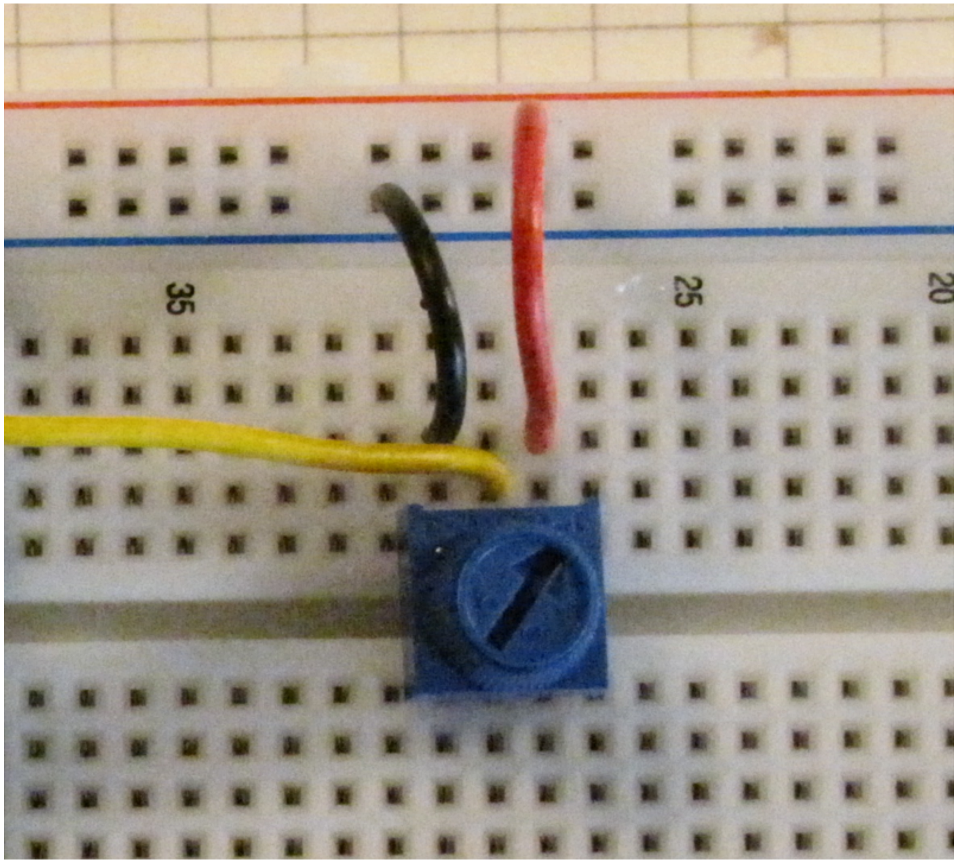
Arduino Analog Input

Plug pot directly into breadboard

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

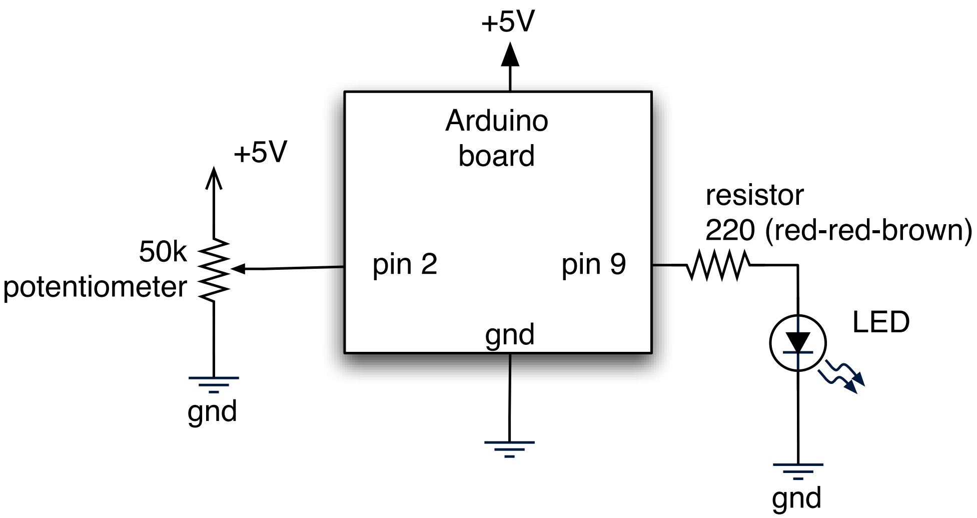
Middle "post" plugs into a row

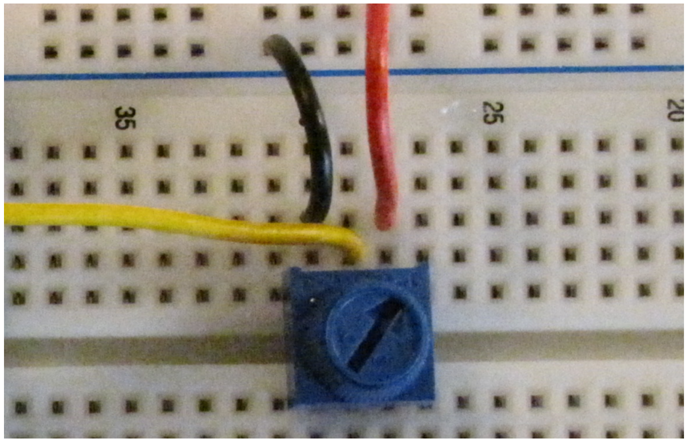
Run a wire from that row to Analog In 2

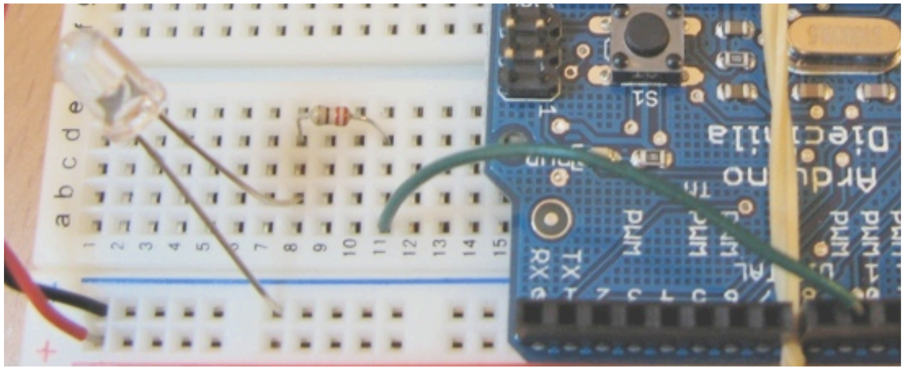


Pot & LED Circuit

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”

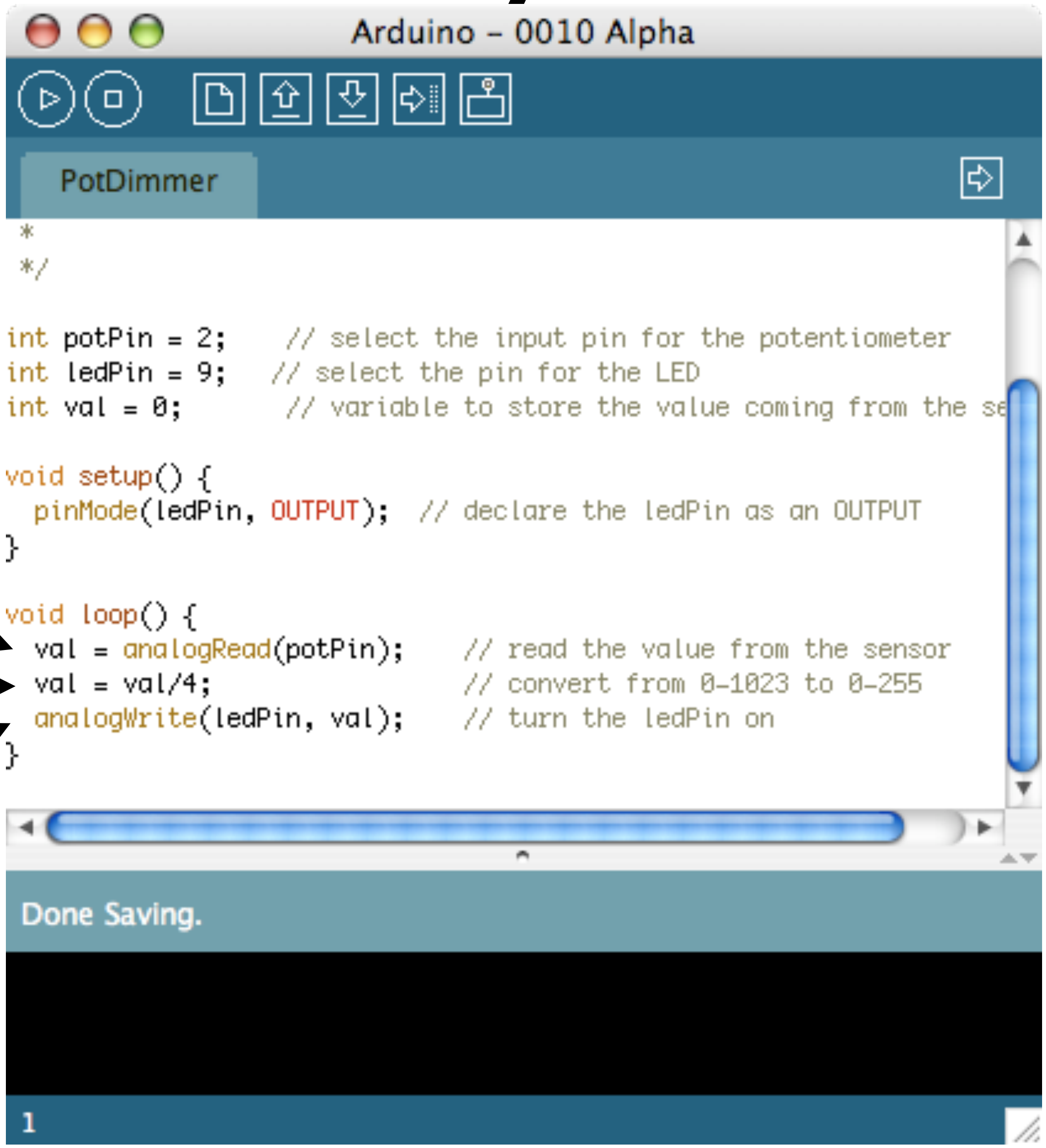
Turn the knob to change LED brightness

input

process the input data

output

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

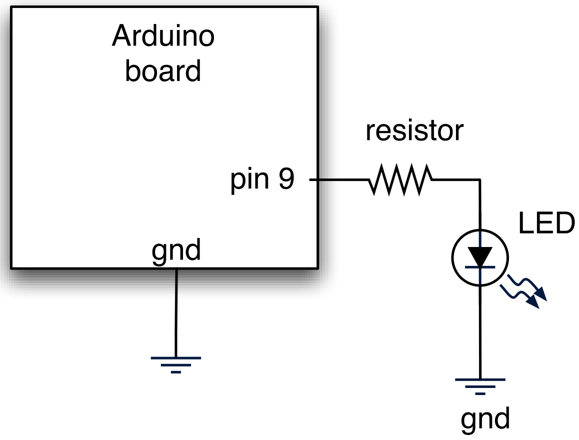


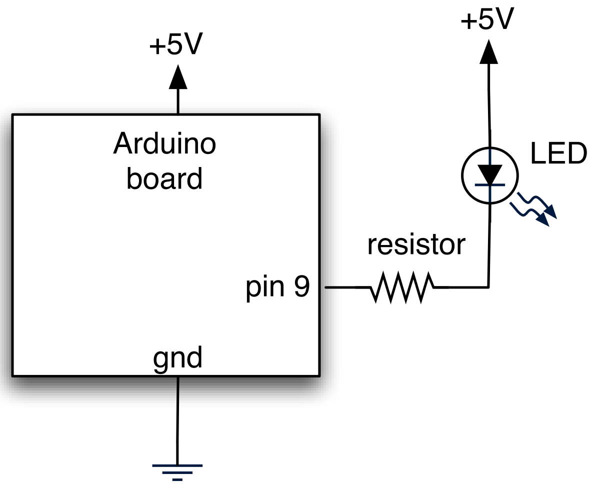
```
 *
 */

int potPin = 2;    // select the input pin for the potentiometer
int ledPin = 9;   // select the pin for the LED
int val = 0;       // variable to store the value coming from the se

void setup() {
  pinMode(ledPin, OUTPUT);  // declare the ledPin as an OUTPUT
}

void loop() {
  val = analogRead(potPin);    // read the value from the sensor
  val = val/4;                 // convert from 0-1023 to 0-255
  analogWrite(ledPin, val);    // turn the ledPin on
}
```

Done Saving.

1

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)`

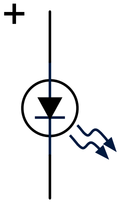
To turn OFF: `digitalWrite(9,HIGH)`

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

RGB LEDs

Normal LED

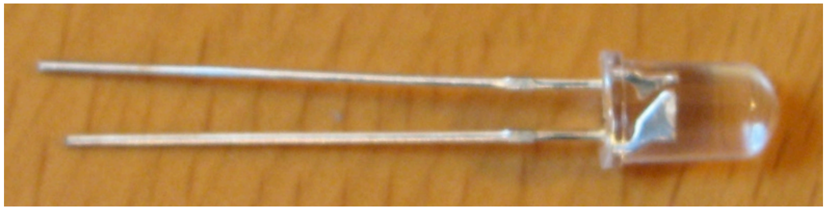
anode +



cathode –

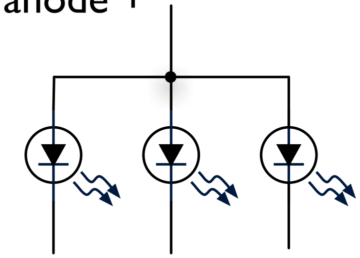
anode +

cathode –



RGB LED

anode +



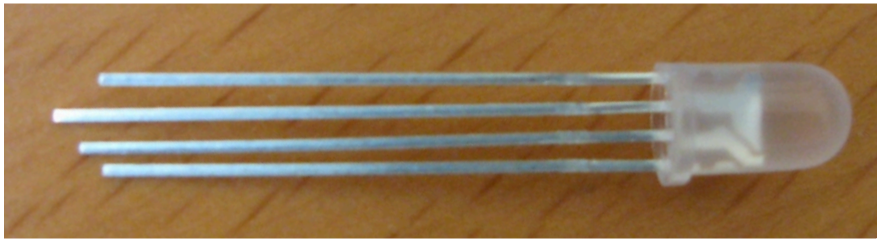
red blue green

red cathode –

anode +

green cathode –

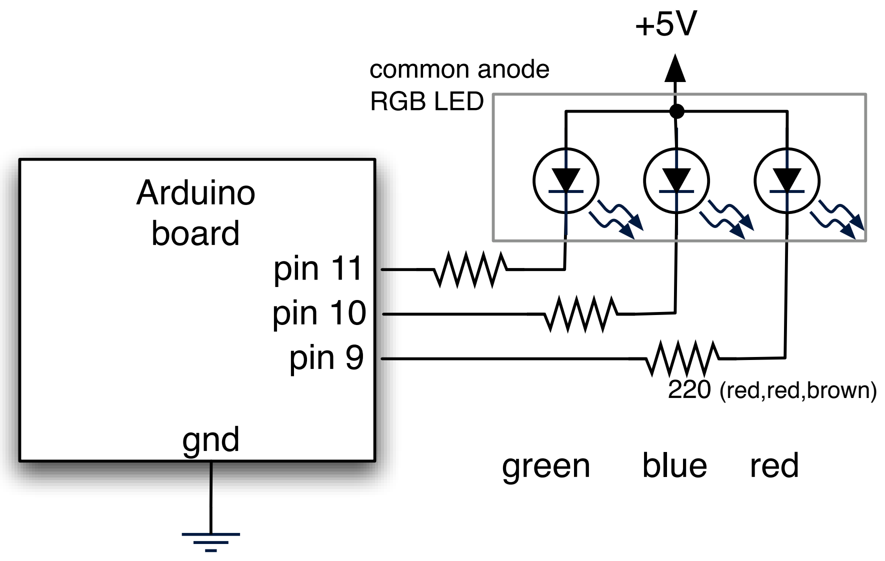
blue cathode –

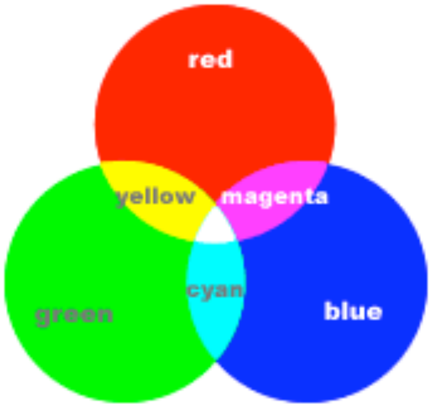


actually 3 LEDs in one package

Color Mixing

With just 3 LEDs you can make any* color



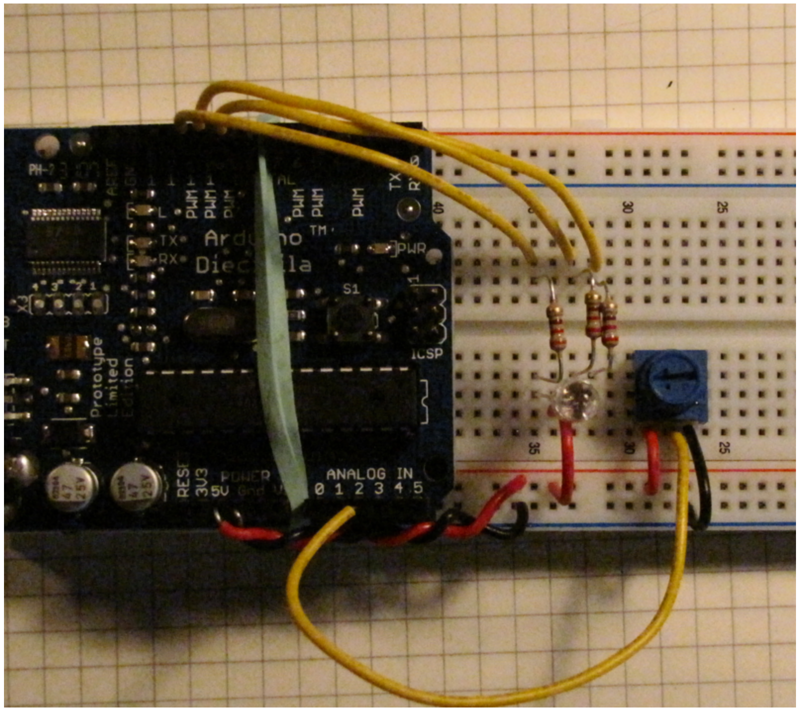


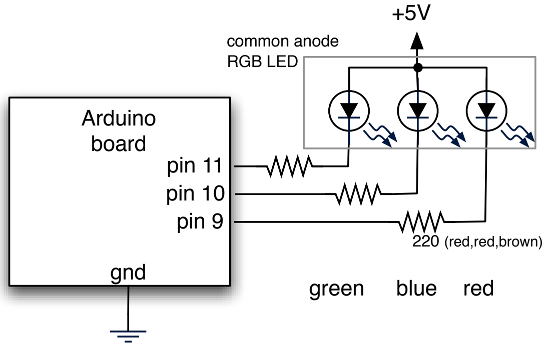
With RGB you can make any color
(except black)

Mixing light is the additive color model

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

Laying out RGB LED Circuit





+5V

common anode
RGB LED

Arduino
board

pin 11

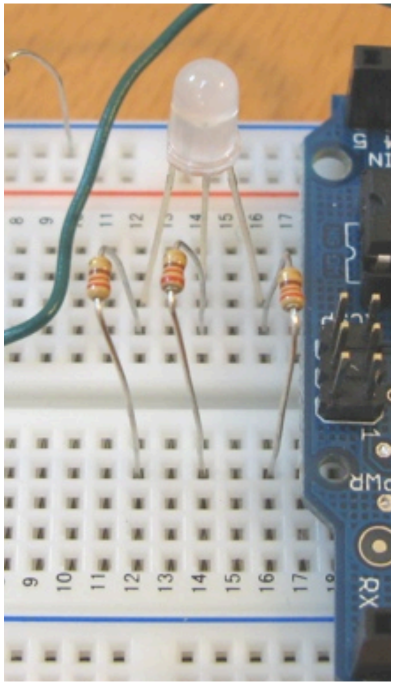
pin 10

pin 9

gnd

220 (red,red,brown)

green blue red



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

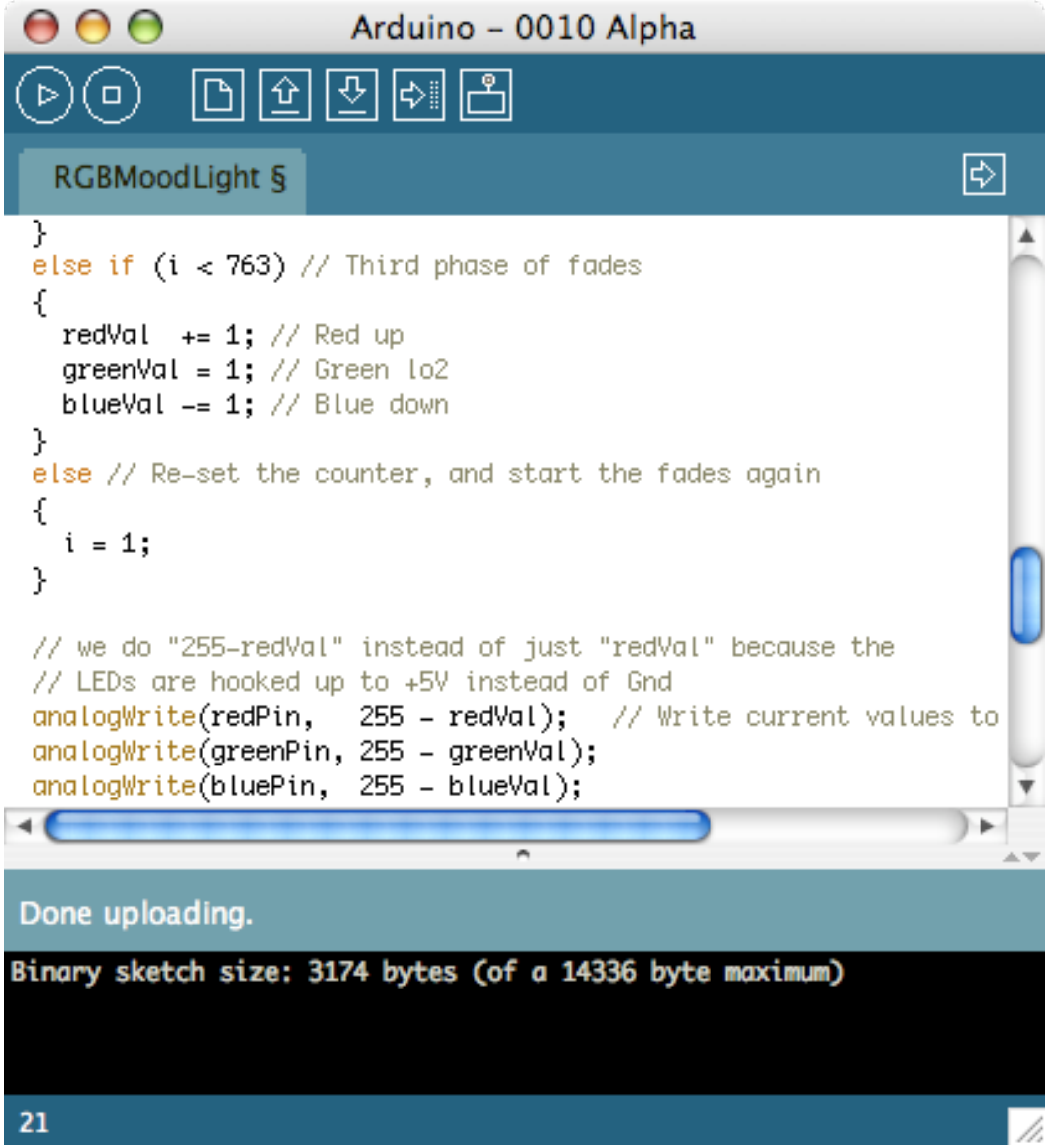
run wires from resistors to pins 9,10,11 of Arduino, can color-code if you want

RGB Color Fading

“RGBMoodLight”

Slow color fading and mixing

Also outputs the current color values to the serial port

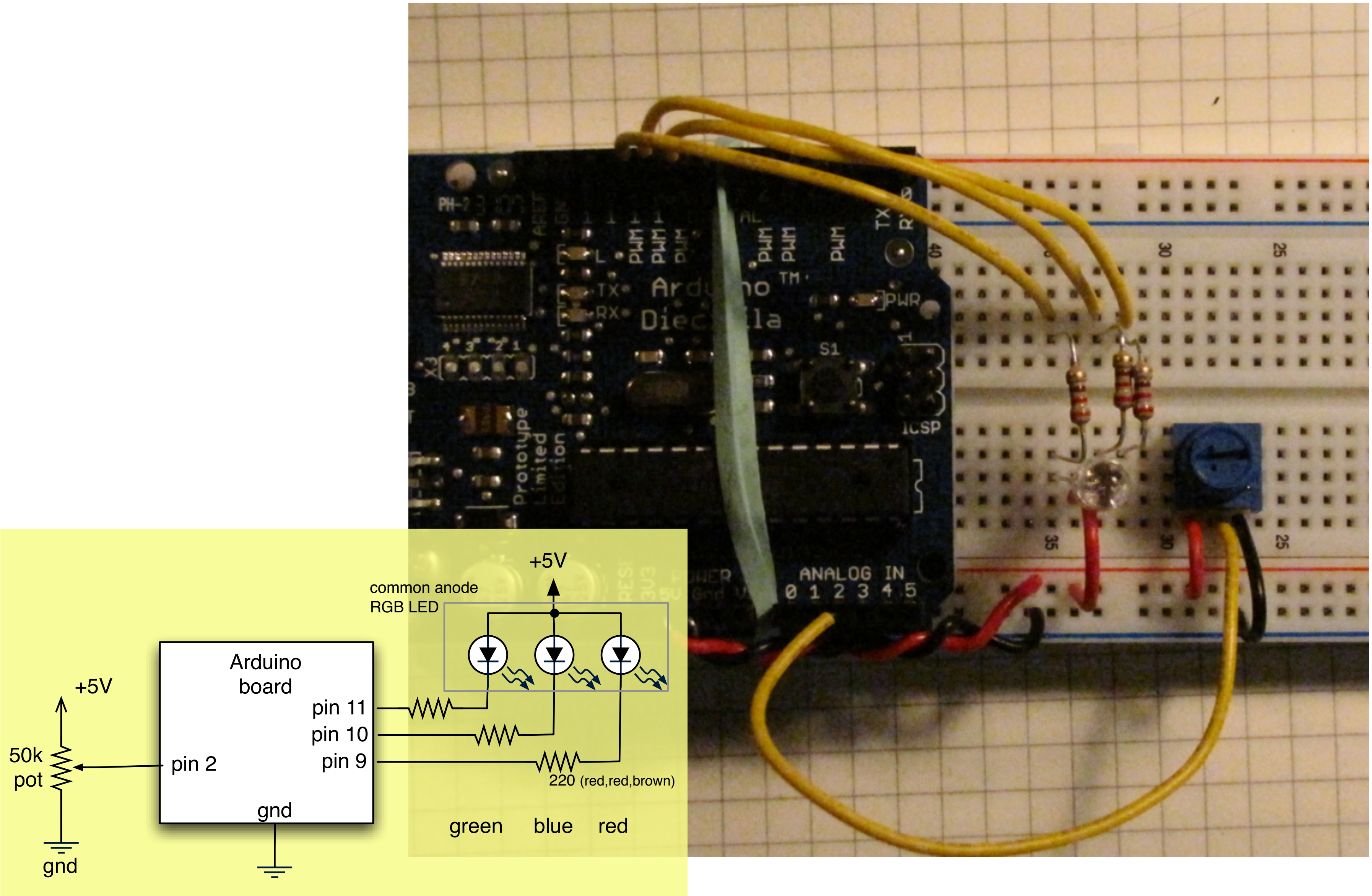


```
}
else if (i < 763) // Third phase of fades
{
  redVal  += 1; // Red up
  greenVal = 1; // Green lo2
  blueVal -= 1; // Blue down
}
else // Re-set the counter, and start the fades again
{
  i = 1;
}

// we do "255-redVal" instead of just "redVal" because the
// LEDs are hooked up to +5V instead of Gnd
analogWrite(redPin,   255 - redVal);   // Write current values to
analogWrite(greenPin, 255 - greenVal);
analogWrite(bluePin,  255 - blueVal);
```

Done uploading.

```
Binary sketch size: 3174 bytes (of a 14336 byte maximum)
```

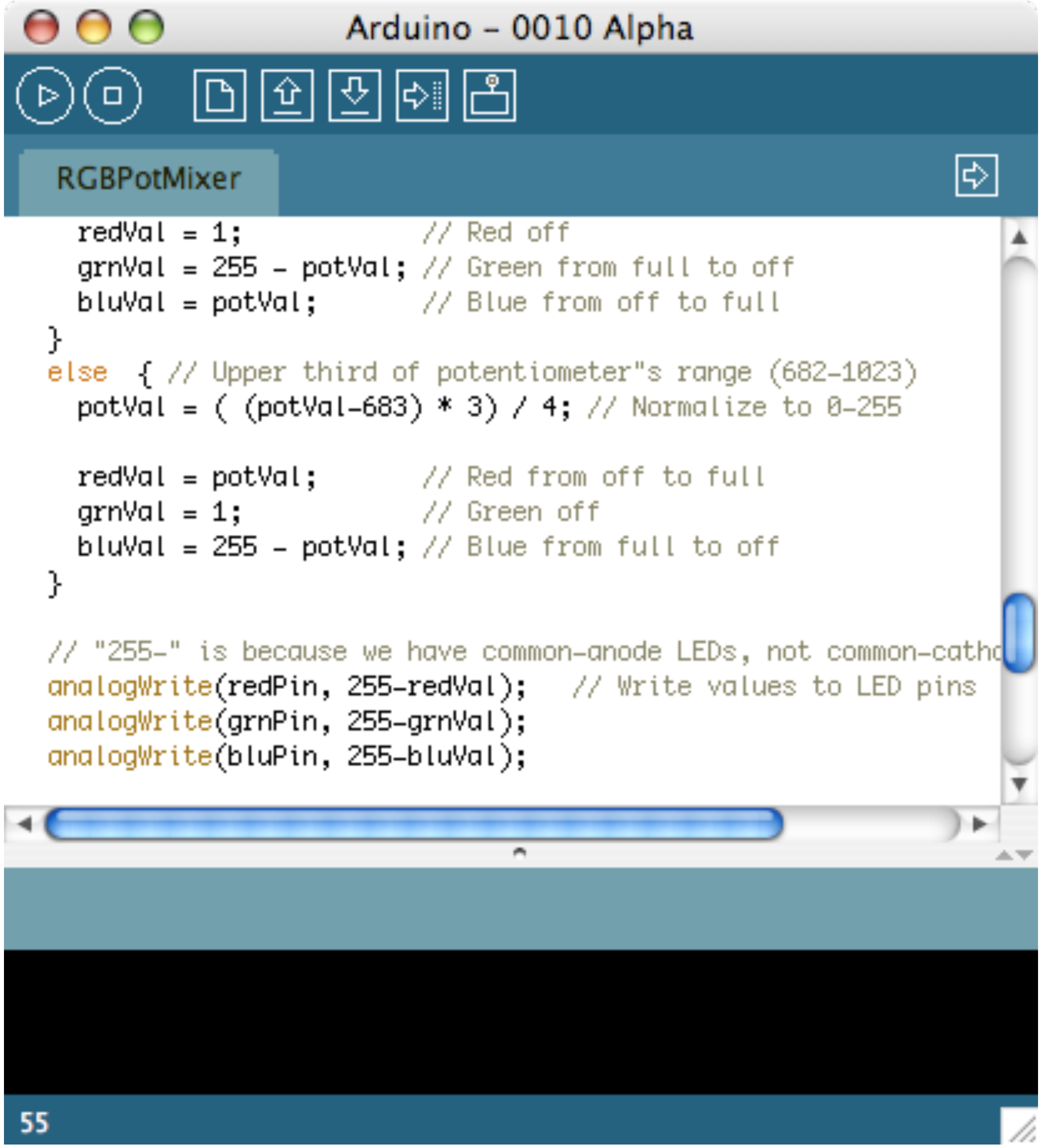
Pot-controlled RGB



Pot-controlled RGB

“RGBPotMixer”

Use the pot from before to control the color mix

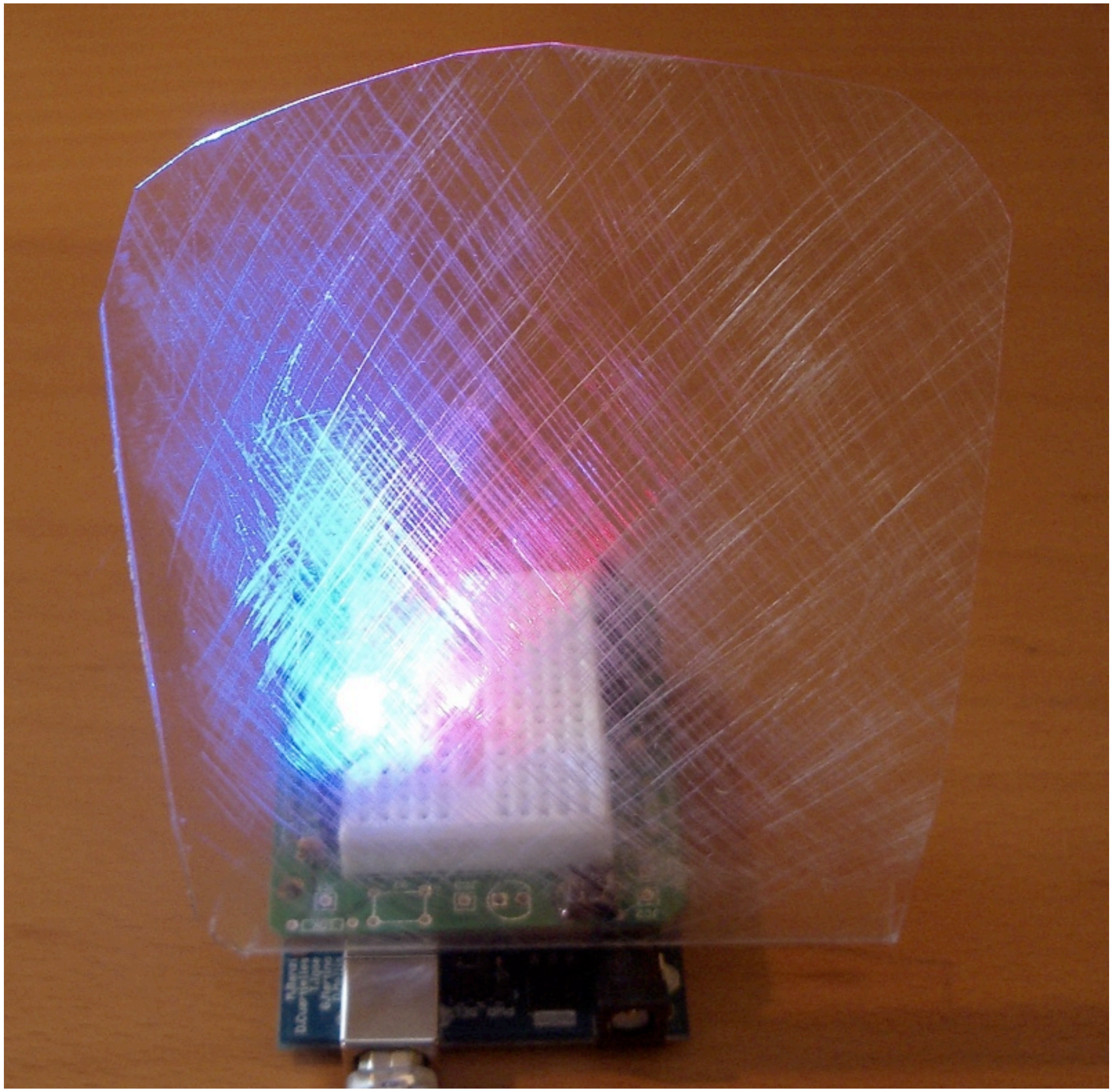


```
  redVal = 1;            // Red off
  grnVal = 255 - potVal; // Green from full to off
  bluVal = potVal;       // Blue from off to full
}
else  { // Upper third of potentiometer"s range (682-1023)
  potVal = ( (potVal-683) * 3) / 4; // Normalize to 0-255

  redVal = potVal;       // Red from off to full
  grnVal = 1;            // Green off
  bluVal = 255 - potVal; // Blue from full to off
}

// "255-" is because we have common-anode LEDs, not common-catho
analogWrite(redPin, 255-redVal);   // Write values to LED pins
analogWrite(grnPin, 255-grnVal);
analogWrite(bluPin, 255-bluVal);
```

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

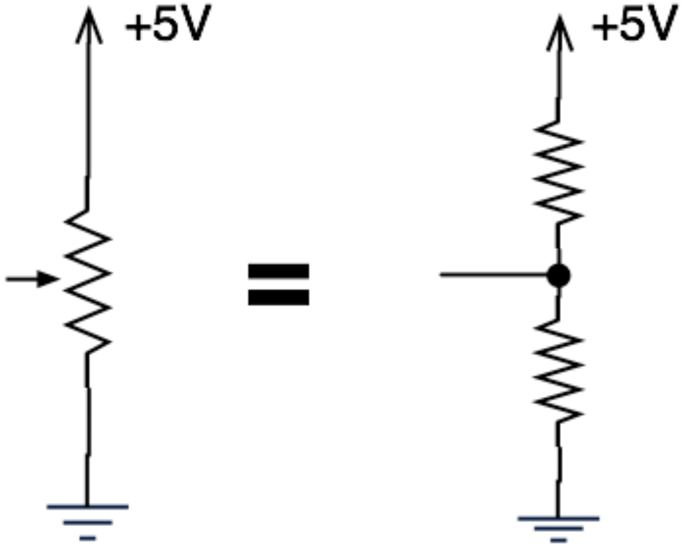
Mood Light

Diffuser made from piece of plastic scratched with sandpaper



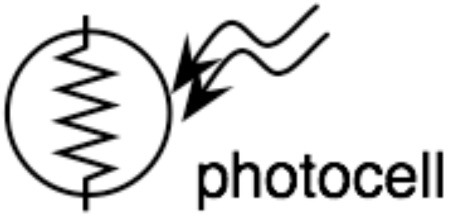
Sensing the Dark

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

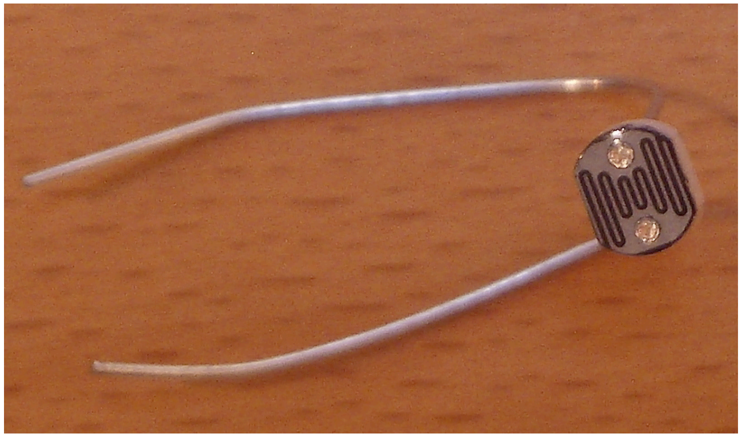


Sensing the Dark: Photocells

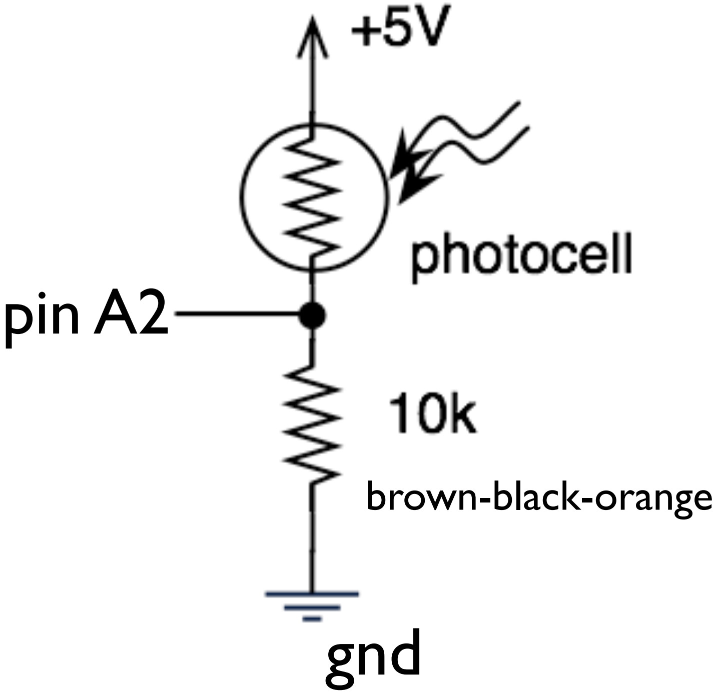
- aka. photoresistor, light-dependent resistor
- A *variable* resistor
- Brighter light == lower resistance
- Photocells you have range approx. 0-10k-1M



schematic symbol



Photocell Circuit



+5V

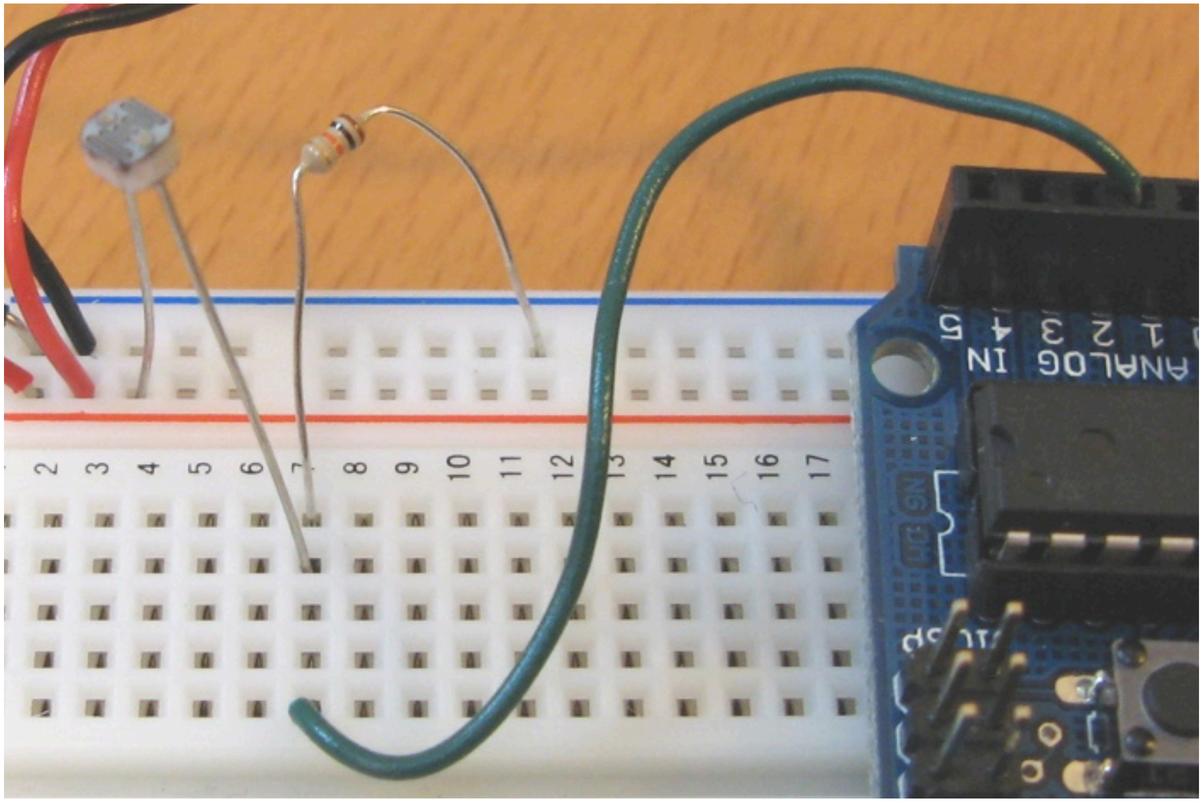
photocell

pin A2

10k

brown-black-orange

gnd

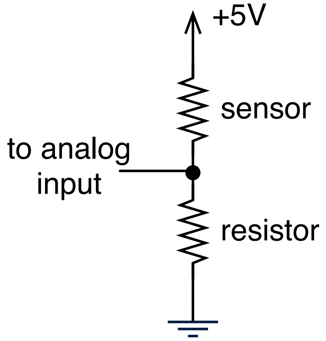


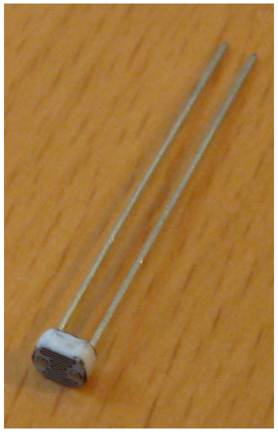
Try it with RGBPotMixer from before

Resistive sensors

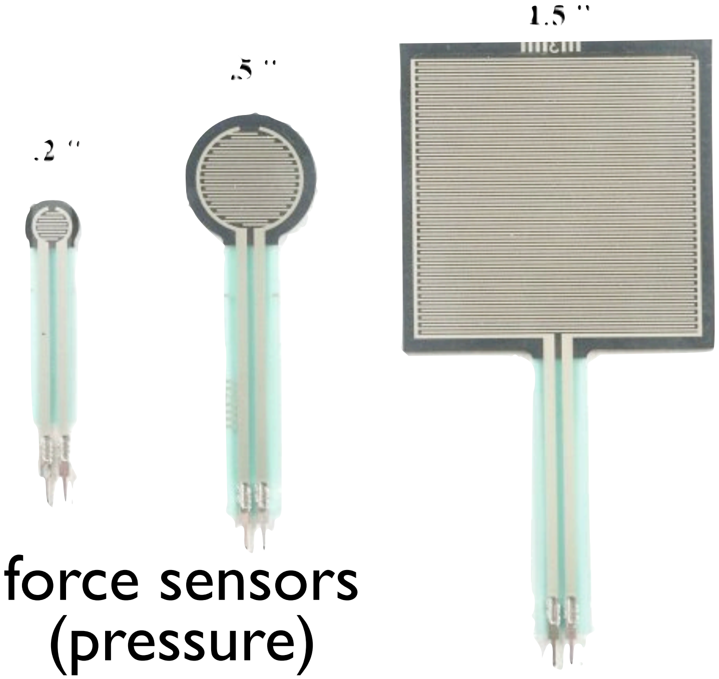
thermistor
(temperature)

circuit is the same
for all these

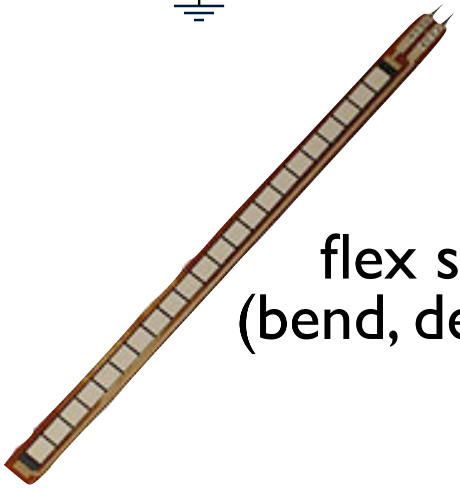




photocell
(light)



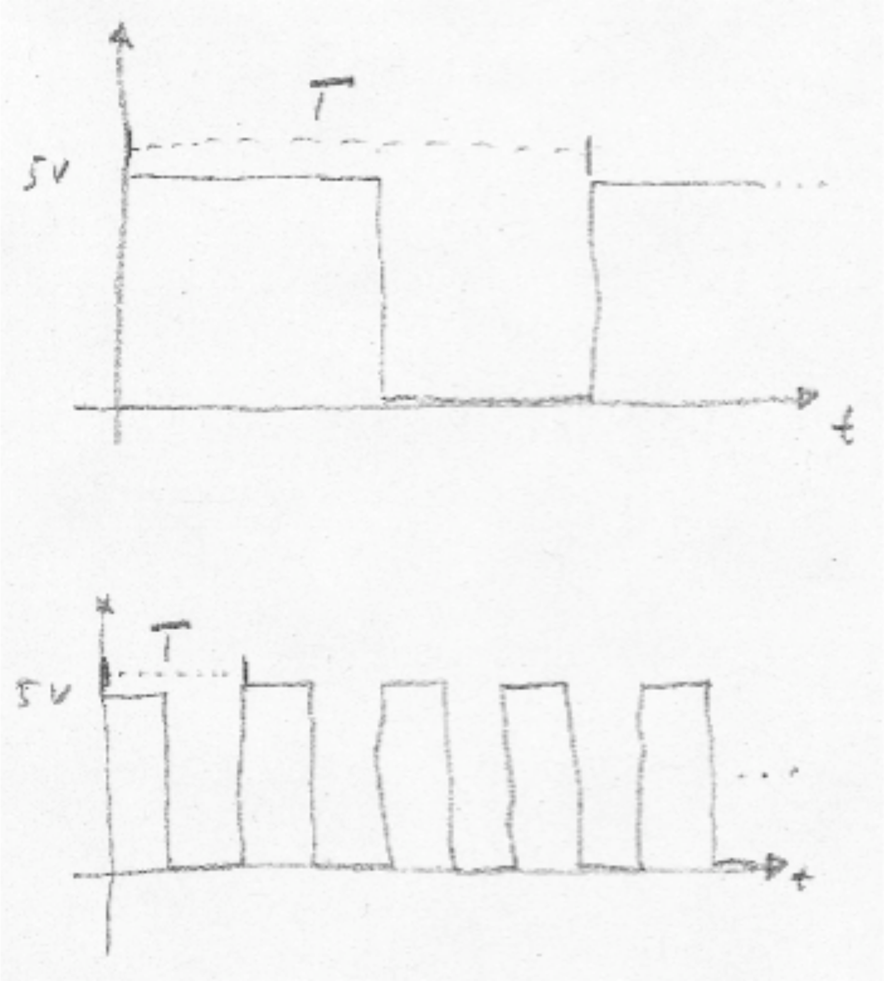
force sensors
(pressure)



flex sensor
(bend, deflection)

also air pressure
and others

Simulating Soundwave



Low pitched tone – long period T

High pitched tone – short period T

Note: For a 20Hz sound wave we have T = 50ms, for a 200Hz sound wave we have T = 5ms.

Make a Theremin

"ooo-weee-ooooo"

The original spooky sound machine

Works by measuring your body's electric field

No touching needed!

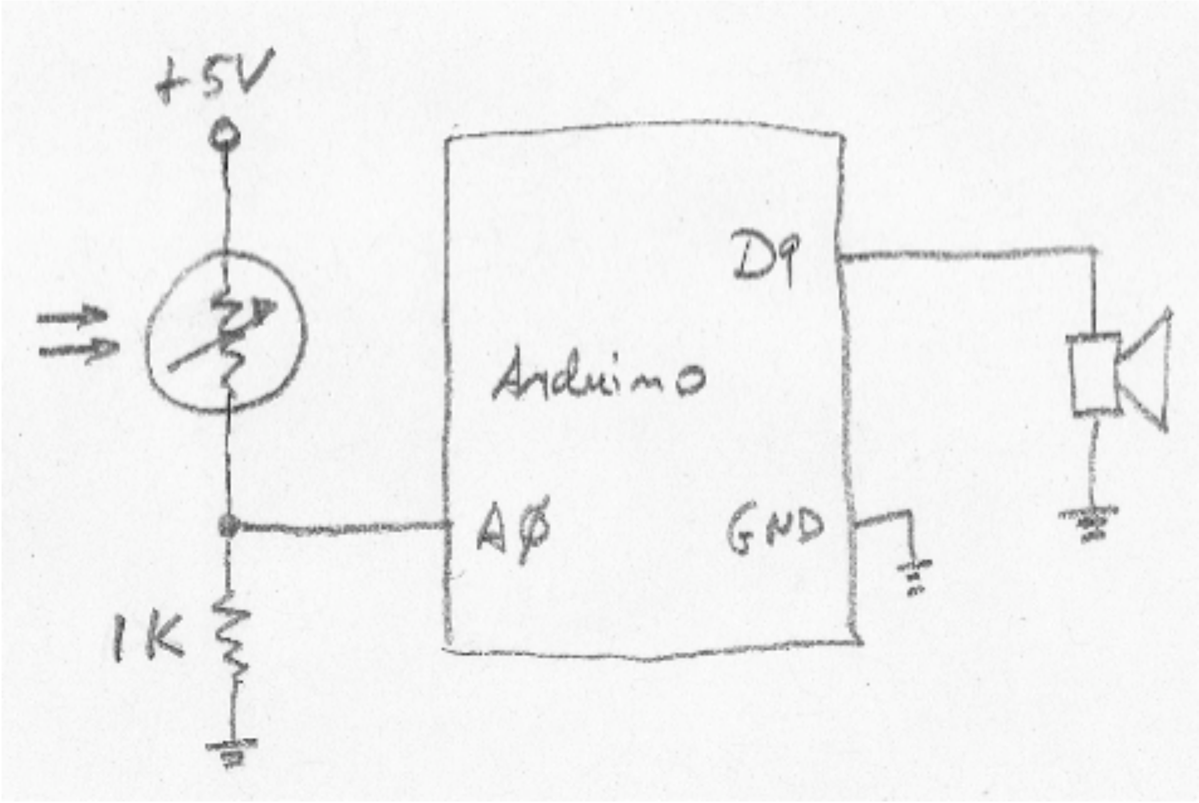
We'll use light in lieu of RF

Leon Theremin

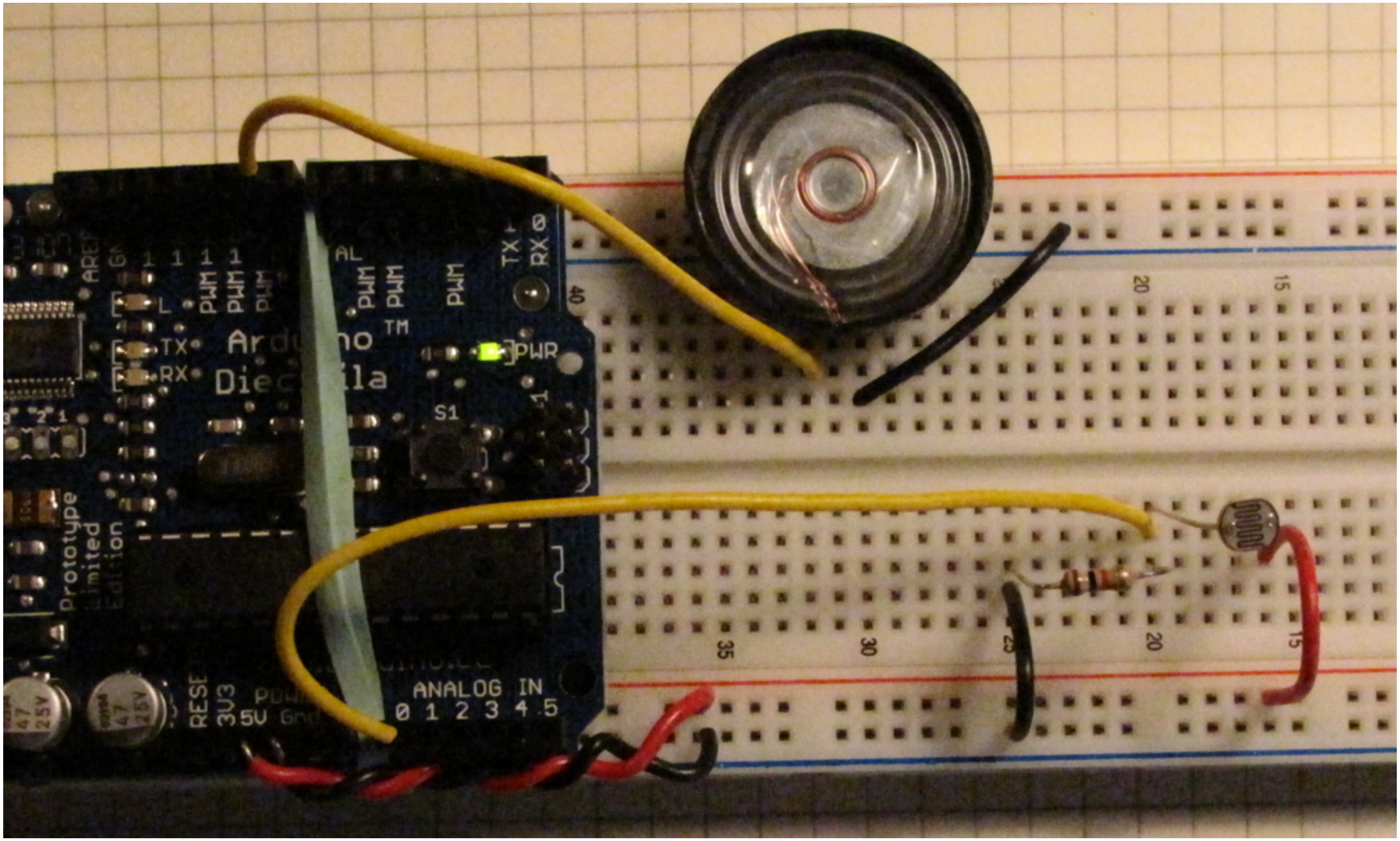
Optical Theremin

- Read an analog signal generated through a photoresistor
- We interpret the digitized value from the A/D conversion as the period of the sound wave we want to generate
- Generate one period of the sound wave, output it to the speaker and then sample the input again

Theremin Setup



Theremin Setup

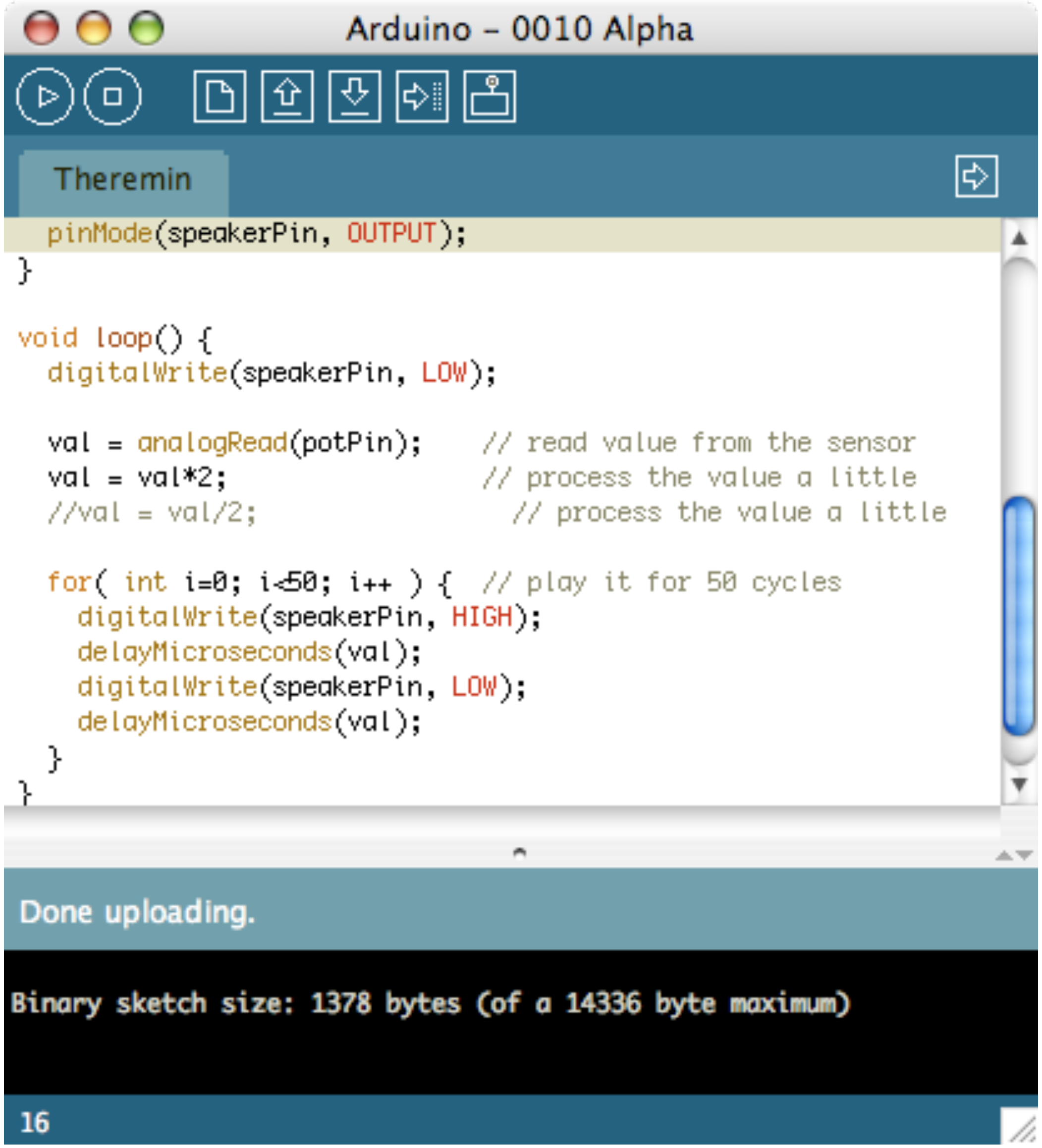


Light Theremin

“Theremin”

Move hand over photocell to change pitch

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



```
  pinMode(speakerPin, OUTPUT);
}

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.

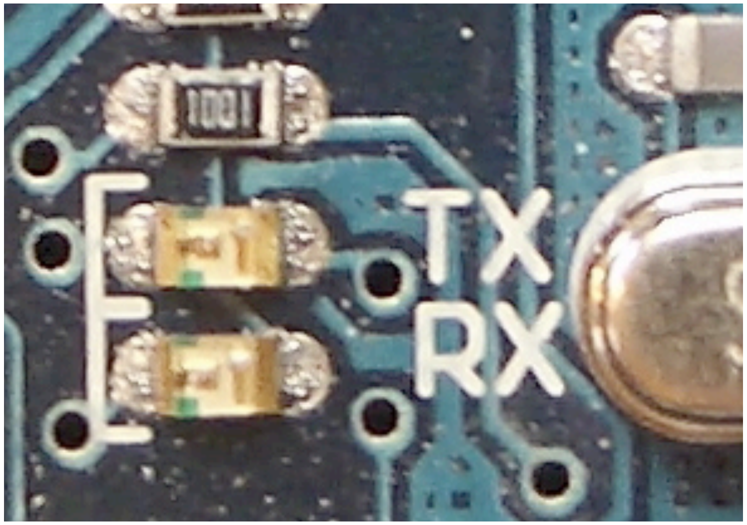
```
Binary sketch size: 1378 bytes (of a 14336 byte maximum)
```

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

Watch the TX/RX LEDS

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



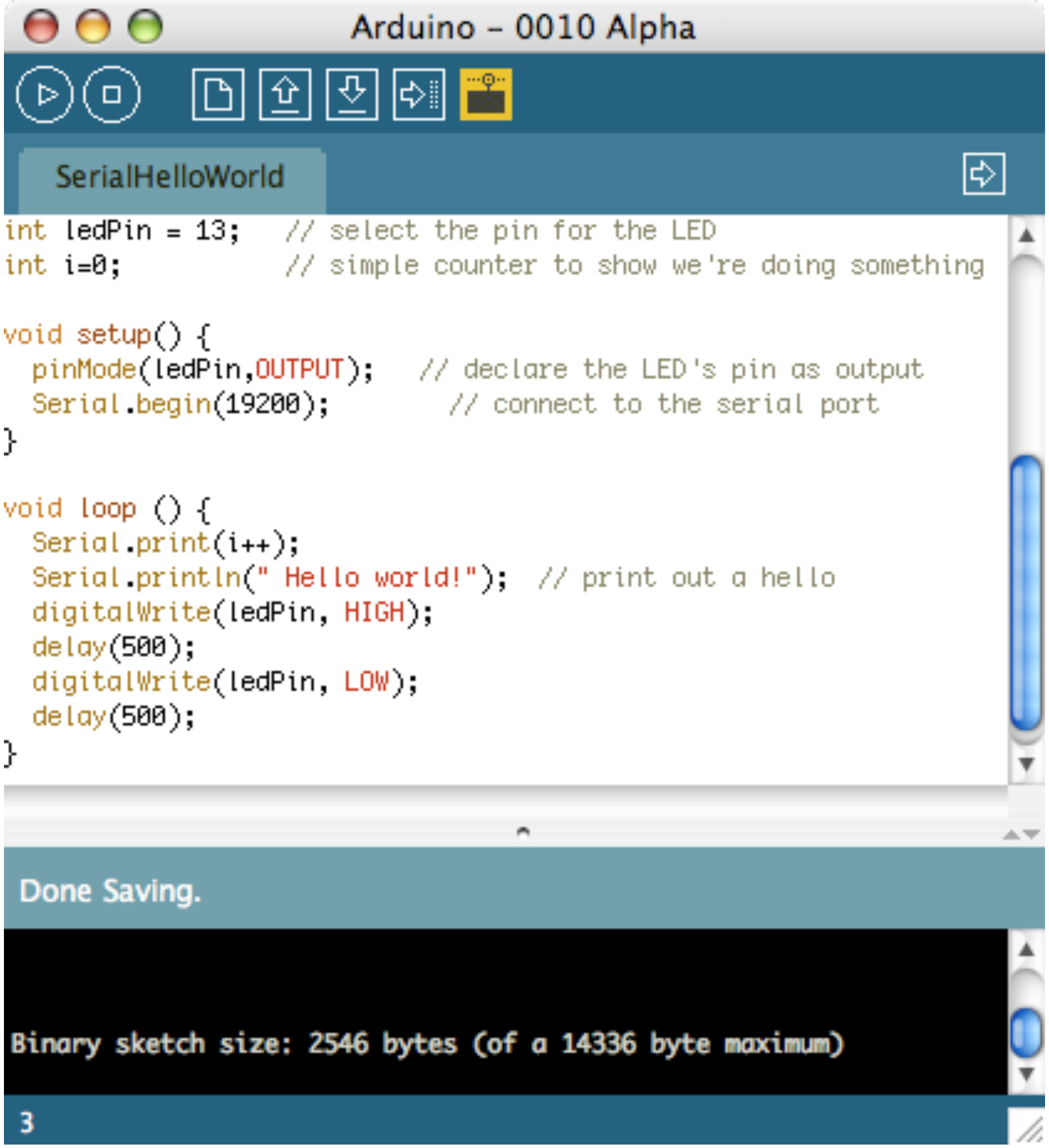
Arduino Says "Hi"

"SerialHelloWorld"

Sends "Hello world!" to your computer

Click on "Serial Monitor" button to see output

Watch TX LED compared to pin 13 LED



```
int ledPin = 13;    // select the pin for the LED
int i=0;            // simple counter to show we're doing something

void setup() {
  pinMode(ledPin,OUTPUT);   // declare the LED's pin as output
  Serial.begin(19200);        // connect to the serial port
}

void loop () {
  Serial.print(i++);
  Serial.println(" Hello world!");  // print out a hello
  digitalWrite(ledPin, HIGH);
  delay(500);
  digitalWrite(ledPin, LOW);
  delay(500);
}
```

Done Saving.

```
Binary sketch size: 2546 bytes (of a 14336 byte maximum)
```

3

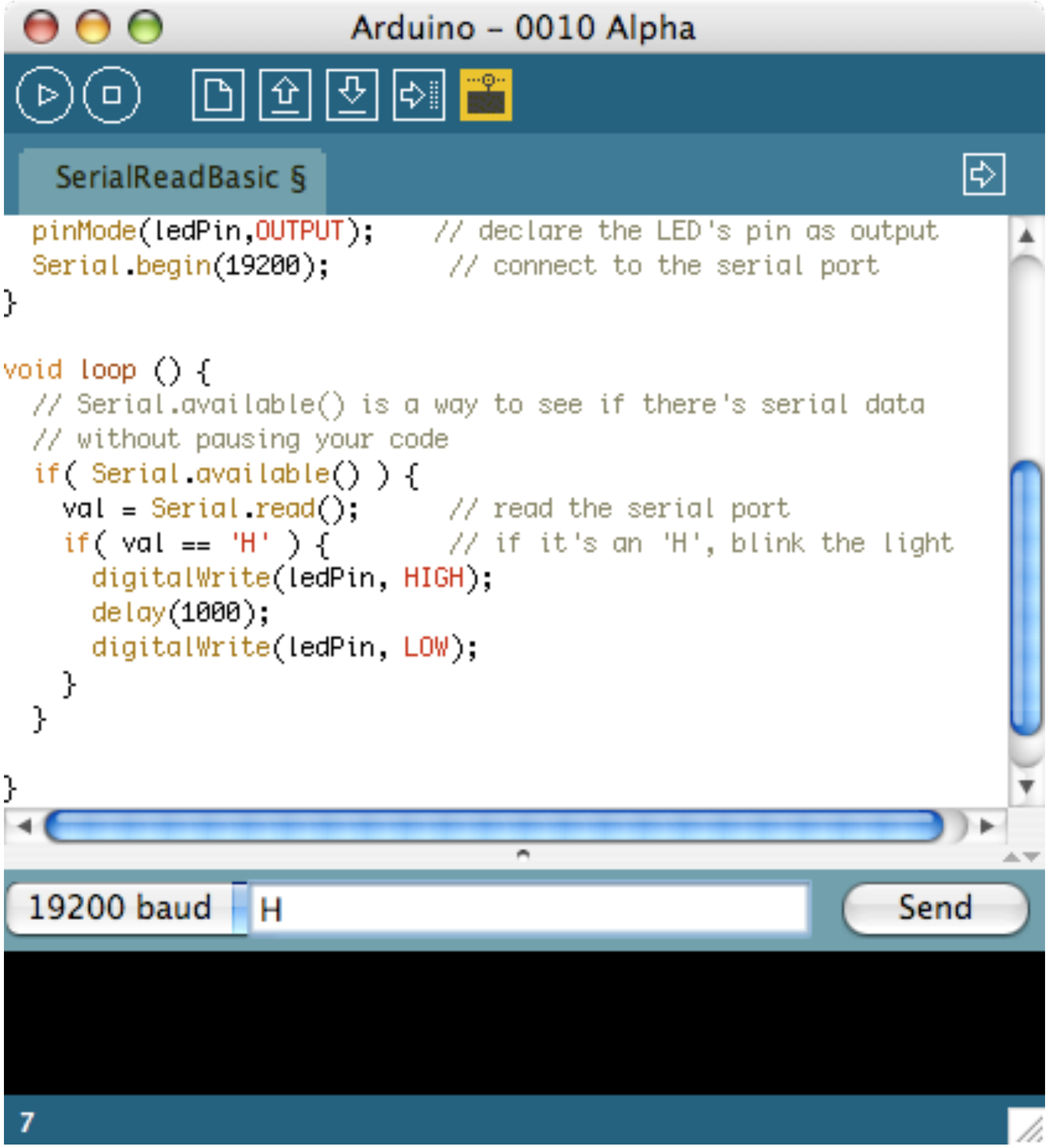
Telling Arduino What To Do

“SerialReadBasic”

You type “H”, LED blinks

In “Serial Monitor”, type “H”, press Send

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



```
  pinMode(ledPin,OUTPUT);     // declare the LED's pin as output
  Serial.begin(19200);         // connect to the serial port
}

void loop () {
  // Serial.available() is a way to see if there's serial data
  // without pausing your code
  if( Serial.available() ) {
    val = Serial.read();        // read the serial port
    if( val == 'H' ) {          // if it's an 'H', blink the light
      digitalWrite(ledPin, HIGH);
      delay(1000);
      digitalWrite(ledPin, LOW);
    }
  }

}
```

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

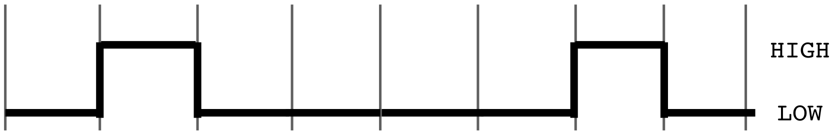
serial terminal from the olde days

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
```

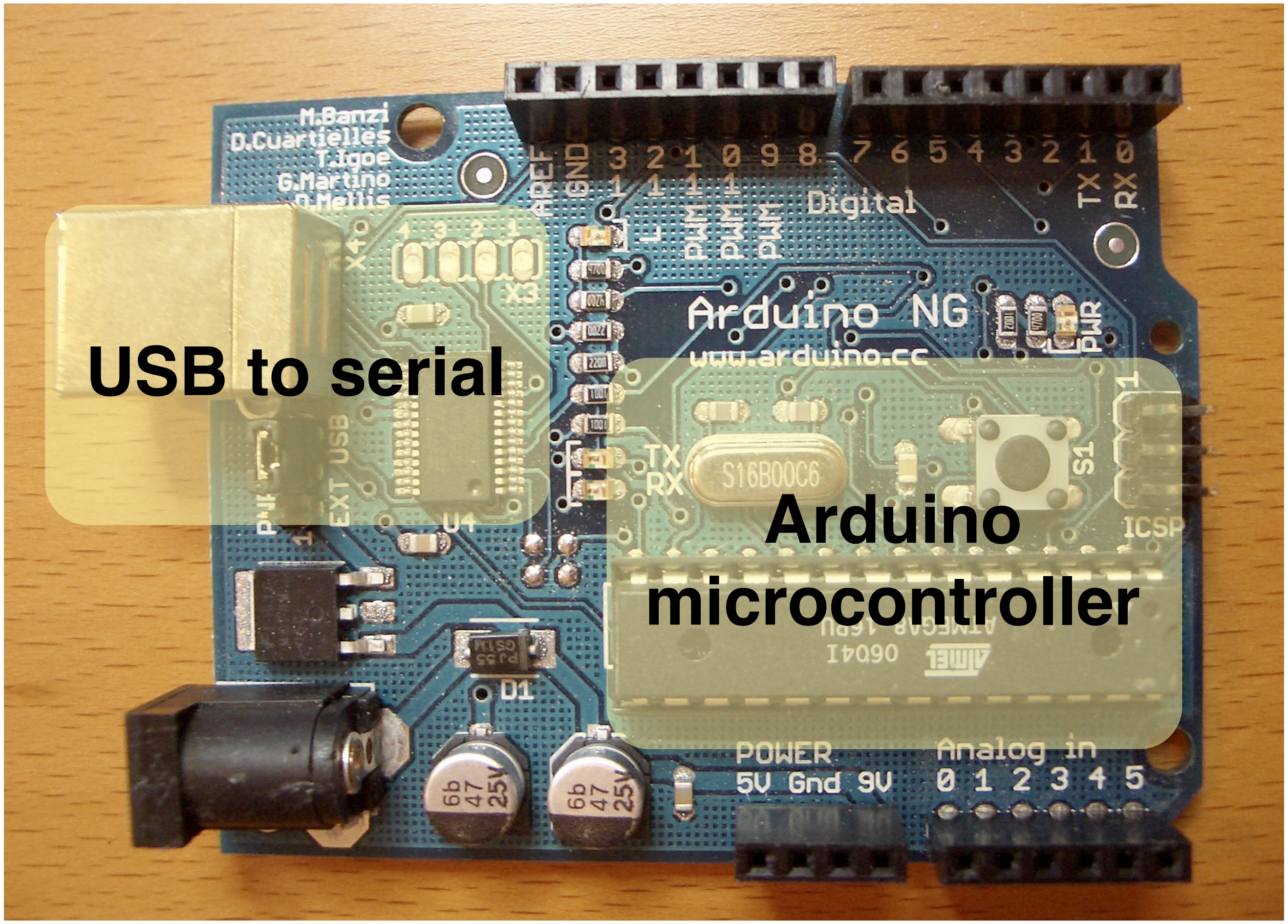
=



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

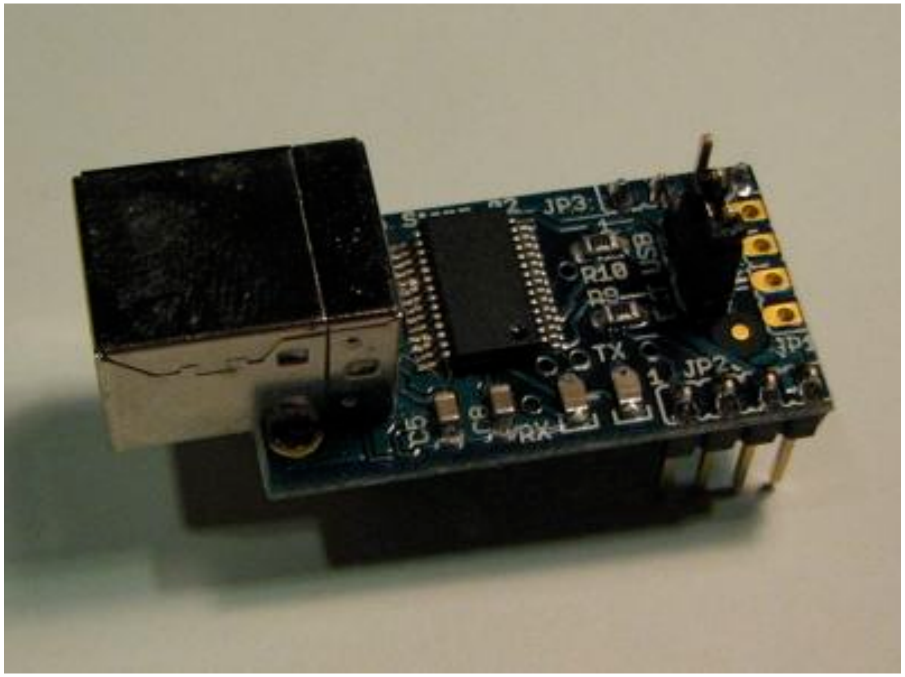
Arduino & USB-to-serial

Arduino board is really two circuits

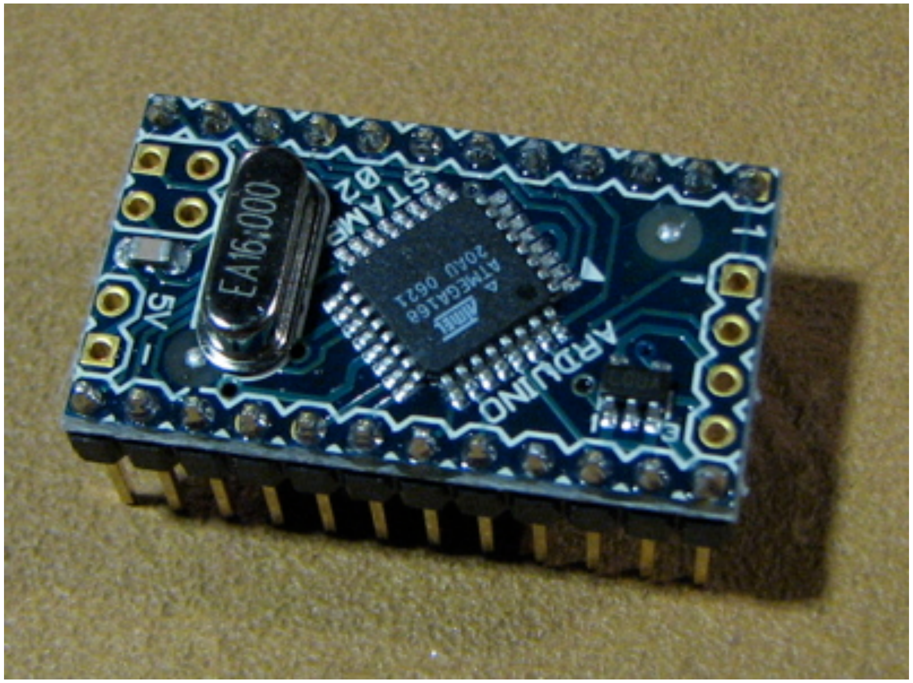


Arduino Mini

Arduino Mini separates the two circuits

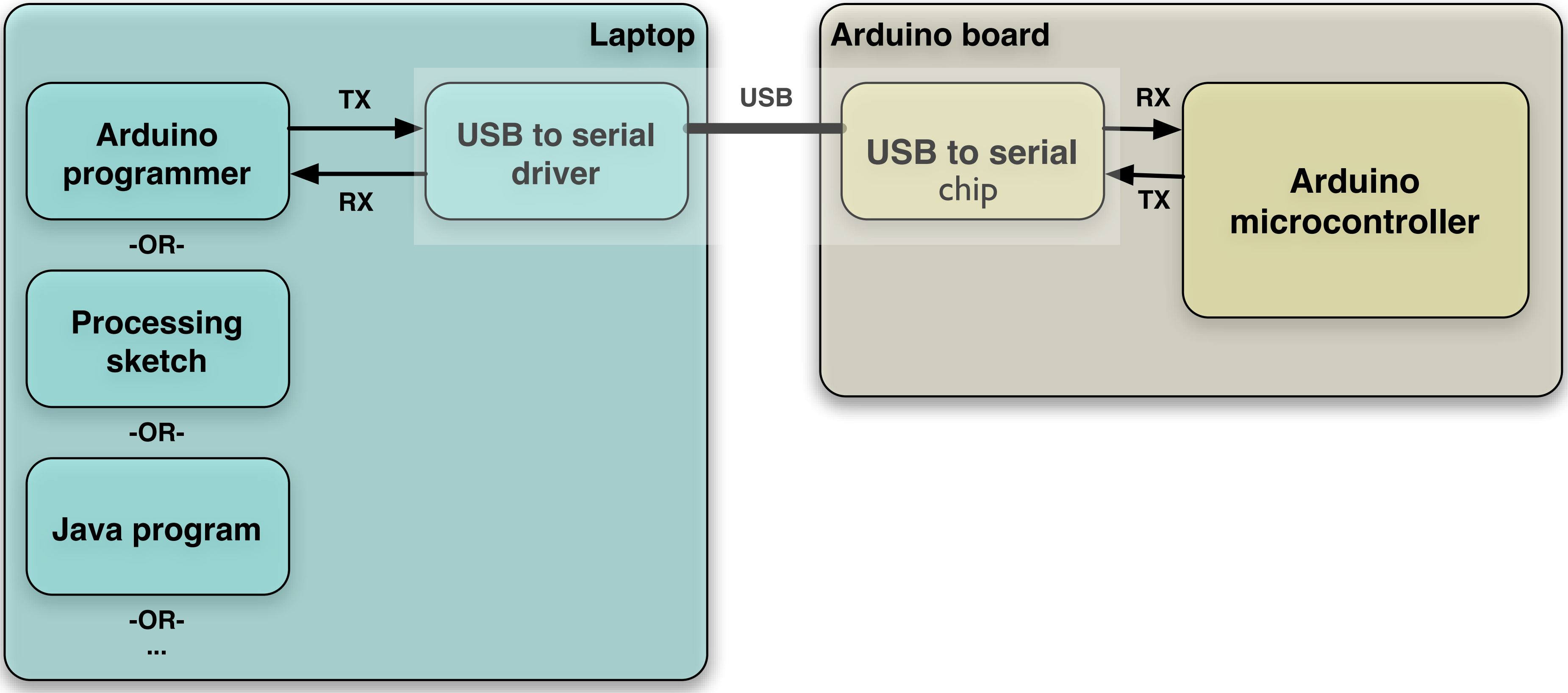


Arduino Mini USB adapter



Arduino Mini

Arduino to Computer



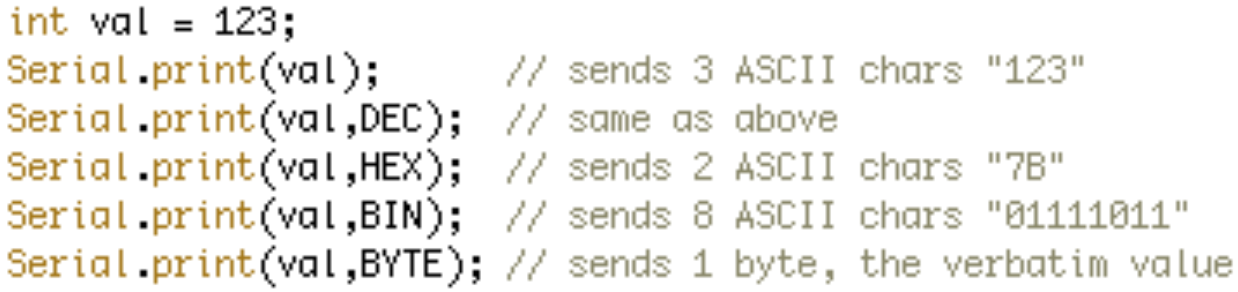
USB is totally optional for Arduino
But it makes things easier

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

Controlling the Computer

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

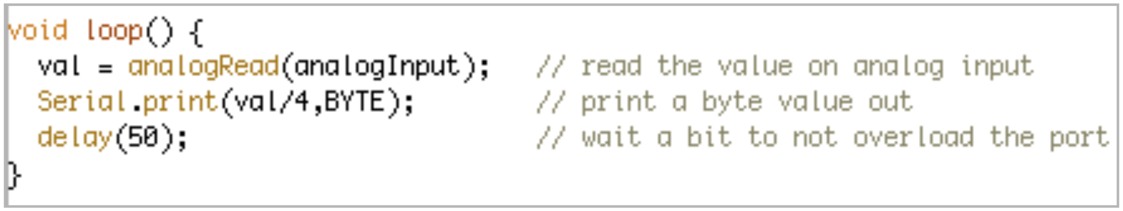


```
int val = 123;
Serial.print(val);       // sends 3 ASCII chars "123"
Serial.print(val,DEC);  // same as above
Serial.print(val,HEX);  // sends 2 ASCII chars "7B"
Serial.print(val,BIN);  // sends 8 ASCII chars "01111011"
Serial.print(val,BYTE); // sends 1 byte, the verbatim value
```

Controlling the Computer

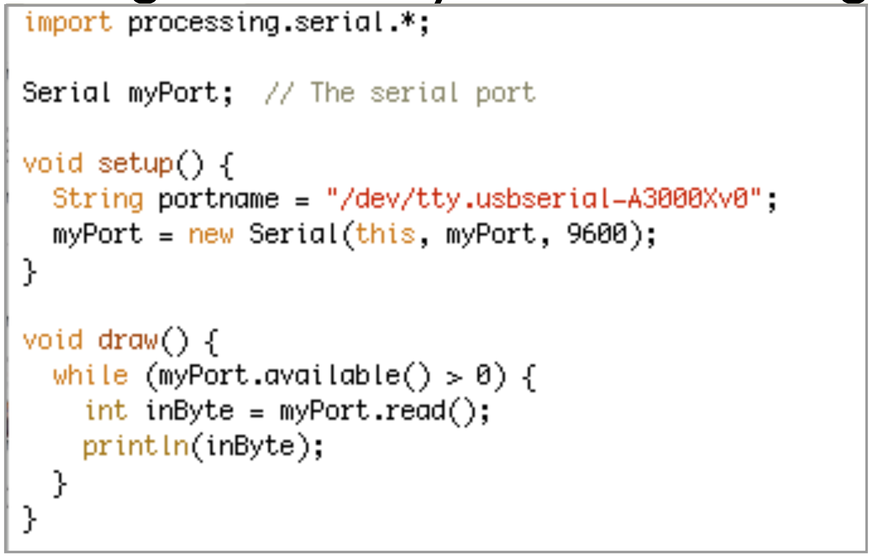
You write one program on Arduino, one on the computer

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);
  }
}
```

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

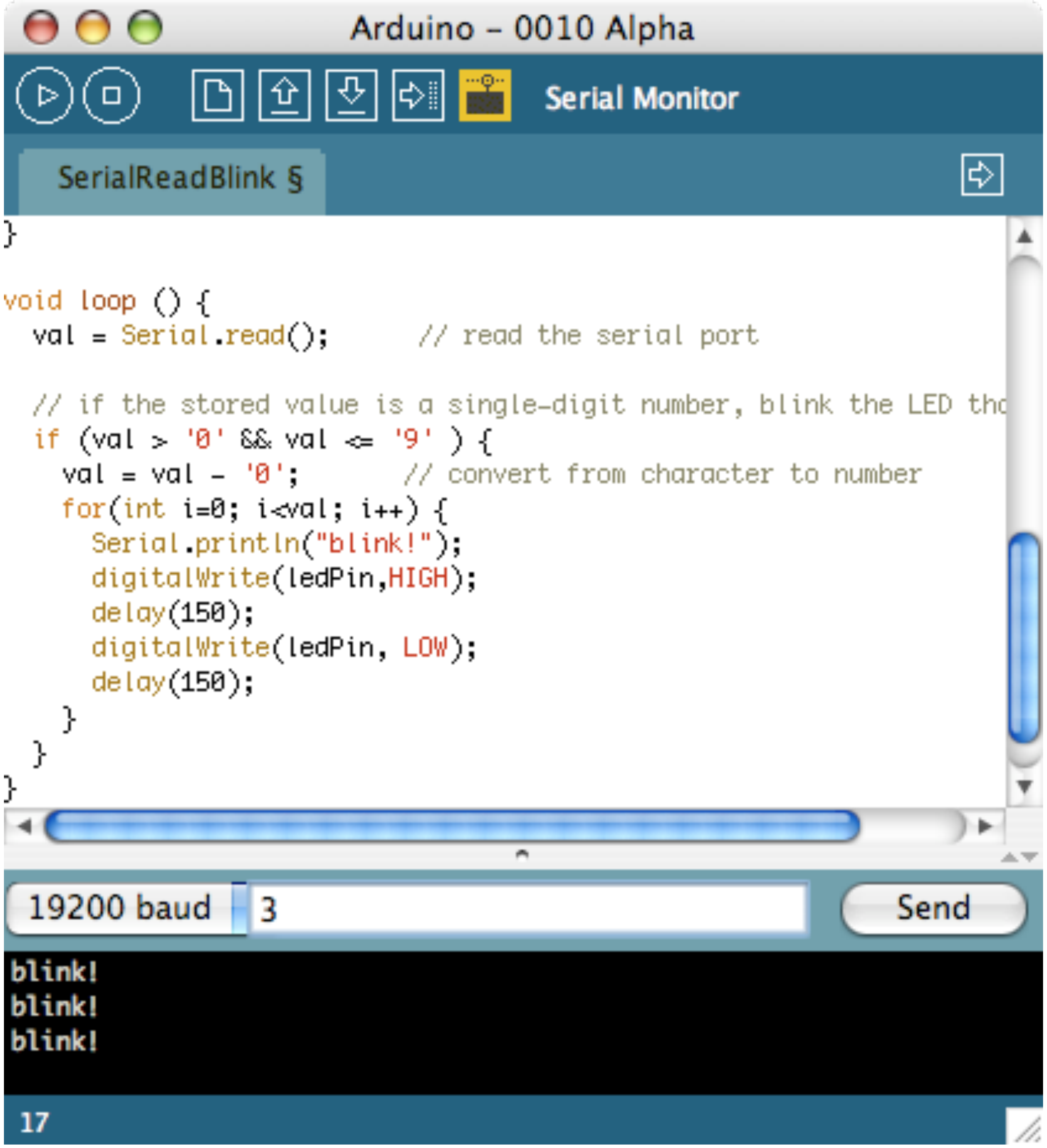
Controlling Arduino, Again

“`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



```
}

void loop () {
  val = Serial.read();      // read the serial port

  // if the stored value is a single-digit number, blink the LED tha
  if (val > '0' && val <= '9' ) {
    val = val - '0';        // convert from character to number
    for(int i=0; i<val; i++) {
      Serial.println("blink!");
      digitalWrite(ledPin,HIGH);
      delay(150);
      digitalWrite(ledPin, LOW);
      delay(150);
    }
  }
}
```

3

```
blink!
blink!
blink!
```

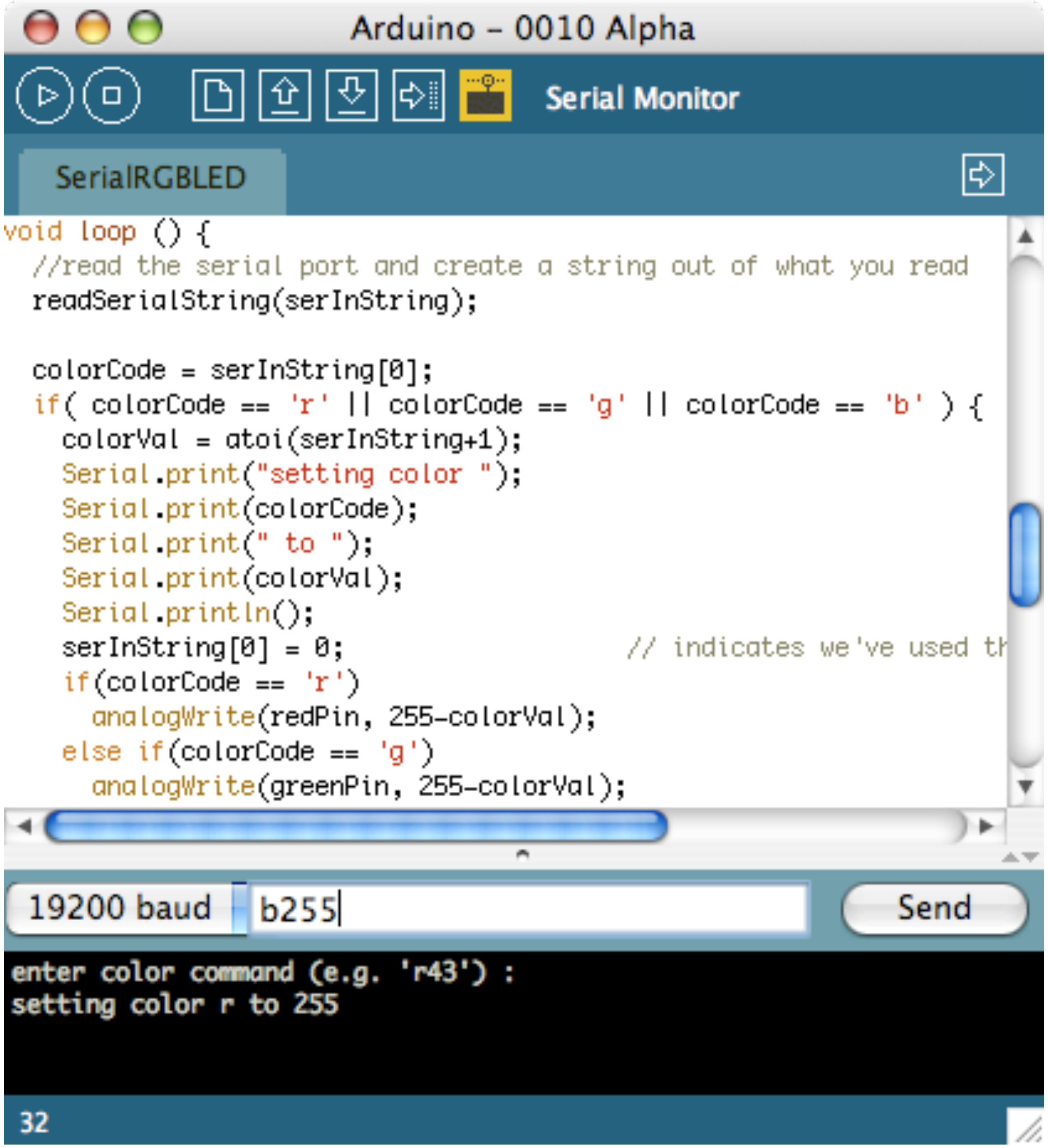
Serial-controlled RGB

"SerialRGBLED"

Send color commands to Arduino

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

Sketch parses what you type, changes LEDs

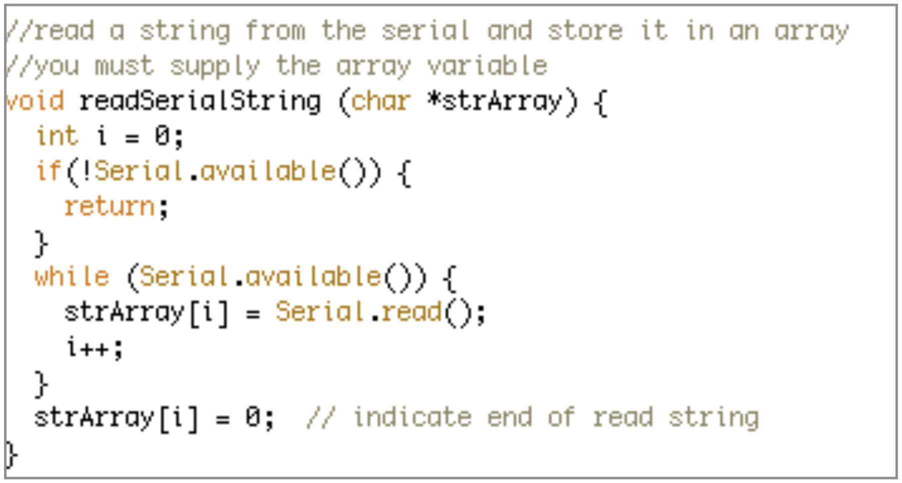


```
void loop () {
  //read the serial port and create a string out of what you read
  readSerialString(serInString);

  colorCode = serInString[0];
  if( colorCode == 'r' || colorCode == 'g' || colorCode == 'b' ) {
    colorVal = atoi(serInString+1);
    Serial.print("setting color ");
    Serial.print(colorCode);
    Serial.print(" to ");
    Serial.print(colorVal);
    Serial.println();
    serInString[0] = 0;                   // indicates we've used th
    if(colorCode == 'r')
      analogWrite(redPin, 255-colorVal);
    else if(colorCode == 'g')
      analogWrite(greenPin, 255-colorVal);
```

```
enter color command (e.g. 'r43') :
setting color r to 255
```

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



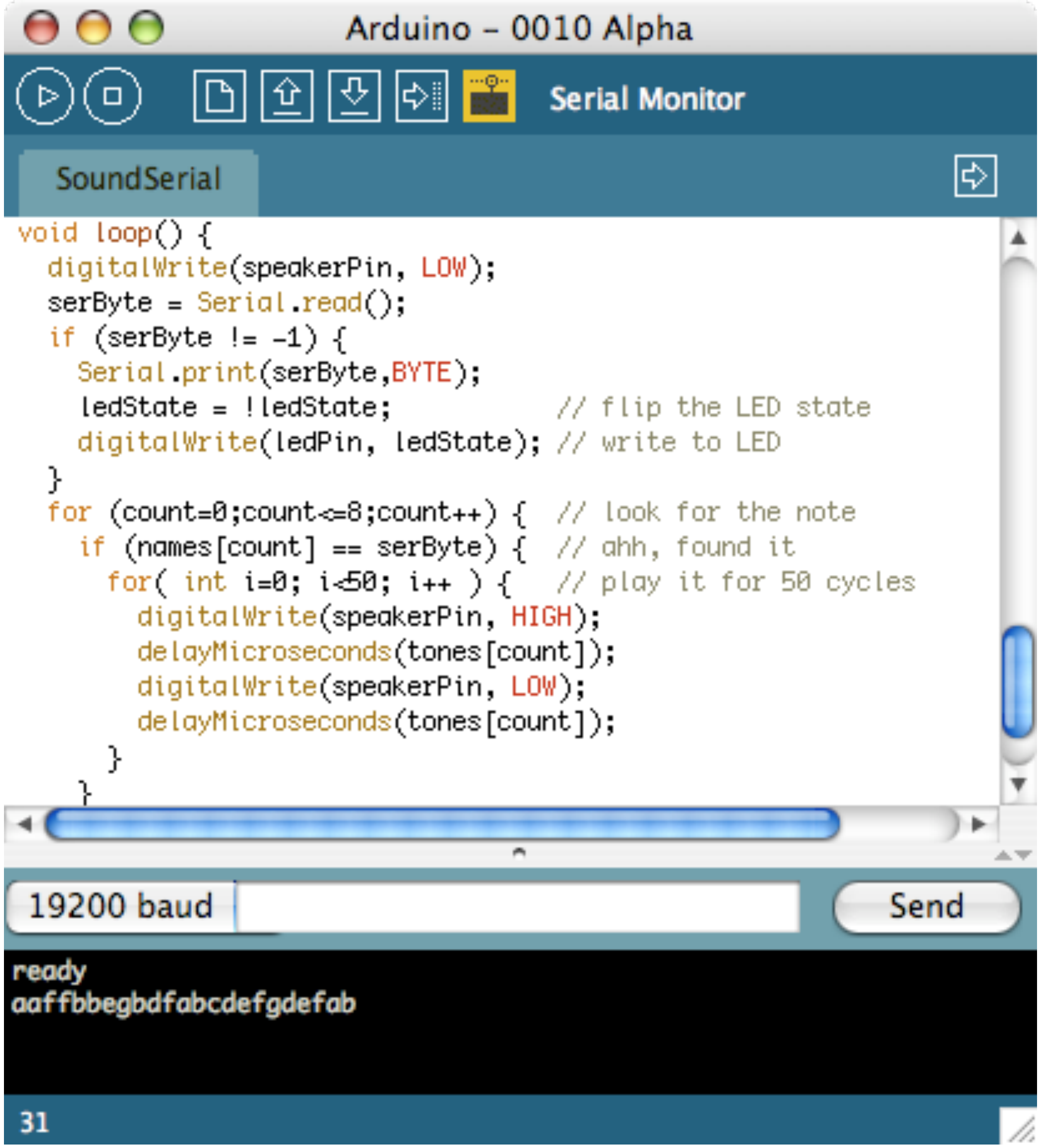
```
//read a string from the serial and store it in an array
//you must supply the array variable
void readSerialString (char *strArray) {
  int i = 0;
  if(!Serial.available()) {
    return;
  }
  while (Serial.available()) {
    strArray[i] = Serial.read();
    i++;
  }
  strArray[i] = 0;  // indicate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

Play a Melody

“SoundSerial”

Play the Speaker with the Serial Monitor

Type multiple letters from “cdefgabC” to make melodies

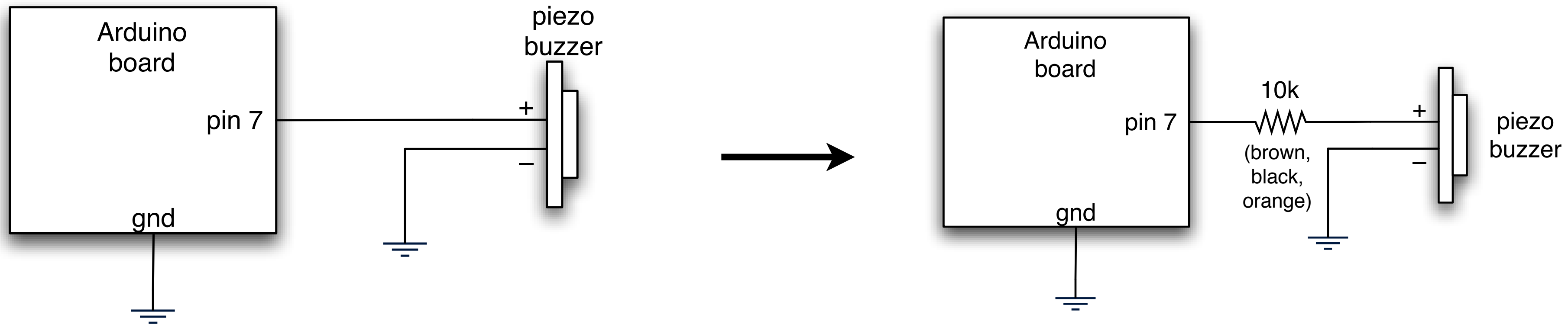


```
void loop() {
  digitalWrite(speakerPin, LOW);
  serByte = Serial.read();
  if (serByte != -1) {
    Serial.print(serByte,BYTE);
    ledState = !ledState;           // flip the LED state
    digitalWrite(ledPin, ledState); // write to LED
  }
  for (count=0;count<=8;count++) {  // look for the note
    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]);
      }
    }
```

```
ready
aaffbbegbdfabcdefgdefab
```

Making it Quieter

Easiest way: add a resistor

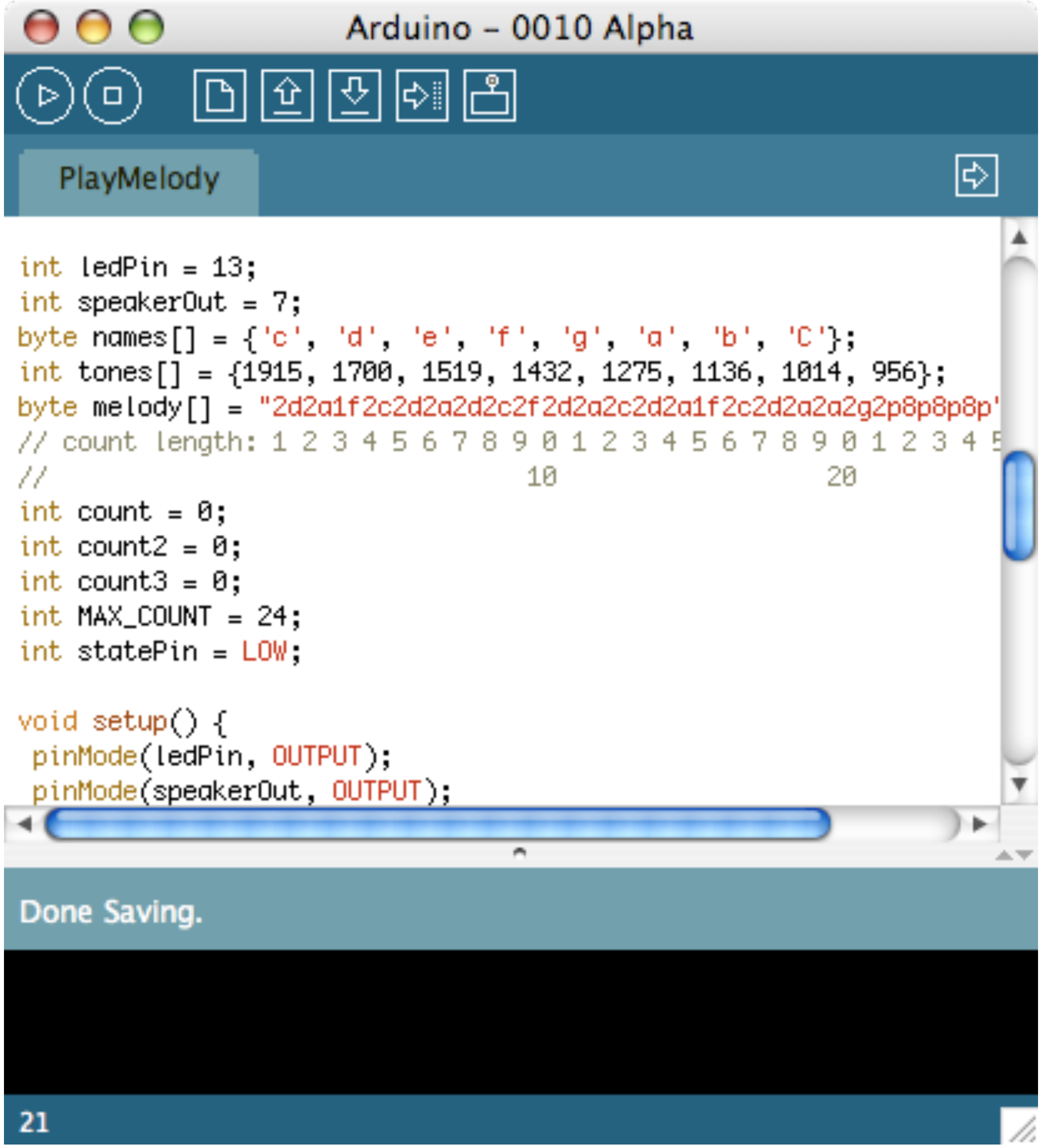


Play a Stored Melody

“PlayMelody”

Plays a melody stored in the Arduino

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



```
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
//                                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);
```

Done Saving.

21

Other Serial Devices

to Wi-Fi   to Ethernet

to graphic LCD

to 8-servo controller

Serial Examples

to Roomba

Going Further

- Can hook up multiple speakers 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)

END Class 2

http://duksta.org/electronics/arduinoclass

John Duksta

john@duksta.org

Giving Credit

This courseware is a mashup of Tod E. Kurt's Bionic Arduino course, taught at Machine Project in LA and Lutz Hamel's Intro to Arduino course taught here at AS220