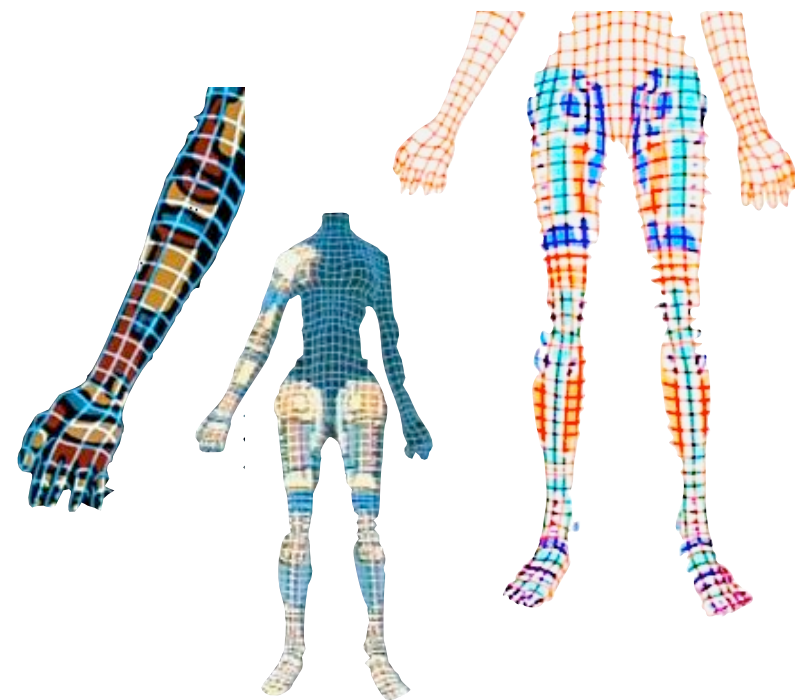


Bionic Arduino

Introduction to Microcontrollers with Arduino

Class I



11 Nov 2007 - machineproject - Tod E. Kurt

Class Info

- Thumbdrive is being passed around, with:
 - PDF version of these notes
 - Arduino software for Mac OS X & Windows
 - Source code (“sketches”) used in class
 - Copy files off, then pass thumbdrive around
- Sunday classes: 3 hours
 - two ~1.5 hour chunks, w/ 15 min. break in middle
- Tuesday classes: ~2.5 hours
 - with some review at the beginning

What's for Today

- Introduction to Arduino
- Setting up your Arduino Environment
- Your first Arduino sketch
- Basic digital & analog output control
- Basic digital sensor inputs
- Making LEDs glow and blink on command
- How to read buttons & switches

Bionic?

Can electronic senses mimic human ones?

Do electronic “muscles” work as well as biological ones? Or better?

What can electronic senses detect that humans can't?

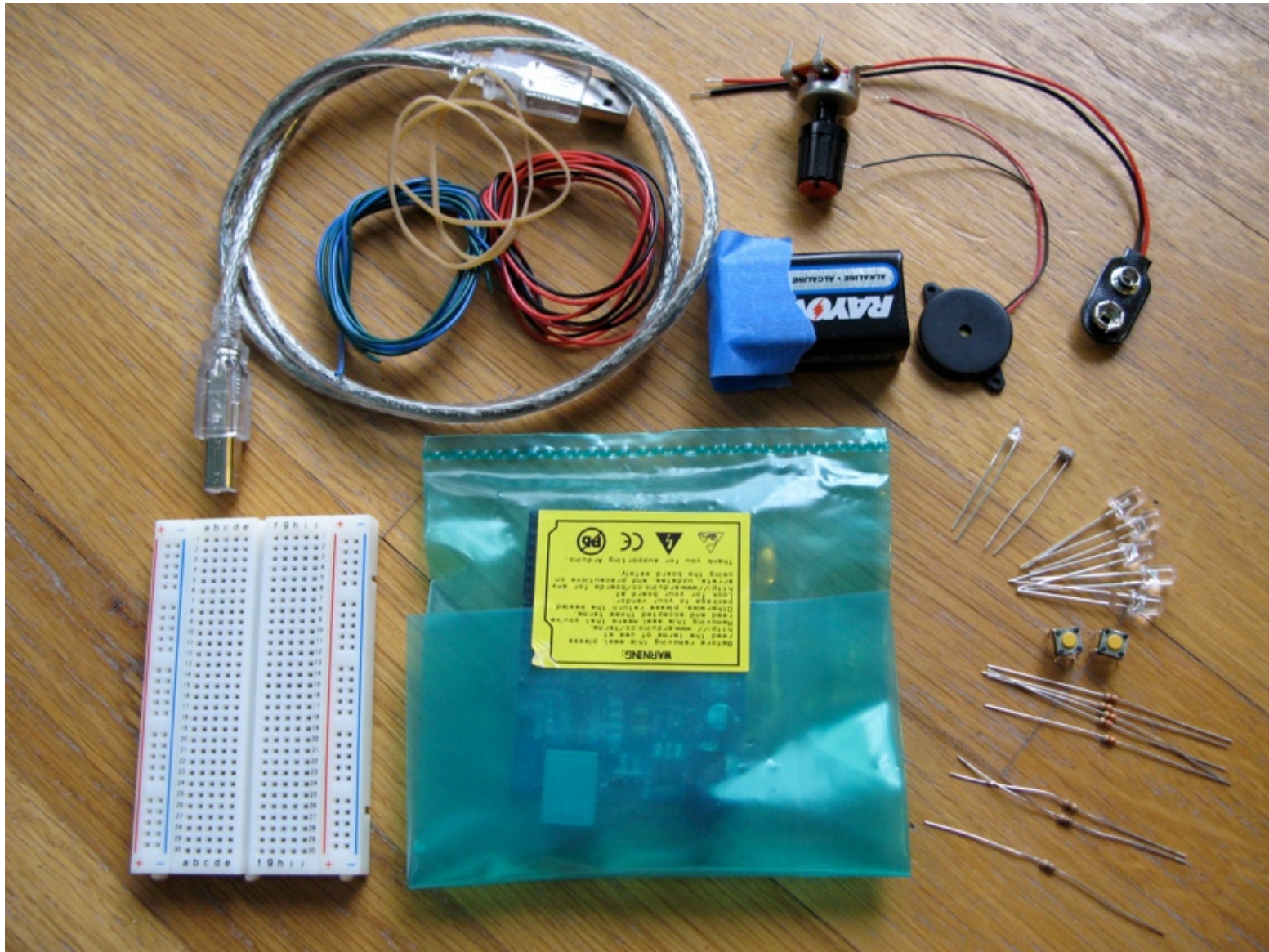
How would you augment yourself with these new abilities?

This class is about exploring the various input & output components used in robots, cell phones, video games, and automobiles, using the friendly Arduino board.

Your devices are watching and responding to you, know their limitations so you can defeat them when the machine uprising comes.

At worst, you'll be able to fashion a convincing disguise from pasting Arduinos on your body.

Class Kit I Contents



Class Kit 2 comes next week

A little shoebox-sized plastic storage bin makes a good holder for your electronics stuff.

Not shown, RGB LED. oops. It showed up late to the photoshoot.

Class Kit I Manifest

Setup and “light & sound”

- Arduino Diecimila USB board
- Solderless breadboard
- USB cable
- piezo buzzer
- potentiometer with knob
- 5 orange LEDs (large, clear)
- 1 RGB LED (diffuse, com. anode)
- two push switches
- 9V battery and connector
- resistors:
 - 6 x 220 ohm (red-red-brown)
 - 2 x 10k (brown-black-orange)
 - 1 x 1M (brown-black-green)
- photocell
- phototransistor (small, clear)
- 4 colors of hookup wire
- rubber bands

There will be a second update kit next week: “motion & motors”

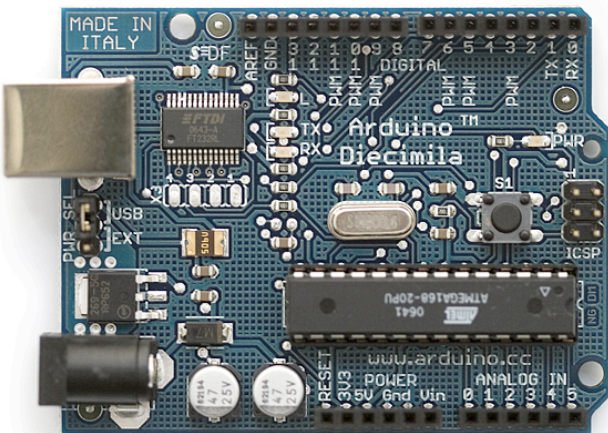
A Word on Safety

- Electronics can hurt you
 - Lead in some of the parts
 - Wash up afterwards
- You can hurt electronics
 - Static-sensitive: don't shuffle your feet & touch
 - Wires only bend so much

What is Arduino?

The word “Arduino” can mean 3 things

A physical piece
of hardware



A programming
environment

A community
& philosophy

Arduino playground

Search: [] [G]

Edit | History | Attachments | About Arduino playground

Projects Built with Arduino

Manuals

Arduino Tutorials

- Official tutorial page
- Information on the Arduino Mini
- Bluetooth tutorial
- Xbee (Zigbee) tutorial
- DIY Arduino Shield
- Serial LCD tutorial
- Learning Electronics
- Learning Programming
- Asuro Robot and Arduino
- Let the Arduino Sleep
- Burning Bootloaders
- CourseWare
- Stand alone 3.3V 8 MHz
- Programming the Bluetooth WT11
- Retrofitting Decimila AutoReset
- Bare Bones "Decimila" Upgrade

Run Arduino on...

- Linux
- Ubuntu Linux
- Debian Linux
- Gentoo Linux
- Slackware Linux
- FreeBSD
- Fedora Linux
- The command line

Interface Arduino with...

- Instant Reality (X3D)
- Flash
- Processing
- PD (Pure Data)

:: About the Arduino Playground ::

Welcome to the Arduino Playground, a wiki where all the users of [Arduino](#) can contribute and benefit from their collective research.

This is the place to post and share your own code, circuit diagrams, tutorials, DIY instructions, tips and tricks, and after all the hard work, to show off your projects!

Arduino Playground is a **work in progress**. We can use all the help you can give, so please read the [Participate](#) section and get your fingers typing!

:: Roadmap: What Needs to be Done? ::

There is a lot to do...most of the pages are just stubs, simple placeholders waiting for you to fill them up. However here is a small roadmap of things I personally think should be developed first (again, this is a wiki so you are more than welcome to get in

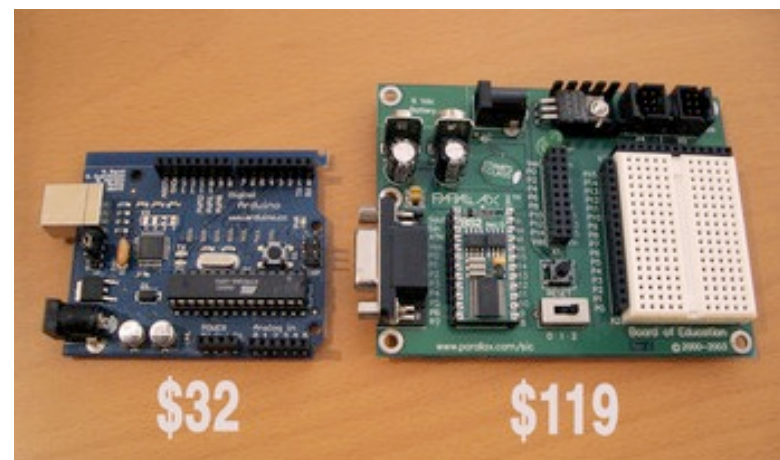
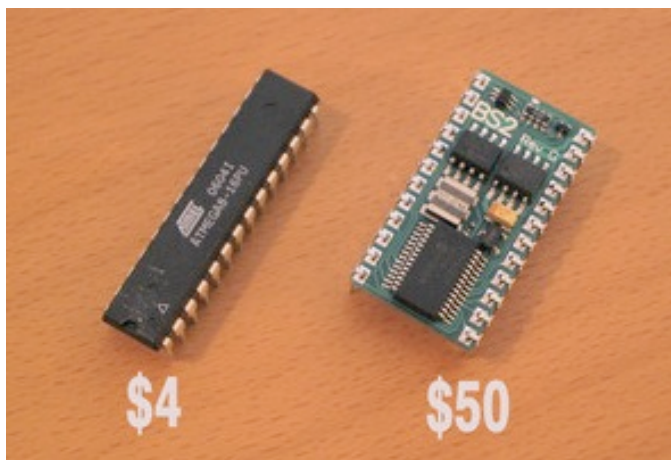
Arduino

Philosophy & Community

- Open Source Physical Computing Platform
 - “open source hardware”
 - open source: free to inspect & modify
 - physical computing. er, what? ubiquitous computing, pervasive computing, ambient intelligence, calm computing, everywhere, spimes, blogjects, smart objects...
- Community-built
 - Examples wiki (the “playground”) editable by anyone
 - Forums with lots of helpful people

Arduino Hardware

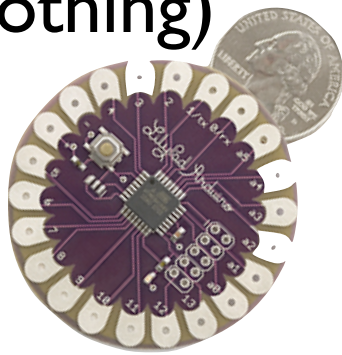
- Similar to Basic Stamp (if you know of it)
 - but cheaper, faster, & open
- Uses AVR ATmega168 microcontroller chip
 - chip was designed to be used with C language



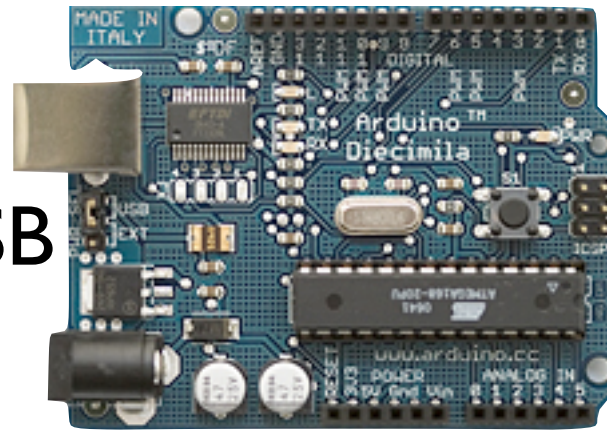
The designer of the AVR purposefully arranged its registers and instruction set so that C programs would compile efficiently on it. This is a big deal, compared to previous microcontrollers where C programs were almost always less efficient than a hand-coded assembly language variant.

Arduino Hardware Variety

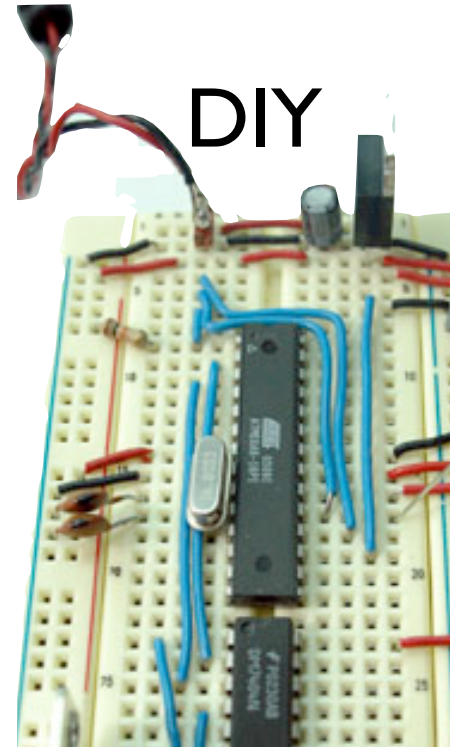
LilyPad
(for clothing)



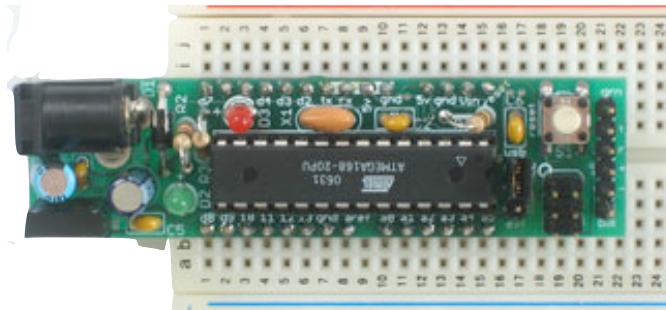
USB



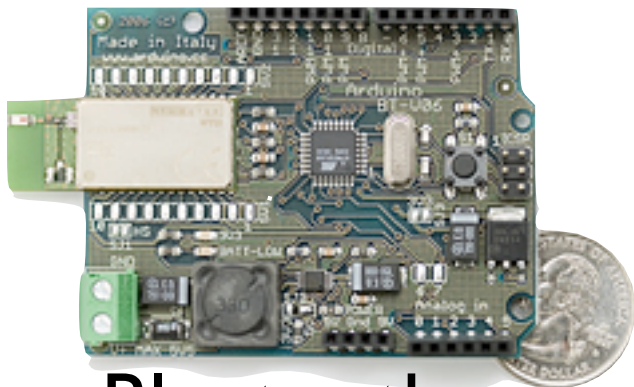
DIY



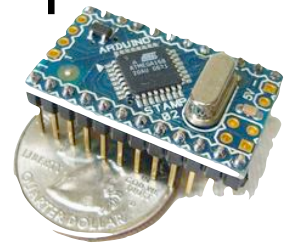
Boarduino Kit



Bluetooth



“Stamp”-sized



many different variations to suite your needs

Openness has its advantages, many different varieties.
Anyone can build an Arduino work-alike in any form-factor they want.
Product images from Sparkfun.com and Adafruit.com

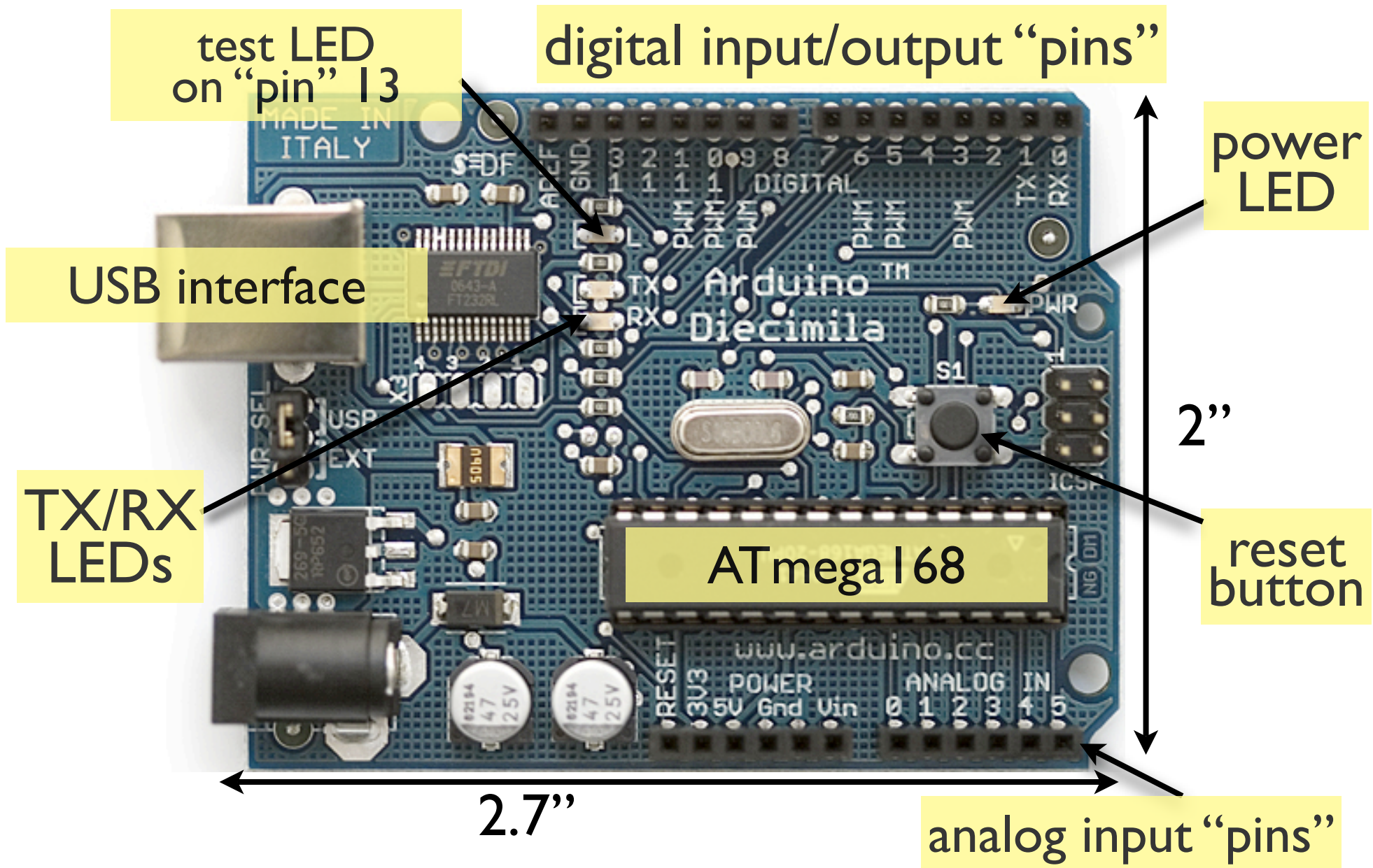
Arduino Capabilities

- 16 kBytes of Flash program memory
- 1 kByte of RAM
- 16 MHz (Apple II: 1 MHz)
- Inputs and Outputs
 - 13 digital input/output pins
 - 5 analog input pins
 - 6 analog output pins*
- Completely stand-alone: doesn't need a computer once programmed

* only sorta analog, uses PWM , which we'll talk about later.

Don't worry if the above doesn't make sense, you don't really need to know it.

Arduino Diecimila Board



Arduino Terminology

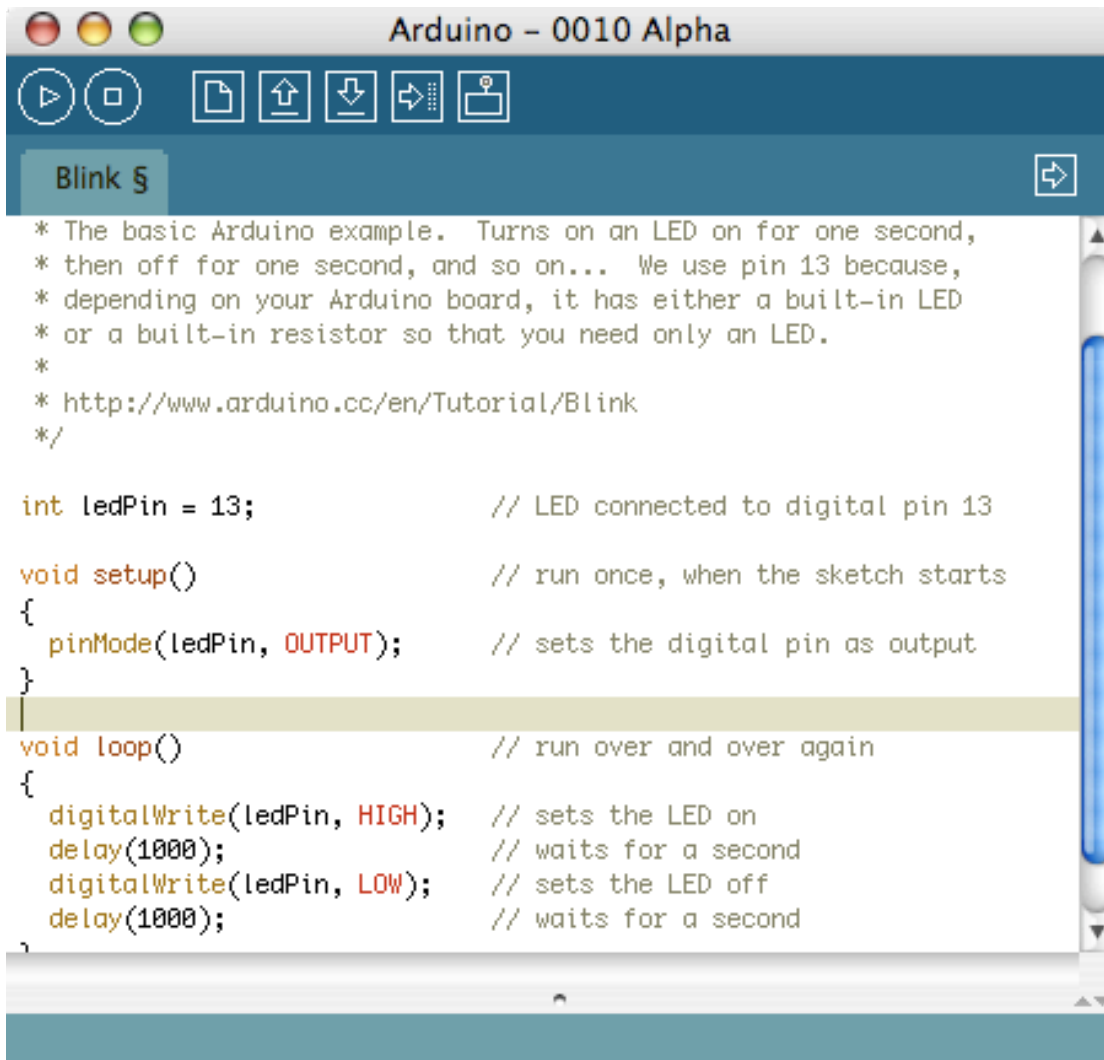
“*sketch*” – a program you write to run on an Arduino board

“*pin*” – an input or output connected to something.
e.g. output to an LED, input from a knob.

“*digital*” – value is either HIGH or LOW.
(aka on/off, one/zero) e.g. switch state

“*analog*” – value ranges, usually from 0-255.
e.g. LED brightness, motor speed, etc.

Arduino Software



The screenshot shows the Arduino IDE interface. The title bar reads "Arduino - 0010 Alpha". The menu bar includes icons for Run, Stop, New, Open, Save, Undo, Redo, and Help. The sketch name "Blink 5" is displayed in the top left. The code is as follows:

```
* The basic Arduino example. Turns on an LED on for one second,
* then off for one second, and so on... We use pin 13 because,
* depending on your Arduino board, it has either a built-in LED
* or a built-in resistor so that you need only an LED.
*
* http://www.arduino.cc/en/Tutorial/Blink
*/

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

void setup()              // run once, when the sketch starts
{
  pinMode(ledPin, OUTPUT); // sets the digital pin as output
}

void loop()               // run over and over again
{
  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
}
```

- Like a text editor
- View/write/edit sketches
- But then you program them into hardware

If you've used Processing to write little Java programs, you'll notice the interface looks familiar. Arduino takes the editor GUI from Processing and some of its philosophy, but Arduino code and Processing code are totally unrelated.

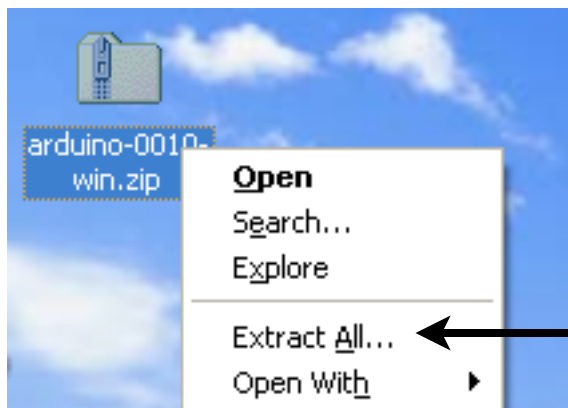
Installing Arduino

The Steps

1. Get the Arduino software & unzip it
2. Plug in Arduino board
3. Install the driver
4. Reboot
5. Run the Arduino program
6. Tell Arduino (program) about Arduino (board)

Getting and Unpacking

- On the thumbdrives
 - “arduino-0010-win.zip” for Windows
 - “arduino-0010-mac.zip” for Mac OS X
- Unzip the zip file. Double-click on Mac



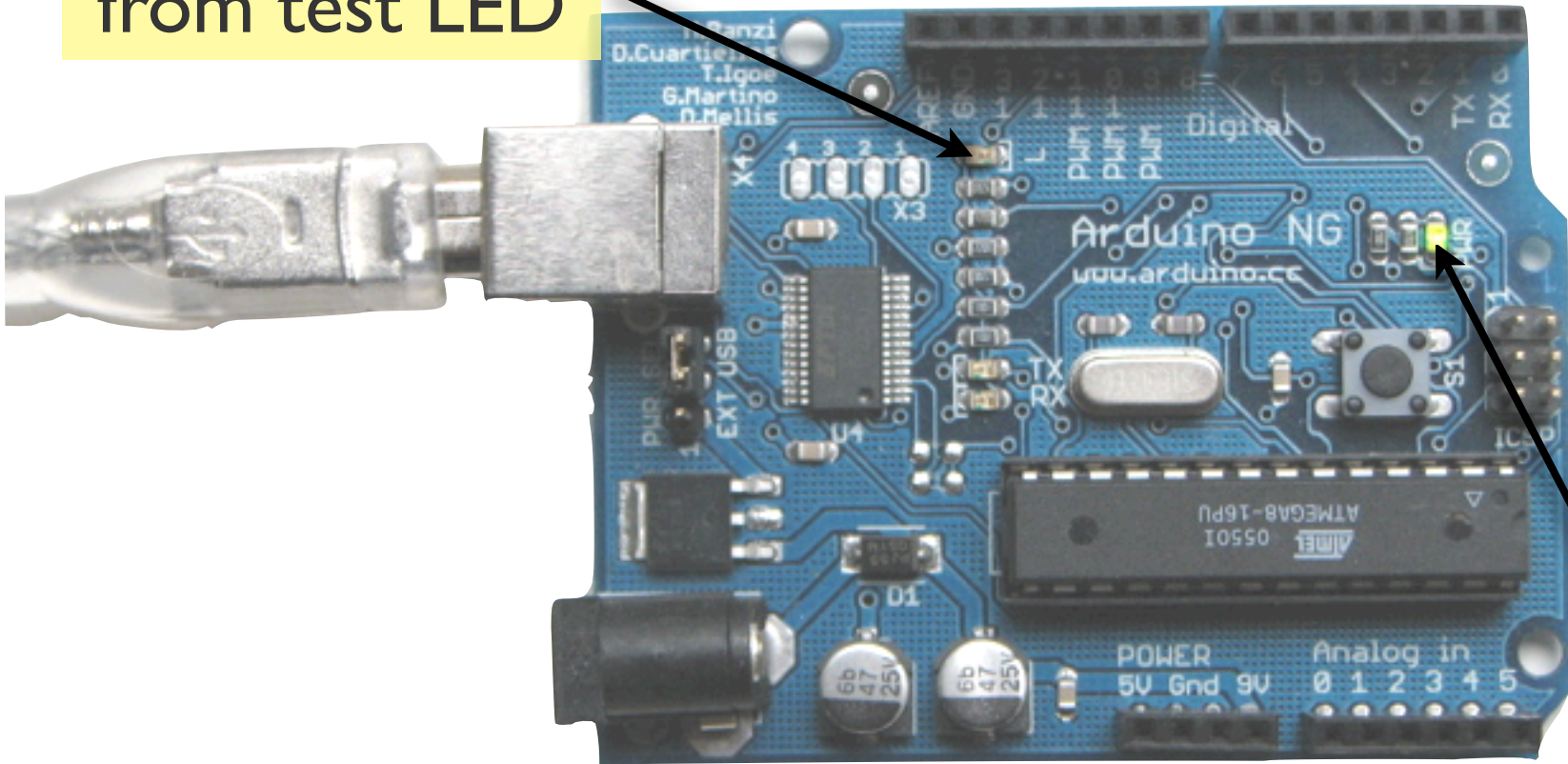
On Windows, right-click

Use “Extract All...”

- Find the “drivers” directory inside

Plug in Arduino board

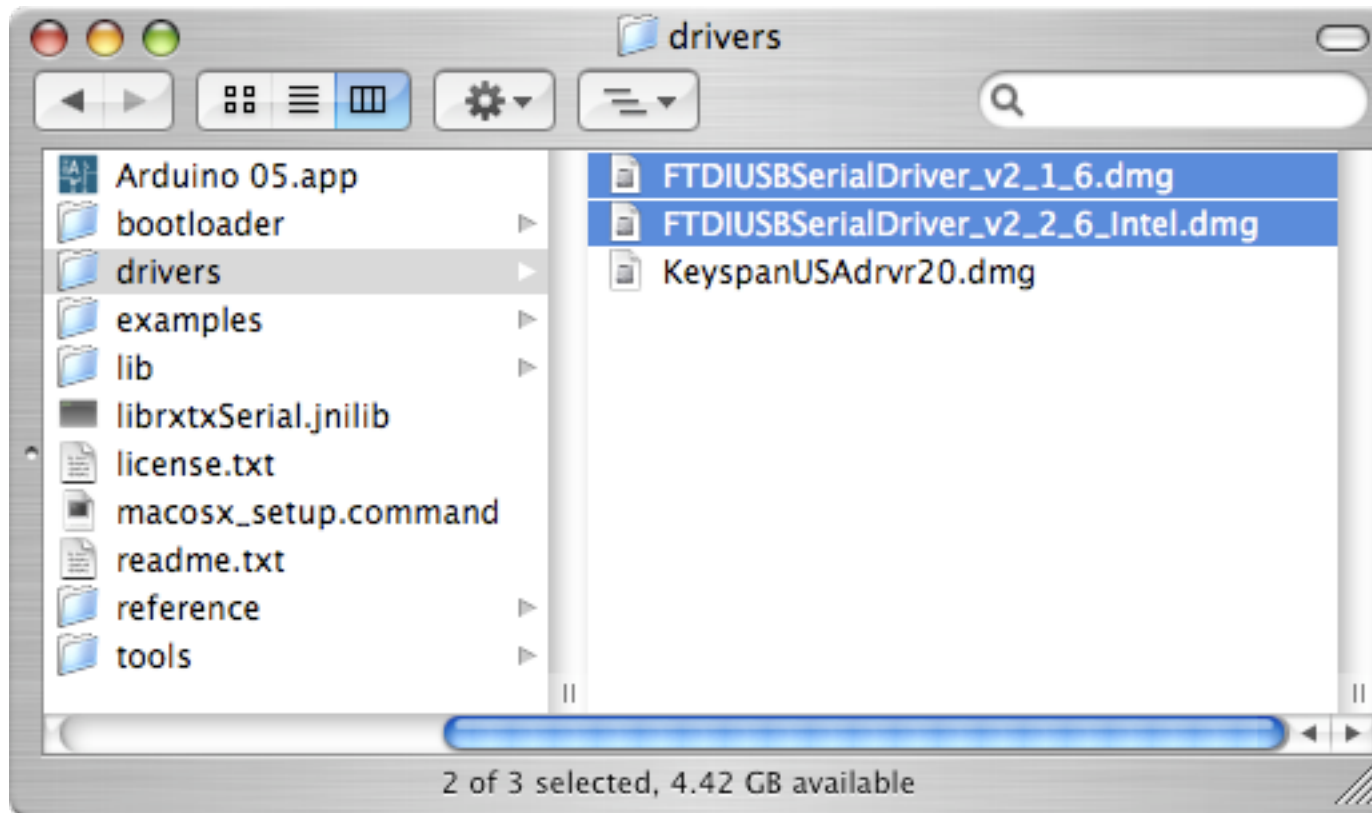
quick blink
from test LED



Power LED should stay on

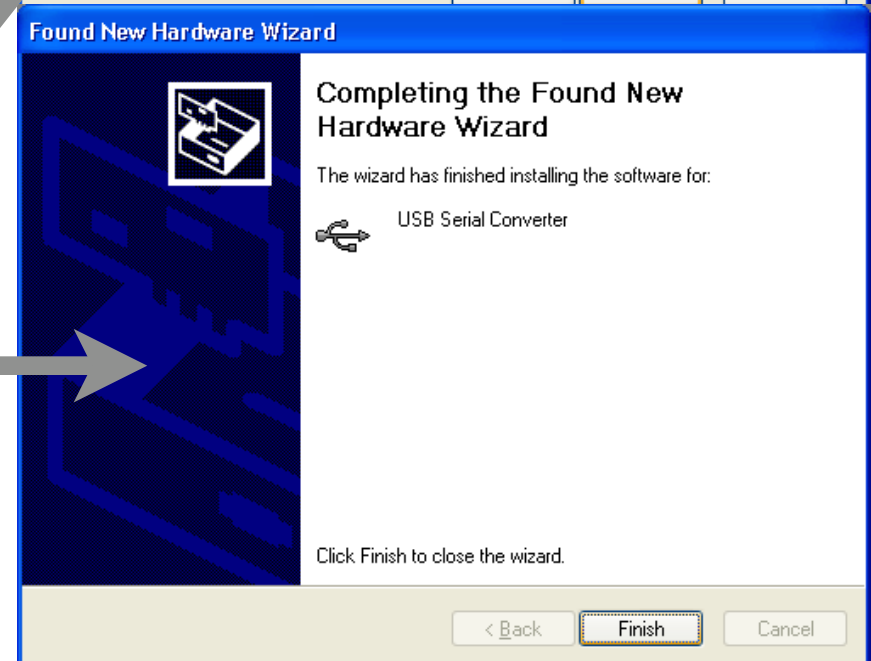
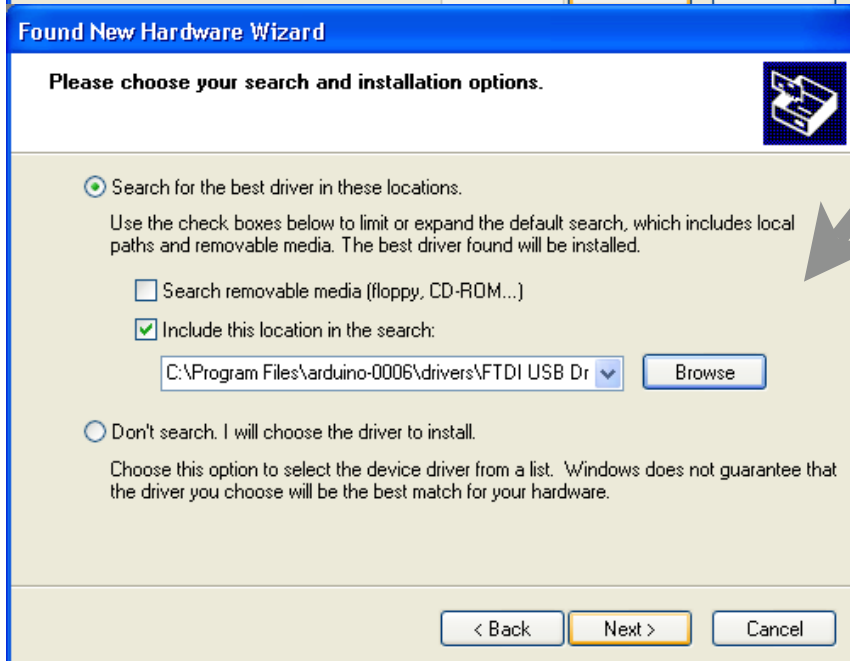
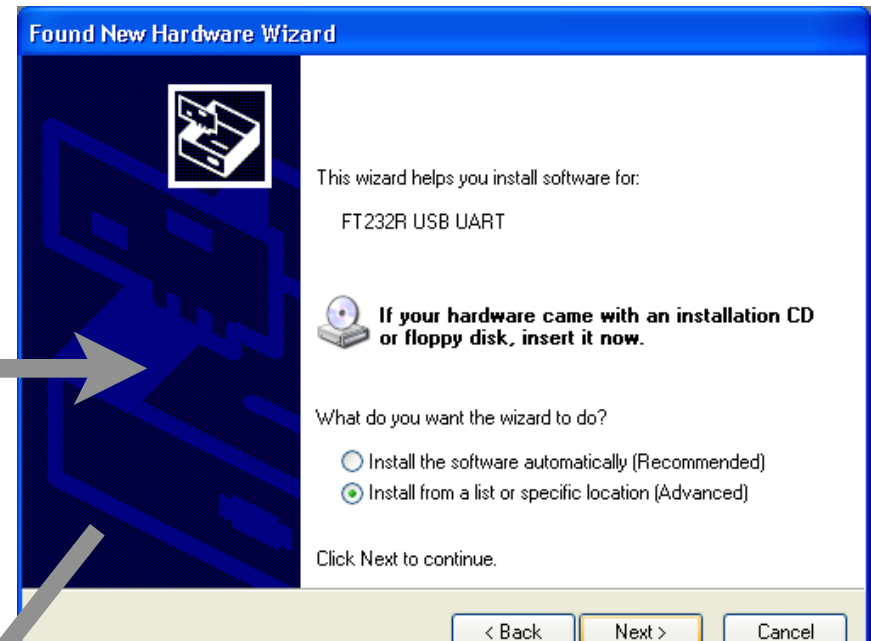
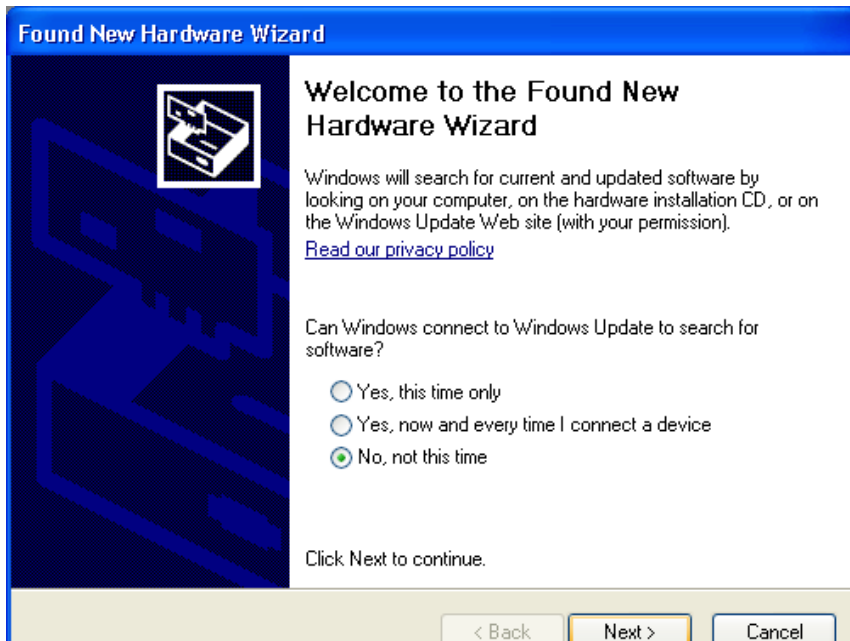
Mac Driver Install

Double-click on .dmg Installer

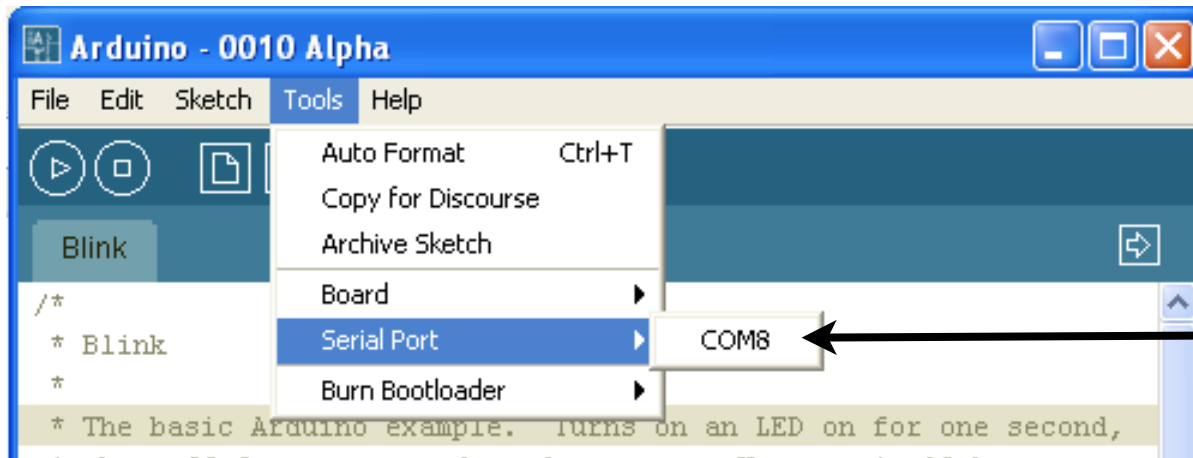


- v2_1_6 for PPC Macs
- v2_2_6 for Intel Macs

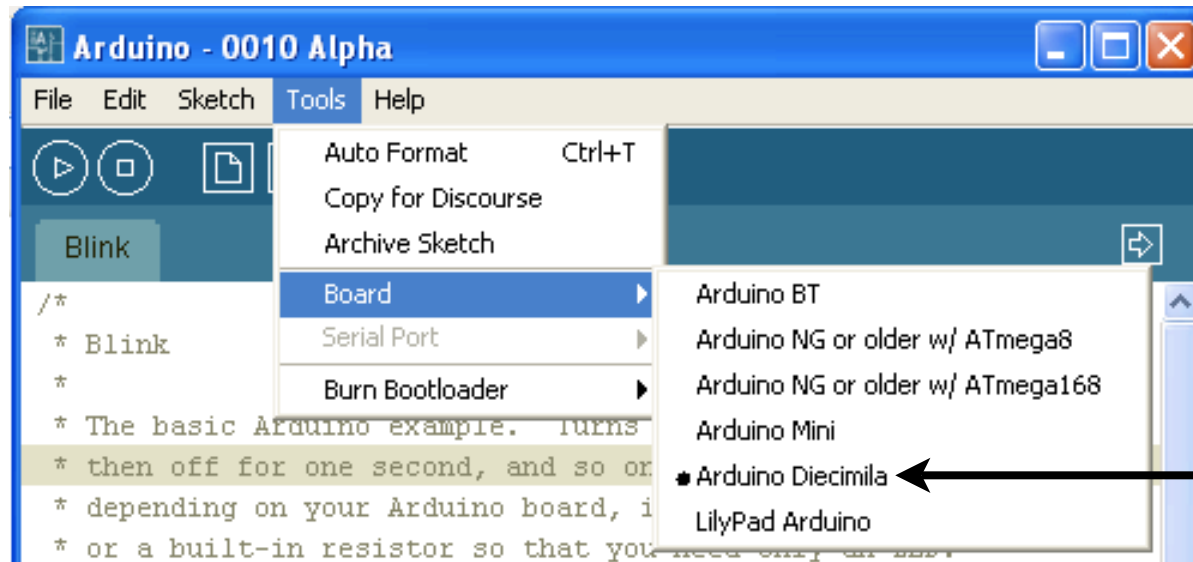
Windows Driver Install



Selecting Location & Type

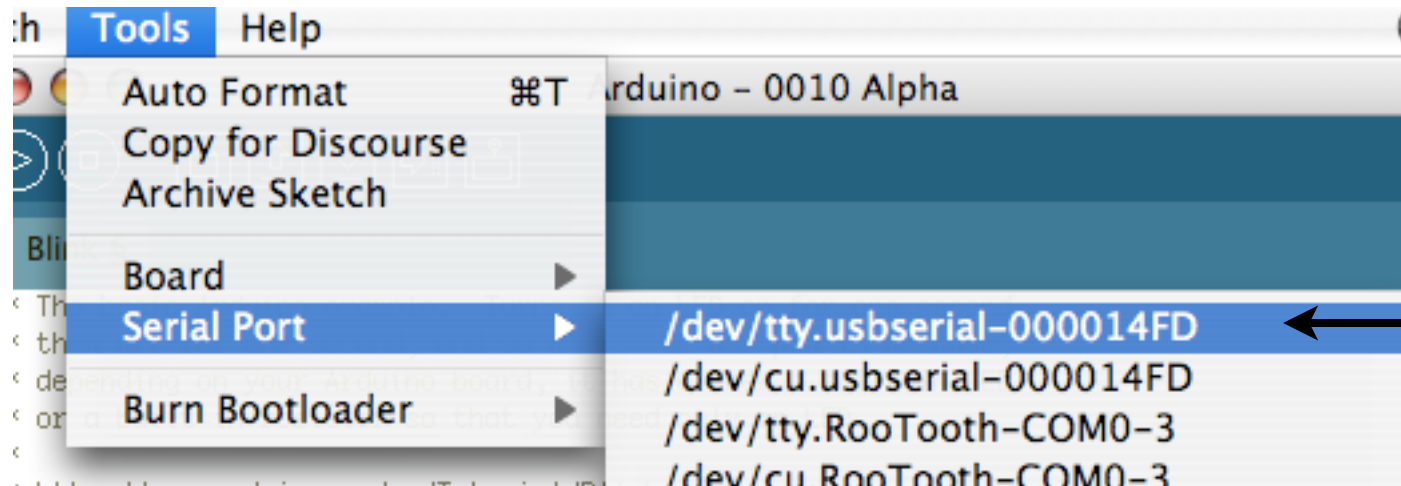


usually highest-numbered port

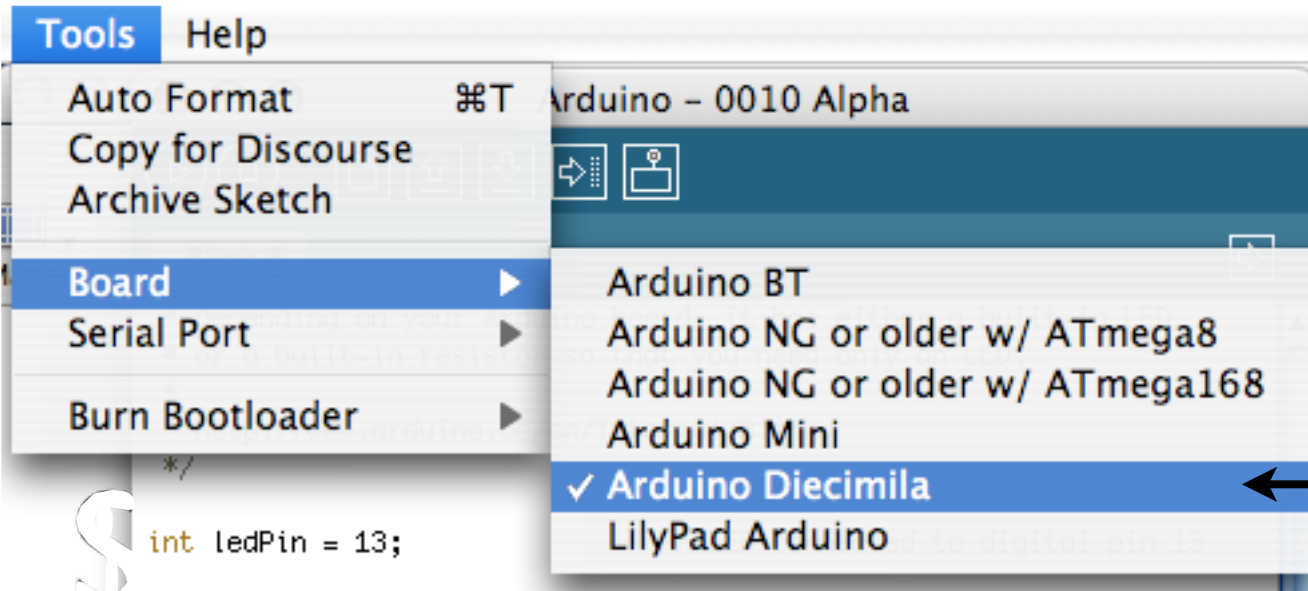


pick "Diecimila"

Selecting Location & Type



starts with
tty.usbserial-



pick "Diecimila"

Arduino Software

compile
(verify)

upload to board

status
area



Using Arduino

- Write your sketch
- Press Compile button (to check for errors)
- Press Upload button to program Arduino board with your sketch

Try it out with the “Blink” sketch!

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

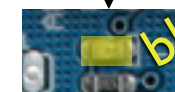
Done compiling.



upload



TX/RX flash



sketch runs


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

Status Messages

Uploading worked

```
Done uploading.  
Binary sketch size: 1110 bytes (of a 14336 byte maximum)
```

Size depends on
complexity of your sketch



Wrong serial port selected

```
Serial port '/dev/tty.usbserial-A4001qa8' not found. Did you select the  
java.awt.EventQueueThread.pumpEvents(EventDispatchThread.java:170  
)  
at  
java.awt.EventQueueThread.run(EventDispatchThread.java:110)
```

Wrong board selected

```
Wrong microcontroller found. Did you select the right board from the T  
Binary sketch size: 800 bytes (of a 7100 byte maximum)  
avrdude: Expected signature for ATMEGA8 is 1E 93 07  
Double check chip, or use -F to override this check.
```

nerdy cryptic error messages



Troubleshooting

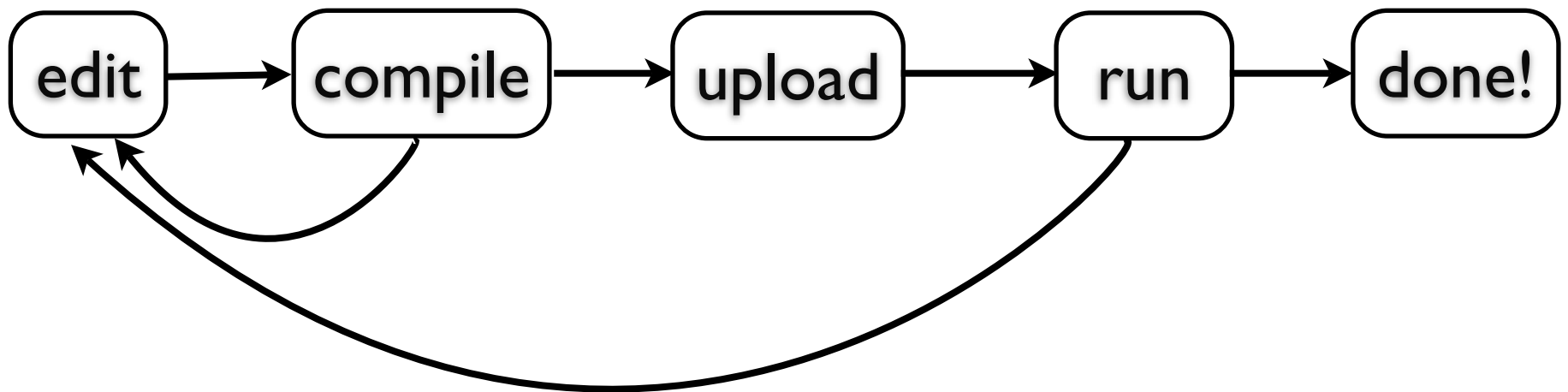
- Most common problem is incorrect serial port setting
- If you ever have any “weird” errors from the Arduino environment, just try again.
- The red text at the bottom is debugging output in case there may be a problem
- Status area shows summary of what’s wrong

I made an LED blink, so what?

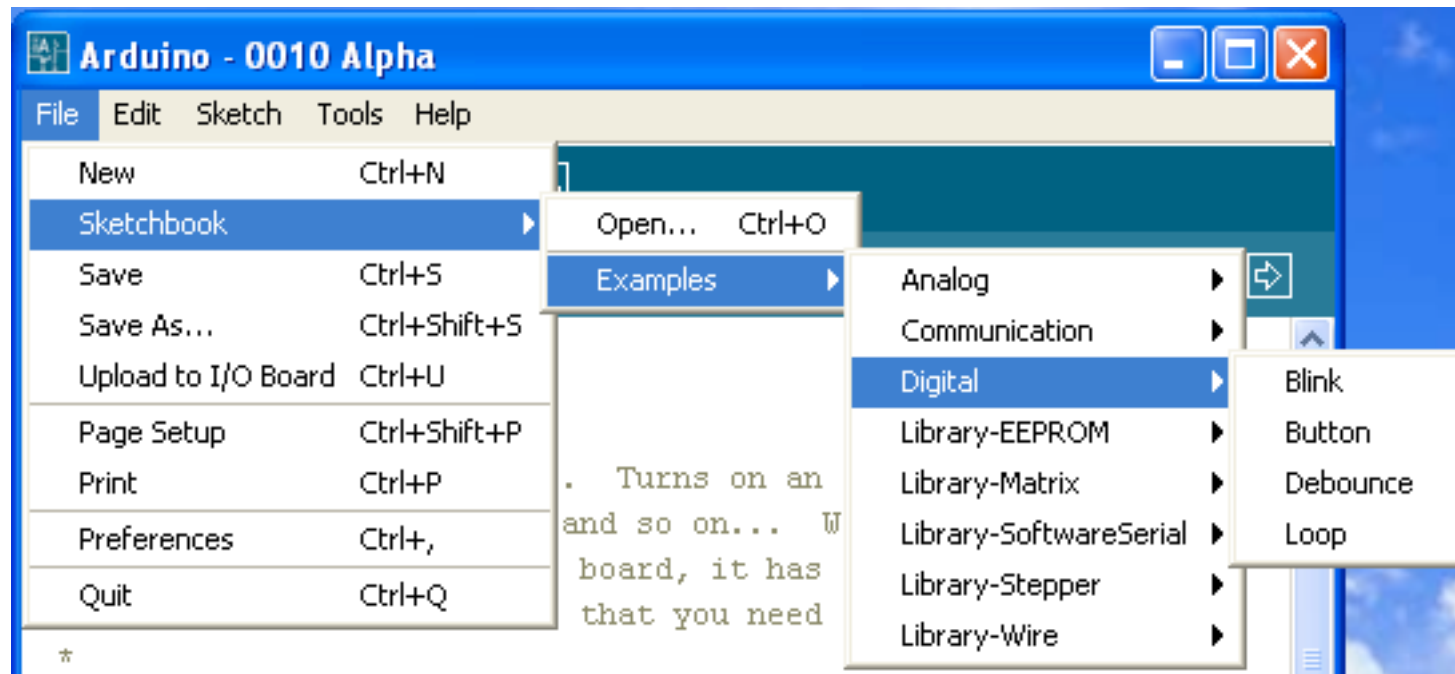
- Most actuators are switched on and off with a digital output
- The `digitalWrite()` command is the software portion of being able to control just about anything
- LEDs are easy, motors come in a bit
- Arduino has up to 13 digital outputs, and you easily can add more with helper chips

Development Cycle

- Make as many changes as you want
- Not like most web programming: edit → run
- Edit → compile → upload → run



Lots of Built-in Examples



And more here:

<http://www.arduino.cc/en/Tutorial/HomePage>

And all over the Net. Search for "Arduino tutorial" or "Arduino notes" or whatever you're interested in and "Arduino" and likely you'll find some neat pages.

Take a Break

Grab a coffee upstairs at Downbeat Cafe.

Arduino “Language”

- Language is standard C (but made easy)
- Lots of useful functions
 - `pinMode()` – set a pin as input or output
 - `digitalWrite()` – set a digital pin high/low
 - `digitalRead()` – read a digital pin’s state
 - `analogRead()` – read an analog pin
 - `analogWrite()` – write an “analog” value
 - `delay()` – wait an amount of time
 - `millis()` – get the current time
- And many others. And libraries add more.

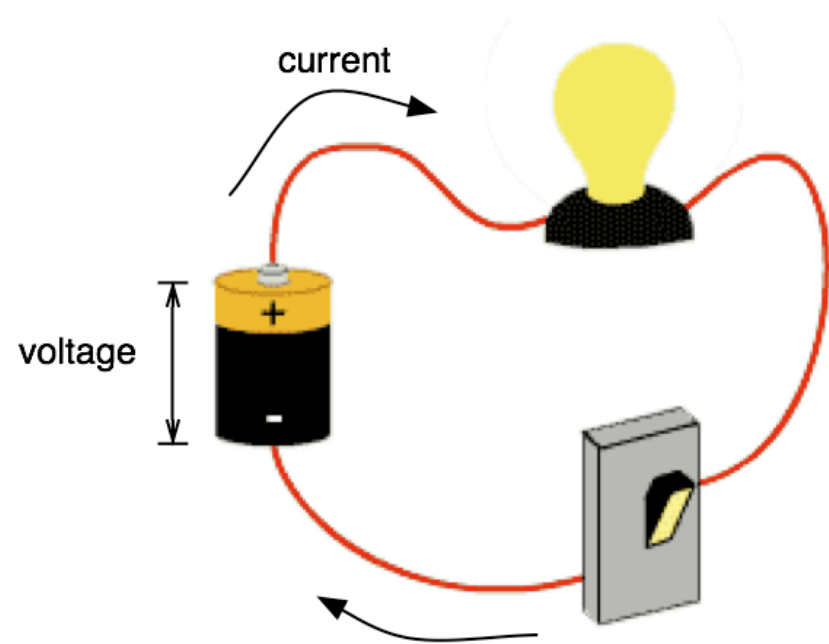
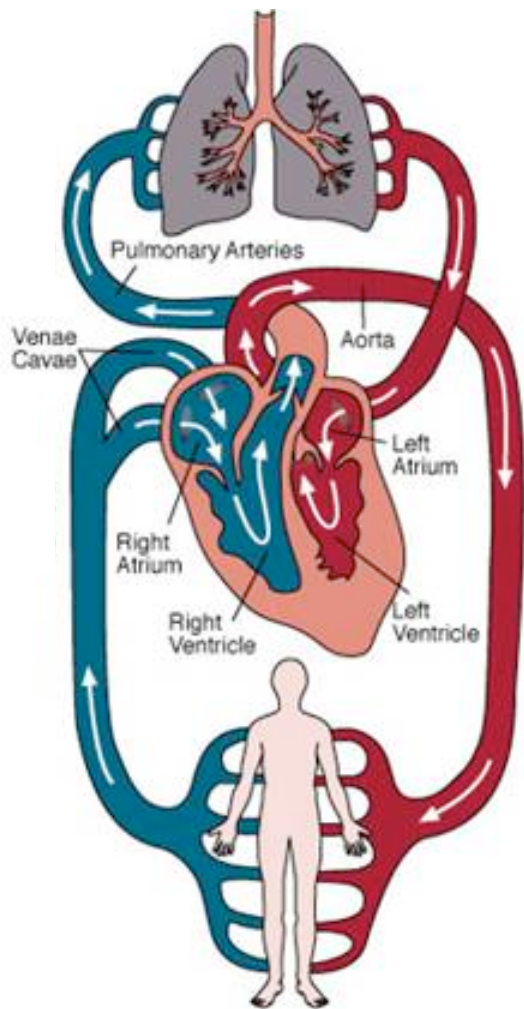
Also: serial library, LCD library, servo examples

Sketch structure

- Declare variables at top
- Initialize
 - `setup()` – run once at beginning, set pins
- Running
 - `loop()` – run repeatedly, after `setup()`

Pins can be changed in `loop()` too, but conceptually easier in `setup()`

Making Circuits

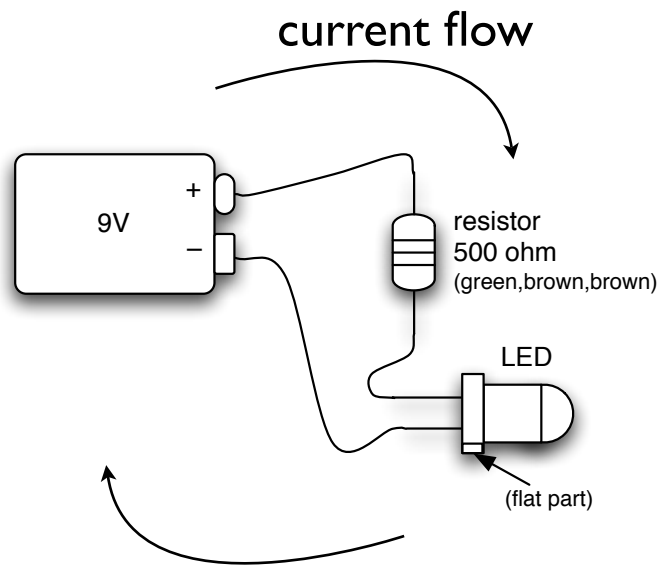


heart pumps, blood flows

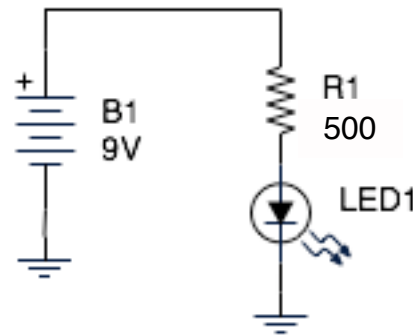
voltage pushes, current flows

It's all about the flow of current. Kinda like the flow of liquid.
Some electronics devices hold back current, like a tiny hose. These are "resistors".

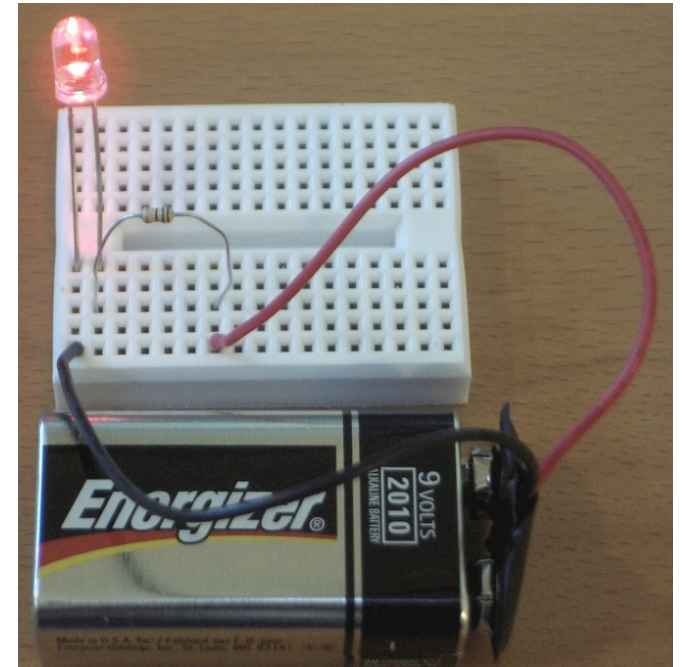
Example: LED flashlight



wiring diagram



schematic



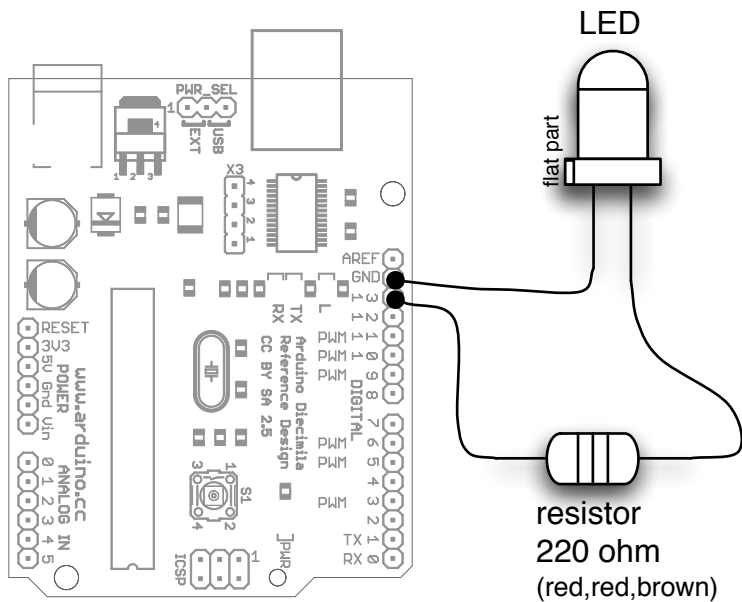
wiring it up

Electricity flows in a loop. Can stop flow by breaking the loop

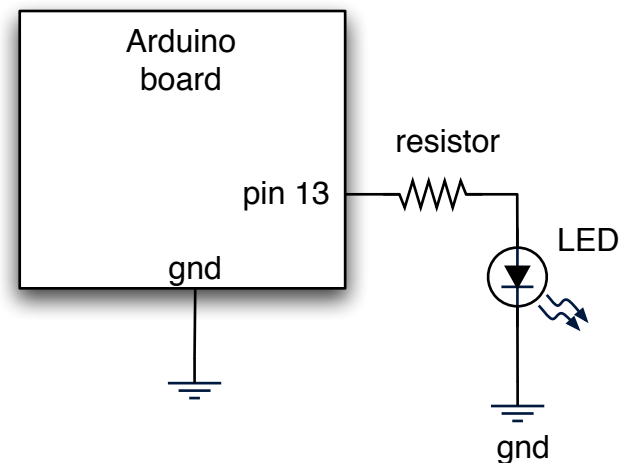
All LED circuits are essentially this: power source, current limiter, LED
Flat part of LED goes to negative, like bar in schematic
The higher the resistance, the dimmer the LED; the lower, the brighter
You don't have to wire this up, but the following circuits are just the same

The Circuit for LED Blink

“hello world” of microcontrollers



wiring diagram



schematic

Arduino Diecimila board has this circuit built-in

To turn on LED use `digitalWrite(13,HIGH)`

This is a “computer-controlled LED flashlight”.

In schematics signals often flow from top-left to bottom-right.

Common nodes like “gnd” are given their own symbol.

You could wire this circuit up on any digital pin, doesn't matter which.

Same circuit as last page, but “battery” is pin 13 of Arduino, and you can turn it on and off.

Schematics are pretty easy to learn, not many people use wiring diagrams.

LEDs & Resistors

On LEDs, polarity matters.
Shorter lead is “negative” side, goes to ground



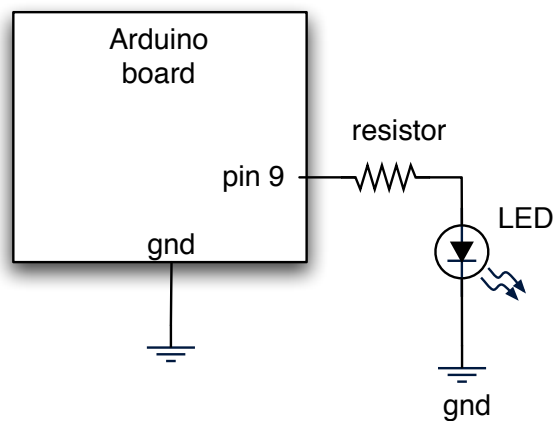
Flat edge here for neg. side



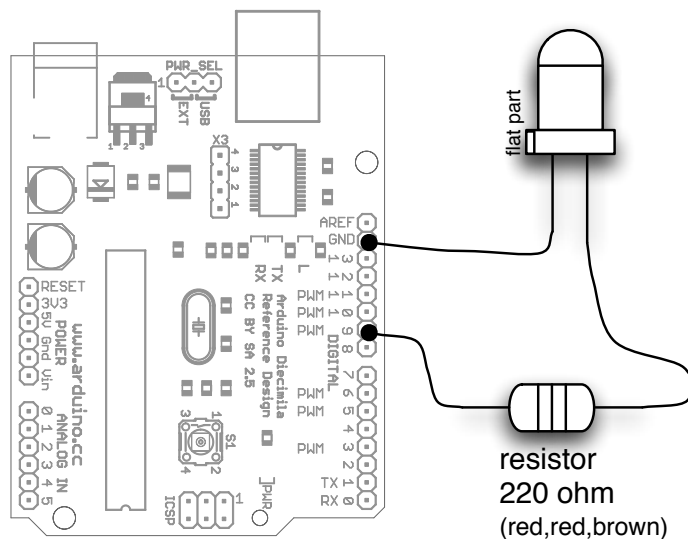
Polarity doesn't matter on resistors

Varying LED Brightness

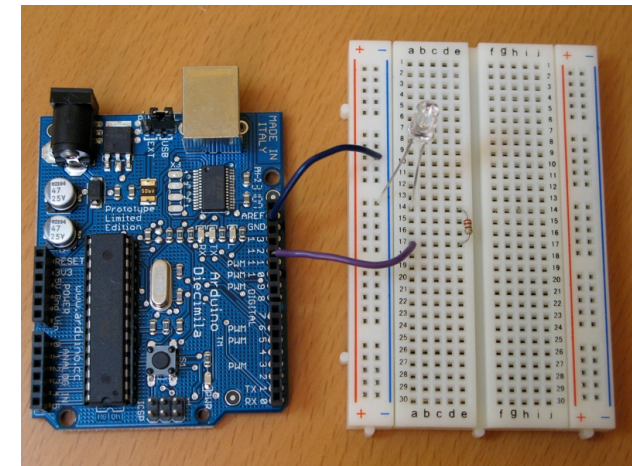
Same circuit as Blink circuit but pin 9 instead of pin 13



schematic



wiring diagram



wired up

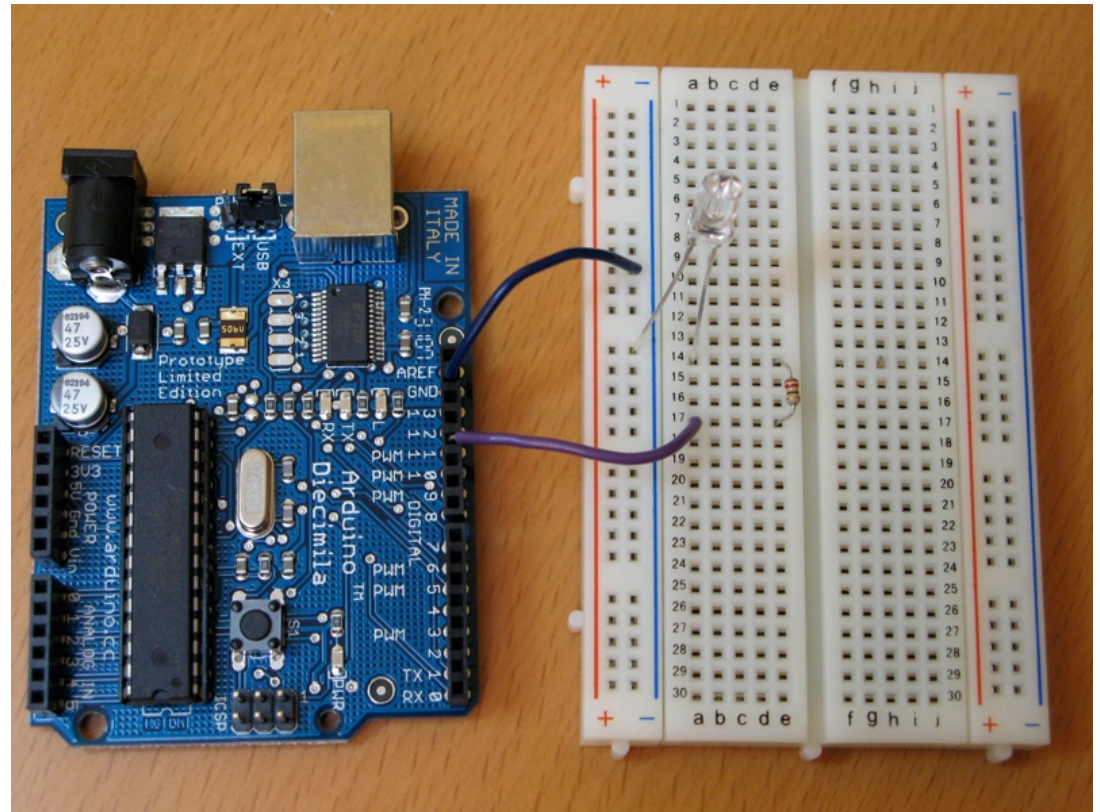
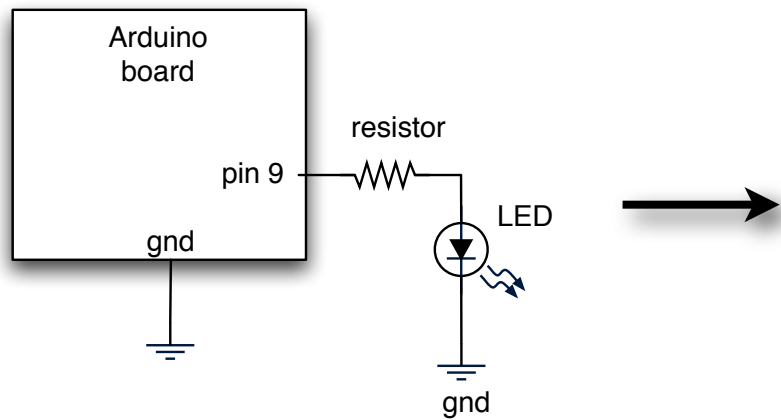
The PWM pins work with the “`analogWrite(value)`” command where “value” ranges between 0 and 255.

To turn LED to half-bright, use `analogWrite(9,128)`

More about PWM later, but it only works on those pins labeled “PWM”.

Very quickly, it works by making and breaking the flow several hundred times a second. So really it's flashing, just like blink, but doing it very fast. Our eyes make it look like brighter or dimmer. We'll have to build this circuit.

Let's Wire It Up



Going from schematic to physical circuit.

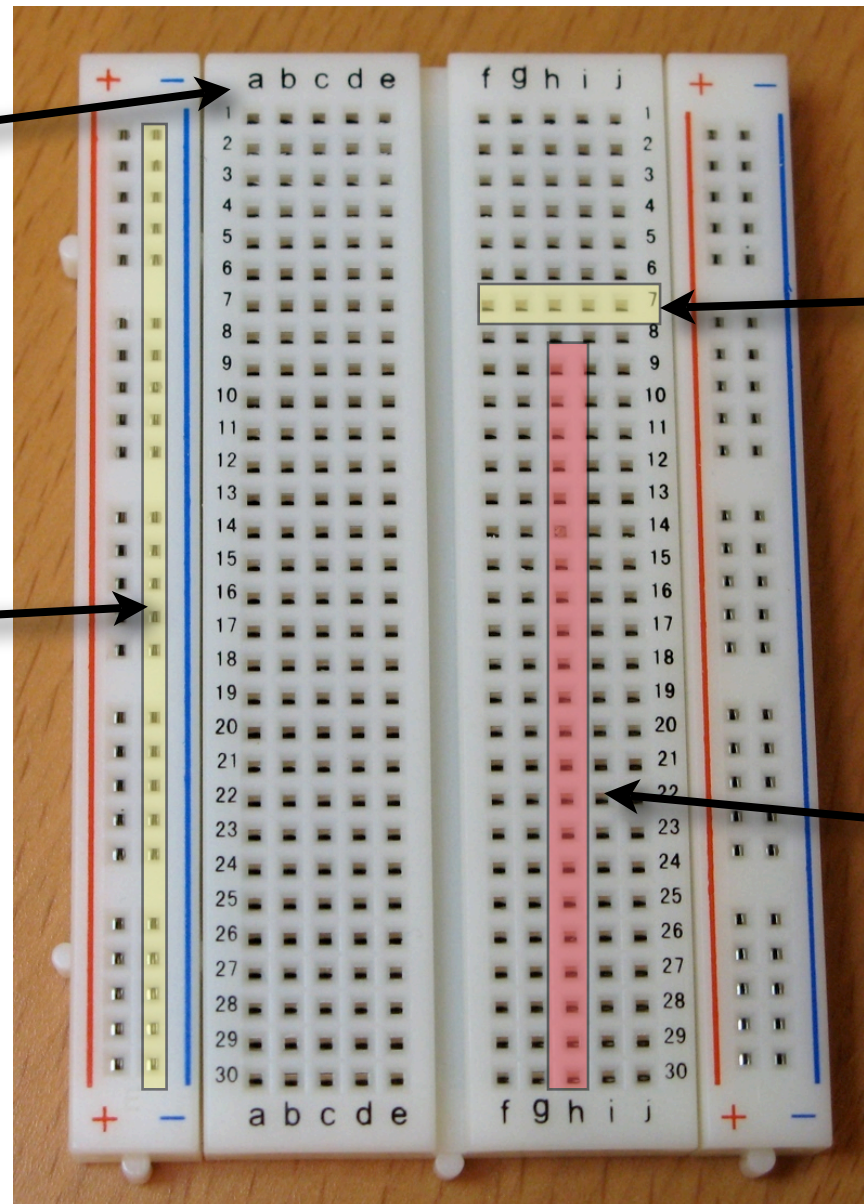
Solderless Breadboards

numbers & letter labels just for reference

All connected, a "bus"

groups of 5 connected

not connected



Insert wires into holes to make a connection.

Much easier, quicker than soldering

But, they wear out, are kind of expensive (\$5 for this one, at that was a bargain)

Useful Tools



Wire stripper

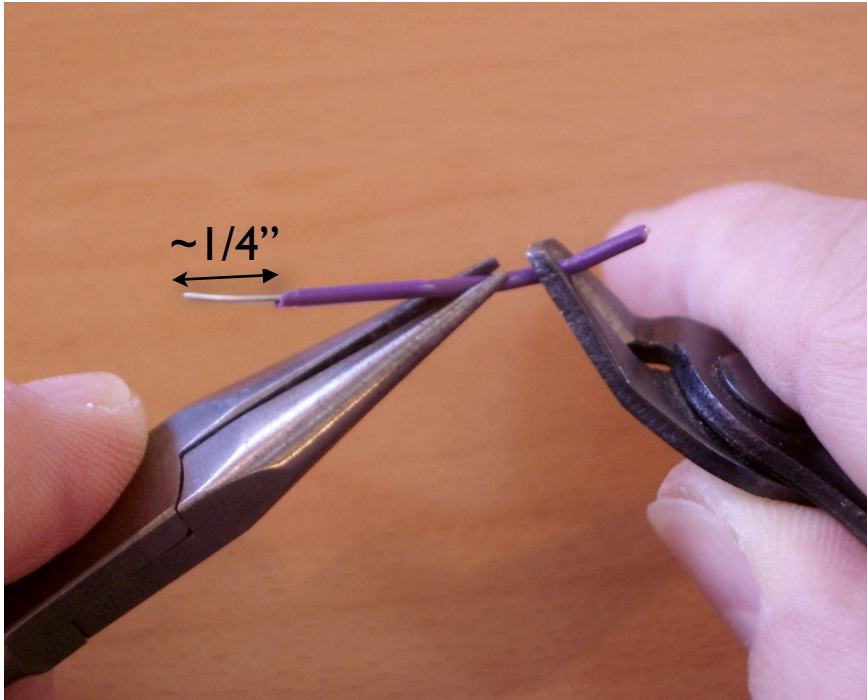
Wire cutters

Needle-nose pliers

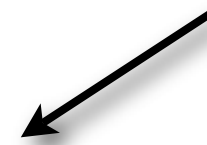
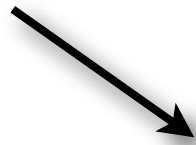
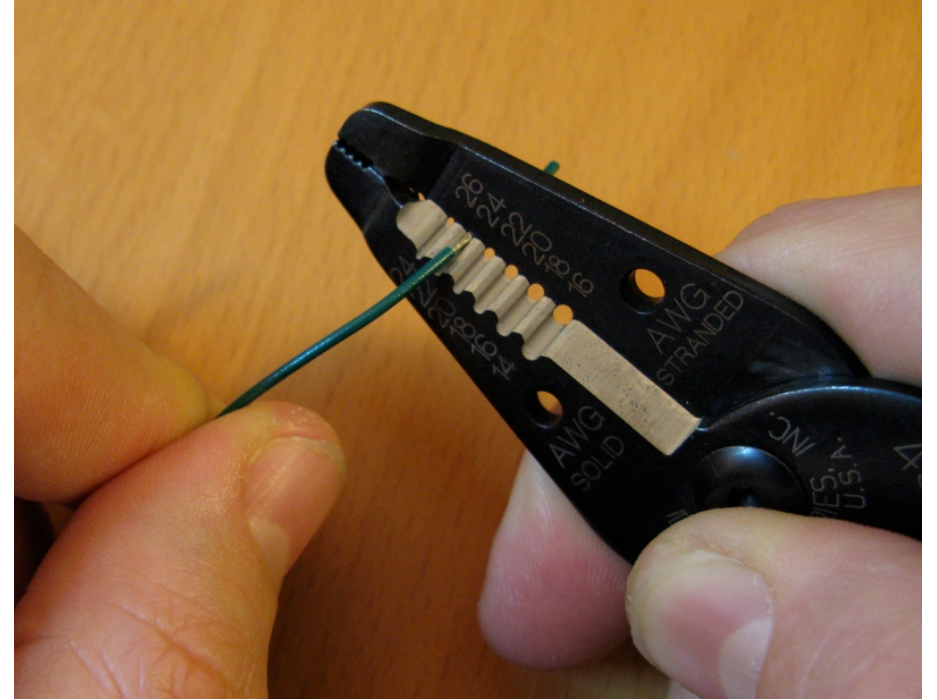
Even with solderless breadboards you'll need to cut and strip wire. Each of these costs around \$5 each. If you have to get just one, get the wire stripper.

Making Jumper Wires

pliers & cutter



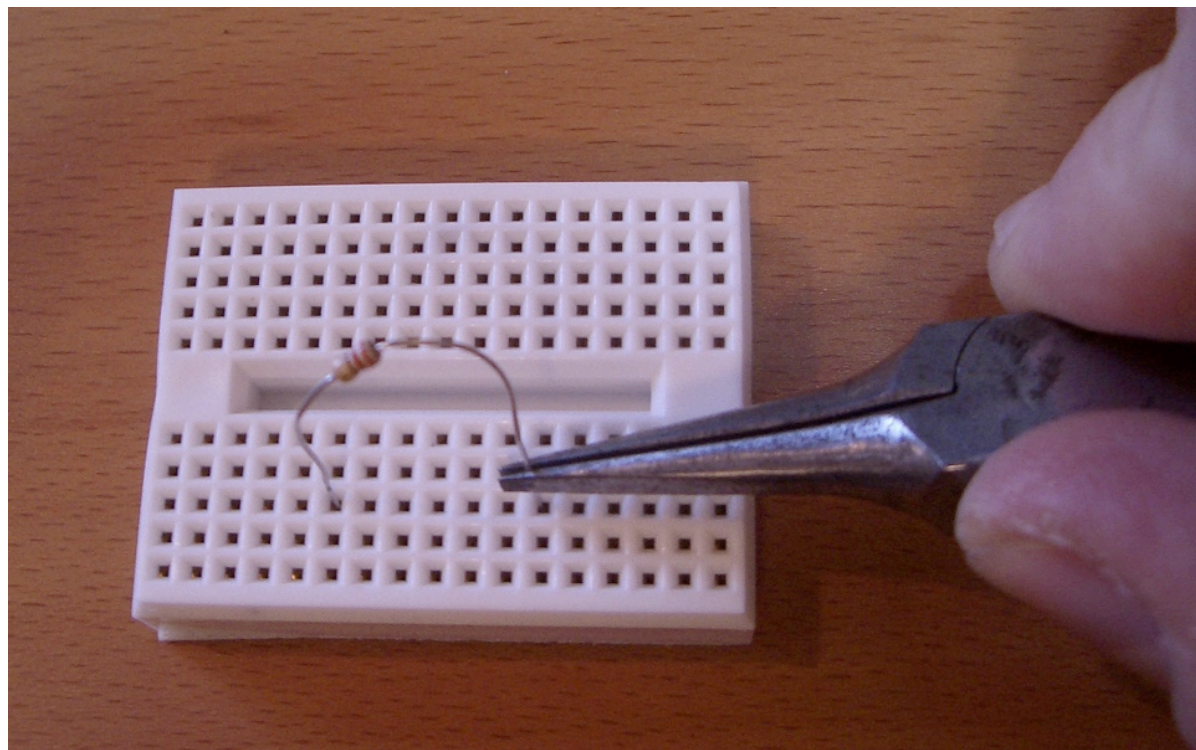
wire stripper



About 1/4'' for the stripped parts.
And as long as you need for your circuit.

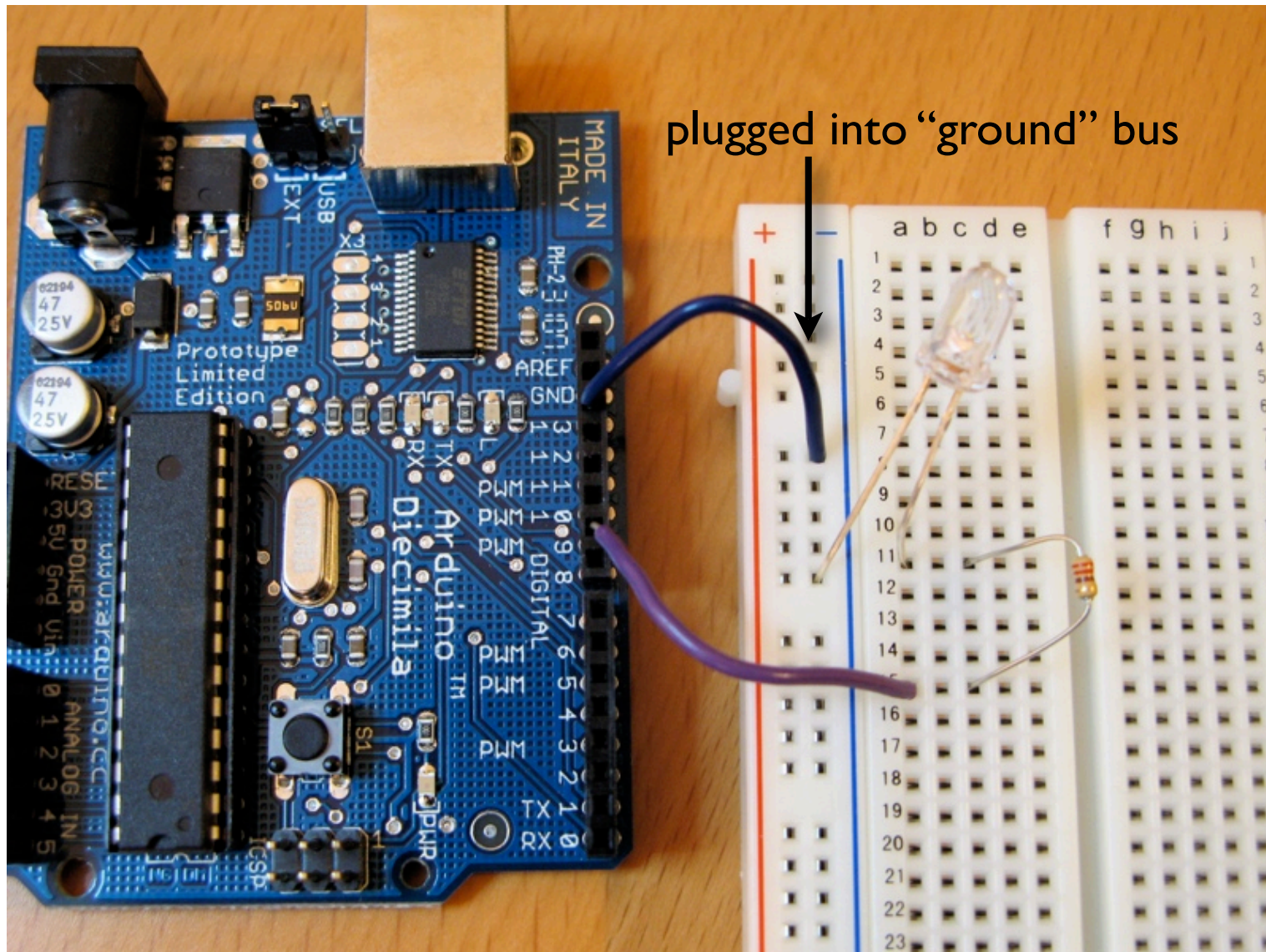
Using Solderless Breadboards

Using needle nose pliers can help push wires & components into holes



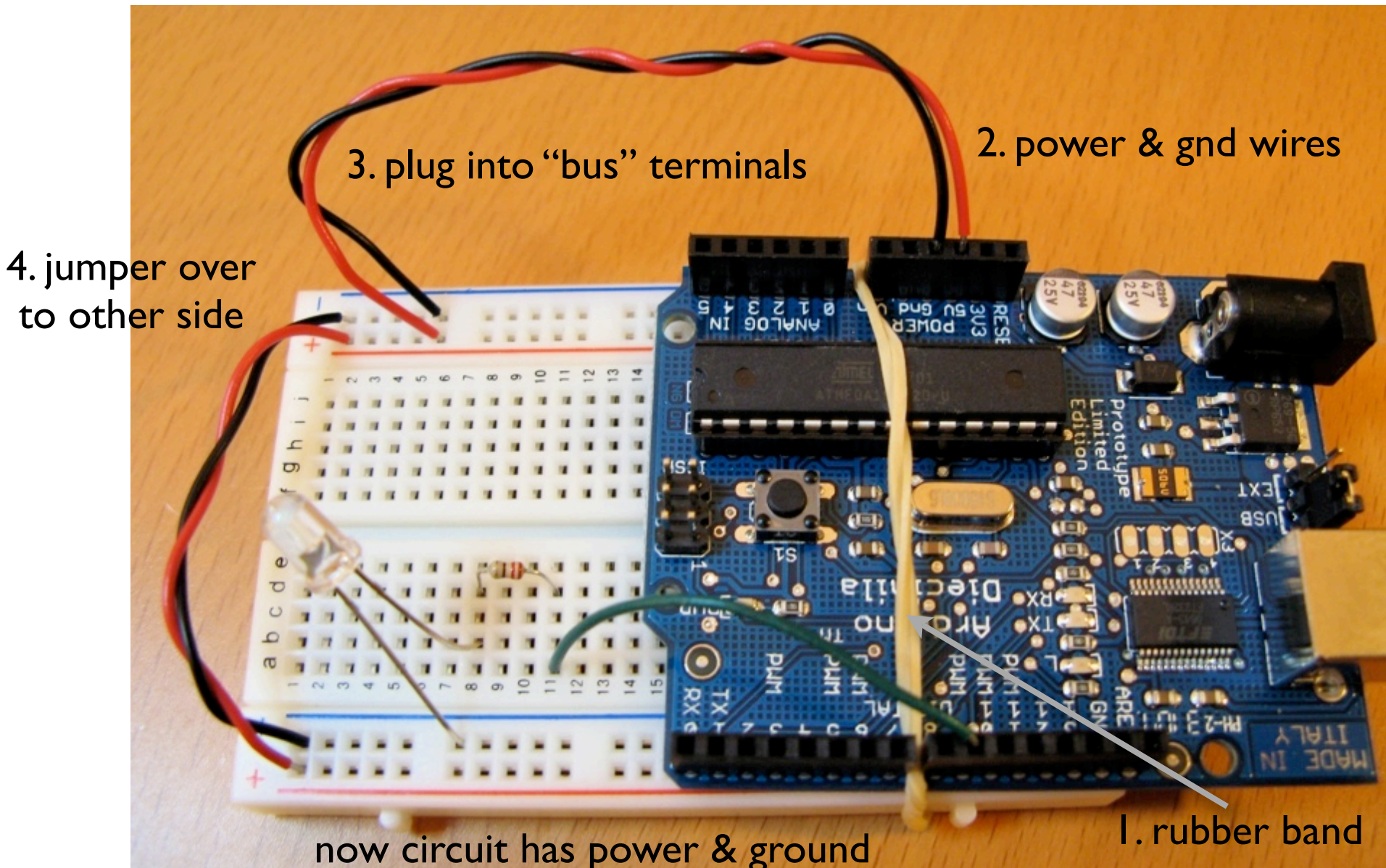
Grab wire or lead toward end and push into hole

All Wired Up



Alternate Way

Or, adding a breadboard to Arduino for I¢



This makes it a bit easier to deal with wiring up circuits for two reasons. First, it secures the breadboard and Arduino together, so wires are less likely to come loose. Secondly, it gives you lots of power and ground holes, which you usually need a lot of.

Use this setup for the rest of your circuits.

Rubber band trick around Arduino & solderless breadboard shameless stolen from Kimiko Ryokai's Tangible User Interface class (INFO290-13): <http://courses.ischool.berkeley.edu/i290-13/f07/>

LED “Fading” Sketch

Load “File/Sketchbook/Examples/Analog/Fading”

note



```
int value = 0; // variable to keep the actu
int ledpin = 9; // light connected to digit

void setup()
{
  // nothing for setup
}

void loop()
{
  for(value = 0 ; value <= 255; value+=5) // fade in (from min to max)
  {
    analogWrite(ledpin, value); // sets the value (range fro
    delay(30); // waits for 30 milli second
  }
  for(value = 255; value >=0; value-=5) // fade out (from max to min)
  {
    analogWrite(ledpin, value);
    delay(30);
  }
}
```

Press “Upload”. After a second, LED will “throb” on and off

Reduce “delay()” values to make it go faster

Try other PWM pins (remember: you have to rewire)

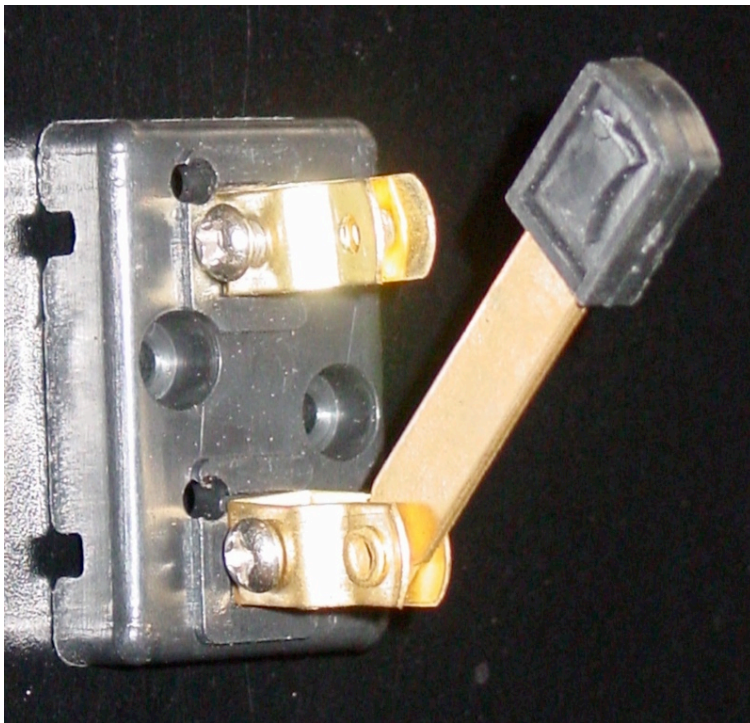
Things to Try With “Fading”

- Make it go really fast or really slow
- Fading from half- to full-bright
- Try other PWM pins
- Multiple fading LEDs, at different rates

Sensors & Inputs

Many sensors are variations on switches

Switches make or break a connection



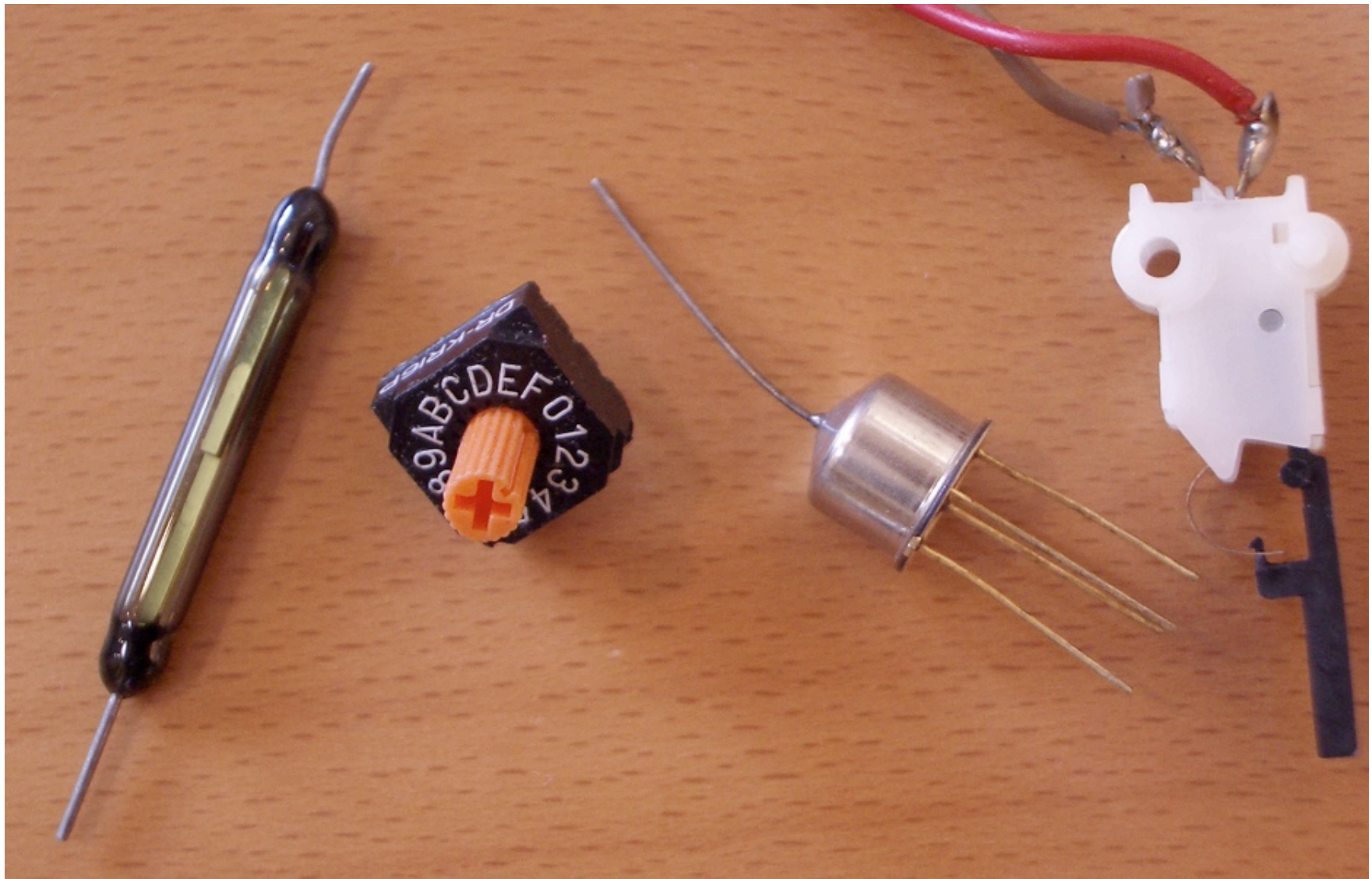
knife switch
(SPST)



toggle switch
(SPDT)

Fundamentally, they're all like the simple knife switch
Single pole = only one circuit is being controlled
Double pole = two circuits are being controlled at once
Single throw = only one path for circuit
Double throw = two potential paths for circuit

Many Kinds of Switches



magnetic

hexadecimal

tilt

lever

Tilt sensor has a little ball inside you can hear.

Used to have mercury switches, with real metallic mercury inside. Not so much now tho'.

Magnetic reed switches are cool, but delicate.

The hex switch is actually many switches in one, and outputs 4 signals

Homemade Switches

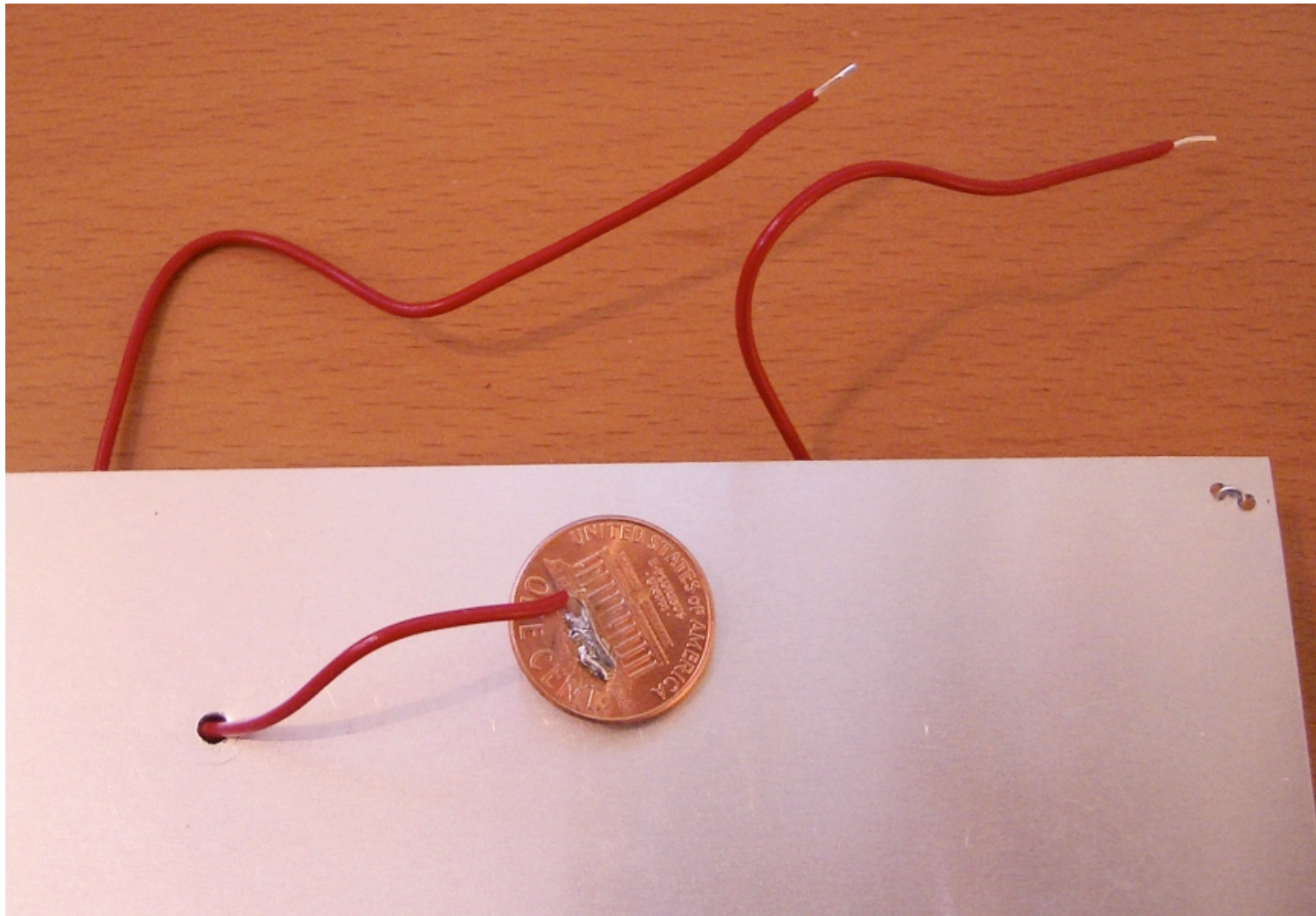
“Trick Penny”

Penny on a surface.
When the penny is lifted, alarms go off



Homemade Switches

“Trick Penny”

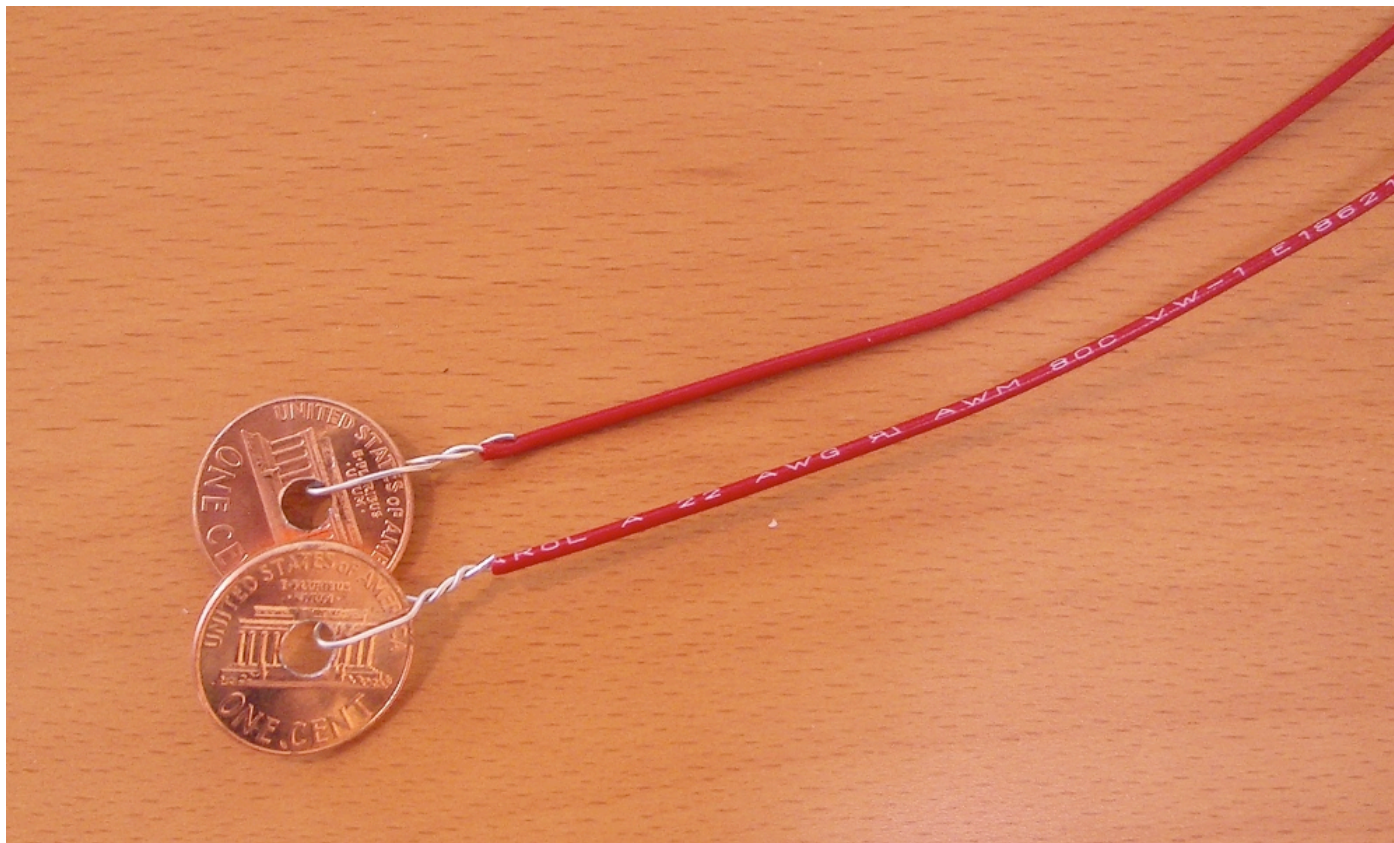


Surface is conductive metal sheet.
Wire soldered to penny.
Wire looped or crimped to metal sheet.

Homemade Switches

“Smart Wind Chimes”

When the wind blows hard enough,
you're sent email

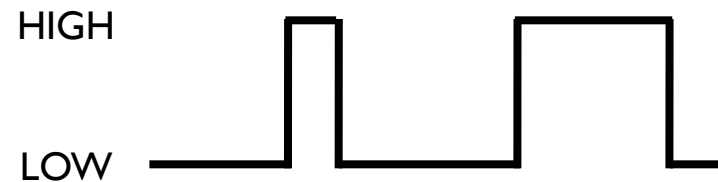


Should use stranded wire, not solid.

Code analyzes series of on/off/on/off pulses to determine wind.

Digital Input

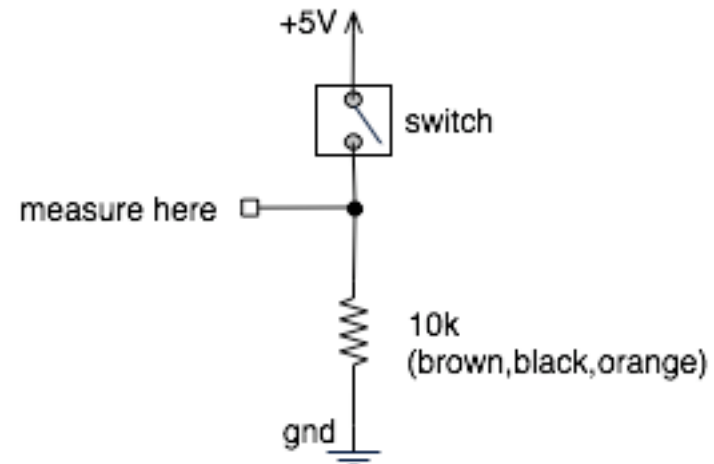
- Switches make or break a connection
- But Arduino wants to see a voltage
 - Specifically, a “HIGH” (5 volts)
 - or a “LOW” (0 volts)



How do you go from make/break to HIGH/LOW?

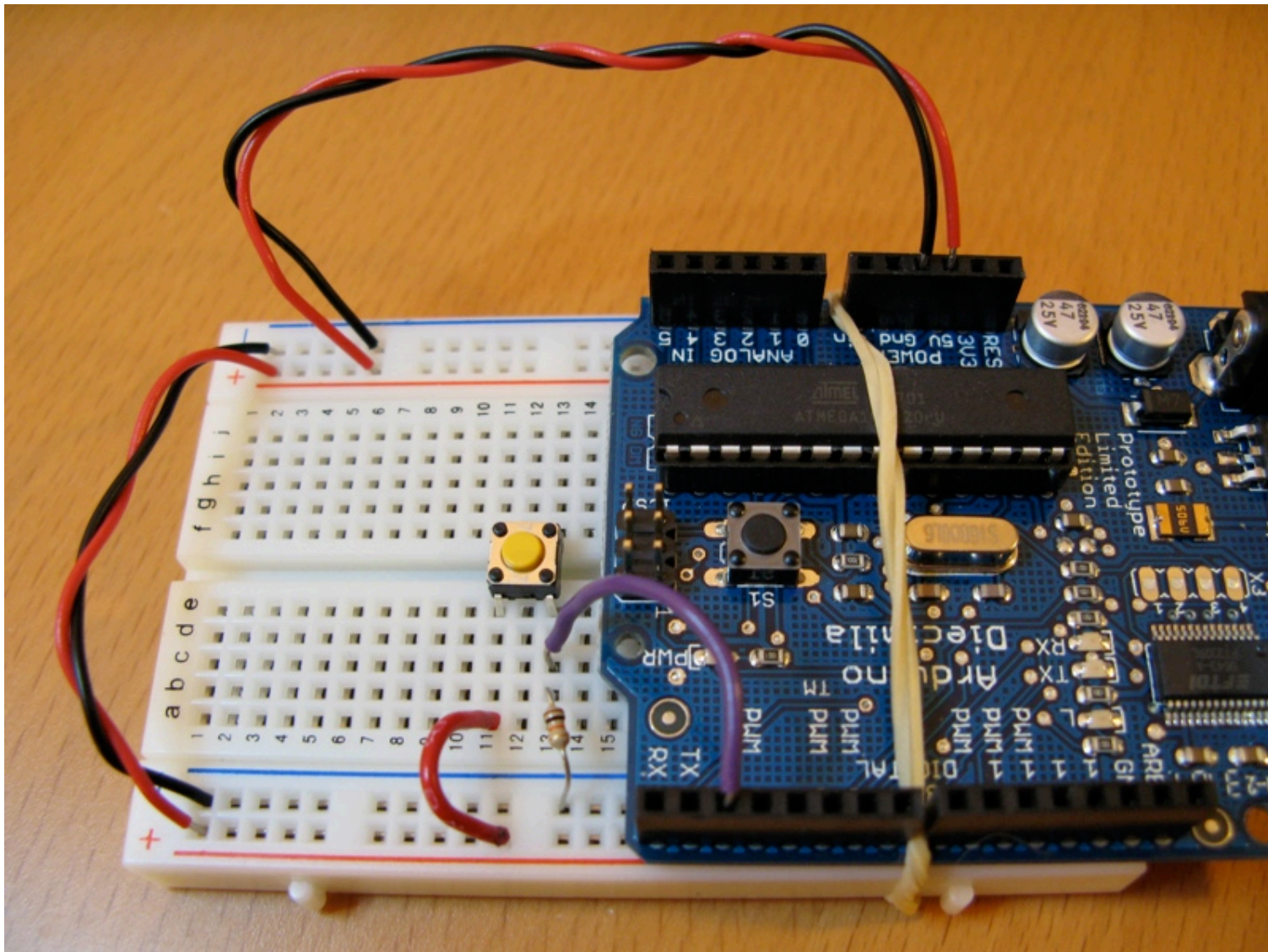
From Switch to HIGH / LOW

- With no connection, digital inputs “float” between 0 & 5 volts (LOW & HIGH)
- Resistor “pulls” input to ground (0 volts)
- Pressing switch “pushes” input to 5 volts
- Press is HIGH
Not pressed is LOW



Don't want “pull-down” to be too small, or it uses a lot of current

Wiring it up



Let's plug it into pin 2

You can leave the last project on the board if you want.

Using digitalRead()

- In `setup()`: `pinMode(myPin, INPUT)` makes a pin an input
- In `loop()`: `digitalRead(myPin)` gets switch's position
 - If doing many tests, use a variable to hold the output value of `digitalRead()`.
 - e.g. `val = digitalRead(myPin)`

Digital Input Sketch

Load “Sketchbook/Examples/Digital/Button”

```
int ledPin = 13;           // choose the pin for the LED
int inputPin = 2;         // choose the input pin (for a pushbutton)
int val = 0;              // variable for reading the pin status

void setup() {
  pinMode(ledPin, OUTPUT); // declare LED as output
  pinMode(inputPin, INPUT); // declare pushbutton as input
}

void loop(){
  val = digitalRead(inputPin); // read input value
  if (val == HIGH) {           // check if the input is HIGH
    digitalWrite(ledPin, LOW); // turn LED OFF
  } else {
    digitalWrite(ledPin, HIGH); // turn LED ON
  }
}
```

Now you control the blinking

(How would you change it to blink the external LED you wired up?)

Press to turn off, release to turn on.
Notice it blinks the LED on-board the Arduino.
Change the code to make it blink the pin 9 LED.

Using Switches to Make Decisions

- Often you'll want to choose between actions, based on how a switch-like sensor
 - E.g. "If person is detected, fire super soaker"
 - E.g. "If flower pot soil is dry, turn on sprinklers"
- Define actions, choose them from sensor inputs
- Let's try that with the actions we currently know

FadeOrBlink

Load “FadeOrBlink” sketch from the handout

Schematic is same as for
“Fading” sketch

Combines “Blink” & “Fading”
sketches into one, selected by
the button

```
int ledPin = 9;           // choose the pin for the LED
int inputPin = 2;        // choose the input pin (for a pushbut
int val = 0;             // variable for reading the pin status
int fadeval = 0;

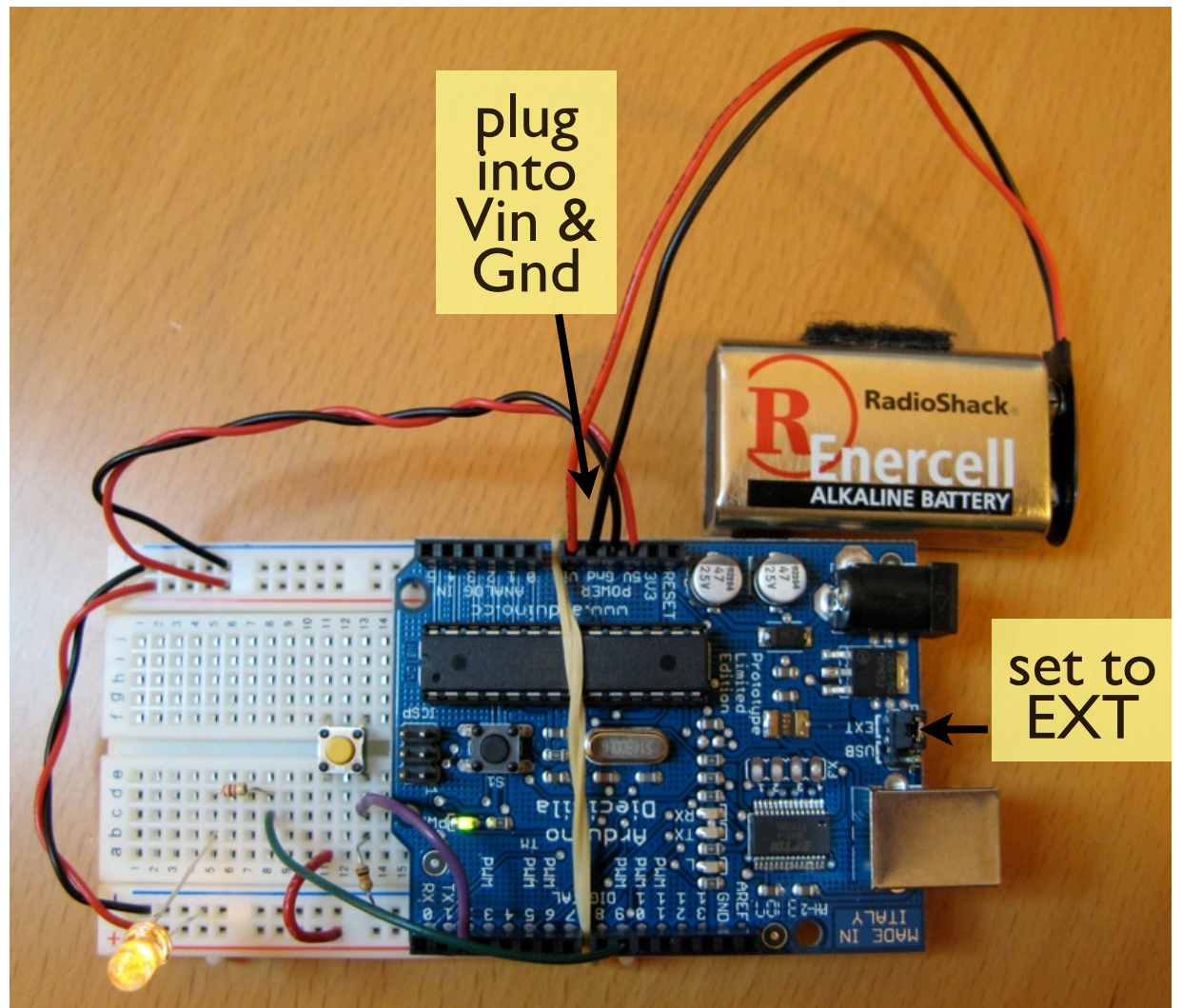
void setup() {
  pinMode(ledPin, OUTPUT); // declare LED as output
  pinMode(inputPin, INPUT); // declare pushbutton as input
}

void loop(){
  val = digitalRead(inputPin); // read input value
  if (val == HIGH) {           // pushed button means do blinking
    digitalWrite(ledPin, LOW); // turn LED OFF
    delay(50);
    digitalWrite(ledPin, HIGH); // turn LED ON
    delay(50);
  }
  else { // else button isn't pressed so do fading
    for(fadeval = 0 ; fadeval <= 255; fadeval+=5) { // fade in (from m
      analogWrite(ledPin, fadeval); // sets the value (range
      delay(10);
    }
    for(fadeval = 255; fadeval >=0; fadeval-=5) { // fade out (from m
      analogWrite(ledPin, fadeval);
      delay(10);
    }
  }
}
```

Battery Power

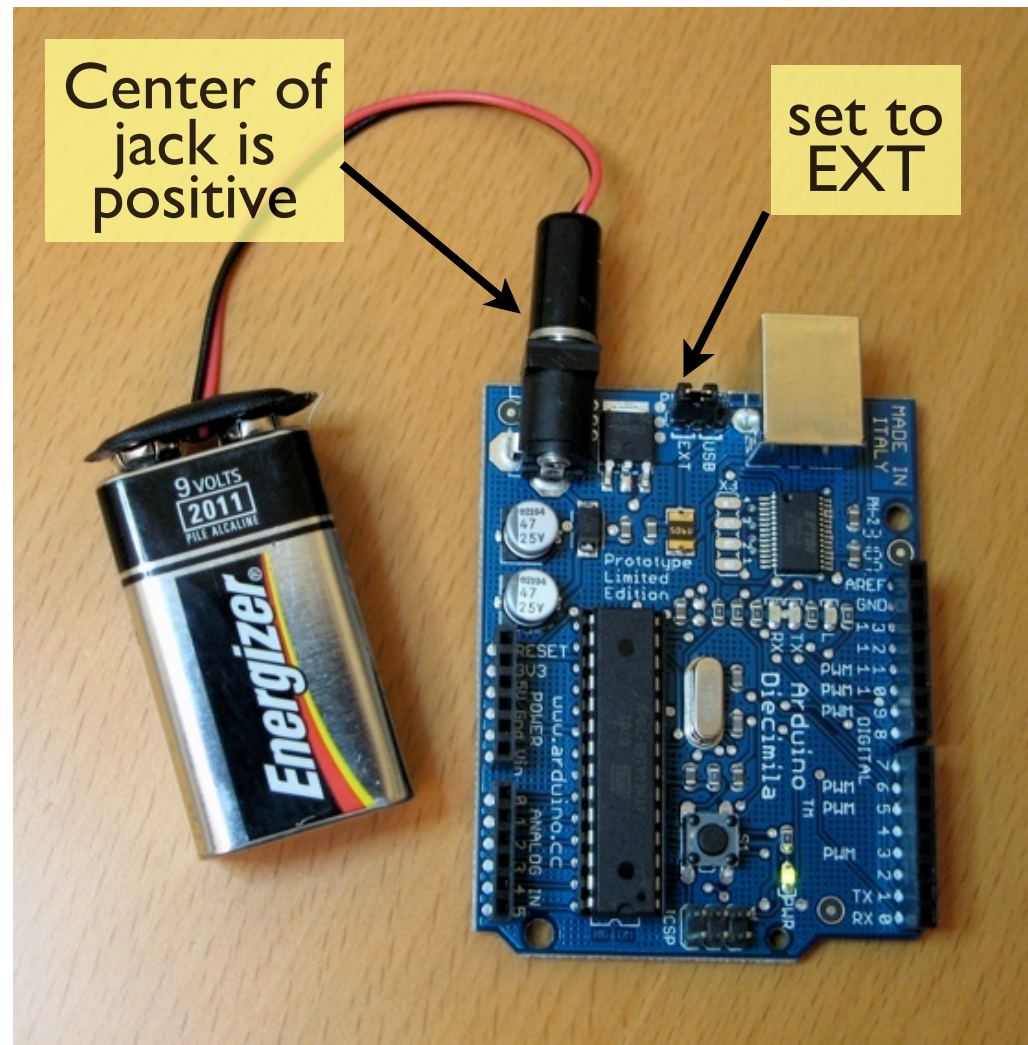
Arduino can work totally stand-alone. It's easy

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



Battery Power

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



Going Further

- Make your own switches: aluminum foil, pennies, etc.
- Build a Knight Rider / Cylon scanning light
- Build a bike light that only works when you peddle
- Make an Annoy-a-Tron™ (blink-blink-blink, wait.... blink-blink-blink)

Lots of things you can do with just LEDs & switches

END Class I

<http://todbot.com/blog/bioniscarduino/>

Tod E. Kurt

tod@todbot.com

Feel free to email me if you have any questions.

Resources

<http://arduino.cc/>

Official homepage. Also check out the Playground & forums

<http://ladyada.net/learn/arduino/>

Great Arduino tutorials

<http://todbot.com/blog/category/arduino/>

Various movies, hacks, tutorials on Arduino

<http://freeduino.org/>

Index of Arduino knowledge

<http://adafruit.com/>

Arduino starter kits, Boarduino Arduino clone, lots of cool kits

<http://sparkfun.com/>

Sells Arduino boards and lots of neat sensors & stuff

Books:

“Physical Computing”, Dan O’Sullivan & Tom Igoe

“Making Things Talk”, Tom Igoe

“Hacking Roomba”, Tod E. Kurt

obligatory book plug