



Lab 120

Introduction to Arduino and Electronics

Class 3

7 July 2009 - AS220 Labs - John Duksta

What's for Today

- Controlling Servos
- About DC motors
- Transistors as switches
- Controlling DC motors
- Introduction to Processing
- Controlling your computer with Arduino

Recap: Blinky LED

Make sure things still work

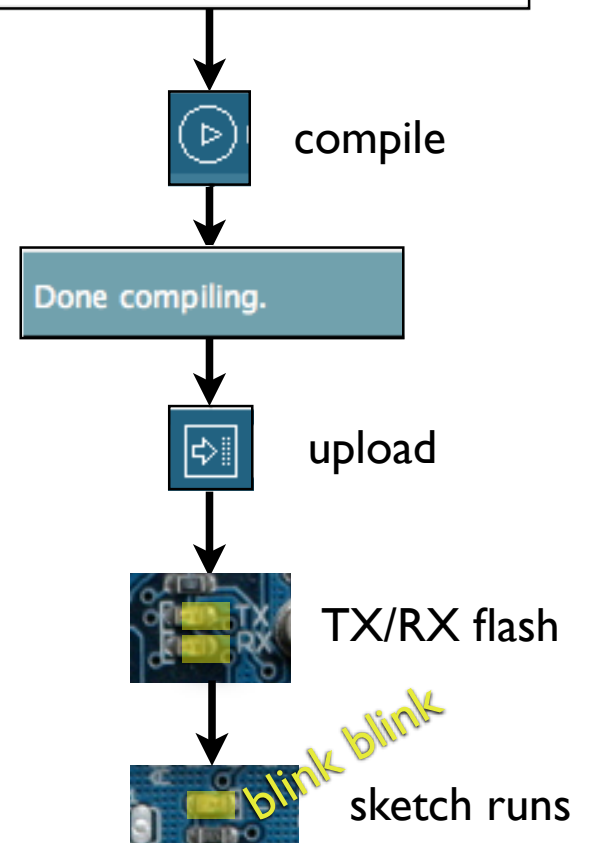
```
int ledPin = 13;           // LED connected to digital pin 13

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
}
```



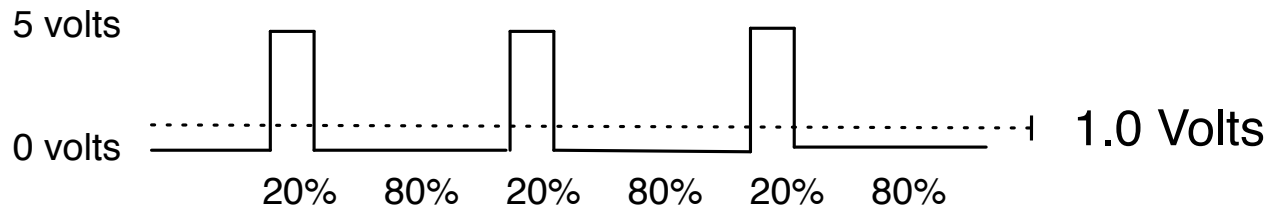
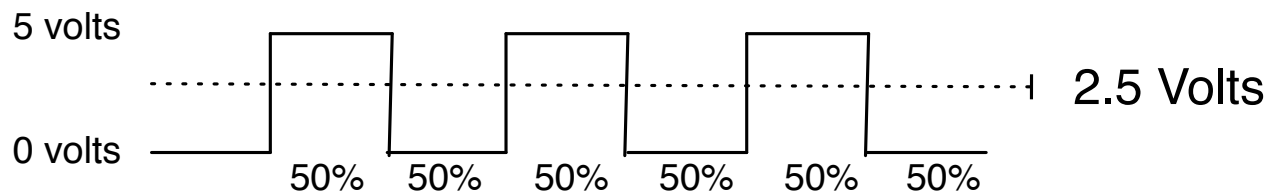
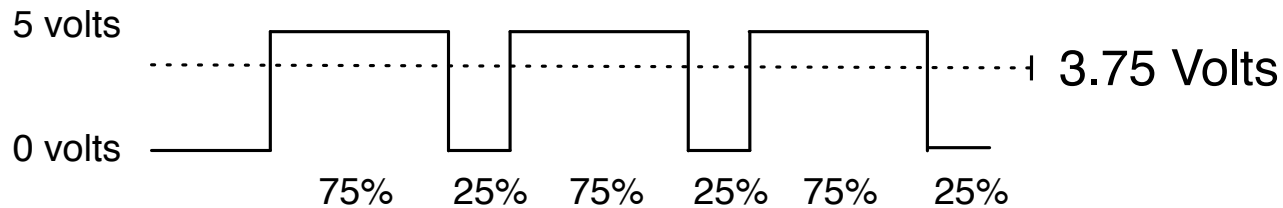
Pulse Width Modulation

- More commonly called “PWM”
- Computers can't output analog voltages
 - Only digital voltages (0 volts or 5 volts)
- But you can fake it
 - if you average a digital signal flipping between two voltages.
- For example...

PWM

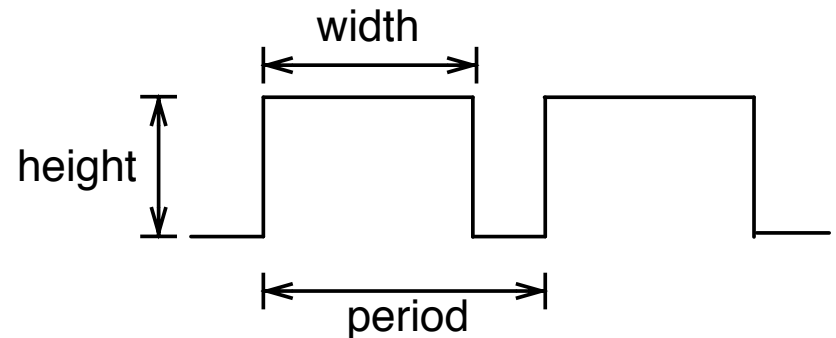
Output voltage is averaged from on vs. off time

$$\text{output_voltage} = (\text{on_time} / \text{off_time}) * \text{max_voltage}$$



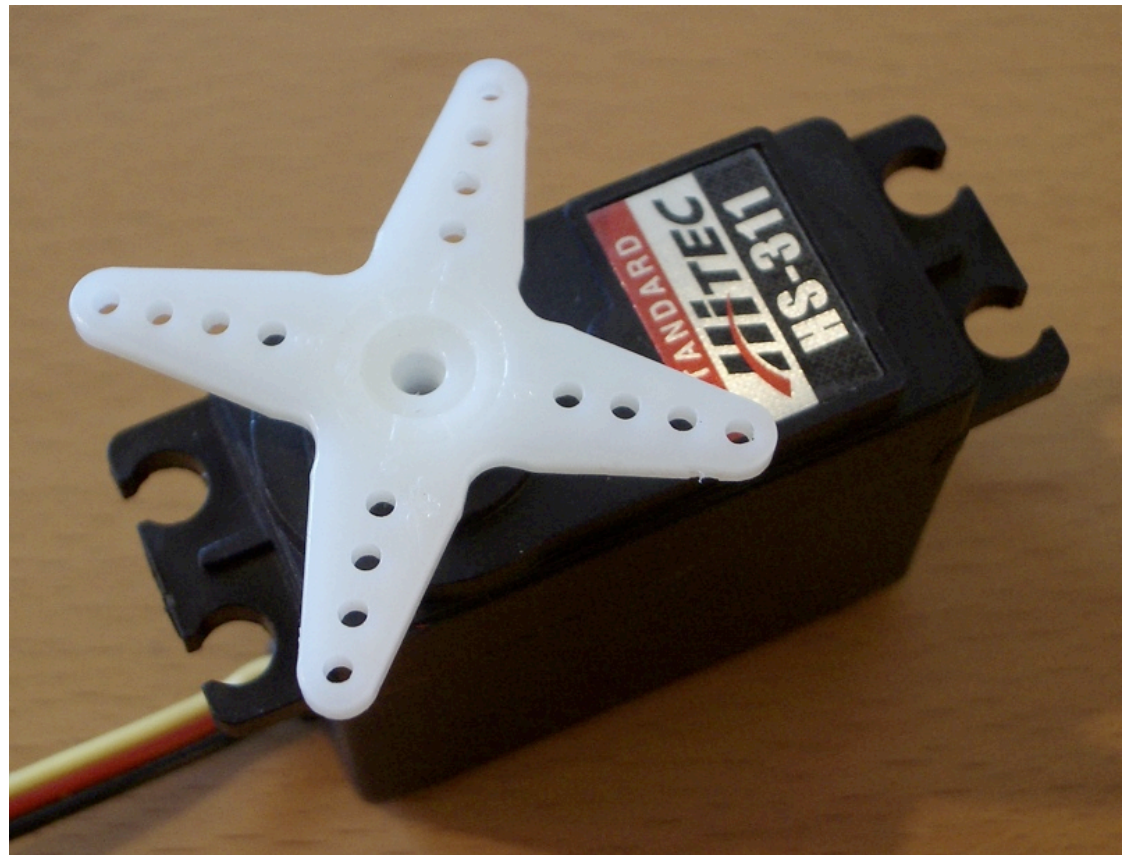
PWM

- Used everywhere
 - Lamp dimmers, motor speed control, power supplies, noise making
- Three characteristics of PWM signals
 - Pulse width range (min/max)
 - Pulse period (= 1/pulses per second)
 - Voltage levels (0-5V, for instance)



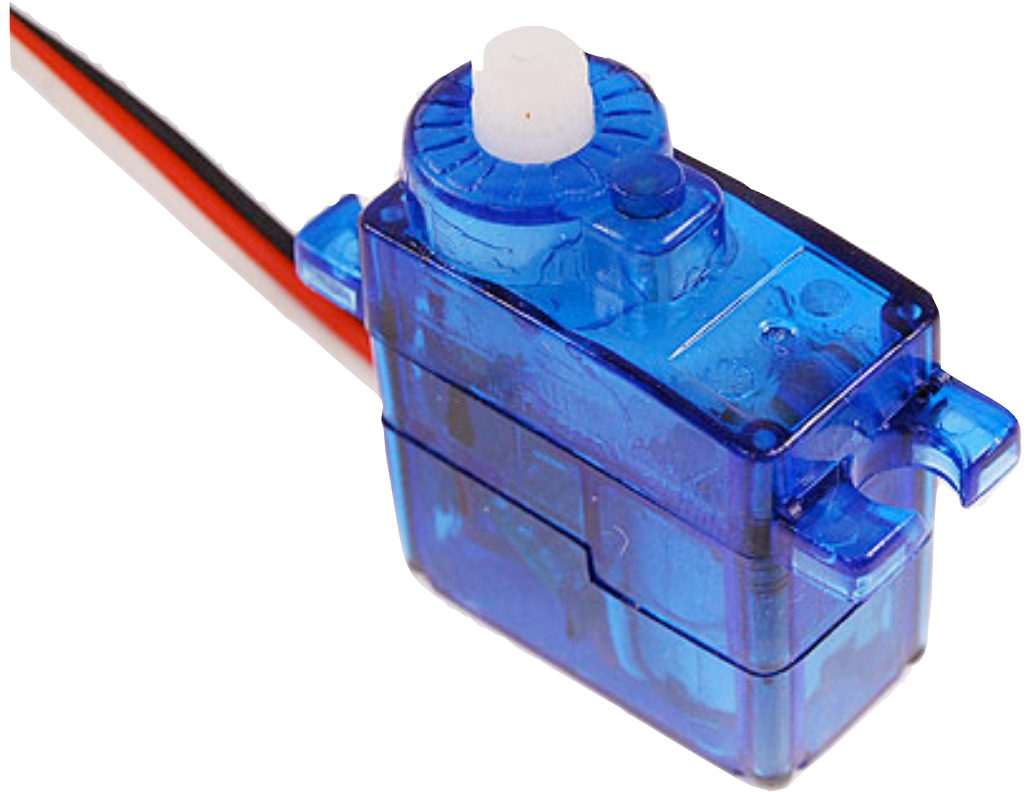
Servomotors

- Can be positioned from 0-180° (usually)
- Internal feedback circuitry & gearing takes care of the hard stuff
- Easy three-wire PWM 5V interface



Servos are Awesome

- DC motor
- High-torque gearing
- Potentiometer to read position
- Feedback circuitry to read pot and control motor
- All built in, you just feed it a PWM signal



Servos, good for what?

- Roboticists, movie effects people, and puppeteers use them extensively
- Any time you need controlled, repeatable motion
- Can turn rotation into linear movement with clever mechanical levers

Servos

- Come in all sizes
 - from super-tiny
 - to drive-your-car
- But all have the same 3-wire interface
- Servos are spec'd by:

weight: 9g
speed: .12s/60deg @ 6V
torque: 22oz/1.5kg @ 6V
voltage: 4.6~6V
size: 21x11x28 mm

9g

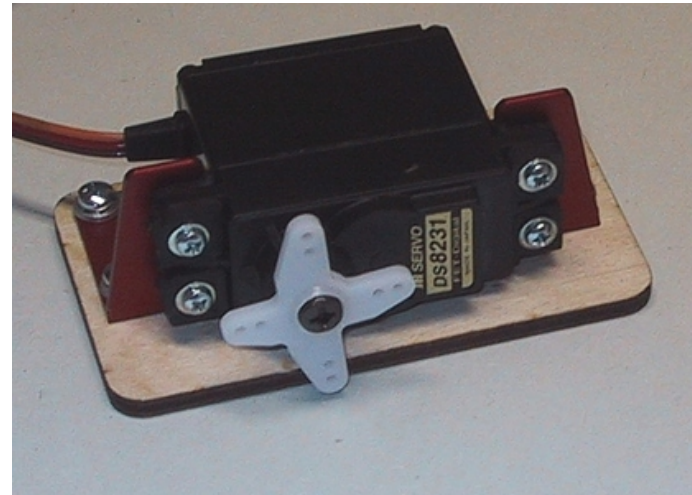


157g

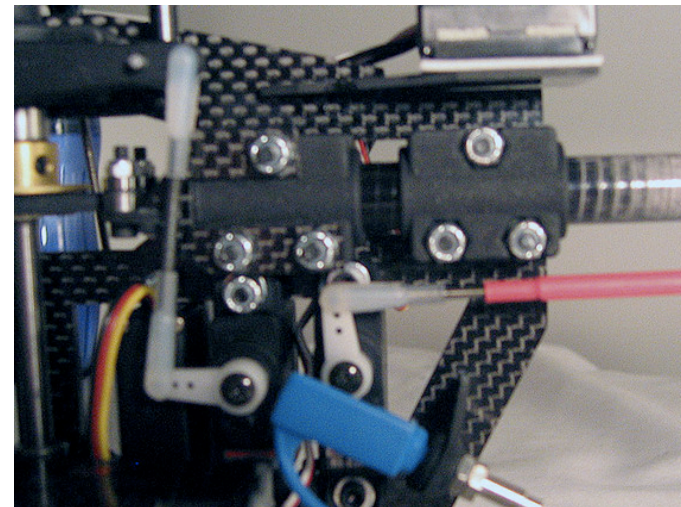


Servo Mounts & Linkages

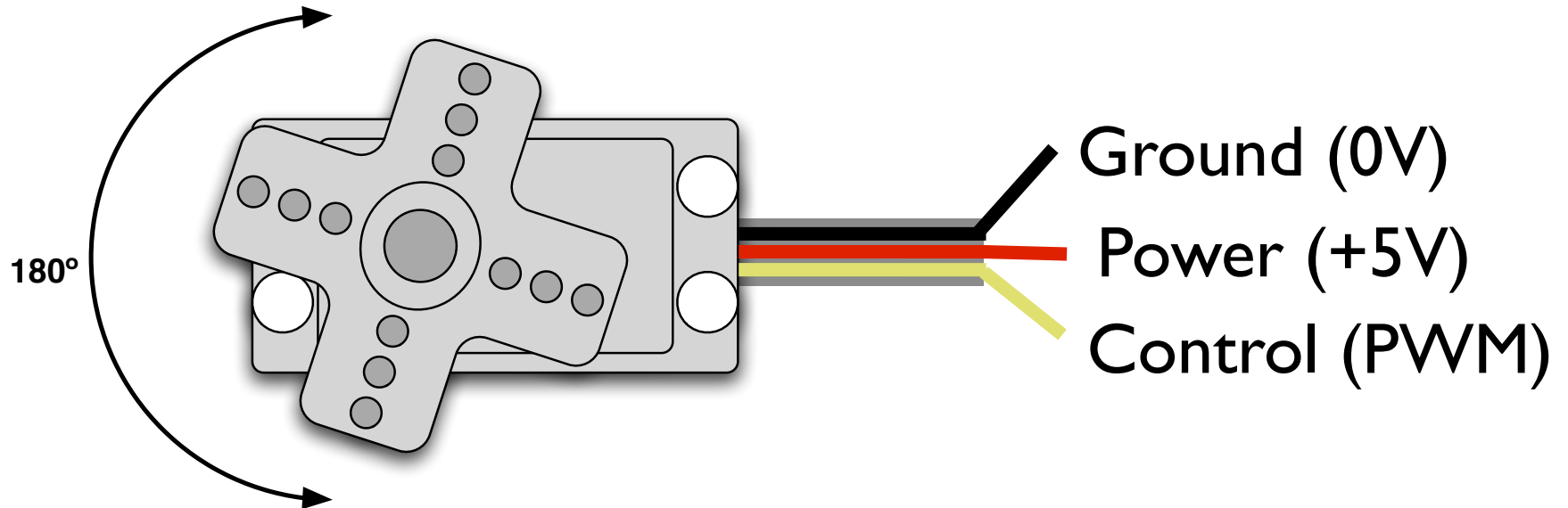
Lots of ways to mount a servo



And turn its rotational motion into other types of motion



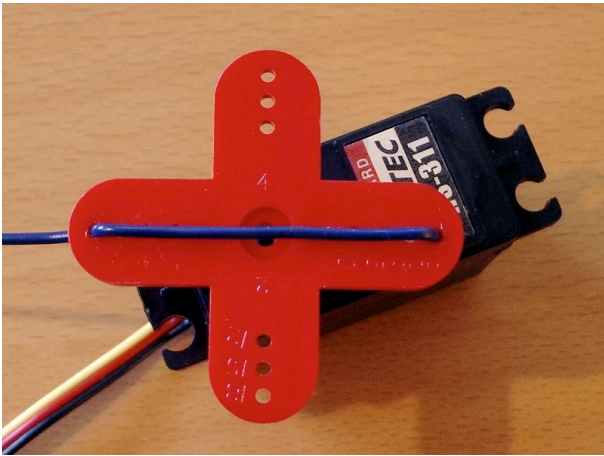
Servo Control



- PWM freq is 50 Hz (i.e. every 20 milliseconds)
- Pulse width ranges from 1 to 2 milliseconds
 - 1 millisecond = full anti-clockwise position
 - 2 millisecond = full clockwise position

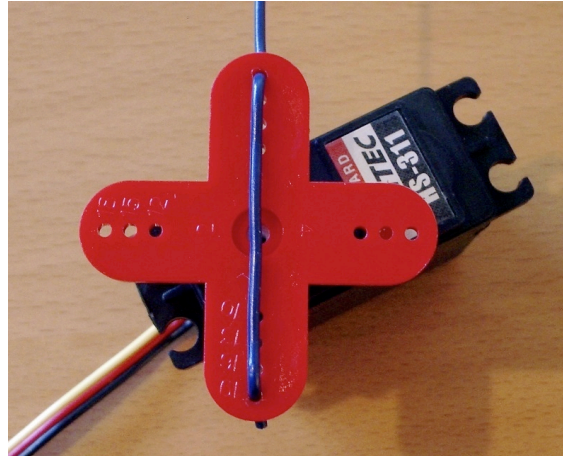
Servo Movement

0 degrees



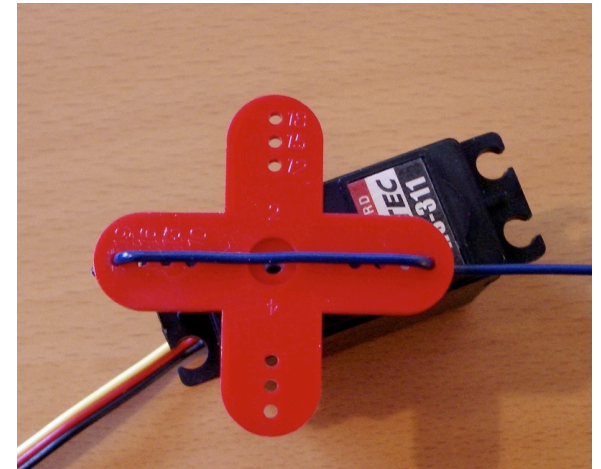
1000 microseconds

90 degrees



1500 microseconds

180 degrees



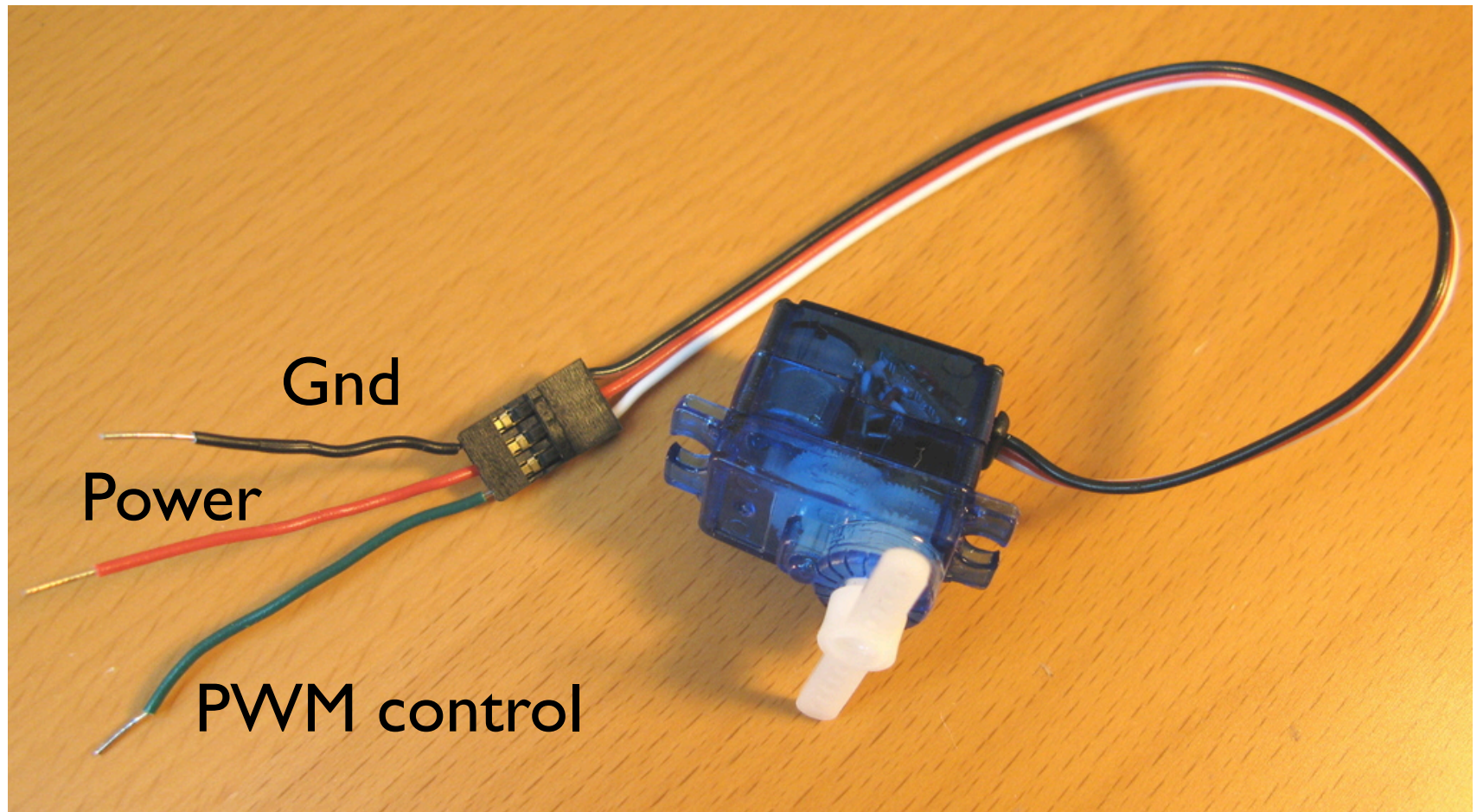
2000 microseconds

In practice, pulse range can range from 500 to 2500 microseconds

(and go ahead and add a wire marker to your servo like the above)

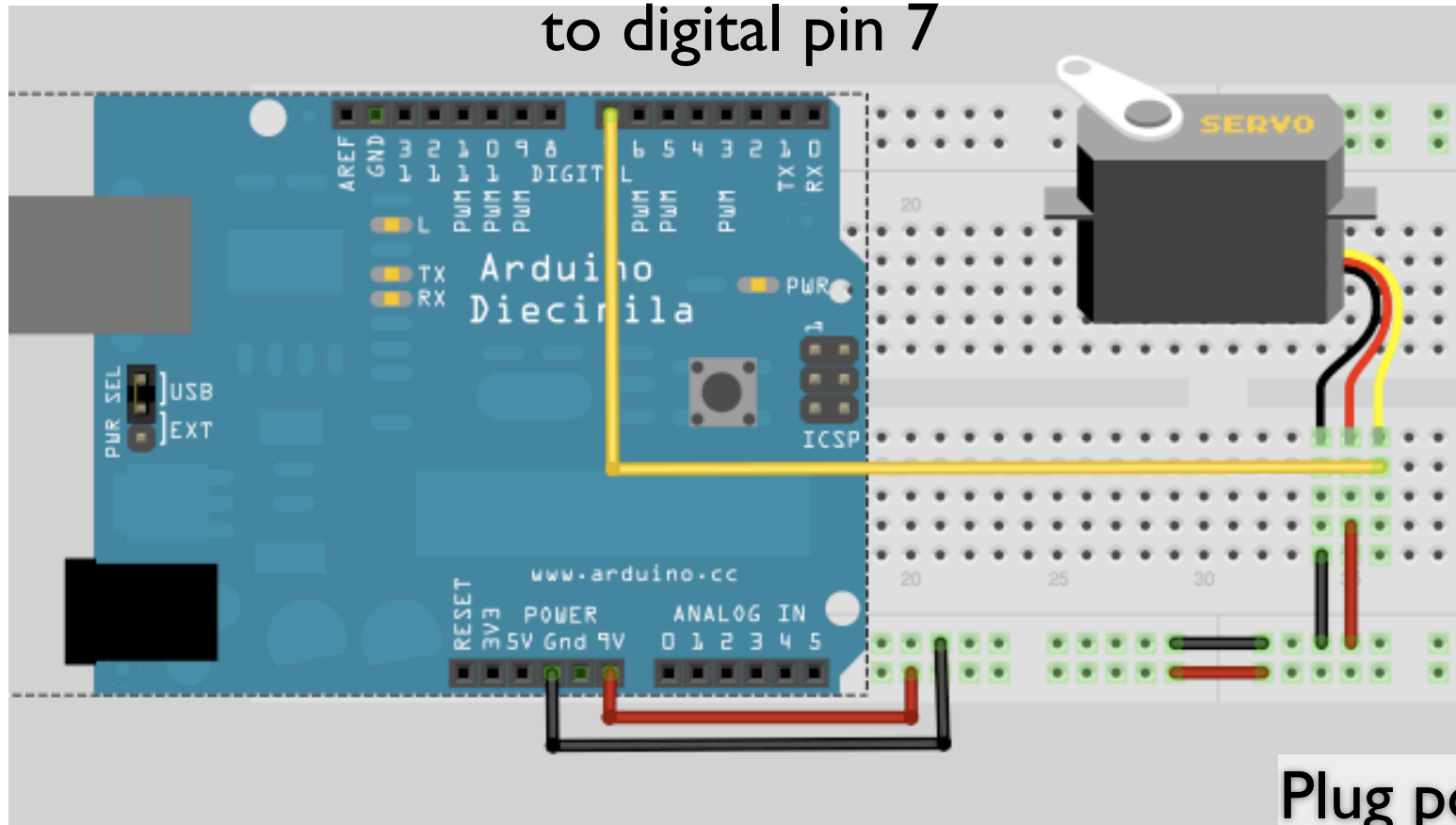
Servo and Arduino

First, add some jumper wires to the servo connector



Servo and Arduino

Plug control wire
to digital pin 7



Plug power
wires in

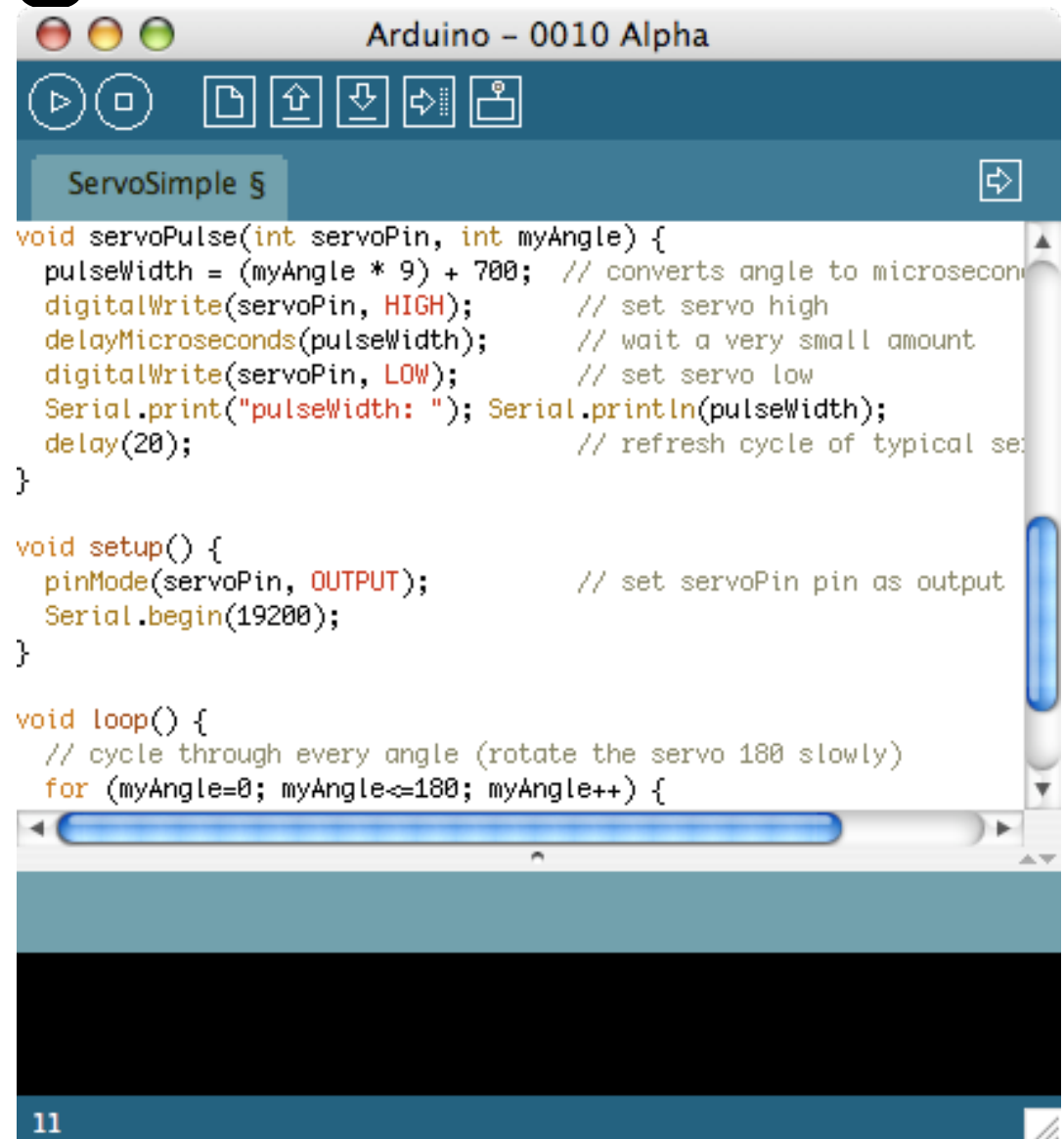
Moving a Servo

“ServoSimple”

Move the servo across
its range of motion

Uses `delayMicroseconds()` for pulse width

Uses `delay()` for pulse frequency



```
void servoPulse(int servoPin, int myAngle) {
  pulseWidth = (myAngle * 9) + 700; // converts angle to microseconds
  digitalWrite(servoPin, HIGH);      // set servo high
  delayMicroseconds(pulseWidth);     // wait a very small amount
  digitalWrite(servoPin, LOW);       // set servo low
  Serial.print("pulseWidth: "); Serial.println(pulseWidth);
  delay(20);                          // refresh cycle of typical servo
}

void setup() {
  pinMode(servoPin, OUTPUT);          // set servoPin pin as output
  Serial.begin(19200);
}

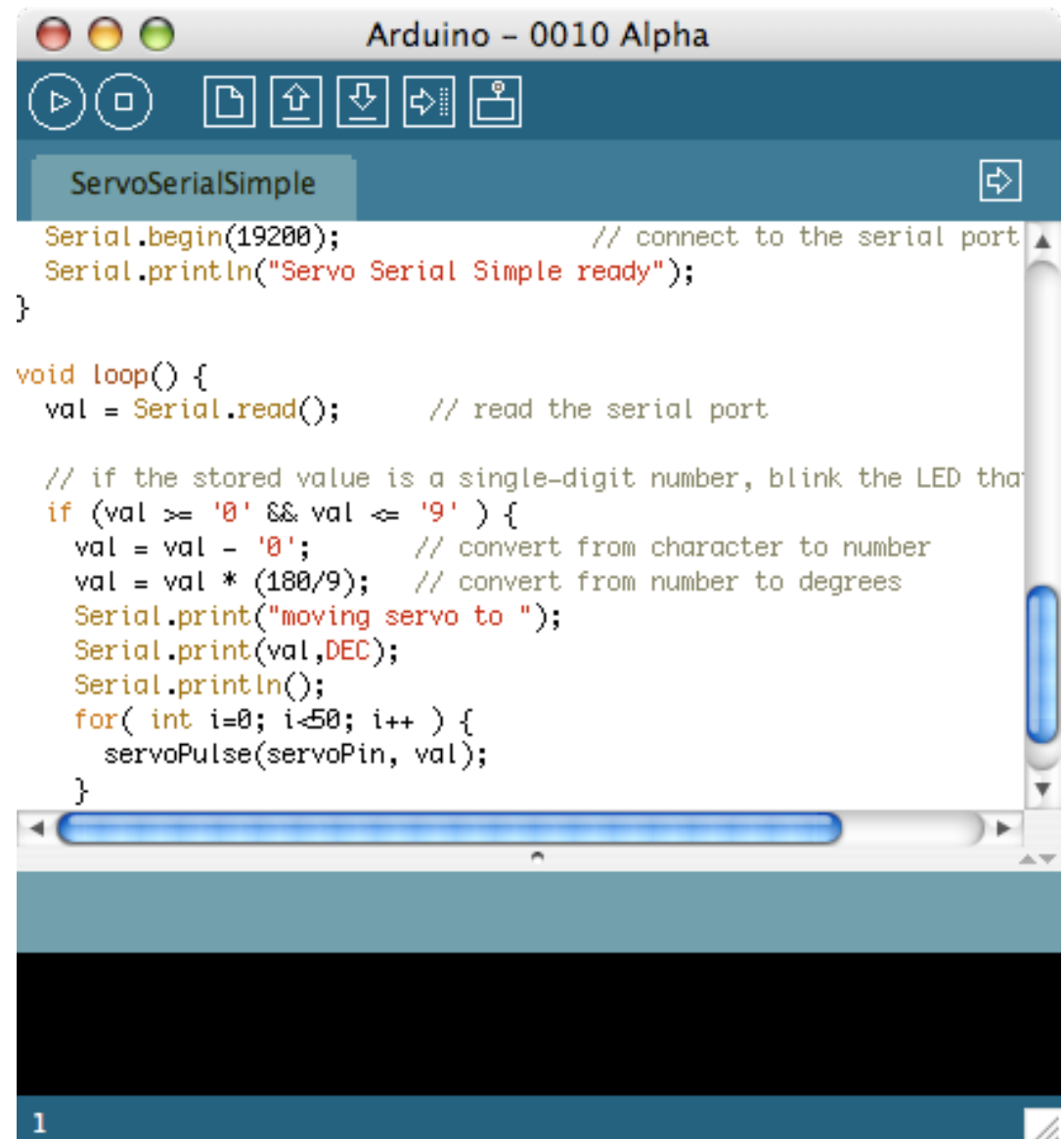
void loop() {
  // cycle through every angle (rotate the servo 180 slowly)
  for (myAngle=0; myAngle<=180; myAngle++) {
```


Serial-controlled Servo

“ServoSerialSimple”

Drive the servo
by pressing
number keys

Takes the last servo
example and adds our
standard serial input to it.

A screenshot of the Arduino IDE window titled "Arduino - 0010 Alpha". The window shows a sketch named "ServoSerialSimple". The code is as follows:

```
Serial.begin(19200);           // connect to the serial port
Serial.println("Servo Serial Simple ready");
}

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

  // if the stored value is a single-digit number, blink the LED that
  if (val >= '0' && val <= '9') {
    val = val - '0';           // convert from character to number
    val = val * (180/9);       // convert from number to degrees
    Serial.print("moving servo to ");
    Serial.print(val,DEC);
    Serial.println();
    for( int i=0; i<50; i++ ) {
      servoPulse(servoPin, val);
    }
  }
}
```

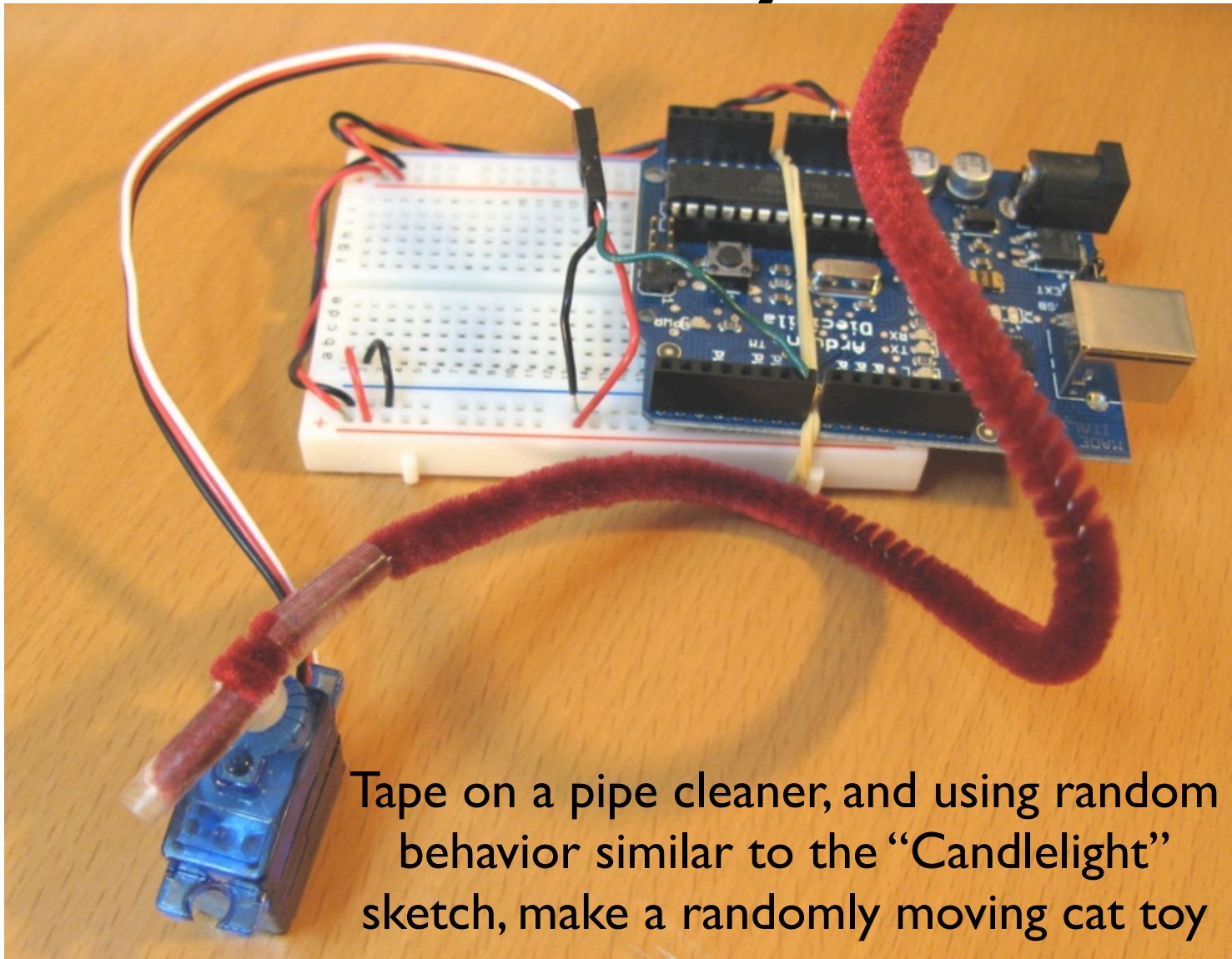
The IDE interface includes a toolbar with icons for running, stopping, saving, and uploading. The bottom status bar shows the line number "1".

Aside: Controlling Arduino

- Any program on the computer, not just the Arduino software, can control the Arduino board
- On Unixes like Mac OS X & Linux, even the command-line can do it:

```
demo% export PORT=/dev/tty.usbserial-A3000Xv0
demo% stty -f $PORT 9600 raw -parenb -parodd cs8 -hupcl -cstopb clocal
demo% printf "1" > $PORT # rotate servo left
demo% printf "5" > $PORT # go to middle
demo% printf "9" > $PORT # rotate servo right
```

Robo Cat Toy Idea



Tape on a pipe cleaner, and using random behavior similar to the “Candlelight” sketch, make a randomly moving cat toy

Servo Timing Problems

- Two problems with the last sketch
 - When `servoPulse()` function runs, nothing else can happen
 - Servo isn't given periodic pulses to keep it at position
- You need to run two different "tasks":
 - one to read the serial port
 - one to drive the servo

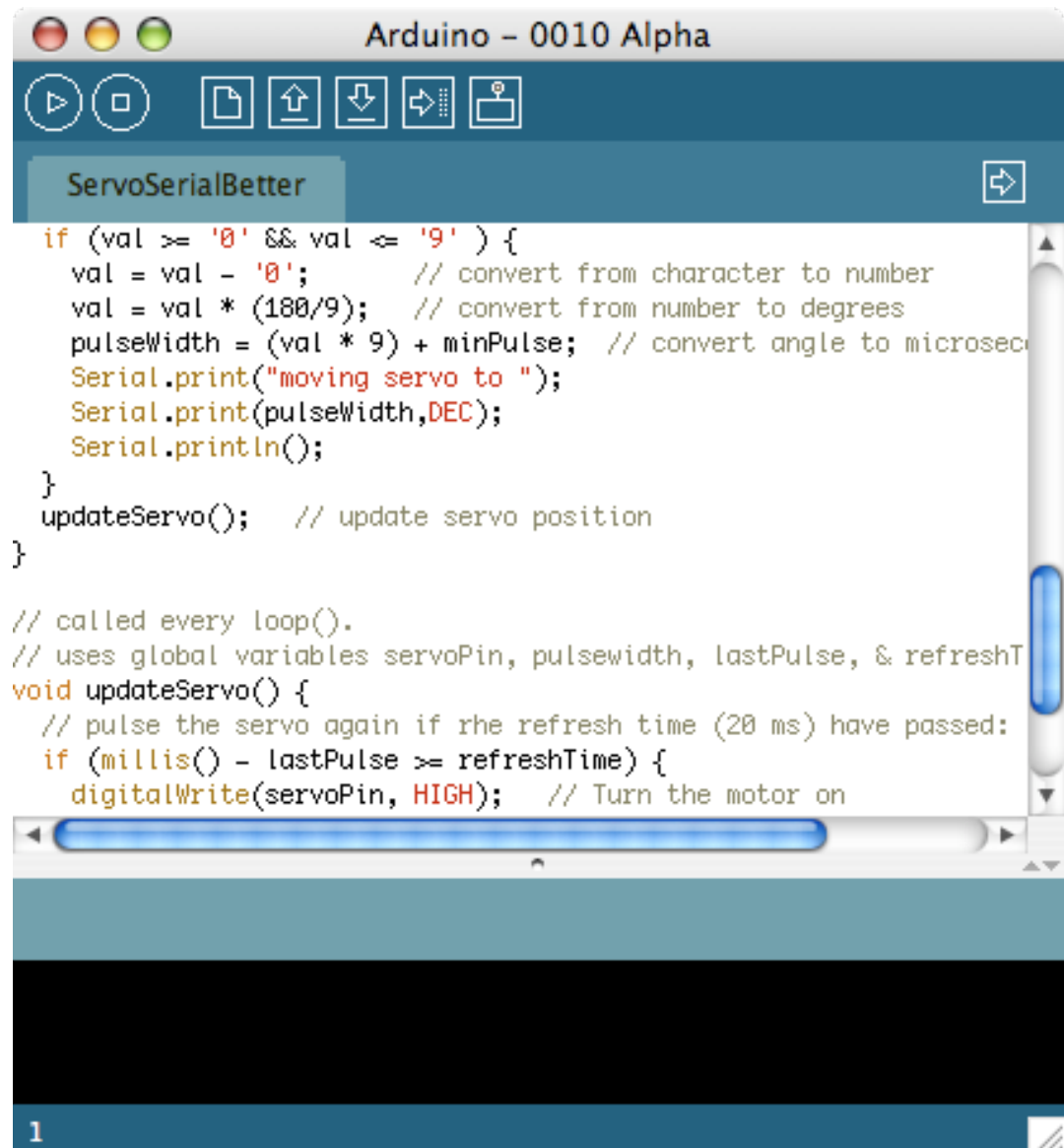
Better Serial Servo

“ServoSerialBetter”

Works just like
ServoSerialSimple
(but better)

Update the servo when
needed, not just when
called at the right time

Uses “millis()” to
know what time it is

A screenshot of the Arduino IDE window titled "Arduino - 0010 Alpha". The code editor shows the implementation of the "ServoSerialBetter" library. The code includes a function to parse a character into a degree value and calculate the pulse width, and a loop function that updates the servo position based on a refresh time interval using the millis() function. The code is as follows:

```
if (val >= '0' && val <= '9') {
  val = val - '0'; // convert from character to number
  val = val * (180/9); // convert from number to degrees
  pulseWidth = (val * 9) + minPulse; // convert angle to microseconds
  Serial.print("moving servo to ");
  Serial.print(pulseWidth,DEC);
  Serial.println();
}
updateServo(); // update servo position
}

// called every loop().
// uses global variables servoPin, pulsewidth, lastPulse, & refreshTime
void updateServo() {
  // pulse the servo again if the refresh time (20 ms) have passed:
  if (millis() - lastPulse >= refreshTime) {
    digitalWrite(servoPin, HIGH); // Turn the motor on
  }
}
```

Multiple Servos

- The `updateServo ()` technique can be extended to many servos
- Only limit really is number of digital output pins you have
- It starts getting tricky after about 8 servos though

Multiple “Tasks”

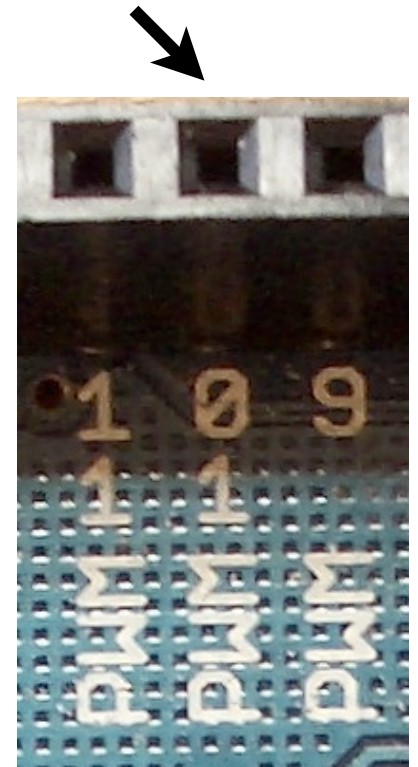
The concept inside `updateServo()` is useful anytime you need to do multiple “things at once” in an Arduino sketch:

- Define your task
- Break it up into multiple time-based chunks (“task slices”)
- Put those task slices in a function
- Use `millis()` to determine when a slice should run
- Call the functions from `loop()`

Arduino PWM

why all the software, doesn't Arduino have PWM?

- Arduino has built-in PWM
- On pins 9, 10, 11
- Use `analogWrite(pin, value)`
- It operates at a high, fixed frequency (thus not usable for servos)
- But great for LEDs and motors
- Uses built-in PWM circuits of the ATmega8 chip -» no software needed

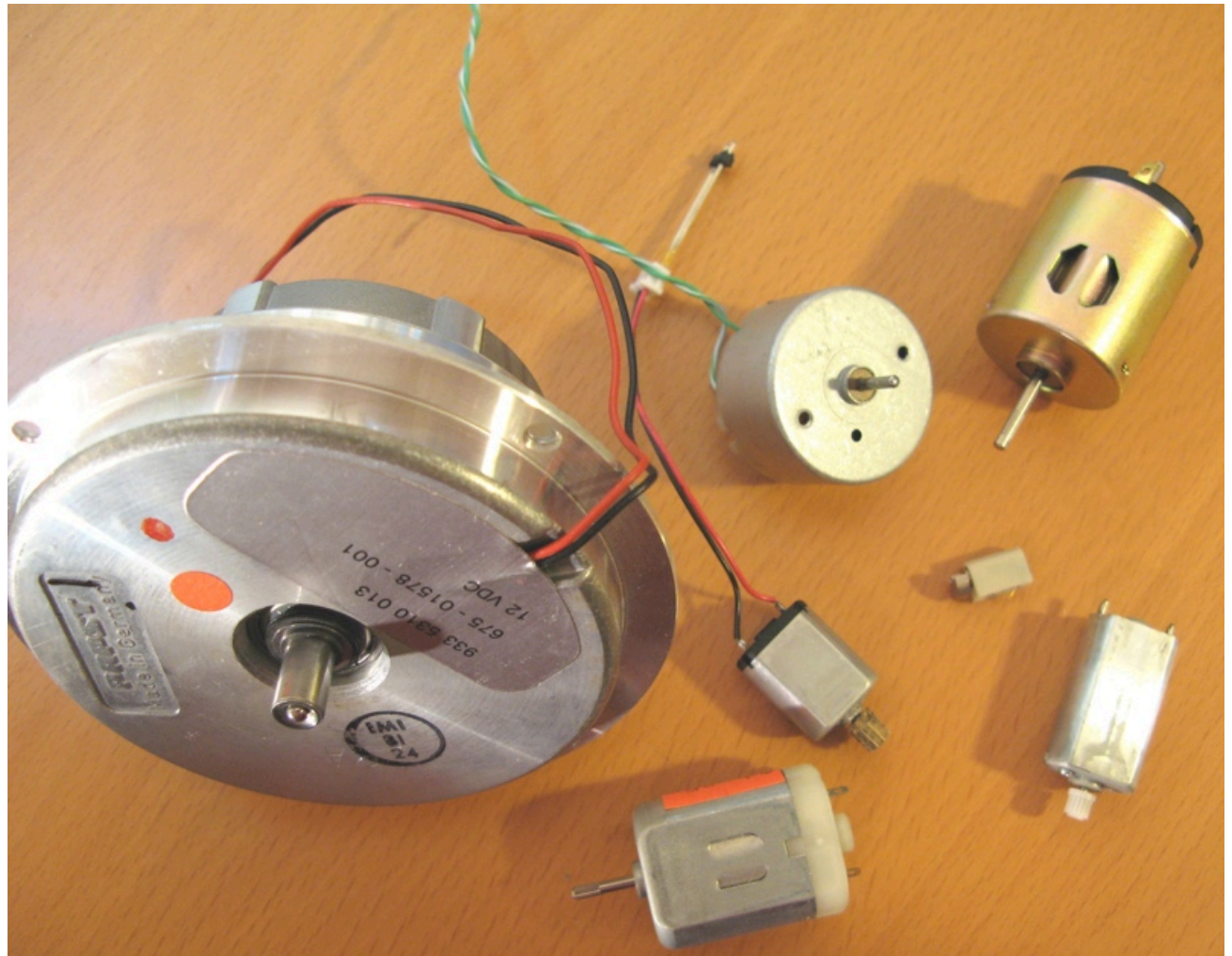


DC Motors

come in all
shapes and sizes

You probably have
3-4 on you right now

(cell vibrate, laptop fan, laptop dvd drive)



DC Motors

A dizzying array of parameters specify a motor

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

The motor you have runs operates on 6-18 volts. It's high torque and runs at 8500 RPM at 12V and pulls 55mA @ 12V with no load

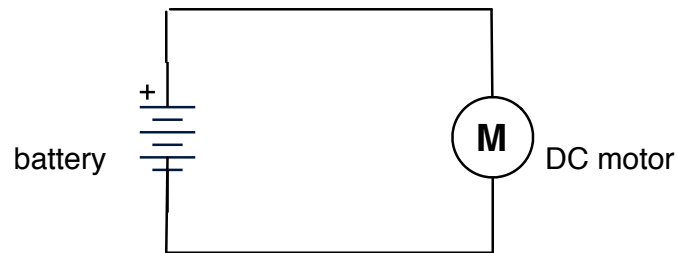
DC Motors

Characteristics

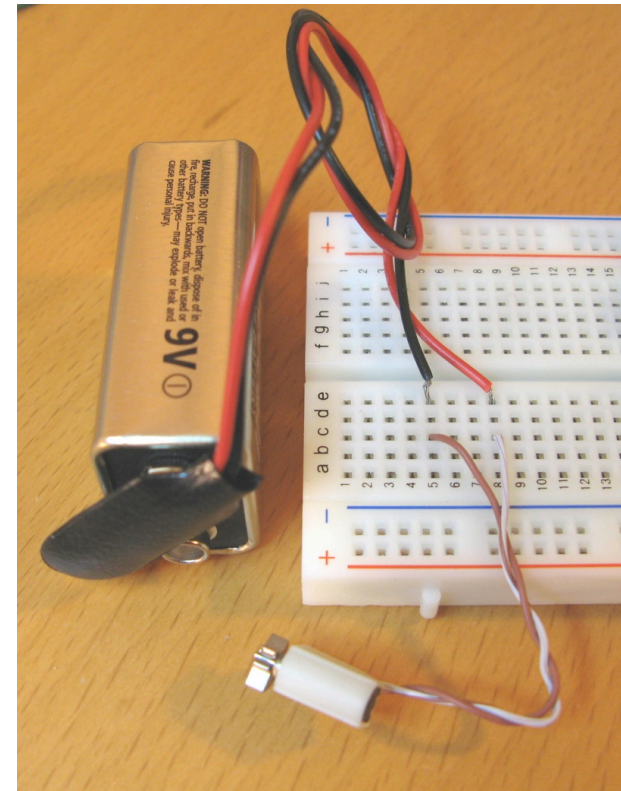
- When the first start up, they draw a *lot* more current, up to 10x more.
- If you “stall” them (make it so they can’t turn), they also draw a lot of current
- They can operate in either direction, by switching voltage polarity
- Usually spin very fast: > 1000 RPM
- To get slower spinning, need gearing

DC Motors

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



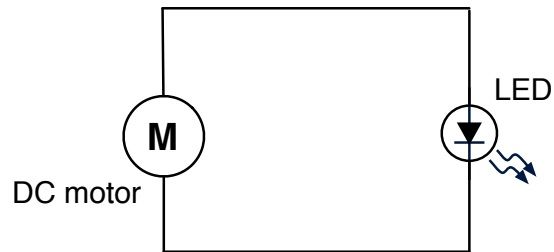
polarity determines which way it rotates



Try this out real quick.
Then swap polarity

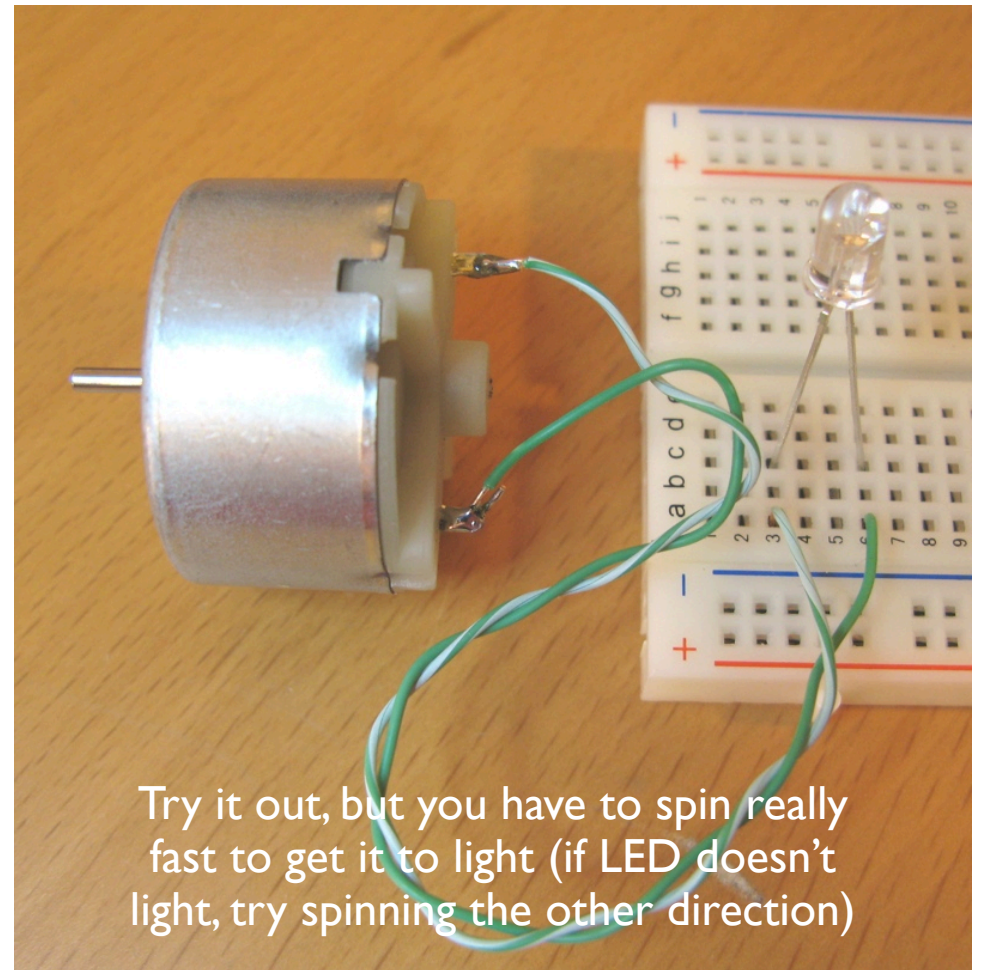
DC Motors as Generators

Just as voltage causes rotation...



...rotation causes voltage

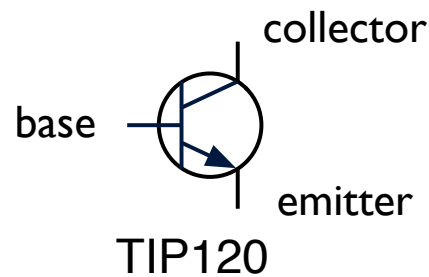
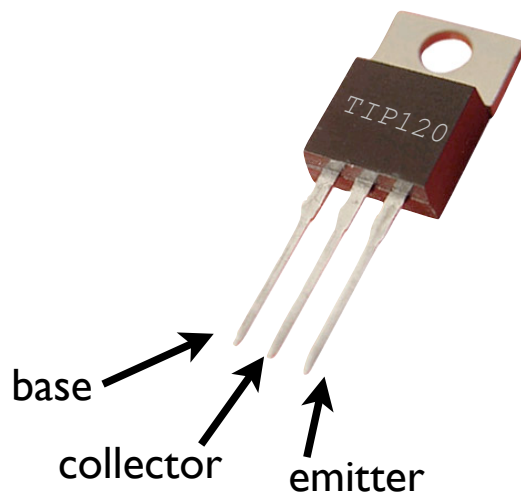
This is used for “regenerative braking” in electric & hybrid cars



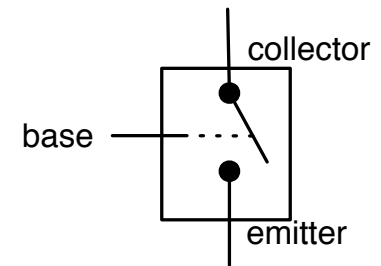
Transistors

Act like switches

electricity flicks the switch instead of your finger



schematic symbol

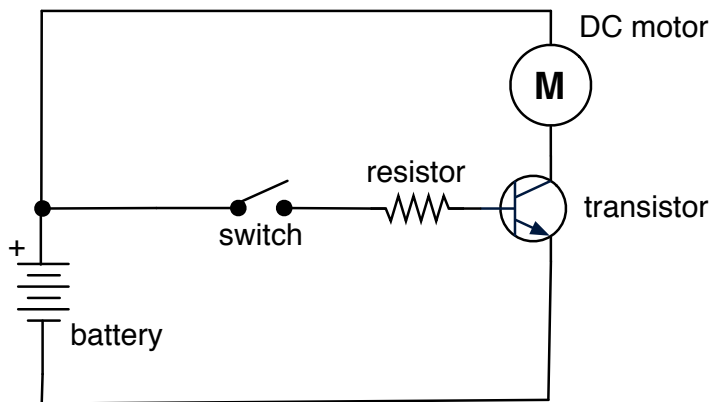


how it kind of works

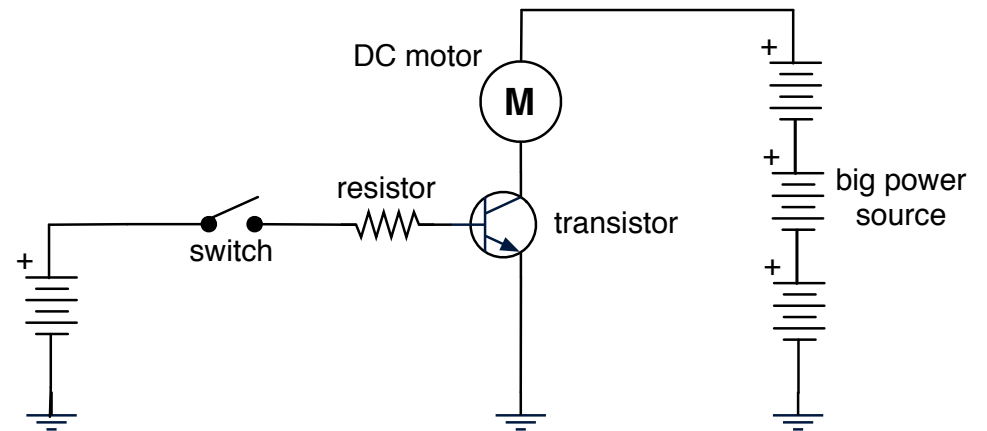
Turning on the “base” connects the “collector” & “emitter” together

Switching Motors with Transistors

little motor



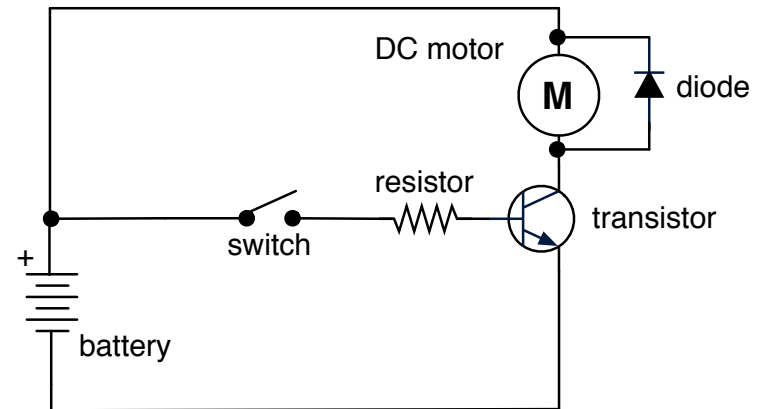
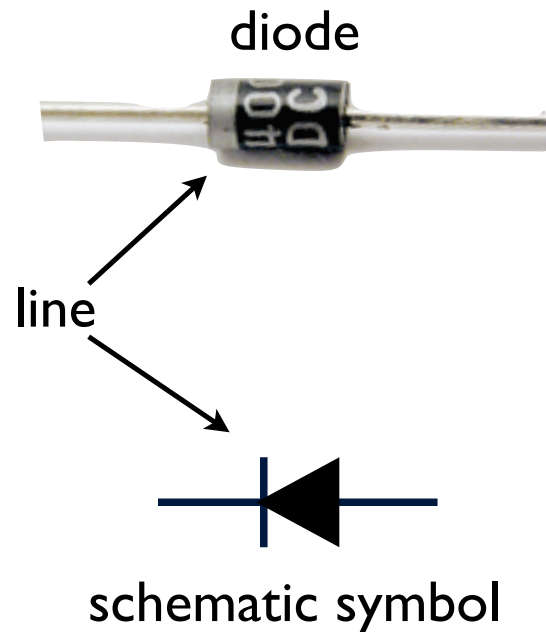
big motor



switching a different power source

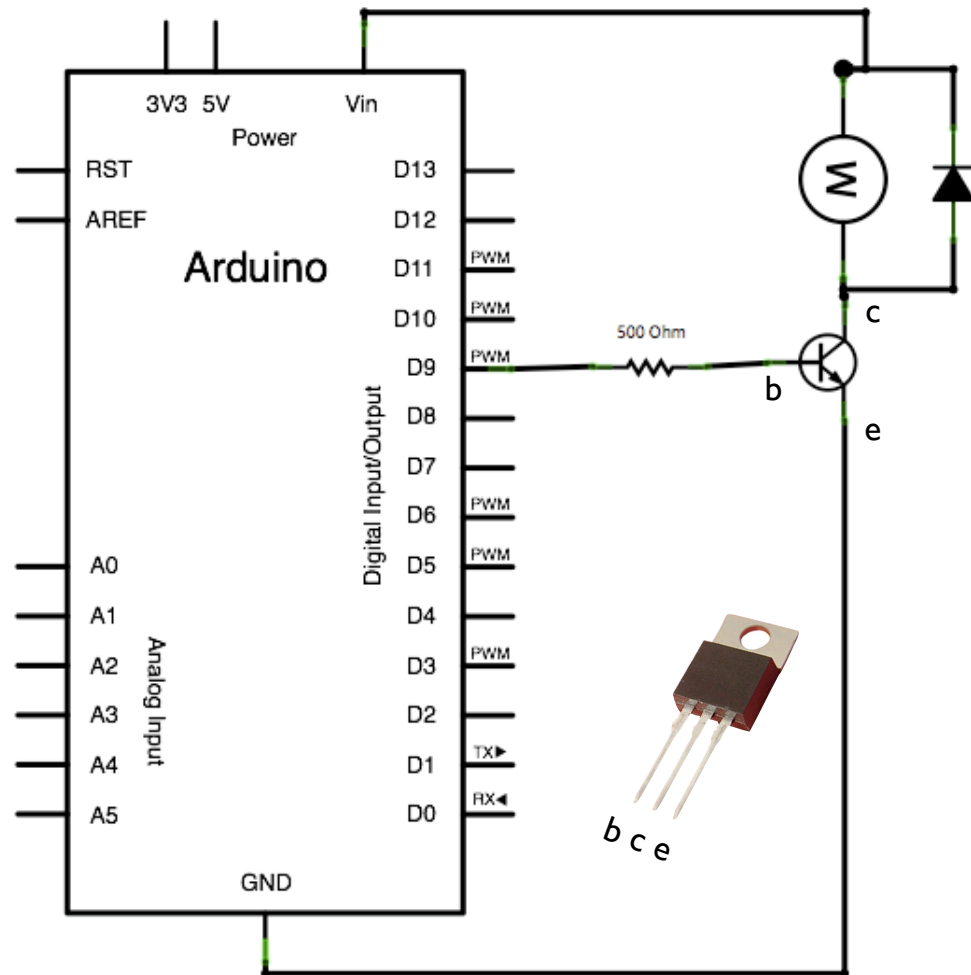
transistors switch big signals with little signals

Need a “Kickback” Diode



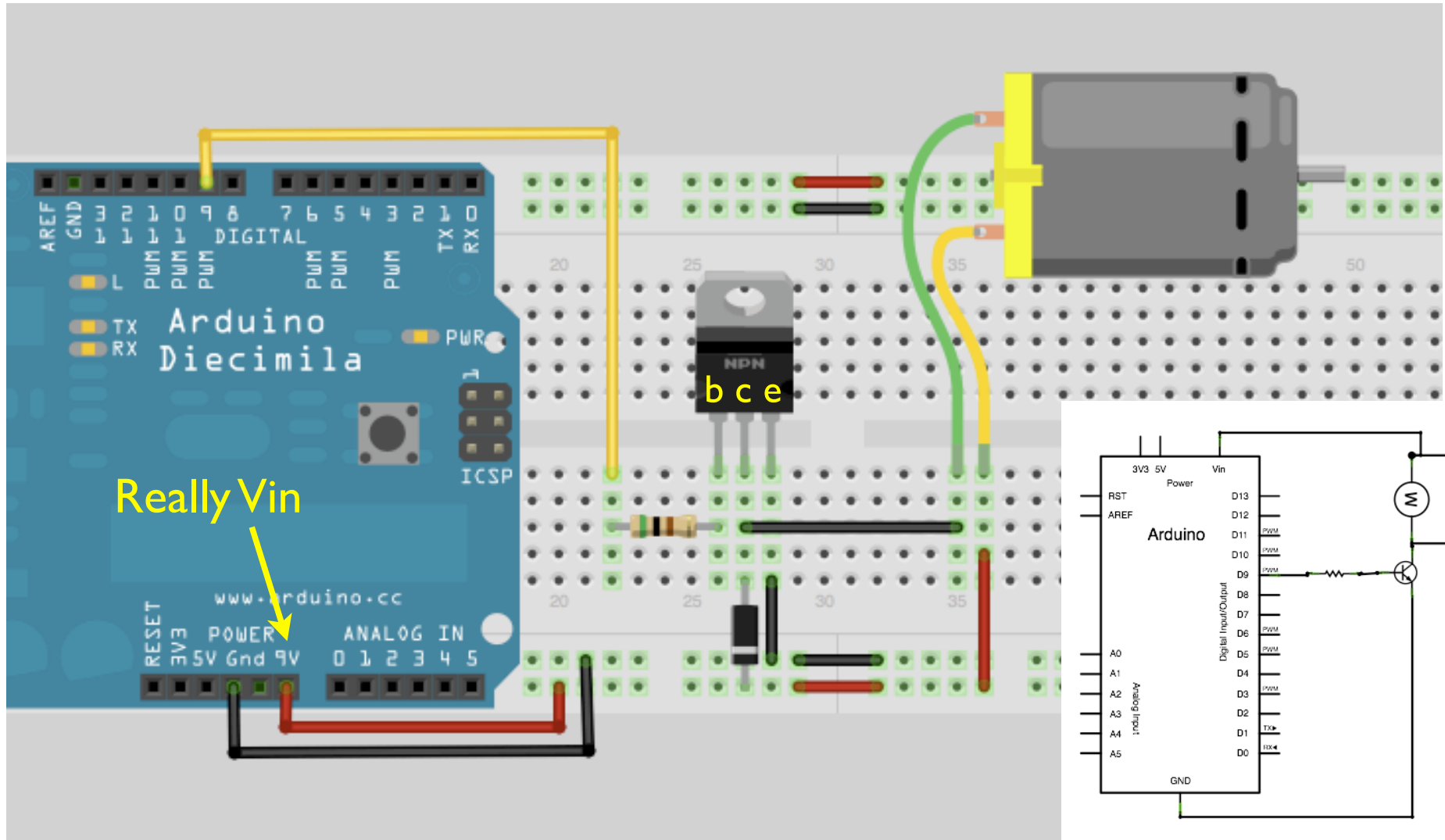
since motors can act like generators,
need to prevent them from generating “kickback” into the circuit

Controlling a Motor



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

Wiring up Motor Circuit

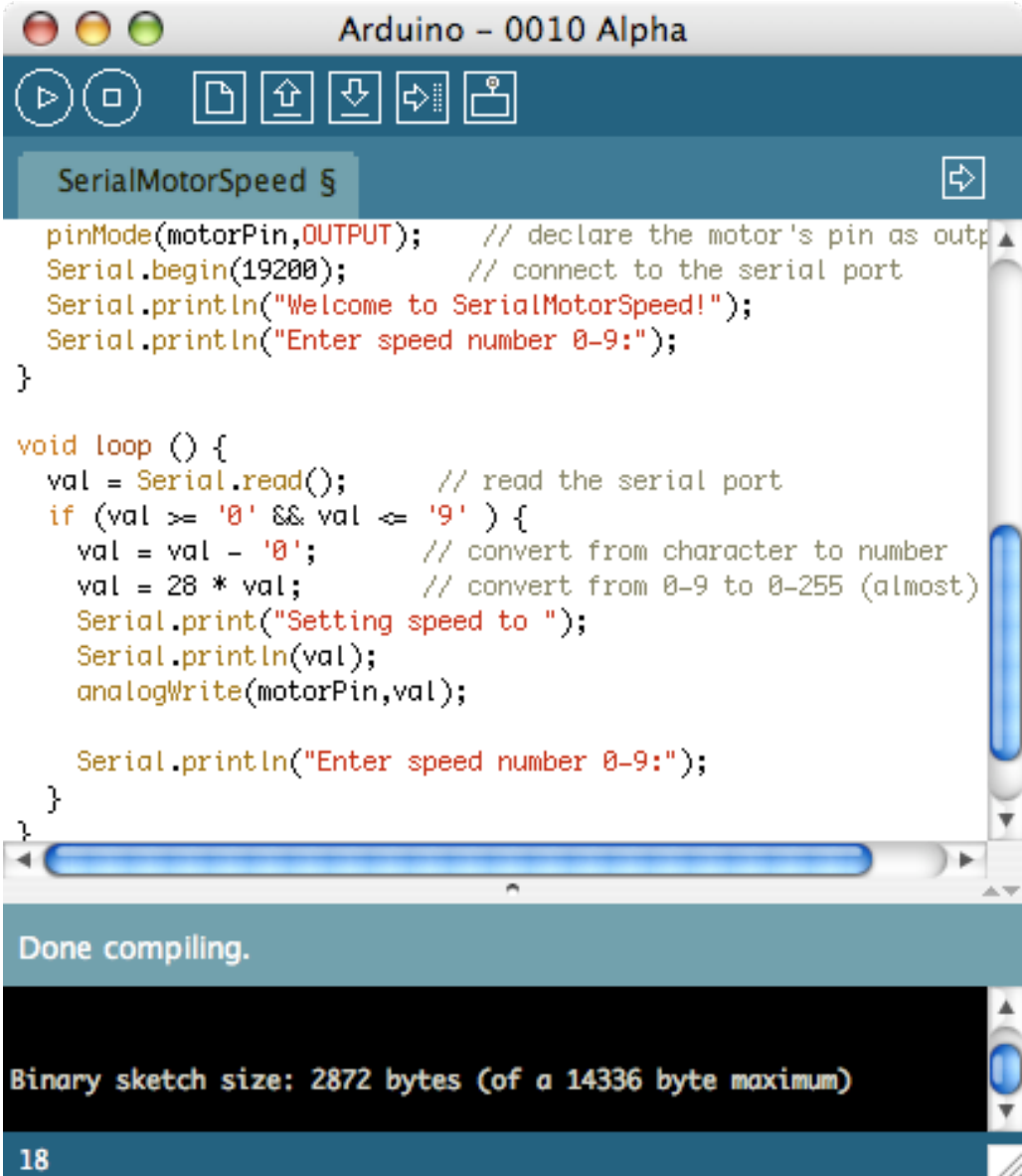


Sketch

“SerialMotorSpeed”

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

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



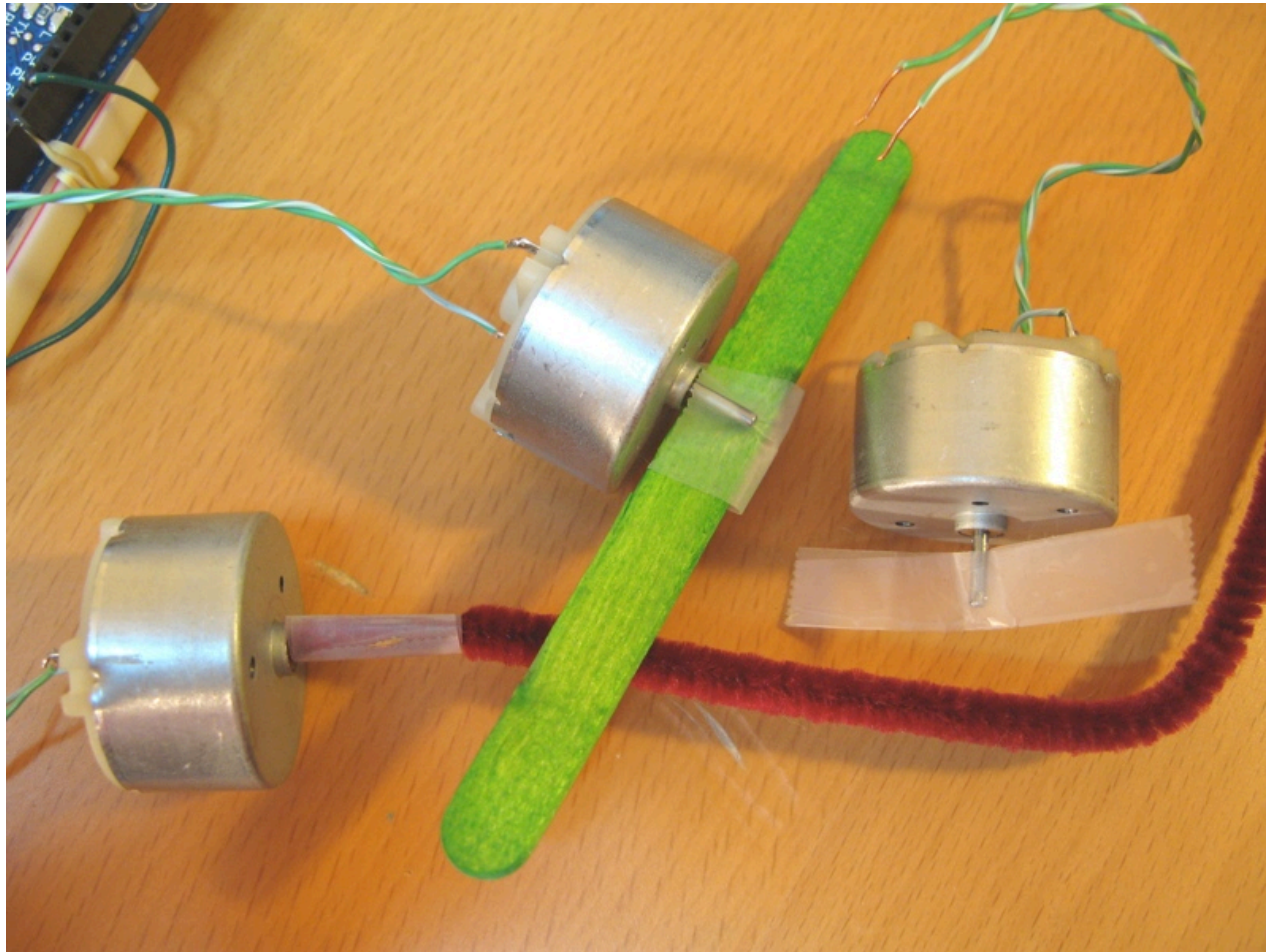
```
Arduino - 0010 Alpha  
SerialMotorSpeed §  
pinMode(motorPin,OUTPUT); // declare the motor's pin as output  
Serial.begin(19200); // connect to the serial port  
Serial.println("Welcome to SerialMotorSpeed!");  
Serial.println("Enter speed number 0-9:");  
}  
  
void loop () {  
  val = Serial.read(); // read the serial port  
  if (val >= '0' && val <= '9') {  
    val = val - '0'; // convert from character to number  
    val = 28 * val; // convert from 0-9 to 0-255 (almost)  
    Serial.print("Setting speed to ");  
    Serial.println(val);  
    analogWrite(motorPin,val);  
  
    Serial.println("Enter speed number 0-9:");  
  }  
}
```

Done compiling.

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

18

Fun Motor Attachments

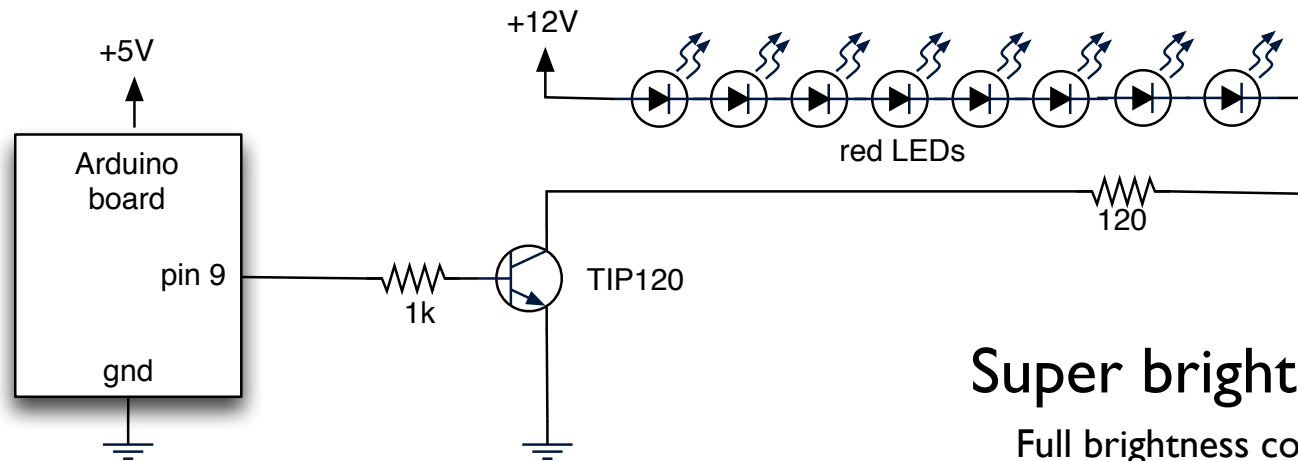


pipe cleaner squiggler

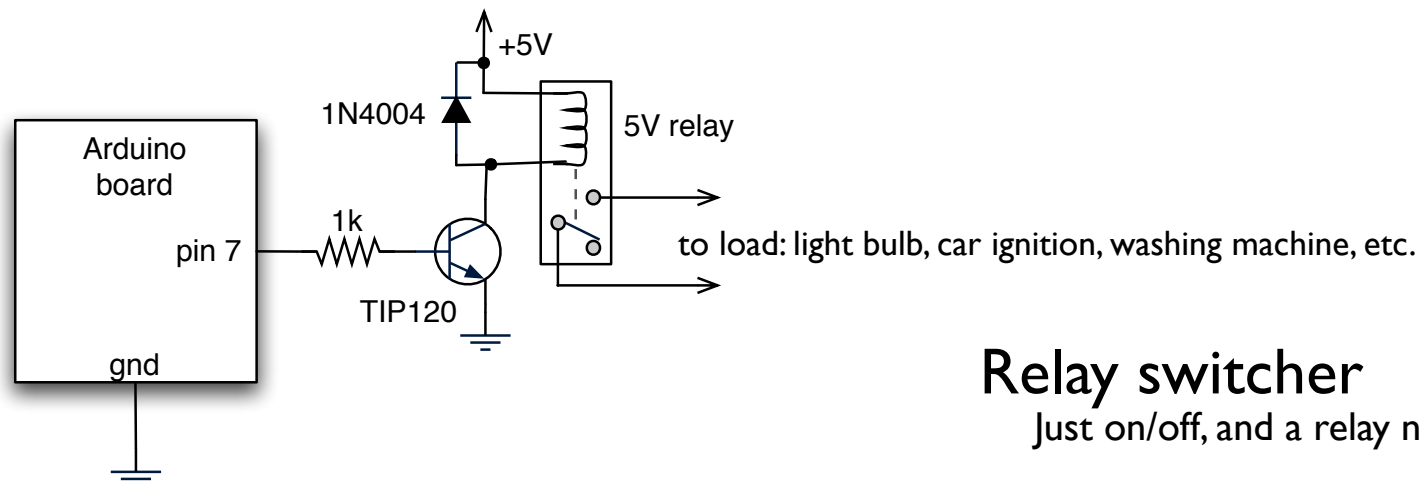
popsicle stick beater

tape propeller

Can Switch Anything*



Super bright LED light
Full brightness control with PWM



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

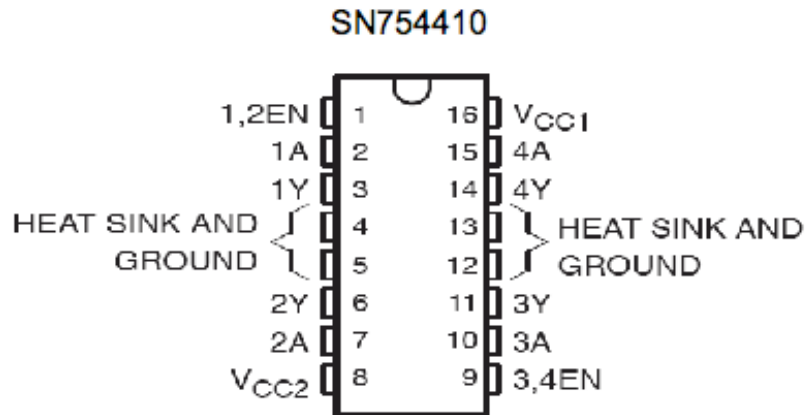
*Anything up to about 1 amp. Need a bigger transistor or a relay after that

H-bridges



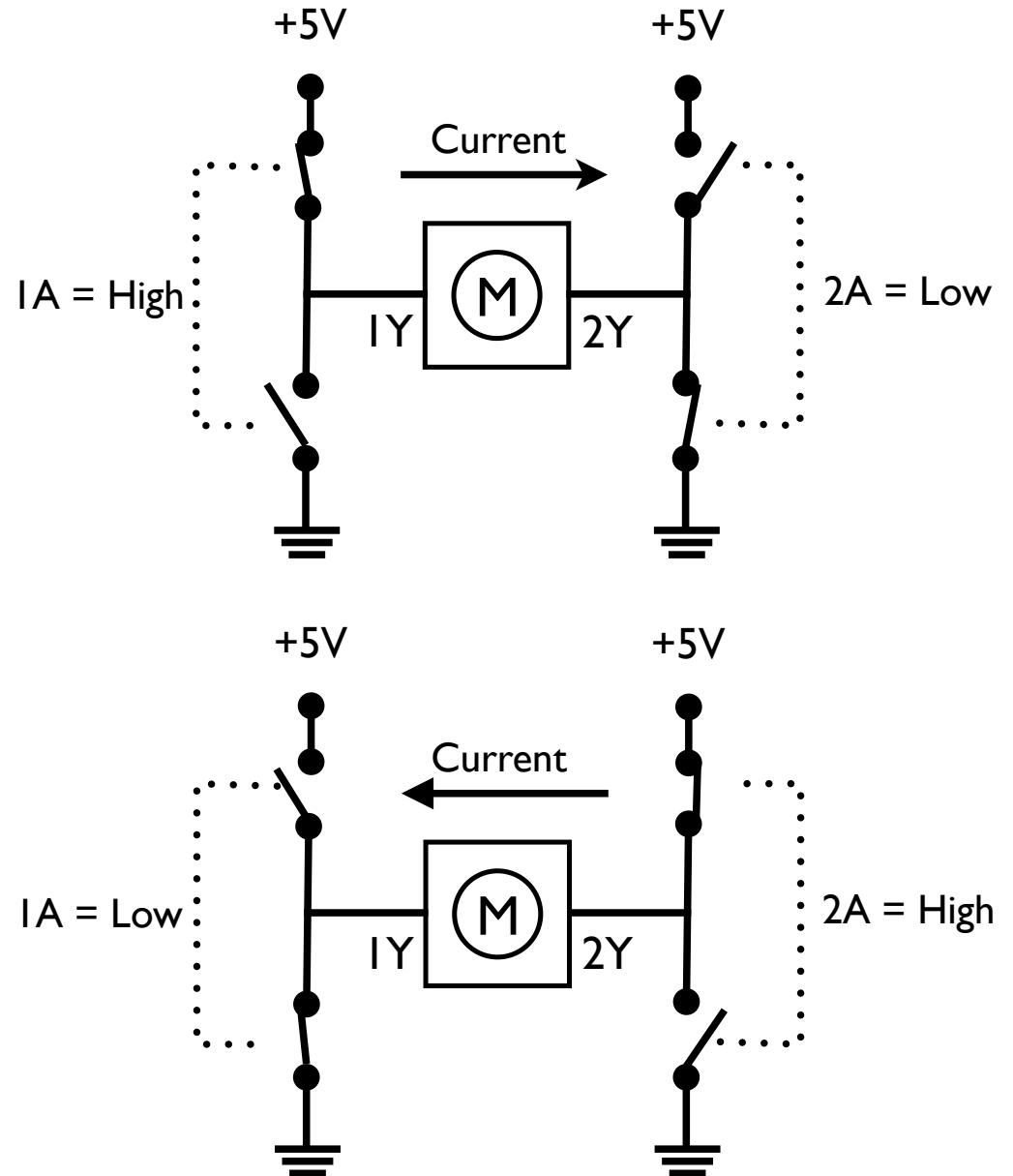
- What if you want to control the rotation of a motor without having to rewire its power connection?
- An H-bridge is an integrated circuit (IC) that is able to interpret digital commands and spin the motor in the appropriate direction

H-bridge Internals

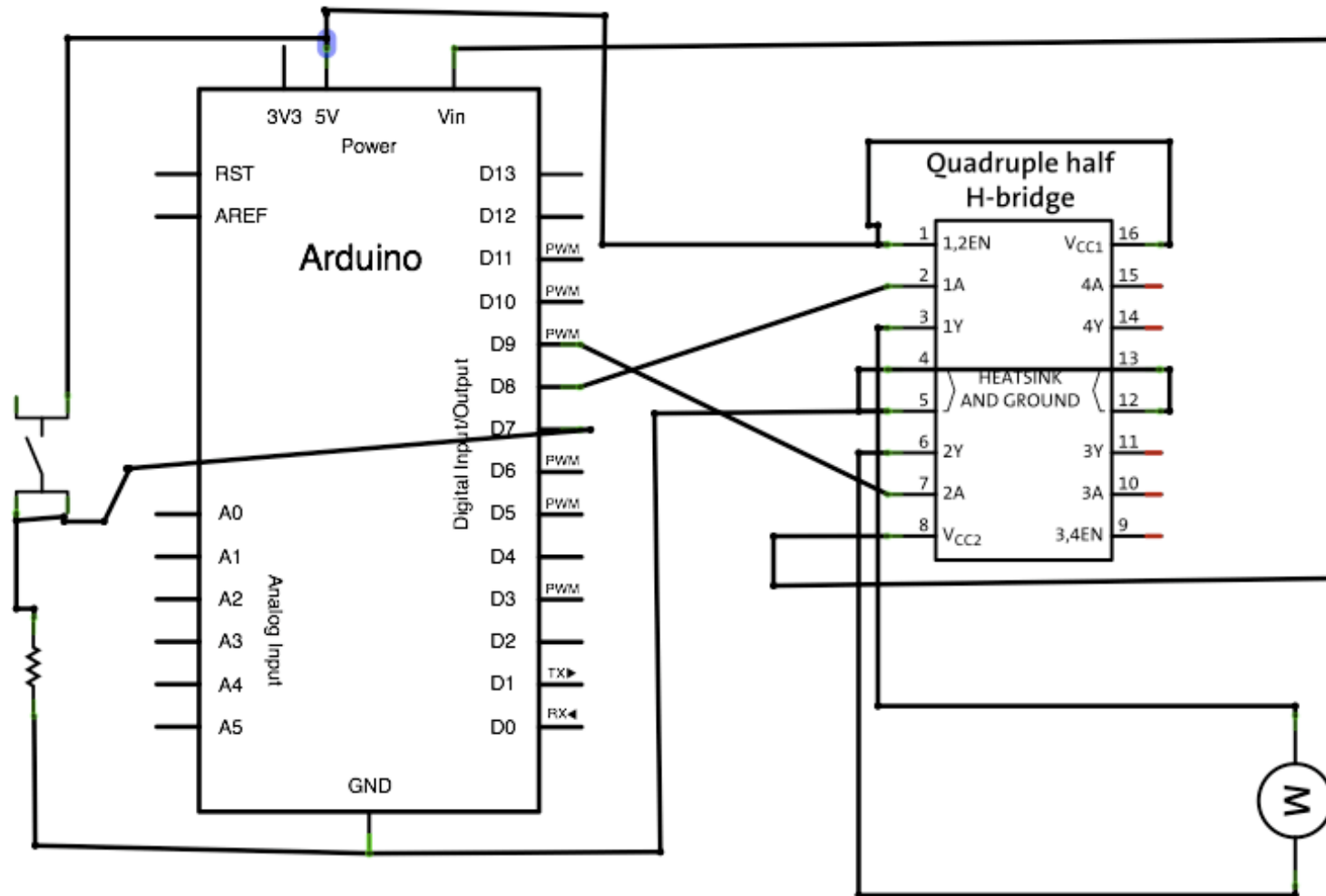


Command Interpretation

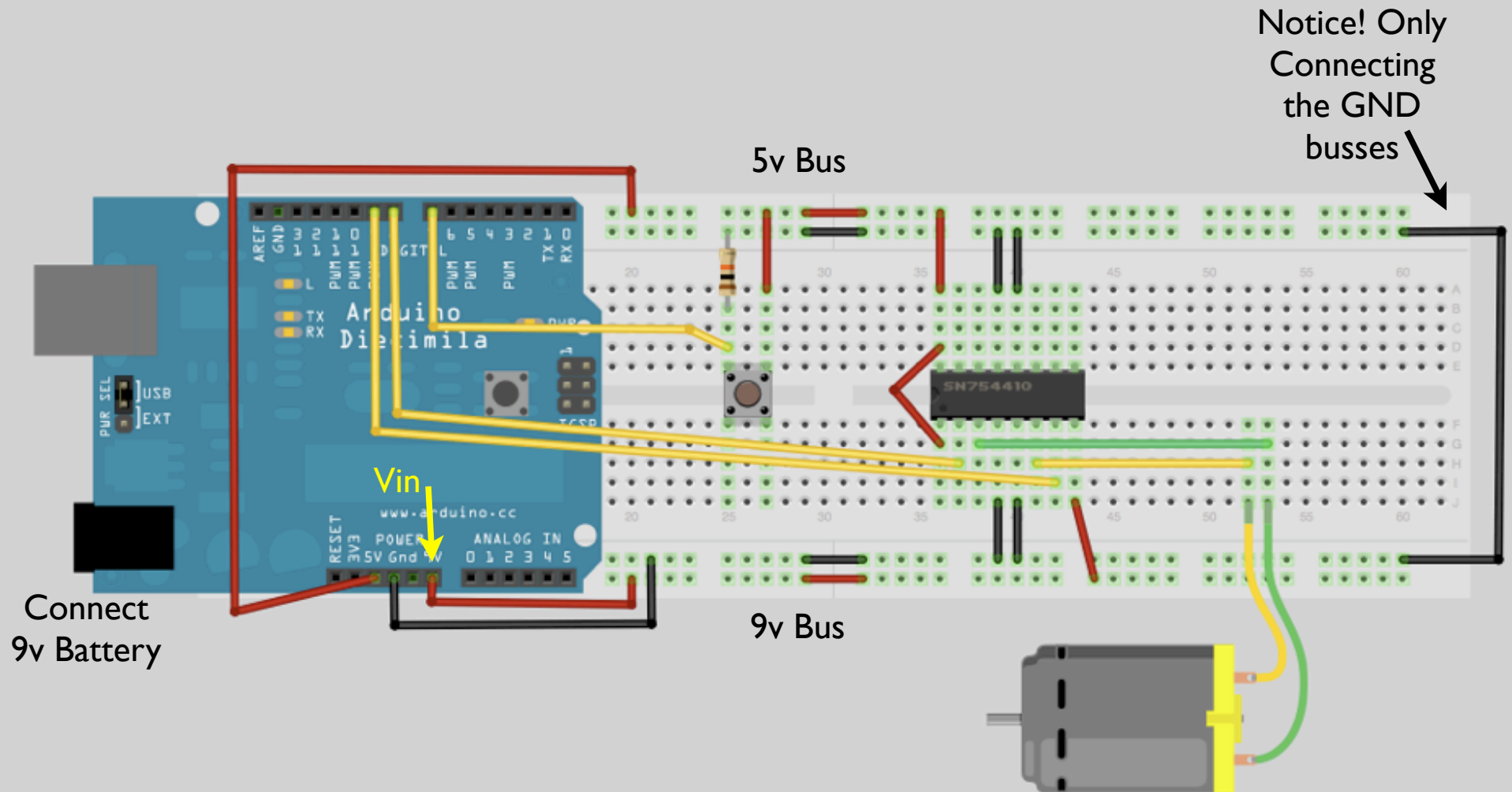
Input	Output
$I_A = \text{High}$ $2A = \text{Low}$	$I_Y = \text{High}$ $2Y = \text{Low}$
$I_A = \text{Low}$ $2A = \text{High}$	$I_Y = \text{Low}$ $2Y = \text{High}$



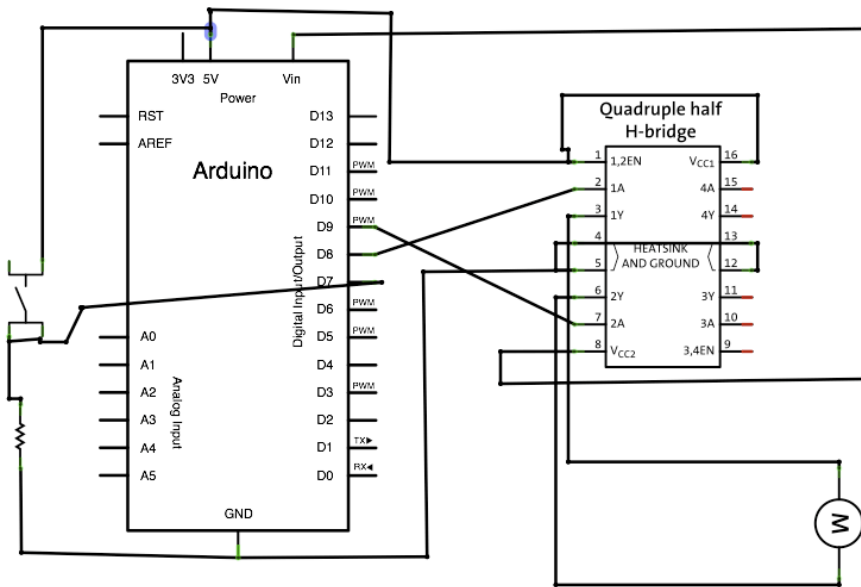
H-bridge Schematic



Putting it together



H-bridge Code



```
// digital directional control of a DC motor  
// using an H-bridge
```

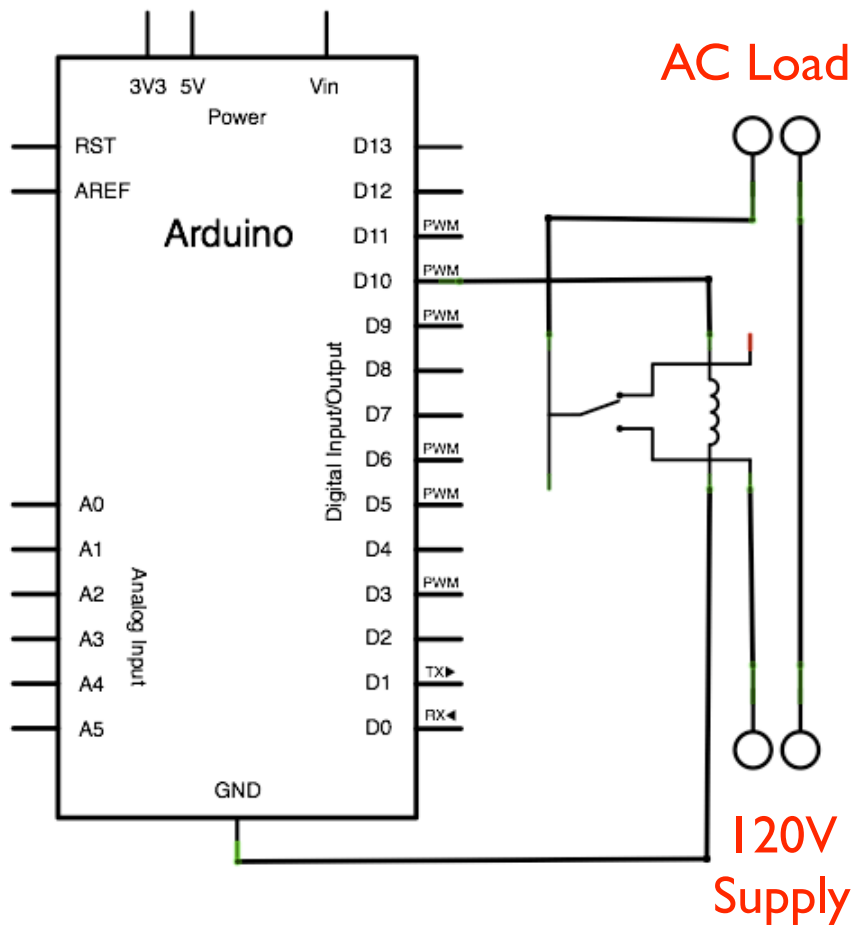
```
int inputPin = 7; // read digital pin 7  
int outputPin1 = 8; // output pin 1a to bridge digital pin 8  
int outputPin2 = 9; // output pin 2a to bridge digital pin 9  
int val = 0;
```

```
void setup()  
{  
  pinMode(inputPin, INPUT);  
  pinMode(outputPin1, OUTPUT);  
  pinMode(outputPin2, OUTPUT);  
}
```

```
void loop()  
{  
  val = digitalRead(inputPin);  
  
  if (val == HIGH) {  
    digitalWrite(outputPin1, HIGH);  
    digitalWrite(outputPin2, LOW);  
  }  
  else {  
    digitalWrite(outputPin1, LOW);  
    digitalWrite(outputPin2, HIGH);  
  }  
  delay(100); // wait 100ms  
}
```

Driving AC Loads

- We can drive 120V AC loads using relays
- Relays are mechanical switches that can be switched by applying power to a secondary circuit
- **WARNING** ☠️: 120V AC current can kill! If you have not done this before consult with somebody who has experience!



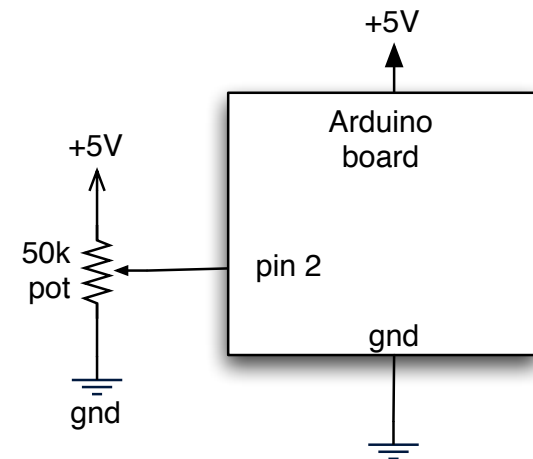
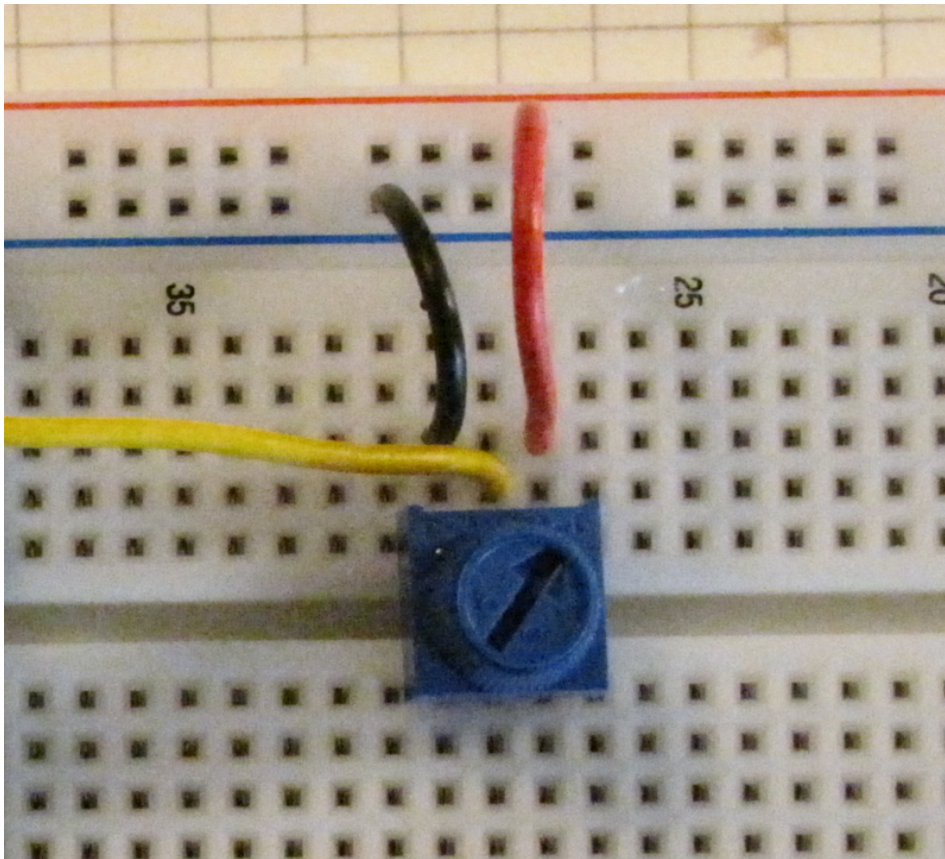
Solid State Relays



- More reliable than mechanical relays
- Less current needed to trigger
- More expensive
- 120VAC still dangerous!

Take a Break

Getting the Board Set Up

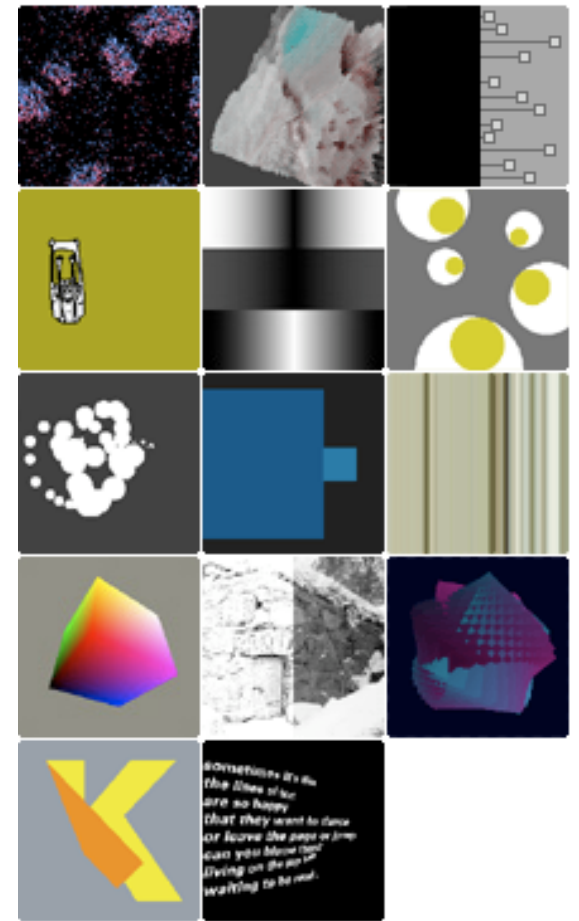


Wire up the potentiometer like from last week

Processing

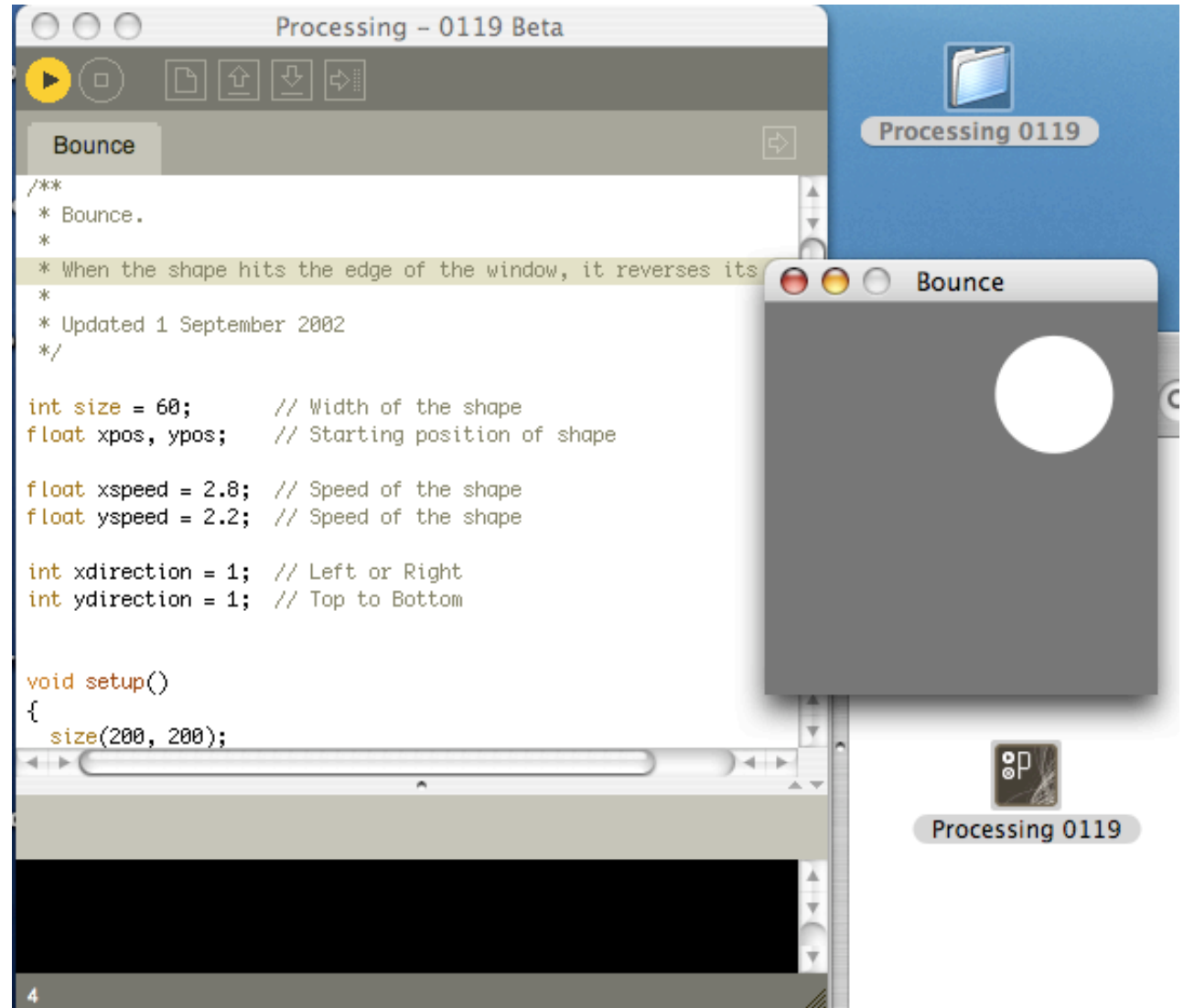


- Processing makes Java programming as fun & easy as Arduino makes AVR programming
- Started as a tool to make generative art
- Is also often used to interface to devices like Arduino
- Think of it as a free Max/MSP



Using Processing

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



About Processing

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

Processing & Arduino

serial communications

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

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

Processing Serial

common Processing serial use

four steps

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

```
import processing.serial.*;

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

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

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

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

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

Arduino Talking to Processing

“PotSend”

Read knob,
send it's value

Note: doesn't send the value as
ASCII text, but as a binary byte
(BYTES are easier to parse in Processing
than other formats)

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



```
Arduino - 0010 Alpha

PotSend

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

int potPin = 2;

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

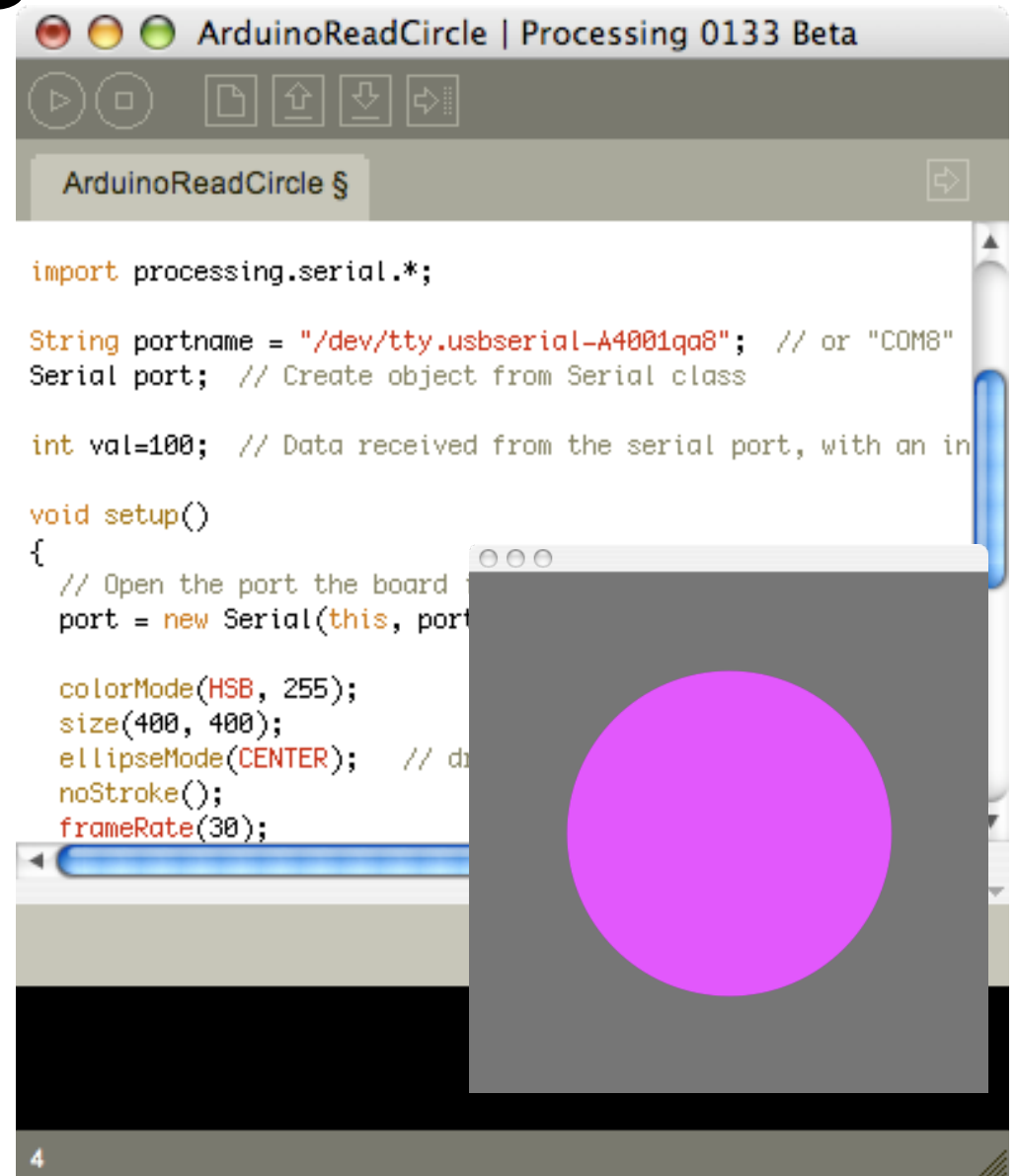
void loop() {
  int val = analogRead(potPin);
  val = val/4;
  Serial.print(val, BYTE);
  delay(75);
}
```

Processing + Arduino

“ArduinoReadCircle”

The pot controls
the hue of the
onscreen circle

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

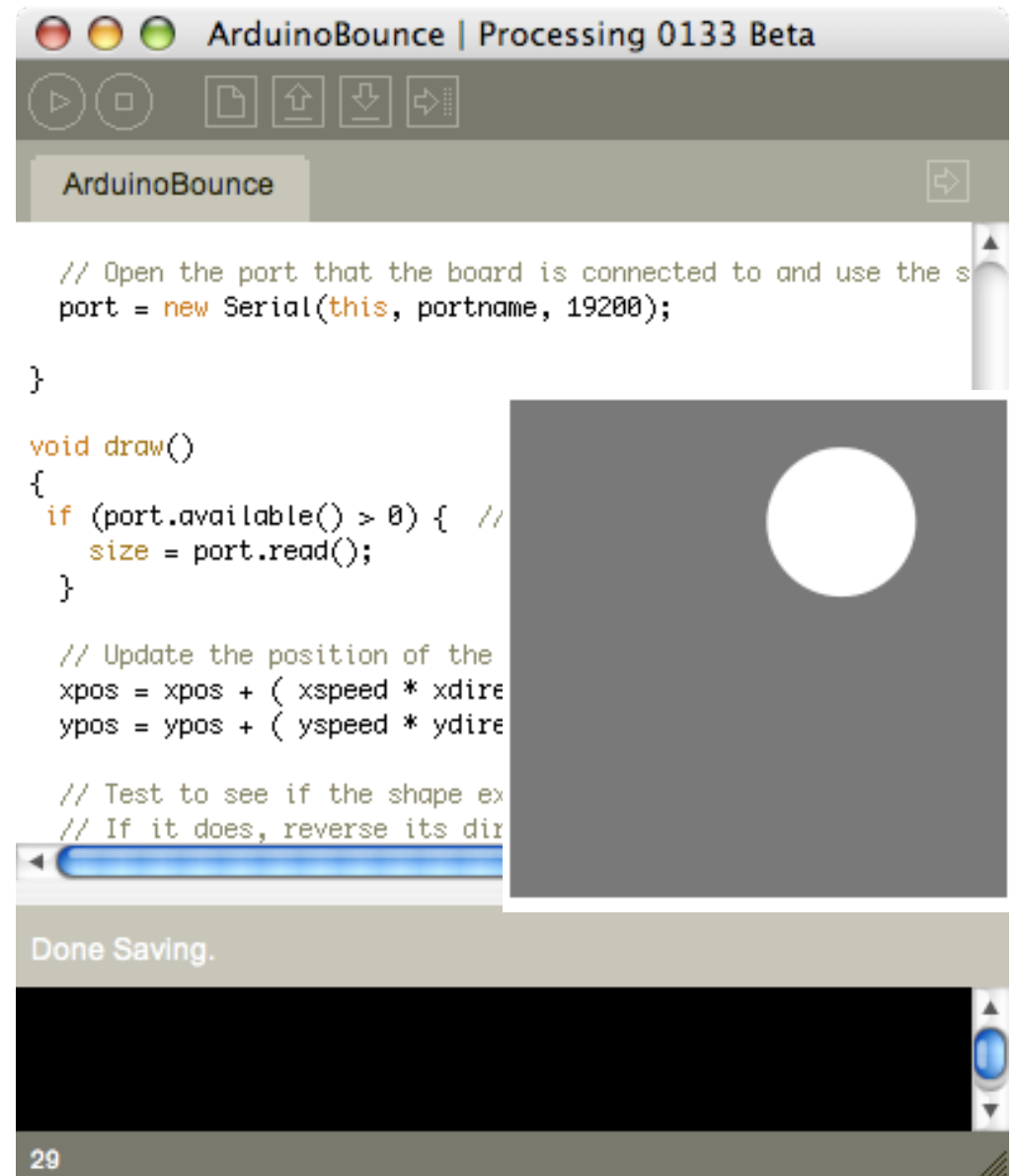


Another One

“ArduinoBounce”

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

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

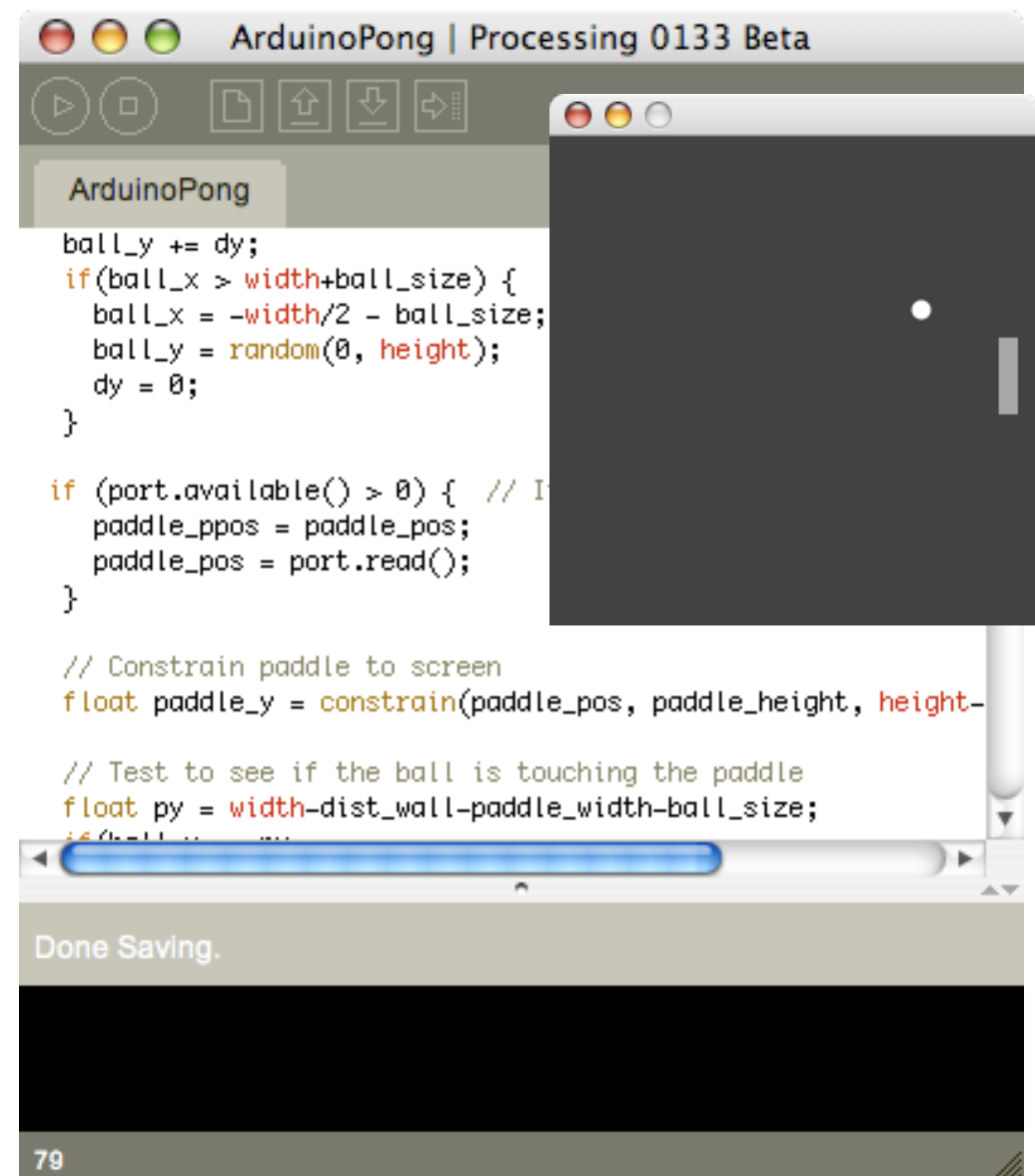


And Another One

“ArduinoPong”

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

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



Processing to Arduino

real quick

"http_rgb_led"

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

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

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

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

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


Going Further

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

Going Further

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

Going Further

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

END Class 3

<http://duksta.org/electronics/arduinooclass>

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