

Lights! Speed! Action!

Fundamentals of Physical Computing for Programmers

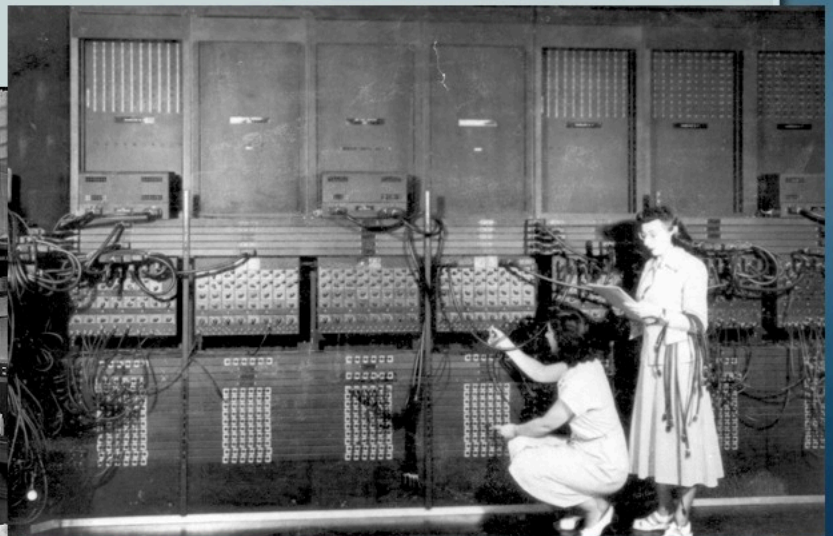
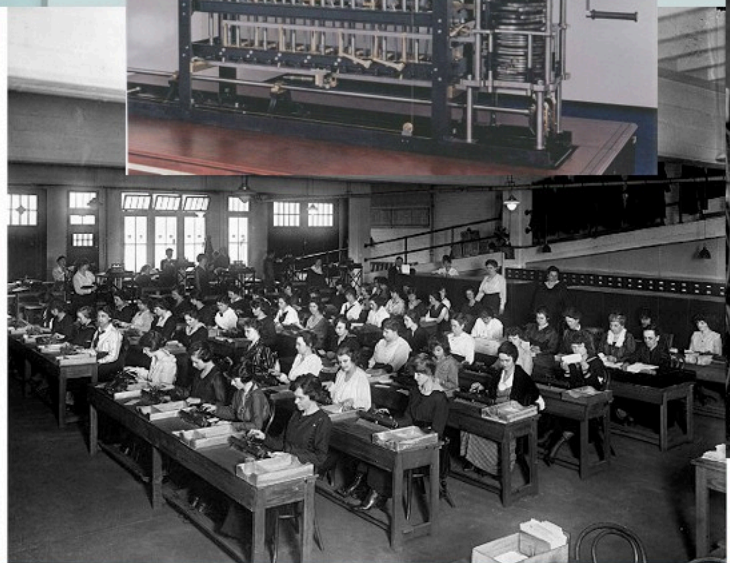
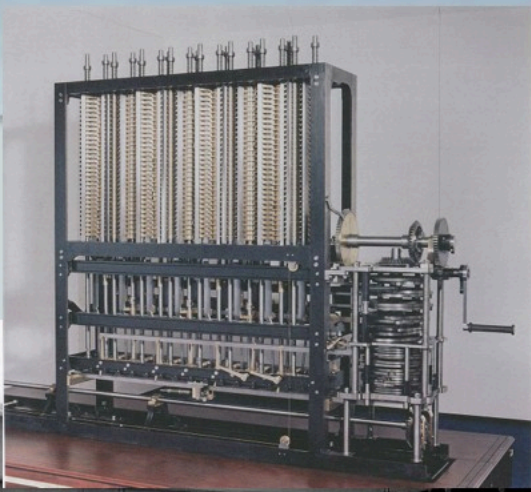


SIGGRAPH2013

*Erik Brunvand
School of Computing
University of Utah*



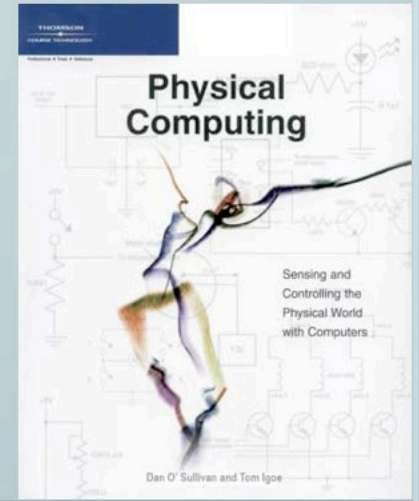
Physical Computing?



Physical Computing

- Dan O'Sullivan and Tom Igoe's book (2004) has a title that nicely captures the idea

– *Physical Computing:
Sensing and Controlling the
Physical World with Computers*



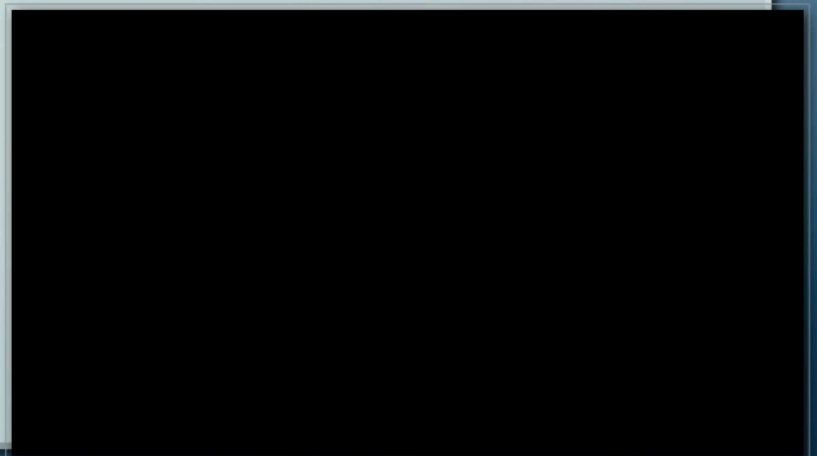
3

Physical Computing

- Why should you care?
You're into computer graphics!

The Bay Lights, Leo Villareal, 2013

– But computer graphics
can be about a LOT
more than putting
images on the screen...



4

SIGGRAPH Art Gallery



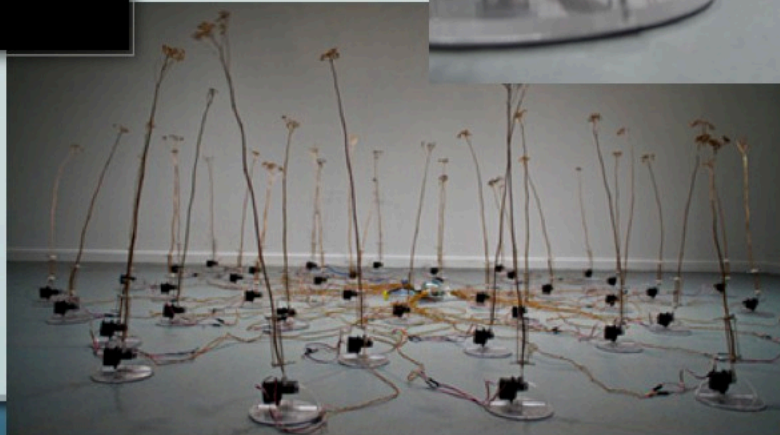
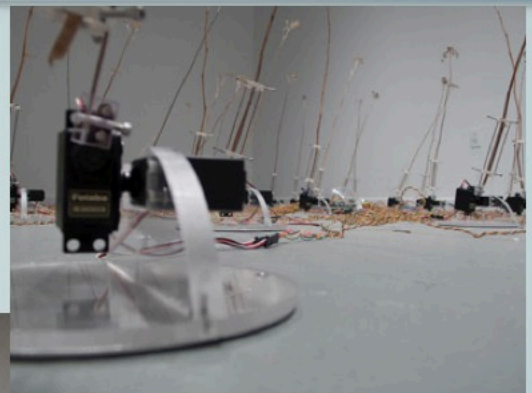
5

SIGGRAPH2013

SIGGRAPH Art Gallery

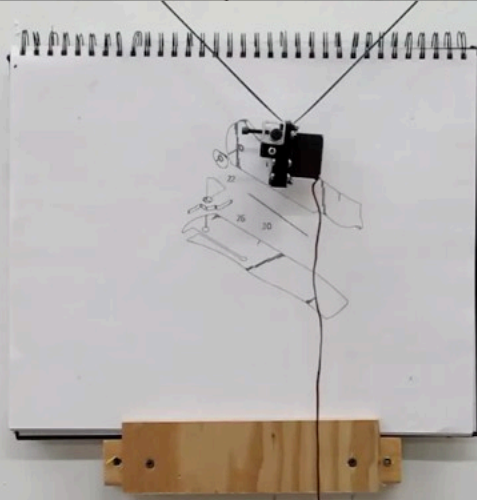
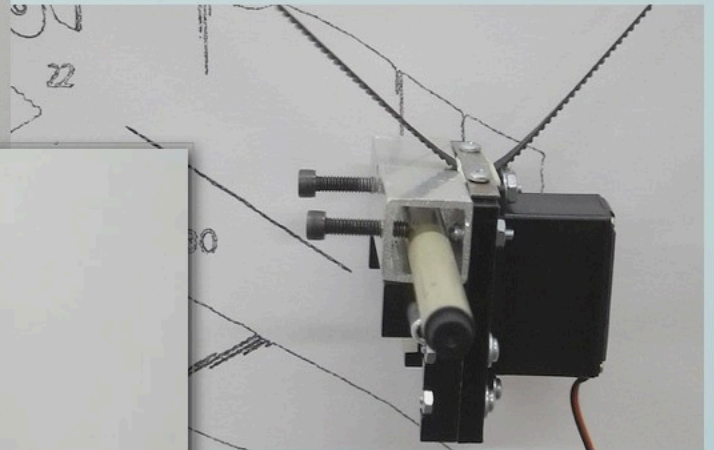
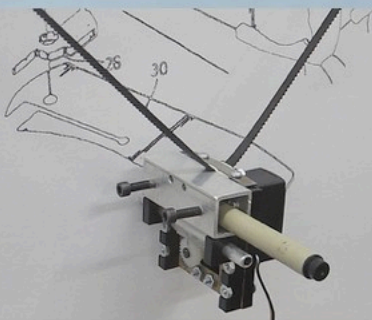
telepresent wind
2009

Telepresent Wind,
David Bowen,
SIGGRAPH 2010



6

SIGGRAPH2013



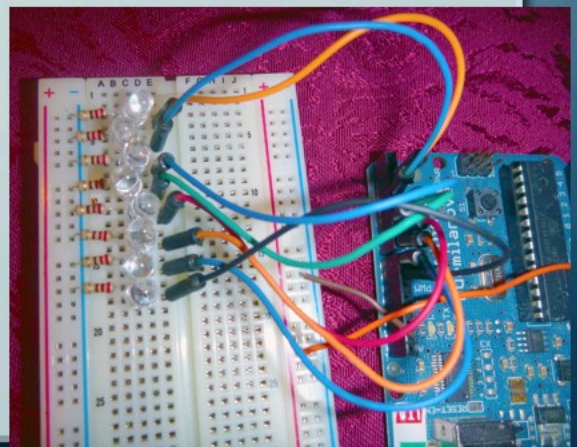
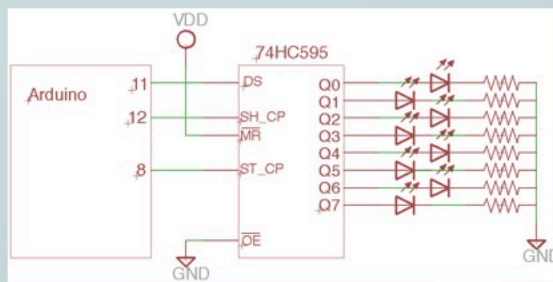
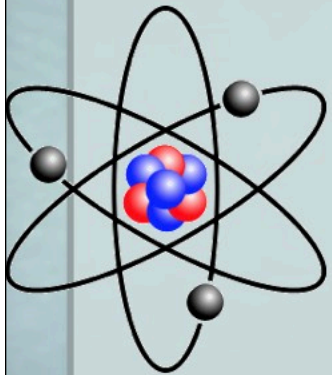
Drawing Machine, Robert Twomey, 2013



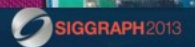
Agenda

Learn enough about electronics to be dangerous!

Think about the right questions to ask

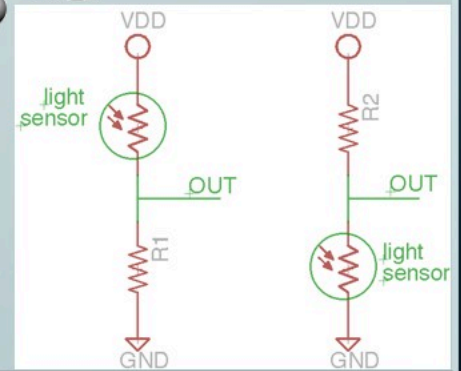
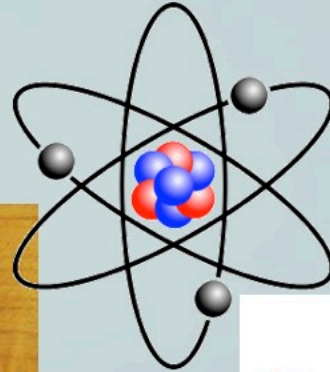
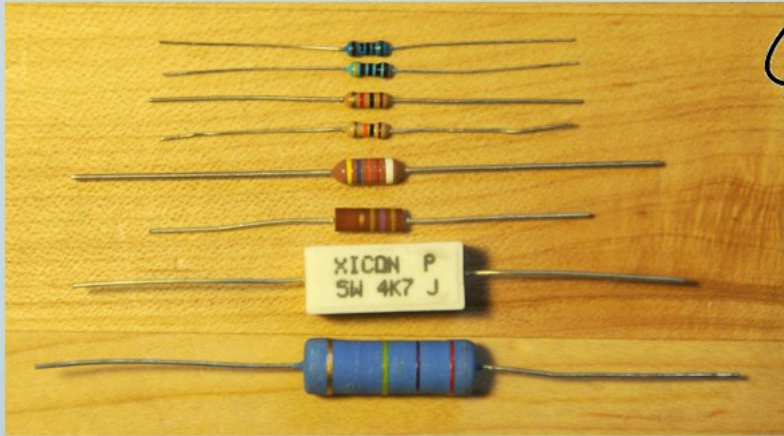


Programming interfaces to physical devices...
...LEDs, sensors, servos and motors



More Detailed Agenda

•Electronics Fundamentals



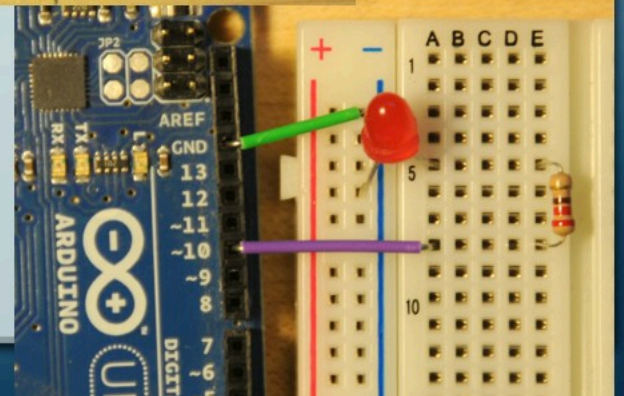
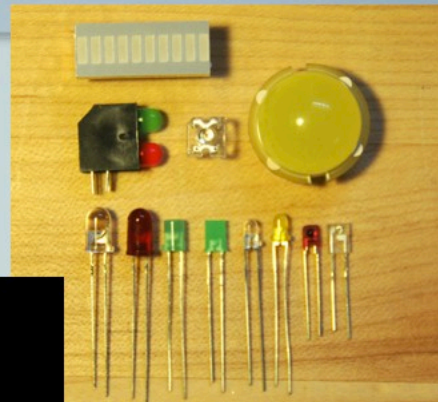
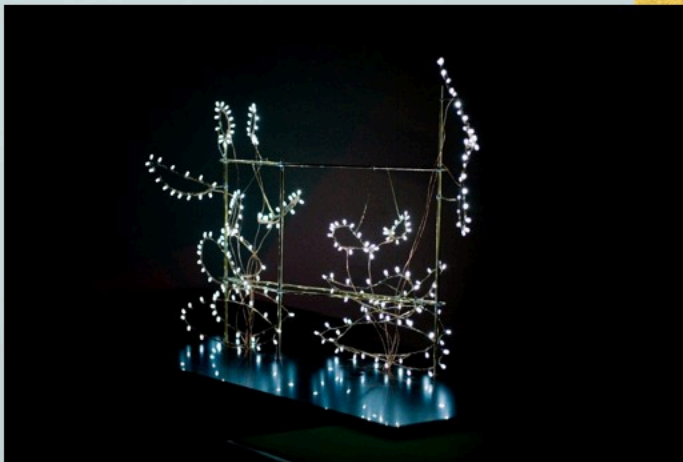
9

SIGGRAPH 2013

More Detailed Agenda

•Electronics Fundamentals

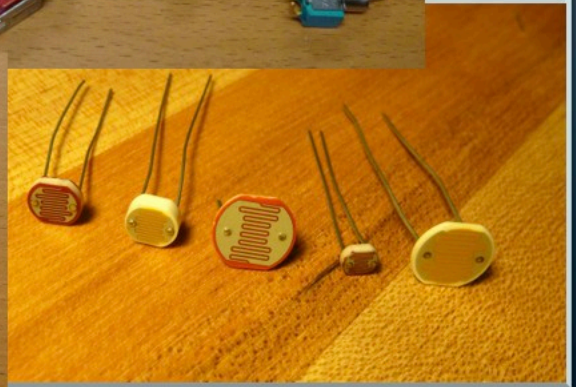
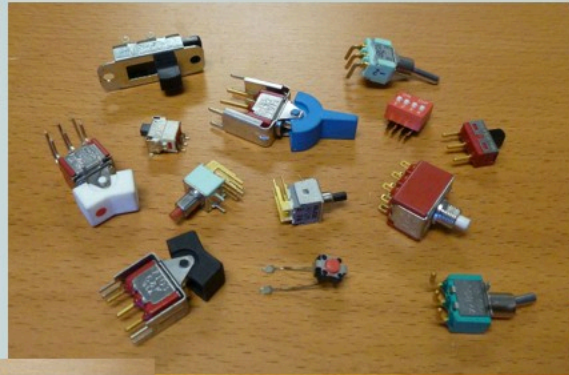
-Lights! (LEDs)



10

More Detailed Agenda

- Electronics Fundamentals
 - Lights!
 - Speed! (Sensors)

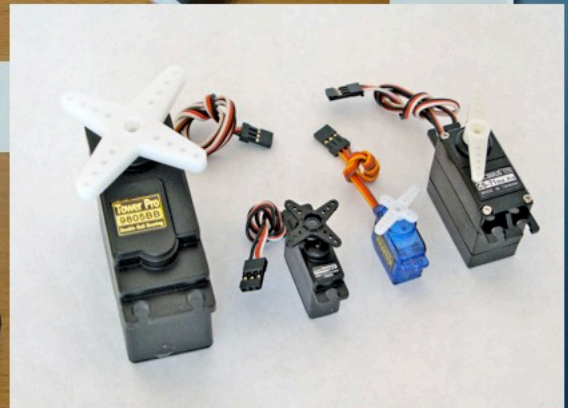
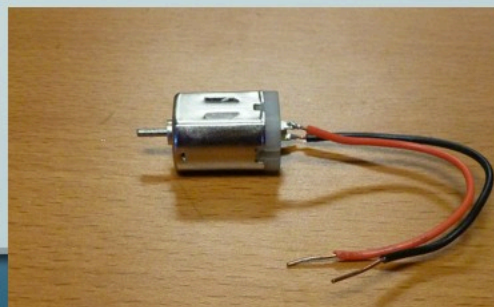
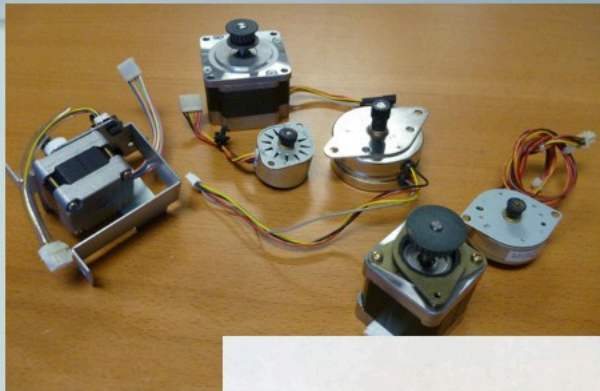


11

SIGGRAPH2013

More Detailed Agenda

- Electronics Fundamentals
 - Lights!
 - Speed!
 - Action! (Servos and motors)

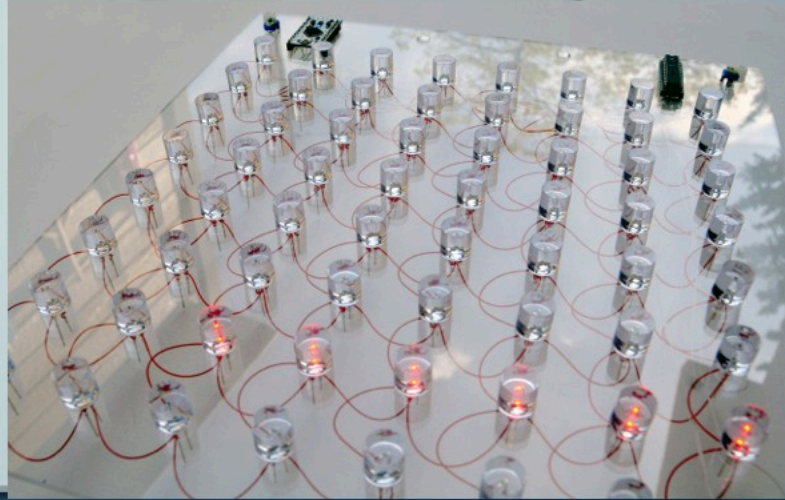


12

SIGGRAPH2013

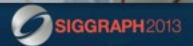
More Detailed Agenda

- Electronics Fundamentals
 - *Lights!*
 - *Speed!*
 - *Action!*
- Conclusions and context



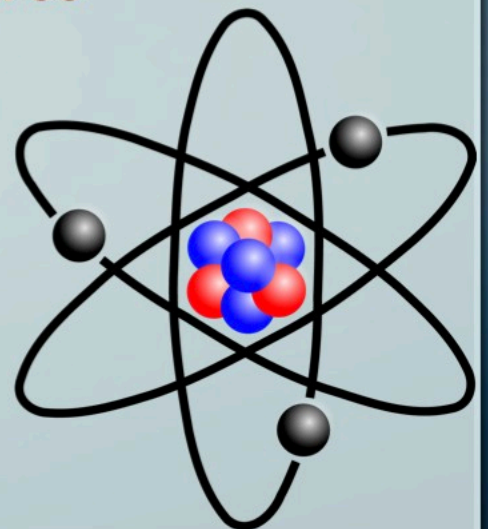
13

Serpente Rosso, Erik Brunvand, 2012



Electronics Fundamentals

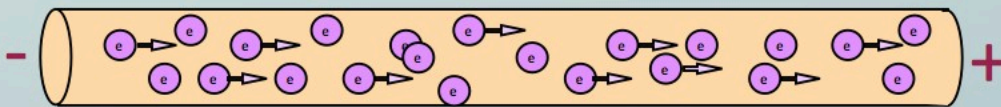
- Electronics: A variety of phenomena related to **charge moving** in response to a **force**
 - **Charge?** Charged subatomic particles
 - Protons and electrons
 - **Moving?** Electronic current
 - Like charges repel; Opposite charges attract
 - **Force?** Electromagnetic fields
 - Measured as voltage



14



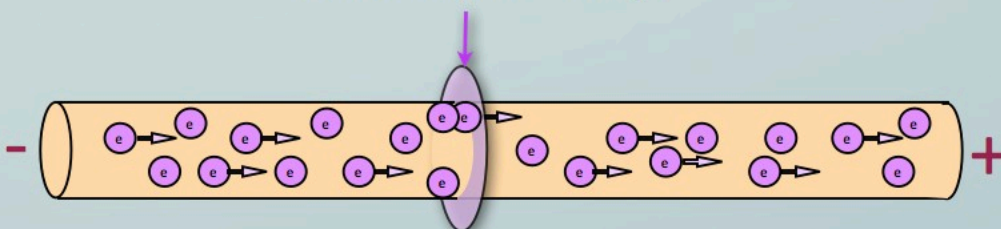
- Apply an electric field
 - electrons are influenced by the field - move in response
- Charge is measured in coulombs
 - One coulomb = charge on 6.241×10^{18} electrons
 - Copper has 1.38×10^{24} free electrons / in^3



Current in Amperes (Amps)

- One **Ampere** of current is one **Coulomb** of charge moving past a point in one **Second**

Coulomb/sec = Amps

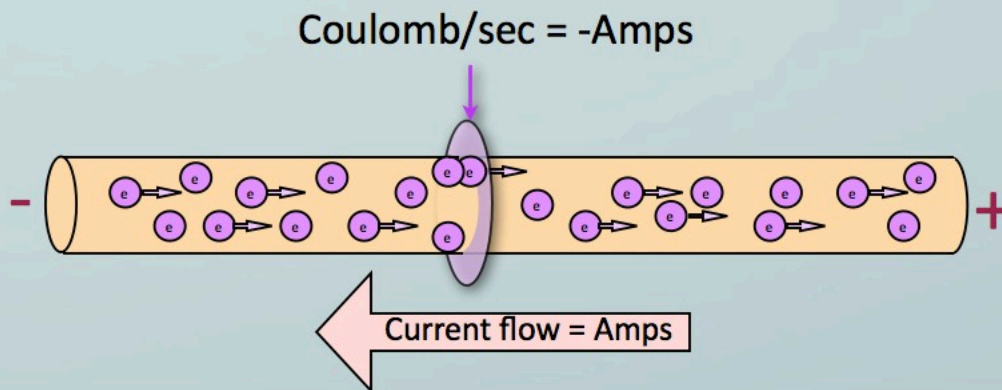


Confusing Concept #1: Current Direction



- Conventional current (positive current) flows from positive to negative!

–As if charge carriers were positive...



17

SIGGRAPH2013

Voltage: Force acting on charge

- Electrical force is measured in volts
 - Voltage is potential energy
 - 1v is the energy required to move 1 coulomb of charge
 - Two points in a circuit characterized by their **voltage difference** (not an absolute quantity)
- Arbitrary reference point for 0v called Ground (GND)
 - In your house, this is actually the ground...



18

SIGGRAPH2013

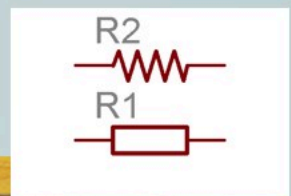
Controlling Charge: Big Idea #1



- **Charge moving in a conductor (current) under the influence of a voltage is the main electrical activity that we're interested in**
 - This phenomenon powers LEDs, makes motors move, and is the property that we'll sense in a sensor to measure our environment.
 - **Causing current to flow, and controlling that current, is one of our main goals!**

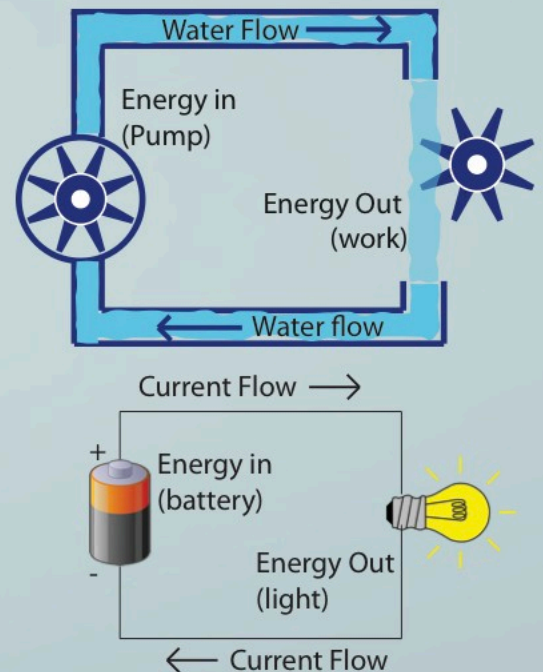
Resistance

- The property of a material to resist current flow
 - Similar to friction in a mechanical system
 - Measured in Ohms
 - Using the symbol Ω
- Color codes for values
 - Look for "resistor calculator" on the web...



Electronics - A Water Analogy

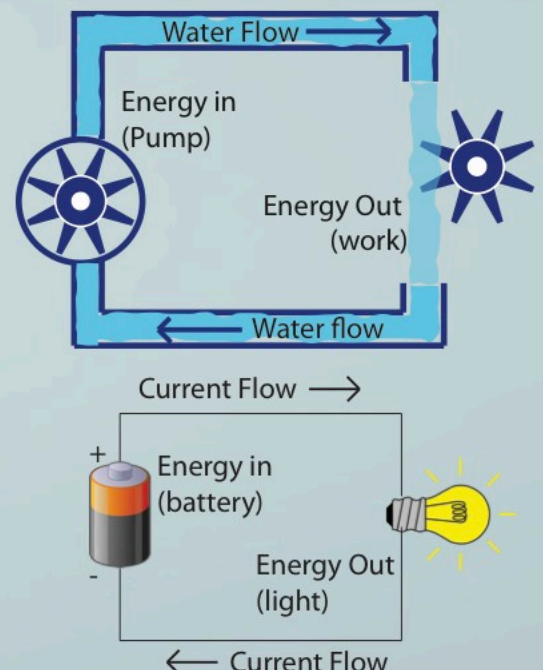
- **Current** is like water flowing
 - **Voltage** is like water pressure
 - **Resistance** is like the diameter of the water pipe
- Water pushed through a pipe can do work (like a water wheel)



21

Electronics - A Water Analogy

- If you have high resistance (small pipe) you have to push harder (more voltage) to get the same amount of water through (current)
- If you have fixed water pressure (voltage) and you lower the resistance (use a bigger pipe), more water will flow (current)



22

Ohm's Law: Big Idea #2



- This relationship is expressed as Ohm's Law
 - *Fundamental relationship between voltage, current, resistance*

$$V = I R$$

V = voltage (in volts)

I = current (in amps)

R = resistance (in ohms)

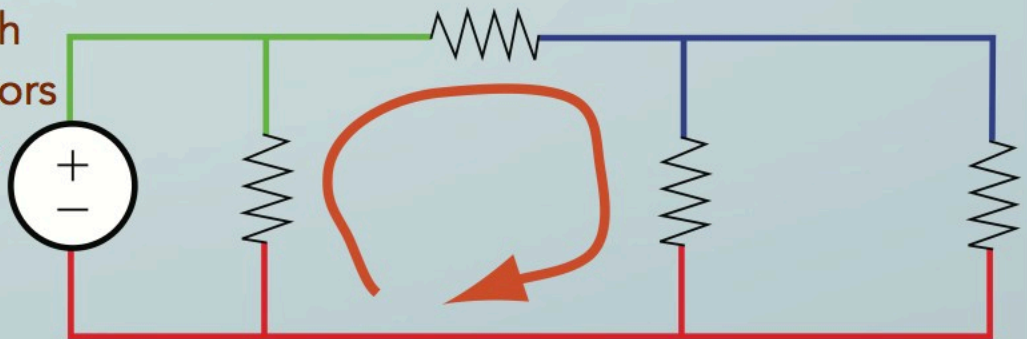
Circuit Nodes: Big Idea #5



- Electrical node in a circuit
 - All points connected through a conductor are at the same electrical potential (same colors below)

- Electrical loop in a circuit

– A connected path through conductors and components that ends up where it started



- **Kirchhoff's Voltage Law (KVL)**

- The sum of voltages around a circuit loop is 0v
- Like a loop hike - just as much uphill as downhill



- **Kirchhoff's Current Law (KCL)**

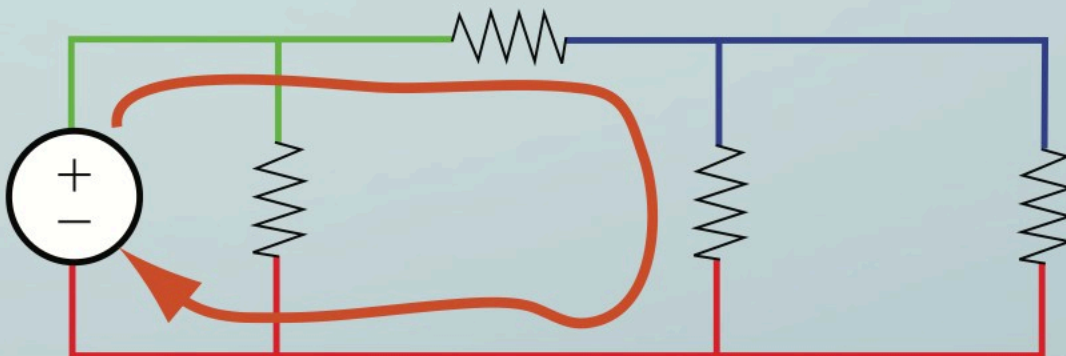
- The sum of currents into and out of a node is 0A
- Like a river splitting into streams:
water in = water out



"Voltage Drop" a la Kirchhoff

- Consider a loop with a battery in it

- 5v supplied by battery, and resistors in the loop
- KVL tells us that voltage is "used up" by the end of the loop
- Where did it go? It's "dropped" across each component



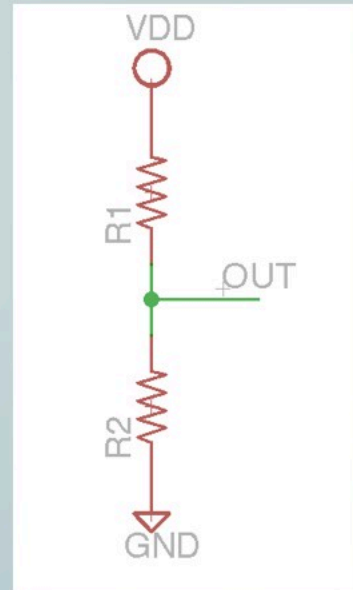
Voltage Divider: Big Idea #6



- Series-connected resistors:

- KVL tells us that all the voltage is dropped
- Ohm's law tells us that the drop is proportional to the resistance values

$$V_{out} = \frac{R_2}{(R_1 + R_2)} V_{dd}$$



Voltage Divider: Big Idea #6

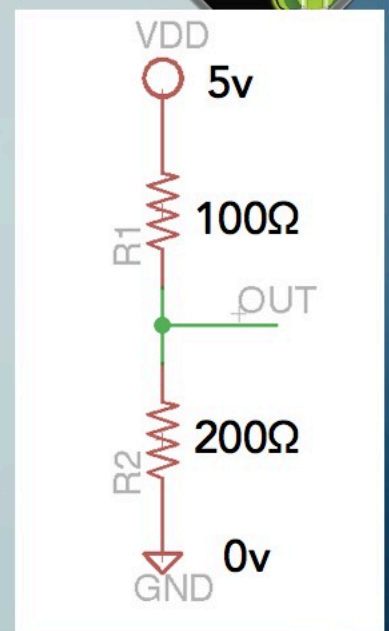


- Series-connected resistors:

- KVL tells us that all the voltage is dropped
- Ohm's law tells us that the drop is proportional to the resistance values

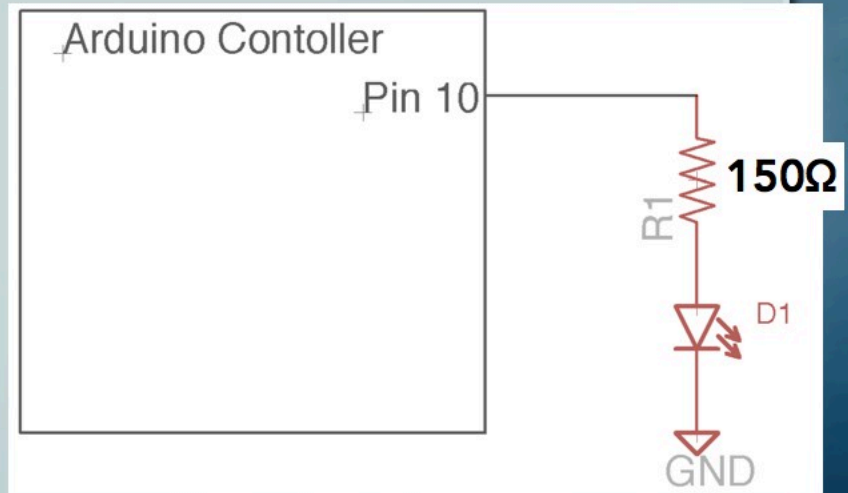
$$V_{out} = \frac{R_2}{(R_1 + R_2)} V_{dd}$$

$$V_{out} = \frac{200\Omega}{(100\Omega + 200\Omega)} 5v = (.667) \times 5v = 3.333v$$

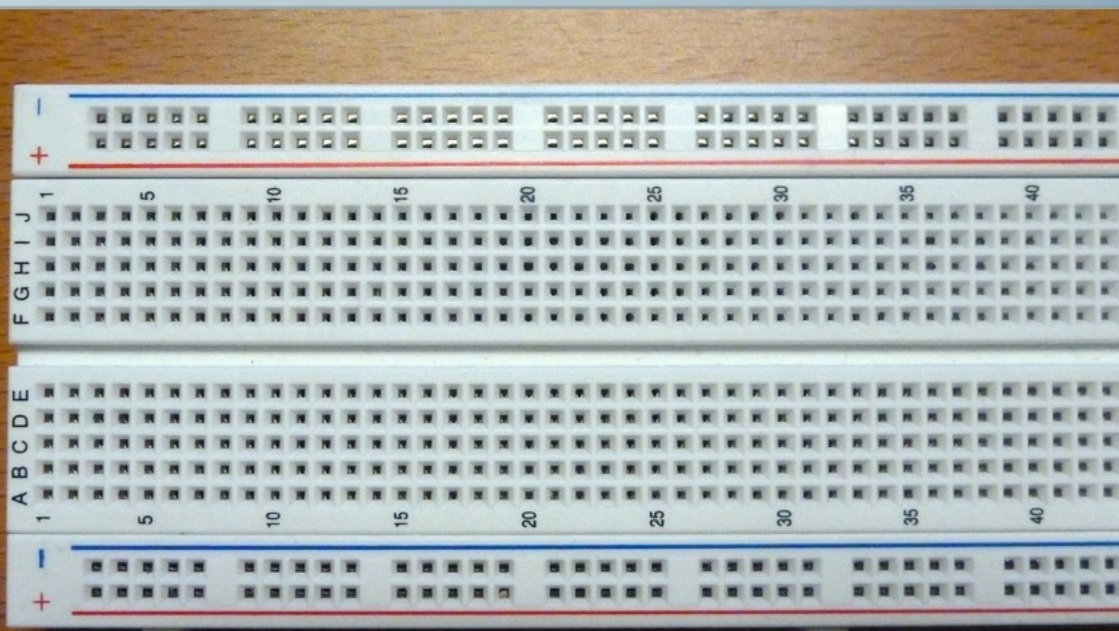


Circuit Practicalities: Wiring Things Up

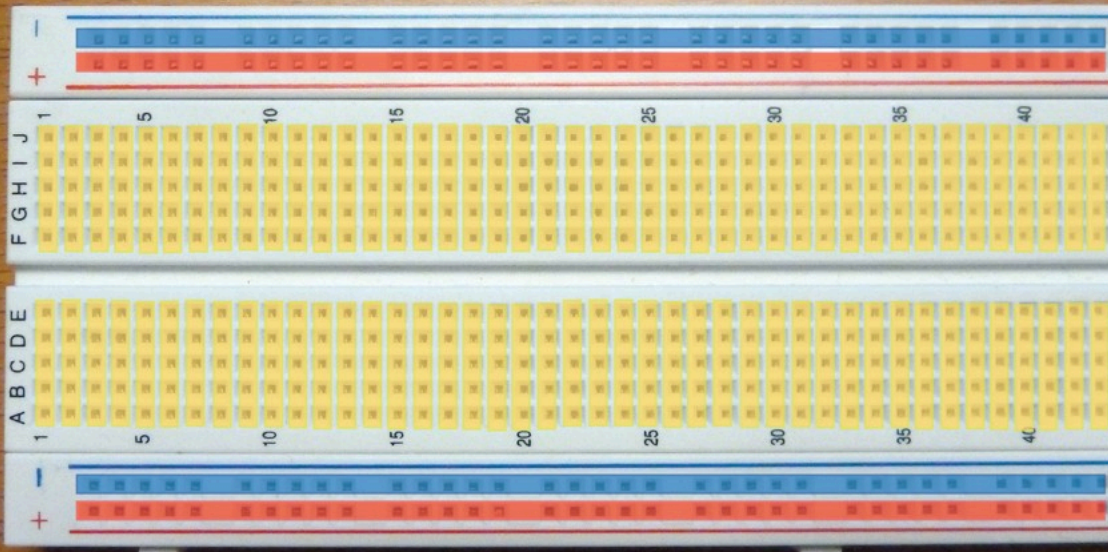
- Schematics describe the logical connections between components



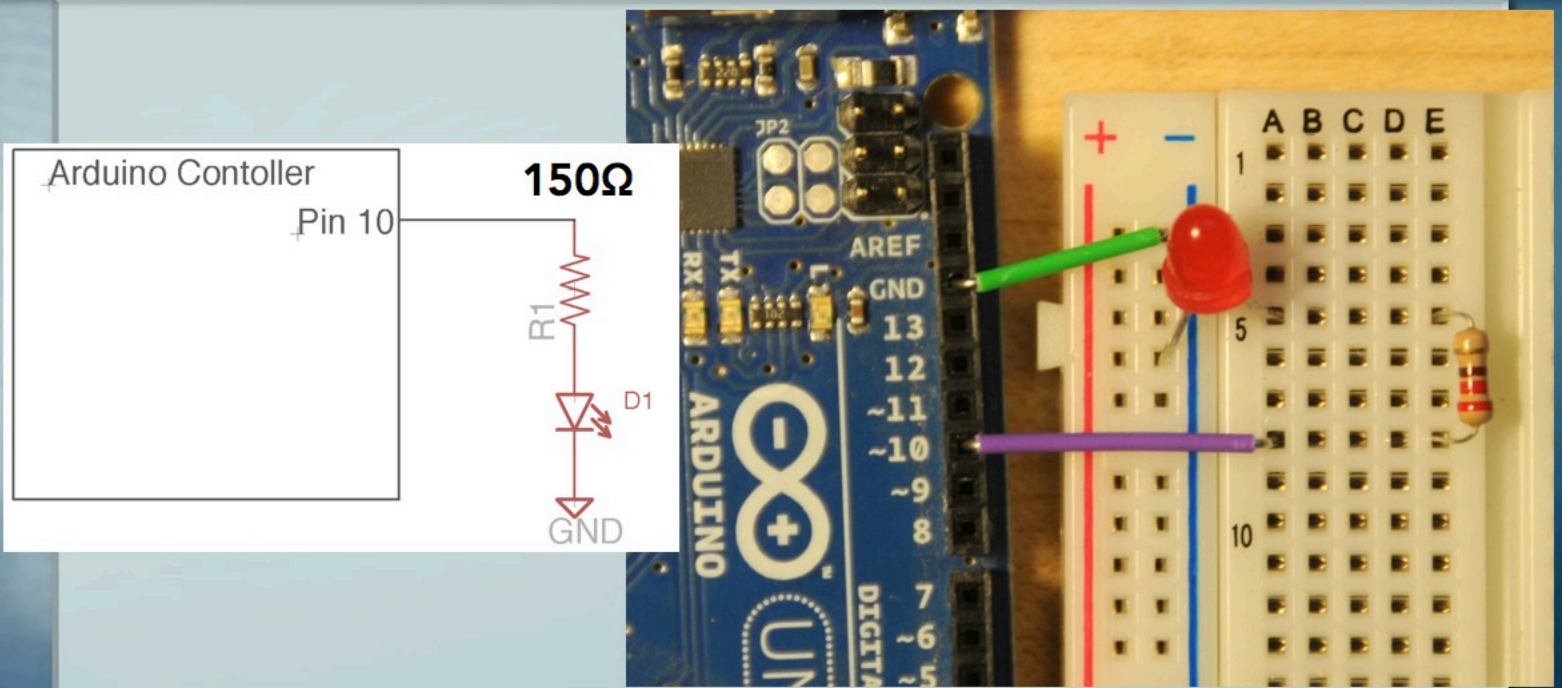
"Solderless Breadboard"



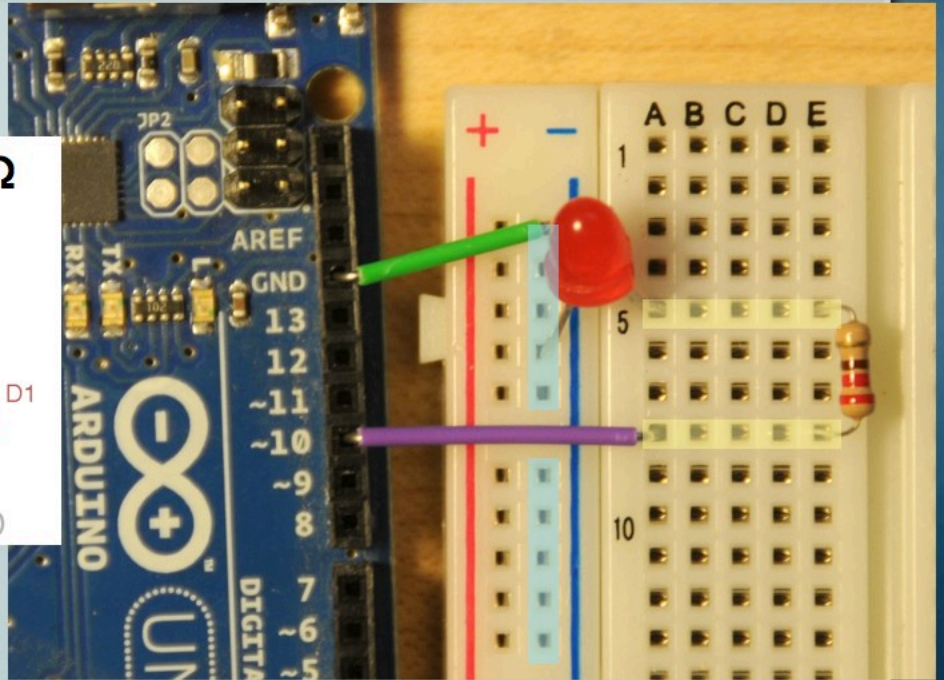
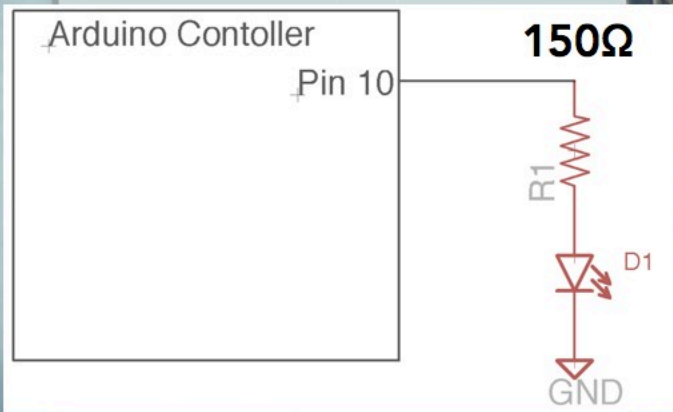
"Solderless Breadboard"



Circuit Practicalities: Wiring Things Up

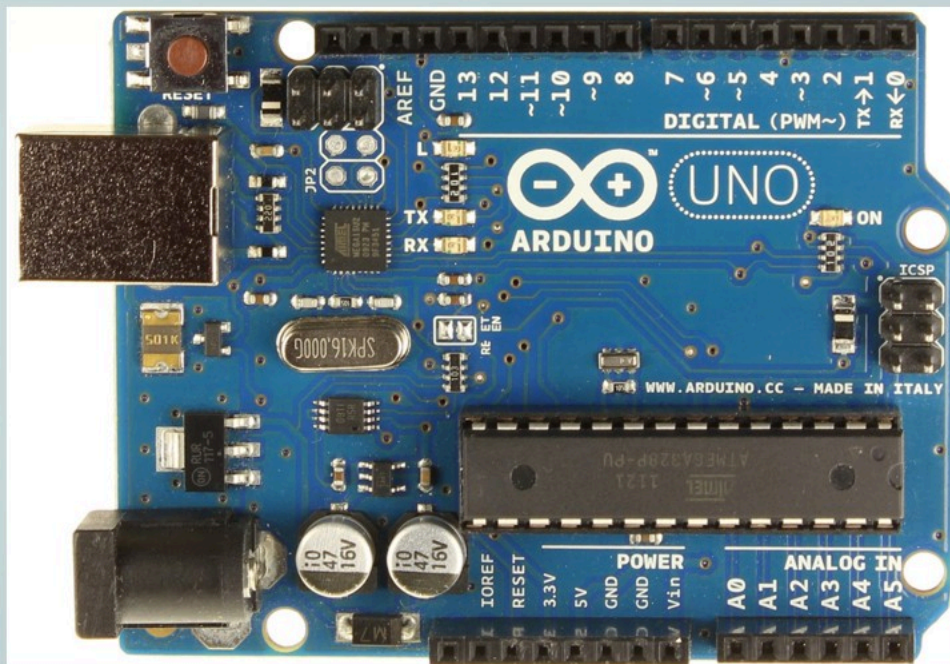


Circuit Practicalities: Wiring Things Up



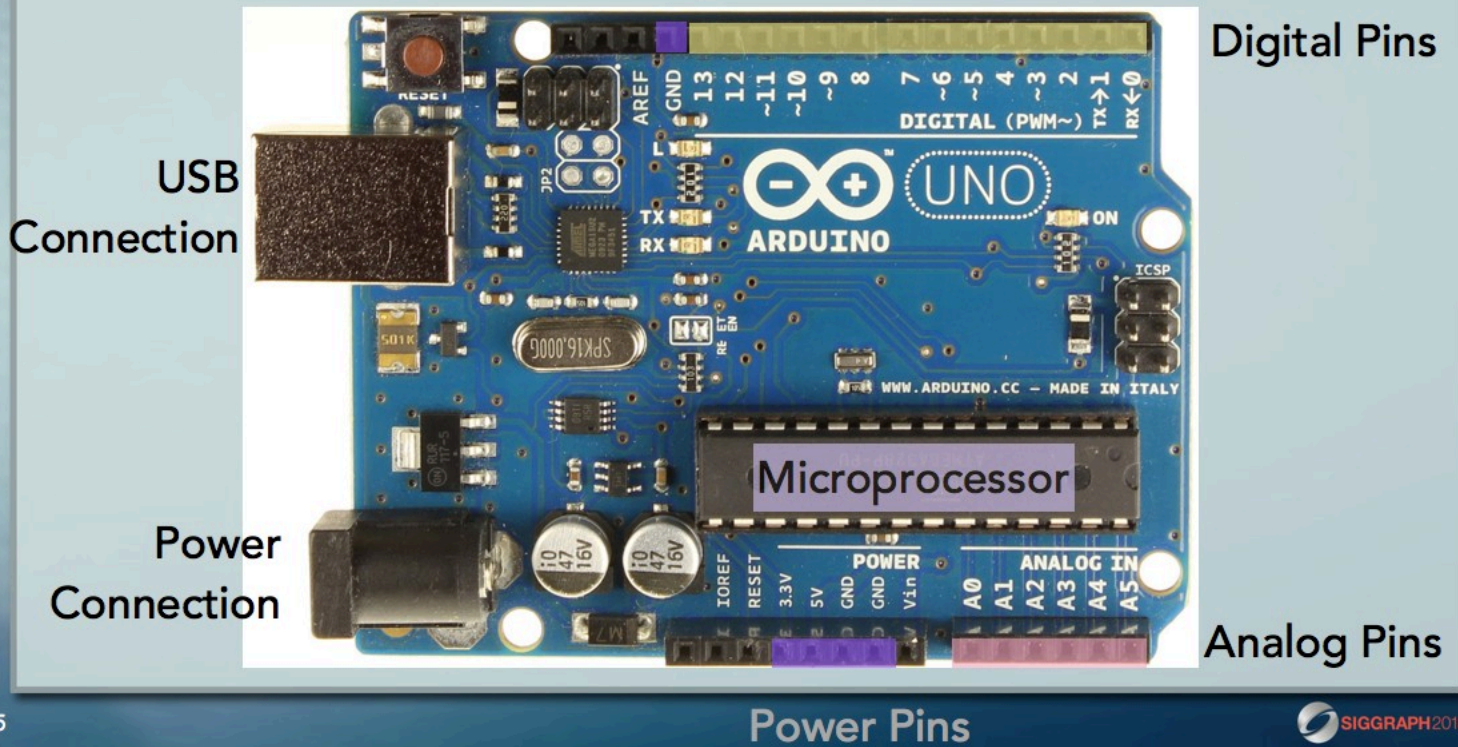
33

Circuit Practicalities: Arduino



34

Circuit Practicalities: Arduino



35

SIGGRAPH2013

Software Platform: Arduino

- www.arduino.cc
 - Simple open source IDE
 - Arduino code is really C/C++
 - avr-gcc is the back end

```
Blink | Arduino 1.0.1
/*
 * Blink
 * Turns on an LED on for one second, then off for one second, repeatedly.
 *
 * This example code is in the public domain.
 */
// Pin 13 has an LED connected on most Arduino boards.
// give it a name:
int led = 13;

// the setup routine runs once when you press reset:
void setup() {
  // initialize the digital pin as an output.
  pinMode(led, OUTPUT);
}

// the loop routine runs over and over again forever:
void loop() {
  digitalWrite(led, HIGH); // turn the LED on (HIGH is the voltage level)
  delay(1000);             // wait for a second
  digitalWrite(led, LOW);  // turn the LED off by making the voltage LOW
  delay(1000);             // wait for a second
}
```

1 Arduino Duemilanove w/ ATmega328 on /dev/tty.usbserial-A9007TW7

SIGGRAPH2013

SW/HW interface: Arduino

- Two required functions

```
void setup(){...} // Runs once at startup
```

```
void loop(){...} // Loops forever after setup()
```

- Standard(ish) C/C++ data types

- Boolean (1 bit)

- char (signed 8 bits), byte (unsigned 8 bits)

- int (16 bits), long (32 bits)

- float (32 bits), double (32 bits)

SW/HW interface: Arduino

- Physical Computing Essentials!

```
pinMode(pinNumber, mode); // declare a pin INPUT or OUTPUT
```

```
digitalRead(pinNumber); // read the HIGH/LOW status of pin
```

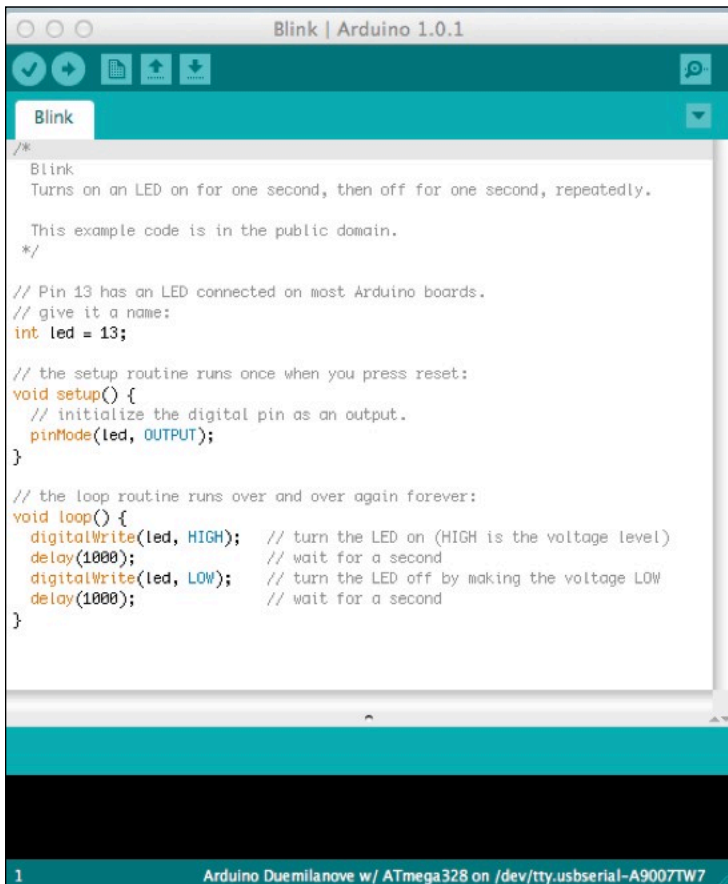
```
digitalWrite(pinNumber, value); // force a pin HIGH/LOW
```

```
analogWrite(pinNumber, value); // use PWM to get intermediate vals
```

```
analogRead(pinNumber); // read analog pin through ADC
```

•Other Helpful Physical Computing Stuff...

```
delay(ms);           // delay for ms milliseconds
millis();            // return total milliseconds since program start
Serial.begin(baud);  // set up serial communication to host
Serial.print(val);   // print var on monitor (number, char, or string)
Serial.println(val); // print with line feed
random(min, max);    // return random between min, max-1
map(val, fromLo, fromHi, toLo, toHi); // interpolate value to range
constrain(val, lo, hi); // constrain value to a range
```



```
Blink | Arduino 1.0.1
Blink
Turns on an LED on for one second, then off for one second, repeatedly.

This example code is in the public domain.
*/

// Pin 13 has an LED connected on most Arduino boards.
// give it a name:
int led = 13;

// the setup routine runs once when you press reset:
void setup() {
  // initialize the digital pin as an output.
  pinMode(led, OUTPUT);
}

// the loop routine runs over and over again forever:
void loop() {
  digitalWrite(led, HIGH); // turn the LED on (HIGH is the voltage level)
  delay(1000);             // wait for a second
  digitalWrite(led, LOW);  // turn the LED off by making the voltage LOW
  delay(1000);             // wait for a second
}

1 Arduino Duemilanove w/ ATmega328 on /dev/tty.usbserial-A9007TW7
```

Example: Blink

```
int led = 13;

void setup() {
  pinMode(led, OUTPUT);
}

void loop() {
  digitalWrite(led, HIGH);
  delay(1000);
  digitalWrite(led, LOW);
  delay(1000);
}
```

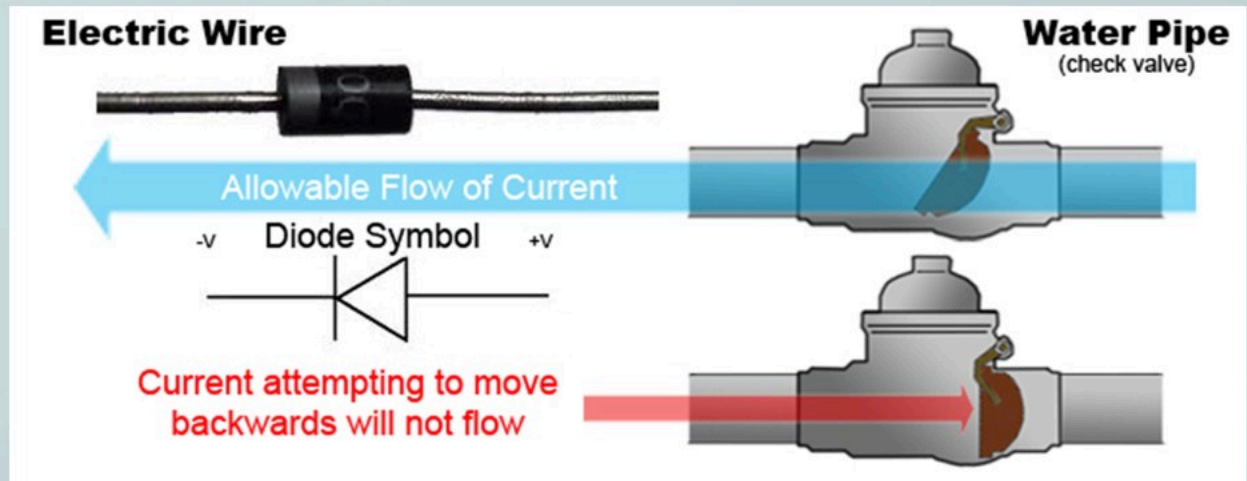
Lights!

LEDs

Diodes

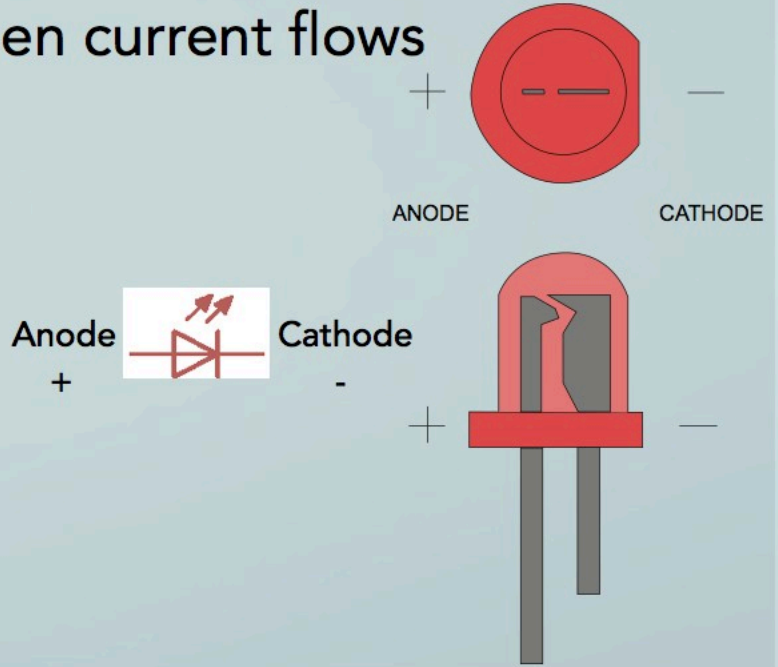
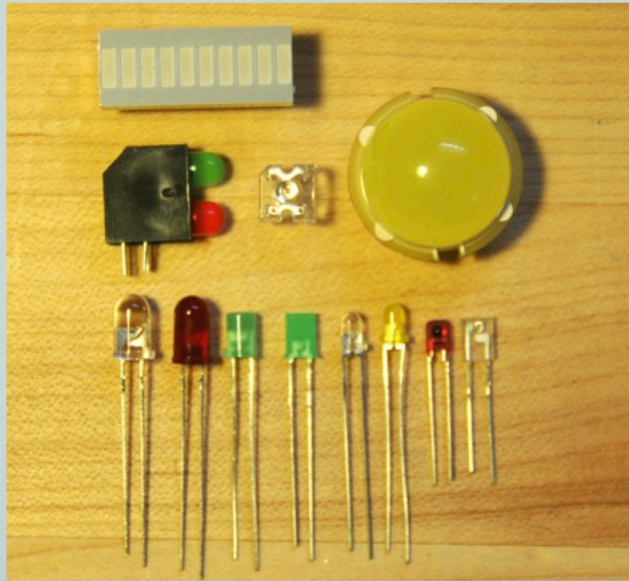
- One-way valves for current

Cathode  Anode



Light Emitting Diodes (LEDs)

- Diodes that light up when current flows



43

SIGGRAPH2013

LED Practicalities

- Diodes have a “forward voltage” or “diode drop”
 - Typically V_f is around 0.7v for a diode, and 1.5v to 3.0v for an LED
- Diodes also have a current limit
 - Typically 20mA for an LED
 - If you don’t limit the current, they’ll burn out



44

CC-BY-SA-2.5:GRL Graffiti research Lab <http://graffitiresearchlab.com>

SIGGRAPH2013

Current-Limiting Resistor: Big Idea #10



Remember, KCL says that all series connected components see the same current!

Arduino Controller

Pin 10

$$R = (V_{dd} - V_f) / I_{desired}$$



-Assume Pin10 can supply 5v

-Assume LED V_f is 2.0v

- $(5v - 2v) = 3v$ remaining for R1

-We want 20mA

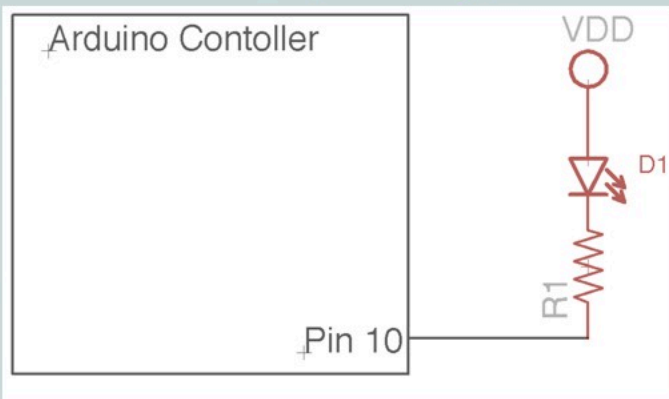
- $R = V/I = 3v / .020A$

- $R = 150\Omega$

-In practice, $150\Omega - 330\Omega$ will work

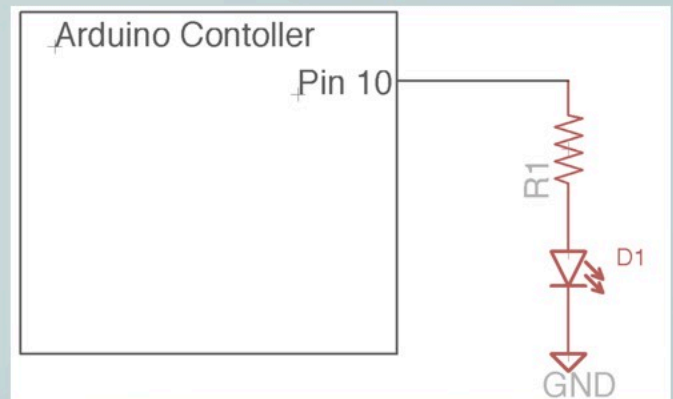
Current Sink vs. Source

Sink means "pull to ground"



LED lights up when Pin 10 is LOW (0v)

Source means "pull to Vdd"

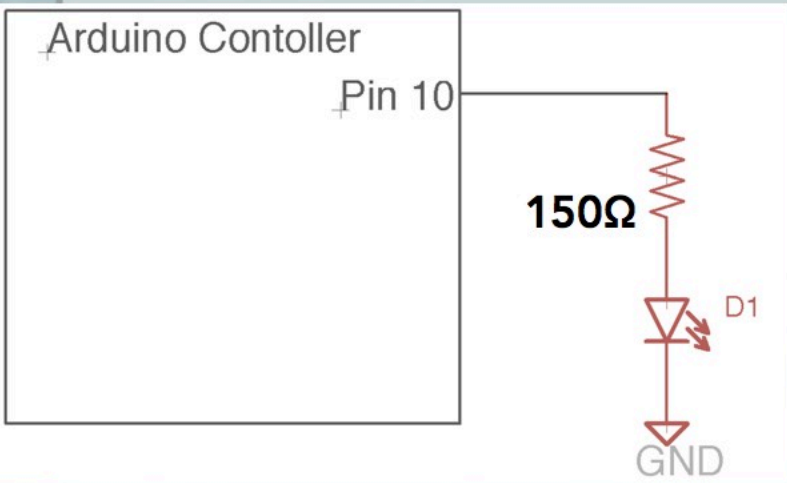


LED lights up when Pin 10 is HIGH (5v)

Arduino digital pins source/sink a maximum of 40mA/pin

Arduino provides a max of 250mA total to all pins

Example Revisited: Blink



```
int led = 10;

void setup() {
  pinMode(led, OUTPUT);
}

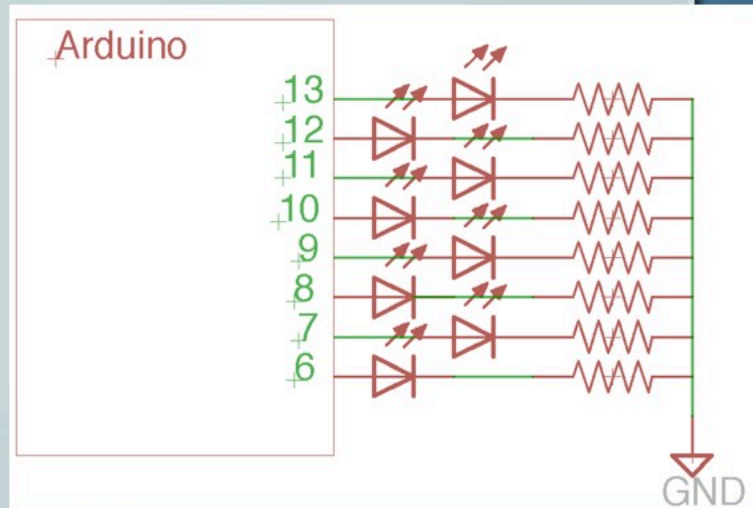
void loop() {
  digitalWrite(led, HIGH);
  delay(1000);
  digitalWrite(led, LOW);
  delay(1000);
}
```

47

Eight LEDs

- Each LED needs its own current-limiting resistor!

```
int leds[] = {6,7,8,9,10,11,12,13};
void setup(){
  for (int i=0; i<8; i++){
    pinMode(leds[i], OUTPUT);
  }
}
void loop(){
  \\... set them HIGH and LOW
  \\ remember to delay(ms)!
}
```

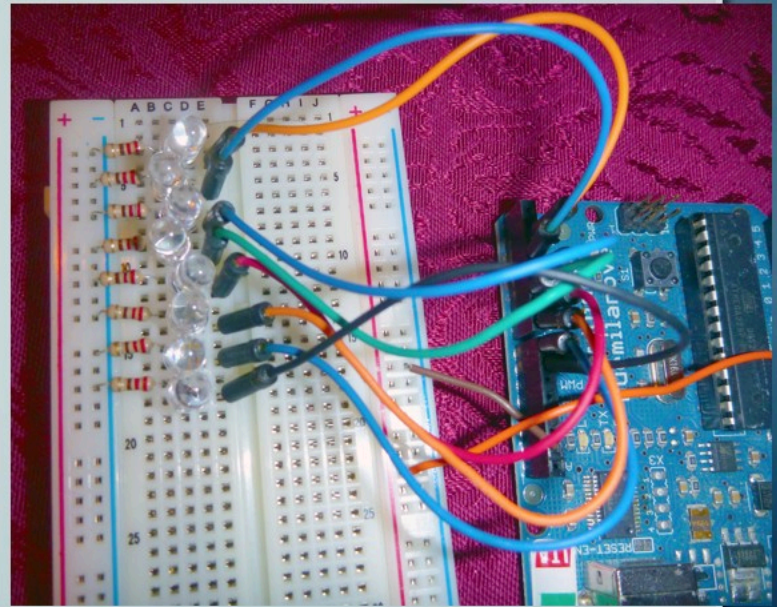


48

Eight LEDs

- *Each LED needs its own current-limiting resistor!*

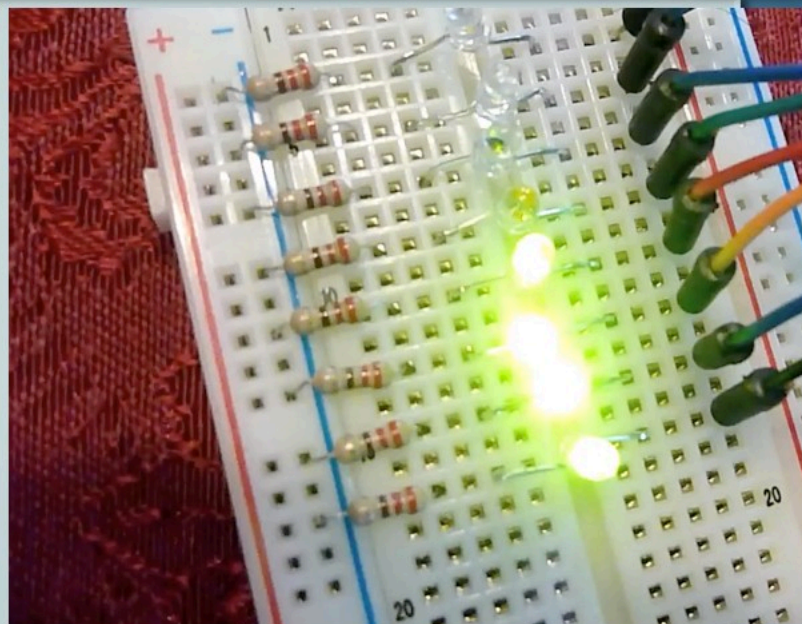
```
int leds[] = {6,7,8,9,10,11,12,13};
void setup(){
  for (int i=0; i<8; i++){
    pinMode(leds[i], OUTPUT);
  }
void loop(){
  \\... set them HIGH and LOW
  \\ remember to delay(ms)!
}
```



Eight LEDs

- *Each LED needs its own current-limiting resistor!*

```
int leds[] = {6,7,8,9,10,11,12,13};
void setup(){
  for (int i=0; i<8; i++){
    pinMode(leds[i], OUTPUT);
  }
void loop(){
  \\... set them HIGH and LOW
  \\ remember to delay(ms)!
}
```



Dimming an LED

- LEDs are either all-on or all-off

- But, they go on and off really fast
- So if you flash them fast enough, they still look on, but dimmer

- **"Pulse Width Modulation" (PWM)**

```
analogWrite(pin, value); // value between 0-255  
// Must be a "PWM pin"
```

D: 0%

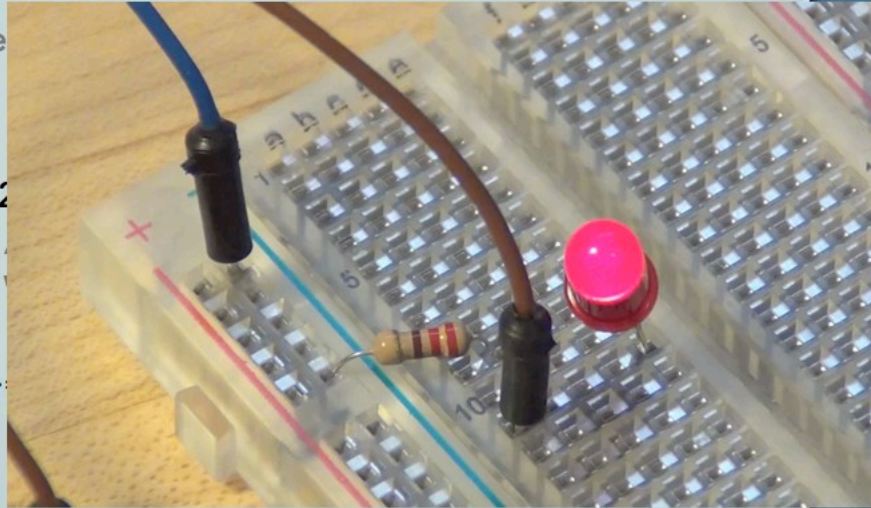


Code Example: Fading

```
int ledPin = 9; // LED connected to digital pin 9 (a PWM pin...)  
  
void setup() {  
  // nothing happens in setup - pins are OUTPUT by default  
}  
  
void loop() {  
  for (int fadeValue = 0 ; fadeValue <= 255; fadeValue +=5) { // fade 0-255 in steps of 5  
    analogWrite(ledPin, fadeValue); // Apply the fade value (brightness) to the LED  
    delay(30); // Wait 30ms so you can see the effect  
  }  
  for (int fadeValue = 255 ; fadeValue >= 0; fadeValue -=5) { // fade back down  
    analogWrite(ledPin, fadeValue);  
    delay(30);  
  }  
}
```

Code Example: Fading

```
int ledPin = 9; // LED connected to digital pin 9 (a PWM pin...)  
  
void setup() {  
  // nothing happens in setup - pins are  
}  
  
void loop() {  
  for (int fadeValue = 0 ; fadeValue <= 255 ; fadeValue++) {  
    analogWrite(ledPin, fadeValue); //  
    delay(30); //  
  }  
  for (int fadeValue = 255 ; fadeValue > 0 ; fadeValue--) {  
    analogWrite(ledPin, fadeValue);  
    delay(30);  
  }  
}
```



53

Driving LOTS of LEDs

- Arduino only has 14 digital I/O pins

- You could connect multiple LEDs to each pin
- Web tools like ledcalculator.net
- Remember current limits!

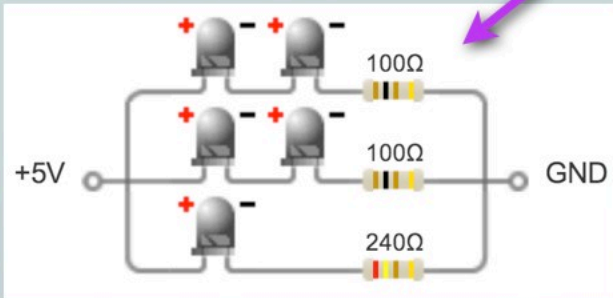
- This circuit has a problem!!!

Power supply voltage (V): 5 ?
LED voltage drop (V): 1.9 ?
LED current rating (mA): 18 ?
Number of LEDs: 5 ?
Output: Wiring Diagram Schematic

54

Driving LOTS of LEDs

- Arduino digital pins source/sink max of 40mA
 - KCL says $18\text{mA} / \text{branch} = 54\text{mA}$
 - $13\text{mA} / \text{LED max}$ for 39mA total



Power supply voltage (V): ?

LED voltage drop (V): ?

LED current rating (mA): ?

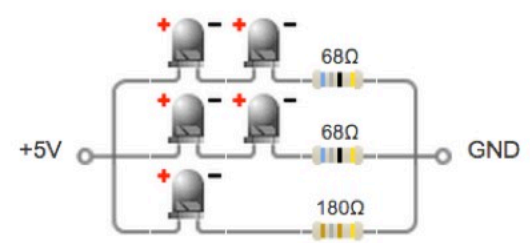
Number of LEDs: ?

Output:

Wiring Diagram

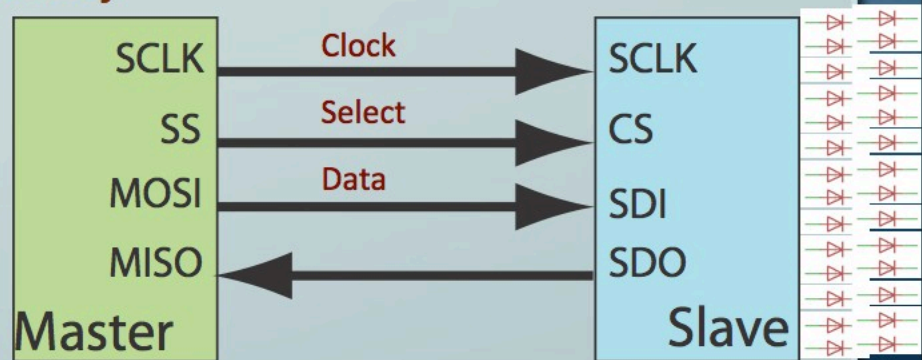
Schematic

Print



Driving Lots of *Individual* LEDs

- A variety of external chips to help
- Most use Serial Peripheral Interface (SPI)
 - Three- or four-wire serial communication protocol
 - **Data** bits are shifted out serially on each **Clock**
 - Captured in external device on **Select**



Driving Lots of *Individual* LEDs

- A variety of external chips to help
- Most use Serial Peripheral Interface (SPI)
 - **Constant Current Outputs!!!!**
 - **STP08DP05 - 8 LEDs**
 - small package
 - **MAX 7219 - 64 LEDs**
 - Automatic cycling
 - **TLC 5940 - 16 LEDs**
 - Individual PWM (4096 levels)

57

Questions to Ask about LEDs

- What is the V_f “forward voltage” (voltage drop) of your LED?
 - Important for sizing the current-limiting resistor
 - Check manufacturer, or measure
- How much current can your LED handle?
 - If you don't have the spec, assume 20mA



58

Questions to Ask about LEDs



- What current-limiting resistor to use?

- Use “LED Calculator”, or formula $R=(V_{dd}-V_f)/I_{desired}$
- Or use constant current driver, and set according to data sheet

- How bright would you like your LED?

```
digitalWrite(pin);           // for full brightness  
analogWrite(pin,value);     // for dimming (value between 0-255)
```

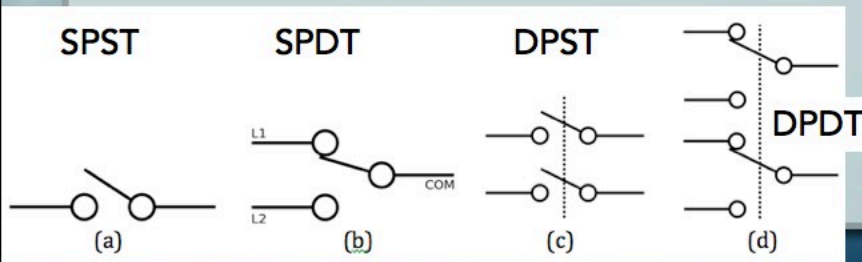
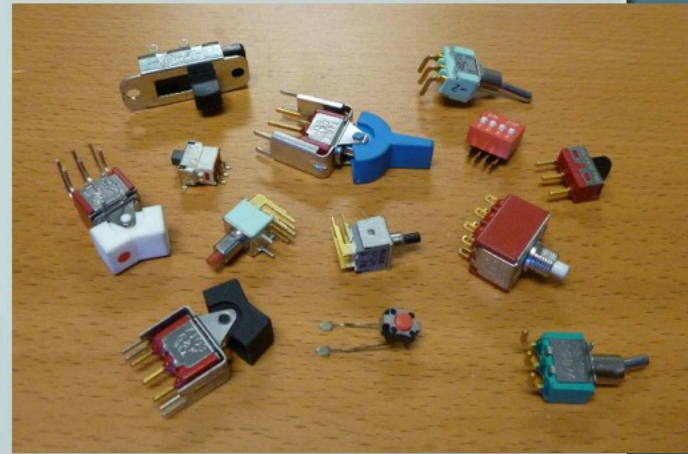
Speed!

Sensors

Sensing and controlling the physical world...

- The simplest sensor - sense **on** or **off**

- Throws and Poles
- Make a connection to Vdd for **HIGH**, GND for **LOW**
- Get value with `digitalRead(pin);`

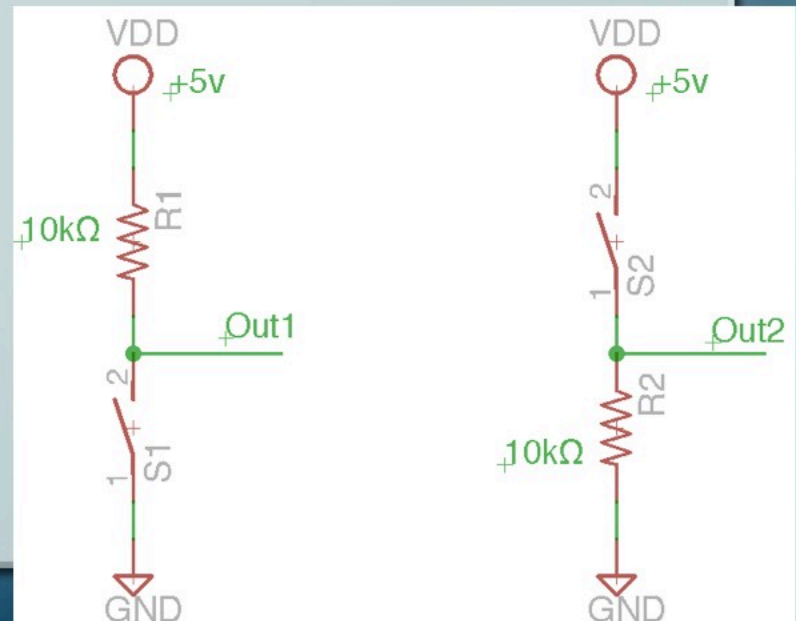


Current-Limiting in Switches: Big Idea #12



- $V = IR$ What if R becomes really low?

- Switches always need a current-limiting resistor!
- Resistance be quite large: very little current is required
- 10kΩ is a common value

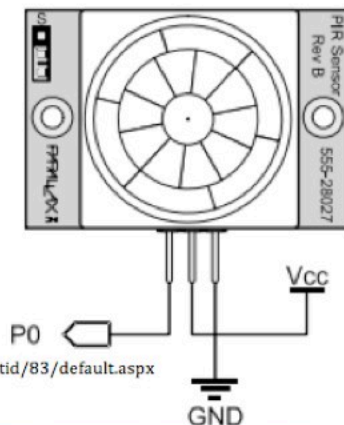
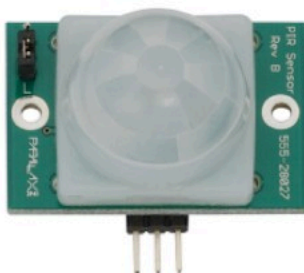


Ridiculously complicated light switch...

```
const int buttonPin = 2; // the number of the pushbutton pin
const int ledPin = 13; // the number of the built-in LED pin
int buttonState = 0; // variable for reading the pushbutton status
void setup() {
  pinMode(ledPin, OUTPUT); // initialize the LED pin as an output
  pinMode(buttonPin, INPUT); // initialize the pushbutton pin as an input
}
void loop(){
  buttonState = digitalRead(buttonPin); // read the state of the pushbutton value
  if (buttonState == HIGH) // check button value
    digitalWrite(ledPin, HIGH); // button is HIGH, turn LED on
  else digitalWrite(ledPin, LOW); // button is LOW, turn LED off
}
```

Motion Sensor: Passive Infrared (PIR)

- Just another type of switch
 - Switch closes when motion is sensed
 - Built-in current-limiting resistor



<http://www.parallax.com/tabid/768/productid/83/default.aspx>

Resistive Sensors

- A large class of sensors that change **resistance** based on some environmental condition
 - **Potentiometer** - turn a knob to change resistance
 - **Light sensor** - resistance based on light intensity
 - **Temperature sensor** - more heat = less resistance
 - **Distance sensor** - resistance based on closest object distance
 - **Pressure Sensor** - resistance changes as you press
 - etc...

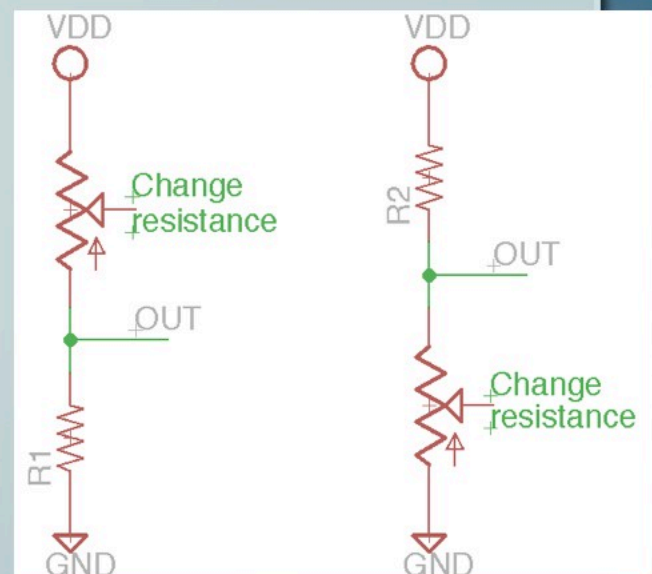
65



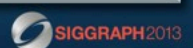
Resistive Sensors

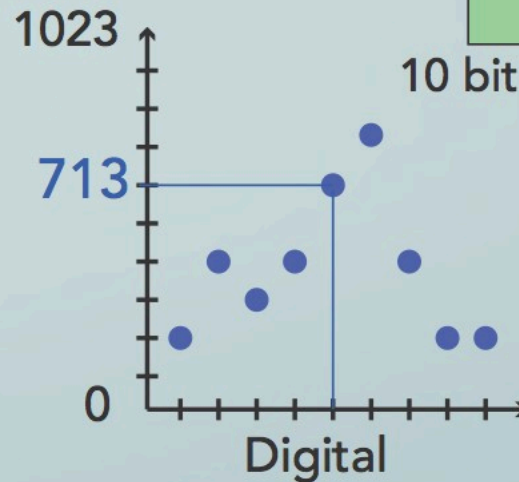
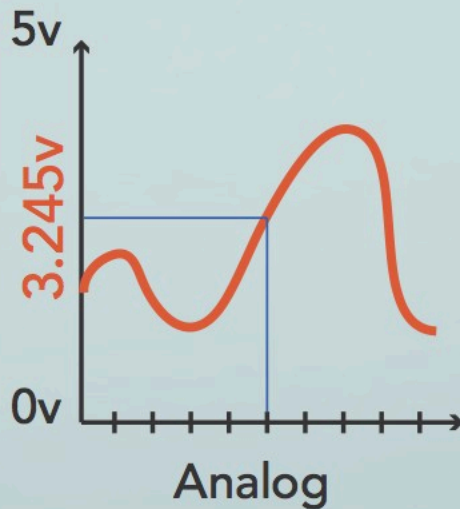
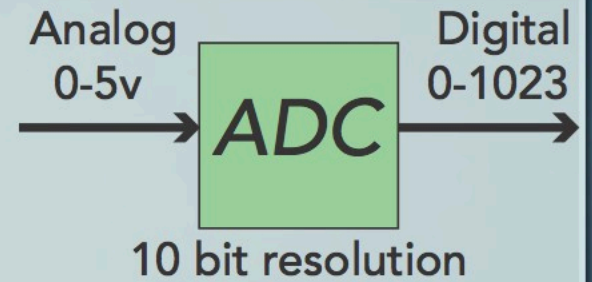
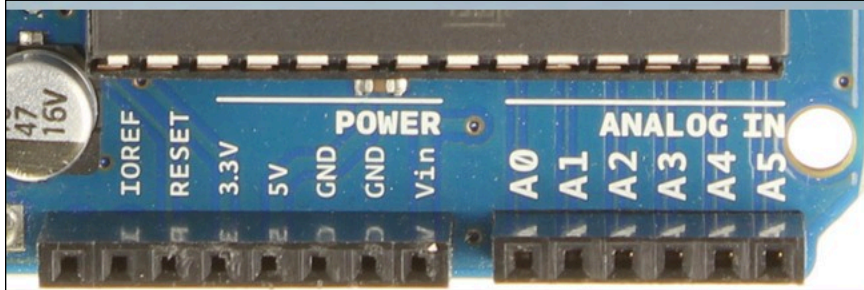
- Use to make a **voltage divider!**

- **Vout** changes as resistance changes
- Read voltage value with `analogRead(pin);` // analog pin only!
- Analog inputs go through analog to digital converter (ADC)



66





Potentiometers (Knobs)

- Direct control of variable resistance

```
int sensorPin = A2; // Analog pin 2
int ledPin = 13;
int sensorValue = 0;
void setup() {
  pinMode(ledPin, OUTPUT);
}
void loop() {
  sensorValue = analogRead(sensorPin);
  val = map(sensorValue, 0, 1023, 100, 255); // map to reasonable values
  digitalWrite(ledPin, val); // write the value to the LED
}
```

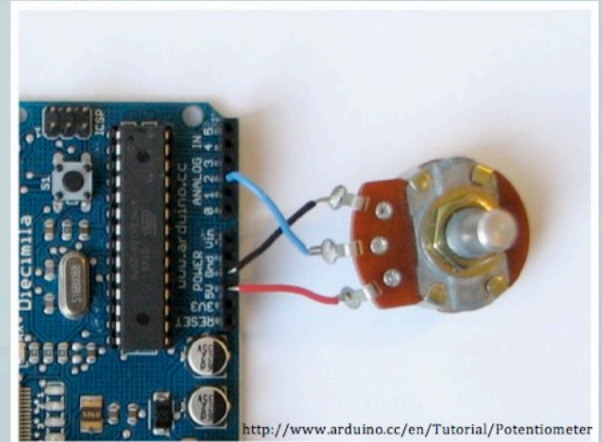


<https://learn.sparkfun.com/tutorials/voltage-dividers/applications>

Potentiometers (Knobs)

• Direct control of variable resistance

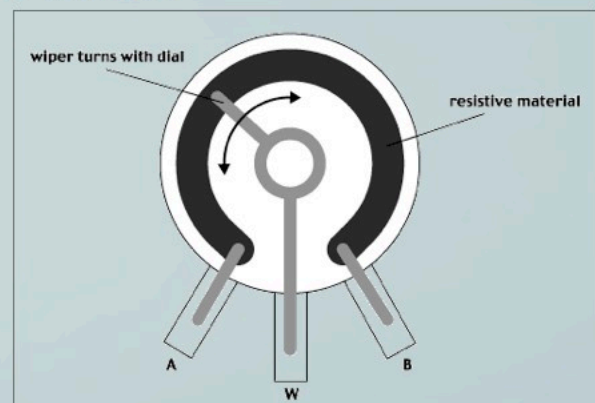
```
int sensorPin = A2; // Analog pin 2
int ledPin = 13;
int sensorValue = 0;
void setup() {
  pinMode(ledPin, OUTPUT);
}
void loop() {
  sensorValue = analogRead(sensorPin);
  val = map(val, 0, 1023, 100, 255); // map to reasonable values
  analogWrite(ledPin, val); // write the value to the LED
}
```



Potentiometers (Knobs)

• Direct control of variable resistance

```
int sensorPin = A2; // Analog pin 2
int ledPin = 13;
int sensorValue = 0;
void setup() {
  pinMode(ledPin, OUTPUT);
}
void loop() {
  sensorValue = analogRead(sensorPin);
  val = map(val, 0, 1023, 100, 255); // map to reasonable values
  analogWrite(ledPin, val); // write the value to the LED
}
```

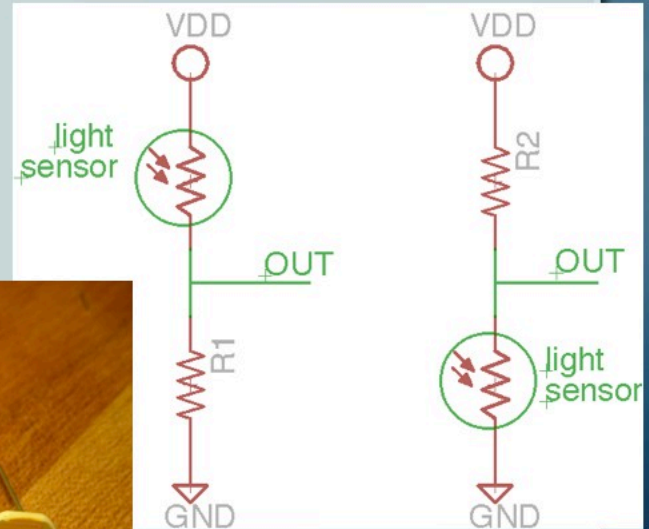
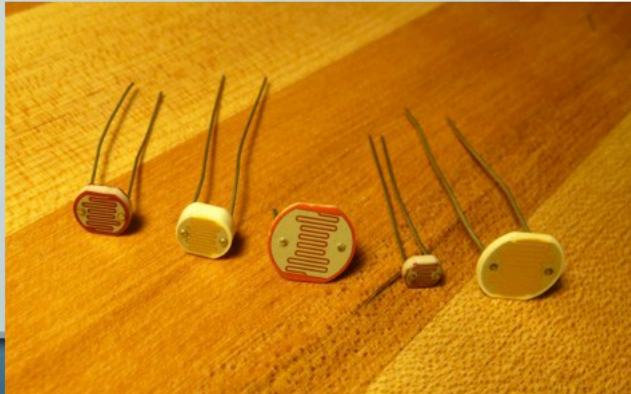


- Increased light means increased resistance

- Add a current-limiting resistor to make a voltage divider

- 2k Ω to 10k Ω ?

- Probably need to calibrate each time you change locations...



Sensor Calibration with Serial Monitor

```
int sensorPin = A0;           // input pin for the resistive sensor
int sensorValue = 0;         // variable for value coming from the sensor

void setup() {
  Serial.begin(9600);        // Init serial communication at 9600 baud
}

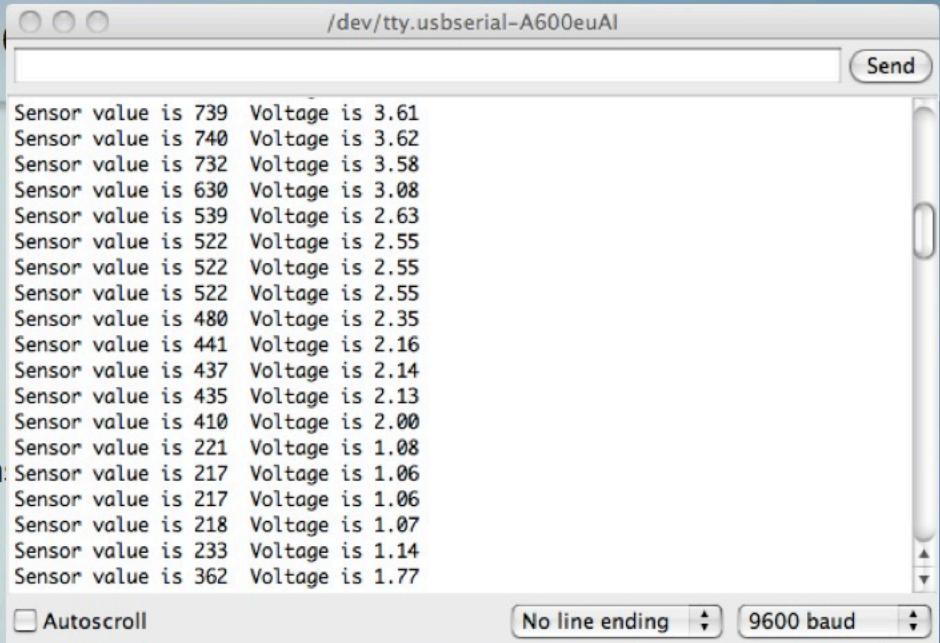
void loop() {
  sensorValue = analogRead(sensorPin); // read value from the sensor
  Serial.print("Sensor value is ");   // print a message
  Serial.print(sensorValue);          // print the value received (0-1023)
  Serial.print(" Voltage is ");       // convert to volts
  Serial.println(sensorValue * (5.0/1023.0)); // 0-5v
  delay(100);                         // wait so you don't print too much!
}
```

Sensor

```
int sensorPin = A0;
int sensorValue = 0;

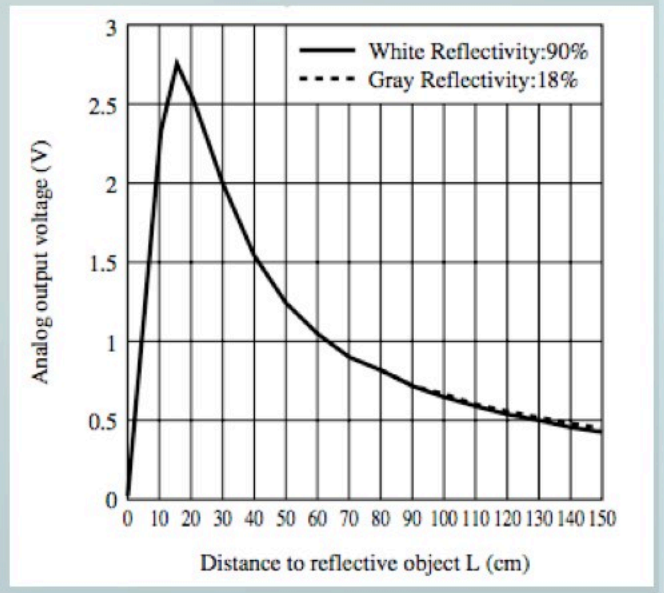
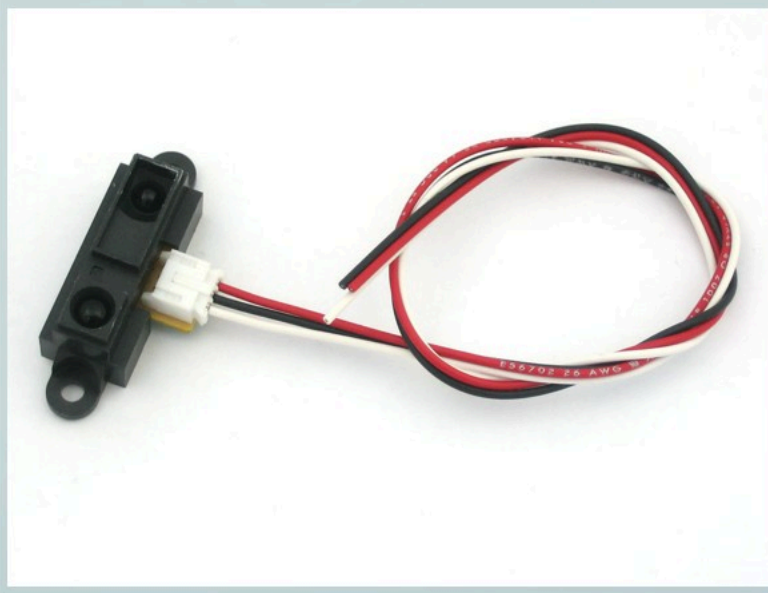
void setup() {
  Serial.begin(9600);
}

void loop() {
  sensorValue = analogRead(sensorPin);
  Serial.print("Sensor value is ");
  Serial.print(sensorValue);
  Serial.print(" Voltage is ");
  Serial.println(sensorValue * (5.0/1023.0)); // 0-5v
  delay(100); // wait so you don't print too much!
}
```



Distance Sensor

- Returns analog voltage that varies with distance



Questions to Ask about Sensors

- What type of switch do you need?
 - Normally open? Normally closed? Momentary?
 - SPST? SPDT?
 - Or make your own! Any conductive material can be used...
- Do you need a current-limiting resistor?
 - Compute how much current you're drawing...
 - Always avoid direct paths between Vdd and GND!



75

Questions to Ask about Sensors

- Potentiometers are easy - will that work?
 - Use the simplest component you can.
- Resistive sensor as voltage divider?
 - What current-limiting resistor should you use?
- Analog voltage? What range will you see?
 - Use `map(...)`; to interpolate values to a useful range



76

Questions to Ask about Sensors

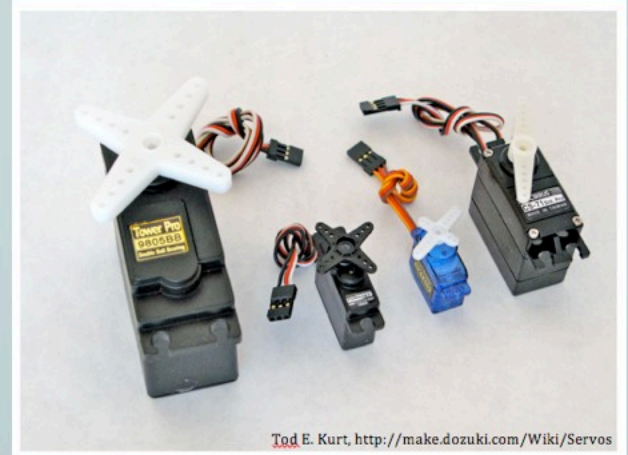
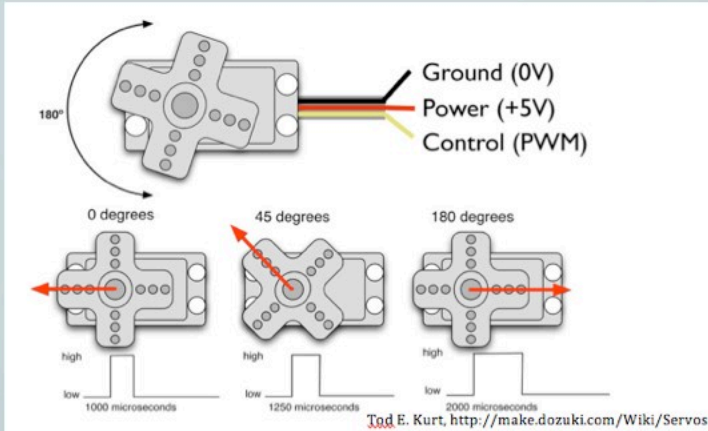


- Can you build calibration into your app?
 - Nearly all sensors need a little calibration!
- How frequently should you check the sensor value?
 - Arduino runs at 16MHz - you can probably check less frequently
 - See *blinkWithoutDelay* from the Arduino Examples...

Action!

Motors

- Easy to control - *great* for smallish objects
 - 180° precise movement, or continuous rotation



Servo Code Example

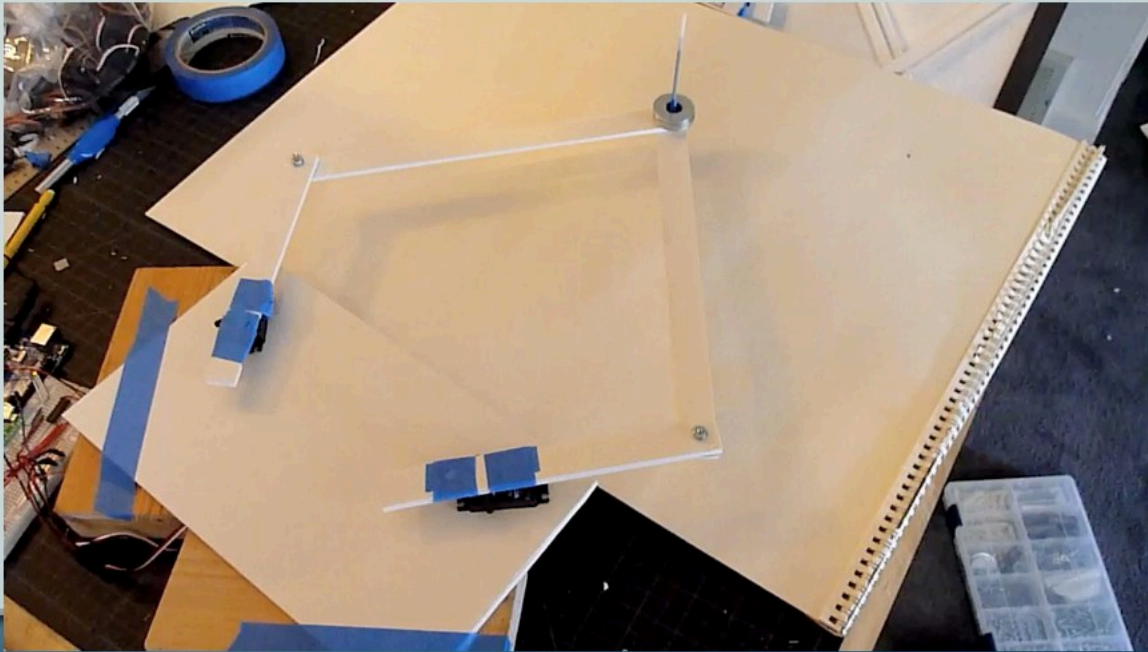
```
#include <Servo.h> // include Servo library
Servo myservo; // create servo object (one for each servo)
int potpin = A0; // analog pin for potentiometer
int val; // variable to hold pot value

void setup() {
  myservo.attach(9); // attaches the servo on pin 9 to the servo object
}

void loop() {
  val = analogRead(potpin); // read from pot (value between 0 and 1023)
  val = map(val, 0, 1023, 0, 179); // scale it for servo (value between 0 and 179)
  myservo.write(val); // set the servo position
  delay(20); // wait for the servo to get there
}
```


Example: Drawing Machine

- Two servos and two knobs!



Wednesday
Studio
2:00-3:30

81

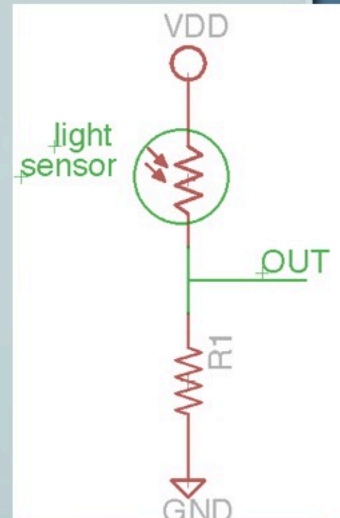


Servo / Light Sensor Code Example

```
#include <Servo.h>           // include Servo library
Servo myservo;              // create servo object (one for each servo)
int CDSPin = A0;           // analog pin for CDS light sensor
int val;                   // variable to hold pot value

void setup() {myservo.attach(9); // attaches servo to pin 9
}

void loop() {
  val = analogRead(CDSPin); // read from pot
  val = map(val, 230,950, 0, 179); // scale it for servo
  val = constrain(val, 0, 179); // keep it in range
  myservo.write(val);        // set the servo position
  delay(20);                // wait for the servo to get there
}
```



82



```
#include <Servo.h>
```

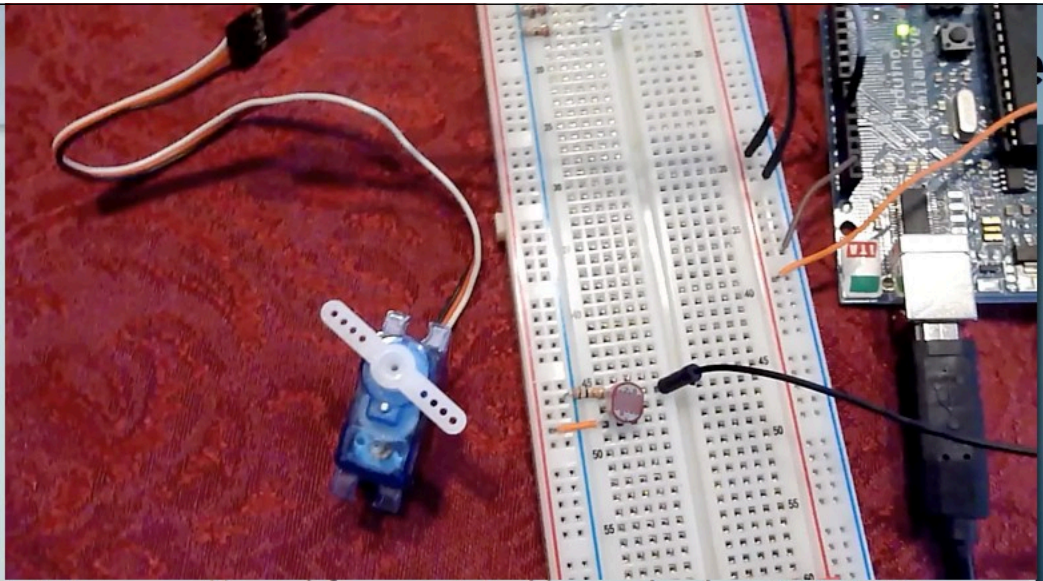
```
Servo myservo;
```

```
int CDSpin = A0;
```

```
int val;
```

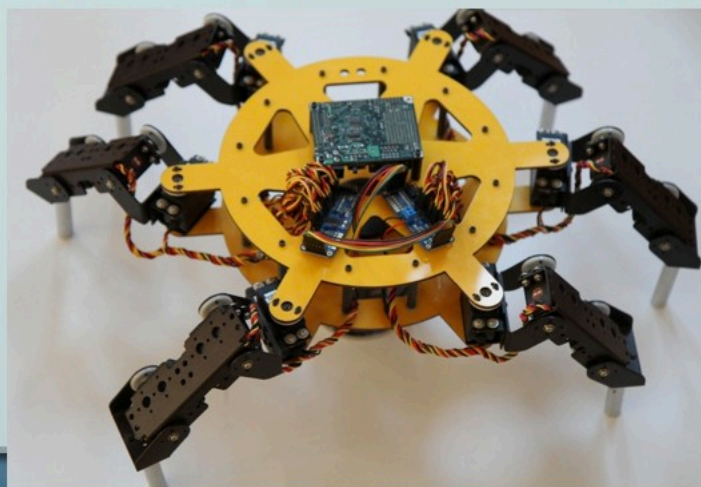
```
void setup() {  
  myservo.attach(9);  
}
```

```
void loop() {  
  val = analogRead(CDSpin); // read from pot (calibrated value 230-950)  
  val = map(val, 230,950, 0, 179); // scale it for servo (value between 0 and 179)  
  myservo.write(val); // set the servo position  
  delay(20); // wait for the servo to get there  
}
```



Servo Power

- Most hobby servos work fine on 4.8v-6v
 - One or two can be powered by Arduino 5v pin
 - More? Probably need a separate power supply



Multiple Power Supplies: Big Idea #7



- ***If you use multiple power supplies always tie their grounds together***

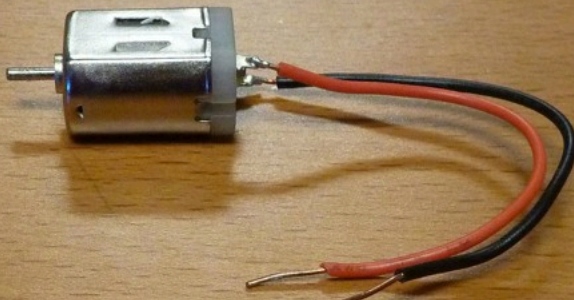
–A common 0v reference point will make everybody happier...

DC Motors

- **Rotate when you apply DC voltage**

–Reverse when you reverse the voltage

–PWM can be used to control speed

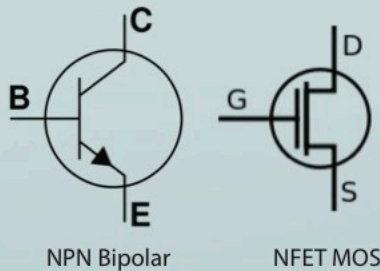
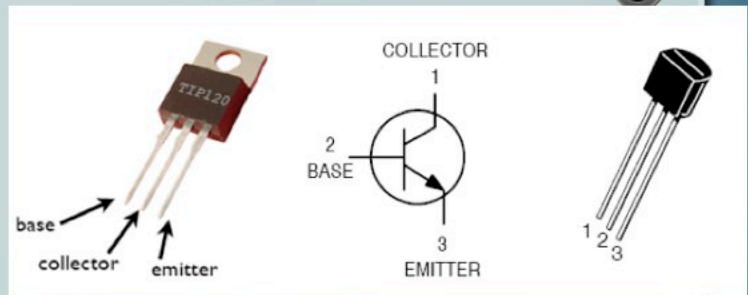


Switching Transistor : Big Idea #13



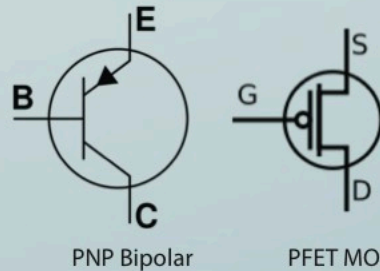
- Voltage higher than 5v?
More than 40mA current?
 - You need a **switching transistor!**
 - Like an electronic switch controlled by Arduino

TIP120



NPN Bipolar

NFET MOS



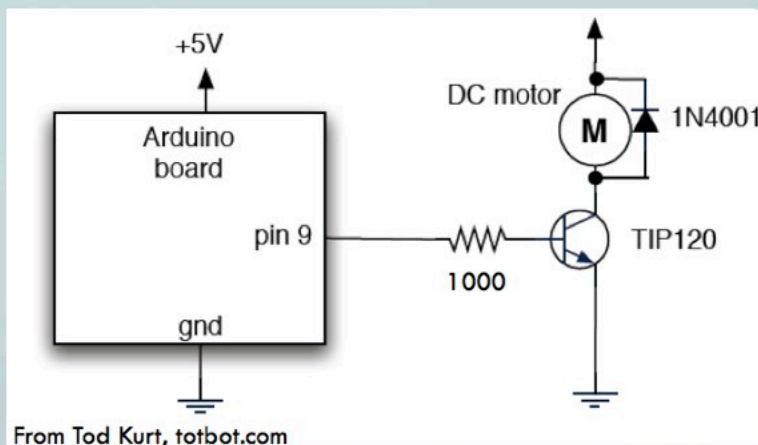
PNP Bipolar

PFET MOS

2N2222

Switching Transistors

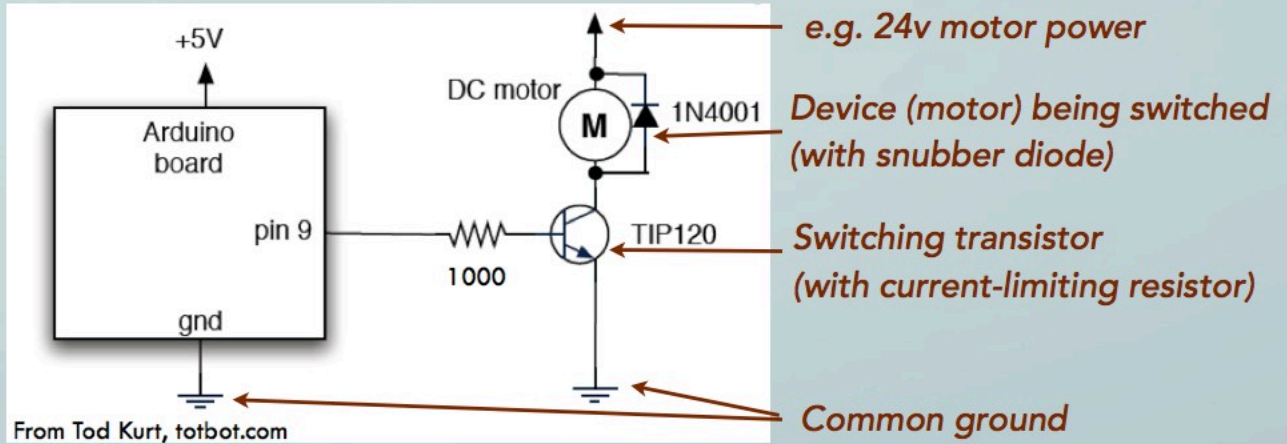
- The thing being switched can be on its own power
 - Remember the **common ground** big idea though!



From Tod Kurt, totbot.com

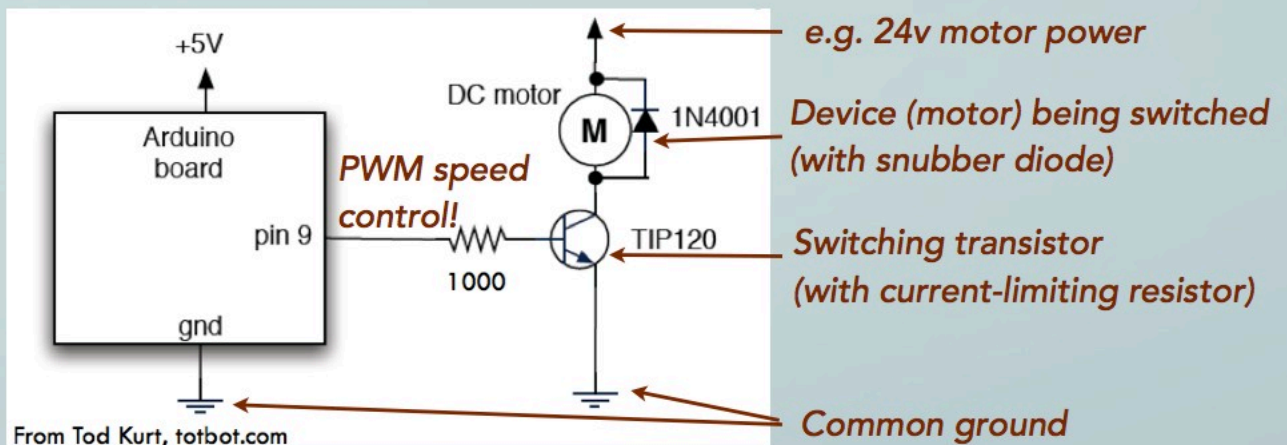
Switching Transistors

- The thing being switched can be on its own power
 - Remember the **common ground** big idea though!



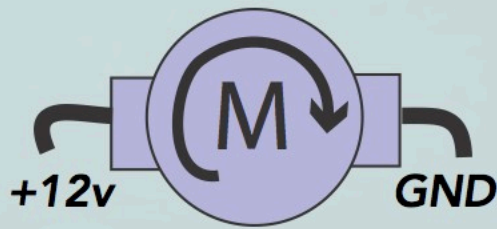
Switching Transistors

- The thing being switched can be on its own power
 - Remember the **common ground** big idea though!

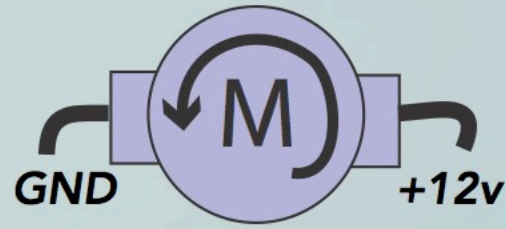


DC Motor Direction Control

- Reverse spin direction? Just reverse power!



Clockwise

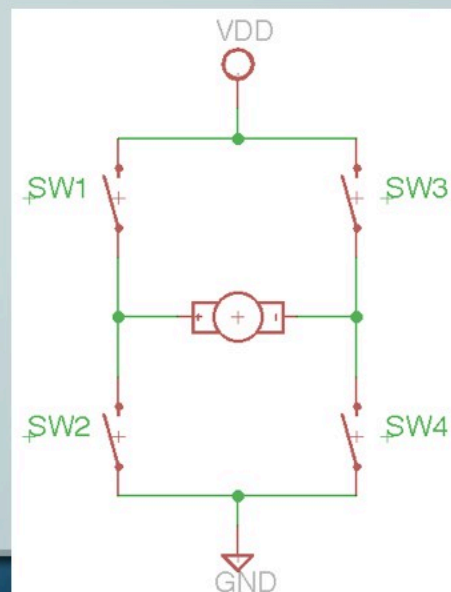


Counter Clockwise

Easier said than done?

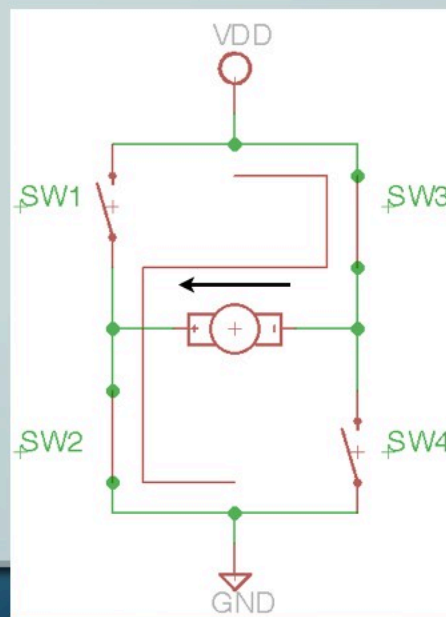
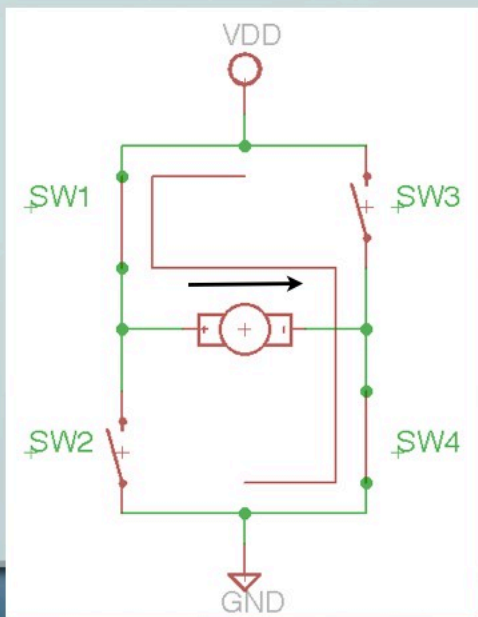
H-Bridge Circuit

- Use two pairs of switches to reverse voltage
 - Arduino controls the switches
 - Switches could be transistors or relays



H-Bridge Circuit

- Use two pairs of switches to reverse voltage



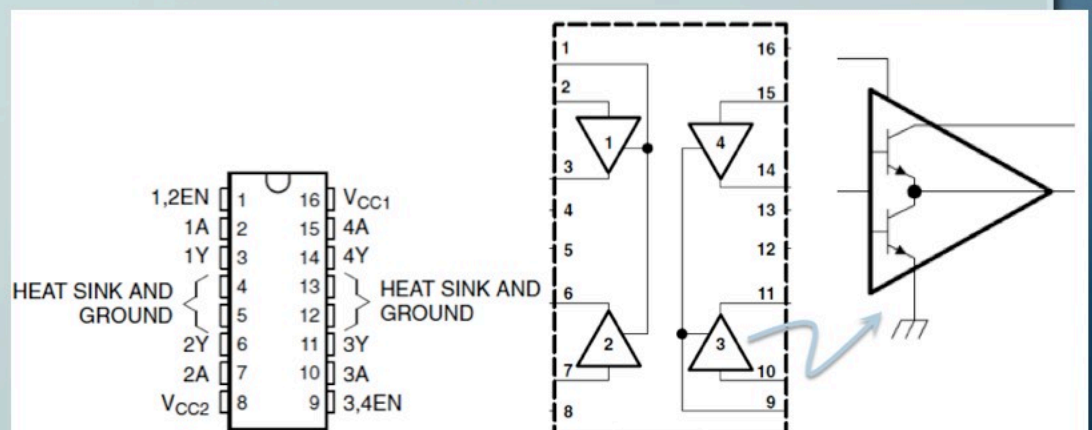
93

SIGGRAPH2013

H-Bridge Chip: L293D or SN754410

- “Quad Half H-Bridge”

- Means you can make two full H-Bridges...
- Power supply for motor (V_{cc2}) can be up to 36v
- V_{cc1} is chip's power: 5v



94

SIGGRAPH2013

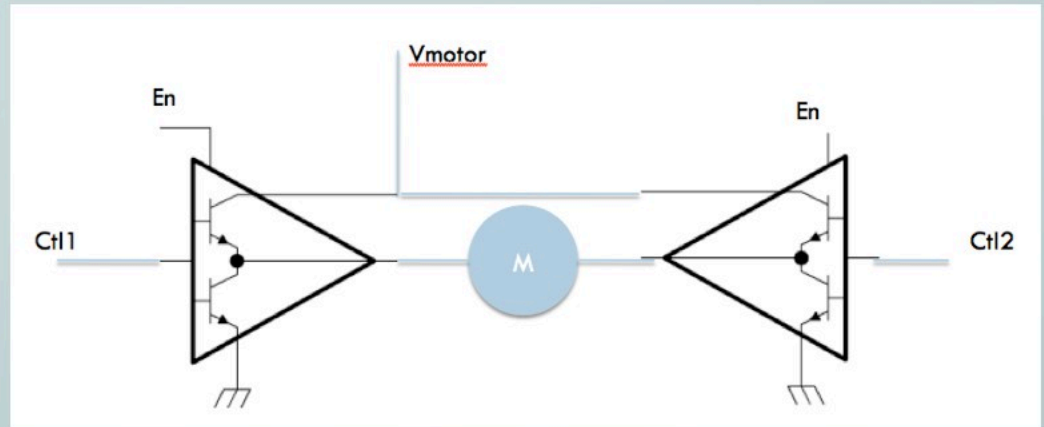
H-Bridge Chip: L293D or SN754410

- "Quad Half H-Bridge"

- Means you can make two full H-Bridges... (Here's one full H-Bridge)

- Ctl1 and Ctl2 should be either HIGH/LOW or LOW/HIGH

- **En** can be used with PWM for speed control

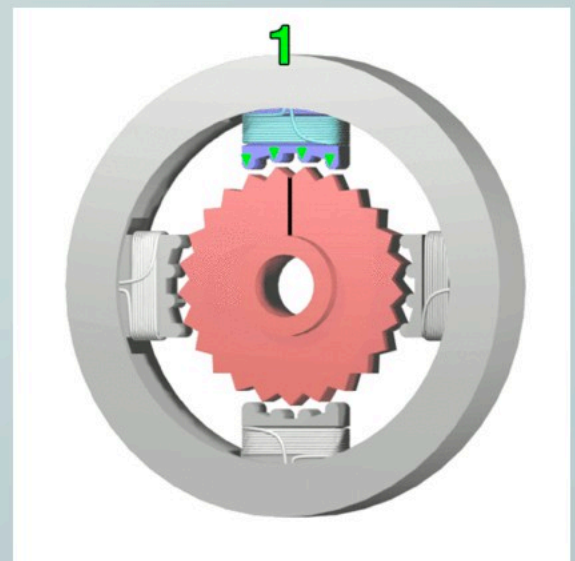


Stepper Motors

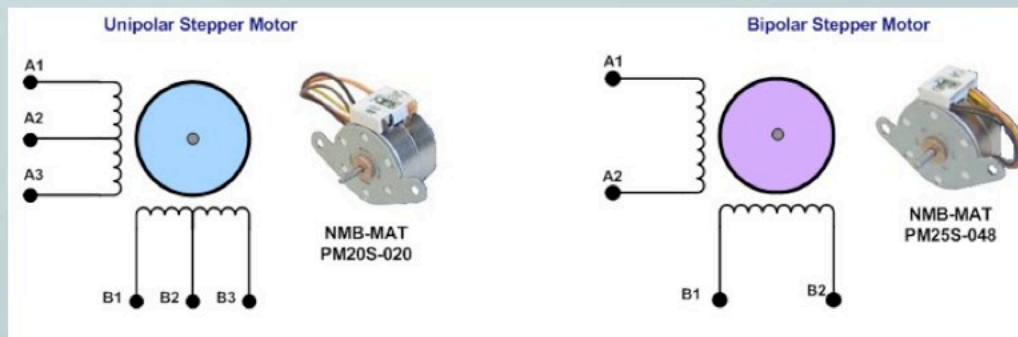
- Combine the best of servos and DC motors?

- Precise positioning and high power

- A DC motor with "steps"



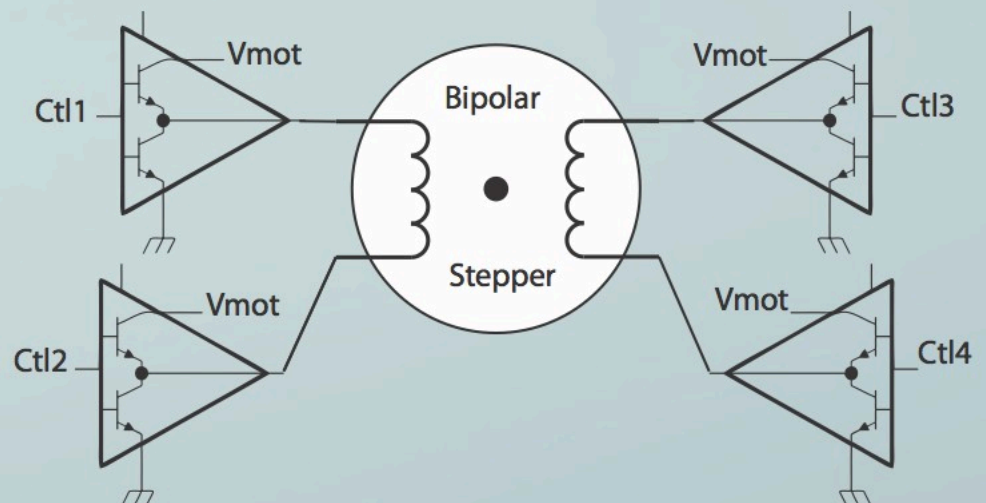
- All steppers have multiple wired connections
 - bipolar steppers have 4
 - Unipolar steppers have 5, 6, or 8
- Pulse them in the right order to make them move



Arduino Stepper Library

- Bipolar Steppers require reversible current on each coil

- Use quad half H-Bridge chip
- One H-bridge for each coil
- Example code is in Arduino IDE



Stepper Code Example

```
#include <Stepper.h>
const int stepsPerRevolution = 200;           // change this for your motor
Stepper myStepper(stepsPerRevolution, 8,9,10,11); // init stepper object

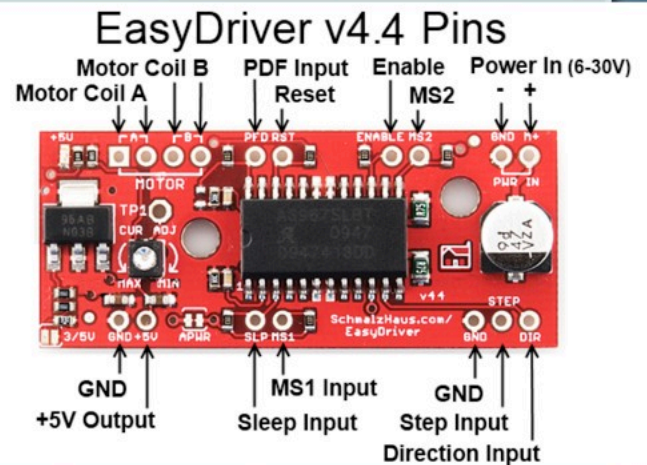
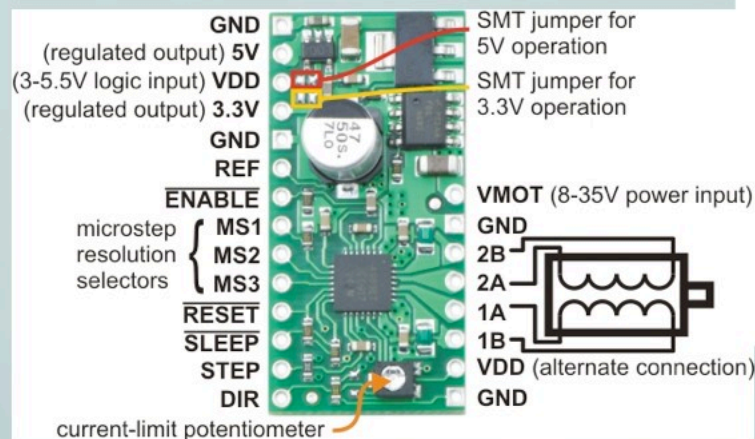
void setup() {
  myStepper.setSpeed(60);                     // set the speed at 60 rpm:
}

void loop() {
  myStepper.step(stepsPerRevolution); // step one revolution in one direction:
  delay(500);                          // give motor time to get there
  myStepper.step(-stepsPerRevolution); // step one revolution in other direction:
  delay(500);                          // give motor time to get there
}
```

Stepper Driver Chips/Boards

• Make life easier - buy a stepper driver board!

- Two control wires: **Step** and **Dir**
- These boards do the rest!

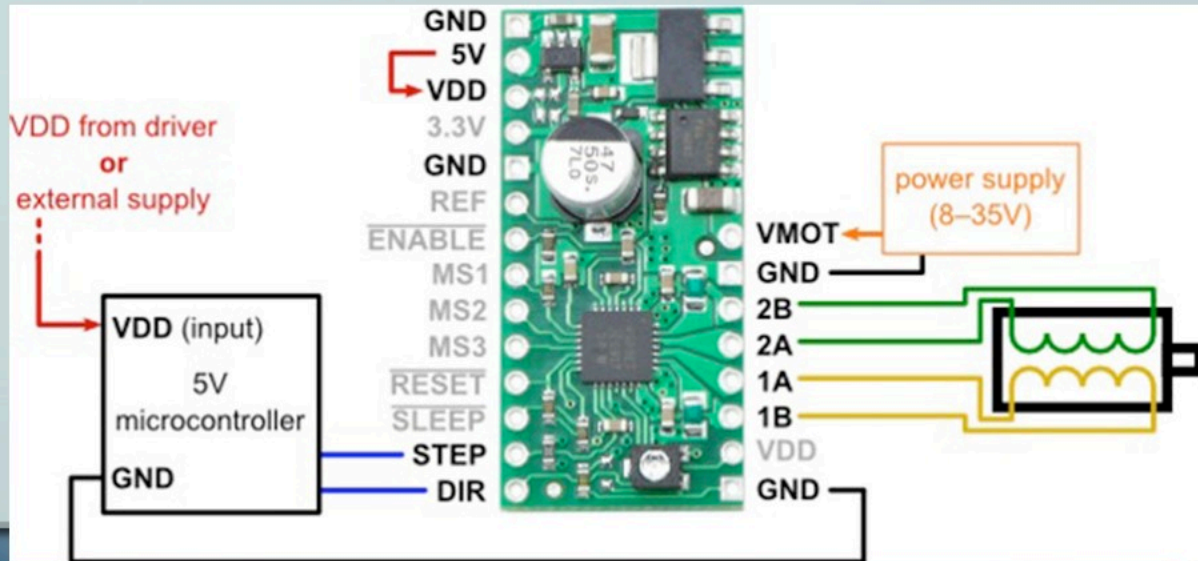


Careful - this doesn't work with built-in Arduino Stepper library!

Stepper Driver Chips/Boards

- Make life easier - buy a stepper driver board!

– Two control wires: **Step** and **Dir**



101

SIGGRAPH2013

Example Stepper Driver Code

```
#define stepPin 2
#define dirPin 3
#define usecDelay 100

void setup(){
  pinMode(stepPin, OUTPUT);
  pinMode(dirPin, OUTPUT);
}

void loop() {
  step(HIGH, 200, 100);
  step(LOW, 800, 100);
}

void step(boolean dir, int steps, int usDelay){
  digitalWrite(dirPin, dir);
  delayMicroseconds(2);
  for (int i=0; i<steps; i++){
    digitalWrite(stepPin, LOW);
    delayMicroseconds(2);
    digitalWrite(stepPin, HIGH);
    delayMicroseconds(usDelay);
  }
}
```

See also accelStepper library on the web

102

SIGGRAPH2013

- Degrees per step

- 7.5°, 3.6, 1.8°, and 0.9° are all fairly common

- 48, 100, 200, and 400 steps per revolution

- Stepper drivers allow “microstepping” if you need more resolution

- Bipolar vs. Unipolar

- Bipolar are required for the stepper drivers mentioned here

- Look for steppers with four wires...

- Rated for max amps per coil

- Volts** and Ω **per coil** are often what you get

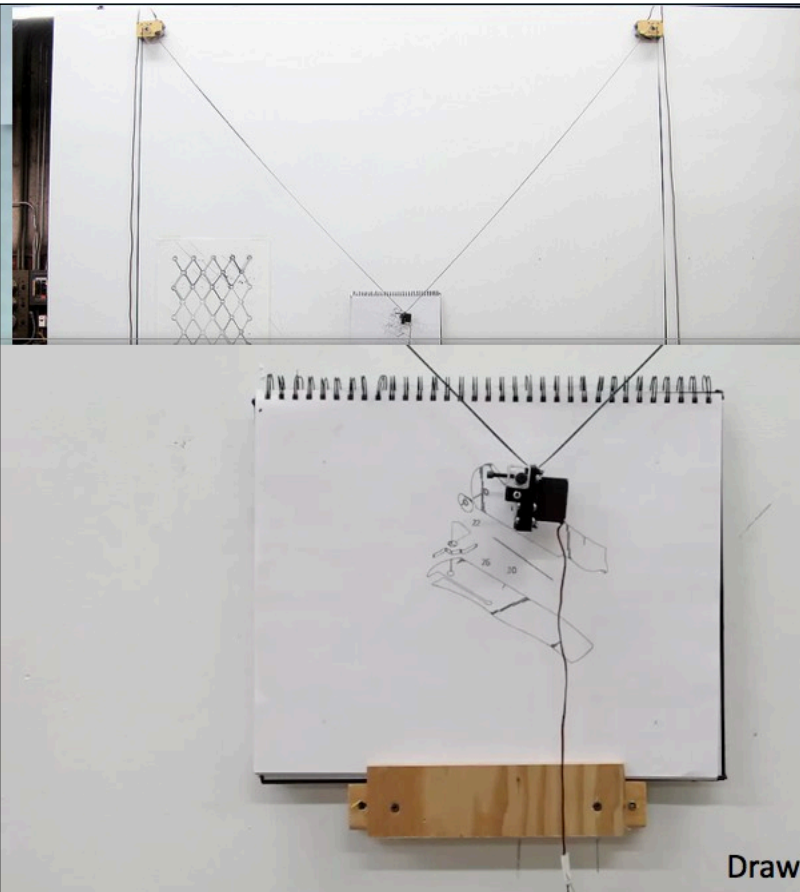
- Ohm’s law to the rescue! $I = V / R$

- Example: 6v stepper with 7.9 Ω /coil = 0.76A

- Stepper drivers let you specify max current!

- This lets you drive steppers at higher voltages and keep the current at safe levels.**

| V REF | Current Limit |
|-------|---------------|
| .1v | .250A |
| .15v | .375A |
| .2v | .500A |
| .25v | .625A |
| .3v | .750A |
| .35v | .875A |
| .4v | 1.000A |
| .45v | 1.125A |



Drawing Machine, Robert Twomey, 2013

Questions to Ask about Motors



- What do you want to move?
 - Smallish things, limited motion? = **servo**
 - Rotating? = **continuous rotation servo** or **DC motor**
 - Heavier, precise motion? = **stepper**
- How many servos?
 - Arduino library supports controlling 12
 - Remember current limits! Extra power supply may be needed

Questions to Ask about Motors



- More than a tiny motor?
 - Use **switching transistor**
 - **TIP120** is common - up to 60vdc and 5A (with heat sink)
- Using multiple power supplies?
 - Make sure all **GNDs** are connected together
- Stepper? How much current / coil?
 - Use proper **voltage** if using H-Bridge (Ohm's Law!)
 - Set **current limit** if using stepper driver board

Questions to Ask about Motors



- Which wires are which on a DC Motor?
 - It doesn't matter! If it's not spinning the right way, reverse them.
- Which wires are which on your Stepper?
 - A little trickier - Use an ohmmeter to figure out connection
 - Then there's coil direction - If you get this wrong, things will "stutter" and you can reverse one coil.

• Sensing and controlling the physical world with computers...

- Sense HIGH/LOW value on digital pin `digitalRead(pin);`
- Sense analog voltage through ADC `analogRead(apin);`
- Set HIGH/LOW value on digital pin `digitalWrite(pin, value);`
- Set PWM duty cycle on digital pin `analogWrite(pin, value);`

- Time delays `delay(msec);` `delayMicroseconds(usec);`

• Connecting external components...

- Always check voltage and current requirements
- Current-limiting resistor
 - Always use for LEDs and switches
- Resistive sensor / voltage divider - analog voltages
 - Calibrate sensors using Serial Monitor
- Switching transistor for larger voltages/currents
- SPI serial communication protocol for external chips
- Motor driver board for steppers

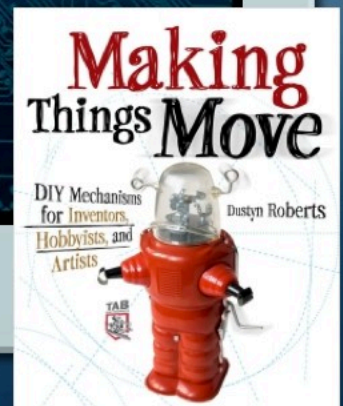
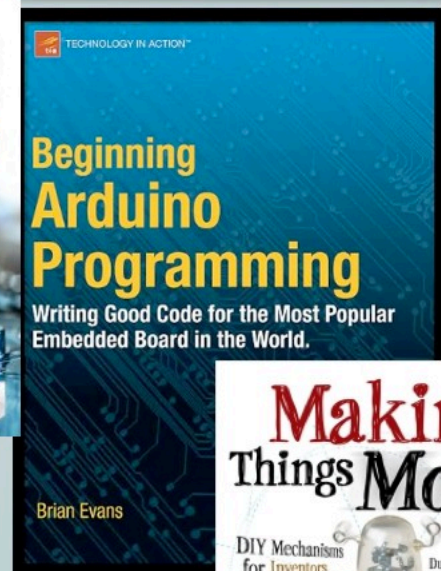
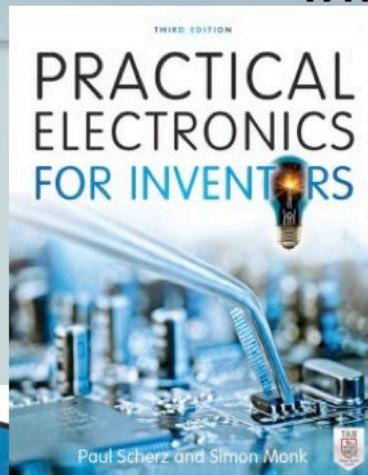
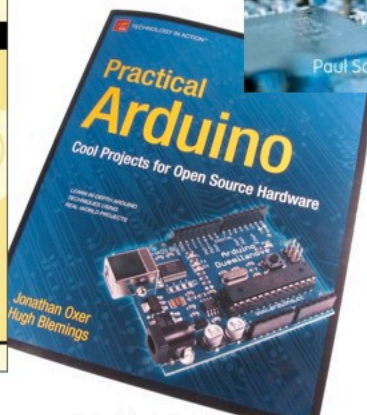
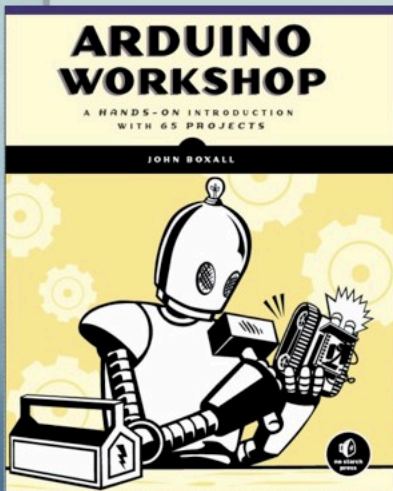
Supply Sources

- Component suppliers



Information Sources

- www.Arduino.cc
- www.Freeduino.org



Whew!

Questions?

