Embedded Systems and Kinetic Art

CS5968: Erik Brunvand
School of Computing

FA3800: Paul Stout
Department of Art and Art History

Logistics

- Class meets Wednesdays from 3:05-6:05
- We'll start meeting in MEB 3133
  - At some point we may also meet in the New Media Wing on the south side of campus
- Web page is www.eng.utah.edu/~cs5968

Kinetic Art

- **Art** that contains moving parts or depends on motion, sound, or light for its effect.
  - The kinetic aspect is often regulated using microcontrollers connected to motors, actuators, transducers, and sensors that enable the sculpture to move and react to its environment.

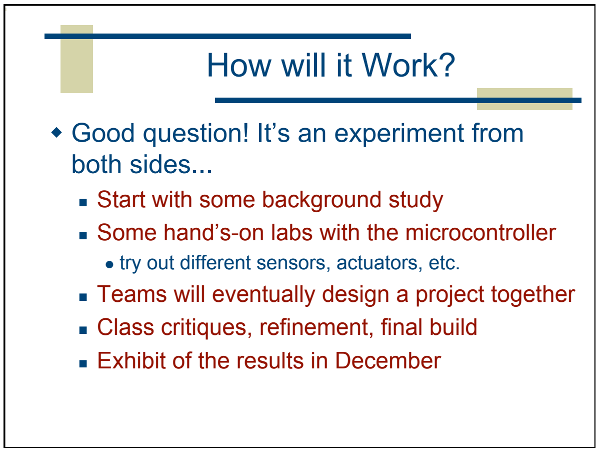
Embedded Systems

- A special-purpose computer system (microcontroller) designed to perform one or a few dedicated functions, often reacting to environmental sensors.
  - It is embedded into a complete device including hardware and mechanical parts rather than being a separate computer system.

Kinetic Art

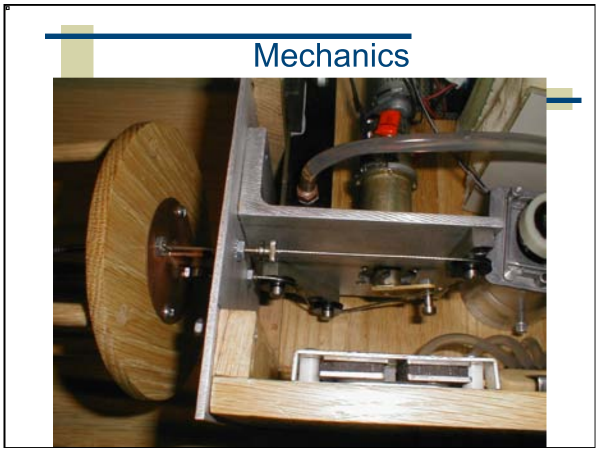
This Class

- Try to get engineers and artists to collaborate to make some interesting kinetic art
  - Force artists and engineers to work on interdisciplinary teams
  - This will be a cross between an engineering class (embedded system design and programming) and an art studio class (designing and building the sculptures) with all students participating fully in both areas.

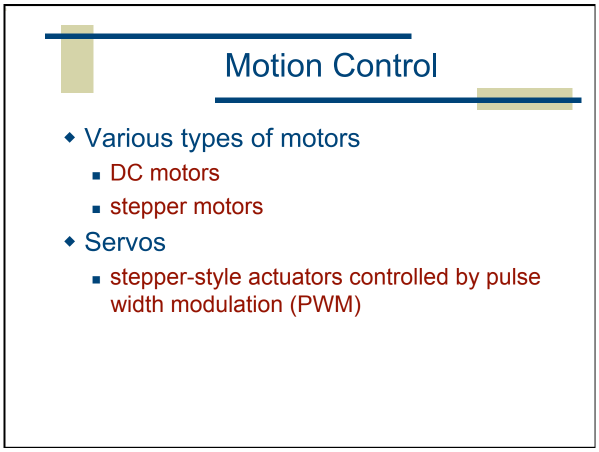


How will it Work?

- Good question! It's an experiment from both sides...
  - Start with some background study
  - Some hand's-on labs with the microcontroller
    - try out different sensors, actuators, etc.
  - Teams will eventually design a project together
  - Class critiques, refinement, final build
  - Exhibit of the results in December

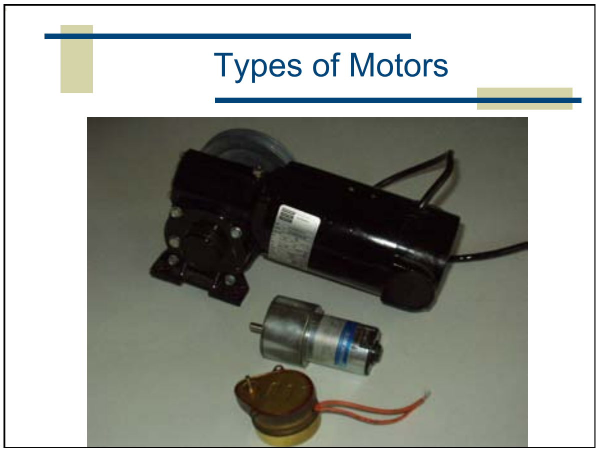


Mechanics

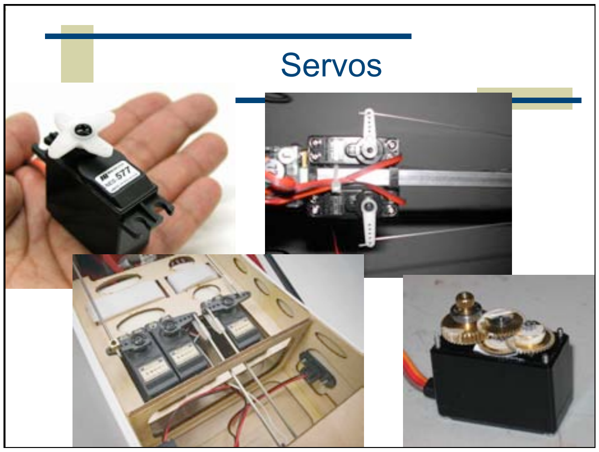


Motion Control

- Various types of motors
  - DC motors
  - stepper motors
- Servos
  - stepper-style actuators controlled by pulse width modulation (PWM)



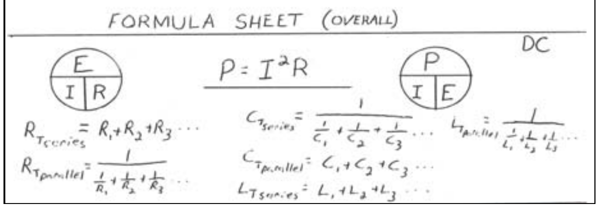
Types of Motors



Servos

Electronics

- You'll need to learn a little electronics
  - Make sure you don't blow things up
  - It's not hard, but you'll need to think a little
- Ohm's Law, etc.



FORMULA SHEET (OVERALL)

DC

$P = I^2R$

$R_{T_{series}} = R_1 + R_2 + R_3 \cdots$

$R_{T_{parallel}} = \dfrac{1}{\frac{1}{R_1} + \frac{1}{R_2} + \frac{1}{R_3} \cdots}$

$C_{T_{series}} = \dfrac{1}{\frac{1}{C_1} + \frac{1}{C_2} + \frac{1}{C_3} \cdots}$

$C_{T_{parallel}} = C_1 + C_2 + C_3 \cdots$

$L_{T_{series}} = L_1 + L_2 + L_3 \cdots$

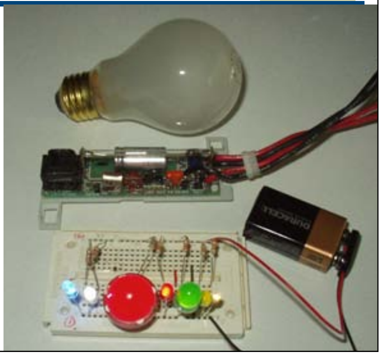
$L_{T_{parallel}} = \dfrac{1}{\frac{1}{L_1} + \frac{1}{L_2} + \frac{1}{L_3} \cdots}$

Outputs

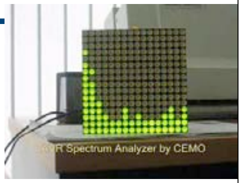
- Cause an action to happen
  - motors and servos cause movement
  - Also light, sound, etc.

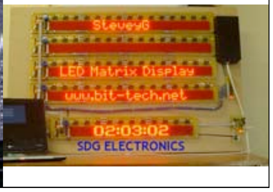
Light Producing Hardware

Light bulbs
strobes
light emitting diodes
(LEDs)



LEDs





Chips to drive LEDs

- Direct control from the microcontroller
- Serial data to external controller ICs
  - some with PWM on each channel
- External LED matrix controllers

- Various ways to drive and control lots of LEDs...

Sound

Speakers
Piezo buzzers

Full audio vs.
PWM buzzing

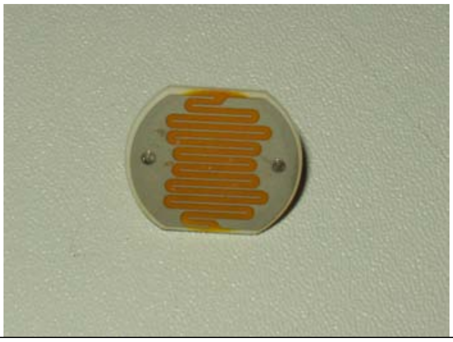
Sound

ISD Digital/Analog solid state recording chip

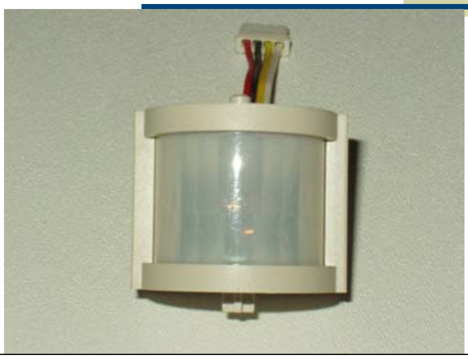
Sensors

- Sense what's going on in the world
- Inputs to your controller
  - light sensors
  - movement detectors
  - rangefinders
  - temperature sensors
  - position sensors

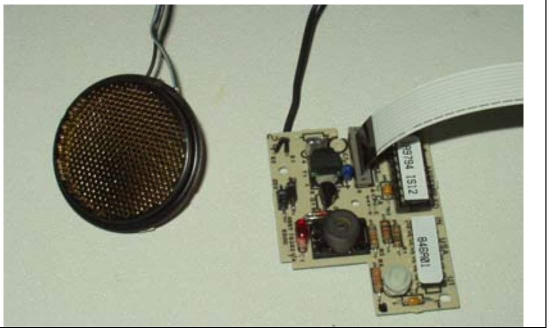
Photocell



Passive infrared (PIR)



Sonar rangefinder

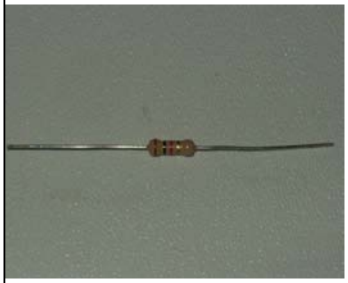


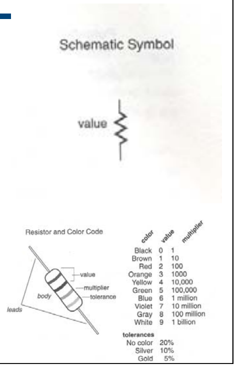
Circuit "glue"

- These electrical components need a little tender loving care
  - so you don't blow them up
  - so the range of values they see or produce is scaled properly
  - so they get the right voltages

  - Can't be sloppy about this!

Resistors



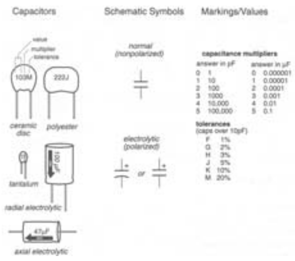


color	value	multiplier
Black	0	1
Brown	1	10
Red	2	100
Orange	3	1000
Yellow	4	10,000
Green	5	100,000
Blue	6	1 million
Violet	7	10 million
Gray	8	100 million
White	9	1 billion

tolerances

No color	20%
Silver	10%
Gold	5%

Capacitors



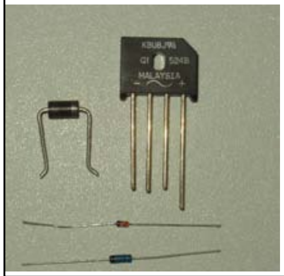
capacitance multipliers

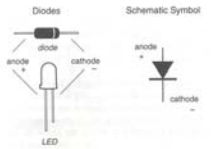
answer in pF		answer in μF	
0	1	0	0.000001
1	10	1	0.00001
2	100	2	0.0001
3	1000	3	0.001
4	10,000	4	0.01
5	100,000	5	0.1

tolerances
(caps over 10pF)

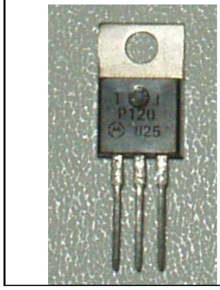
F	1%
G	2%
H	3%
J	5%
K	10%
M	20%

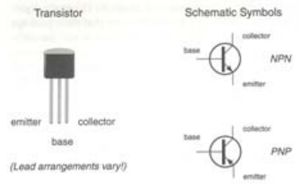
Diodes and LEDs





Transistors





Assembling Components

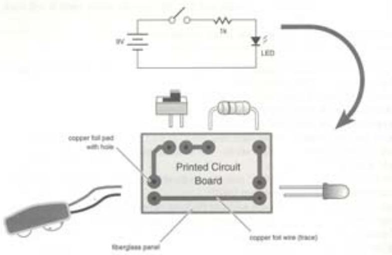
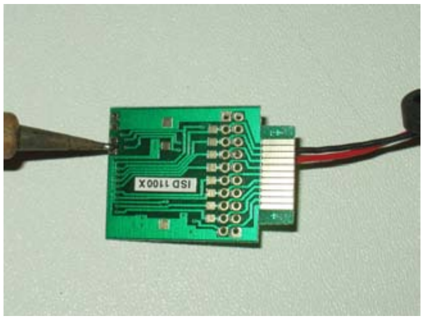
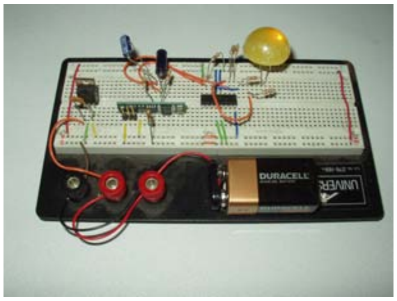


Figure 4-11 How a circuit translates to a circuit board.

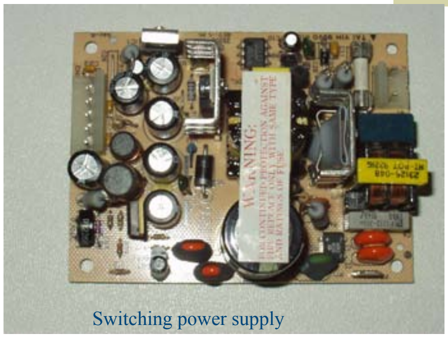
Assembly (soldering)



Assembly (breadboard prototyping)



Power supplies, batteries, etc.



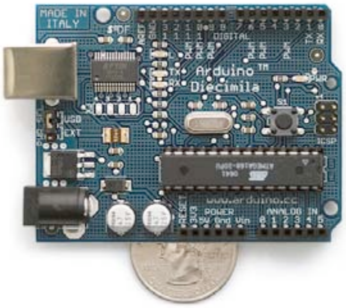
Switching power supply

Batteries, power supplies, etc.

Microcontroller

- The “brains” that coordinates the kinetics
  - Small computers
  - Typically with special support for sensors and actuators
    - Analog-digital converters on inputs
    - pulse-width modulation on outputs

Arduino



What is Arduino?

The word “Arduino” can mean 3 things

A physical piece of hardware

A programming environment

A community & philosophy

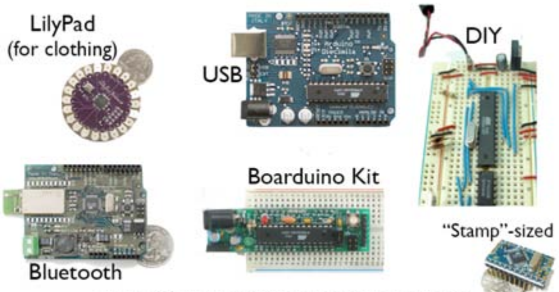
Arduino Community

- Open source physical computing platform
  - "open source" hardware
  - open source software environment
  - physical computing means sensing and controlling the physical world
- Community
  - Examples wiki (the "playground")
  - Forums with helpful people

Arduino Hardware

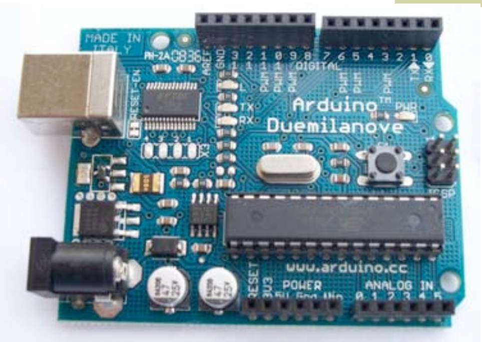
- Similar to Basic Stamp (if you know of it)
  - but cheaper, faster, & open
- Uses AVR ATmega168 microcontroller chip
  - chip was designed to be used with C language

Arduino Hardware Variety

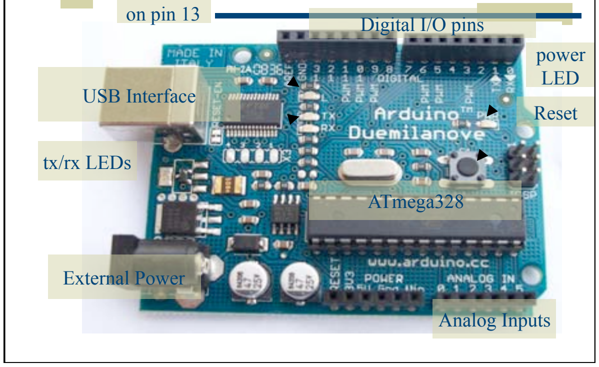


many different variations to suite your needs

Arduino

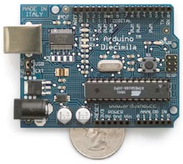


Arduino

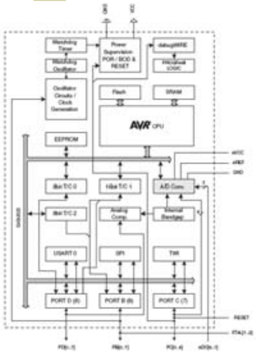


Arduino

- Based on the AVR ATmega328 chip
  - 8 bit microcontroller (RISC architecture)
  - 32k flash for programs
  - 2k RAM, 2k EEPROM, 32 registers
  - 14 digital outputs (pwm on 6)
  - 6 analog inputs
  - Built-in boot loader
  - Powered by USB
    or by external power

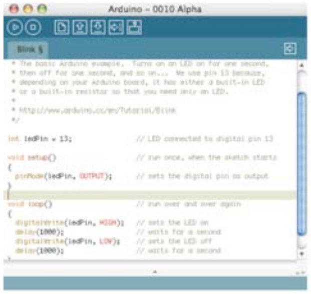


ATmega328P



8-bit RISC CPU – 16MHz
32 registers
32k Flash, 2k SRAM, 1k EEPROM
3 8-bit I/O ports
6 ADC inputs
2 8-bit timers
1 16-bit timer
USART
SPI/TWI serial interfaces

Arduino Software

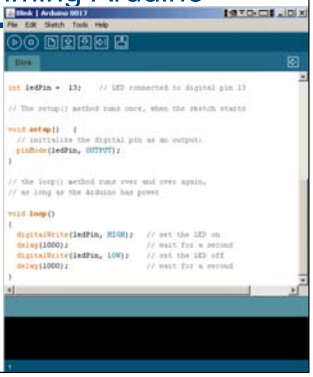


```
 * The basic Arduino example.  Turns on an LED on for one second,
 * then off for one second, and so on...  We use pin 13 because,
 * depending on your Arduino board, it has either a built-in LED
 * or a built-in resistor so that you need only an LED.
 *
 * http://www.arduino.cc/en/Tutorial/Blink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

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

Programming Arduino

- Open-source programming environment
- Arduino language is based on C
  - Actually, it *is* C/C++
  - Hiding under the hood is gcc-avr
  - But, the Ardiuino environment has lots of nice features to make programming less scary...



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

// The setup() method runs once, when the sketch starts

void setup()   {
  // initialize the digital pin as an output:
  pinMode(ledPin, OUTPUT);
}

// the loop() method runs over and over again,
// as long as the Arduino has power

void loop()
{
  digitalWrite(ledPin, HIGH);   // set the LED on
  delay(1000);                  // wait for a second
  digitalWrite(ledPin, LOW);    // set the LED off
  delay(1000);                  // wait for a second
}
```

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.

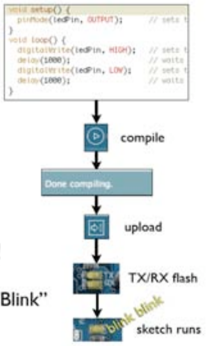
(aka on/off, one/zero) e.g. switch state

"analog" – value ranges, usually from 0-255.

e.g. LED brightness, motor speed, etc.

Using Arduino

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



```
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
}
```

Try it out with the "Blink" sketch!

Load "File/Sketchbook/Examples/Digital/Blink"

More Arduino Info?

- www.arduino.cc/
  - Main Arduino project web site
- www.arduino.cc/playground/Main/HomePage
  - "playground" wiki with lots of users and examples
- www.freeduino.org/
  - "The world famous index of Arduino and Freeduino knowledge"
- www.eng.utah.edu/~cs5968
  - our class web site

Resources for this class

- We have a small grant that can be used to buy supplies for the class
  - Arduino boards
  - sensors of various different types
  - motors and servos
  - LEDs and LED controllers
- You should expect to have to buy a few more parts on your own to complete your project though...
  - We can use this electronics lab, and perhaps wood and metal shop facilities in Art

Next Week

- We'll do a hand's-on session with the Arduino boards
  - Bring a laptop if you have one
  - We'll write some very simple programs
  - Interface to some very simple sensors/LEDs

Next Steps?

- Assignment 1 for next week
  - Look for examples of arts/tech collaborations
  - Find a few examples that you find interesting
  - Make a short powerpoint/keynote presentation on what you found (5-10min)
  - Show it to the class next week