SIGGRAPH2015
Xroads of Discovery

The 42nd International Conference and Exhibition
on Computer Graphics and Interactive Techniques

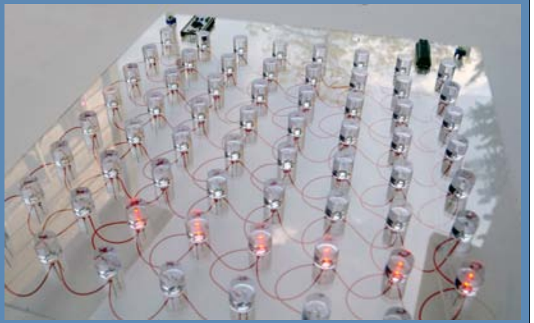
THE UNIVERSITY OF UTAH

Kinetic Sculptures:
Creating Programmable Art

Erik Brunvand
University of Utah

Context - Arts/Technology Collaborations

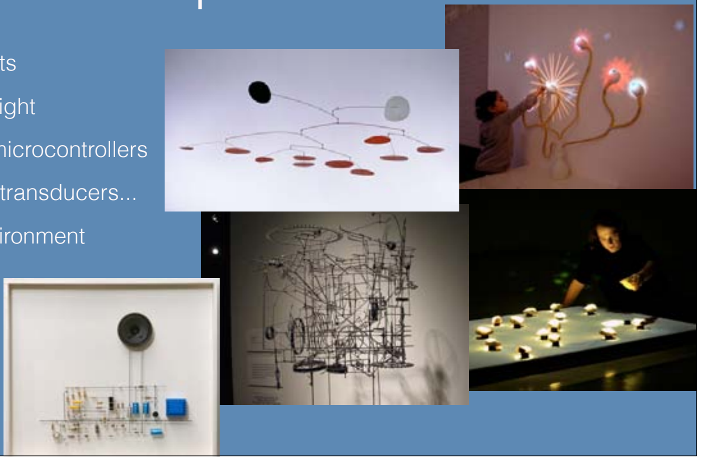
I argue that arts/technology collaboration is a powerful framework for enhancing ideas in both arenas



Serpente Rosso, 2013

Kinetic Sculpture

- Contains moving parts
  - Motion, sound, or light
- Often controlled by microcontrollers
  - Motors, actuators, transducers...
- Often reactive to environment

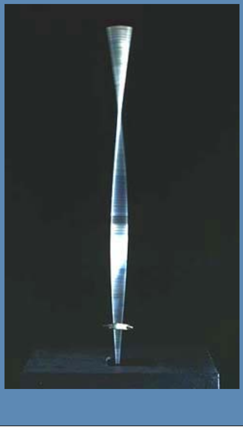


Naum Gabo

Russian - 1890-1977

Kinetic Construction (Standing Wave)

1919-1920

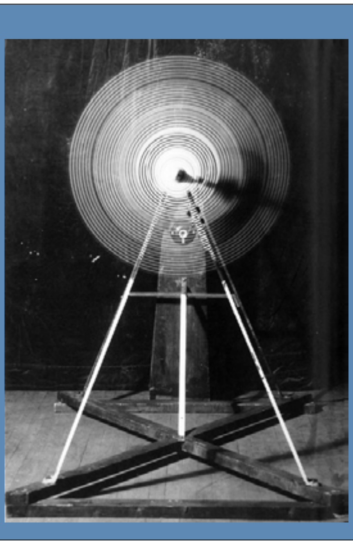


Marcel Duchamp

French (naturalized US) 1887- 1968

Rotary Glass Plates
1920

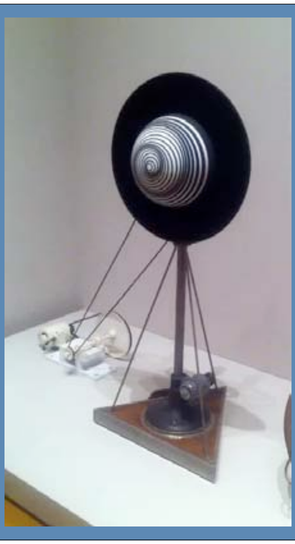
Built with the help
of Man Ray



Marcel Duchamp

French (naturalized US) 1887- 1968

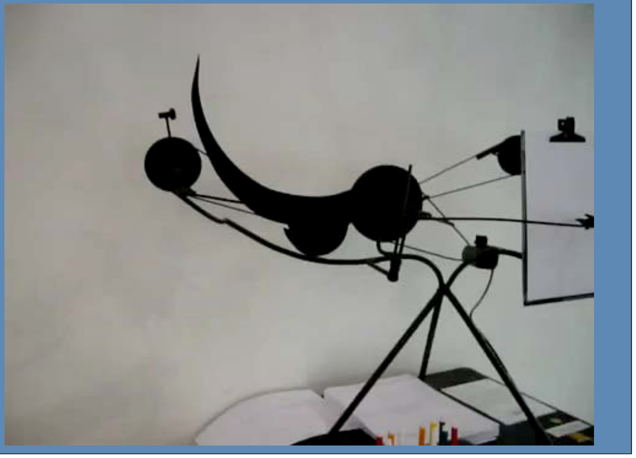
Rotary Demisphere
(Precision Optics)
1925



Jean Tinguely

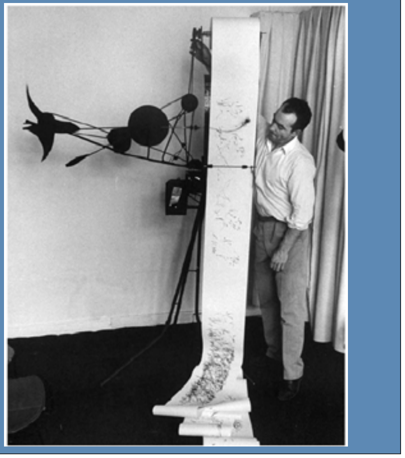
Swiss - 1925 - 1991

Metamatic - 1959

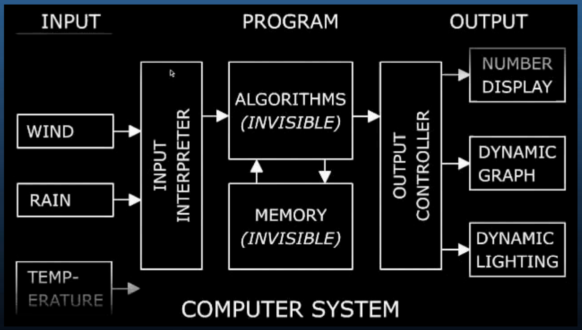


Jean Tinguely

Swiss - 1925 - 1991



Jim Campbell's Algorithm

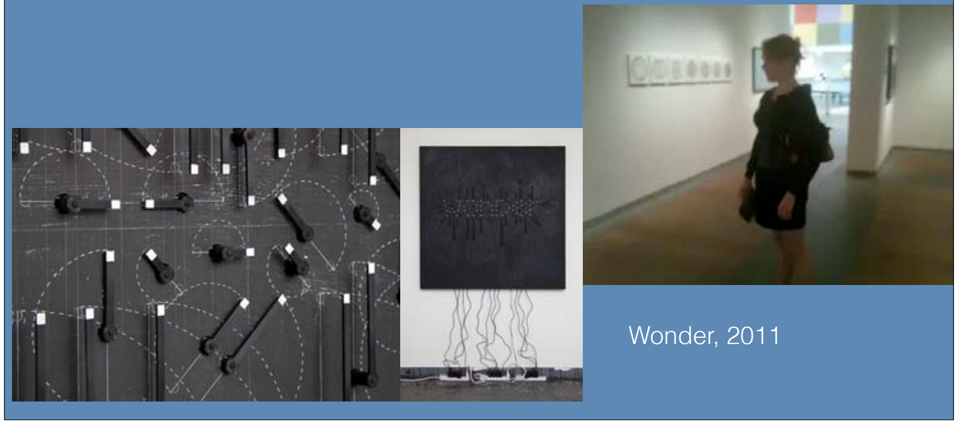

INPUT
PROGRAM
OUTPUT
WIND
RAIN
TEMP-
ERATURE
INPUT INTERPRETER
ALGORITHMS *(INVISIBLE)*
MEMORY *(INVISIBLE)*
OUTPUT CONTROLLER
NUMBER DISPLAY
DYNAMIC GRAPH
DYNAMIC LIGHTING
COMPUTER SYSTEM

Jim Campbell

US - b. 1956

Alicia Eggert

US - b. 1981

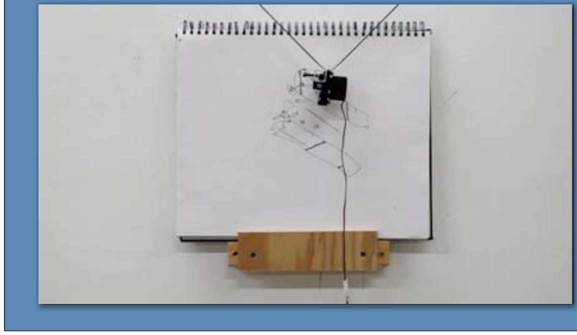


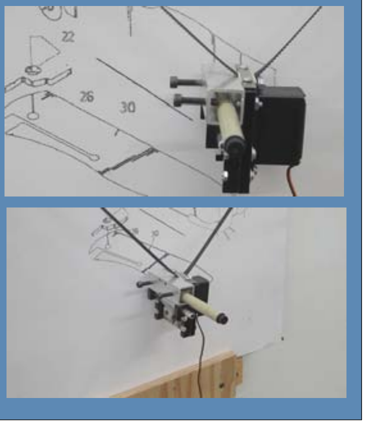
Wonder, 2011

Robert Twomey

US - b. 1979

Drawing Machine - 2013
Showed at SIGGRAPH 2013

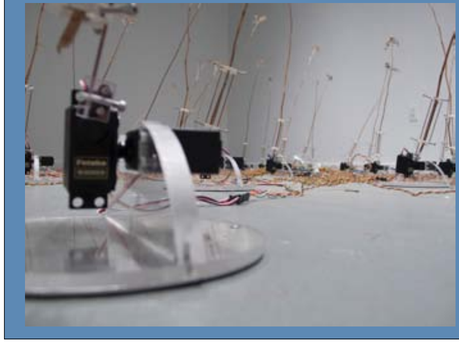




David Bowen

US - b. 1975

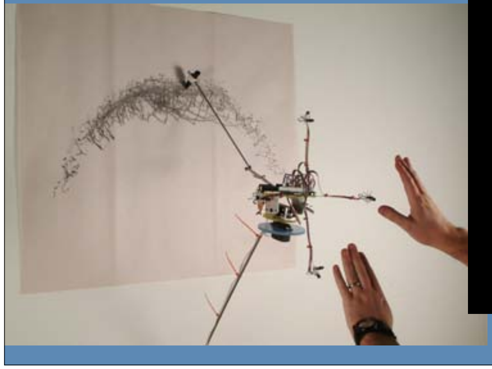
Telepresent Wind (2009)
Showed at SIGGRAPH 2011



telepresent wind
2009

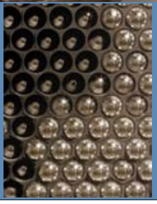
David Bowen

Infrared Drawing Machine (2003)



Daniel Rozen

US - b. 1961

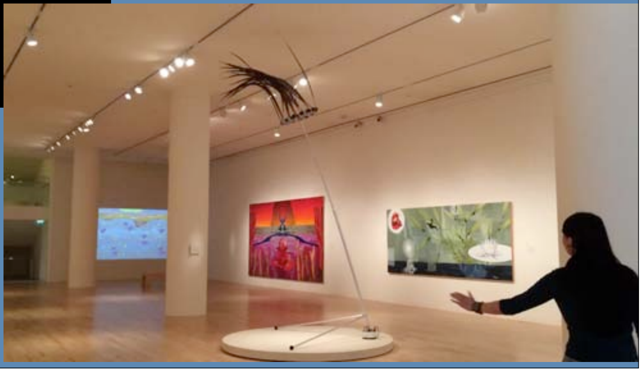


rAndom International

London-based collective

Audience (2008)

Alan Rath



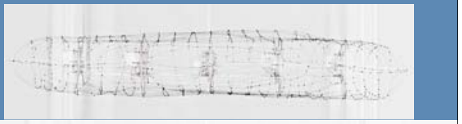
US - b. 1959

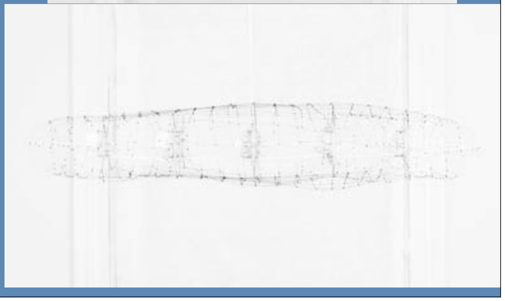
Forever (2012)
Absolutely (2012)

James Leng

Point Cloud, 2012

Architect in LA

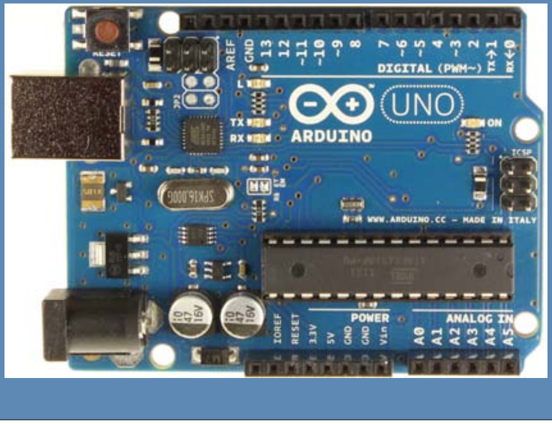


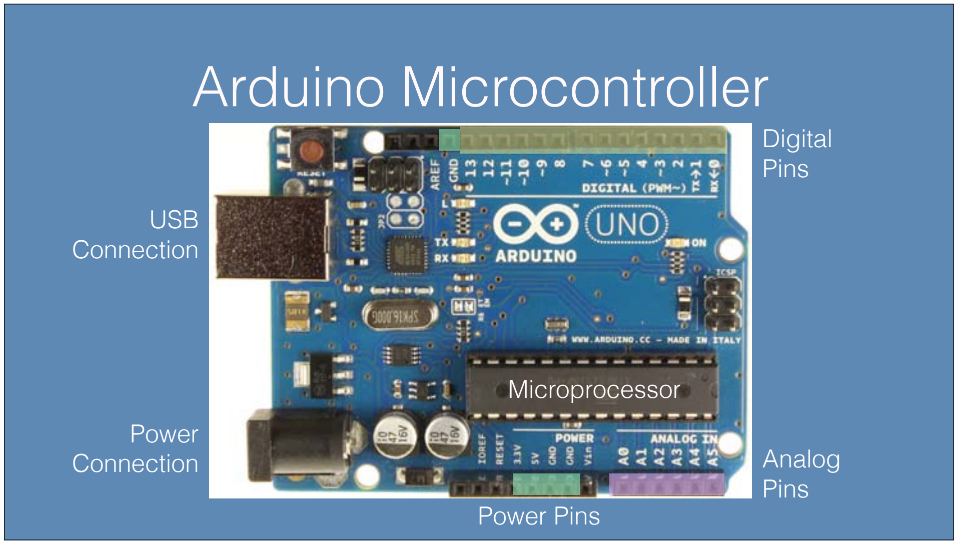


Physical Computing Essentials

- *Get some input from the environment*
  - Light, motion, heat, etc.
- *Cause something to happen*
  - Make something move!

Arduino Microcontroller





Arduino Microcontroller

USB Connection

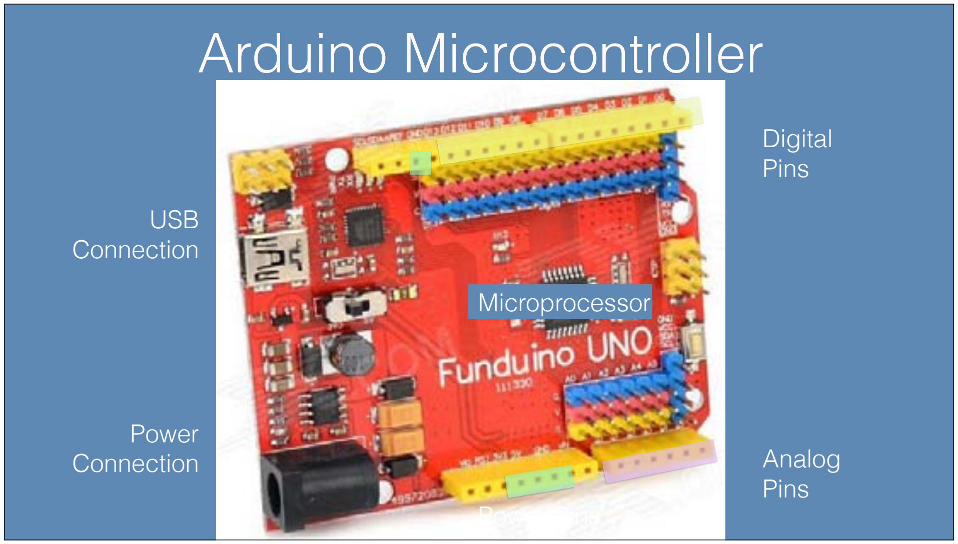
Power Connection

Microprocessor

Digital Pins

Analog Pins

Power Pins



Arduino Microcontroller

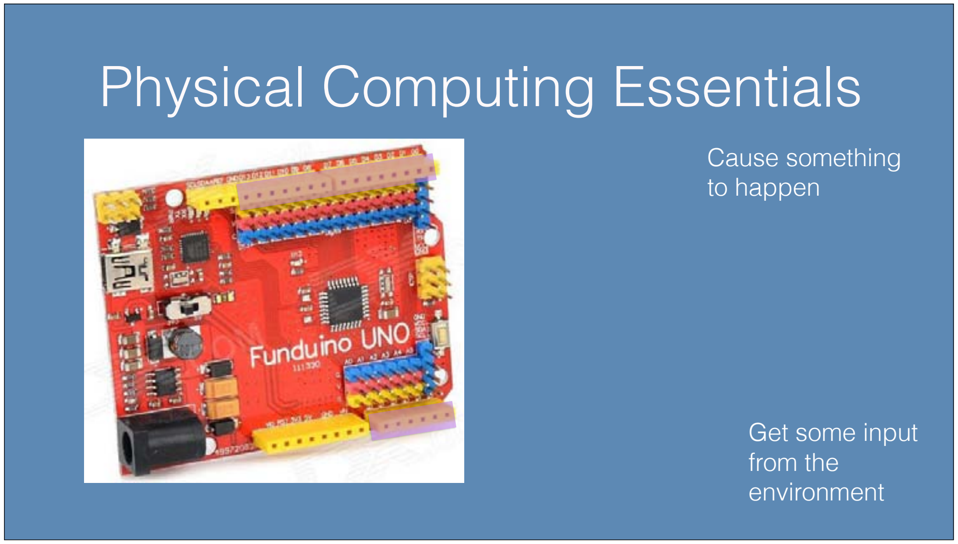
USB Connection

Power Connection

Microprocessor

Digital Pins

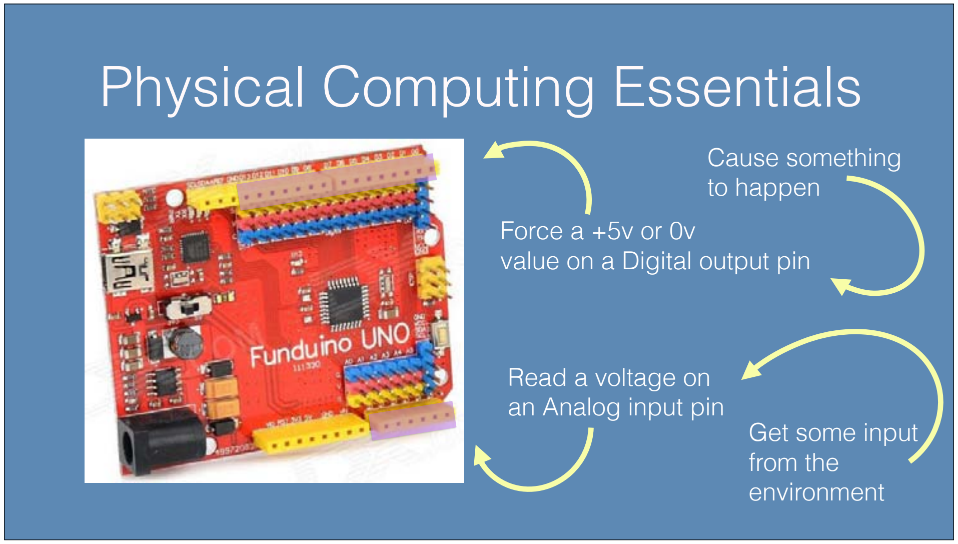
Analog Pins



Physical Computing Essentials

Cause something to happen

Get some input from the environment



Physical Computing Essentials

Cause something to happen

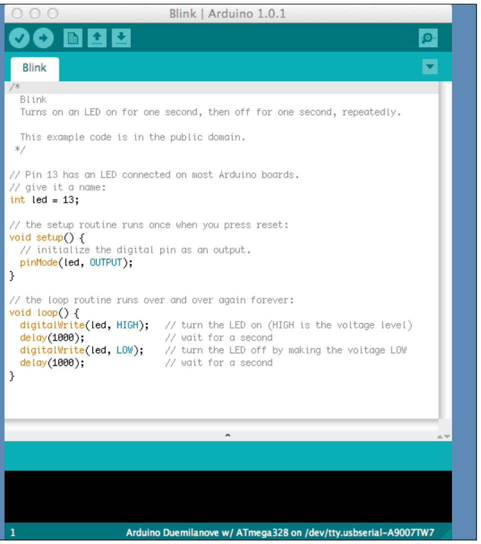
Force a +5v or 0v value on a Digital output pin

Read a voltage on an Analog input pin

Get some input from the environment

Arduino Programming Environment

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

```
/*
  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
}
```

Physical Computing Essentials

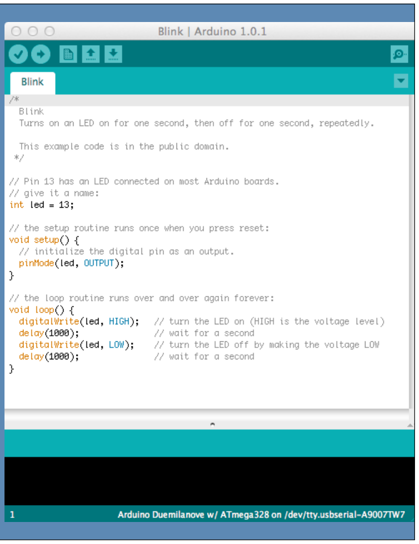
- **pinMode(pinNumber, mode);** // declare a pin INPUT or OUTPUT
- **digitalRead(pinNumber);** // read the HIGH/LOW status of pin
- **digitalWrite(pinNumber, value);** // force a pin HIGH/LOW
- **delay(milliseconds);** // delay processing (spin wait)

Physical Computing Essentials

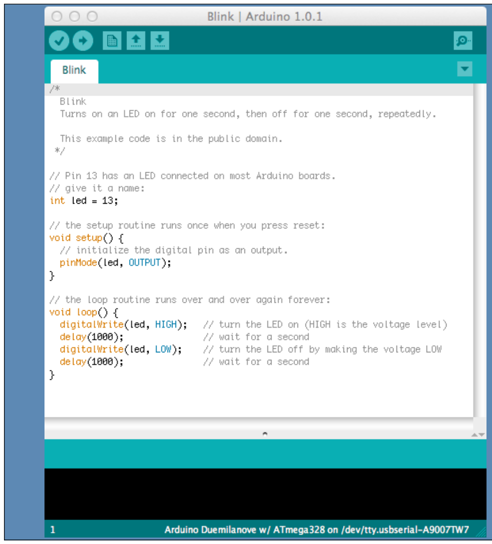
- Each of the digital pins can be set to one of two values
  - High and Low (logic 1 (+5v) and logic 0 (0v))
- **digitalWrite(<pin-number>, <value>);**
  - **digitalWrite(13, HIGH);**
    digitalWrite(13, 1);
  - **digitalWrite(13, LOW);**
    digitalWrite(13, 0);

Arduino Programming

- 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)


Blink | Arduino 1.0.1
Blink
1
Arduino Duemilanove w/ ATmega328 on /dev/tty.usbserial-A9007TW7

```
/*
  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
}
```



```
/*
  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
}
```

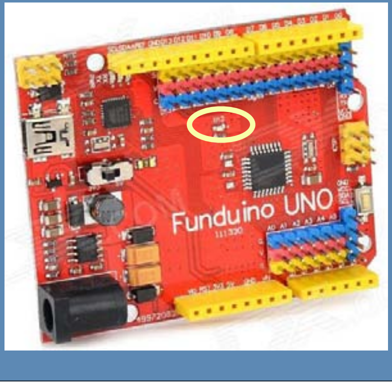
1 Arduino Duemilanove w/ ATmega328 on /dev/tty.usbserial-A9007TW7

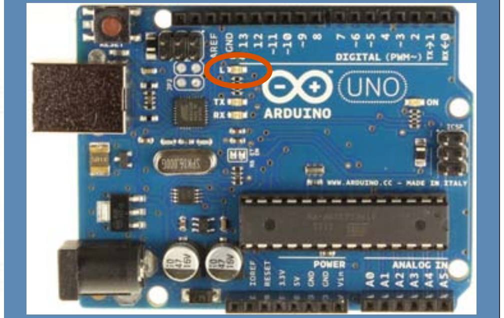
Example: Blink

```
int led = 13;

void setup() { // make pin 13 an output
  pinMode(led, OUTPUT);
}

void loop() {  // turn pin on and off
  digitalWrite(led, HIGH);
  delay(1000);  // delay argument is in ms
  digitalWrite(led, LOW);
  delay(1000);
}
```

What's Blinking?

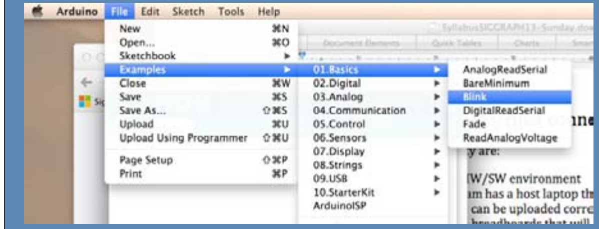
Built-in LED connected to pin 13

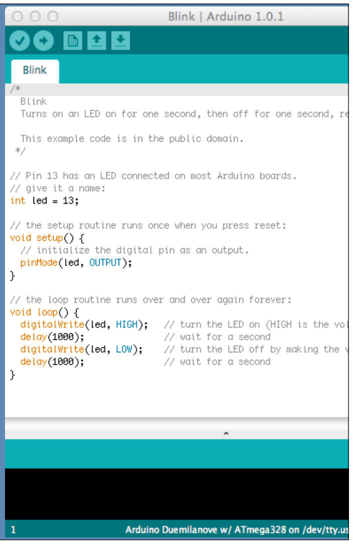




Upload Blink to Arduino

- Load the Blink program from Examples -> Basics -> Blink
- Connect your Arduino with the USB cable



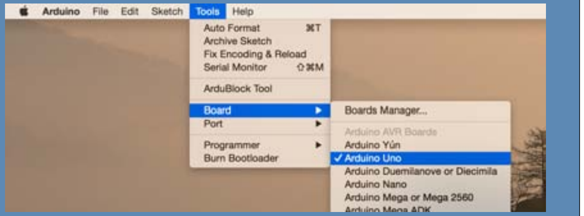


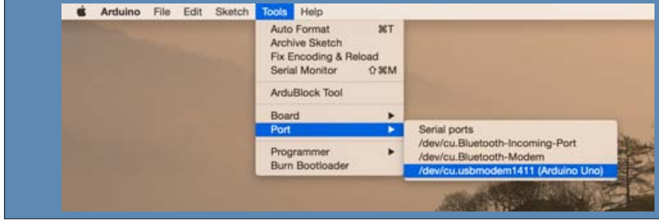
```
/*
  Blink
  Turns on an LED on for one second, then off for one second, re

  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 vo
  delay(1000);               // wait for a second
  digitalWrite(led, LOW);    // turn the LED off by making the v
  delay(1000);               // wait for a second
}
```

1 Arduino Duemilanove w/ ATmega328 on /dev/tty.us

Upload Blink to Arduino

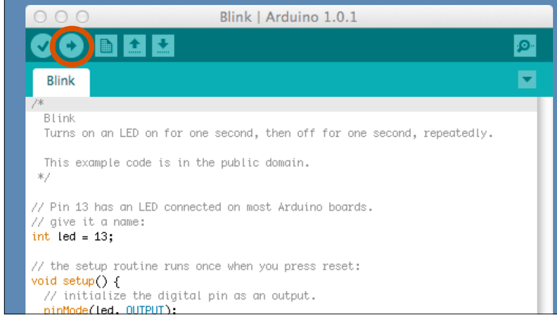
- Make sure you select the correct board
- Tools -> Board -> Uno
- Make sure you select the correct serial port
  - Not the bluetooth ports…



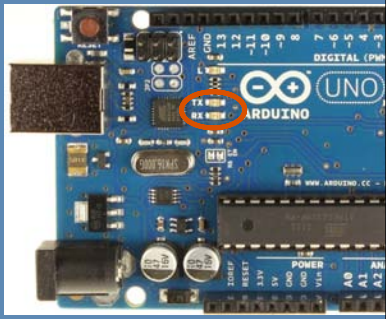


Upload Blink to Arduino

- Click on the upload button
  - Watch for blinky lights during upload

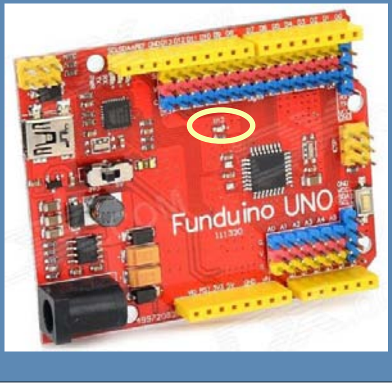


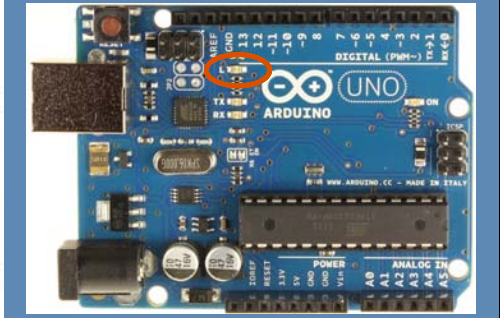
```
/*
  Blink
  Turns on an LED on for one second, then off for one second, repeatedly.

  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);
```



What’s Blinking?

Built-in LED
connected to pin 13



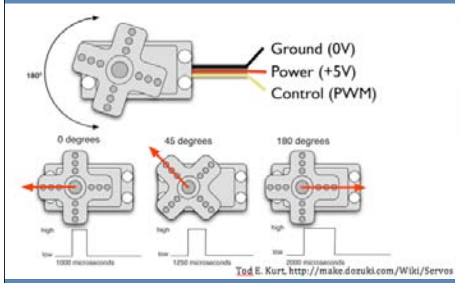


Big Deal?

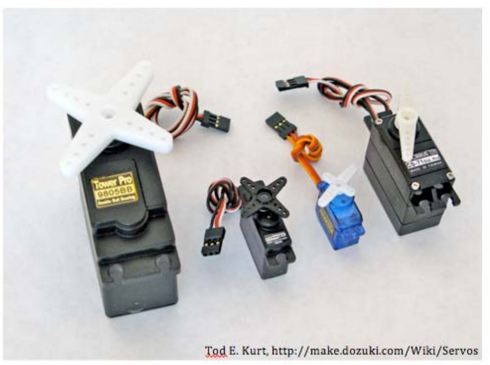
If you can blink an LED
you can control the world!

Turning a pin on and off can
control all sorts of external
devices…

Hobby Servos

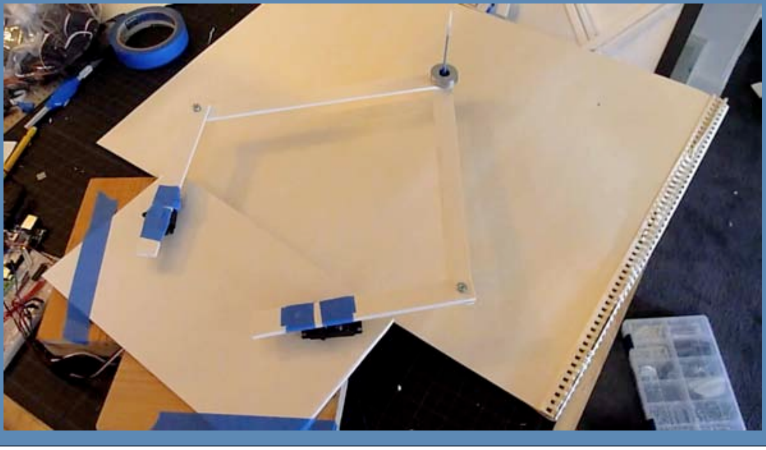


Tod E. Kurt, http://make.dozuki.com/Wiki/Servos



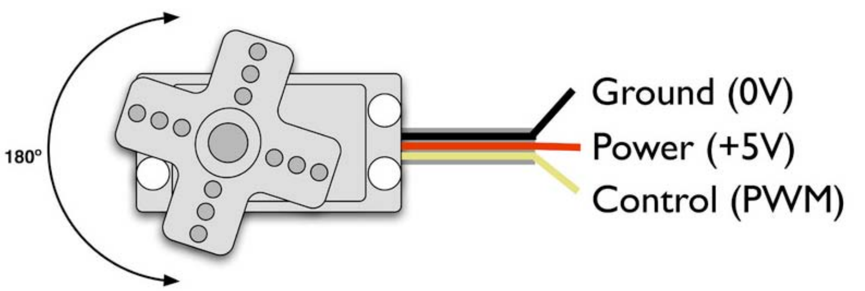
Tod E. Kurt, http://make.dozuki.com/Wiki/Servos

Arduino-Controlled Motion

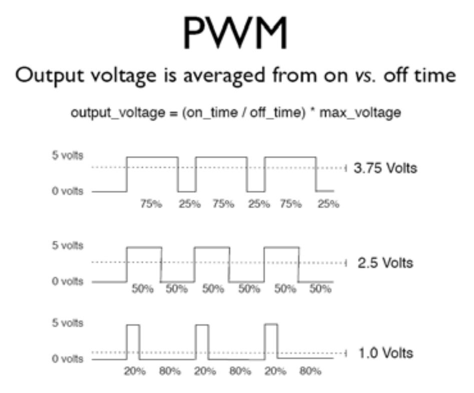


Controlling a Servo

- Pulse Width Modulation (PWM)



Pulse Width Modulation

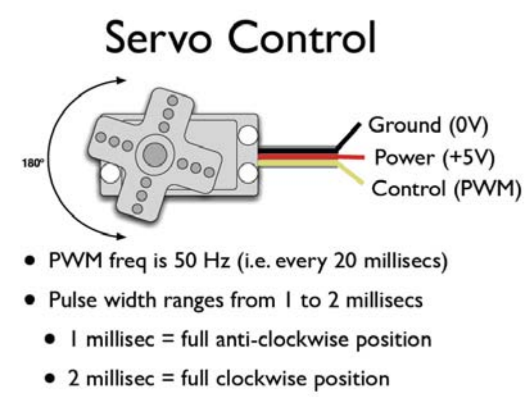


PWM

Output voltage is averaged from on *vs.* off time

output_voltage = (on_time / off_time) * max_voltage

Controlling a Servo

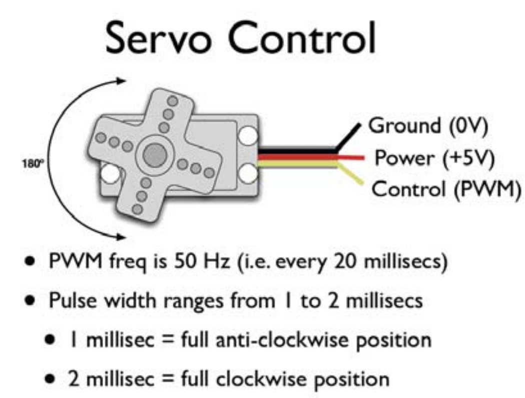


Servo Control

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

Controlling a Servo

Luckily you don't really need to know any of this!



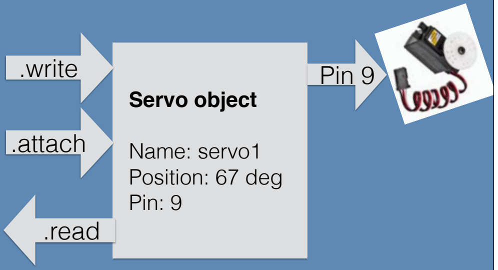
Servo Control

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

There's built-in Arduino code for driving servos!

Servo Object (Class Instance)

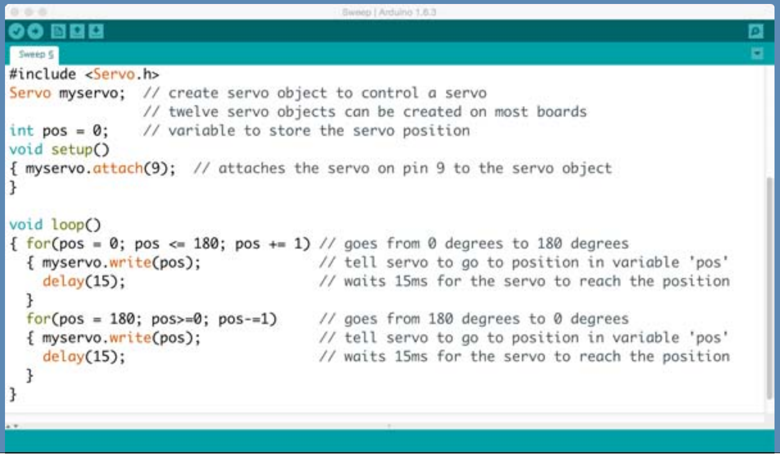
```
#include <Servo.h>   // include servo library

Servo servo1;         // create servo object

void setup() {
   servo1.attach(9);   // attach to pin 9
}

void loop() {
   servo1.write(67);  // move to 67 degrees
   delay(100);         // give it time to move
}
```



Servo Functions (C++ Class)

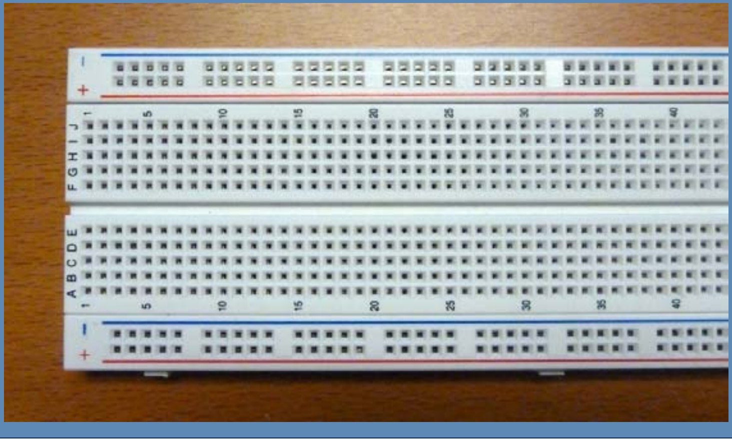
- **Servo myServo;** // creates an instance of Servo class named "myServo"
- **myServo.attach(pin);** // attach myServo to a digital output pin
  - doesn't need to be PWM pin - can be anything from 0-13
  - Servo library can control up to 12 servos on our boards
  - a side effect is that it disables the PWM on pins 9 and 10
- **myServo.write(pos);** // moves myServo – pos ranges from 0-179
- **myServo.read();** // returns current position of myServo (0-179)

Controlling a Servo

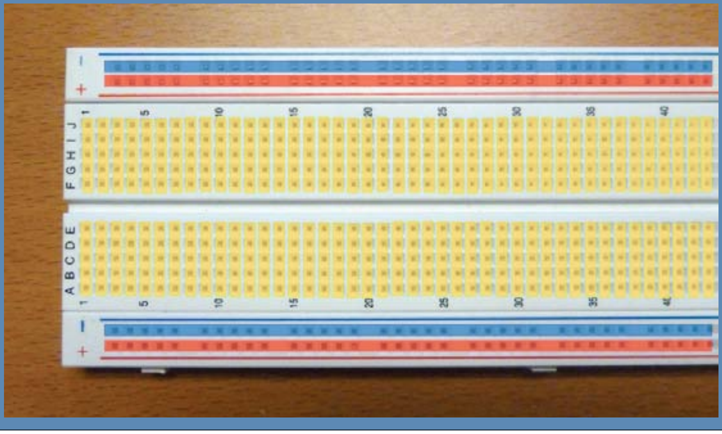


```
#include <Servo.h>
Servo myservo;  // create servo object to control a servo
                // twelve servo objects can be created on most boards
int pos = 0;    // variable to store the servo position
void setup()
{ myservo.attach(9);  // attaches the servo on pin 9 to the servo object
}

void loop()
{ for(pos = 0; pos <= 180; pos += 1) // goes from 0 degrees to 180 degrees
  { myservo.write(pos);              // tell servo to go to position in variable 'pos'
    delay(15);                       // waits 15ms for the servo to reach the position
  }
  for(pos = 180; pos>=0; pos-=1)     // goes from 180 degrees to 0 degrees
  { myservo.write(pos);              // tell servo to go to position in variable 'pos'
    delay(15);                       // waits 15ms for the servo to reach the position
  }
}
```

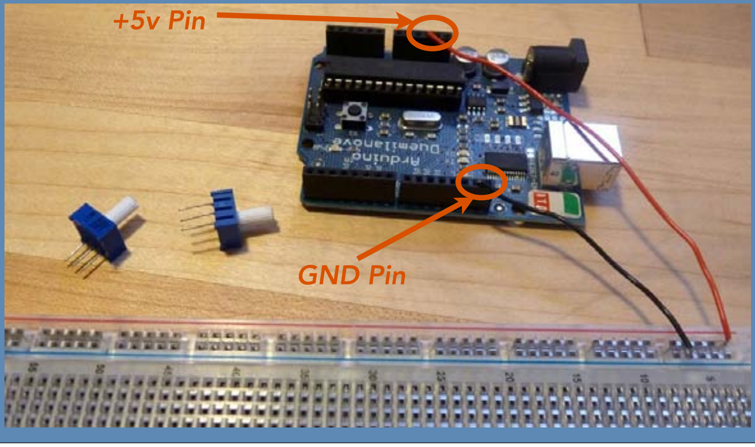
Solderless Breadboard



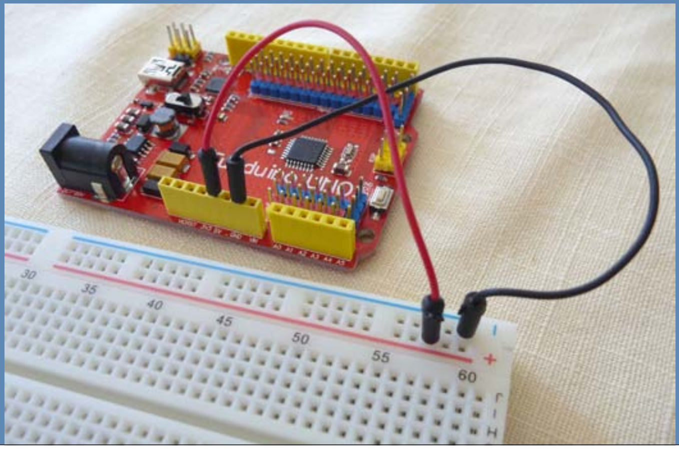
Solderless Breadboard



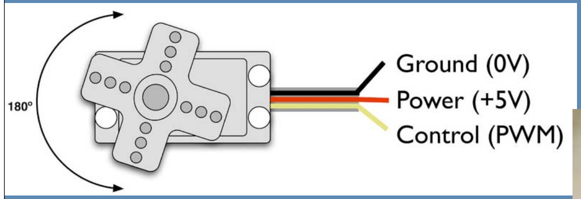
Connecting Power and Ground



Connecting Power and Ground

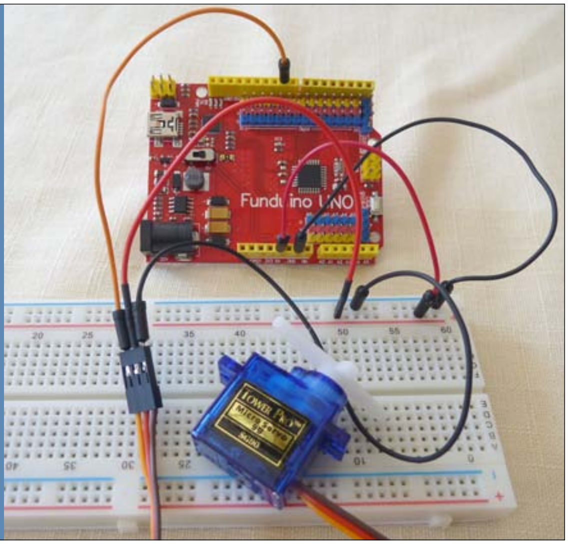


Connecting a Servo

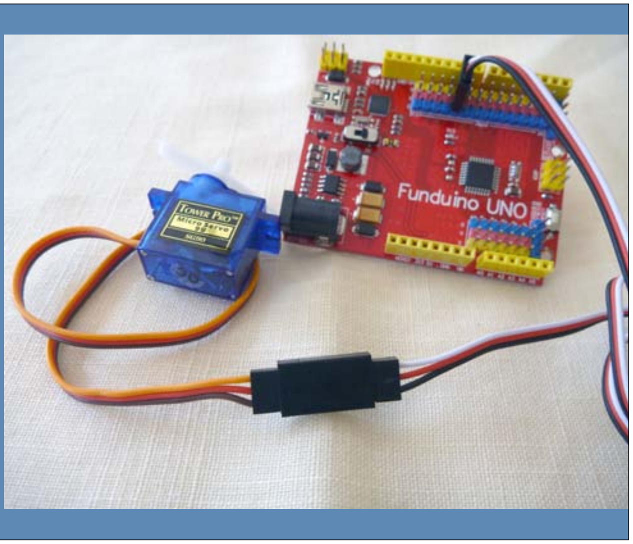


Power is always in the middle
GND is the darker of the two on the edge
Control is the lighter of the two on the edge

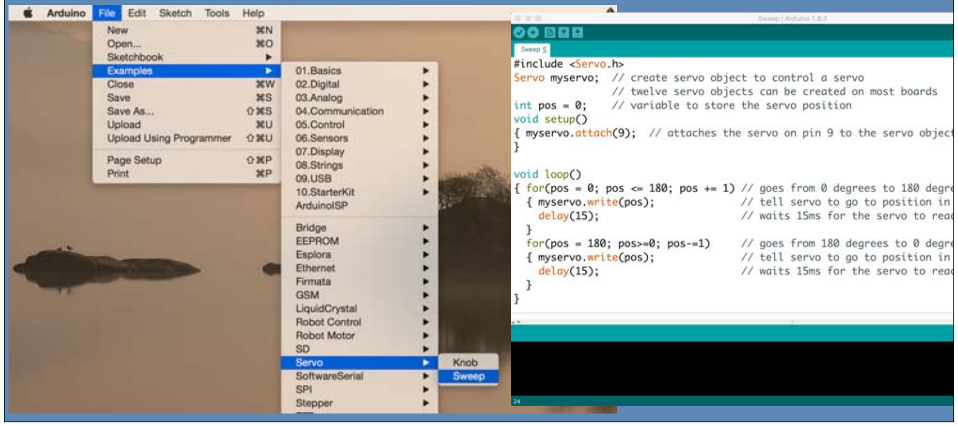
Connecting a Servo



Connecting a Servo



Load and Run Sweep

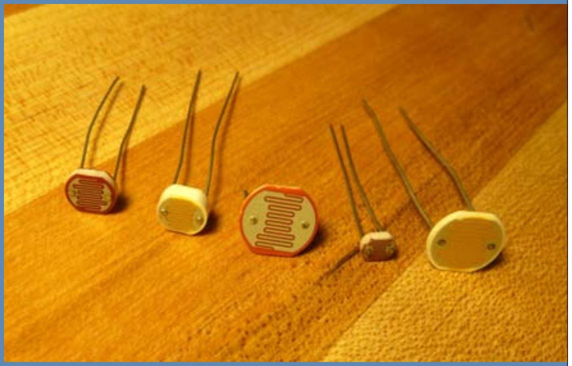


```
#include <Servo.h>
Servo myservo;  // create servo object to control a servo
                // twelve servo objects can be created on most boards
int pos = 0;    // variable to store the servo position
void setup()
{ myservo.attach(9);  // attaches the servo on pin 9 to the servo object
}

void loop()
{ for(pos = 0; pos <= 180; pos += 1) // goes from 0 degrees to 180 degre
  { myservo.write(pos);              // tell servo to go to position in
    delay(15);                       // waits 15ms for the servo to reac
  }
  for(pos = 180; pos>=0; pos-=1)     // goes from 180 degrees to 0 degre
  { myservo.write(pos);              // tell servo to go to position in
    delay(15);                       // waits 15ms for the servo to reac
  }
}
```

Light Sensor

Light-sensitive resistors

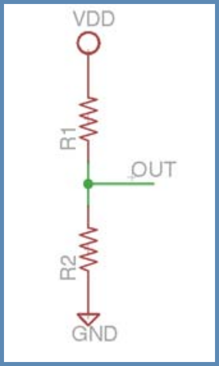
Also called photocells or CdS Sensors



Voltage Divider

- Vout is proportional to the ratio of R1 and R2

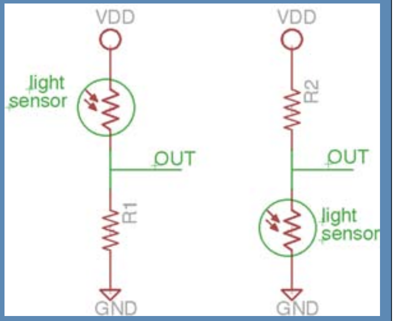
$$Vout = \frac{R_2}{(R_1 + R_2)} Vdd$$



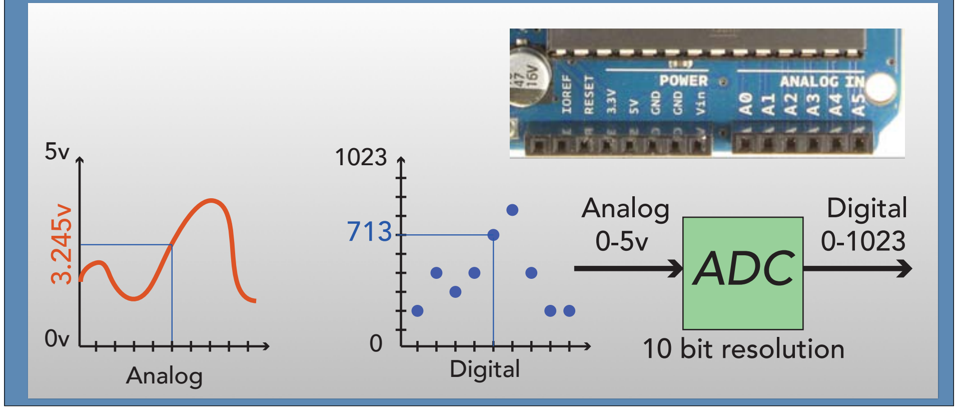
Voltage Divider

$$Vout = \frac{R_2}{(R_1 + R_2)} Vdd$$

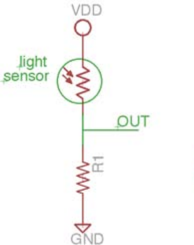
- The changing voltage at OUT can be sensed by the ADC of Arduino
- **analogRead(pinNumber);**
- This senses the voltage (0v to 5v) on the pin and returns a digital value from 0 to 1023

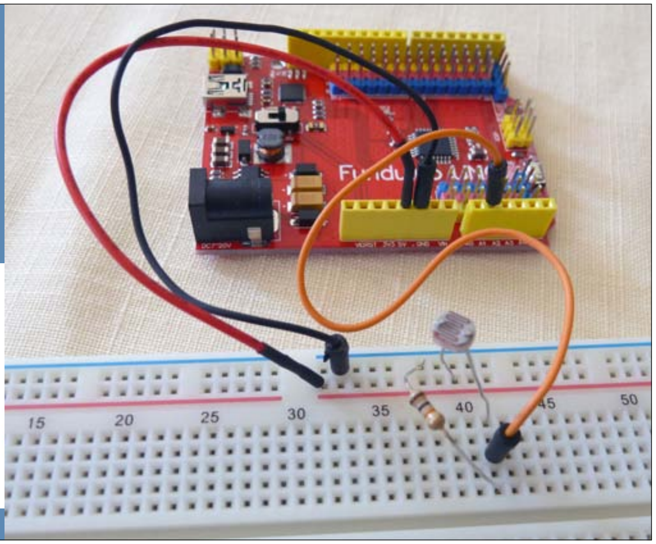


analogIn(pinNum);

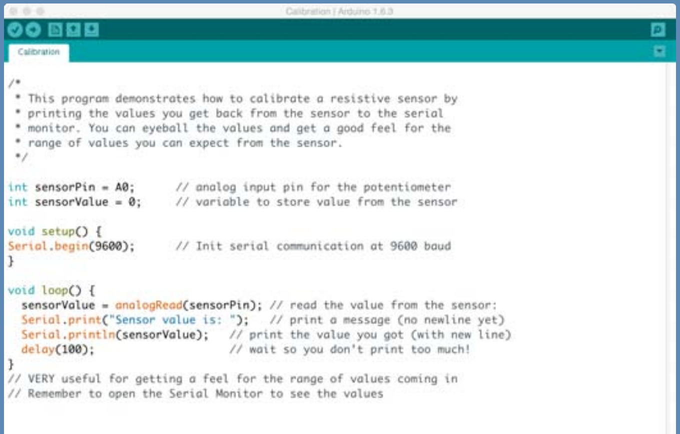


Light Sensor Connection



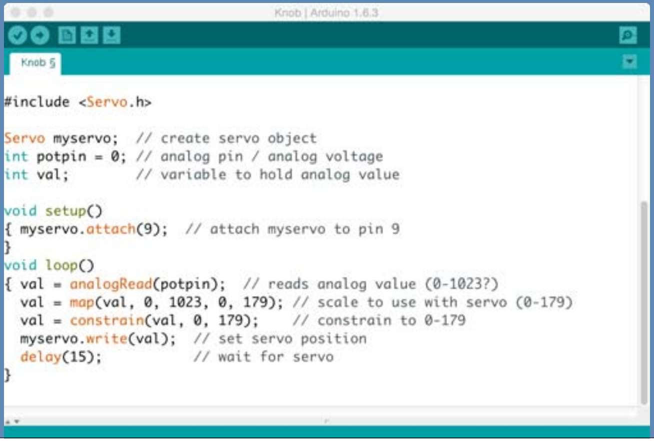


Calibrate Analog Voltage



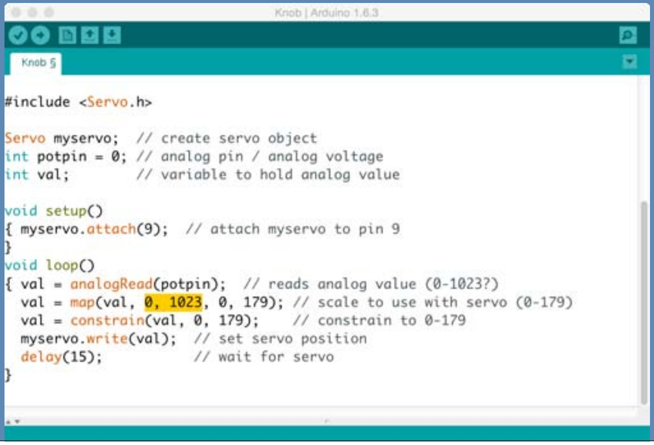
```
/*
 * This program demonstrates how to calibrate a resistive sensor by
 * printing the values you get back from the sensor to the serial
 * monitor. You can eyeball the values and get a good feel for the
 * range of values you can expect from the sensor.
 */

int sensorPin = A0;      // analog input pin for the potentiometer
int sensorValue = 0;     // variable to store value from the sensor

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

void loop() {
  sensorValue = analogRead(sensorPin); // read the value from the sensor:
  Serial.print("Sensor value is: ");   // print a message (no newline yet)
  Serial.println(sensorValue);   // print the value you got (with new line)
  delay(100);                    // wait so you don't print too much!
}
// VERY useful for getting a feel for the range of values coming in
// Remember to open the Serial Monitor to see the values
```

Use the Analog Voltage



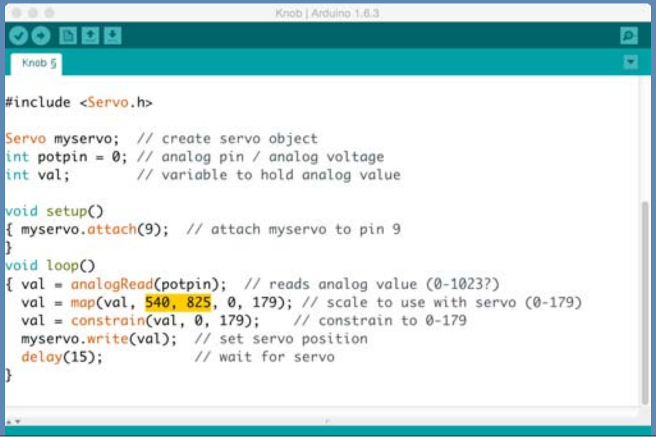
```
#include <Servo.h>

Servo myservo;  // create servo object
int potpin = 0; // analog pin / analog voltage
int val;        // variable to hold analog value

void setup()
{ myservo.attach(9);  // attach myservo to pin 9
}
void loop()
{ val = analogRead(potpin);  // reads analog value (0-1023?)
  val = map(val, 0, 1023, 0, 179); // scale to use with servo (0-179)
  val = constrain(val, 0, 179);    // constrain to 0-179
  myservo.write(val);  // set servo position
  delay(15);           // wait for servo
}
```

Use the Analog Voltage



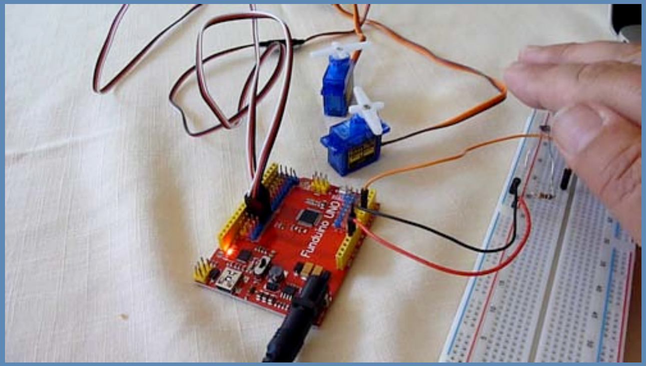
```
#include <Servo.h>

Servo myservo;  // create servo object
int potpin = 0; // analog pin / analog voltage
int val;        // variable to hold analog value

void setup()
{ myservo.attach(9);  // attach myservo to pin 9
}
void loop()
{ val = analogRead(potpin);  // reads analog value (0-1023?)
  val = map(val, 0, 1023, 0, 179); // scale to use with servo (0-179)
  val = constrain(val, 0, 179);    // constrain to 0-179
  myservo.write(val);  // set servo position
  delay(15);           // wait for servo
}
```

Use the Analog Voltage



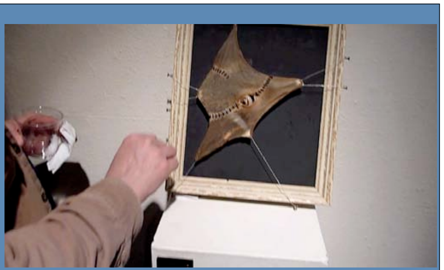
```
#include <Servo.h>

Servo myservo;  // create servo object
int potpin = 0; // analog pin / analog voltage
int val;        // variable to hold analog value

void setup()
{ myservo.attach(9);  // attach myservo to pin 9
}
void loop()
{ val = analogRead(potpin);  // reads analog value (0-1023?)
  val = map(val, 540, 825, 0, 179); // scale to use with servo (0-179)
  val = constrain(val, 0, 179);    // constrain to 0-179
  myservo.write(val);  // set servo position
  delay(15);           // wait for servo
}
```

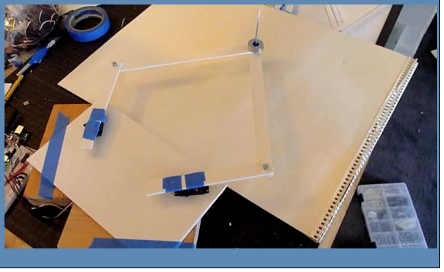
Servo/CdS Light Meter



Go Make Something!

- You have the basic tools you need
  - You can make something move
  - You can respond to light
- Use your imagination and the resources of the Studio
  - Printers
  - Laser cutters
  - Cardboard, foam core, paper, etc.

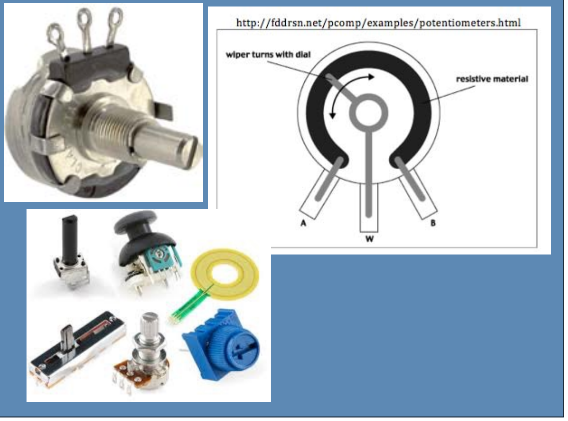




Extra Material

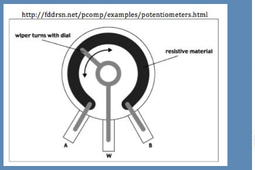
Potentiometers (Knobs)

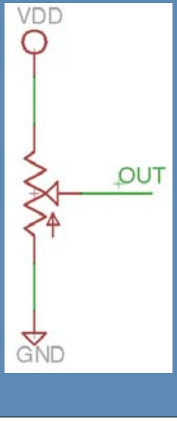
- Variable resistors with a knob



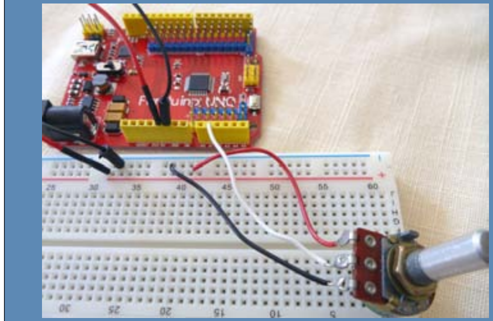
Potentiometers (Knobs)

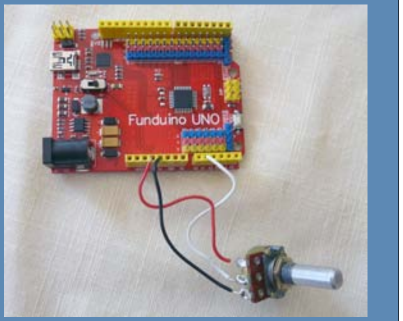
- Variable resistors with a knob
  - Use them just like a CdS light sensor





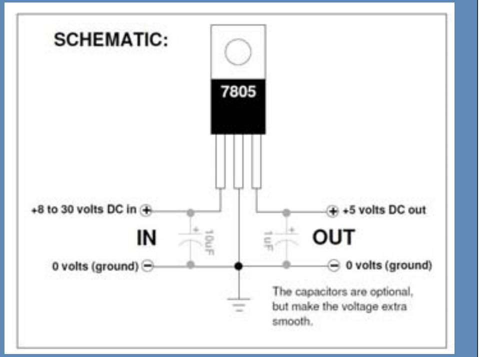
Potentiometers (Knobs)





Voltage Regulation

- Take a higher voltage (e.g. 9v) and reduce it to a regulated lower voltage (e.g. 5v)
  - Extra voltage is converted to heat!
- Provides up to 1.5A of current with an appropriate heat sink
  - Will drive lots of servos!
  - Cap values not critical…

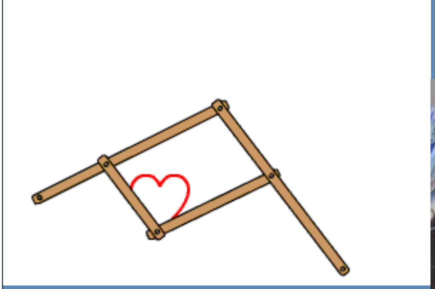


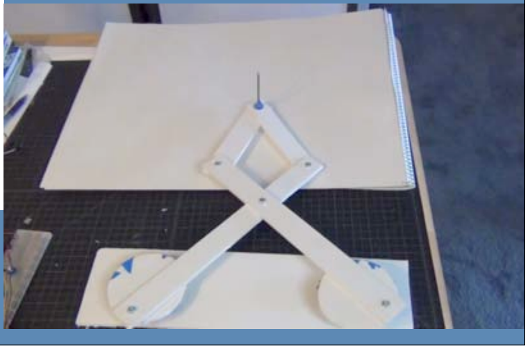
Voltage Regulation

- Take a higher voltage (e.g. 9v) and reduce it to a regulated lower voltage (e.g. 5v)
  - Extra voltage is converted to heat!
- Provides up to 1.5A of current with an appropriate heat sink
  - Will drive lots of servos!
  - Cap values not critical…

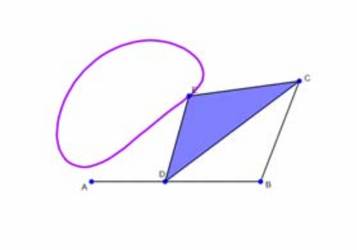
Include picture of regulator on breadboard

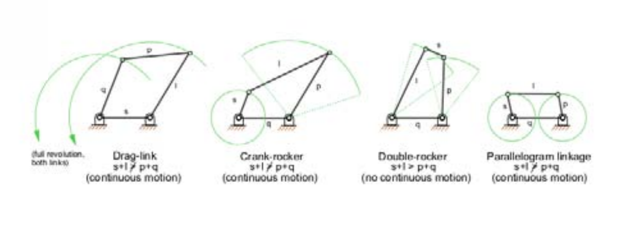
Linkages: Pantograph



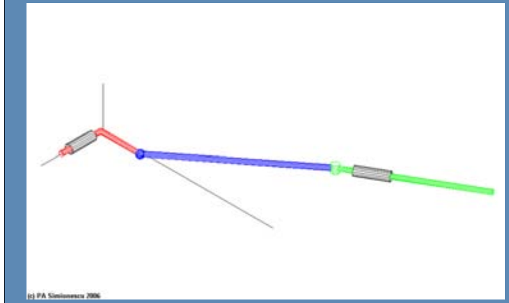


Linkages: Four-Bar

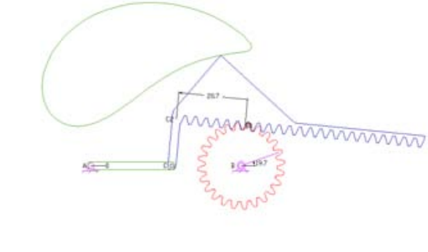




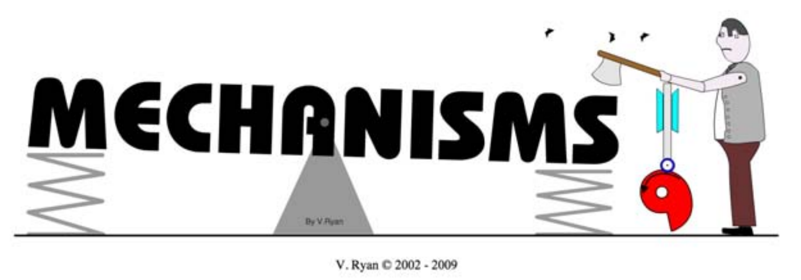
Linkages: Slider-Crank, Rack & Pinion



(c) FA Simionescu 2006



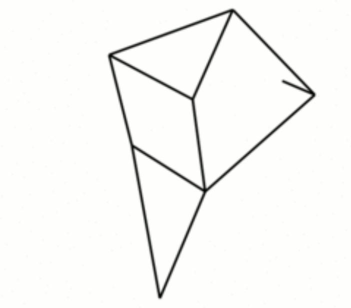
Linkages: Cams

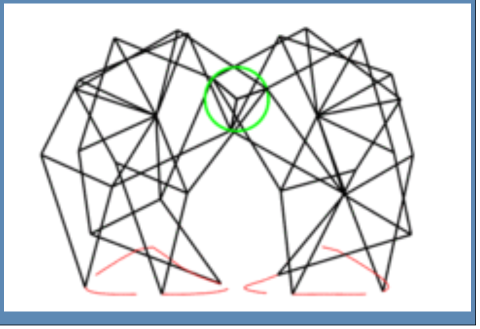


http://www.technologystudent.com/cams/camdex.htm

Linkages: Jansen's Linkage

Theo Jansen, Dutch, b. 1948
Strandbeest





Linkages: Jansen's Linkage

Theo Jansen, Dutch, b. 1948
Strandbeest

Linkages: Klann's Linkage

Patented by Joe Klann, 1994