INTRODUCTION TO PROGRAMMING

Using Arduino

Disclaimer

- Many of these slides are mine
- Others are from various places on the web
  - todbot.com – Bionic Arduino and Spooky Arduino class notes from Tod E.Kurt
  - ladyada.net – Arduino tutorials by Limor Fried

What is a program?

- Essentially just a list of actions to take
  - Each line of the program is step to take
  - The program just walks through the steps one at at time
    - Maybe looping too
- It's like a recipe!

Meatloaf…

Meatloaf Recipe
Ingredients:

1 package Lipton Onion Soup Mix
2 pounds lean ground beef
1 large egg
2/3 cup milk
3 Tablespoons catsup
3 Tablespoons brown sugar
1 Tablespoon yellow mustard

Meatloaf…

Directions:

1. Preheat the oven to 350 degrees F.
2. Mix the onion soup mix, ground beef, egg and milk together.
3. Form the combination into a loaf shape in a 13 X 9 X 2 loaf pan.
4. Combine the rest of the ingredients and spoon onto the top of the meatloaf.
5. Bake uncovered, for about an hour.
6. When done, take the meatloaf out of the pan and place on a serving plate. Let stand for 10 minutes before slicing.

Shampoo

1. Lather
2. Rinse
3. Repeat

- When do you stop?

Shampoo

1. Lather
2. Rinse
3. If this is the first lather, then Repeat
   else stop and towel off

Shampoo

1. Repeat twice {
2. Lather
3. Rinse
4. }

Shampoo

1. For (count=1; count<3; count=count+1)
2. {
3. Lather
4. Rinse
5. }

Shampoo

1. For (count=1; count<3; count=count+1)
2. {
3. Lather
4. Rinse
5. }

count=1
lather
rinse
count=2
lather
rinse
count=3
continue to next instruction…

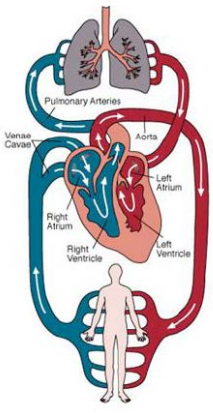
Make a light flash

1. Turn light on
2. Wait for 1 second
3. Turn light off
4. Wait for one second
5. repeat

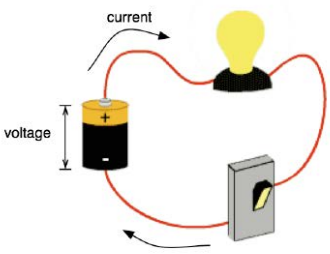
- We'll come back to this… Let's talk about lights

Electricity

Making Circuits



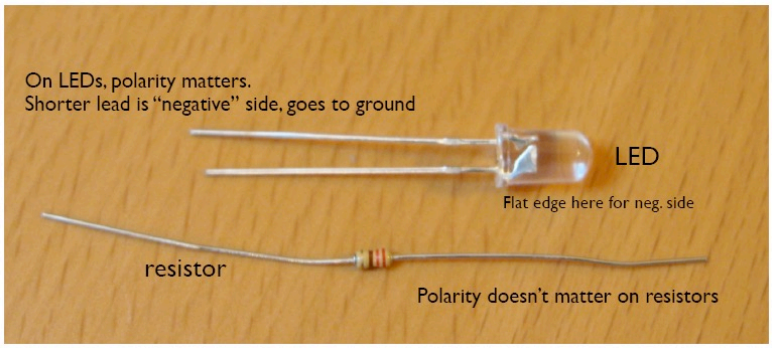
heart pumps, blood flows

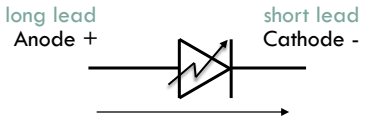


voltage pushes, current flows

www.todbot.com

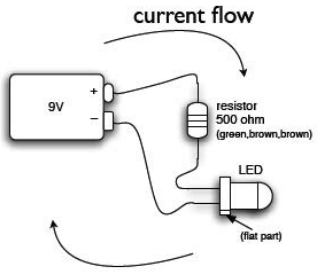
LEDs and Resistors

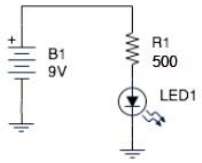


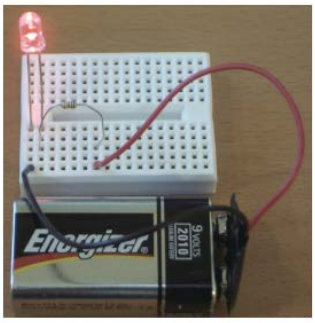


Current flows from Anode to Cathode
Lights up when current flows

Wiring it up



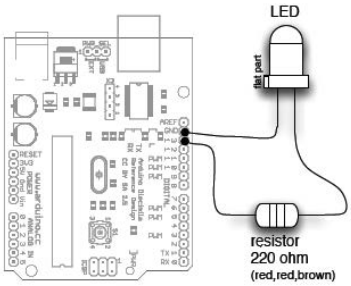




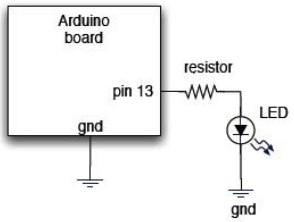
wiring diagram schematic wiring it up

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

Wiring it Up

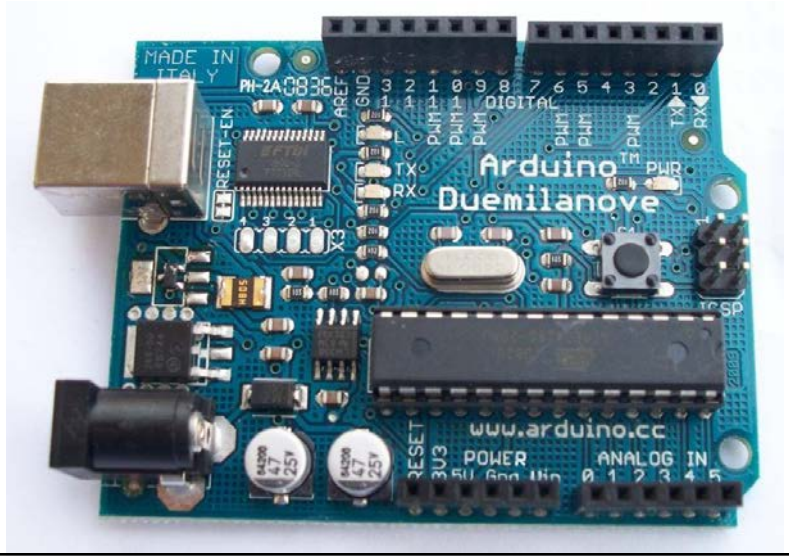


wiring diagram



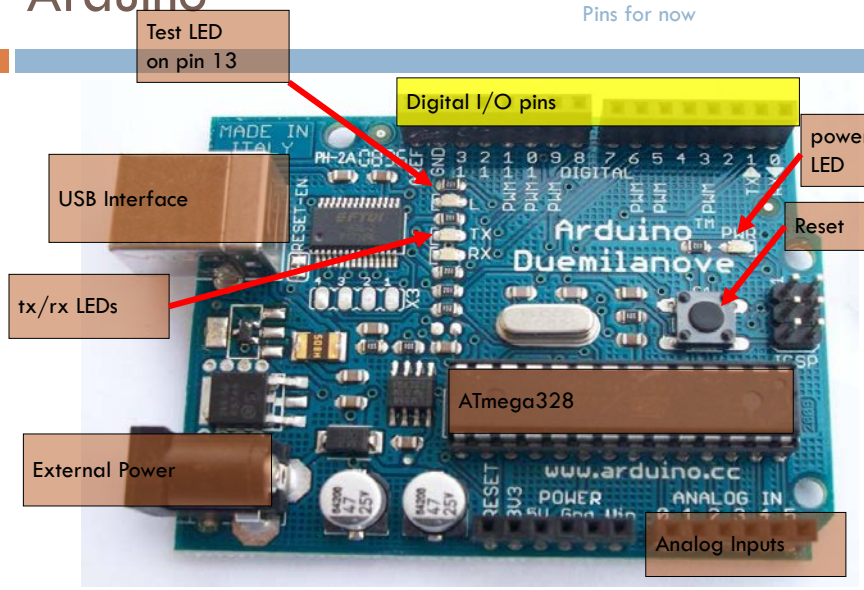
schematic

Arduino

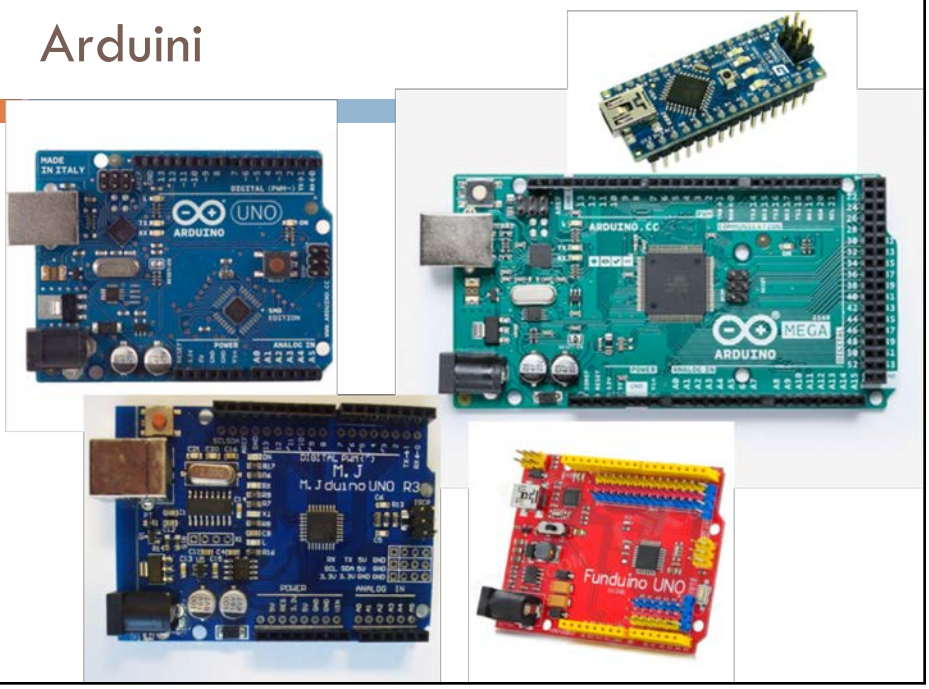


Arduino

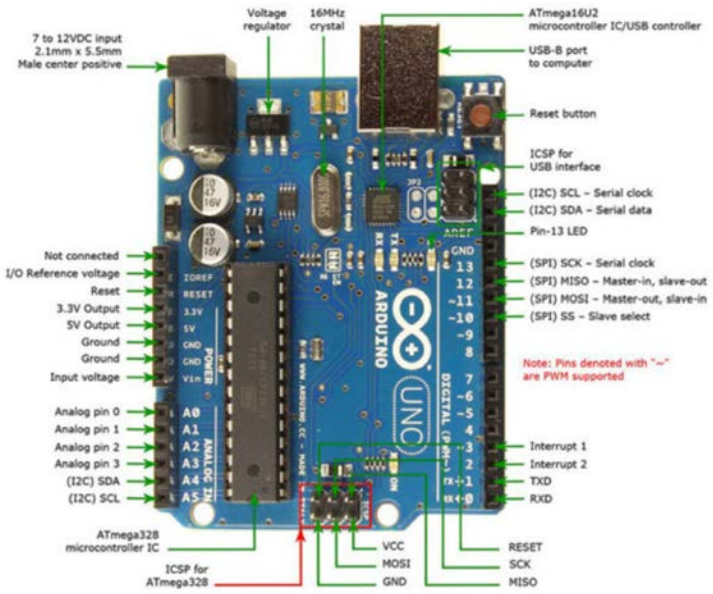
Focus on these Digital Pins for now



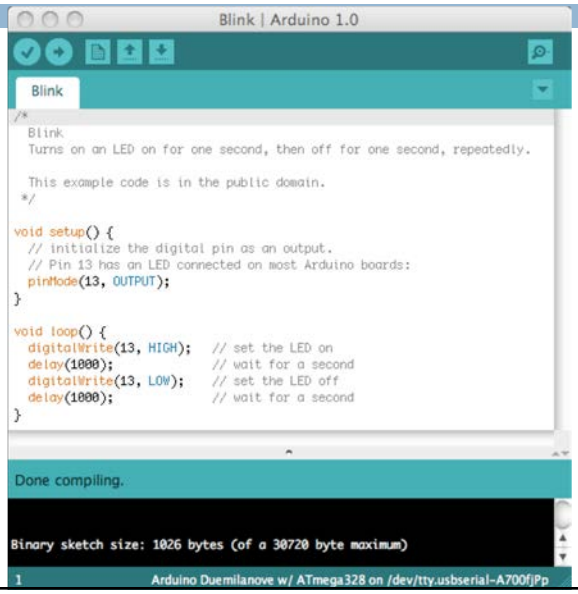
Arduini



Arduino



Arduino Programming

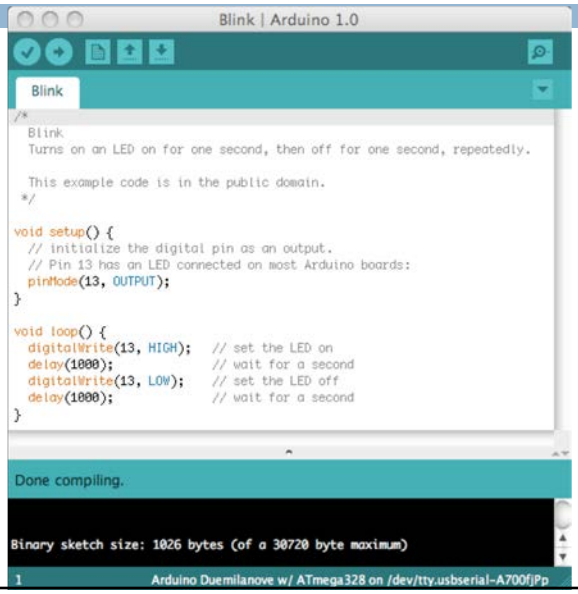


Arduino Programming

Verify, Upload, New, Open, Save

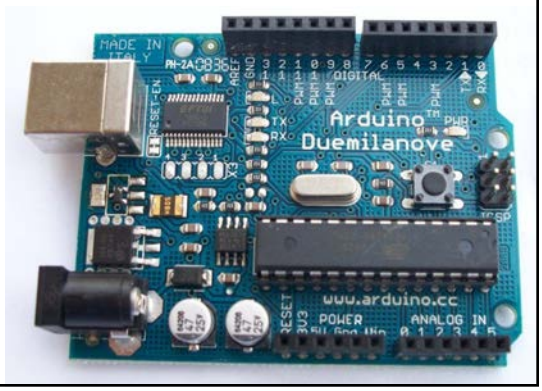
Programming area

Notification area

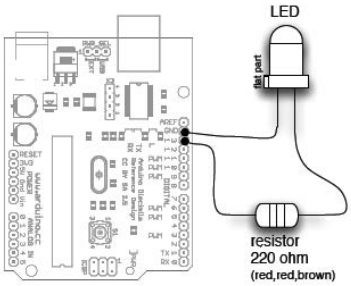


Digital Pins

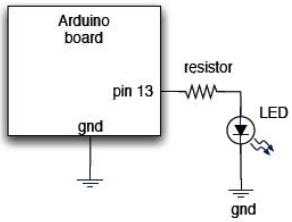
- Each of the digital pins can be set to one of two values
  - High and Low (+5v and 0v)
  - digitalWrite(<pin-number>, <value>);
  - digitalWrite(13, HIGH);
  - digitalWrite(13, LOW);



Wiring it Up



wiring diagram



schematic

Arduino Duemilanove board has this circuit built-in
To turn on LED use `digitalWrite(13,HIGH)`

Make a light flash

1. Turn light on
2. Wait for 1 second
3. Turn light off
4. Wait for one second
5. repeat

Make a light flash

1. Turn light on — digitalWrite(13,HIGH);
2. Wait for 1 second — delay(1sec);
3. Turn light off — digitalWrite(13, LOW);
4. Wait for one second — delay(1sec);
5. repeat — repeat;

Make a light flash

1. Turn light on
2. Wait for 1 second
3. Turn light off
4. Wait for one second
5. repeat

```
loop()
{
  digitalWrite(13,HIGH);
  delay(1000);
  digitalWrite(13,LOW);
  delay(1000);
}
```

Very common to write things in "pseudocode" before writing the real program!

Make a light flash

```
void loop()                    // loop forever
{
  digitalWrite(13, HIGH); // set pin 13 HIGH
  delay(1000);                 // delay 1000ms (1sec)
  digitalWrite(13, LOW); // set pin 13 LOW
  delay(1000);                 // delay 1000ms (1sec)
}                              // go back to loop()
```

Make a light flash

```
void setup() {               // do once at first
pinMode(13, OUTPUT);  // make pin 13 an output
}

void loop() {                 // loop forever
  digitalWrite(13, HIGH); // set pin 13 HIGH
  delay(1000);                 // delay 1000ms (1sec)
  digitalWrite(13, LOW); // set pin 13 LOW
  delay(1000);                 // delay 1000ms (1sec)
}                              // go back to loop()
```

Required Arduino Functions

```
/* define global variables here */

void setup() {                    // run once, when the program starts
    <initialization statement>;   // typically pin definitions
    …                             // and other init stuff
    <initialization statement>;
}

void loop() {                     // run over and over again
   /* define local variables here */
   <main loop statement>;         // the guts of your program
    …                             // which could include calls
   <main loop statement>;         // to other functions…
}
```

"void" means that those functions do not return any values

Variables

- Like mailboxes – you can store a value in them and retrieve it later
- They have a "type"
  - tells you what values can be stored in them

```
// define a variable named "LEDpin"
// start it out with the value 13
int LEDpin = 13;
//you can now use LEDpin in your program
// Wherever you use it, the program will look inside
// and use the 13
```

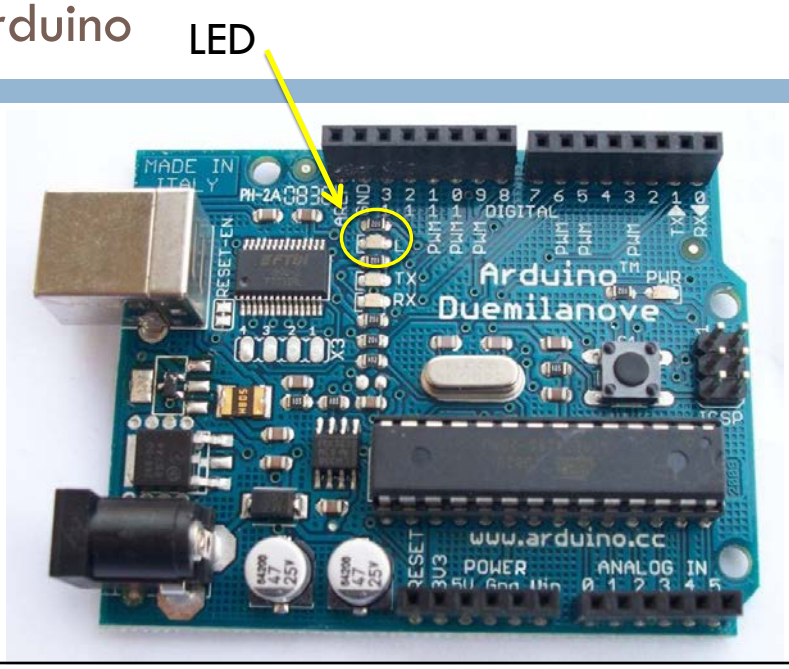
Blink Sketch (program)

```
/*
* Blink
* 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.
*/

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);                // wait for a second
    digitalWrite(ledPin, LOW);  // sets the LED off
    delay(1000);                // wait for a second
}
```

Variables

- Variable names must start with a letter or underscore
  - Case sensitive! - Foo and foo are different variables!
  - After the letter or underscore you can use numbers too
- Are these valid names?
  - Abc
  - 1st_variable
  - _123_
  - pinName
  - another name
  - a23-d
  - aNiceVariableName

Arduino



We just made an LED blink…Big Deal?

- *If you can switch things on and off under program control you can control the world!*
  - Most actuators may be switched on and off with a digital signal

We just made an LED blink…Big Deal?

- Most actuators can be switched on and off with a digital signal
  - The digitalWrite(pin,value); function is the software command that lets you control almost anything
- LEDs are easy!
  - Motors, servos, etc. are a little trickier, but not much
  - More on that later…
- Arduino has 14 digital pins (inputs or outputs)
  - can easily add more with external helper chips
  - More on that later…

Arduino Project

- Group project (groups of 3 – one artist/group)
- Arduino control – ***Sequence and randomize things***
  - Tell things when to happen…
- Outputs – ***make things move and light up…***
  - light (LED), movement (servo), sound (speaker)
- Inputs – ***Sense the environment around you***
  - switches (all sorts), knobs, light sensors

Project Teams (random…)

Alec Bang
Douglas Karmondy
Benjamin Shapiro

Jennifer Bohn
Zhoushiyuan Xue
Aaron Pabst

Ethan Edwards
Steve Corey
Donovan Bidlack

Nate Francis
Kylee Fluckiger
Bryan Hatasala

Laurie Larson
Nate Miller
Thomas Walker

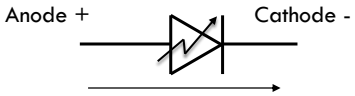
Emily McMurray
Cole Mortensen
Bryce Bartlett

Kate Wilhite
John Fresco
Eli Hebdon

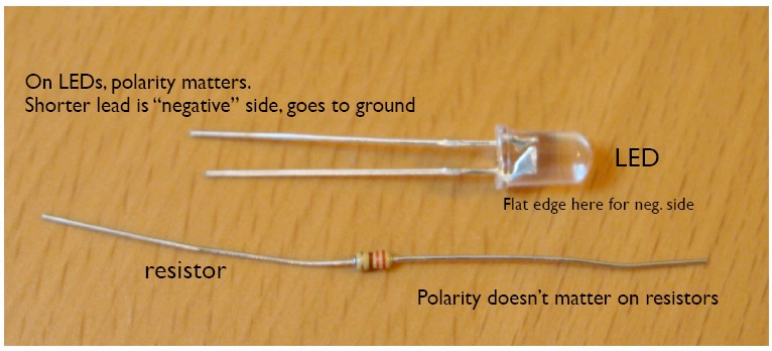
Arduino Project

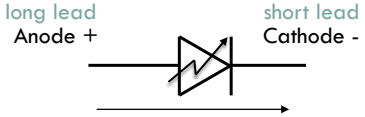
Blink Modifications

- Change to use an external LED rather than the one on the board
  - Connect to any digital pin
  - LED is on if current flows from Anode to Cathode
  - LED is on if the digital pin is HIGH, off if LOW
  - How much current do you use?
    - not more than 20mA
  - How do you make sure you don't use too much?
    - use a resistor
  - Pay attention to current! Use a current-limiting resistor!



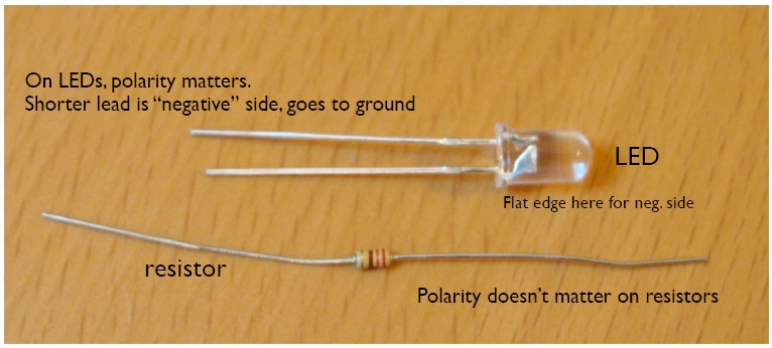
LEDs and Resistors

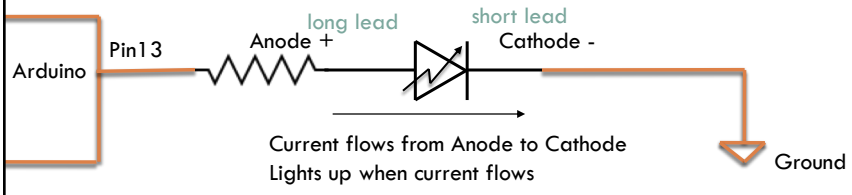




Current flows from Anode to Cathode
Lights up when current flows

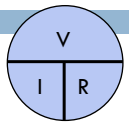
LEDs and Resistors





Current Limiting Resistor

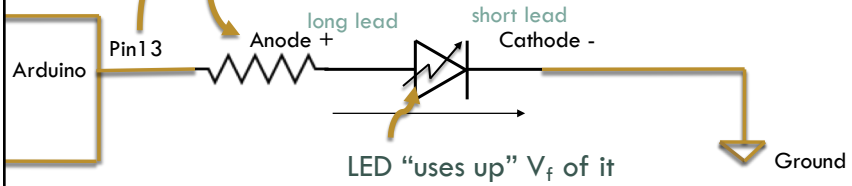
- Ohm's Law
  - V = IR I = V/R R = V/I



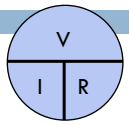
- Every LED has a V_f "Forward Voltage"
  - How much voltage is dropped (used up) passing through the LED

"HIGH" forces output pin to 5v (called V)

Resistor "uses up" the rest ($V - V_f$)

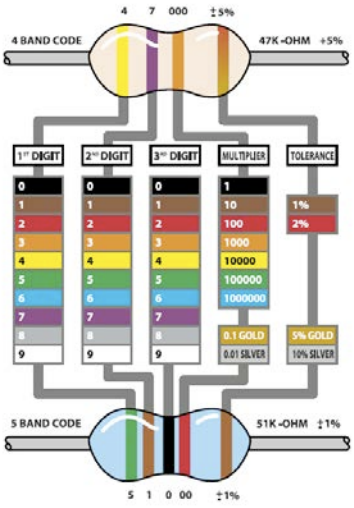


Current Limiting Resistor



- Ohm's Law
  - V = IR I = V/R R = V/I
- Every LED has a V_f "Forward Voltage"
  - How much voltage is dropped (used up) passing through the LED
- $R = (V - V_f) / I$
  - Example – If V_f is 1.9v (red LED), and V = 5v, and you want 15mA of current (0.015A)
  - R = (5 – 1.9)/0.015 = 3.1/0.015 = 206Ω
  - Exact isn't critical – use next size up, i.e. 220Ω
  - **Or be safe and use 330Ω or 470Ω**
  - This would result in 9.4mA or 6.6mA which is fine

Resistor Color Codes

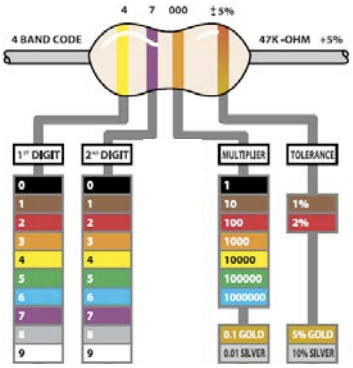


What's the color code for a 330Ω resistor?

What's the color code for a 1kΩ resistor?

What's the color code for a 10kΩ resistor

Resistor Color Codes



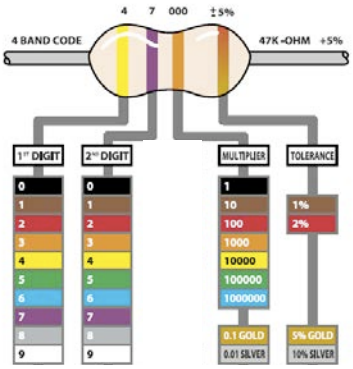
We're using 4-band 5% resistors with a ¼ watt rating

What's the color code for a 330Ω resistor?

What's the color code for a 1kΩ resistor?

What's the color code for a 10kΩ resistor

Resistor Color Codes



We're using 4-band 5% resistors with a ¼ watt rating

What's the color code for a 220Ω resistor?

orange orange brown gold

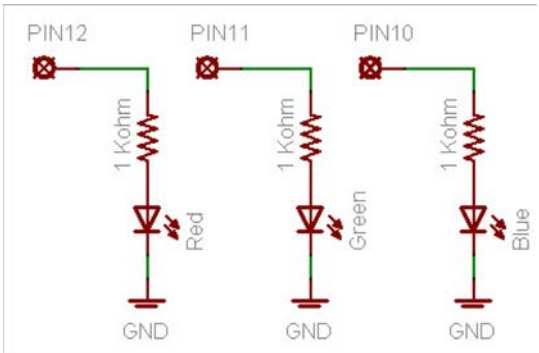
What's the color code for a 1kΩ resistor?

brown black red gold

What's the color code for a 470Ω resistor

brown black orange gold

Multiple LEDs



Arduino Code

```
int redPin = 12;                // Red LED connected to digital pin 12
int greenPin = 11;              // Green LED connected to digital pin 11

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

void loop()   {                 // run over and over again
  digitalWrite(redPin, HIGH);   // sets the Red LED on
  digitalWrite(greenPin, HIGH); // sets the Green LED on
  delay(500);                   // waits for half a second
  digitalWrite(redPin, LOW);    // sets the Red LED off
  digitalWrite(greenPin, LOW);  // sets the Green LED off
  delay(500);                   // waits for half a second
}
```

Add a diffuser

Add a diffuser

Online tutorial: ladyada.net

- http://www.ladyada.net/learn/arduino/lesson3.html

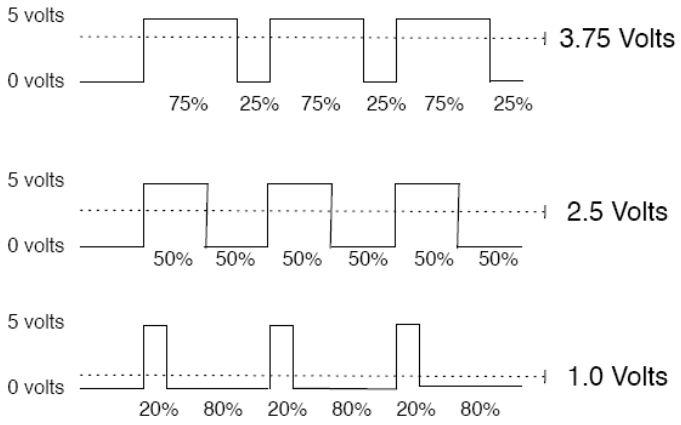
Changing brightness of an LED

- LEDs are (mostly) either on or off
  - No strong dependency on current – unlike incandescent bulbs..
- But, they turn on and off VERY quickly
  - So flash them on and off quickly, and they seem dimmer...

PWM

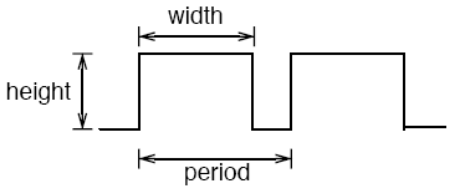
Output voltage is averaged from on *vs.* off time

output_voltage = (on_time / off_time) * max_voltage



PWM

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

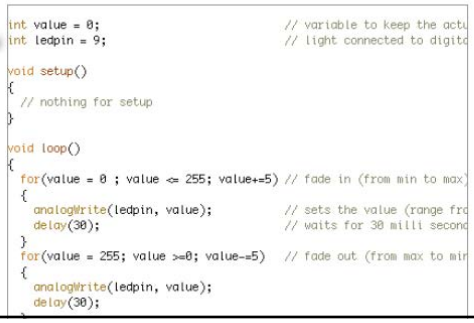


Pulse Width Modulation

- analogWrite(pin, value);
  - value can be 0 to 255
  - Must be one of the "PWM pins" : pins 3, 5, 6, 9, 10, 11
  - Don't need to set pinMode to OUTPUT (but won't hurt)

Load "File/Sketchbook/Examples/Analog/Fading"

note →

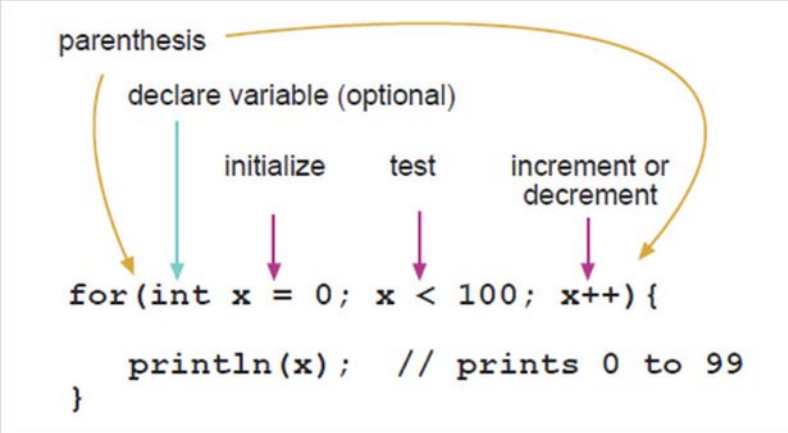


```
int value = 0;                          // variable to keep the actu
int ledpin = 9;                         // light connected to digita

void setup()
{
  // nothing for setup
}

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

C "for loop"

```
for (<initialization>; <condition>; <increment>) {
  // do something…
  }

int i;  // define an int to use as a loop variable
for (i = 0; i <= 255; i=i+1) { // repeat 256 times
  analogWrite(pin, i);       // write a value to the pin
  delay(50);                 // wait 50msec (0.05 sec)
}    // The loop will take 50*256 msec to execute
     // (12.8 sec)
```

Another view of a "for loop"



C "for" loop

```
for (<initialization>; <condition>; <increment>) {
  // do something…
  }

// You can also define the variable right in the loop
for (int i = 0; i <= 255; i=i+1) { // repeat 256 times
  analogWrite(pin, i);       // write a value to the pin
  delay(50);                 // wait 50msec (0.05 sec)
}    // The loop will take 50*256 msec to execute
     // (12.8 sec)
```

Aside: C Compound Operators

```
x = x + 1;  // adds one to the current value of x
x += 5;     // same as x = x + 5
x++;        // same as x = x + 1

x = x – 2;  // subtracts 2 from the current vale of x
x -= 3;     // same as x = x - 3
x--;        // same as x = x – 1

x = x * 3;  // multiplies the current value of x by 3
x *=5;      // same as x = x * 5
```

Fading Program

```
int ledPin = 9;    // LED connected to digital pin 9
void setup() {
    // nothing happens in setup (Why not?)
}
void loop() {
   // fade in from min to max in increments of 5 points:
   for (int fadeValue = 0 ; fadeValue <= 255; fadeValue +=5) {
    analogWrite(ledPin, fadeValue);     // sets the value (range from 0 to 255):
   delay(30);     // wait for 30 milliseconds between brightness steps
   }

    // fade out from max to min in increments of 5 points:
    for (int fadeValue = 255 ; fadeValue >= 0; fadeValue -=5) {
   analogWrite(ledPin, fadeValue); // sets the value (range from 0 to 255):
   delay(30); // wait for 30 milliseconds between dimming steps
    }
}
```

Modified Fading

- What would you change to make things behave differently?
- Can you predict the effect of your changes?
- Loops are important – a general way to repeat things over and over
  - You don't always have to repeat a fixed number of times
  - foo = 30;
    for (int i =0; i < foo; i++) { … } // loop "foo" times

Moving on…

- Write a program to make the LED flicker like a flame
  - Choose a random intensity
  - For a random amount of time
- Use analogWrite(ledPin, val) to change brightness
- Main loop repeats itself forever…
  - Set the value of the brightness to a random value
  - Wait for a random amount of time
  - repeat
- The effect looks like flickering…

Flickering Pseudocode

1. Set the LED to a random brightness
2. Wait for a random amount of time
3. repeat

Additional Programming

- Generate random number
  random(<min>,<max>)
  - Returns random number between min and max-1

```
random(2, 5); // returns random number between 2 and 4

random(30); // returns random number between 0 and 29
```

Flickering Pseudocode

1. Pick a random number between 100-254
2. Set LED to that brightness (use analogWrite)
3. Pick another random number between 10-150
4. Wait for that amount of time (in ms)
5. Repeat

```
int brightness;
brightness = random(100, 255);
```

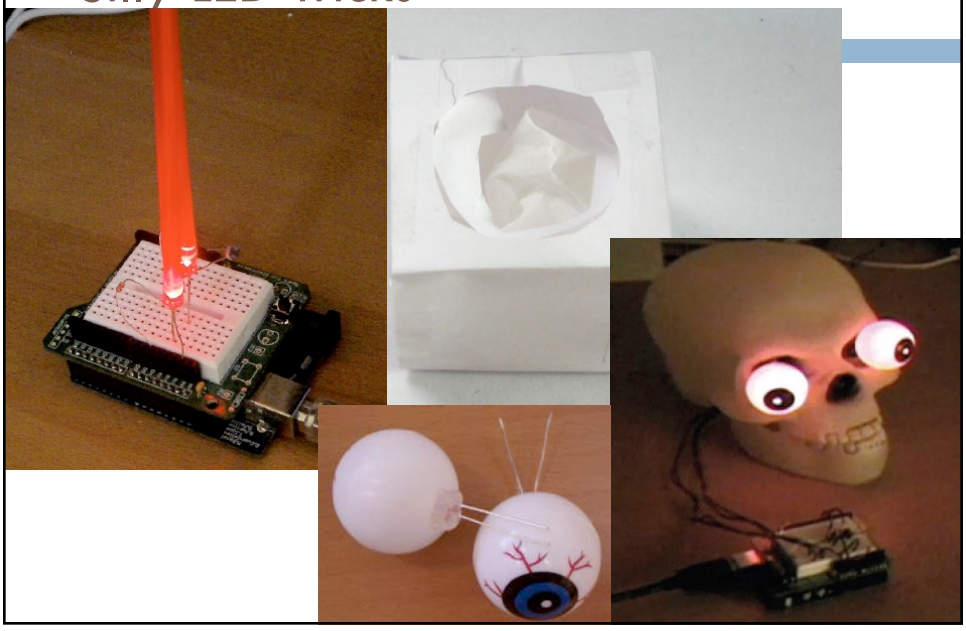
Candle Program

```
int ledPin = 9;                 // select pin for LED output
int bright = 0;                 // Variable to hold LED brightness
int time = 0;                   // variable to hold delay time

void setup () {
    randomSeed(0);              // initialize the random function
    pinMode(ledPin, OUTPUT);    // ledPin should be an output
}

void loop () {
    bright = random(100, 255);      // random brightness value
    analogWrite(ledPin, bright);    // set the LED brightness

    time = random(10,150);          // random time in ms
    delay(time);                    // delay for that time
}
```

Candle Program (smaller)

```
int ledPin = 9;                 // select pin for LED output
void setup () {
   pinMode(ledPin, OUTPUT);   // ledPin should be output
}

void loop () {
   analogWrite(ledPin, random(100, 255));  // LED brightness
   delay(random(10,150)) // delay for random time
}
```

Silly LED Tricks



Summary – Whew!

- Digital Pins
  - use pinMode(<pin>, <INPUT/OUTPUT>) for setting direction
    - Put these in the setup() function
    - pinMode(13, OUTPUT); // set pin 13 as an output
  - use digitalWrite(<pin>, <HIGH/LOW>) for on/off
    - digitalWrite(LEDpin, HIGH); // turn on pin "LEDpin"
  - use analogWrite(<pin>, <val>) for PWM dimming
    - values from 0 – 255
    - PWM pins are 3, 5, 6, 9, 10, 11
    - analogWrite(9, 235); // set LED on pin 9 to somewhat bright

More Summary

- delay(val) delays for val-number of milliseconds
  - milliseconds are thousandths of a sec (1000msec = 1sec)
  - delay(500); // delay for half a second
- random(min,max) returns a random number between min and max-1
  - You get a new random number each time you call the function
  - foo = random(10, 255); // assign foo a random # from // 10 to 254

More Summary

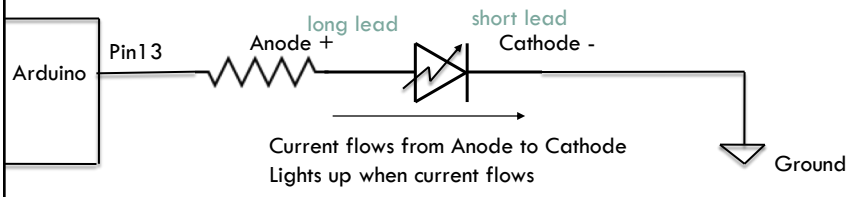
- Two required Arduino functions
  - void setup() { … } // executes once at start for setup
  - void loop() { … } // loops forever
    - statements execute one after the other inside loop, then repeat after you run out
- int i = 10; // define an int variable, initial value 10
- Other types of variables:
  - char – 8 bits
  - long - 32 bits
  - unsigned…
  - float – 32 bit floating point number

Still More Summary

- for (<start>; <stop>; <change>) { … }
  - for (int i=0; i<8; i++) { … } // loop 8 times
    // the value of i in each iteration is 0, 1, 2, 3, 4, 5, 6, 7
- if (<condition>) { … }
  - if (foo < 10) {digitalWrite(ledPin, HIGH);}
- if (<condition>) { …} else { … }
  - if (num == 10) { <do something> }
    else { <do something else> }

Last Summary (for now)

- LEDs – turn on when current flows from anode to cathode
  - Always use a current-limiting resistor!
  - Remember your resistor color codes
  - 220-470 ohm are good, general-purpose values for LEDs
  - Drive from Arduino on digital pins
  - Use PWM pins if you want to use analogWrite for dimming



Resources

- http://arduino.cc/en/Tutorial/HomePage
- http://www.ladyada.net/learn/arduino/index.html
- http://todbot.com/blog/bionicarduino/
- http://todbot.com/blog/spookyarduino/
- http://sheepdogguides.com/arduino/aht0led.htm

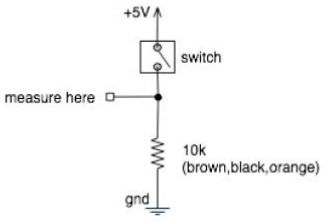
Getting Input (Digital)

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

How do you go from make/break to high/low?

Switches

- Digital inputs can "float" between 0 and 5 volts
- Resistor "pulls down" input to ground (0 volts)
- Pressing switch sets input to 5 volts
- Press is HIGH
  Release is LOW

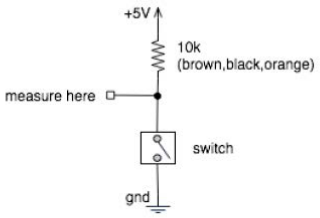


"pull-down"

Why do we need the "pull down" resistor?

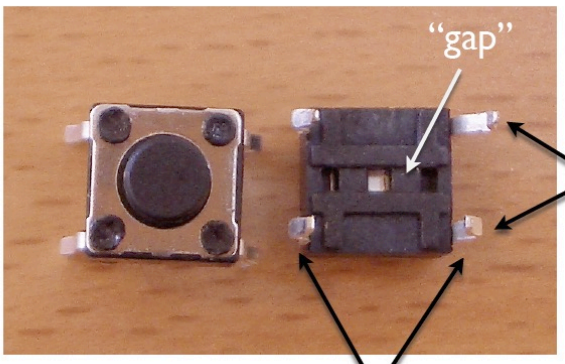
Another Switch

- Resistor pulls up input to 5 volts
- Switch sets input to 0 volts
- But now the sense is inverted
  - Press is LOW
  - Release is HIGH



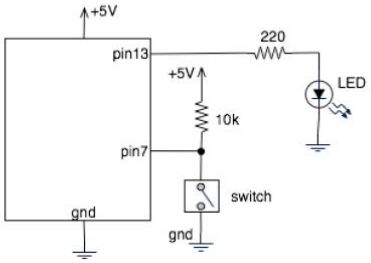
"pull-up"

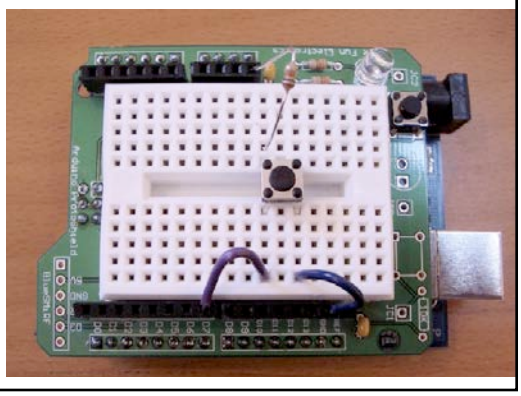
A Switch



Pressing the button, "closes the gap"

Using a Switch





Using digitalRead()

- Assume int myPin = 5; // pick a pin
- in setup() – use pinMode(myPin, INPUT);
- in loop() – use digitalRead(myPin)
  - int foo;
    foo = digitalRead(myPin);
    if (foo == 1) {do something}

digitalRead(pin);

```
// constants won't change. They're used here to set pin numbers:
    const int buttonPin = 2;     // the number of the pushbutton pin
    const int ledPin =  13;      // the number of the LED pin
// variables hold values that will change:
    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) {              // buttonState HIGH means pressed
    digitalWrite(ledPin, HIGH); }        // turn LED on:
    else { digitalWrite(ledPin, LOW); }// turn LED off:
    }
}
```

digitalRead(pin);

```
// define's are also constants They're used here to set pin numbers:
    #define buttonPin  2     // the number of the pushbutton pin
    #define ledPin 13        // the number of the LED pin
// variables hold values that will change:
    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) {              // buttonState HIGH means pressed
    digitalWrite(ledPin, HIGH); }        // turn LED on:
    else { digitalWrite(ledPin, LOW); }// turn LED off:
    }
}
```

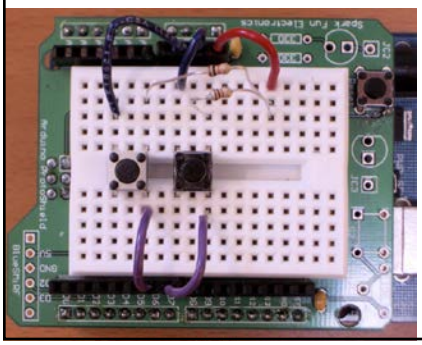
Moving on…

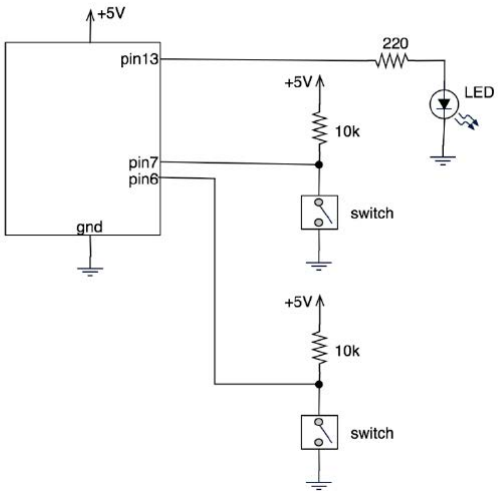
- Write a program that reads the value on an input pin
  - Use the button to change from blinking fast to blinking slow

```
int ledPin = 13; // choose the pin for the LED
int inPin = 7;   // choose the input pin (for a pushbutton)
int val = 0;     // variable for reading the pin status
int delayval = 100;

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

void loop(){
  val = digitalRead(inPin);  // read input value

  if( val == HIGH )
    delayval = 1000;
  else
    delayval = 100;

  digitalWrite(ledPin, HIGH);  // blink the LED and go OFF
  delay(delayval);
  digitalWrite(ledPin, LOW);
  delay(delayval);
}
```

Multiple Switches

Same sub-circuit, just duplicate



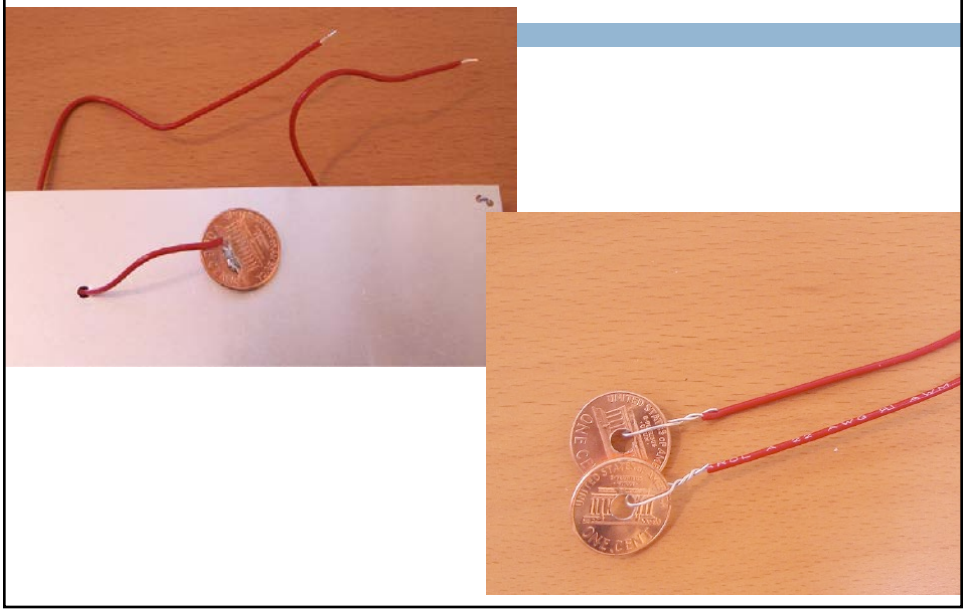


Make Your Own Switches

- Anything that makes a connection
- Wires, tin foil, tinfoil balls, ball bearings
- Pennies!
- Nails, bolts, screws

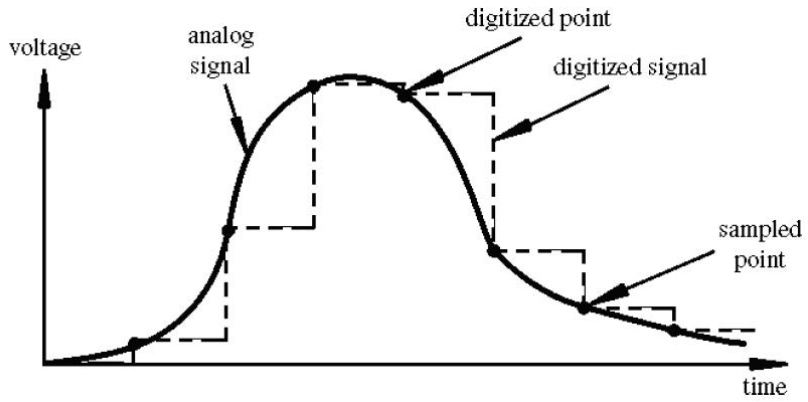
- Or repurpose these tiny switches as bump detectors or closure detectors

Make Your Own Switches



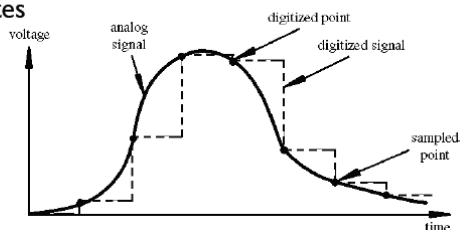
Analog Input

To computers, analog is chunky



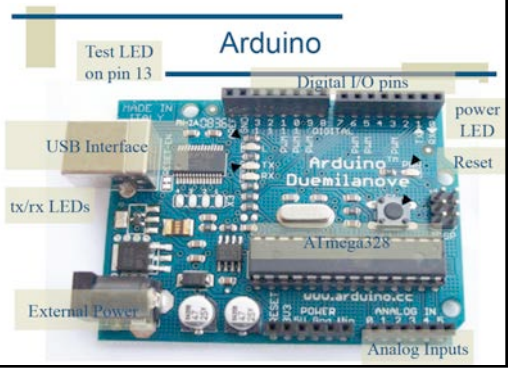
Analog Input

- Many states, not just two (HIGH/LOW)
- Number of states (or "bins") is *resolution*
- Common computer resolutions:
  - 8-bit = 256 states
  - 16-bit = 65,536 states
  - 32-bit = 4,294,967,296 states



Analog Input on Arduino

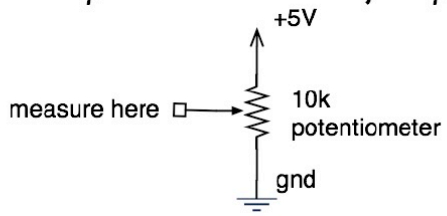
- Our version uses ATMega328p
  - six ADC inputs (Analog to Digital Converter)
  - Voltage range is 0-5v
  - Resolution is 10 bits (digital values between 0-1023)
  - In other words, 5/1024 = 4.8mV is the smallest voltage change you can measure
- analogRead(pin);
  - reads an analog pin
  - returns a digital value between 0-1023
  - analog pins need no pinMode declaration

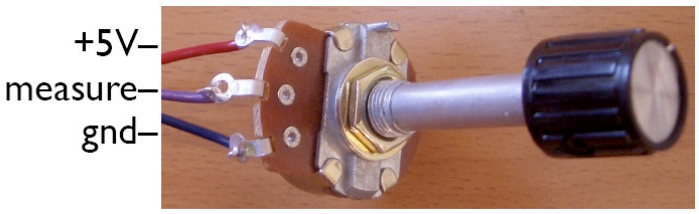


Analog Input

Sure sure, but how to make a varying voltage?

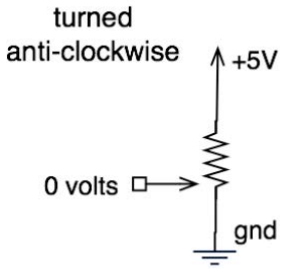
With a *potentiometer*. Or just *pot*.

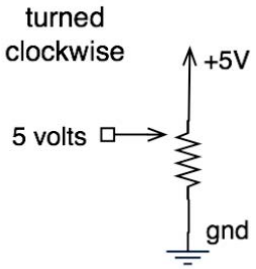


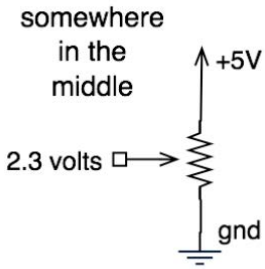


Potentiometers

Moving the knob is like moving
where the arrow taps the voltage on the resistor





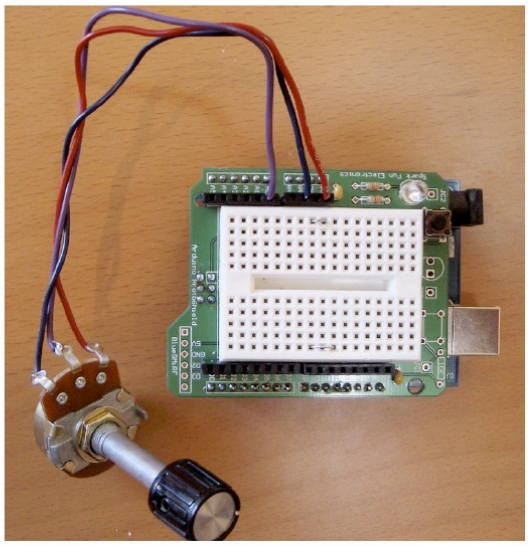


Arduino Analog Input

Red to Vcc

Purple to A0

Blue to Gnd



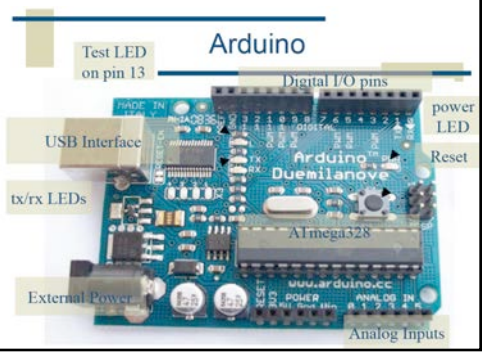
```
int sensorPin = A0;    // select the input pin for the potentiometer
int ledPin = 13;       // select the pin for the LED
int sensorValue = 0;  // variable to store the value coming from the sensor

void setup() {
pinMode(ledPin, OUTPUT); // declare the ledPin as an OUTPUT:
// Note that you don't need to declare the Analog pin – it's always input
}

void loop() {
  sensorValue = analogRead(sensorPin); // read the value from the sensor:
  digitalWrite(ledPin, HIGH); // turn the ledPin on
  delay(sensorValue); // stop the program for <sensorValue> milliseconds:
  digitalWrite(ledPin, LOW); // turn the ledPin off:
  delay(sensorValue); // stop the program for for <sensorValue> milliseconds:
}
```

Moving on…

- Write a program to read an analog value from a pot and use that value to control the brightness of an LED
  - Fade the LED by turning the pot
  - Useful function is
    map(value, fromlow, fromhigh, tolow, tohigh);

    y = map(x, 0, 1023, 50, 150);

  - Also remember
    analogWrite(pin,value);
    - PWM value from 0-255



potFade

```
int potPin = 0;          // the analog input pin from the pot
int ledPin = 9;          // pin for LED (a PWM pin)
int val;                 // Variable to hold pot value

void setup () {
   pinMode(ledPin, OUTPUT);          // declare ledPin as output
   pinMode(potPin, INPUT);           // potPin is an input
}

void loop () {
   val = analogRead(potPin);         //read the value from the pot
   val = map(val, 0, 1023, 100, 255); // map to reasonable values
   analogWrite(ledPin, val);         // write it to the LED
}
```

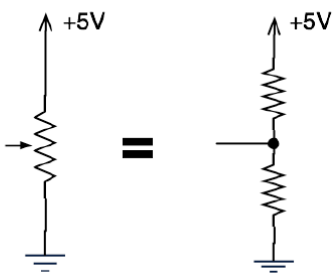
What good are pots?

- Anytime you need a ranged input
  - (we're used to knobs)
- Measure rotational position
  - steering wheel, etc.

- But more importantly for us, potentiometers are a good example of a *resistive sensor*

Sensing the Dark

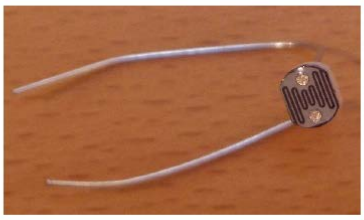
- Pots are example of a *voltage divider*
- Voltage divider splits a voltage in two
- Same as two resistors, but you can vary them



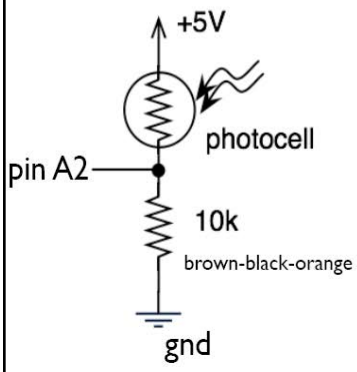
Sensing the Dark: Photocells

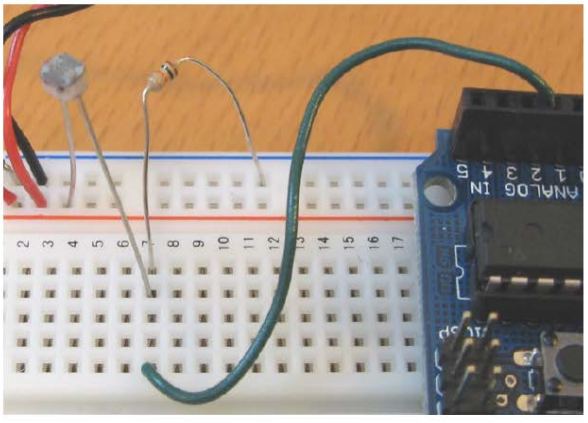
- aka. photoresistor, light-dependent resistor
- A *variable* resistor
- Brighter light == lower resistance
- Photocells you have range approx. 0-10k

schematic symbol



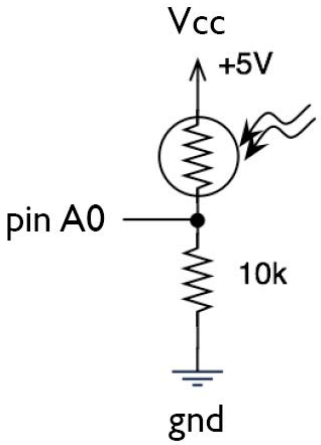
Photocell Circuit





Moving on…

- Connect a photocell instead of a pot to your fading circuit
  - Do you get the same range of fade as with the pot?
  - Why or why not?



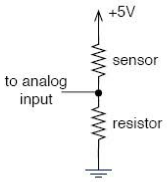
potFade

```
int lightSensePin = A0;    // the analog input pin from the light sensor
int ledPin = 9;            // pin for LED (a PWM pin)
int val;                   // Variable to hold light sensor value

void setup () {
   pinMode(ledPin, OUTPUT);          // declare ledPin as output
   pinMode(lightSensePin, INPUT);    // lightSensorPin is an input
}

void loop () {
   val = analogRead(lightSensePin);  //read the value from the sensor
   val = map(val, 0, 1023, 100, 255); // map to reasonable values
   analogWrite(ledPin, val);          // write it to the LED
}
```

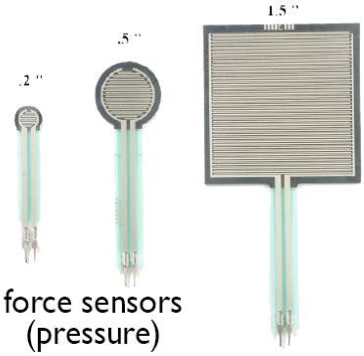
Resistive sensors

thermistor
(temperature)

circuit is the same
for all these



photocell
(light)



force sensors
(pressure)

flex sensor
(bend, deflection)

also air pressure
and others

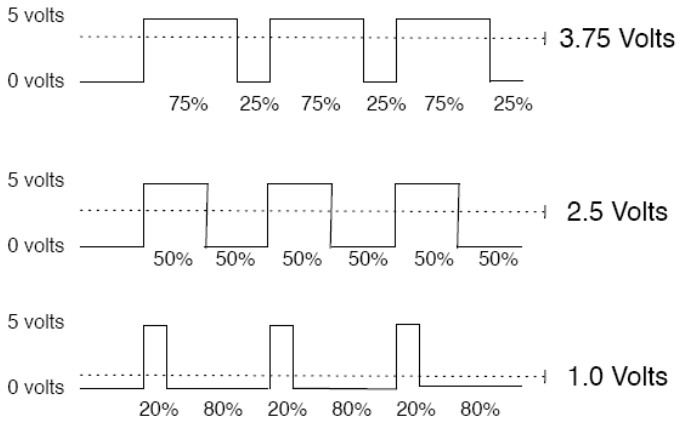
Moving on… Servos

- Servo motors are small DC motors that have a range of motion of 0-180°
  - Internal feedback and gearing to make it work
  - easy three-wire interface
  - position is controlled by PWM signals

PWM

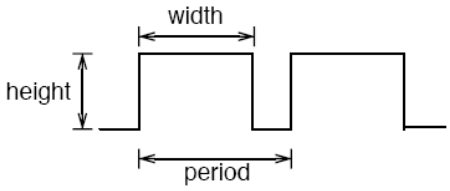
Output voltage is averaged from on *vs.* off time

output_voltage = (on_time / off_time) * max_voltage



PWM

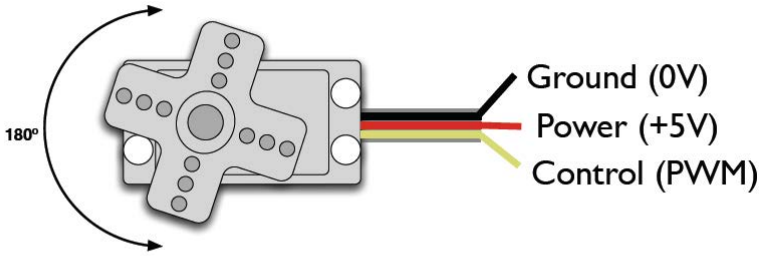
- Used everywhere
  - Lamp dimmers, motor speed control, power supplies, noise making
- Three characteristics of PWM signals
  - Pulse width range (min/max)
  - Pulse period (= 1/pulses per second)
  - Voltage levels (0-5V, for instance)



Servomotors

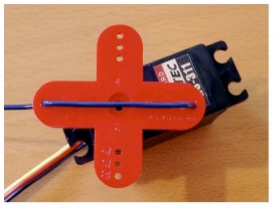
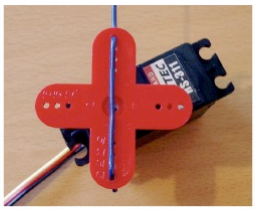
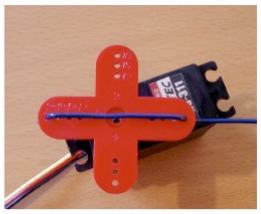
- Can be positioned from 0-180° (usually)
- Internal feedback circuitry & gearing takes care of the hard stuff
- Easy three-wire PWM 5V interface

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

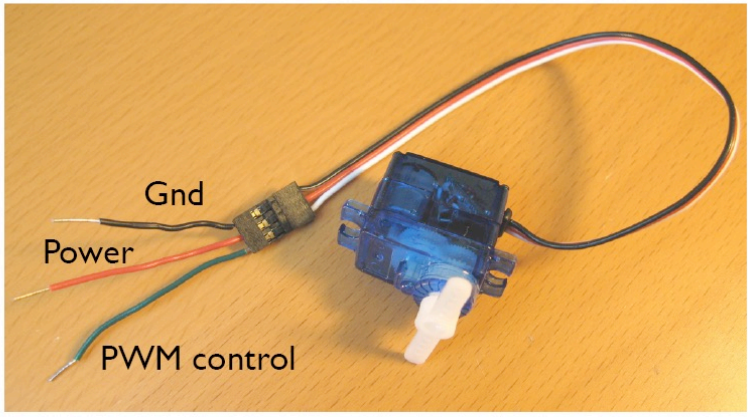
Servo Movement

0 degrees	90 degrees	180 degrees
		
1000 microsecs	1500 microsecs	2000 microsecs

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

Servo and Arduino

First, add some jumper wires to the servo connector



Servo Example Program

```
#include <Servo.h> // include the built-in servo library
Servo myservo;      // create a servo object to control the servo (one per servo)
int pos = 0;        // variable to store the servo position

void setup() {
    myservo.attach(9);          // attach servo control to pin 9
}
void loop() {
    for (pos = 0; pos < 180; pos++) {     // go from 0 to 180 degrees
            myservo.write(pos);             // move the servo

            delay(15);l                     // give it time to get there
    }
    for (pos = 180; pos>=1; pos--) {    // wave backwards
            myservo.write(pos);
            delay(15);
    }
}
```

Servo Functions

- Servo is a class
  - Servo myservo; // creates an instance of that class
- myservo.attach(pin);
  - attach to an output pin (doesn't need to be PWM pin!)
  - Servo library can control up to 12 servos on our boards
  - but a side effect is that it disables the PWM on pins 9 and 10
- myservo.write(pos);
  - moves servo – pos ranges from 0-180
- myservo.read();
  - returns the current position of the servo (0-180)

Moving on…

- Write a program to control the position of the servo from a pot, or from a photocell
  - remember pot analogRead(); values are from 0-1023
  - measure the range of values coming out of the photocell first?
  - use Serial.print(val); for example
  - use map(val, in1, in2, 0, 180); to map in1-in2 values to 0-179
  - Can also use constrain(val, 0, 179);

Side Note - Power

- Servos can consume a bit of power
  - We need to make sure that we don't draw so much power out of the Arduino that it fizzles
  - If you drive more than a couple servos, you probably should put the servo power pins on a separate power supply from the Arduino
  - Use a wall-wart 5v DC supply, for example

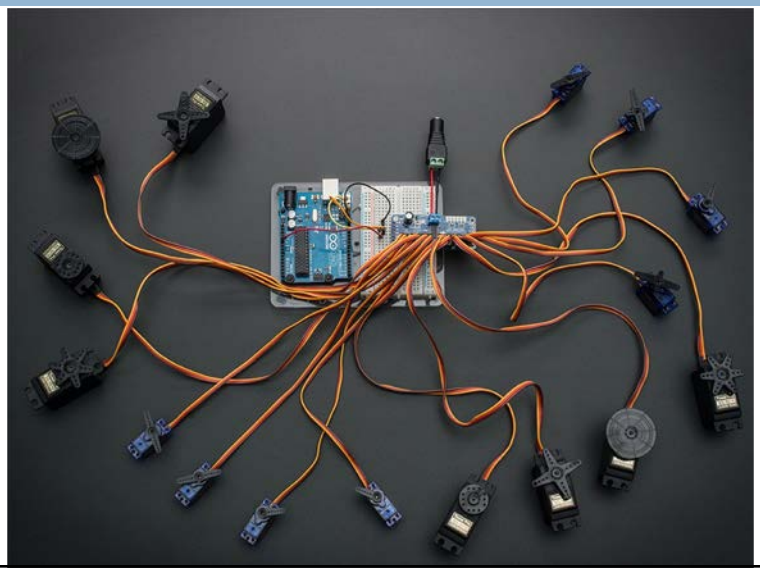
Servo Driver Board

Control connection to Arduino

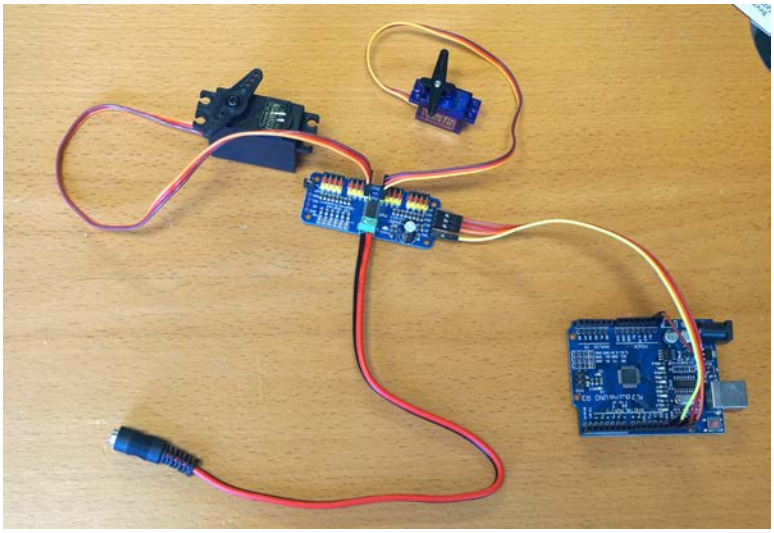
Power connector for servos

16 servo connections

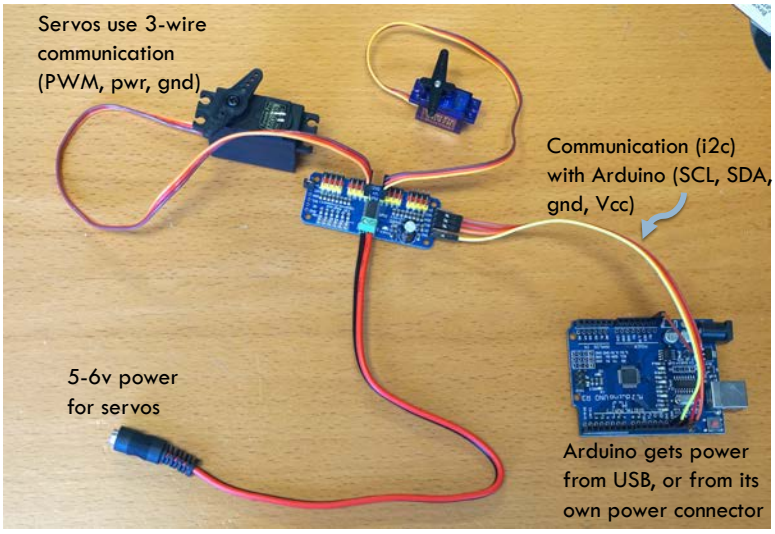
Servo Driver Board



Servo driver board

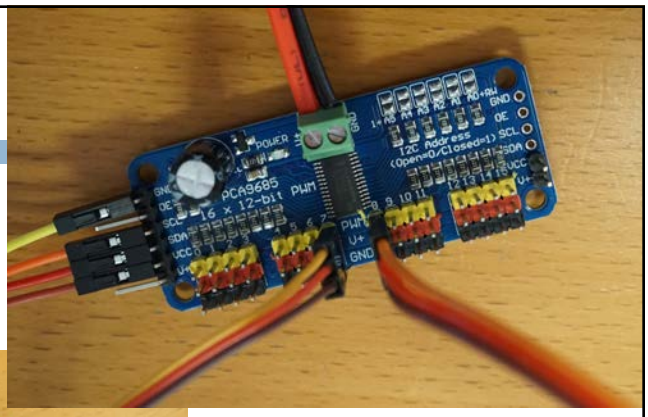


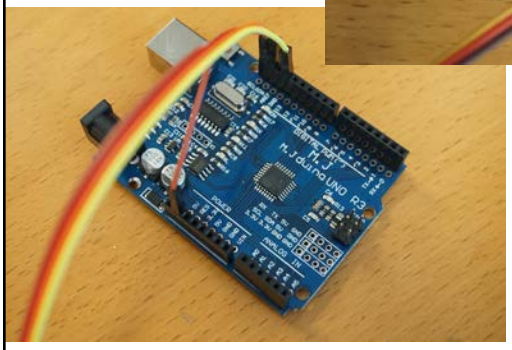
Servo driver board



Servo driver

Servos plug in to the 3-prong connectors. Middle is pwr, darker wire is gnd, lighter wire is PWM.





Vcc comes from Arduino (5v) and powers the logic on the servo driver board

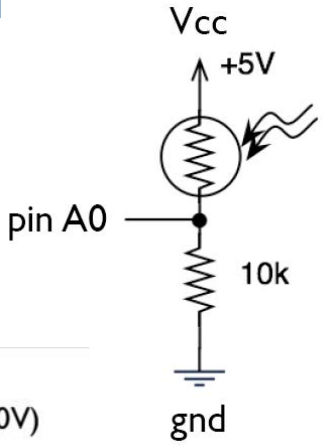
V+ comes to the green connector and is for the servos. It's separate from the Vcc power!

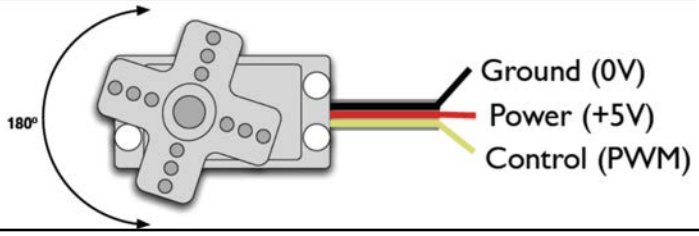
Communication is on the i2c connections (SCL, SDA).

```
#include <Wire.h>                    // library for i2c communication
#include <Adafruit_PWMServoDriver.h>  // library for the servo driver board

Adafruit_PWMServoDriver driverBoard = Adafruit_PWMServoDriver();

#define SERVOMIN  160       //  'minimum' pulse length count (out of 4096)
#define SERVOMAX  520       //  'maximum' pulse length count (out of 4096)

// define a couple of positions on the servo board for the test servos
#define SERVO1 7 // poistion 7 on the servo board
#define SERVO2 8 // position 8 on the servo board

void setup() {
  driverBoard.begin();
  driverBoard.setPWMFreq(60);
  delay(10); // delay to get things settled...
}

void loop() {
  int pulse = 0; // A variable to hold the pulse width that corresponds to a degree value
  // Take both servos and sweep them from 0 to 179 degrees
   for (int pos = 0; pos <= 179; pos += 1) {
    servoDegree(SERVO1, pos); // set servo to the degree position in variable 'pos'
    servoDegree(SERVO2, pos); // set servo to the degree position in variable 'pos'
    delay(15);                // waits 15ms for the servo to reach the position
  }
  for (int pos = SERVOMAX; pos >= SERVOMIN; pos -= 1) {
    driverBoard.setPWM(SERVO1, 0, pos); // set servo to the position in variable 'pos'
    driverBoard.setPWM(SERVO2, 0, pos); // set servo to the to position in variable 'pos'
    delay(15);                  // waits 15ms for the servo to reach the position
  }
}

// Helper function to set the servo number servoNum to a degree position
void servoDegree(int servoNum, int degree) {
  driverBoard.setPWM(servoNum, 0, map(degree, 0, 179, SERVOMIN, SERVOMAX));
}
```

Servo/Light Practice

- Use a photocell on the input
  - put in series with 10k ohm resistor
- use a servo on the output
  - connect to a PWM pin
- make the servo do something in response to the amount of light falling on the photocell





Summary – Whew!

- LEDs – use current limiting resistors (330Ω) (remember color code!)
  - drive from digitalWrite(pin,val); for on/off
  - drive from analogWrite(pin,val); for PWM dimming (values from 0-255)
- buttons – current limiting resistors again (10k Ω)
  - active-high or active low (pullup or pulldown)
  - read with digitalRead(pin);
- potentiometers (pots)– voltage dividers with a knob
  - use with analogRead(pin); for values from 0-1023

Knob-Servo

```
#include <Servo.h>

Servo myservo;    // create servo object to control a servo
int servoPin = 9; // a pin to connect the servo to

int potPin = A0;  // analog pin used to connect the potentiometer
int val;          // variable to read the value from the analog pin

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

void loop()
{
  val = analogRead(potPin);            // reads the value of the potentiometer (value between 0 and
  val = map(val, 0, 1023, 0, 179);     // scale it to use it with the servo (value between 0 and 180
  myservo.write(val);                  // sets the servo position according to the scaled value
  delay(20);                           // waits for the servo to get there
}
```

Summary – Whew!

- photocells – variable resistors
  - use with current-limiting resistors (1k-10k) (to make voltage divider)
- Servos – use Servo library to control motion
  - might need external power supply
  - range of motion 0-180°
- Also setup( ) and loop( ) functions, and various libraries