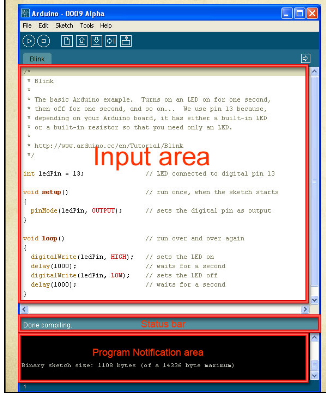
Arduino Hands-On

CS5968 / ART4455

Disclaimer

- Many of these slides are mine
- But, some are stolen 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

Part 1 – Arduino SW

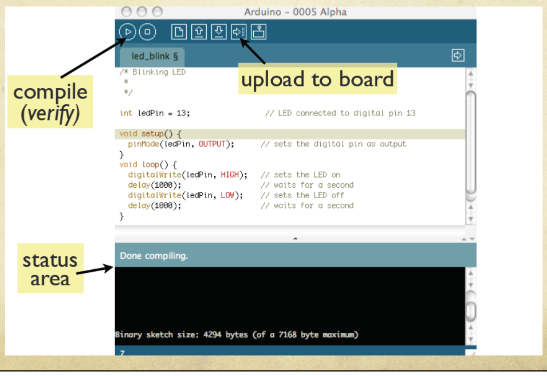


```
/*
 * Blink
 *
 * The basic Arduino example.  Turns on an LED on for one second,
 * then off for one second, and so on...  We use pin 13 because,
 * depending on your Arduino board, it has either a built-in LED
 * or a built-in resistor so that you need only an LED.
 *
 * http://www.arduino.cc/en/Tutorial/Blink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

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
}
```

```
Done compiling.
Binary sketch size: 1108 bytes (of a 14336 byte maximum)
```

Remember, Arduino calls programs "sketches"

Part 1 – Arduino SW



```
/* Blinking LED
 *
 */

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

void setup() {
  pinMode(ledPin, OUTPUT);      // sets the digital pin as output
}
void loop() {
  digitalWrite(ledPin, HIGH);   // sets the LED on
  delay(1000);                  // waits for a second
  digitalWrite(ledPin, LOW);    // sets the LED off
  delay(1000);                  // waits for a second
}
```

```
Done compiling.
Binary sketch size: 4294 bytes (of a 7168 byte maximum)
```

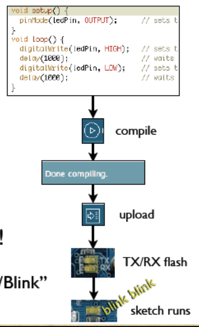
Procedure

Using Arduino

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

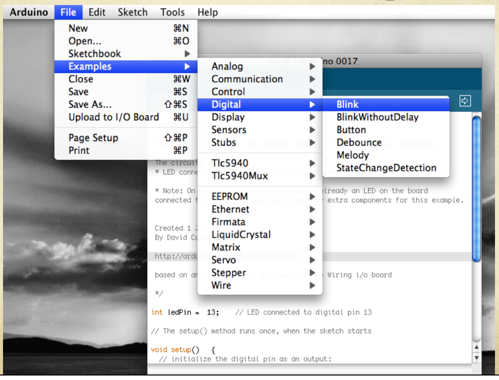
Try it out with the "Blink" sketch!

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



```
void setup() {
  pinMode(ledPin, OUTPUT);      // sets t
}
void loop() {
  digitalWrite(ledPin, HIGH);   // sets t
  delay(1000);                  // waits
  digitalWrite(ledPin, LOW);    // sets t
  delay(1000);                  // waits
}
```

Get the Blink Example

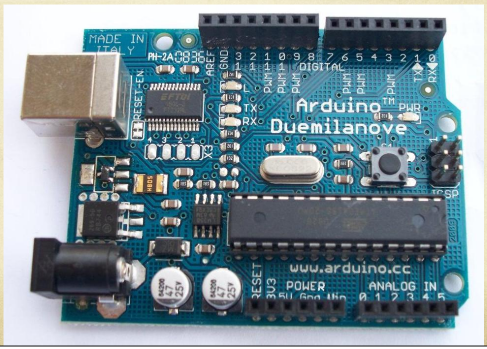


```
int ledPin =  13;    // LED connected to digital pin 13

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

void setup()   {
  // initialize the digital pin as an output:
```

Blink Sketch (program)

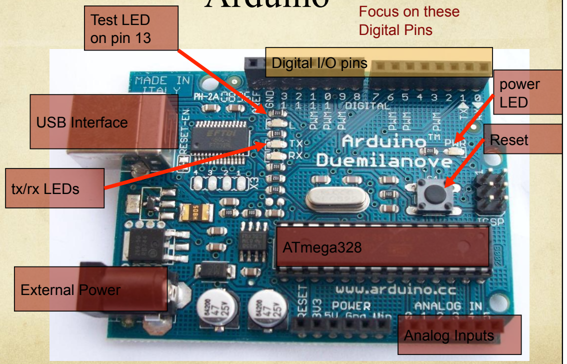
```
/*
 * Blink
 *
 * The basic Arduino example.  Turns on an LED on for one second,
 * then off for one second, and so on...  We use pin 13 because,
 * depending on your Arduino board, it has either a built-in LED
 * or a built-in resistor so that you need only an LED.
 */

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
}
```

Arduino



Arduino

Focus on these Digital Pins



Arduino Functions

- Each of the 14 digital pins is controlled by program statements
  - pins are numbered 13 to 0
- pinMode(<pinNumber>, <INPUT/OUTPUT>)
  - Define whether the pin is used for input or output
  - e.g. pinMode(13, OUTPUT);
  - Pins are OUTPUT by default…
- digitalWrite(<pinNumber>, <HIGH/LOW>)
  - Drive the output to a HIGH or LOW voltage (5v or 0v)
  - e.g. digitalWrite(13,HIGH);
- digitalRead(<pinNumber>)
  - read a value on an input pin
  - e.g. digitalRead(8);

(almost) all statements end with a semicolon!

Arduino Program

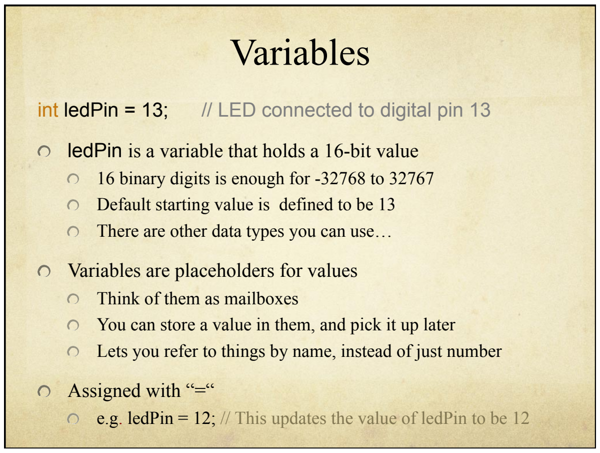
- One section for setting things up
  - pinMode(13, OUTPUT);
    pinMode(12, INPUT);
- One section repeats forever – lines of code execute one at a time
  - digitalWrite(13,HIGH);
    delay(1000);
    digitalWrite(13,LOW);
    delay(1000);
  - repeat forever…

Add Comments…

Comments are just notes to the reader. They are NOT code

- One section for setting things up
  - pinMode(13, OUTPUT); // pin 13 is the output LED
    pinMode(12, INPUT);   // pin 12 is the pushbutton
- One section repeats forever – lines of code execute one at a time
  - digitalWrite(13,HIGH); // Set 13 high (LED lit)
    delay(1000);   // delay for 1 sec (1000 ms)
    digitalWrite(13,LOW); // set 13 low (LED Off)
    delay(1000);   // wait for 1sec
  - repeat forever…

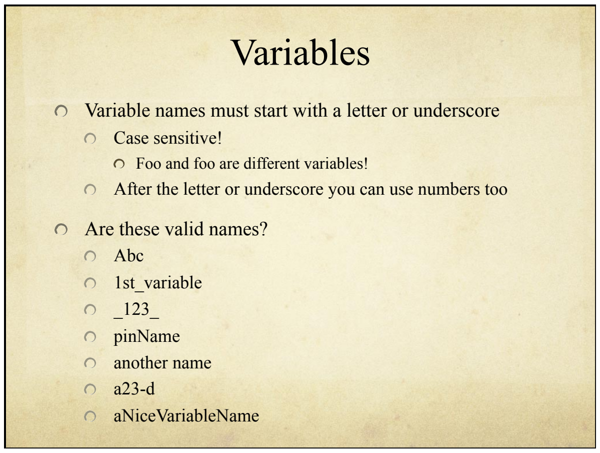
// means everything to the end of the line is a comment
/* starts a comment, (which might be multiple lines).
the comment is ended with a */



Variables

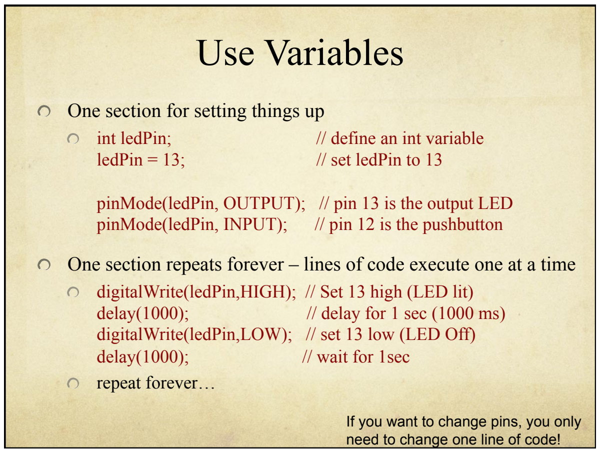
```
int ledPin = 13;     // LED connected to digital pin 13
```

- ledPin is a variable that holds a 16-bit value
  - 16 binary digits is enough for -32768 to 32767
  - Default starting value is defined to be 13
  - There are other data types you can use…
- Variables are placeholders for values
  - Think of them as mailboxes
  - You can store a value in them, and pick it up later
  - Lets you refer to things by name, instead of just number
- Assigned with "="
  - e.g. ledPin = 12; // This updates the value of ledPin to be 12



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



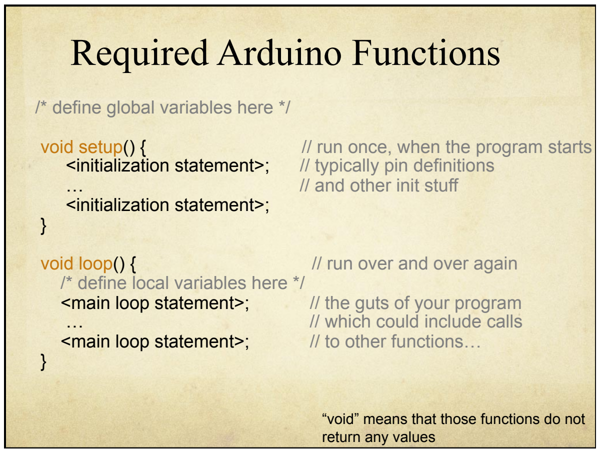
Use Variables

- One section for setting things up

  -
    ```
    int ledPin;                    // define an int variable
    ledPin = 13;                   // set ledPin to 13

    pinMode(ledPin, OUTPUT);   // pin 13 is the output LED
    pinMode(ledPin, INPUT);    // pin 12 is the pushbutton
    ```
- One section repeats forever – lines of code execute one at a time

  -
    ```
    digitalWrite(ledPin,HIGH);  // Set 13 high (LED lit)
    delay(1000);                // delay for 1 sec (1000 ms)
    digitalWrite(ledPin,LOW);   // set 13 low (LED Off)
    delay(1000);                // wait for 1sec
    ```
  - repeat forever…

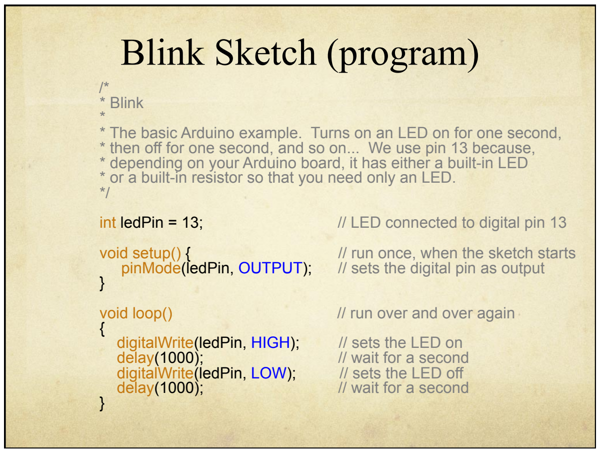
If you want to change pins, you only need to change one line of code!



Required Arduino Functions

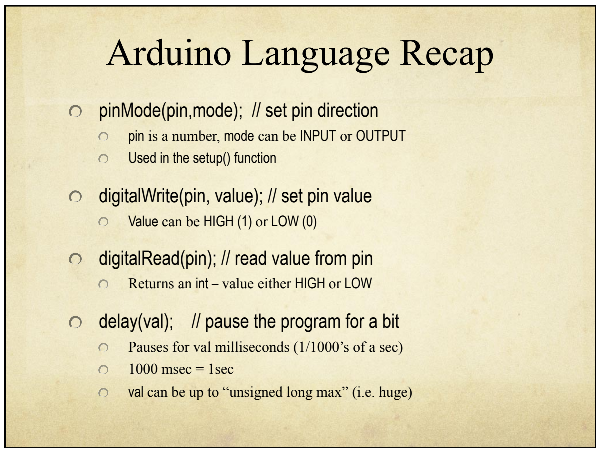
```
/* define global variables here */

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



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
}
```



Arduino Language Recap

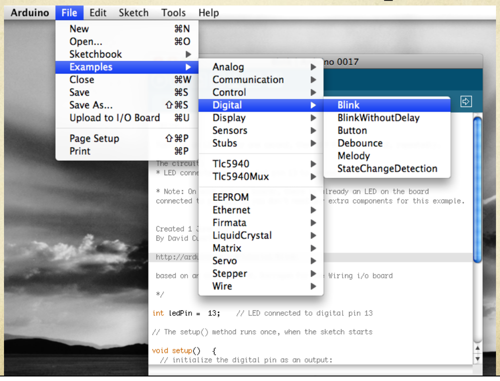
- pinMode(pin,mode); // set pin direction
  - pin is a number, mode can be INPUT or OUTPUT
  - Used in the setup() function
- digitalWrite(pin, value); // set pin value
  - Value can be HIGH (1) or LOW (0)
- digitalRead(pin); // read value from pin
  - Returns an int – value either HIGH or LOW
- delay(val);   // pause the program for a bit
  - Pauses for val milliseconds (1/1000's of a sec)
  - 1000 msec = 1sec
  - val can be up to "unsigned long max" (i.e. huge)

Data Types on Arduino

- By default, types are signed unless you say "unsigned"…

Type	Size (bits)	Size (bytes)	Minimum	Maximum
boolean	1	1	0 (false)	1 (true)
unsigned byte	8	1	0	255
byte	8	1	-128	127
unsigned int	16	2	0	65,535
int	16	2	-32,768	32,767
unsigned long	32	4	0	4,294,967,295
long	32	4	-2,147,483,648	-2,147,483,647
float (double)	32	4	-3.4028235E+38	3.4028235E+38

Load "Blink" example



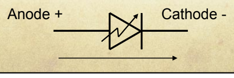
```
The circuit
* LED conne
* Note: On                 lready an LED on the board
connected t                extra components for this example.

Created 1 J
By David Cu

http://ardu

based on an                Wiring i/o board

*/

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:
```

Blink Modifications

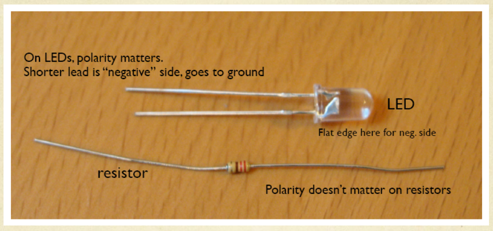
- Change so that blink is on for 500msec and off for 100msec
  - What happens?
- Change so that blink is on for 50msec and off for 50msec
  - What happens?
- Change so that blink is on for 10ms and off for 10ms
  - What happens?

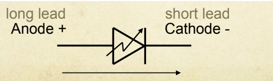
Blink Modifications

- Change to use an external LED rather than the one on the board
  - Connect to pin 13
  - 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!



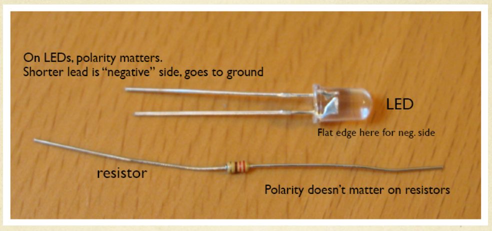
LEDs and Resistors

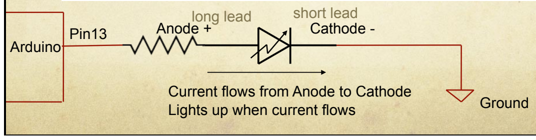

On LEDs, polarity matters.
Shorter lead is "negative" side, goes to ground
LED
Flat edge here for neg. side
resistor
Polarity doesn't matter on resistors



Current flows from Anode to Cathode
Lights up when current flows

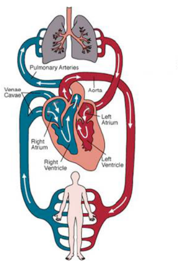
LEDs and Resistors

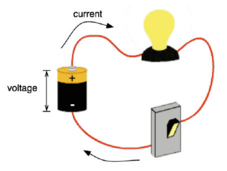

On LEDs, polarity matters.
Shorter lead is "negative" side, goes to ground
LED
Flat edge here for neg. side
resistor
Polarity doesn't matter on resistors



Current flows from Anode to Cathode
Lights up when current flows

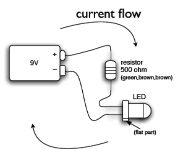
Making Circuits

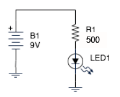


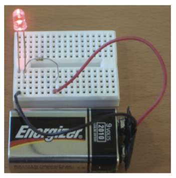


heart pumps, blood flows   voltage pushes, current flows

Wiring it Up



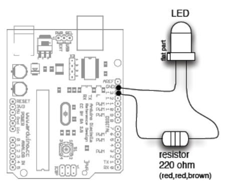


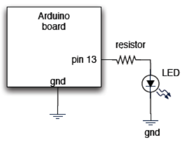


wiring diagram   schematic   wiring it up

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

Wiring it Up

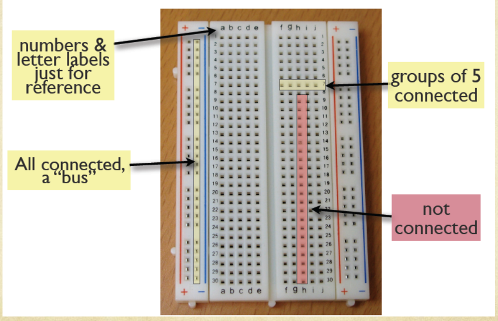




wiring diagram   schematic

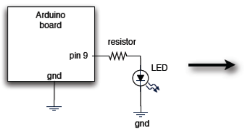
Arduino Duemilanove board has this circuit built-in
To turn on LED use `digitalWrite(13,HIGH)`

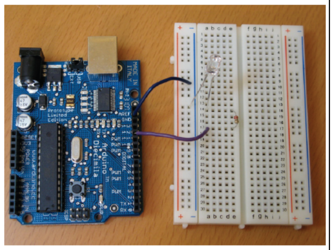
Proto Boards



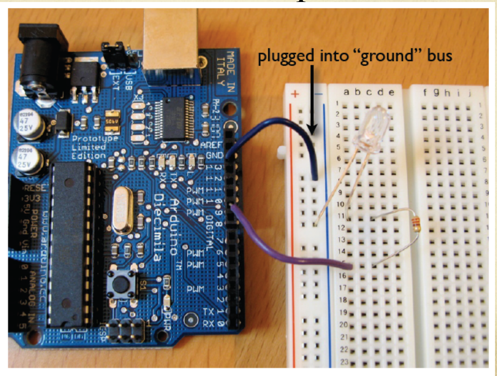
AKA Solderless Breadboards

Wire it Up





Wire it Up

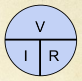


We just made an LED blink
Big Deal?

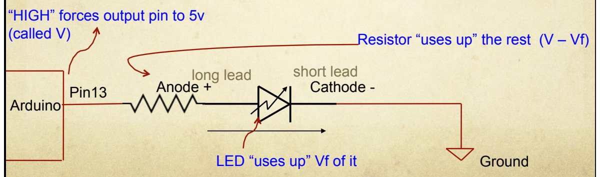
- Most actuators are switched on and off with a digital output
  - The digitalWrite(pin,value); function is the software command that lets you control almost anything
- LEDs are easy!
  - Motors, servos, etc. are a little trickier, but not much
  - More on that later…
- Arduino has 14 digital pins (inpts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

Current Limiting Resistor

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

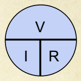


- Every LED has a Vf "Forward Voltage"
  - How much voltage is dropped (used up) passing through the LED


"HIGH" forces output pin to 5v (called V)
Resistor "uses up" the rest (V – Vf)
long lead
short lead
Anode +
Cathode -
Pin13
Arduino
LED "uses up" Vf of it
Ground

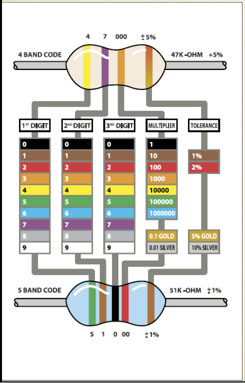
Current Limiting Resistor

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



- Every LED has a Vf "Forward Voltage"
  - How much voltage is dropped (used up) passing through the LED
- R = (V – Vf) / I
  - Example – If Vf 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

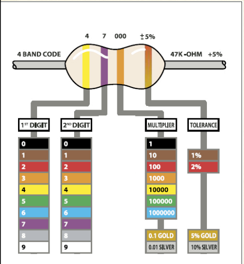


What's the color code for a 220Ω resistor?

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

What's the color code for a 470Ω resistor

Resistor Color Codes



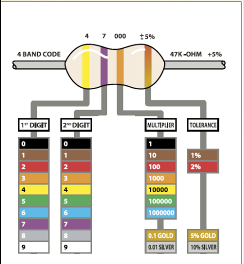
We're using 4-band 5% resistors with a ¼ watt rating

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

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

What's the color code for a 470Ω 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?

red red brown gold

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

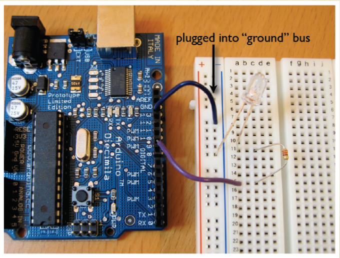
brown black red gold

What's the color code for a 470Ω resistor

yellow violet brown gold

Wire it Up

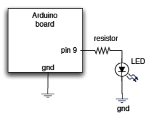
- Wire up an external LED of your choice, and change the Blink program to use that external LED
  - Choose your resistor based on the Vf of the LED you're using



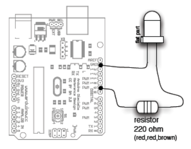
Moving on…

Varying LED Brightness

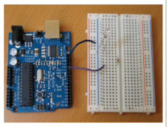
Same circuit as Blink circuit but pin 9 instead of pin 13



schematic



wiring diagram



wired up

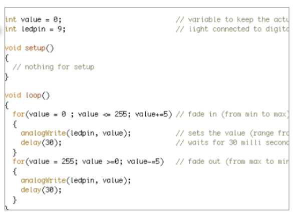
The PWM pins work with the "`analogWrite(value)`" command where "value" ranges between 0 and 255.
To turn LED to half-bright, use `analogWrite(9,128)`

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 mir
  {
    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)
```

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
  - Assume that "foo" is a variable that you can set in your program
  - 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…

Candle Program

- random(min,max); will return a random number between min and max.
  - randomSeed(int); will initialize the random function
  - Not really needed…
  - foo = random(10, 200); // assign foo to a random number between 10-200
- Remember delay(val); // waits for "val" milliseconds

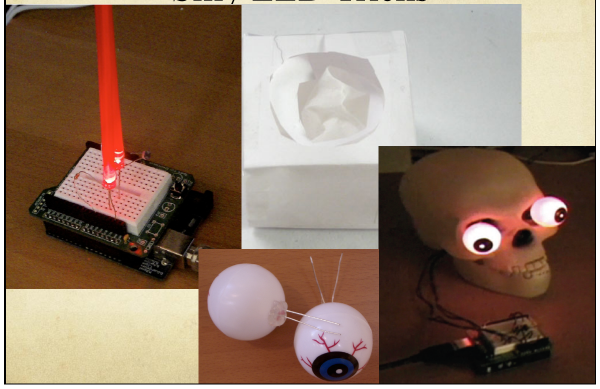
hints…

```
int bright; // make a new variable called bright
bright = random(100, 255); // set "bright" to a random value
                                           // between 100 and 255
Remember: analogWrite(pin,value); // sets a brightness on a pin
       // "pin" is the pin number, "value" is between 0 – 255
```

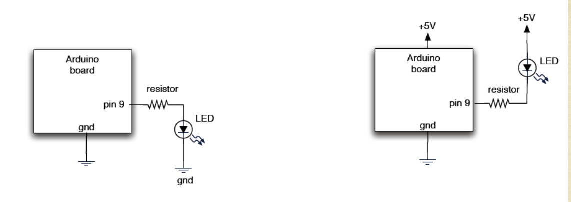
Candle Program

Blocked out for now…

Silly LED Tricks



LED Wiring – 2 ways



To turn ON: `digitalWrite(9,HIGH)`
To turn OFF: `digitalWrite(9,LOW)`

To set brightness: `analogWrite(9,val)`

To turn ON: `digitalWrite(9,LOW)`
To turn OFF: `digitalWrite(9,HIGH)`

To set brightness: `analogWrite(9,255-val)`

Next Task: 8 LEDs

- connect LEDs (through resistors!) to 8 Arduino pins
  - use pins 0, 1, 2, 3, 4, 5, 6, 7
    - Remember, pwm on pins 3, 5, 6, 9, 10, 11 only…
  - Now you can turn the LEDs on and off with

    ```
    digitalWrite(0, HIGH);  // turn LED 0 on
    digitalWrite(1, LOW);   // turn LED 1 off
    analogWrite(3, 180);    // turn LED 3 partly on
    ```
  - Use those commands, also delay(), also perhaps loops, and random(min,max) to make the 8 LEDs do something!

Hints… Overall Algorithm

```
void setup() {
    … set pin directions…
    … set global values if needed…
    }

void loop() {
    … set LED on/off values…
    … delay for some amount of time …

    … set LED on/off values…
    … delay for some amount of time…

    … more LED values followed by more delays…
    … etc. …

} // this code repeats when you get to the end…
```

Hints…setup()

```
void setup() {
    pinMode(0,OUTPUT);
    pinMode(1,OUTPUT);
    pinMode(2,OUTPUT);
    pinMode(3,OUTPUT);
    pinMode(4,OUTPUT);
    pinMode(5,OUTPUT);
    pinMode(6,OUTPUT);
    pinMode(7,OUTPUT);
    }
```

OR…

```
void setup(){
    // do nothing (why?)
}
```

OR…

```
void setup() {
    for (int i=0; i<8; i++) {              // this loop will repeat 8 times
       pinMode(i, OUTPUT);  // set each pin to OUTPUT
       }  // i will be 0, 1, 2, 3, 4, 5, 6, 7 on each iteration of the loop
    }
```

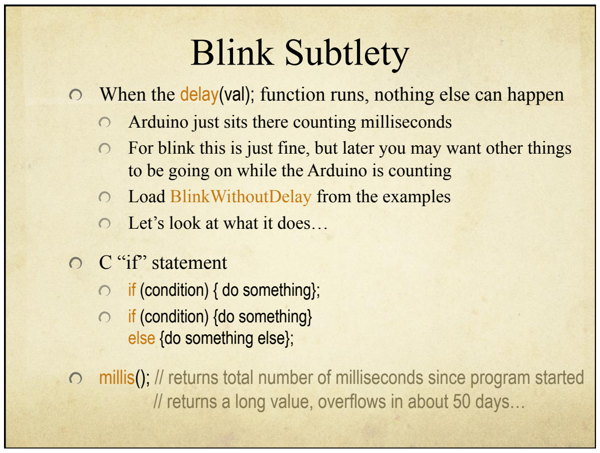
Hints…loop()

```
// loop is the function that repeats forever

void loop() {
    int delayTime = 100;       // a basic unit of delay (in msec)

    digitalWrite(0, HIGH);     // set LED 0 on
    delay(delayTime);          // wait delayTime milliseconds

    digitalWrite(0, LOW);      // set LED 0 off
    digitalWrite(1, HIGH);     // set LED 1 on
    delay(delayTime);          // wait delayTime milliseconds
    …// more setting and delaying…
    }
```

Or use for (int i=0; i<foo; i++), or random(min,max), etc…

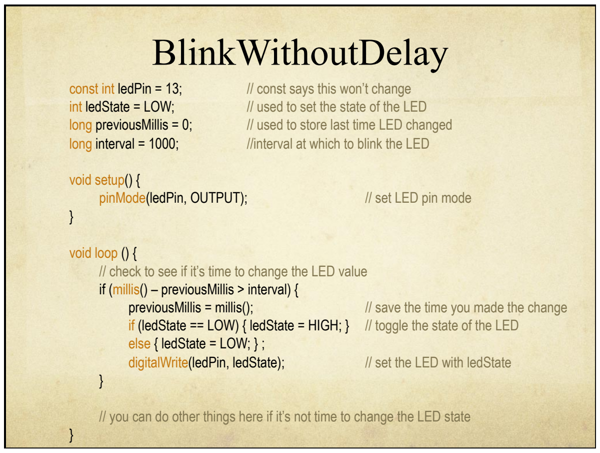
Everybody start coding!

- We'll have demos in a few minutes…



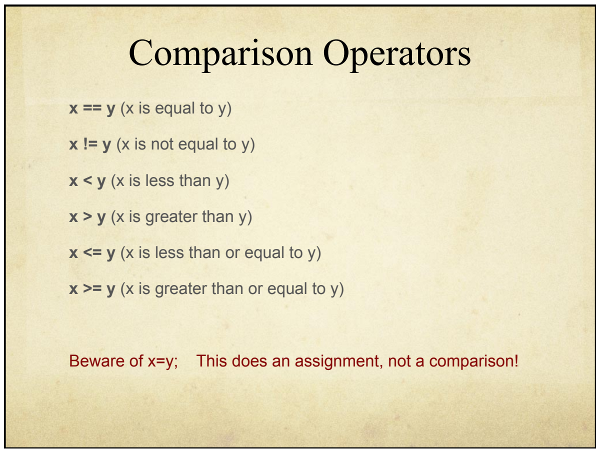
Blink Subtlety

- When the delay(val); function runs, nothing else can happen
  - Arduino just sits there counting milliseconds
  - For blink this is just fine, but later you may want other things to be going on while the Arduino is counting
  - Load BlinkWithoutDelay from the examples
  - Let's look at what it does…
- C "if" statement
  - if (condition) { do something};
  - if (condition) {do something}
    else {do something else};
- millis(); // returns total number of milliseconds since program started
  // returns a long value, overflows in about 50 days…



BlinkWithoutDelay

```
const int ledPin = 13;          // const says this won't change
int ledState = LOW;             // used to set the state of the LED
long previousMillis = 0;        // used to store last time LED changed
long interval = 1000;           //interval at which to blink the LED

void setup() {
     pinMode(ledPin, OUTPUT);                       // set LED pin mode
}

void loop () {
     // check to see if it's time to change the LED value
     if (millis() – previousMillis > interval) {
          previousMillis = millis();                // save the time you made the change
          if (ledState == LOW) { ledState = HIGH; } // toggle the state of the LED
          else { ledState = LOW; } ;
          digitalWrite(ledPin, ledState);           // set the LED with ledState
     }

     // you can do other things here if it's not time to change the LED state
}
```



Comparison Operators

x == y (x is equal to y)

x != y (x is not equal to y)

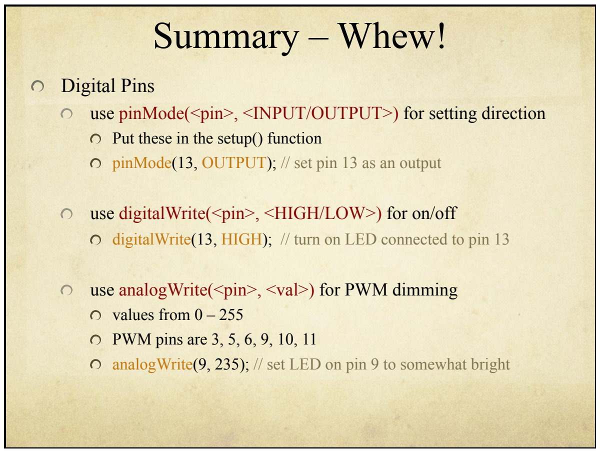
x < y (x is less than y)

x > y (x is greater than y)

x <= y (x is less than or equal to y)

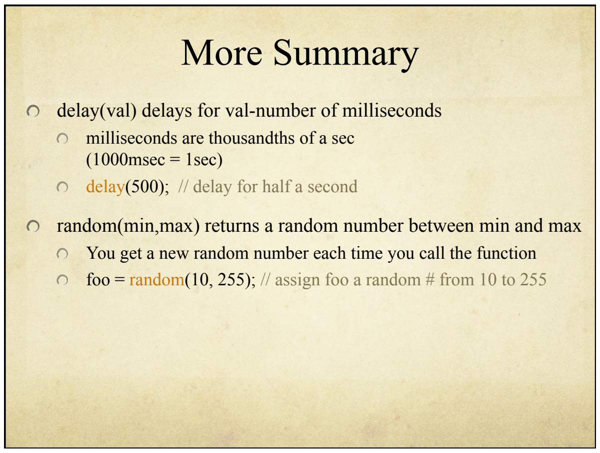
x >= y (x is greater than or equal to y)

Beware of x=y; This does an assignment, not a comparison!



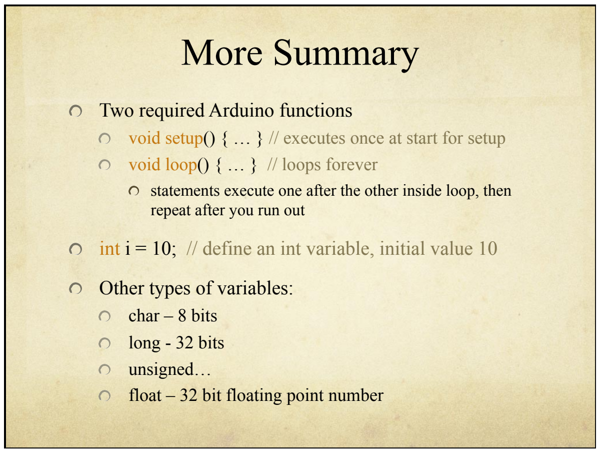
Summary – Whew!

- 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(13, HIGH); // turn on LED connected to pin 13
  - 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



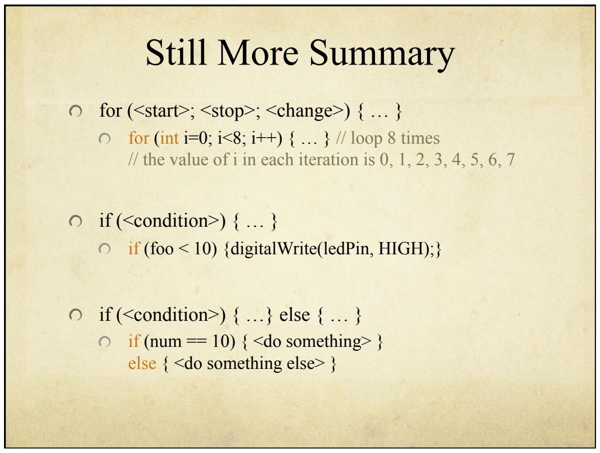
More Summary

- 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
  - You get a new random number each time you call the function
  - foo = random(10, 255); // assign foo a random # from 10 to 255



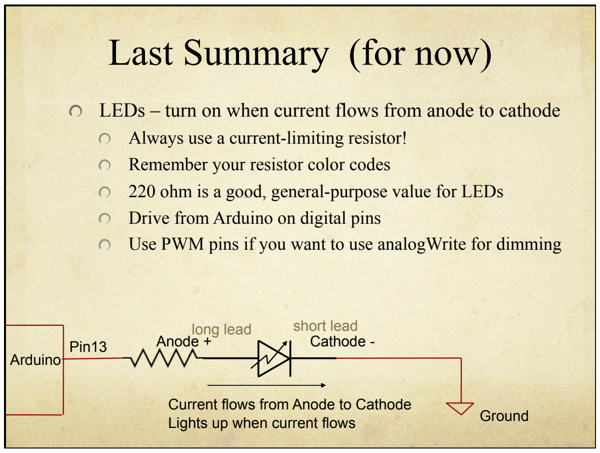
More Summary

- 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



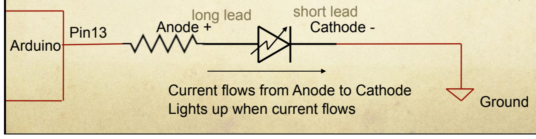
Still More Summary

- 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 ohm is a good, general-purpose value for LEDs
  - Drive from Arduino on digital pins
  - Use PWM pins if you want to use analogWrite for dimming


Arduino
Pin13
long lead
Anode +
short lead
Cathode -
Ground
Current flows from Anode to Cathode
Lights up when current flows