FOR INSTRUCTIONS ON HOW TO INSTALL ARDUINO SOFTWARE ON A MAC:

HTTP://WWW.ARDUINO.CC/EN/GUIDE/MACOSX

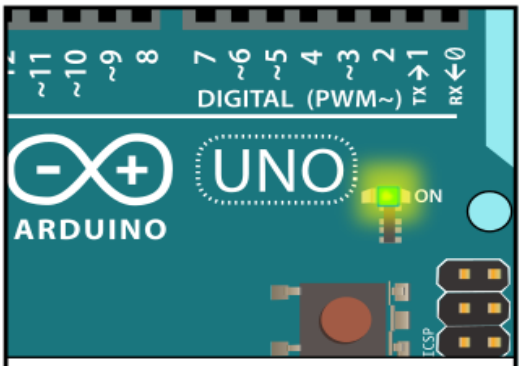
FOR INSTRUCTIONS ON HOW TO INSTALL ON WINDOWS:

HTTP://WWW.ARDUINO.CC/EN/GUIDE/WINDOWS

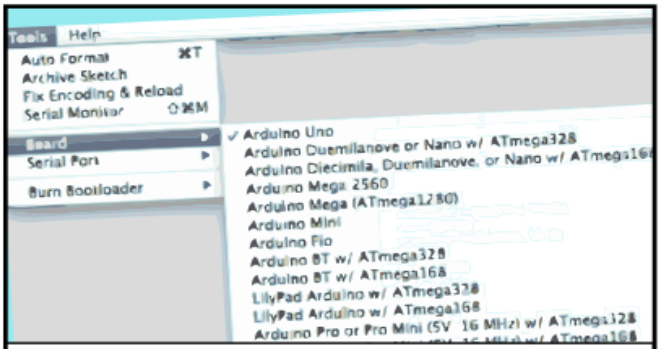
FOR INSTRUCTIONS ON HOW TO INSTALL ON LINUX:

HTTP://WWW.ARDUINO.CC/PLAYGROUND/LEARNING/LINUX

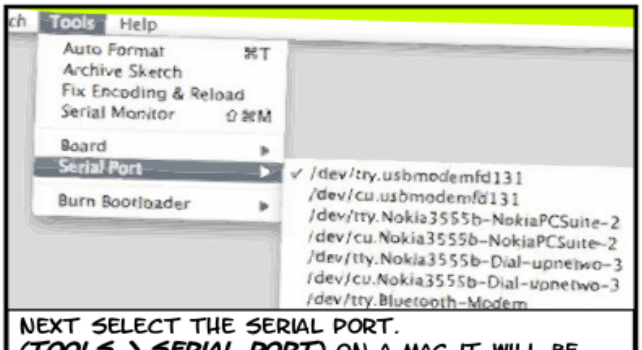
GO TO THE URLS ABOVE FOR DETAILED INSTRUCTIONS ON INSTALLING THE SOFTWARE ON THESE PLATFORMS.



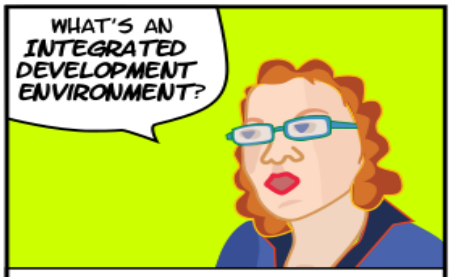
WHEN YOU HAVE INSTALLED THE SOFTWARE, CONNECT THE ARDUINO. AN LED MARKED ***ON*** SHOULD LIGHT UP ON THE BOARD.



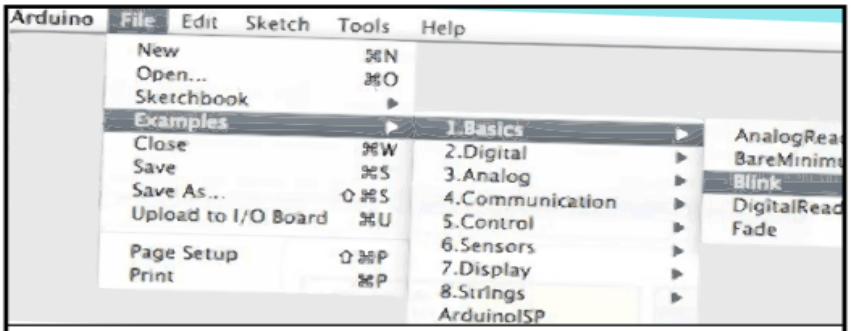
LAUNCH THE ARDUINO SOFTWARE. IN THE TOOLS MENU, SELECT THE BOARD YOU ARE USING ***(TOOLS > BOARD)***. FOR EXAMPLE, ***ARDUINO UNO***.



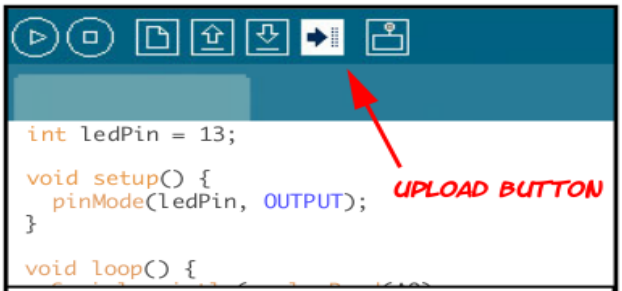
NEXT SELECT THE SERIAL PORT. ***(TOOLS > SERIAL PORT)*** ON A MAC IT WILL BE SOMETHING LIKE ***/DEV/TTY.USBMODEM***. ON A WINDOWS MACHINE, IT WILL BE ***COM3*** OR SOMETHING LIKE THAT.



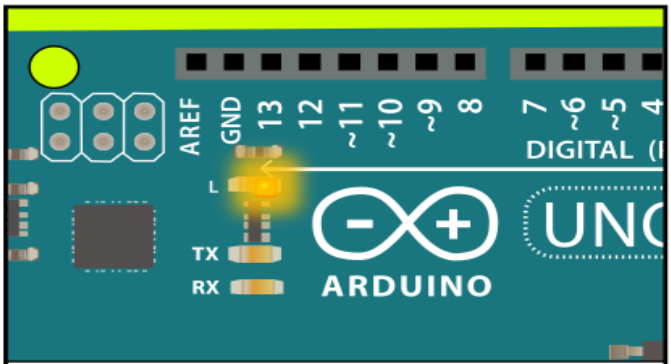
WHEN YOU DOWNLOADED THE ARDUINO SOFTWARE, YOU DOWNLOADED AN ***IDE***. IT COMBINES A TEXT EDITOR WITH A COMPILER AND OTHER FEATURES TO HELP PROGRAMMERS DEVELOP SOFTWARE.



THE ARDUINO IDE ALLOWS YOU TO WRITE ***SKETCHES***, OR ***PROGRAMS*** AND UPLOAD THEM TO THE ARDUINO BOARD. OPEN THE ***BLINK*** EXAMPLE IN THE FILE MENU. ***FILE > EXAMPLES > 1.BASICS > BLINK***.



TO UPLOAD THE SKETCH TO THE ARDUINO BOARD, CLICK THE ***UPLOAD BUTTON*** ON THE STRIP OF BUTTONS AT THE TOP OF THE WINDOW. SOME MESSAGES WILL APPEAR IN THE BOTTOM OF THE WINDOW, FINALLY ***DONE UPLOADING***.

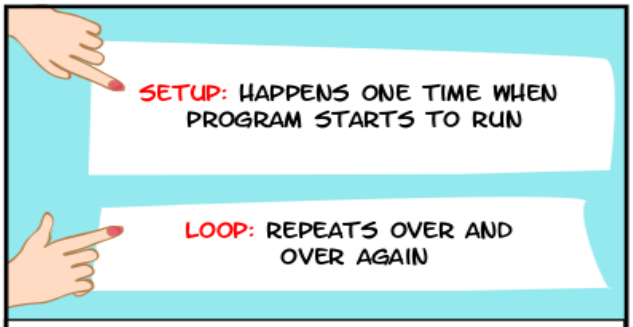


THE LED AT PIN 13 ON THE ARDUINO STARTS BLINKING.

```
void setup() {
  // initialize the digital pin as an output.
  // Pin 13 has LED connected on most Arduino boards
  pinMode(13, OUTPUT);
}

void loop() {
  digitalWrite(13, HIGH);   // set the LED on
  delay(1000);              // wait for a second
  digitalWrite(13, LOW);    // set the LED off
  delay(1000);              // wait for a second
}
```

A SKETCH, LIKE A PROGRAM WRITTEN IN ANY LANGUAGE, IS A SET OF INSTRUCTIONS FOR THE COMPUTER. IF WE LOOK CLOSELY AT THE BLINK SKETCH, WE SEE THERE ARE 2 MAJOR PARTS, ***SETUP*** AND ***LOOP***.



THESE ARE BOTH BLOCKS OF CODE CALLED ***FUNCTIONS*** THAT EVERY SKETCH WILL HAVE. THEY ARE BLOCKED OUT BY CURLY BRACES { }.

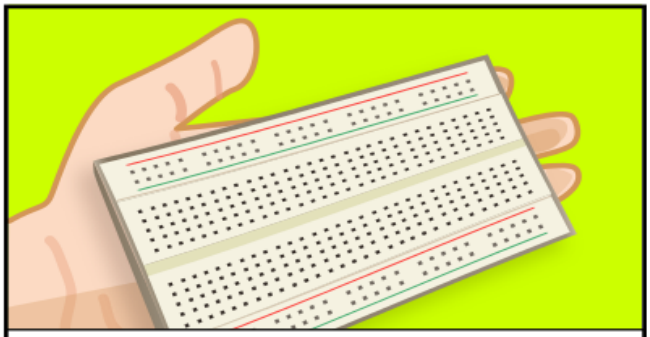
HTTP://ARDUINO.CC/EN/REFERENCE/HOMEPAGE

CHECK OUT THE ARDUINO WEBSITE FOR THE ARDUINO REFERENCE GUIDE AND MANY OTHER RESOURCES TO LEARN THE LANGUAGE.

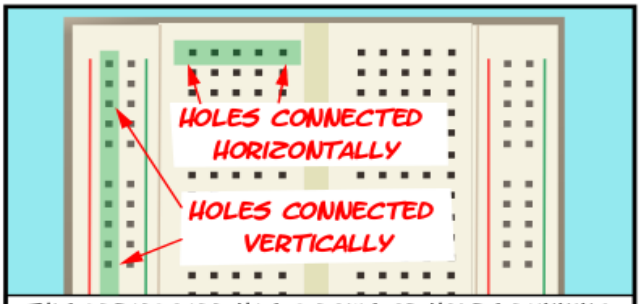
```
void setup() {       //DECLARES BLOCK OF CODE
  pinMode(13, OUTPUT); //SETS PIN 13 TO OUTPUT
}                      //END BLOCK OF CODE


void loop() {       //DECLARES BLOCK OF CODE
  digitalWrite(13, HIGH);  //SETS PIN 13 HIGH
  delay(1000);              //PAUSE 1 SECOND
  digitalWrite(13, LOW);    //SETS PIN 13 LOW
  delay(1000);              //PAUSE 1 SECOND
}                         //END BLOCK OF CODE
```

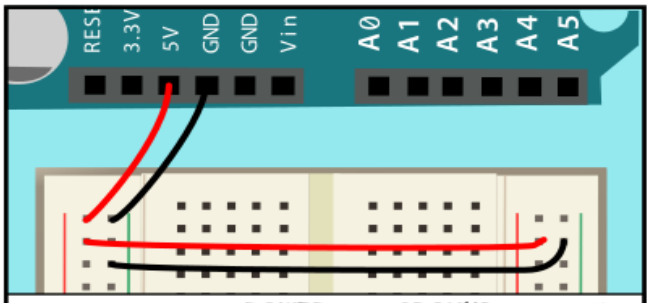
FOR NOW, LET'S LOOK AT THIS SIMPLE SCRIPT LINE BY LINE & SEE WHAT EACH LINE DOES.



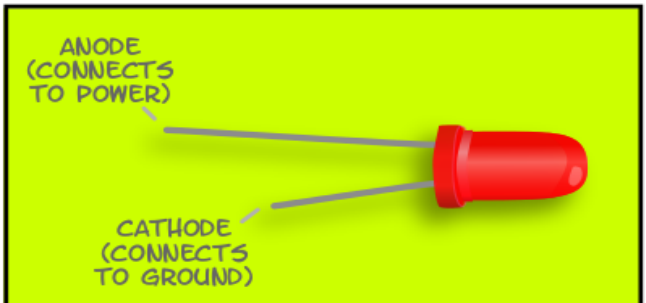
HOW DO WE CONTROL OBJECTS THAT ARE NOT ON THE ARDUINO BOARD? WE WILL CONNECT THE ARDUINO TO A ***SOLDERLESS BREADBOARD***. THIS WILL ALLOW US TO QUICKLY SET UP AND TEST CIRCUITS.



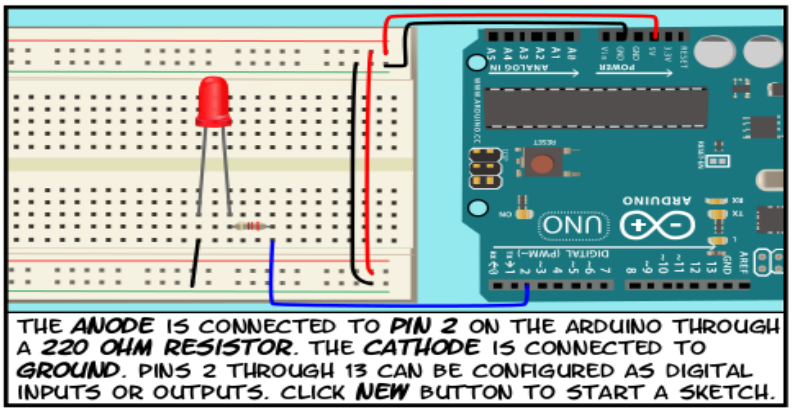
THIS BREADBOARD HAS 2 ROWS OF HOLES RUNNING DOWN THE LEFT AND RIGHT SIDE, AND 5 ROWS OF HOLES ON EITHER SIDE OF A MIDDLE INDENTATION. THE SIDE ROWS ARE CONNECTED ***VERTICALLY***, EACH ROW OF 5 HOLES IN THE MIDDLE ARE CONNECTED ***HORIZONTALLY***.

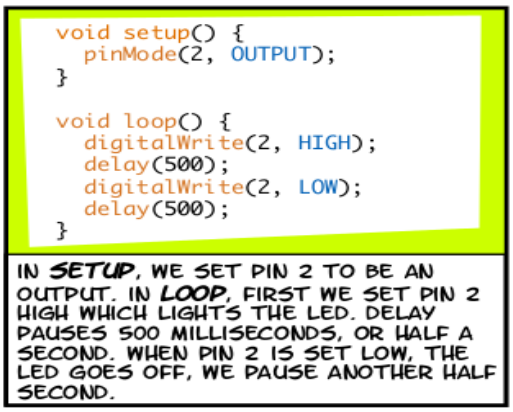


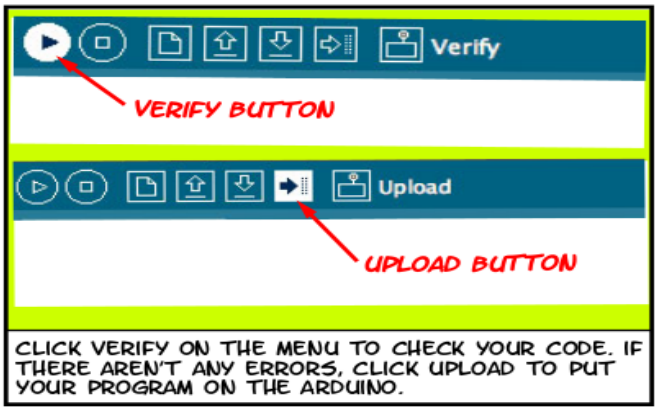
WE WILL CONNECT ***POWER*** AND ***GROUND*** FROM THE ARDUINO BOARD TO THE VERTICALLY CONNECTED STRIPS ON THE LEFT AND RIGHT WITH 22 GAUGE WIRE. OTHER COMPONENTS CAN BE ATTACHED TO THE HOLES IN THE MIDDLE AND TO POWER AND GROUND AS NEEDED.

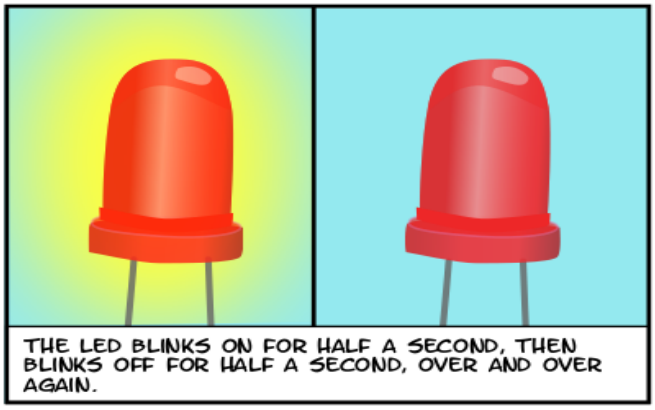


WHEN CURRENT FLOWS THROUGH A ***LED (LIGHT EMITTING DIODE)*** IN THE RIGHT DIRECTION, IT LIGHTS UP. WE'LL ATTACH AN LED TO THE BREADBOARD, THEN TO THE ARDUINO SO WE CAN CONTROL IT WITH CODE.





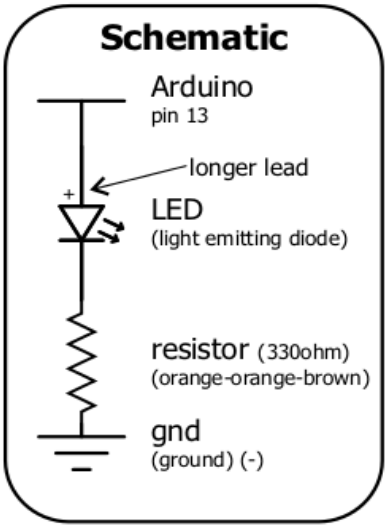




Projects:

1. S.O.S. with a L.E.D.

Your task here is to use an LED to signal the letters S.O.S., which is the international distress signal in Morse Code. Morse Code is an old coding system where you can signal letters and numbers using only two symbols: OFF and ON, or a dot and a dash. So it's nicely suited to Arduino, which is a digital system. (In the old days, Morse Code was used to signal between ships using blinking lights, between continents using telegram, and in war correspondence using radio).



S = dot dot dot (dot means a short flash)
O = dash dash dash (dash means a long flash)

Give it a go, using the digital output of the Arduino. You can use the code from the comic above as a reference. **Remember to put the LED in series with a resistor, to limit the current.**

Once you're done, here's a tip to improve your code. If you find yourself repeating a particular line of code over and over, such as

line of code;
line of code;
line of code;
line of code;

You can replace this with a **for loop**:

```
for(int i = 0; i < 4; i ++){
  line of code;
}
```

This tells the Arduino to create a counter called i and set it equal to zero. It will then keep repeating the line of code, incrementing i by 1 every time it does so. When i reaches the upper limit (in this case 4) the loop will end.

2. LED Fader

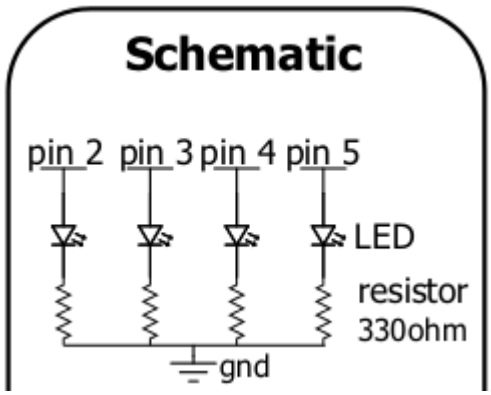
Now let's try and get the LED light to fade in and out, like we did in class. In the very top of your code, before the setup function, create two variables:

int brightness = 0
int fadeamount = 5

Try and work out how to proceed from here.

Tips:
1. You now want to use **analogWrite** instead of digitalWrite.
2. analogWrite must send out a value between 0 and 255. Be careful not to overshoot this range.
3. you can use **if statements**, like we discussed in class, to help you out here.

3. LED pulse pattern



Set up 4 LEDs, each connected its own 330 ohm resistor and an output of the Arduino. Once this circuit is set up, your goal is to have the LEDs blink in a chasing pattern, similar to the lights on the car in Knight Rider (80s reference.. never mind).

It should go through these sequences (O = LED on, X = LED off):
OXXX -> XOXX -> XXOX -> XXXO -> XXOX -> XOXX and then loop back to the start.

To make things a bit easier to a handle, we're going to learn how to work with **arrays**.

An array is a list of variables. For example, you can store the pin numbers of the LEDs in an integer array:

int ledPins[] = {2,3,4,5}; //an array to store the pin numbers each LED is connected to

For example, if you want to refer to pin 2, you can now just type in ledPins[0] and it will give you the first element. Similarly, ledPins[1] is 3, and so on..

You can now use a for loop to initialize all the pins in your setup function:

```
void setup(){
  for(int i = 0; i <4; i++){
    pinMode(ledPins[i], OUTPUT);
  }
}
```

This will do the same thing as:

```
void setup(){
 pinMode(ledPins[0],output);   // initializes pin 2 for output
 pinMode(ledPins[1],output);   // initializes pin 3 for output
 pinMode(ledPins[2],output);   // initializes pin 4 for output
 pinMode(ledPins[3],output);   // initializes pin 5 for output
}
```

Suppose you want to switch on all the LEDs one by one. You can do this with a for loop inside your loop part of the code.

```
void loop(){
for(int i=0;i<4;i++)
        {
          digitalWrite(ledPins[i], HIGH);
        }
}
```

Make sure you understand what the for loop is doing in these examples. It'll come in VERY handy.

Now, modify this code to make the pulse pattern described above. You'll want to use the delay function to slow things down so our eyes can see the pattern.

Also try out a few new patterns of your own.

Take on the next section only if you have time to spare:

4. Analog Input

The task is to wire up a sensor to the analog input, and use it to control the blinking rate of the pulsed LEDs from the previous activity.

To do this, wire up a sensor according to its wiring diagram, and plug it in to an analog port. Here is some example code that will return the value measured by an analog port. It uses 'serial communication', which just means the Arduino is sending data to your computer, 1 bit at a time.

```
void setup(){
  Serial.begin(9600); // Tells Arduino to send back 9600 bits of data per second.
                      // 9600 is an old standard from the modem days
}

void loop(){
  int brightness = analogRead(0); //Read in the value from Analog Port 0
  Serial.println(brightness);     //And print it to the serial port
  delay(500);                     // Wait half a second.

}
```

Run the code, and open the Serial Monitor to spy on these readings. Figure out the minimum and maximum readings from your sensor. Next, you'll want to use the map function to scale this number to something appropriate for a LED blinking rate.

The command for scaling a number is **map**:

brightness = map(brightness, lowvalue, highvalue, newlowvalue, newhighvalue);

This will scale the reading that goes from lowvalue to highvalue, so that it now goes from newlowvalue to newhighvalue.

Below are the circuit diagram for some different types on sensors: