Fun with the Arduino and Electronics – Session A

Tech Resort Innovators Sessions #2103A

In these sessions you are going to experiment with building electronic circuits. You will use an 'Arduino' microcontroller board to interface to your circuits in order to build computer controlled systems.

Skills involved:	**Coding**	**√**
	Electronics	**√**
Suitable for ages:	**10-14**	
You will need:	**An Arduino Microcontroller**	
	A PC with the Arduino programming language loaded	
	Some electronics components as set out below	
	A 'Solderless Breadboard'	

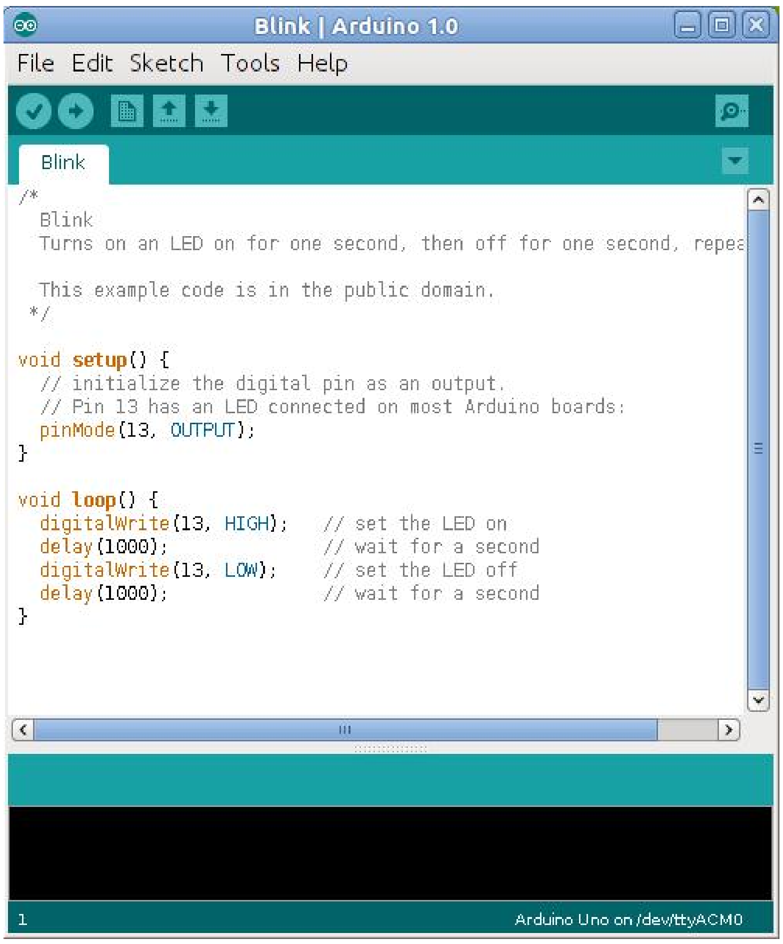
Introduction

- Here, we're going to explore how to program an Arduino microcontroller using the Arduino programming language. As this is text based it's a good introduction to using text to create a program rather than graphics as you may have been doing so far with languages like Scratch.
- We're also going to see how basic electronics circuits work and examine the principles of current and voltage.
- Try to read through the text of the instructions rather than just following the pictures. This will help you get a much better understanding so you can complete the challenges.

Programming an Arduino

- Let's start by plugging our Arduino in and learning how to program it.
- The Arduino has a small LED on it that we can program to flash on and off. An example of how to do this is built into the Arduino IDE which is what we'll use to program it.
- Plug the Arduino Uno board into your PC using a USB cable and start the Arduino IDE (Integrated Development Environment) which you'll find on your desktop.
- Check in Tools → Board that the board is set to Arduino/Genuino Uno. If it isn't, go and change it.
- Now check Port setting – you need to select the port that has your Arduino on it.
- Now, on the top menu bar, click **File → Examples → 1.Basics → Blink**
- This code does the following things:
    - Has a piece of code in **setup()** which runs only once. This sets up Pin 13 of the Arduino (to which the on board LED is connected) to be an Output.
    - Then repeatedly runs a **loop()** which turns Pin 13 ON (or HIGH) for 1 second or (1000 milliseconds) to turn the LED on, and then sets it to OFF (or LOW) for 1 second to turn the LED off.

- The code should open in a new window and is shown below:



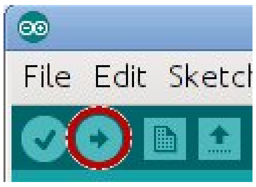
```
/*
  Blink
  Turns on an LED on for one second, then off for one second, repea
 
  This example code is in the public domain.
 */

void setup() {
  // initialize the digital pin as an output.
  // Pin 13 has an 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
}
```

```
1                                   Arduino Uno on /dev/ttyACM0
```

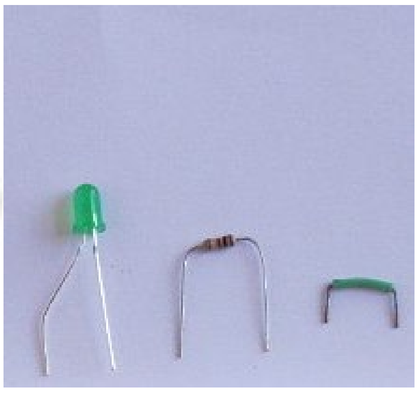
- This program can now be loaded into the Arduino: click the Upload button (shown in red below) on the top toolbar to load the program to the Arduino.



- The program should load and then start running - you will see the LED marked 'L' on the board start to flash on and off.
- If the program doesn't load correctly check the following:
  - In the **Tools** menu item, check that we have set **Board: "Arduino/Genuino Uno"**
  - Also in the **Tools** menu item, select **Port** and check that we have set the COM port into which our Arduino is plugged.
- Try changing this program to alter the time the LED is on and off for: edit the values in the brackets after the **delay()** command.

Building a Circuit

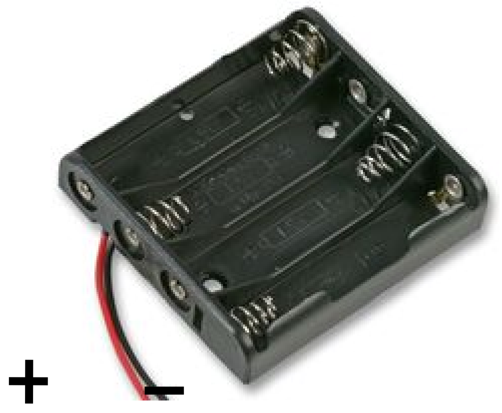
- We’re now going to put our Arduino to one side and do some very basic experiments with electronic components so we understand how they work.
- To start we’re going to build a very simple circuit which lights up a single Light Emitting Diode (LED). ‘Component’ is a word we use to refer to any single part of an electronic circuit. Here are the components we will use:

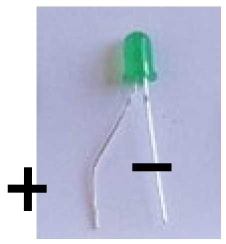


These are, from left to right:

A battery holder, an LED, a 390 Ohm resistor and a wire link.

- You’ll notice that all of these have two leads, but in two of the components the two leads are different: the battery holder has a red lead and a black lead and the LED has a long leg and a short leg. This is because it’s important which way round we connect them: these components have a ‘positive’ side and a ‘negative’ side. The red lead is the positive side of the battery terminal and the long leg is the positive side of the LED. We can denote this using a ‘+’ and a ‘-‘ sign.
  BEWARE! It’s important that the positive and negative wires of the battery holder don’t touch when there are batteries in the holder. This would cause a short circuit and damage the batteries and the box.

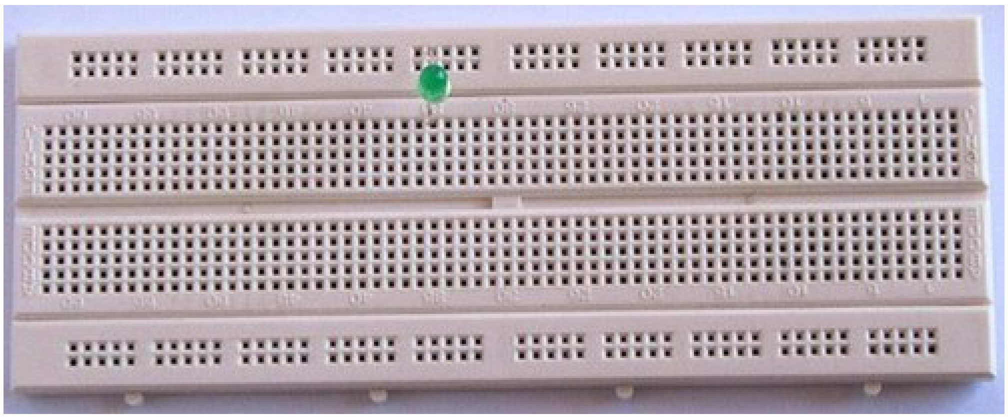




- Now let’s have a go at building our first circuit using our four components and our solderless breadboard

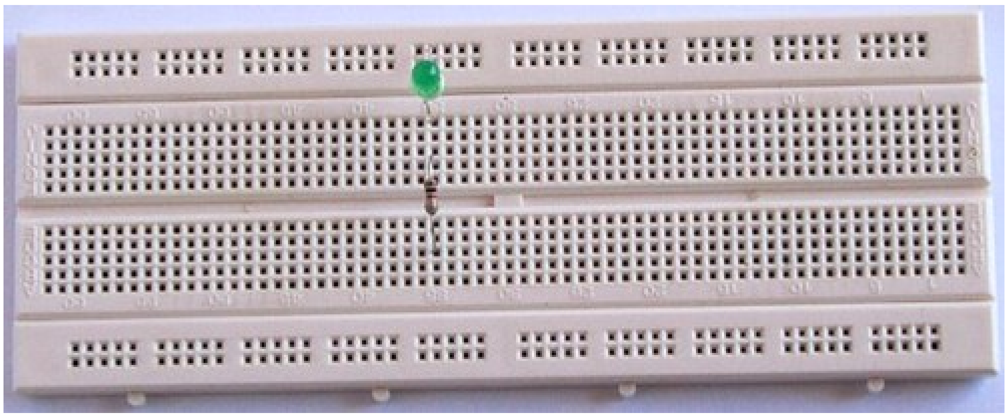
Step 1: Insert the LED into the Breadboard

- Start by bending the longer lead of the LED as shown in the previous photo. Plug the longer 'positive' lead of the LED into the top rail of the breadboard and the other lead into a hole in the main part of the breadboard as shown:



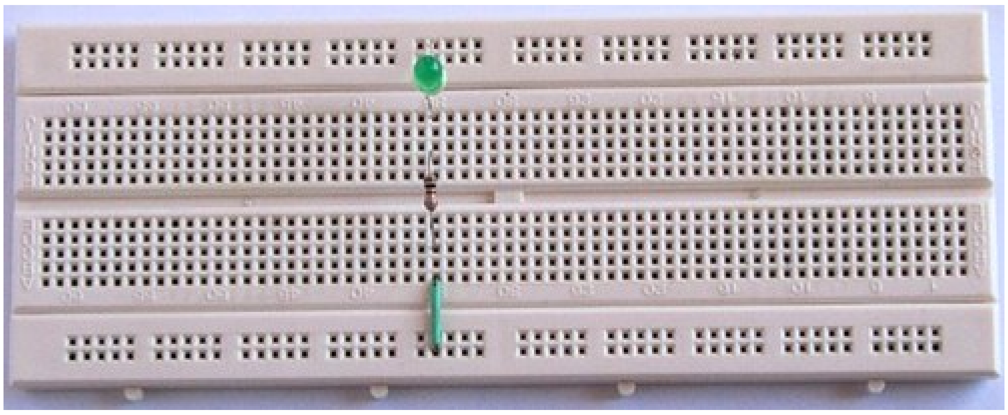
Step 2: Insert the Resistor into the Breadboard

- Use the side cutters to remove a 1k resistor from the string of resistors if they are taped together. Cut the resistor lead as near to the tape as possible. Don't try to remove the tape as this will leave a sticky mess on the end of the resistor lead which will then end up in your breadboard.
- Bend the leads of the resistor as shown below. Plug one of the resistor leads into a hole directly below the 'negative' lead of the LED and the other lead into a hole below the middle channel of the breadboard. This connects the negative side of the LED to one of the resistor leads. It does not matter which way around the resistor is plugged into the breadboard.



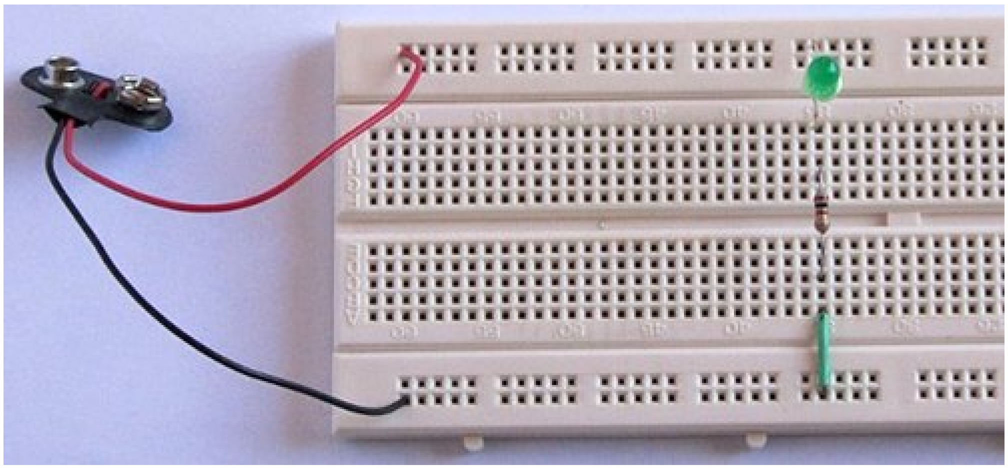
Step 3: Insert the Wire Link into the Breadboard

- Insert a wire connector into a hole directly below the resistor lead and into the bottom rail of the breadboard. If you don't have one exactly like the one below, don't worry: any piece of wire that fits into the holes in the breadboard will do!



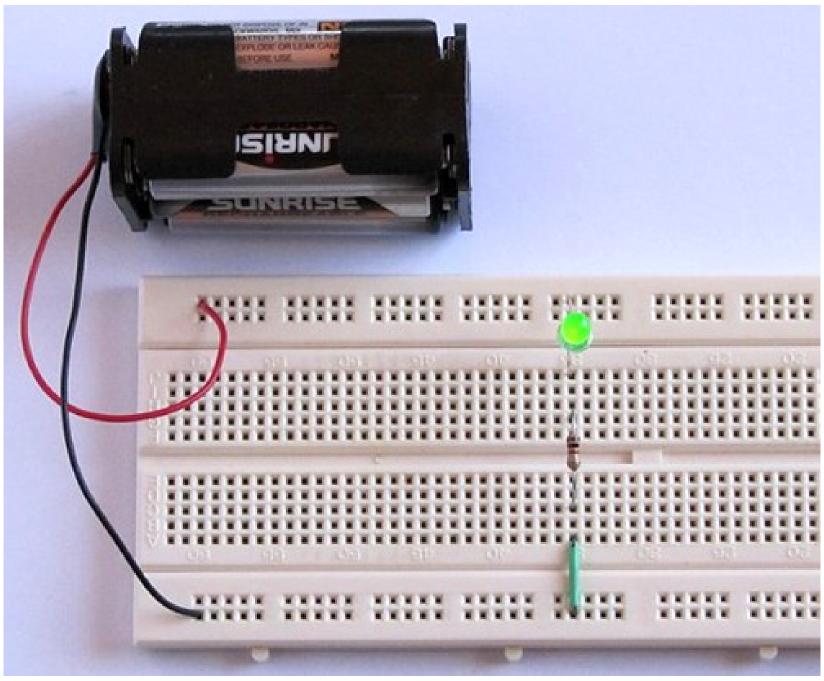
Step 4: Insert the Battery Holder into the Breadboard

- Plug the red (positive) wire of the battery holder into the top rail of the breadboard. Plug the black (negative) wire of the battery holder into the bottom rail of the breadboard. Note that we’ve shown a clip rather than a holder here.



Step 5: Put four Batteries into the Battery Holder

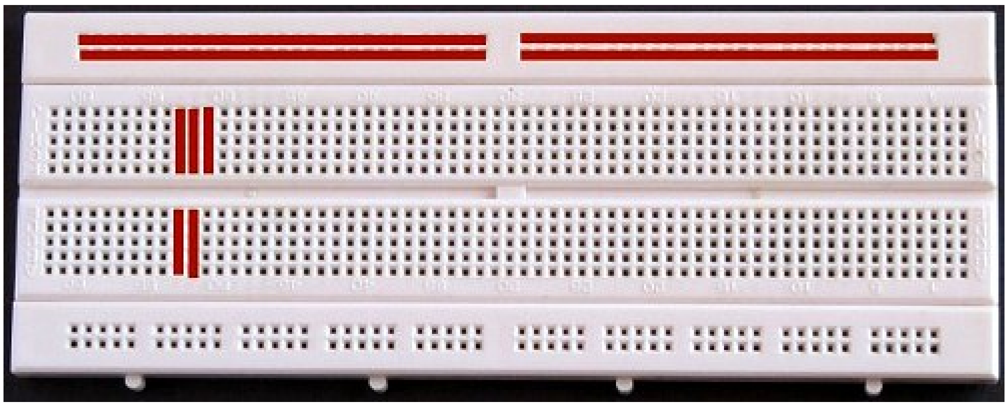
- Finally put your batteries into theholder to power up the circuit and switch the LED on.



- Does the LED light up: Yes? Well done!
- No? Check everything is connected as shown and is the right way round. The try wiggling each of the components to see if it's made a proper connection in the breadboard.

How the Breadboard and Circuit Works

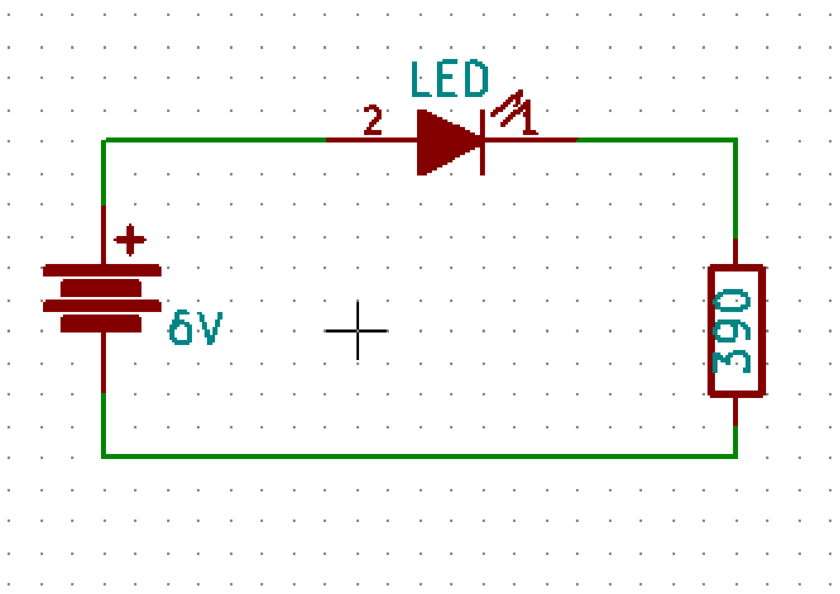
- The red lines in photo below show how the breadboard is connected internally.



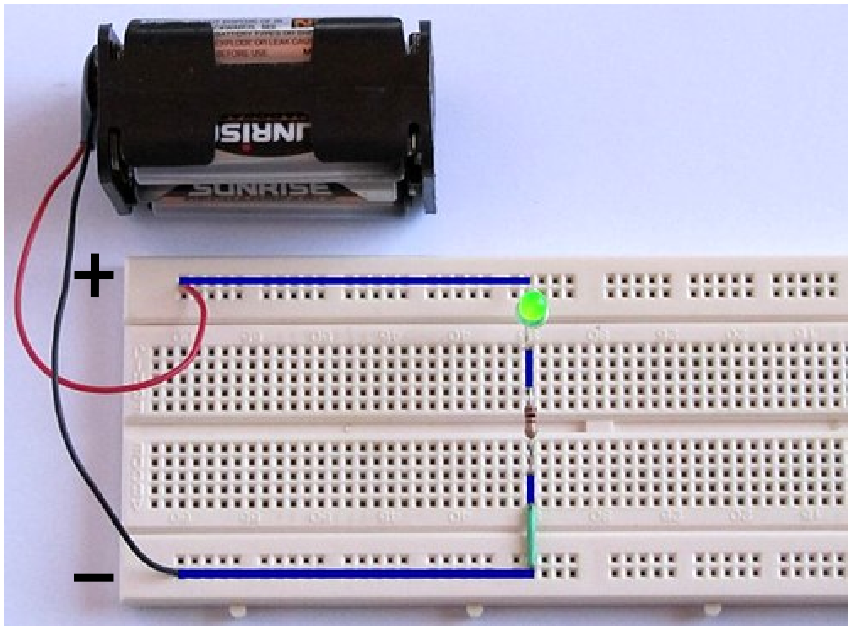
- The top two and bottom two rows of holes are connected horizontally. The middle of the breadboard has vertically connected strips separated by a horizontal channel in the middle.
- The red lines show how the holes are connected together inside the breadboard. Any component lead that is plugged into a hole of the breadboard will be connected to whatever is plugged into another hole on the same connecting strip.

Understanding Circuit Diagrams

- This is the circuit diagram for the circuit we've just created. The 'positive' lead of the battery is connected to the positive lead of the LED. The 'negative' lead of the LED is connected to one end of the resistor and the other end of the resistor is connected to the 'negative' lead of the battery.
- The resistor is shown as a rectangular block with it value in (390 Ohms) here



- We don't show the piece of wire we used on the diagram: it's just there to make sure the right parts of the circuit are connected together in the way that the diagram shows.
- The blue lines in this picture show how the internal connections in the breadboard connect everything up as we want.

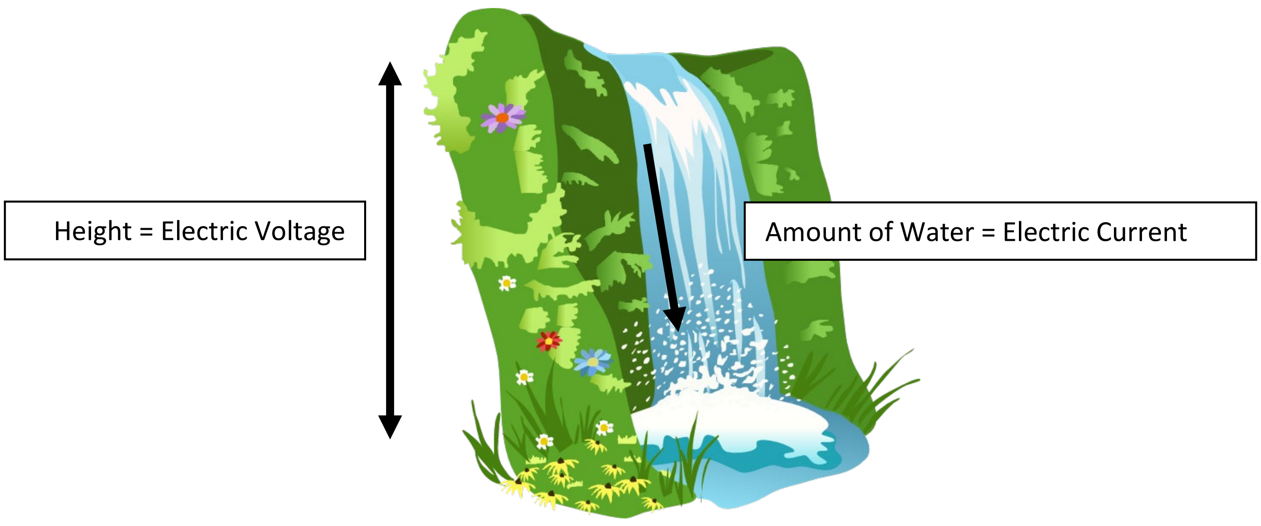


<u>What's the Resistor For?</u>

- Ok so we can see that the battery is there to supply the electrical power and the LED is there to supply the light but what is the resistor for?
- The resistor is there to control what is called the current in our circuit: too much current and the LED will blow up. Not enough current and the LED will not light up.

<u>Understanding Voltage and Current</u>

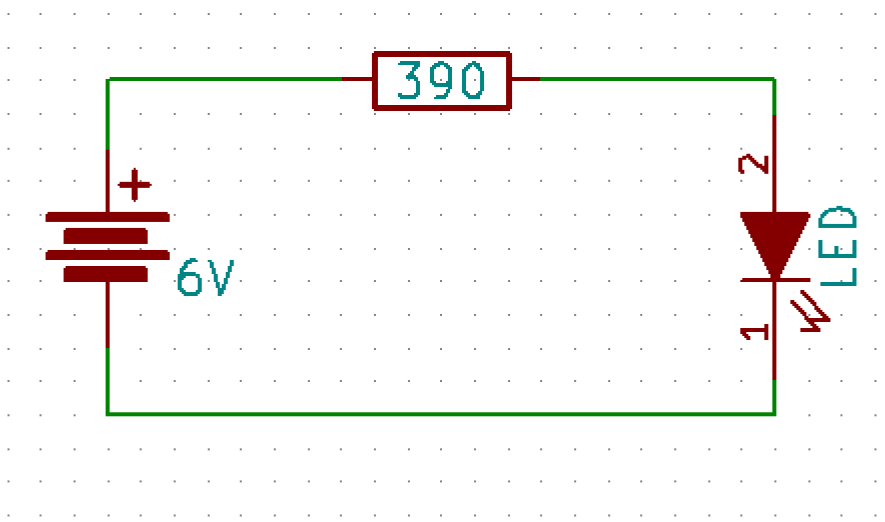
- Voltage and Current are two very simple properties of electric circuits but they can be difficult to understand at first.
- Think of your circuit as being like a waterfall: it has a fixed height, but the amount of water flowing over it can change.
- The height is like the voltage of our battery and the amount of water is like the electric current flowing through our circuit:



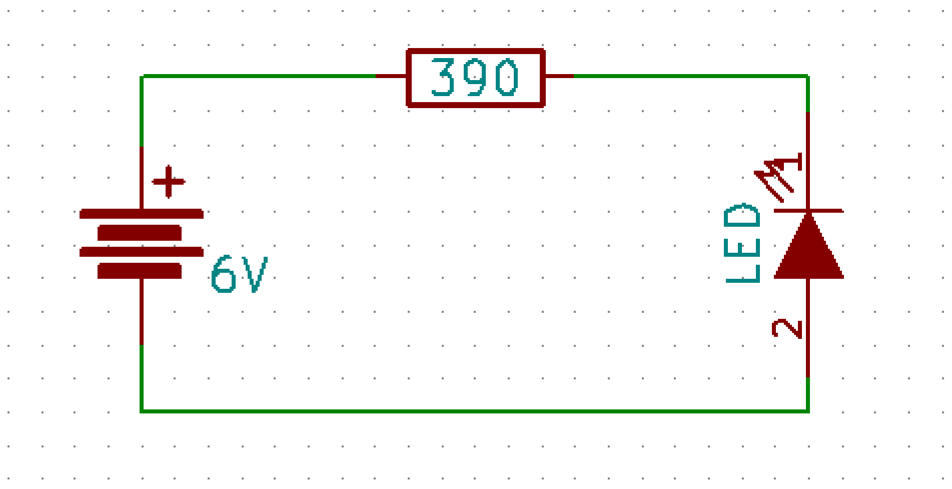
- If you think about it, you can have a high waterfall with not much water or a low waterfall with loads of water. A bit like this we can change the amount of current that flows in our electric circuit by changing the components we use in it.
- Our 1k resistor ensures that just the right amount of current flows to light up the LED.

More Electric Circuits

- What's different about the circuit diagram below? Can you change your circuit so it looks like this?



- Does the LED still light? Yes! This is because the same current is flowing in both the LED and the resistor.
- Now turn your LED round so your circuit looks like this:

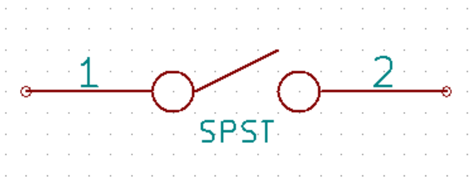


- What happens now? The LED doesn't light at all!
- This is because, when it's connected the wrong way round, the LED takes over from the resistor and doesn't allow any current to flow. It must be connected the right way round to light up.

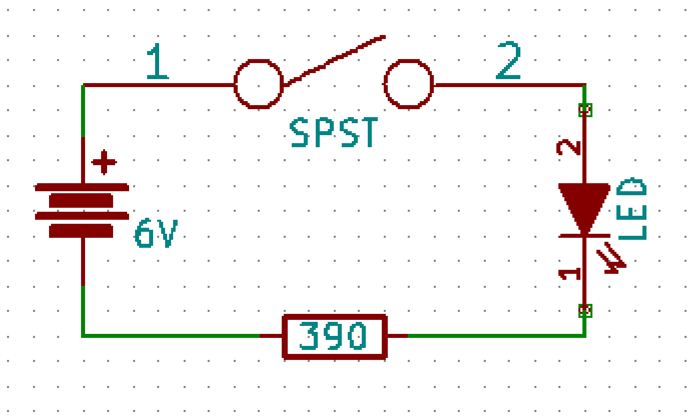
<u>One More Circuit</u>

- Let’s find a miniature push-button switch. It looks like this:

- And the symbol for it in a circuit diagram is this:



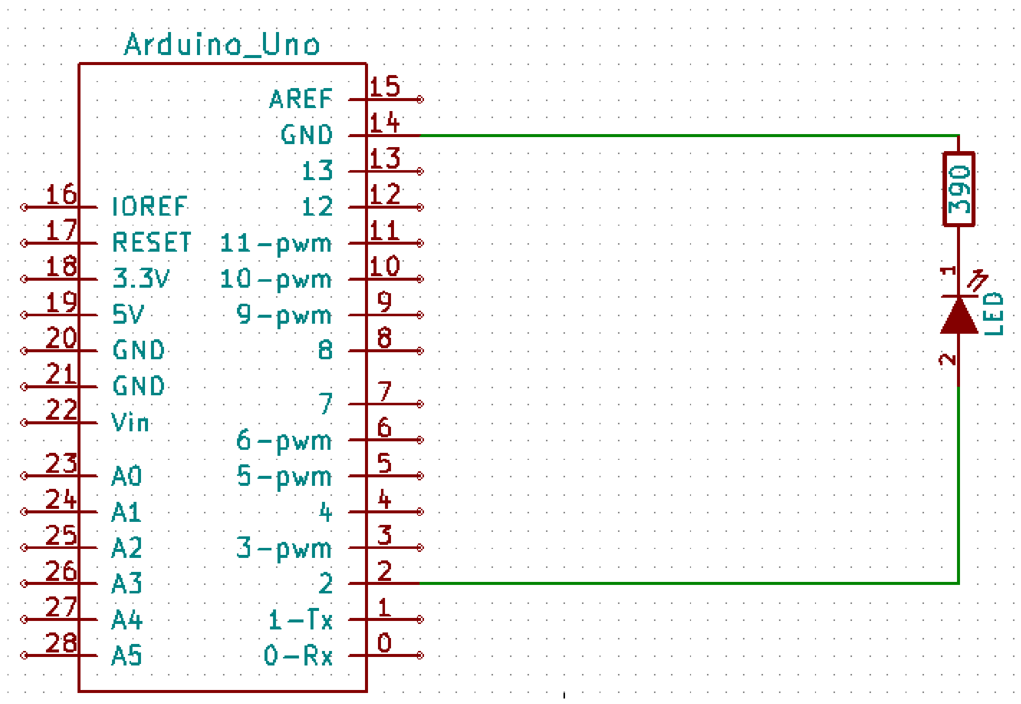
- Can you make a circuit like the following diagram from your existing components, the breadboard and your new switch?



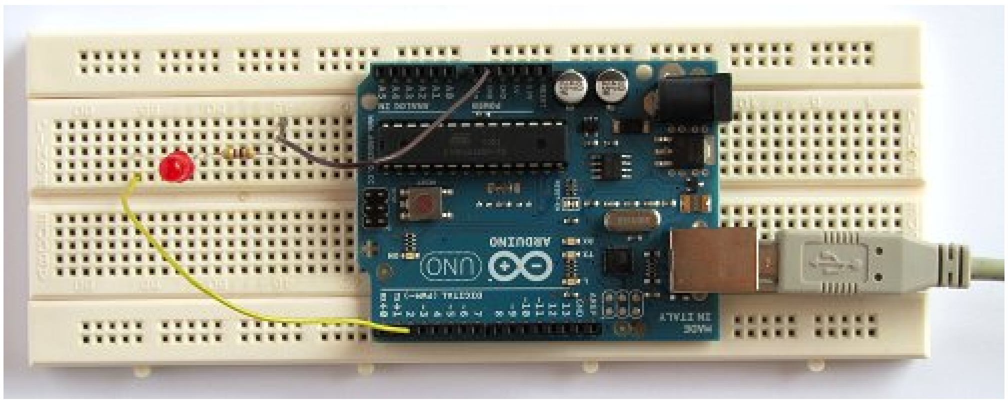
- If you’ve done this correctly, you now use the switch to turn the LED on and off.

Using the Arduino to Control a Circuit

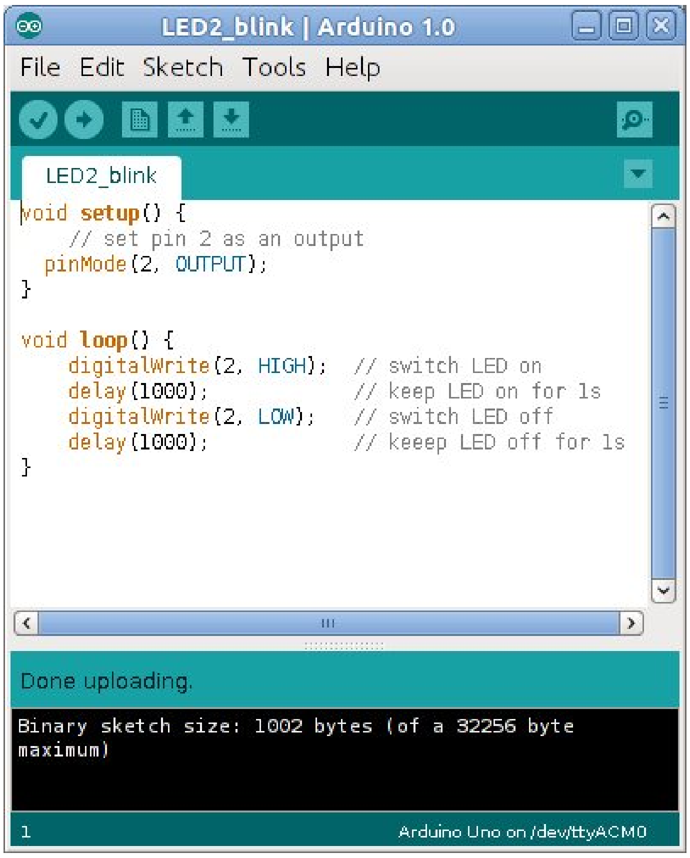
- It's time to use your Arduino again!
- The Arduino can be programmed to act as *both* the battery and the switch in our LED circuit so we can put those to one side for now.
- The circuit diagram below shows us how to connect our Arduino up. Pin '2' of the Arduino will be programmed to be an output and is connected to positive lead of the LED. Pin 2 is marked on the writing on the board. The negative LED lead connects to another resistor and the other lead of the resistor connects to a 'GND' pins on the Arduino (the Arduino Uno has three pins marked as GND, use any one of them).



- In this way we are using Pin 2 of the Arduino in the same way as the 'positive' terminal of or battery and the 'GND' pin as the negative terminal.
- If you've connected everything correctly your circuit might look a bit like this. The yellow wire goes from Pin 2 to the positive leg of the LED. The black wire goes from one end of the resistor to the 'GND' pin:



- Now we need a new program to control the LED on Pin 2 of the Arduino rather than Pin 13.
- In your 'Blink' program window, click on **File → Save As** and give it a new name like "LED2_blink"
- Edit your program so it blinks Pin 2. This is quite simple, it should look like this:



```
void setup() {
    // set pin 2 as an output
  pinMode(2, OUTPUT);
}

void loop() {
    digitalWrite(2, HIGH);   // switch LED on
    delay(1000);             // keep LED on for 1s
    digitalWrite(2, LOW);    // switch LED off
    delay(1000);             // keeep LED off for 1s
}
```

```
Done uploading.
Binary sketch size: 1002 bytes (of a 32256 byte
maximum)
```

Verify the Program

- In the Arduino IDE, save your new program.
- Click the **Verify** button on the top toolbar of the IDE to make sure that the program has no errors in it (the verify button is the first icon on the toolbar - the tick mark).
- If the program has any errors, then make sure that you typed the program in exactly as shown. Make corrections, save and verify again.

Upload the Program

- Click the **Upload** button to load the program to the Arduino board.
- After the upload completes, the external LED interfaced to pin 2 of the Arduino board should start to flash on and off just like the on-board LED did after loading the first program.

Challenge

- Can you add a second LED and resistor controlled via Pin 3 of the Arduino board and then program it to flash alternately with your first LED?