

**Arduino UNO Activity**

**Grade Levels 6-8**

What is Arduino?

Arduino is an open-source electronics platform based on easy-to-use hardware and software. [Arduino boards](https://www.arduino.cc/en/Main/Products) are able to read inputs - light on a sensor, a finger on a button, or a Twitter message - and turn it into an output - activating a motor, turning on an LED, publishing something online. You can tell your board what to do by sending a set of instructions to the microcontroller on the board. To do so you use the [Arduino programming language](https://www.arduino.cc/en/Reference/HomePage) (based on [Wiring](http://wiring.org.co/)), and [the Arduino Software (IDE)](https://www.arduino.cc/en/Main/Software), based on [Processing](https://processing.org/).

Why use Arduino?

Thanks to its simple and accessible user experience, Arduino has been used in thousands of different projects and applications. The Arduino software is easy-to-use for beginners, yet flexible enough for advanced users. It runs on Mac, Windows, and Linux. Teachers and students use it to build low cost scientific instruments, to prove chemistry and physics principles, or to get started with programming and robotics.

Arduino also simplifies the process of working with microcontrollers, but it offers some advantage for teachers, students, and interested amateurs over other systems:

* - Inexpensive - Arduino boards are relatively inexpensive compared to other microcontroller platforms. The least expensive version of the Arduino module can be assembled by hand, and even the pre-assembled Arduino modules cost less than $50
* - Cross-platform - The Arduino Software (IDE) runs on Windows, Macintosh OSX, and Linux operating systems. Most microcontroller systems are limited to Windows.
* - Simple, clear programming environment - The Arduino Software (IDE) is easy-to-use for beginners, yet flexible enough for advanced users to take advantage of as well. For teachers, it's conveniently based on the Processing programming environment, so students learning to program in that environment will be familiar with how the Arduino IDE works.
* - Open source and extensible software - The Arduino software is published as open source tools, available for extension by experienced programmers. The language can be expanded through C++ libraries, and people wanting to understand the technical details can make the leap from Arduino to the AVR C programming language on which it's based. Similarly, you can add AVR-C code directly into your Arduino programs if you want to.
* - Open source and extensible hardware - The plans of the Arduino boards are published under a Creative Commons license, so experienced circuit designers can make their own version of the module, extending it and improving it. Even relatively inexperienced users can build the [breadboard version of the module](https://www.arduino.cc/en/Main/Standalone) in order to understand how it works and save money.

Reference:

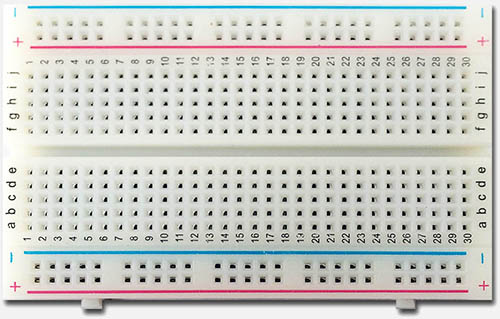
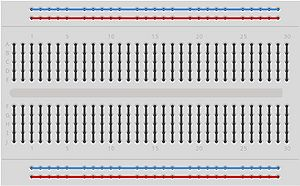
Arduino - Introduction. (2017). Retrieved September 13, 2017, from <https://www.arduino.cc/en/Guide/Introduction>

**Simple Circuit Activity - Using A Breadboard and Arduino UNO Micro Controller**

To start, it is imperative that the individual who is using the Arduino UNO understands the concepts of circuitry and the movement of electrons before the coding aspect is introduced into their repertoire of knowledge. This activity will give students the background knowledge of how to wire the breadboard to the Arduino UNO, in order for the coding that has been completed on the computer, to work.

Introduction

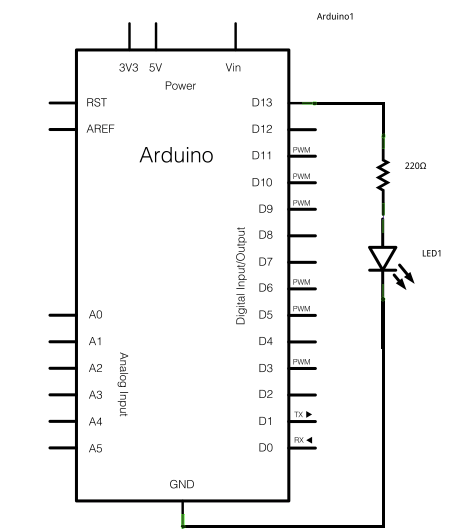
On it’s own, the Arduino UNO is an incredible piece of technology, but along with the breadboard, resistors, switches, lightbulbs and various other pieces, it becomes a tool that can be used to teach a variety of different aspects of STEAM.

The piece to the left is called the **BREADBOARD**. The image on the left is how the breadboard looks. It has a grid-like configuration, and under each port is copper wire, allowing for the flow of electricity when probes are plugged into the ports. The image on the right is a map of the copper wire within the breadboard. Copper wire runs **vertically** within the anode (positive, red) and cathode (negative, blue) columns. These are not connected to each other, or any other row on the board. The alphanumeric grid beside the anode and cathode columns, have copper wire running horizontally, with each numbered row. The copper does not connect between rows - that is how the user creates the circuit. 

Simple Circuit Building

In order for an electric current to flow through a circuit, the circuit must be complete - ie. it must run from the negative end of the battery to the positive end of the battery. This means that you must connect the negative end of the battery to the cathode (blue) column, and the positive end of the battery must be connected to the anode (red) column. From there, lightbulbs, resistors and motors can be placed into ports, and connected to the anode and cathode using connectors.

Below is the schematic of the simple circuit that we will be creating:



The different aspects of the Arduino UNO starter kit that you will need to use are:

* 1 LED light bulb
* 220Ω resistor (Red, Red, Brown stripes)
* Arduino UNO
* Connectors
* Breadboard

Procedure

Step 1 – Insert black jumper wire into the GND (Ground) pin on the Arduino and then in the GND (cathode; blue) rail of the breadboard.

Step 2 – Insert red jumper wire into the 5V pin on the Arduino and then the other end into the anode (red) rail on the breadboard

Step 3 – Place the LONG leg of the LED into any pin hole

Step 4 – Place the SHORT leg of the LED into any pin hole in a different row from the LONG leg.

Step 5 – Bend both legs of a 220 Ohm resistor and place one leg in the GND rail, and the other in the same row as the SHORT leg of the LED.

Step 6 - Use a connector to connect the LONG leg of the LED to the anode rail of the breadboard.

Step 7 – Connect the Arduino Uno to your computer via USB cable

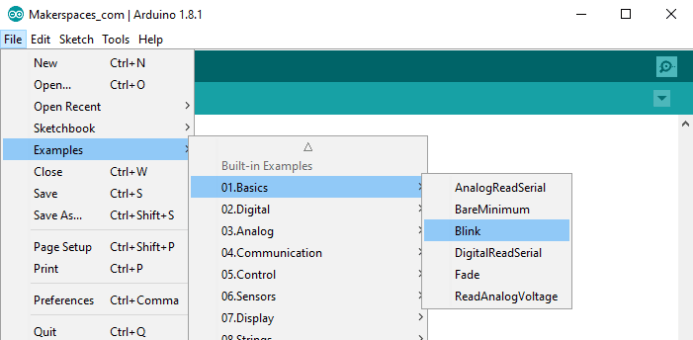
At this point, your light bulb should be turned on - you have created a simple circuit that allows the flow of electricity from your computer, to the Arduino UNO, to the breadboard and back.

Coding Your Simple Circuit

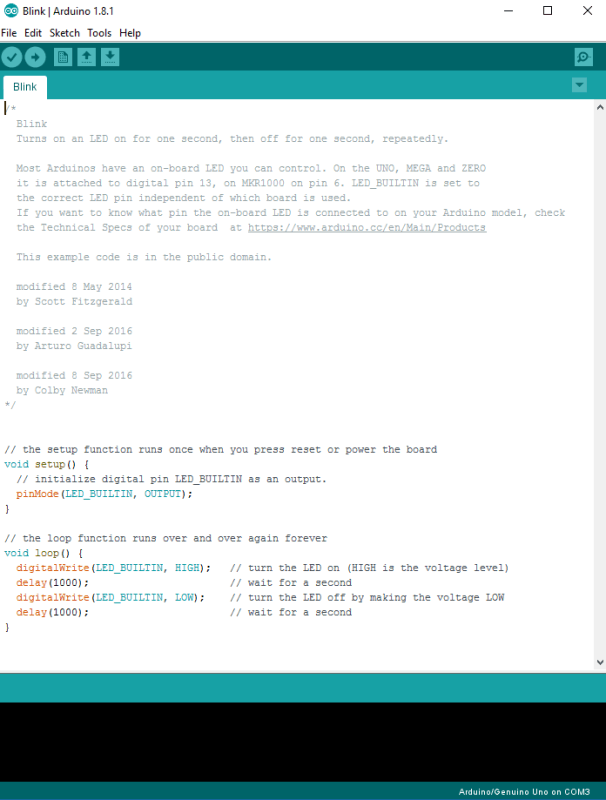
With the basic circuitry complete, we are going to go over the basics of the Arduino IDE ("**Integrated Development Environment**") or coding environment. Before the code will work, we need to switch the anode from the 5V to the Digital Pin (these pins can be coded to differ the output,

In the IDE, there are built-in example sketches that you can use which make it easy for beginners.

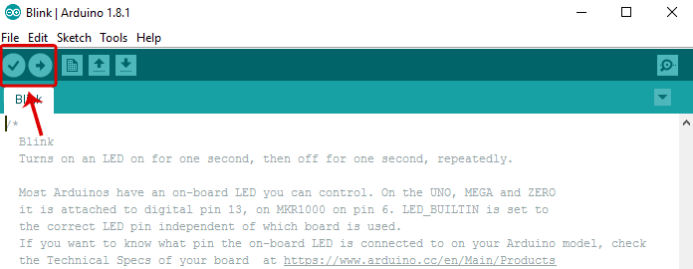
To open the blink sketch, you will need to go to **File > Examples > Basics > Blink**



Now you should have a fully coded blink sketch that looks like the image below.



Next, you need to click on the verify button (check mark) that’s located in the top left of the IDE box. This will compile the sketch and look for errors. Once it says “Done Compiling” you are ready to upload it. Click the upload button (forward arrow) to send the program to the Arduino board.



The built-in LEDs on the Arduino board will flash rapidly for a few seconds and then the program will execute. If everything went correctly, the LED on the breadboard should turn on for a second and then off for a second and continue in a loop.

\*NOTE: Arduino measures time in milliseconds and 1000 milliseconds = 1 second. The original code (1000) turns on the LED for 1 second and then off for 1 second. By adjusting the code from (1000) to (200) it shortens the time between on and off which makes it blink faster.

Projects adapted from: Simple Arduino Uno Projects For Beginners - Step-by-Step Tutorial. (2017, June 06). Retrieved September 19, 2017, from https://www.makerspaces.com/simple-arduino-projects-beginners/