# Living Machines

**Whether we can't live without them or are trying to get away from them, electronics are a permanent fixture in our everyday lives. From the new IPhone we're speaking through, or the satellite hovering a thousand miles above earth, electronics are everywhere in our lives.**

**Often the secrets of how these our devices work are hidden behind a thick case. But in their simplest form they every piece of technology senses something in the real world (touch, sound, temperature) and does something useful for us, whether it's turning on a TV, starting your car or playing your favorite music.**

**In this lesson and the lessons to follow, we are going to build our own electronics projects, and tell them what to do with the help of a tiny computer called "Arduino" that we'll program. While we're at it, we'll also explore the physics behind our project and "do the math" in a way that shouldn't make anyone want to cry. This lesson is tailored for students who are starting or have taken an introduction to Physics or Chemistry class. So if we're ready, let's get started!**

## Little Packets of Light

**What is light? It's responsible for the vibrant colors of flowers, the bursts of red and blue fireworks in the sky, or the shade of cyan on your room wall. White light reflects off of and is absorbed by the atoms in your shirt, and the color that we see is the color that's reflected. But if we could see the light down at the quantum level, we would see that light is made of little packets of energy called "photons." Their speed is second to nothing else in the universe. In fact, the speed of a photon is approximately 3.00 x 10^8 meters per second.**

**The component we will use for our first project gives away photons too. It's called a "Light Emitting Diode", or as you might know it better, an LED. If I hook our LED up to a circuit, like the one in the drawing or "schematic" shown below, we can figure out how it gives off light.**

**See, our 15 mm Red LED can handle at most 2 Volts (V) and 20 Milliamps (mA). If we connected our LED directly to +5V, our LED will explode. Don't try it. If you think**

**you'll see a cool explosion, you won't. The LED will just fizzle out and you'll have wasted an LED. So the resistor is there to protect our LED.**

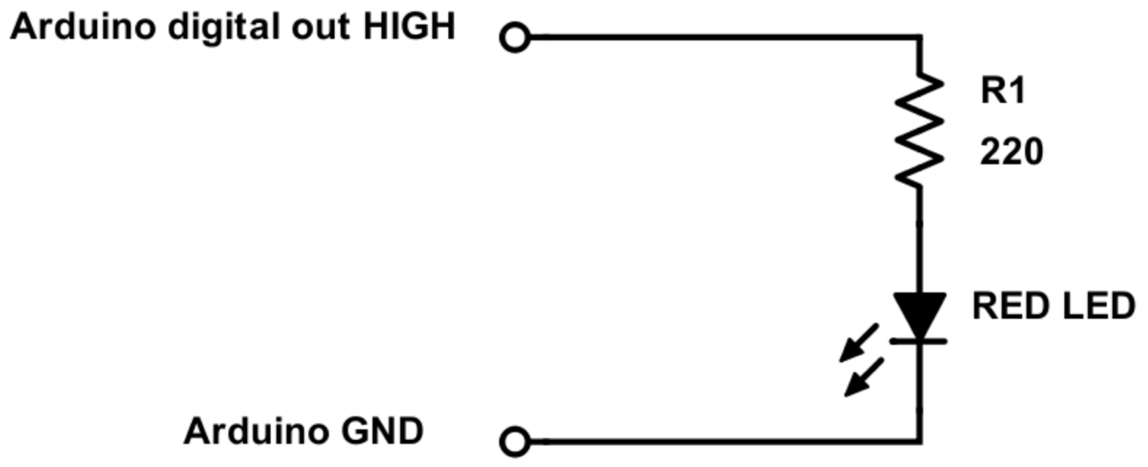

**Figure 1** This figure was taken from in-class lecture notes for Physics 124, created by Professor Kam Arnold

Arnold, Kam. "PHYSICS 124 Lecture 2." TritonEd. 2019, tritoned.ucsd.edu/bbcswebdav/pid-1377146-dt-content-rid-17636313_1/courses/PHYS124_FA18_A00/PHYS124_Arnold_02.pdf.

**The light that we can get from our LED depends on the wavelength of the photon that gets released from it. To find out our wavelength, we'll use two formulas you may have already seen before.**

$$(1)\ E = h\,\nu = \frac{h\,c}{\lambda} \quad \& \quad (2)\ V = I\,R$$

**The first one is the energy of one photon, and the second is Ohm's Law, which you've encountered when solving circuit problems. If one photon were to eject from our LED, it would take with it exactly 0.6 Electron Volts (eV). That means our LED we would have a drop in voltage, V, of 0.6 eV as well. If we rearrange our formula for E solving for λ and plug in our energy, E, we would get that our wavelength would be:**

$$\lambda = \frac{hc}{E} = \frac{200\ eV * nm}{0.6\ eV} = 2000\ nm$$

**If we use the familiar pneumonic for the spectrum of visible light (ROYGBIV), we see that the largest wavelength of visible light is Red with a wavelength of 700 nm. The wavelength of this photon is not visible light. It's Infrared; about three times the wavelength that we need.**

**However, by looking in the denominator of our equation for the energy of one photon, we see that we can get a smaller photon wavelength by increasing the voltage. Plugging the wavelength we want, we get that the voltage drop we need from our LED should be:**

$$\mathrm{E} = \frac{\mathrm{h\ c}}{\lambda} = \frac{200\ \mathrm{eV} * \mathrm{nm}}{700\ \mathrm{nm}} = 1.7\ \mathrm{eV}$$

**Changing from eV, we get that this is about 2 volts. We might now be worried that our new voltage will make our current that is greater than our 20 mA limit. Let's double check that the current that will pass through our LED is up to spec, using Ohm's Law, where V is  our voltage, I is our current and R is our 220 Ω resistor.**

$$\mathrm{I} = \frac{\mathrm{V}}{\mathrm{R}} = \frac{3.3\ V}{220\ \Omega} = 15\ mA$$

**This current is 15 mA is safely below the limit of 20 mA. So we won't have to worry that our LED will blow up in smoke. Now we can assemble our circuit, toting along all our information. Now we can put our circuit together. Here is a list of the supplies that you will need.**

## *Supplies*

*1) A breadboard*
*2) 220 Ω resistor*
*3) Jumper wires of various colors (preferably black and red)*
*4) 20 AWG wire (red and black)*
*5) Wire strippers*

**Take your black jumper wire and attach it from the digital pin labeled GND on your Arduino to the blue rail on your breadboard. Then attach a red jumper wire to digital pin 8 of your Arduino and the other end to the red power rail of your breadboard. These will be your ground and + 5V rail, respectively.**

**An important note about grounding your circuit, is that this is a very commonly forgotten step for beginners, professionals and myself (many times). Forgetting to do so can destroy parts of your circuit, and in some cases cause them to catch fire. Don't worry though. Instead, make it a habit to ground your circuit first.**

**To connect from our breadboard rails to the pins where we are going to build our circuit, we'll use our 20 AWG wire. Cutting our own sized wire will help make our circuits neat and tidy, and make it easier to find loose connections if there are any.**

**Take your wire strippers and cut one 1" piece of red wire and strip 1/8" off of each end. If you're having trouble getting the plastic from the wire to come off, bite down with the wire strippers, twist and pull.**

**Connect the wire to the power rail on your breadboard and the other end to one of the pin holes on your breadboard. Next to the pin, attach your 220 Ω resistor. On the other end of your resistor attach the positive (+), or longer leg of your LED and the negative (-) leg next to it.**

**Now take your wire strippers and cut a black wire just like the red one. Attach this black wire from the negative (-) end of your LED to your blue ground rail. If you finished putting together your circuit, it should look like Fig (2) below.**

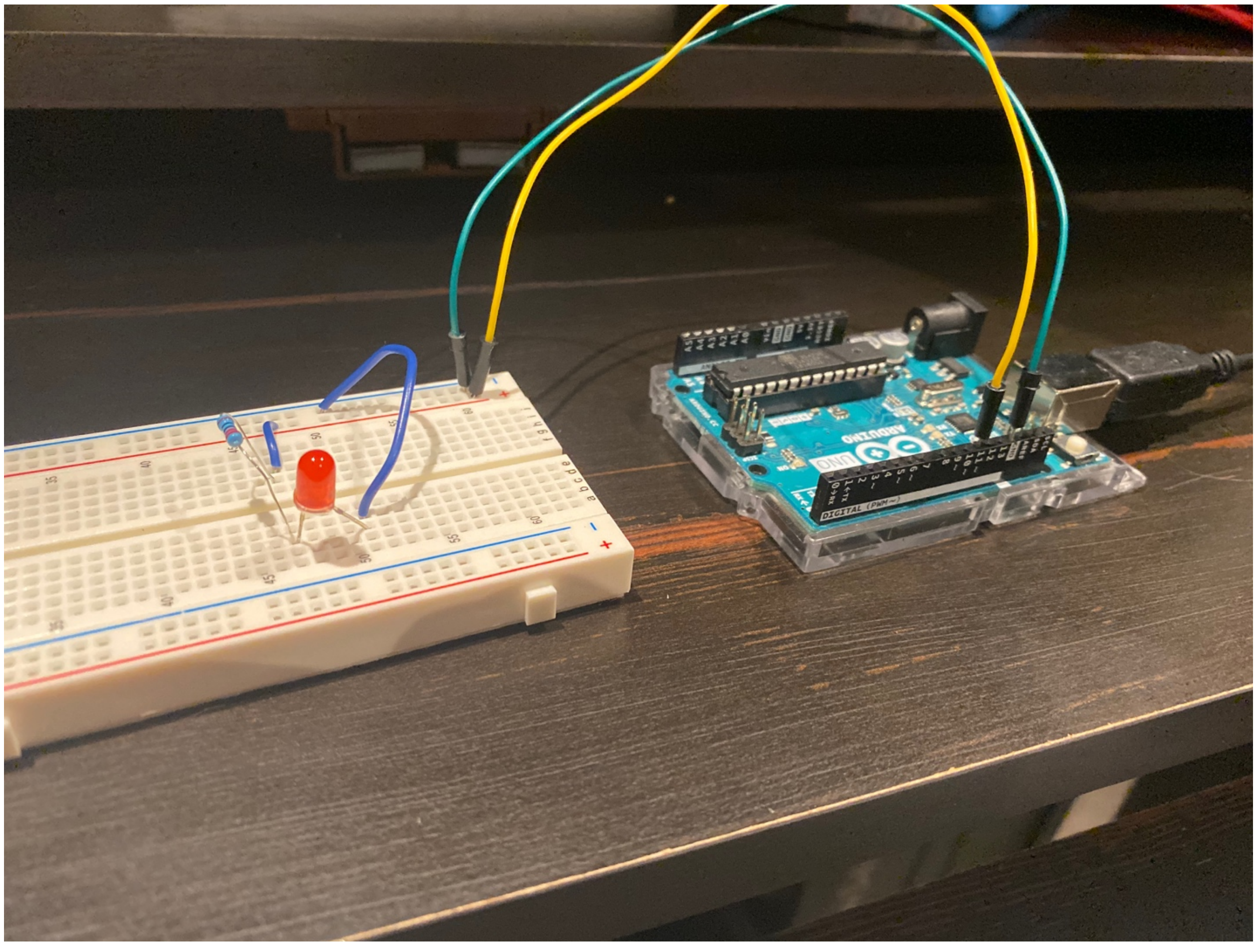

*Figure 2 LED circuit attached to Arduino.*

**Now go ahead and open up your Arduino Sketch. Once open, go to File > Save As, and name your project "BlinkingLED." Go ahead and save your project somewhere useful.**

**Make a space above void() setup{} and type two slashes like //. The two slashes should turn gray, along with whatever text you write after it. This is a comment. After this, describe what your program does, so that a person who doesn't know a lick of programming can understand. While what BlinkingLED does may seem obvious, as your code gets longer and more complex, so will explaining in simple terms what it does. For our program, a description like "This program, BlinkingLED, blinks an LED!"**

**Since we're still on the topic, let's break down the operation of our blinking LED into a series of steps that we can relate to functions. Let's introduce two functions that will be helpful.**

$$digitalWrite(pin\ \#, value)$$
$$delay(time)$$

**To blink our LED, we need it to turn ON, wait for a moment, turn OFF, then wait again. To turn the LED on, in void() loop{} we'll write digitalWrite() under void() loop{}. In the first entry in the parentheses, pin #, we'll put 8, which is our digital pin that will send +5V to our LED. To specify ON and send current to our LED, we'll type HIGH in the second entry.**

**To tell our Arduino to wait, we'll use the function delay() and in its parentheses write the amount of time we want it to wait. Time in the parentheses is in milliseconds, so write 1000, which is equal to a one second delay. Finally, repeat these two commands again, except now the 'value' in our digitalWrite() will be OFF instead of ON. Go ahead and compare your code with the example below.**

**Example 1: Blinking LED**

```
//Blink An LED: This program makes an LED blink ON and OFF

void setup() {
  // put your setup code here, to run once:
}

void loop() {
  // put your main code here, to run repeatedly:

  digitalWrite(11,HIGH); //Set pin 11 to HIGH
  delay(1000); //Delay for 1 second
  digitalWrite(11,LOW); //Set pin 11 to LOW
  delay(1000); //Delay 1 second
}
```

**One VERY important note. If you've noticed, at the end of every line of code under void() loop{} there's a semicolon (;). This is like the period at the end of a sentence. This tells Arduino that an action has been completed, and that it is ready to move onto the next line.**

**Without even one semicolon, your code will refuse to work. Dropping a semicolon is an extremely common coding mistake, and can mean the difference between finishing a program in thirty minutes and spending hours searching for your mistake.**

**Once you are done with your four lines of code, don't worry about adding anything more. These four lines are all that we will need, since our program will repeat in void() loop {}, unless we specify a stopping condition, or simply pull the plug. Go ahead and hit the 'Verify' button in the top left of your Arduino sketch. If you see any red errors pop up in the bottom of the screen, go through your code one line at a time. Check for common bugs like dropping a semicolon (;) or misspelling a function. Remember that spelling and the letter case of your functions matter, and Arduino will give you an error if either of these mistakes are made.**

**Otherwise, if there are no error, hit 'Send' next to 'Verify.' If all of your steps were successful, you should see a blinking LED. If not, then go back to your circuit and make sure all your parts on your breadboard are connected. If you don't remember how all the pins on the breadboard are connected, look at this guide.**

## What Next?

**Congratulations! You just completed your first Arduino project, so pat yourself on the back. We learned to 'talk' to our Arduino and tell it to do something, i.e. blink a light. In addition, we covered a lot of the physics behind light. But there is still a ways to go, before we have a basic understanding of how to build interactive projects with Arduino.**

**So far, we have yet to sense anything from the real world. In the next lesson we'll work with sensors that detect light and touch; learn to read signals that change continuously; and combine what we've learned to build our first truly interactive circuit.**