02 - Not just ON & OFF? ~ ~ ~ ~

In our last project, hopefully you created some really cool light effects with multiple LEDs. Our last project used `digitalWrite()` to turn the LEDs either ON or OFF. What about dimming or fading the LEDs? Let's look at how we can do this with a technique called PWM.

SKILL REQUIREMENTS

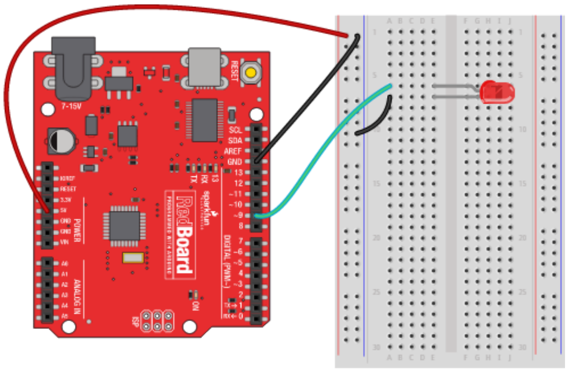
ELECTRICAL PROTOTYPING	■□□□□
ROBOTICS	□□□□□
SOLDERING	□□□□□
PROGRAMMING	■□□□□
DIY	□□□□□

MATERIALS LIST

- Computer
- Arduino UNO
- USB Cable
- LEDs
- Breadboard

STEP 1: Wiring

In our last project, we used a lot of LEDs. To get started with this one, let's start with just one LED - but, we're going to connect this LED to pin 9 instead of pin 13. Pin 9 is one of the special pins on the Arduino. Notice that it has a tilde symbol (~) next to it. More on that later.



Made with Fritzing.org

STEP 2: Example Code

Open up this example code (https://codebender.cc/sketch:136781), and upload it to your board.

What is the LED doing? How is this different from the original blink example? Now, let's look at the code.

STEP 3: Code Review

First, notice at the very top of the code, we have a line that reads:

```
int brightness = 2;
```

This declares an integer variable in the code that's called *`brightness`*. Variables are basically placeholders for values. The letters ***`int`*** before the name is what declares this variable as an integer type. This means that the value can be any non-decimal number, positive, negative, or zero.

Now, in the `loop()`, we see a single line of code (Remember that anything that has \\ in front of it is a comment).

```
analogWrite(9, brightness);
```

This sets the intensity of the LED using a technique called ***PWM*** or ***pulse-width-modulation***. The brightness can be any number from 0 to 255. A value of 255 represents full brightness (the same as setting a pin `HIGH`), and a value of 0 represents fully off (or setting a pin to `LOW`).

STEP 4: Incrementing along

The basic code example doesn’t do much. It just turns the LED on - dim. Let’s see if we can add some neat fading effects to our toolbox of code. Add these two lines of code to the loop(). The `loop()` code should look something like this:

```
void loop()
{
  analogWrite(9, brightness);
  brightness = brightness + 2;
  delay(20);
}
```

Upload this code. What do you see? Each time the loop runs, the `brightness` variable will be incremented by 2. And, recall that on each cycle the variable, `brightness`, is used to to set the intensity of the LED. Play around with this code a little. How might you change it so that it gets brighter faster? How about slower? Notice how the LED gets brighter, brighter, and brighter...and then resets? This is because it is going over the 255 limit. Let’s see how we might be able to increment up and then decrement back down.

STEP 5: for() loops

Sometimes you want to loop through a certain snippet of code a specific number of times. Introducing the for() loop. The for loop is a way to repeat a block of code a specific number of times and count through an index. The for() loop has three main parts - each separated by a semicolon. The first part initializes a variable, the second is a test / condition statement, and the third is an increment or decrement statement. Let’s look at the example that replicates what our code did above. Replace the code in your `loop()` with this.

```
for(int x = 0; x <=255; x = x + 2)
{
  analogWrite(9, x);
  delay(20);
}
```

Instead of using the brightness variable, this `for()` loop initializes a temporary variable called x and sets it equal to 0. It repeats while the variable is less than or equal to (<=) 255. And, each cycle, it increments the variable by 2. Now, let’s add the decrement so that we can get the LED to dim back down. This example here does that.

```
for(int x = 255; x >=0; x = x - 2)
{
  analogWrite(9, x);
  delay(20);
}
```

Notice that in this case, we initialize x to 255, the condition statement loops while x is greater than or equal to zero (>=), and we decrement x by 2 each time.

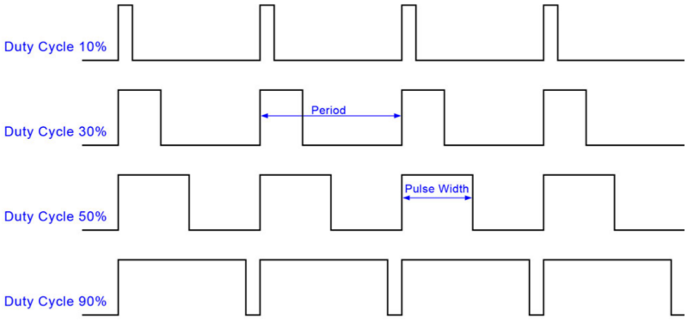
STEP 6: Challenge: Alternating LEDs

Now, without a diagram to help you, add a second LED and connect this LED to pin 10. Can you modify the code so that as one LED gradually gets brighter, the other LED gets dimmer? (Hint: Inside the `for()` loop, what is 255 - x?)

STEP 7: Build first / Teach second

We fully believe that you should play first before we lecture or go too deep into any hard content. So here's the "hard" content.

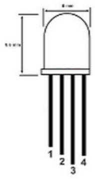
The Arduino is a purely digital device. What that means is that the pins are either HIGH (ON) or their LOW (OFF). But, how do we achieve something in between? Because the Arduino is a really fast device (running at 16 MHz - this is 62.5 nanoseconds per instruction), we can turn the LEDs on and off really fast -- so fast, you can't see it blink. Now, if we modify the amount of time the LED is ON compared to the amount of time that it's OFF, we can trick the eye into thinking that the LED is dimmer.



This is what is called ***PWM*** or **pulse-width modulation**. The diagram here shows an example of what's going on. So, basically, the Arduino is *faking* an analog signal. This is why we use the ~ symbol. One person we worked with said it's the "sort-of" symbol. So Arduino makes it "sort-of" analog.

Going Further

- Find the RGB LED in your kit. It has 4 legs. Blue, Green, GND, and Red. Wire this up to pins 9, 10, & 11 on the Arduino.
- Use the internet to find various color schemes. Colorpicker.com is a great example. Notice that the colors are all defined by their Red / Green / Blue values? And, these values range from 0 - 255? Find 3 - 4 of your favorite colors.
- Program your RGB LED to cycle through your favorite colors.
- Program the RGB LED to fade gradually from Red to Green to Blue and back to Red?



1 - Blue
2 - Green
3 - GND (common cathode)
4 - Red