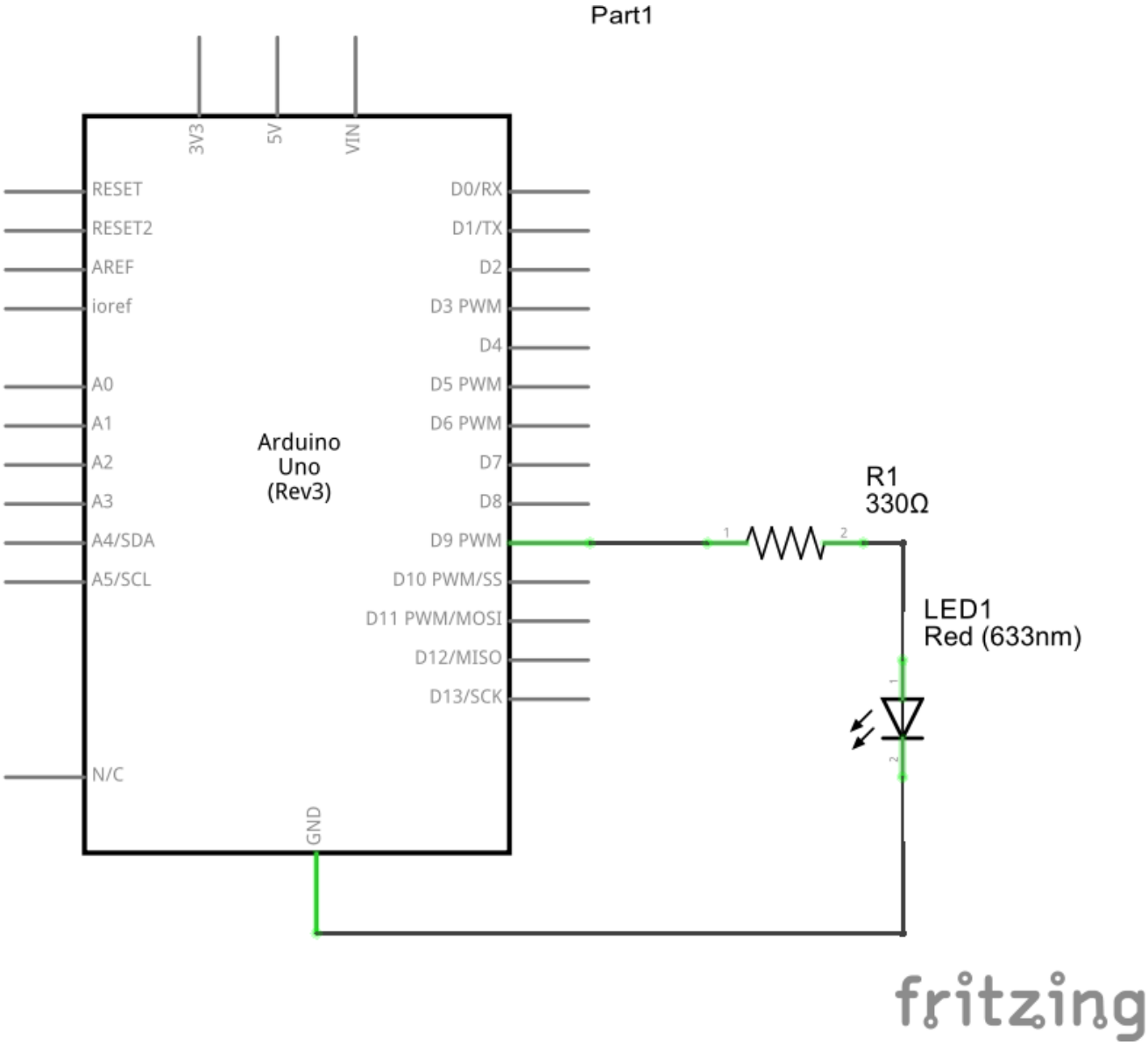
Pretend Analog

Before You Start

You should know the basics of inputs and outputs, as well as how to connect circuits on a breadboard. You should also know the basics of programming the Arduino (what setup is, what the loop does). There are many online tutorials, as well as the ones in this series that can help with these concepts.

The Circuit

Before we begin to build, lets take a quick look at what the circuit schematic looks like.



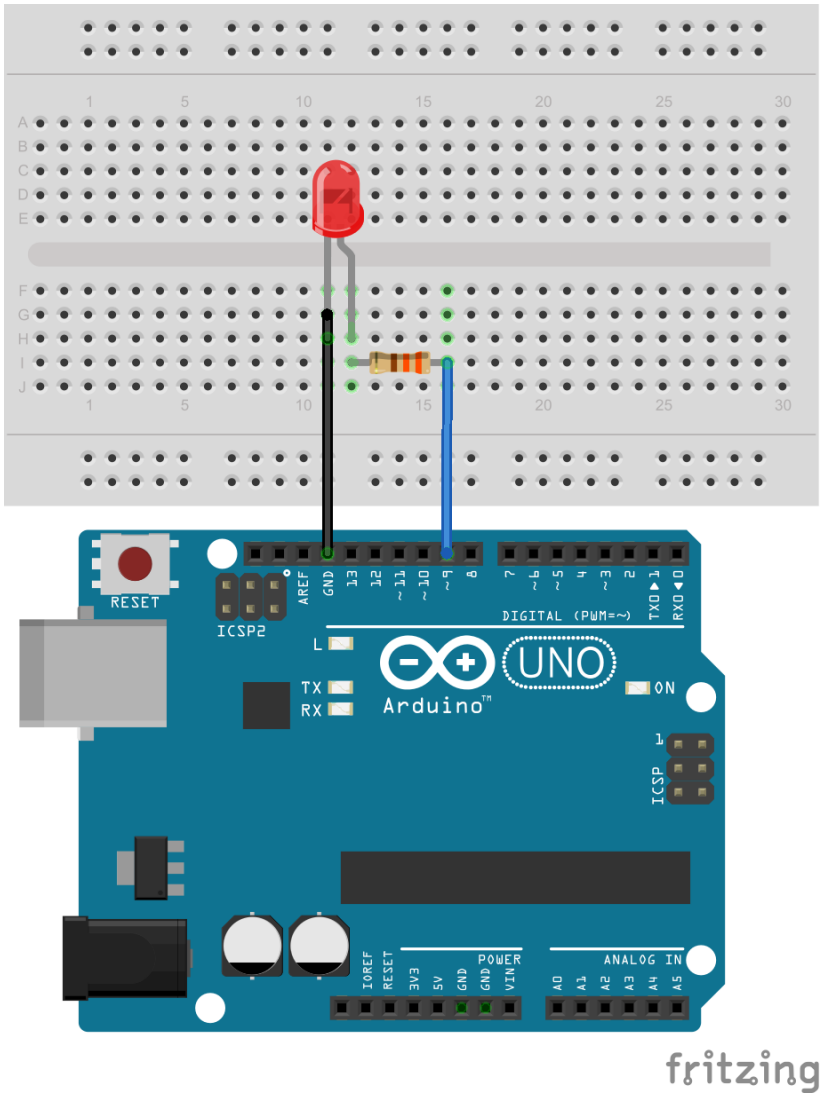
PWM PINS

This circuit is incredibly simple. All of the tricks we are performing are happening in software. However, there is something special about the pin that we connected the resistor and LED to (Digital Pin 9). This pin has the “PWM” designation, marked by a (~) on the board. Only pins with this marking are able to simulate an analog value.

The Breadboard

Below is an illustration of how the circuit is laid out on a breadboard. You can see the mark beside some of the pins, noting that they are PWM pins. The meaning of PWM will be explained later. Right now it is important to note that we connected the LED through a resistor, to prevent the LED from burning out.

At this point, knowing what resistor to select is very important? What is the rule of thumb for most breadboard LED’s? How can you calculate the value precisely?



The Code

The code for this project is below. The comments walk you through step by step on what each part means. To learn about some of the main concepts in this tutorial's code, look of the following on the Arduino website.

- analogWrite()
- 'for' loops

Before we begin writing the code, we need to understand how the analogWrite() function works. analogRead() is able to take in a real world value, not a 1 or 0, and convert it to a number, so we can perform math on it. However, the microcontroller does not have the ability to output anything except a 1 or 0. Instead, it turns the pin on and off quickly, which gives an average value between 5V and 0V. This concept is called "Pulse Width Modulation", or PWM. For a more detailed explanation, see here: https://www.arduino.cc/en/Tutorial/PWM

```
/*
 * In this sketch we create a blinking LED, but instead of
 * blinking on and off rapidly and sharply, it slowly fades
 * on and off.
 */

// To begin, we set value of the pin that we want the led to
// be connected to. We are using a variable definition, which
// is as valid as the #define statements from previous tutorials.
int led_pin = 9;

void setup() {
  // Much like analogRead(), analogWrite() understands that
  // we are going to be outputting a value, and so we do
  // not need to specify INPUT or OUTPUT. Therefore, nothing
  // needs to be done in the setup loop.
}

void loop() {
  // Below are several new concepts. Make sure that you
  // understand each concept, as all are essential in progressing
  // further.

  // Below we define a for loop. In this case, we are using the
  // variable fadeValue to tell the loop how long it should
  // continue running for. See below: We set fadeValue equal to
  // 0. Then, we say "As long as fadeValue is less than or equal
  // to 255, add 5 to fadeValue every loop". Once fadeValue
  // reaches this value, the loop will stop running and the code
  // will continue to the next block.
  for (int fadeValue = 0 ; fadeValue <= 255; fadeValue += 5) {

    // Inside the loop, we write an analog value to led_pin. The
    // value we are writing is fadeValue
    analogWrite(led_pin, fadeValue);

    // We implement a small delay so that the dimming effect can
    // be seen.
```

```
    delay(30);
  }

  // Below, we define another loop. However, in this loop we
  // let fadeValue start at 255, and decrement by 5 until
  // it reaches 0.
  for (int fadeValue = 255 ; fadeValue >= 0; fadeValue -= 5) {


    // Again, we set the led_pin analog value equal to fadeValue.
    analogWrite(led_pin, fadeValue);

    delay(30);
  }
}
```

Connect your Arduino to your computer, and open the Arduino IDE. Ensure that your Arduino is connected. Next, enter the code above (you don't need to include the comments) into your Arduino IDE.

Click verify (the check mark) to ensure that your code is error free. If everything looks good, then upload the code to the Arduino.

The Result

The result should be an LED that is slowly getting brighter and then darker. Some things to think about…

1. Why did we choose 255? Is there any significance to this value? What does analogWrite() expect?
2. How does PWM work? Does the Arduino change the frequency of the pulses?