Now, suppose one of those resistors is replaced with a variable resistor, su as a photoresistor. Photoresistors (see Figure 3-10) change resistance deper ing on the amount of light that hits them. In this case, I'll opt to use a 200 photoresistor. When in complete darkness, its resistance is about 200kΩ; wh saturated with light, the resistance drops nearly to zero. Whether you choo to replace R1 or R2 and what value you choose to make the fixed resistor w affect the scale and precision of the readings you receive. Try experimenti with different configurations and using the serial monitor to see how your valu change. As an example, I will choose to replace R1 with the photoresistor, a I'll make R2 a 10kΩ resistor (see Figure 3-11). You can leave the RGB LED in pla for now, though you'll only use one of the colors for this exercise.

Credit: element14, www.element14.com

Figure 3-10: Photoresistor

Load up your trusty serial printing sketch again (Listing 3-1) and try chan ing the lighting conditions over the photoresistor. Hold it up to a light and c it with your hands. Odds are, you aren't going to be hitting the full range fro 0 to 1023 because the variable resistor will never have a resistance of zero. Rathe you can probably figure out the maximum and minimum values that you a likely to receive. You can use the data from your photoresistor to make a mo intelligent nightlight. The nightlight should get brighter as the room gets dark and vice versa. Using your serial monitor sketch, pick the values that represe when your room is at full brightness or complete darkness. In my case, I foun that a dark room has a value of around `200` and a completely bright room ha a value around `900`. These values will vary for you based upon your lightin conditions, the resistor value you are using, and the value of your photoresisto

Using Analog Inputs to Control Analog Outputs

Recall that you can use the `analogWrite()` command to set the brightness of a LED. However, it is an 8-bit value; that is, it accepts values between 0 and 25 only, whereas the ADC is returning values as high as 1023. Conveniently, th Arduino programming language has two functions that are useful for mappin between two sets of values: the `map()` and `constrain()` functions. The map function looks like this:

```
output = map(value, fromLow, fromHigh, toLow, toHigh)
```

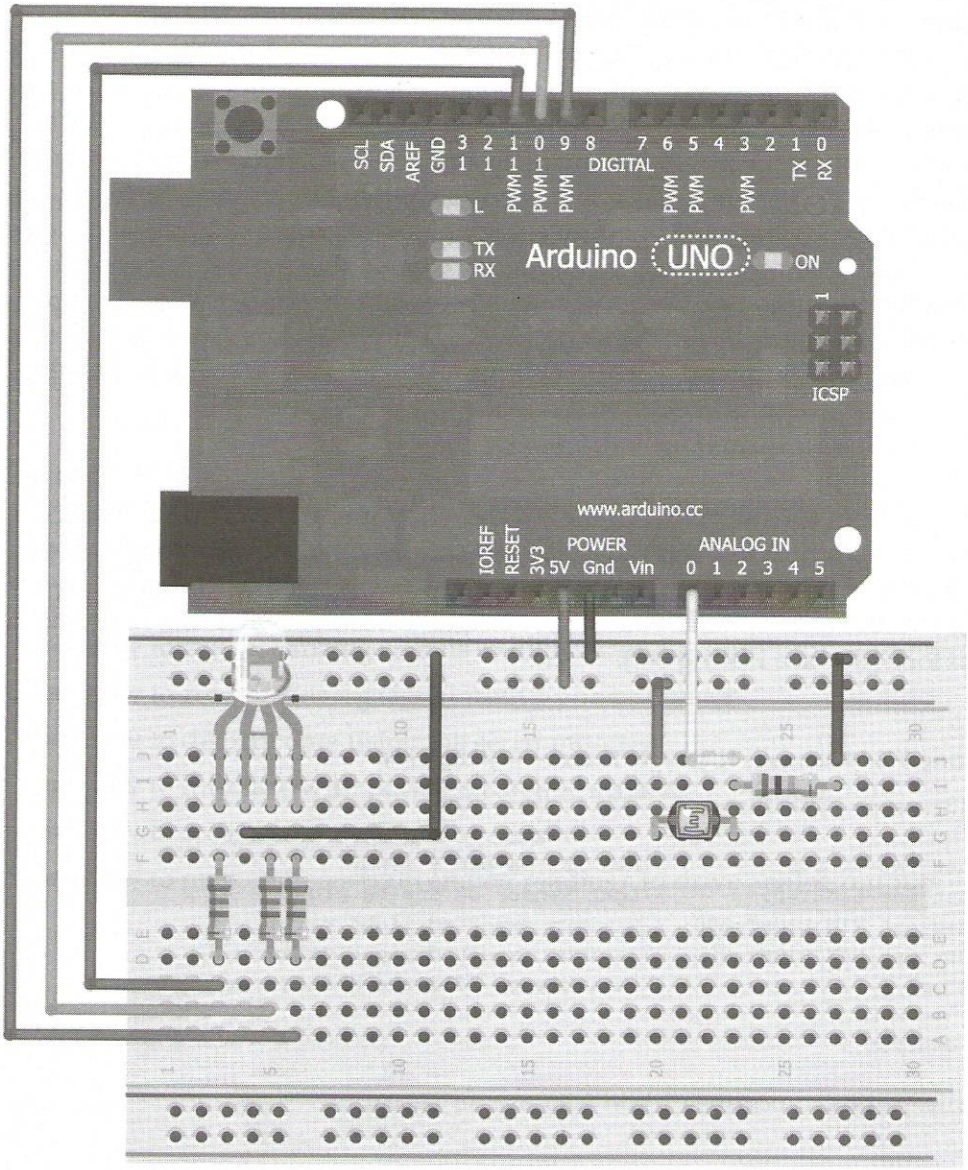

Image created with Fritzing.

Figure 3-11: Photoresistor circuit

`value` is the information you are starting with. In your case, that's the most recent reading from the analog input. `fromLow` and `fromHigh` are the input boundaries. These are values you found to correspond to the minimum and maximum brightness in your room. In my case, they were `200` and `900`. `toLow` and `toHigh` are the values you want to map them to. Because `analogWrite()` expects value between `0` and `255`, you use those values. However, we want a darker room to map to a brighter LED. Therefore, when the input from the ADC is a low value, you want the output to the LED to be a high value, and vice versa.

Conveniently, the `map` function can handle this automatically; simply swap the high and low values so that the low value is `255` and the high value is 0. The `map()` function creates a linear mapping. For example, if your `fromLow` and `fromHigh` values are `200` and `900`, respectively, and your `toLow` and `toHigh` values are `255` and `0`, respectively, `550` maps to `127` because `550` is halfway between `200` and `900` and `127` is halfway between `255` and `0`. Importantly, however, the `map()` function does not constrain these values. So, if the photoresistor does measure a value below `200`, it is mapped to a value above `255` (because you are inverting the mapping). Obviously, you don't want that because you can't pass a value greater than `255` to the `analogWrite()` function. You can deal with this by using the `constrain()` function. The `constrain()` function looks like this:

```
output = constrain(value, min, max)
```

If you pass the output from the `map` function into the `constrain` function, you can set the `min` to `0` and the `max` to `255`, ensuring that any numbers above or below those values are constrained to either `0` or `255`. Finally, you can then use those values to command your LED! Now, take a look at what that final sketch will look like (see Listing 3-3).

Listing 3-3: Automatic Nightlight Sketch—nightlight.ino

```
//Automatic Nightlight

const int RLED=9;          //Red LED on pin 9 (PWM)
const int LIGHT=0;         //Lght Sensor on analog pin 0
const int MIN_LIGHT=200;   //Minimum expected light value
const int MAX_LIGHT=900;   //Maximum Expected Light value
int val = 0;               //variable to hold the analog reading

void setup()
{
  pinMode(RLED, OUTPUT); //Set LED pin as output
}

void loop()
{
  val = analogRead(LIGHT);                          //Read the light sensor
  val = map(val, MIN_LIGHT, MAX_LIGHT, 255, 0); //Map the light reading
  val = constrain(val, 0, 255);                     //Constrain light value
  analogWrite(RLED, val);                           //Control the LED
}
```

Note that this code reuses the `val` variable. You can alternatively use a different variable for each function call. In functions such as `map()` where `val` is both the input and the output, the previous value of `val` is used as the input, and its value is reset to the updated value when the function has completed.

Play around with your nightlight. Does it work as expected? Remember, you can adjust the sensitivity by changing the minimum and maximum bounds of the mapping function or changing the fixed resistor value. Use the serial monitor to observe the differences with different settings until you find one that works the best. Can you combine this sketch with the color-selection nightlight that you designed in the preceding chapter? Try adding a button to switch between colors, and use the photoresistor to adjust the brightness of each color.

Summary

In this chapter you learned about the following:

- The differences between analog and digital signals
- How to convert analog signals to digital signals
- How to read an analog signal from a potentiometer
- How to display data using the serial monitor
- How to interface with packaged analog sensors
- How to create your own analog sensors
- How to map and constrain analog readings to drive analog outputs