Circuit 1C: Photoresistor

In circuit 1B, you got to use a potentiometer, which varies resistance based on the twisting of a knob. In this circuit, you'll be using a photoresistor, which changes resistance based on how much light the sensor receives. Using this sensor you can make a simple night-light that turns on when the room gets dark and turns off when it is bright.

YOU NEED

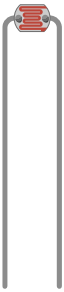

LED


PHOTORESISTOR


330Ω RESISTOR


10KΩ RESISTOR


7 JUMPER WIRES



NEW COMPONENTS

PHOTORESISTORS are light-sensitive, variable resistors. As more light shines on the sensor's head, the resistance between its two terminals decreases. They're an easy-to-use component in projects that require ambient-light sensing.

NEW CONCEPTS

ANALOG TO DIGITAL CONVERSION:
In order to have the RedBoard sense analog signals, we must first pass them through an Analog to Digital Converter (or ADC). The six analog inputs (A0–A5) covered in the last circuit all use an ADC. These pins sample the analog signal and create a digital signal for the microcontroller to interpret. The **resolution** of this signal is based on the resolution of the ADC. In the case of the RedBoard, that resolution is 10-bit. With a 10-bit ADC, we get 2 ^ 10 = 1024 possible values, which is why the analog signal can vary between 0 and 1023.

VOLTAGE DIVIDERS CONTINUED:
Since the RedBoard can't directly interpret resistance (rather, it reads voltage), we need to use a voltage divider to use our photoresistor, a part that doesn't output voltage. The resistance of the photoresistor changes as it gets darker or lighter. That changes or "divides" the voltage going through the divider circuit. That divided voltage is then read in on the analog to digital converter of the analog input.

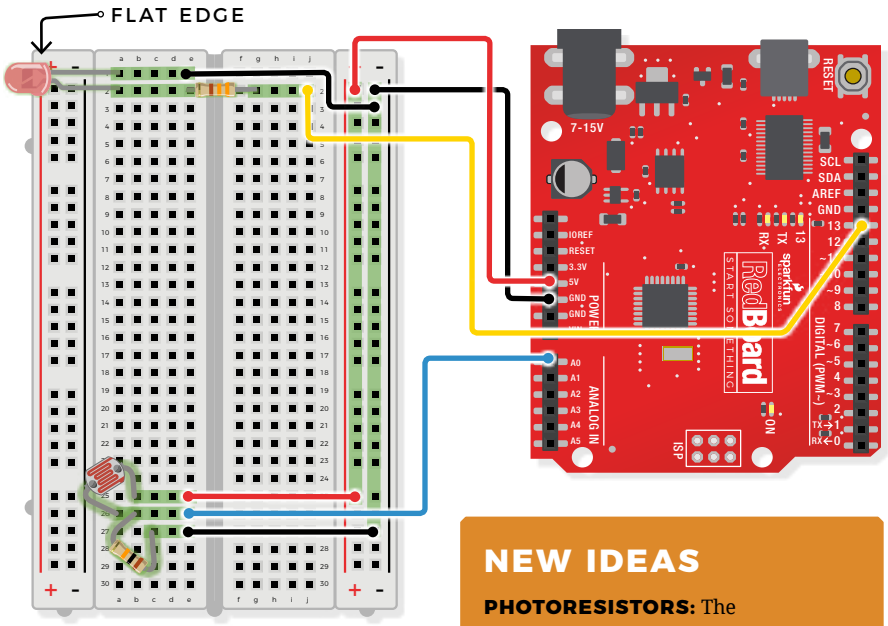
The voltage divider equation:

$$V_{out} = V_{in} \bullet \frac{R_2}{R_1 + R_2}$$

assumes that you know three values of the above circuit: the input voltage (V_{in}), and both resistor values (R_1 and R_2). If R1 is a constant value (the resistor) and R2 fluctuates (the photoresistor), the amount of voltage measured on the Vout pin will also fluctuate.

HOOKUP GUIDE

READY TO START HOOKING EVERYTHING UP? Check out the circuit diagram and hookup table below to see how everything is connected.



NEW IDEAS

PHOTORESISTORS: The photoresistors, like regular resistors, are not polarized and can be installed in either direction.

CONNECTION TYPES ◆ REDBOARD CONNECTION ■ BREADBOARD CONNECTION

JUMPER WIRES	◆ 5V to ■ 5V(+)
	◆ GND to ■ GND (-)
	◆ D13 to ■ J2
	◆ A0 to ■ E26
	■ E1 to ■ GND(-)
	■ E25 to ■ 5V(+)
	■ E27 to ■ GND(-)
LED	■ A1(-) to ■ A2(+)
330Ω RESISTOR (ORANGE, ORANGE, BROWN)	■ E2 to ■ F2
10KΩ RESISTOR (BROWN, BLACK, ORANGE)	■ B26 to ■ C27
PHOTORESISTOR	■ A26 to ■ B25

Open the Arduino IDE

Connect the RedBoard to a USB port on your computer.

Open the Sketch:
File > Examples > SIK_Guide_Code-V_4 > **CIRCUIT_1C-PHOTORESISTOR**

Select **UPLOAD** to program the sketch on the RedBoard.

WHAT YOU SHOULD SEE

The program stores the light level in a variable. Using an **if**/**else** statement, the variable value is compared to the threshold. If the variable is above the threshold (it's bright), turn the LED off. If the variable is below the threshold (it's dark), turn the LED on. Open the Serial Monitor in Arduino. The value of the photoresistor should be printed every so often. When the photoresistor value drops below the threshold, the LED should turn on (you can cover the photoresistor with your finger for testing).

NEW IDEAS

LIGHT LEVELS: If the room you are in is very bright or dark, you may have to change the value of the `threshold` variable in the code to make your night-light turn on and off. See the Troubleshooting section for instructions.

PROGRAM OVERVIEW

1	Store the light level in the variable **`photoresistor`**.
2	If the value of the **`photoresistor`** is above the **`threshold`** (it's bright), turn the LED off.
3	Otherwise, the value of the **`photoresistor`** is below the **`threshold`** (it's dark), turn the LED on.

CODE TO NOTE

IF ELSE STATEMENTS:

```
if(logic statement){
//run if true
}
else{
//run if false
}
```

The **if else** statement lets your code react to the world by running one set of code when the logic statement in the round brackets is true and another set of code when the logic statement is false. For example, this sketch uses an **if** statement to turn the LED on when it is dark, and **else** statement to turn the LED off when it is light.

LOGICAL OPERATORS:

```
(photoResistor <
threshold)
```

Programmers use logic statements to translate things that happen in the real world into code. Logic statements use logical operators like 'equal to' ==, 'greater than' > and 'less than' <, to make comparisons. When the comparison is true (e.g., 4 < 5), then the logic statement is true. When the comparison is false (e.g., 5 < 4) then the logic statement is false. This example is asking whether the variable `photoresistor` is less than the variable `threshold`.

CODING CHALLENGE

RESPONSE PATTERN: Right now your **if** statement turns the LED on when it gets dark, but you can also use the light sensor like a no-touch button. Try using `digitalWrite()` and `delay()` to make the LED blink a pattern when the light level drops, then calibrate the threshold variable in the code so that the blink pattern triggers when you wave your hand over the sensor.

REPLACE 10KΩ RESISTOR WITH AN LED: Alter the circuit by replacing the 10KΩ resistor with an LED (the negative leg should connect to GND). Now what happens when you place your finger over the photoresistor? This is a great way to see Ohm's law in action by visualizing the effect of the change in resistance on the current flowing through the LED.

TROUBLESHOOTING

Nothing is printing in the Serial Monitor	Try unplugging your USB cable and plugging it back in. In the Arduino IDE, go to Tools > Port, and make sure that you select the right port.
The light never turns on or always stays on	Start the Serial Monitor in Arduino. Look at the value that the photoresistor is reading in a bright room (e.g., 915). Cover the photoresistor, or turn the lights off. Then look at the new value that the photoresistor is reading (e.g., 550). Set the threshold in between these two numbers (e.g., 700) so that the reading is above the threshold when the lights are on and below the threshold when the lights are off.

You've completed Circuit 1C!

Continue to circuit 1D to learn about RGB LEDs, functions and pulse-width modulation.