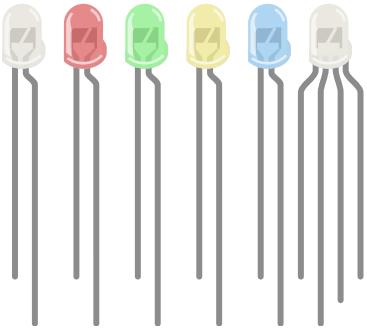
Circuit 1A: Blinking an LED

You can find LEDs in just about any source of light, from the bulbs lighting your home to the tiny status lights flashing on your home electronics. Blinking an LED is the classic starting point for learning how to program embedded electronics. It's the "Hello, World!" of microcontrollers. In this circuit, you'll write code that makes an LED blink on and off.

YOU NEED

NEW COMPONENTS

LIGHT-EMITTING DIODES (LEDS) are small lights made from a silicon diode. They come in different colors, brightnesses and sizes. LEDs (pronounced el-ee-dees) have a positive (+) leg and a negative (-) leg, and they will only let electricity flow through them in one direction. LEDs can also burn out if too much electricity flows through them, so you should always use a



resistor to limit the current when you wire an LED into a circuit.

RESISTORS resist the flow of electricity. You can use them to protect sensitive components like LEDs. The strength of a resistor (measured in ohms) is marked on the body of the resistor using small colored bands. Each color stands for a number, which you can look up using a resistor chart. One can be found at the back of this book.

NEW CONCEPTS

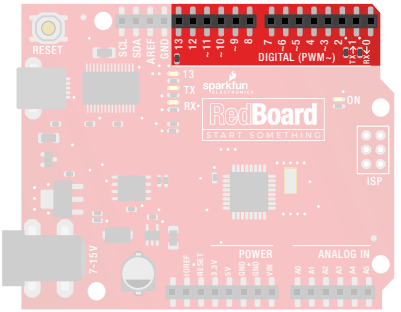
POLARITY: Many electronics components have polarity, meanig electricity can (and should) flow through them in only one direction. Polarized components, like an LED, have a positive and a negative leg and only work when electricity flows through them in one direction. Some components, like resistors, do not have polarity; electricity can flow through them in either direction.

OHM'S LAW describes the relationship between the three fundamental elements of electricity: **voltage**, **resistance** and **current**. This relationship can be represented by this equation:

$$V=I \cdot R$$

V = Voltage in volts

I = Current in amps



R = Resistance in ohms (Ω)

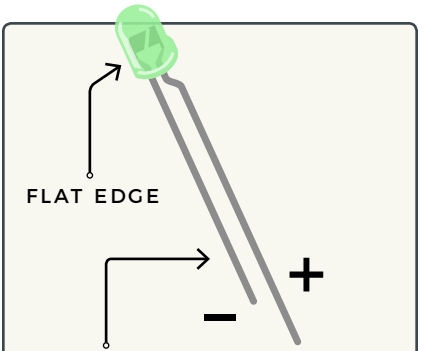
This equation is used to calculate what resistor values are suitable to sufficiently limit the current flowing to the LED so that it does not get too hot and burn out.

DIGITAL OUTPUT: When working with microcontrollers such as the RedBoard, there are a variety of pins to which you can connect electronic components. Knowing which pins perform which functions is important when building your circuit. In this circuit, we will be using what is known as a digital output. There are 14 of these pins found on the RedBoard. A digital output only has **two states:**

NEW IDEAS

ELECTRICAL SAFETY: Never work on your circuits while the board is connected to a power source. The SparkFun RedBoard operates at 5 volts, which, while not enough to injure you, is enough to damage the components in your circuit.

COMPONENT ORIENTATION & POLARITY: Instructions on how to orient each of the new components will be given before each circuit diagram. Many components have polarity and have only one correct orientation, while others are nonpolarized.

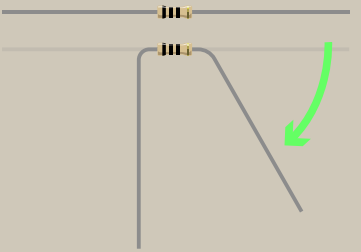


POLARIZED COMPONENTS

Pay close attention to the LED. The negative side of the LED is the short leg, marked with a flat edge.

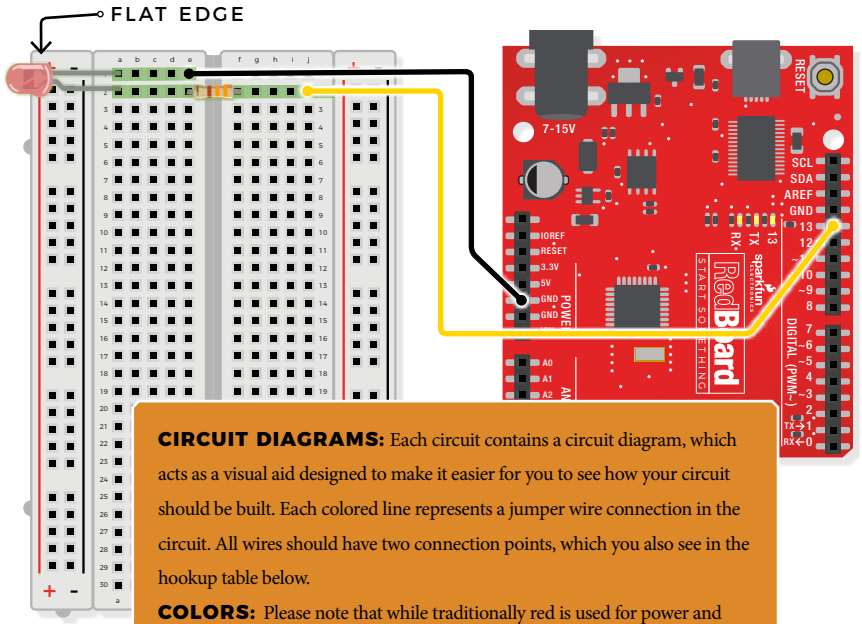
RESISTOR LEADS

Components like resistors need to have their legs bent into 90° angles in order to correctly fit in the breadboard sockets.



HOOKUP GUIDE

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



CIRCUIT DIAGRAMS: Each circuit contains a circuit diagram, which acts as a visual aid designed to make it easier for you to see how your circuit should be built. Each colored line represents a jumper wire connection in the circuit. All wires should have two connection points, which you also see in the hookup table below.

COLORS: Please note that while traditionally red is used for power and black is used for ground, all wires, no matter their color, function the same.

HOOKUP TABLES: Many electronics beginners find it helpful to have a coordinate system when building their circuits. For each circuit, you'll find a hookup table that lists the coordinates of each component or wire and where it connects to the RedBoard, the breadboard, or both. The breadboard has a letter/number coordinate system, just like the game Battleship.

◆ D13 *to* ■ J2

...means one end of a component connects to digital pin 13 on your RedBoard and the other connects to J2 on the breadboard

CONNECTION TYPES	◆ REDBOARD CONNECTION ■ BREADBOARD CONNECTION
JUMPER WIRES	◆D13 *to* ■ J2 ◆ GND *to* ■ E1
LED	■ A1(-) *to* ■ A2(+)
330Ω RESISTOR (ORANGE, ORANGE, BROWN)	■ E2 *to* ■ F2

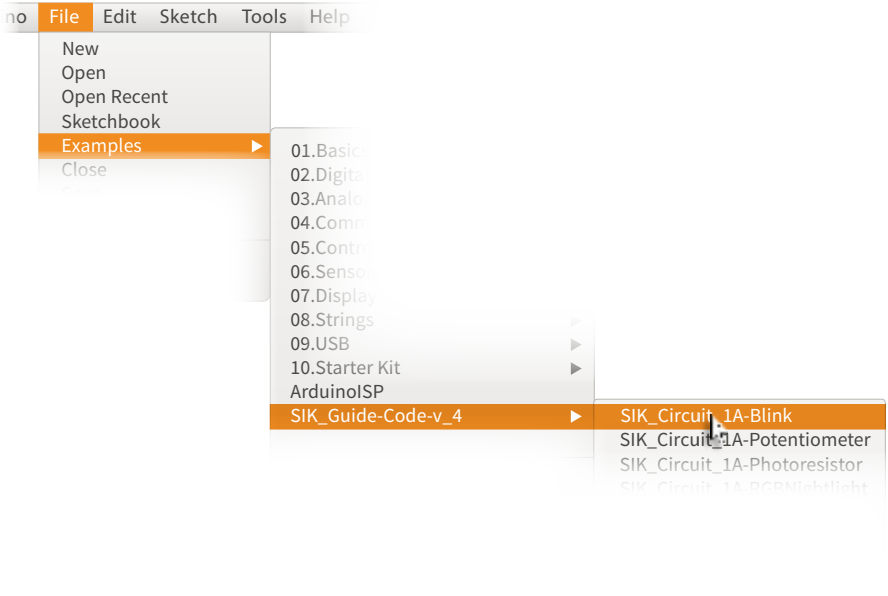
In this table, a yellow highlight indicates that a component has polarity and will only function if properly oriented.

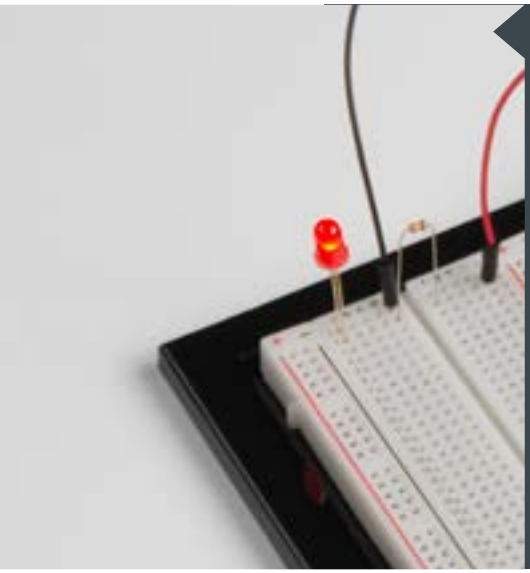
Open the Arduino IDE

Connect the RedBoard to a USB port on your computer.

Open the Sketch: File > Examples > SIK_Guide_Code-V_4 > **CIRCUIT_1A-BLINK**

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



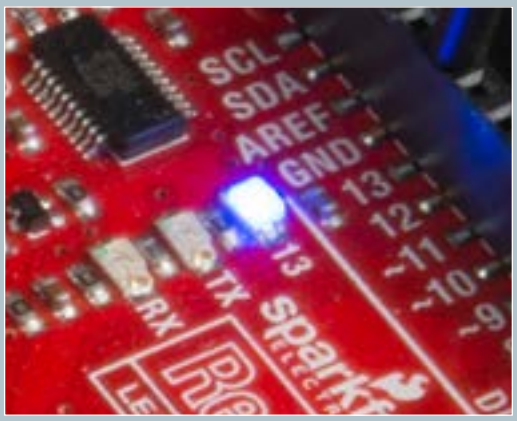


WHAT YOU SHOULD SEE

The LED will flash on for two seconds, then off for two seconds. If it doesn't, make sure you have assembled the circuit correctly and verified and uploaded the code to your board. See the Troubleshooting section at the end of this circuit if that doesn't work. One of the best ways to understand the code you uploaded is to change something and see how it affects the behavior of your circuit. What happens when you change the number in one or both of the `delay(2000);` lines of code (try 100 or 5000)?

PROGRAM OVERVIEW

1	Turn the LED on by sending power (5V) to digital pin 13.
2	Wait 2 seconds (2000 milliseconds).
3	Turn the LED off by cutting power (0V) to digital pin 13.
4	Wait 2 seconds (2000 milliseconds).
5	Repeat.



ONBOARD LED PIN 13: You may have noticed a second, smaller LED blinking in unison with the LED in your breadboard circuit. This is known as the onboard LED, and you can find one on almost any Arduino or Arduino-compatible board. In most cases, this LED is connected to **digital pin 13 (D13)**, the same pin used in this circuit.

NEW IDEAS

CODE TO NOTE: The sketches that accompany each circuit introduce new programming techniques and concepts as you progress through the guide. The Code to Note section highlights specific lines of code from the sketch and explains them in greater detail.

CODE TO NOTE

SETUP AND LOOP:

```
void setup(){} &
void loop(){}
```

Every Arduino program needs these two functions. Code that goes in between the curly brackets `{}` of `setup()` runs once. The code in between the `loop()` curly brackets `{}` runs over and over until the RedBoard is reset or powered off.

INPUT OR OUTPUT?:

```
pinMode(13, OUTPUT);
```

Before you can use one of the digital pins, you need to tell the RedBoard whether it is an `INPUT` or `OUTPUT`. We use a built-in "function" called `pinMode()` to make pin 13 a digital output. You'll learn more about digital inputs in Project 2.

CODE TO NOTE

DIGITAL OUTPUT:

```
digitalWrite(D13, HIGH);
```

When you're using a pin as an `OUTPUT`, you can command it to be HIGH (output 5 volts) or LOW (output 0 volts).

DELAY:

```
delay(2000);
```

Causes the program to wait on this line of code for the amount of time in between the brackets, represented in milliseconds (2000ms = 2s). After the time has passed, the program will continue to the next line of code.

COMMENTS:

```
//This is a comment

/* So is this */
```

Comments are a great way to leave notes in your code explaining why you wrote it the way you did. Single line comments use two forward slashes `//`, while multi-line comments start with a `/*` and end with a `*/`.

NEW IDEAS

CODING CHALLENGES: The Coding Challenges section is where you will find suggestions for changes to the circuit or code that will make the circuit more challenging. If you feel underwhelmed by the tasks in each circuit, visit the Coding Challenges section to push yourself to the next level.

CODING CHALLENGES

PERSISTENCE OF VISION: Computer screens, movies and the lights in your house all flicker so quickly that they appear to be on all of the time but are actually blinking faster than the human eye can detect. See how much you can decrease the delay time in your program before the light appears to be on all the time but is still blinking.

MORSE CODE: Try adding and changing the `delay()` values and adding more `digitalWrite()` commands to make your program blink a message in Morse code.

TROUBLESHOOTING

I get an error when uploading my code

The most likely cause is that you have the wrong board selected in the Arduino IDE. Make sure you have selected **Tools > Board > Arduino/Genuino Uno.**

TROUBLESHOOTING

I still get an error when uploading my code	If you're sure you have the correct Board selected but you still can't upload, check that you have selected the correct serial port. You can change this in **Tools > Serial Port > your_serial_port**.
Which serial port is the right one?	Depending on how many devices you have plugged into your computer, you may have several active serial ports. Make sure you are selecting the correct one. A simple way to determine this is to look at your list of serial ports. Unplug your RedBoard from your computer. Look at the list again. Whichever serial port has disappeared from the list is the one you want to select once you plug your board back into your computer.
My code uploads, but my LED won't turn on	LEDs will only work in one direction. Try taking it out of your breadboard, turning it 180 degrees and reinserting it.
Still not working?	Jumper wires unfortunately can go "bad" from getting bent too much. The copper wire inside can break, leaving an open connection in your circuit. If you are certain that your circuit is wired correctly and that your code is error-free and uploaded, but you are still encountering issues, try replacing one or more of the jumper wires for the component that is not working.

You've completed Circuit 1A!

Continue to circuit 1B to learn about analog signals and potentiometers.