Chipsmall Limited consists of a professional team with an average of over 10 year of expertise in the distribution of electronic components. Based in Hongkong, we have already established firm and mutual-benefit business relationships with customers from,Europe,America and south Asia,supplying obsolete and hard-to-find components to meet their specific needs.

With the principle of “Quality Parts,Customers Priority,Honest Operation,and Considerate Service”,our business mainly focus on the distribution of electronic components. Line cards we deal with include Microchip,ALPS,ROHM,Xilinx,Pulse,ON,Everlight and Freescale. Main products comprise IC,Modules,Potentiometer,IC Socket,Relay,Connector.Our parts cover such applications as commercial,industrial, and automotives areas.

We are looking forward to setting up business relationship with you and hope to provide you with the best service and solution. Let us make a better world for our industry!


## Contact us

Tel: +86-755-8981 8866 Fax: +86-755-8427 6832
Email & Skype: info@chipsmall.com Web: www.chipsmall.com
Address: A1208, Overseas Decoration Building, #122 Zhenhua RD., Futian, Shenzhen, China

# SparkFun Inventor's Kit

VERSION 4.0

Your Guide to the SIK
for the SparkFun RedBoard

sparkfun
START SOMETHING

# SparkFun Inventor's Kit, Version 4.0

## WELCOME TO THE SPARKFUN INVENTOR'S KIT (SIK) GUIDE.

This is your map for navigating beginning embedded electronics. This booklet contains all the information you will need to build five projects encompassing the 16 circuits of the SIK for the SparkFun RedBoard. At the center of this manual is one core philosophy: that anyone can (and should) play around with electronics. When you're done with this guide, you will have built five great projects and acquired the know-how to create countless more. Now enough talk — let's start something!

For a digital version of this guide with more in-depth information for each circuit and links explaining relevant terms and concepts, visit:

www.sparkfun.com/SIKguide

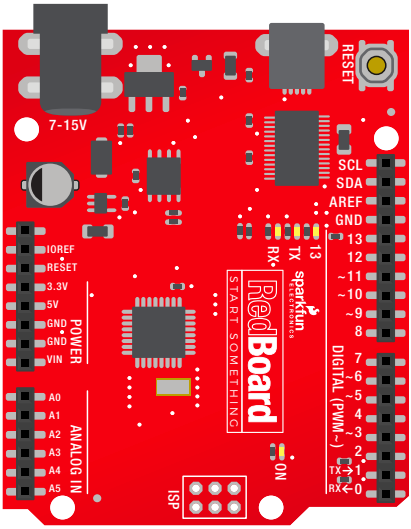
# Contents

# The RedBoard Platform

**THE DIY REVOLUTION:** At SparkFun we believe that an understanding of electronics is a core literacy that opens up a world of opportunities in the fields of robotics, Internet of Things (IoT), engineering, fashion, medical industries, environmental sciences, performing arts and more. This guide is designed to explore the connection between software and hardware, introducing Arduino code and SparkFun parts as they are used in the context of building engaging projects. The circuits in this guide progress in difficulty as new concepts and components are introduced. Completing each circuit means much more than just "experimenting"; you will walk away with a fun project you can use — and a sense of accomplishment that is just the beginning of your electronics journey. At the end of each circuit, you'll find coding challenges that extend your learning and fuel ongoing innovation.

## A COMPUTER FOR THE PHYSICAL WORLD

The SparkFun RedBoard is your development platform. At its roots, the RedBoard is essentially a small, portable computer, also known as a microcontroller. It is capable of taking inputs (such as the push of a button or a reading from a light sensor) and interpreting that information to control various outputs (like blinking an LED light or spinning an electric motor). That's where the term "physical computing" comes in; this board is capable of taking the world of electronics and relating it to the physical world in a real and tangible way.




**THE SPARKFUN REDBOARD** is one of a multitude of development boards based on the ATmega328 microprocessor. It has 14 digital input/output pins (six of which can be PWM outputs), six analog inputs, a 16MHz crystal oscillator, a USB connection, a power jack, and a reset button. You'll learn more about each of the RedBoard's features as you progress through this guide.

# Baseplate Assembly

Before you can build circuits, you'll want to first assemble the breadboard baseplate. This apparatus makes circuit building easier by keeping the RedBoard microcontroller and the breadboard connected without the worry of disconnecting or damaging your circuit.

**TO BEGIN,** collect your parts: the RedBoard, breadboard, included screwdriver, baseplate and two baseplate screws.

Your screwdriver has both Phillips and flatheads. If it is not already in position, pull the shaft out and switch to the Phillips head.

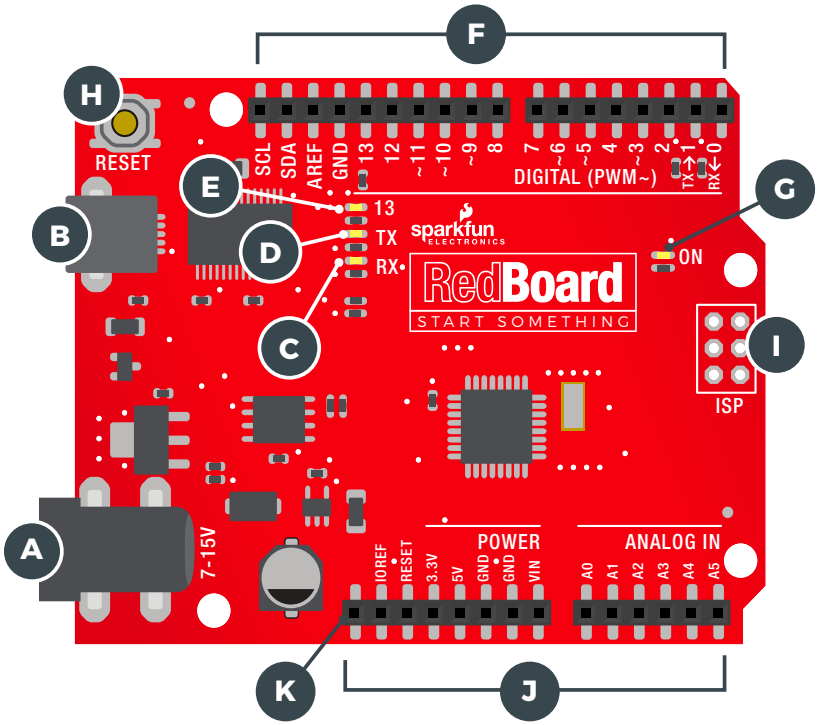
**PEEL** the adhesive backing off the breadboard.

**CAREFULLY ALIGN** the breadboard over its spot on the baseplate. The text on the breadboard should face the same direction as the text on the baseplate. Firmly press the breadboard to the baseplate to adhere it.

**ALIGN THE REDBOARD** with its spot on the baseplate. The text on it should face the same direction as the text on the breadboard and the baseplate. Using one of the two included screws, affix the RedBoard to one of the four stand-off holes found on the baseplate. The plastic holes are not threaded, so you will need to apply pressure as you twist the screwdriver.

Screw the second screw in the stand-off hole diagonally across from the first. With that, your baseplate is now assembled.

# Anatomy of the SparkFun RedBoard




## REDBOARD HARDWARE OVERVIEW

| | | |
|---|---|---|
| **A** | **POWER IN (BARREL JACK)** | Can be used with either a 9V or 12V "wall-wart" or a battery pack. |
| **B** | **POWER IN (USB PORT)** | Provides power and communicates with your board when plugged into your computer via USB. |
| **C** | **LED (RX: RECEIVING)** | Shows when the FTDI chip is receiving data bits from the computer. |
| **D** | **LED (TX: TRANSMITTING)** | Shows when the FTDI chip is transmitting data bits to the computer. |
| **E** | **ONBOARD LED PIN D13** | This LED, connected to digital pin 13, can be controlled in your program and is great for troubleshooting. |
| **F** | **PINS AREF, GROUND, DIGITAL, RX, TX, SDA, SCL** | These pins can be used for inputs, outputs, power and ground. |
| **G** | **POWER LED** | Illuminated when the board is connected to a power source. |
| **H** | **RESET BUTTON** | A manual reset switch that will restart the RedBoard and your code. |
| **I** | **ISP HEADER** | This is the In-System Programming header. It is used to program the ATMega328 directly. It will not be used in this guide. |
| **J** | **ANALOG IN, VOLTAGE IN, GROUND, 3.3 AND 5V OUT, RESET** | The power bus has pins to power your circuits with various voltages. The analog inputs allow you to read analog signals. |
| **K** | **RFU** | This stands for Reserved for Future Use. |

# Anatomy of the Breadboard

A breadboard is a circuit-building platform that allows you to connect multiple components without using a soldering iron.

### POWER BUS

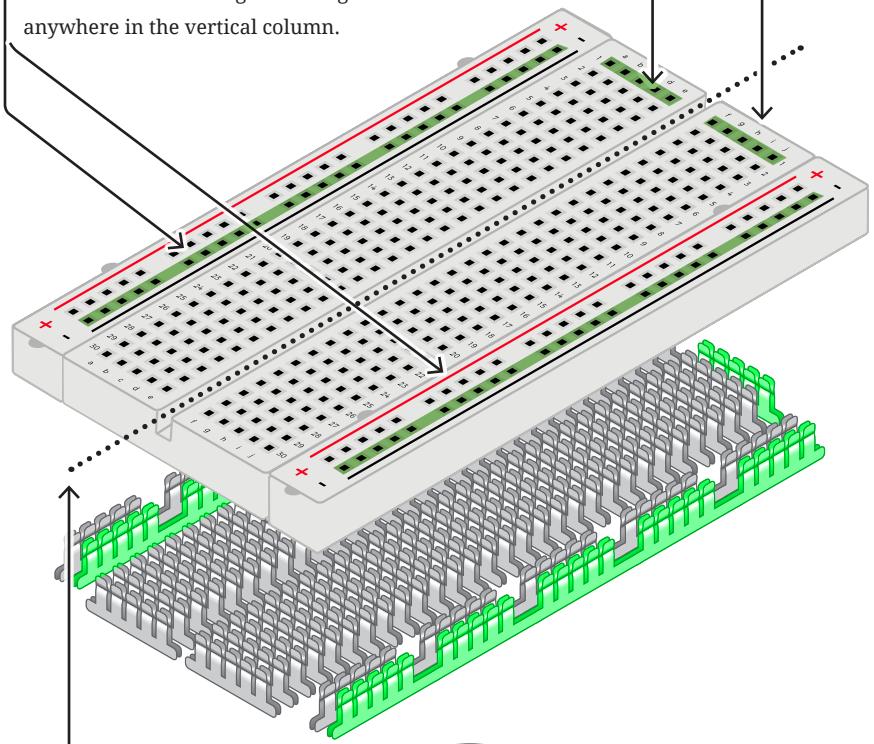
Each side of the breadboard has a pair of vertical connections marked – and +

**+ POWER:** Each + sign runs power anywhere in the vertical column.

**– GROUND:** Each – sign runs to ground anywhere in the vertical column.

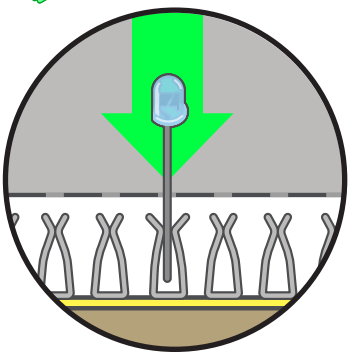
### HORIZONTAL ROWS

Each series of 5 sockets marked a–e and f–j are connected. Components connected to a row will be connected to any other part inserted in the same row.



### CENTERLINE

This line divides the breadboard in half, restricting electricity to one half or the other.



### MAKING A CONNECTION

Most of the components in this kit are breadboard-friendly and can be easily installed and removed.

# The Arduino IDE


**IN ORDER TO GET YOUR REDBOARD UP AND RUNNING**, you'll need to download the newest version of the Arduino software from **www.arduino.cc** (it's free!). This software, known as the Arduino IDE (Integrated Development Environment), will allow you to program the RedBoard to do exactly what you want. It's like a word processor for coding. With an internet-capable computer, open up your favorite browser and type the following URL into the address bar:

**DOWNLOAD THE SOFTWARE HERE:** arduino.cc/downloads

## 1. DOWNLOAD AND INSTALL ARDUINO IDE

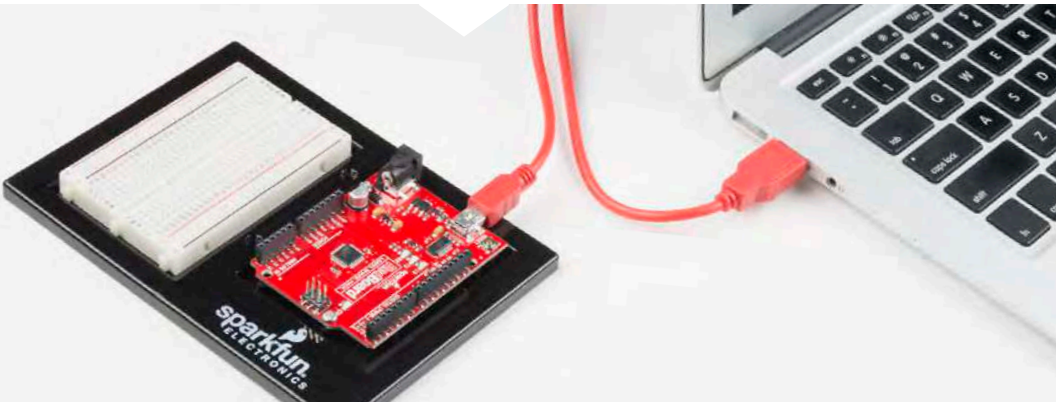
Select the installer option appropriate for the operating system you are using. Once finished downloading, open the file and follow the instructions to install.

## 2. INSTALL USB DRIVERS

In order for the RedBoard hardware to work with your computer's operating system, you will need to install a few drivers. Please go to **www.sparkfun.com/FTDI** for specific instructions on how to install the USB drivers onto your computer.

## 3. CONNECT THE REDBOARD TO A COMPUTER

Use the USB cable provided in the SIK to connect the RedBoard to one of your computer's USB inputs.



# 4. DOWNLOAD AND INSTALL THE SIK CODE

Each of the circuits you will build in the SparkFun Inventor's Kit has an Arduino code sketch already written for it. This guide will show you how to manipulate that code to control your hardware.

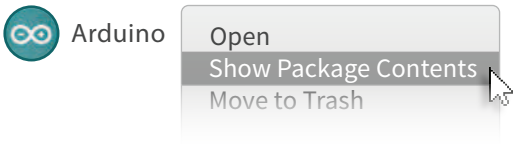
**DOWNLOAD THE CODE HERE:** sparkfun.com/SIKcode

## COPY "SIK GUIDE CODE" INTO "EXAMPLES" LIBRARY IN ARDUINO FOLDER

Your browser will download the code automatically or ask you if you would like to download the .zip file. Select "Save File." Locate the code (usually in your browser's "Downloads" folder). You'll need to relocate it to the "Examples" subfolder in your Arduino IDE installation in order for it to function properly.

Unzip the file **"SIK GUIDE CODE."** It should be located in your browser's "Downloads" folder. Right-click (or ctrl + click) the zipped folder and choose **"unzip."**

**MAC OS:** Find "Arduino" in your "Applications" folder in Finder. Right-click (ctrl + click) on "Arduino" and select "Show Package Contents."




Copy or move the unzipped "SIK Guide Code" folder from your "Downloads" folder into the Arduino application's folder named "Examples."

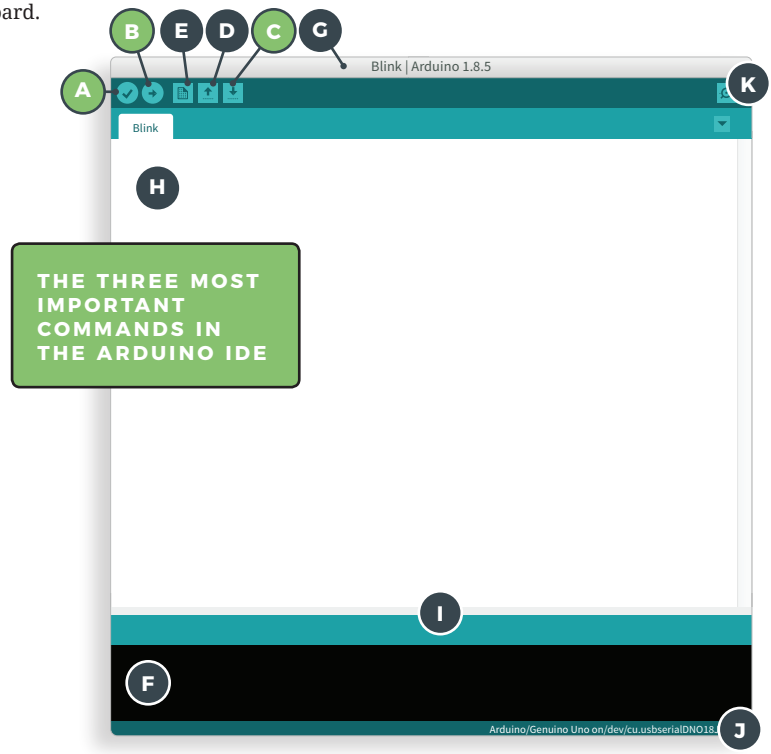

**WINDOWS:** Copy or move the unzipped "SIK Guide Code" files from "Downloads" to the Arduino application's "Examples" folder.


**LINUX:** Distribution-specific setup instructions for Linux can be found at:

http://arduino.cc/playground/learning/linux

## 5. OPEN THE ARDUINO IDE:

Open the Arduino IDE software on your computer. Poke around and get to know the interface. We aren't going to code right away; this step is to set your IDE to identify your RedBoard.

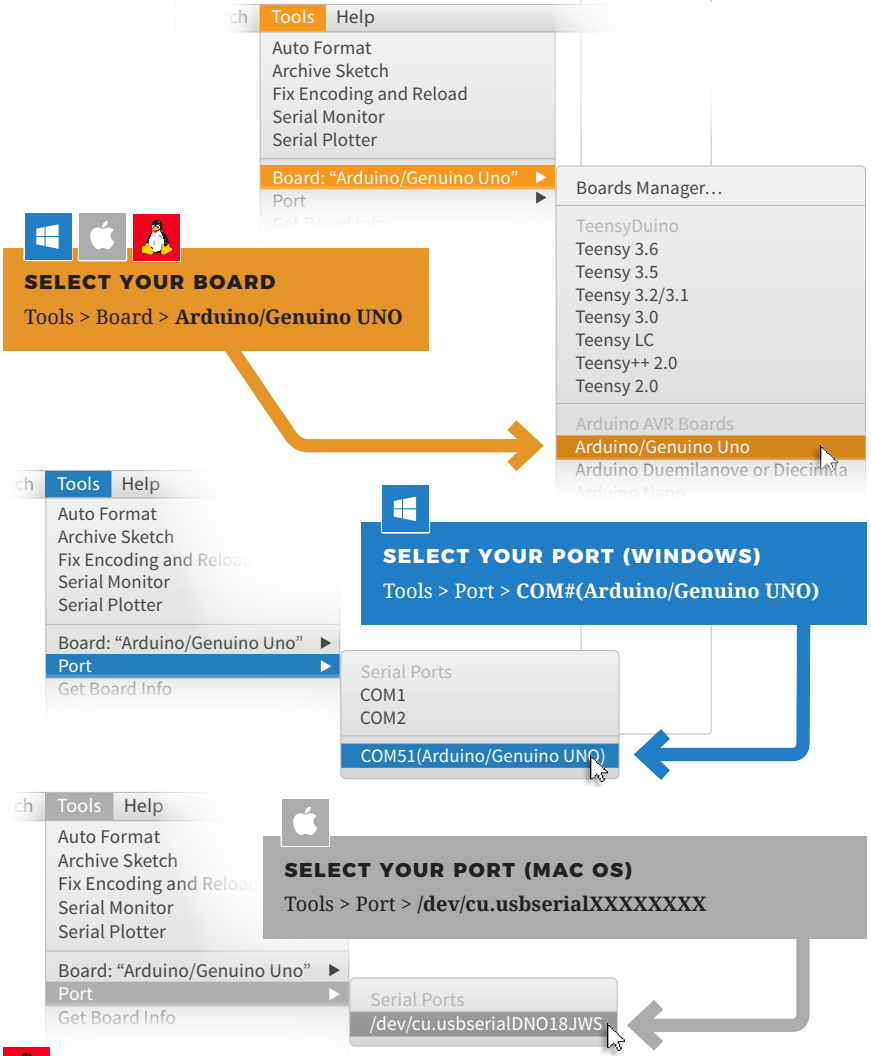



## GRAPHICAL USER INTERFACE (GUI)

| | | |
|---|---|---|
| A | VERIFY | Compiles and approves your code. It will catch errors in syntax (like missing semicolons or parentheses). |
| B | UPLOAD | Sends your code to the RedBoard. When you click it, you should see the lights on your board blink rapidly. |
| C | SAVE | Saves the currently active sketch. |
| D | OPEN | Opens an existing sketch. |
| E | NEW | Opens up a new code window tab. |
| F | DEBUG WINDOW | Displays any errors generated by your sketch. |
| G | SKETCH NAME | Displays the name of the sketch you are currently working on. |
| H | CODE AREA | Where you compose or edit the code for your sketch. |
| I | MESSAGE AREA | Indicates if the code is compiling, uploading or has errors. |
| J | CONNECTION AREA | Displays the board and serial port currently selected. |
| K | SERIAL MONITOR | Opens a window that displays any serial information your RedBoard is transmitting (useful for debugging). |

# 6. SELECT YOUR BOARD AND SERIAL DEVICE

**NOTE:** Your SparkFun RedBoard and the Arduino/Genuino UNO are interchangeable, but you won't find the RedBoard listed in the Arduino software. Select **"ARDUINO/GENUINO UNO"** instead.




**SELECT YOUR PORT (LINUX)**

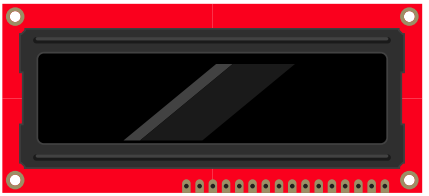
Distribution-specific serial device setup instructions can be found **HERE:**

http://arduino.cc/playground/learning/linux

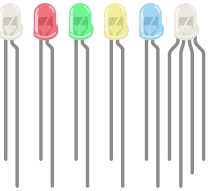
# Inventory of Parts

The SparkFun Inventor's Kit contains an extensive array of electronic components. As you work your way through each circuit, you will learn to use new and more complicated parts to accomplish increasingly complex tasks.

LCD DISPLAY



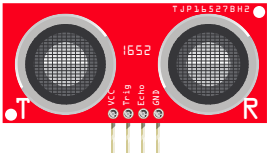
LEDS



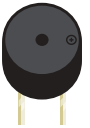
POTENTIOMETER

SWITCH

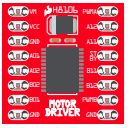
ULTRASONIC DISTANCE SENSOR



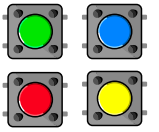

PIEZO BUZZER



MOTOR DRIVER




PUSH BUTTONS



10KΩ RESISTORS



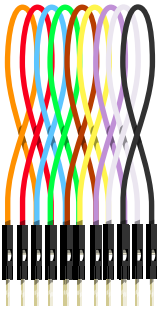
330Ω RESISTORS



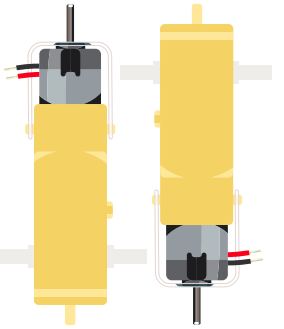
PHOTORESISTOR

TEMPERATURE SENSOR

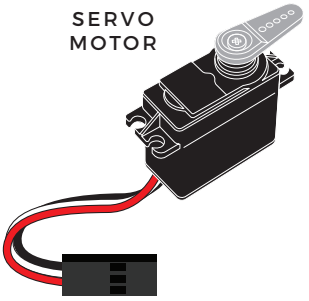

JUMPER WIRES



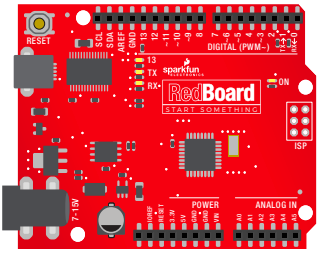
GEARMOTORS



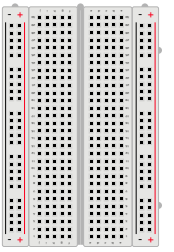
SERVO MOTOR

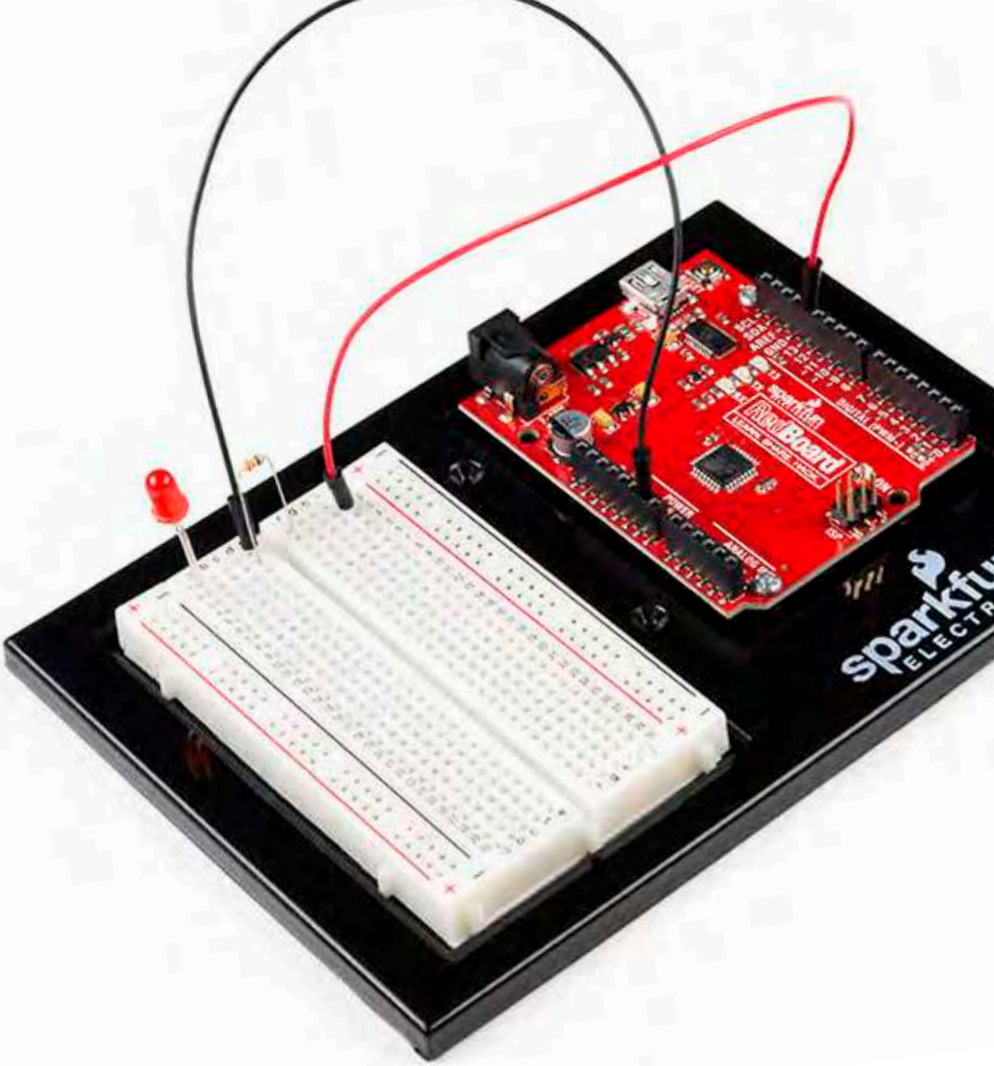


SPARKFUN REDBOARD

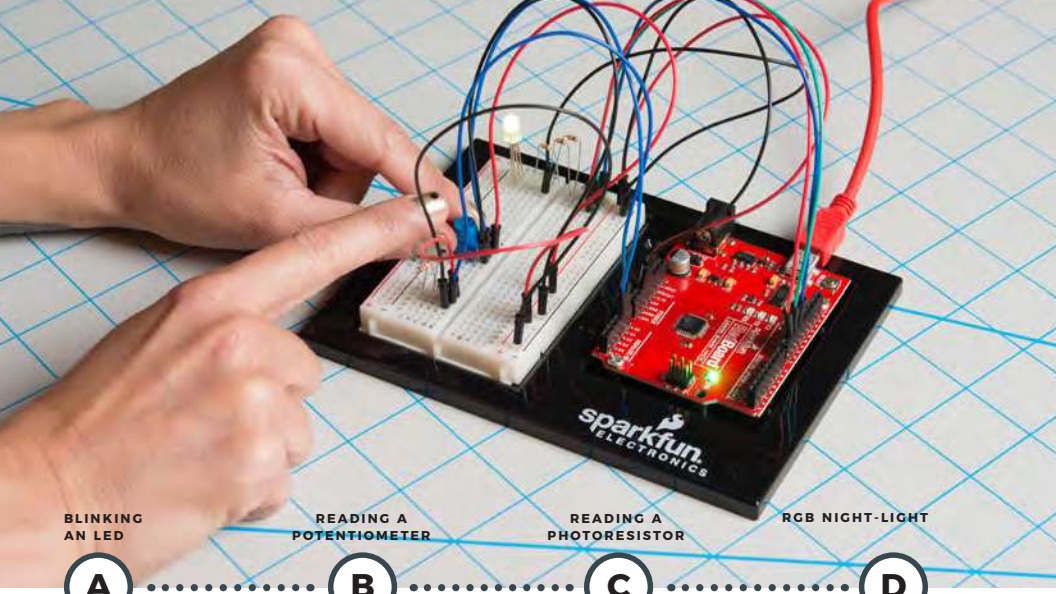



BREADBOARD






# Let's Get Started With Your First Circuit!




# PROJECT 1

Welcome to your first SparkFun Inventor's Kit project. Each **project** is broken up into several **circuits,** the last circuit being a culmination of the technologies that came before. There are five projects total, each designed to help you learn about new technologies and concepts. This first project will set the foundation for the rest and will aid in helping you understand the fundamentals of circuit building and electricity!

In Project 1, you will learn about **Light-Emitting Diodes (LEDs)**, **resistors**, **inputs**, **outputs** and **sensors**. The first project will be to build and program your own multicolored night-light! The night-light uses a sensor to turn on an RGB (Red, Green, Blue) LED when it gets dark, and you will be able to change the color using an input knob.

## NEW IDEAS

Each project will introduce new concepts and components, which will be described in more detail as you progress through the circuits.

**NEW COMPONENTS INTRODUCED IN THIS PROJECT**

- LEDS
- RESISTORS
- POTENTIOMETERS
- PHOTORESISTORS

**NEW CONCEPTS INTRODUCED IN THIS PROJECT**

- POLARITY
- OHM'S LAW
- DIGITAL OUTPUT
- ANALOG VS. DIGITAL
- ANALOG INPUT
- ANALOG TO DIGITAL CONVERSION
- VOLTAGE DIVIDER
- PULSE-WIDTH MODULATION
- FUNCTIONS

**YOU WILL LEARN**

- HOW TO UPLOAD A PROGRAM TO YOUR REDBOARD
- CIRCUIT-BUILDING BASICS
- HOW TO CONTROL LEDS WITH DIGITAL OUTPUTS
- HOW TO READ SENSORS WITH ANALOG INPUTS

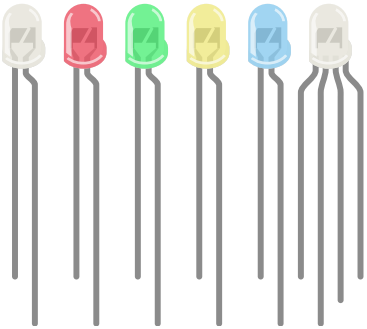
# Circuit 1A: Blinking an LED

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:

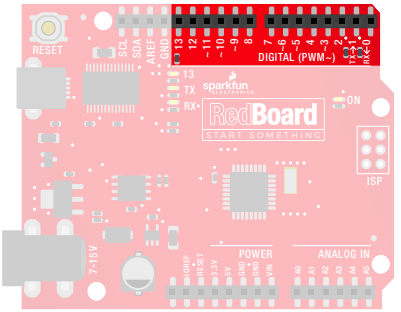
$$V=I \bullet R$$

**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.

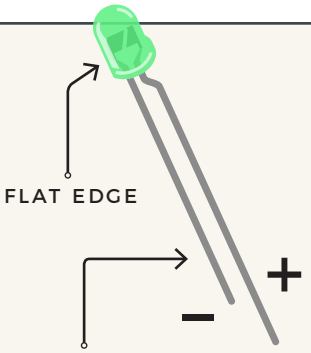



**DIGITAL OUTPUT:** When working with microcontrollers such as the RedBoard, there are a variety of pins to which you can connect electronic components. Knowing which pins perform which functions is important when building your circuit. In this circuit, we will be using what is known as a digital output. There are 14 of these pins found on the RedBoard. A digital output only has **two states: ON or OFF**. These two states can also be thought of as **HIGH or LOW, TRUE or FALSE**. When an LED is connected to one of these pins, the pin can only perform two jobs: turning on the LED and turning off the LED. We'll explore the other pins and their functions in later circuits.

## 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.

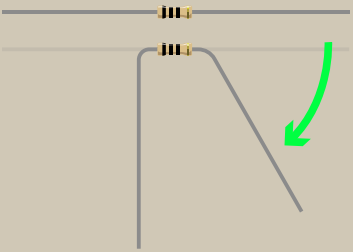



### POLARIZED COMPONENTS

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

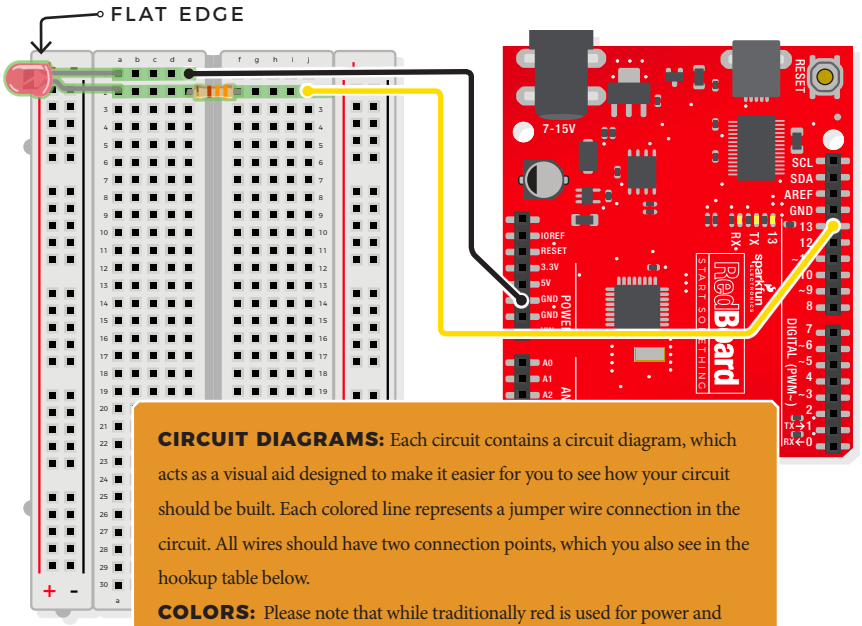
### RESISTOR LEADS

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** |

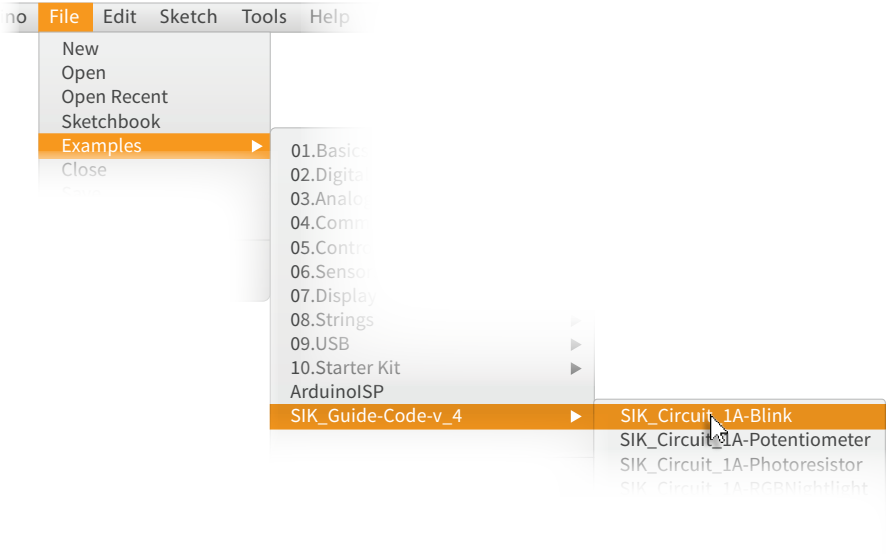
In this table, a yellow highlight indicates that a component has polarity and will only function if properly oriented.

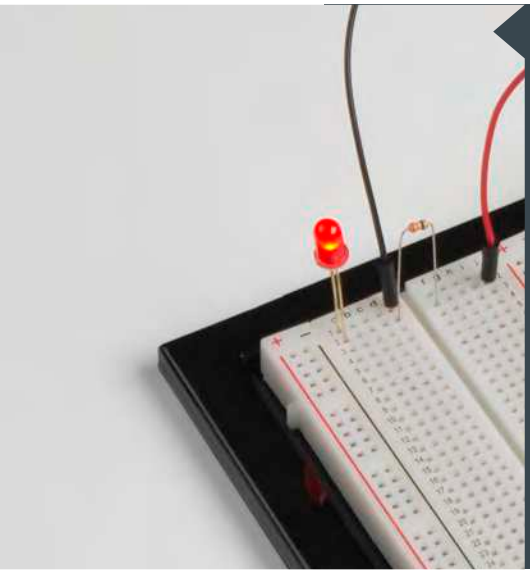
## Open the Arduino IDE

**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.



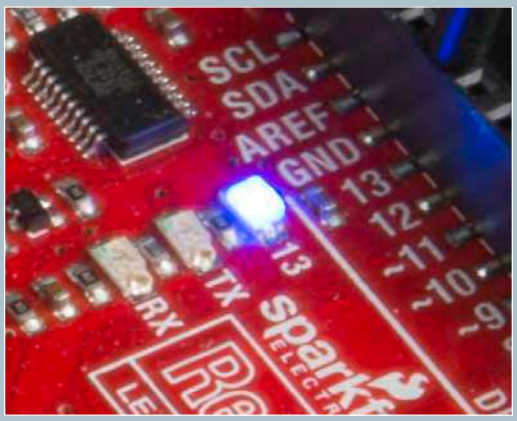



## WHAT YOU SHOULD SEE

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.

# Circuit 1B: Potentiometer

Potentiometers (also known as “trimpots” or “knobs”) are one of the basic inputs for electronic devices. By tracking the position of the knob with your RedBoard, you can make volume controls, speed controls, angle sensors and a ton of other useful inputs for your projects. In this circuit, you’ll use a potentiometer as an input device to control the speed at which your LED blinks.

**YOU NEED**


LED


POTENTIOMETER


330Ω RESISTOR


7 JUMPER WIRES

## NEW COMPONENTS

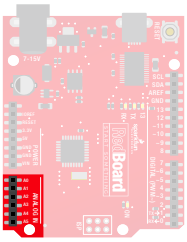
**POTENTIOMETER:** A potentiometer is a 3-pin variable resistor. When powered with 5V, the middle pin outputs a voltage between 0V and 5V, depending on the position of the knob on the potentiometer. Internal to the trimpot is a single resistor and a wiper, which cuts the resistor in two and moves to adjust the ratio between both halves.

## NEW CONCEPTS

**ANALOG VS. DIGITAL:** We live in an analog world. There are an infinite number of colors to paint an object, an infinite number of tones we can hear, and an infinite number of smells we can smell. The common theme among these analog signals is their infinite possibilities.

Digital signals deal in the realm of the discrete or finite, meaning there is a limited set of values they can be. The LED from the previous circuit had only two states it could exist in, ON or OFF, when connected to a digital output.

**ANALOG INPUTS:** So far, we’ve only dealt with outputs. The RedBoard also has inputs. Both inputs and outputs can be analog or digital. Based on our previous definition of analog and digital, that means an analog input can sense a wide range of values versus a digital input, which can only sense two values, or states.



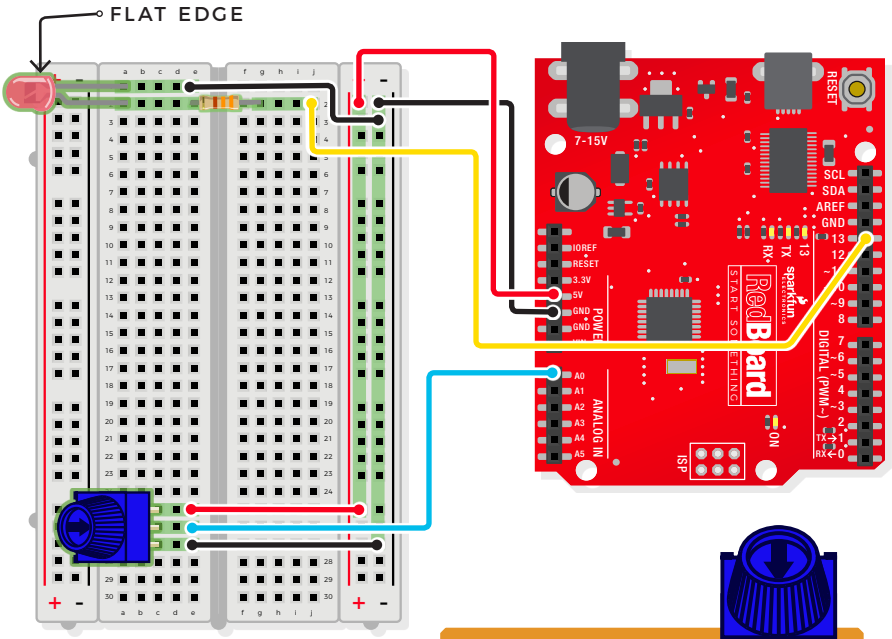
You may have noticed some pins labeled **Digital** and some labeled **Analog In** on your RedBoard. There are only six pins that function as analog inputs; they are labeled A0–A5.

## VOLTAGE DIVIDER

**VOLTAGE DIVIDERS** are simple circuits that turn some voltage into a smaller voltage using two resistors. A potentiometer is a variable resistor that can be used to create an adjustable voltage divider. A wiper in the middle position means the output voltage will be half of the input. Voltage dividers will be covered in more detail in the next circuit.

## 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

**POTENTIOMETERS** are not polarized and can be installed in either direction. Note that swapping the 5V and GND pins will reverse its behavior.

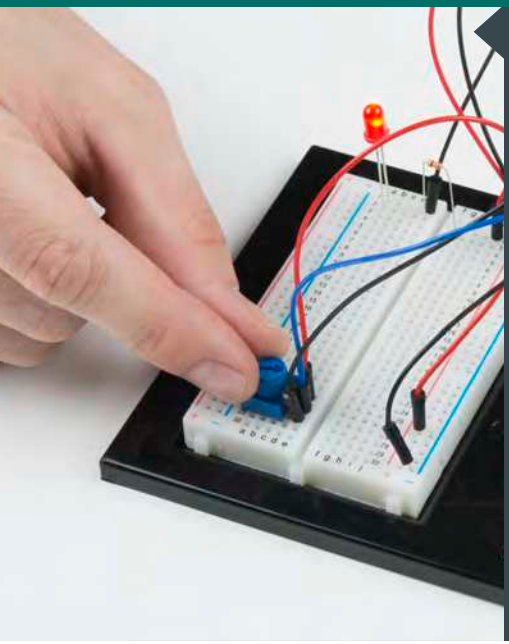
| CONNECTION TYPES | ◆ REDBOARD CONNECTION ■ BREADBOARD CONNECTION |
|---|---|
| JUMPER WIRES | ◆ 5V to ■ 5V<br>◆ GND to ■ GND (-)<br>◆ A0 to ■ E26<br>■ E25 to ■ 5V (+)<br>■ E27 to ■ GND (-)<br>■ E1 to ■ GND (-)<br>◆ D13 to ■ J2 |
| LED | ■ A1(-) to ■ A2(+) |
| 330Ω RESISTOR (ORANGE, ORANGE, BROWN) | ■ E2 to ■ F2 |
| POTENTIOMETER | ■ C25 + ■ C26 + ■ C27 |

## Open the Arduino IDE

**Connect the RedBoard to a USB port on your computer.**

**Open the Sketch:**
File > File > Examples > SIK_Guide_Code-V_4 > **CIRCUIT_1B-POTENTIOMETER**

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



## WHAT YOU SHOULD SEE

You should see the LED blink faster or slower in accordance with your potentiometer. The delay between each flash will change based on the position of the knob. If it isn't working, make sure you have assembled the circuit correctly and verified and uploaded the code to your board. If that doesn't work, see the Troubleshooting section.

## PROGRAM OVERVIEW

| | |
|---|---|
| 1 | Read the position of the potentiometer (from 0 to 1023) and store it in the variable **`potPosition`**. |
| 2 | Turn the LED on. |
| 3 | Wait from 0 to 1023 milliseconds, based on the position of the knob and the value of **`potPosition`**. |
| 4 | Turn the LED off. |
| 5 | Wait from 0 to 1023 milliseconds, based on the position of the knob and the value of **`potPosition`**. |
| 6 | Repeat. |

## ARDUINO PRO TIP

**ARDUINO SERIAL MONITOR:** The Serial Monitor is one of the Arduino IDE's many great included features. When working with embedded systems, it helps to see and understand the values that your program is trying to work with, and it can be a powerful debugging tool when you run into issues where your code is not behaving the way you expected it to. This circuit introduces you to the Serial Monitor by showing you how to print the values from your potentiometer to it. To see these values, click the Serial Monitor button, found in the upper-right corner of the IDE in most recent versions. You can also select **Tools** > **Serial Monitor** from the menu.


*Serial Monitor button in the upper-right of the Arduino IDE.*


*Serial Monitor printout and baud-rate menu.*

You should see numeric values print out in the monitor. Turning the potentiometer changes the value as well as the delay between each print.

If you are having trouble seeing the values, ensure that you have selected 9600 baud and have auto scroll checked.

## CODE TO NOTE

**INTEGER VARIABLES:**

```
int potPosition;
```

A variable is a placeholder for values that may change in your code. You must introduce, or "declare," variables before you use them. Here we're declaring a variable called **potPosition** of type **int** (integer). We will cover more types of variables in later circuits. Don't forget that variable names are case-sensitive!