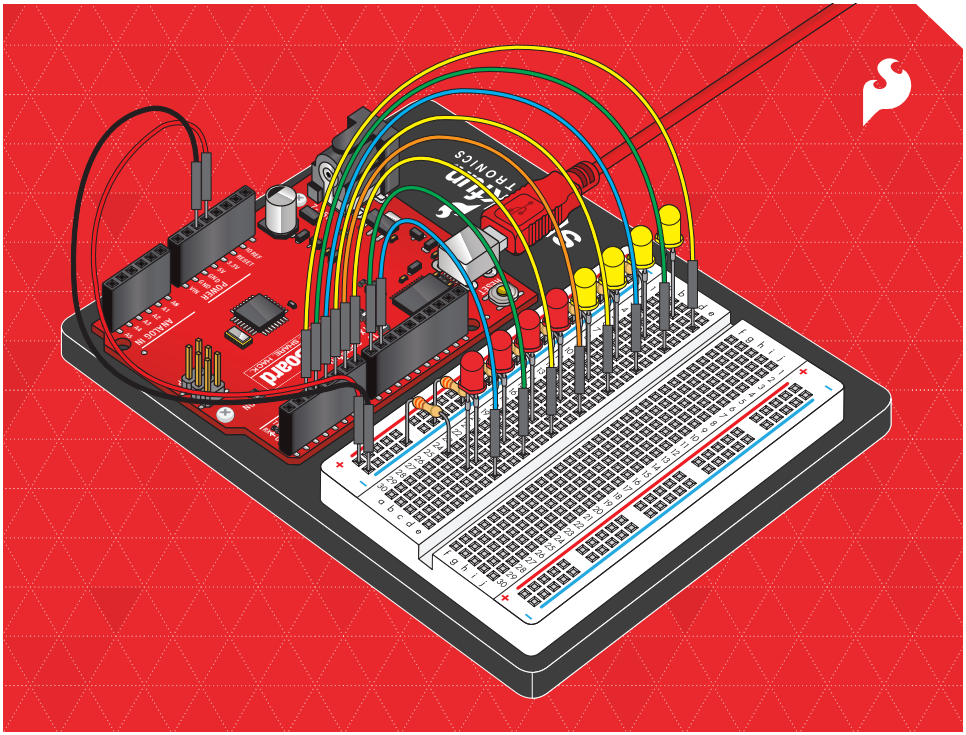


SIK GUIDE

Your guide to the SparkFun Inventor's Kit for the SparkFun RedBoard

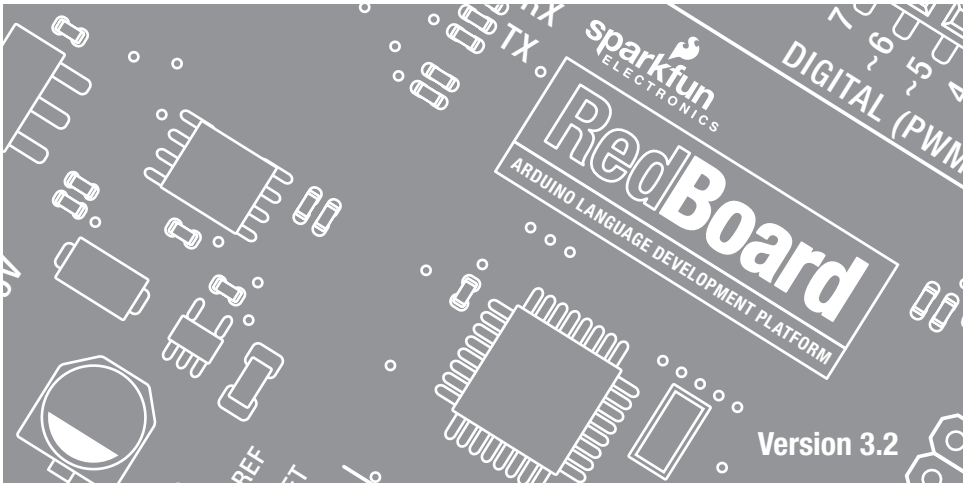


Table of Contents

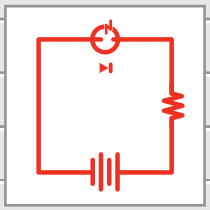
Welcome to the SparkFun Inventor's Guide

The SparkFun Inventor's Guide is your map for navigating the waters of beginning embedded electronics. This booklet contains all the information you will need to explore the 16 circuits of the SparkFun Inventor's Kit for Educators. 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'll have the know-how to start creating your own projects and experiments. Now enough talking - let's get inventing!

www.sparkfun.com

Section 1: *Getting Started*

Section 2: *Getting Started with Circuits*



What is the RedBoard platform?

The DIY Revolution

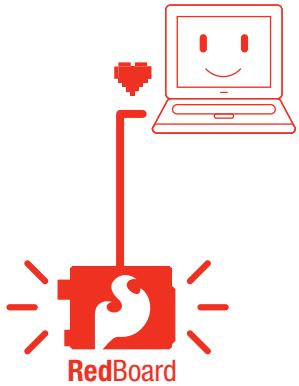
We live in a unique time where we have access to resources that allow us to create our own solutions and inventions. The DIY revolution is composed of hobbyists, tinkerers and inventors who would rather craft their own projects than let someone do it for them.

www.sparkfun.com

A Computer for the Physical World

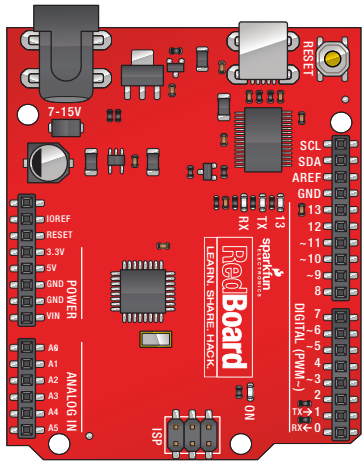
The RedBoard in your hand (or on your desk) is your development platform. At its roots, the RedBoard is essentially a small portable computer. It is capable of taking **inputs** (such as the push of a button or a reading from a light sensor) and interpreting that information to control various **outputs** (like a blinking LED light or an electric motor).

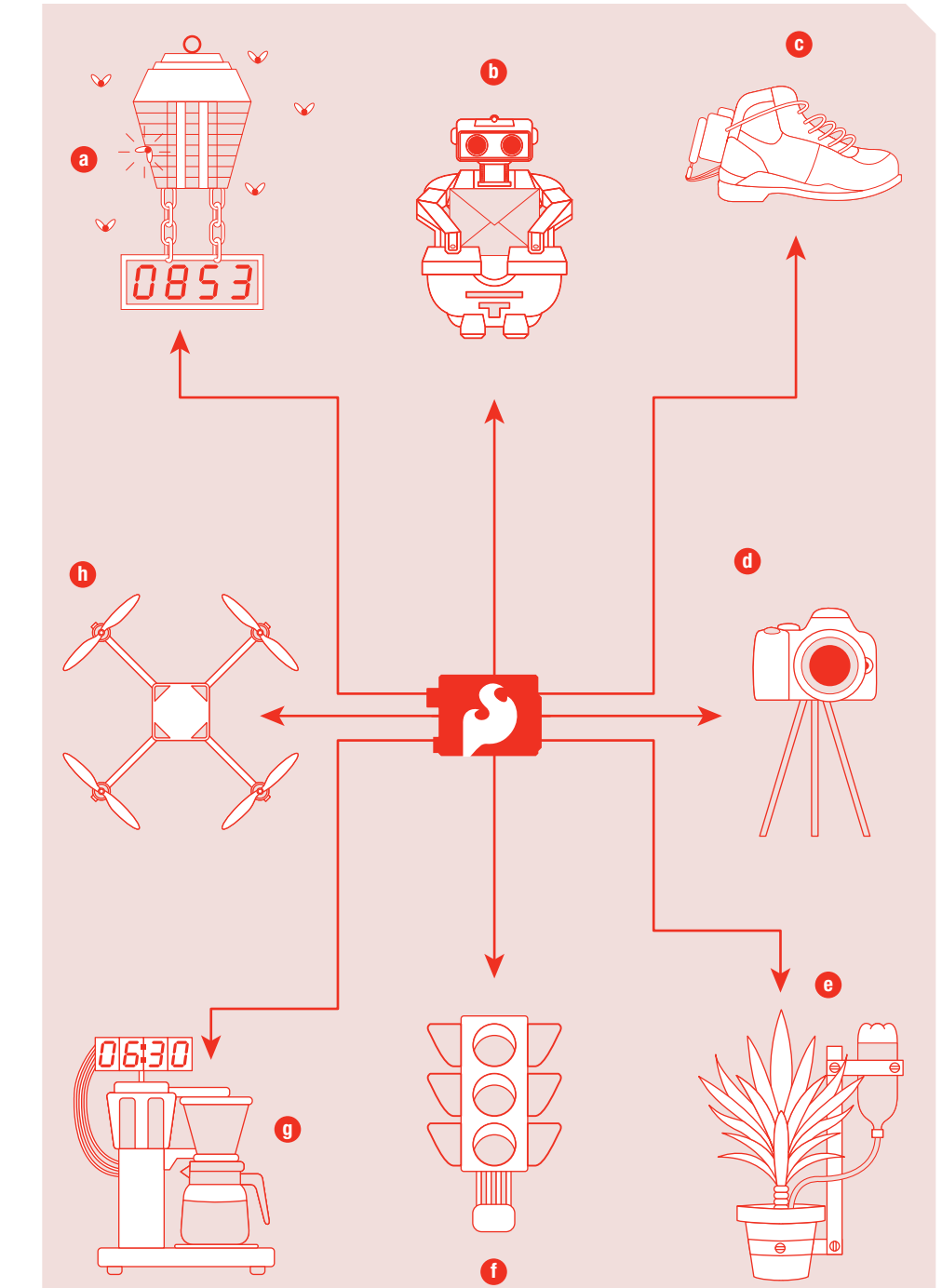
That's where the term "physical computing" is born - this board is capable of taking the world of electronics and relating it to the physical world in a real and tangible way. Trust us - this will all make more sense soon.



// SparkFun RedBoard

The SparkFun RedBoard is one of a multitude of development boards based on the ATmega328. It has 14 digital input/output pins (6 of which can be PWM outputs), 6 analog inputs, a 16 MHz crystal oscillator, a USB connection, a power jack, an ISP header, and a reset button. Don't worry, you'll learn about all these later.





a Bug Zapper Counter
b Old Toy Email Notifer
c Power-Lacing High Tops
d Camera Time-lapse operation
e Auto-Plant Watering
f Re-Programmed Traffic Light
g Auto-Coffee Maker
h Quadcopter

Download the Arduino IDE (Integrated Development Environment)

Access the Internet

In order to get your RedBoard up and running, you'll need to download the newest version of the Arduino software first from www.arduino.cc (it's free!). This software, known as the Arduino IDE, will allow you to program the board to do exactly what you want. It's like a word processor for writing programs. With an internet-capable computer, open up your favorite browser and type in the following URL into the address bar:

arduino.cc/en/main/software

1

Download

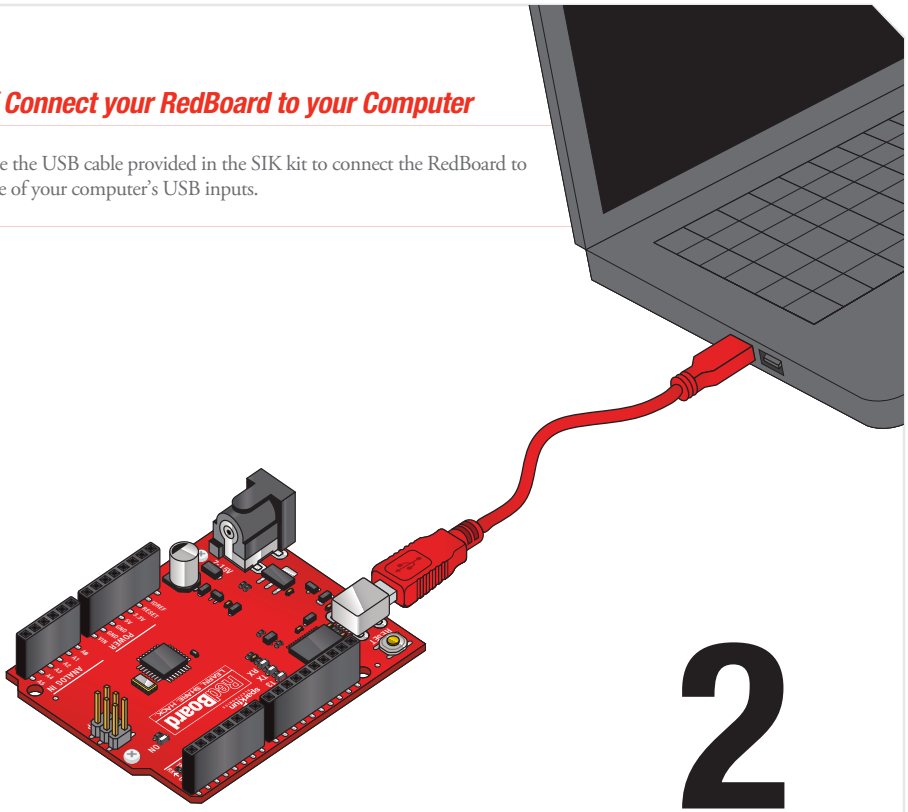
Click on your appropriate computer operating system next to the " + " sign.

- + Windows
- + Mac OS X
- + Linux: 32 bit, 64 bit
- + source

Choose the appropriate Operating System installation package for your computer.

// Connect your RedBoard to your Computer

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



2

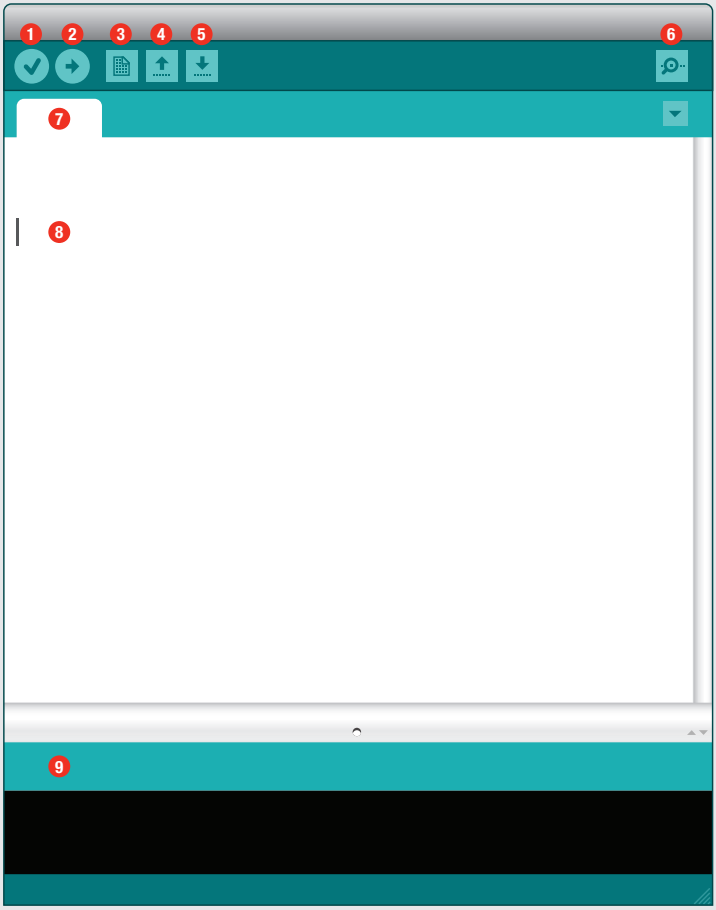
3

// Install Arduino Drivers

Depending on your computer's operating system, you will need to follow specific instructions. Please go to **www.sparkfun.com/FTDI** for specific instructions on how to install the FTDI drivers onto your RedBoard.

// 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 is just an introduction. This step is to set your IDE to identify your RedBoard.



GUI (Graphical User Interface)

1. **Verify:** Compiles and approves your code. It will catch errors in syntax (like missing semi-colons or parenthesis). // See Diagram Below
2. **Upload:** Sends your code to the RedBoard. When you click it, you should see the lights on your board blink rapidly. // See Diagram Below
3. **New:** This buttons opens up a new code window tab.
4. **Open:** This button will let you open up an existing sketch. // See Diagram Below
5. **Save:** This saves the currently active sketch.
6. **Serial Monitor:** This will open a window that displays any serial information your RedBoard is transmitting. It is very useful for debugging.
7. **Sketch Name:** This shows the name of the sketch you are currently working on.
8. **Code Area:** This is the area where you compose the code for your sketch.
9. **Message Area:** This is where the IDE tells you if there were any errors in your code.

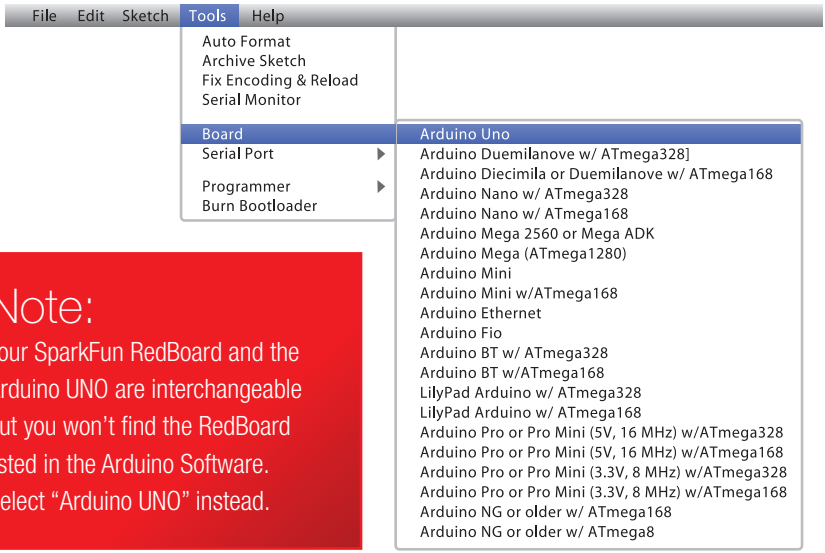
// The three most important commands for this guide are seen below:

Open

Verify

Upload

4 // Select your board: Arduino Uno



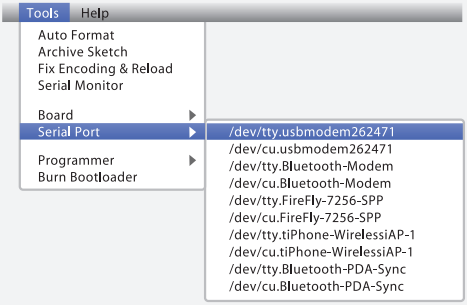
Note:

Your SparkFun RedBoard and the Arduino UNO are interchangeable but you won't find the RedBoard listed in the Arduino Software. Select "Arduino UNO" instead.

// Select your Serial Device

Select the serial device of the RedBoard from the Tools | Serial Port menu. This is likely to be **com3 or higher** (COM1 and COM2 are usually reserved for hardware serial ports). To find out, you can disconnect your RedBoard and re-open the menu; the entry that disappears should be the RedBoard. Reconnect the board and select that serial port.

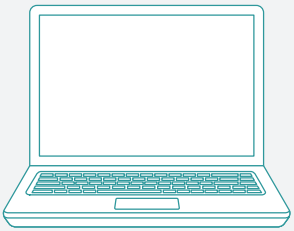
Select the serial device of the RedBoard from the Tools > Serial Port menu. On the Mac, this should be something with **/dev/tty.usbmodem** or **/dev/tty.usbserial** in it.



http://www.arduino.cc/playground/Learning/Linux

Download Arduino Code (For use with the circuits in this guide)

5



Type in the following URL to download the code:

sparkfun.com/sikcode

// Copy "SIK Guide Code" into "Examples" library in Arduino folder

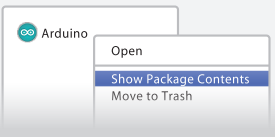
Unzip the file "SIK Guide Code". It should be located in your browser's **"Downloads"** folder. Right click the zipped folder and choose **"unzip"**.

Start → Programs → arduino → examples

Copy the "SIK Guide Code" folder into Arduino's folder named "examples".

Unzip the file **"SIK Guide Code"**. It should be loacted in your browser's **"Downloads"** folder. Right click the zipped folder and choose **"unzip"**.

Find "Arduino" in your applications folder. Right click(ctrl + click) on "Arduino". Select "Show Package Contents".



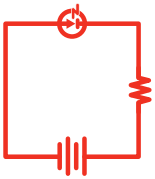
Contents
↓
Resources
↓
Java
↓
examples

Copy the "SIK Guide Code" folder into Arduino's folder named "examples".

http://www.arduino.cc/playground/Learning/Linux

WHAT'S NEXT? Read on to learn more about getting started with circuits. Then you can start on your first circuit on page 17!

Getting Started with Circuits



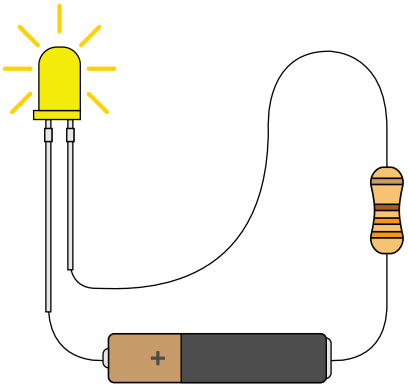
What is an Electrical Circuit?

A circuit is basically an electrical loop with a starting point and an ending point - with any number of components in between. Circuits can include resistors, diodes, inductors, sensors of all sizes and shapes, motors, and any other handful of hundreds of thousands of components.

Circuits are usually divided into three categories - analog circuits, digital circuits, or mixed-signal circuits. In this guide, you will explore all three sets of circuits.

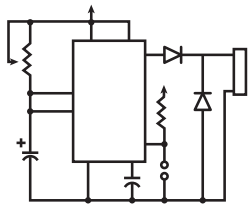
The World Runs on Circuits:

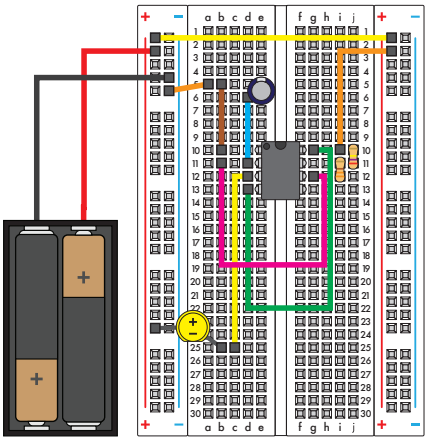
Everywhere you look, you'll find circuits. The cell phone in your pocket, the computer that controls your car's emissions system, your video game console - all these things are chock full of circuits. In this guide, you'll experiment with some simple circuits and learn the gist of the world of embedded electronics.

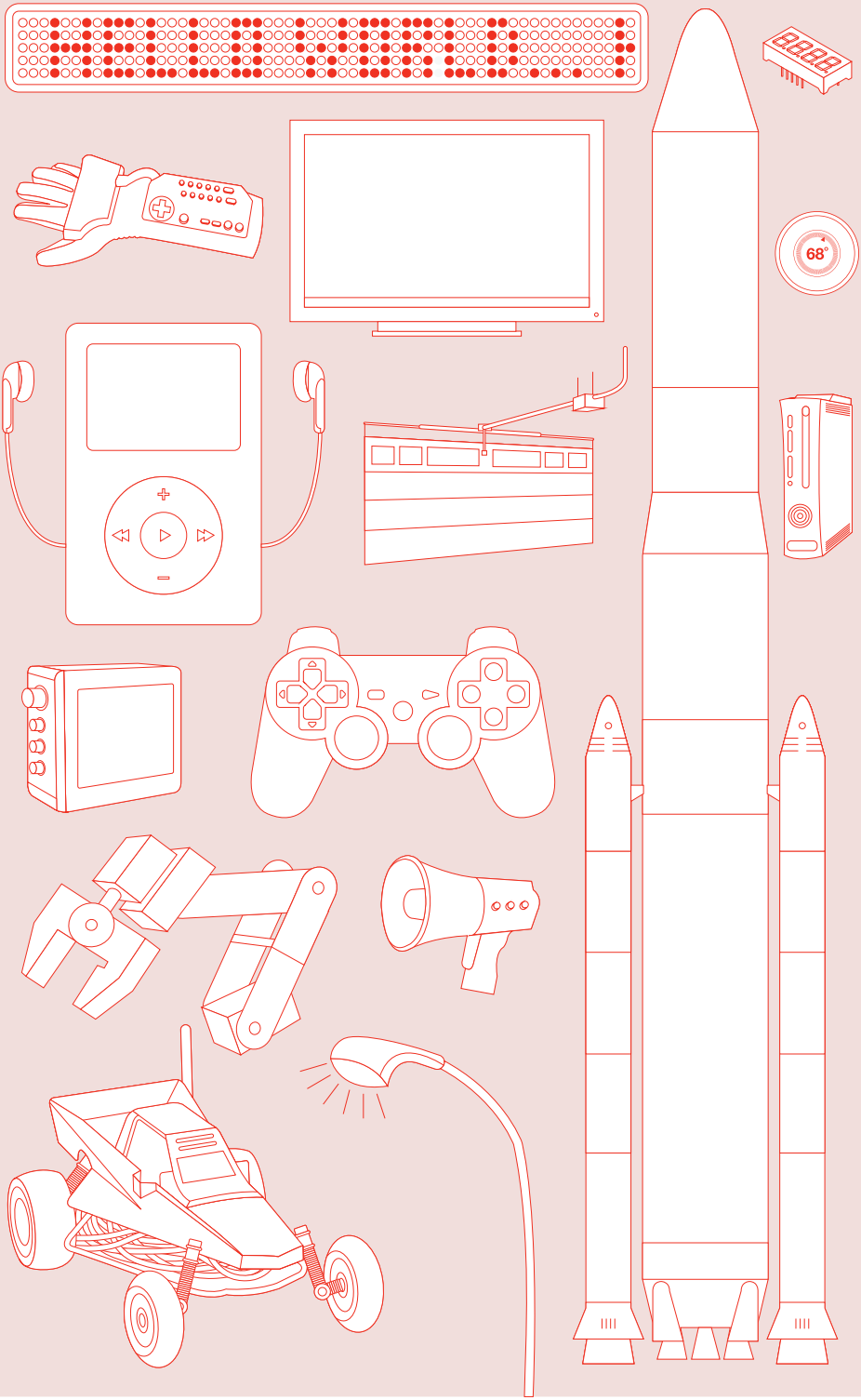


// Simple and Complex Circuits

In this guide, you will be primarily exploring simple circuits - but that doesn't mean you can't do amazing things with simple tools! When you've finished the SIK, your knowledge of circuits will enable you to explore amazing projects and unleash the power of your imagination.

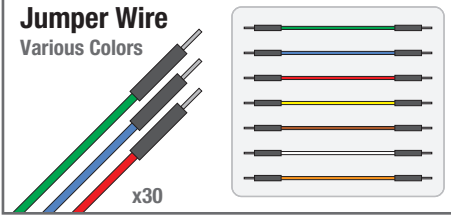




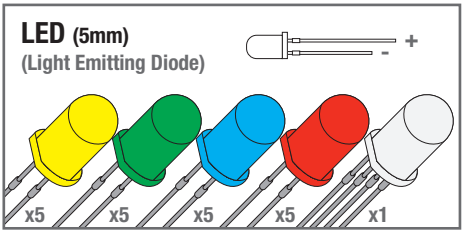

68°

Inventory of Parts

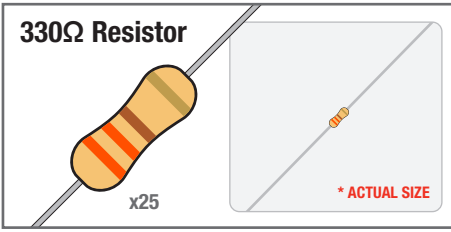
Jumper Wire
Various Colors



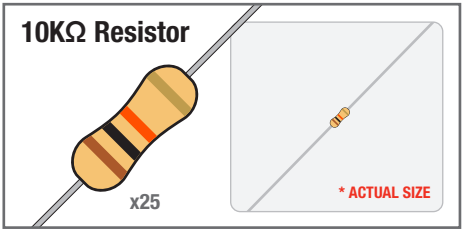
LED (5mm)
(Light Emitting Diode)



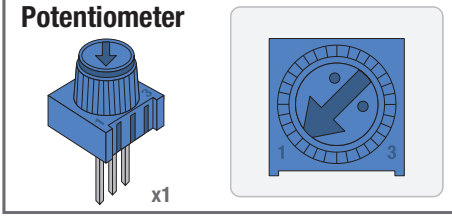
330Ω Resistor



10KΩ Resistor



Potentiometer



Diode
(1N4148)

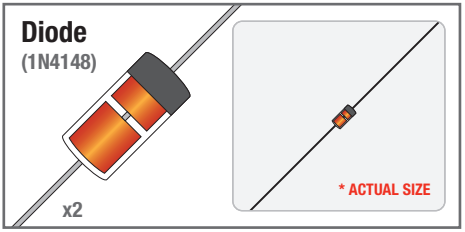
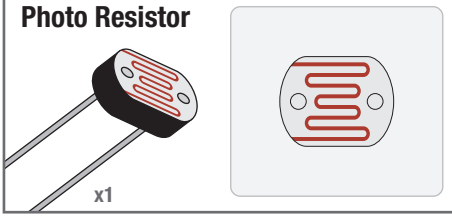
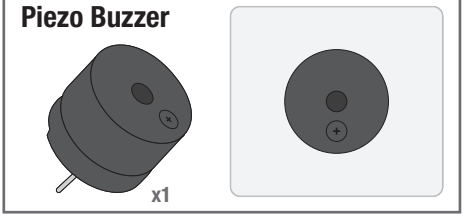


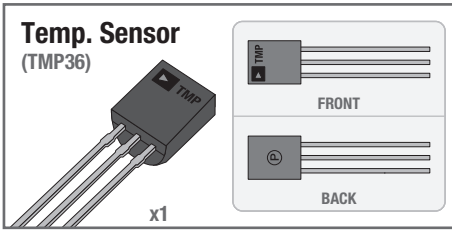
Photo Resistor



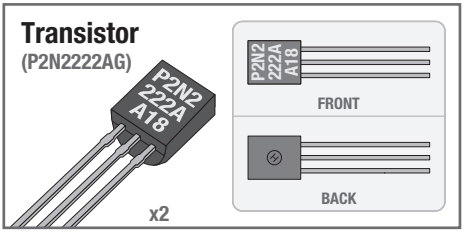
Piezo Buzzer



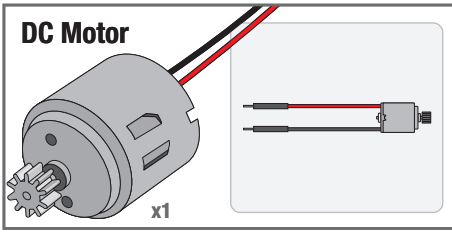
Temp. Sensor
(TMP36)



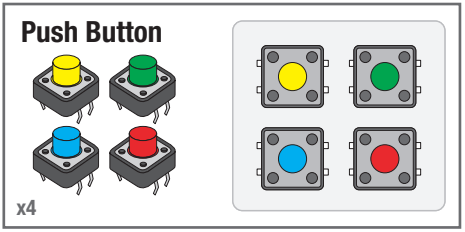
Transistor
(P2N2222AG)



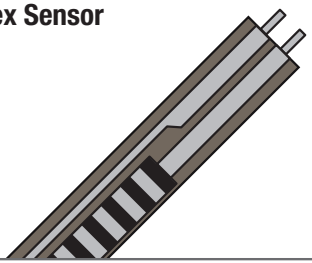
DC Motor



Push Button

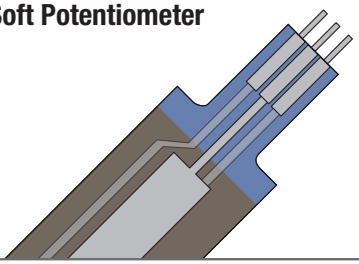


Flex Sensor



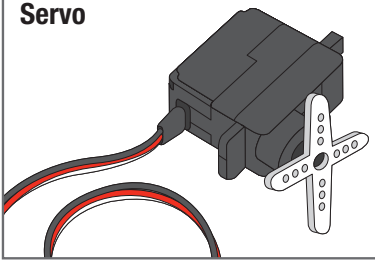
x1

Soft Potentiometer



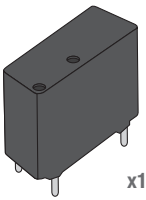
x1

Servo



x1

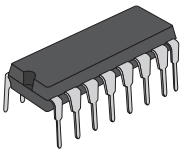
Relay



x1

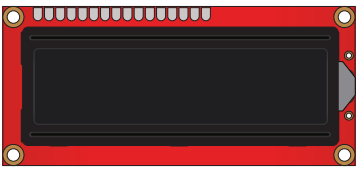
Integrated Circuit

(IC)



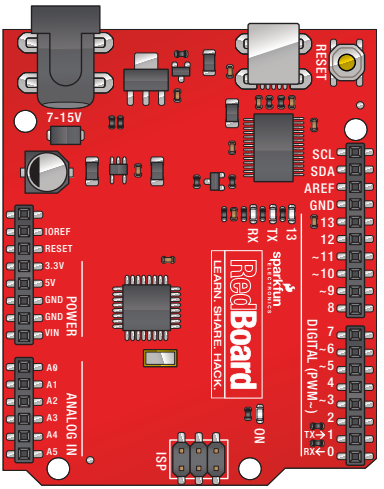
x1

LCD



x1

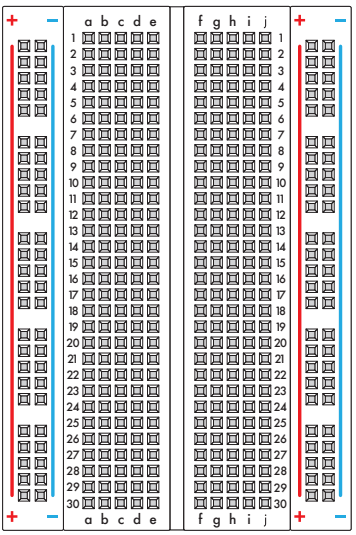
SparkFun RedBoard



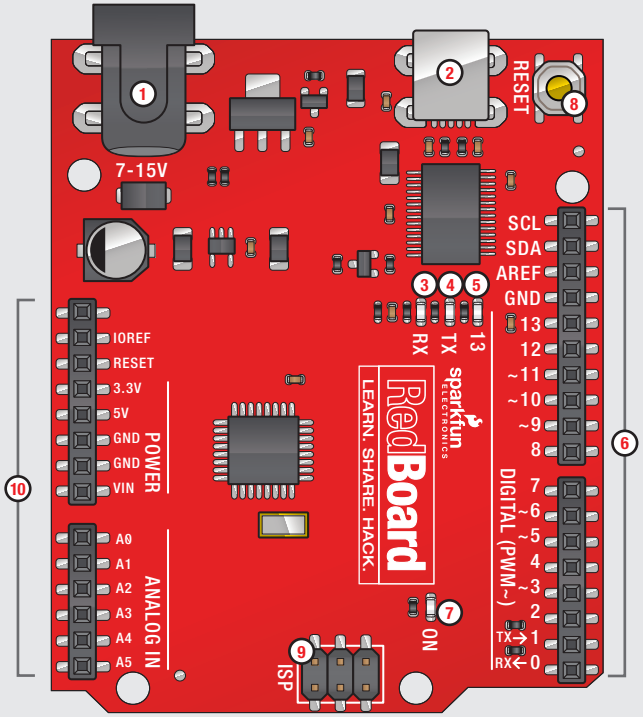
x1

Breadboard

Standard Solderless (Color may vary)



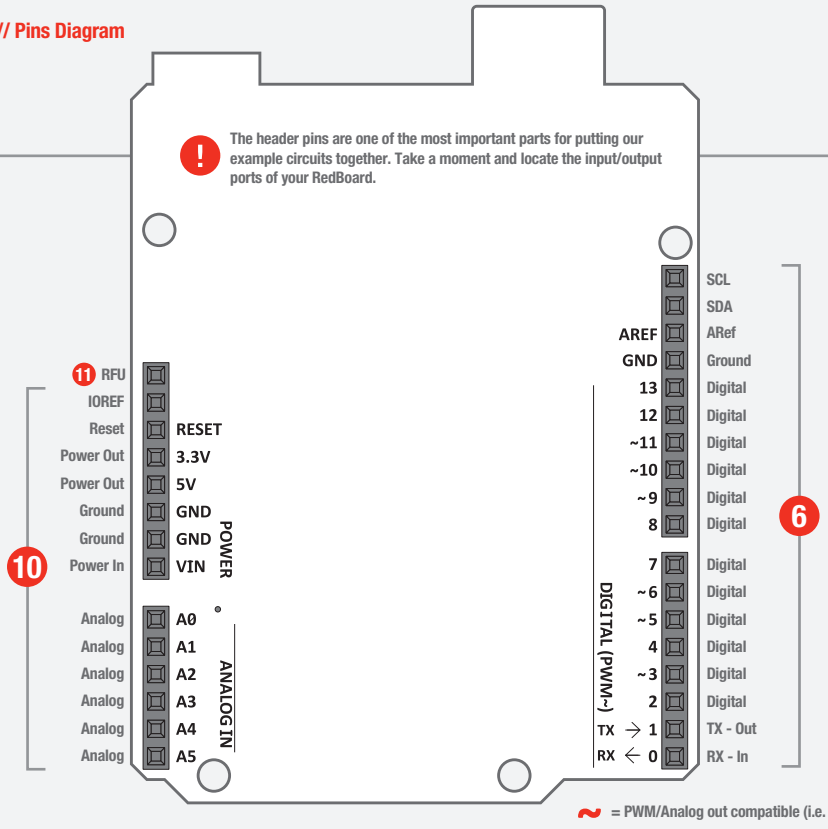
x1

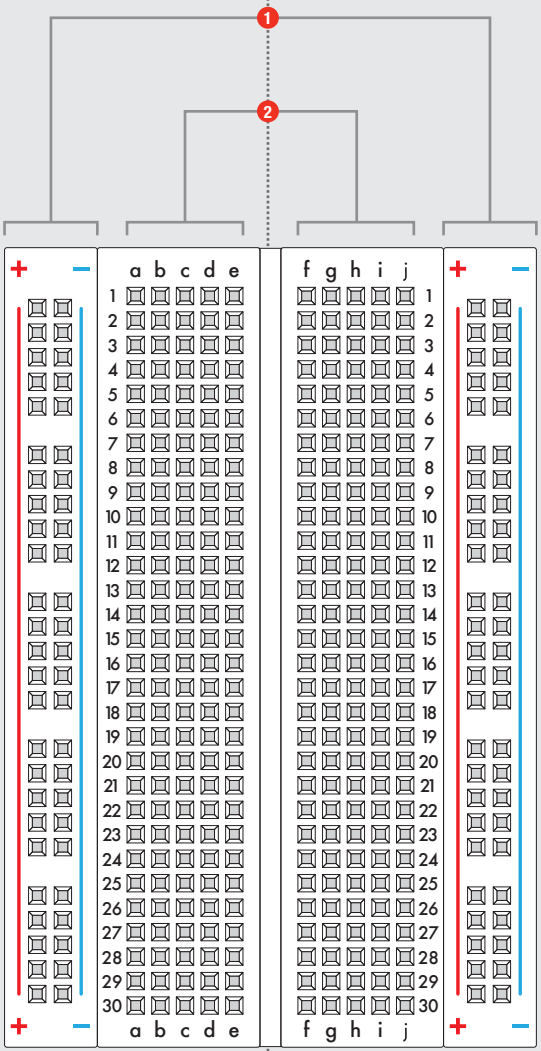

1
2
RESET
8
7-15V
SCL
SDA
AREF
GND
13
12
~11
~10
~9
8
3
4
5
RX
TX
13
IOREF
RESET
3.3V
5V
GND
GND
VIN
POWER
10
RedBoard
LEARN. SHARE. HACK.
sparkfun
ELECTRONICS
6
7
~6
~5
4
~3
2
TX→1
RX←0
DIGITAL (PWM~)
A0
A1
A2
A3
A4
A5
ANALOG IN
ON
7
9
ISP

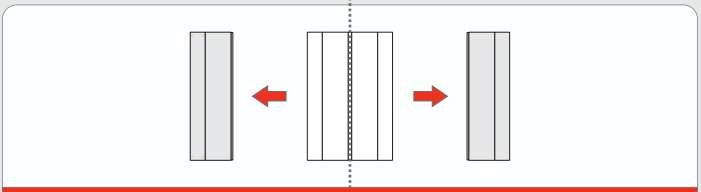
SparkFun RedBoard

1. **Power In** (Barrel Jack) - Can be used with either a 9V or 12V wall-wart or battery.
2. **Power In** (USB Port) - Provides power and communicates with your board when plugged into your computer via USB.
3. **LED** (RX: Receiving) - This shows when the FTDI chip is receiving data bits from the microcontroller. This happens when the microcontroller is sending data bits back to the computer.
4. **LED** (TX: Transmitting) - This shows when the FTDI chip is transmitting data bits to the microcontroller. This happens when the microcontroller is receiving this data from the computer.
5. **LED** (Pin 13: Troubleshooting) - This LED is incorporated into your sketch to show if your program is running properly.
6. **Pins** (ARef, Ground, Digital, Rx, Tx) - These various pins can be used for inputs, outputs, power, and ground. // See Diagram Below
7. **LED** (Indicates RedBoard is ON) - This is a simple power indicator LED.
8. **Reset Button** - This is a way to manually reset your RedBoard, which makes your code restart.
9. **ICSP Pins** (Uploading Code without Bootloader) - This is for "In-Circuit Serial Programming," used if you want to bypass the bootloader.
10. **Pins** (Analog In, Power In, Ground, Power Out, Reset) - These various pins can be used for inputs, outputs, power, and ground. // See Diagram
11. **RFU** - This pin is reserved for future use.

// Pins Diagram





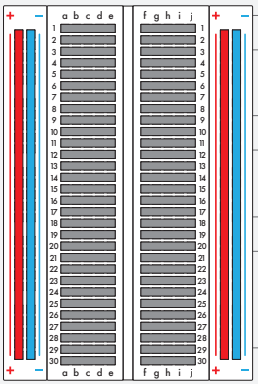


This line divides the board in half, restricting electricity to one half or the other.

Breadboard

1. **Vertical Connection** (+ Power and - Ground) - Power bus // See Diagram Below
2. **Horizontal Connection** (a-e & f-j) // See Diagram Below)

How's it all connected?



+ Power:

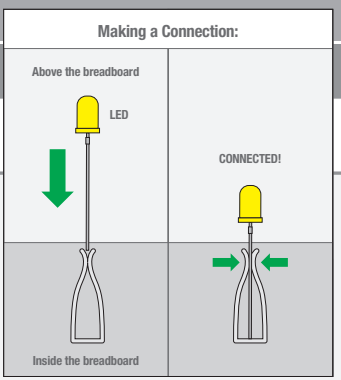
Each + sign runs power anywhere in the vertical column.

− Ground:

Each - sign runs to ground anywhere in the vertical column.

Horizontal Rows:

Each of these rows numbered 1-30 are comprised of five horizontal sockets. Components placed in the same row will be connected in a circuit when power is running.



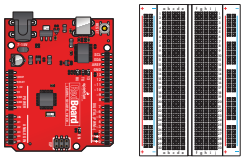
View of the inside >>>

CIRCUIT #1 - *Your First Circuit*

How It Works:

1 ASSEMBLE **2 WRITE** **3 UPLOAD**

+ Make sure the text on the RedBoard and breadboard are facing up so you can read them.



+ Connect the USB cable.

+ Screw the RedBoard down and into place.

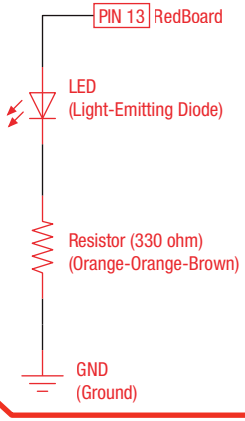
+ Peel sticker off back of breadboard and stick into place.

Your RedBoard runs on 5V. This is the power that will be supplied from your computer via USB and will be the driving force behind any components you use in your circuits. By plugging your RedBoard into your computer, you are supplying it with just the right voltage it needs to thrive! 5V can't hurt you, so don't be afraid to touch anything in your circuit. You can also power the RedBoard through the barrel jack. The on-board voltage regulator can handle anything from 7 to 15VDC.

1

Blinking an LED

LEDs (light-emitting diodes) are small, powerful lights that are used in many different applications. To start off the SIK, we will work on blinking an LED. That's right - it's as simple as turning a light on and off. It might not seem like much, but establishing this important baseline will give you a solid foundation as we work toward more complex experiments.



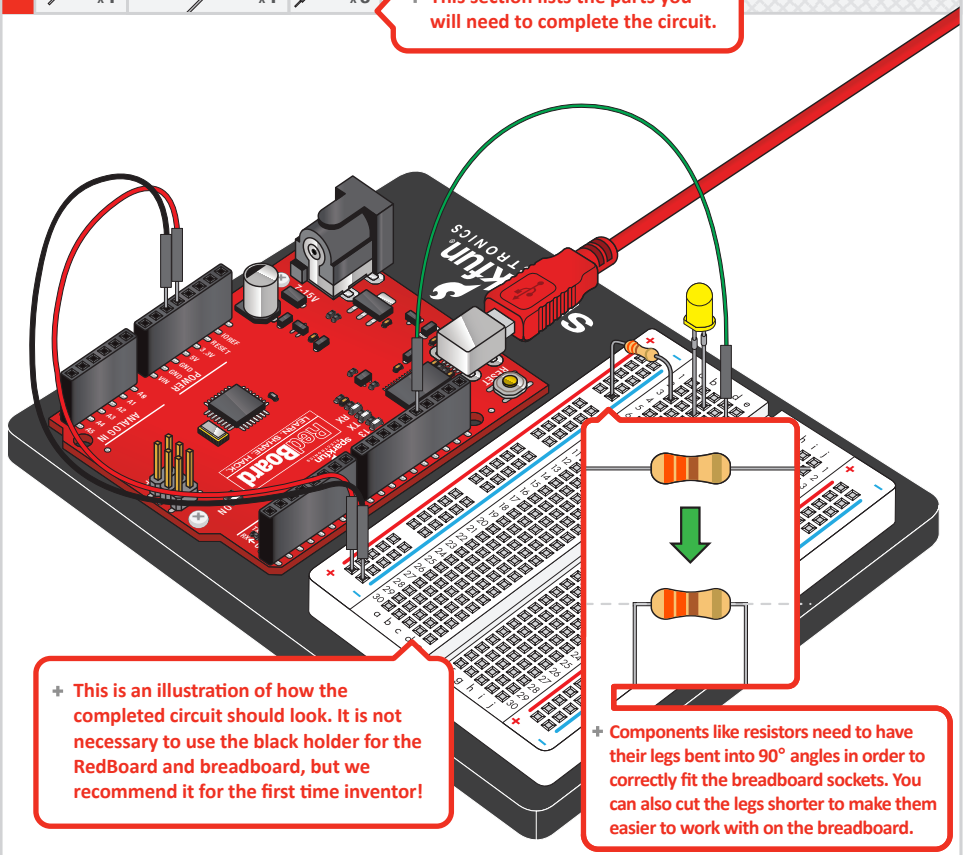
+ This is a schematic of your circuit.

+ Each circuit begins with a brief description of the what you are putting together and the expected result.

PARTS:

LED	330Ω Resistor	Wire
x1	x1	x3

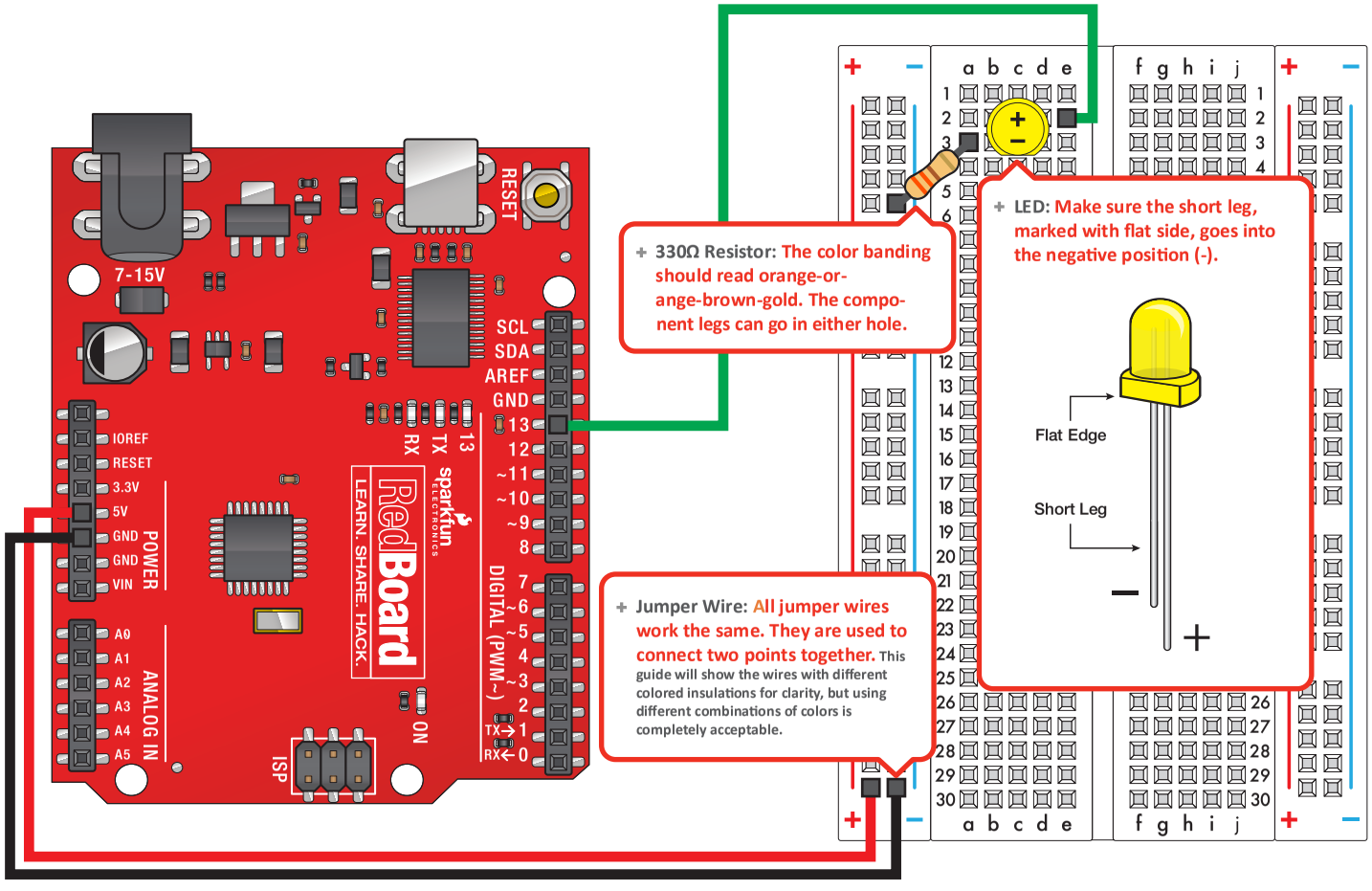
+ This section lists the parts you will need to complete the circuit.

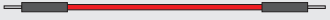
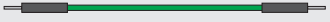


+ This is an illustration of how the completed circuit should look. It is not necessary to use the black holder for the RedBoard and breadboard, but we recommend it for the first time inventor!

+ Components like resistors need to have their legs bent into 90° angles in order to correctly fit the breadboard sockets. You can also cut the legs shorter to make them easier to work with on the breadboard.

Circuit 1: Blinking an LED



Component:	Image Reference:		
LED (5mm) ⚠			c2 - c3 + -
330Ω Resistor			a3 - -
Jumper Wire		GND	-
Jumper Wire		5V	+
Jumper Wire		Pin 13	e2

+ Components like LEDs are inserted into the breadboard sockets c2(long leg) c3(short leg). Steps highlighted with a yellow warning triangle represent a polarized component. Pay special attention to the component's markings indicating how to place it on the breadboard.

+ Resistors are placed in breadboard sockets only. The "-" symbol represents any socket in its vertical column on the Power bus.

+ "GND" on the RedBoard should be connected to the row marked "-" on the breadboard.

+ "5V" on the RedBoard connects to the row marked "+" on the breadboard.

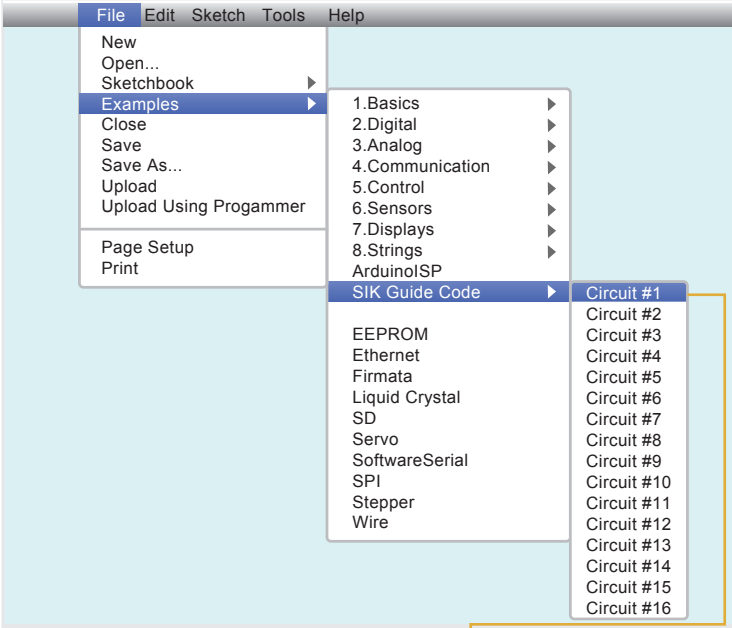
+ "Pin 13" on the RedBoard connects to socket "e2" on the breadboard.

+ **RedBoard:** The red background represents a connection to one of the RedBoard header pins.

+ **Breadboard:** The white background represents a connection to a breadboard socket specified by a letter-number coordinate such as e2. These coordinates are merely suggestions that align with the graphic image.

Open Your First Sketch:

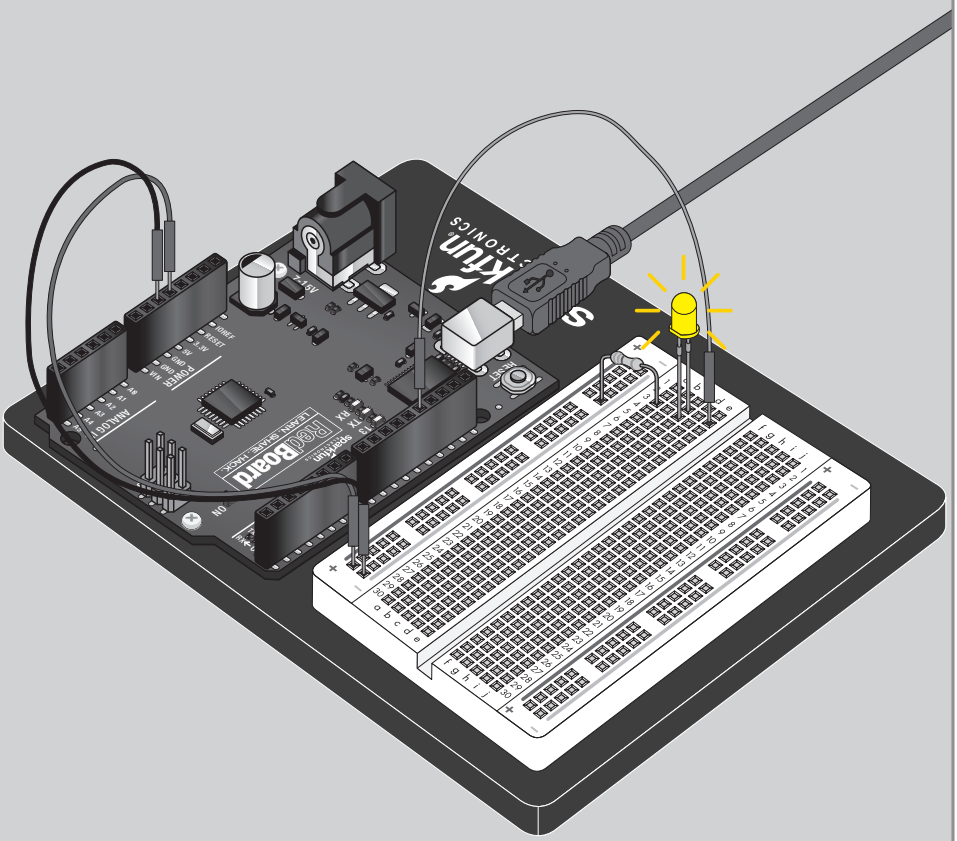
Open Up the Arduino IDE software on your computer. Coding in the Arduino language will control your circuit. Open the code for Circuit 1 by accessing the "SIK Guide Code" you downloaded and placed into your "Examples" folder earlier.



// Circuit #1

Verify	This compiles your code. The IDE changes it from text into instructions the computer can understand.
Upload	This sends the instructions via the USB cable to the computer chip on the RedBoard. The RedBoard will then begin running your code automatically.

// The result of a completed circuit with correct code after verified and uploaded.



1

+ This is where you will find the Arduino code for each circuit.

Open Arduino IDE // File > Examples > SIK Guide > Circuit # 1

Code to Note:

+ Remember to Verify and Upload your code.

+ Begin to understand how the Arduino code works. See below.

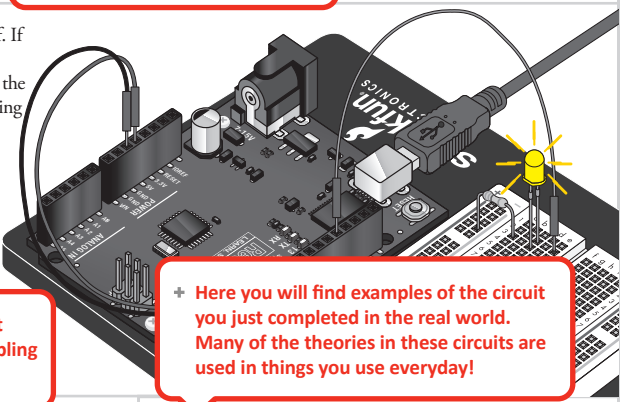
pinMode(13, OUTPUT); ⇨ Before you can use one of the RedBoard's pins, you need to tell the RedBoard whether it is an INPUT or OUTPUT. We use a built-in "function" called pinMode() to do this.

digitalWrite(13, 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).

What you Should See:

+ See if your circuit is complete and working in this section.

You should see your LED blink on and off. If it isn't, make sure you have assembled the circuit correctly and verified and uploaded the code to your board or see the troubleshooting tips below.



+ Here you will find examples of the circuit you just completed in the real world. Many of the theories in these circuits are used in things you use everyday!

+ This is a section dedicated to the most common mistakes made while assembling the circuit.

Troubleshooting:

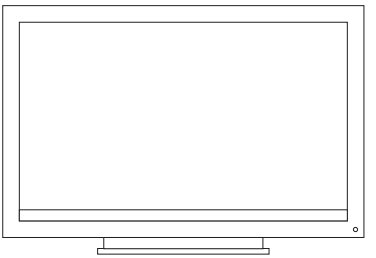
LED Not Lighting Up?
LEDs will only work in one direction. Try taking it out and twisting it 180 degrees (no need to worry, installing it backward does no permanent harm).

Program Not Uploading
This happens sometimes, the most likely cause is a confused serial port, you can change this in tools>serial port>

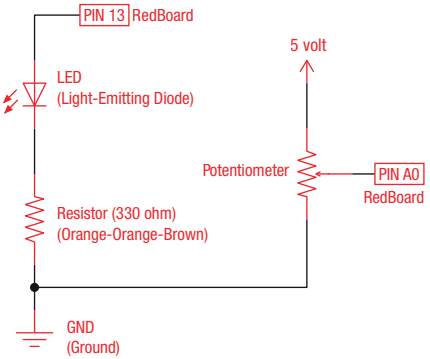
Still No Success?
A broken circuit is no fun, send us an e-mail and we will get back to you as soon as we can: **techsupport@sparkfun.com**

Real World Application:

Almost all modern flat screen televisions and monitors have LED indicator lights to show they are on or off.



2

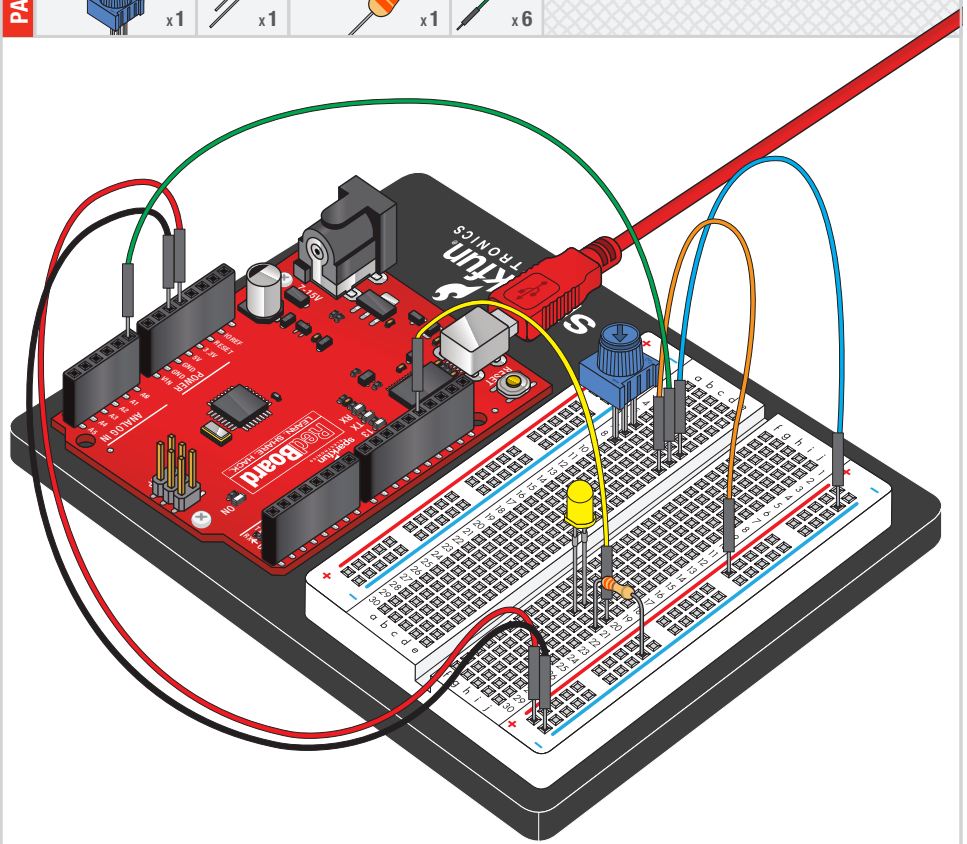


Potentiometer

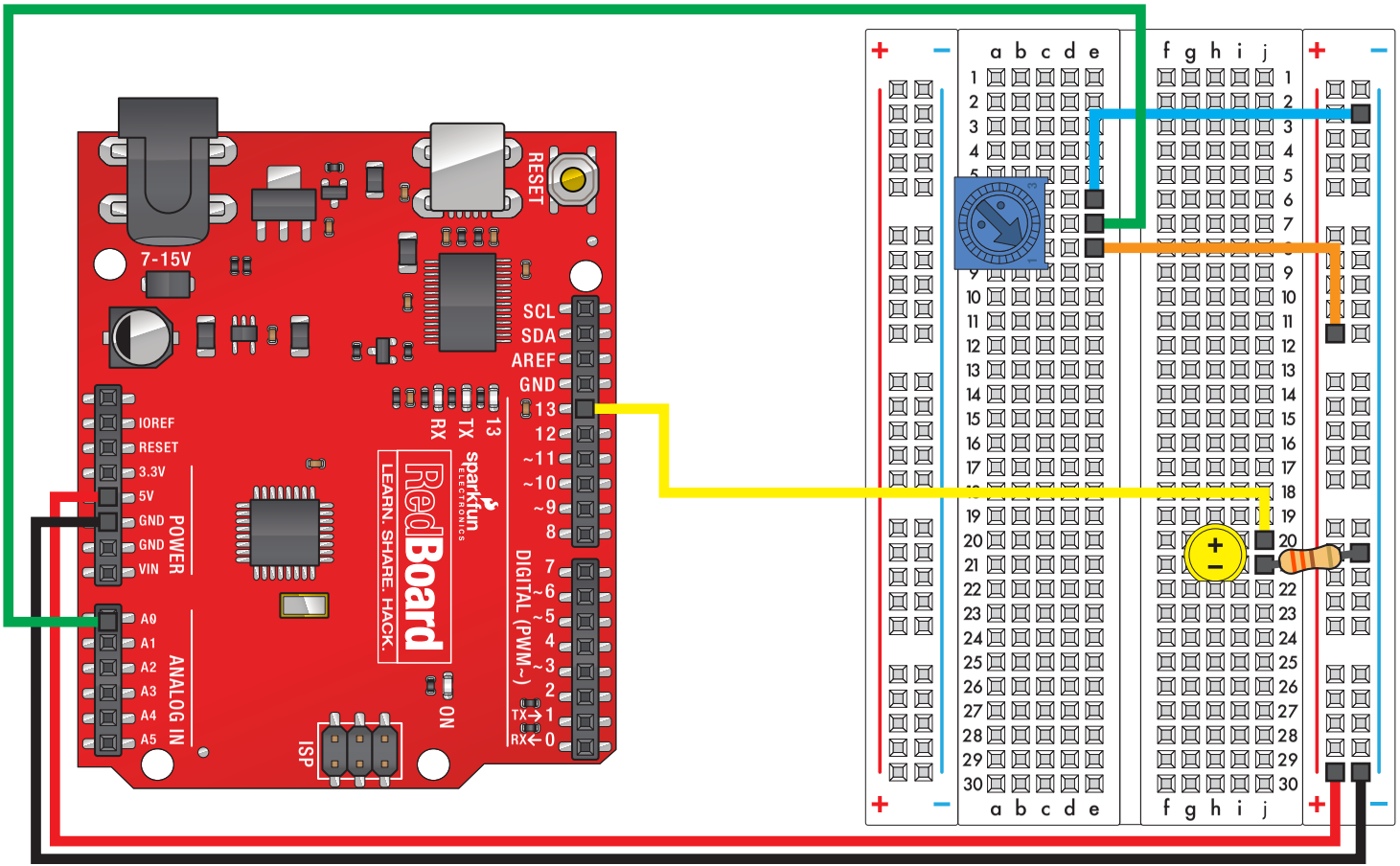
In this circuit you'll work with a potentiometer. A potentiometer is also known as a variable resistor. When it's connected with 5 volts across its two outer pins, the middle pin outputs a voltage between 0 and 5, depending on the position of the knob on the potentiometer. A potentiometer is a perfect demonstration of a variable voltage divider circuit. The voltage is divided proportionate to the resistance between the middle pin and the ground pin. In this circuit, you'll learn how to use a potentiometer to control the brightness of an LED.

PARTS:

Potentiometer	LED	330Ω Resistor	Wire
x1	x1	x1	x6



Circuit 2: Potentiometer



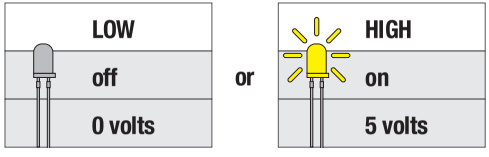
Component:	Image Reference:		
Potentiometer			a6 a7 a8
LED (5mm) ⚠			h20 (+) h21 (-)
330Ω Resistor			j21 -
Jumper Wire			e6 -
Jumper Wire		A0	e7
Jumper Wire			e8 +
Jumper Wire		Pin 13	j20
Jumper Wire		5V	+
Jumper Wire		GND	-

Digital versus Analog:

If you look closely at your RedBoard, you'll see some pins labeled "DIGITAL", and some labeled "ANALOG". What's the difference?

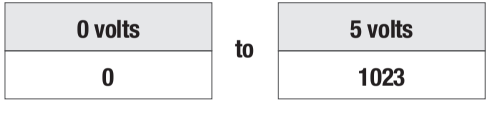
Many of the devices you'll interface to, such as LEDs and pushbuttons, have only two possible states: on and off, or as they're known to the RedBoard, "HIGH" (5 volts) and "LOW" (0 volts). The digital pins on an RedBoard are great at getting these signals to and from the outside world, and can even do tricks like simulated dimming (by blinking on and off really fast), and serial communications (transferring data to another device by encoding it as patterns of HIGH and LOW).

DIGITAL



But there are also a lot of things out there that aren't just "on" or "off". Temperature levels, control knobs, etc. all have a continuous range of values between HIGH and LOW. For these situations, the RedBoard offers six analog inputs that translate an input voltage into a number that ranges from 0 (0 volts) to 1023 (5 volts). The analog pins are perfect for measuring all those "real world" values, and allow you to interface the RedBoard to all kinds of things.

ANALOG



2

Arduino Code:

Open Arduino IDE // File > Examples > SIK Guide > Circuit # 2

Code to Note:

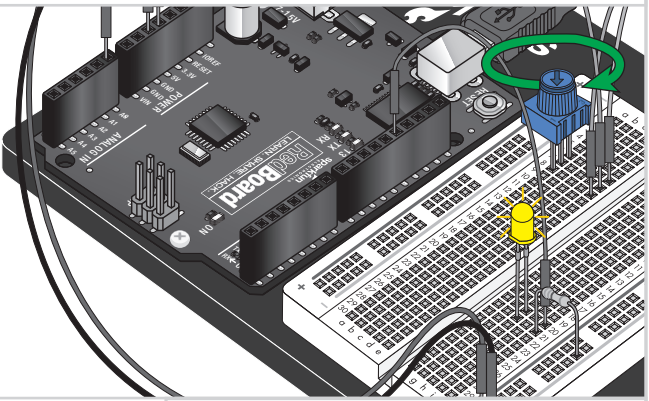
`int sensorValue;` ⇨ A "variable" is a stored value you've given a name to. You must introduce, or "declare" variables before you use them; here we're declaring a variable called sensorValue, of type "int" (integer). Don't forget that variable names are case-sensitive!

`sensorValue = analogRead(sensorPin);` ⇨ We use the analogRead() function to read the value on an analog pin. analogRead() takes one parameter, the analog pin you want to use ("sensorPin"), and returns a number ("sensorValue") between 0 (0 volts) and 1023 (5 volts).

`delay(sensorValue);` ⇨ The Arduino is very very fast, capable of running thousands of lines of code each second. To slow it down so that we can see what it's doing, we'll often insert delays into the code. delay() counts in milliseconds; there are 1000 ms in one second.

What you Should See:

You should see the LED blink faster or slower in accordance with your potentiometer. If it isn't working, make sure you have assembled the circuit correctly and verified and uploaded the code to your board or see the troubleshooting tips below.



Troubleshooting:

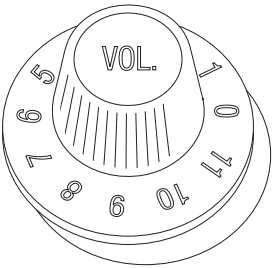
Sporadically Working
This is most likely due to a slightly dodgy connection with the potentiometer's pins. This can usually be conquered by holding the potentiometer down.

Not Working
Make sure you haven't accidentally connected the wiper, the resistive element in the potentiometer, to digital pin 0 rather than analog pin 0. (the row of pins beneath the power pins).

LED Not Lighting Up?
LEDs will only work in one direction. Try taking it out and twisting it 180 degrees (no need to worry, installing it backward does no permanent harm).

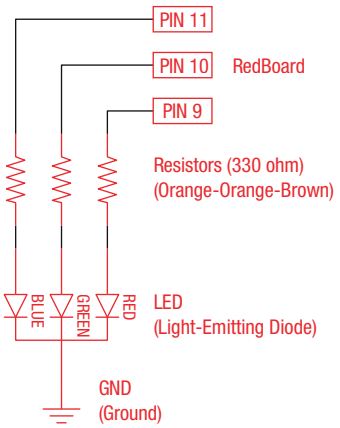
Real World Application:

Most traditional volume knobs employ a potentiometer.



CIRCUIT #3

3

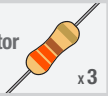


RGB LED

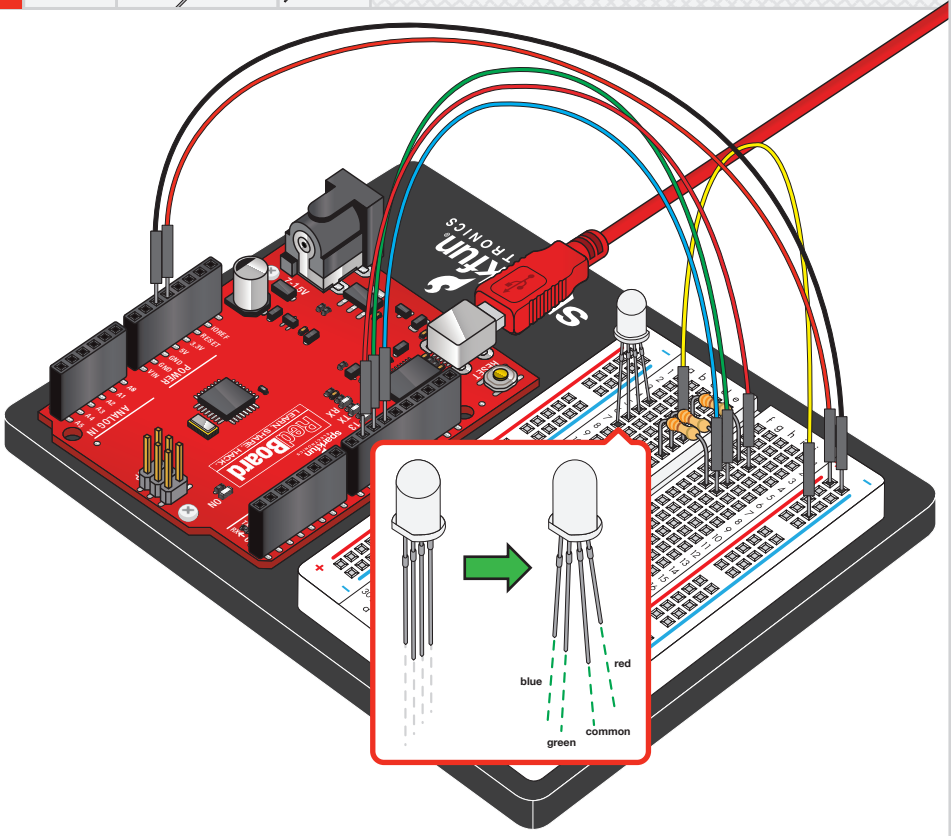
You know what's even more fun than a blinking LED? Changing colors with one LED. RGB, or red-green-blue, LEDs have three different color-emitting diodes that can be combined to create all sorts of colors. In this circuit, you'll learn how to use an RGB LED to create unique color combinations. Depending on how bright each diode is, nearly any color is possible!

PARTS:

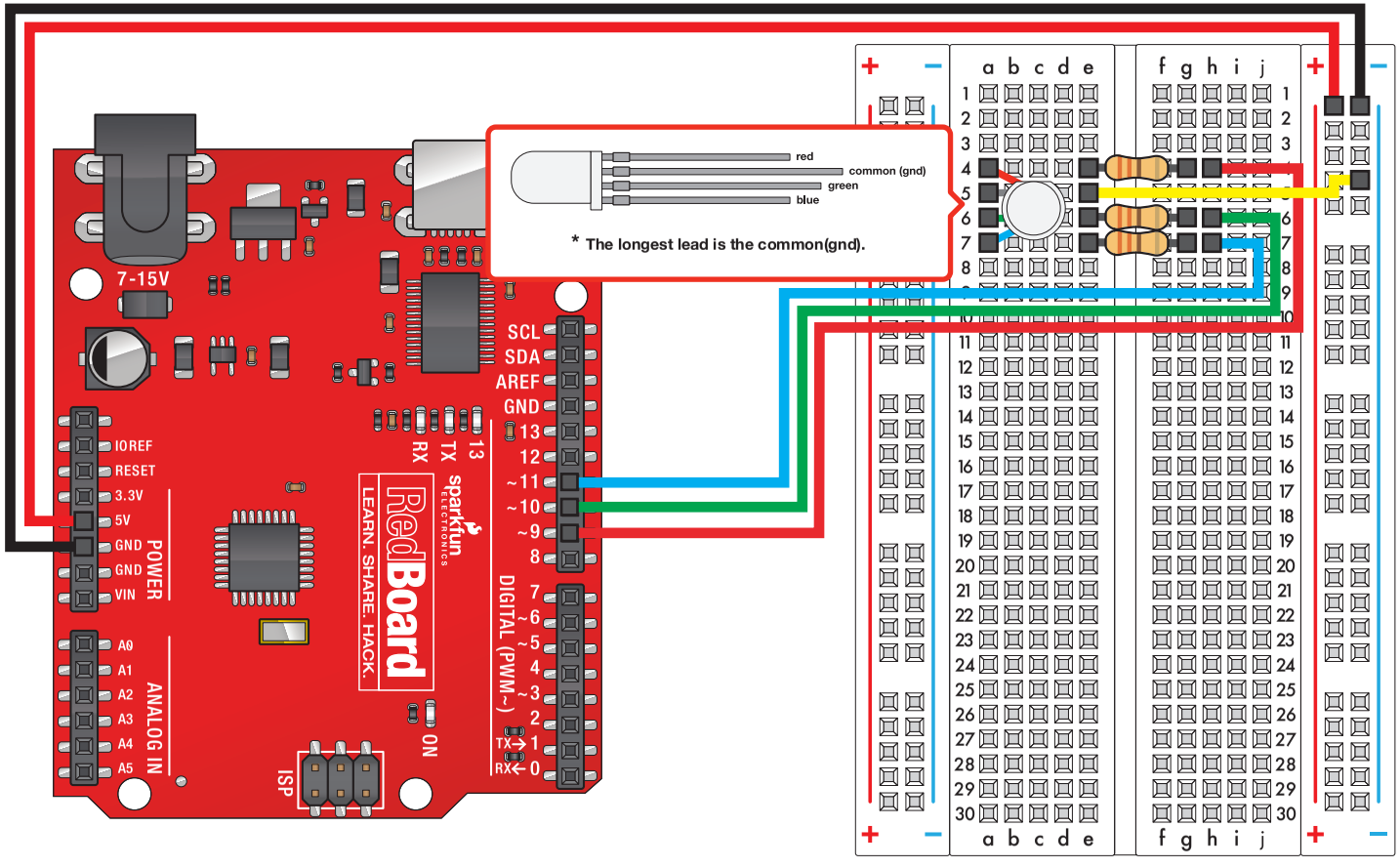

LED x1

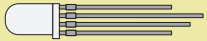
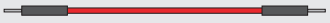
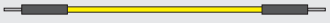
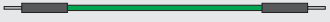
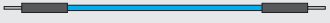
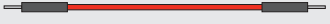

330Ω Resistor x3


Wire x6



Circuit 3: RGB LED



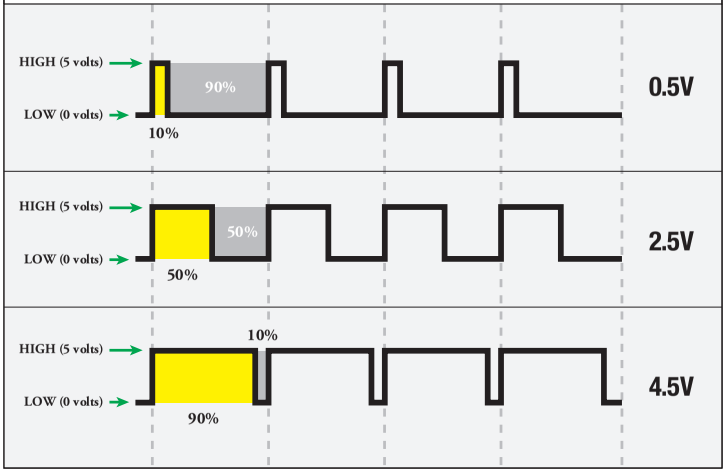
Component:	Image Reference:		
RGB LED (5mm) ⚠			a4 a5 a6 a7
330Ω Resistor			e4 g4
330Ω Resistor			e6 g6
330Ω Resistor			e7 g7
Jumper Wire		Pin 9	h4
Jumper Wire			e5 -
Jumper Wire		Pin 10	h6
Jumper Wire		Pin 11	h7
Jumper Wire		5V	+
Jumper Wire		GND	-

The shocking truth behind analogWrite():

We've seen that the Arduino can read analog voltages (voltages between 0 and 5 volts) using the **analogRead()** function. Is there a way for the RedBoard to output analog voltages as well?

The answer is no... and yes. The RedBoard does not have a true analog voltage output. But, because the RedBoard is so fast, it can fake it using something called **PWM ("Pulse-Width Modulation")**. The pins on the RedBoard with "~" next to them are PWM/Analog out compatible.

The RedBoard is so fast that it can blink a pin on and off almost 1000 times per second. PWM goes one step further by varying the amount of time that the blinking pin spends HIGH vs. the time it spends LOW. If it spends most of its time HIGH, a LED connected to that pin will appear bright. If it spends most of its time LOW, the LED will look dim. Because the pin is blinking much faster than your eye can detect, the RedBoard creates the illusion of a "true" analog output.



3 Arduino Code:

Open Arduino IDE // File > Examples > SIK Guide > Circuit # 3

Code to Note:

```
for (x = 0; x < 768; x++)
{}
```

A for() loop is used to step a number across a range, and repeatedly runs code within the brackets {}. Here the variable "x" starts a 0, ends at 767, and increases by one each time ("x++").

```
if (x <= 255)
{}
else
{}
```

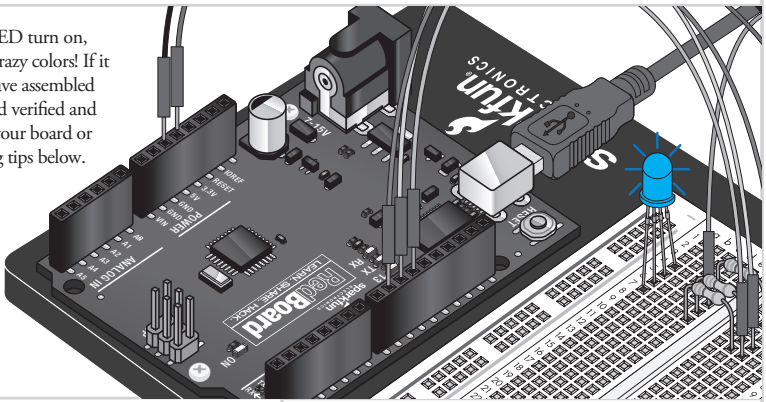
"If / else" statements are used to make choices in your programs. The statement within the parenthesis () is evaluated; if it's true, the code within the first brackets {} will run. If it's not true, the code within the second brackets {} will run.

```
delay(sensorValue);
```

The RedBoard is very very fast, capable of running thousands of lines of code each second. To slow it down so that we can see what it's doing, we'll often insert delays into the code. delay() counts in milliseconds; there are 1000 ms in one second.

What you Should See:

You should see your LED turn on, but this time in new, crazy colors! If it isn't, make sure you have assembled the circuit correctly and verified and uploaded the code to your board or see the troubleshooting tips below.



Troubleshooting:

LED Remains Dark or Shows Incorrect Color
With the four pins of the LED so close together, it's sometimes easy to misplace one. Double check each pin is where it should be.

Seeing Red
The red diode within the RGB LED may be a bit brighter than the other two. To make your colors more balanced, use a higher Ohm resistor. Or adjust in code.

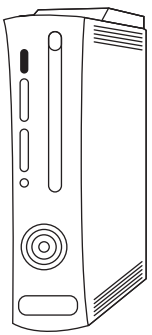
analogWrite(RED_PIN, redIntensity);

to

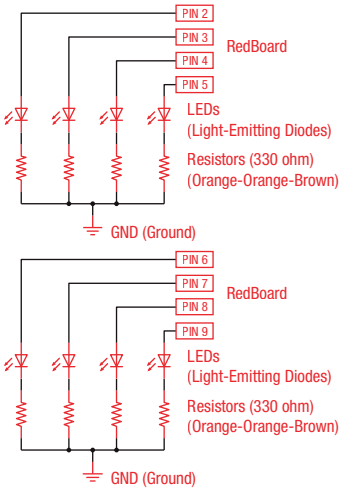
analogWrite(RED_PIN, redIntensity/3);

Real World Application:

Many electronics such as videogame consoles use RGB LEDs to have the versatility to show different colors in the same area. Often times the diffent colors represent different states of working condition.



4



Multiple LEDs

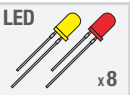
So you have gotten one LED to blink on and off – fantastic! Now it's time to up the stakes a little bit – by connecting EIGHT LEDS AT ONCE. We'll also give our RedBoard a little test by creating various lighting sequences. This circuit is a great setup to start practicing writing your own programs and getting a feel for the way RedBoard works.

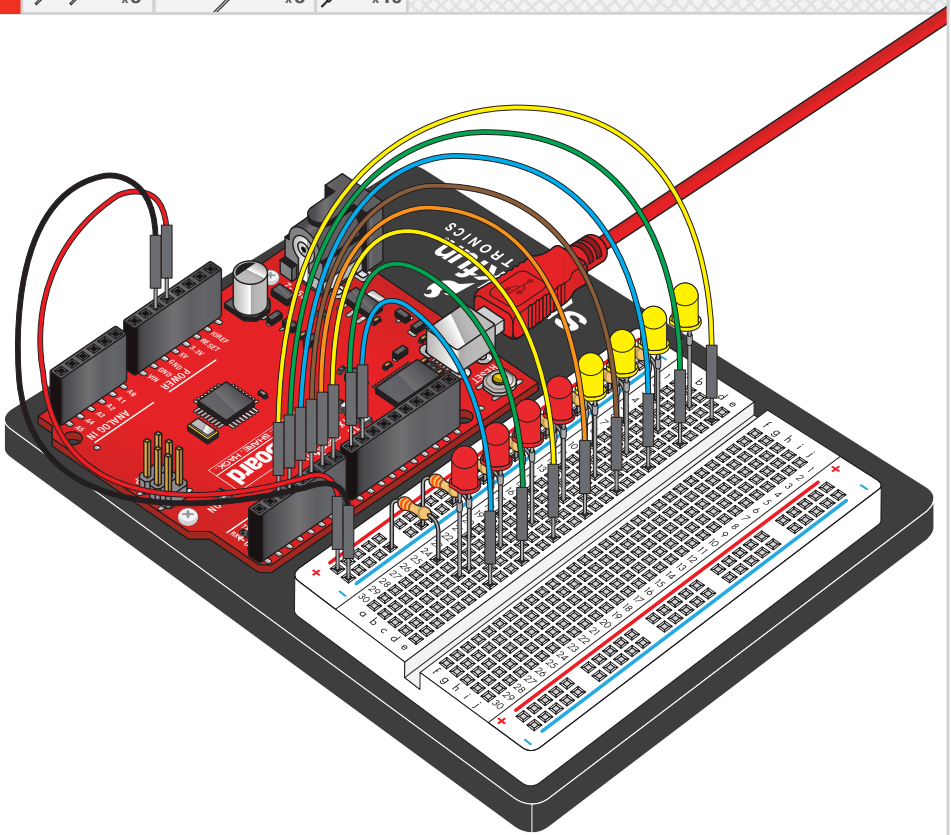
Along with controlling the LEDs, you'll learn about a couple programming tricks that keep your code neat and tidy:

for() loops - used when you want to run a piece of code several times

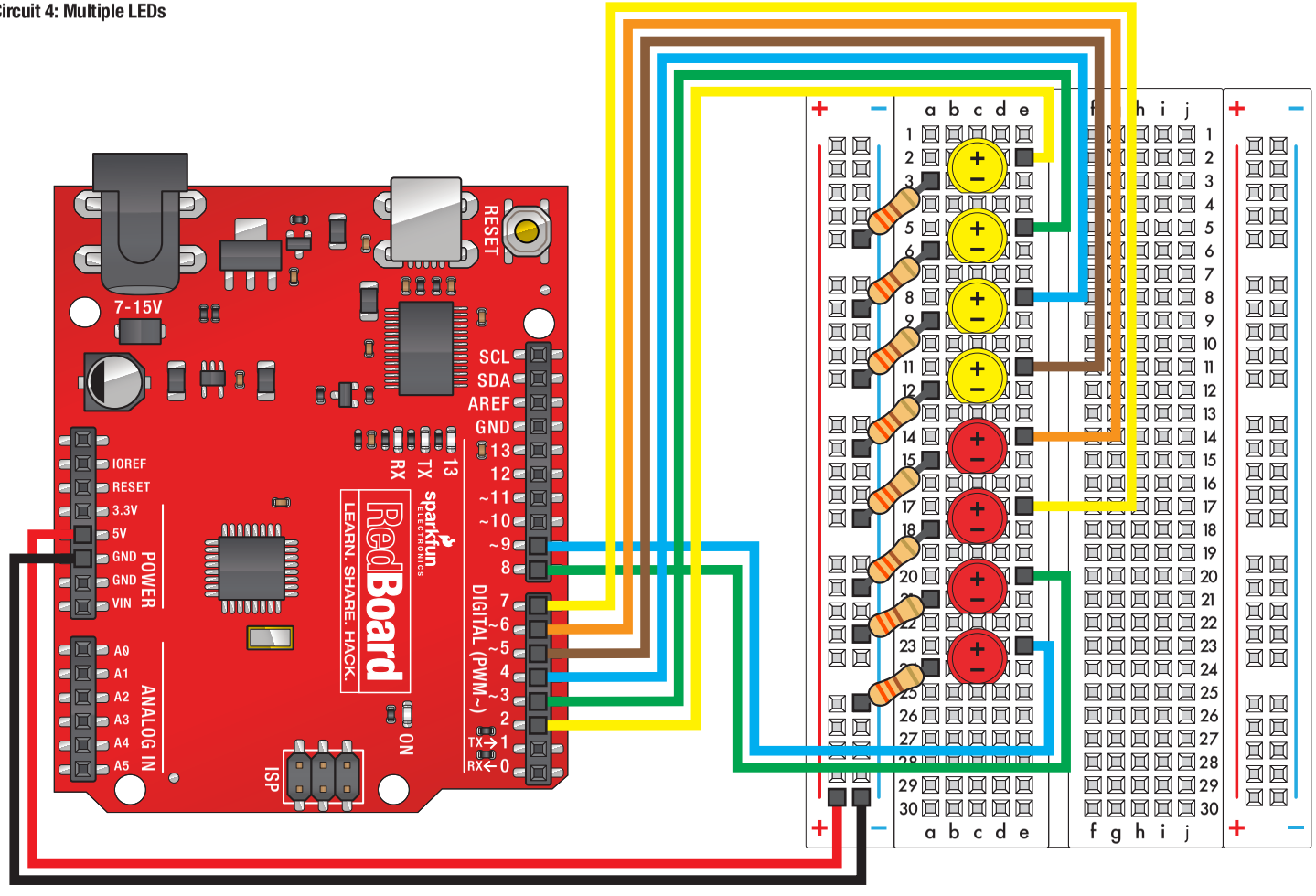
arrays[ ] - used to make managing variables easier by grouping them together

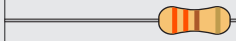
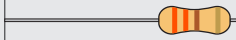
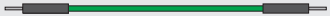
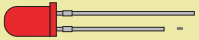
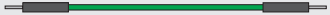
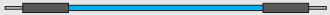
PARTS:





Circuit 4: Multiple LEDs



Component:	Image Reference:			Component:	Image Reference:		
LED (5mm) ⚠			c2 (+) – c3 (-)	330Ω Resistor			a18 –
LED (5mm) ⚠			c5 (+) – c6 (-)	330Ω Resistor			a21 –
LED (5mm) ⚠			c8 (+) – c9 (-)	330Ω Resistor			a24 –
LED (5mm) ⚠			c11 (+) – c12 (-)	Jumper Wire		Pin 2	e2
LED (5mm) ⚠			c14 (+) – c15 (-)	Jumper Wire		Pin 3	e5
LED (5mm) ⚠			c17 (+) – c18 (-)	Jumper Wire		Pin 4	e8
LED (5mm) ⚠			c20 (+) – c21 (-)	Jumper Wire		Pin 5	e11
LED (5mm) ⚠			c23 (+) – c24 (-)	Jumper Wire		Pin 6	e14
330Ω Resistor			a3 –	Jumper Wire		Pin 7	e17
330Ω Resistor			a6 –	Jumper Wire		Pin 8	e20
330Ω Resistor			a9 –	Jumper Wire		Pin 9	e23
330Ω Resistor			a12 –	Jumper Wire		5V	+
330Ω Resistor			a15 –	Jumper Wire		GND	–

4

Arduino Code:

Open Arduino IDE // File > Examples > SIK Guide > Circuit # 4

Code to Note:

`int ledPins[] = {2,3,4,5,6,7,8,9};`

When you have to manage a lot of variables, an "array" is a handy way to group them together. Here we're creating an array of integers, called ledPins, with eight elements.

`digitalWrite(ledPins[0], HIGH);`

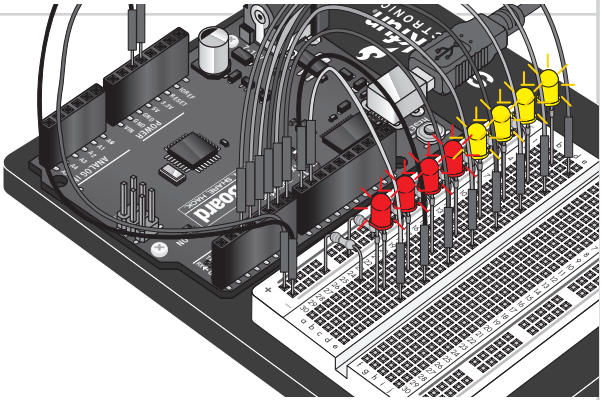
You refer to the elements in an array by their position. The first element is at position 0, the second is at position 1, etc. You refer to an element using "ledPins[x]" where x is the position. Here we're making digital pin 2 HIGH, since the array element at position 0 is "2".

`index = random(8);`

Computers like to do the same things each time they run. But sometimes you want to do things randomly, such as simulating the roll of a dice. The random() function is a great way to do this.
See **http://arduino.cc/en/reference/random** for more information.

What you Should See:

This is similar to circuit number one, but instead of one LED, you should see all the LEDs blink. If they aren't, make sure you have assembled the circuit correctly and verified and uploaded the code to your board or see the troubleshooting tips below.



Troubleshooting:

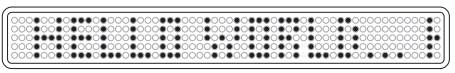
Some LEDs Fail to Light
It is easy to insert an LED backwards. Check the LEDs that aren't working and ensure they the right way around.

Operating out of sequence
With eight wires it's easy to cross a couple. Double check that the first LED is plugged into pin 2 and each pin there after.

Starting Afresh
Its easy to accidentally misplace a wire without noticing. Pulling everything out and starting with a fresh slate is often easier than trying to track down the problem.

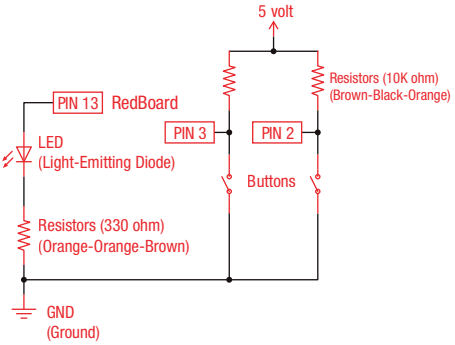
Real World Application:

Scrolling marquee displays are generally used to spread short segments of important information. They are built out of many LEDs.



5

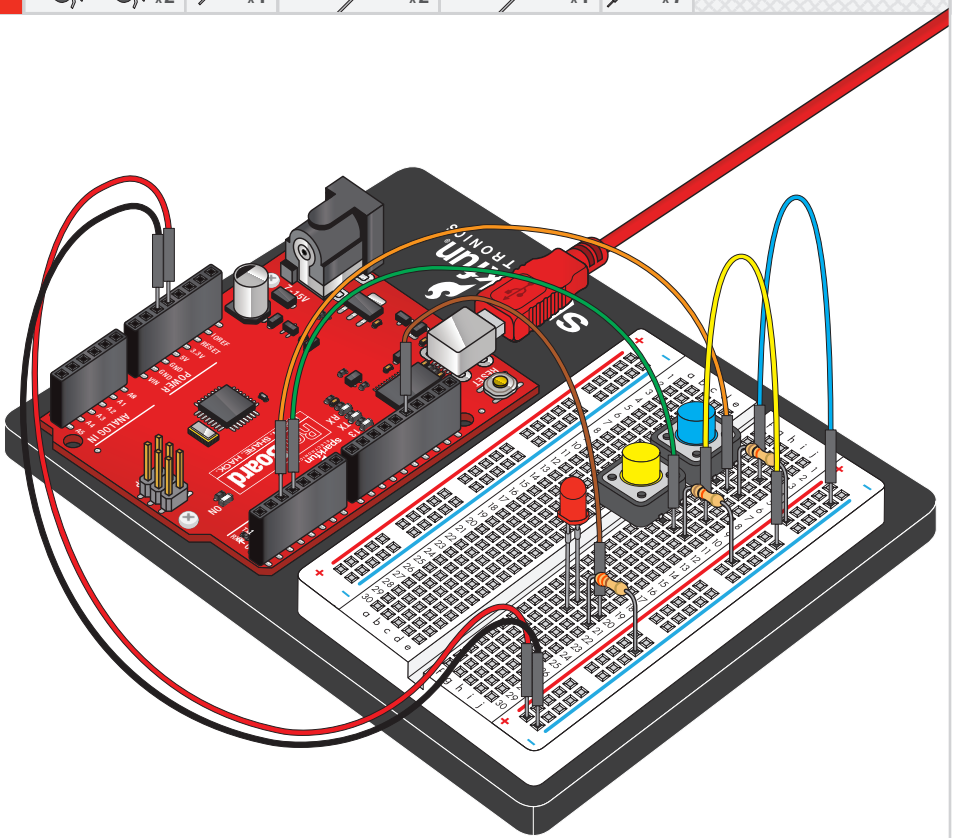
Push Buttons

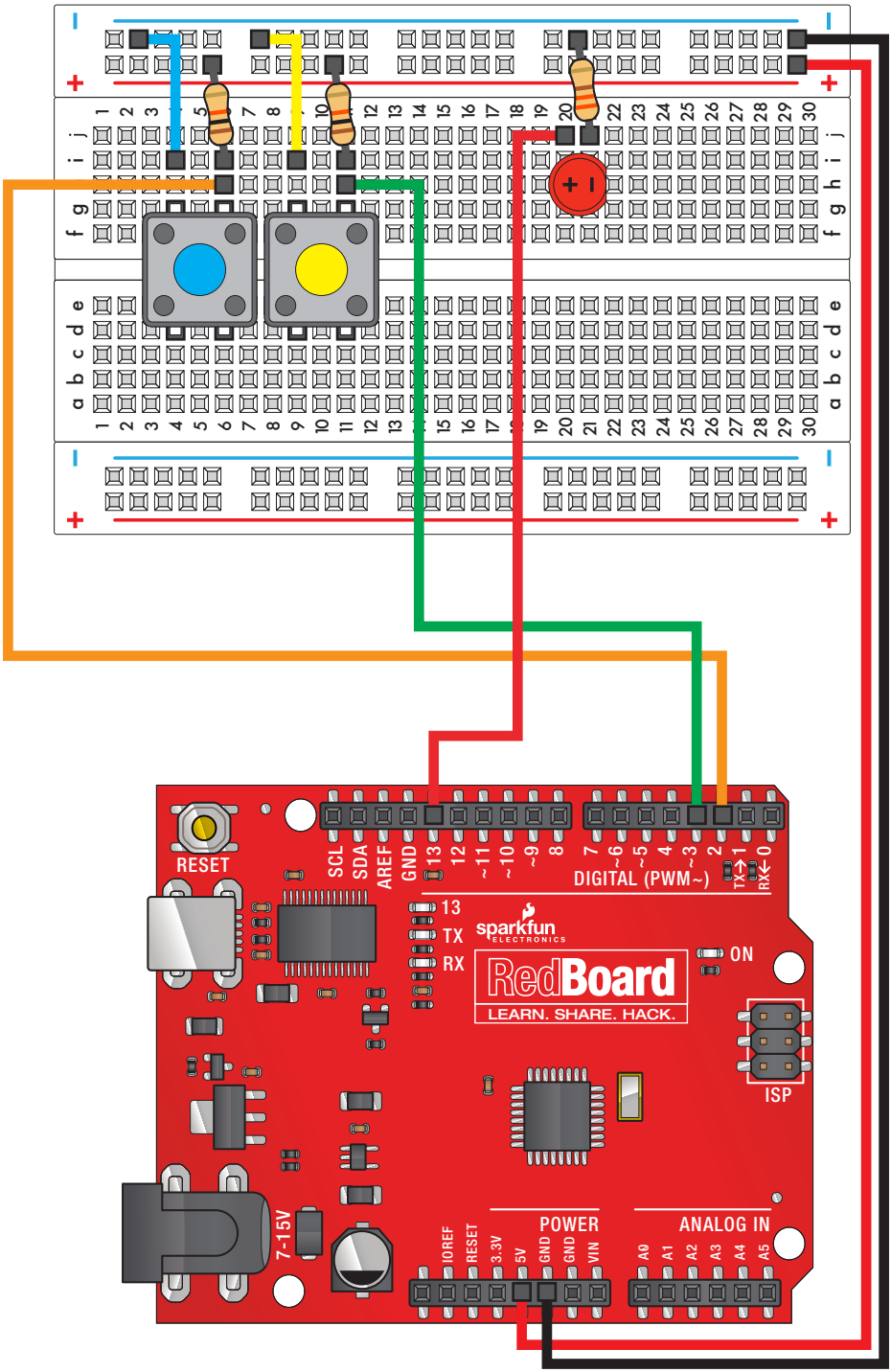


Up until now, we've focused solely on outputs. Now we're going to go to the other end of spectrum and play around with inputs. In this circuit, we'll be looking at one of the most common and simple inputs – a push button. The way a push button works with RedBoard is that when the button is pushed, the voltage goes LOW. The RedBoard reads this and reacts accordingly. In this circuit, you will also use a pull-up resistor, which keeps the voltage HIGH when you're not pressing the button.

PARTS:

Push Button	LED	10KΩ Resistor	330Ω Resistor	Wire
x2	x1	x2	x1	x7





Circuit 5: Push Buttons

Component:	Image Reference:		
Push Button			d4 g4 d6 g6
Push Button			d9 g9 d11 g11
LED (5mm) ⚠			h20 - h21 + + -
10KΩ Resistor			i6 - +
10KΩ Resistor			i11 - +
330Ω Resistor			j21 - -
Jumper Wire			i4 - -
Jumper Wire			i9 - -
Jumper Wire		Pin 2	h6
Jumper Wire		Pin 3	h11
Jumper Wire		Pin 13	j20
Jumper Wire		5V	+
Jumper Wire		GND	-

How to use logic like a Vulcan:

One of the things that makes the RedBoard so useful is that it can make complex decisions based on the input it's getting. For example, you could make a thermostat that turns on a heater if it gets too cold, a fan if it gets too hot, waters your plants if they get too dry, etc.

In order to make such decisions, the Arduino environment provides a set of logic operations that let you build complex "if" statements. They include:

==	EQUIVALENCE	**A == B** is true if A and B are the **SAME.**
!=	DIFFERENCE	**A != B** is true if A and B are **NOT THE SAME.**
&&	AND	**A && B** is true if **BOTH** A and B are **TRUE.**
||	OR	**A || B** is true if **A** or **B** or **BOTH** are **TRUE.**
!	NOT	**!A** is **TRUE** if A is **FALSE.** **!A** is **FALSE** if A is **TRUE.**

You can combine these functions to build complex if() statements.

For example:

```
if ((mode == heat) && ((temperature < threshold) || (override == true)))
{
digitalWrite(HEATER, HIGH);
}
```

...will turn on a heater if you're in heating mode **AND** the temperature is low, **OR** if you turn on a manual override. Using these logic operators, you can program your RedBoard to make intelligent decisions and take control of the world around it!

5

Arduino Code:

Open Arduino IDE // File > Examples > SIK Guide > Circuit # 5

Code to Note:

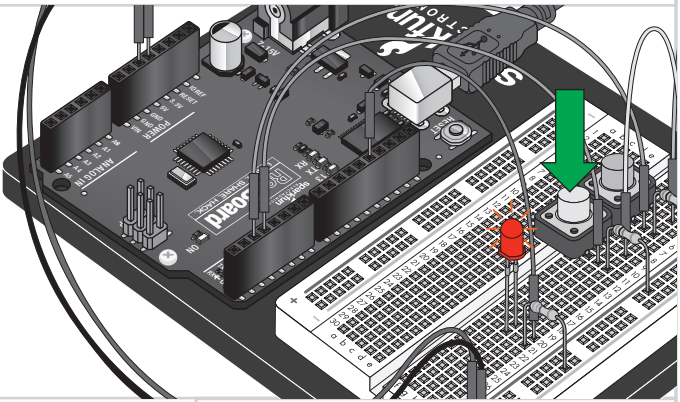
pinMode(button2Pin, INPUT); ⇨ The digital pins can be used as inputs as well as outputs. Before you do either, you need to tell the RedBoard which direction you're going.

button1State = digitalRead(button1Pin); ⇨ To read a digital input, you use the digitalRead() function. It will return HIGH if there's 5V present at the pin, or LOW if there's 0V present at the pin.

if (button1State == LOW) ⇨ Because we've connected the button to GND, it will read LOW when it's being pressed. Here we're using the "equivalence" operator ("==") to see if the button is being pressed.

What You Should See:

You should see the LED turn on if you press either button, and off if you press both buttons. (See the code to find out why!) If it isn't working, make sure you have assembled the circuit correctly and verified and uploaded the code to your board or see the troubleshooting tips below.



Troubleshooting:

Light Not Turning On
The pushbutton is square, and because of this it is easy to put it in the wrong way. Give it a 90 degree twist and see if it starts working.

Underwhelmed
No worries, these circuits are all super stripped down to make playing with the components easy, but once you throw them together the sky is the limit.

Real World Application:

The buttons we used here are similar to the buttons in most video game controllers.