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

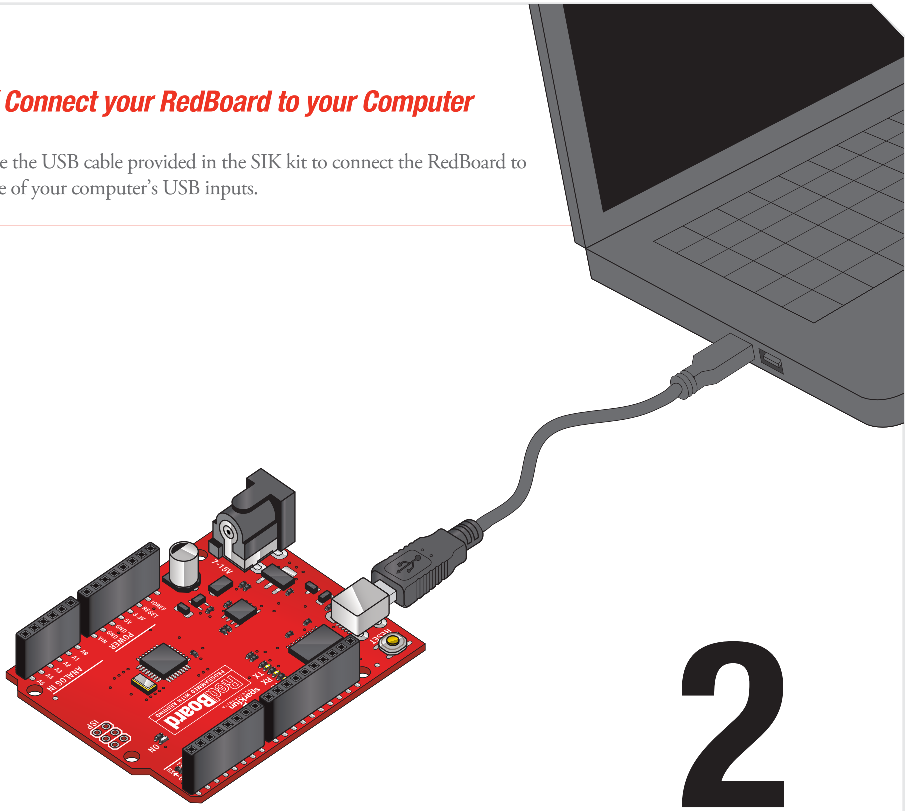
Download	**Click on your appropriate computer operating system next to the " + " sign.**

- ✚ **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.

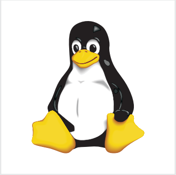


2

3

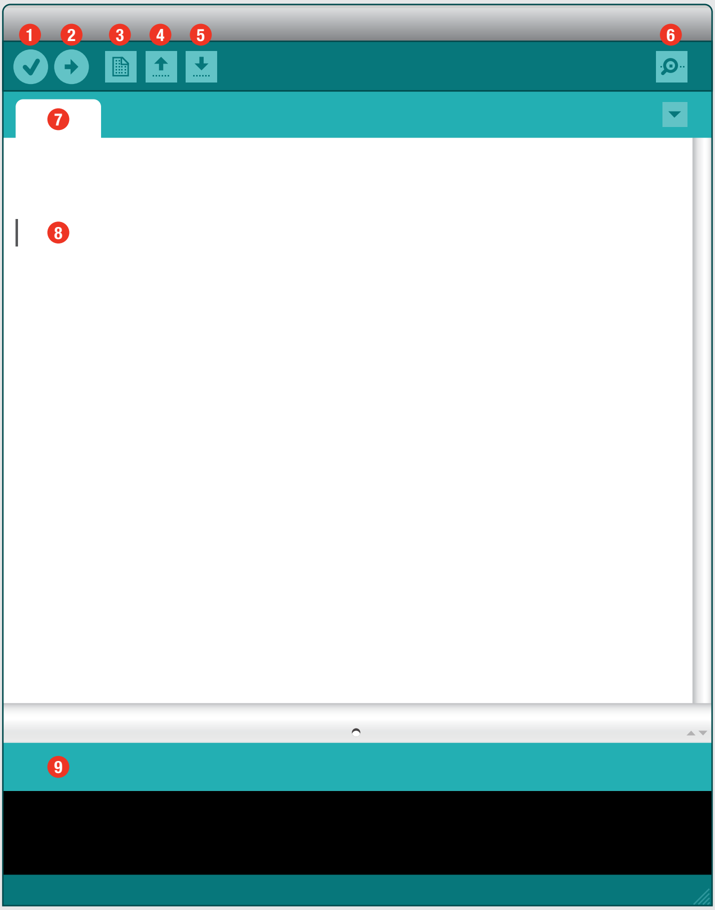
// Install Arduino Drivers

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



// Open the Arduino IDE:

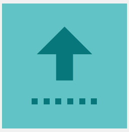
Open the Arduino IDE software on your computer. Poke around and get to know the interface. We aren't going to code right away, this is just an introduction. This step is to set your IDE to identify your RedBoard.



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:

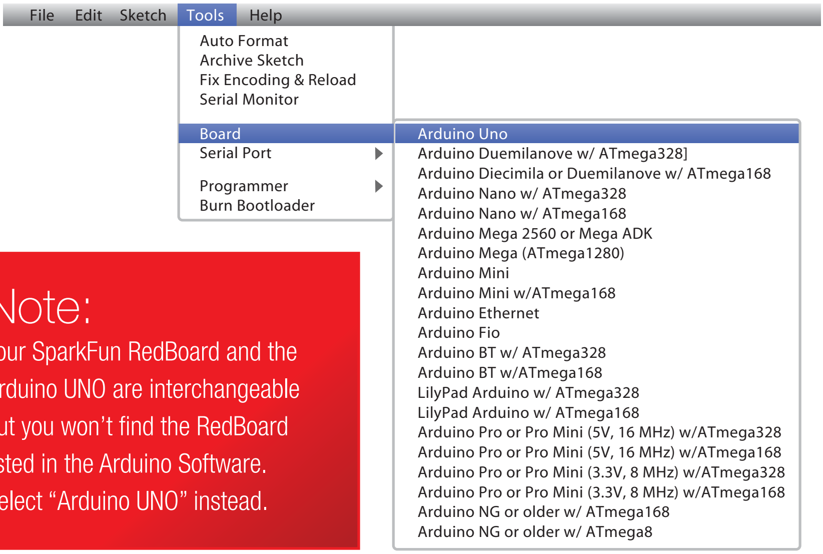


Open

Verify

Upload

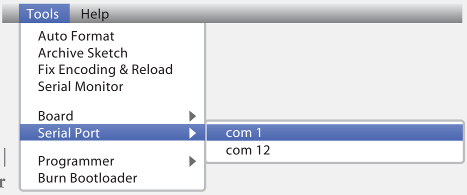
4 // Select your board: Arduino Uno


File
Edit
Sketch
Tools
Help
Auto Format
Archive Sketch
Fix Encoding & Reload
Serial Monitor
Board
Serial Port
Programmer
Burn Bootloader
Arduino Uno
Arduino Duemilanove w/ ATmega328]
Arduino Diecimila or Duemilanove w/ ATmega168
Arduino Nano w/ ATmega328
Arduino Nano w/ ATmega168
Arduino Mega 2560 or Mega ADK
Arduino Mega (ATmega1280)
Arduino Mini
Arduino Mini w/ATmega168
Arduino Ethernet
Arduino Fio
Arduino BT w/ ATmega328
Arduino BT w/ATmega168
LilyPad Arduino w/ ATmega328
LilyPad Arduino w/ ATmega168
Arduino Pro or Pro Mini (5V, 16 MHz) w/ATmega328
Arduino Pro or Pro Mini (5V, 16 MHz) w/ATmega168
Arduino Pro or Pro Mini (3.3V, 8 MHz) w/ATmega328
Arduino Pro or Pro Mini (3.3V, 8 MHz) w/ATmega168
Arduino NG or older w/ ATmega168
Arduino NG or older w/ ATmega8

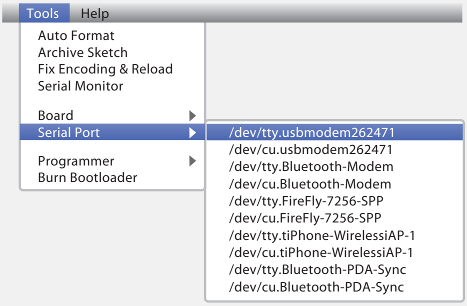
Note:

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.

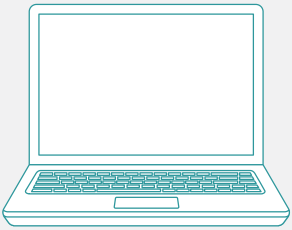


Select the serial device of the RedBoard from the Tools > Serial Port menu. On the Mac, this should be something with **/dev/tty.usbmodem** or **/dev/tty.usbserial** in it.

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

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

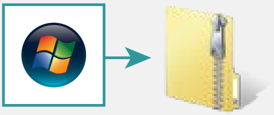
5



Type in the following URL to download the code:

sparkfun.com/sikcode

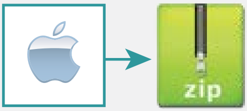
// Copy “SIK Guide Code” into “Examples” library in Arduino folder



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

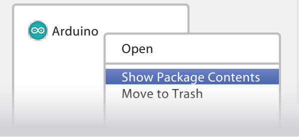
Start → Programs → arduino → examples

Copy the “SIK Guide Code” folder into Arduino’s folder named “examples”.



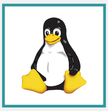
Unzip the file **“SIK Guide Code”**. It should be loacted in your browser’s **“Downloads”** folder. Right click the zipped folder and choose **“unzip”**.

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



Contents
↓
Resources
↓
Java
↓
examples

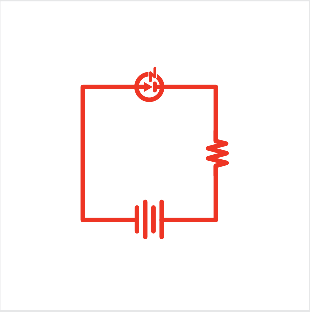
Copy the “SIK Guide Code” folder into Arduino’s folder named “examples”.



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



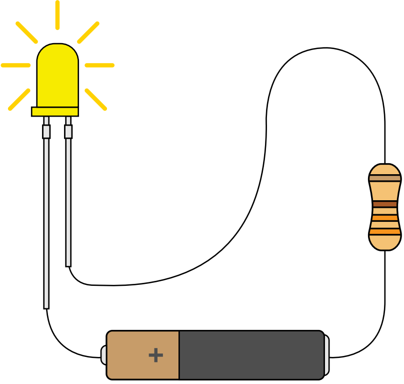
What is an Electrical Circuit?

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.

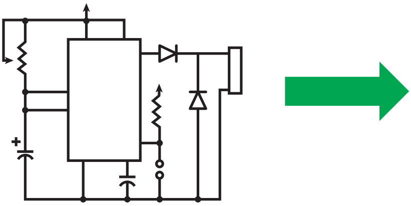
The World Runs on Circuits:

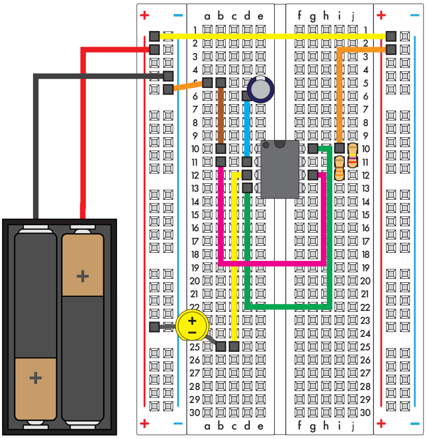
Everywhere you look, you'll find circuits. The cell phone in your pocket, the computer that controls your car's emissions system, your video game console - all these things are chock full of circuits. In this guide, you'll experiment with some simple circuits and learn the gist of the world of embedded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

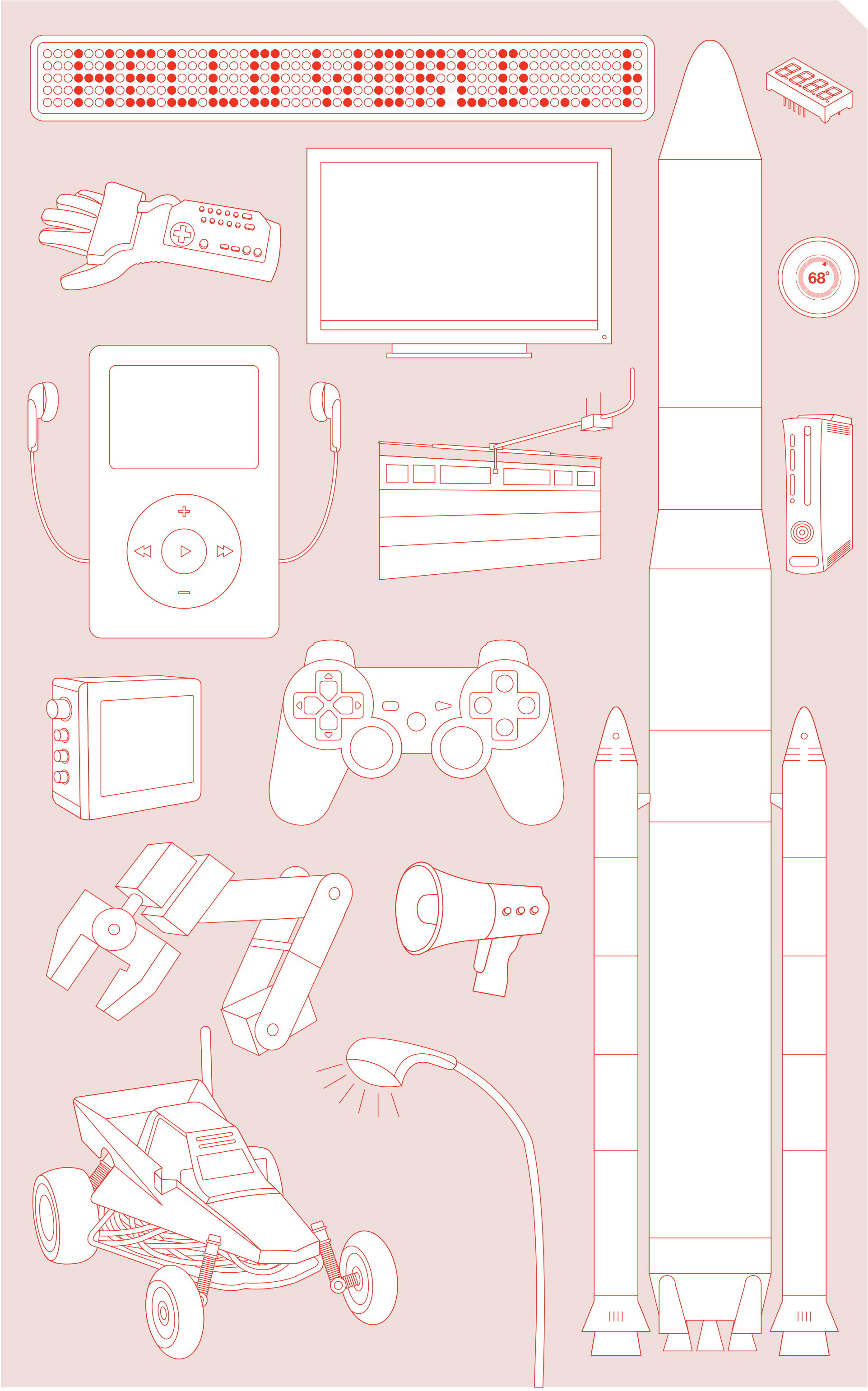


// Simple and Complex Circuits

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.



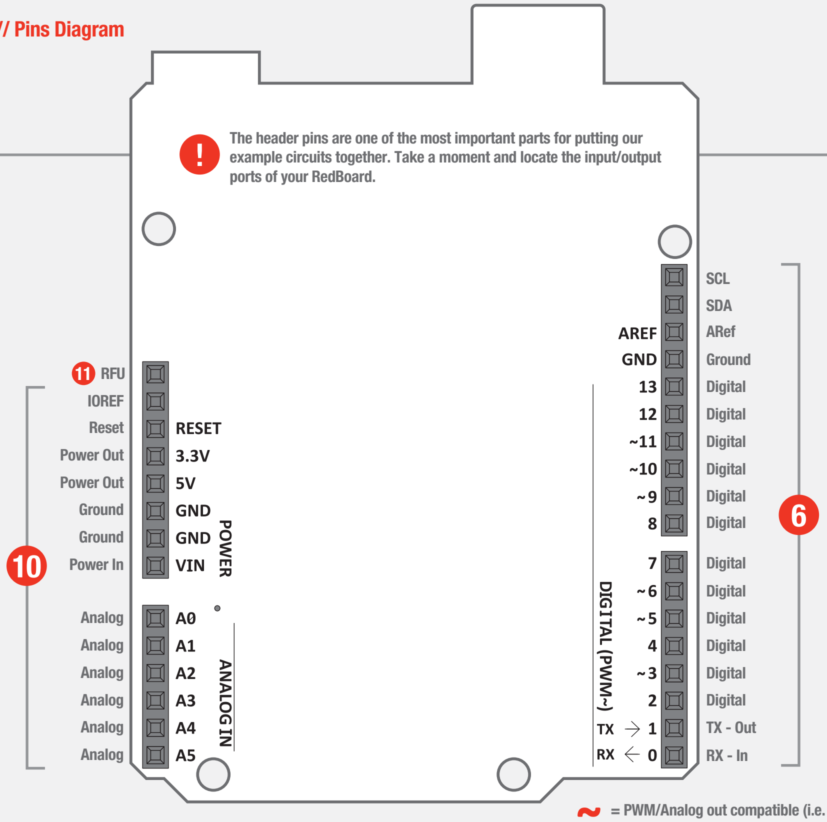



HELLO WORLD...
68°

SparkFun RedBoard

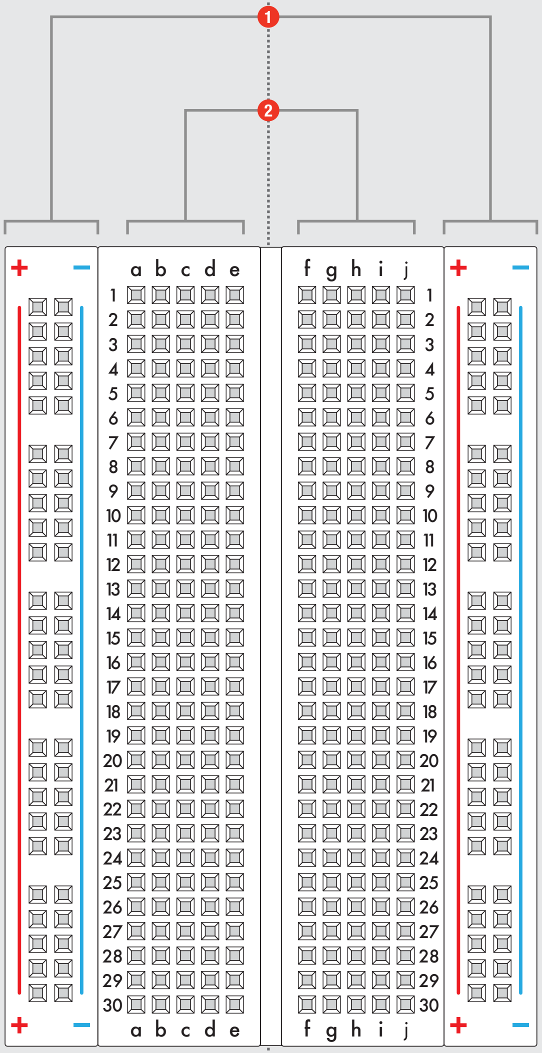
1. **Power In** (Barrel Jack) - Can be used with either a 9V or 12V wall-wart or battery.
2. **Power In** (USB Port) - Provides power and communicates with your board when plugged into your computer via USB.
3. **LED** (RX: Receiving) - This shows when the FTDI chip is receiving data bits from the microcontroller. 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.

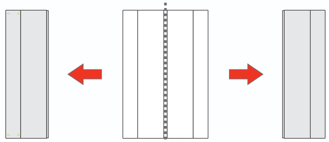
// Pins Diagram



! The header pins are one of the most important parts for putting our example circuits together. Take a moment and locate the input/output ports of your RedBoard.

~ = PWM/Analog out compatible (i.e. ~3)



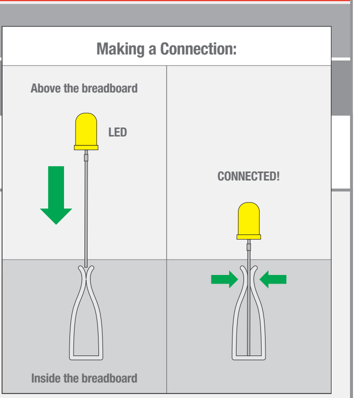


This line divides the board in half, restricting electricity to one half or the other.

Breadboard

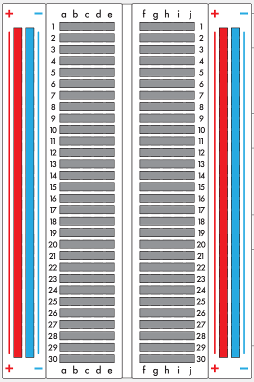
1 **Vertical Connection** (+ Power and - Ground) - Power bus // See Diagram Below

2 **Horizontal Connection** (a-e & f-j) // See Diagram Below)



Making a Connection:

How’s it all connected?



+ Power:

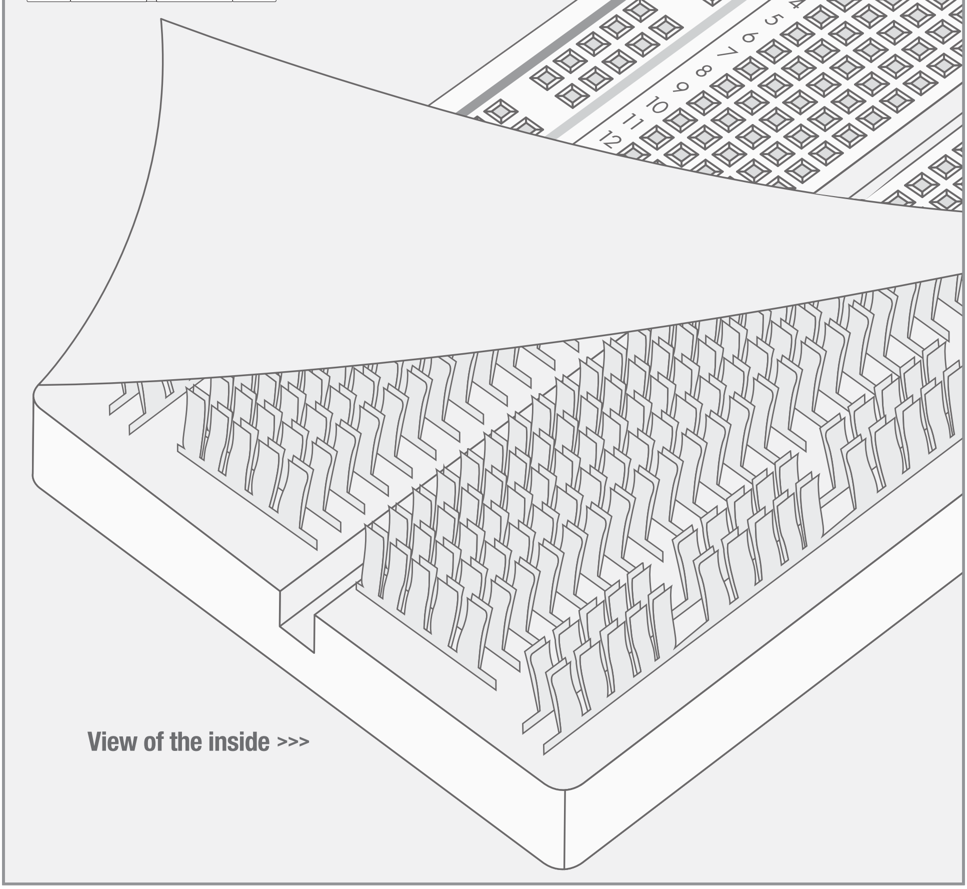
Each + sign runs power anywhere in the vertical column.

− 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.

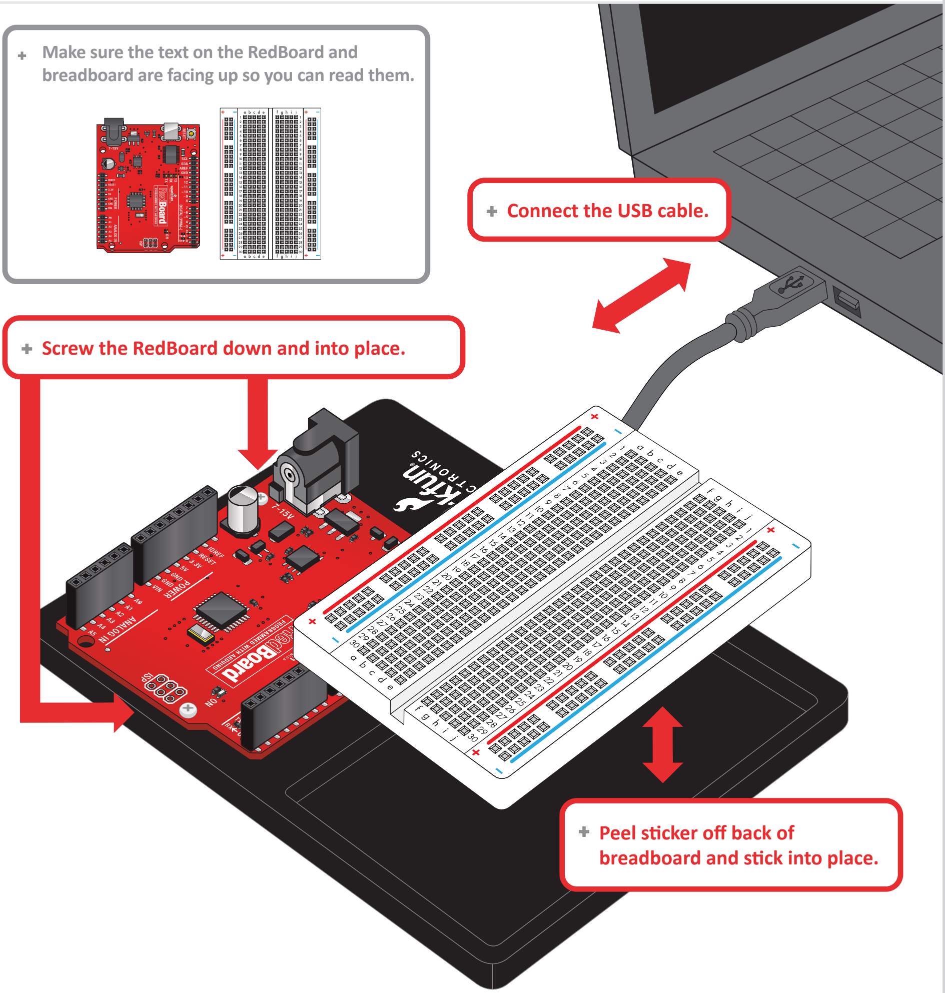


View of the inside >>>

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.

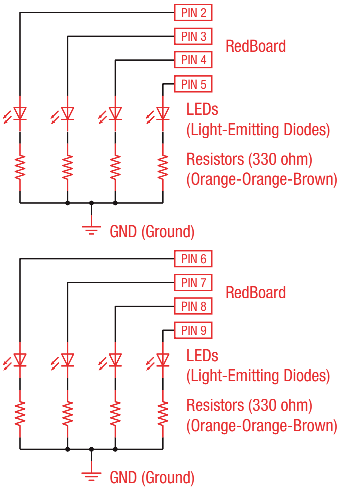
+ Screw the RedBoard down and into place.

+ 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.

CIRCUIT #4

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

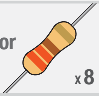
arrays[ ] - used to make managing variables easier by grouping them together

PARTS:

LED

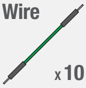
x 8

330Ω Resistor

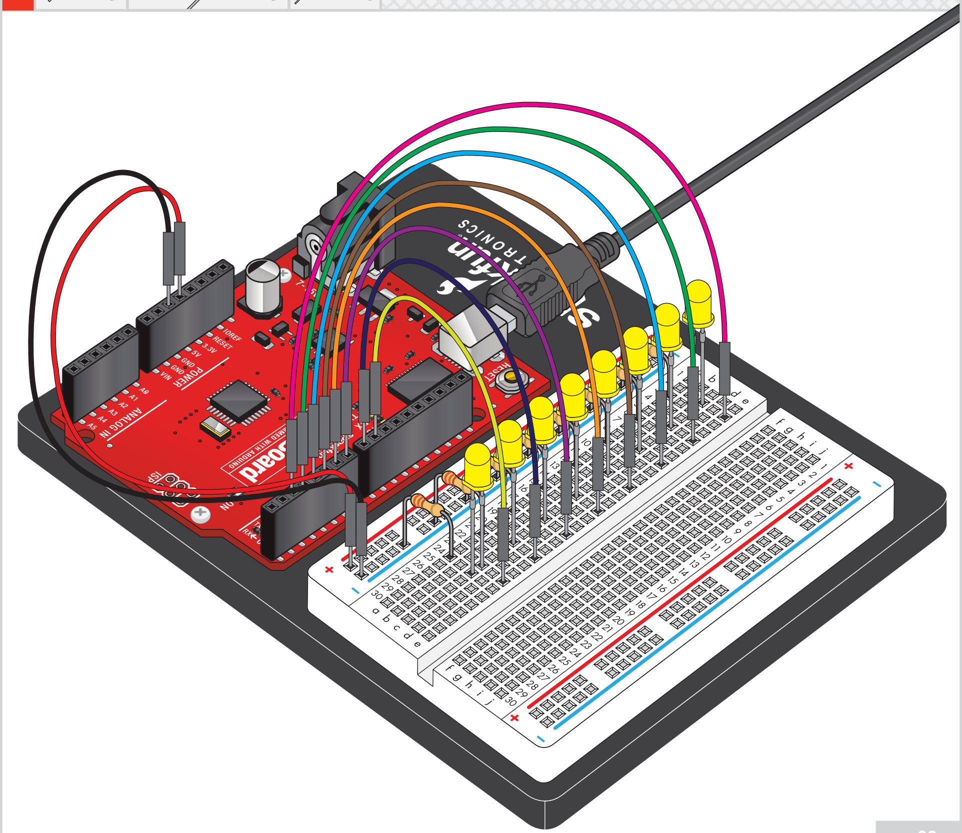


x 8

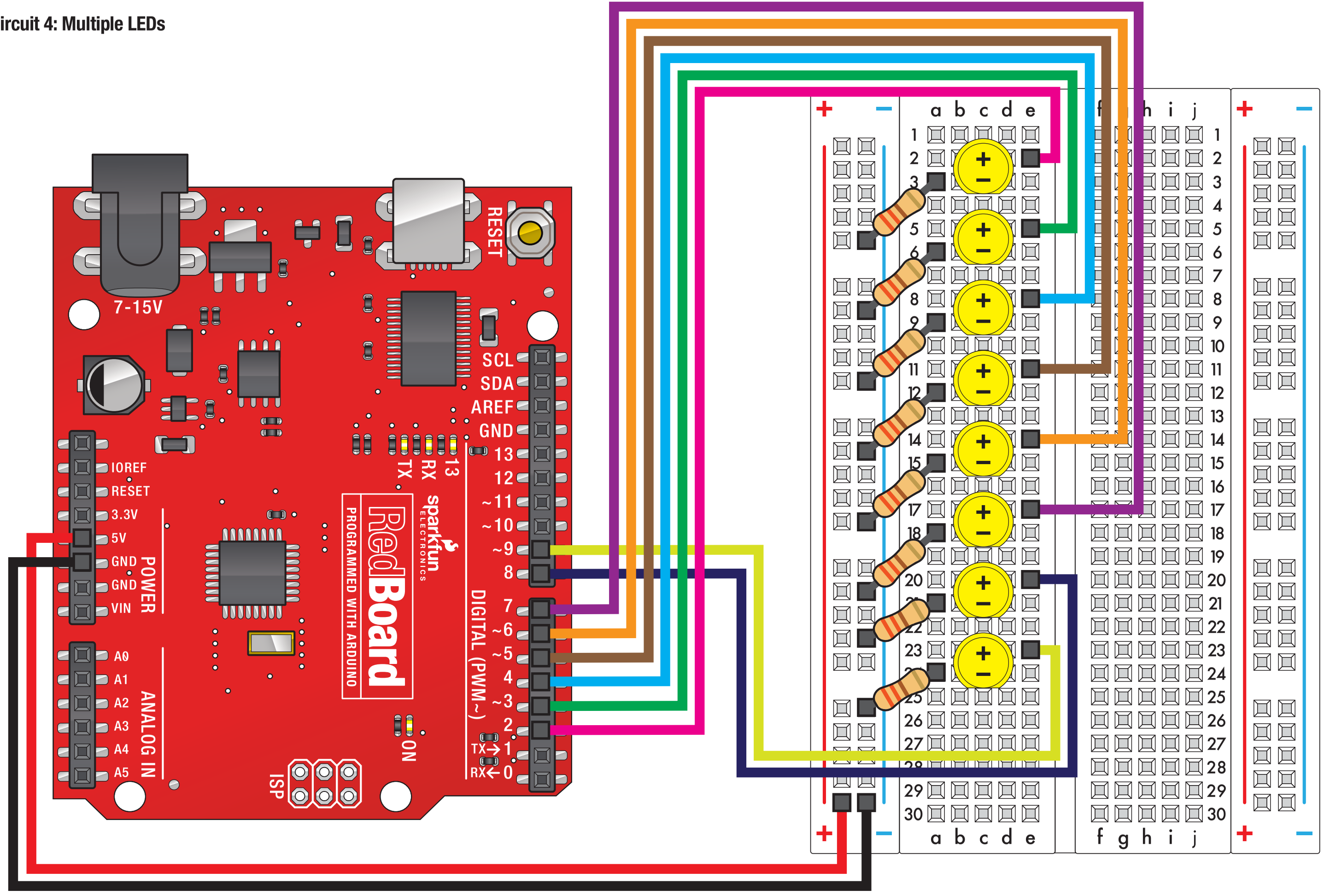
Wire

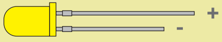
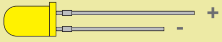
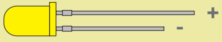
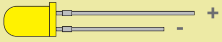
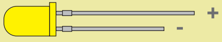
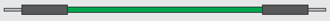
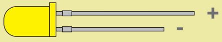
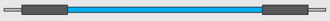
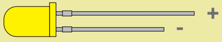
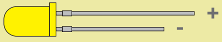
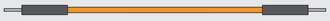
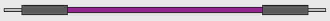


x 10



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);
```

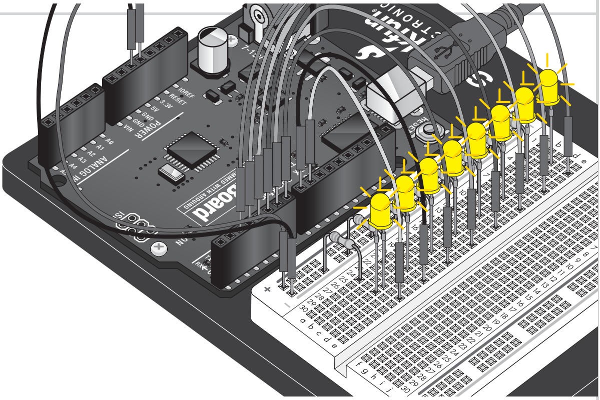
You refer to the elements in an array by their position. The first element is at position 0, the second is at position 1, etc. You refer to an element using "ledPins[x]" where x is the position. Here we're making digital pin 2 HIGH, since the array element at position 0 is "2".

```
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.