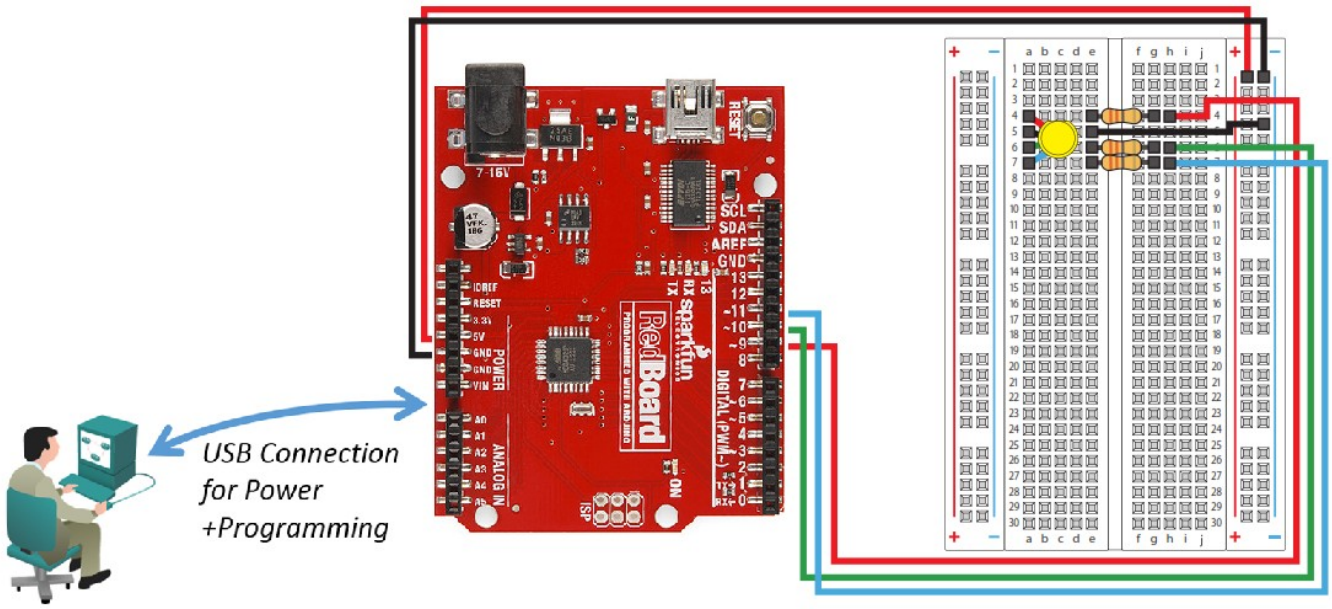
Lab – RGB LED using RedBoard and Arduino IDE

Topology



Objectives

Part 1: Set up the RGB circuit on the breadboard

- **Connect the circuits between the breadboard and RedBoard**
- **Load and run the sketch in the Arduino IDE**

Background / Scenario

Arduino, with its open source hardware prototyping platform, started a revolution in the maker community. Access to electronics, control, and programming has dramatically simplified and revolutionized how “smart devices” are built. The RedBoard included in the SparkFun Inventors Kit is an Arduino compatible board that can be used with the Arduino IDE installed previously. In this lab, you will learn to use the RedBoard and Arduino IDE to change the colors of an RGB LED.

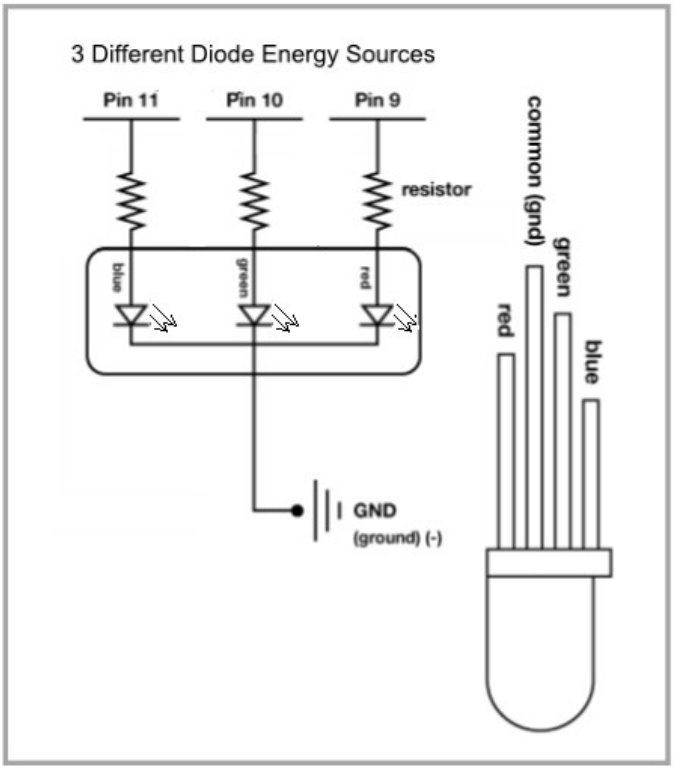
Inside of the SparkFun Inventors Kit you will find all the resources needed to start to prototype smart things. Arduino boards are able to read inputs - a light on a sensor, a finger on a button, or a Twitter message - and turn it into an output - activating a motor, turning on an LED, or publishing something online. All this is defined by a set of instructions programmed through the Arduino Software (Arduino IDE).

Required Resources

- SparkFun Inventors Kit or equivalent components
- PC configured with Arduino drivers and Arduino software
- SparkFun SIK Guide Code example files

Part 1: Set up the RGB Circuit on the Breadboard.

This lab will make use of a special RGB LED to change colors based on varying voltages from the RedBoard. The RGB LED has four leads, one is a common ground and the other three connect to 3 separate energy sources as depicted in the diagram. The pins that are used on the RedBoard are noted on the diagram, Pin 9 for red, Pin 10 for green and Pin 11 for blue. These three pins along with Pins 3,5, and 6 have the ability to simulate analog output and vary the voltage individually using Pulse Wave Modulation. Each of the lines with voltage applied requires a 330 Ohm (Ω) resistor.



Step 1: Connect the circuits between the breadboard and RedBoard.

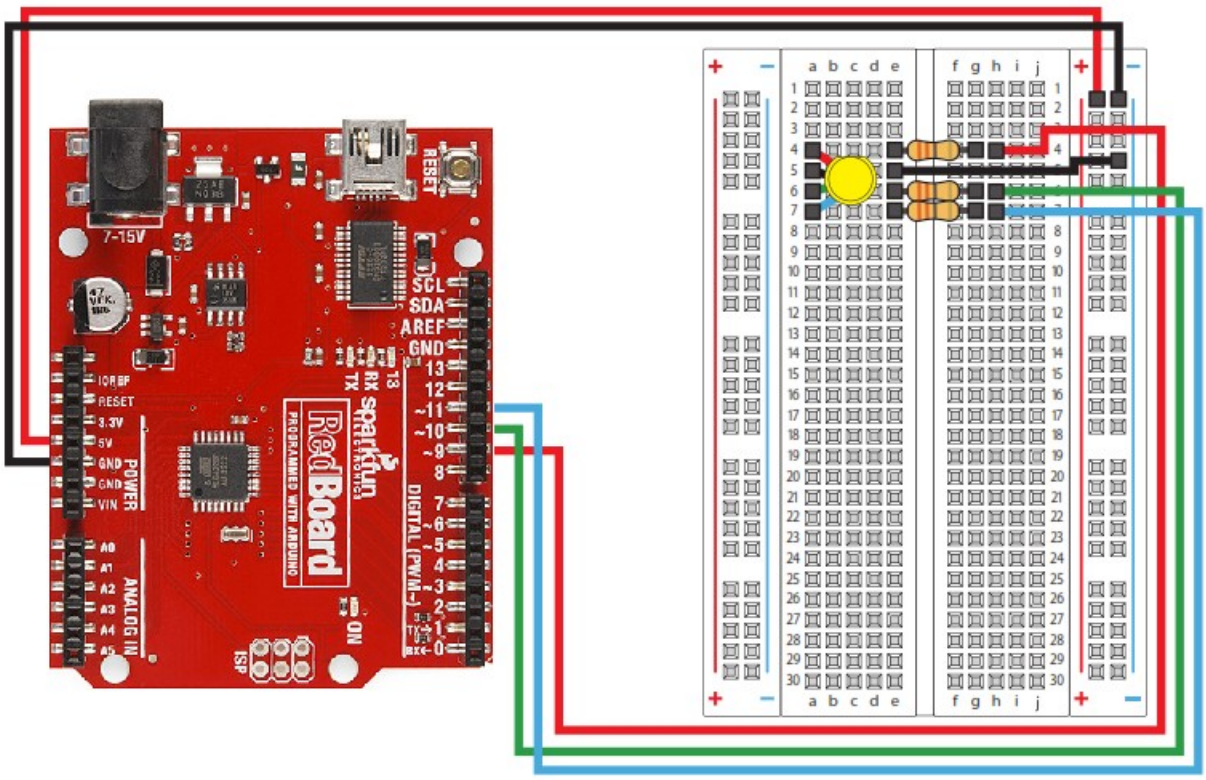
a. Select 3 - 330 Ohm (Ω) resistors, 2 red wire leads, 2 black wire leads, 1 green wire lead, 1 blue wire lead and the RGB LED.

b. Insert the **resistors** with one resistor connected to **e4** and **g4**, the second resistor connected to **e6** and **g6**, and the third resistor connected to **e7** and **g7**. It doesn't make any difference which lead goes in which hole for the resistors.

c. Insert the **RGB LED** making certain that the longest leg of the LED is inserted in **a5** on the breadboard. The remaining pins occupy **a4**, **a6** and **a7** in the same order as they are in the LED.

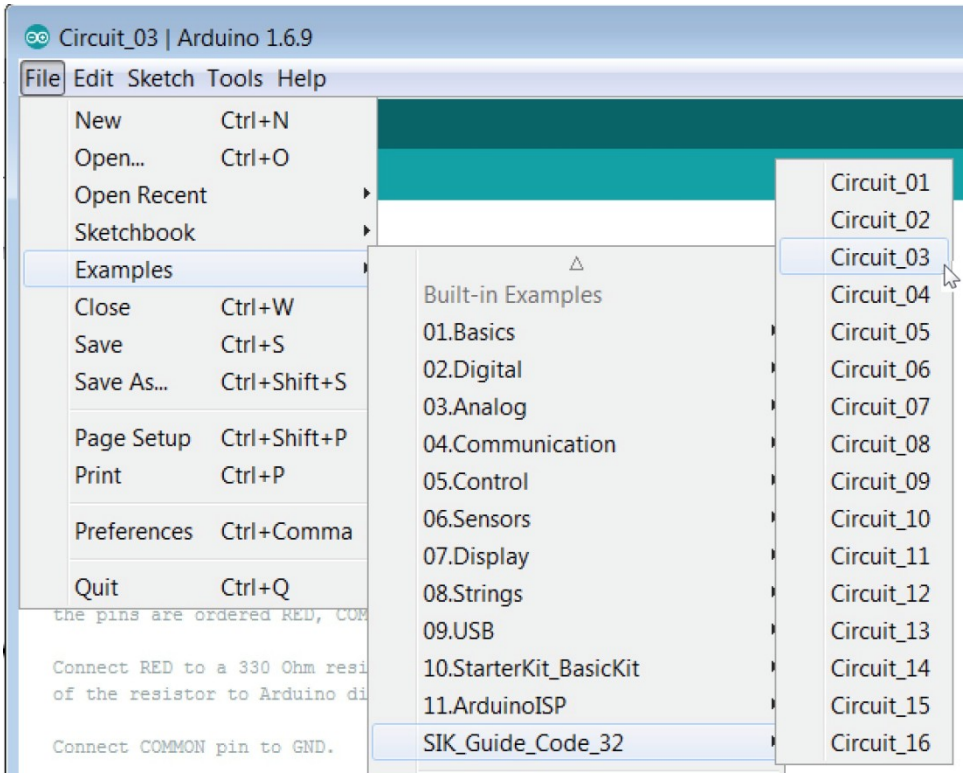
d. Connect one black wire lead from **e5** to the negative (**-**) voltage bar being used. Connect the wire leads from **h4**, **h6** and **h7** to the respective pins on the RedBoard as depicted using the same color scheme for clarity.

e. Connect the black (**-**) lead from the breadboard to the **GND** on the RedBoard and the red (**+**) lead from the breadboard to the 5V pin on the RedBoard.



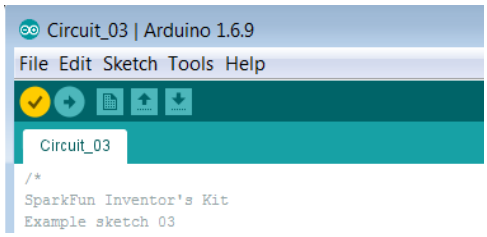
Step 2: Load the code in the Arduino IDE.

a. Click **File > Examples > SIK_Guide_Code_32 > Circuit_03** to load the sketch to control the RBG LED.



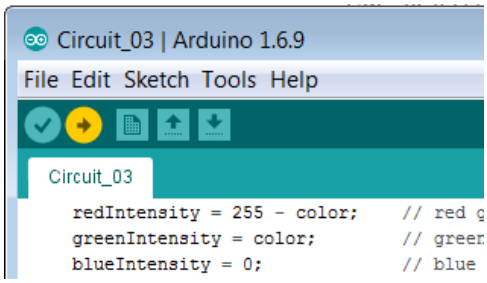
Note: The **SIK_Guide_Code_32** folder may need to be downloaded from http://sparkfun.com/sikcode and copied to the examples folder located under the Arduino program files, e.g. C:\Program Files\Arduino\examples.

b. Click the **Verify** icon to verify that the sketch does not have any coding issues that would prevent the compiler from being able to create an executable firmware that can be uploaded to the Flash memory of the microcontroller.




```
// There are a few new programming features introduced in this lab.
const int RED_LED_PIN = 9;
const int GREEN_LED_PIN = 10;
const int BLUE_LED_PIN = 11;
// The lines above make it easy to move wire leads to one of the other PWM pins
//  (3,5 or 6) by changing one value in the code.
int redIntensity = 0;
int greenIntensity = 0;
int blueIntensity = 0;
const int DISPLAY_TIME = 100;
// Another constant that would make varying the delay easy to adjust.
void setup() {
// No setup required but you still need it
}
void loop(){
  for (greenIntensity = 0; greenIntensity <= 255; greenIntensity+=5) {
      redIntensity = 255-greenIntensity;
      analogWrite(GREEN_LED_PIN, greenIntensity);
      analogWrite(RED_LED_PIN, redIntensity);
      delay(DISPLAY_TIME);
  }
  // The analogWrite functions above are simulating voltage changes for the LED
  //  color changes. This will only work on the PWM pins.
  for (blueIntensity = 0; blueIntensity <= 255; blueIntensity+=5) {
      greenIntensity = 255-blueIntensity;
      analogWrite(BLUE_LED_PIN, blueIntensity);
      analogWrite(GREEN_LED_PIN, greenIntensity);
      delay(DISPLAY_TIME);
  }
  for (redIntensity = 0; redIntensity <= 255; redIntensity+=5) {
      blueIntensity = 255-redIntensity;
      analogWrite(RED_LED_PIN, redIntensity);
      analogWrite(BLUE_LED_PIN, blueIntensity);
      delay(DISPLAY_TIME);
  }
}
```

c. Make sure the RedBoard is connected to the PC with the USB cable. Click the **Upload** icon to compile and upload the sketch to the RedBoard. After this step, the LED should continuously change colors between red, blue, green and many shades in between. If the RGB LED is not changing colors make sure all connections have been made as outlined and that every lead is making a good connection with the breadboard.



Note: The only way to stop the color change cycle is to remove power from the RedBoard. Until the RedBoard receives new code, this code will continue to loop as long as there is power to the board.

Reflection

Which variable setting would you change to increase or decrease the length of time it takes for the RBG LED to cycle through its color changes? Make changes to the value of this variable and upload the change to the red to verify your answer.

_____

DISPLAY_TIME_____
_____