# RoutinesRGB

v1.8.5

Generated by Doxygen 1.8.11

# Contents

# Chapter 1

# Module Index

## 1.1 Modules

Here is a list of all modules:

# Chapter 2

# Class Index

## 2.1 Class List

Here are the classes, structs, unions and interfaces with brief descriptions:

# Chapter 3

# Module Documentation

## 3.1 Getters and Setters

### Functions

- void RoutinesRGB::setMainColor (byte r, byte g, byte b)
- void RoutinesRGB::setColor (uint16_t colorIndex, byte r, byte g, byte b)
- void RoutinesRGB::setBrightness (uint8_t brightness)
- void RoutinesRGB::setFadeSpeed (uint8_t fadeSpeed)
- void RoutinesRGB::setBlinkSpeed (uint8_t blinkSpeed)
- Color RoutinesRGB::getMainColor ()
- Color RoutinesRGB::getColor (uint16_t i)
- uint8_t RoutinesRGB::getR (uint16_t i)
- uint8_t RoutinesRGB::getG (uint16_t i)
- uint8_t RoutinesRGB::getB (uint16_t i)

### 3.1.1 Detailed Description

These are the getters and setters for RoutinesRGB that are used to control the settings and the colors.

### 3.1.2 Function Documentation

#### 3.1.2.1 uint8_t RoutinesRGB::getB ( uint16_t *i* )

Retrieve the b value at a given index in the buffer.

#### 3.1.2.2 RoutinesRGB::Color RoutinesRGB::getColor ( uint16_t *i* )

Retrieve the color at the given index.

#### 3.1.2.3 uint8_t RoutinesRGB::getG ( uint16_t *i* )

Retrieve the g value at a given index in the buffer.

#### 3.1.2.4 RoutinesRGB::Color RoutinesRGB::getMainColor ( )

Retrieve the main color, which is used for single color routines.

#### 3.1.2.5 uint8_t RoutinesRGB::getR ( uint16_t *i* )

Retrieve the r value at a given index in the buffer.

#### 3.1.2.6 void RoutinesRGB::setBlinkSpeed ( uint8_t *blinkSpeed* )

Sets how many updates to wait before changing the light state in the blink routine and in routines that switch between solid colors.

**Parameters**

| | |
|---|---|
| *blinkSpeed* | a value between 1 and 255 representing how fast to blink. A value of 1 will make it blink on every frame, which may be too fast when used with other routines. |

#### 3.1.2.7 void RoutinesRGB::setBrightness ( uint8_t *brightness* )

Set the brightness between 0 and 100. 0 is off, 100 is full power.

#### 3.1.2.8 void RoutinesRGB::setColor ( uint16_t *colorIndex,* byte *r,* byte *g,* byte *b* )

Set the color at a given index with the RGB values provided. colorIndex must be less than the colorCount provided to the constructor or else it will not have any effect.

#### 3.1.2.9 void RoutinesRGB::setFadeSpeed ( uint8_t *fadeSpeed* )

Sets the speed of routines that fade between colors between 1 and 100. A fade speed of 1 is the slowest possible fade.

#### 3.1.2.10 void RoutinesRGB::setMainColor ( byte *r,* byte *g,* byte *b* )

Sets the color used for single color routines.

## 3.2 Single Color Routines

### Functions

- void RoutinesRGB::solid (uint8_t red, uint8_t green, uint8_t blue)
- void RoutinesRGB::blink (uint8_t red, uint8_t green, uint8_t blue)
- void RoutinesRGB::fade (uint8_t red, uint8_t green, uint8_t blue, uint8_t fadeSpeed, boolean shouldUpdate)
- void RoutinesRGB::glimmer (uint8_t red, uint8_t green, uint8_t blue, long percent, boolean shouldUpdate)

### 3.2.1 Detailed Description

These routines each take a R, G, and B value as parameters to generate a color. This color is the only color used by the routine.

Blink, fade, and glimmer, should be called repeatedly on a loop for their full effect. The speed of the loop determines how fast the LEDs update.

### 3.2.2 Function Documentation

#### 3.2.2.1 void RoutinesRGB::blink ( uint8_t *red,* uint8_t *green,* uint8_t *blue* )

Switches between ON and OFF states using the provided color.

**Parameters**

| | |
|---|---|
| *red* | strength of red LED, between 0 and 255 |
| *green* | strength of green LED, between 0 and 255 |
| *blue* | strength of blue LED, between 0 and 255 |

#### 3.2.2.2 void RoutinesRGB::fade ( uint8_t *red,* uint8_t *green,* uint8_t *blue,* uint8_t *fadeSpeed,* boolean *shouldUpdate* )

Fades the LEDs on and off based on the provided color. Uses the parameter fadeSpeed to determine how fast to fade. A larger number leads to a slower fade.

**Parameters**

| | |
|---|---|
| *red* | strength of red LED, between 0 and 255 |
| *green* | strength of green LED, between 0 and 255 |
| *blue* | strength of blue LED, between 0 and 255 |
| *fadeSpeed* | how many ticks it takes to fade. Higher numbers are slower. |

#### 3.2.2.3 void RoutinesRGB::glimmer ( uint8_t *red,* uint8_t *green,* uint8_t *blue,* long *percent,* boolean *shouldUpdate* )

Set every LED to the provided color. A subset of the LEDs based on the percent parameter will be less bright than the rest of the LEDs.

**Parameters**

| | |
|---|---|
| *red* | strength of red LED, between 0 and 255 |
| *green* | strength of green LED, between 0 and 255 |
| *blue* | strength of blue LED, between 0 and 255 |
| *percent* | determines how many LEDs will be slightly dimmer than the rest |

#### 3.2.2.4 void RoutinesRGB::solid ( uint8_t *red*, uint8_t *green*, uint8_t *blue* )

Set every LED to the provided color.

**Parameters**

| | |
|---|---|
| *red* | strength of red LED, between 0 and 255 |
| *green* | strength of green LED, between 0 and 255 |
| *blue* | strength of blue LED, between 0 and 255 |

## 3.3 Multi Color Routines

### Functions

- void RoutinesRGB::randomSolid ()
- void RoutinesRGB::randomIndividual ()
- void RoutinesRGB::fadeBetweenAllColors ()

### 3.3.1 Detailed Description

These routines use their own set of predefined colors and require no additional parameters.

Blink, fade, and glimmer, should be called repeatedly on a loop for their full effect. The speed of the loop determines how fast the LEDs update.

### 3.3.2 Function Documentation

#### 3.3.2.1 void RoutinesRGB::fadeBetweenAllColors ( )

Fade through all the colors in the rainbow!

#### 3.3.2.2 void RoutinesRGB::randomIndividual ( )

Each LED gets assigned a different random value for its R,G, and B components. This results in a lot of pastel colored lights and an overall white glow.

#### 3.3.2.3 void RoutinesRGB::randomSolid ( )

A random color is chosen and applied to each LED.

## 3.4 Array Colors Routines

### Functions

- void RoutinesRGB::arrayGlimmer (uint16_t colorCount, long percent)
- void RoutinesRGB::arrayRandomIndividual (uint16_t colorCount)
- void RoutinesRGB::arrayRandomSolid (uint16_t colorCount)
- void RoutinesRGB::arrayFade (uint16_t colorCount)
- void RoutinesRGB::arrayBarSolid (uint16_t colorCount, byte barSize)
- void RoutinesRGB::arrayBarMoving (uint16_t colorCount, byte barSize)

### 3.4.1 Detailed Description

These routines use the `colors` array as a basis for their colors. They all take the parameter of `colorCount` which is used to determine how many colors to use.

Blink, fade, and glimmer, should be called repeatedly on a loop for their full effect. The speed of the loop determines how fast the LEDs update.

### 3.4.2 Function Documentation

#### 3.4.2.1 void RoutinesRGB::arrayBarMoving ( uint16_t *colorCount*, byte *barSize* )

Provides a similar effect as arrayBarSolid, but the alternating patches move up one LED index on each frame update to create a "scrolling" effect.

**Parameters**

| | |
|---|---|
| *colorCount* | the number of colors in the array used for the routine. |
| *barSize* | how many LEDs before switching to the other bar. |

#### 3.4.2.2 void RoutinesRGB::arrayBarSolid ( uint16_t *colorCount*, byte *barSize* )

Uses the color array to set the LEDs in alternating patches with a size of barSize.

**Parameters**

| | |
|---|---|
| *colorCount* | the number of colors in the array used for the routine. |
| *barSize* | how many LEDs before switching to the other bar. |

#### 3.4.2.3 void RoutinesRGB::arrayFade ( uint16_t *colorCount* )

Fades between the number of colors in the array.

**Parameters**

| | |
|---|---|
| *colorCount* | the number of colors in the array used for the routine. |

#### 3.4.2.4 void RoutinesRGB::arrayGlimmer ( uint16_t *colorCount,* long *percent* )

This method uses its percent parameter to dim LEDs randomly, similar to the standard glimmer mode. It also uses the percent to randomly change the color of select LEDs to a color in the `colors` array. The base color is `color[0]` from the colors array.

**Parameters**

| | |
|---|---|
| *colorCount* | the number of colors in the array used for the routine. |
| *percent* | percent of LEDs that will get the glimmer applied |

#### 3.4.2.5 void RoutinesRGB::arrayRandomIndividual ( uint16_t *colorCount* )

sets each individual LED as a random color from the color array.

**Parameters**

| | |
|---|---|
| *colorCount* | the number of colors in the array used for the routine. |

#### 3.4.2.6 void RoutinesRGB::arrayRandomSolid ( uint16_t *colorCount* )

A random color is chosen from the color array and applied to each LED.

**Parameters**

| | |
|---|---|
| *colorCount* | the number of colors in the array used for the routine. |

# Chapter 4

# Class Documentation

## 4.1 RoutinesRGB Class Reference

An Arduino library that provides a set of RGB lighting routines for compatible LED array hardware.

```
#include <RoutinesRGB.h>
```

### Public Member Functions

- RoutinesRGB (uint16_t ledCount, uint16_t colorCount)
- void setMainColor (byte r, byte g, byte b)
- void setColor (uint16_t colorIndex, byte r, byte g, byte b)
- void setBrightness (uint8_t brightness)
- void setFadeSpeed (uint8_t fadeSpeed)
- void setBlinkSpeed (uint8_t blinkSpeed)
- Color getMainColor ()
- Color getColor (uint16_t i)
- uint8_t getR (uint16_t i)
- uint8_t getG (uint16_t i)
- uint8_t getB (uint16_t i)
- void solid (uint8_t red, uint8_t green, uint8_t blue)
- void blink (uint8_t red, uint8_t green, uint8_t blue)
- void fade (uint8_t red, uint8_t green, uint8_t blue, uint8_t fadeSpeed, boolean shouldUpdate)
- void glimmer (uint8_t red, uint8_t green, uint8_t blue, long percent, boolean shouldUpdate)
- void randomSolid ()
- void randomIndividual ()
- void fadeBetweenAllColors ()
- void arrayGlimmer (uint16_t colorCount, long percent)
- void arrayRandomIndividual (uint16_t colorCount)
- void arrayRandomSolid (uint16_t colorCount)
- void arrayFade (uint16_t colorCount)
- void arrayBarSolid (uint16_t colorCount, byte barSize)
- void arrayBarMoving (uint16_t colorCount, byte barSize)

### 4.1.1 Detailed Description

An Arduino library that provides a set of RGB lighting routines for compatible LED array hardware.

**Version**

v1.8.3

**Date**

March 27, 2016

**Author**

Tim Seemann

**Copyright**

`MIT License`

This library has been tested with SeeedStudio Rainbowduinos, quite a few of the Adafruit Neopixels products, and a standard RGB LED. Sample code is provided in the git repo for all tested hardware in the samples folder of the git repository.

If you are starting a project from scratch, first you'll need to make a global object in the arduino sketch:

```
RoutinesRGB routines = RoutinesRGB(LED_COUNT, COLOR_COUNT);
```

where `LED_COUNT` is the number of LEDs in your array, and `COLOR_COUNT` is the maximum number of colors you want to use in the array color routines.

After setting up the global object, it will be showing a solid green color with a glimmer by default. To update the colors, first call the proper functions to change it to the mode you want. For instance, to update to a red blinking light, call this function:

```
routines.blink(255, 0, 0);
```

Then, update the LED array with the values from the library's RGB buffer. The way to do this will vary from hardware to hardware, but for a NeoPixels sample, it would look something like this:

```
for (int x = 0; x < LED_COUNT; x++) {
   pixels.setPixelColor(x, pixels.Color(routines.getR(x),
                                        routines.getG(x),
                                        routines.getB(x)));
}
pixels.show();
```

Some routines, such as the random solid and all of the fades, change their values overtime. For these, put the routine's API call and the hardware update in your `loop()` and use your loop's update speed to determime how fast the LEDs change.

### 4.1.2 Constructor & Destructor Documentation

#### 4.1.2.1 RoutinesRGB::RoutinesRGB ( uint16_t *ledCount,* uint16_t *colorCount* )

Required constructor. This library can support as many LEDs and colors as your arduino's memory can support. The library should be stored in global memory and allocated only once at startup.

It will allocate `(4 * ledCount) + (3 * colorCount)` bytes.

**Parameters**

| | |
|---|---|
| *ledCount* | number of individual RGB LEDs. |
| *number* | of colors that are allocated for the array. |

# Index