RoutinesRGB

v1.9.0

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

File Index

3.1 File List

Here is a list of all documented files with brief descriptions:

Chapter 4

Module Documentation

4.1 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::setColorCount (uint8_t count)
- uint8_t RoutinesRGB::colorCount ()
- void RoutinesRGB::brightness (uint8_t brightness)
- void RoutinesRGB::fadeSpeed (uint8_t fadeSpeed)
- void RoutinesRGB::blinkSpeed (uint8_t blinkSpeed)
- Color RoutinesRGB::mainColor ()
- Color RoutinesRGB::color (uint16_t i)
- uint8_t RoutinesRGB::red (uint16_t i)
- uint8_t RoutinesRGB::green (uint16_t i)
- uint8_t RoutinesRGB::blue (uint16_t i)

4.1.1 Detailed Description

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

4.1.2 Function Documentation

4.1.2.1 void RoutinesRGB::blinkSpeed (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.

4.1.2.2 uint8_t RoutinesRGB::blue (uint16_t *i*)

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

4.1.2.3 void RoutinesRGB::brightness (uint8_t *brightness*)

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

4.1.2.4 RoutinesRGB::Color RoutinesRGB::color (uint16_t *i*)

Retrieve the color at the given index.

4.1.2.5 uint8_t RoutinesRGB::colorCount ( )

Retrieve the amount of colors that are used from the custom array.

4.1.2.6 void RoutinesRGB::fadeSpeed (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.

4.1.2.7 uint8_t RoutinesRGB::green (uint16_t *i*)

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

4.1.2.8 RoutinesRGB::Color RoutinesRGB::mainColor ( )

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

4.1.2.9 uint8_t RoutinesRGB::red (uint16_t *i*)

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

4.1.2.10 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.

4.1.2.11 void RoutinesRGB::setColorCount (uint8_t *count*)

Sets the amount of colors used in custom array routines. This is useful when you want to use a subset of the custom array. The value given must be less than the size of the custom array or else it will be set to use the entire array.

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

Sets the color used for single color routines.

4.2 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)

4.2.1 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.

4.2.2 Function Documentation

4.2.2.1 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

4.2.2.2 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.

4.2.2.3 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

4.2.2.4 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

4.3 Multi Colors Routines

Functions

- void RoutinesRGB::arrayGlimmer (EColorPreset preset, long percent)
- void RoutinesRGB::arrayFade (EColorPreset preset)
- void RoutinesRGB::arrayRandomIndividual (EColorPreset preset)
- void RoutinesRGB::arrayRandomSolid (EColorPreset preset)
- void RoutinesRGB::arrayBarsSolid (EColorPreset preset, byte barSize)
- void RoutinesRGB::arrayBarsMoving (EColorPreset preset, byte barSize)

4.3.1 Detailed Description

These routines use multiple colors. They all take the parameter of `preset` which is used to determine which set of colors to use. The custom color array is eCustom, all other values for `preset` come from predefined color groups. Go to the project's github for a full list of the presets and their corresponding values.

All routines except eArrayBarsSolid should be called repeatedly on a loop for their full effect. The speed of the loop determines how fast the LEDs update.

4.3.2 Function Documentation

4.3.2.1 void RoutinesRGB::arrayBarsMoving (EColorPreset *preset*, 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

preset	the color array to use for the routine. eCustom is the custom array, all other values are preset arrays.
barSize	how many LEDs before switching to the other bar.

4.3.2.2 void RoutinesRGB::arrayBarsSolid (EColorPreset *preset*, byte *barSize*)

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

Parameters

preset	the color array to use for the routine. eCustom is the custom array, all other values are preset arrays.
barSize	how many LEDs before switching to the other bar.

4.3.2.3 void RoutinesRGB::arrayFade (EColorPreset *preset*)

Fades between the number of colors in the array.

Parameters

preset	the color array to use for the routine. eCustom is the custom array, all other values are preset arrays.

4.3.2.4 void RoutinesRGB::arrayGlimmer (EColorPreset *preset,* 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 chosen array. The base color is the first from the chosen array.

Parameters

preset	the color array to use for the routine. eCustom is the custom array, all other values are preset arrays.
percent	percent of LEDs that will get the glimmer applied

4.3.2.5 void RoutinesRGB::arrayRandomIndividual (EColorPreset *preset*)

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

Parameters

preset	the color array to use for the routine. eCustom is the custom array, all other values are preset arrays

4.3.2.6 void RoutinesRGB::arrayRandomSolid (EColorPreset *preset*)

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

Parameters

preset	the color array to use for the routine. eCustom is the custom array, all other values are preset arrays.

Chapter 5

Class Documentation

5.1 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 resetToDefaults ()
- void setMainColor (byte r, byte g, byte b)
- void setColor (uint16_t colorIndex, byte r, byte g, byte b)
- void setColorCount (uint8_t count)
- uint8_t colorCount ()
- void brightness (uint8_t brightness)
- void fadeSpeed (uint8_t fadeSpeed)
- void blinkSpeed (uint8_t blinkSpeed)
- Color mainColor ()
- Color color (uint16_t i)
- uint8_t red (uint16_t i)
- uint8_t green (uint16_t i)
- uint8_t blue (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 arrayGlimmer (EColorPreset preset, long percent)
- void arrayFade (EColorPreset preset)
- void arrayRandomIndividual (EColorPreset preset)
- void arrayRandomSolid (EColorPreset preset)
- void arrayBarsSolid (EColorPreset preset, byte barSize)
- void arrayBarsMoving (EColorPreset preset, byte barSize)

5.1.1 Detailed Description

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

Version

v1.9.0

Date

April 24, 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 available in the custom color array.

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.red(x),
                                        routines.green(x),
                                        routines.blue(x)));
}
pixels.show();
```

Some routines, change their values over time. 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.

5.1.2 Constructor & Destructor Documentation

5.1.2.1 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.

5.1.3 Member Function Documentation

5.1.3.1 void RoutinesRGB::resetToDefaults ( )

Resets all internal values to the original values.

Chapter 6

File Documentation

6.1 /Users/tim/ArduinoAmy/RGB-LED-Routines/RoutinesRGB/LightingProtocols.h File Reference

Enumerations

6.1.1 Detailed Description

Version

v1.9.0

Date

April 24, 2016

Author

Tim Seemann

Copyright

This file defines the protocols used for the Arduino libraries and the GUI.

A slightly modified version of this file exists in the Qt GUI project. None of the modifications change the naming, documentation, or order of the protocols. Instead, the changes allow the GUI version to use the strongly typed enums that were made available in C++11.

6.1.2 Enumeration Type Documentation

6.1.2.1 enum EColorPreset

used during multi color routines to determine which colors to use in the routine. eCustom uses the custom color array, while all other values use presets based around overall themes.

Enumerator

eCustom **0**
Use the custom color array instead of a preset routine.

eWater **1**
Shades of blue with some teal.

eFrozen **2**
Shades of teal with some blue, white, and light purple.

eSnow **3**
Shades of white with some blue and teal.

eCool **4**
Based on the cool colors: blue, green, and purple.

eWarm **5**
Based on the warm colors: red, orange, and yellow.

eFire **6**
Similar to the warm set, but with an emphasis on oranges to give it a fire-like glow.

eEvil **7**
Mostly red, with some other, evil highlights.

eCorrorsive **8**
Greens and whites, similar to radioactive goo from a 90s kids cartoon.

ePoison **9**
A purple-based theme. Similar to poison vials from a 90s kids cartoon.

eRose **10**
Shades of pink, red, and white.

ePinkGreen **11**
The colors of watermelon candy. bright pinks and bright green.

eRedWhiteBlue **12**
Bruce Springsteen's favorite color scheme, good ol' red, white, and blue.

eAll **13**
Rather than using using presets, it uses all possible colors.

eRGB **14**
red, green, and blue.

eCMY **15**
Cyan, magenta, yellow.

eSixColor **16**
Red, yellow, green, cyan, blue, magenta.

eSevenColor **17**
Red, yellow, green, cyan, blue, magenta, white.

6.1.2.2 enum ELightingMode

The mode is for determining what LED routines to use. Some routines use a single color and others use multiple colors.

Enumerator

eOff **0**
Turns off the LEDs.

eSingleSolid **1**
Shows a single color at a fixed brightness.

eSingleBlink **2**
Alternates between showing a single color at a fixed brightness and turning the LEDs completely off.

eSingleFade **3**
Linear fade of the brightness of the LEDs.

eSingleGlimmer **4**
Single Color Routine, Randomly dims some of the LEDs to give a glimmer effect.

eMultiGlimmer **5**
Uses the first color of the array as the base color and uses the other colors for a glimmer effect.

eMultiFade **6**
Fades slowly between each color in the array.

eMultiRandomSolid **7**
Chooses a random color from the array and lights all all LEDs to match that color.

eMultiRandomIndividual **8**
Chooses a random color from the array for each individual LED.

eMultiBarsSolid **9**
Draws the colors of the array in alternating groups of equal size.

eMultiBarsMoving **10**
Draws the colors of the array in alternating groups of equal size. Then, on each update, it moves those groups one index to the right, creating a scrolling effect.

Index