Impact Factor (SJIF): 5.301

International Journal of Advance Research in Engineering, Science & Technology

e-ISSN: 2393-9877, p-ISSN: 2394-2444

Volume 5, Issue 9, September-2018

Development of RGB LED Library to create multiple colors using high frequency PWM pulses from Arduino

Mr. Vihang H Trivedi[1] Ms. Purvi Katmore[2]

[1 2]*Assistant Professor , Department of Electronics & Communication K.J. Institute of Engineering & Technology, Savli*

Abstract - *This paper is about how to develop library for RGBLED with possible 2^{24} combination of Red, Green and blue. RGB LED has two options either common anode or common cathode. When interfacing RGB LED with ARDUINO one can directly write a code without using library but if library is developed then more than one RGB LEDs can be interfaced with ARDUINO. All we need is to declare the RGB LEDs as objects. Here we are using PWM pulses at frequency 5000Hz. So Using PWM we can control intensity of Red, Blue and Green that is how we are able to create 2^{24} combinations 8-bit for each color. The library contains 3 main functions Setcolor(), on() and off().*

Key Words- , *RGB LED, Analog pulses to LED, Multiple colors, Library,header file , .cpp files*

I. INTRODUCTION

To develop an application that can display the color on RGB LED as per given color code, first thing is to write library for RGB LED. This paper explores section by section how to write library, followed by a common Arduino code describing interfacing with RGB LED. RGB LED require 3 pins to be connected with Arduino and 1 pin is to be connected with common ground. Arduino pins 9 to 11 are occupied for RGB LED in form of Blue, Green and Red respectively. The scenario is quite simple but development of library is the most critical part for Arduino. Without developing library, the code for the application is lengthy and hence ambiguous sometimes.

II. RGB LED pin outs and circuit

The pin out of RGB LED(Common Cathode) is shown in figure. Pin number 1 is for RED, 2 for ground, 3 for GREEN and 4 for BLUE. While connecting supplies to the RGB pins, a current limiting resistor of 220Ω must be placed with the RGB pins.

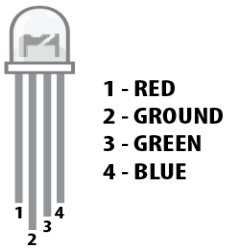

Figure 2.1 RGB LED pinout

The circuit diagram for interfacing RGB with Arduino is shown in figure 2.2. Pin number 1, 3 and 4 of RGB LED is connected with pin number 11, 10 and 9 of Arduino respectively through the current limiting resistors.Pin number 2 of RGB LED is connected with ground pin of Aurdino.

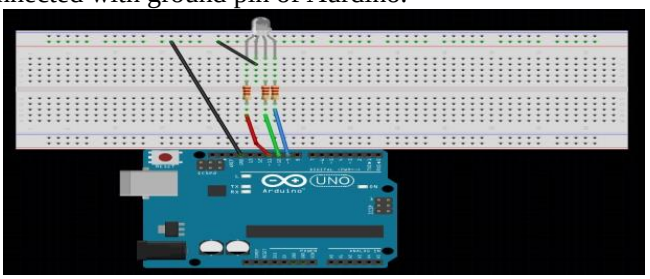

Figure 2.2 circuit diagram

Figure 2.3 Next 10 blocks of lower energy

III. LIBRARY DEVELOPMENT.

The library for interfacing RGB LED with the arduino is explained below.
The library contains one header file and one C++ file. In header files contains the functions for LED, Constructor and declaration of pins of Arduino to be interfaced with RGB LED.The C++ file contains the definition for constructor and functions. There are mainly 3 functions for RGB LED, namely, SetColor(Red value, Green Value, Blue Value), on() and off(). The function SetColor(Red value, Green Value, Blue Value) accepts 8-bit value ranging from 0 to 255 to be compatible with RGB color coding scheme. The on() and off() functions has no passing parameters. When on() is called the RGB LED glows white and when off() is called the RGB LED turns off. The intensity of LED can be controlled via current limiting resistors so no extra functions are needed.
The header file " RGBLed1.h "for the discussed library is mentioned below:

```
#ifndef RGBLed1_h
#define RGBLed1_h
#include "arduino.h"

class RGBLed1
{
public:
RGBLed1(int r_pin,int g_pin,int b_pin);
void SetColor(int r_val,int g_val,int b_val);
void on(void);
void off(void);
private:
int _r_pin,_g_pin,_b_pin;
};
#endif
```

As discussed above, the 3 functions are declared public so that it can be called via any object associated with RGBLed1 class. Constructor RGBLed1(int r_pin,int g_pin,int b_pin) is declared so that the programmer can select the pins of arduino which makes it programmer friendly. The pins are declared private so that the direct access to pin variable can be avoided and hence these private variables get values from constructor defined in C++ file.
The C++ file "RGBLed1.cpp" for the discussed library is mentioned below:

```
#include "arduino.h"
#include "RGBLed1.h"
RGBLed1::RGBLed1(int r_pin,int g_pin,int b_pin)
{
pinMode(r_pin,OUTPUT);
pinMode(g_pin,OUTPUT);
pinMode(b_pin,OUTPUT);
_r_pin=r_pin;
_g_pin=g_pin;
_b_pin=b_pin;
}
```

The above mentioned constructor will define pins as OUTPUT which can send PWM pulses to RGB LED. Private variables are assigned with values of the passing parameters so that Multiple objects can be declared. Hence more than one RGB LEDs can be interfaced with Arduino with ease of operation.

The following SetColor function accepts 3 parameters: 1. Red Value 2. Green Value 3. Blue Value. The values can be ranged from 0 to 255. These values are converted in form of duration between 0 to 200µSeconds. The mapping is simple. These mapped values are used to give high pulses to RGB LED for that duration and remaining time for low pulses. The idea of providing high frequency PWM pulses is to remove flickering when RGB LED glows. The generated pulses are sent to RGB LED for 1000 times to create a small amount of delay. Extra delays can be used by programmer while setting any color. The code for SetColor is mentioned below:

```
void RGBLed1::SetColor (char r_val,char g_val,char b_val)
{
char r,g,b;
int i;
r=(r_val*200)/256;
g=(g_val*200)/256;
```

```
b=(b_val*200)/256;
for (i=0;i<1000;i++)
{
delayMicroseconds(r);
digitalWrite(_r_pin,HIGH);
delayMicroseconds((200-r));
digitalWrite(_r_pin,LOW);
delayMicroseconds(g);
digitalWrite(_g_pin,HIGH);
delayMicroseconds((200-g));
digitalWrite(_g_pin,LOW);
delayMicroseconds(b);
digitalWrite(_b_pin,HIGH);
delayMicroseconds((200-b));
digitalWrite(_b_pin,LOW);
}
}
```

The "on()" function simply provides "HIGH" vlaues to all pins. That means RED+GREEN+BLUE=WHITE color is created. The function works as shown below. Also, if any pin of RGB LED is not connected properly or not working then WHITE color will not be created and therefore the programmer can identify the problem.

```
void RGBLed1::on()
{
digitalWrite(_r_pin,HIGH);
digitalWrite(_g_pin,HIGH);
digitalWrite(_b_pin,HIGH);
}
```

The "off()" function simply turns off the RGB LED by providing " LOW" value to RGB LED anode pins. The following code does that work easily.

```
void RGBLed1::off()
{
digitalWrite(_r_pin,LOW);
digitalWrite(_g_pin,LOW);
digitalWrite(_b_pin,LOW);
}
```

IV. Working

To use the library, an Arduino code and Arduino Uno are required. The code can be written in Arduino software where library is added previously for use and initialized in code so that the functions can be used. For example one RGB LED is interfaced so one object is created named "RGBL1". The code shown below elaborates the functionality :

```
#include "RGBLed1.h"
const int r=11;
const int g=10;
const int b=9;
RGBLed1 RGBL1(r,g,b);
```

There is no setup() function required, so it is left blank. The loop() section can be used to create as many colors as programmer wants. The following code will create some of colors using RGB color model.

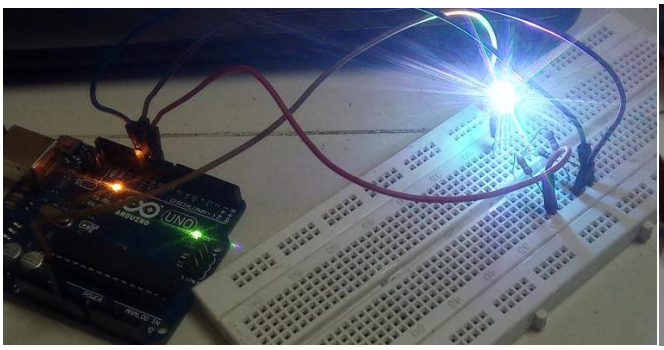

Figure 4.1 White Colored RGB LED

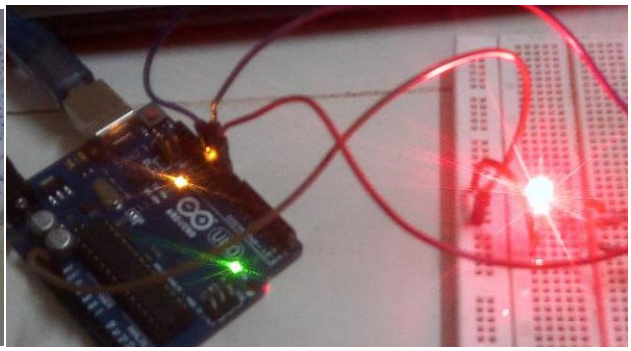

Figure 4.2 Red Colored RGB LED

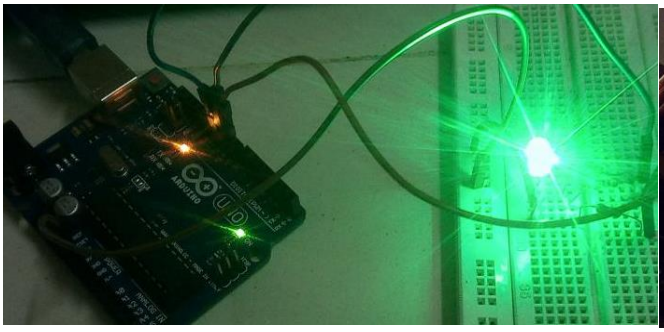

Figure 4.3 Green colored RGB LED

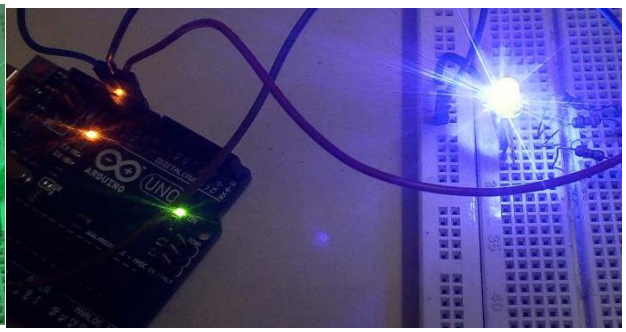

Figure 4.4 Blue colored RGB LED

To generate above colors the following code is required:

```
void loop() {
 RGBL1.on(); // White
delay(2000);
RGBL1.SetColor(255,0,0); //Red
delay(2000);
RGBL1.SetColor(0,255,0); //Green
delay(2000);
RGBL1.SetColor(0,0,255); //Blue
delay(10000);
RGBL1.off();
delay(2000);
}
```

Delay is used so that color can be observed properly. As RGBL1 object is created by writing RGBL1.SetColor(255,0,0) programmer can provide full ON 100% duty cycle of PWM to RED pin and 0%duty cycle to Green and blue so that Red color can be generated. Same is applicable for Green and Blue color. To generate White color RGBL1.on() calling is used.

V. Result

For results, various sample values to generate purple,cyan, gold, orange and Yellow colors were taken. The combination of Red, Green and Blue colors as per color coding scheme is shown below

Table 5.1 RGB Values of sample[1]

Sr. No	Color	R Value	G Value	B value
1	Purple	200	0	200
2	Cyan	40	255	255
3	Gold	255	215	0
4	Orange	255	178	102
5	Yellow	255	255	0

Figure 5.1 Purple Colored RGB LED.

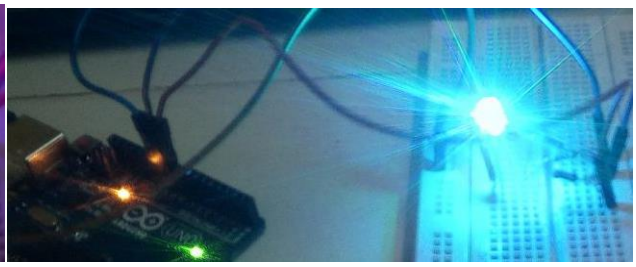

Figure 5.2 Cyan colored LED

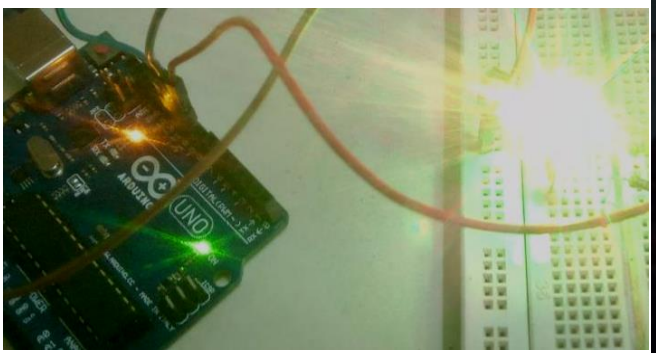

Figure 5.3 Gold Colored RGB LED.

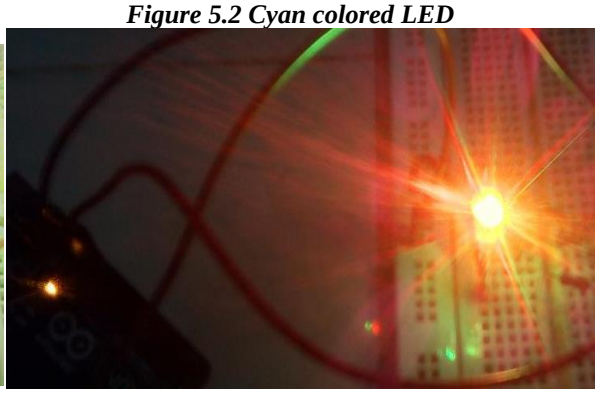

Figure 5.4 Orange colored LED

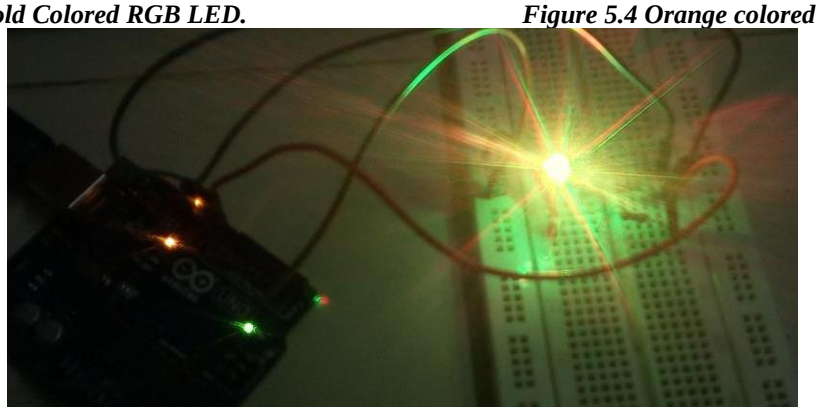

Figure 5.5 Yellow Colored RGB LED.

Conclusion

The simplicity is achieved due to library development of RGB LED. The programmer only needs RGB color scheme to generate or create any color. The header and c++ files for RGB LED can be used for other devices or software. The high frequency PWM has also given quite impressive results for color stability. The very next advantage is that the added library can be used for other hardware also like STM32f103c8t6 ARM device.

References

[1] Rafael C. Gonzalez and Richard E. Woods , "Digital Image Processing," 3/e
[2]Datasheet of ATMEGA328P http://ww1.microchip.com/downloads/en/DeviceDoc/Atmel-42735-8-bit-AVR-Microcontroller-ATmega328-328P_Datasheet.pdf