Grove - RGB LED Ring (20 - WS2813 Mini)

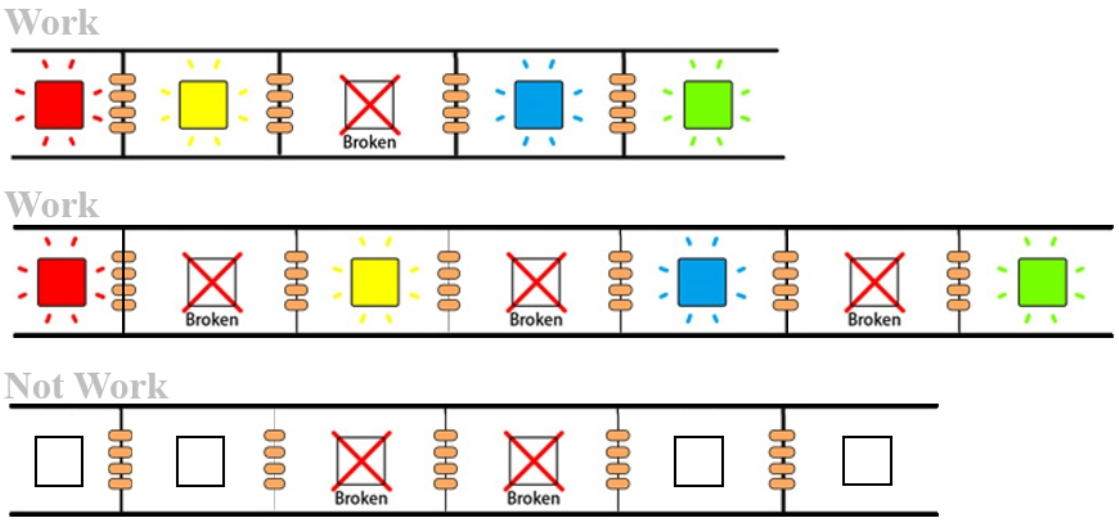
The Grove - RGB LED Ring (20 - WS2813 Mini) is a mini version of WS2813 Digital RGB LED Ring [https://www.seeedstudio.com/WS2813-Digital-RGB-LED-Ring-p-2871.html]. The RGB LED Ring are 3535-sized LEDs with an embedded microcontroller inside the LED. The WS2813s are each addressable as the driver chip is located inside the LED. Each LED has a constrant current drive so the color will be very consistent even if the change of the voltage.

Get One Now [https://www.seeedstudio.com/Grove-RGB-LED-Ring-20-WS2813-Min-p-3227.html]

Feature

- The control circuit and RGB chip are integrated in a 3535 components, to form an external control pixel.
- Intelligent Reverse-connection protection.
- Built-in signal reshaping circuit
- 256 gray level and 16777216 full-color display
- Serial cascade interface, data receiving and decoding depend on just one signal line.
- Data transmitting at speeds of up to 800Kbps.
- Dual-signal wires version, signal break-point continuous transmission.

Signal break-point continuous transmission



As long as not two or more adjacent LEDs are broken, the remaining LEDs will be able to work normally.

Specification

Item	Value
Power	3.3V/5V
Quiescent Current	0.7mA/LED
RGB Channel Constant Current	16mA/LED
Refresh Frequency	2KHz
Reset Time	>280μs
Operating Temperature	-25 ~ 85℃
Storage Temperature	-40 ~ 105℃

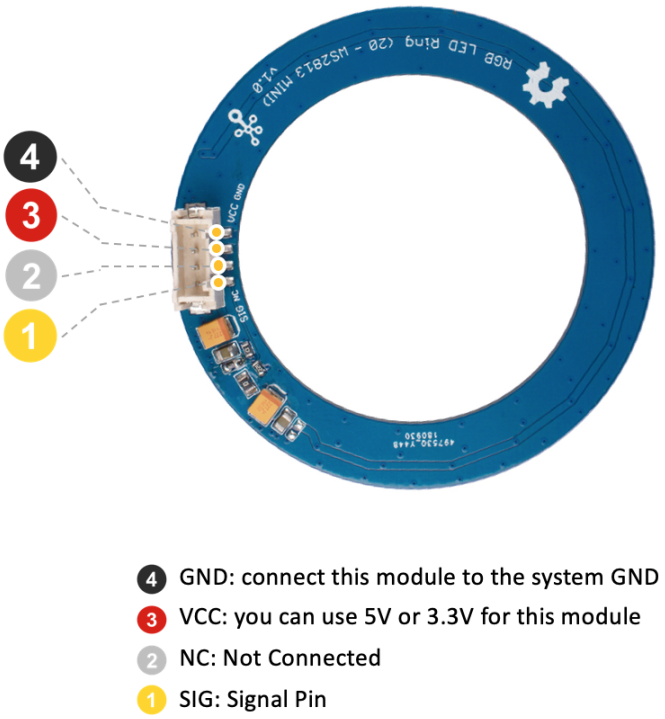
Typical applications

- Guardrail tube series, point light display series, flexible/rigid strips series, module series applications.

- Lighting stage costumes, innovative gadgets or any other electronic products.

Hardware Overview

Pin Out



Hardware Detail

WS2813B-Mini

WS2813-Mini is an intelligent control LED light source that the control circuit and RGB chip are integrated in a package of 3535 components. Its internal include intelligent digital port data latch and signal reshaping amplification drive circuit. Also include a precision internal oscillator and a 12V voltage programmable constant current control part, which achieves highly consistent color effect.

Platforms Supported

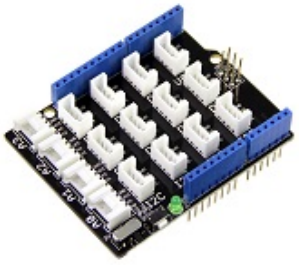
Arduino	Raspberry Pi	
	TO DO	

Caution
The platforms mentioned above as supported is/are an indication of the module's software or theoritical compatibility. We only provide software library or code examples for Arduino platform in most cases. It is not possible to provide software library / demo code for all possible MCU platforms. Hence, users have to write their own software library.

Getting Started

Play With Arduino

Hardware

Seeeduino V4.2	Base Shield	Grove - RGB LED Ring
		
Get ONE Now [https://www.seeedstudio.com/Seeeduino-V4.2-p-2517.html]	Get ONE Now [https://www.seeedstudio.com/Base-Shield-V2-p-1378.html]	Get ONE Now [https://www.seeedst RGB-LED-Ring-20-WS 3227.html]

Note

1. Please plug the USB cable gently, otherwise you may damage the port. Please use the USB cable with 4 wires inside, the 2 wires cable can't transfer data. If you are not sure about the wire you have, you can click here [https://www.seeedstudio.com/Micro-USB-Cable-48cm-p-1475.html] to buy

2. Each Grove module comes with a Grove cable when you buy. In case you lose the Grove cable, you can click here [https://www.seeedstudio.com/Grove-Universal-4-Pin-Buckled-20cm-Cable-%285-PCs-pack%29-p-936.html] to buy.

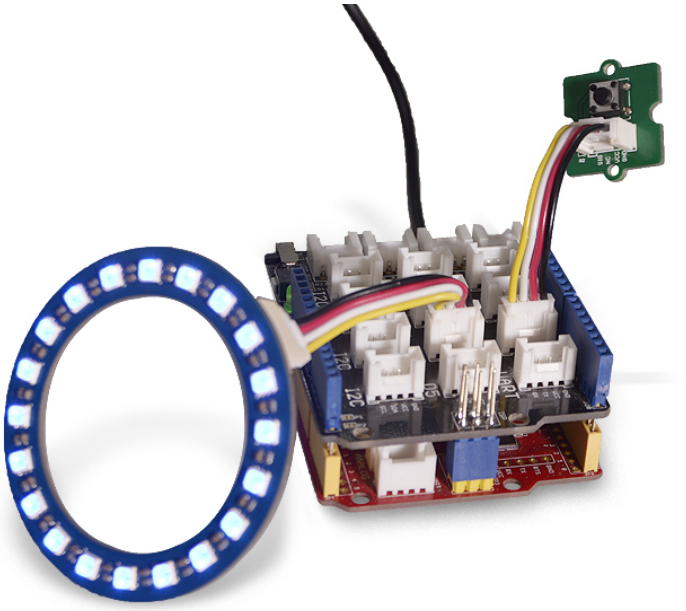
- **Step 1.** Connect Grove - Button to port D2 of Grove - Base Shield.
- **Step 2.** Connect the Grove - RGB LED Ring to port D6 of Grove-Base Shield.
- **Step 3.** Plug Grove - Base Shield into Seeeduino.

Caution

1. If you are using Arduino UNO, connect to the DC power supply is recommended to avoid the maximum Vcc voltage ripple to exceed 100mV.

2. If you are using Seeeduino V4.2, you do not need to connect to DC power supply. However, if you change to supply Grove with 3V3 the motherboard will restart when power is on. Please note, this will not affect the usage.

- **Step 4.** Connect Seeeduino to PC via a USB cable.



Warning

Hot swap is not supported, you may want to disconnect arduino from the power source before any replacement or change.

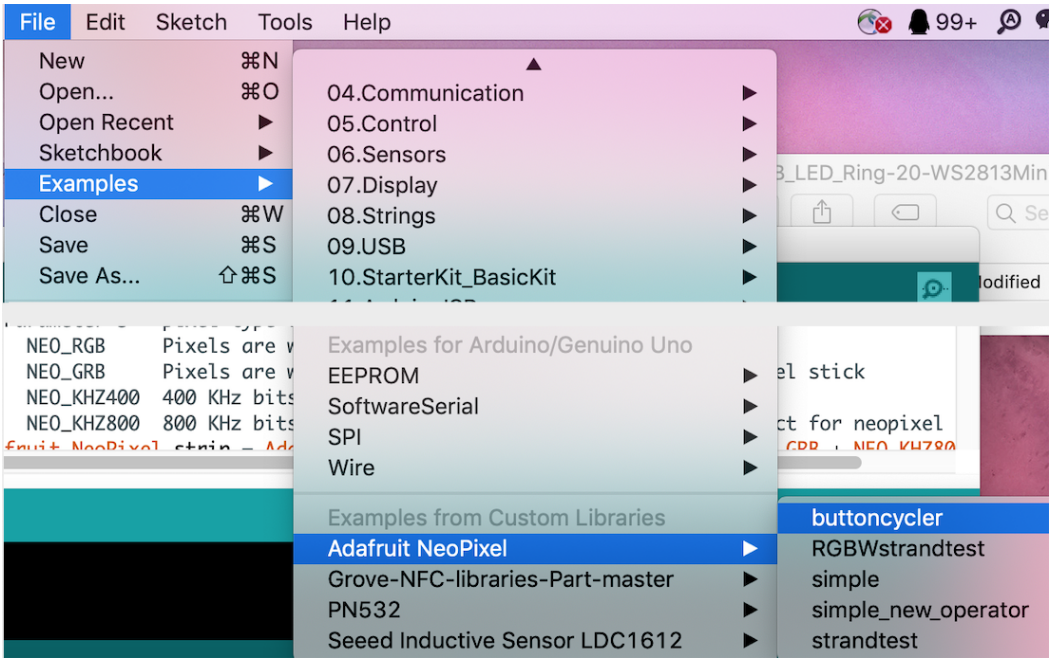
Now, we will demonstrate you how to run the code 'buttoncycler'. This is a demonstration on how to use an additional input device(button) to trigger changes on your LED ring. Similar procedure if you wish to run other programs, the only change is you need to disconnect button from port D2 of base shield as you are not using it.

Software

> **Attention**
> If this is the first time you work with Arduino, we strongly recommend you to see Getting Started with Arduino [https://wiki.seeedstudio.com/Getting_Started_with_Arduino/] before the start.

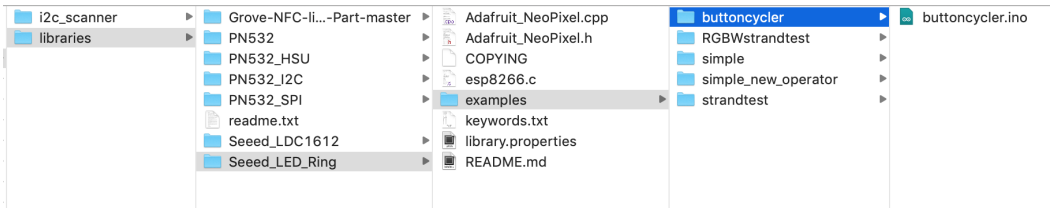
- **Step 1.** Download the Grove-RGB_LED_Ring-20-WS2813Mini [https://github.com/Seeed-Studio/Seeed_LED_Ring.git] Library from Github.
- **Step 2.** Refer to How to install library [https://wiki.seeedstudio.com/How_to_install_Arduino_Library] to install library for Arduino.
- **Step 3.** Restart the Arduino IDE. Open the example, you can open it in the following three ways:

  a. Open it directly in the Arduino IDE via the path: **File → Examples →Adafruit NeoPixel→buttoncycler**.



  b. Open it in your computer by click the **basic_demo.ino** which you can find in the folder **XXXX\Arduino\libraries\Seeed_LED_Ring\examples\buttoncycler\buttoncycler.ino**,

XXXX is the location you installed the Arduino IDE.



c. Or, you can just click the icon in upper right corner of the code block to copy the following code into a new sketch in the Arduino IDE.

> **Notice**
> Due to the current limitation, the brightness of the LED is limited in the program. If you insist on adjusting the brightness limit, you can modify the setBrightness() function. But this may cause the light to not work properly.

buttoncycler

After run this code, when you press the button it will change to a new pixel animation. Note that you need to press the button once to start the first animation!

```
#include "Adafruit_NeoPixel.h"

#define BUTTON_PIN   2    // Digital IO pin connected to the button.  This w
                          // driven with a pull-up resistor so the switch sh
                          // pull the pin to ground momentarily.  On a high
                          // transition the button press logic will execute.

#define PIXEL_PIN    6    // Digital IO pin connected to the NeoPixels.

#define PIXEL_COUNT 20

// Parameter 1 = number of pixels in strip,  neopixel stick has 8
// Parameter 2 = pin number (most are valid)
// Parameter 3 = pixel type flags, add together as needed:
//   NEO_RGB     Pixels are wired for RGB bitstream
//   NEO_GRB     Pixels are wired for GRB bitstream, correct for neopixel st
//   NEO_KHZ400  400 KHz bitstream (e.g. FLORA pixels)
//   NEO_KHZ800  800 KHz bitstream (e.g. High Density LED strip), correct fo
Adafruit_NeoPixel strip = Adafruit_NeoPixel(PIXEL_COUNT, PIXEL_PIN, NEO_GRB

bool oldState = HIGH;
int showType = 0;

void setup() {
  pinMode(BUTTON_PIN, INPUT_PULLUP);
  strip.setBrightness(255);
  strip.begin();
  strip.show(); // Initialize all pixels to 'off'
}
```

```
30
31  void loop() {
32    // Get current button state.
33    bool newState = digitalRead(BUTTON_PIN);
34
35    // Check if state changed from high to low (button press).
36    if (newState == LOW && oldState == HIGH) {
37      // Short delay to debounce button.
38      delay(20);
39      // Check if button is still low after debounce.
40      newState = digitalRead(BUTTON_PIN);
41      if (newState == LOW) {
42        showType++;
43        if (showType > 9)
44          showType=0;
45        startShow(showType);
46      }
47    }
48
49    // Set the last button state to the old state.
50    oldState = newState;
51  }
52
53  void startShow(int i) {
54    switch(i){
55      case 0: colorWipe(strip.Color(0, 0, 0), 50);    // Black/off
56              break;
57      case 1: colorWipe(strip.Color(255, 0, 0), 50);  // Red
58              break;
59      case 2: colorWipe(strip.Color(0, 255, 0), 50);  // Green
60              break;
61      case 3: colorWipe(strip.Color(0, 0, 255), 50);  // Blue
62              break;
63      case 4: theaterChase(strip.Color(127, 127, 127), 50); // White
64              break;
65      case 5: theaterChase(strip.Color(127,   0,   0), 50); // Red
66              break;
67      case 6: theaterChase(strip.Color(  0,   0, 127), 50); // Blue
68              break;
69      case 7: rainbow(20);
70              break;
71      case 8: rainbowCycle(20);
72              break;
73      case 9: theaterChaseRainbow(50);
74              break;
75    }
76  }
77
78  // Fill the dots one after the other with a color
79  void colorWipe(uint32_t c, uint8_t wait) {
80    for(uint16_t i=0; i < strip.numPixels(); i++) {
81      strip.setPixelColor(i, c);
82      strip.show();
83      delay(wait);
```

```
  }
}

void rainbow(uint8_t wait) {
  uint16_t i, j;

  for(j=0; j<256; j++) {
    for(i=0; i<strip.numPixels(); i++) {
      strip.setPixelColor(i, Wheel((i+j) & 255));
    }
    strip.show();
    delay(wait);
  }
}

// Slightly different, this makes the rainbow equally distributed throughout
void rainbowCycle(uint8_t wait) {
  uint16_t i, j;

  for(j=0; j<256*5; j++) { // 5 cycles of all colors on wheel
    for(i=0; i< strip.numPixels(); i++) {
      strip.setPixelColor(i, Wheel(((i * 256 / strip.numPixels()) + j) & 255
    }
    strip.show();
    delay(wait);
  }
}

//Theatre-style crawling lights.
void theaterChase(uint32_t c, uint8_t wait) {
  for (int j=0; j<10; j++) {  //do 10 cycles of chasing
    for (int q=0; q < 3; q++) {
      for (int i=0; i < strip.numPixels(); i=i+3) {
        strip.setPixelColor(i+q, c);    //turn every third pixel on
      }
      strip.show();

      delay(wait);

      for (int i=0; i < strip.numPixels(); i=i+3) {
        strip.setPixelColor(i+q, 0);        //turn every third pixel off
      }
    }
  }
}

//Theatre-style crawling lights with rainbow effect
void theaterChaseRainbow(uint8_t wait) {
  for (int j=0; j < 256; j++) {     // cycle all 256 colors in the wheel
    for (int q=0; q < 3; q++) {
      for (int i=0; i < strip.numPixels(); i=i+3) {
        strip.setPixelColor(i+q, Wheel( (i+j) % 255));    //turn every third
      }
      strip.show();
```

```
138
139       delay(wait);
140
141       for (int i=0; i < strip.numPixels(); i=i+3) {
142         strip.setPixelColor(i+q, 0);        //turn every third pixel off
143       }
144     }
145   }
146 }
147
148 // Input a value 0 to 255 to get a color value.
149 // The colours are a transition r - g - b - back to r.
150 uint32_t Wheel(byte WheelPos) {
151   WheelPos = 255 - WheelPos;
152   if(WheelPos < 85) {
153     return strip.Color(255 - WheelPos * 3, 0, WheelPos * 3);
154   }
155   if(WheelPos < 170) {
156     WheelPos -= 85;
157     return strip.Color(0, WheelPos * 3, 255 - WheelPos * 3);
158   }
159   WheelPos -= 170;
160   return strip.Color(WheelPos * 3, 255 - WheelPos * 3, 0);
161 }
```

Attention
The library file may be updated. This code may not be applicable to the updated library file, so we recommend that you use the first two methods.

Success
If everything goes well, you will be able to see the first animation of the LED ring, and you will be able to trigger the new animation once you press the button.

Other Examples:

RGBW strand test

```
1  #include "Adafruit_NeoPixel.h"
2  #ifdef __AVR__
3    #include <avr/power.h>
4  #endif
5
6  #define PIN 6
7
8  #define NUM_LEDS 20
9
10 #define BRIGHTNESS 255
11
```

```
Adafruit_NeoPixel strip = Adafruit_NeoPixel(NUM_LEDS, PIN, NEO_GRBW + NEO_KH

byte neopix_gamma[] = {
    0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,
    0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  1,  1,  1,  1,
    1,  1,  1,  1,  1,  1,  1,  1,  1,  2,  2,  2,  2,  2,  2,  2,
    2,  3,  3,  3,  3,  3,  3,  3,  4,  4,  4,  4,  4,  5,  5,  5,
    5,  6,  6,  6,  6,  7,  7,  7,  7,  8,  8,  8,  9,  9,  9, 10,
   10, 10, 11, 11, 11, 12, 12, 13, 13, 13, 14, 14, 15, 15, 16, 16,
   17, 17, 18, 18, 19, 19, 20, 20, 21, 21, 22, 22, 23, 24, 24, 25,
   25, 26, 27, 27, 28, 29, 29, 30, 31, 32, 32, 33, 34, 35, 35, 36,
   37, 38, 39, 39, 40, 41, 42, 43, 44, 45, 46, 47, 48, 49, 50, 50,
   51, 52, 54, 55, 56, 57, 58, 59, 60, 61, 62, 63, 64, 66, 67, 68,
   69, 70, 72, 73, 74, 75, 77, 78, 79, 81, 82, 83, 85, 86, 87, 89,
   90, 92, 93, 95, 96, 98, 99,101,102,104,105,107,109,110,112,114,
  115,117,119,120,122,124,126,127,129,131,133,135,137,138,140,142,
  144,146,148,150,152,154,156,158,160,162,164,167,169,171,173,175,
  177,180,182,184,186,189,191,193,196,198,200,203,205,208,210,213,
  215,218,220,223,225,228,231,233,236,239,241,244,247,249,252,255 };


void setup() {
  // This is for Trinket 5V 16MHz, you can remove these three lines if you a
  #if defined (__AVR_ATtiny85__)
    if (F_CPU == 16000000) clock_prescale_set(clock_div_1);
  #endif
  // End of trinket special code
  strip.setBrightness(BRIGHTNESS);
  strip.begin();
  strip.show(); // Initialize all pixels to 'off'
}

void loop() {
  // Some example procedures showing how to display to the pixels:
  colorWipe(strip.Color(255, 0, 0), 50); // Red
  colorWipe(strip.Color(0, 255, 0), 50); // Green
  colorWipe(strip.Color(0, 0, 255), 50); // Blue
  colorWipe(strip.Color(0, 0, 0, 255), 50); // White

  whiteOverRainbow(20,75,5);

  pulseWhite(5);

  // fullWhite();
  // delay(2000);

  rainbowFade2White(3,3,1);


}

// Fill the dots one after the other with a color
void colorWipe(uint32_t c, uint8_t wait) {
  for(uint16_t i=0; i < strip.numPixels(); i++) {
```

```
    strip.setPixelColor(i, c);
    strip.show();
    delay(wait);
  }
}

void pulseWhite(uint8_t wait) {
  for(int j = 0; j < 256 ; j++){
      for(uint16_t i=0; i < strip.numPixels(); i++) {
          strip.setPixelColor(i, strip.Color(0,0,0, neopix_gamma[j] ) );
        }
        delay(wait);
        strip.show();
      }

  for(int j = 255; j >= 0 ; j--){
      for(uint16_t i=0; i<strip.numPixels(); i++) {
          strip.setPixelColor(i, strip.Color(0,0,0, neopix_gamma[j] ) );
        }
        delay(wait);
        strip.show();
      }
}


void rainbowFade2White(uint8_t wait, int rainbowLoops, int whiteLoops) {
  float fadeMax = 100.0;
  int fadeVal = 0;
  uint32_t wheelVal;
  int redVal, greenVal, blueVal;

  for(int k = 0 ; k < rainbowLoops ; k ++){

    for(int j=0; j<256; j++) { // 5 cycles of all colors on wheel

      for(int i=0; i< strip.numPixels(); i++) {

        wheelVal = Wheel(((i * 256 / strip.numPixels()) + j) & 255);

        redVal = red(wheelVal) * float(fadeVal/fadeMax);
        greenVal = green(wheelVal) * float(fadeVal/fadeMax);
        blueVal = blue(wheelVal) * float(fadeVal/fadeMax);

        strip.setPixelColor( i, strip.Color( redVal, greenVal, blueVal ) );

      }

      //First loop, fade in!
      if(k == 0 && fadeVal < fadeMax-1) {
          fadeVal++;
      }

      //Last loop, fade out!
      else if(k == rainbowLoops - 1 && j > 255 - fadeMax ){
```

```
            fadeVal--;
        }

          strip.show();
          delay(wait);
      }

    }



    delay(500);


    for(int k = 0 ; k < whiteLoops ; k ++){

      for(int j = 0; j < 256 ; j++){

          for(uint16_t i=0; i < strip.numPixels(); i++) {
              strip.setPixelColor(i, strip.Color(0,0,0, neopix_gamma[j] ) );
            }
            strip.show();
          }

          delay(2000);
      for(int j = 255; j >= 0 ; j--){

          for(uint16_t i=0; i < strip.numPixels(); i++) {
              strip.setPixelColor(i, strip.Color(0,0,0, neopix_gamma[j] ) );
            }
            strip.show();
          }
    }

    delay(500);


}

void whiteOverRainbow(uint8_t wait, uint8_t whiteSpeed, uint8_t whiteLength

    if(whiteLength >= strip.numPixels()) whiteLength = strip.numPixels() - 1;

    int head = whiteLength - 1;
    int tail = 0;

    int loops = 3;
    int loopNum = 0;

    static unsigned long lastTime = 0;


    while(true){
      for(int j=0; j<256; j++) {
```

```
      for(uint16_t i=0; i<strip.numPixels(); i++) {
        if((i >= tail && i <= head) || (tail > head && i >= tail) || (tail >
          strip.setPixelColor(i, strip.Color(0,0,0, 255 ) );
        }
        else{
          strip.setPixelColor(i, Wheel(((i * 256 / strip.numPixels()) + j) &
        }

      }

      if(millis() - lastTime > whiteSpeed) {
        head++;
        tail++;
        if(head == strip.numPixels()){
          loopNum++;
        }
        lastTime = millis();
      }

      if(loopNum == loops) return;

      head%=strip.numPixels();
      tail%=strip.numPixels();
        strip.show();
        delay(wait);
    }
  }

}
void fullWhite() {

    for(uint16_t i=0; i<strip.numPixels(); i++) {
        strip.setPixelColor(i, strip.Color(0,0,0, 255 ) );
    }
      strip.show();
}


// Slightly different, this makes the rainbow equally distributed throughout
void rainbowCycle(uint8_t wait) {
  uint16_t i, j;

  for(j=0; j<256 * 5; j++) { // 5 cycles of all colors on wheel
    for(i=0; i< strip.numPixels(); i++) {
      strip.setPixelColor(i, Wheel(((i * 256 / strip.numPixels()) + j) & 255
    }
    strip.show();
    delay(wait);
  }
}

void rainbow(uint8_t wait) {
  uint16_t i, j;

```

```
  for(j=0; j<256; j++) {
    for(i=0; i<strip.numPixels(); i++) {
      strip.setPixelColor(i, Wheel((i+j) & 255));
    }
    strip.show();
    delay(wait);
  }
}

// Input a value 0 to 255 to get a color value.
// The colours are a transition r - g - b - back to r.
uint32_t Wheel(byte WheelPos) {
  WheelPos = 255 - WheelPos;
  if(WheelPos < 85) {
    return strip.Color(255 - WheelPos * 3, 0, WheelPos * 3,0);
  }
  if(WheelPos < 170) {
    WheelPos -= 85;
    return strip.Color(0, WheelPos * 3, 255 - WheelPos * 3,0);
  }
  WheelPos -= 170;
  return strip.Color(WheelPos * 3, 255 - WheelPos * 3, 0,0);
}

uint8_t red(uint32_t c) {
  return (c >> 16);
}
uint8_t green(uint32_t c) {
  return (c >> 8);
}
uint8_t blue(uint32_t c) {
  return (c);
}
```

Simple

```
#include "Adafruit_NeoPixel.h"
#ifdef __AVR__
  #include <avr/power.h>
#endif

// Which pin on the Arduino is connected to the NeoPixels?
// On a Trinket or Gemma we suggest changing this to 1
#define PIN            6

// How many NeoPixels are attached to the Arduino?
#define NUMPIXELS      20

// When we setup the NeoPixel library, we tell it how many pixels, and which
// Note that for older NeoPixel strips you might need to change the third par
// example for more information on possible values.
Adafruit_NeoPixel pixels = Adafruit_NeoPixel(NUMPIXELS, PIN, NEO_GRB + NEO_KH
```

```
17
18  int delayval = 500; // delay for half a second
19
20  void setup() {
21    // This is for Trinket 5V 16MHz, you can remove these three lines if you ar
22  #if defined (__AVR_ATtiny85__)
23    if (F_CPU == 16000000) clock_prescale_set(clock_div_1);
24  #endif
25    // End of trinket special code
26    pixels.setBrightness(255);
27    pixels.begin(); // This initializes the NeoPixel library.
28  }
29
30  void loop() {
31
32    // For a set of NeoPixels the first NeoPixel is 0, second is 1, all the way
33
34    for(int i=0;i<NUMPIXELS;i++){
35
36      // pixels.Color takes RGB values, from 0,0,0 up to 255,255,255
37      pixels.setPixelColor(i, pixels.Color(0,150,0)); // Moderately bright gree
38
39      pixels.show(); // This sends the updated pixel color to the hardware.
40
41      delay(delayval); // Delay for a period of time (in milliseconds).
42
43    }
44  }
```

Simple New Operator

```
1   #include "Adafruit_NeoPixel.h"
2   #ifdef __AVR__
3     #include <avr/power.h>
4   #endif
5
6   // Which pin on the Arduino is connected to the NeoPixels?
7   // On a Trinket or Gemma we suggest changing this to 1
8   #define PIN            6
9
10  // How many NeoPixels are attached to the Arduino?
11  int numPixel = 20;
12
13  // Color order, for more information see https://github.com/adafruit/Adafruit_
14  uint8_t colorOrder = 0x52; //or just use NEO_GBR
15
16  // Define new pointer for NeoPixel
17  Adafruit_NeoPixel *pixels;
18
19
20  int delayval = 500; // delay for half a second
21
```

```
void setup() {
  // This is for Trinket 5V 16MHz, you can remove these three lines if you ar
#if defined (__AVR_ATtiny85__)
  if (F_CPU == 16000000) clock_prescale_set(clock_div_1);
#endif
  // End of trinket special code

  // Here is a good place to read numPixel & colorOrder from EEPROM or what e
  // create a new NeoPixel instance with new values
  pixels = new Adafruit_NeoPixel(numPixel, PIN, colorOrder);
  pixels->setBrightness(255);
  pixels->begin(); // This initializes the NeoPixel library.
}

void loop() {

  // For a set of NeoPixels the first NeoPixel is 0, second is 1, all the way

  for(int i=0;i<numPixel;i++){

    // pixels.Color takes RGB values, from 0,0,0 up to 255,255,255
    pixels->setPixelColor(i, pixels->Color(0,150,0)); // Moderately bright gr

    pixels->show(); // This sends the updated pixel color to the hardware.

    delay(delayval); // Delay for a period of time (in milliseconds).

  }
}
```

Strand test

```
#include "Adafruit_NeoPixel.h"
#ifdef __AVR__
  #include <avr/power.h>
#endif

#define PIN 6

// Parameter 1 = number of pixels in strip
// Parameter 2 = Arduino pin number (most are valid)
// Parameter 3 = pixel type flags, add together as needed:
//   NEO_KHZ800  800 KHz bitstream (most NeoPixel products w/WS2812 LEDs)
//   NEO_KHZ400  400 KHz (classic 'v1' (not v2) FLORA pixels, WS2811 drivers
//   NEO_GRB     Pixels are wired for GRB bitstream (most NeoPixel products)
//   NEO_RGB     Pixels are wired for RGB bitstream (v1 FLORA pixels, not v2
//   NEO_RGBW    Pixels are wired for RGBW bitstream (NeoPixel RGBW products
Adafruit_NeoPixel strip = Adafruit_NeoPixel(20, PIN, NEO_GRB + NEO_KHZ800);

// IMPORTANT: To reduce NeoPixel burnout risk, add 1000 uF capacitor across
// pixel power leads, add 300 - 500 Ohm resistor on first pixel's data input
// and minimize distance between Arduino and first pixel.  Avoid connecting
```

```
// on a live circuit...if you must, connect GND first.

void setup() {
  // This is for Trinket 5V 16MHz, you can remove these three lines if you a
  #if defined (__AVR_ATtiny85__)
    if (F_CPU == 16000000) clock_prescale_set(clock_div_1);
  #endif
  // End of trinket special code

  strip.begin();
  strip.setBrightness(255);
  strip.show(); // Initialize all pixels to 'off'
}

void loop() {
  // Some example procedures showing how to display to the pixels:
  colorWipe(strip.Color(255, 0, 0), 50); // Red
  colorWipe(strip.Color(0, 255, 0), 50); // Green
  colorWipe(strip.Color(0, 0, 255), 50); // Blue
//colorWipe(strip.Color(0, 0, 0, 255), 50); // White RGBW
  // Send a theater pixel chase in...
  theaterChase(strip.Color(127, 127, 127), 50); // White
  theaterChase(strip.Color(127, 0, 0), 50); // Red
  theaterChase(strip.Color(0, 0, 127), 50); // Blue

  rainbow(20);
  rainbowCycle(20);
  theaterChaseRainbow(50);
}

// Fill the dots one after the other with a color
void colorWipe(uint32_t c, uint8_t wait) {
  for(uint16_t i=0; i < strip.numPixels(); i++) {
    strip.setPixelColor(i, c);
    strip.show();
    delay(wait);
  }
}

void rainbow(uint8_t wait) {
  uint16_t i, j;

  for(j=0; j<256; j++) {
    for(i=0; i<strip.numPixels(); i++) {
      strip.setPixelColor(i, Wheel((i+j) & 255));
    }
    strip.show();
    delay(wait);
  }
}

// Slightly different, this makes the rainbow equally distributed throughout
void rainbowCycle(uint8_t wait) {
  uint16_t i, j;
```

```
75
76    for(j=0; j<256*5; j++) { // 5 cycles of all colors on wheel
77      for(i=0; i< strip.numPixels(); i++) {
78        strip.setPixelColor(i, Wheel(((i * 256 / strip.numPixels()) + j) & 255
79      }
80      strip.show();
81      delay(wait);
82    }
83  }
84
85  //Theatre-style crawling lights.
86  void theaterChase(uint32_t c, uint8_t wait) {
87    for (int j=0; j<10; j++) {  //do 10 cycles of chasing
88      for (int q=0; q < 3; q++) {
89        for (uint16_t i=0; i < strip.numPixels(); i=i+3) {
90          strip.setPixelColor(i+q, c);    //turn every third pixel on
91        }
92        strip.show();
93
94        delay(wait);
95
96        for (uint16_t i=0; i < strip.numPixels(); i=i+3) {
97          strip.setPixelColor(i+q, 0);        //turn every third pixel off
98        }
99      }
100   }
101 }
102
103 //Theatre-style crawling lights with rainbow effect
104 void theaterChaseRainbow(uint8_t wait) {
105   for (int j=0; j < 256; j++) {     // cycle all 256 colors in the wheel
106     for (int q=0; q < 3; q++) {
107       for (uint16_t i=0; i < strip.numPixels(); i=i+3) {
108         strip.setPixelColor(i+q, Wheel( (i+j) % 255));    //turn every third
109       }
110       strip.show();
111
112       delay(wait);
113
114       for (uint16_t i=0; i < strip.numPixels(); i=i+3) {
115         strip.setPixelColor(i+q, 0);        //turn every third pixel off
116       }
117     }
118   }
119 }
120
121 // Input a value 0 to 255 to get a color value.
122 // The colours are a transition r - g - b - back to r.
123 uint32_t Wheel(byte WheelPos) {
124   WheelPos = 255 - WheelPos;
125   if(WheelPos < 85) {
126     return strip.Color(255 - WheelPos * 3, 0, WheelPos * 3);
127   }
128   if(WheelPos < 170) {
```

```
129       WheelPos -= 85;
130       return strip.Color(0, WheelPos * 3, 255 - WheelPos * 3);
131     }
132     WheelPos -= 170;
133     return strip.Color(WheelPos * 3, 255 - WheelPos * 3, 0);
134   }
```

Schematic Online Viewer

Resources

- **[Zip]** Grove - RGB LED Ring(20 WS2813 Mini) Eagle Files [https://files.seeedstudio.com/wiki/Grove-RGB_LED_Ring-20-WS2813Mini/res/Grove%20-%20RGB%20LED%20Ring%20(20%20-%20WS2813%20Mini).zip]
- **[Zip]** Grove - RGB LED Ring(20 WS2813 Mini) Software Library [https://github.com/Seeed-Studio/Seeed_LED_Ring/archive/master.zip]

- **[PDF]** Datasheet WS2813- Mini [https://files.seeedstudio.com/wiki/Grove-RGB_LED_Ring-20-WS2813Mini/res/WS2813-Mini.pdf]

Tech Support

Please do not hesitate to submit the issue into our forum [https://forum.seeedstudio.com/] .