# Neo Pixels Ring with Arduino Nano

Will guide you to integrate the Neo Pixel Ring with Arduino Nano

Difficulté **Très facile** | Durée **1 heure(s)** | Catégories **Électronique** | Coût **10 USD ($)**

## Sommaire



## Introduction

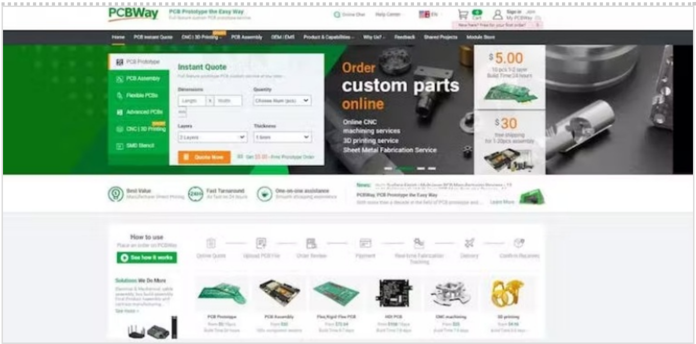
If you are looking for a way to add some colorful and dynamic lighting effects to your Arduino projects, you might want to try using neo-pixel rings. Neo-pixel rings are circular arrays of RGB LEDs that can be controlled individually by a single data line. They are easy to use and can create amazing patterns and animations with Arduino code.
In this article, I will show you how to integrate a pixel ring with Arduino Nano, a small and cheap microcontroller board that can be programmed to interact with various sensors and devices. You will learn how to wire the neo-pixel ring to the Arduino Nano, how to install and use the Adafruit Neo Pixel library, and how to code some basic lighting effects using the neo-pixel ring.
By the end of this article, you can create your custom lighting effects using a neo-pixel ring and Arduino Nano. You will also be able to modify and customize the code according to your preferences and needs. Let's get started!

## Matériaux

## Outils

## Étape 1 - Get PCBs For Your Projects Manufactured

You must check out PCBWAY for ordering PCBs online for cheap! You get 10 good-quality PCBs manufactured and shipped to your doorstep for cheap. You will also get a discount on shipping on your first order. Upload your Gerber files onto PCBWAY to get them manufactured with good quality and quick turnaround time. PCBWay now could provide a complete product solution, from design to enclosure production. Check out their online Gerber viewer function. With reward points, you can get free stuff from their gift shop. Also, check out this useful blog on PCBWay Plugin for KiCad from here. Using this plugin, you can directly order PCBs in just one click after completing your design in KiCad.



# Étape 2 - Materials and tools:

- 8-bit neo-pixel ring
- Arduino nano
- Jumper wires
- Mini USB cable
- Adafruit Neo Pixel library for Arduino

# Étape 3 - Wiring:

- Connect the IN pin of the neo-pixel ring to the D2 pin of the Arduino nano
- Connect the VCC pin of the neo-pixel ring to the +5V pin of the Arduino nano
- Connect the GND pin of the neo-pixel ring to the GND pin of the Arduino nano

# Étape 4 - Programming:

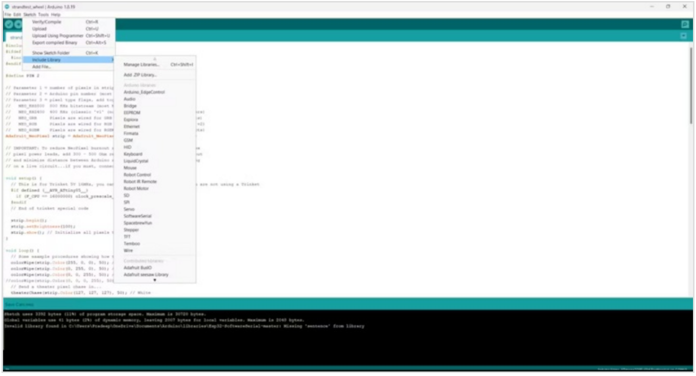
Once all the connections are made, open up the Arduino IDE and go to the Include library option.
Then add the downloaded zip library that we have previously mentioned.
Once the library installation is successful, you can see the neo pixel library examples in the examples sketch.
Next, open up the strandtest_wheel sketch and change the pin to 2. Because we have connected our Neo pixel to the D2 pin of the Arduino.

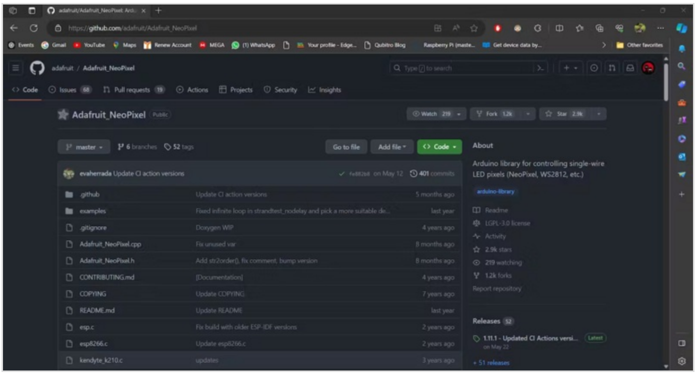
```
#include <Adafruit_NeoPixel.h>
#ifdef __AVR__
 #include <avr/power.h>
#endif

#define PIN 2

// 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(8, 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 are not using a Trinket
 #if defined (__AVR_ATtiny85__)
   if (F_CPU == 16000000) clock_prescale_set(clock_div_1);
 #endif
 // End of trinket special code

 strip.begin();
 strip.setBrightness(100);
 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);
```

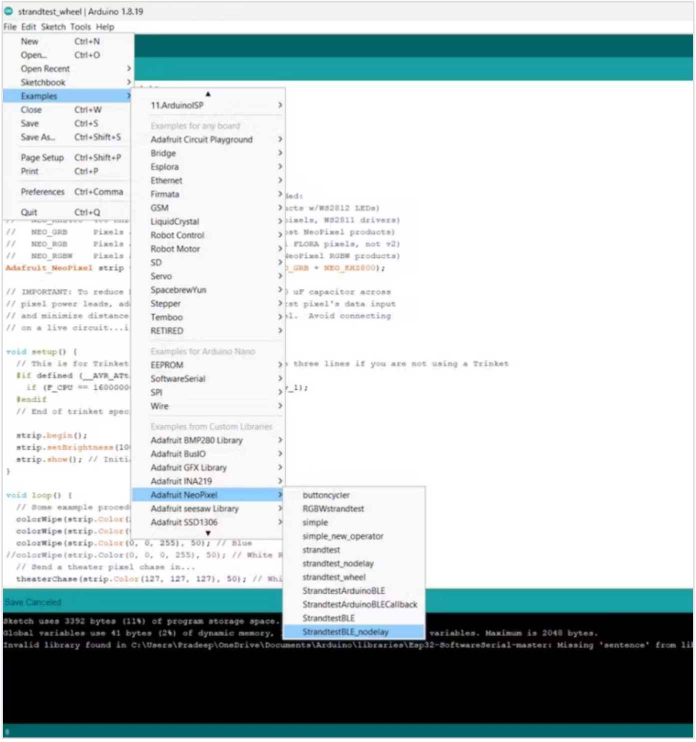
```
  theaterChaseRainbow(50);
}

// 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;

  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 (uint16_t i=0; i < strip.numPixels(); i=i+3) {
        strip.setPixelColor(i+q, c);    //turn every third pixel on
      }
      strip.show();

      delay(wait);

      for (uint16_t 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 (uint16_t i=0; i < strip.numPixels(); i=i+3) {
        strip.setPixelColor(i+q, Wheel( (i+j) % 255));    //turn every third pixel on
      }
      strip.show();

      delay(wait);

      for (uint16_t i=0; i < strip.numPixels(); i=i+3) {
        strip.setPixelColor(i+q, 0);        //turn every third pixel off
      }
    }
  }
}

// Input a value 0 to 255 to get a color value.
```

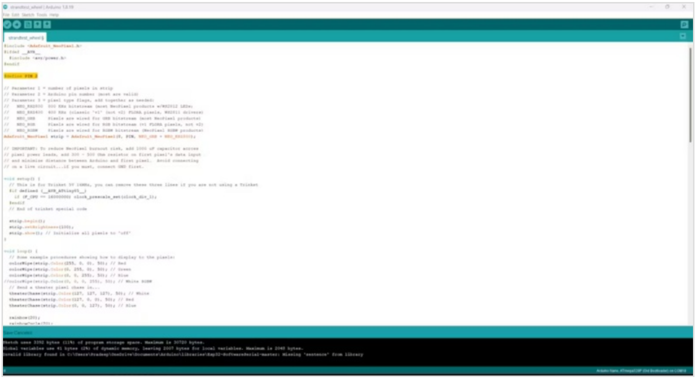
```
// The colours are a transition r - g - b - back to r.
uint32_t Wheel(byte WheelPos) {
  WheelPos = 255 - WheelPos;
  if(WheelPos < 85) {
    return strip.Color(255 - WheelPos * 3, 0, WheelPos * 3);
  }
  if(WheelPos < 170) {
    WheelPos -= 85;
    return strip.Color(0, WheelPos * 3, 255 - WheelPos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

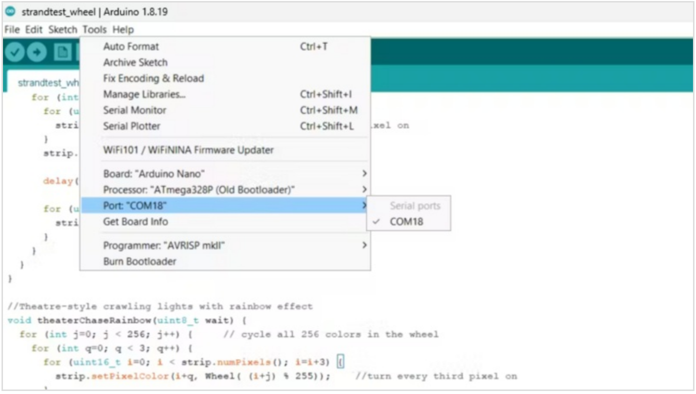
Next, select the correct port and board.









# Étape 5 - Conclusion:

You have learned how to connect neo pixel ring with Arduino Nano and create some lighting effects using Arduino code, You can try different colors, patterns, and speeds for your animations by changing the code parameters. You can also use other types of neo-pixel products, such as strips, matrices, or jewels, for more variety and complexity.