Lesson 3-Testing WS2812 LED Module

1. Knowledege of WS2812 LED

1.1 Overview：

WS2812B is an intelligently controlled LED that integrates the control circuit and RGB chip in a package of 5050 components.The interior includes intelligent digital ports, data latches and signal amplification drive circuits. It also includes a precise internal oscillator and a voltage programmable constant current control part to effectively ensure that the light color of the pixels is highly consistent.

The data transmission protocol adopts a single NZR communication mode. After the pixel is powered on and reset, the DIN port receives data from the controller, the first pixel collects the initial 24-bit data, and then sends it to the internal data latch, and other data processed by the internal signal amplification circuit are sent to the next one cascade pixels through the DO port. After each pixel is transmitted, the signal is reduced by 24 bits. The pixel adopts self-shaping transmission technology, so that the number of pixel cascading is not limited by signal transmission, but only depends on the speed of signal transmission.

Reset time >280us, no false reset when interrupted;
Support low frequency, with cheap MCU.
The refresh rate is updated to 2KHz, flicker-free, which improves the excellent display effect.

Characteristic:
* The control circuit and LED share the only power supply.
* The control circuit and RGB chip are integrated in a package of 5050 components to form a complete addressable circuit
* The pixel has a built-in signal-shaping circuit. After the waveform is shaped to the next driver, it ensures that the waveform distortion does not accumulate.
* Built-in electronic reset circuit and power failure reset circuit.
* The three primary colors of each pixel can display 256 luminances, completing the display of 16777216 colors, and the scanning frequency is 2KHz.
* The cascade port transmits signals over a single wire.
* If the distance between any two points does not exceed 5m, no additional circuit is required to transmit the

signal.
* When the refresh rate is 30fps, the number of cascades should not be less than 1024 pixels.
* Send data at 800Kbps.
* The color of the light is highly consistent and requires no external electronic components, not even capacitors.

1.2 Hardware Introduction:

WS2812B light strip: 30 beads per meter 9w, 60 beads per meter 18w, 144 beads per meter 43W, voltage: (DC) DC5V, the power consumption of each lamp bead is about 0.3W
Power supply: 5V
Power when each lamp bead is fully lit: 0.3W
Current when each lamp bead is fully lit: 0.6mA
Each chip has four pins, and each pin is defined in the following table

NO.	Pin	Function
1	5V	Positive power supply
2	GND	Negative pole of power supply
3	DI	Data input pin, control data entry of MCU
4	DO	Data output pin, cascading multiple SW2812 outputs, connected to the DI of the next SW2812

1.3 cascading wiring method

The DO of the previous chip is connected to the DI of the next chip
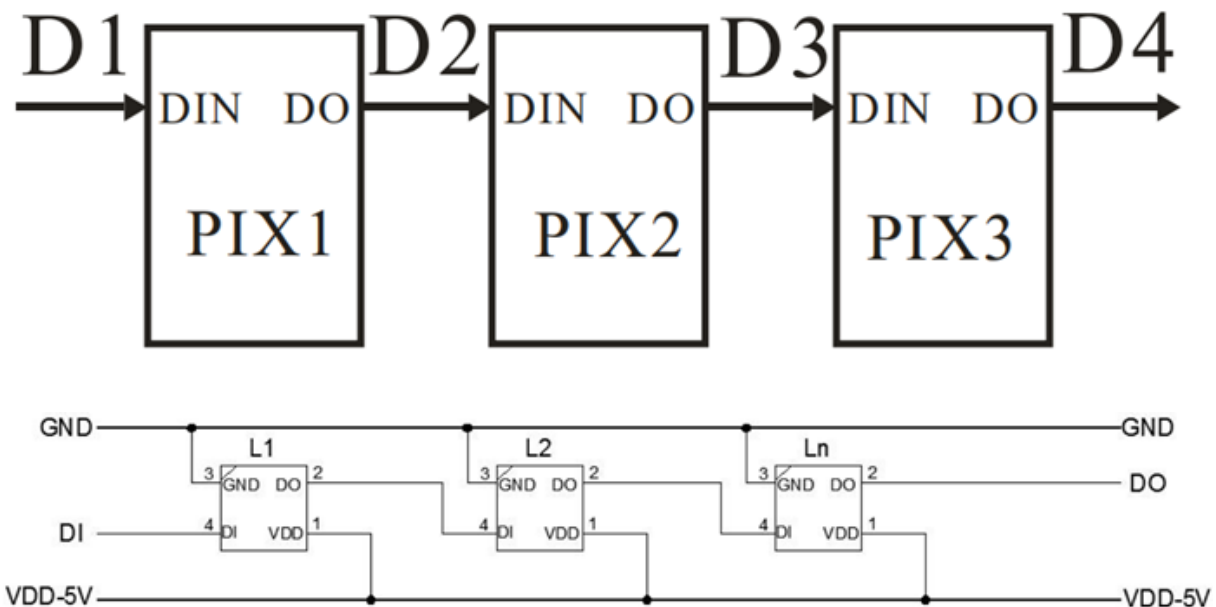
Control principle:
SW2812 is an RGB chip, so it has three colors of red, green and blue, and each color has corresponding 8 bits.

Usually, a pixel is represented by three colors of RGB. For example, #FFFFFF means that the value of R (red) is 255, the value of G (green) is 255, the value of B (blue) is 255, #FFFFFFF is finally displayed as white. So a SW2812 consists of 3 U8s, that is 3*8=24 bits. To determine the color of a SW28122 chip, it is necessary to send 24 bits of data.

1.4 Cascade data transmission

l The cache of the first screen data
u The first 24 bits are received and cached by the first module
u The second 24 bits will be forwarded by the first module to the second module and cached
u The third 24 bits will be forwarded to the third module by the first and second, and cached
u The fourth 24 bits...
u The Nth 24 bits...
l Reset signal, that is, the real embodiment of the cached data on the display
l Second screen data cache
u The first 24 bits are received and cached by the first module
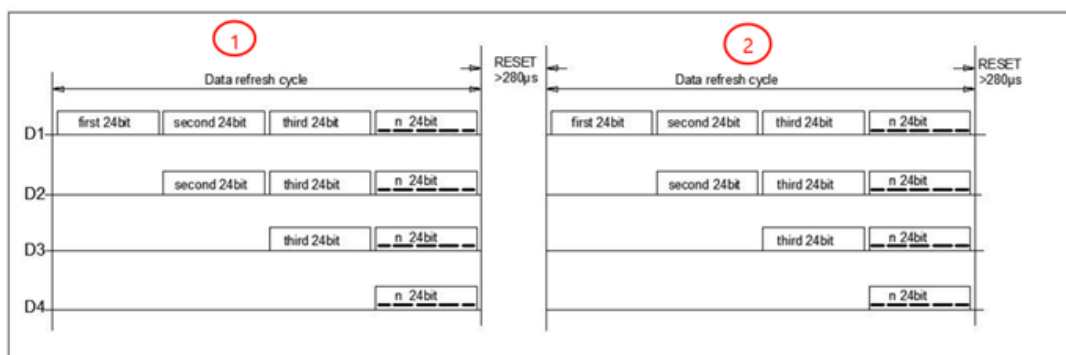u The second 24 bits will be forwarded by the first module to the second module and cached
u The third 24 bits will be forwarded to the lower three modules by the first and second and cached
u The fourth 24 bits...
u Lower N 24 bits...
l Reset signal, that is, the real embodiment of the cached data on the display
l...

1.5 The meaning of each 24-bit data

G7	G6	G5	G4	G3	G2	G1	G0	R7	R6	R5	R4	R3	R2	R1	R0	B7	B6	B5	B4	B3	B2	B1	B0

Data is transmitted in GRB order, the high-order data bits are transmitted first
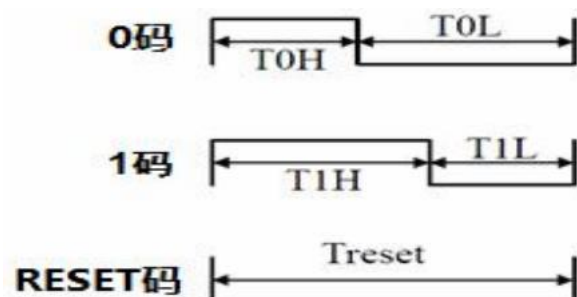
T0H	0 code, high voltage time	220ns~380ns
T1H	1 code, high voltage time	580ns~1μs
T0L	0 code, low voltage time	580ns~1μs
T1L	1 code, low voltage time	**580ns~1μs**
RES	Frame unit, low voltage time	>280μs

## 1.6 About the HSV color model in Adafruit_NeoPixel

In Adafruit_NeoPixel, the RGB color model can be used to control the red, green, and blue lights to synthesize various colors. The HSV color model can also be used to control the hue, saturation, and brightness of the lights to adjust the color.

The advantage of HSV control is that it is more convenient to control the brightness of the light and adjust the color to make it more in line with human intuition.
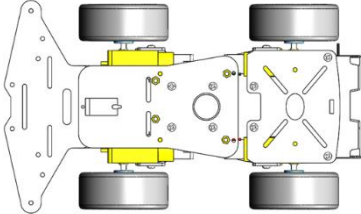
H in HSV: Hue parameter range 0--65535
S in HSV: Saturation parameter range 0--255
V in HSV: Brightness parameter range 0--255
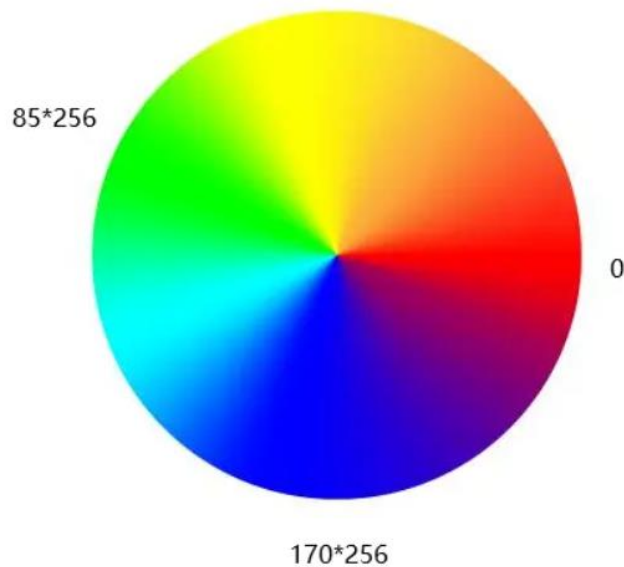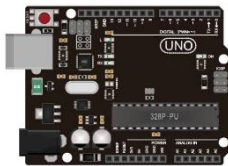
2. Component/Module List

Component/Module	QTY	Picture	Remark
4WD Car Chassis	1		Provided by the 4WD Car Chassis Kit

UNO R3 Board	1		You need to prepare these by yourself.These just as an example,you can DIY what is you want and prepared
3-WS2812 LED Module	1		
Dupont line	Some		

3. Circuit connection

3-WS2812 LED Module Pin	UNO R3 Board Pin
VCC	5V
GND	GND
IN	2
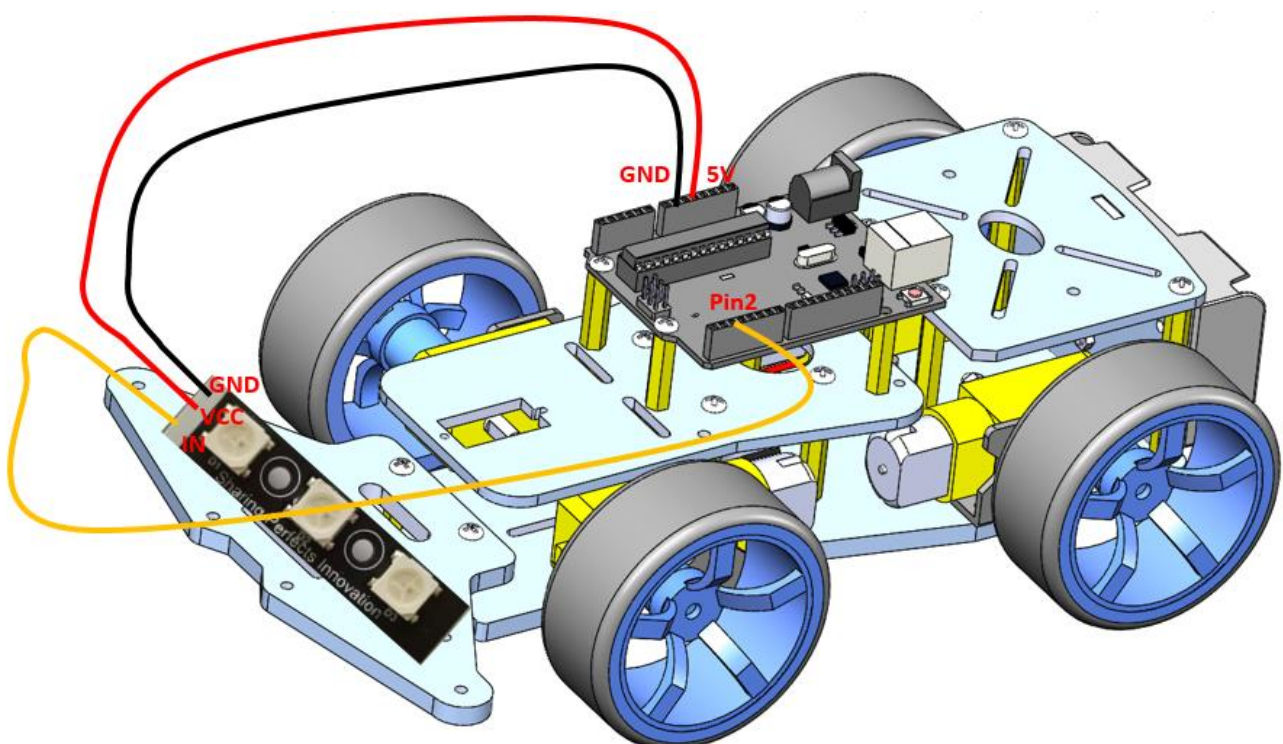
Before powering on, please carefully check whether the connected circuit has open circuit or short circuit, especially GND and 5V, GND and 3.3V, must not be short-circuited.A short circuit can cause high current in your circuit, create excessive component heat and cause permanent damage to your hardware!

4. Upload code and test the ws2812 module

Install Adafruit_NeoPixel.h libraries

In order to establish communication with the sw2812 module, we need to install the Adafruit_NeoPixel.h library first, so that the UNO board can issue commands to control the LED lights of the sw2812 module.

Open to click Arduino IDE，then click "Sketch">"Include library">"Manage Libraries..."，Wait for the library manager to download the library index and update the list of installed libraries.
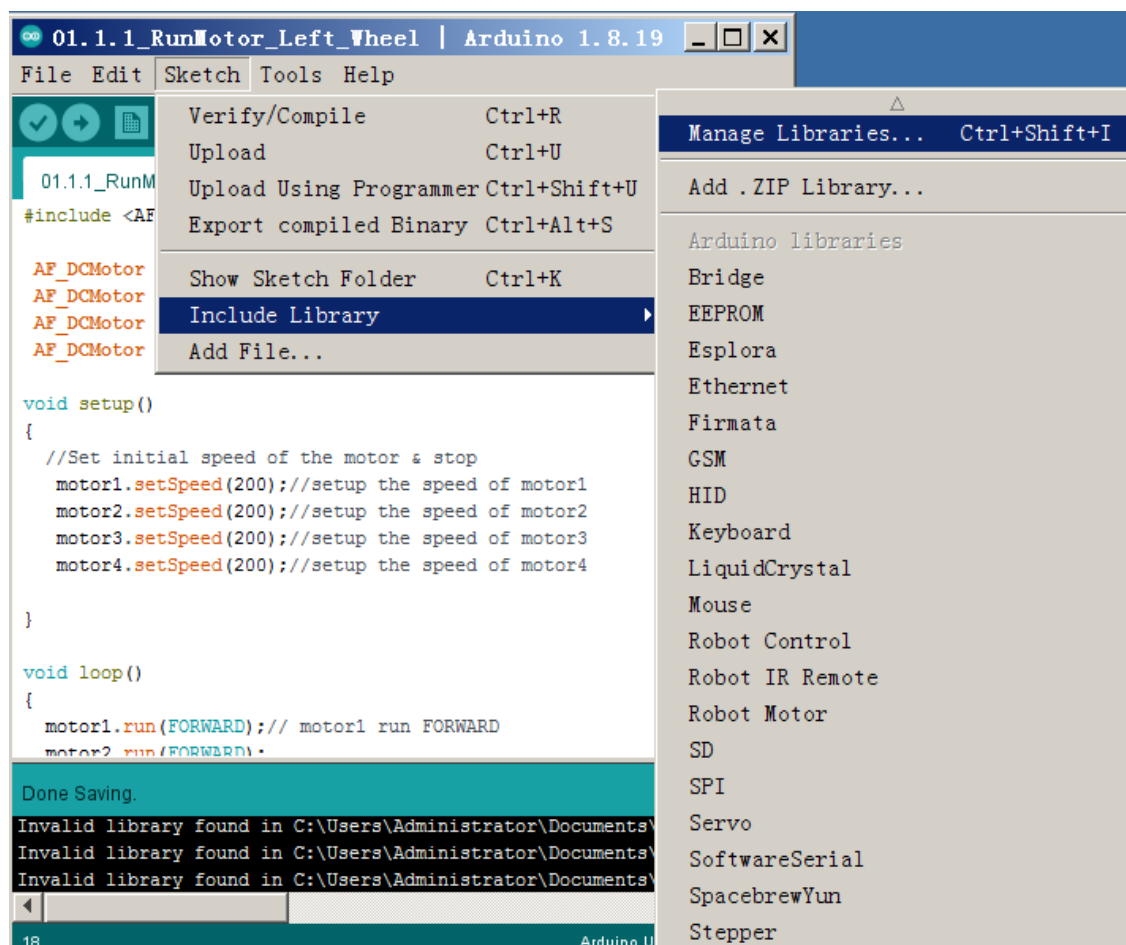
Type "NeoPixel" to filter your search. Look for the Adafruit_NeoPixel library provided by Adafruit. Once found, click the entry, select the latest version, and finally click Install. Wait for the automatic installation to complete, and the SW2812 module library file is ready.
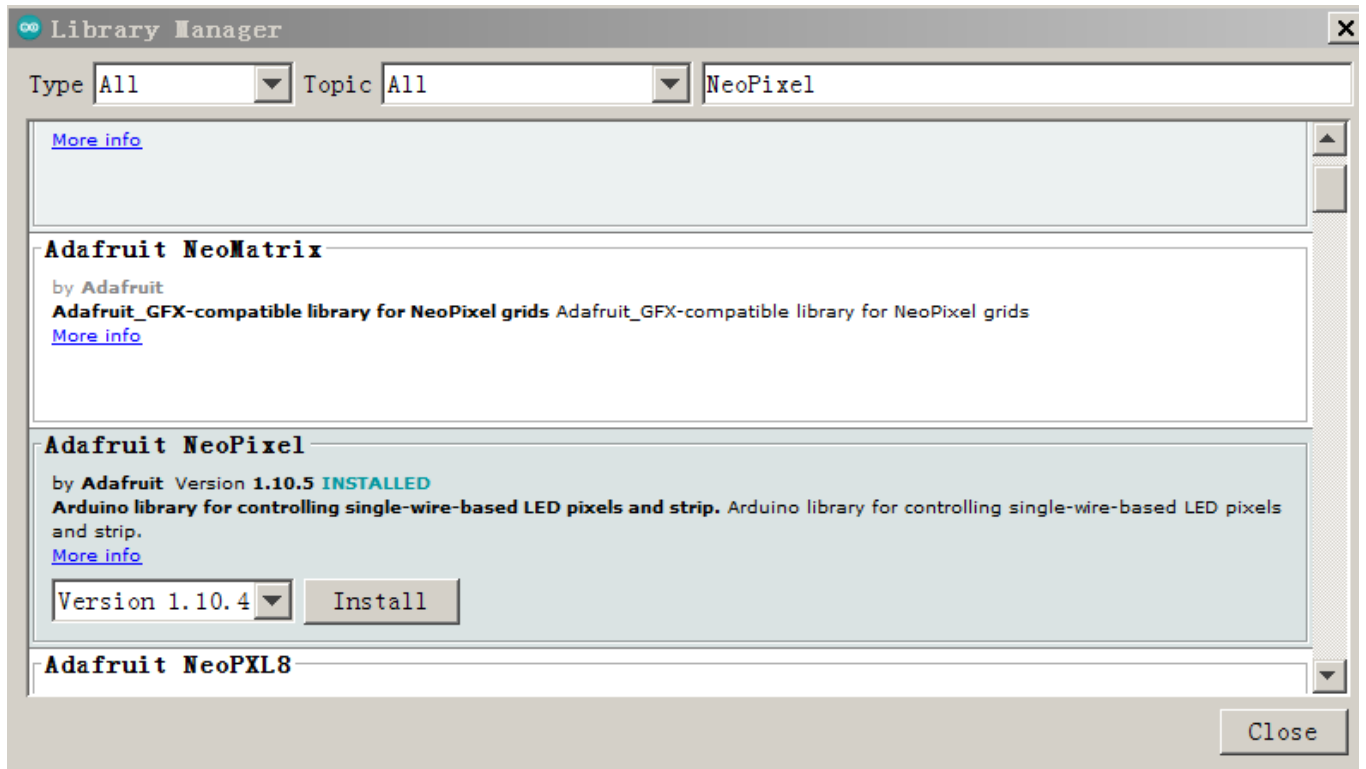
Click "File"---"Open" in the IDE interface, select the code in the path "4WD Car Chassis Kit Tutorial\ Sketches\ 03.1_Testing WS2812 LED". After the compilation is successful, connect the UNO board on the 4WD car body to the computer with a USB cable, upload the code.

After the upload is successful, 3 LED lights on the sw2812 module are cyclically changed in the order of red---green---blue---white---off.

You can also set different color changes according to your own preferences.

The corresponding relationship between program setting value and RGB is shown in the following figure

You can find more rgb values and colors here
https://www.rapidtables.com/web/color/RGB_Color.html

Code as below:

```
/**********************************************************************
  Product     : Cokoino 4WD Car chassis kit
  Auther      : www.cokoino.com
  Modification: 2022/07/03
**********************************************************************/
#include <Adafruit_NeoPixel.h>
#ifdef __AVR__
#include <avr/power.h> // Required for 16 MHz Adafruit Trinket
#endif

#define LED_PIN 2
#define LED_COUNT 3
#define Led_delay 100
Adafruit_NeoPixel strip(LED_COUNT, LED_PIN, NEO_GRB + NEO_KHZ800);

void setup()
{
  #if defined(__AVR_ATtiny85__) && (F_CPU == 16000000)
    clock_prescale_set(clock_div_1);
  #endif
  // END of Trinket-specific code.
  strip.begin();           // INITIALIZE NeoPixel strip object (REQUIRED)
  strip.show();            // Turn OFF all pixels ASAP
  strip.setBrightness(50); // Set BRIGHTNESS to about 1/5 (max = 255)
  Serial.begin(9600);
}

void loop()
{
  SW2812_Test();
}

void SW2812_Test()
{
  colorWipe(strip.Color(255,   0,    0), Led_delay); // Red
  colorWipe(strip.Color(  0, 255,    0), Led_delay); // Green
  colorWipe(strip.Color(  0,   0,  255), Led_delay); // Blue
  colorWipe(strip.Color(  255,255, 255), Led_delay); // While
  //rainbow(10);               // Flowing rainbow cycle along the whole strip
  //theaterChaseRainbow(50); // Rainbow-enhanced theaterChase variant
}

void colorWipe(uint32_t color, int wait) {
  for(int i=0; i<strip.numPixels(); i++) { // For each pixel in strip...
    strip.setPixelColor(i, color);         //  Set pixel's color (in RAM)
    strip.show();                          //  Update strip to match
```

```cpp
      delay(wait);                              //   Pause for a moment
    }
  }
}

void theaterChase(uint32_t color, int wait)
{
  for(int a=0; a<10; a++) {  // Repeat 10 times...
    for(int b=0; b<3; b++) { //  'b' counts from 0 to 2...
      strip.clear();         //    Set all pixels in RAM to 0 (off)
      // 'c' counts up from 'b' to end of strip in steps of 3...
      for(int c=b; c<strip.numPixels(); c += 3) {
        strip.setPixelColor(c, color); // Set pixel 'c' to value 'color'
      }
      strip.show(); // Update strip with new contents
      delay(wait);  // Pause for a moment
    }
  }
}

void rainbow(int wait)
{
  for(long firstPixelHue = 0; firstPixelHue < 5*65536; firstPixelHue += 256)
  {
    strip.rainbow(firstPixelHue);
    strip.show(); // Update strip with new contents
    delay(wait);  // Pause for a moment
  }
}

void theaterChaseRainbow(int wait)
{
  int firstPixelHue = 0;     // First pixel starts at red (hue 0)
  for(int a=0; a<30; a++) {  // Repeat 30 times...
    for(int b=0; b<3; b++) { //  'b' counts from 0 to 2...
      strip.clear();         //    Set all pixels in RAM to 0 (off)
      for(int c=b; c<strip.numPixels(); c += 3)
      {
        int      hue   = firstPixelHue + c * 65536L / strip.numPixels();
        uint32_t color = strip.gamma32(strip.ColorHSV(hue)); // hue -> RGB
        strip.setPixelColor(c, color); // Set pixel 'c' to value 'color'
      }
      strip.show();                   // Update strip with new contents
      delay(wait);                    // Pause for a moment
      firstPixelHue += 65536 / 90; // One cycle of color wheel over 90 frames
    }
  }
}
```

5. Make your suggestion and get support

THANK YOU for participating in this learning experience!

If you find errors, omissions or you have suggestions and/or questions about this lesson, please feel free to contact us: cokoino@outlook.com

We will make every effort to make changes and correct errors as soon as feasibly possible and publish a revised version.

If you want to learn more about Arduino, Raspberry Pi, Smart Cars, Robotics and other interesting products in science and technology, please continue to visit our website. We will continue to launch fun, cost-effective, innovative and exciting products.
 http://cokoino.com/

Thank you again for choosing Cokoino products.