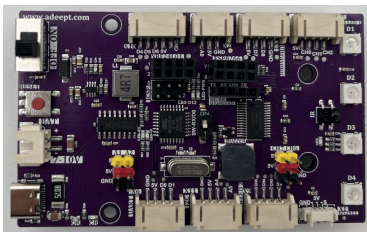




Lesson 8 Light up the WS2812

8.1 Overview

This lesson focuses on how to use the WS2812 LEDs with the Adeept Robot Control Board. By the end of this lesson, you'll be able to make the WS2812 LEDs display different lighting effects, such as breathing lights and flowing lights, through programming.

8.2 Required Components

Components	Quantity	Picture
Adeept Robot Control Board	1	
Type-C USB Cable	1	
WS2812 LED	1	

8.3 Principle Introduction

Red, green, and blue are called the three primary colors. When you combine these three primary colors of different brightness, it can produce almost all kinds of visible light.

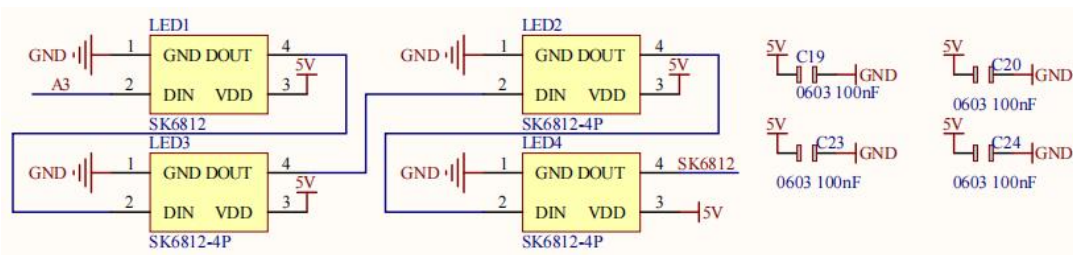
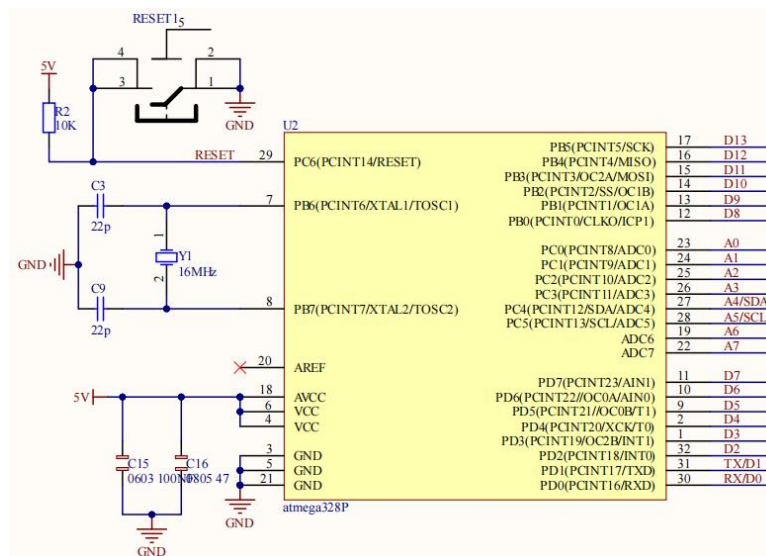
WS2812 RGB module is a low-power RGB tri-color lamp with integrated current control chip. Its appearance is the same as a 5050LED lamp bead, and each element is a pixel. The pixel contains an intelligent digital interface data latch signal shaping amplifier driving circuit, and also contains a high-precision internal oscillator and a 12V high-voltage programmable constant current control part, which effectively guarantees that the color of the pixel light is highly consistent.

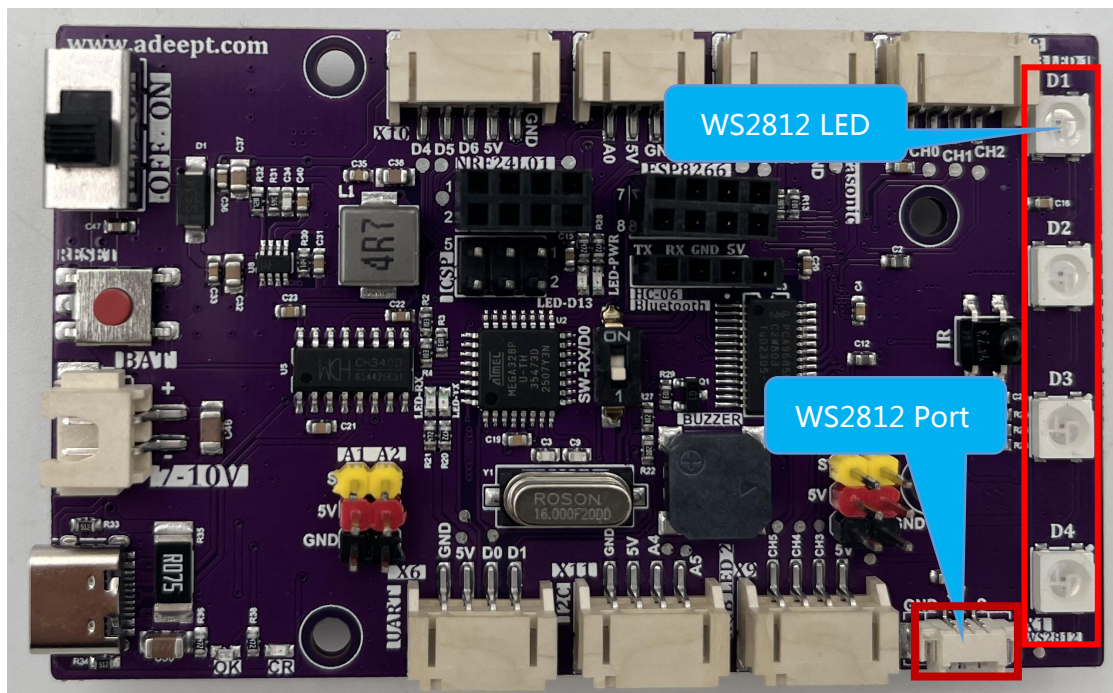
Four WS2812 LEDs are integrated on the Adeept Robot Control Board, and the number of WS2812 LEDs can also be increased through the WS2812 LED port.



WS2812 LED is a very commonly used module on our robot products. There are three WS2812 LEDs on each module. Pay attention to the direction of the signal line when connecting. The signal line needs to be connected to the "IN" port of WS2812 LED after being led from the Adeept Robot Control Board WS2812 Port. When the next WS2812 LED needs to be connected, we connect a signal wire drawn from the "OUT" port of the previous WS2812 LED with the "IN" port of the next WS2812 LED.

8.4 Wiring Diagram





8.5 Demonstration

1. Connect your computer and Adept Robot Control Board (Arduino Board) with a USB cable.
2. Open "04_WS2812/BreathingLight/" folder in "/Code" , double-click "BreathingLight.ino" .

```
BreathingLight | Arduino 1.8.19
File Edit Sketch Tools Help

BreathingLight
#include <Adafruit_NeoPixel.h>

#define LED_PIN    A3      // WS2812 connect to pin A3 .
#define NUM_LEDS   7       // LED number.
Adafruit_NeoPixel pixels(NUM_LEDS, LED_PIN, NEO_GRB + NEO_KHZ800);

void setup() {
  pixels.begin();          // Initialize the NeoPixel library.
}

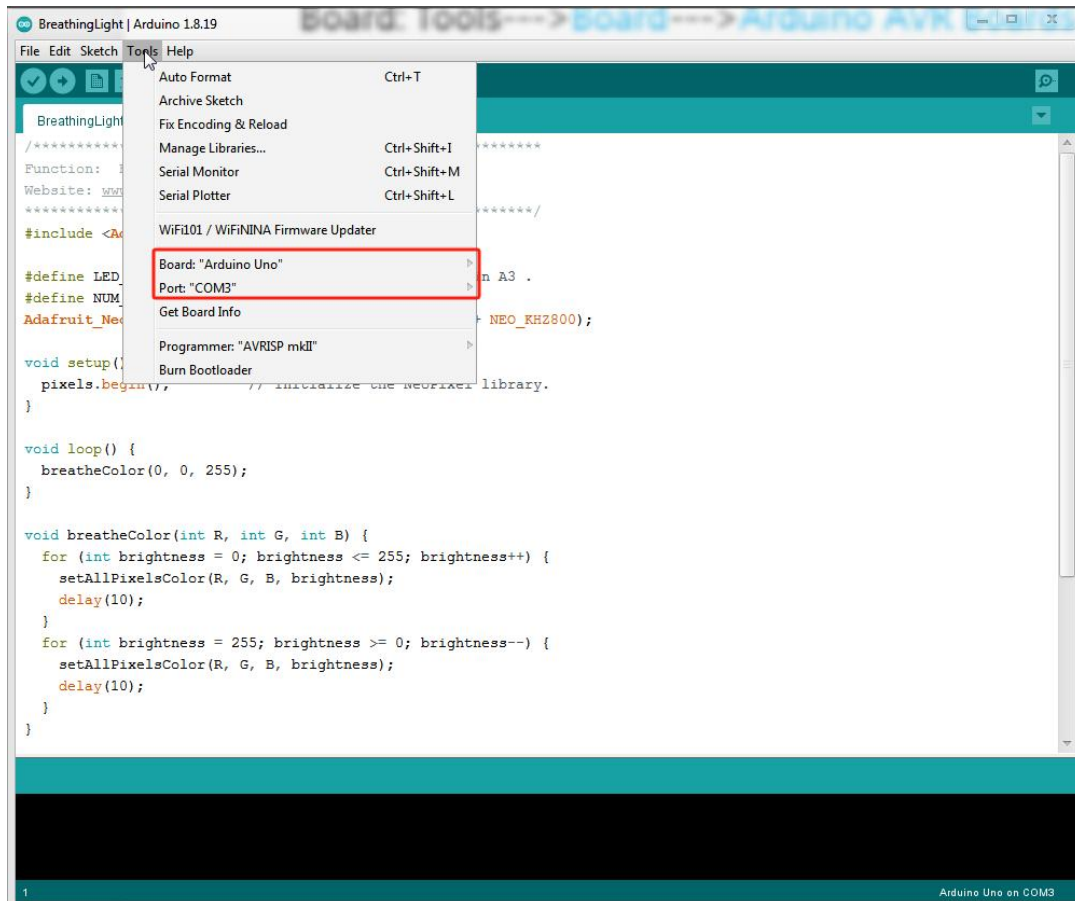
void loop() {
  breatheColor(0, 0, 255);
}
```


3. Select development board and serial port.

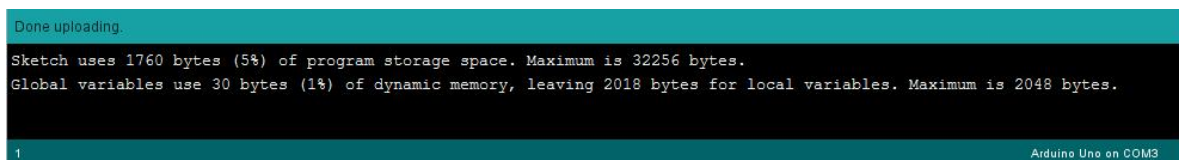
Board: Tools--->Board--->Arduino AVR Boards--->Arduino Uno

Port: Tools --->Port--->COMx

Note: The port number will be different in different computers.

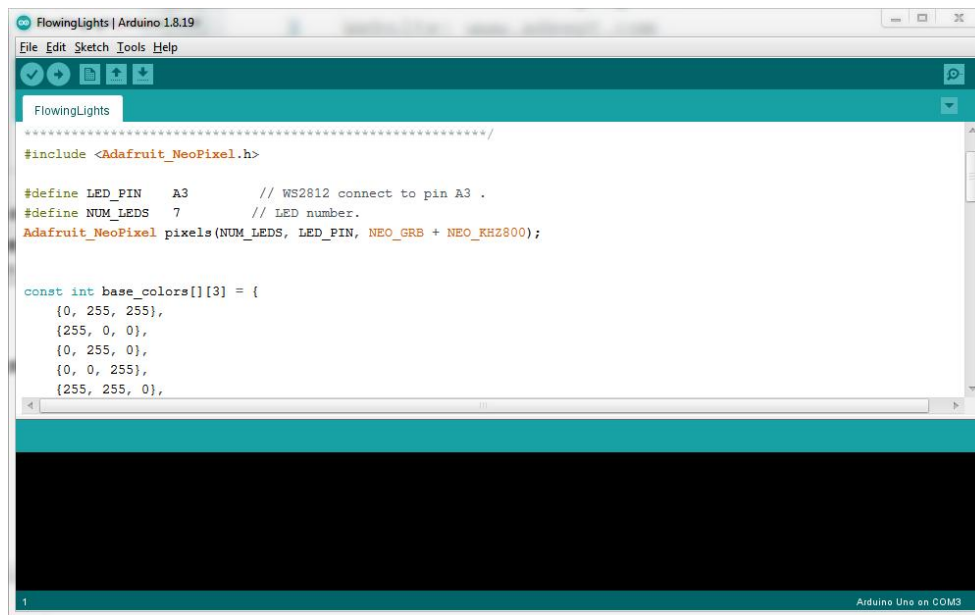



4. After opening, click  to upload the code program to the Arduino. If there is no error warning in the console below, it means that the Upload is successful.

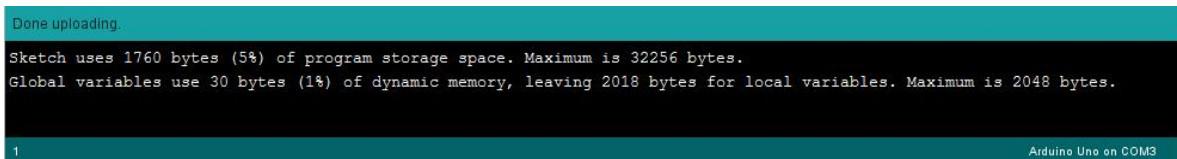


5. After the program runs successfully, the WS2812 LEDs on the AdeepT Robot Control Board will display a breathing light effect. The LEDs will gradually brighten and then dim in a smooth, cyclic manner.

6. Open "04_WS2812/FlowingLights/" folder in "/Code" , double-click "FlowingLights.ino" .



7. After opening, click  to upload the code program to the Arduino. If there is no error warning in the console below, it means that the Upload is successful.



8. After the program runs successfully, the WS2812 LEDs will display a flowing lights effect. The lights will move across the LEDs in a predefined sequence, creating an eye-catching visual display.

8.6 Code

Complete code refer to [BreathingLight.ino](#)

```

01  /*****
02  Function: BreathingLight LED
03  Website: www.adeept.com
04  *****/
05  #include <Adafruit_NeoPixel.h>
06
07  #define LED_PIN    A3          // WS2812 connect to pin A3 .
08  #define NUM_LEDS   7          // LED number.
09  Adafruit_NeoPixel pixels(NUM_LEDS, LED_PIN, NEO_GRB + NEO_KHZ800);
10
11  void setup() {
12    pixels.begin();           // Initialize the NeoPixel library.
13  }
14
15  void loop() {
16    breatheColor(0, 0, 255);

```

```

17 }
18
19 void breatheColor(int R, int G, int B) {
20     for (int brightness = 0; brightness <= 255; brightness++) {
21         setAllPixelsColor(R, G, B, brightness);
22         delay(10);
23     }
24     for (int brightness = 255; brightness >= 0; brightness--) {
25         setAllPixelsColor(R, G, B, brightness);
26         delay(10);
27     }
28 }
29
30 void setAllPixelsColor(int R, int G, int B, int brightness) {
31     for (int i = 0; i < NUM_LEDS; i++) {
32         pixels.setPixelColor(i, scaleColor(R, brightness), scaleColor(G, brightness), scaleColor(B,
33         brightness));
34     }
35     pixels.show();
36 }
37
38 int scaleColor(int color, int brightness) {
39     return (color * brightness) / 255;
40 }

```

Complete code refer to [FlowingLights.ino](#)

```

01 /*****
02 Function: FlowingLights LED
03 Website: www.adeept.com
04 *****/
05 #include <Adafruit_NeoPixel.h>
06
07 #define LED_PIN    A3          // WS2812 connect to pin A3 .
08 #define NUM_LEDS   7           // LED number.
09 Adafruit_NeoPixel pixels(NUM_LEDS, LED_PIN, NEO_GRB + NEO_KHZ800);
10
11
12 const int base_colors[][3] = {
13     {0, 255, 255},
14     {255, 0, 0},
15     {0, 255, 0},
16     {0, 0, 255},
17     {255, 255, 0},
18     {255, 0, 255},
19     {0, 128, 255}
20 };
21
22
23 void generate_color_sequences(int color_sequences[][NUM_LEDS][3]) {
24     for (int i = 0; i < sizeof(base_colors) / sizeof(base_colors[0]); i++) {
25         for (int j = 0; j < NUM_LEDS; j++) {
26             int index = (j + i) % (sizeof(base_colors) / sizeof(base_colors[0]));
27             color_sequences[i][j][0] = base_colors[index][0];
28             color_sequences[i][j][1] = base_colors[index][1];
29             color_sequences[i][j][2] = base_colors[index][2];
30         }

```



```
31     }
32 }
33
34 void setup() {
35     pixels.begin();          // Initialize the NeoPixel library.
36     pixels.setBrightness(5); // Set WS2812 LED brightness.
37 }
38
39 void loop() {
40     int color_sequences[sizeof(base_colors) / sizeof(base_colors[0])][NUM_LEDS][3];
41     generate_color_sequences(color_sequences);
42
43     for (int s = 0; s < sizeof(base_colors) / sizeof(base_colors[0]); s++) {
44         for (int i = 0; i < NUM_LEDS; i++) {
45             pixels.setPixelColor(i, pixels.Color(color_sequences[s][i][0], color_sequences[s][i][1],
46 color_sequences[s][i][2]));
47         }
48         pixels.show();
49         delay(300);
50     }
51 }
52
53
54 void WS2812Color(int num, int R, int G, int B){
55     pixels.setPixelColor(num,pixels.Color(R,G,B));
56     pixels.show();
57 }
58
59 void WS2812ColorAll(int R, int G, int B){
60     for(int i=0; i<=NUM_LEDS; i++){
61         pixels.setPixelColor(i,pixels.Color(R,G,B));
62     }
63     pixels.show();
64 }
```

Code explanation

BreathingLight.ino

Initialization Stage:

At the beginning of the code, the 'Adafruit NeoPixel' library was introduced, which is used to manipulate the WS2812 LED light bar. Created an instance of the Adafruit_NeoPixel class called 'pixels' and initialized it, including specifying the number of LEDs, connection pins, and communication protocol.

Loop Control Process:

A loop is used to create the breathing effect. This loop gradually increases or decreases the RGB values of the LEDs over time. For example, it could use a sine-wave-like function to smoothly change the brightness, making the LEDs appear to breathe.

In the Arduino environment, the 'setup' and 'loop' functions will continue to run, and when the program stops running or the Arduino board loses power, the related resources will be automatically released.

FlowingLights.ino

Initialization Stage:

At the beginning of the code, the 'Adafruit NeoPixel' library was introduced, which is used to manipulate the WS2812 LED light bar. Created an instance of the Adafruit_NeoPixel class called 'pixels' and initialized it, including specifying the number of LEDs, connection pins, and communication protocol.

Loop Control Process:

Loop is used to control the movement of flowing lights. It updates the status of the LED according to the defined order and moves the "flow" of light on the LED strip.

In the Arduino environment` The setup and loop functions will continue to run, and when the program stops running or the Arduino board loses power, the related resources will be automatically released.