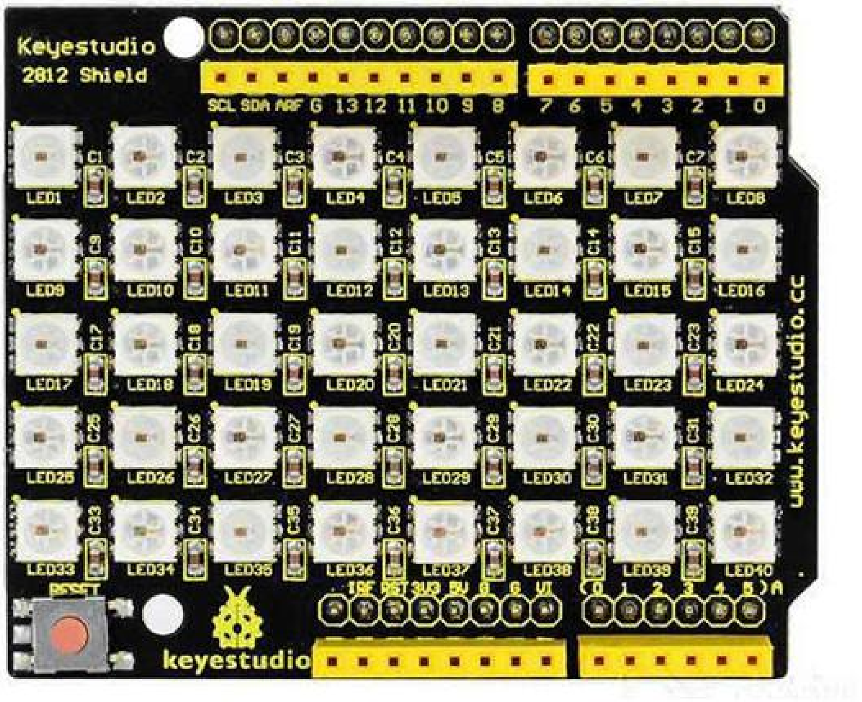
## keyestudio 40 RGB LED 2812 Pixel Matrix Shield


Keyestudio
2812 Shield
SCL SDA ARF G 13 12 11 10 9 8
7 6 5 4 3 2 1 0
LED1 LED2 LED3 LED4 LED5 LED6 LED7 LED8
LED9 LED10 LED11 LED12 LED13 LED14 LED15 LED16
LED17 LED18 LED19 LED20 LED21 LED22 LED23 LED24
LED25 LED26 LED27 LED28 LED29 LED30 LED31 LED32
LED33 LED34 LED35 LED36 LED37 LED38 LED39 LED40
www.keyestudio.cc
RESET
IRF RST 3V3 5V G G VI
0 1 2 3 4 5 A
keyestudio

**Introduction:**

keyestudio 2812 shield adopts stackable design, compatible with UNO board.

It is an intelligent controlled LED light source that the control circuit and RGB chip are integrated into a 5050 SMD component.

Each pixel interior not only includes intelligent digital port data latch and signal reshaping amplification drive circuit, but also includes a precision internal oscillator and a 12V programmable constant current control part, effectively ensuring the highly consistency of the pixel point light color.

The data transfer protocol uses single RZ (return-to-zero) communication mode. After the pixel power-on to reset, the DI port receives data from controller, the first 24bit data is extracted by the first pixel,and then sent to the data latch inside the pixel.

LED has the advantages of low driving voltage, environmental-friendly, energy saving, high brightness, large scattering angle, good consistency, long life span, etc.

**Specifications of Single LED:**

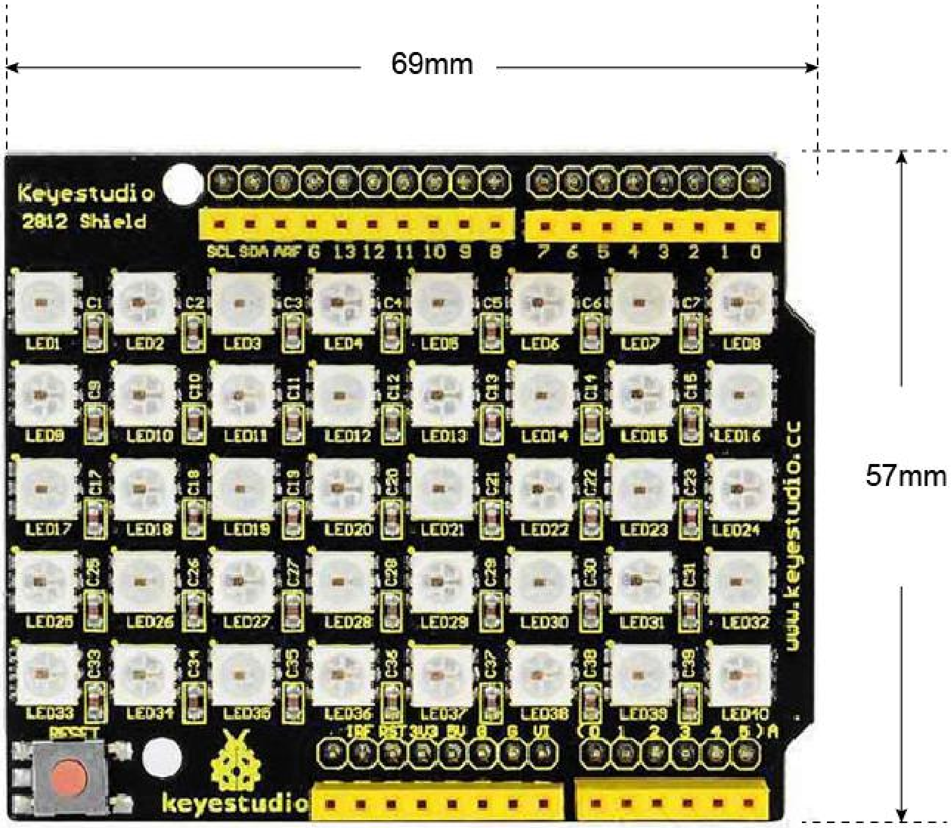
1. Anti-reverse protection circuit; the reverse power connection will not damage the internal IC of the LED.
2. IC control circuit and LED point light source use the same power supply.
3. Control circuit and the RGB chip are integrated into a 5050 SMD component, forming a complete control of pixel point.
4. Built-in signal reshaping circuit; signals received will be wave-reshaped first and then output to the next driver, ensuring wave-form distortion not accumulate.
5. Built-in power-on reset circuit and power-down reset circuit.
6. Each pixel of the three primary color can achieve 256 brightness display; complete full color display of 16777216 colors; scan frequency no less than 400Hz/s.
7. Serial cascade interface, to complete the reception and decoding of data via a signal line.
8. When transmission distance between two arbitrary points is no more than five meters, no extra circuit needed.
9. When the refresh rate is 30fps, cascade number no less than 1024 points.
10. Data sending speed can reach 800Kbps.
11. The color of the light is highly consistent, cost-effective.

**Advantages:**

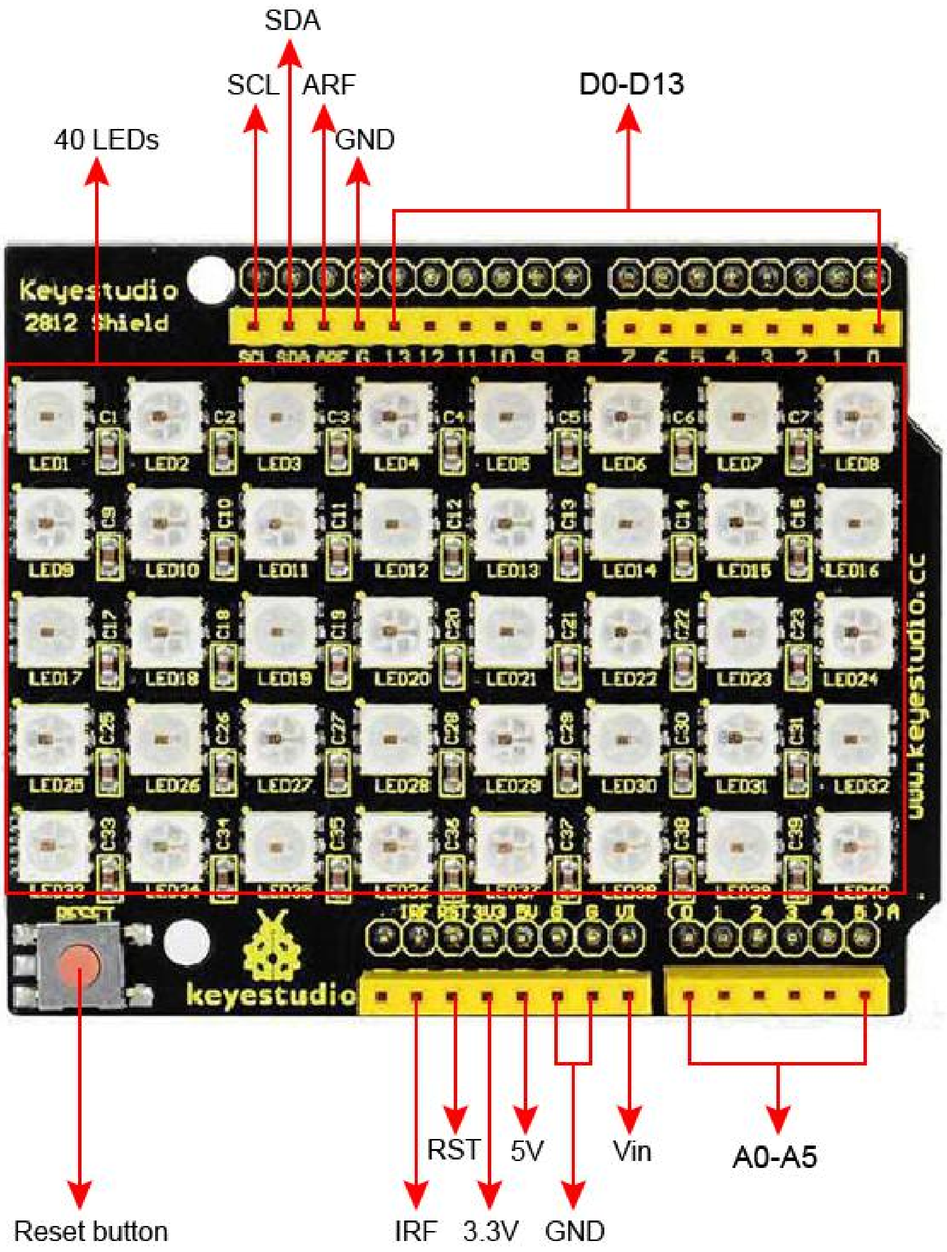
1. LED with built-in IC is brighter than common LED
2. High consistency of RGB chip inside all LEDs
3. Reliable performance of built-in driver IC
4. Use hard plastic packaging, preventing press damage.

**Details:**

- Dimensions: 69mm x 57mm x 18mm
- Weight: 21.1g

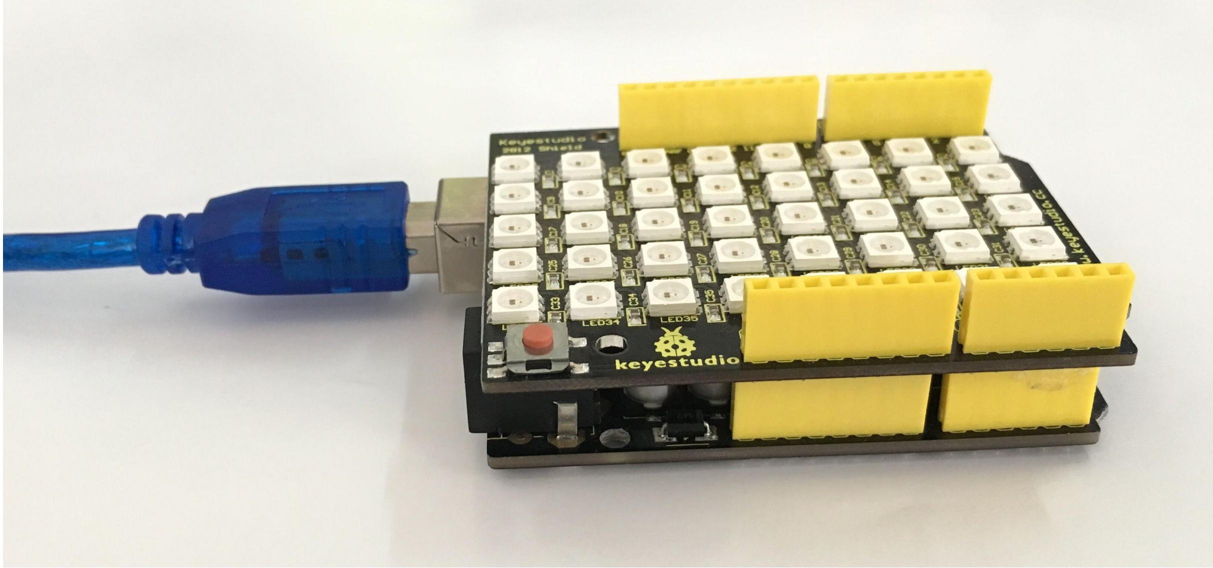


**PINOUTS:**


SDA
SCL
ARF
GND
D0-D13
40 LEDs
Keyestudio 2812 Shield
Reset button
IRF
RST
3.3V
5V
GND
Vin
A0-A5
www.keyestudio.cc

### Simple Hookup

Simply stack the shield on the UNO board.



### Sample Code

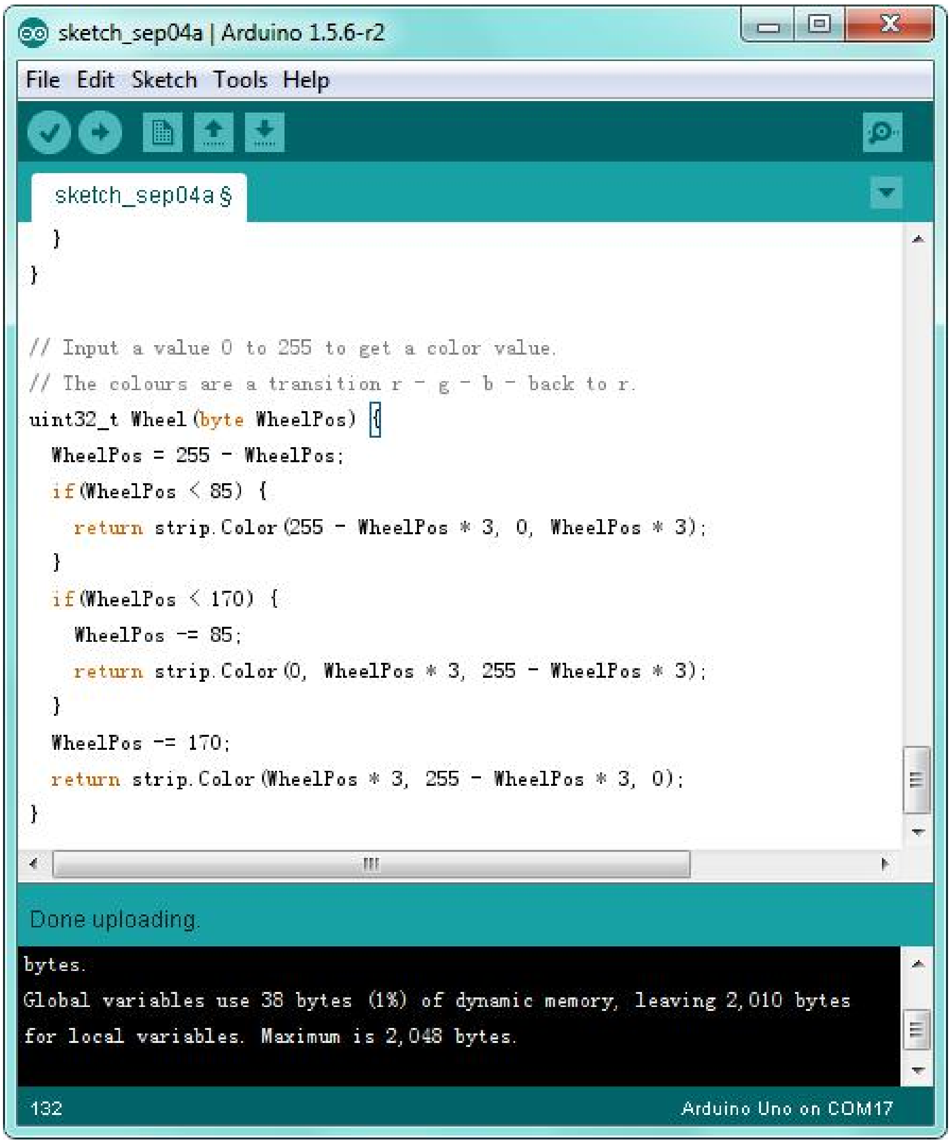
- How to download and install Arduino IDE and driver.
- Or click here to download the code or directly copy the code below.
- Click here to download the libraries
- **Note:** Before compile the code, do remember to add the necessary libraries inside the libraries directory of Arduino IDE.

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

#define PIN 13

// 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)
Adafruit_NeoPixel strip = Adafruit_NeoPixel(40, 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.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
  // 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 (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 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
      }
    }
  }
}

// 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);
}
```

*****************************************************************



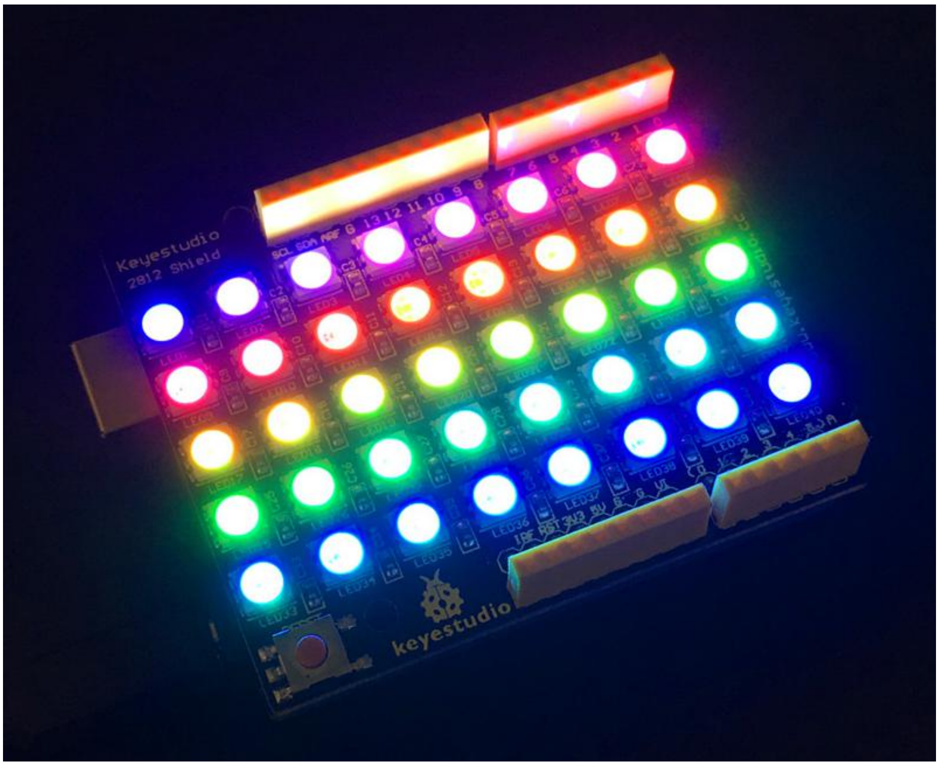
```
  }
}

// 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);
}
```

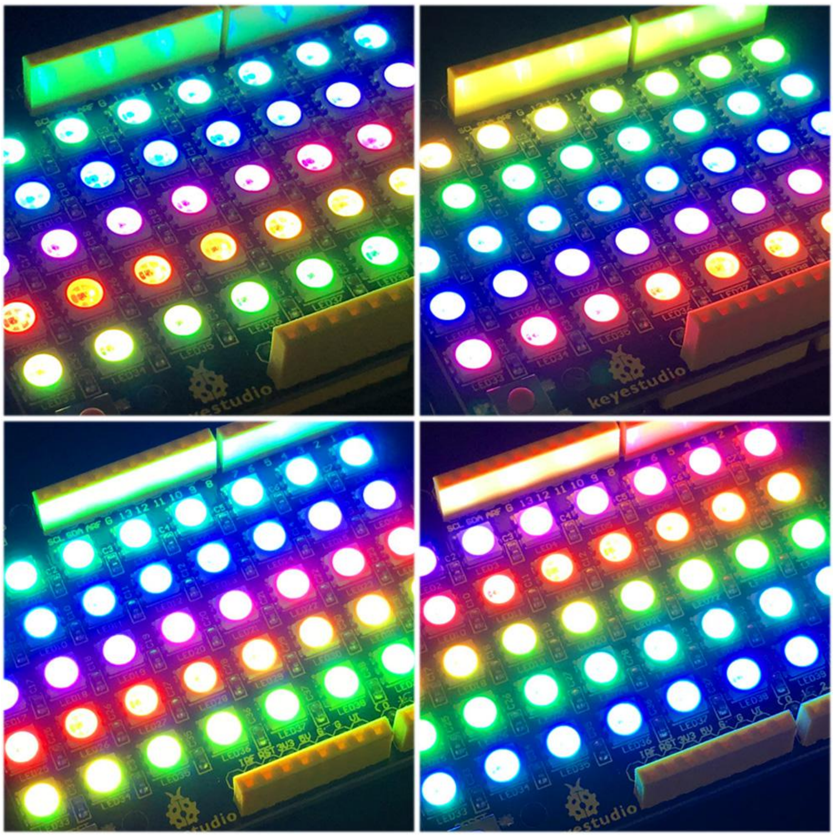
Done uploading.

```
bytes.
Global variables use 38 bytes (1%) of dynamic memory, leaving 2,010 bytes
for local variables. Maximum is 2,048 bytes.
```

### Test Result

Done uploading the code above to the board, you should see the LED matrix blink with shiny colors.





**Resource Links**

Download code and libraries:

https://fs.keyestudio.com/KS0163