Gravity: Flexible 16x16 RGB LED Matrix SKU: DFR0463

Introduction

Good news everyone! We have a new member to the Gravity family - The Flexible 16x16 RGB LED Matrix! This module is a square panel with a soldered XH2.54 3-pin interface with pin mappings that perfectly match the Arduino I/O expansion shield.
This flexible 16x16 RGB full-color LED matrix module is based on SK6812 or WS2812 intelligent control LEDs. Each LED can be independently addressed with RGB pixels that can achieve 256 levels of brightness. That's 16777216 colors in total with a scanning frequency no less than 400Hz!
The 16x16 RGB LED Matrix is single wire control board. We have customized a special cable to make it compatible with Arduino I/O expansion shields without any soldering. The module also supports cascading control. All you need to do is connect Din to the DOUT port. In combination with the open source Arduino library, you can control an entire array of LEDs using just one pin!

Note: Each LED requires a maximum current of 18mA. The highest current arrives 2A. it is recommended to use the DFRobot Gravity IO expansion board to power multiple LEDs matrix via the Servo Power Port..

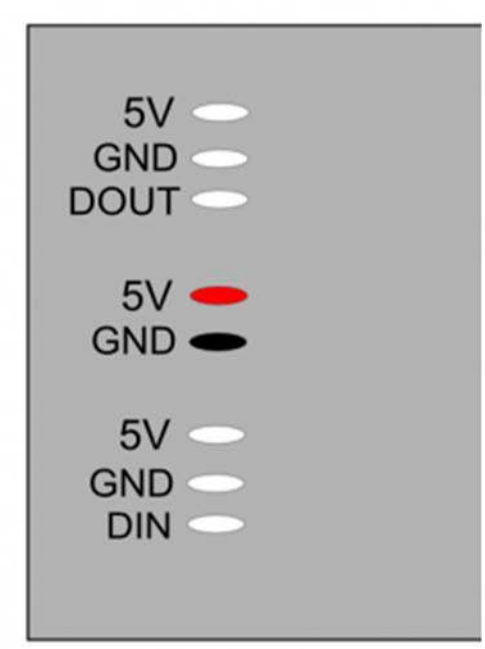
Features

- Full Color RGB
- Supports Cascade Control
- Compatible with Gravity Interface
- Supports External Power Supply

Specification

- Operating Voltage: 5V
- LED Type: SK6812/WS2812
- Communication Interface: Gravity 3Pin interface
- Operating Temperature: -25 ~ +80 ℃
- Dimension: 160*160 mm/6.3*6.3 inches
- Weight: 172g

Board Overview



Num	Label	Description
1	5V	Power +
2	GND	Power -
3	DIN	Data IN
4	DOUT	Data OUT

⚠ Note: All “5V” ports are interconnecting. Please be careful with the module power supply.

Tutorial

In this tutorial, we'll show you how to light the Matrix in 5 minutes.

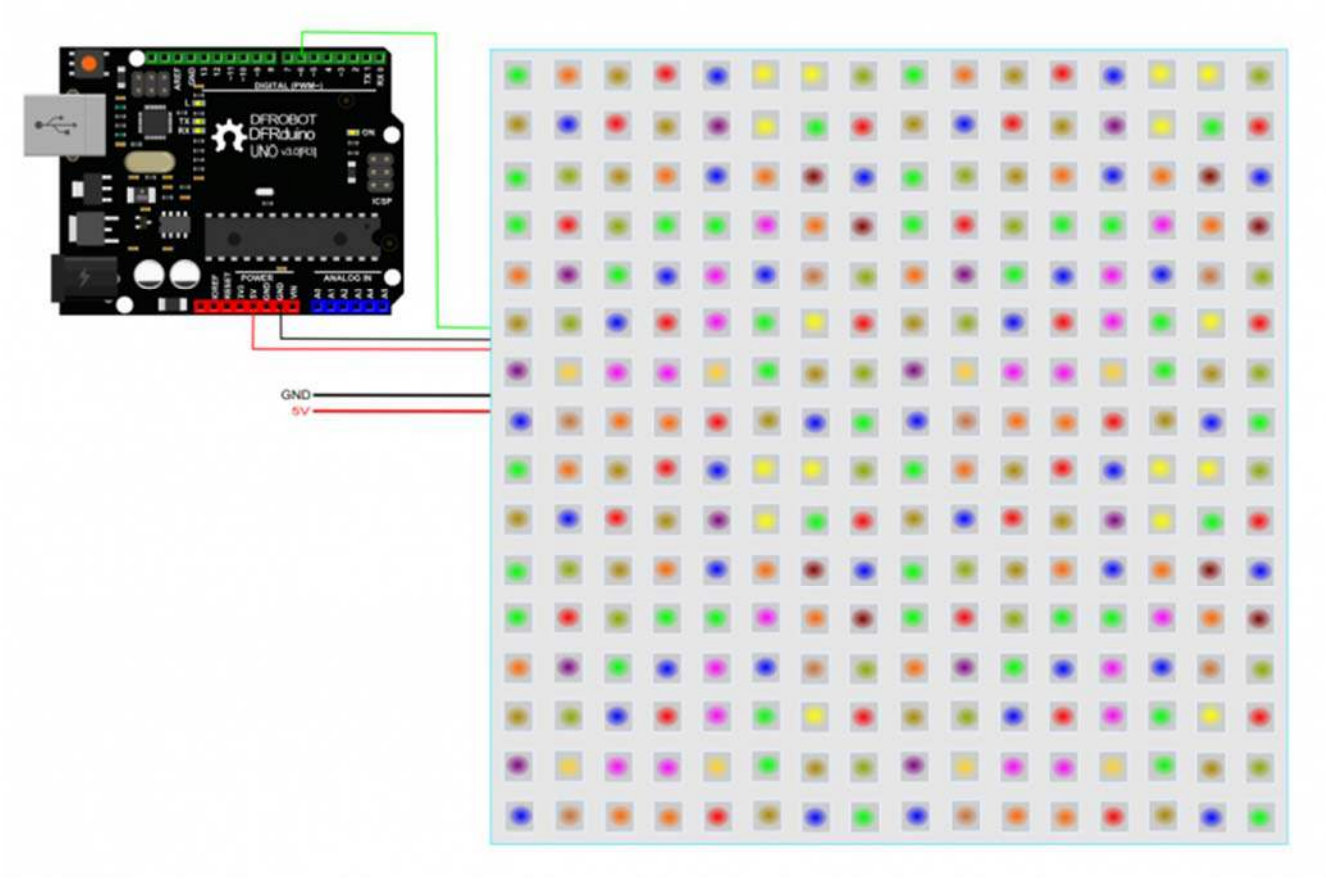
Requirements

- **Hardware**
  DFRduino UNO x 1
  16x16 RGB LED Matrix x1

- **Software**

  Arduino IDE, Click to Download Arduino IDE from Arduino®
  https://www.arduino.cc/en/Main/Software%7C

Connection Diagram



Flexible 16x16 RGB LED Matrix SKU: DFR0463

Sample Code

In this section, we'll use Adafruit_NeoPixel RGB LED Lbrary. How to install Libraries in Arduino IDE
https://github.com/adafruit/Adafruit_NeoPixel https://www.arduino.cc/en/Guide/Libraries#.UxU8mdzF9H0%7C

```
#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(256, 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 ar
e 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
//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);
}
```

Expected Results

FAQ

For any questions, advice or cool ideas to share, please visit the **DFRobot Forum**.