Datenblatt RBS15043

32Bit RGB LED Ring WS2812 5V ähnl. Neopixel

Inhaltsverzeichnis

Beschreibung 3
Sicherheitshinweise 3
Technische Spezifikationen 4
Pinbelegung & Anschlussplan 5
Beispielcode Adafruit Neopixel Library 6

Beschreibung

Großer LED-Ring bestehend aus 32 WS2812b RGB-Modulen; Der Ring hat einen Durchmesser von ca. 11 cm.

Jede LED ist mit Hilfe eines Microcontrollers (z.B. Arduino) einzeln ansteuerbar, so lassen sich super Funktionen oder Effekte wie Lauflichter, buntes Aufblitzen und vieles mehr programmieren. Über den Do-Pin (Data-Out) können weitere RGB-Ringe angeschlossen und somit kaskadiert werden.

Der LED-Ring ist kompatibel zu der Adafruit Neopixel-Library die das Programmieren stark vereinfacht und bereits viele tolle Beispiele mitbringt. Weitere Beispiele und Projekte und Videos findet man bei Adafruit oder mit einer Youtube suche nach "Neopixel Adafruit".

Sicherheitshinweise

Stromversorgung

- Verwenden Sie eine stabile 5V Stromquelle mit ausreichender Leistung. Berechnen Sie den Strombedarf basierend auf der LED-Anzahl (ca. 60 mA pro LED bei voller Helligkeit).
- Schließen Sie einen Kondensator (100-1000 µF) zwischen VCC und GND an, um Spannungsschwankungen zu reduzieren.

Anschluss und Verkabelung

- Achten Sie auf die korrekte Polarität beim Anschließen. Vertauschen von VCC und GND kann die LEDs beschädigen.
- Verwenden Sie Kabel mit ausreichendem Querschnitt, besonders bei längeren Strecken oder hohen Strömen.
- Isolieren Sie alle Anschlüsse sorgfältig, um Kurzschlüsse zu vermeiden.

Betrieb

- Vermeiden Sie direkte Blicke in die LEDs bei voller Helligkeit, um Ihre Augen zu schützen.
- Begrenzen Sie die Helligkeit in Software auf etwa 50%, um die Lebensdauer der LEDs zu verlängern und Überhitzung zu vermeiden.
- Sorgen Sie für ausreichende Belüftung, besonders bei Dauerbetrieb oder in geschlossenen Gehäusen.

Umgebungsbedingungen

- Schützen Sie den LED-Ring vor Feuchtigkeit und Staub, es sei denn, er ist speziell dafür ausgelegt (IP-Schutzklasse beachten).
- Vermeiden Sie den Einsatz in Umgebungen mit extremen Temperaturen. Die typische Betriebstemperatur liegt zwischen -20°C und 50°C.

Wartung und Handhabung

- Trennen Sie immer die Stromversorgung, bevor Sie Änderungen oder Wartungsarbeiten vornehmen.
- Behandeln Sie den LED-Ring vorsichtig, um mechanische Beschädigungen zu vermeiden.
- Reinigen Sie den Ring nur im ausgeschalteten Zustand mit einem trockenen oder leicht feuchten Tuch.

Technische Spezifikationen

Allgemein

- LED-Typ: WS2812B (5050 SMD, RGB)
- Anzahl der LEDs: 32
- Betriebsspannung: DC 5V
- Steuerprotokoll: 1-Wire

Elektrische Eigenschaften

- Stromverbrauch (max): 640 mA (20 mA pro LED)
- Datenübertragungsrate: 800 Kbps

Optische Eigenschaften

- Abstrahlwinkel: 120°
- Helligkeit: 18-20 Lumen pro LED
- Farbtiefe: 24-Bit (16.777.216 Farben)

Abmessungen

- Außendurchmesser: 11 cm
- Innendurchmesser: 9,5 cm
- PCB-Dicke: 1,6 mm

Besonderheiten

- Einzeln adressierbare LEDs
- Integrierter Controller in jeder LED
- Kaskadierbar für größere Installationen
- Kompatibel mit gängigen Mikrocontrollern (z.B. Arduino, Raspberry Pi)

Empfohlenes Zubehör

- 5V Netzteil (mind. 1A)
- Levelshifter für 3,3V Mikrocontroller
- Kondensator (1000 µF) zur Glättung der Stromversorgung

Kompatible Bibliotheken

- Adafruit NeoPixel
- FastLED
- NeoPixelBus

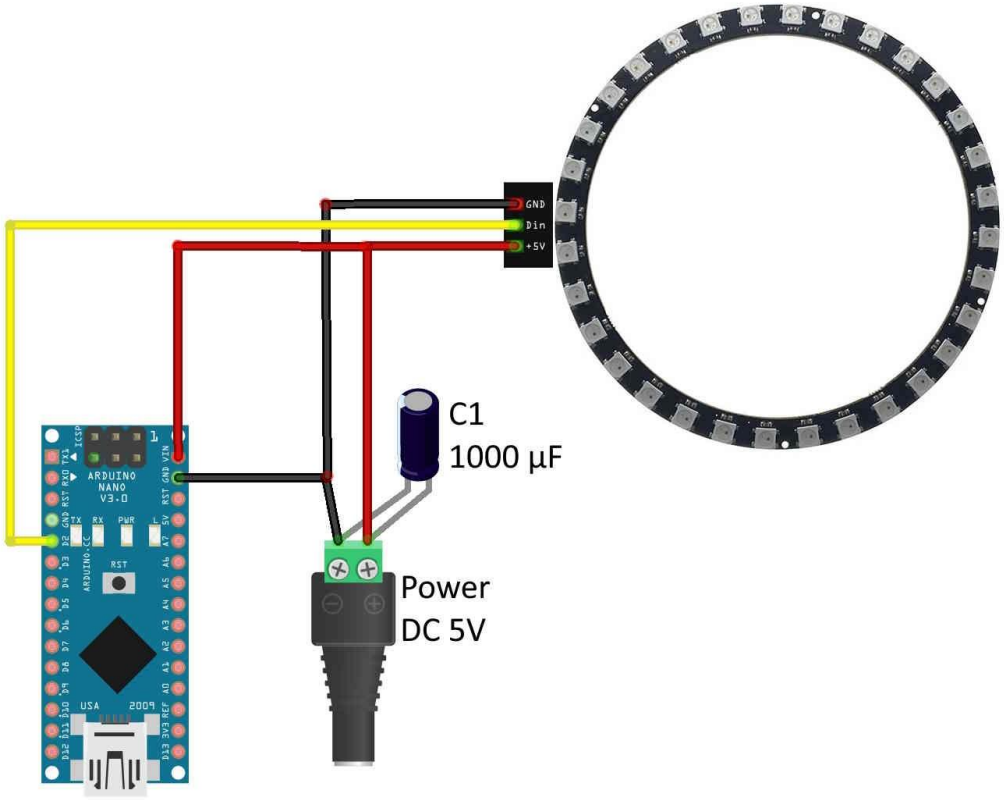
Anwendungsbeispiele

- Uhren und Zeitanzeigen
- Fortschrittsanzeigen
- Ambiente-Beleuchtung
- Interaktive Kunstinstallationen

Pinbelegung & Anschlussplan

Pinbelegung

- VCC: 5V Stromversorgung
- GND: Masse
- DIN: Dateneingang
- DOUT: Datenausgang (für Kaskadierung)



Der Kondensator dient zum Glätten der Eingangsspannung und Abfangen von Spannungsspitzen.

Beispielcode Adafruit Neopixel Library

Dieser Code wurde auf einem Uno R3 und Nano V3.0 Board getestet.

```
#include <Adafruit_NeoPixel.h>
#ifdef __AVR__
 #include <avr/power.h> // Required for 16 MHz Adafruit Trinket
#endif

// Which pin on the Arduino is connected to the NeoPixels?
// On a Trinket or Gemma we suggest changing this to 1:
#define LED_PIN    2

// How many NeoPixels are attached to the Arduino?
#define LED_COUNT 32

// Declare our NeoPixel strip object:
Adafruit_NeoPixel strip(LED_COUNT, LED_PIN, NEO_GRB + NEO_KHZ800);
// Argument 1 = Number of pixels in NeoPixel strip
// Argument 2 = Arduino pin number (most are valid)
// Argument 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)


// setup() function -- runs once at startup ----------------------------
---

void setup() {
  // These lines are specifically to support the Adafruit Trinket 5V 16
MHz.
  // Any other board, you can remove this part (but no harm leaving it):
#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)
}


// loop() function -- runs repeatedly as long as board is on ------------
---

void loop() {
  // Fill along the length of the strip in various colors...
  colorWipe(strip.Color(255,   0,   0), 50); // Red
```

```
  colorWipe(strip.Color(  0, 255,   0), 50); // Green
  colorWipe(strip.Color(  0,   0, 255), 50); // Blue

  // Do a theater marquee effect in various colors...
  theaterChase(strip.Color(127, 127, 127), 50); // White, half brightness
  theaterChase(strip.Color(127,   0,   0), 50); // Red, half brightness
  theaterChase(strip.Color(  0,   0, 127), 50); // Blue, half brightness

  rainbow(10);             // Flowing rainbow cycle along the whole strip
  theaterChaseRainbow(50); // Rainbow-enhanced theaterChase variant
}


// Some functions of our own for creating animated effects -------------
---

// Fill strip pixels one after another with a color. Strip is NOT cleared
// first; anything there will be covered pixel by pixel. Pass in color
// (as a single 'packed' 32-bit value, which you can get by calling
// strip.Color(red, green, blue) as shown in the loop() function above),
// and a delay time (in milliseconds) between pixels.
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
    delay(wait);                           //  Pause for a moment
  }
}

// Theater-marquee-style chasing lights. Pass in a color (32-bit value,
// a la strip.Color(r,g,b) as mentioned above), and a delay time (in ms)
// between frames.
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
    }
  }
}

// Rainbow cycle along whole strip. Pass delay time (in ms) between
frames.
void rainbow(int wait) {
  // Hue of first pixel runs 5 complete loops through the color wheel.
  // Color wheel has a range of 65536 but it's OK if we roll over, so
  // just count from 0 to 5*65536. Adding 256 to firstPixelHue each time
  // means we'll make 5*65536/256 = 1280 passes through this loop:
  for(long firstPixelHue = 0; firstPixelHue < 5*65536; firstPixelHue +=
256) {
```

```
    // strip.rainbow() can take a single argument (first pixel hue) or
    // optionally a few extras: number of rainbow repetitions (default
1),
    // saturation and value (brightness) (both 0-255, similar to the
    // ColorHSV() function, default 255), and a true/false flag for
whether
    // to apply gamma correction to provide 'truer' colors (default
true).
    strip.rainbow(firstPixelHue);
    // Above line is equivalent to:
    // strip.rainbow(firstPixelHue, 1, 255, 255, true);
    strip.show(); // Update strip with new contents
    delay(wait);  // Pause for a moment
  }
}

// Rainbow-enhanced theater marquee. Pass delay time (in ms) between
frames.
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)
      // 'c' counts up from 'b' to end of strip in increments of 3...
      for(int c=b; c<strip.numPixels(); c += 3) {
        // hue of pixel 'c' is offset by an amount to make one full
        // revolution of the color wheel (range 65536) along the length
        // of the strip (strip.numPixels() steps):
        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
    }
  }
}
```