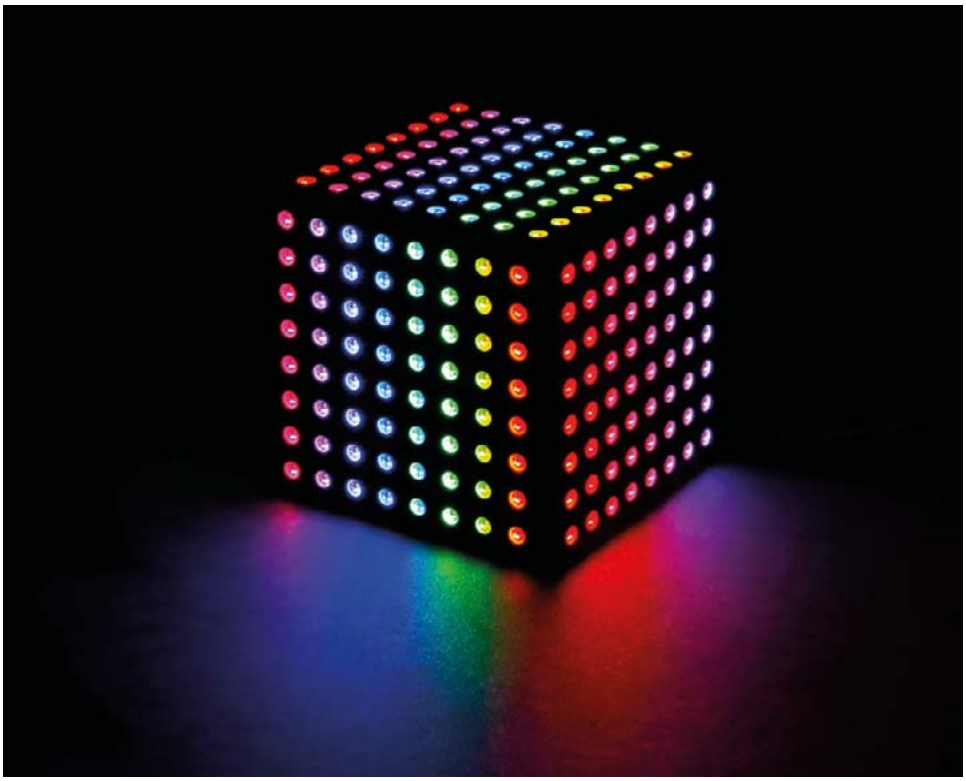
BEDIENUNGSANLEITUNG

64-LED RGB-MATRIX



Inhaltsverzeichnis

Los geht’s!

Die 64-LED RGB-Matrix vorbereiten

Bevor wir beschreiben wie Sie die Software schreiben müssen und wie Sie etwas in einem Panel (mehrere Panels) anzeigen können, möchten wir erklären, wie Sie ein Panel (mehrere Panels) korrekt vorbereiten.

Datenleitungen

Bevor Sie das Gerät mit dem Netz verbinden, müssen Sie unbedingt wissen, wie die WS2812-LEDs miteinander verbunden sind. Jede LED funktioniert wie ein Shieberegister und verschiebt Display-Daten vom Eingang zum Ausgang und danach zur nächsten LED. Dies ist die Art und Weise auf der die Daten durch ein Panel von einer LED auf eine andere verbreitet werden. Im VM207 sind die LEDs in Reihen verbunden und jede Reihe hat eine Rücklaufzeile damit der Beginn der nächsten Reihe mit dem Ende der vorigen Reihe verbunden werden kann. Dank dieser Funktion zickzacken die Reihen nicht und können, mehrere Panels miteinander verbunden, werden.

Studieren Sie folgende Abbildung sorgfältig, um zu begreifen, wie diese LEDs verdrahtet sind.

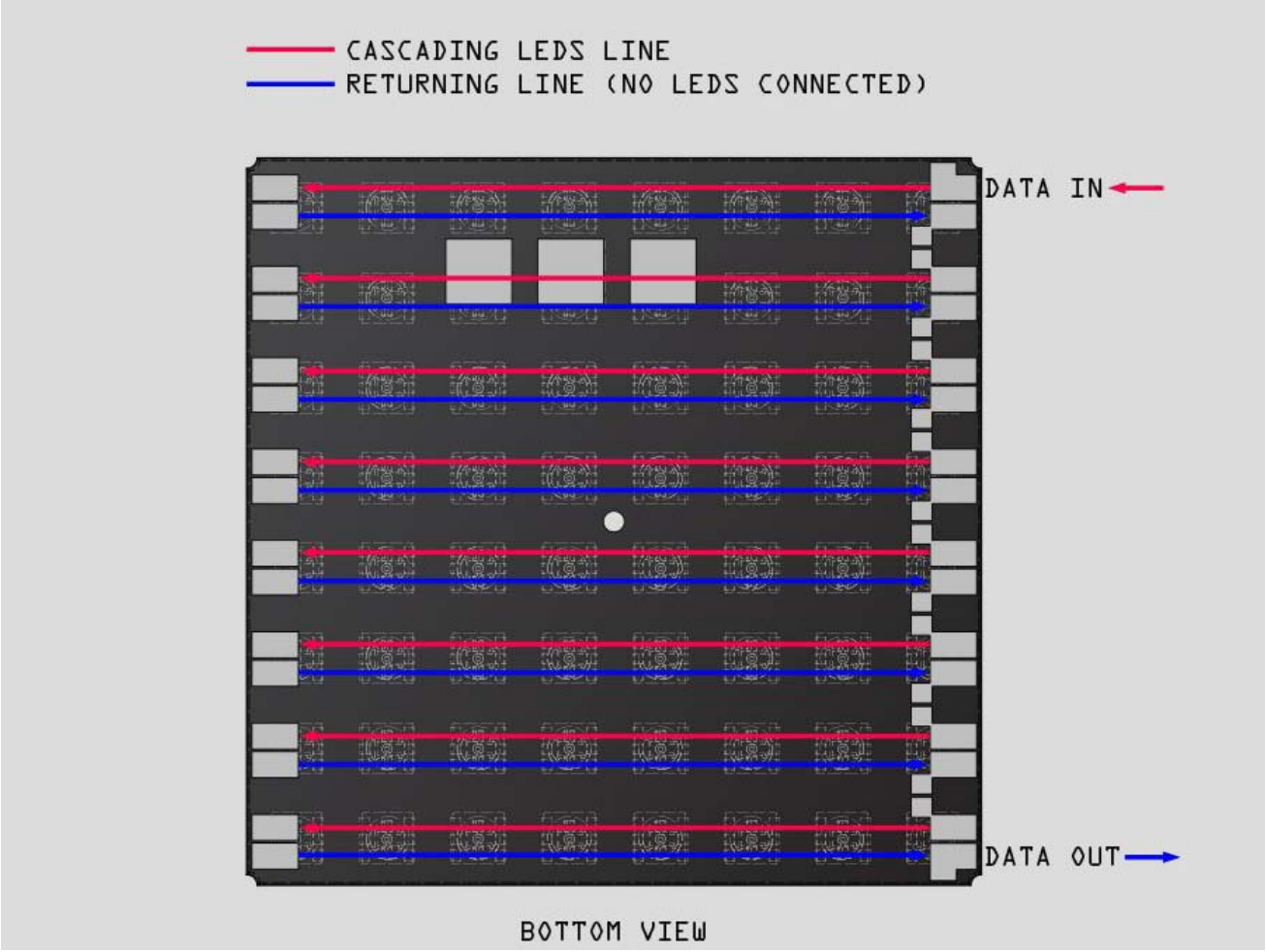


fig. 1

Das Panel wird nicht sofort funktionieren weil die erste Reihe nicht mit der folgenden Reihe verbunden ist. Die Display-Daten werden einfach am Ende der ersten LED-Reihe stoppen und nicht weiter gehen. Um das Panel als 8 x 8 Display zu verwenden und, um die Display-Daten durch jede LED gehen zu lassen, verbinden Sie die Pads wie auf der Abbildung gezeigt (gelbe Lötanschlüsse). Verwenden Sie nur ein bisschen Lötzinn für diese Pads.

Befolgen Sie nun das Display Datapad, dann sehen Sie, dass alle Reihen miteinander verbunden sind und tatsächlich eine lange Kette von WS2812-LEDs darstellen.

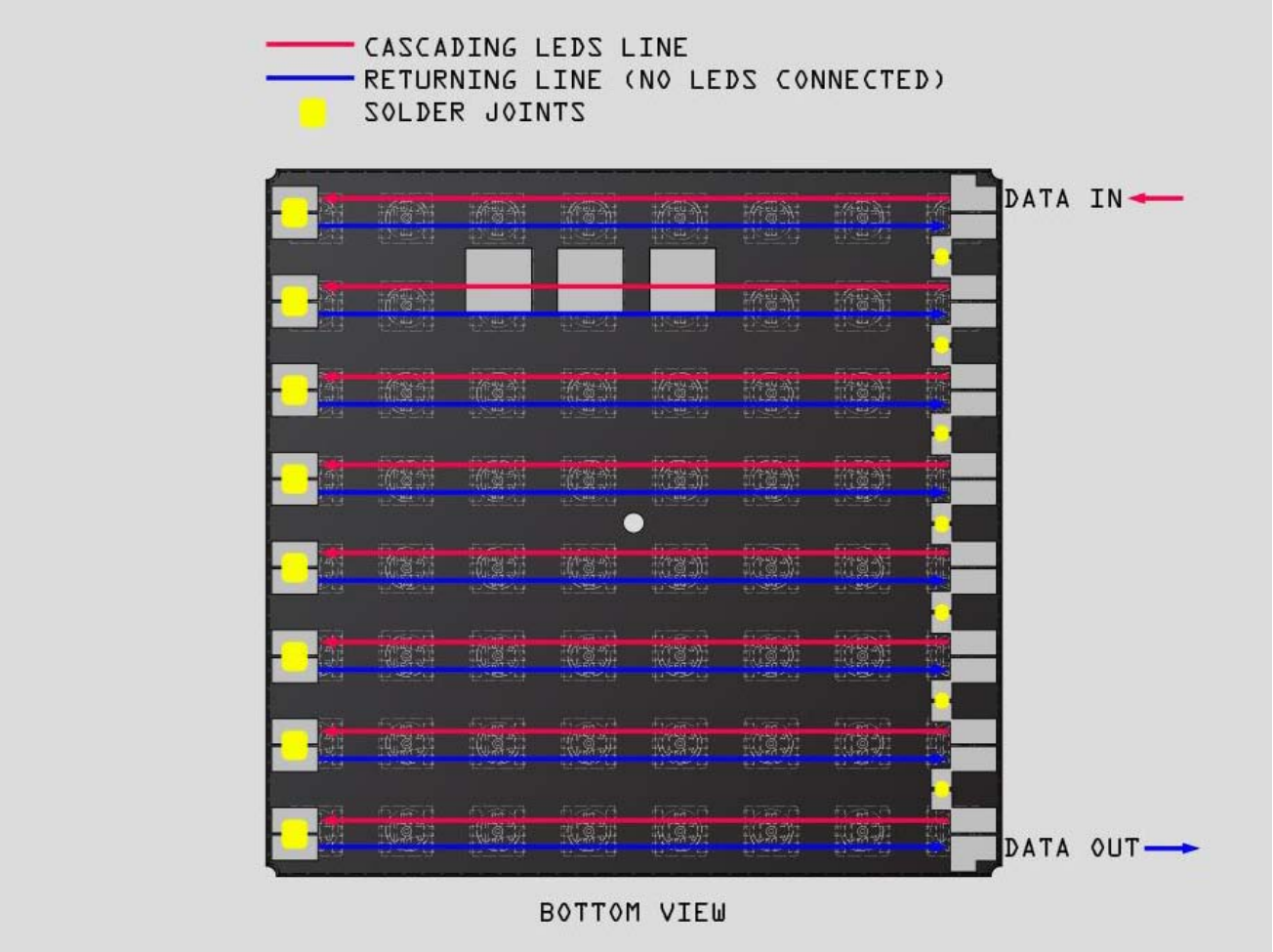


fig. 2

Folgende Abbildung zeigt an, wie Sie zwei (oder mehrere) Panels miteinander verbinden können, indem Sie die Lötpads auf den Seiten mit ein bisschen Blankdraht verbinden.

Eine Reihe besteht nun aus 16 LEDs statt 8 LEDs.

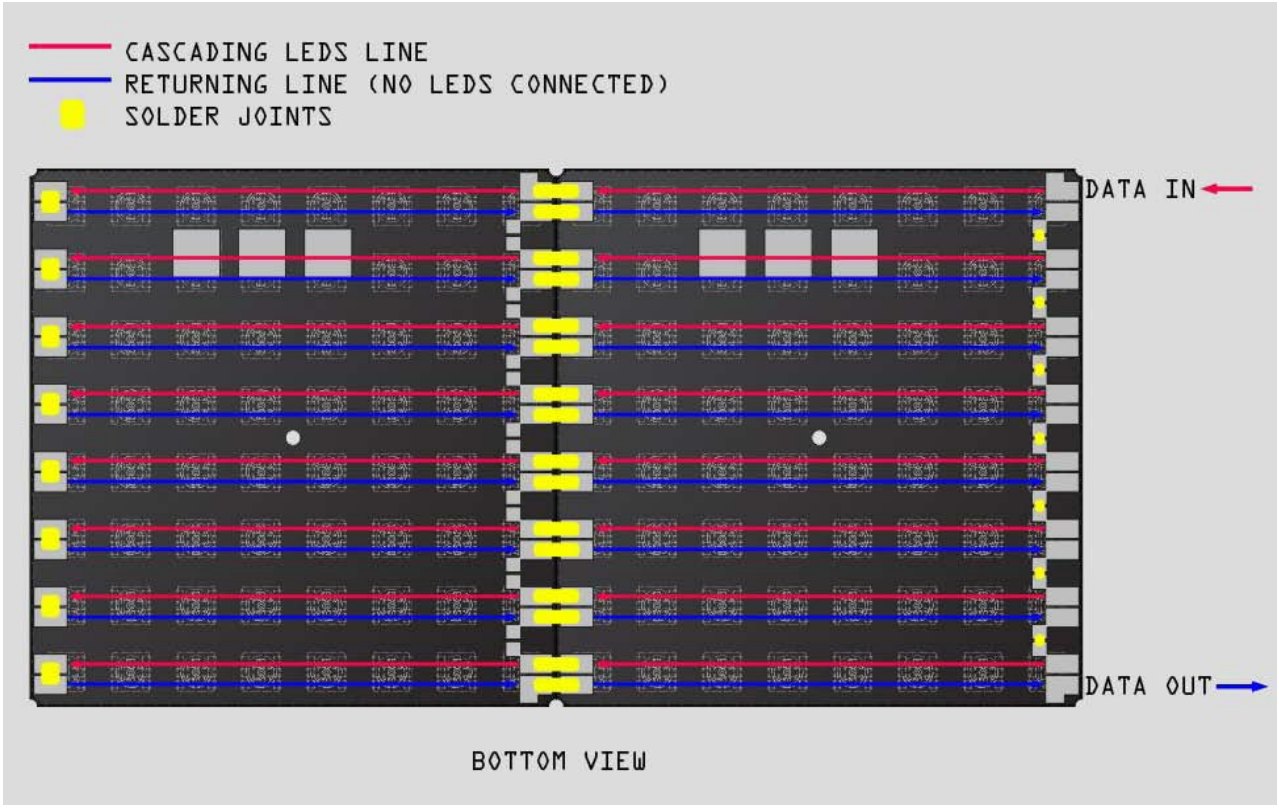


fig. 3

Sie können die Panels auch vertikal verbinden, indem Sie den DATA OUT der vorigen Reihe von Panels an den DATA IN einer neuen Reihe von Panels anschließen. Tun Sie dies vorsichtig weil größere Montagen fragil sein können.

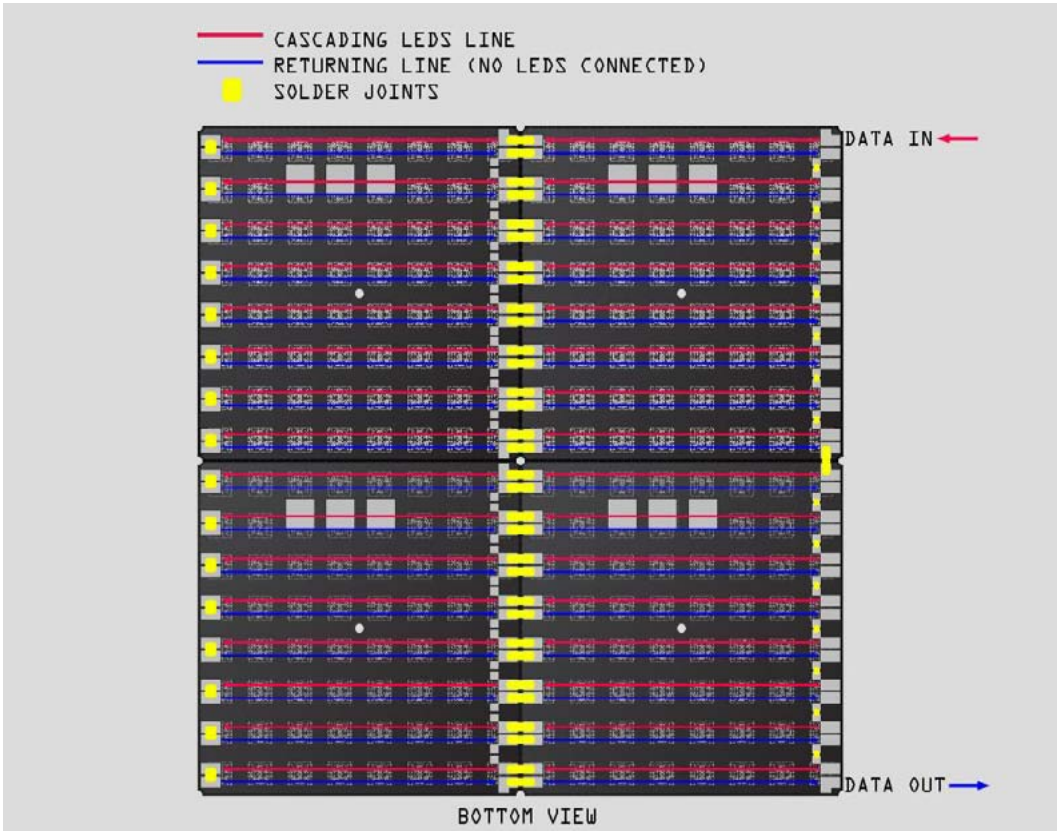


fig. 4

WICHTIGE HINWEISE

- Halten Sie die Datenverbindungen möglichst kurz.
- Teilen Sie die Datenleitung nicht weil dies nicht funktionieren wird. (Eine LED kann keine Daten an zwei oder mehr LEDs senden).
- Die LEDs funktionieren normalerweise mit einem TTL-Pegel (5 V) Datensignal aber funktionieren auch mit einem 3.3 V Mikrokontroller-Ausgang. 5 V ist aber besser….
- Installieren Sie einen Widerstand (470 Ohm) zwischen dem Pin vom Datenausgang des Controllers und dem Eingang des ersten Panels. (Sie brauchen das nicht immer zu tun, aber dies hilft bei Störungen auf den Datenleitungen).

Stromversorgung

Es ist ein bisschen einfacher, die VM207 mit Strom zu versorgen. Es gibt 3 Kontakte: **DV+**, **LV+** y **GND** (**Data Voltage +**, **LED Voltage +** y **Ground**. Die WS2812-LEDs, die auf den Panels montiert werden, enthalten 6 Pins und haben separate Pins für die 5 V für den LED-Die und die 5 V für den CI-Die. Der Pad DV+ ist mit allen IC-Die-Pins verbunden, die Pin LD+ mit allen LED-Die-Pins. In den meisten Fällen können Sie diese beiden einfach miteinander verbinden und wird alles funktionieren. Besteht Ihr Projekt aus mehreren LED-Panels und/oder brauchen Sie eine große Lichtintensität, dann raten wir, die Stromversorgung der LED-Panels und die Ansteuerung getrennt durchzuführen. Die Masse beider Stromversorgungen muss gemeinschaftlich sein.

Siehe folgende Abbildung für die einfachste Art und Weise, um das Panel mit Strom zu versorgen.

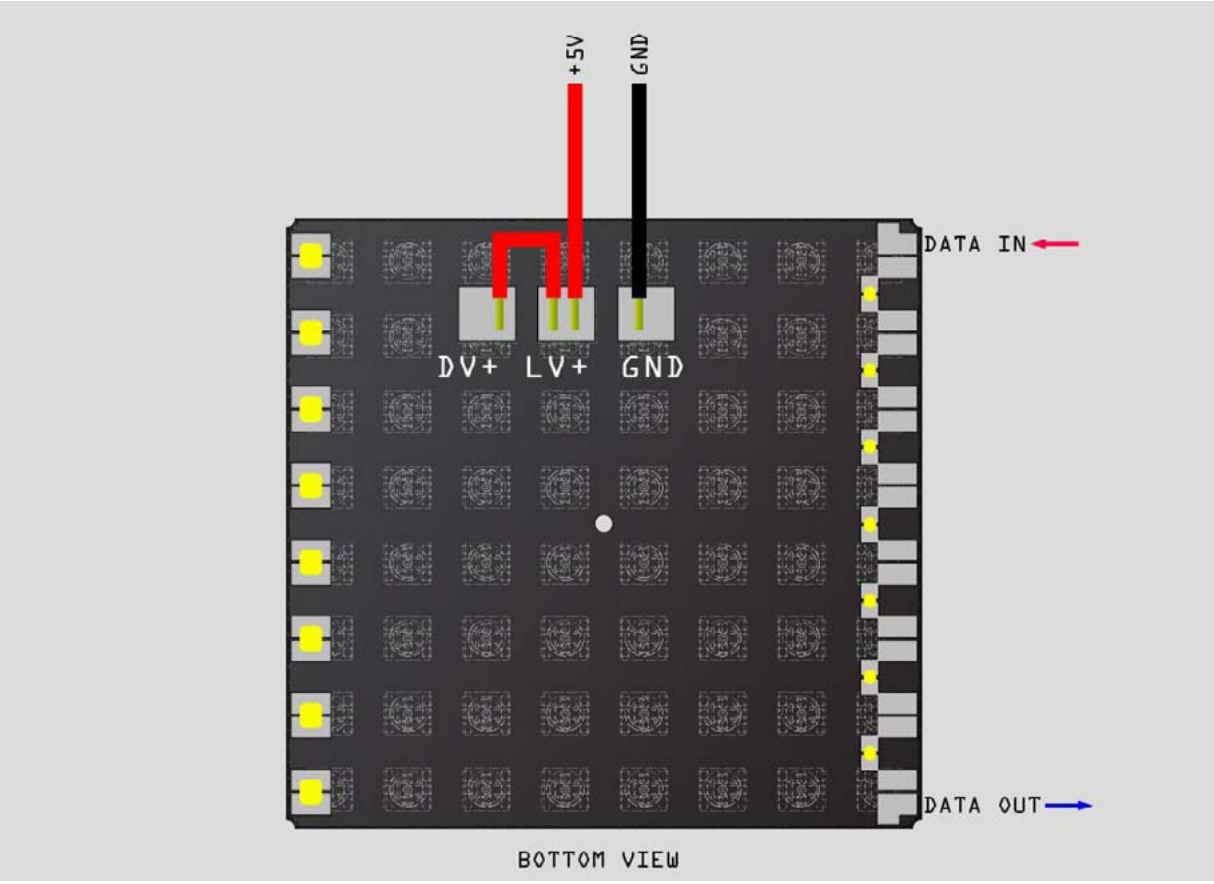


fig. 5

Haben Sie mehrere Panels, so installieren Sie diese in Parallel. **Benutzen Sie Verdrahtung mit einer angepassten Dicke und verwenden Sie Sternverdrahtung!**

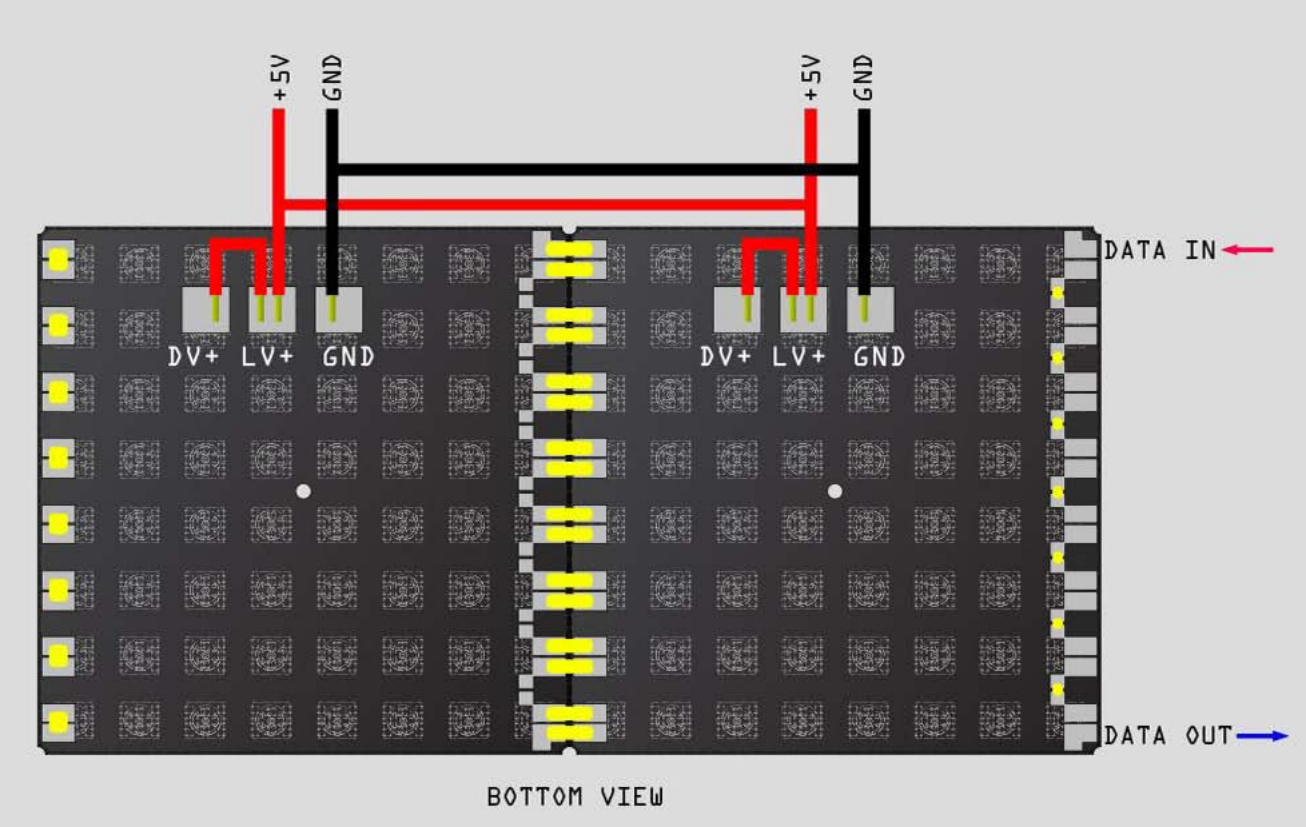


fig. 6

WICHTIGE HINWEISE

- Verwenden Sie immer einen 1000 µf Kondensator (mitgeliefert) in Parallel mit der Stromversorgung.
- Schalten Sie die Stromversorgung nicht ein und aus wenn die Panels angeschlossen sind weil die Spannungsspitzen die LEDs beschädigen können.
- Schließen Sie immer zuerst die Masse an.
- Benutzen Sie ein geeignetes Kabel, um die Panels mit Strom zu versorgen weil diese eine hohe Stromstärke erzeugen können (3.5 A pro Panel bei máx. Helligkeit). Durch die große Stromaufnahme kann ein Spannungsverlust die Lichtintensität beeinflussen.
- Starren Sie niemals die Panels an wenn die LEDs auf max. Helligkeit brennen. Dies könnte desorientierend sein.
- Es wird viel Wärme erzeugt wenn Sie viele Panels auf max. Helligkeit benutzen. Ist dies der Fall, benutzen Sie dann einen Ventilator, ein Kühlkörper usw.) um die Lebendauer der Panels zu erhöhen

Die 64-LED RGB-Matrix ansteuern

Der Mikrokontroller

Sie können viele Mikrokontroller oder Mikrokontroller-Plattformen verwenden, um das Panel anzusteuern. Um es Ihnen leicht zu machen, werden wir erklären, wie Sie das Panel mit einer Mikrokontroller-Plattform, kompatibel mit Arduino IDE (p.ej Uno, Mega, Teensy, etc.) benutzen müssen.

Möchten Sie keine dieser Plattformen verwenden, müssen Sie lernen, einen Datenstrom zu kreieren, um eine Serie WS2812 LEDs anzusteuern. Hier finden Sie die benötigte Information: WS2812 DATASHEET.

Ein Panel mit einem Arduino™ Uno

Verwenden Sie folgende Abbildung, um ein Panel mit einem Arduino Uno zu verbinden. Im Moment wird Pin 6 als Datenausgang verwenden aber dies können Sie später im Code ändern. Verbinden Sie das Arduino und die VM207 mit dem Netz und verbinden Sie die Massen miteinander.

> ⚠ **ACHTUNG**
>
> Verwenden Sie niemals die 5 V vom Arduino, um ein oder mehrere Panels mit Strom zu versehen. Diese Panels benutzen zu viel Strom für den Regulator auf dem Arduino. Tun Sie dies, dann können Sie das Arduino-Board brechen.

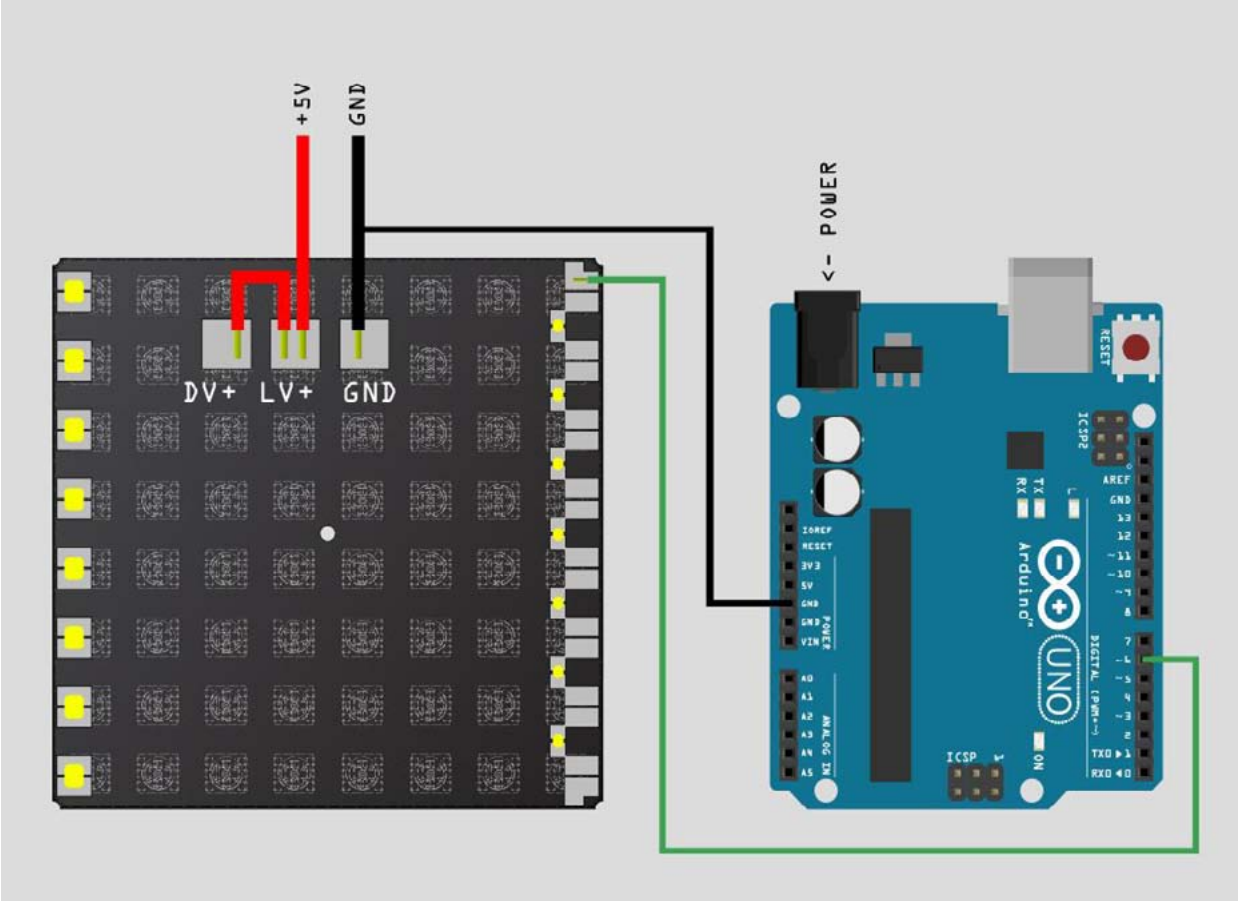


fig. 7

Die Adafruit™ neopixel Bibliothek

Wir werden zuerst die ADAFRUIT NEOPIXEL Bibliothek erklären. Diese Bibliothek wurde von Adafruit geschrieben und kann eine Menge WS2812-LEDs separat ansteuern. Diese Bibliothek eignet sich nicht zum Schreiben von Text oder zum Zeichnen von Figuren aber ist ideal wenn Sie jede LED separat ansteuern möchten oder selber etwas kreieren möchten.

Laden Sie die ADAFRUIT NEOPIXEL Bibliothek herunter: https://github.com/adafruit/Adafruit_NeoPixel (drücken Sie **Download ZIP**)

Nun können Sie diese Bibliothek in die Arduino-Installation installieren (installieren Sie den heruntergeladenen Ordner im Bibliotheksordner der Arduino-Installation) und starten Sie danach die Arduino-Software (Installieren Sie auch die FRUIT NEOPIXEL Bibliothek).

Gehen Sie zu: **File > Examples > Adafruit Neopixel > simple**, dann öffnet Arduino diese Skizze.

```
// NeoPixel Ring simple sketch (c) 2013 Shae Erisson
// released under the GPLv3 license to match the rest of the AdaFruit NeoPixel library

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

// Which pin on the Arduino is connected to the NeoPixels?
// On a Trinket or Gemma we suggest changing this to 1
#define PIN            6

// How many NeoPixels are attached to the Arduino?
#define NUMPIXELS      16

// When we setup the NeoPixel library, we tell it how many pixels, and which pin to use to send signals.
// Note that for older NeoPixel strips you might need to change the third parameter--see the strandtest
// example for more information on possible values.
Adafruit_NeoPixel pixels = Adafruit_NeoPixel(NUMPIXELS, PIN, NEO_GRB + NEO_KHZ800);
```

Was ist hier los?

Im ersten Teil geht's nur um das Einfügen der Adafruit-Bibliothek.

Danach sagt #define PIN 6 dem Programm, dass der PIN 6 gleicht und, dass der Datenstrom aus Pin 6 vom Arduino kommt. Möchten Sie dies nach 13 wechseln, dann ändern Sie diese Zeile des Codes in: **#define PIN 13**

Danach kommt die #define NUMPIXELS 16 Zeile, die dem Programm sagt wie viele LEDs angesteuert werden. Um 1 Panel anzusteuern muss die Zeile so aussehen: #define NUMPIXELS 64. Haben Sie 2 Panels, dann sieht es so aus: 128. Bei 3 Panels = 192 usw.

Danach erscheint folgende Zeile: **Adafruit_NeoPixel pixels = Adafruit_NeoPixel(NUMPIXELS, PIN, NEO_GRB + NEO_KHZ800);**

Diese Zeile gibt unseren LEDs einen Namen. Hier ist das "pixels". Sie können dies einfach wechseln nach "panel" oder etwas anderes. Vergessen Sie aber nicht alle anderen "pixels" zu ändern.

Danach sagen wir dem Programm woraus "pixels" (unser Panel in diesem Fall) besteht. **Adafruit_ NeoPixel(NUMPIXELS, PIN, NEO_GRB + NEO_KHZ800);**

- NUMPIXELS = Dieser Wert wurde schon früher im Programm definiert. Dies ist die Anzahl Pixel die Sie angesteuert werden müssen.
- PIN = Dieser Wert wurde schon früher im Programm definiert. • Dies ist der Display Datenstrom Pin .
- NEO_GRB = Erhalten Sie dies für die Standard WS2812-LEDs.
- NEO_KHZ800 = Lassen Sie dies für die Standard WS2812-LEDs.

Danach erscheint der folgende Code, der nur eine Variable ist. Diese Variable speichert einen Wert, der als eine Verzögerung in der Hauptfunktion verwendet wird. Ändern Sie diesen Wert, dann ändert sich die Geschwindigkeit, mit der die For-Loop-Funktion in der Loop-Funktion ausgeführt wird.

```
int delayval = 500; // delay for half a second

Below that piece of code there is the setup function that looks like this:

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();
}
```

Der wichtigste Teil des Codes ist die Zeile **strip.begin**();. Hierdurch werden die LEDs gestartet. Vergessen Sie diesen Schritt nicht in Ihrem eigenen Code.

Danach kommt die Loop-Funktion:

```
void loop() {

  // For a set of NeoPixels the first NeoPixel is 0, second is 1, all the way up to the count of pix-
els minus one.

  for(int i=0;i<NUMPIXELS;i++){

    // pixels.Color takes RGB values, from 0,0,0 up to 255,255,255
    pixels.setPixelColor(i, pixels.Color(0,150,0)); // Moderately bright green color.

    pixels.show(); // This sends the updated pixel color to the hardware.

    delay(delayval); // Delay for a period of time (in milliseconds).

  }
}
```

Hier geschieht etwas besonders. Die Loop-Funktion wiederholt sich selber jedes Mal und innerhalb dieser Loop-Funktion gibt es eine For-Loop-Funktion, die diese Zeilen jede halbe Sekunde ausführen wird:

```
    pixels.setPixelColor(i, pixels.Color(0,0,255)); // Bright blue color.

    pixels.show(); // This sends the updated pixel color to the hardware.

    delay(delayval); // Delay for a period of time (in milliseconds).
```

Die erste Zeile gibt dem Pixel, wo Wert "i" gespeichert ist, eine grüne Farbe. Diese Information wird aber nur aktualisiert wenn die zweite Zeile "pixels.show" aufgerufen wird. Diese zwei Zeilen werden alle 500 ms aufgerufen und "i" erhöht sich immer um 1. Für mehr Informationen über das Funktionieren der For-Loops: https://www.arduino.cc/en/reference/for Normalerweise werden alle LEDs des Panels langsam grün wenn Sie diesen Code für Arduino verwenden.

Versuchen Sie dies, indem Sie diesen Code erstellen und auf dem Arduino hochladen und verbinden Sie alles. Das Panel leuchtet langsam (alle 500 ms) eine neue grüne LED. Leuchten alle grünen LEDs, dann bleibt das Panel grün. Um dies zu ändern, können Sie versuchen die Loop-Funktion und den Verzögerungswert (siehe oben) zu ändern (Klicken Sie zwei Mal um auszuwählen und zu kopieren):

```
void loop() {

  // For a set of NeoPixels the first NeoPixel is 0, second is 1, all the way up to the count of pix-
els minus one.

  for(int i=0;i<NUMPIXELS;i++){
    // pixels.Color takes RGB values, from 0,0,0 up to 255,255,255
    pixels.setPixelColor(i, pixels.Color(0,0,255)); // Brigh blue color.
```

```
    pixels.show(); // This sends the updated pixel color to the hardware.

    delay(delayval); // Delay for a period of time (in milliseconds).
  }
  for(int i=0;i<NUMPIXELS;i++){
    pixels.setPixelColor(i, pixels.Color(0,0,0)); // No color (dark).
  }
  pixels.show(); //Updating the panel to show nothing.
}
```

Dieser Code löscht alle Werte nach der ersten For-Loop-Funktion, durch eine zweite For-Loop-Funktion zu benutzen. Wir haben auch die Farbe gewechselt nach Blau (völlig gesättigt). Dies ist eine sehr helle Farbe. Um die Helligkeit des ganzen Panels zu ändern, können Sie Folgendes verwenden: **pixels.setBrightness(20);** Diese Funktion akzeptiert einen Wert zwischen 20 und 255 (0 = min. Helligkeit, 255 = max. Helligkeit) . Vergewissern Sie sich davon, wie das Display beeinflusst wird wenn Sie die Loop-Funktion ändern (Kicken Sie zwei Mal, um auszuwählen und zu kopieren):

```
void loop() {
  pixels.setBrightness(20); // Setting the brightness really low.
  // For a set of NeoPixels the first NeoPixel is 0, second is 1, all the way up to the count of pix-
els minus one.
  for(int i=0;i<NUMPIXELS;i++){
    // pixels.Color takes RGB values, from 0,0,0 up to 255,255,255
    pixels.setPixelColor(i, pixels.Color(0,0,255)); // Bright blue color.

    pixels.show(); // This sends the updated pixel color to the hardware.

    delay(delayval); // Delay for a period of time (in milliseconds).
  }
  for(int i=0;i<NUMPIXELS;i++){
    pixels.setPixelColor(i, pixels.Color(0,0,0)); // No color (dark).
  }
  pixels.show(); // Updating the panel to show nothing.
}
```

Unten sehen Sie wie der vollständige Code im Moment aussieht. Bevor Sie weitergehen, experimentieren Sie mit alles was Sie in diesem Kapitel gelernt haben. Klicken Sie zwei Mal, um auszuwählen und zu kopieren.

```
// NeoPixel Ring simple sketch (c) 2013 Shae Erisson
// released under the GPLv3 license to match the rest of the AdaFruit NeoPixel library

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

```
// Which pin on the Arduino is connected to the NeoPixels?
// On a Trinket or Gemma we suggest changing this to 1
#define PIN            6

// How many NeoPixels are attached to the Arduino?
#define NUMPIXELS      64

// When we setup the NeoPixel library, we tell it how many pixels, and which pin to use to send signals.
// Note that for older NeoPixel strips you might need to change the third parameter--see the strandtest
// example for more information on possible values.
Adafruit_NeoPixel pixels = Adafruit_NeoPixel(NUMPIXELS, PIN, NEO_GRB + NEO_KHZ800);

int delayval = 50; // delay for half a second

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

 pixels.begin(); // This initializes the NeoPixel library.
}




void loop() {
 pixels.setBrightness(20); // Setting the brightness really low.
 // For a set of NeoPixels the first NeoPixel is 0, second is 1, all the way up to the count of pixels minus one.
 for(int i=0;i<NUMPIXELS;i++){
   // pixels.Color takes RGB values, from 0,0,0 up to 255,255,255
   pixels.setPixelColor(i, pixels.Color(0,0,255)); // Bright blue color.

   pixels.show(); // This sends the updated pixel colour to the hardware.

   delay(delayval); // Delay for a period of time (in milliseconds).
 }
 for(int i=0;i<NUMPIXELS;i++){
   pixels.setPixelColor(i, pixels.Color(0,0,0)); // Moderately bright green color.
 }
 pixels.show(); // Updating the panel to show nothing.
}
```

Die Adafruit™ neomatrix

Möchten Sie einen Text anzeigen oder Figuren zeichnen, verwenden Sie dann die ADAFRUIT NEOMATRIX Bibliothek. Um diese Bibliothek funktionieren zu lassen, brauchen Sie auch die ADAFRUIT GFX Bibliotheken. Diese Bibliotheken brauchen Sie, um die Figuren, die Buchstaben und die Farben zu kreieren. Die neomatrix dient, um alle Daten zu den Panels zu senden.

Laden Sie die ADAFRUIT NEOMATRIX-Bibliothek hier herunter: https://github.com/adafruit/Adafruit_NeoMatrix (drücken Sie **Download ZIP**)

Laden Sie die ADAFRUIT GFX-Bibliothek hier herunter: https://github.com/adafruit/Adafruit-GFX-Library drücken Sie **Download ZIP**)

Nun können Sie diese Bibliothek in die Arduino-Installation installieren (installieren Sie den heruntergeladenen Ordner im Bibliotheksordner der Arduino-Installation) und starten Sie danach die Arduino-Software (Installieren Sie auch die FRUIT NEOPIXEL Bibliothek).

Gehen Sie zu: **File > Examples > Adafruit Neopixel > simple**, dann öffnet Arduino diese Skizze.

Der obere Teil der geöffnete Datei sieht so aus:

```
#include <Adafruit_GFX.h>
#include <Adafruit_NeoMatrix.h>
#include <Adafruit_NeoPixel.h>
#ifndef PSTR
 #define PSTR // Make Arduino Due happy
#endif

#define PIN 6

// MATRIX DECLARATION:
// Parameter 1 = width of NeoPixel matrix
// Parameter 2 = height of matrix
// Parameter 3 = pin number (most are valid)
// Parameter 4 = matrix layout flags, add together as needed:
//   NEO_MATRIX_TOP, NEO_MATRIX_BOTTOM, NEO_MATRIX_LEFT, NEO_MATRIX_RIGHT:
//     Position of the FIRST LED in the matrix; pick two, e.g.
//     NEO_MATRIX_TOP + NEO_MATRIX_LEFT for the top-left corner.
//   NEO_MATRIX_ROWS, NEO_MATRIX_COLUMNS: LEDs are arranged in horizontal
//     rows or in vertical columns, respectively; pick one or the other.
//   NEO_MATRIX_PROGRESSIVE, NEO_MATRIX_ZIGZAG: all rows/columns proceed
//     in the same order, or alternate lines reverse direction; pick one.
//   See example below for these values in action.
// Parameter 5 = pixel type flags, add together as needed:
//   NEO_KHZ800  800 KHz bitstream (most NeoPixel products w/WS2812 LEDs)
//   NEO_KHZ400  400 KHz (classic •vĬ (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)
```

```
// Example for NeoPixel Shield.  In this application wĕd like to use it
// as a 5x8 tall matrix, with the USB port positioned at the top of the
// Arduino.  When held that way, the first pixel is at the top right, and
// lines are arranged in columns, progressive order.  The shield uses
// 800 KHz (v2) pixels that expect GRB colour data.
Adafruit_NeoMatrix matrix = Adafruit_NeoMatrix(5, 8, PIN,
  NEO_MATRIX_TOP     + NEO_MATRIX_RIGHT +
  NEO_MATRIX_COLUMNS + NEO_MATRIX_PROGRESSIVE,
  NEO_GRB            + NEO_KHZ800);
```

Der erste Teil enthält die 3 Bibliotheken (NEOPIXEL, NEOMATRIX und GFX).

Danach bestimmen wir wieder einen Ausgangspin. Dies ist das gleiche Prinzip als im vorigen Kapitel.

Nun werden wir erklären wie Sie die Bibliothek konfigurieren müssen, damit diese weißt wie Ihr Panel aussieht. Geben Sie dem Panel zuerst einen Namen. In diesem Fall:”matrix”. Danach gibt es 5 Parameter, die Sie berücksichtigen müssen:

- die Breite der Matrix (Anzahl LEDs, die die Breite der Matrix formen (8, 16, 32, ...)
- die Höhe der Matrix (Anzahl LEDs, die die Höhe der Matrix formen (8, 16, 32, ...)
- PIN-Nummer vom DATA OUT-Pin.
- Dieser Parameter beschreibt das Layout vom Panel und besteht aus Fahnen, die Sie miteinander verbinden müssen. Bestimmen Sie zuerst, wo sich die erste LED in der Matrix befindet. Diese LED befindet sich meistens oben links: **NEO_MATRIX_TOP + NEO_MATRIX_LEFT**. Wählen Sie dann aus, ob die LEDs in Reihen oder in Spalten verbinden möchten. Verwenden Sie die Panels horizontal, wählen Sie dann: **NEO_MATRIX_ROWS**. Wählen Sie dann aus, ob die Reihen (oder Spalten) progressiv oder im Zickzack laufen müssen. Die Reihen (oder Spalten) der VM207 bewegen sich immer progressiv. **NEO_MATRIX_PROGRESSIVE**.
- Beim letzten Parameter handelt es sich um die LEDs, die Sie für die VM207 verwenden müssen. Benutzen Sie IMMER: **NEO_GRB + NEO_KHZ800**.

Um 1 Panel anzusteuern, fügen Sie alle Parameter wie folgt zusammen. Zum Beispiel: wechseln Sie den Code des Beispiels und benutzen Sie den folgenden Code weil diese gemacht ist, um ein 5 x 8 Panel anzusteuern):

```
Adafruit_NeoMatrix matrix = Adafruit_NeoMatrix(8, 8, PIN,
  NEO_MATRIX_TOP     + NEO_MATRIX_LEFT +
  NEO_MATRIX_ROWS + NEO_MATRIX_PROGRESSIVE,
  NEO_GRB            + NEO_KHZ800);
```

Möchten Sie ein größeres Display mit den VM207-Panels bauen, dann müssen Sie die Parameter demgemäß anpassen.

Der nächste Code im Programm ist nur eine Matrix mit 3 Farben damit wir durch einige Farben gehen können, um das Beispiel ein bisschen spezieller zu machen:

```
const uint16_t colors[] = {
  matrix.Color(255, 0, 0), matrix.Color(0, 255, 0), matrix.Color(0, 0, 255) };
```

In der Setup-Funktion müssen Sie mit einigen Dingen rechnen, aber der wichtigste Teil ist die Funktion matrix. begin(); weil diese das Panel startet. Beim folgenden Code handelt es sich um den Text, der im Panel angezeigt wird. Wir werden diesen Code nun erstellen und auf dem Arduino Uno hochladen: Brauchen Sie den ganzen Code, dann können Sie ihn hier kopieren:

```
// Adafruit_NeoMatrix example for single NeoPixel Shield.
// Scrolls •Howdỹ across the matrix in a portrait (vertical) orientation.

#include <Adafruit_GFX.h>
#include <Adafruit_NeoMatrix.h>
#include <Adafruit_NeoPixel.h>
#ifndef PSTR
 #define PSTR // Make Arduino Due happy
#endif

#define PIN 6

// MATRIX DECLARATION:
// Parameter 1 = width of NeoPixel matrix
// Parameter 2 = height of matrix
// Parameter 3 = pin number (most are valid)
// Parameter 4 = matrix layout flags, add together as needed:
//   NEO_MATRIX_TOP, NEO_MATRIX_BOTTOM, NEO_MATRIX_LEFT, NEO_MATRIX_RIGHT:
//     Position of the FIRST LED in the matrix; pick two, e.g.
//     NEO_MATRIX_TOP + NEO_MATRIX_LEFT for the top-left corner.
//   NEO_MATRIX_ROWS, NEO_MATRIX_COLUMNS: LEDs are arranged in horizontal
//     rows or in vertical columns, respectively; pick one or the other.
//   NEO_MATRIX_PROGRESSIVE, NEO_MATRIX_ZIGZAG: all rows/columns proceed
//     in the same order, or alternate lines reverse direction; pick one.
//   See example below for these values in action.
// Parameter 5 = pixel type flags, add together as needed:
//   NEO_KHZ800  800 KHz bitstream (most NeoPixel products w/WS2812 LEDs)
//   NEO_KHZ400  400 KHz (classic •vĭ (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)


// Example for NeoPixel Shield.  In this application wĕd like to use it
// as a 5x8 tall matrix, with the USB port positioned at the top of the
// Arduino.  When held that way, the first pixel is at the top right, and
// lines are arranged in columns, progressive order.  The shield uses
// 800 KHz (v2) pixels that expect GRB color data.
```

```
Adafruit_NeoMatrix matrix = Adafruit_NeoMatrix(8, 8, PIN,
  NEO_MATRIX_TOP     + NEO_MATRIX_LEFT +
  NEO_MATRIX_ROWS + NEO_MATRIX_PROGRESSIVE,
  NEO_GRB            + NEO_KHZ800);

const uint16_t colors[] = {
  matrix.Color(255, 0, 0), matrix.Color(0, 255, 0), matrix.Color(0, 0, 255) };

void setup() {
  matrix.begin();
  matrix.setTextWrap(false);
  matrix.setBrightness(40);
  matrix.setTextColor(colors[0]);
}

int x    = matrix.width();
int pass = 0;

void loop() {
  matrix.fillScreen(0);
  matrix.setCursor(x, 0);
  matrix.print(F("Howdy"));
  if(--x < -36) {
    x = matrix.width();
    if(++pass >= 3) pass = 0;
    matrix.setTextColor(colors[pass]);
  }
  matrix.show();
  delay(100);
}
```

Normalerweise erscheint nun das Wort "Howdy" in drei verschiedenen Farben.
Dies ist nur ein Beispiel der vielen Möglichkeiten, die die ADAFRUIT GFX-Bibliothek bietet. Für mehr Informationen, lesen Sie folgende Bedienungsanleitung: https://learn.adafruit.com/downloads/pdf/adafruit-gfx-graphics-library.pdf .

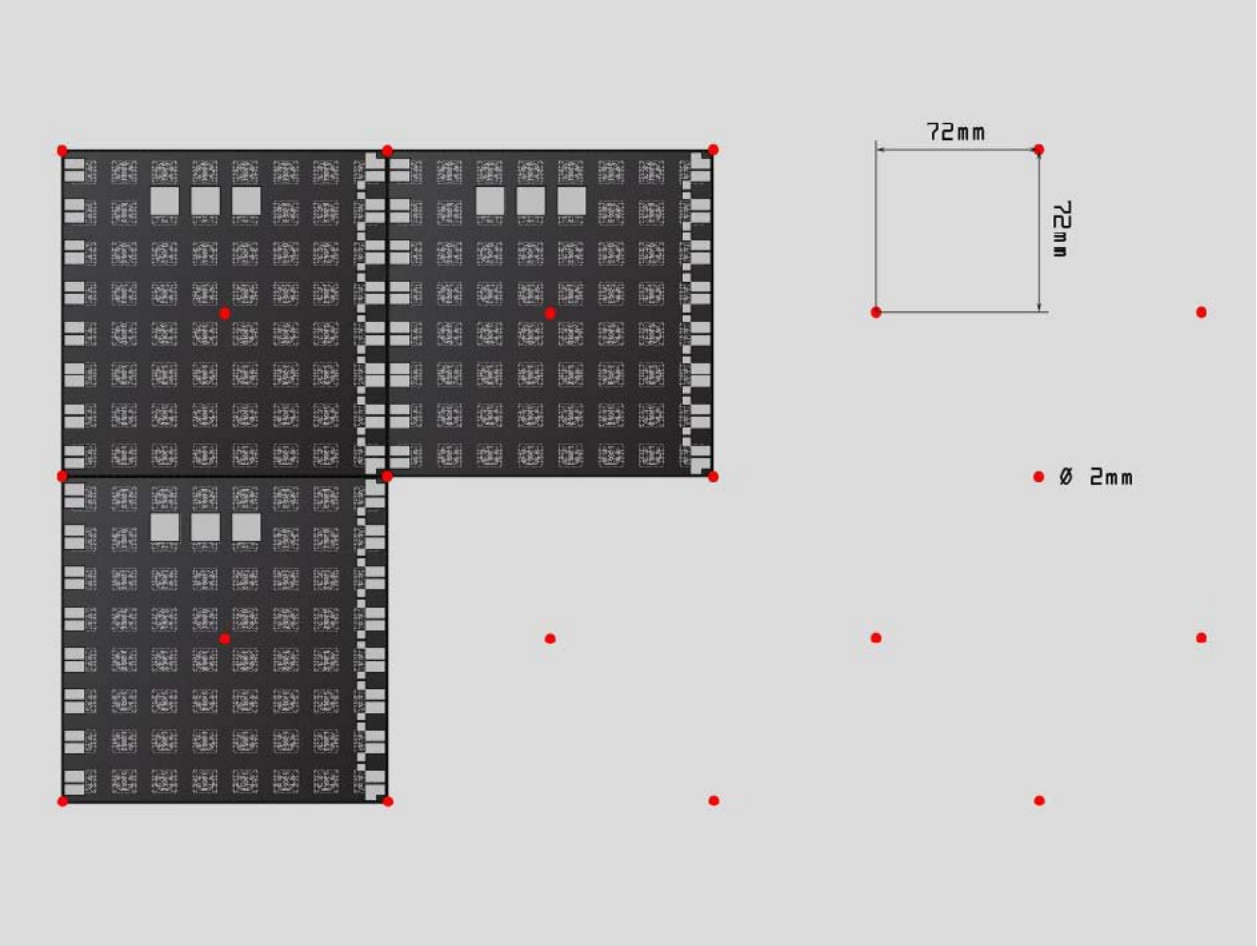
Diese Bedienungsanleitung basiert auf einem echten Display. Die Panels funktionieren gemäß dem gleichen Prinzip wenn Sie die NEOMATRIX-Bibliothek zusammen mit der GFX-Bibliothek verwenden.

¡Entdecken Sie alles was Sie mit diesen Bibliotheken tun können und kreieren Sie coole Displays!

Die 64-LED RGB-Matrix montieren

Die Befestigungslöcher

Die Panels haben Befestigungslöcher, mit denen Sie verschiedene Panels verbinden können. Diese Befestigungslöcher haben ein Durchmesser von 2 mm. Verwenden Sie eine kleine M2-Schraube, um die Panels miteinander zu verbinden. Sie befinden sich in einem Abstand von 72 mm und sind gestaffelt. Siehe Abbildung:

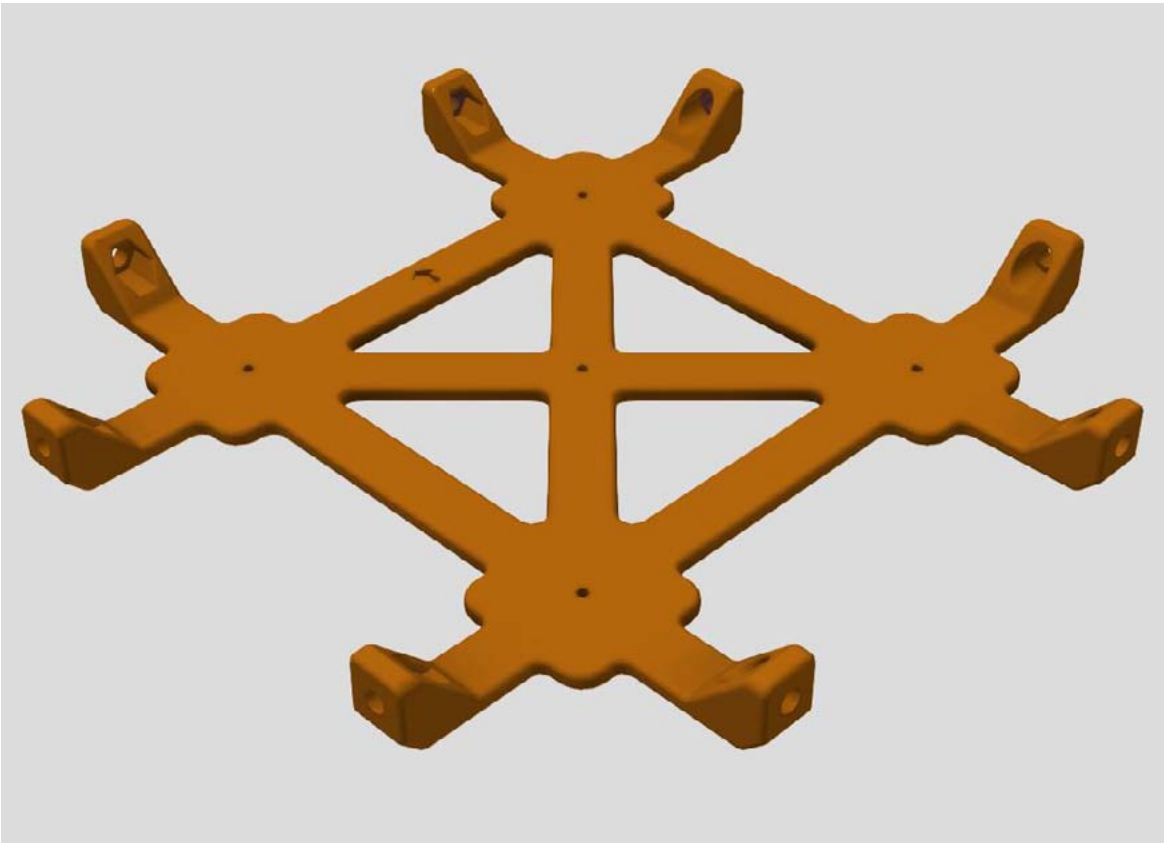


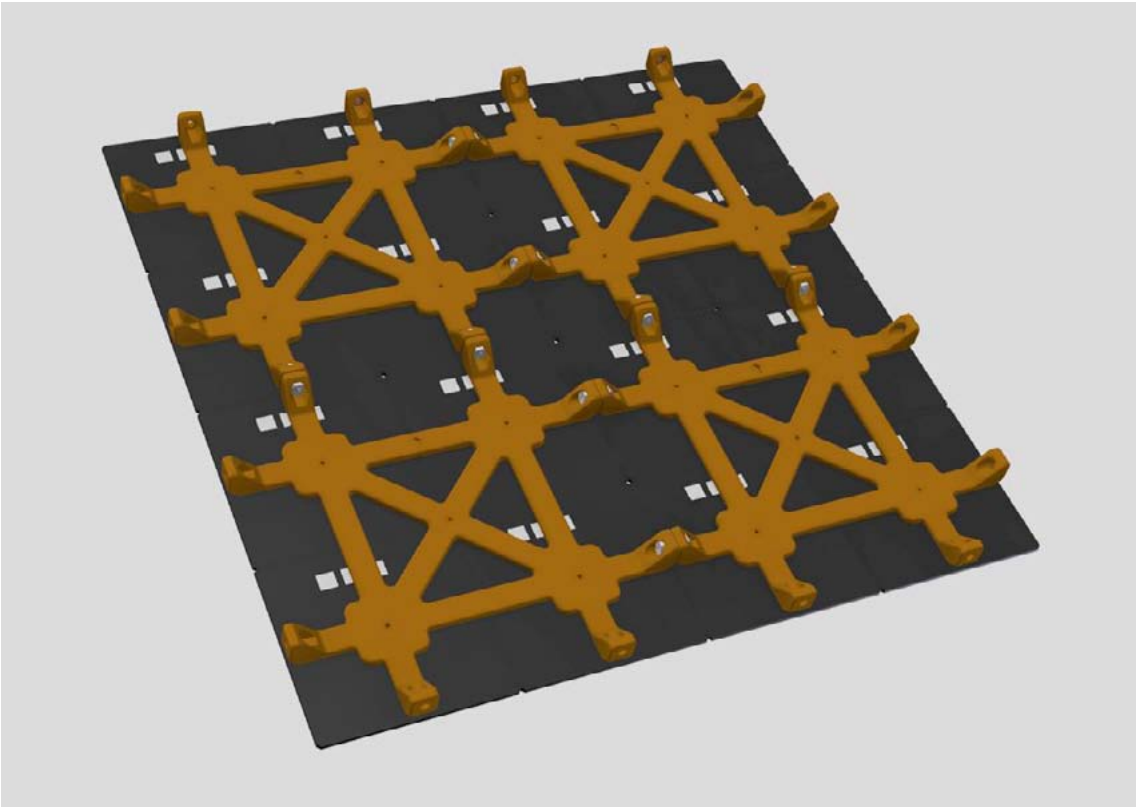
Montagebügel (mit 3D-Drucker gedruckt)

Verfügen Sie über einen 3D-Drucker, dann können Sie folgende Montagebügel herunterladen:

vm207-tile_mount.stl

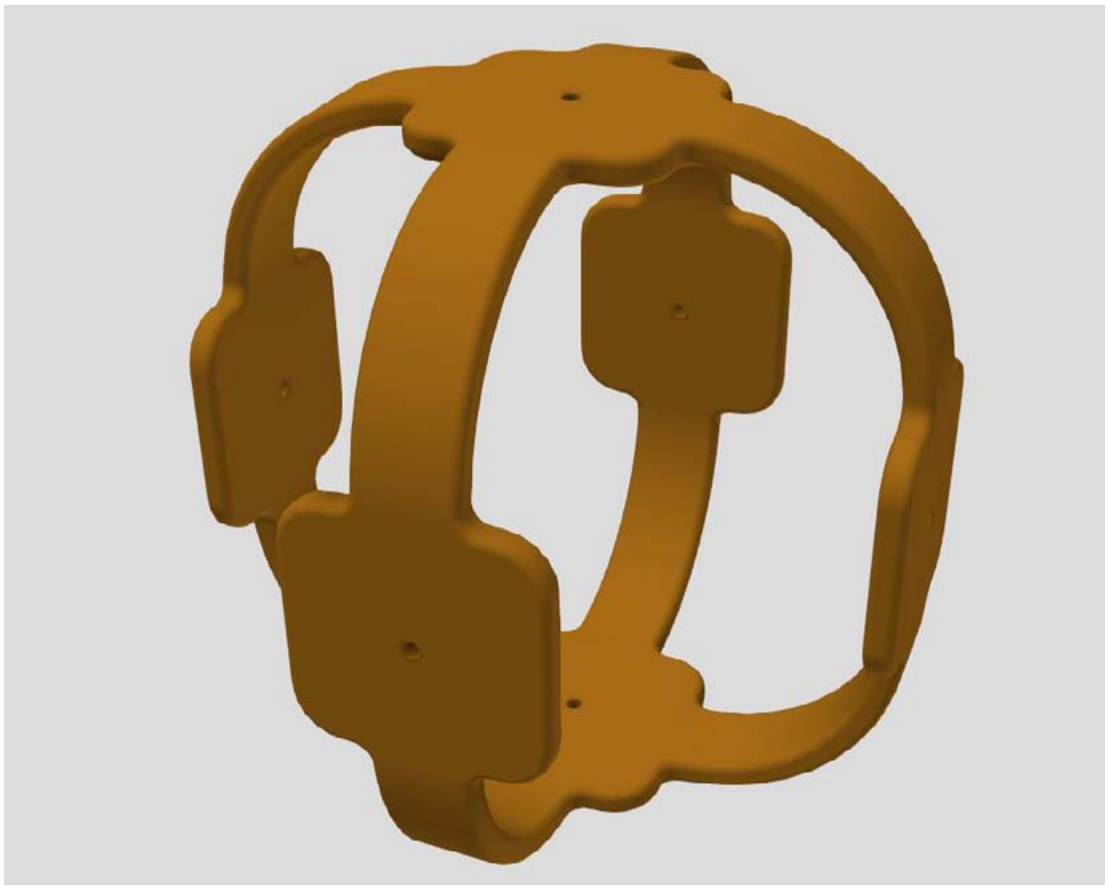
Die verschiedene Montagebügel werden mit 10 M3-Schrauben miteinander verbunden.

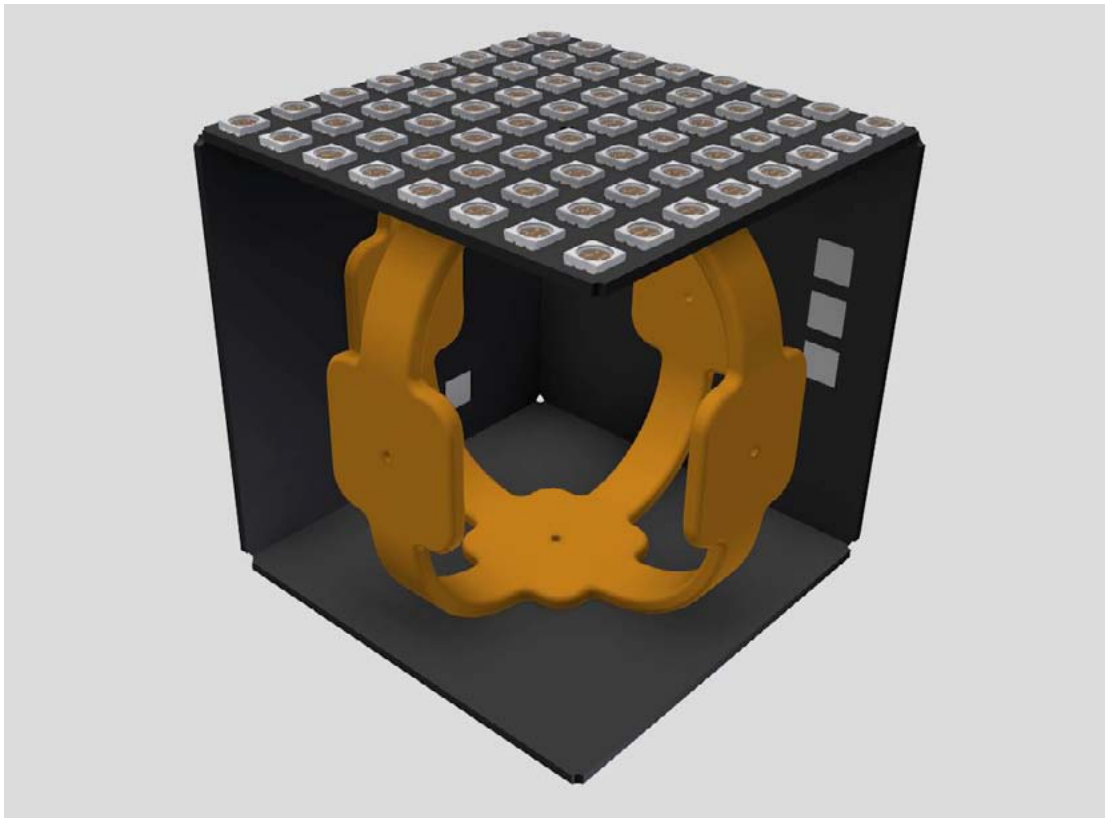




vm207-cube_mount.stl

Benötigt 6 x VM207-Panels. Achten Sie darauf, wie Sie die Panels miteinander verbinden müssen!





ORDERCODE: VM207

REVISION: HVM207'1

VellemanProjects @Velleman_RnD
VELLEMAN nv - Legen Heirweg 33, Gavere (Belgium)
vellemanprojects.com