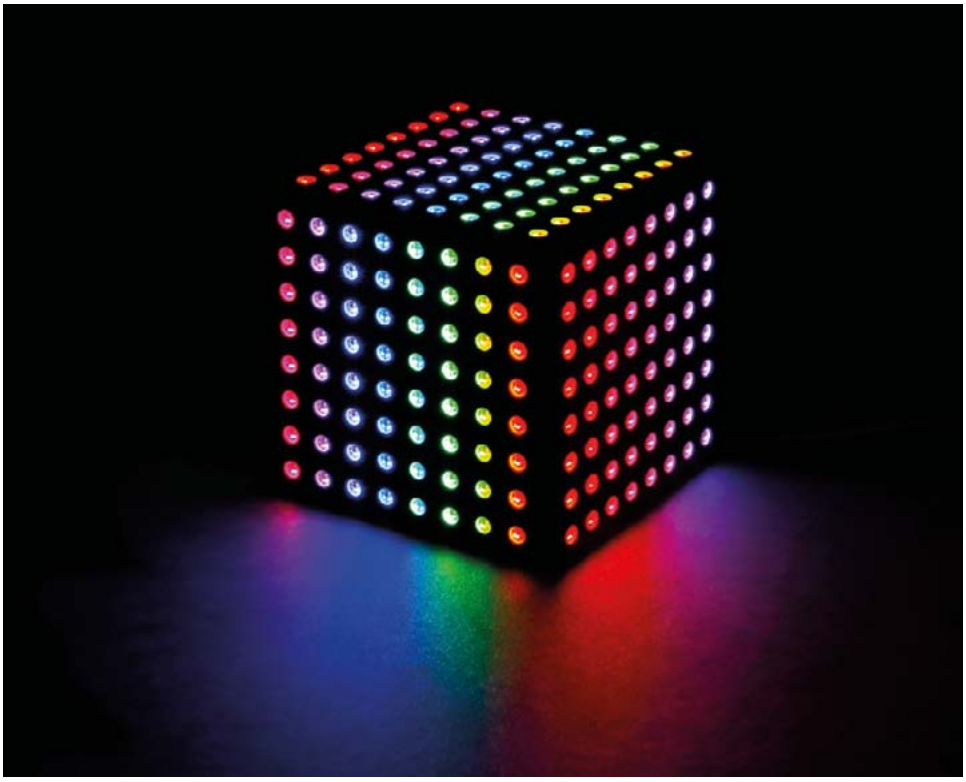
MODE D'EMPLOI

MATRICE à 64 LEDs



Sommaire

Préparer la matrice à LED RVB 3

Lignes de données 3

Alimentation 6

Contrôler la matrice à LED RVB 8

Quel microcontrôleur choisir ? 8

Connecter un panneau à une carte Arduino™ Uno 8

Bibliothèque NeoPixel d'Adafruit™ 9

Bibliothèque NeoMatrix d'Adafruit™ 13

Monter la matrice à LED 17

Trous de montage 17

Supports de montage imprimés en 3D 18

Démarrer!

Préparer la matrice à LED

Avant d'entrer dans les détails du fonctionnement, passons à la préparation correcte du panneau.

Lignes de données

Avant de connecter l'alimentation, il est nécessaire de comprendre comment les LEDs WS8212 sont connectées. Chaque LED fonctionne comme un registre à décalage qui décale les données d'affichage depuis l'entrée vers la sortie, et passe à la LED suivante. C'est ainsi que les données d'affichage se propagent dans un panneau, d'une LED à la LED suivante. Dans la VM207, les LEDs sont connectées en lignes. Le point de départ de chaque ligne est reliée au point de fin de la ligne précédente. L'avantage est que les lignes sont alignées et que vous pouvez facilement connecter plusieurs panneaux.

Observez la figure (ci-dessous) pour voir comment ces LEDs sont connectées.

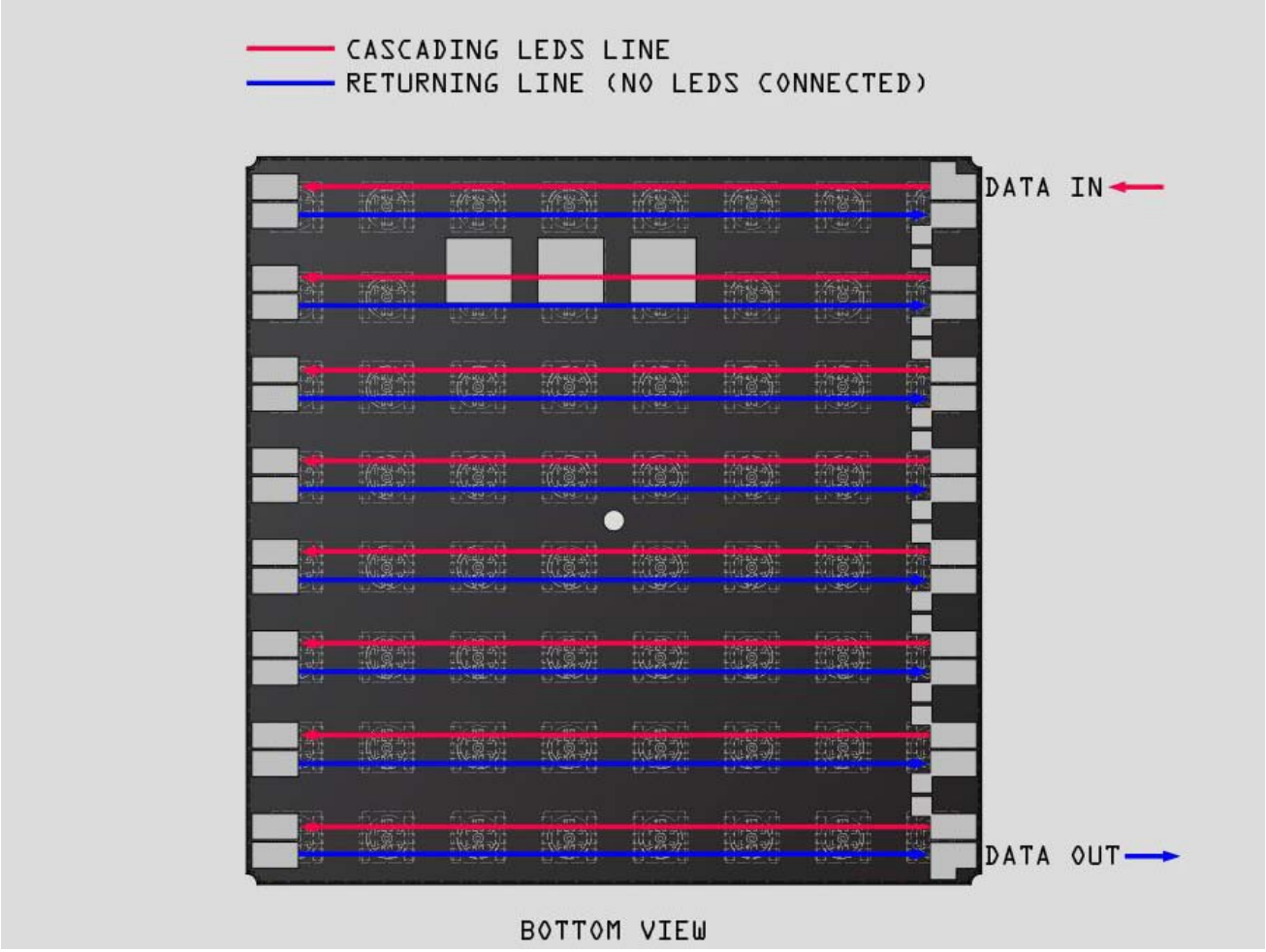


fig. 1

Comme vous le voyez , le panneau ne fonctionne pas immédiatement puisque l'extrémité de la première ligne n'est connectée à rien. Les données d'affichage s'arrêtent à la fin de la première ligne de LEDs et ne sont pas envoyées à travers le panneau. Pour utiliser le panneau comme un afficheur 8 x 8 et transmettre les données d'affichage à chaque LED, connectez les pattes comme illustré (indiquées en jaune). N'appliquez qu'une toute petite quantité de soudure.

Les LEDs sont interconnectées et forment une série de LEDs WS2812.

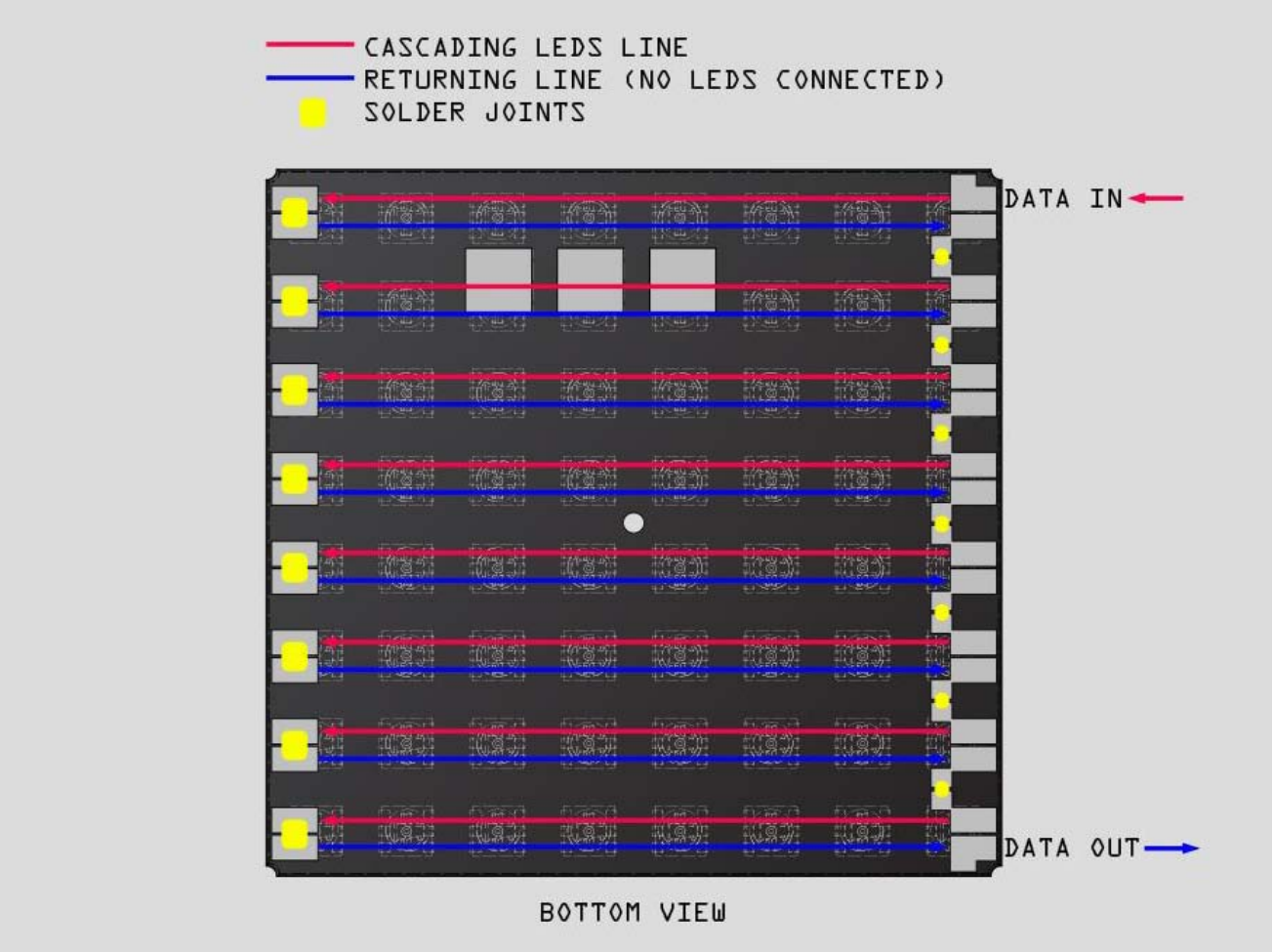


fig. 2

La figure ci-après montre comment assembler horizontalement 2 (ou plusieurs) panneaux en connectant les pattes à souder avec un fil nu.

Une ligne est composée de 16 LEDs au lieu de 8.

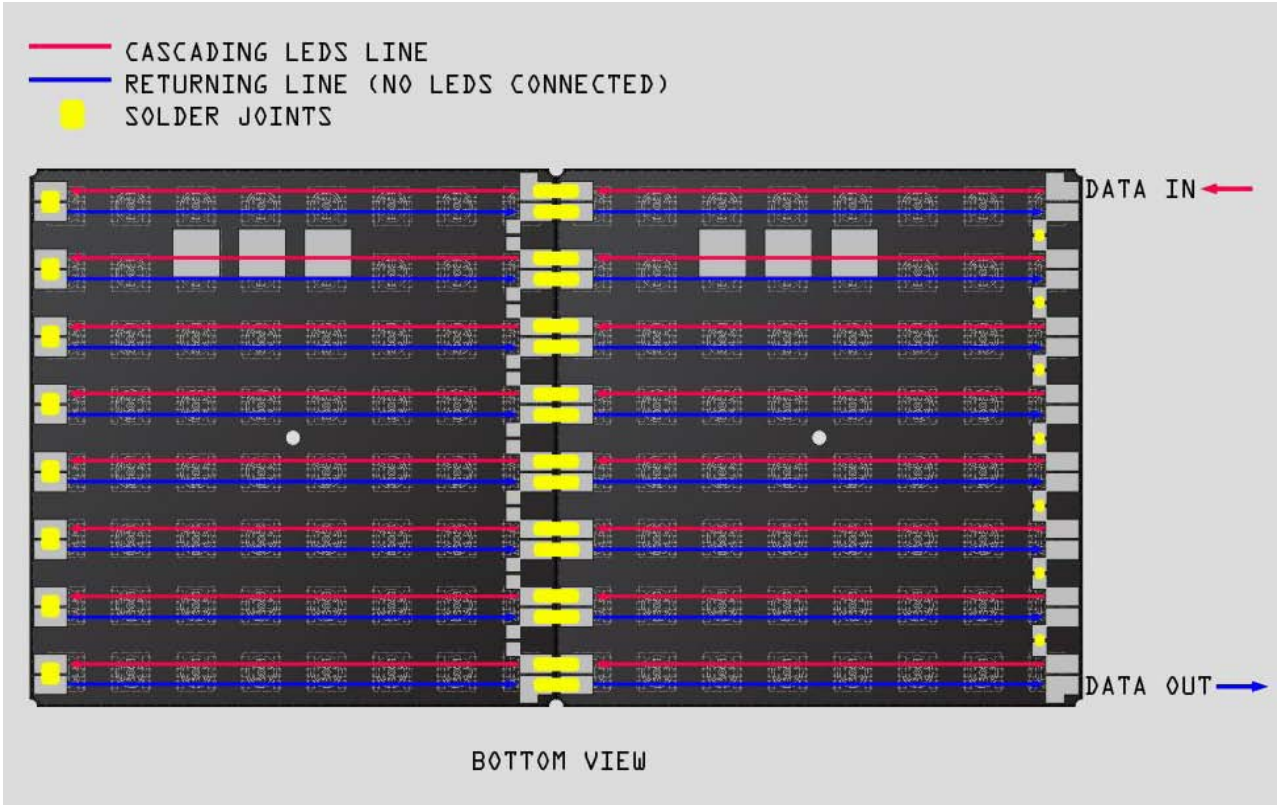


fig. 3

Vous pouvez également monter les panneaux verticalement en connectant la sortie DATA OUT de la ligné précédente des panneaux à l'entrée DATA IN de la ligne suivante des panneaux. Soyez prudent. Cette connexion est fragile dans un assemblage soudé.

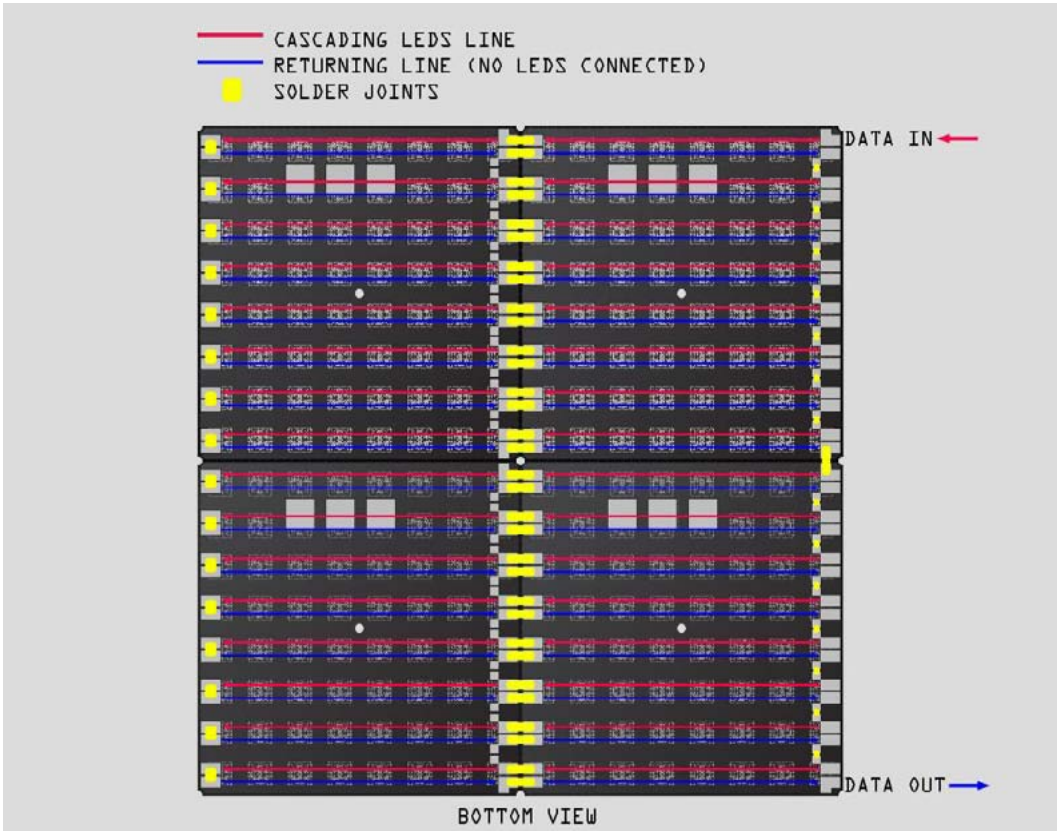


fig. 4

⚠ À RETENIR

- Les connexions de données doivent être aussi courtes que possibles.
- Ne séparez pas la ligne de données, puisque cela ne fonctionnera pas. (Une LED ne peut pas envoyer des données à deux ou plusieurs LEDs).
- Les LEDs fonctionnent avec un niveau TTL (5 V), mais également avec un microcontrôleur d'une tension de sortie de 3.3 V (5 V est préférable).
- Ajoutez une résistance de 470 Ohm entre la broche de sortie de données et la sortie du premier panneau. (Ce n'est pas toujours nécessaire mais peut être utile en cas de lignes de données sensibles au bruit).

Power supply

Powering the VM207 is a bit simpler. There are 3 contacts labeled **DV+**, **LV+** and **GND**, they respectively stand for **Data Voltage +**, **LED Voltage +**, and **Ground**. The WS2812 LEDs that are mounted on the panels are the 6 pin variant and these have separate pins for the 5 V for the LED die and the 5 V for the IC die inside the package. So the DV+ pad is connected to all the IC die pins and the LD+ pin is connected to all the LED die pins. In most occasions you can just connect these two and all will be OK. If you are worried by brown-outs due to line-loss when you have a lot of panels that are operating at high brightness, you can have the IC dies on a seperate 5 V supply (common ground!)

So the easiest way to power the panel is as shown in the drawing below:

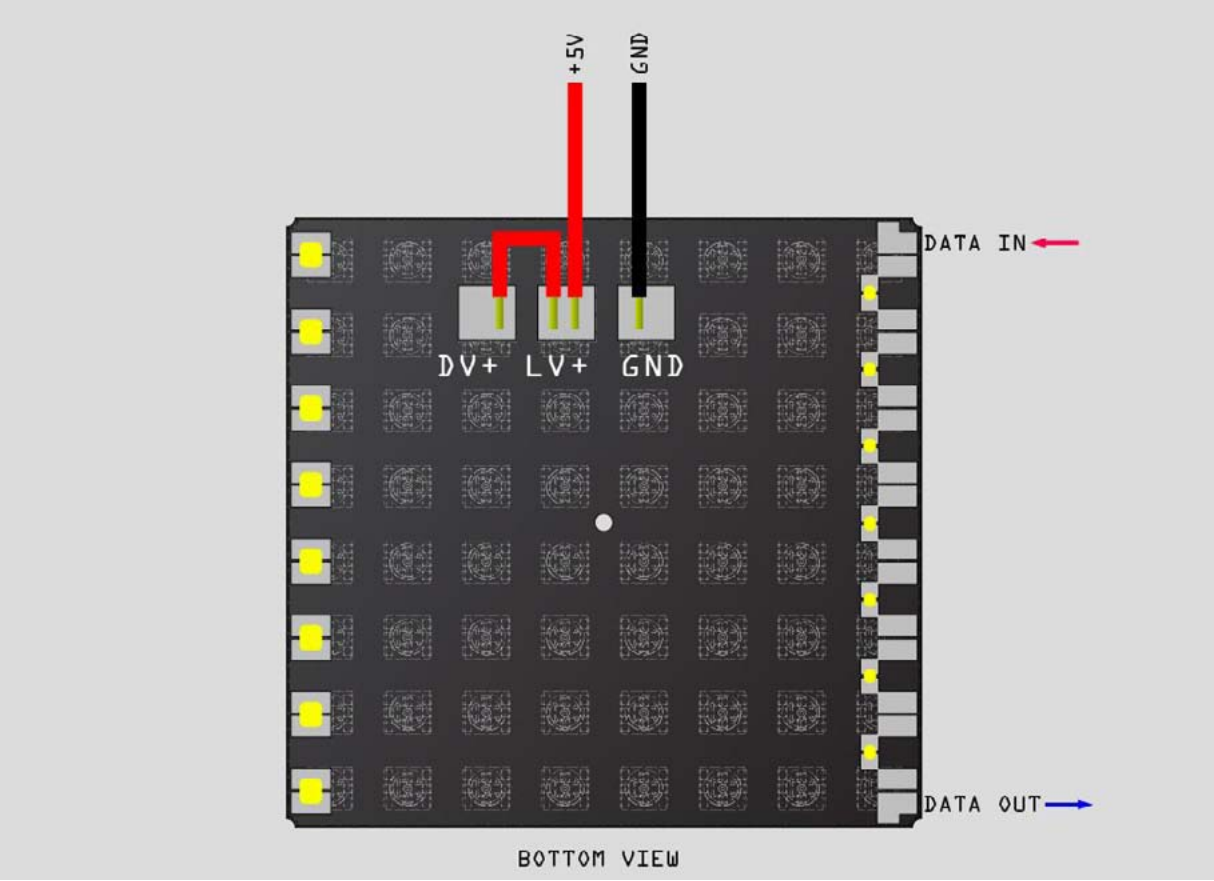


fig. 5

Si vous utilisez plusieurs panneaux, montez-les en parallèle. **Utilisez une section de câble appropriée ou plusieurs câbles depuis l'alimentation jusqu'aux panneaux!**

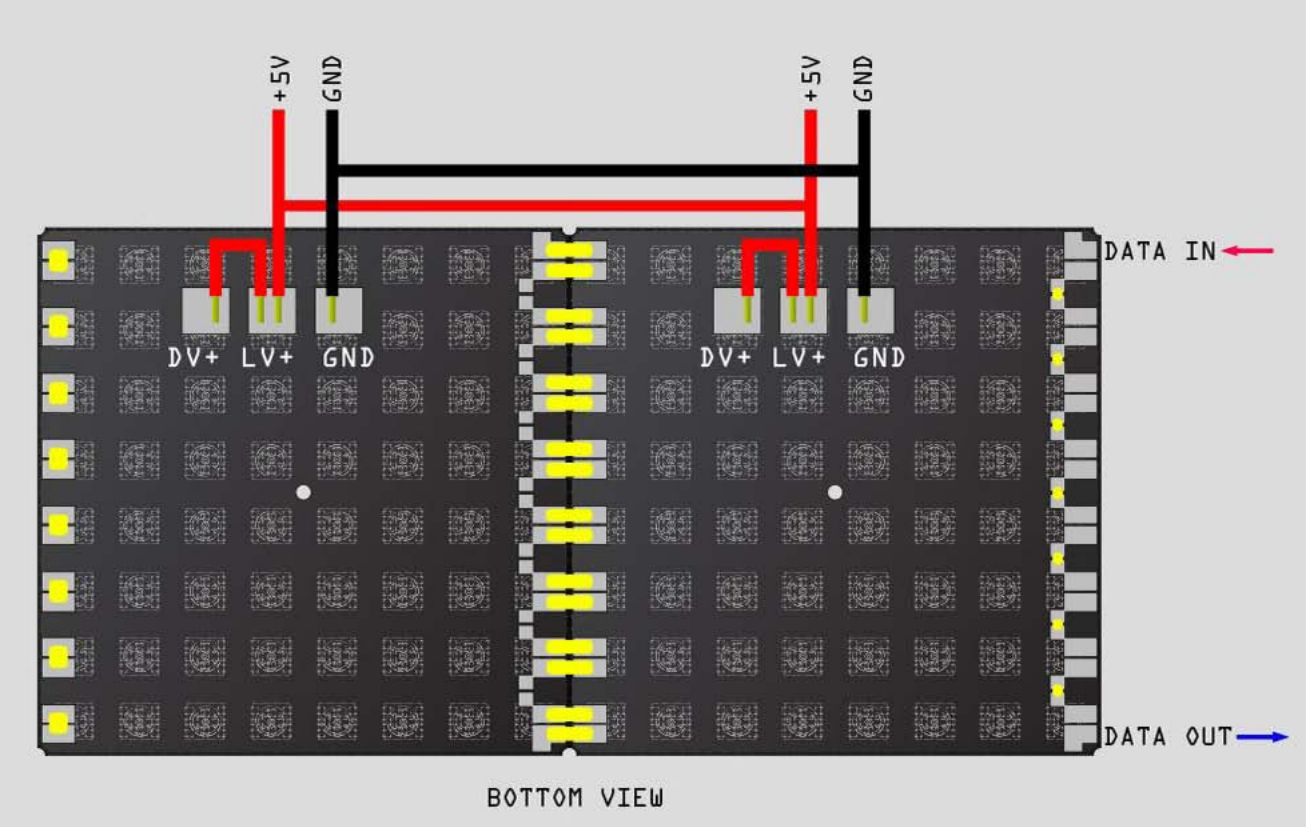


fig. 6

A RETENIR

- Connectez toujours un condensateur de 1000 µf (incl.) en parallèle à la source d'alimentation.
- N'allumez ou n'éteignez pas l'alimentation lorsque les panneaux sont connectés. Les crêtes lors de l'allumage et l'extinction pourraient endommager les LEDs de votre panneau.
- Connectez toujours la masse en premier.
- Utilisez un câble approprié pour alimenter les panneaux, puisqu'ils consomment beaucoup de courant (3.5 A par panneau à puissance maximale). La perte de tension dans le câble peut être un facteur lorsque vous utilisez de tels panneaux.
- Ne regardez pas directement dans les LEDs à luminosité maximale, cela pourrait vous désorienter.
- L'usage de plusieurs panneaux à puissance maximale peut générer beaucoup de chaleur. Dans certaines applications il faudra dissiper cette chaleur (ventilateur, dissipateur, ...) pour une max. durée de vie des panneaux.

Contrôler la matrice à LED

Quel microcontrôleur choisir?

Vous pouvez utiliser plusieurs microcontrôleurs ou des plateformes microcontrôleur pour contrôler votre panneau. La section suivante décrit comment utiliser votre panneau avec une plateforme microcontrôleur, qui est compatible avec Arduino IDE (p. ex. Uno, Mega, Teensy, etc.).

Si vous ne souhaitez pas utiliser de plateformes, créez un flux de données pour piloter une série de LEDs WS2812. Pour plus d'informations, consultez: WS2812 DATASHEET.

Connecter un panneau à une carte Arduino Uno

Consultez la figure ci-dessous pour connecter un panneau à une carte Arduino Uno. La broche 6 est utilisée comme sortie de données mais cette valeur peut facilement être modifiée dans le code. Alimentez l'Arduino et la VM207. Connectez les câbles de masse.

ATTENTION

N'utilisez jamais la 5 V de l'Arduino pour alimenter les panneaux. Ces panneaux consomment trop de courant pour le régulateur de l'Arduino. Cela pourrait endommager la carte Arduino.

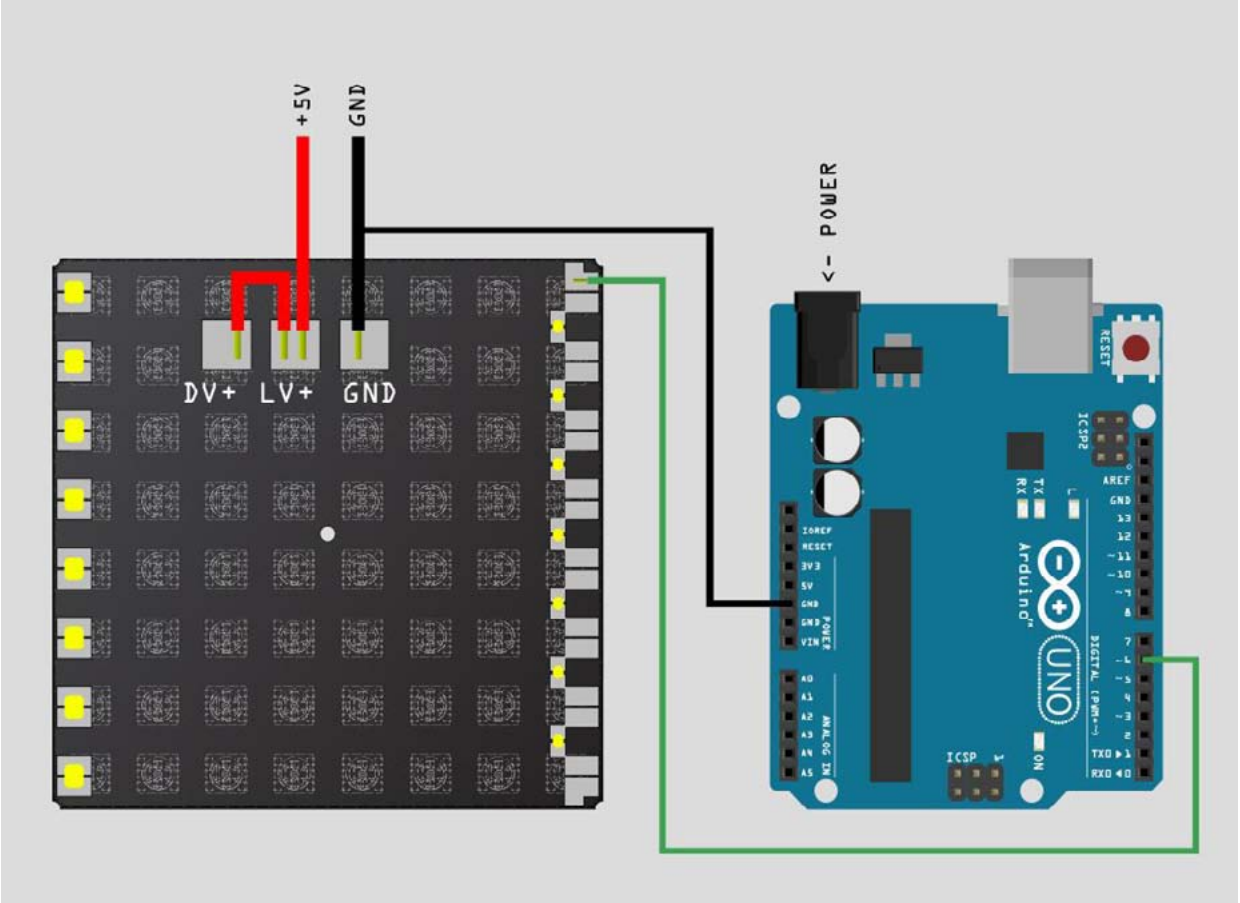


fig. 7

Bibliothèque NeoPixel d'Adafruit

Tout d'abord, un petit mot sur la bibliothèque NeoPixel d'Adafruit. La bibliothèque créée par Adafruit, permet de piloter individuellement une série de LEDs WS2812. Vous n'avez pas besoin de la bibliothèque pour afficher des caractères ou des formes graphiques sur le panneau, mais peut parfois être utile pour piloter individuellement chaque LED.

Téléchargez la bibliothèque NeoPixel sur: https://github.com/adafruit/Adafruit_NeoPixel (appuyez sur **Download ZIP**)

Ensuite, installez la bibliothèque dans votre installation Arduino (placez le dossier téléchargé et décompressé dans le dossier "Libraries" de l'installation Arduino) et lancez le logiciel Arduino.

Allez à : **File > Examples > Adafruit Neopixel > simple**. Arduino ouvre une fenêtre contenant le sketch/croquis.

```
// 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);
```

Que se passe-t-il ?

Le sketch commence par l'inclusion de la bibliothèque Adafruit.

#define PIN 6 indique que la constante 'PIN' = 6 et définit la broche 6 comme broche de sortie numérique. Pour changer la position de la broche (broche 13), ajustez la ligne de code comme suit : **#define PIN 13**

La ligne **#define NUMPIXELS 16** indique le nombre de LEDs connectées. Pour contrôler 1 panneau, saisissez la ligne comme suit : **#define NUMPIXELS 64**. Pour contrôler 2 panneaux, remplacez la valeur par 128, 3 panneaux = 192, etc.

La ligne de code suivante s'affiche comme suit : **Adafruit_NeoPixel pixels = Adafruit_ NeoPixel(NUMPIXELS, PIN, NEO_GRB + NEO_KHZ800);**

Cette ligne donne un nom au panneau ou à l'assemblage des panneaux, dans l'exemple : "**pixels**". Vous pouvez facilement changer le nom en "panel", mais il faudra remplacer toutes les occurrences du nom "pixels" par "pixels" dans le reste du programme.

La ligne suivante permet de spécifier le modèle des "pixels" (notre panneau) : **Adafruit_ NeoPixel(NUMPIXELS, PIN, NEO_GRB + NEO_KHZ800);**

- NUMPIXELS = valeur précédemment initialisée dans le programme. Correspond au nombre de LEDs connectées.
- PIN = valeur précédemment initialisée dans le programme. Correspond à la broche de sortie sur laquelle sont connectées les LEDs.
- NEO_GRB = maintenir pour des LEDs WS2812 standard.
- NEO_KHZ800 = maintenir pour des LEDs WS2812 standard.

Le code suivant est une variable pour stocker une valeur, qui sera utilisée comme un délai dans la fonction principale. Si vous modifiez cette valeur, la vitesse à laquelle la fonction loop 'for' est exécutée dans la fonction 'loop', changera conformément.

```
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();
}
```

La ligne la plus importante du code est **strip.begin();** qui permet d'initialiser les LEDs. N'oubliez pas cette étape dans votre code.

La fonction 'loop' principale :

```
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).

  }
}
```

C'est le coeur du programme. La fonction 'loop' se répète et contient la fonction 'for', qui exécutera les lignes toutes les 500 ms :

```
    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).
```

La première ligne affecte la couleur (vert) au pixel stocké dans la valeur "i". L'appel à la deuxième ligne "pixels. show" permettra de rafraîchir le panneau. Ces 2 lignes sont appelées en boucle toutes les 500 ms et la valeur 'i' s'incrémente toujours de 1. Consultez: https://www.arduino.cc/en/reference/for pour plus d'informations. Les LEDs vertes s'allument progressivement à l'exécution de ce code sur l'Arduino.

Testez, compilez et téléversez le code sur votre Arduino. Connectez le tout. Les LEDs vertes s'allument progressivement (toutes les 500 ms), les unes après les autres. Une fois le panneau allumé en vert, les LEDs restent allumées. Pour changer, ajustez la fonction 'loop' et la valeur de délai réglée:

```
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.
}
```

Ce code supprimera les valeurs après la boucle initiale 'for', en utilisant une seconde boucle 'for'. De plus, nous avons changé la couleur de vert en bleu très saturé. Le niveau de luminosité est très élevé. Pour régler la luminosité du panneau complet, utilisez : **pixels.setBrightness(20);** cette fonction acceptera une valeur comprise entre 0 (éteint) et 255 (luminosité max.). Pour voir l'effet sur l'afficheur, changez le code de la fonction 'loop' (boucle):

```
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.
}
```

Voici maintenant le code complet : Il est conseillé de tester et d'expérimenter avant de continuer.

```
// 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.
}
```

Bibliothèque NeoMatrix

La bibliothèque NeoMatrix vous permet d'afficher des caractères ou des formes graphiques sur le(s) panneau(x). Pour utiliser cette bibliothèque, installez également la bibliothèque GFX d'Adafruit. Cette bibliothèque gère les opérations graphiques (formes, caractères et couleurs). La bibliothèque NeoMatrix envoie ces données aux panneaux.

Téléchargez la bibliothèque NeoMatrix sur: https://github.com/adafruit/Adafruit_NeoMatrix (appuyez sur le bouton **Download ZIP**)

Téléchargez la bibliothèque GFX sur: https://github.com/adafruit/Adafruit-GFX-Library (appuyez sur le bouton **Download ZIP**)

Installez les bibliothèques dans votre installation Arduino (placez les dossiers téléchargés et décompressés dans le dossier "Libraries" de l'installation Arduino) et lancez le logiciel Arduino. Assurez-vous d'avoir installé la bibliothèque NeoPixel.

Allez à: **File > Examples > Adafruit Neopixel > simple.** Arduino ouvre le sketch (croquis).

Le fichier d'en tête s'affiche comme suit :

```
#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 •vĨ (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);
```

La première partie inclut les 3 bibliothèques (NeoPixel, NeoMatrix et GFX).

Ensuite, définissez une broche de sortie. Procédez de la même manière que précédemment.

La section ci-dessous décrit la configuration de la bibliothèque, permettant ainsi de spécifier la structure du montage. Donnez un nom au panneau, dans notre cas : " matrix ". Il faut tenir compte de 5 paramètres :

- la largeur de la matrice (nombre de LEDs : 8, 16, 32, ...)
- la hauteur de la matrice (nombre de LEDs : 8, 16, 32, ...)
- le numéro de la broche de données sur laquelle les LEDs sont connectées.
- Cet argument indique la disposition du montage et est constitué d'un ensemble de drapeaux (flags) additionnés. Indiquez la position de la première LED dans la matrice. La première LED doit se trouver en haut à gauche : **NEO_MATRIX_TOP + NEO_MATRIX_LEFT**. Indiquez comment les LEDs sont connectées, en lignes ou en colonnes. Si vous utilisez les panneaux horizontalement, choisissez : **NEO_MATRIX_ROWS**. Ensuite indiquez l'ordre des lignes (ou colonnes) , progressif ou zigzag. Avec la VM207, l'ordre est TOUJOURS progressif : **NEO_MATRIX_PROGRESSIVE**.
- La dernière constante indique le type de LEDs utilisées dans la VM207, donc TOUJOURS : **NEO_GRB + NEO_KHZ800**.

Ces constantes sont additionnées en une seule valeur pour piloter 1 panneau. Ajustez le code de l'exemple, puisque il s'agit d'un code pour contrôler une matrice de 5 x 8 avec des caractéristiques différentes.

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

Pour un afficheur composé de plusieurs panneaux VM207, ajustez le code conformément.

Le code suivant est une série de 3 couleurs pour faire défiler les couleurs, afin d'obtenir un effet plus spécial.

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

Dans la fonction setup, la partie la plus importante du code est la fonction **matrix.begin()** permettant d'initialiser le panneau. Le code qui suit fait référence au contenu graphique affiché sur le panneau. Compilez et téléversez le code sur l'Arduino Uno. Si nécessaire, copiez le code complet ci-dessous:

```
// Adafruit_NeoMatrix example for single NeoPixel Shield.
// Scrolls •Howdў 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);
}
```

Si tout est correct, le mot “Howdy” s’affiche sur le panneau en 3 différentes couleurs.
Ce n’est qu’un exemple des nombreuses possibilités que vous offre la bibliothèque GFX. Consultez le mode d’emploi sur: https://learn.adafruit.com/downloads/pdf/adafruit-gfx-graphics-library.pdf .

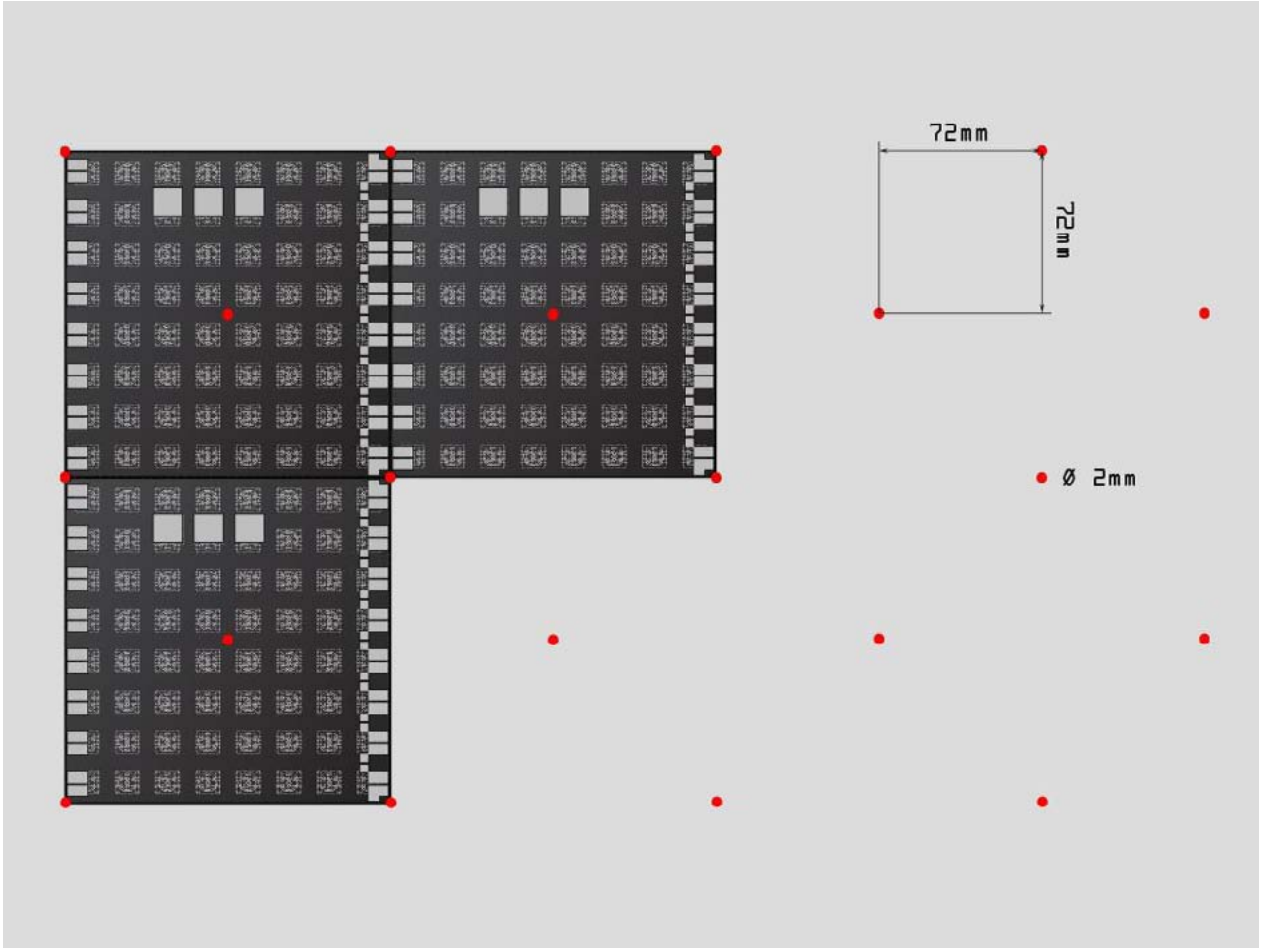
Les explications fournies dans ce mode d’emploi sont basées sur un afficheur. Le principe reste le même pour des panneaux si vous utilisez les bibliothèques NeoMatrix et GFX.

Découvrez toutes les possibilités des bibliothèques et créez des projets cool!

Monter la matrice à LED

Trous de montage

Les panneaux ont des trous de montage formant un motif lorsque vous utilisez plusieurs panneaux. Les trous ont un diamètre de 2 mm. Utilisez une petite vis M2 pour monter les panneaux. L'espacement entre les trous est de 72 mm et forment une matrice en damier. Voir ci-dessous:

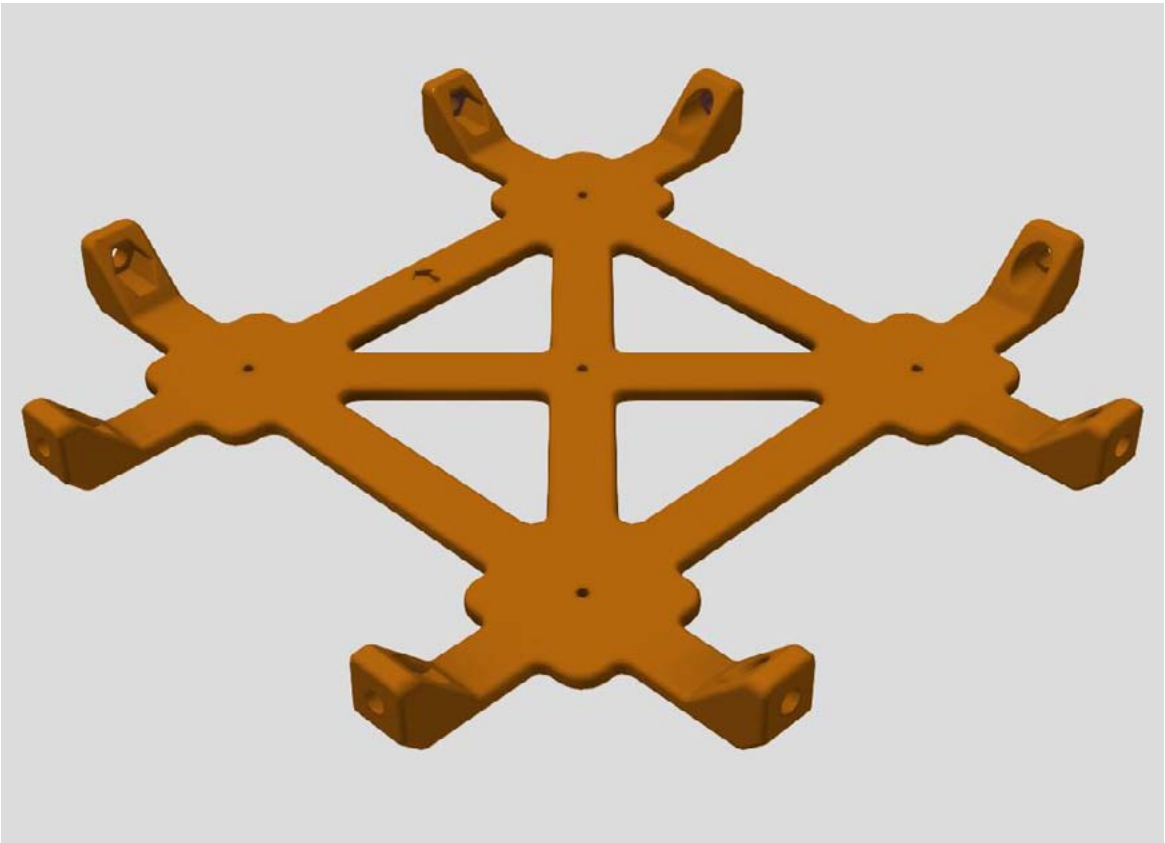


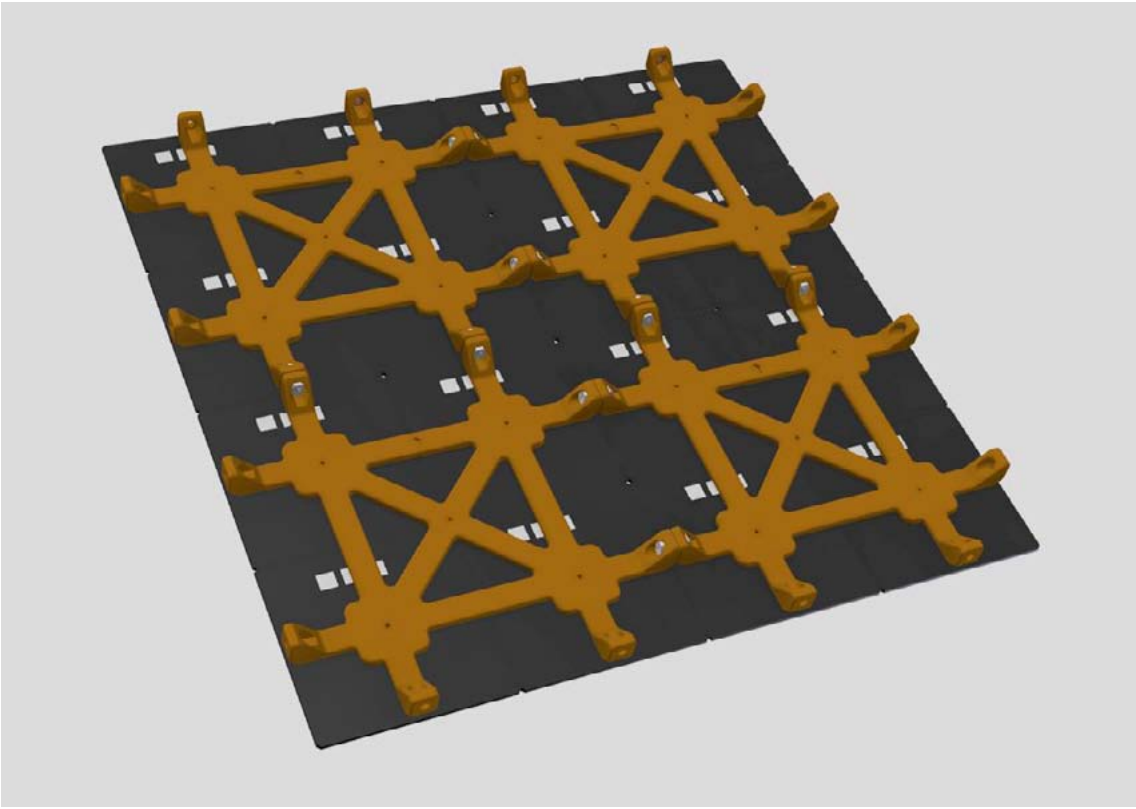
Supports de montage imprimés en 3D

Si vous avez une imprimante 3D , téléchargez le fichier pour imprimer les supports de montage :

vm207-tile_mount.stl

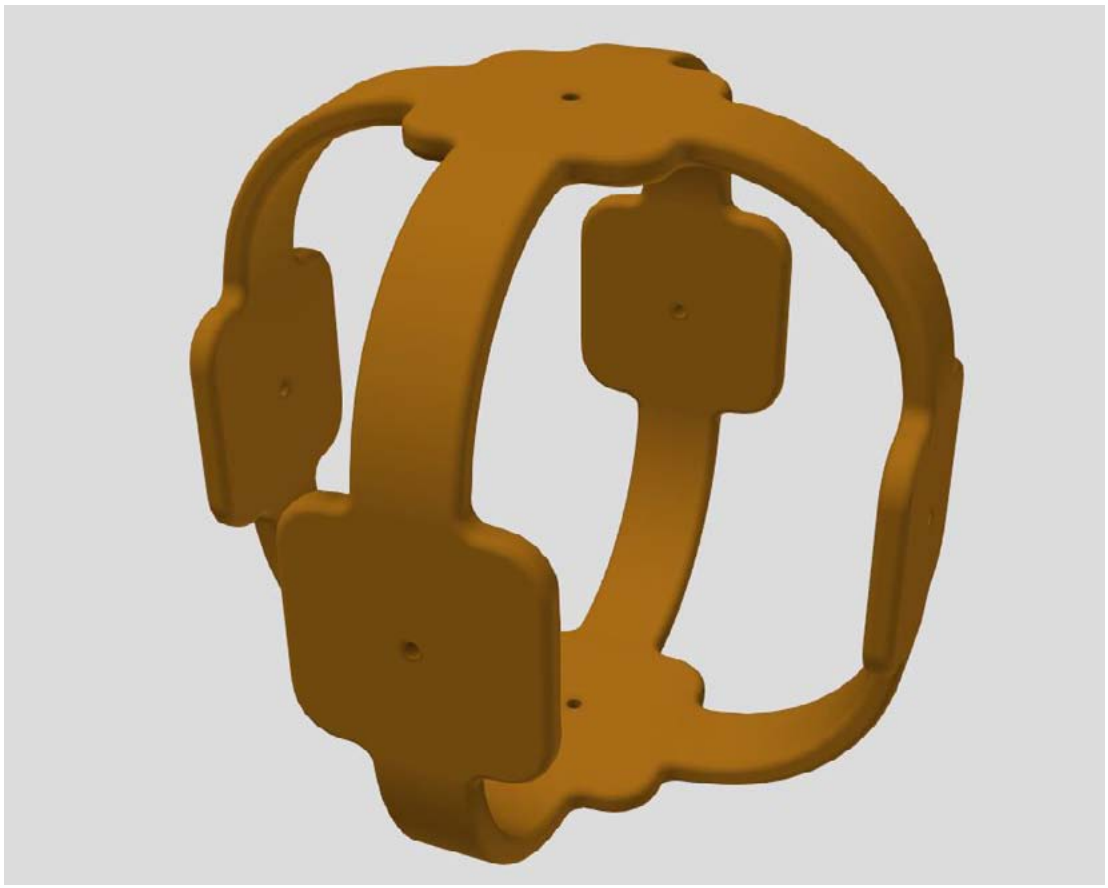
Connectez plusieurs supports de montage avec des boulons M3 x 10.

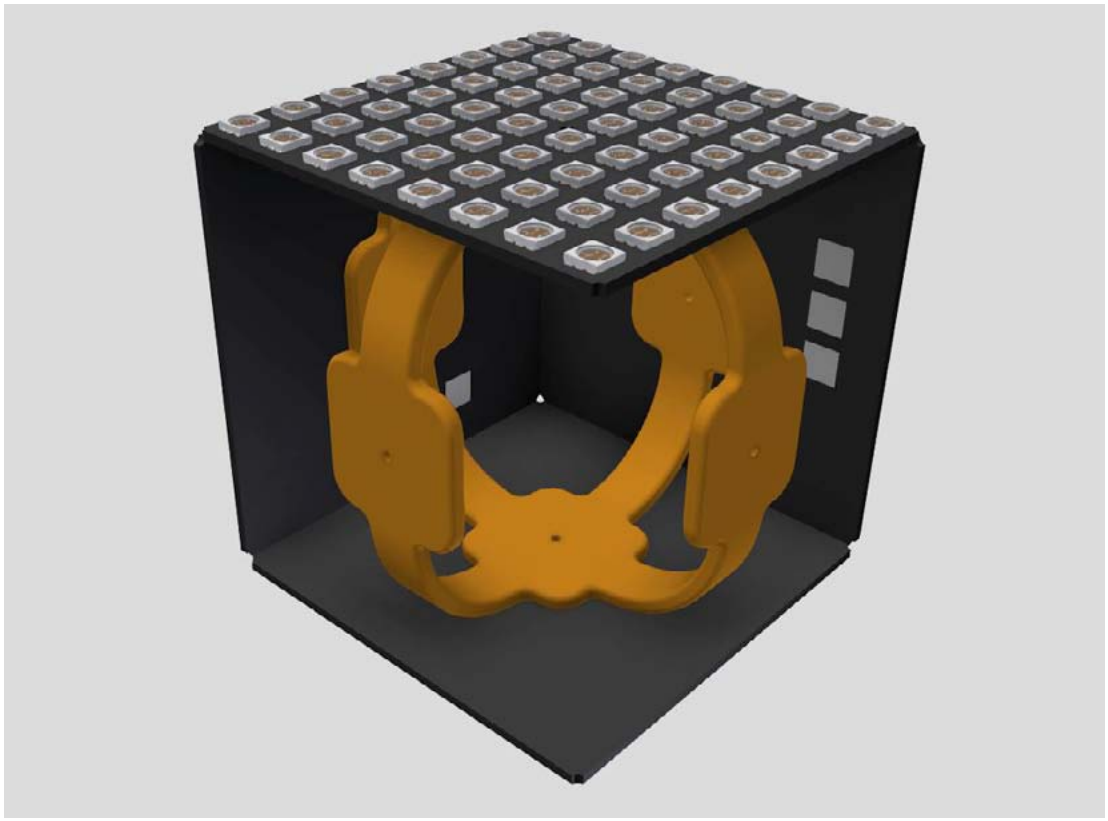




vm207-cube_mount.stl

Nécessite 6 x panneau VM207. Veillez à connecter les panneaux correctement!





ORDERCODE: VM207

REVISION: HVM207'1

UPC 8 16353 01511 6

EAN 5 410329 612955