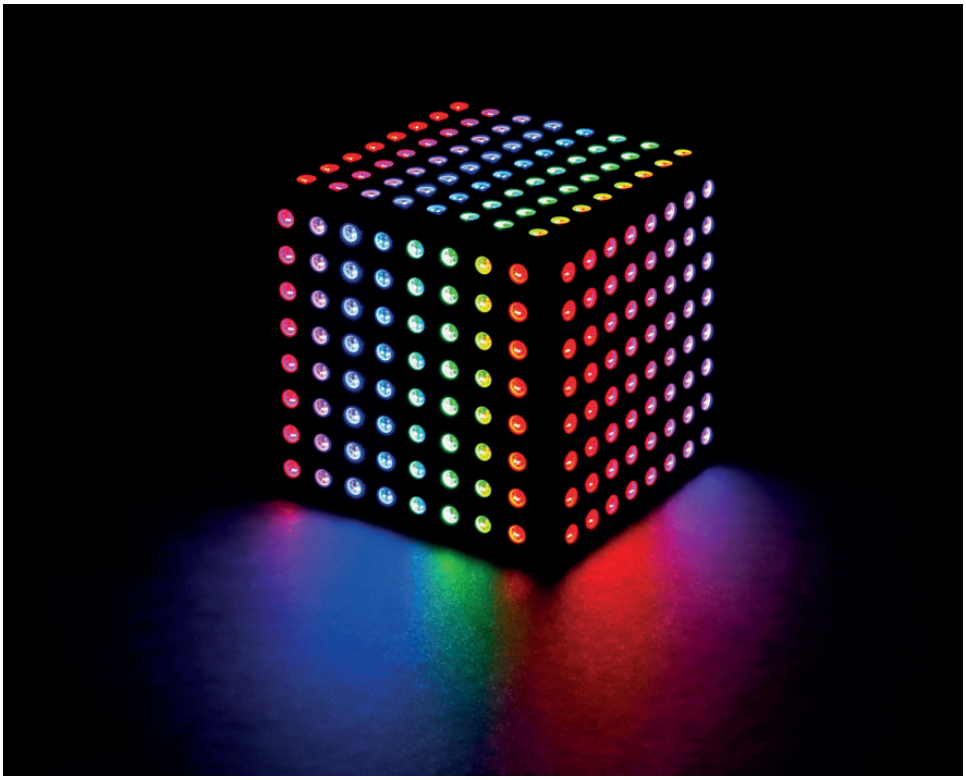
MANUAL DEL USUARIO

MATRIZ DE 64 LEDs RGB



Sumario

Preparar la matriz de 64 LEDs RGB 3

Cables de datos 3

Alimentación 6

Controlar la matriz de 64 LEDs RGB 8

El microcontrolador 8

Conectar la matriz a una Arduino™ Uno 8

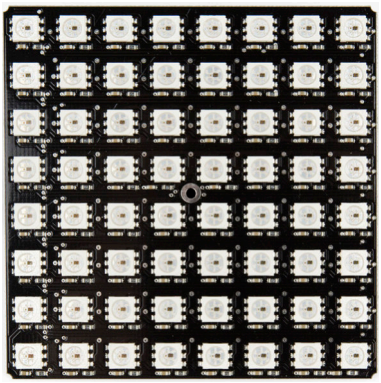
La biblioteca Adafruit™ neopixel 9

La biblioteca Adafruit™ neomatrix 13

Montar la matriz de 64 LEDs RGB 17

Los agujeros de montaje 17

Soportes de montaje (imprimidos con una impresora 3D) 18



¡vamos a ello!

Preparar la matriz de 64 LEDs RGB

Antes de explicar cómo escribir el software y visualizar algo en el (los) panel(es), indicaremos cómo preparar el (los) panel(es) de manera correcta.

Cables de datos

Antes de enchufar el aparato, es imprescindible saber cómo están interconectados los LEDs WS2812. Cada LED funciona como un registro de desplazamiento (shift register) y mueve datos de la entrada a la salida y luego al siguiente LED. Esto es la manera de la que se propagan los datos por un panel, de un LED a otro. En el VM207, los LEDs están conectados en filas y cada fila tiene una línea de regreso para que pueda conectar el principio de la siguiente fila al final de la fila anterior. Gracias a esta función, las filas no se mueven en zigzag y es posible conectar más paneles.

Examine cuidadosamente la siguiente figura para comprender cómo están cableados estos LEDs.
Study the drawing below carefully to understand how these LEDs are wired.

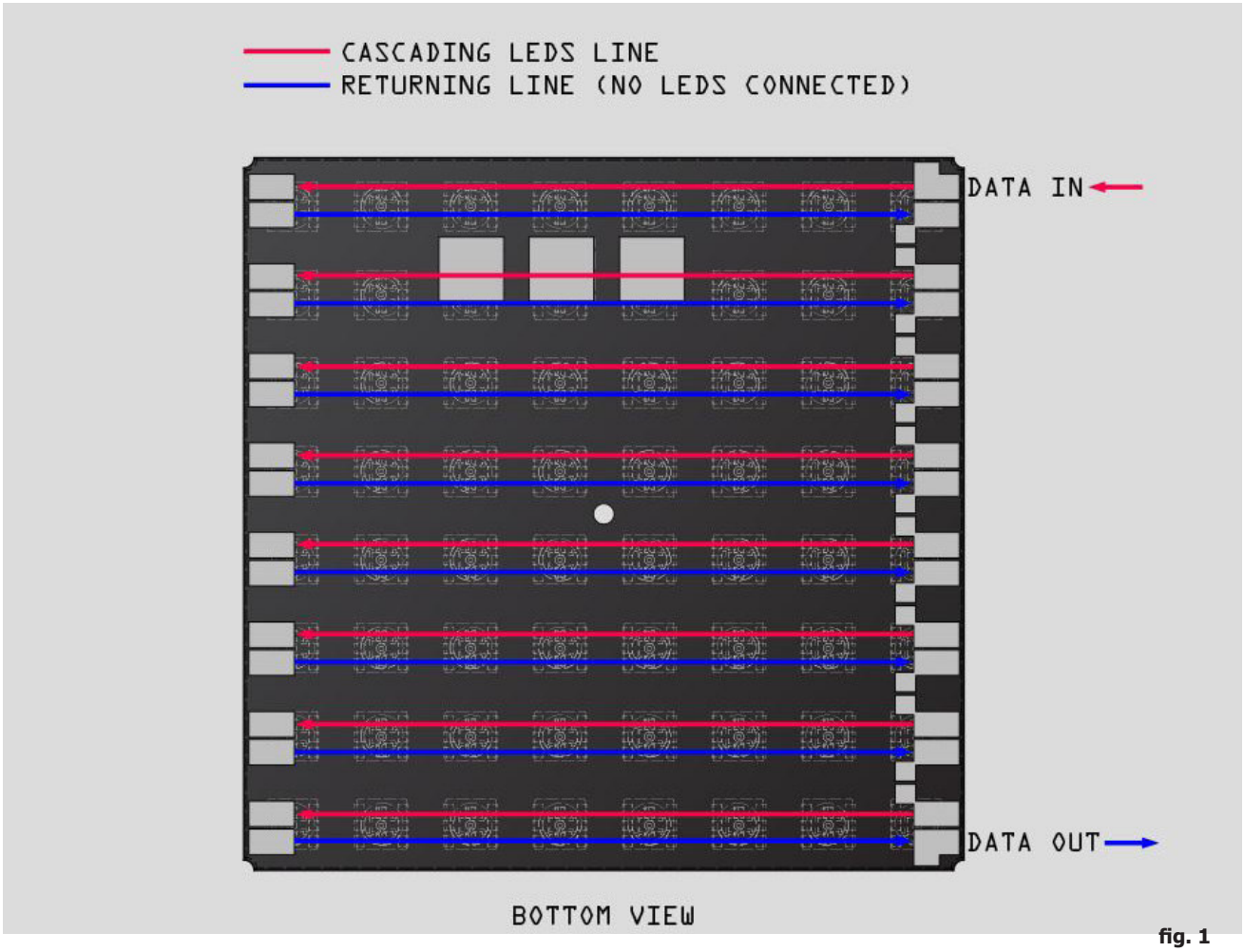


fig. 1

El panel no funcionará inmediatamente porque una vez llegue al final de la primera fila, no está conectada a nada. Los datos se pararán al final de la primera fila de LEDs y no continuará por los demás. Para utilizar el panel como una pantalla 8 x 8 y para dejar pasar los datos por cada LED, conecte las pistas como está indicado en la siguiente figura (conexiones de soldadura amarillas). Utilice sólo un poco de soldadura para conectar estas pistas.

Si sigue el display data pad ahora, verá que todas las filas están interconectadas y representan una cinta larga de LEDs WS2812.

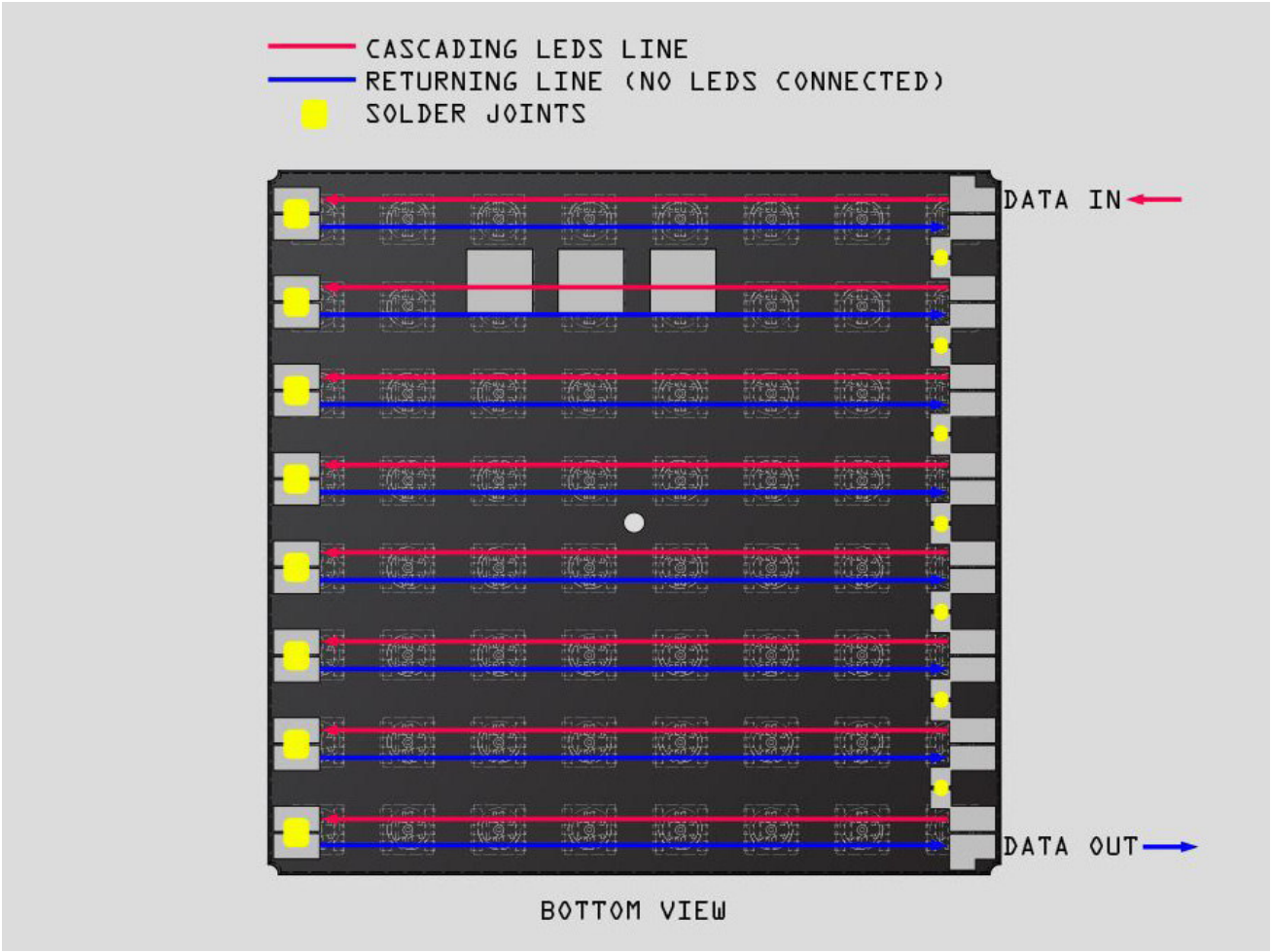


fig. 2

En la siguiente figura se visualiza cómo unir dos (o más) paneles uno al lado de otro conectando las juntas de los laterales con un poco de cable desnudo.

Ahora, una fila constará de 16 LEDs en lugar de 8.

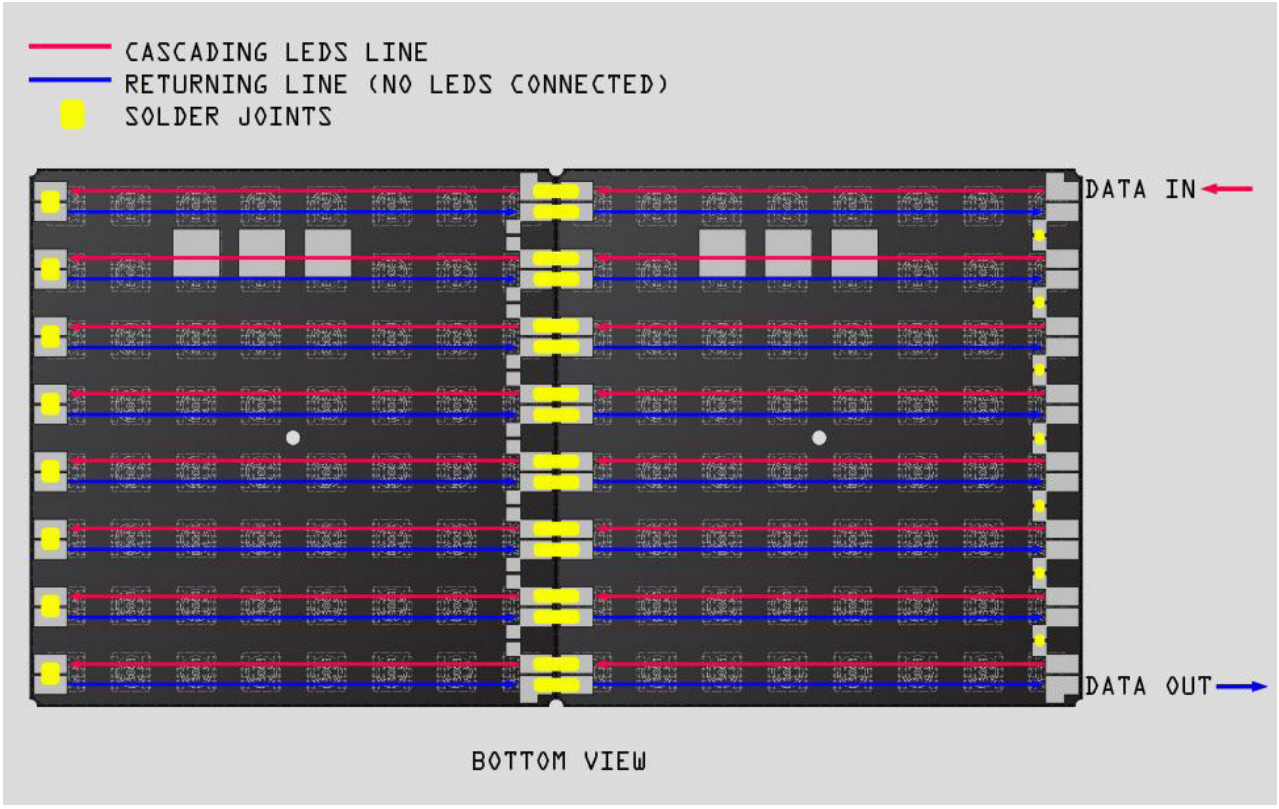


fig. 3

También es posible unir los paneles de manera vertical conectando el DATA OUT de la fila anterior de paneles al DATA IN de la nueva fila de paneles. Hágalo con cuidado, porque esta conexión puede ser frágil si cuenta con ensamblajes más grandes.

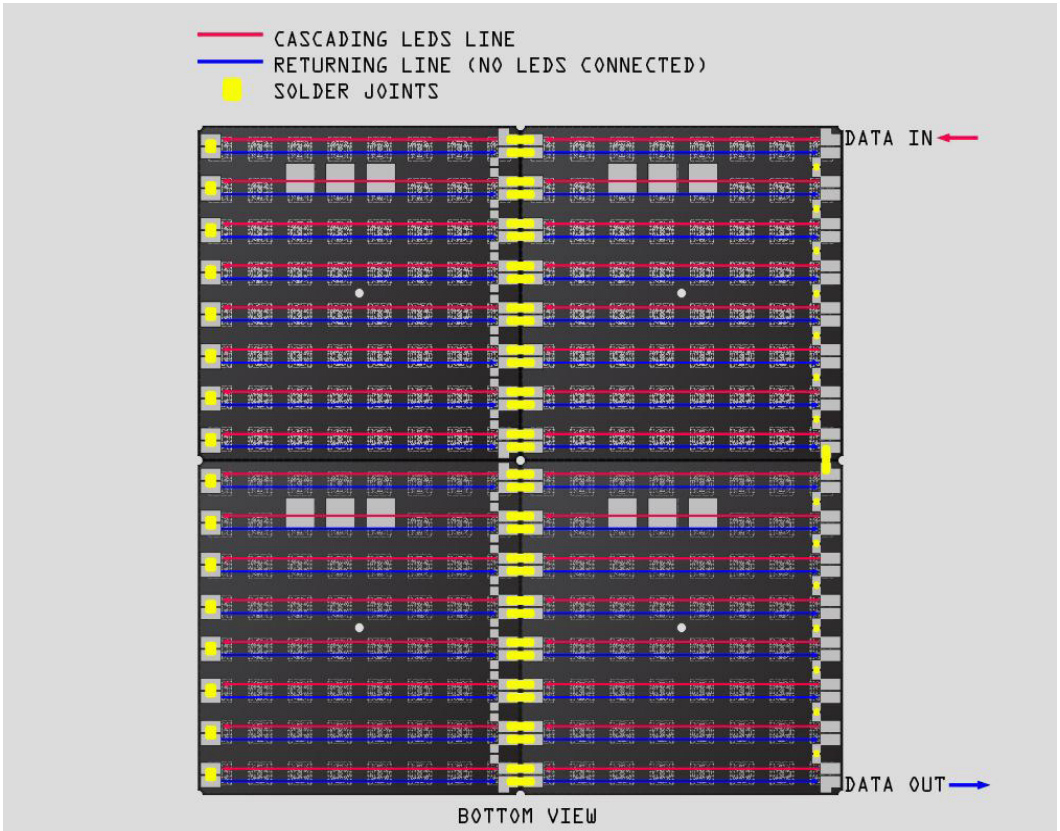


fig. 4

CONSEJOS A TENER EN CUENTA

- Asegúrese de que las conexiones de datos sean lo más cortas posibles.
- No divida la línea de datos porque esto no funcionará. (Un LED no puede enviar datos a dos o más LEDs).
- Los LEDs suelen funcionar con una señal de datos de nivel TTL (5 V) pero también funciona con una salida microcontrolador de 3.3 V. Sin embargo, 5 V es mejor.
- Instale un resistor de 470 Ohm entre el pin de la salida de datos del controlador y la entrada del primer panel. (Esto no siempre es necesario pero ayudará si tiene problemas con interferencias en las líneas de datos).

Alimentación

Es un poco más sencillo alimentar el VM207. Hay 3 contactos: **DV+**, **LV+** y **GND** (**Data Voltage +**, **LED Voltage +** y **Ground**. Los LEDs WS2812 que se montan en los paneles incluyen 6 pines y tienen pines separados para el 5 V para el dado LED y el 5 V para el dado CI. El pad DV+ está conectado a todos los pins dado CI y el pin LD+ a todos los pins dado LED. En la mayoría de los casos, basta con conectar estos dos y todo funcionará. Si su proyecto consta de muchos paneles LED y/o si necesita mucha intensidad luminosa, separe la alimentación de los paneles LED y el control. Asegúrese de que la masa de ambas alimentaciones sea común.

Consulte la siguiente figura para ver la manera más fácil de alimentar el panel:

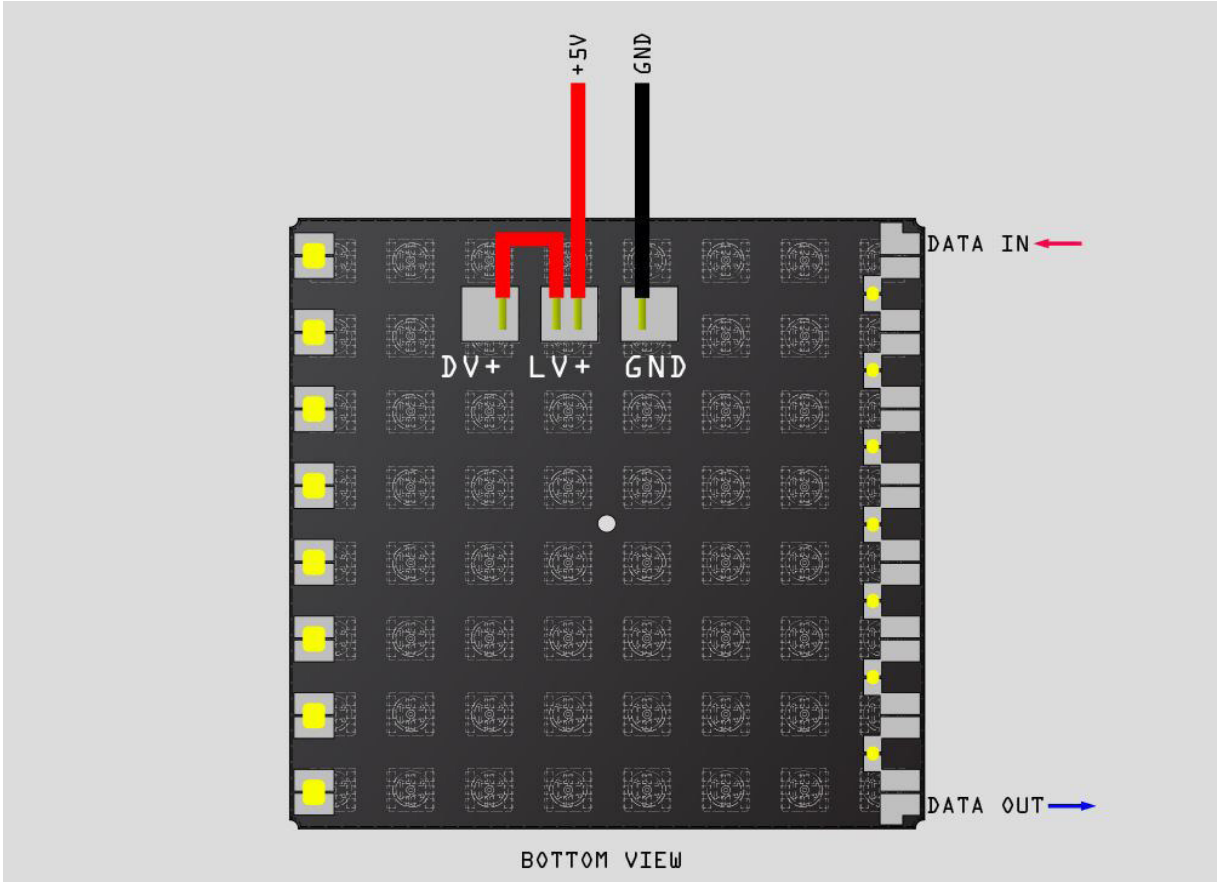


fig. 5

Si tiene más paneles, instálelos en paralelo. **¡Utilice cables con una sección adecuada y utilice cableado estrella!**

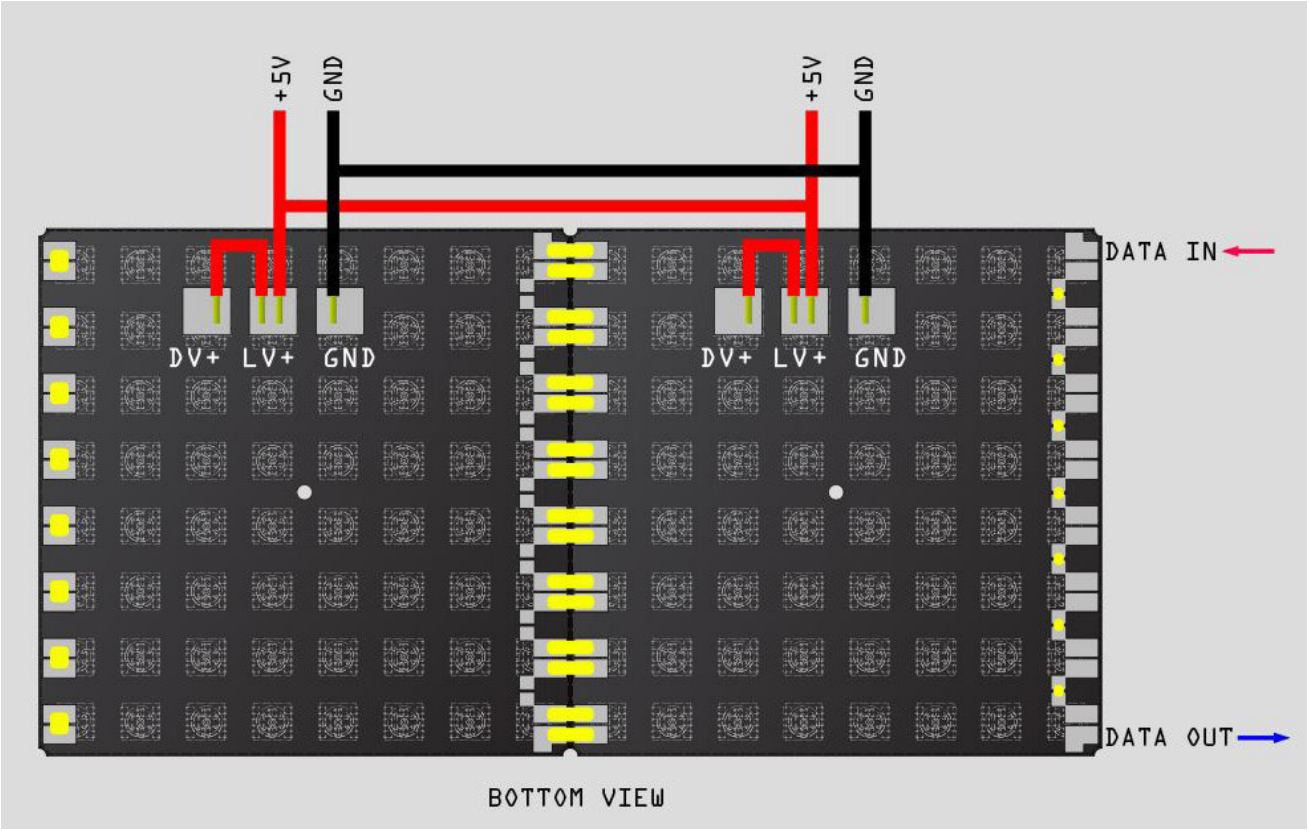


fig. 6

CONSEJOS A TENER EN CUENTA

- Utilice siempre un capacitor de 1000 µf (incl.) en paralelo con la alimentación.
- No active y desactive la alimentación cuando los paneles estén conectados porque los picos de tensión podrían dañar los LEDs.
- Primero, conecte siempre la masa.
- Utilice un cable adecuado para alimentar sus paneles porque podrían producir un alto amperaje (3.5 A por panel en caso de luminosidad máx.). La caída de tensión puede influenciar la intensidad luminosa a causa de la gran absorción de corriente.
- No se quede mirando a los paneles si los LEDs funcionan a la intensidad luminosa máx. Esto podría desorientarle.
- Se generará mucho calor si se utilizan muchos paneles a intensidad luminosa máx. Si fuera el caso, utilice un ventilador, disipador de calor, etc.) para aumentar la duración de vida de los paneles

Controlar la matriz de 64 LEDs RGB

El microcontrolador

Puede utilizar muchos controladores o plataformas de microcontroladores para controlar el panel. Para facilitar las cosas, explicaremos cómo utilizar el panel con una plataforma de microcontrolador compatible con la Arduino IDE (p.ej. Uno, Mega, Teensy, etc.).

Si no quiere utilizar una de estas plataformas, deberá aprender a crear la corriente de datos necesaria para controlar una serie de LEDs WS2812. Encontrará la información necesaria aquí: WS2812 DATASHEET.

Conectar un panel a una Arduino Uno

Utilice la siguiente figura para conectar un panel a una Arduino Uno. De momento, pin 6 es la salida de datos pero es posible cambiar esto más tarde. Conecte la Arduino y el VM207 a la red eléctrica y conecte las masas.

ADVERTENCIA

No utilice el 5 V de la Arduino para alimentar uno o más paneles. Estos paneles utilizan demasiada corriente para el regulador de la Arduino. Podría destruir la placa board Arduino al hacer esto.

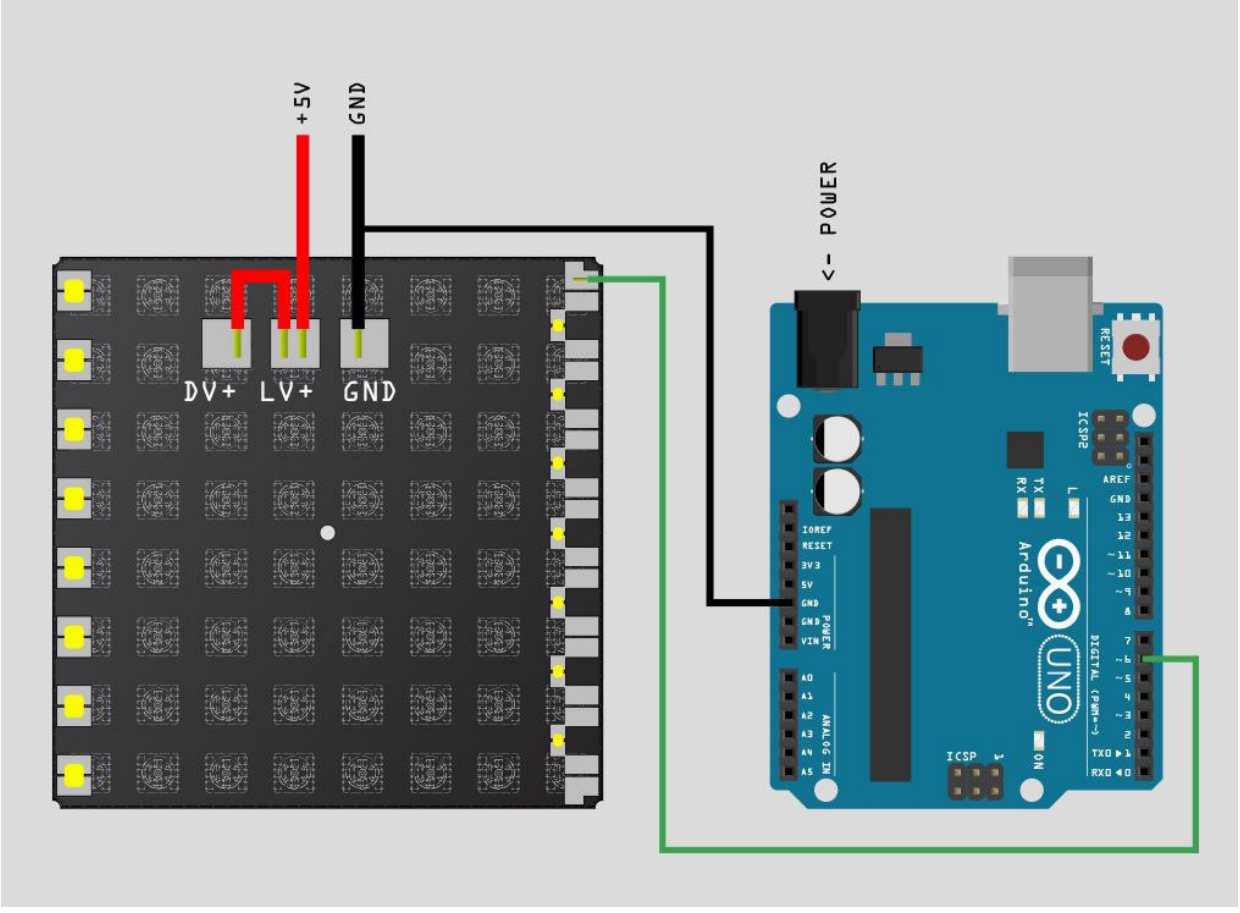


fig. 7

La biblioteca Adafruit neopixel

Primero, explicaremos la biblioteca ADAFRUIT NEOPIXEL Esta biblioteca, escrita por Adafruit puede controlar una serie de LEDs WS2812 por separado. Esto no es útil si quiere crear texto o formas en el panel pero es ideal si quiere controlar cada LED por separado.

Descargue la biblioteca ADAFRUIT NEOPIXEL: https://github.com/adafruit/Adafruit_NeoPixel (pulse **Download ZIP**)

Ahora, puede instalar esta biblioteca en la instalación Arduino (ponga el fichero descargado en el fichero de la instalación Arduino). Luego, inicie el software Arduino (Instale también la biblioteca ADAFRUIT NEOPIXEL).

Ahora, haga clic en: **File > Examples > Adafruit Neopixel > simple**, para que Arduino abrirá este 'sketch'.

La parte superior del fichero que acaba de abrir se visualizará:

```
// 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 sig-
nals.
// Note that for older NeoPixel strips you might need to change the third parameter--see the strand-
test
// example for more information on possible values.
Adafruit_NeoPixel pixels = Adafruit_NeoPixel(NUMPIXELS, PIN, NEO_GRB + NEO_KHZ800);
```

¿Qué pasa aquí?

La primera parte sólo trata de cómo incluir la biblioteca Adafruit.

Luego, #define PIN 6 explica al programa que PIN es igual a 6 y que la corriente de datos viene del pin 6 de la Arduino. Por ello, si quiere cambiar esto a 13, deberá cambiar este código de línea a: **#define PIN 13**

Luego, tenemos la línea #define NUMPIXELS 16 que explica al programa cuántos LEDs van a ser controlados. Para controlar 1 panel, esta línea debería parecerse a: #define NUMPIXELS 64. Si tiene 2 paneles, ponga 128. En caso de utilizar 3 paneles ponga 192, etc.

Luego, esta línea se visualizará: **Adafruit_NeoPixel pixels = Adafruit_NeoPixel(NUMPIXELS, PIN, NEO_GRB + NEO_KHZ800);**

Esta línea da un nombre a los LEDs. En este caso el nombre es 'pixels'. Puede cambiar esto fácilmente en "panel" o´lo que quiera. Sin embargo tenga en cuenta que deberá cambiar cada "pixel" en el resto del programa.

Luego, le explicaremos al programa de qué "pixels" consta (nuestro panel en este caso): **Adafruit_ NeoPixel(NUMPIXELS, PIN, NEO_GRB + NEO_KHZ800);**

- NUMPIXELS = Hemos determinado este valor anteriormente en el programa. Esto es el número de píxeles que debe ser controlado.
- PIN = Hemos determinado este valor anteriormente en el programa. Esto es el pin 'display datastream' .
- NEO_GRB = Deje esto para LEDs WS2812 estándar.
- NEO_KHZ800 = Deje esto para LEDs WS2812 estándar.

Luego se visualizará el siguiente código que sólo es una variable que almacena un valor que se utilizará como una temporización en la función principal. Si cambia este valor, la velocidad con la que el 'for-loop' se ejecuta en la función 'loop' se cambiará.

```
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 parte más importante del código es la línea **strip.begin();**. Esto iniciará los LEDs. No olvide este paso en su propio código.

Luego, tenemos la función 'loop' principal:

```
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).

  }
}
```

Aquí pasa algo mágico. La función 'loop' se repite cada vez y dentro de esta función 'loop' hay una función 'for-loop' que ejecutará estas líneas cada medio segundo:

```
    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 primera línea dará al pixel en la posición donde está almacenado el valor "i" un color verde. Sin embargo, el panel sólo se actualizará en cuanto la segunda línea "pixels.show" esté llamada. Estas dos líneas se llaman cada 500ms y el valor 'i' siempre se incrementa con 1. Para más información sobre el funcionamiento de 'for-loops', consulte: https://www.arduino.cc/en/reference/for about how for-loops work) . El panel se llena poco a poco de LEDs verdes al iniciar este código en una Arduino.

Intentemos eso, suba este código a la placa board Arduino y conecte todo. Se irá iluminando lentamente un LED verde más (cada 500 ms). Después de que todos los LEDs se hayan iluminado en verde el panel se quedará verde. Para cambiar esto, puede intentar cambiar la función 'loop' y el valor de temporización mencionado anteriormente (haga clic dos veces para seleccionar y copiar):

```
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.
}
```

Este código borrará todos los errores después de la función 'for-loop' inicial utilizando una segunda función 'for-loop'. También hemos cambiado el color a azul (completamente saturado).

Esto es un color muy luminoso. Para cambiar la luminosidad del panel entero, puede utilizar: pixels. setBrightness(20);Esta función aceptará un valor entre 20 y 255 (0 = luminosidad mín., 255 = luminosidad máx.). Puede ver cómo esto afectará la pantalla cuando cambie la función 'loop' (haga clic dos veces para seleccionar y copiar):

```
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.
}
```

A continuación puede ver cómo el código entero se visualiza de momento. Experimente con todo lo que ha aprendido en este capítulo antes de continuar. Haga clic dos veces para seleccionar y copiar.

```
// 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 sig-
nals.
// Note that for older NeoPixel strips you might need to change the third parameter--see the strand-
test
// 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.
}
```

La biblioteca Adafruit neomatrix

Si quiere visualizar un texto o dibujar figuras en el (los) panel(es), utilice la biblioteca ADAFRUIT NEOMATRIX. Para hacer funcionar esta biblioteca, necesitará también las bibliotecas ADAFRUIT GFX. Necesitará estas bibliotecas para crear las formas, las letras y los colores. La biblioteca neomatrix servirá para enviar todos los datos a los paneles.

Descargue la biblioteca ADAFRUIT NEOMATRIX aquí: https://github.com/adafruit/Adafruit_NeoMatrix (pulse **Download ZIP**)

Descargue la biblioteca ADAFRUIT GFX aquí: https://github.com/adafruit/Adafruit-GFX-Library (pulse **Download ZIP**)

Ahora, puede instalar esta biblioteca en la instalación Arduino (ponga el fichero descargado en el fichero de la instalación Arduino). Luego, inicie el software Arduino (Instale también la biblioteca ADAFRUIT NEOPIXEL).

Ahora, haga clic en: **File > Examples > Adafruit Neopixel > simple** para que Arduino abrirá este 'sketch'.

La parte superior del fichero que acaba de abrir se visualizará:

```
#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);
```

La primera parte incluye las 3 bibliotecas (NEOPIXEL, NEOMATRIX y GFX).

Luego, volveremos a determinar un pin de salida. Es el mismo principio que el del capítulo anterior.

Ahora, explicaremos cómo configurar la biblioteca para que sepa cómo es su panel. Primero, dele un nombre a su panel. En este caso: ”matrix”. Luego, hay 5 parámetros con que debe contar:

- La anchura de su matriz (número de LEDs que forman la anchura de su matriz (8, 16, 32, ...))
- La altura de su matriz (número de LEDs que forman la altura de su matriz (8, 16, 32, ...))
- El número PIN del pin DATA OUT
- Este parámetro describe la disposición del panel y consta de banderas que debes conectar. Primero, especifique dónde se encuentra el primer LED de la matriz. En la mayoría de los casos, este LED se encuentra en la esquina superior izquierdo: **NEO_MATRIX_TOP + NEO_MATRIX_LEFT**. Luego, seleccione si quiere conectar los LEDs en filas o en columnas. Si utiliza los paneles horizontalmente, seleccione: **NEO_MATRIX_ROWS**. Luego, escoja si las filas (o columnas) deben moverse de manera progresiva o en zigzag. Las filas (o columnas) del VM207 siempre se mueven de manera progresiva: **NEO_MATRIX_PROGRESSIVE**.
- El último parámetro trata de los LEDs que debe utilizar con el VM207. SIEMPRE utiliza: **NEO_GRB + NEO_KHZ800**.

Una todos estos parámetros de la siguiente manera para controlar un panel, por ejemplo (tendrá que cambiar el código del ejemplo por el siguiente código porque ha sido creado para controlar un panel 5 x 8 con otras características):

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

Si quiere construir una pantalla más grande con los paneles VM207, piense en ello como si fuera un solo panel grande y ajuste los parámetros.

El siguiente código en el programa sólo es una matriz con 3 colores para que podamos pasar por algunos colores para hacer que el ejemplo sea un poco más especial.

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

En la función ‘setup’ tiene que contar con algunas cosas pero la parte más importante es la función matriz. El siguiente código es para crear el texto animado que se visualizará en el panel. Por ello, compilaremos el código y subirlo a la Arduino Uno Si necesita el código entero, puede copiarlo a continuación:

```
// 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);
}
```

La palabra “Howdy” se visualizará en 3 colores diferentes.
Sólo es un ejemplo de las numerosas posibilidades que le ofrece la biblioteca GFX. Para más información, consulte el siguiente manual del usuario: https://learn.adafruit.com/downloads/pdf/adafruit-gfx-graphics-library.pdf

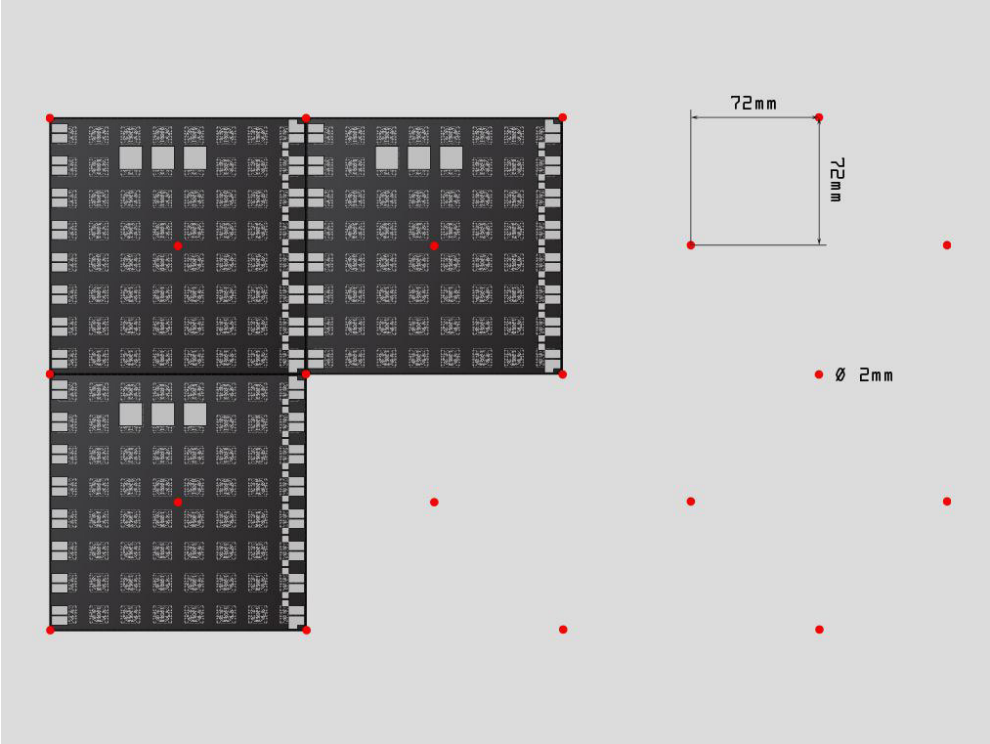
Este manual del usuario se base en una pantalla auténtica. Los paneles funcionan según el mismo principio si utiliza la biblioteca NEOMATRIX con la biblioteca GFX.

¡Descubra todo lo que puede hacer con estas bibliotecas y cree pantallas chulas!

Montar la matriz de 64 LEDs RGB

Los agujeros de montaje

Los paneles están equipados con agujeros de montaje a los que puede conectar diferentes paneles. Los agujeros tienen un diámetro de 2 mm. Utilice un pequeño tornillo M2 para fijar los paneles. Se encuentran en una distancia de 72 mm y son escalonados. Véase la siguiente figura:

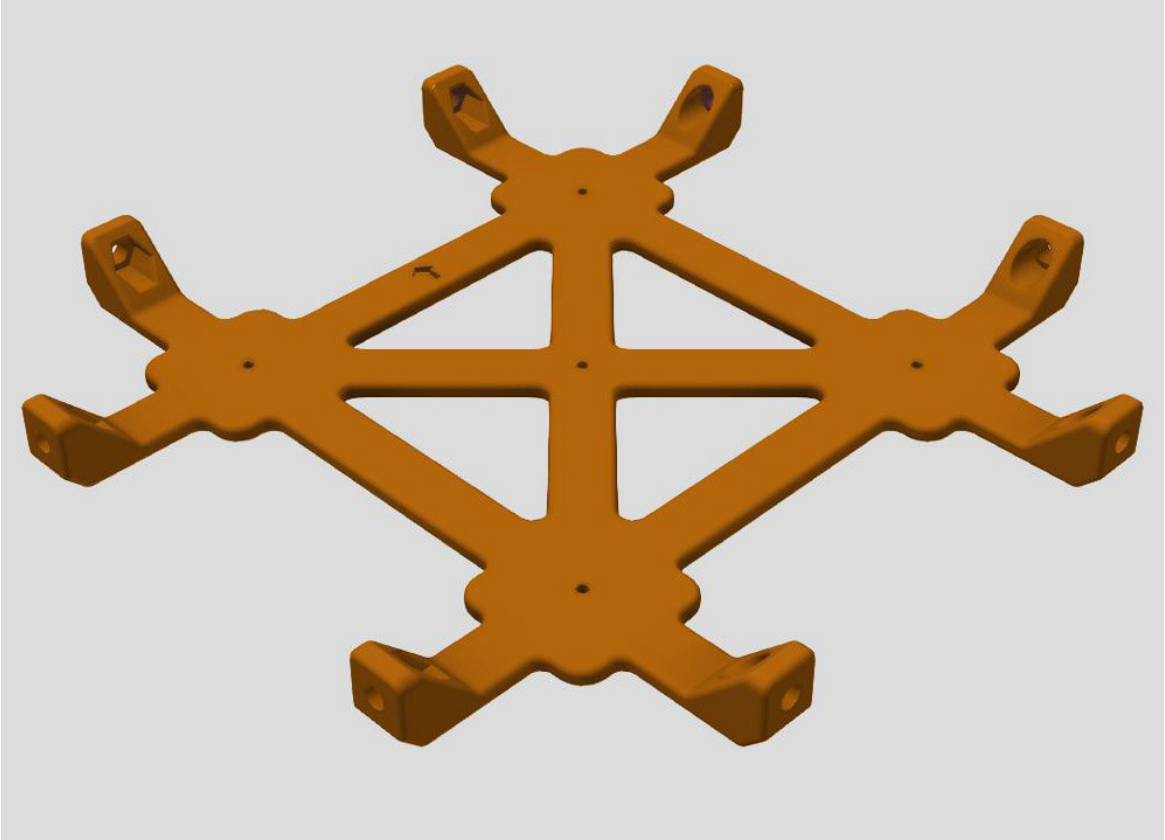


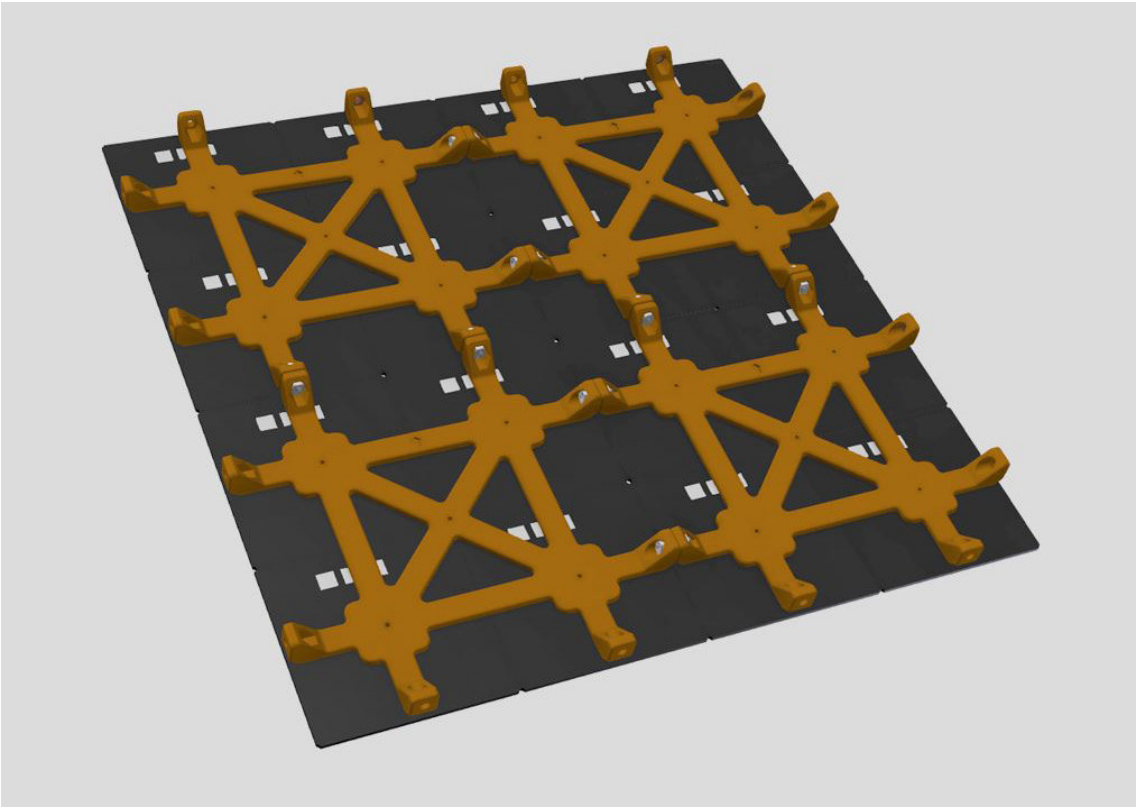
Soportes de montaje (imprimidos con una impresora 3D)

Puede descargar los siguientes soportes de montaje si tiene una impresora 3D:

vm207-tile_mount.stl

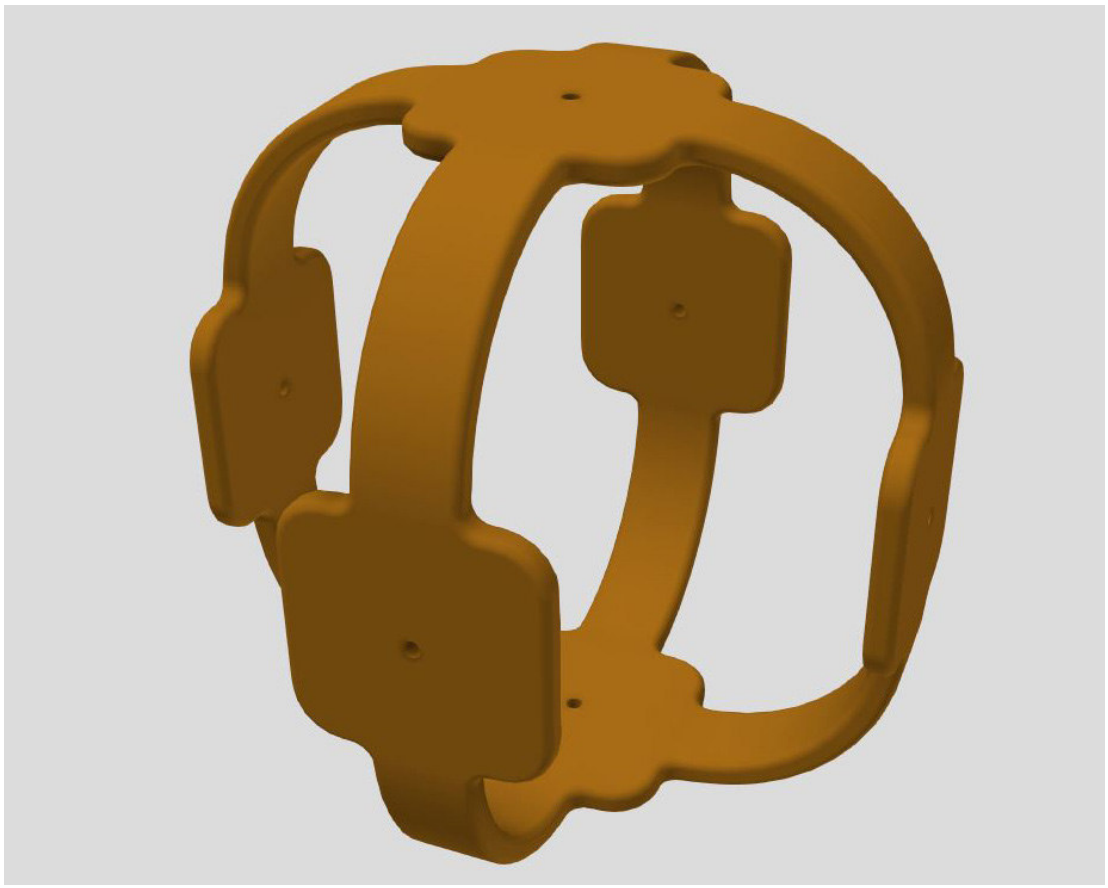
Los soportes de montaje se conectan con 10 tornillos M3.

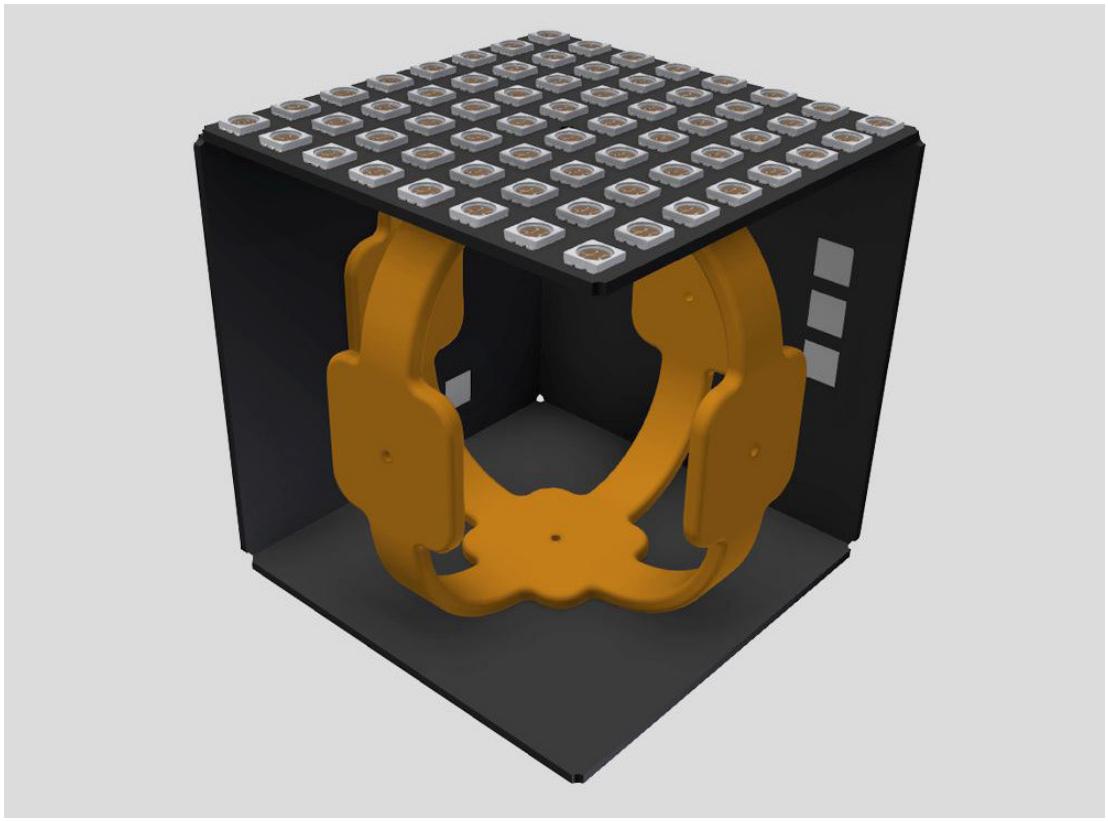




vm207-cube_mount.stl

Requiere 6 x panel VM207. Preste especial atención a la conexión de los diferentes paneles.





velleman®

ORDERCODE: VM207

REVISION: HVM207'1

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