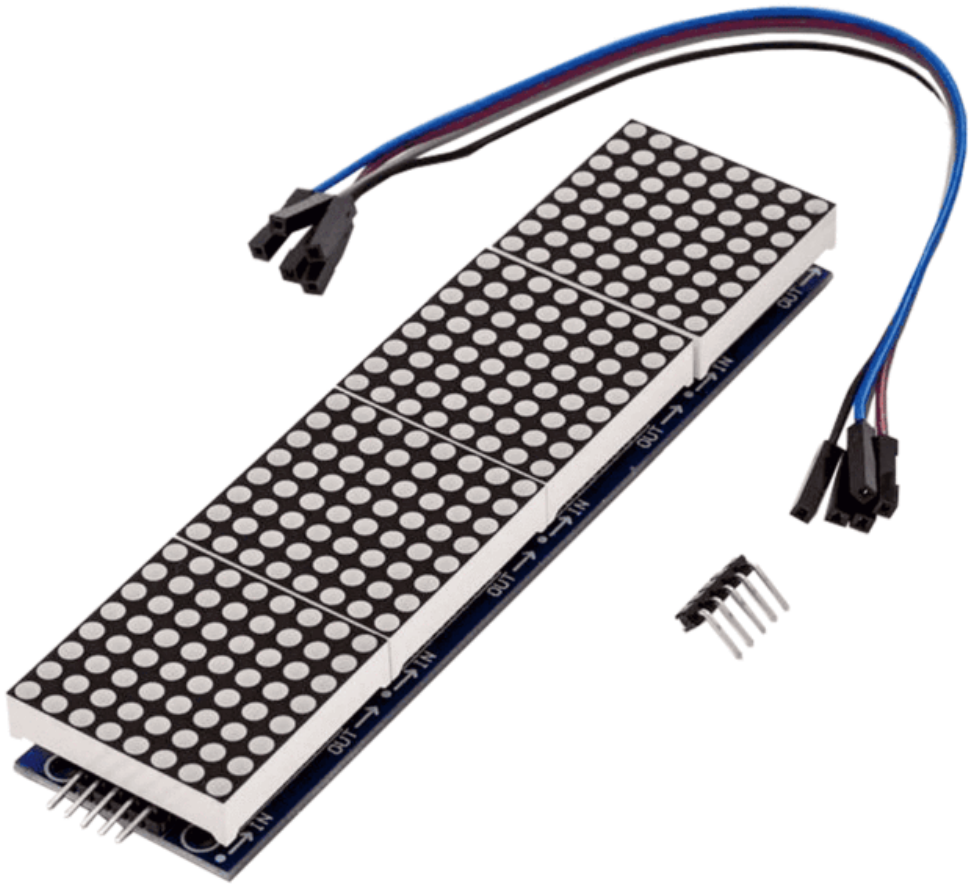
¡Bienvenido!

Muchas gracias por comprar nuestro AZ-Delivery Módulo de Pantalla de Matriz LED 4x64. En las siguientes páginas se presentará como utilizar y configurar este práctico dispositivo.

¡Diviértase!



Areas de aplicación
Educación y enseñanza: uso en escuelas, universidades e instituciones de formación para enseñar los conceptos básicos de electrónica, programación y sistemas integrados. Investigación y desarrollo: Uso en proyectos de investigación y desarrollo para crear prototipos y experimentos en los campos de la electrónica y la informática. Desarrollo de prototipos: Uso en el desarrollo y prueba de nuevos circuitos y dispositivos electrónicos. Proyectos Hobby and Maker: utilizado por entusiastas y aficionados a la electrónica para desarrollar e implementar proyectos de bricolaje.

Conocimientos y habilidades requeridos.
Conocimientos básicos de electrónica e ingeniería eléctrica. Conocimientos de programación, especialmente en el lenguaje de programación C/C++. Capacidad para leer esquemas y diseñar circuitos simples. Experiencia trabajando con componentes electrónicos y soldadura.

Condiciones de operación
El producto sólo puede funcionar con los voltajes especificados en la hoja de datos para evitar daños. Se requiere una fuente de alimentación CC estabilizada para su funcionamiento. Al realizar la conexión a otros componentes y circuitos electrónicos, se deben observar los límites máximos de corriente y voltaje para evitar sobrecargas y daños.

Condiciones ambientales
El producto debe utilizarse en un ambiente limpio y seco para evitar daños causados por la humedad o el polvo. Proteja el producto de la luz solar directa (UV), ya que esto puede afectar negativamente la vida útil de la pantalla.

Uso previsto
El producto está diseñado para su uso en entornos educativos, de investigación y desarrollo. Se utiliza para desarrollar, programar y crear prototipos de proyectos y aplicaciones electrónicos. El producto no pretende ser un producto de consumo terminado, sino más bien una herramienta para usuarios con conocimientos técnicos, incluidos ingenieros, desarrolladores, investigadores y estudiantes.

Uso inadecuado previsible
El producto no es adecuado para uso industrial o aplicaciones relevantes para la seguridad. No se permite el uso del producto en dispositivos médicos o para fines de aviación y viajes espaciales.

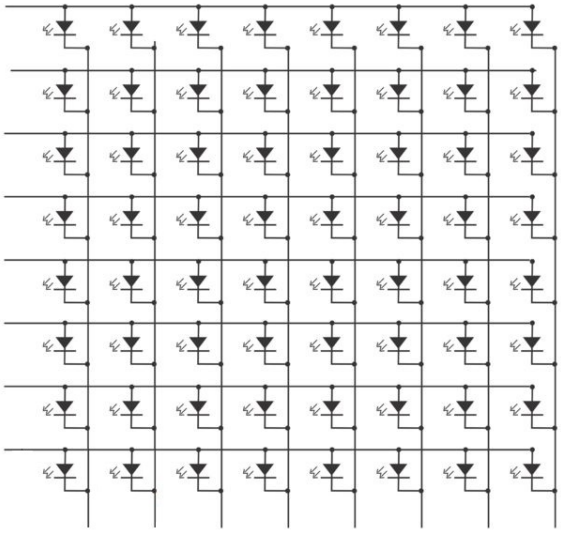
desecho
¡No lo deseche con la basura doméstica! Su producto es acorde al europeo. Directiva sobre residuos de aparatos eléctricos y electrónicos que deben eliminarse de forma respetuosa con el medio ambiente. Las valiosas materias primas contenidas en ellos se pueden reciclar. convertirse en. La aplicación de esta directiva contribuye a la protección del medio ambiente y la salud. Utilice el punto de recogida habilitado por su municipio para devolver y Reciclaje de aparatos eléctricos y electrónicos viejos. N.º registro RAEE: DE 62624346

descarga electrostática
La pantalla es sensible a las descargas electrostáticas (ESD), que pueden dañar o destruir los componentes electrónicos. Tenga en cuenta las siguientes instrucciones de seguridad para evitar riesgos de ESD: Atención: Las cargas electrostáticas en su cuerpo pueden dañar la pantalla. Nota: Conéctese a tierra usando una muñequera antiestática conectada a una superficie con conexión a tierra o tocando una superficie metálica con conexión a tierra antes de manipular la pantalla. Atención: Utilice bolsas y tapetes antiestáticos para proteger la pantalla. Nota: Coloque la pantalla sobre una alfombra de trabajo antiestática y guárdela en bolsas antiestáticas cuando no esté en uso. Nota: Un lugar de trabajo limpio y conectado a tierra minimiza el riesgo de ESD. Acción: Mantenga su lugar de trabajo limpio y libre de materiales que puedan generar cargas electrostáticas. Asegúrese de que todas las superficies utilizadas estén conectadas a tierra. Atención: Un lugar de trabajo limpio y conectado a tierra minimiza el riesgo de ESD. Nota: Mantenga su lugar de trabajo limpio y libre de materiales que puedan generar cargas electrostáticas. Asegúrese de que todas las superficies utilizadas estén conectadas a tierra.

instrucciones de seguridad
Aunque la pantalla cumple con los requisitos de la Directiva RoHS (2011/65/UE) y no contiene sustancias peligrosas en cantidades superiores a los límites permitidos, aún pueden existir riesgos químicos residuales. Tenga en cuenta las siguientes instrucciones de seguridad: Atención: La parte posterior de la pantalla y la placa de circuito pueden liberar residuos químicos de fabricación o durante el funcionamiento. Nota: Use guantes protectores cuando manipule o instale la pantalla durante

mucho tiempo para evitar irritación de la piel. Precaución: Los componentes electrónicos pueden emitir pequeñas cantidades de compuestos orgánicos volátiles (COV), especialmente si la pantalla es nueva. Nota: asegúrese de trabajar en un área bien ventilada para minimizar la concentración de vapores en el aire. Precaución: No utilice productos químicos ni disolventes agresivos para limpiar la pantalla, ya que pueden dañar la capa protectora o los componentes electrónicos. Nota: Utilice un paño de limpieza antiestático o un limpiador especial para componentes electrónicos para limpiar cuidadosamente la pantalla. Aunque la pantalla cumple con los requisitos de la Directiva RoHS (2011/65/UE) y no contiene sustancias peligrosas en cantidades superiores a los límites permitidos, aún pueden existir riesgos químicos residuales. Tenga en cuenta las siguientes instrucciones de seguridad: Atención: La parte posterior de la pantalla y la placa de circuito pueden liberar residuos químicos de fabricación o durante el funcionamiento. Nota: Use guantes protectores cuando manipule o instale la pantalla durante mucho tiempo para evitar irritación de la piel. Precaución: Los componentes electrónicos pueden emitir pequeñas cantidades de compuestos orgánicos volátiles (COV), especialmente si la pantalla es nueva. Nota: asegúrese de trabajar en un área bien ventilada para minimizar la concentración de vapores en el aire. Precaución: No utilice productos químicos ni disolventes agresivos para limpiar la pantalla, ya que pueden dañar la capa protectora o los componentes electrónicos. Nota: Utilice un paño de limpieza antiestático o un limpiador especial para componentes electrónicos para limpiar cuidadosamente la pantalla. La pantalla contiene componentes electrónicos sensibles y una capa superior. Un manejo inadecuado o una presión excesiva pueden causar daños a la pantalla o lesiones. Observe las siguientes instrucciones de seguridad para evitar riesgos mecánicos: Atención: La tapa del display es frágil y puede romperse si se manipula incorrectamente. Nota: Evite aplicar una presión fuerte o doblar la pantalla. Manipule la pantalla con cuidado y sólo por la placa de circuito para evitar roturas. Precaución: Las caídas o los impactos pueden agrietar la superficie de la pantalla y dañar los componentes electrónicos de la parte posterior. Nota: Evite dejar caer la pantalla y protéjala de impactos. Utilice una superficie suave cuando trabaje para evitar rayones. Atención: Si la pantalla se rompe, los trozos de vidrio afilados pueden provocar lesiones. Nota: Si la pantalla se rompe, manipule los fragmentos con cuidado y use guantes protectores para evitar cortes. Deseche las piezas de vidrio de forma segura. Nota: Una fijación incorrecta puede provocar tensión mecánica y rotura de la pantalla. Acción: Coloque la pantalla de forma segura y sin presión excesiva. Utilice soportes o carcasas adecuados para montar la pantalla de forma estable. Precaución: Los métodos de limpieza inadecuados pueden rayar o dañar la superficie. Nota: Utilice únicamente paños suaves y antiestáticos para limpiar la pantalla. Evite agentes de limpieza agresivos y fricciones fuertes. La pantalla funciona con voltajes y corrientes eléctricas que, si se usan incorrectamente, pueden provocar descargas eléctricas, cortocircuitos o incendios. Tenga en cuenta las siguientes instrucciones de seguridad: Atención: Utilice el producto únicamente con los voltajes especificados. Nota: Los límites de rendimiento del producto se pueden encontrar en la hoja de datos asociada. Nota: Las fuentes de voltaje inadecuadas pueden dañar la pantalla o provocar situaciones peligrosas. Acción: Utilice únicamente fuentes de alimentación o baterías probadas y adecuadas para alimentar sus circuitos. Asegúrese de que la fuente de voltaje cumpla con los requisitos de la pantalla. Precaución: Evite cortocircuitos entre los conectores y componentes del producto. Nota: Asegúrese de que ningún objeto conductor toque o puentee la placa de circuito. Utilice herramientas aisladas y preste atención a la disposición de las conexiones. Precaución: No realice ningún trabajo en el producto cuando esté conectado a una fuente de alimentación. Nota: Desconecte el producto de la alimentación antes de realizar cambios en el circuito o conectar o quitar componentes. Nota: busque señales de daños eléctricos, como humo, olores inusuales o decoloración. Acción: Si ocurren tales signos, apague la alimentación inmediatamente e inspeccione minuciosamente el circuito para detectar errores. La pantalla puede generar calor durante el funcionamiento, lo que podría provocar sobrecalentamiento, quemaduras o incendios si se manipula incorrectamente. Tenga en cuenta las siguientes instrucciones de seguridad: Atención: Algunos componentes de la pantalla pueden calentarse durante el funcionamiento o en caso de error. Medida: Después de apagarla, deje que la pantalla se enfríe lo suficiente antes de tocar directamente los componentes individuales de la parte posterior. Evite el contacto directo con componentes calientes. Precaución: La sobrecarga puede provocar un calentamiento excesivo de los componentes electrónicos. Nota: Asegúrese de que la fuente de alimentación y voltaje cumpla con las especificaciones de la pantalla y no cause sobrecarga.

La pantalla de matriz LED de *4x64* es una pantalla con matrices de LED juntas una por una en una *"matrix"*. La pantalla LED *4x64* está compuesta por cuatro pantallas más pequeñas de *8x8* que se denominan *"segments"*. Se denominan de *8x8* porque hay *8* LEDs en una fila, y hay *8* filas en un segmento que crea una matriz de *64* LEDs (*64* píxeles). Cada LED necesita de dos cables para trabajar (uno para energía y otro para tierra). Para evitar tener que cablear los *64* LEDs uno por uno, se conectan en una matriz de LEDs, como se observa en la siguiente imagen:



Colocar los LEDs en una matriz reduce el número de pines de salida a *16*, lo que sigue siendo demasiado, por esto se desarrollaron los controladores de la matriz de LED. En este caso, el controlador de matriz de LED que se utiliza es "*MAX7219*", que utiliza la interfaz SPI. La pantalla de matriz LED *4x64* tiene cuatro segmentos, un chip controlador por segmento. Los chips controladores están conectados en

serie en *"Daisy chain"*. Es posible acumular más, para lo cual hay que utilizar una fuente de alimentación externa.

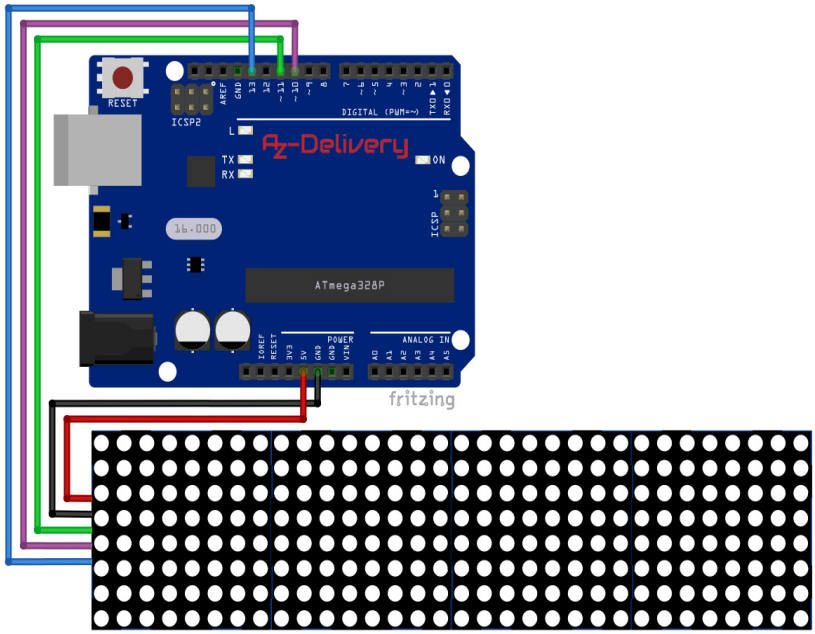
El chip MAX7219 utiliza la interfaz SPI para la comunicación con los microcontroladores. SPI es una interfaz de comunicación periférica serial síncrona, que se utiliza para la comunicación a corta distancia, principalmente en sistemas integrados. Lo síncrono permite que los datos se envíen y reciban dentro de un ciclo de reloj específico, y lo serial permite que los datos se envíen en serie, bit a bit.

Especificaciones:

- » Fuente de alimentación y rango de voltaje lógico: +3.3V a +5V DC
- » Color de LEDs: rojo
- » Control del brillo: digital y analógico
- » Dimensiones: 32x129mm [1.26x5.1in]
- » Interfaz SPI 10MHz
- » Control individual del segmento de LED
- » El Display se limpia al encenderse
- » Conduce Pantallas LED de Cátodo Común.

Conectando la pantalla con Atmega328P Board

Conectar el módulo con el tablero del microcontrolador como se observa en el siguiente diagrama de conexión:

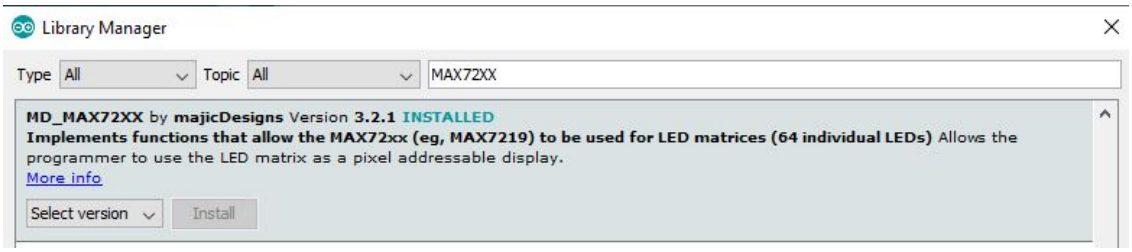


Pin del Módulo		**>**	**Pin de tablero**	
VCC		>	5V	**Cable rojo**
GND		>	GND	**Cable negro**
CS	(SS)	>	D10 pin	**Cable púrpura**
DATA	(MOSI)	>	D11 pin	**Cable verde**
CLK	(SCK)	>	D13 pin	**Cable azul**

Se debe asegurar de leer las designaciones de los pines antes de empezar a conectarlos, ya que hay diferentes versiones del módulo. Se han incluido algunos nombres alternativos de los pines dentro de los paréntesis.

Librería Arduino IDE

Para utilizar el módulo con tablero del microcontrolador, se recomienda descargar una librería externa para esto. Ir a: *Tools* > *Manage Libraries*, una nueva ventana aparecerá, escribir *"MAX72XX"* en el cuadro de búsqueda. La librería que se va a utilizar es *"MD_MAX72XX"* de *"majicDesigns",* como se observa en la siguiente imagen:



La instalación es muy simple, solo hacer clic en el botón *"Install"* que aparece al pasar el ratón por encima del resultado de la búsqueda.

Hay muchas versiones de estas pantallas. Para detectar cuál está utilizando, puede ejecutar el sketch de Hardware Mapper que incluye la librería. Para abrir el sketch, ir a:
File > *Examples* > *MD_MAX72XX* > *MD_MAX72xx_HW_Mapper,*
y cambiar la siguiente línea del código:

```
#define SERIAL_SPEED 57600
```

a:

```
#define SERIAL_SPEED 9600
```

Cargar el sketch al tablero del microcontrolador y abrir el Serial Monitor (*Tools* > *Serial Monitor*). Después de eso, seguir las instrucciones de

tres pasos que tablero del microcontrolador envía al Serial Monitor, como se observa en la imagen de la siguiente página:

```
[MD_MAX72xx Hardware mapping utility]

INTRODUCTION
------------
How the LED matrix is wired is important for the MD_MAX72xx library as different
LED modules are wired differently. The library can accommodate these, but it
needs to know what transformations need to be carried out to map your board to the
standard coordinate system. This utility shows you how the matrix is wired so that
you can set the correct *_HW module type for your application.

The standard functions in the library expect that:
o COLUMNS are addressed through the SEGMENT selection lines, and
o ROWS are addressed through the DIGIT selection lines.

The DISPLAY always has its origin in the top right corner of a display:
o LED matrix module numbers increase from right to left,
o Column numbers (ie, the hardware segment numbers) increase from right to left (0..7), and
o Row numbers (ie, the hardware digit numbers) increase down (0..7).

There are three hardware setting that describe your hardware configuration:
- HW_DIG_ROWS - HardWare DIGits are ROWS. This will be 1 if the digits map to the rows
                of the matrix, 0 otherwise
- HW_REV_COLS - HardWare REVerse COLumnS. The normal column coordinates orientation
                is col 0 on the right side of the display. This will be 1 if reversed.
                (ie, hardware 0 is on the left).
- HW_REV_ROWS - HardWare REVerse ROWS. The normal row coordinates orientation is row
                0 at top of the display. This will be 1 if reversed (ie, row 0
                is at the bottom).

These individual setting then determine the model type of the hardware you are using.

INSTRUCTIONS
------------
1. Wire up one matrix only, or cover up the other modules, to avoid confusion.
2. Enter the answers to the question in the edit field at the top of Serial Monitor.

=====================================================

STEP 1 - DIGITS MAPPING (rows)
------------------------------
In this step you will see a line moving across the LED matrix.
You need to observe whether the bar is scanning ROWS or COLUMNS,
and the direction it is moving.
>> Enter Y when you are ready to start: Y
Dig-0-1-2-3-4-5-6-7
>> Enter Y if you saw ROWS animated, N if you saw COLUMNS animated: Y
>> Enter Y if you saw the line moving BOTTOM to TOP, or enter N otherwise: N

STEP 2 - SEGMENT MAPPING (columns)
----------------------------------
In this step you will see a dot moving along one edge of the LED matrix.
You need to observe the direction it is moving.
>> Enter Y when you are ready to start: Y
Seg-G-F-E-D-C-B-A-DP
>> Enter Y if you saw the LED moving LEFT to RIGHT, or enter N otherwise: N

STEP 3 - RESULTS
----------------
Your responses produce these hardware parameters

HW_DIG_ROWS     1
HW_REV_COLS     0
HW_REV_ROWS     0

Your hardware matches the setting for FC-16 modules. Please set FC16_HW.
```

Para obtener el resultado correcto, se tiene que hacer la orientación de la pantalla como en el diagrama de conexión. En el lado derecho de la pantalla están los pines de entrada (donde se conectan los cables).

El mapa de hardware resultante es “*FC16_HW*”, por lo que al utilizar el módulo de pantalla matriz de 64 LEDs de AZ-Delivery se necesita ajustar el macro "*HARDWARE_TYPE*" a este valor de resultado.

Código sketch:

```
#include <MD_MAX72xx.h >

#define PRINT(s, x) { Serial.print(F(s)); Serial.print(x); }
#define PRINTS(x) Serial.print(F(x))
#define PRINTD(x) Serial.println(x, DEC)
#define PRINT(s, x)
#define PRINTS(x)
#define PRINTD(x)

#define HARDWARE_TYPE MD_MAX72XX::FC16_HW

#define MAX_DEVICES  4

#define CLK_PIN   13  // or SCK
#define DATA_PIN  11  // or MOSI
#define CS_PIN    10  // or SS

MD_MAX72XX mx = MD_MAX72XX(HARDWARE_TYPE, CS_PIN, MAX_DEVICES);

#define  DELAYTIME  100  // in milliseconds
```

```
void scrollText(char *p) {
  uint8_t charWidth;
  uint8_t cBuf[8];  // this should be ok for all built-in fonts
  PRINTS("\nScrolling text");
  mx.clear();
  while (*p != '\0') {
    charWidth = mx.getChar(*p++, sizeof(cBuf) / sizeof(cBuf[0]), cBuf);
    // allow space between characters
    for (uint8_t i=0; i<=charWidth; i++) {
      mx.transform(MD_MAX72XX::TSL);
      if (i < charWidth) {
        mx.setColumn(0, cBuf[i]);
      }
      delay(DELAYTIME);
    }
  }
}

void rows() {
  mx.clear();
  for (uint8_t row=0; row<ROW_SIZE; row++) {
    mx.setRow(row, 0xff); delay(2*DELAYTIME);
    mx.setRow(row, 0x00);
  }
}

void columns() {
  mx.clear();
  for (uint8_t col=0; col<mx.getColumnCount(); col++) {
    mx.setColumn(col, 0xff); delay(DELAYTIME/MAX_DEVICES);
    mx.setColumn(col, 0x00);
  }
}
```

```
void stripe() {
  const uint16_t maxCol = MAX_DEVICES*ROW_SIZE;
  const uint8_t stripeWidth = 10;
  mx.clear();
  for (uint16_t col=0; col<maxCol + ROW_SIZE + stripeWidth; col++) {
    for(uint8_t row=0; row < ROW_SIZE; row++) {
      mx.setPoint(row, col-row, true);
      mx.setPoint(row, col-row - stripeWidth, false);
    }
    delay(DELAYTIME);
  }
}

void spiral() {
  int  rmin = 0, rmax = ROW_SIZE-1;
  int  cmin = 0, cmax = (COL_SIZE*MAX_DEVICES)-1;
  mx.clear();
  while ((rmax > rmin) && (cmax > cmin)) {
    for (int i=cmin; i<=cmax; i++) { // do row
      mx.setPoint(rmin, i, true); delay(DELAYTIME/MAX_DEVICES);
    }
    rmin++;
    for (uint8_t i=rmin; i<=rmax; i++) { // do column
      mx.setPoint(i, cmax, true); delay(DELAYTIME/MAX_DEVICES);
    }
    cmax--;
    for (int i=cmax; i>=cmin; i--) { // do row
      mx.setPoint(rmax, i, true); delay(DELAYTIME/MAX_DEVICES);
    }
    rmax--;
    for (uint8_t i=rmax; i>=rmin; i--) { // do column
      mx.setPoint(i, cmin, true); delay(DELAYTIME/MAX_DEVICES);
    }
    cmin++;
  }
}
```

```
void bounce() {
  const int minC = 0;
  const int maxC = mx.getColumnCount()-1;
  const int minR = 0;
  const int maxR = ROW_SIZE-1;
  int  nCounter = 0;
  int  r = 0, c = 2;
  int8_t dR = 1, dC = 1;  // delta row and column
  mx.clear();
  while (nCounter++ < 200) {
    mx.setPoint(r, c, false);
    r += dR; c += dC;
    mx.setPoint(r, c, true);
    delay(DELAYTIME/2);
    if ((r == minR) || (r == maxR)) {
      dR = -dR;
    }
    if ((c == minC) || (c == maxC)) {
      dC = -dC;
    }
  }
}

void intensity() {
  uint8_t row;
  mx.clear();
  for (int8_t i=0; i<=MAX_INTENSITY; i++) {
    mx.control(MD_MAX72XX::INTENSITY, i);
    mx.setRow(0, 0xff); delay(DELAYTIME*10);
  }
  mx.control(MD_MAX72XX::INTENSITY, 8);
}
```

```
void transformation() {
  uint8_t arrow[COL_SIZE] = {
    0b00001000,
    0b00011100,
    0b00111110,
    0b01111111,
    0b00011100,
    0b00011100,
    0b00111110,
    0b00000000 };
  MD_MAX72XX::transformType_t  t[] = {
    MD_MAX72XX::TSL, MD_MAX72XX::TSL, MD_MAX72XX::TSL, MD_MAX72XX::TSL,
    MD_MAX72XX::TSL, MD_MAX72XX::TSL, MD_MAX72XX::TSL, MD_MAX72XX::TSL,
    MD_MAX72XX::TSL, MD_MAX72XX::TSL, MD_MAX72XX::TSL, MD_MAX72XX::TSL,
    MD_MAX72XX::TSL, MD_MAX72XX::TSL, MD_MAX72XX::TSL, MD_MAX72XX::TSL,
    MD_MAX72XX::TFLR,
    MD_MAX72XX::TSR, MD_MAX72XX::TSR, MD_MAX72XX::TSR, MD_MAX72XX::TSR,
    MD_MAX72XX::TSR, MD_MAX72XX::TSR, MD_MAX72XX::TSR, MD_MAX72XX::TSR,
    MD_MAX72XX::TSR, MD_MAX72XX::TSR, MD_MAX72XX::TSR, MD_MAX72XX::TSR,
    MD_MAX72XX::TSR, MD_MAX72XX::TSR, MD_MAX72XX::TSR, MD_MAX72XX::TSR,
    MD_MAX72XX::TRC,
    MD_MAX72XX::TSD, MD_MAX72XX::TSD, MD_MAX72XX::TSD, MD_MAX72XX::TSD,
    MD_MAX72XX::TSD, MD_MAX72XX::TSD, MD_MAX72XX::TSD, MD_MAX72XX::TSD,
    MD_MAX72XX::TFUD,
    MD_MAX72XX::TSU, MD_MAX72XX::TSU, MD_MAX72XX::TSU, MD_MAX72XX::TSU,
    MD_MAX72XX::TSU, MD_MAX72XX::TSU, MD_MAX72XX::TSU, MD_MAX72XX::TSU,
    MD_MAX72XX::TINV,
    MD_MAX72XX::TRC, MD_MAX72XX::TRC, MD_MAX72XX::TRC, MD_MAX72XX::TRC,
    MD_MAX72XX::TINV };
  mx.clear();
  mx.control(MD_MAX72XX::UPDATE, MD_MAX72XX::OFF);
  for (uint8_t j=0; j<mx.getDeviceCount(); j++) {
    mx.setBuffer(((j+1)*COL_SIZE)-1, COL_SIZE, arrow);
  }
  mx.control(MD_MAX72XX::UPDATE, MD_MAX72XX::ON);
  delay(DELAYTIME);
  mx.control(MD_MAX72XX::WRAPAROUND, MD_MAX72XX::ON);
```

```
// one tab
  for (uint8_t i=0; i<(sizeof(t)/sizeof(t[0])); i++) {
    mx.transform(t[i]); delay(DELAYTIME*4);
  }
  mx.control(MD_MAX72XX::WRAPAROUND, MD_MAX72XX::OFF);
}

void showCharset() {
  mx.clear();
  mx.update(MD_MAX72XX::OFF);
  for (uint16_t i=0; i<256; i++) {
    mx.clear(0);
    mx.setChar(COL_SIZE-1, i);
    if (MAX_DEVICES >= 3) {
      char hex[3];
      sprintf(hex, "%02X", i);
      mx.clear(1); mx.setChar((2*COL_SIZE)-1, hex[1]);
      mx.clear(2); mx.setChar((3*COL_SIZE)-1, hex[0]);
    }
    mx.update();
    delay(DELAYTIME*2);
  }
  mx.update(MD_MAX72XX::ON);
}
void setup() { mx.begin(); }
void loop() {
  scrollText("Graphics");
  rows();
  columns();
  stripe();
  bounce();
  spiral();
  intensity();
  transformation();
  showCharset();
  delay(3000);
}
```

El sketch comienza con la inclusión de la librería "*MD_MAX72XX.h*", y luego continua con varias definiciones de macros. Un ejemplo especial del uso de uno de los macros es `Serial.print(F(x))`. `F(x)` es un macro que almacena una cadena "*x*" en la memoria flash, no SRAM. La memoria flash es una memoria de los microcontroladores donde se almacenan los programas, y la memoria SRAM es un tipo de memoria RAM. Se disminuye el límite de memoria para nuestros programas cuando se utiliza el macro `F(x)`, lo cual incrementa la memoria SRAM (muy importante para ejecutar los programas en los microcontroladores). Si no utiliza el macro `F(x)`, lo más probable es que su programa no funcione como está diseñado. Esto se debe a que, en el sketch, hay demasiadas cadenas, que en el lenguaje de programación *C* son arreglos de caracteres y que agotan la memoria SRAM fácilmente.

Después, se especifica que pantalla se está utilizando, estableciendo el macro `HARDWARE_TYPE` a `FC16_HW` (el valor resultante que se obtuvo en el capítulo anterior).

Después de esto, se especifican cuantos segmentos tiene la pantalla de matriz de LED estableciendo el macro `MAX_DEVICES` a `4`. Si conecta múltiples pantallas de matriz de LED en la cadena Daisy, tiene que cambiar este valor en consecuencia. Hay tres macros más que definen los pines de la interfaz SPI.

A continuación, se crea el objeto "*mx*", el objeto de la pantalla que se utiliza en todo el sketch.

Después de todo esto, se crean varias funciones.

La primera función es *scrollingText()*, que se utiliza para el desplazamiento de texto en la pantalla. La función *scrollingText()* utiliza la función *setColumn()* de la librería "*MD_MAX72XX.h*". Esta función enciende o apaga un conjunto de columnas de LEDs en la pantalla. El resto de la función es un algoritmo para el efecto de desplazamiento. Se toma la cadena "*p*" de la memoria flash, y se la convierte en una matriz de números enteros sin signo que representa los LEDs en la matriz de columnas. Luego se muestra esa matriz en la pantalla. Al final de la función se especifica el tiempo de retraso, que es la velocidad de desplazamiento del texto.

La segunda función es *rows()*, que utiliza la función *setRow()*de la librería "*MD_MAX72XX.h*". La función *setRow()* acepta dos argumentos. El primer argumento es para conocer qué fila se utilizará, donde *0* representa que la primera fila comienza desde arriba. Y el segundo argumento es el número hexadecimal, que representa el valor de *8* bits para la matriz de filas de píxeles (*8* píxeles en una fila). En este caso se utiliza *0xFF* que enciende todos los píxeles de una fila específica.

La tercera función es *columns()*, que es la misma que *rows(),* pero en este caso se activan los conjuntos de columnas de píxeles. La función utiliza la función *setColumn()* de la librería "*MD_MAX72XX.h*".

La función *setColumn()* es la misma que *setRow()*, pero solo para columnas.

La cuarta función es *stripe()*, que muestra una franja diagonal de desplazamiento (*4* pixeles de ancho), en toda la pantalla. La función utiliza la función *setPoint()* de la librería "*MD_MAX72XX.h*". La función *setPoint()* se utiliza para encender un LED específico en la pantalla, que acepta tres argumentos. El primer y segundo argumento son las posiciones *X* e *Y* del LED (la esquina superior izquierda de la pantalla es *X=0* e *Y=0*, la esquina inferior derecha de la pantalla es *X=31* e *Y=7*). El tercer argumento es un valor booleano que representa el estado del LED, encendido = *true*, y apagado = *false*. Al final de la función se especifica el tiempo de retraso, que se utiliza para cambiar la velocidad de la banda de desplazamiento.

La quinta función es *spiral()*, que muestra una línea como de serpiente que gira todos los píxeles en la pantalla en un efecto de movimiento espiral. La función utiliza la función *setPoint()* de la librería "*MD_MAX72XX.h*". El resto de la función es el algoritmo de encender LED por LED hasta que todos los LEDs se enciendan.

La sexta función es *bounce()*, que muestra el punto de movimiento que rebota en los bordes de la pantalla de la matriz de LED. La función utiliza pocas funciones que ya se han explicado, y una función adicional *getColumnCont()* que devuelve el número de columnas de la pantalla de matriz de LED. El resto de la función es el algoritmo para el movimiento de pixel.

La séptima función es *`intensity()`* que se utiliza para encender sólo la primera fila de los LEDs en la pantalla. La función utiliza la función *`control()`* de la librería "*MD_MAX72XX.h*". La función *`control()`* acepta dos argumentos, el primero es definir cuál propiedad del objeto "*mx*" se va a cambiar, donde se almacena el valor "*MD_MAX72XX::INTENSITY*". Así que se va a cambiar la intensidad del brillo. El segundo argumento es un número que representa el nivel de intensidad y su valor está en el rango de *0* a *15*, o *16* niveles diferentes (valor por defecto es *8*).

La octava función es *`transformation()`*, que se utiliza para animar la imagen en mapa de bits en la pantalla. Los datos de la imagen en mapa de bits están almacenados en la matriz de números enteros sin signo. Esta matriz contiene *8* elementos, que contiene números binarios. Estos números binarios representan las filas de un segmento de la pantalla, donde el valor binario *0* representa el LED apagado, y el valor *1* representa el LED encendido. Se define una matriz más denominada "*t*", que contiene enumeraciones utilizadas para la animación. El formato de estas enumeraciones es "*MD_MAX72XX::{enumeration}*". Las enumeraciones utilizadas son:

TSL – Mover a la izquierda para un pixel;
TSR – Mover a la derecha para un pixel
TSU – Subir para un pixel;
TSD – Bajar para un pixel
TFLR – Voltear de izquierda a derecha;
TFUD – Voltear de arriba a abajo;

TRC – Girar en sentido horario 90 grados

TINV - Invertir pixeles (todos los píxeles invertidos, los que estaban encendidos, se apagan, y viceversa).

A continuación, se utiliza la función *`setBuffer()`* para enviar los datos para su visualización al chip MAX7219. Esta función acepta tres argumentos, el primero y el segundo son las posiciones *X* e *Y* de la imagen, y el tercero son los datos de la imagen en mapa de bits.

Para mostrar los datos se utiliza la siguiente línea del código:

```
mx.control(MD_MAX72XX::UPDATE, MD_MAX72XX::ON);
```

Después, se animan los datos que se muestran. Esto se realiza con la propiedad *`WRAPAROUND`* de la función *`control()`* y la función *`transform()`*. Se utiliza *`for`* loop a loop a través de todas las enumeraciones en la matriz *`"t"`*.

La última función es *`showCharsset()`* que se utiliza para mostrar todos los caracteres UNICODE que se pueden utilizar en la pantalla de matriz de LED. Aquí se utiliza *built* en la función *`update()`* que actúa como la función *`control()`* con la propiedad *`UPDATE`*. La función *`update()`* acepta un argumento, cuyo valor puede ser uno de los dos: *`MD_MAX72XX::OFF`* y *`MD_MAX72XX::ON`*

Primero se fija el valor del argumento en *`MD_MAX72XX::OFF`*, luego se realizan los cambios, cambiar el carácter que se mostrará, y luego cambiar el valor del argumento a *`MD_MAX72XX::ON`*. Se utiliza *`for`* loop a loop a través de todos los caracteres UNICODE.

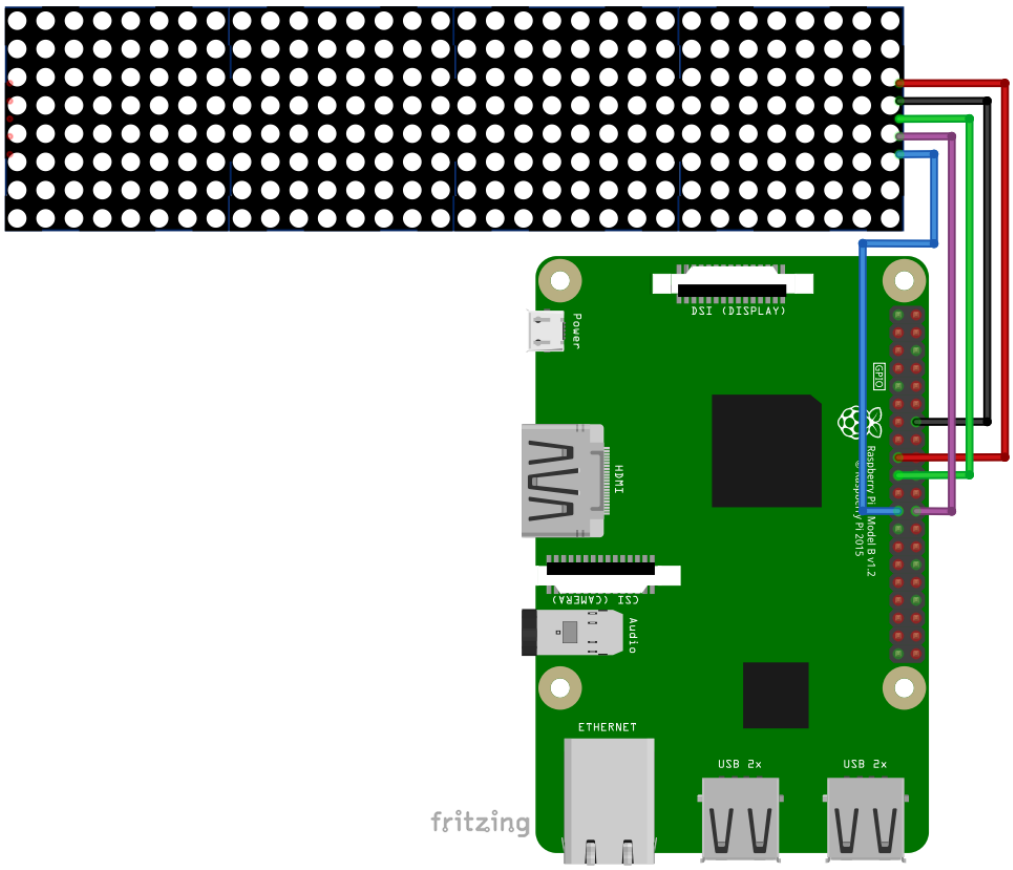
En la función *setup()* se inicia el objeto de la pantalla con la siguiente línea del código: `mx.begin()`

En la función *loop()* se nombran todas las funciones que se han creado, una por una.

Si desea conocer más información sobre estas funciones de la librería "*MD_MAX72XX.h*", puede ir al sitio de documentación oficial de la librería:
https://majicdesigns.github.io/MD_MAX72XX/class_m_d___m_a_x72_x_x.html

Conectando la pantalla con Raspberry Pi

Conectar el módulo con el Raspberry Pi como se observa en el siguiente diagrama de conexión:



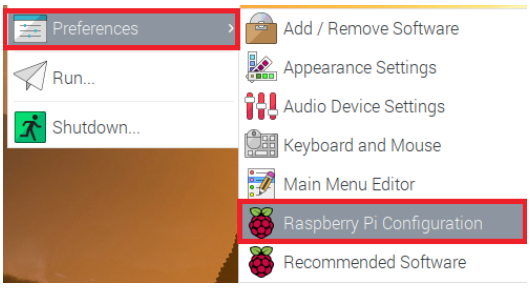
Pin del Módulo		>	Pin de Raspberry Pi	
VCC		>	5V	**Cable rojo**
GND		>	GND	**Cable negro**
DATA	(MOSI)	>	GPIO10 [pin 19]	**Cable azul**
CLK	(SCK)	>	GPIO11 [pin 23]	**Cable café**
CS	(SS)	>	GPIO8 [pin 24]	**Cable verde**

Habilitando la Interfaz SPI

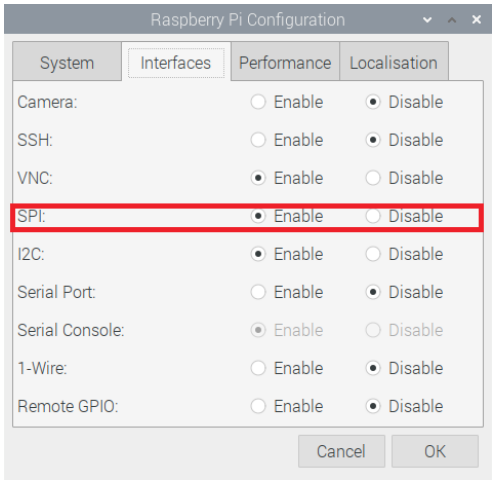
Para utilizar el módulo con Raspberry Pi, primero se tiene que habilitar la Interfaz SPI en el Raspbian. Para esto, ir a:

Preferences > *Raspberry Pi Configuration*

como se observa en la siguiente imagen:



A continuación, abrir la pestaña "*Interfaces*", buscar el botón de radio "*SPI*" y habilitarlo, como se observa en la siguiente imagen:



Se le requerirá que reinicie el sistema. Reiniciarlo.

Librerías y herramientas para Python

Para utilizar el módulo con el Raspberry Pi, se recomienda descargar una librería externa. La librería que se va a utilizar es "*luma.led_matrix*". Para utilizar la librería, primero se necesita instalar sus dependencias. Para esto, abrir el terminal y los siguientes comandos:

```
sudo apt install build-essential python-dev
sudo apt install python-pip libfreetype6-dev
sudo apt install libjpeg-dev libopenjp2-7 libtiff5
```

Se tienen que actualizar *"pip"* y "*setuptools*", y para hacerlo, ejecutar el siguiente comando:

```
sudo -H pip install --upgrade --ignore-installed pip setuptools
```

Después, instale la última versión de la librería "*luma.led_matrix*" ejecutando el siguiente comando:

```
sudo -H pip install --upgrade luma.led_matrix
```

Script Python:

```
import re
import time
import argparse
from luma.led_matrix.device import max7219
from luma.core.interface.serial import spi, noop
from luma.core.render import canvas
from luma.core.virtual import viewport
from luma.core.legacy import text, show_message
from luma.core.legacy.font import proportional, CP437_FONT, TINY_FONT,
SINCLAIR_FONT, LCD_FONT

def demo(n, block_orientation, rotate, inreverse):

    serial = spi(port=0, device=0, gpio=noop())
    device = max7219(serial, cascaded=n or 1,
                     block_orientation=block_orientation, rotate=rotate or 0,
                     blocks_arranged_in_reverse_order=inreverse)
    print('Created device')
    msg = 'MAX7219 LED Matrix Demo'
    print(msg)
    show_message(device, msg, fill='white',font=proportional(CP437_FONT),
                                              scroll_delay=0)
    time.sleep(1)
    print('Vertical scrolling')
    words = ['Victor', 'Echo', 'Romeo', 'Tango', 'India', 'Charlie',
             'Alpha', 'Lima', ' ', 'Sierra', 'Charlie', 'Romeo', 'Oscar',
             'Lima', 'Lima', 'India', 'November', 'Golf', ' ']

    virtual = viewport(device, width=device.width, height=len(words) * 8)
    with canvas(virtual) as draw:
        for i, word in enumerate(words):
           text(draw, (0, i * 8), word,     fill='white',
                                            font=proportional(CP437_FONT))
```

```
# one tab
    for i in range(virtual.height - device.height):
        virtual.set_position((0, i))
        time.sleep(0.05)

    msg = 'Brightness'
    print(msg)
    show_message(device, msg, fill='white')
    time.sleep(1)

    for _ in range(5):
        for intensity in range(16):
            device.contrast(intensity * 16)
            time.sleep(1)

    device.contrast(0x80)
    time.sleep(1)
    msg = 'Alternative font!'
    print(msg)
    show_message(device, msg, fill='white', font=SINCLAIR_FONT)
    time.sleep(1)
    msg = 'Proportional font - characters are squeezed together!'
    print(msg)
    show_message(device,msg,fill='white',font=proportional(SINCLAIR_FONT))
    time.sleep(1)
    msg = 'Tiny is, I believe, the smallest possible font \
    (in pixel size). It stands at a lofty four pixels \
    tall (five if you count descenders), yet it still \
    contains all the printable ASCII characters.'
    msg = re.sub(' +', ' ', msg)
    print(msg)
    show_message(device, msg, fill='white', font=proportional(TINY_FONT))
    time.sleep(1)
```

```
# one tab
    msg = 'CP437 Characters'
    print(msg)
    show_message(device, msg)
    time.sleep(1)
    for x in range(256):
        with canvas(device) as draw:
            text(draw, (0, 0), chr(x), fill='white')
            time.sleep(0.3)

if __name__ == '__main__':
    parser = argparse.ArgumentParser(description='matrix_demo arguments',
                    formatter_class=argparse.ArgumentDefaultsHelpFormatter)
    parser.add_argument('--cascaded', '-n', type=int, default=1,
                    help='Number of cascaded MAX7219 LED matrices')
    parser.add_argument('--block-orientation', type=int, default=0,
                    choices=[0, 90, -90],
                    help='Corrects block orientation when wired vertically')
    parser.add_argument('--rotate',type=int,default=0,choices=[0, 1, 2, 3],
                    help='Rotate display 0=0°, 1=90°, 2=180°, 3=270°')
    parser.add_argument('--reverse-order', type=bool, default=False,
                    help='Set to true if blocks are in reverse order')
    args = parser.parse_args()
    try:
        while True:
        demo(args.cascaded,args.block_orientation,
                                            args.rotate,args.reverse_order)
        time.sleep(1)

    except KeyboardInterrupt:
        print('Script end!')
```

El código se basa en el código del siguiente link:

https://github.com/rm-hull/luma.led_matrix/blob/master/examples/matrix_demo.py

El script comienza con la importación de las librerías necesarias. A continuación, se crea la función *demo()* que se utiliza para ejecutar todas las funciones que se incluyen con la librería "*luma.led_matrix*". La función acepta cuatro argumentos y no devuelve ningún valor. Los argumentos son:

» *n* – que representa el número de segmentos de matriz de LED en cascada

» *block_orientation* – que se utiliza para corregir la orientación del contenido en la pantalla; sus valores son: *0*, *90* y *-90*

» *rotate* – que gira el contenido en la pantalla; sus valores son: *0*, *1*, *2* y *3*, que representan los ángulos *0°*, *90°*, *180°* y *270°*, respectivamente

» *reverse* – es un valor booleano y cuando se establece como *true*, el orden de los segmentos está en reversa.

Dentro de la función, se crea un objeto de comunicación SPI y un objeto de dispositivo.

Para mostrar un mensaje de desplazamiento (horizontal) en la pantalla, se utiliza la función *show_message()* que se incluye con la librería "*luma.led_matrix*". Esta función acepta cinco argumentos y no devuelve ningún valor. Los tres últimos argumentos son opcionales. El primer argumento es un objeto de dispositivo, el segundo es el mensaje, el tercero es el argumento *fill*, el cuarto es el argumento *font* y el quinto es el argumento *scroll_delay*. El valor del argumento *fill* es el nombre del color: "*white*", "*blue*", "*red*", etc. Si utiliza cualquier otro

color que no sea “*black*” el mensaje se mostrará en rojo, debido a que el color rojo es el color de los LEDs de la pantalla.

Si utiliza “*black*” en el argumento *`fill`*, no se mostrará nada.

El argumento *`font`* se utiliza para establecer la fuente específica y el argumento *`scroll_delay`* se utiliza para cambiar la velocidad de desplazamiento del texto.

Para poder mostrar el mensaje de desplazamiento vertical tiene que utilizar las funciones de la librería *`viewport`*. Estas funciones se utilizan para crear un objeto, o un lienzo, con el contenido del mensaje, después el contenido de este lienzo se mueve sobre la pantalla a través del *`for`* loop generando un efecto de desplazamiento vertical.

Para cambiar el nivel de brillo, se utiliza la función *`contrast()`*, como en la siguiente línea del código: `device.contrast(number)`
donde *`number`* es el nivel de brillo. Hay *16* valores diferentes de nivel de brillo, pero este valor tiene que ser un número dividido que al dividirlo por *16* representa un número entero, porque los registros internos del chip controlador del MAX7219 se establecen de esta manera. .

Hay dos maneras de utilizar las fuentes. La primera es poner el nombre de la fuente al argumento de la fuente de la función *`show_message()`*. La segunda es poner la función *`proportional()`*al mismo argumento. Esta función acepta un argumento, cuyo valor es el nombre de la fuente.

La diferencia es que con la función *proportional()* la fuente se muestra en proporción al tamaño de la pantalla.

Para mostrar un carácter específico en un segmento de la pantalla se utiliza la función *text()*. La función acepta cuatro argumentos y no devuelve ningún valor. El primer argumento es un objeto *draw*. El objeto tiene una parte de lienzo, o una pieza de datos del objeto de lienzo. Se explica el objeto *canvas* para el desplazamiento vertical del mensaje. El segundo argumento es una lista de dos elementos, las posiciones *X* e *Y* de la esquina superior izquierda del carácter. El tercer argumento es el carácter que se mostrará. El cuarto argumento es el argumento *fill*.

Guardar el script con el nombre "*ledMatrix.py*". Para ejecutarlo, abrir el terminal en el directorio donde guardó el script, y ejecutar el comando:
python3 ledMatrix.py

Existe la opción de enviar cuatro argumentos al script cuando se lo ejecuta. Estos argumentos son opcionales. Esto es posible gracias a la librería utilizada que se incluyó, denominada *argparse*. Hay cuatro argumentos opcionales que puede enviar al script:
» *cascaded* - es el número de segmentos de matriz de LED en cascada
» *block-orientation* - corrige la orientación de los segmentos
» *rotate* - gira el contenido de la pantalla
» *reverse-order* - cambia la dirección de desplazamiento del contenido

Estos argumentos se utilizan cuando se nombre a la función *demo()*. La función *demo()* se nombre en el bloque de loop infinito (*while True:*).

El bloque de loop infinito está en el bloque *try-except*. Se utiliza el bloque *try-except* para esperar por la interrupción del teclado. La interrupción del teclado se produce cuando se presiona *CTRL* + *C* en el teclado, y esto detiene el script.

Un ejemplo de utilizar todos los argumentos cuando se ejecuta el script:

```
python3 ledMatrix.py --cascaded 4 --block-orientation 90
                     --rotate 2   --reverse-order false
```

¡Lo ha logrado!

Ahora puede usar su módulo para varios proyectos.

Ahora es el momento de aprender y hacer sus propios proyectos. Puede hacerlo con la ayuda de muchos scripts de ejemplo y otros tutoriales, que puede encontrar fácilmente en Internet.

Si está buscando microelectrónica y accesorios de alta calidad, AZ-Delivery Vertriebs GmbH es la compañía adecuada donde podrá obtenerlos. Se le proporcionarán numerosos ejemplos de aplicación, guías completas de instalación, libros electrónicos, librerías y la asistencia de nuestros expertos técnicos.

https://az-delivery.de

¡Disfrute!

Impressum

https://az-delivery.de/pages/about-us