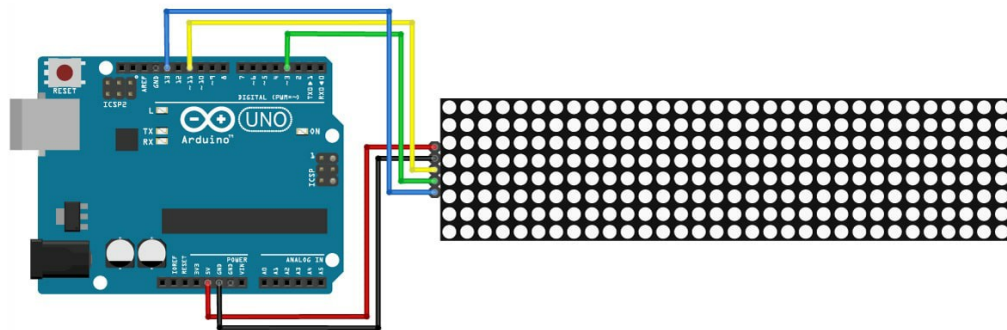


MAX7219 LED dot matrix display Arduino tutorial



25 Comments



fritzing

In this tutorial, you will learn how to control a [MAX7219 LED dot matrix display](#) with Arduino. I have included a wiring diagram and many example codes! The code in this tutorial can be used for 8×8, 8×32, and even larger displays.

For this tutorial, I will be using the **MD_Parola** in combination with the **MD_MAX72XX** Arduino library. These libraries make displaying scrolling text and other animations super easy. In the first part of this article, I will cover the basics of printing text on the display. Next, we will look at scrolling text and other text animations. Lastly, I will show you how to use text sprites.

If you would like to learn more about other types of displays, check out the articles below:

Recommended articles

- [TM1637 4 Digit 7 Segment Display Arduino Tutorial](#)
- [How to use a 16×2 character LCD with Arduino](#)
- [How to control a character I2C LCD with Arduino](#)

If you have any questions, please leave a comment below.

Supplies

Hardware components

	8×32 MAX7219 LED dot matrix display	× 1	Amazon
	8×8 MAX7219 LED dot matrix display (alternative)	× 1	Amazon
	Generic 8×8 MAX7219 LED dot matrix display (alternative)	× 1	Amazon



[Arduino Uno Rev3](#)

× 1 [Amazon](#)

[Jumper wires \(male to female\)](#)

× 4 [Amazon](#)

[USB cable type A/B](#)

× 1 [Amazon](#)

[Male headers](#)

~ 20 [Amazon](#)

[Jumpers](#)

~ 20 [Amazon](#)

Software



[Arduino IDE](#)

Makerguides.com is a participant in the Amazon Services LLC Associates Program, an affiliate advertising program designed to provide a means for sites to earn advertising fees by advertising and linking to products on Amazon.com.

About the MAX7219 LED driver

The MAX7219 LED driver can be used to control 7-segment displays up to 8 digits, bar-graph displays, or 64 individual LEDs. The driver communicates with the Arduino through SPI so you only need three wires to control the display.

Since the MAX7219 can control a maximum of 64 LEDs, the maximum size dot matrix display it can drive is 8×8 pixels. However, you can daisy chain multiple drivers and matrices together and easily control much larger displays like 8×32, 8×64, or even bigger. Still, you only need three wires to control all of the ICs so you need very few I/O pins of the Arduino.

Below you can find the specifications of a typical MAX7219 8×32 LED dot matrix display.

MAX7219 LED dot matrix display specifications

Operating voltage	5 V
Display driver	MAX7219 x 4
Brightness levels	16
Display dimensions	32 x 128 x 15 mm
Pixels	8×32, 3 mm
Cost	Check price

For more information, you can check out the datasheet:

[MAX7219 Datasheet](#)

Almost all of the displays I have used in the past used a 1088AS type 8×8 LED matrix. You can find a datasheet from one of the companies that makes them below:

[1088AS Datasheet](#)

How to connect the dot matrix display to the Arduino

The MAX7219 LED display driver communicates with the Arduino through SPI (Serial Peripheral Interface). To learn more about this data protocol, please see [this page](#) on the Arduino website.

With an SPI interface there is always one master device (the Arduino) that controls the peripheral devices (also known as slaves). You can control the display either through the Arduino's AVR microcontroller hardware SPI interface or three arbitrary digital pins (software SPI).

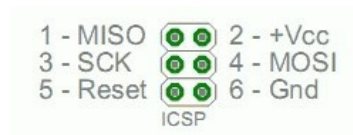
The hardware SPI pins (MOSI, MISO, and SCK) are at a specific location on each Arduino board. This interface is faster than using software SPI, but you will need to use the following fixed output pins:

Hardware SPI pin locations

Board	MOSI	MISO	SCK	Level
Arduino Uno	11 or ICSP-4	12 or ICSP-1	13 or ICSP-3	5 V
Arduino Mega	51 or ICSP-4	50 or ICSP-1	52 or ICSP-3	5 V
Arduino Leonardo	ICSP-4	ICSP-1	ICSP-3	5 V
Arduino Due	SPI-4	SPI1	SPI-3	3.3 V
Arduino MKR1000	8	10	9	3.3 V

Hardware SPI pin locations on different Arduino boards.

Note that the MOSI, MISO, and SCK pins are also at a consistent physical location on the 6-pin ICSP header:



Source: Arduino.cc

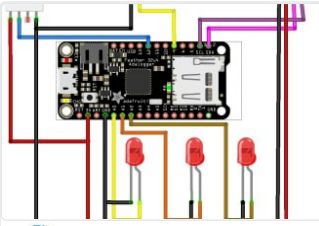
To control MAX7219 displays you only need to make three connections:

- **MOSI** (Master Out Slave In) connected to DIN – The Master line sending data to the peripherals.
- **SCK** (Serial Clock) connected to CLK – The clock pulses which synchronize data transmission generated by the master.
- **SS** (Slave Select) connected to CS – The pin on each device that the master can use to enable and disable specific devices.


You can daisy chain multiple displays to create one large display by connecting DOUT of the first display to DIN of the next display. VCC, GND, CLK, and CS are shared between all of the displays.

You can select any of the digital pins of the Arduino for the SS/CS pin. Note that for this tutorial I used pin 3 (see table below).


Cheap Arduino Coding Services Done For You:



- ✓ CIRCUIT LAYOUT
- ✓ PCB DESIGNING
- ✓ ARDUINO CODING
- ✓ SIMULATIONS





ORDER NOW




ARDUINO PROGRAMMING, CODE AND PROJECTS



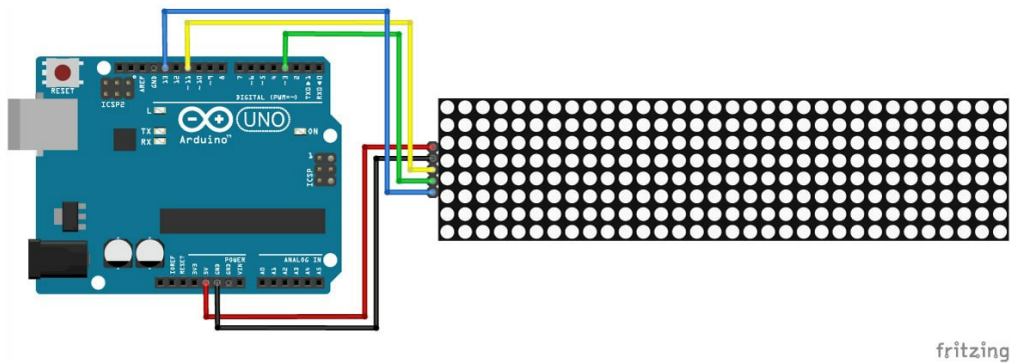

arduino_yard1
 I will do arduino, esp32 and es... projects for you
 ★★★★★ 5.0 (23)
 STARTING AT \$5


a4anas
 I will do arduino, raspberry... programming
 ★★★★★ 5.0 (6)
 STARTING AT \$20


asadaslam7
 I will do arduino progra... projects for you
 ★★★★★ 5.0 (330)
 STARTING AT \$10

[Explore more Arduino services](#)
[Services on fiverr.](#)

The wiring diagram below shows you how to connect the MAX7219 LED dot matrix display to the Arduino. **Note that when using the MD_Parola library, you need to orient the display with the DIN connector on the right, otherwise the text will be printed upside down.** For more information see [the section below](#).



MAX7219 LED dot matrix display with Arduino wiring diagram

The connections are also given in the table below:

MAX7219 LED dot matrix display connections

MAX7219 Display	Arduino
VCC	5 V
GND	GND
DIN	11 (MOSI)
CS	3 (SS)
CLK	13 (SCK)

If you want to use software SPI instead, you can connect DIN, CS, and CLK to any of the digital pins of the Arduino. You just need to specify the pin numbers in the setup of the Arduino code (see examples below).

Power requirements

The maximum power that the Arduino Uno can safely deliver when powered from USB is around 400 mA at 5 V. If you want to control a large display it is therefore advised to use an external power supply.

Installing the MD_Parola and MD_MAX72XX Arduino libraries

To control the MAX7219 display we will be using two awesome Arduino libraries created by Marco Colli from MajicDesigns. The **MD_Parola** library can be used to create many different text animations like scrolling and sprite text effects. This library depends on the **MD_MAX72XX** library which implements the hardware functions of the LED matrix.

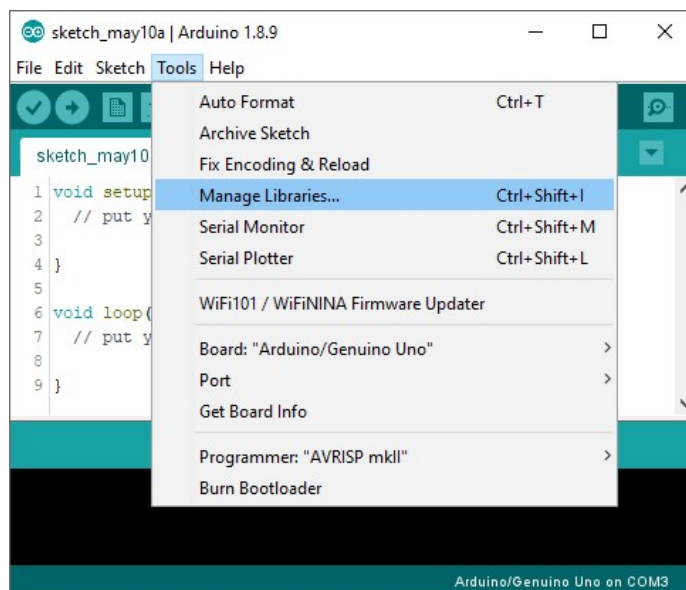
This are some functions and features of the library:

- Left, right, or center text justification
- Text scrolling with entry and exit effects
- Control display parameters and animation speed
- Multiple virtual displays (zones) in each string of LED modules
- Support for hardware SPI interface
- User-defined fonts and/or individual characters substitutions
- Support for double-height displays
- Support for mixing text and graphics on the same display

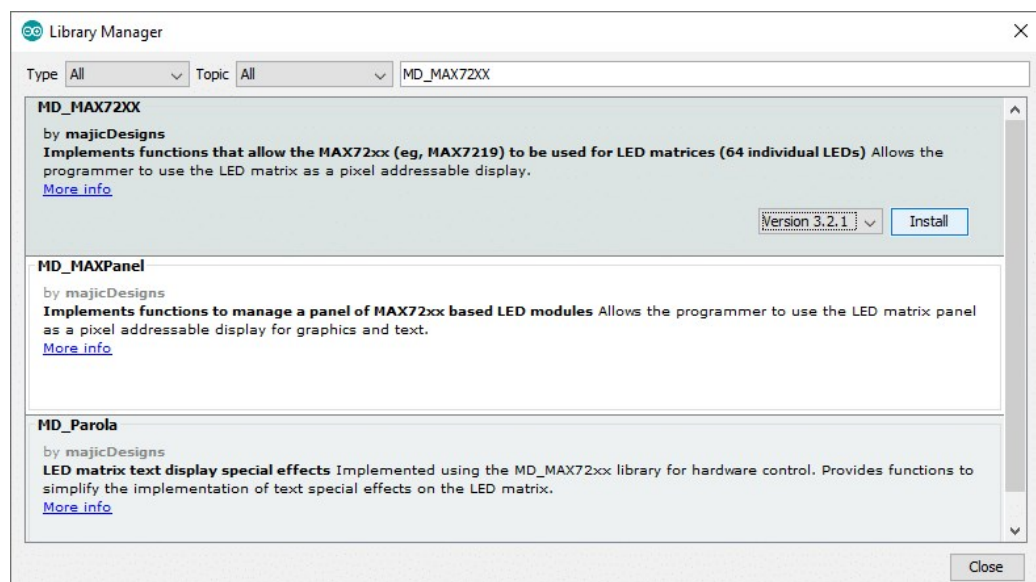
Marco has been working on this library for several years and has written some excellent tutorials on [his blog](#). The source code and documentation for the libraries can be found here:

- [MD_Parola source code on GitHub](#)
- [MD_Parola documentation](#)
- [MD_MAX72XX source code on GitHub](#)
- [MD_MAX72XX documentation](#)

You can install the libraries via the Library Manager of the [Arduino IDE](#). Go to Tools > Manage Libraries... or type Ctrl + Shift + I on Windows. The Library Manager will open and update the list of installed libraries.



Search for 'MD_MAX72XX' and look for the libraries by majicDesigns. Select the latest version and then click install. Make sure that you install both the MD_MAX72XX library and the MD_Parola library.



Different types of LED dot matrix displays

Many different types and sizes of MAX7219 LED dot matrix displays are available on the market. The MD_MAX72XX library supports almost all of these displays but you need to configure the library correctly for the type of matrix being used.

Below you can find information about connecting and configuring the most common MAX7219 LED dot matrix displays that you can buy on Amazon, AliExpress, and eBay.

FC-16 8×8 or 8×32 module



This is probably the most common MAX7219 display that you can find. It typically comes as either an 8×8 or 8×32 LED matrix and you can buy them with different colors of LEDs.

Module orientation and connections

You can easily connect multiple 8×8 or 8×32 modules together to create one larger display. I typically solder [straight male headers](#) on the back of the modules and connect them together using [jumpers](#). This way you can take them apart without having to desolder any connections.



The display is oriented with the DIN side on the right. Note that the screen-printed text on the back of the PCB might be upside down in this orientation.

	DP	A	B	C	D	E	F	G	
	7	6	5	4	3	2	1	0	D0
CLK <---								1	D1 <--- CLK
CS <---								2	D2 <--- CS
DOUT <---								3	D3 <--- DIN
GND <---							0	4	D4 <--- GND
VCC <---						0	0	5	D5 <--- VCC
					0	0	0	6	D6
				0	0	0	0	7	D7

Hardware configuration in Arduino code

When setting up the display in your Arduino code you need to set the **HARDWARE_TYPE** to **FC16_HW** and specify the number of devices you have connected. An 8×8 matrix counts as 1 device, so if you want to control an 8×32 module you need to set **MAX_DEVICES** to 4 (an 8×32 display contains 4 MAX7219 ICs).

```
// Hardware SPI:
#define HARDWARE_TYPE MD_MAX72XX::FC16_HW
#define MAX_DEVICES 4
#define CS_PIN 2

// Create a new instance of the MD_MAX72XX class:
MD_Parola matrix = MD_Parola(HARDWARE_TYPE, CS_PIN, MAX_DEVICES);

// For software SPI you also need to specify the DATA_PIN and the CLK_PIN connections:
// #define DATA_PIN 3
// #define CLK_PIN 4

// Create a new instance of the MD_MAX72XX class:
// MD_Parola matrix = MD_Parola(HARDWARE_TYPE, DATA_PIN, CLK_PIN, CS_PIN, MAX_DEVICES);
```

Generic 8×8 module



This is an 8×8 module mounted on a green PCB with the MAX7219 IC below the LED matrix. They are characterized by the 5-pin connectors at the short ends of the rectangular PCB.

Module orientation and connections

The generic module needs to be oriented with the MAX7219 IC at the top. You can connect multiple modules together with some short [female to female jumper wires](#). Simply connect all the pins of the DOUT side of the first module to the DIN side of the next module.

	C	C	D	G	V	
	L	S	I	N	C	
	K		N	D	C	
	V	V	V			
D7 D6 D5 D4 D3 D2 D1 D0						
+-----+						
7	6	5	4	3	2	1 0 DP
						1 A
						2 B
						3 C
0						4 D
0	0					5 E
0	0	0				6 F
0	0	0	0			7 G
+-----+						
	V	V	V			
	C	C	D	G	V	
	L	S	O	N	C	
	K		U	D	C	
	T					

Hardware configuration in Arduino code

For the generic display modules, you need to set the **HARDWARE_TYPE** to **GENERIC_HW**. The rest of the setup and MAX_DEVICES is the same as for the FC-16 modules.

```
// Hardware SPI:
#define HARDWARE_TYPE MD_MAX72XX::GENERIC_HW
```



```
#define MAX_DEVICES 1
#define CS_PIN 2

// Create a new instance of the MD_MAX72XX class:
MD_Parola matrix = MD_Parola(HARDWARE_TYPE, CS_PIN, MAX_DEVICES);

// For software SPI you also need to specify the DATA_PIN and the CLK_PIN connections:
// #define DATA_PIN 3
// #define CLK_PIN 4

// Create a new instance of the MD_MAX72XX class:
// MD_Parola matrix = MD_Parola(HARDWARE_TYPE, DATA_PIN, CLK_PIN, CS_PIN, MAX_DEVICES);
```

Arduino example codes

Below you will find several example codes that cover the basic functions of the MD_Parola Arduino library. After each example I explain how the code works so you should be able to modify it to suit your needs. You can also find more examples when you go to File > Examples > MD_Parola in the Arduino IDE, but they don't include any explanation so they might be a bit hard to follow.

Basic Arduino example code to print text

With the example code below you can print text on the display without any animations.

You can upload the example code to your Arduino via the Arduino IDE. For this tutorial, I used [this standard 8x32 LED dot matrix display](#) but you can use other types and/or sizes as well (see code explanation below).

You can copy the code by clicking on the button in the top right corner of the code field.

```
/* Basic example code for MAX7219 LED dot matrix display with Arduino. More info: https://www.makerguides.com */

// Include the required Arduino libraries:
#include <MD_Parola.h>
#include <MD_MAX72xx.h>
#include <SPI.h>

// Define hardware type, size, and output pins:
#define HARDWARE_TYPE MD_MAX72XX::FC16_HW
#define MAX_DEVICES 4
#define CS_PIN 3

// Create a new instance of the MD_Parola class with hardware SPI connection:
MD_Parola myDisplay = MD_Parola(HARDWARE_TYPE, CS_PIN, MAX_DEVICES);

// Setup for software SPI:
// #define DATAPIN 2
// #define CLK_PIN 4
// MD_Parola myDisplay = MD_Parola(HARDWARE_TYPE, DATA_PIN, CLK_PIN, CS_PIN, MAX_DEVICES);

void setup() {
  // Initialize the object:
  myDisplay.begin();
  // Set the intensity (brightness) of the display (0-15):
  myDisplay.setIntensity(0);
  // Clear the display:
  myDisplay.displayClear();
}

void loop() {
  myDisplay.setTextAlignment(PA_CENTER);
  myDisplay.print("Center");
  delay(2000);
  myDisplay.setTextAlignment(PA_LEFT);
```

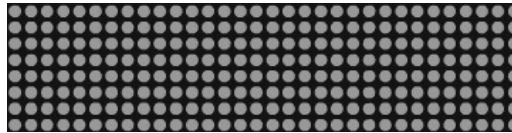


```

myDisplay.print("Left");
delay(2000);
myDisplay.setTextAlignment(PA_RIGHT);
myDisplay.print("Right");
delay(2000);
myDisplay.setTextAlignment(PA_CENTER);
myDisplay.setInvert(true);
myDisplay.print("Invert");
delay(2000);
myDisplay.setInvert(false);
myDisplay.print(1234);
delay(2000);
}

```

You should see the following output:



How the code works

The first step is to include all the required Arduino libraries. As I mentioned before, the MD_MAX72XX library implements the hardware functions of the LED matrix and the MD_Parola library the text effects. You will also need to include the SPI library, which comes pre-installed in the Arduino IDE. This library is used for the Serial Peripheral Interface communication between the display and the Arduino.

```

// Include the required Arduino libraries:
#include <MD_Parola.h>
#include <MD_MAX72xx.h>
#include <SPI.h>

```

Next, we need to specify which hardware we are using. Since I used a standard 8×32 display (also known as FC-16), I set the `HARDWARE_TYPE` to `FC16_HW`. The number of MAX7219 ICs in an 8×32 display is 4 so I set `MAX_DEVICES` to 4. Lastly, I defined to which pin the CS pin of the display is connected (output pin 3 in this case). See the [section about display types](#) for a more detailed explanation on how to set up other types of displays.

The statement

```
#define
```

is used to give a name to a constant value. The compiler will replace any references to this constant with the defined value when the program is compiled. So everywhere you mention

```
CS_PIN
```

, the compiler will replace it with the value 3 when the program is compiled.

```

// Define hardware type, size, and output pins:
#define HARDWARE_TYPE MD_MAX72XX::FC16_HW
#define MAX_DEVICES 4
#define CS_PIN 3

```

After this, a new instance of the MD_Parola class is created with the function

```
MD_Parola()
```

. This function needs three parameters, the first is the hardware type, the second the CS pin, and the third the number of max devices connected.

Note that I have called the MD_Parola object 'myDisplay' but you can use other names as well. You will need to change 'myDisplay' to the new name in the rest of the sketch.

When you want to use software SPI instead of hardware SPI, you also need to define the data and clock output pins and pass these as parameters when setting up the display object.

```
// Create a new instance of the MD_Parola class with hardware SPI connection:
MD_Parola myDisplay = MD_Parola(HARDWARE_TYPE, CS_PIN, MAX_DEVICES);

// Setup for software SPI:
// #define DATAPIN 2
// #define CLK_PIN 4
// MD_Parola myDisplay = MD_Parola(HARDWARE_TYPE, DATA_PIN, CLK_PIN, CS_PIN, MAX_DEVICES
);
```

In the setup section of the code, we first initialize the object with the function

```
begin()
```

. The brightness of the display can be set with the function

```
setIntensity()
```

. You can enter a value between 0 (minimum brightness) and 15 (maximum brightness). The display is cleared with the function

```
displayClear()
```

```
void setup() {
  // Intialize the object:
  myDisplay.begin();
  // Set the intensity (brightness) of the display (0-15):
  myDisplay.setIntensity(0);
  // Clear the display:
  myDisplay.displayClear();
}
```

In the loop section of the code, we first set the alignment of the text to be printed with the function `setTextAlignment()`. You can left, center, and right align the text with `PA_LEFT`, `PA_CENTER`, and `PA_RIGHT` respectively.

Next, the string 'Center' is printed with

```
myDisplay.print("Center")
```

. Note that you need to place quotation marks (" ") around the text since we are printing a [text string](#). When you want to print numbers, no quotation marks are necessary. For example

```
myDisplay.print(1234)
```

. You can invert the display, i.e. LEDs normally on turn off and vice versa, with

```
myDisplay.setInvert(true)
```

```
void loop() {
  myDisplay.setTextAlignment(PA_CENTER);
  myDisplay.print("Center");
  delay(2000);
  myDisplay.setTextAlignment(PA_LEFT);
  myDisplay.print("Left");
  delay(2000);
  myDisplay.setTextAlignment(PA_RIGHT);
  myDisplay.print("Right");
}
```

```

delay(2000);
myDisplay.setTextAlignment(PA_CENTER);
myDisplay.setInvert(true);
myDisplay.print("Invert");
delay(2000);
myDisplay.setInvert(false);
myDisplay.print(1234);
delay(2000);
}

```

Scrolling text Arduino example code

When you want to print a message on a dot matrix display, you will often find that the display is too small to fit the entire message. The solution is in the MD_Parola library, which makes it super easy to create scrolling text effects. In the following examples, I will show you how to set this up, as well as how to use some of the other available text effects.



You can copy the code below by clicking on the button in the top right corner of the code field.

```

/* Example code for scrolling text effect on MAX7219 LED dot matrix display with Arduino
. More info: https://www.makerguides.com */

// Include the required Arduino libraries:
#include <MD_Parola.h>
#include <MD_MAX72xx.h>
#include <SPI.h>

// Define hardware type, size, and output pins:
#define HARDWARE_TYPE MD_MAX72XX::FC16_HW
#define MAX_DEVICES 4
#define CS_PIN 3

// Create a new instance of the MD_Parola class with hardware SPI connection:
MD_Parola myDisplay = MD_Parola(HARDWARE_TYPE, CS_PIN, MAX_DEVICES);

// Setup for software SPI:
// #define DATA_PIN 2
// #define CLK_PIN 4
// MD_Parola myDisplay = MD_Parola(HARDWARE_TYPE, DATA_PIN, CLK_PIN, CS_PIN, MAX_DEVICES);

void setup() {
  // Initialize the object:
  myDisplay.begin();
  // Set the intensity (brightness) of the display (0-15):
  myDisplay.setIntensity(0);
  // Clear the display:
  myDisplay.displayClear();
  myDisplay.displayText("Scrolling text", PA_CENTER, 100, 0, PA_SCROLL_LEFT, PA_SCROLL_LEFT);
}

void loop() {
  if (myDisplay.displayAnimate()) {
    myDisplay.displayReset();
  }
}

```

How the code works

The first part of the code up to the end of the setup section is exactly the same as in the previous example. At the end of the setup section, we specify how we want to display the text with the function

```
displayText(pText, align, speed, pause, effectIn, effectOut)
```

. This function takes 5 arguments.

The first parameter is the text string, in this case "Scrolling text".

The second argument sets the alignment of the text during the optional pause. You can use the same alignment options as in the previous example, i.e. PA_CENTER, PA_LEFT, or PA_RIGHT.

The third and fourth arguments set the speed of the animation and pause time respectively. The speed of the display is the time in milliseconds between animation frames. The lower this time the faster the animation. If you want to pause the text in between the in and out animation, you can set the pause time in milliseconds. I set it to zero so the text scrolls continuously.

Next, the in and out effects are specified. In this case I used PA_SCROLL_LEFT for both. See the example below for other text effects.

```
myDisplay.displayText("Scrolling text", PA_CENTER, 100, 0, PA_SCROLL_LEFT, PA_SCROLL_L  
EFT);
```

In the loop section, you only need two functions to create a scrolling text display.

First, we use the function

```
displayAnimate()
```

in an

```
if
```

statement. This function animates the display using the currently specified text and animation parameters and returns true when the animation has finished. When the animation has finished, we reset the display with the function

```
displayReset()
```

so the text is displayed in a loop.

```
void loop() {  
  if (myDisplay.displayAnimate()) {  
    myDisplay.displayReset();  
  }  
}
```

Other text effects

The library includes several other text effects that you can use:

- PA_PRINT,
- PA_SCAN_HORIZ,
- PA_SCROLL_LEFT,
- PA_WIPE,
- PA_SCROLL_UP_LEFT,
- PA_SCROLL_UP,
- PA_OPENING_CURSOR,
- PA_GROW_UP,

- PA_MESH,
- PA_SCROLL_UP_RIGHT,
- PA_BLINDS,
- PA_CLOSING,
- PA_RANDOM,
- PA_GROW_DOWN,
- PA_SCAN_VERT,
- PA_SCROLL_DOWN_LEFT,
- PA_WIPE_CURSOR,
- PA DISSOLVE,
- PA_OPENING,
- PA_CLOSING_CURSOR,
- PA_SCROLL_DOWN_RIGHT,
- PA_SCROLL_RIGHT,
- PA_SLICE,
- PA_SCROLL_DOWN

The example code below steps through the different effect so you can see what they look like.

```
/* Example code for scrolling text and other text effects on MAX7219 LED dot matrix display with Arduino. More info: https://www.makerguides.com */
```

```
// Include the required Arduino libraries:
```

```
#include <MD_Parola.h>
```

```
#include <MD_MAX72xx.h>
```

```
#include <SPI.h>
```

```
int i = 0;
```

```
textEffect_t texteffect[] =
```

```
{
  PA_PRINT,
  PA_SCAN_HORIZ,
  PA_SCROLL_LEFT,
  PA_WIPE,
  PA_SCROLL_UP_LEFT,
  PA_SCROLL_UP,
  PA_OPENING_CURSOR,
  PA_GROW_UP,
  PA_MESH,
  PA_SCROLL_UP_RIGHT,
  PA_BLINDS,
  PA_CLOSING,
  PA_RANDOM,
  PA_GROW_DOWN,
  PA_SCAN_VERT,
  PA_SCROLL_DOWN_LEFT,
  PA_WIPE_CURSOR,
  PA DISSOLVE,
  PA_OPENING,
  PA_CLOSING_CURSOR,
  PA_SCROLL_DOWN_RIGHT,
  PA_SCROLL_RIGHT,
  PA_SLICE,
  PA_SCROLL_DOWN
};
```

```
// Define hardware type, size, and output pins:
```

```
#define HARDWARE_TYPE MD_MAX72XX::FC16_HW
```

```
#define MAX_DEVICES 4
```

```
#define CS_PIN 3
```

```
// Create a new instance of the MD_Parola class with hardware SPI connection:
```

```
MD_Parola myDisplay = MD_Parola(HARDWARE_TYPE, CS_PIN, MAX_DEVICES);
```

```
// Setup for software SPI:
// #define DATA_PIN 2
// #define CLK_PIN 4
// MD_Parola myDisplay = MD_Parola(HARDWARE_TYPE, DATA_PIN, CLK_PIN, CS_PIN, MAX_DEVICES
);

void setup() {
  myDisplay.begin();
  myDisplay.setIntensity(0);
  myDisplay.setTextAlignment(PA_CENTER);
  myDisplay.setPause(1000);
  myDisplay.setSpeed(100);
  myDisplay.displayClear();
}

void loop() {
  if (myDisplay.displayAnimate()) {
    if (i < sizeof(texteffect)) {
      i++;
    }
    else {
      i = 0;
    }
    myDisplay.displayText("Hello", myDisplay.getTextAlignment(), myDisplay.getSpeed(), m
yDisplay.getPause(), texteffect[i], texteffect[i]);
    myDisplay.displayReset();
  }
}
```

Text sprites

A relatively new function of the MD_Parola library is animated text sprites. In computer graphics, a sprite is a two-dimensional bitmap that is integrated into a larger scene (in this case, the matrix display).

A sprite is made up of a number of frames that run sequentially to make the animation on the display. Once the animation reaches the last frame it restarts from the first frame.

Note that I used an 8×64 matrix display for this example by connecting two 8×32 displays together (MAX_DEVICES is set to 8).

```
/* Example code for sprite text effect on MAX7219 LED dot matrix display with Arduino. M
ore info: https://www.makerguides.com */

// Include the required Arduino libraries:
#include <MD_Parola.h>
#include <MD_MAX72xx.h>
#include <SPI.h>

// Define hardware type, size, and output pins:
#define HARDWARE_TYPE MD_MAX72XX::FC16_HW
#define MAX_DEVICES 8
#define CS_PIN 3

// Create a new instance of the MD_Parola class with hardware SPI connection:
MD_Parola myDisplay = MD_Parola(HARDWARE_TYPE, CS_PIN, MAX_DEVICES);

// Setup for software SPI:
// #define DATA_PIN 2
// #define CLK_PIN 4
// MD_Parola myDisplay = MD_Parola(HARDWARE_TYPE, DATA_PIN, CLK_PIN, CS_PIN, MAX_DEVICES
);

// Sprite definitions:
const uint8_t F_PMAN1 = 6;
const uint8_t W_PMAN1 = 8;
const uint8_t PROGMEM pacman1[F_PMAN1 * W_PMAN1] = // gobbling pacman animation
{
```

```

0x00, 0x81, 0xc3, 0xe7, 0xff, 0x7e, 0x7e, 0x3c,
0x00, 0x42, 0xe7, 0xe7, 0xff, 0xff, 0x7e, 0x3c,
0x24, 0x66, 0xe7, 0xff, 0xff, 0xff, 0x7e, 0x3c,
0x3c, 0x7e, 0xff, 0xff, 0xff, 0xff, 0x7e, 0x3c,
0x24, 0x66, 0xe7, 0xff, 0xff, 0xff, 0x7e, 0x3c,
0x00, 0x42, 0xe7, 0xe7, 0xff, 0xff, 0x7e, 0x3c,
});

const uint8_t F_PMAN2 = 6;
const uint8_t W_PMAN2 = 18;
const uint8_t PROGMEM pacman2[F_PMAN2 * W_PMAN2] = // pacman pursued by a ghost
{
    0x00, 0x81, 0xc3, 0xe7, 0xff, 0x7e, 0x7e, 0x3c, 0x00, 0x00, 0x00, 0xfe, 0x7b, 0xf3, 0x
7f, 0xfb, 0x73, 0xfe,
    0x00, 0x42, 0xe7, 0xe7, 0xff, 0xff, 0x7e, 0x3c, 0x00, 0x00, 0x00, 0xfe, 0x7b, 0xf3, 0x
7f, 0xfb, 0x73, 0xfe,
    0x24, 0x66, 0xe7, 0xff, 0xff, 0xff, 0x7e, 0x3c, 0x00, 0x00, 0x00, 0xfe, 0x7b, 0xf3, 0x
7f, 0xfb, 0x73, 0xfe,
    0x3c, 0x7e, 0xff, 0xff, 0xff, 0xff, 0x7e, 0x3c, 0x00, 0x00, 0x00, 0xfe, 0x7b, 0xf3, 0x
7f, 0xfb, 0x73, 0xfe,
    0x24, 0x66, 0xe7, 0xff, 0xff, 0xff, 0x7e, 0x3c, 0x00, 0x00, 0x00, 0xfe, 0x7b, 0xf3, 0x
7f, 0xfb, 0x73, 0xfe,
    0x00, 0x42, 0xe7, 0xe7, 0xff, 0xff, 0x7e, 0x3c, 0x00, 0x00, 0x00, 0xfe, 0x7b, 0xf3, 0x
7f, 0xfb, 0x73, 0xfe,
});

const uint8_t F_WAVE = 14;
const uint8_t W_WAVE = 14;
const uint8_t PROGMEM wave[F_WAVE * W_WAVE] = // triangular wave / worm
{
    0x08, 0x04, 0x02, 0x01, 0x02, 0x04, 0x08, 0x10, 0x20, 0x40, 0x80, 0x40, 0x20, 0x10,
0x10, 0x08, 0x04, 0x02, 0x01, 0x02, 0x04, 0x08, 0x10, 0x20, 0x40, 0x80, 0x40, 0x20,
0x20, 0x10, 0x08, 0x04, 0x02, 0x01, 0x02, 0x04, 0x08, 0x10, 0x20, 0x40, 0x80, 0x40,
0x40, 0x20, 0x10, 0x08, 0x04, 0x02, 0x01, 0x02, 0x04, 0x08, 0x10, 0x20, 0x40, 0x80,
0x80, 0x40, 0x20, 0x10, 0x08, 0x04, 0x02, 0x01, 0x02, 0x04, 0x08, 0x10, 0x20, 0x40,
0x40, 0x80, 0x40, 0x20, 0x10, 0x08, 0x04, 0x02, 0x01, 0x02, 0x04, 0x08, 0x10, 0x20,
0x20, 0x40, 0x80, 0x40, 0x20, 0x10, 0x08, 0x04, 0x02, 0x01, 0x02, 0x04, 0x08, 0x10,
0x10, 0x20, 0x40, 0x80, 0x40, 0x20, 0x10, 0x08, 0x04, 0x02, 0x01, 0x02, 0x04, 0x08,
0x08, 0x10, 0x20, 0x40, 0x80, 0x40, 0x20, 0x10, 0x08, 0x04, 0x02, 0x01, 0x02, 0x04,
0x04, 0x08, 0x10, 0x20, 0x40, 0x80, 0x40, 0x20, 0x10, 0x08, 0x04, 0x02, 0x01, 0x02,
0x02, 0x04, 0x08, 0x10, 0x20, 0x40, 0x80, 0x40, 0x20, 0x10, 0x08, 0x04, 0x02, 0x01,
0x01, 0x02, 0x04, 0x08, 0x10, 0x20, 0x40, 0x80, 0x40, 0x20, 0x10, 0x08, 0x04, 0x02,
0x02, 0x01, 0x02, 0x04, 0x08, 0x10, 0x20, 0x40, 0x80, 0x40, 0x20, 0x10, 0x08, 0x04,
0x04, 0x02, 0x01, 0x02, 0x04, 0x08, 0x10, 0x20, 0x40, 0x80, 0x40, 0x20, 0x10, 0x08,
});

const uint8_t F_ROLL1 = 4;
const uint8_t W_ROLL1 = 8;
const uint8_t PROGMEM roll1[F_ROLL1 * W_ROLL1] = // rolling square
{
    0xff, 0x8f, 0x8f, 0x8f, 0x81, 0x81, 0x81, 0xff,
    0xff, 0xf1, 0xf1, 0xf1, 0x81, 0x81, 0x81, 0xff,
    0xff, 0x81, 0x81, 0x81, 0xf1, 0xf1, 0xf1, 0xff,
    0xff, 0x81, 0x81, 0x81, 0x8f, 0x8f, 0x8f, 0xff,
});

const uint8_t F_ROLL2 = 4;
const uint8_t W_ROLL2 = 8;
const uint8_t PROGMEM roll2[F_ROLL2 * W_ROLL2] = // rolling octagon
{
    0x3c, 0x4e, 0x8f, 0x8f, 0x81, 0x81, 0x42, 0x3c,
    0x3c, 0x72, 0xf1, 0xf1, 0x81, 0x81, 0x42, 0x3c,
    0x3c, 0x42, 0x81, 0x81, 0xf1, 0xf1, 0x72, 0x3c,
    0x3c, 0x42, 0x81, 0x81, 0x8f, 0x8f, 0x4e, 0x3c,
});

const uint8_t F_LINES = 3;
const uint8_t W_LINES = 8;
const uint8_t PROGMEM lines[F_LINES * W_LINES] = // spaced lines
{
    0xff, 0xff, 0xff, 0x00, 0x00, 0xff, 0x00, 0x00,
    0xff, 0xff, 0x00, 0xff, 0x00, 0x00, 0xff, 0x00,

```



```

0xff, 0xff, 0x00, 0x00, 0xff, 0x00, 0x00, 0xff,
};

const uint8_t F_ARROW1 = 3;
const uint8_t W_ARROW1 = 10;
const uint8_t PROGMEM arrow1[F_ARROW1 * W_ARROW1] = // arrow fading to center
{
    0x18, 0x3c, 0x7e, 0xff, 0x7e, 0x00, 0x00, 0x3c, 0x00, 0x00,
    0x18, 0x3c, 0x7e, 0xff, 0x00, 0x7e, 0x00, 0x00, 0x18, 0x00,
    0x18, 0x3c, 0x7e, 0xff, 0x00, 0x00, 0x3c, 0x00, 0x00, 0x18,
};

const uint8_t F_ARROW2 = 3;
const uint8_t W_ARROW2 = 9;
const uint8_t PROGMEM arrow2[F_ARROW2 * W_ARROW2] = // arrow fading to outside
{
    0x18, 0x3c, 0x7e, 0xe7, 0x00, 0x00, 0xc3, 0x00, 0x00,
    0x18, 0x3c, 0x7e, 0xe7, 0xe7, 0x00, 0x00, 0x81, 0x00,
    0x18, 0x3c, 0x7e, 0xe7, 0x00, 0xc3, 0x00, 0x00, 0x81,
};

const uint8_t F_SAILBOAT = 1;
const uint8_t W_SAILBOAT = 11;
const uint8_t PROGMEM sailboat[F_SAILBOAT * W_SAILBOAT] = // sail boat
{
    0x10, 0x30, 0x58, 0x94, 0x92, 0x9f, 0x92, 0x94, 0x98, 0x50, 0x30,
};

const uint8_t F_STEAMBOAT = 2;
const uint8_t W_STEAMBOAT = 11;
const uint8_t PROGMEM steamboat[F_STEAMBOAT * W_STEAMBOAT] = // steam boat
{
    0x10, 0x30, 0x50, 0x9c, 0x9e, 0x90, 0x91, 0x9c, 0x9d, 0x90, 0x71,
    0x10, 0x30, 0x50, 0x9c, 0x9c, 0x91, 0x90, 0x9d, 0x9e, 0x91, 0x70,
};

const uint8_t F_HEART = 5;
const uint8_t W_HEART = 9;
const uint8_t PROGMEM heart[F_HEART * W_HEART] = // beating heart
{
    0x0e, 0x11, 0x21, 0x42, 0x84, 0x42, 0x21, 0x11, 0x0e,
    0x0e, 0x1f, 0x33, 0x66, 0xcc, 0x66, 0x33, 0x1f, 0x0e,
    0x0e, 0x1f, 0x3f, 0x7e, 0xfc, 0x7e, 0x3f, 0x1f, 0x0e,
    0x0e, 0x1f, 0x33, 0x66, 0xcc, 0x66, 0x33, 0x1f, 0x0e,
    0x0e, 0x11, 0x21, 0x42, 0x84, 0x42, 0x21, 0x11, 0x0e,
};

const uint8_t F_INVADER = 2;
const uint8_t W_INVADER = 10;
const uint8_t PROGMEM invader[F_INVADER * W_INVADER] = // space invader
{
    0x0e, 0x98, 0x7d, 0x36, 0x3c, 0x3c, 0x36, 0x7d, 0x98, 0x0e,
    0x70, 0x18, 0x7d, 0xb6, 0x3c, 0x3c, 0xb6, 0x7d, 0x18, 0x70,
};

const uint8_t F_ROCKET = 2;
const uint8_t W_ROCKET = 11;
const uint8_t PROGMEM rocket[F_ROCKET * W_ROCKET] = // rocket
{
    0x18, 0x24, 0x42, 0x81, 0x99, 0x18, 0x99, 0x18, 0xa5, 0x5a, 0x81,
    0x18, 0x24, 0x42, 0x81, 0x18, 0x99, 0x18, 0x99, 0x24, 0x42, 0x99,
};

const uint8_t F_FBALL = 2;
const uint8_t W_FBALL = 11;
const uint8_t PROGMEM fireball[F_FBALL * W_FBALL] = // fireball
{
    0x7e, 0xab, 0x54, 0x28, 0x52, 0x24, 0x40, 0x18, 0x04, 0x10, 0x08,
    0x7e, 0xd5, 0x2a, 0x14, 0x24, 0x0a, 0x30, 0x04, 0x28, 0x08, 0x10,
};

const uint8_t F_CHEVRON = 1;

```

```

const uint8_t W_CHEVRON = 9;
const uint8_t PROGMEM chevron[F_CHEVRON * W_CHEVRON] = // chevron
{
    0x18, 0x3c, 0x66, 0xc3, 0x99, 0x3c, 0x66, 0xc3, 0x81,
};

const uint8_t F_WALKER = 5;
const uint8_t W_WALKER = 7;
const uint8_t PROGMEM walker[F_WALKER * W_WALKER] = // walking man
{
    0x00, 0x48, 0x77, 0x1f, 0x1c, 0x94, 0x68,
    0x00, 0x90, 0xee, 0x3e, 0x38, 0x28, 0xd0,
    0x00, 0x00, 0xae, 0xfe, 0x38, 0x28, 0x40,
    0x00, 0x00, 0x2e, 0xbe, 0xf8, 0x00, 0x00,
    0x00, 0x10, 0x6e, 0x3e, 0xb8, 0xe8, 0x00,
};

void setup() {
    myDisplay.begin();
    myDisplay.setIntensity(0);
    myDisplay.displayClear();
    myDisplay.setSpriteData(pacman2, W_PMAN2, F_PMAN2, pacman2, W_PMAN2, F_PMAN2);
    myDisplay.displayText("Parola sprites", PA_CENTER, 50, 1000, PA_SPRITE, PA_SPRITE);
}

void loop() {
    if (myDisplay.displayAnimate()) {
        myDisplay.displayReset();
    }
}

```

How the code works

After setting up the display like before, the text sprites are defined.

Two constants are used to define the sprite, one for the width (number of bytes) data for one sprite and the other for the number of frames contained in the animation. The total number of bytes required is the width * number of frames. Note that the sprite data is stored in PROGMEM to save RAM space.

Each row of the array is made up of hexadecimal numbers that set which LEDs need to light up in each column of the sprite.

The example includes many different sprite definitions, which you can also find in one of the examples that come with the library.

```

// Sprite definitions:
const uint8_t F_PMAN1 = 6;
const uint8_t W_PMAN1 = 8;
const uint8_t PROGMEM pacman1[F_PMAN1 * W_PMAN1] = // gobbling pacman animation
{
    0x00, 0x81, 0xc3, 0xe7, 0xff, 0x7e, 0x7e, 0x3c,
    0x00, 0x42, 0xe7, 0xe7, 0xff, 0xff, 0x7e, 0x3c,
    0x24, 0x66, 0xe7, 0xff, 0xff, 0xff, 0x7e, 0x3c,
    0x3c, 0x7e, 0xff, 0xff, 0xff, 0xff, 0x7e, 0x3c,
    0x24, 0x66, 0xe7, 0xff, 0xff, 0xff, 0x7e, 0x3c,
    0x00, 0x42, 0xe7, 0xe7, 0xff, 0xff, 0x7e, 0x3c,
};

```

In the setup, the function

```
setSpriteData(inData, inWidth, inFrames, outData, outWidth, outFrames)
```

is used to set up user data needed so that the library can display the sprite when the PA_SPRITE animation type is selected in the

```
displayText()
```

function.

You can select any of the predefined sprites, in this case I used pacman2 (pacman pursued by a ghost).

```
myDisplay.setSpriteData(pacman2, W_PMAN2, F_PMAN2, pacman2, W_PMAN2, F_PMAN2);  
myDisplay.displayText("Parola sprites", PA_CENTER, 50, 1000, PA_SPRITE, PA_SPRITE);
```

The loop section is the same as before.

Conclusion

In this article, I have shown you how you can use a MAX7219 LED dot matrix display with Arduino. We looked at the basics of printing text, scrolling text, other text effects, and text sprites. There still are some less frequently used functions in the MD_Parola library that I haven't covered in this tutorial but you can check those out in the MD_Parola documentation on GitHub.

I hope you found this tutorial useful and informative. If you did, please **share it with a friend** who also likes electronics and making things!

I would love to know what projects you plan on building (or have already built) with this display. If you have any questions, suggestions, or if you think that things are missing in this tutorial, **please leave a comment down below**.

Note that comments are held for moderation to prevent spam.



This work is licensed under a [Creative Commons Attribution-NonCommercial-ShareAlike 4.0 International License](#).

Other Useful Links From Around The Web:

- [Project Example with the I2C](#)
- [Another simple project example](#)
- [Circuit Geeks I2C LCD tutorial](#)
- [Micro Controllers Lab I2C LCD tutorial](#)
- [Useful YouTube Tutorial on the I2C LCD with Arduino](#)



By Benne

Published: May 24, 2020 - Last updated: March 2, 2022

Arduino, Displays, Tutorials

[Home](#) > [Tutorials](#) > MAX7219 LED dot matrix display Arduino tutorial

1088AS

Arduino

Display

Electronics

LED

Matrix

MAX7219

Tutorial

← [TB6600 Stepper Motor Driver with Arduino Tutorial](#)

[Automatic plant watering system with Arduino IoT Cloud](#) →

25 COMMENTS

Comment

Write your comment...

Name *

Email *

Email

CO
MM
ENT



Doug

June 22, 2022 at 07:37 PM

Thanks for this. Due to learning about hardware types, I changed the sketch I had to FC16_HW and no more garbled clock display.



Sirhornet

June 4, 2022 at 10:35 AM

Hi, I was wondering if anyone had a code so I could interface a DHT11 with a MAX 7219, when I've tried all I get is a "0" in the centre of the display, even though the serial monitor tells me the DHT11 is reading correctly.



Dzil

April 8, 2022 at 07:49 PM

Fantastic Tutorial, just right for beginner like me, many thanks to contributor with great explanations.



giuseppe traversa

June 27, 2021 at 09:38 AM

Grazie infinite hai risolto una buona parte dei miei dubbi è un tutorial descritto con particolare semplicità. Vorrei chiedere se è possibile un tutorial sullo scorrimento della data l'ora con esp32 o esp8266. Grazie



Eka

June 12, 2021 at 01:16 PM

Thanks for your tutorial. This Parola's lib really make my life easier.

But, in my case, I'm installing 3 MAX_Devices, in serial, how to make them drive individually?

For example, 1st MAX is arrow head with scrolling left, 2nd MAX will be a number (0-9) fix, no scrolling, and the 3rd is arrow head with scrolling right, like in the lift application.

Should I separate 3 of them and make individual connection of each (consuming 9 MCU pins).

Please, guide me on this matter.

Thanks, Eka



Reilly

May 20, 2021 at 10:32 PM

Hey, looking for some assistance. I tried setting this up, but when I upload the code from the Basic Arduino Print to Code step, nothing happens. I am using Pin 12 instead of pin 3, but i changed that in the code. I am not using SPI. It doesnt seem to work with the scrolling text code loop either. I'm very new to this arduino thing though so maybe I messed something else up? I did double and triple check my wiring. Weirdly the first time I tried to run it, before I changed my CS pin to match in the code because I forgot to the first time, it lit up but in a garbled mess, ending with all the lights lighting up red. Any help is appreciated



Aditya

July 7, 2021 at 05:47 AM

Hi, in the void loop function, try changing the display "setIntensity" to something other than 0, up till 15



Paul

May 5, 2021 at 01:08 PM

Hi,

Thank you for this tutorial.

On the Arduino Nano 33 BLE I had to use the 3.3V instead of the 5V (if it can help).

Paul.



Freddie

March 7, 2021 at 11:52 AM

Is there any way on making a game on this like snake?



charles

February 18, 2021 at 05:22 PM

can i use home made 8x8 LED module with the library?



English



Benne de Bakker

March 2, 2021 at 08:58 AM

If you used the MAX7219 LED driver and wired the module as described in the datasheet, I think it should work without a problem.



juan jose

February 6, 2021 at 07:00 AM

Thanks for a wonderful tutorial. One question though, I tried stepping through all the sprites the way you stepped through the text effects. But it either only shows the final one or they overlap each other during the same animation. Any suggestions? Thanks again for getting me this far.



Alex

January 23, 2021 at 01:38 AM

Great tutorial, thank you for sharing your knowledge with us !
I love the effects I get now with my matrix LED displays.
Alex.



Rolandow

November 30, 2020 at 10:51 AM

I don't understand how to connect this is you have a 3.3V arduino. The "Hardware SPI pin locations"

table shows two arduino's that use 3.3V, but how can this work if the dot matrix requires 5V?



Benne de Bakker

December 1, 2020 at 02:31 PM

Hi Rolandow,

marco_c, the author of the Parola library, discusses this on his website:

<https://arduinoplusplus.wordpress.com/2020/08/29/parola-a-to-z-frequently-asked-issues/>

The MAX7219 should work on 3.3 V logic but you might need a voltage level shifter if you connect more than a few matrices in the chain.

Benne



antonio1

November 23, 2020 at 11:32 PM

I found your site by looking for adjusting the intensity. With all parola examples I couldn't find how to reduce the intensity to a comfortable reading in night. Your examples were so clear and simple, really nice knowledge sharing with us.

My project is displaying during the night the time, outside temperature, outside humidity, inside temperature and inside humidity. During the winter for me is very important when I wake up to know immediately how start the day, how much negative degrees and controlling the humidity.

Thank you again, you solved my part so easy.



Benne de Bakker

November 28, 2020 at 12:59 PM

Thanks!



Ian Stockdale

November 12, 2020 at 10:20 AM

Many thanks for this as it was nice to get back into Arduinos after a lay off. when I was a lecturer I had a colleague who specialised in this and so was hooked. I'm looking to find out how to add sprites before and after a scrolling text message – is it possible to do a tutorial on this. Once again many thanks.



Roger Ashley

November 7, 2020 at 06:57 AM

Very good, found it easy to understand so far. But have one problem. How does one write a line to include a value. I have tried many ways including:

```
myDisplay.setTextAlignment(PA_LEFT);
```

```
myDisplay.print("Valentina ",val1);
```

I have put an int val1 = 0: in top section of sketch;

I'm 76 and trying to learn so any help please.



Mike Bonham

November 12, 2020 at 12:10 AM

I got it to work by defining an array of characters like this just before setting the message:

```
char text[20]="Your message here ";
```

```
myDisplay.displayText(text, PA_CENTER, 50, 0, PA_SCROLL_LEFT, PA_SCROLL_LEFT);
```



Andrew

September 9, 2020 at 12:51 AM

Thank you So very much, thanks to you I now have a number of physical examples to show my students how simple it can be to create a custom LED display, I'm sure they are going to have fun building and designing their own, I am them going to get them to design and build a small timber case in woodwork for them to house it in and take home.

Thanks Again, Cheers

Andrew



Philip Ainscough Hart

July 30, 2020 at 03:47 PM

Awesome libraries – thanks!



Wes

July 2, 2020 at 02:58 AM

I am using the Generic displays, and it seems that the unit connected to the Arduino is the rightmost unit with the rest of the daisy chain fitting to the left. Is this correct?



Benne de Bakker

July 6, 2020 at 08:13 AM

Yes, this is correct!



JOHN

June 28, 2020 at 07:38 PM

I have wasted weeks and weeks trying to figure how to make these fc-16 displays work . They are backwards and upside down. I'm very glad you took the time to put this work together . I'm 77 years old and it's very hard to learn this new stuff . Thank you . John Okaly


Makerguides

Guides, Tutorials & Projects For The
Maker Community



Navigation

- ▶ Arduino Displays
- ▶ Distance Sensors
- ▶ Motor Controls
- ▶ Temperature & Humidity Sensors
- ▶ Other Tutorials
- ▶ Projects
- ▶ FAQs
- ▶ Contact
- ▶ Disclaimer
- ▶ Privacy Policy
- ▶ Terms & Conditions

 English

