

LedMatrix/LedMatrixSPI

An Arduino library for controlling multiple 8x8 LED displays in a matrix style format using the MAX7219/7221 display drivers. It does not support 7-segment displays.

The two libraries are identical except that LedMatrix uses a software SPI implementation and LedMatrixSPI uses a hardware SPI implementation. The software version was initially written, but the refresh rates were not sufficient for a large number of displays, so a second hardware SPI version was written. The only advantage of using LedMatrix is you do not have to use the dedicated SPI pins.

Written by Aaron Groom, groom.aaron@gmail.com, 2015

Features

- Supports up to 32 rows and columns of displays⁽¹⁾.
 - Basic drawing functions (line, rect, quad, triangle, circle, ellipse).
 - Scrolling text and characters.
 - Two built-in fonts (8x8 and 5x7).
- (1) The library can support up to 32 rows and columns of 8x8 displays (or 256 x 256 leds); however, due to display buffering it's unlikely the MCU will have sufficient memory to support more than 180 8x8 displays.

Functions

LedMatrix(`uint8_t` rows, `uint8_t` cols)
LedMatrixSPI(`uint8_t` rows, `uint8_t` cols)

Constructor for library. Rows and columns refer to the number of 8x8 display rows and columns. This uses the default data, clock and cs pins. If you want to change the pins you must use the LedMatrix library and use alternate constructor.

Default pins:

- Data: digital 11
- Clock: digital 10
- CS: digital 13

LedMatrix(`uint8_t` rows, `uint8_t` cols, `uint8_t` data, `uint8_t` clk, `uint8_t` cs)

Alternate constructor for the LedMatrix library only. Sets alternate data, clock and cs pins. Currently the only pins supported are digital 8-13, or those on PORTB.

void setIntensity(`uint8_t` intensity)

Sets the brightness of all displays (0 = lowest, 15 = brightest). The default brightness is 1.

void shutdown(**bool** value)

Sets the shutdown mode for power saving (1 = shutdown mode, 0 = normal operation). The default mode is normal operation.

void setFont(**font** f)

Sets the font to use for the scrollChar() and drawChar() functions. There are currently two supported fonts, 5x7 and 8x8. The default 8x8.

void setLed(**uint8_t** x, **uint8_t** y, **bool** value)

Sets the value of an individual led in the array buffer at the x, y coordinates specified. This only changes the value of the led in the display buffer, but does not update the display. You must use the update() function to update the display. The coordinate grid starts in the upper left corner (0,0). Coordinates outside the matrix's range are ignored.

void update(**void**)

Updates the display with changes in the led matrix array. Must be called to update the display after any drawing function is called. This was done so multiple changes could be made before updating the screen to improve refresh rates. The only exception is when using the scrollChar() function.

void clear(**void**)

Clears the current values stored in the led matrix array, but does not clear the display.

void clearScreen(**void**)

Clears the current values stored in the led matrix array, and clears the screen. If you wish to clear the screen only, but not alter the values in the led matrix array, use the shutdown() function.

void drawChar(**char** c, **uint8_t** x , **uint8_t** y)

Draws a character at the specified x (left) and y (top) coordinates specified. The x and y coordinates must begin on the display, but may extend outside the display's limits.

void scrollChar(**char** c, **uint8_t** row)

Scrolls a single character onto the display, from right to left, on the specified row. Row refers to the 8x8 grid row, where the top row = 1.

void scrollDelay(uint8_t value)

Sets the delay for the scrollChar() function. Because the LedMatrix is much slower than LedMatrixSPI, the values used for each will differ. A zero for LedMatrixSPI is much too fast to read, but a zero for LedMatrix is still readable.

void line(uint8_t x1, uint8_t y1, uint8_t x2, uint8_t y2)

Draws a line from (x1, y1) to (x2, y2). All coordinates must be within the grid's limits or the call is ignored.

void triangle(uint8_t x1, uint8_t y1, uint8_t x2, uint8_t y2, uint8_t x3, uint8_t y3)

Draws a triangle. All coordinates must be within the grid's limits or the call is ignored.

void quad(uint8_t x1, uint8_t y1, uint8_t x2, uint8_t y2, uint8_t x3, uint8_t y3, uint8_t x4, uint8_t y4)

Draws any quadrilateral, with the points listed as parameters in a clockwise or counter-clockwise direction. All coordinates must be within the grid's limits or the call is ignored.

void rect(uint8_t x, uint8_t y, uint8_t width, uint8_t height)

Draws a rectangle, using the x and y coordinates as the top left corner, of the specified height and width. The x and y coordinates must begin on the display, but may extend outside the display's limits.

void circle(uint8_t x, uint8_t y, uint8_t radius)

Draws a circle with the specified radius, using the x and y coordinates as the center. The x and y coordinates must begin on the display, but may extend outside the display's limits.

void ellipse(uint8_t x, uint8_t y, uint8_t width, uint8_t height)

Draws an ellipse, using the x and y coordinates as the center, of the specified height and width. The x and y coordinates must begin on the display, but may extend outside the display's limits.