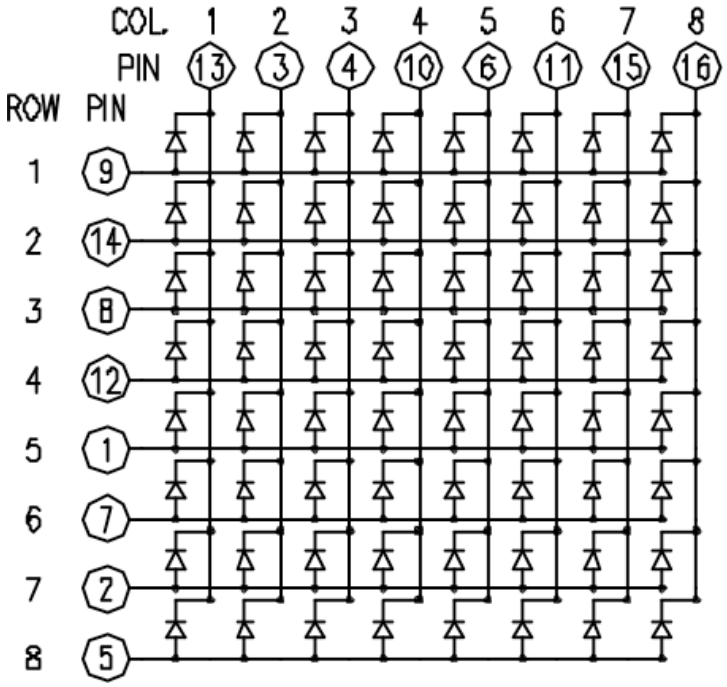
MODULES: Example DOT MATRIX

For our example, you must use a **8x8 LED matrix**

This form is often packaged as an array of LEDs where the anodes are connected to the column while the cathodes per line (or vice versa). An LED lights up when the row corresponding to its cathode is connected through a resistor to ground (**LOW**), while the column corresponding to its anode is connected to + 5V. (**HIGH**)

The 8x8 matrix LED has 16 pins. Each leg is associated with a row or a column. To check the 8x8 matrix LED via Arduino need to connect both the rows and the columns. The LED lights up when the leg corresponding to its column is set to **HIGH** while on its line is set to **LOW** . If the column and the row are placed respectively at **HIGH** and **LOW** no voltage flowing through the LED and so this is not lights up .

Here are the typical connection diagram of a LED matrix 8x8. **COL** and **ROW** refer to the rows and columns while the numbering on the **PIN** is relative to its pinout.

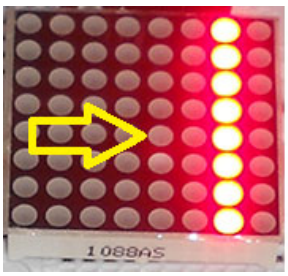


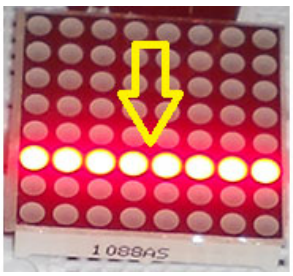
Pin Dot Matrix	(R) iga / (C) all unit Dot Matrix
1	R5
2	R7
3	C2
4	C3
5	R8
6	C5
7	R6
8	R3
9	R1
10	C4
11	C6
12	R4
13	C1
14	R2
15	C7
16	C8

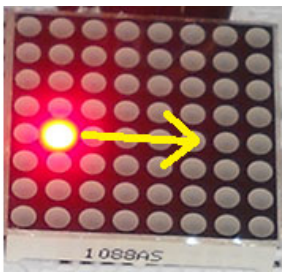
To control this device, without the aid of an appropriate integrated, well I use 16 ports Arduino as can be seen in the example below:

Source Code A

Project Objective : Build a circuit and a program that turns on the LED matrix in sequence as shown in Figure

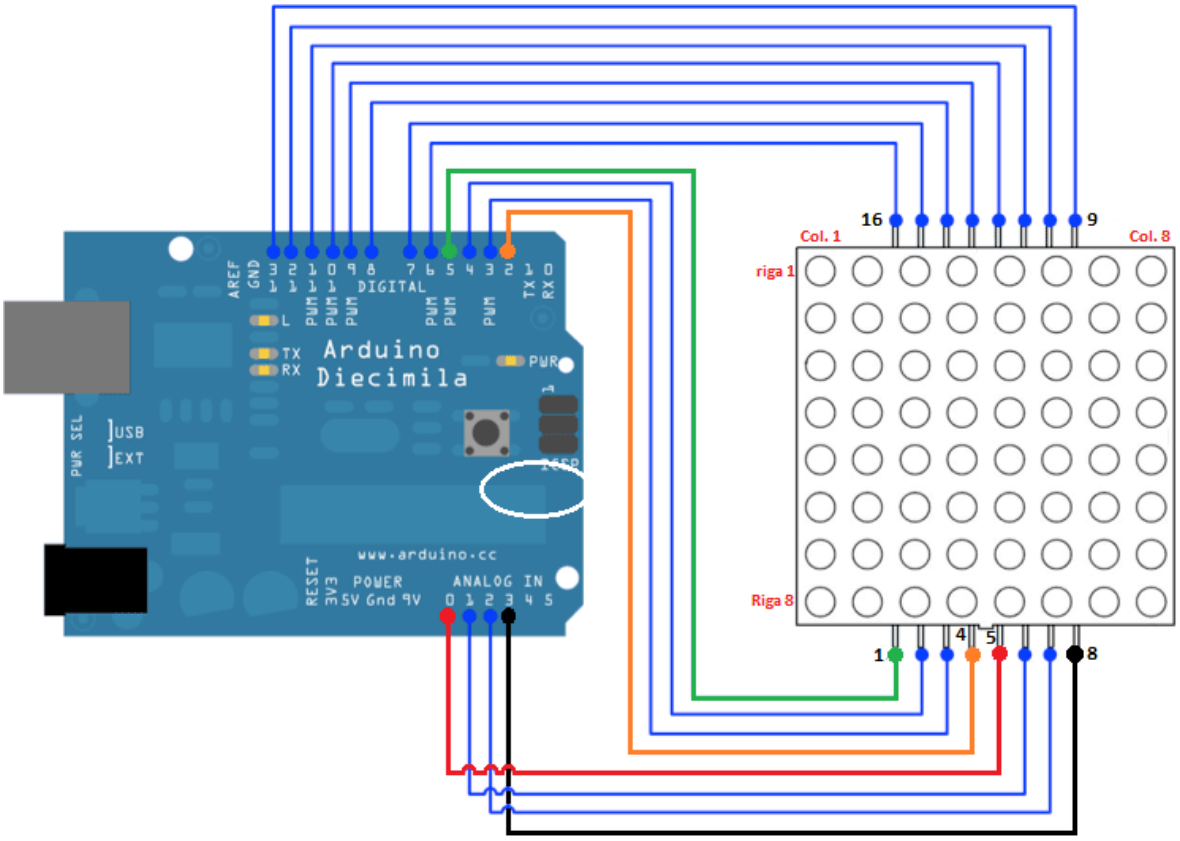






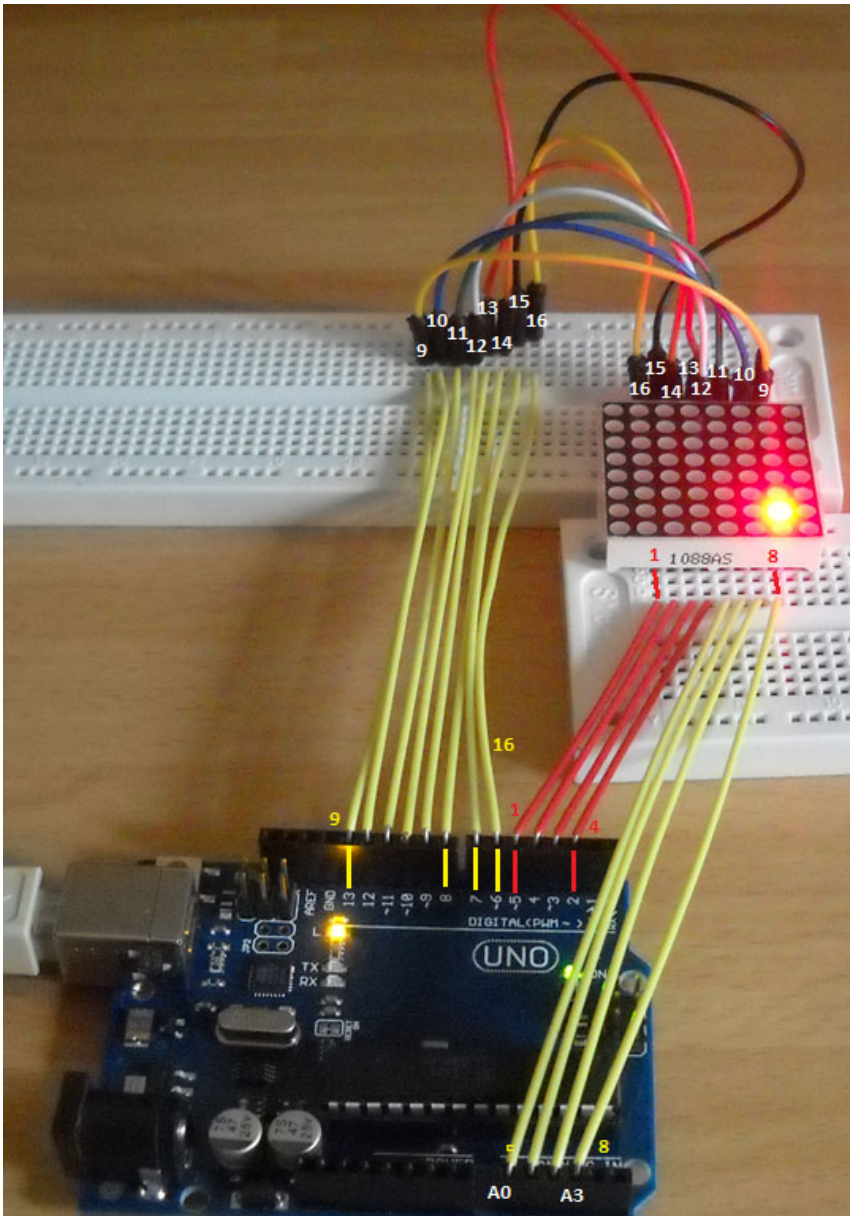
solution :

We build the following circuit

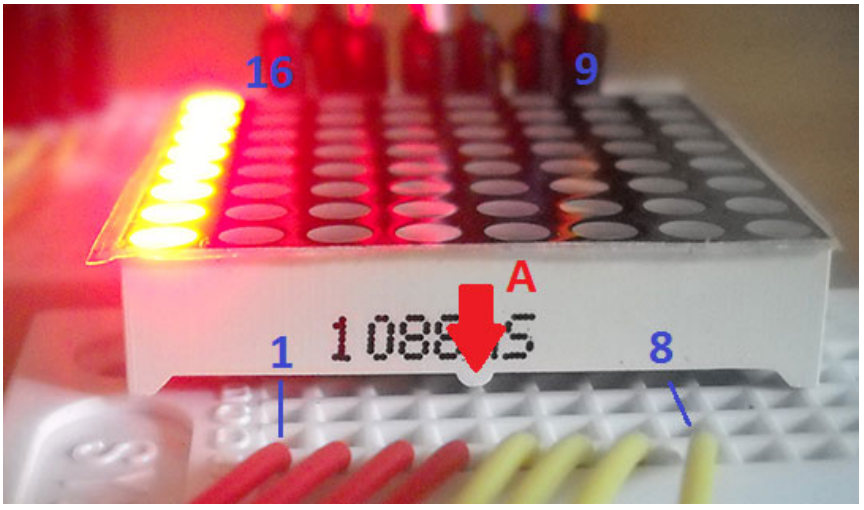


The connections are shown in the following table:

Pin Dot Matrix	Arduino Pin (D) igital / (A) nalogico	(R) iga / (C) all unit Dot Matrix
1	D5	R5
2	D4	R7
3	D3	C2
4	D2	C3
5	A0	R8
6	A1	C5
7	A2	R6
8	A3	R3
9	D13	R1
10	D12	C4
11	D11	C6
12	D10	R4
13	D9	C1
14	D8	R2
15	D7	C7
16	D6	C8



Note that the LED matrix has a protuberance on the side containing the series of pins which goes from 1 to 8.



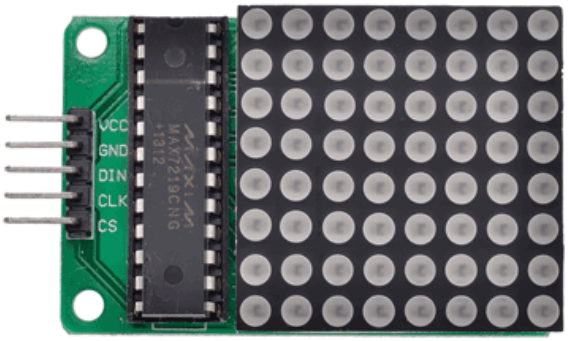
Here is the code associated

```
/ * Code that performs in series:
   - LEDs light column from left to right
   - To switch the LEDs on the line from top to bottom
   - Ignition sequence of the first led to the 64-th
* /
In the array // pin [0..16] (position 0 is fictitious and is used to start the array index 1)
// Is shown schematically the connection of the pin of the LED matrix with arduino pin.
In // pin [i] is added to the arduino pin connected to the i-th pin of the LED matrix.
// Please note that the arduino pin 14 ... 17 correspond respectively to:
// A0 => 14-th pin of arduino
```

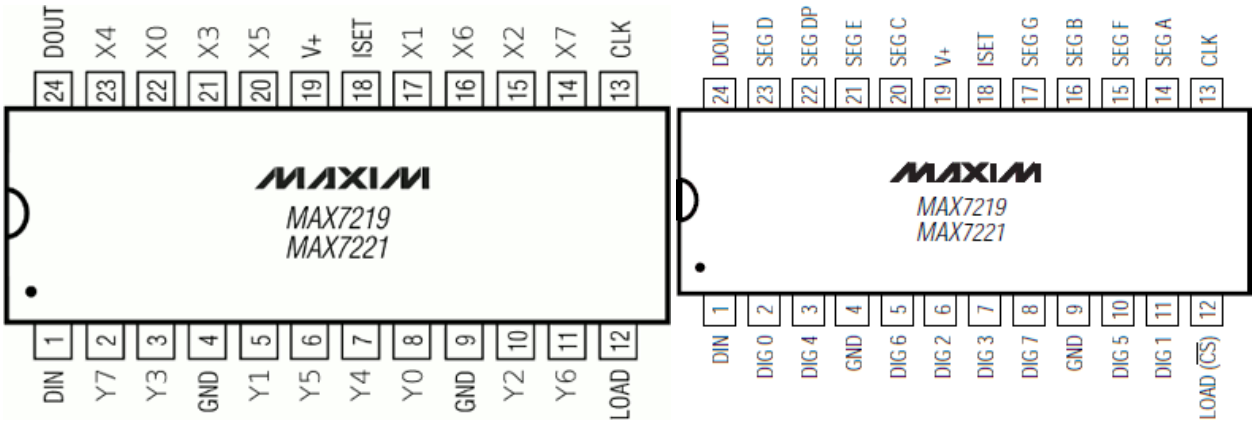
```
// A1 => 15-th pin of arduino
// A2 => 16-th pin of arduino
// A3 => 17-th pin of arduino
int pins [17] = {-1, 5, 4, 3, 2, 14, 15, 16, 17, 13, 12, 11, 10, 9, 8, 7, 6};

// The column "c" of the LED matrix is connected to the pin of arduino cols [c] which corresponds to
// Pin [j] where "j" is the pin of the LED matrix associated with the column "c"
int cols [8] = {pins [13], pins [3], pins [4], pins [10], pins [06], pins [11], pins [15], pins [16]};

// The line "r" of the LED array is connected to pin arduino rows [r] which corresponds to
// Pin [j] where "j" is the pin-matrix LED associated with the line "r"
int rows [8] = {pins [9], pins [14], pins [8], pins [12], pins [1], pins [7], pins [2], pins [5]};

/ * Assuming all the columns and rows to HIGH to LOW all the LEDs light * /
void AllLedON ()
{
  for (int i = 1; i <= 8, the ++)
  {
        digitalWrite (cols [i - 1], HIGH);
        digitalWrite (rows [i - 1], LOW);
  }
  delay (1000);
  AllLedOFF ();
}

/ *
   By putting all the columns and rows, respectively, to a different state from HIGH and LOW
   all the LEDs turn off
* /
void AllLedOFF ()
{
  for (int i = 1; i <= 8, the ++)
  {
        digitalWrite (cols [i - 1], LOW);
        digitalWrite (rows [i - 1], HIGH);
  }
}
/ * Turns the LED on the line king column C * /
ledon void (int r, int c)
{
  AllLedOFF ();
  digitalWrite (cols [c - 1], HIGH);
  digitalWrite (rows [r - 1], LOW);
  delay (100);
}

void setup ()
{
  // Set the Arduino pin as output pin
  for (int i = 1; i <= 16; i ++)
      pinMode (pins [i], OUTPUT);
  // Puts all the LED matrix OFF
  AllLedOFF ();
}

void loop ()
{
  int i, c, r;

  // Turns on the c-th column from left to right
  for (c = 1, c <= 8; ++ c)
  {
    for (r = 1, r <= 8; r ++)
       digitalWrite (rows [r - 1], LOW);
    digitalWrite (cols [c - 1], HIGH);
    delay (300);
    digitalWrite (cols [c - 1], LOW);
    for (r = 1, r <= 8; r ++)
       digitalWrite (rows [r - 1], HIGH);
  }

  // Lights the r-th row from top to bottom
  for (r = 1, r <= 8; r ++)
  {
    for (c = 1, c <= 8; ++ c)
       digitalWrite (cols [c - 1], HIGH);
    digitalWrite (rows [r - 1], LOW);
    delay (300);
    digitalWrite (rows [r - 1], HIGH);
    for (i = 1, i <= 8, the ++)
      digitalWrite (cols [i - 1], LOW);
  }

  // Turns a single LED from the first position to the 64-th
  for (r = 1, r <= 8; r ++)
    for (c = 1, c <= 8; ++ c)
      Ledon (r, c);
  AllLedOFF ();
  // AllLedON ();
}
```

INTEGRATED MAX7219

Analyzing the previous project you notice a considerable amount of pins. Instead, using the integrated **MAX7219** can drive all 64 LEDs by reducing the number of PIN used Arduino only 3 wires (plus those for the power supply).



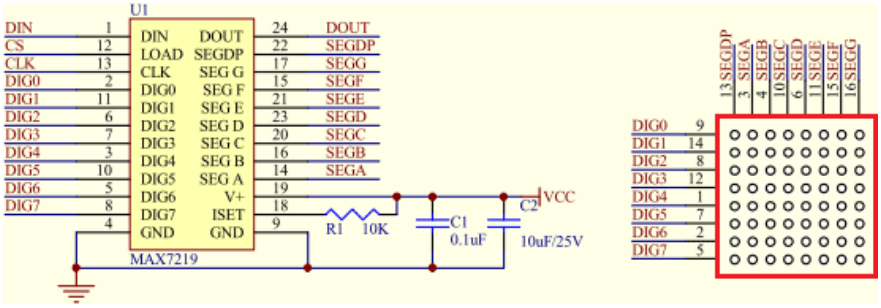
The pin of the chip **MAX7219** is as follows (see datasheet):



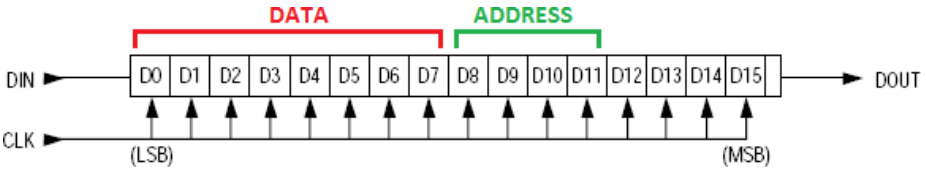
The meaning of the PIN is illustrated in the table below

PIN	CODE	MEANING
1	**DIN** = DataIn	**Serial-Data Input:** the data is loaded into the shift register (shift register) extension on the rising edge of **CLK**
2, 3, 5-8, 10, 11	**Y0** ... **Y7** or **DIG7** ... **DIG0**	Pin connected to the lines. The **MAX7219** port these outputs to V + when turned off
4, 9	**GND**	2 and all the pins must be connected to earth
12	**LOAD** - **$\overline{\text{CS}}$**	When this pin is set to **LOW** (Chip Select Input) the serialized data is loaded into the shift register. The last 16-bit serial input **DataIn** are blocked on the rising edge of pin **LOAD** .
13	**CLK**	**Serial-Clock Input.** The maximum permissible frequency is 10MHz .. The bits sent (in packets of 16 bits) to pin **DataIn** are shifted into the shift register at each rising edge of the **CLK** regardless of the state of pin **LOAD** . On the falling edge of **CLK** , the data is replicated on pin **DOUT** . The input **CLK** pin is only active when **CS** is LOW.
14-17, 20-23	**SEGA -SEGG, SEGDP or X0 ... X7**	Pin connected to the columns. In the **MAX7219** , when a segment is off the pin is set to **GND** .
18	The **SSET**	E 'connected to V_{DD} through a resistor (**RSET**) to set the current peak on the segments **SEGA** .. **ET SEQ** and **DP** . The brightness of the display can be controlled with an external resistor **RSET** connected between V + and ISET. The peak current in the segments (**SEGx** and DP) is nominally 100 ISET. **RSET** can be fixed or variable but must be ≥ 9.53 Ω.
19	V +	Pin connected to +5 V
24	DOUT	**Serial-Data Output.** serialized The data on **DIN** will be repeated at **DOUT** 16.5 clock cycles after the falling edge of CLK. This pin is used to put in different series **MAX7219** .

That are connected to the LED matrix in this way



The chip **MAX7219** has a 16-bit register is divided into 2 parts: **ADDRESS** and **DATA** , each of 8 bits. The data format is as follows:

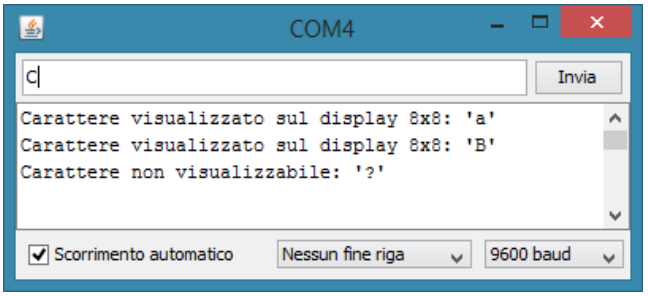


Bits **D12-D15** have no special meaning and are used for scrolling. Bits **D8 ... D11** are used to address the internal registers of the **MAX7219** and correspond to the commands available in the chip. These commands are described in the table below:

Address	Meaning
0x00	**NOP** : no operation
0x01	Digit 0
0x02	Digit 1
...	...
0x08	Digit 7
0x09	**DECODE MODE** - defines, for each digit, when using the BCD code B or code no.
0x0A	**INTENSITY Register** - Sets the intensity of the brightness of the LED. The values range from 0 to 15.
0x0B	**SCAN_LIMIT Register** - Sets the number of bits used
0x0C	**Register SHUTDOWN** - When the **MAX7219** is in shutdown mode the oscillator is stopped, the driver of the segments (**SEGY**) are placed in mass while those of the display (**DIGx**) to V +, so turning off all LEDs. data in the registers remain unchanged and the display driver can be programmed. The shutdown mode can be used to save power or to flash the display.
0x0F	**DISPLAY_TEST Register** - if set to 1, all the LEDs are lit

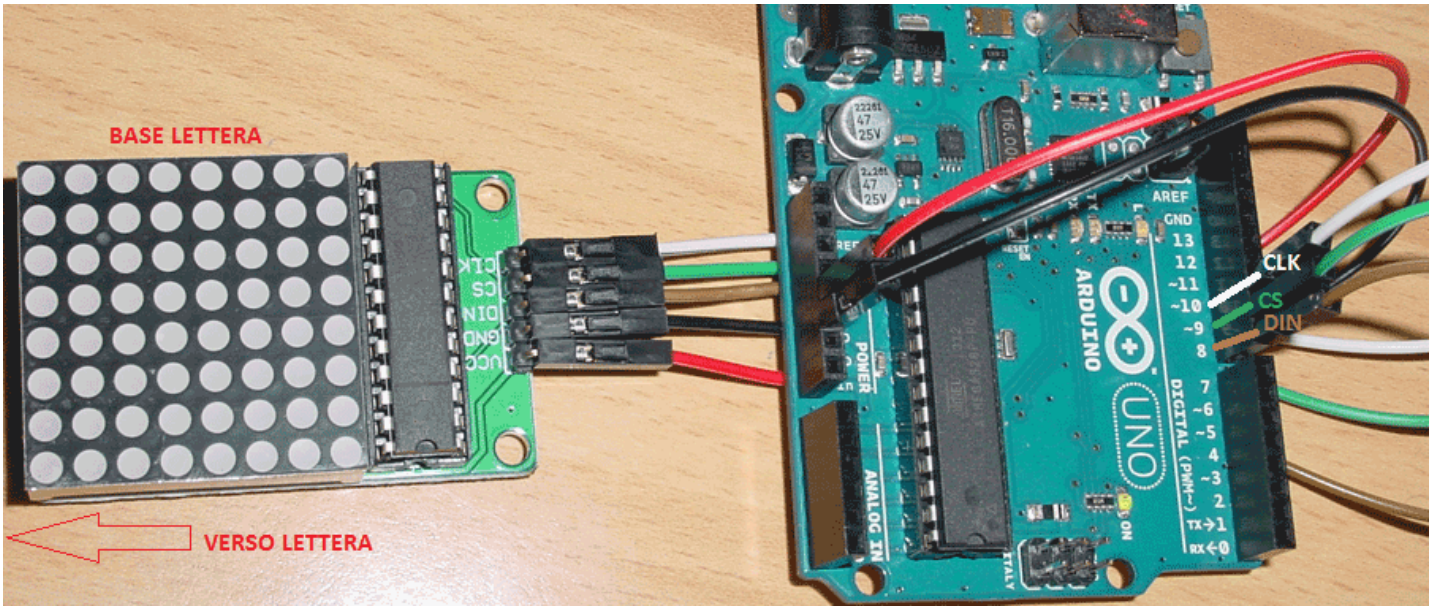
Source Code B

Project Objective : The objective of our project is to display on the LCD 8x8 alphanumeric characters (**0..9** , **a ... z** , **A ... Z**) typed on the PC keyboard. If you enter an invalid character, the display LEDs are all off. At the same time on the Serial Monitor will instead receive a message that indicates the character typed or an error notice if the symbol is not alphanumeric.



solution :

We set the following circuit visible in the 'picture below



Here is the code associated with our project. The most interesting thing about this code is that it does not use any external libraries.

```
/ *
 Example created taking inspiration from the example at this address:
 http://linksprite.com/wiki/index.php5?title=LED_Matrix_Kit
 Displays an alphanumeric character on the LED display 8x8
* /
Max7219_pinCLK int = 10; // On arduino connected to the CLK pin of MAX7219
int Max7219_pinCS = 9; // Pin on arduino connected to the DAC MAX7219
int Max7219_pinDIN = 8; // On arduino connected to the DIN pin of MAX7219

DISP1 unsigned char [38] [8] =
{
  {B00111100, B01111110, B01100110, B01100110, B01100110, B01100110, B01111110, B00111100}, // 0.
  {B00011000, B00111000, B00011000, B00011000, B00011000, B00011000, B00111100, B00111100}, // 1
  {B00111100, B01111110, B01100110, B00001110, B00011000, B00110000, B01111110, B01111110}, // 2
  {B00111100, B01111110, B01100110, B00001110, B00001100, B01100110, B01111110, B00111100}, // 3
  {B00001110, B00011110, B00111110, B01110110, B01100110, B01111111, B00111111, B00000110}, // 4
  {B01111110, B01111110, B01100000, B01111100, B00011110, B01000110, B01111110, B00111110}, // 5
  {B00111110, B01111110, B01100000, B01111100, B01111110, B01100110, B01111110, B00111100}, // 6
  {B01111110, B01111110, B00000110, B00001110, B00011100, B00111000, B01110000, B01100000}, // 7
  {B00111100, B01111110, B01100110, B00111100, B01100110, B01100110, B01111110, B00111100}, // 8
  {B00111100, B01111110, B01100110, B01111110, B00111110, B00000110, B01111110, B00111100}, // 9
  {B01111110, B11111111, B11000011, B11000011, B11111111, B11111111, B11000011, B11000011}, // A
  {B11111110, B11111111, B11000011, B11111111, B11111110, B11000011, B11111111, B11111110}, // B
  {B01111110, B11111111, B11000011, B11000000, B11000000, B11000011, B11111111, B01111110}, // C
  {B11111100, B11111110, B11000111, B11000011, B11000011, B11000111, B11111110, B11111100}, // D
  {B11111111, B11111111, B11000000, B11110000, B11110000, B11000000, B11111111, B11111111}, // E //
  {B11111111, B11111111, B11000000, B11110000, B11110000, B11000000, B11000000, B11000000}, // F //
  {B01111110, B11111110, B11000000, B11001110, B11001111, B11000011, B11111110, B01111110}, // G //
  {B11000011, B11000011, B11000011, B11111111, B11111111, B11000011, B11000011, B11000011}, // H //.
  {B00111100, B00011000, B00011000, B00011000, B00011000, B00011000, B00011000, B00111100}, // I //
  {B00111100, B00011000, B00011000, B00011000, B00011000, B11011000, B11111000, B01110000}, // J //
  {B11001110, B11011100, B11111000, B11110000, B11111000, B11011100, B11001110, B11000111}, // K //
  {B01100000, B01100000, B01100000, B01100000, B01100000, B01100000, B01111110, B01111110}, // L //
  {B11000011, B11100111, B11111111, B11111111, B11011011, B11000011, B11000011, B11000011}, // M //
  {B11000011, B11100011, B11110011, B11111011, B11011111, B11001111, B11000111, B11000011}, // N //
  {B01111100, B11111110, B11000110, B11000110, B11000110, B11000110, B11111110, B01111100}, // O //
  {B01111110, B01111111, B01100011, B01111111, B01111110, B01100000, B01100000, B01100000}, // P //
  {B01111100, B11111110, B11000110, B11000110, B11001110, B11001110, B11111110, B01111011}, // Q //
  {B01111110, B01111111, B01100011, B01111111, B01111110, B01111100, B01101110, B01100111}, // R //
  {B01111110, B11111110, B11100000, B11111000, B00111110, B00001110, B11111110, B11111100}, // S //
  {B11111111, B11111111, B00011000, B00011000, B00011000, B00011000, B00011000, B00011000}, // T //
  {B11000110, B11000110, B11000110, B11000110, B11000110, B11000110, B11111110, B01111100}, // U //
  {B11000011, B11000011, B01100110, B01100110, B01100110, B00111100, B00111100, B00011000}, // V //
  {B11000011, B11011011, B11111111, B11111111, B01111110, B01100110, B01100110, B00100100}, // W //
  {B11000011, B01100110, B00111100, B00011000, B00111100, B01100110, B11000011, B11000011}, // X //
  {B01100110, B01100110, B01100110, B01111110, B00111100, B00011000, B00011000, B00011000}, // Y //
  // {0xFF, 0xFF 0x0E, 0x1C, 0x38,0x70,0xFF, 0xFF}, // Z in Hex
  {B11111111, B11111111, B00001110, B00011100, B00111000, B01110000, B11111111, B11111111}, // Z
  {B00000000, B00000000, B00000000, B00000000, B00000000, B00000000, B00000000, B00000000}, // All off [36]
  {B11111111, B11111111, B11111111, B11111111, B11111111, B11111111, B11111111, B11111111} // Full On, [37]
};

Write_Max7219_byte void (unsigned char DATA)
{
    unsigned char i;
    for (i = 1, i <= 8, the ++)
    {
       digitalWrite (Max7219_pinCLK, LOW);
       digitalWrite (Max7219_pinDIN, DATE & 0x80); // Pull out the bit in position 8
       DATA = DATA << 1; // Shift to the left of a place the bits in DATA
       digitalWrite (Max7219_pinCLK, HIGH); // Arc edge of the CLK is the DIN
                                            // Shifted within the shift register
    }
}

Write_Max7219 void (unsigned char address, unsigned char dat)
```

```
{
    digitalWrite (Max7219_pinCS, LOW); // I enable writing to the Data IN
    Write_Max7219_byte (address); // Register address / command to execute (bit D8 .. D11)
    Write_Max7219_byte (dat); // Bit to put in D0..D7 (representing the value used by the command)
    digitalWrite (Max7219_pinCS, HIGH); // Conclude by writing to the Data IN
}

Init_MAX7219 void (void)
{
   Write_Max7219 (0x09, 0x00); // Decoding: BCD
   Write_Max7219 (0x0a, 0x0F); // Brightness 1..15
   Write_Max7219 (0x0b, 0x07); // Boundary scan, I run all 8 lines
   Write_Max7219 (0x0c, 0x01); // Shutdown command: Display off: 0x00 - ON (normal mode): 0x01
   Write_Max7219 (0x0f, 0x00); // Test command: Test Mode (all LEDs on): 0x01 - Normal mode: 0x00)
}

void setup ()
{
    // Septum pins as output
    pinMode (Max7219_pinCLK, OUTPUT);
    pinMode (Max7219_pinCS, OUTPUT);
    pinMode (Max7219_pinDIN, OUTPUT);
    delay (50);
    Init_MAX7219 ();
    Serial.begin (9600);
}

void loop ()
{

   int value, j, i;
   String phrase = "";
   if (Serial.available ()) // if I font on the serial monitor
   {
       Value = Serial.read (); // Read the data on the Arduino IDE Serial Monitor
       if (value <= '9' && value> = '0') // I establish the position in the vector
          Value- j = (int) '0';
       else if (value <= 'Z' && value> = 'A')
          Value- j = (int) 'A' + 10;
       else if (value <= 'z' && value> = 'a')
          Value- j = (int) 'a' + 10;
       else
          j = 36;
       for (i = 0; i <8, the ++)
          Write_Max7219 (i + 1, DISP1 [j] [i]); // Write l '(i + 1) th row of the bit map
       if (j> = 36)
          Sentence = "Character can not be displayed: '" + String ((char) value) + "'";
       else
          Sentence = "font on the display 8x8: '" + String ((char) value) + "'";
       Serial.println (Phrase);
  }
}
```

To generate the bitmap to be loaded on the 8x8 display we can use the following javascript program

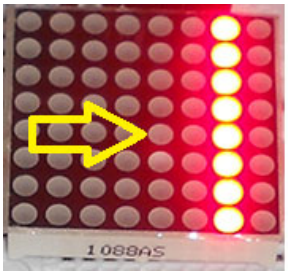
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0

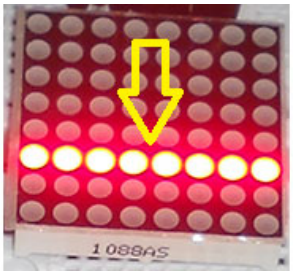
Reset

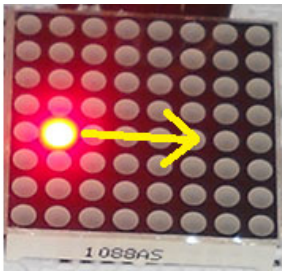
Binary Encoding:

Source Code C

Objective of the project : Building a program equivalent to the first example using the library **LEDCONTROL** . Therefore, on the 8x8 matrix we need to see this execution

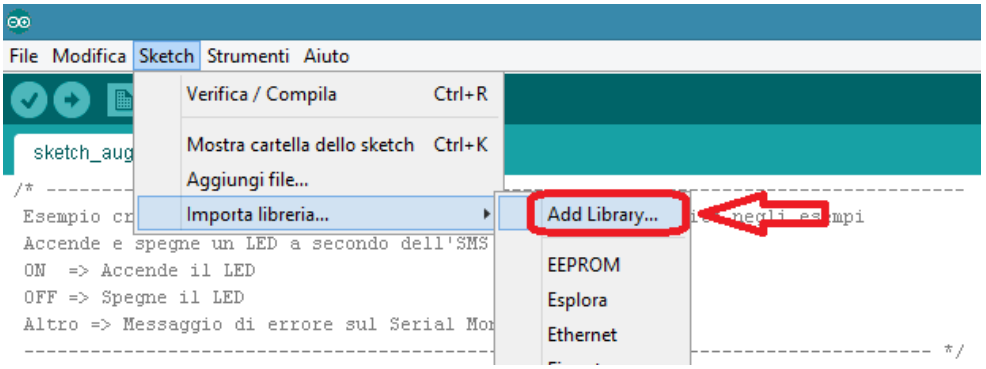






solution :

In order to use this code you need to install the library **LEDCONTROL** available at this link . To install it just unzip the zip file and copy the folder produced inside the folder containing the libraries of Arduino (usually **C: \ Program Files \ Arduino \ libraries**). Finish by adding the newly copied folder, using the menu " *Add Library* "within the IDE of Arduino



after you import the files are copied to the folder **C: \ Users \ <User> \ Documents \ Arduino \ libraries** . If there are problems with the installation you need to delete the folder just copied the library and retry the operation.

Here is the code associated with our project.

```
# include <LedControl.h>
/ *
 We set the PIN used:
 Pin 8: Connected to pin DataIn
 Pin 10: connected to the CLK
 Pin 9: connected to the LOAD / CS
 With the last parameter to set the nr LedControl display of used 8x8
 * /
LedControl lc = LedControl (8,10,9,1);

/ * Turns on all LED DISPLAY * /
void AllLedON ()
{
  for (int r = 0, r <8; ++ r)
    lc.setRow (0, r, B11111111);
  delay (1000);
  lc.clearDisplay (0);
}

void setup ()
{
  / *
   During the construction of the object LEDControl MAX72XX is the place
   in "power-saving". We therefore need to "wake up" the integrated
  * /
  lc.shutdown (0, false); // 0 is relative to where MAX72XX act
                        // Equivalent to "Write_Max7219 (0x0c, 0x01);"
  lc.setIntensity (0.15); // Brightness 1..15
  lc.setScanLimit (0.7); // Boundary scan, I run all 8 lines
                          // Not necessary: the library puts on initialization
                          // Object LEDControl this value to the maximum (7)
  lc.clearDisplay (0);
  // Decoding BCD and setting the "Display Test Register" are not available
  // As set directly in the class constructor LCDControl
}

void loop ()
{
  int r, c;
  lc.clearDisplay (0);

  // Turns on the c-th column from left to right
  for (c = 0, c <8 c ++)
  {
     lc.setColumn (0, c, B11111111);
     delay (300);
     lc.setColumn (0, c, (byte) 0); // Turn off the column
  }

  // Lights the r-th row from top to bottom
  for (r = 0, r <8; r ++)
  {
     lc.setRow (0, r, B11111111);
     delay (300);
     lc.setRow (0, r, (byte) 0); // Turn off the line
  }

  // Turns a single LED from the first position to the 64-th
  for (r = 0, r <8; r ++)
    for (c = 0, c <8 c ++)
    {
      lc.setLed (0, r, c, true); // I turn on the LEDs
      delay (100);
      lc.setLed (0, r, c, false); // Turn off the single LED
    }

  // Turn on all the LEDs
  // AllLedON ();
}
```