**GROUP 17**

**EL213 ANALOG CIRCUITS**

LED MATRIX DISPLAY

INSTRUCTOR: -Dr. RUTU PAREKH

* **GroupMembersDetails:-**

1. Gamita Banaphal-201401101

gamitabanaphal2751@gmail.com

1. Archana Bhusura-201401172

[archana.bhusura84@gmail.com](mailto:archana.bhusura84@gmail.com)

1. Rahul Parmar-201401166

[rahul.parmar.58173000@gmail.com](mailto:rahul.parmar.58173000@gmail.com)

1. Hardik Ghusar-201401167

[ghusarhardikr@gamil.com](mailto:ghusarhardikr@gamil.com)

1. Dhaval Prajapati-201401138

[dhavalprajapati12015@gmail.com](mailto:dhavalprajapati12015@gmail.com)

1. Bhawana Jaloria-201401043

[bhawana.daiict@gmail.com](mailto:bhawana.daiict@gmail.com)

1. Dimple Dodia-201401152

[dodia123.dd@gmail.com](mailto:dodia123.dd@gmail.com)

1. Drashti Bharti-201401173

drashti.bharti96@gmail.com

1. Ishan Changela-201401447

[icishanchangela@gmail.com](mailto:icishanchangela@gmail.com)

10-Mayur Detroja-201401459

detrojamayur184@gmail.com

* INDEX

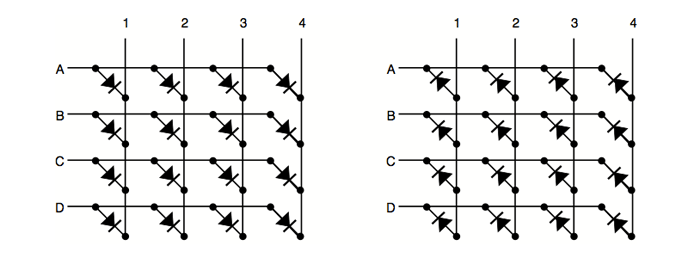
1. Introduction
2. Role of components
3. Circuit diagram
4. Working
5. Application
6. Bibliography

* INTRODUCTION

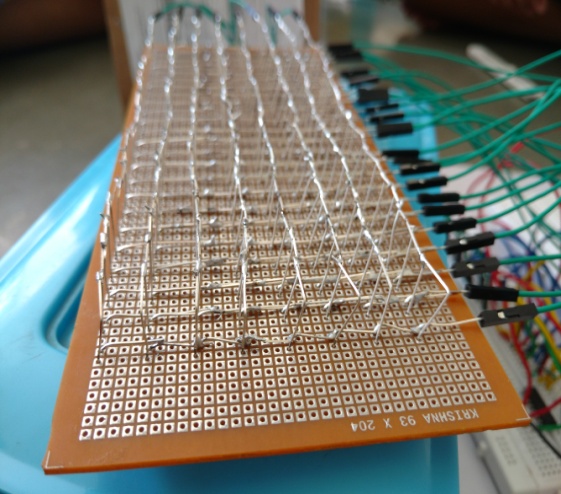
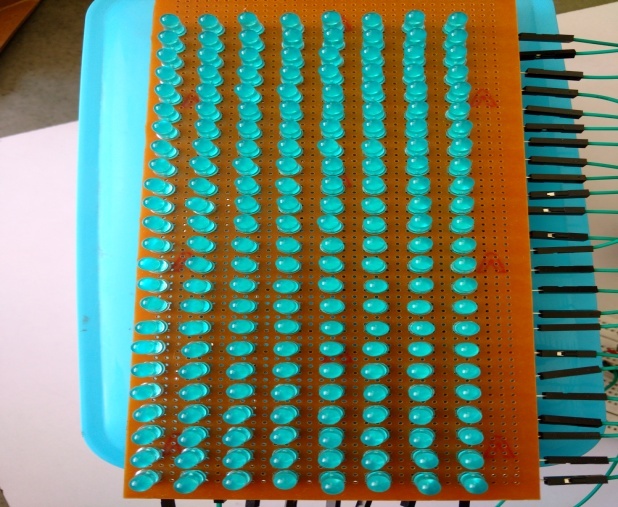
LED–based signage and matrix displays are bringing new dimensions of versatility and eye-pleasing effects to growing number of outdoor and indoor application. It is a flat panel display, which uses an array of light emitting diodes. LED displays are capable of providing general illumination in addition to visual display, as when used for stage lighting or other decorative purposes. There are numerous other applications some of them are explained later.

There are two types of LED matrix

* Common-row anode (left)
* Common-row cathode(right)



The difference between these two configurations is how LED is lit. With common-row anode current sources (positive voltage) are attached to rows A..D and negative voltage (ground) to columns 1..4 With common-row cathode negativevoltages are attached to rows A..D and currents sources to columns 1..4.



We made our 24x8 LED Matrix with 192 5mm blue LEDs, which operates on 3.4 V forward voltage and 30mA forward current. Our matrix configuration is common-row cathode, in which we connected all anode in a columns together by bending them and we connected all the cathodes in a row by bridging them over the anodes at some height.

* Components
* Our project comprises of following components :-

1. 192 LEDs
2. 3 x 74HC595 shift registers
3. 24 x 100 ohm resistors
4. 1 x Arduino board
5. 8 x 1k ohm resistors
6. 8 x 2N3904 transistors
7. 1 x 4017 decade counter

* Role of Components

1. Arduino :-

Arduino is a platform based on easy-to-use hardware and software. Information is displayed on the 24x8 led matrix using arduino.First, code is uploaded in the arduino through the computer. Based on this code arduino gives serial input to shift register.Provide clock cycles to shift register and counter .It is also used as source of voltage and ground for the circuit.

1. Role of shift register:-

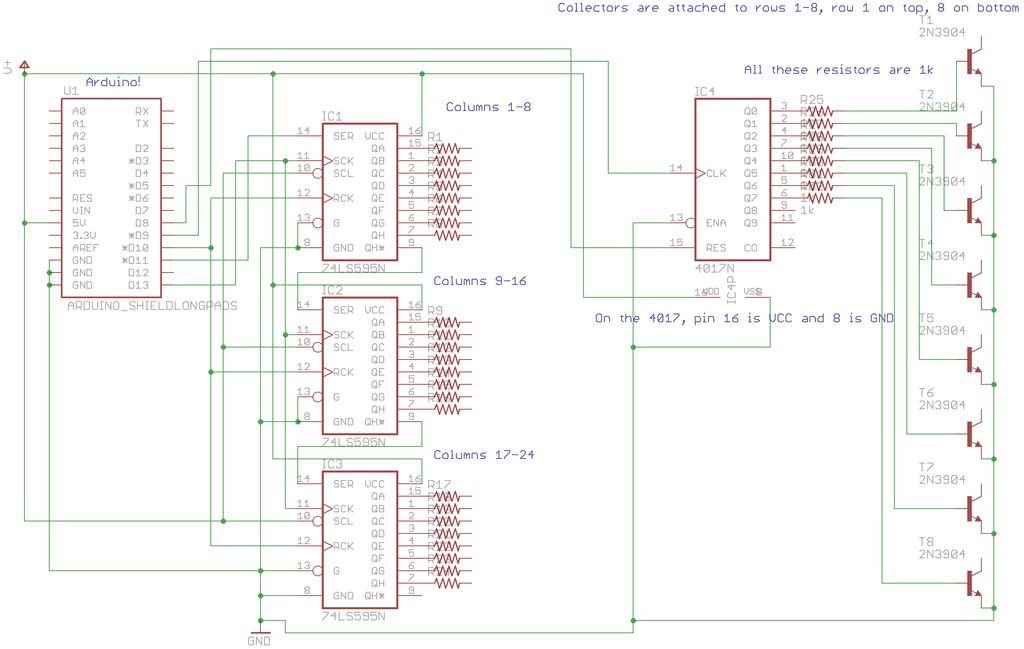
The use of shift registers minimizes the number of I/O pins required to drive the columns of the LED matrix. For driving 24 columns separately, we need 24 I/O pins of arduino, however, with the use of 74HC595 ICs, this number is reduced to 3. 74HC595 is an 8-stage serial-in, serial or parallel-out shift register, with a storage register. The shift register and storage register have separate clocks: SH\_CP (pin 11) and ST\_CP (pin 12). Data is fed serially into the register through DS pin (14) and is shifted on the positive-going transitions of the SH\_CP input.

1. Role of counter :-

Eight I/O pins are required to scan 8 rows in sequence. A port expander, such as CD4017 (counter), can be used for this purpose which uses only two I/O pins of arduino. The counter is cleared to zero count by a logical “1” on its reset line (15). The counter is advanced on the positive edge of the clock signal (pin 14), when the clock inhibit (pin 13) is grounded. The 8 decoded outputs are normally in the logical “0” state and go to the logical “1” state only at their respective time slot. Each decoded output remains high for 1 full clock cycle. The carry-out signal completes a full cycle for every 8 clock input cycles and is used as a ripple carry signal to any succeeding stages. The 8 rows of LED matrix are sequentially connected to the decoded outputs, Q0- Q7, of CD4017 through 8 transistors each of which provides a ground path to sink the combined current of all LEDs in a row. At the end of every 8th clock cycle, the arduino will reset the counter by issuing a logical “1” to its Reset pin (15)

1. 29304 Transistor :-

It is used as Low Side Switch.Since the cathode part of the matrix should be given low voltage and anode part needs high voltage so that it can blink properly. Transistor is used to control the supply for cathode. Base part of the transistor take input from the Counter.

* CIRCUITDIAGRAM
* WORKING :-

To display the alphabet A(any and every character will occupy 8x8 grid), first the row R1 is selected (which means R1 is pulled low in this case), and deselect other rows by blocking their ground paths (one way of doing that is by pulling R2 through R8 pins to logic high). Now, the first row is active, and the LEDs in the column C2 through C7 of this row are to be turned on, which can be done by applying forward bias voltages to these column. Next, select the row R2 (and deselect all other rows), and apply forward bias to C1 and C8, and so on. Therefore, by scanning across the row quickly **(> 100 times per second)**, and turning on the respectiveLEDs in each column of that row, the persistence of vision comes in to play, and we perceive the display image as still.

To display any information using Arduino uno, we have to program the required information in the software for Arduino coding. The code is uploaded in the ardunio which then controls the displayof information.

In the code, firstly we defined all the symbols in 8 rows and 8 columns of bits. When we give input from the list of symbols defined above, it will call (display\_word ()) function to display that symbol to led matrix. The display\_word () function will take first row of bits defined for that symbol and it will give all bits serially as input for shift register. We used 74HC595 IC (shift register). It is done by connecting pin D9 of arduino to the input pin of the shift register. Above process executed for the first row of bits since there are 8 rows and to see character properly we have to blink all 8 rows according to bits of symbols, for that CD4017counter is used. Counter will provide voltage to first row of led matrix for given clock cycle while not providing voltage to other rows, then when the next positive edge of clock comes it will provide voltage to second row of led matrix (not providing voltage to the rest of the rows and so for the rest of the row.) and arduino code makes that clock cycle for 2ms by doing that all 8 rows will blink in 16 ms. This happens so fast that human eye can’t catch and it looks like all LEDs are blinking at same time. After 16ms arduino will send signal to reset pin of the counter then the above process repeats.

For the scrolling effect, the symbols are shifted according to the time defined, clock cycle is provided by arduino to all shifts registers, then the output of the shift registers is shifted to next column and the first column is provided with new input by arduino. In order to shift output of first shift register, output of last column of last shift register is deleted.

Here outputs of shift registers and counter are given to anodes and cathodes of LEDs respectively. In order to blink LEDs properly voltage at anodes must be higher than voltage at cathodes, so outputs of counters are followed by 1k registers and NPN transistors. This configuration will act as low side switch and it will provide low voltage to cathode of LEDs irrespective of output voltage of counter. Since output voltage of shift registers will be near 5V and LED’s forward voltage is 3.5V so we have connected 100ohm registers between LEDs and outputs of shift registers.

* Arduino code

#define A{B00111100,B01000010,B01000010,B01000010,B01111110,B01000010,B01000010,B01000010}

#define B{B01111100,B01000010,B01000010,B01111100,B01000010,B01000010,B01000010,B01111100}

#define C{B00111100,B01000010,B01000000,B01000000,B01000000,B01000000,B01000010,B00111100}

#define D{B01111000,B01000100,B01000010,B01000010,B01000010,B01000010,B01000100,B01111000}

#define E{B01111110,B01000000,B01000000,B01111100,B01000000,B01000000,B01000000,B01111110}

#define F{B01111110,B01000000,B01000000,B01111100,B01000000,B01000000,B01000000,B01000000}

#define G{B00111100,B01000010,B01000000,B01000000,B01001110,B01000010,B01000010,B00111100}

#define H{B01000010,B01000010,B01000010,B01111110,B01000010,B01000010,B01000010,B01000010}

#define I{B01111100,B00010000,B00010000,B00010000,B00010000,B00010000,B00010000,B01111100}

#define J{B00111110,B00001000,B00001000,B00001000,B00001000,B00001000,B01001000,B00110000}

#define K{B01000100,B01001000,B01010000,B01100000,B01100000,B01010000,B01001000,B01000100}

#define L{B01000000,B01000000,B01000000,B01000000,B01000000,B01000000,B01000000,B01111110}

#define M{B10000010,B11000110,B10101010,B10101010,B10010010,B10010010,B10000010,B10000010}

#define N{B01000010,B01100010,B01010010,B01010010,B01001010,B01001010,B01000110,B01000010}

#define O{B00111100,B01000010,B01000010,B01000010,B01000010,B01000010,B01000010,B00111100}

#define P{B01111100,B01000010,B01000010,B01000010,B01111100,B01000000,B01000000,B01000000}

#define Q{B00111100,B01000010,B01000010,B01000010,B01000010,B01000010,B01000100,B00111010}

#define R{B01111100,B01000010,B01000010,B01111100,B01010000,B01001000,B01000100,B01000010}

#define S{B00111100,B01000010,B01000000,B00110000,B00001100,B00000010,B01000010,B00111100}

#define T{B11111110,B00010000,B00010000,B00010000,B00010000,B00010000,B00010000,B00010000}

#define U{B01000010,B01000010,B01000010,B01000010,B01000010,B01000010,B01000010,B00111100}

#define V{B01000100,B01000100,B01000100,B01000100,B01000100,B01000100,B00101000,B00010000}

#define W{B10000010,B10000010,B10000010,B10000010,B10010010,B10010010,B10101010,B01101100}

#define X{B10000010,B10000010,B01000100,B00101000,B00010000,B00101000,B01000100,B10000010}

#define Y{B10000010,B10000010,B01000100,B00101000,B00010000,B00010000,B00010000,B00010000}

#define Z{B01111110,B00000010,B00000100,B00001000,B00010000,B00100000,B01000000,B01111110}

#define full{B11111111,B11111111,B11111111,B11111111,B11111111,B11111111,B11111111,B11111111}

#define guk{B00111000,B01111100,B01000000,B00100000,B01111100,B00001000,B00000100,B01111100}

#define gum{B00000000,B11100100,B00100100,B00100100,B01111100,B00100100,B00000100,B00000100}

#define guch{B00011000,B00000100,B01100001,B10000001,B01101101,B10001101,B01111001,B00000001}

#define LA{B00000000,B00000000,B00000000,B00111000,B00000100,B00111100,B01000100,B00111000}

#define LB{B01000000,B01000000,B01000000,B01000000,B01111000,B01000100,B01000100,B01111000}

#define LC{B00000000,B00000000,B00000000,B00111100,B01000000,B01000000,B01000000,B00111100}

#define LD{B00000100,B00000100,B00000100,B00000100,B00111100,B01000100,B01000100,B00111100}

#define LE{B00000000,B00000000,B00000000,B00111000,B01000100,B01011000,B01000000,B00111100}

#define LF{B00111000,B01000100,B01000000,B01000000,B01111000,B01000000,B01000000,B01000000}

#define LG{B00111000,B01000100,B01000100,B01000100,B00111000,B00000100,B00000100,B01111000}

#define LH{B01000000,B01000000,B01000000,B01111000,B01000100,B01000100,B01000100,B01000100}

#define LI{B00000000,B00000000,B00010000,B00000000,B00010000,B00010000,B00010000,B00010000}

#define LJ{B00000000,B00001000,B00000000,B00001000,B00001000,B00001000,B01001000,B00110000}

#define LK{B00000000,B01000000,B01000000,B01001000,B01010000,B01100000,B01010000,B01001000}

#define LL{B00100000,B00100000,B00100000,B00100000,B00100000,B00100000,B00100000,B00011000}

#define LM{B00000000,B00000000,B00000000,B01000000,B01101100,B01010100,B01000100,B01000100}

#define LN{B00000000,B00000000,B00000000,B01000000,B01111000,B01000100,B01000100,B01000100}

#define LO{B00000000,B00000000,B00000000,B00111000,B01000100,B01000100,B01000100,B00111000}

#define LP{B00000000,B00000000,B00111000,B01000100,B01000100,B01111000,B01000000,B01000000}

#define LQ{B00000000,B00000000,B00111000,B01000100,B01000100,B00111100,B00000100,B00000100}

#define LR{B00000000,B00000000,B00111000,B01000100,B01000000,B01000000,B01000000,B01000000}

#define LS{B00000000,B00000000,B00111000,B01000100,B01100000,B00011100,B01000100,B00111000}

#define LT{B00100000,B00100000,B00100000,B01111000,B00100000,B00100000,B00100000,B00011000}

#define LU{B00000000,B00000000,B00000000,B01000100,B01000100,B01000100,B01000100,B00111100}

#define LV{B00000000,B00000000,B00000000,B01000100,B01000100,B01000100,B00101000,B00010000}

#define LW{B00000000,B00000000,B00000000,B01000100,B01000100,B01010100,B01010100,B00111000}

#define LX{B00000000,B00000000,B00000000,B01000100,B00101000,B00010000,B00101000,B01000100}

#define LY{B00000000,B00000000,B00000000,B01000100,B01000100,B00111100,B00000100,B01111000}

#define LZ{B00000000,B00000000,B00000000,B01111100,B00000100,B00111000,B01000000,B01111100}

#define SPACE{B00000000,B00000000,B00000000,B00000000,B00000000,B00000000,B00000000,B00000000}

#define NUM0{B00111100,B01000010,B01000110,B01001010,B01010010,B01100010,B01000010,B00111100}

#define NUM1{B00011000,B00111000,B00011000,B00011000,B00011000,B00011000,B00011000,B00111100}

#define NUM2{B00111100,B01000010,B00000010,B00000010,B00001100,B00110000,B01000000,B01111110}

#define NUM3{B00111100,B01000010,B00000010,B00000010,B00011100,B00000010,B01000010,B00111100}

#define NUM4{B01000010,B01000010,B01000010,B01000010,B00111110,B00000010,B00000010,B00000010}

#define NUM5{B01111110,B01000000,B01000000,B01111100,B00000010,B00000010,B01000010,B00111100}

#define NUM6{B01111110,B01000000,B01000000,B01000000,B01111110,B01000010,B01000010,B01111110}

#define NUM7{B01111110,B00000010,B00000100,B00000100,B00001000,B00001000,B00010000,B00010000}

#define NUM8{B00111100,B01000010,B01000010,B01111110,B01000010,B01000010,B01000010,B00111100}

#define NUM9{B01111110,B01000010,B01000010,B01111110,B00000010,B00000010,B01000010,B01111110}

#define DEVIDE{B00000100,B00000100,B00001000,B00001000,B00010000,B00010000,B00100000,B00100000}

#define TWODOTS{B00000000,B00011000,B00011000,B00000000,B00000000,B00011000,B00011000,B00000000}

#define DOT{B00000000,B00000000,B00000000,B00000000,B00000000,B00000000,B01100000,B01100000}

#define COMA{B00000000,B00000000,B00000000,B00000000,B00000000,B00110000,B00110000,B01100000}

#define LINE{B00000000,B00000000,B00000000,B01111110,B01111110,B00000000,B00000000,B00000000}

#define QUASTION{B00111000,B01000100,B00000100,B00000100,B00001000,B00010000,B00000000,B00010000}

#define MARK{B00011000,B00011000,B00011000,B00011000,B00011000,B00000000,B00011000,B00011000}

#define DA {B00000000,B0000000,B0000000,B00111100,B00000000,B00000000,B00000000,B00000000}

#define SM {B00111100,B01000010,B10100101,B10000001,B10100101,B10011001,B01000010,B00111100}

#define SM1 {B00111100,B01000010,B10100101,B10000001,B10011001,B10100101,B01000010,B00111100}

#define SM2 {B00111100,B01000010,B10100101,B10000001,B10000001,B10111101,B01000010,B00111100}

#define END {B11111111,B11111111,B11111111,B11111111,B11111111,B11111111,B11111111,B11111111}

#define DG {B00111000,B10010010,B10111010,B01010100,B00010000,B00101000,B01000100,B10000010}

#define DG1 {B00111000,B00010000,B10111010,B11010110,B00010000,B00101000,B01000100,B10000010}

int latchPin = 10;

int clockPin = 13;

int dataPin = 11;

int clock = 9;

int Reset = 8;

int latchPinPORTB = latchPin - 8;

int clockPinPORTB = clockPin - 8;

int dataPinPORTB = dataPin - 8;

int i = 0;

long scrolling\_word[16];

int array\_turn=0;

byte your\_text[][8]={E,L,DA,NUM2,NUM1,NUM3,SPACE,G,R,O,U,P,DA,NUM1,NUM7,SPACE,SM,SPACE,H,E,L,L,O,SPACE,R,U,T,U,SPACE,M,A,M,SPACE,guk,gum,SPACE,guch,QUASTION,SPACE,SM};//PUT YOUR TEXT HERE

int len = sizeof(your\_text)/8;

void setup(){

Serial.begin(9600);

pinMode(dataPin,OUTPUT);

pinMode(clockPin,OUTPUT);

pinMode(latchPin,OUTPUT);

pinMode(clock,OUTPUT);

pinMode(Reset,OUTPUT);

digitalWrite(Reset,HIGH);

digitalWrite(Reset,LOW);

setupSPI();

}

void display\_word(int loops,byte word\_print[][8],int num\_patterns,int delay\_langth){// this function displays your symbols

i = 0;// resets the counter fot the 4017

for(int g=0;g<8;g++)//resets the the long int where your word goes

scrolling\_word[g] = 0;

for(int x=0;x<num\_patterns;x++){//main loop, goes over your symbols

// you will need to find a better way to make the symbols scroll my way is limited for 24 columns

for(int r=0;r<8;r++)//puts the buildes the first symbol (also controls how many columns are displayed)

scrolling\_word[r] |= word\_print[x][r];

for (int z=0;z<8;z++){//the scrolling action (Z is also pretty much a kerning variable- larger vale = more space between letters)

for(int p=0;p<8;p++)

scrolling\_word[p] = scrolling\_word[p] << 1;

// end of the scrolling funcion

for(int t=0;t<delay\_langth;t++){// delay function, it just loops over the same display

for(int y=0;y<8;y++){// scaning the display

// delay(10); //for exaplanation

if(i == 8){// counting up to 8 with the 4017

digitalWrite(Reset,HIGH);

digitalWrite(Reset,LOW);

i = 0;

}

latchOff();

spi\_transfer(make\_word(0x01000000,y));// sending the data

spi\_transfer(make\_word(0x00010000,y));

spi\_transfer(make\_word(0x00000100,y));

latchOn();

delayMicroseconds(2000);//waiting a bit

latchOff();

spi\_transfer(0);// clearing the data

spi\_transfer(0);

spi\_transfer(0);

latchOn();

digitalWrite(clock,HIGH);//counting up with the 4017

digitalWrite(clock,LOW);

i++;

}

}

}

}

finish\_scroll(delay\_langth);

}

void finish\_scroll(int delay\_scroll){// this function is the same as the funcion above, it just finishing scrolling

for (int n=0;n<24;n++){

for(int h=0;h<8;h++)

scrolling\_word[h] = scrolling\_word[h] << 1;

for(int w=0;w<delay\_scroll;w++){

for(int k=0;k<8;k++){

// delay(10);

if(i == 8){

digitalWrite(Reset,HIGH);

digitalWrite(Reset,LOW);

i = 0;

}

latchOff();

spi\_transfer(make\_word(0x01000000,k));

spi\_transfer(make\_word(0x00010000,k));

spi\_transfer(make\_word(0x00000100,k));

latchOn();

delayMicroseconds(2000);

latchOff();

spi\_transfer(0);

spi\_transfer(0);

spi\_transfer(0);

latchOn();

digitalWrite(clock,HIGH);

digitalWrite(clock,LOW);

i++;

}

}

}

}

byte make\_word (long posistion,byte turn){

byte dummy\_word = 0;

for(int q=0;q<8;q++){

if(scrolling\_word[turn] & (posistion<<q))

dummy\_word |= 0x01<<q;

}

return dummy\_word;

}

void loop() {

display\_word(1,your\_text,len,15);// calls for the display\_pattern function and says that int loop = 15(if you do more loop the pattern whould scrole slower).

// the seconds variable in this (20 in this case), controls how many letters your display can hold at one time.

}

void latchOn(){

bitSet(PORTB,latchPinPORTB);

}

void latchOff(){

bitClear(PORTB,latchPinPORTB);

}

void setupSPI(){

byte clr;

SPCR |= ( (1<<SPE) | (1<<MSTR) ); // enable SPI as master

//SPCR |= ( (1<<SPR1) | (1<<SPR0) ); // set prescaler bits

SPCR &= ~( (1<<SPR1) | (1<<SPR0) ); // clear prescaler bits

clr=SPSR; // clear SPI status reg

clr=SPDR; // clear SPI data reg

SPSR |= (1<<SPI2X); // set prescaler bits

//SPSR &= ~(1<<SPI2X); // clear prescaler bits

delay(10);

}

byte spi\_transfer(byte data)

{

SPDR = data; // Start the transmission

while (!(SPSR & (1<<SPIF))) // Wait the end of the transmission

{

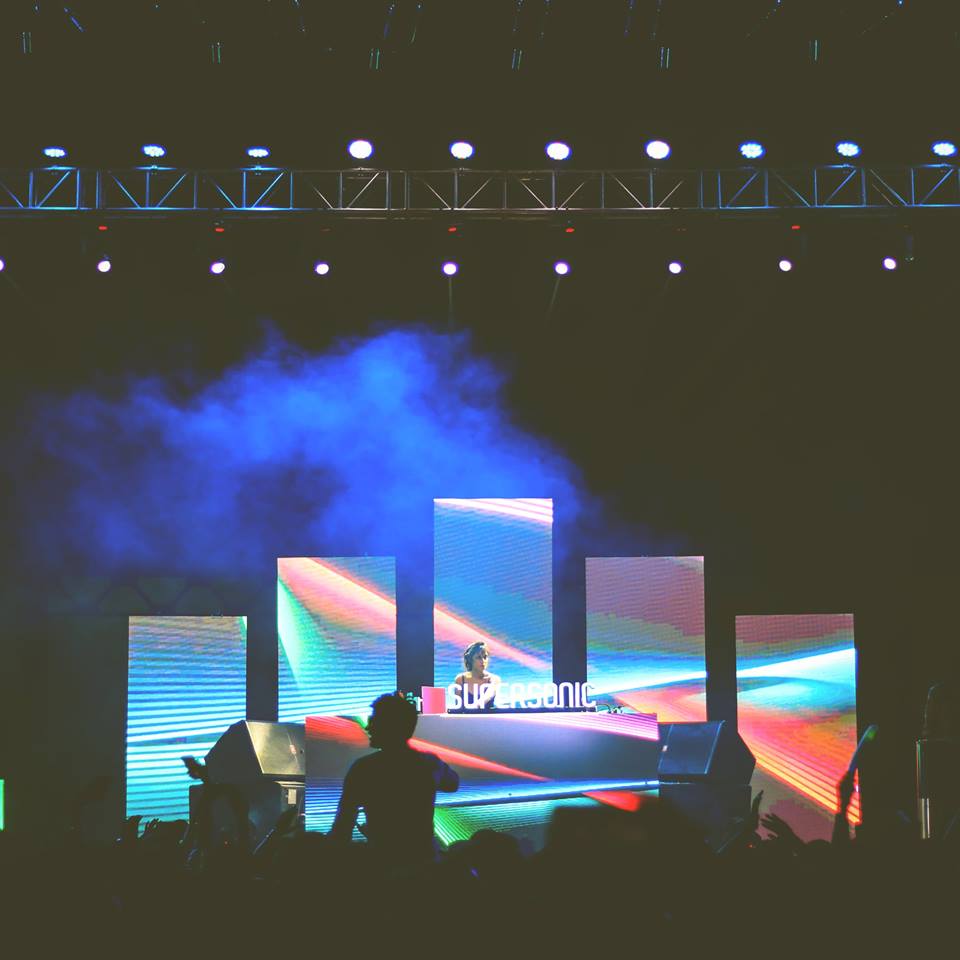
};

return SPDR; // return the received byte, we don't need that

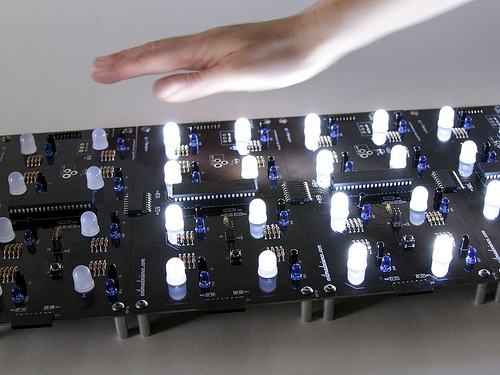
}

* APPLICATIONS**:-**
* There are numerous applications of LED matrix. Some of them are explained below:



1. Display signs: LED matrixes are widely used as display signs as they are appropriate to display a message to a crowd of people. They are used in restaurants, hotels, shops to display names or order numbers. They used along the roads to display scrolling announcements on billboards.They are used in BRTS public bus transport system in Ahmedabad, Surat etc. to display destination, current location and time in the buses and in different stations.
2. Decorative: LED matrixes are attractive and beautiful. With some changes in code and circuit and circuit components, like using multicolour LEDs, different types of patterns and designs are displayed. LED matrix is also used to display visual effects. For example, in the events like Sunburn, VH1 Supersonic , EDM night etc. the stage is made of numerous smaller LED matrix displays connected together to make one huge wall to display different patterns, designs and texts.



1. Interactive LED matrix display: In this, infrared LEDs are added in the matrix with normal LEDs. That makes a motion sensor within the matrix. Whenever we move our hand over it, the LED lights up because of the disturbance created by moving our hand closer. There are also fixed on the floor to make Interactive LED matrix floor. The Interactive LED matrix also responds to touch. These kinds of matrix are not widely available as it is expensive to make. Interactive LED matrixes are also made sensitive to frequencies, where it responds to sound vibration. They are very attractive if made to display different patterns and designs according to the sound or voice produced or music played.



1. LED TV: This the most common application of LED matrix display. It consists of a huge matrix with high density of LEDs. These are tricoloured (Red, Green and Blue) LEDs with high intensity, which can be controlled according to requirement. Millions of colours are displayed by controlling the intensity of red, green and blue lights in the LEDs. This is how colourful videos are displayed.

* Bibliography

* <http://www.instructables.com/id/Make-a-giant-LED-sign/>
* <http://www.instructables.com/id/Make-a-24X6-LED-matrix/>