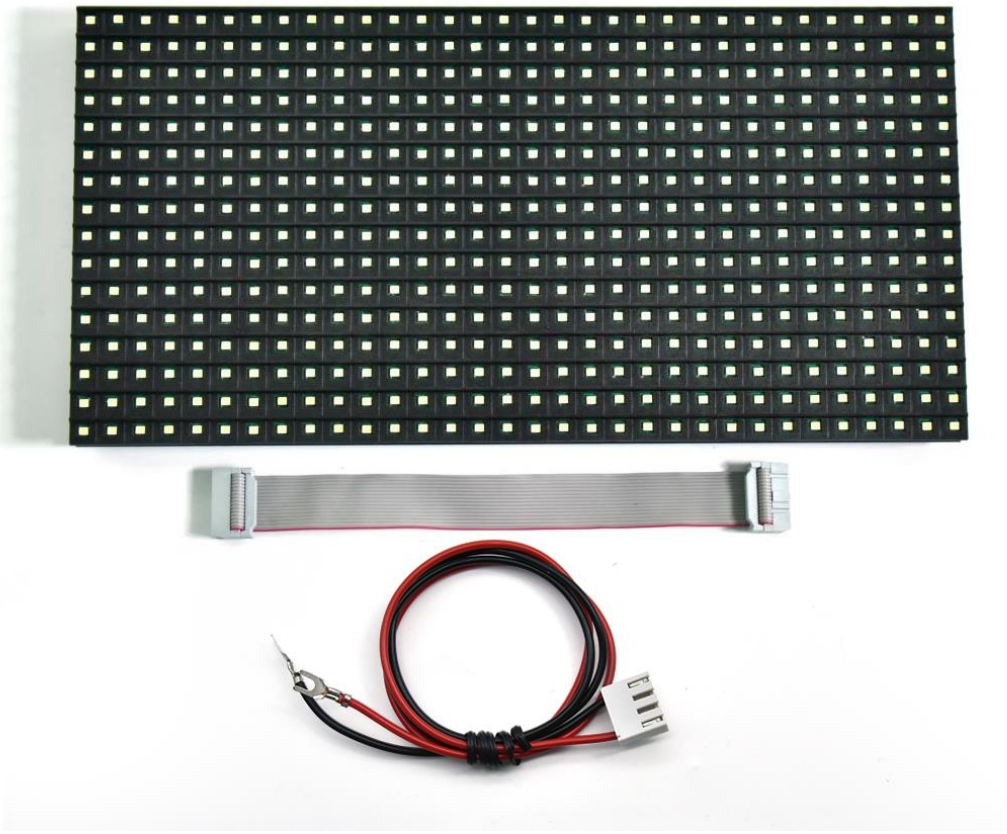
keyestudio

P10 32*16 Mono Color LED Matrix



Introduction：

Billboard is an outdoor media that transmits information. The P10 32*16 monocolor LED matrix is one of billboards.

It can form into a larger billboard. It is composed of 512 RGB lamp beads, which can display red, green, blue and white color

Parameters：

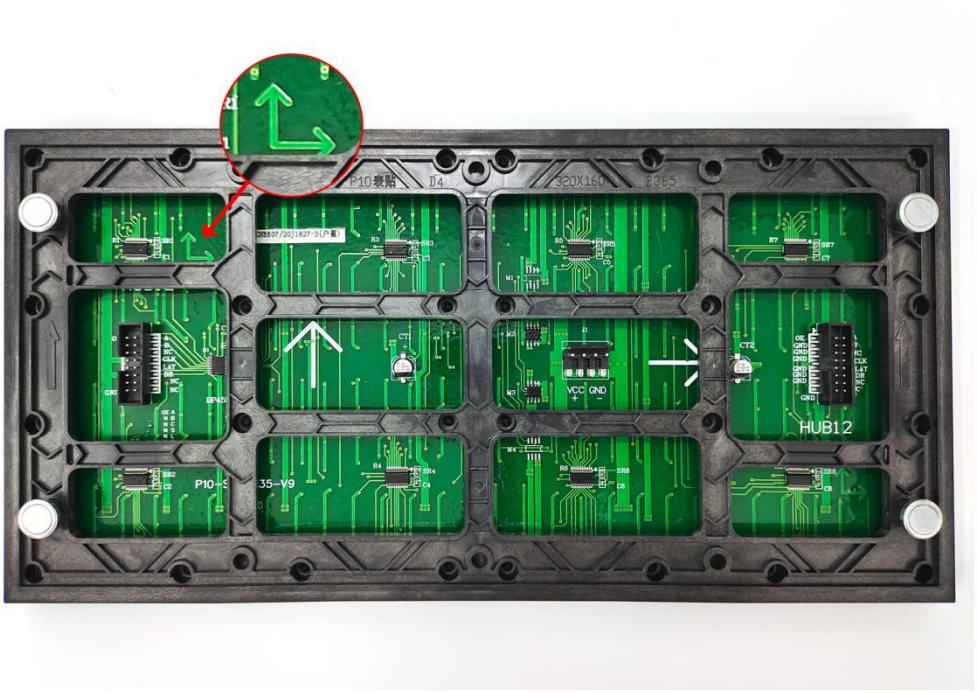
Physical point spacing	10	Best viewing angle	Horizontal 80° Vertical 75°
Physical density (点/m^2)	10000	Driving way	Constant current drive, dynamic
Luminous brightness(cd/m^2)	3000	Scanning way	1/4
Cell board size （mm）	320x160	Cell board interface	HUB12
Cell board pixels (dot)	512	Operating Voltage	DC 5V ±5%
Pixel composition	1R	Average power consumption	7W/Cell board
Cell board weight	266g	Maximum power consumption	≤15W/Cell board
Best sight distance	>5m	Life span	10W hour
Control way	U disk card, wireless WIF card, network card, serial port cards, GPRS wireless card		

Wiring Diagram：

How to determine data input/output ports?

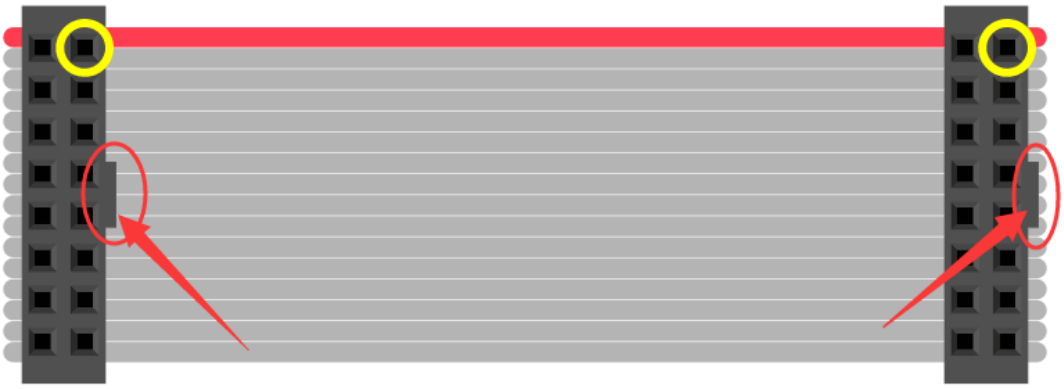
Ignore vertical arrows(up and down arrows), view horizontal arrows(left and right arrows). Horizontal arrows imply the direction of data input to data output. In this way, you can determine the port of inputs

As you can see horizontal arrows below point at the right side. That means data coming from the left to the right. So, the left is the input port.

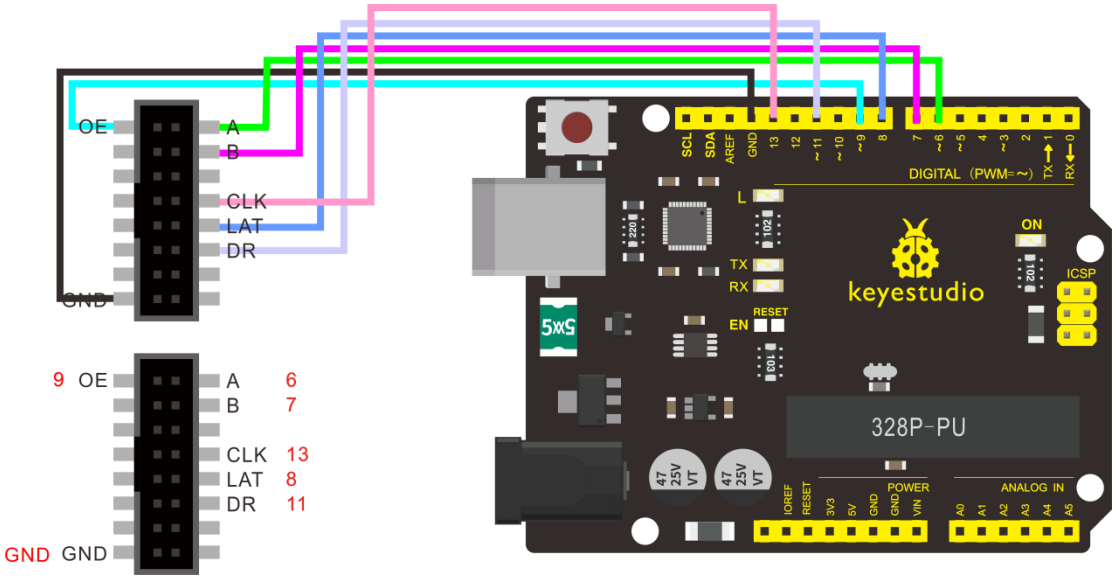


Take the raised point of the cable connector as a reference(red circles).

Interfaces marked by yellow circles are connected.



Connect flat cables to the left port, then wire the UNO board and flat cables using breadboard wires, as shown below; (red numbers are UNO ports)



Test Code：

```
#include <SPI.h>              //SPI.h must be included as DMD is written by SPI (the
IDE complains otherwise)
#include "DMD.h"              //
#include <TimerOne.h>      //
#include "SystemFont5x7.h"
#include "Arial_black_16.h"


//Fire up the DMD library as dmd
#define DISPLAYS_ACROSS 1
#define DISPLAYS_DOWN 1
DMD dmd(DISPLAYS_ACROSS, DISPLAYS_DOWN);


/*--------------------------------------------------------------------------------
```

```
  Interrupt handler for Timer1 (TimerOne) driven DMD refresh scanning, this gets
  called at the period set in Timer1.initialize();
---------------------------------------------------------------------------*/
void ScanDMD()
{
  dmd.scanDisplayBySPI();
}

/*---------------------------------------------------------------------------
  setup
  Called by the Arduino architecture before the main loop begins
---------------------------------------------------------------------------*/
void setup(void)
{

   //initialize TimerOne's interrupt/CPU usage used to scan and refresh the
display
   Timer1.initialize( 5000 );          //period in microseconds to call
ScanDMD. Anything longer than 5000 (5ms) and you can see flicker.
   Timer1.attachInterrupt( ScanDMD );  //attach the Timer1 interrupt to
ScanDMD which goes to dmd.scanDisplayBySPI()
```

```
    //clear/init the DMD pixels held in RAM
    dmd.clearScreen( true );    //true is normal (all pixels off), false is negative
(all pixels on)

}

/*--------------------------------------------------------------------------------
  loop
  Arduino architecture main loop
--------------------------------------------------------------------------------*/
void loop(void)
{
    byte b;

    // 10 x 14 font clock, including demo of OR and NOR modes for pixels so that
the flashing colon can be overlayed
    dmd.clearScreen( true );
    dmd.selectFont(Arial_Black_16);
    dmd.drawChar(  0,  3, '2', GRAPHICS_NORMAL );
    dmd.drawChar(  7,  3, '3', GRAPHICS_NORMAL );
    dmd.drawChar( 17,  3, '4', GRAPHICS_NORMAL );
    dmd.drawChar( 25,  3, '5', GRAPHICS_NORMAL );
```

```
dmd.drawChar( 15,    3, ':', GRAPHICS_OR       );    // clock colon overlay on
delay( 1000 );
dmd.drawChar( 15,    3, ':', GRAPHICS_NOR      );    // clock colon overlay off
delay( 1000 );
dmd.drawChar( 15,    3, ':', GRAPHICS_OR       );    // clock colon overlay on
delay( 1000 );
dmd.drawChar( 15,    3, ':', GRAPHICS_NOR      );    // clock colon overlay off
delay( 1000 );
dmd.drawChar( 15,    3, ':', GRAPHICS_OR       );    // clock colon overlay on
delay( 1000 );

dmd.drawMarquee("Scrolling Text",14,(32*DISPLAYS_ACROSS)-1,0);
long start=millis();
long timer=start;
boolean ret=false;
while(!ret){
   if ((timer+30) < millis()) {
      ret=dmd.stepMarquee(-1,0);
      timer=millis();
   }
}
// half the pixels on
```

```
    dmd.drawTestPattern( PATTERN_ALT_0 );
    delay( 1000 );

    // the other half on
    dmd.drawTestPattern( PATTERN_ALT_1 );
    delay( 1000 );

    // display some text
    dmd.clearScreen( true );
    dmd.selectFont(System5x7);
    for (byte x=0;x<DISPLAYS_ACROSS;x++) {
      for (byte y=0;y<DISPLAYS_DOWN;y++) {
        dmd.drawString(  2+(32*x),  1+(16*y),  "freet",  5,
GRAPHICS_NORMAL );
        dmd.drawString(  2+(32*x),  9+(16*y),  "ronic",  5,
GRAPHICS_NORMAL );
      }
    }
    delay( 2000 );

    // draw a border rectangle around the outside of the display
    dmd.clearScreen( true );
```

```
    dmd.drawBox(    0,    0, (32*DISPLAYS_ACROSS)-1, (16*DISPLAYS_DOWN)-1,
GRAPHICS_NORMAL );
    delay( 1000 );

    for (byte y=0;y<DISPLAYS_DOWN;y++) {
       for (byte x=0;x<DISPLAYS_ACROSS;x++) {
          // draw an X
          int ix=32*x;
          int iy=16*y;
          dmd.drawLine(    0+ix,    0+iy, 11+ix, 15+iy, GRAPHICS_NORMAL );
          dmd.drawLine(    0+ix, 15+iy, 11+ix,    0+iy, GRAPHICS_NORMAL );
          delay( 1000 );

          // draw a circle
          dmd.drawCircle( 16+ix,    8+iy,    5, GRAPHICS_NORMAL );
          delay( 1000 );

          // draw a filled box
          dmd.drawFilledBox( 24+ix, 3+iy, 29+ix, 13+iy, GRAPHICS_NORMAL );
          delay( 1000 );
       }
    }
```

```
    // stripe chaser
    for( b = 0 ; b < 20 ; b++ )
    {
        dmd.drawTestPattern( (b&1)+PATTERN_STRIPE_0 );
        delay( 200 );
    }
    delay( 200 );

}
```