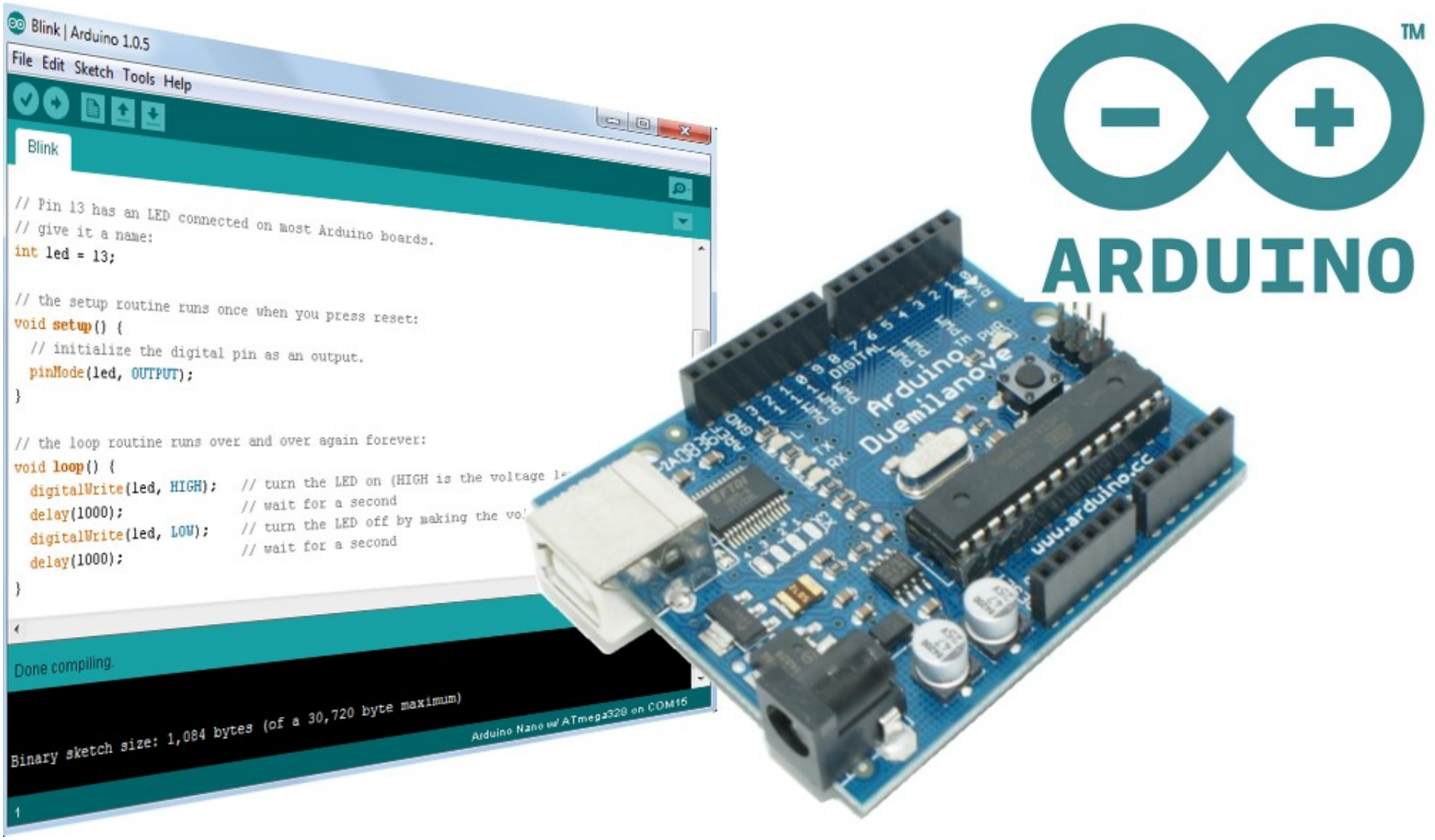
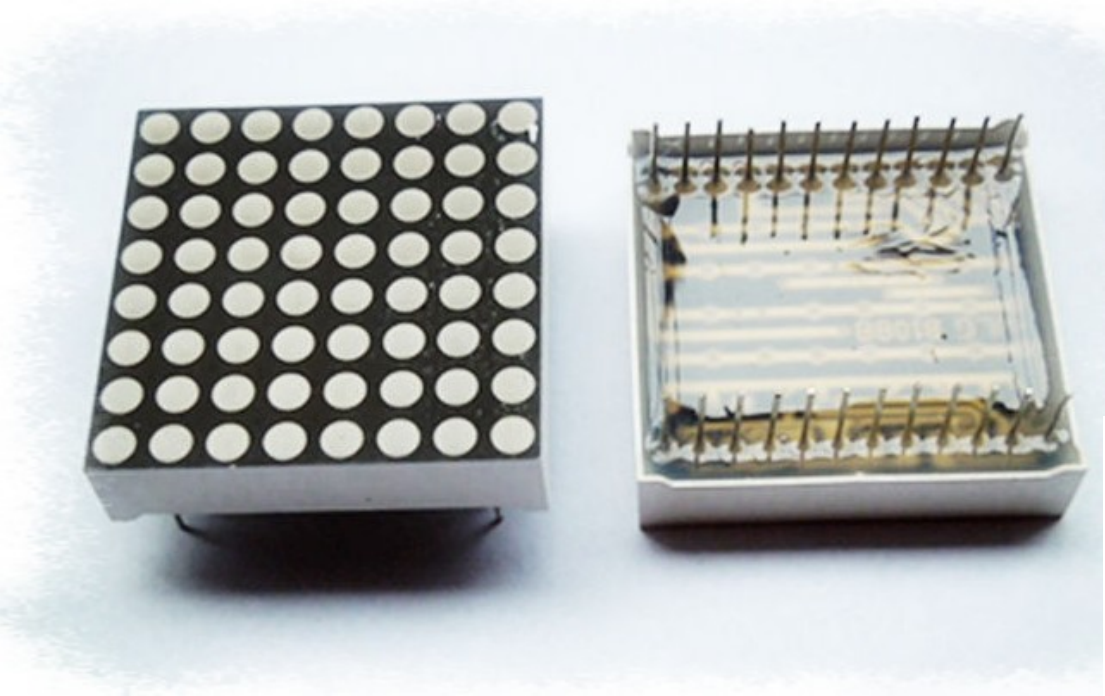
Arduino tanfolyam kezdőknek és haladóknak



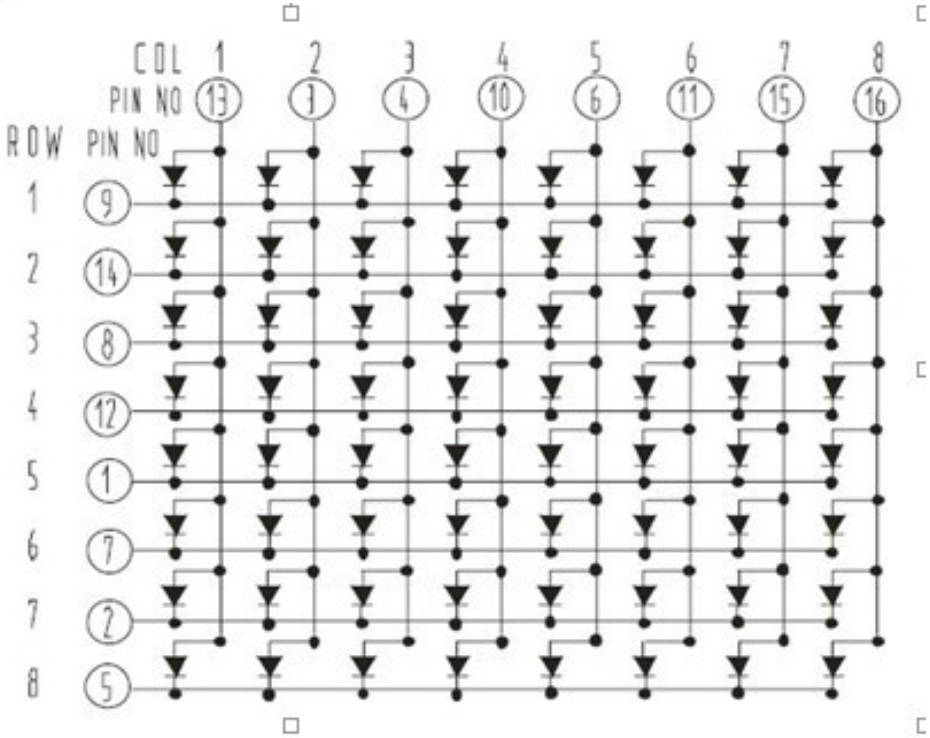
8. LED mátrix vezérlése MAX7219 IC-vel

8x8 LED mátrix



3 mm-es piros LED-ek 8x8-as mátrixba szervezve

A **1088AS** típusnál a sorkiválasztó vonalak a katódokat közösítik



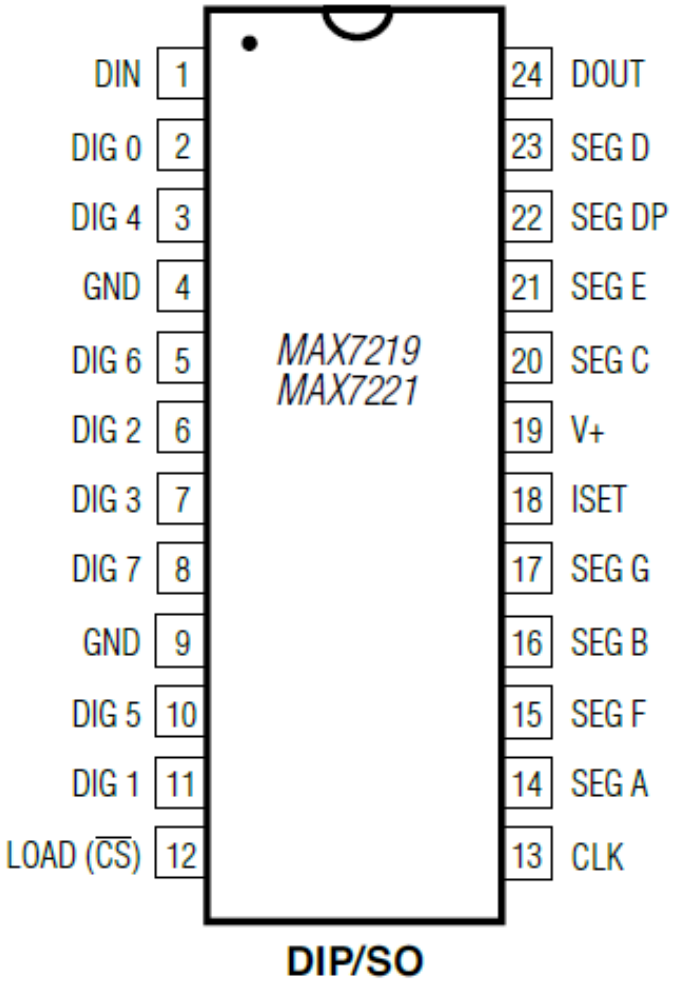
Multiplex kijelzés, egyidejűleg legfeljebb egy sor, vagy egy oszlop lehet aktív.

Kényelmes meghajtás:

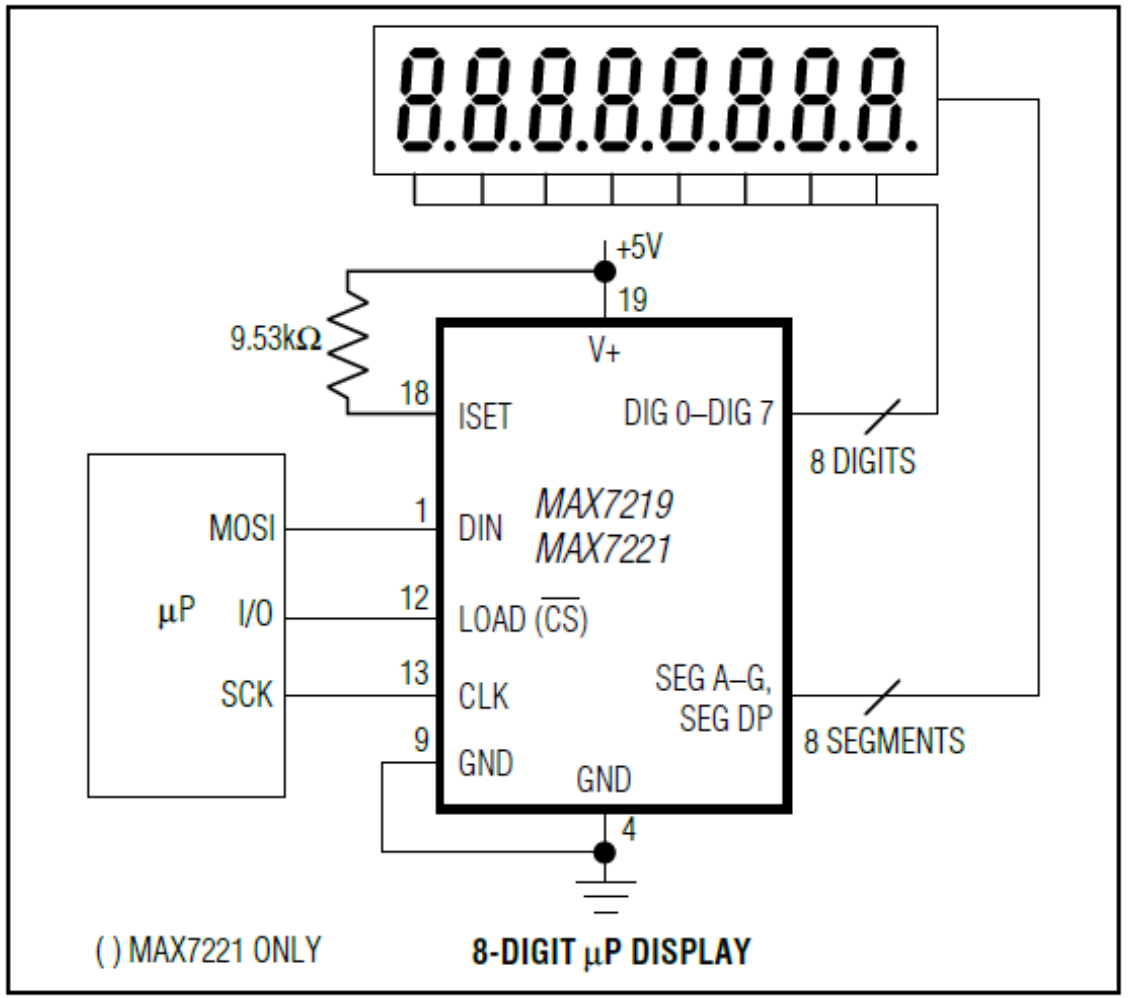
- 1 db **MAX7219**, vagy
- 2 db **74HC595** (+ meghajtó +áramkorlátozás)
- 1 db **MCP23017** (+ meghajtó +áramkorlátozás)

MAX7219

LED meghajtó IC, beépített áramkorlátozással. 7 szegmens kijelző esetén 1-8 db számjegy meghajtása (opcionálisan beépített dekódolással), vagy 8x8 LED mártix meghajtása. Az IC vezérlése SPI buszon történhet.

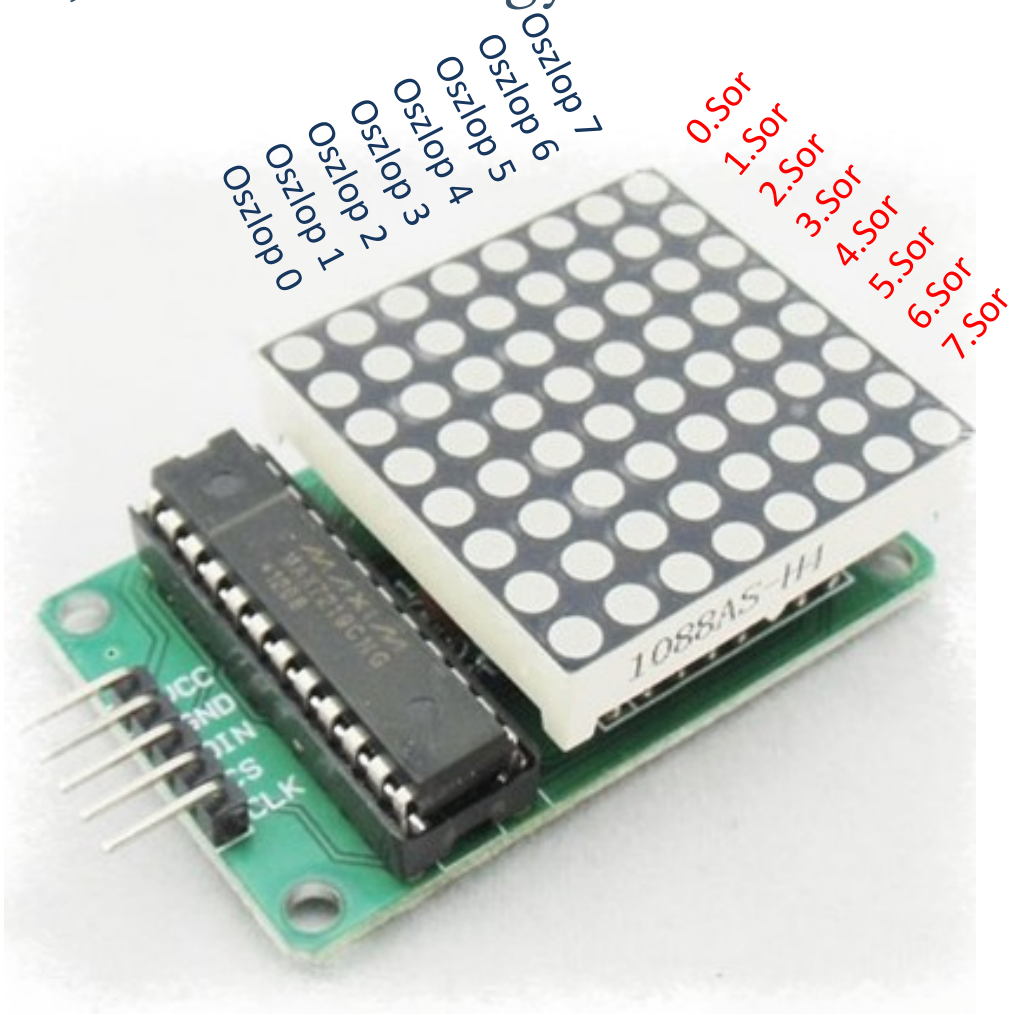


DIP/SO



Kijelző modul MAX7219 IC-vel

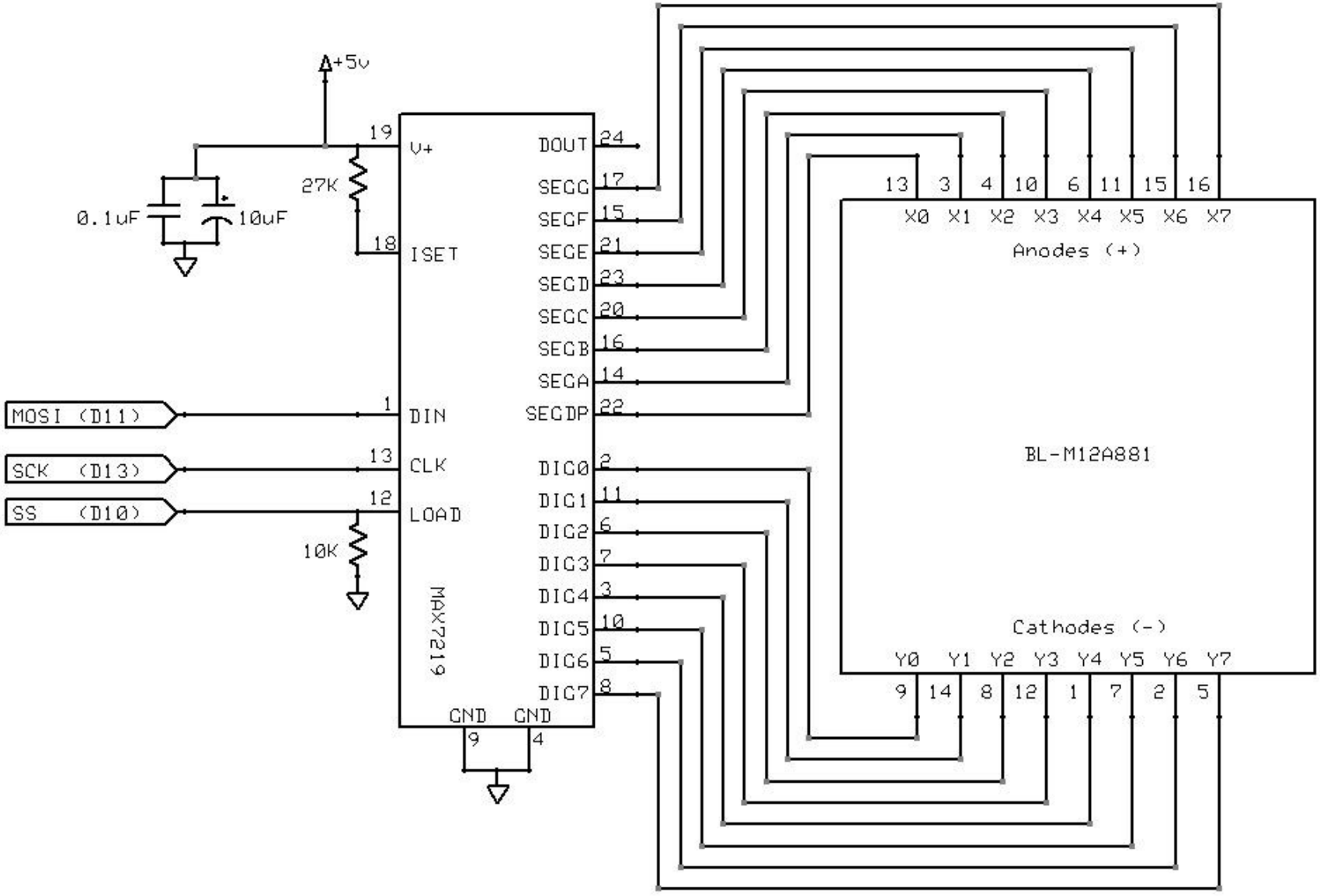
- Az E-bay kínálatában kapható, furatszerelt vagy felületszerelt kivitelben
  - 8x8 LED mátrix
  - MAX7219 vezérlő
  - Felfűzhető kivitel
  - Tápellátás: 3,5 – 5 V



Bemenetek	**Kimenetek**
1 VCC	1 VCC
2 GND	2 GND
3 DIN	3 DOUT
4 CS	4 CS
5 CLK	5 CLK

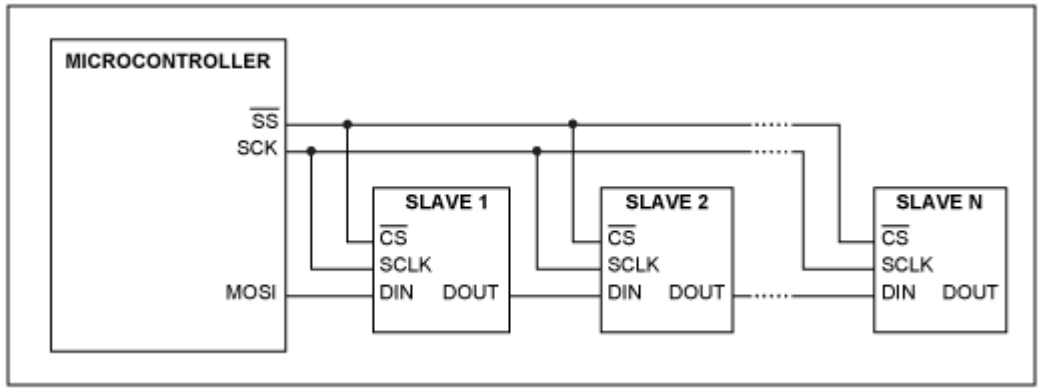
- **DIN/DOUT** – soros adat, **CLK** – szinkron órajel, **CS** – eszközkiválasztó jel, **VCC** – tápfeszültség, **GND** – a tápegység közös pontja („föld”)

Kapcsolási rajz

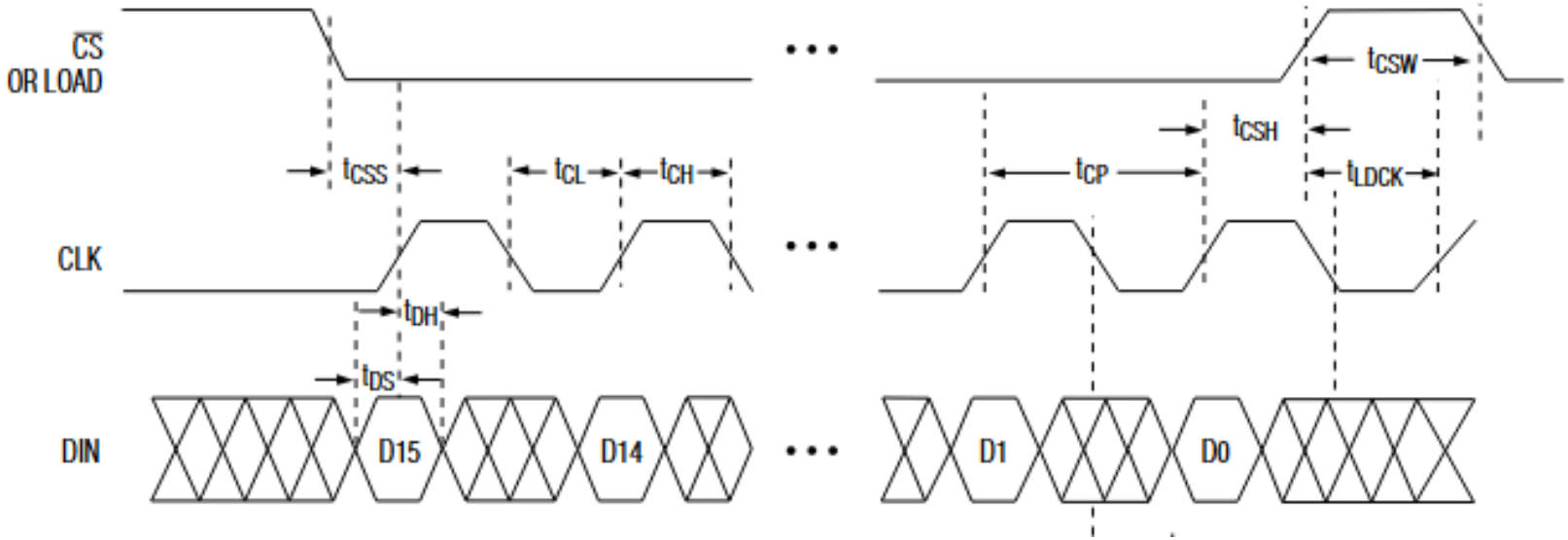


Hogy működik az SPI adatküldés?

- **SPI** – soros periféria illesztőt jelent, ahol az adatbitek nem több vezetéken párhuzamosan, hanem egy adatvezetéken, sorban, egymás után haladnak



- Az adatbitek szinkronizálását egy külön vezetéken kiküldött órajel impulzusok végzik, a tranzakció szinkronizálását pedig a **LOAD** (**CS**) jel végzi (felfutó élére íródik be a 16 bites adat)



Példaprogramok

- **LED8x8_MAX7219** – egyszerű példaprogram a **MAX7219** IC inicializálásának és használatának bemutatására (I♥He kiírása)
- **LedControl_8x8** – a fenti feladat megvalósítása a **LedControl** programkönyvtár segítségével (I♥He kiírása, picit lassítva)
- **LedControl_8x8_scroll** – szöveg görgetése egy **MAX7219** meghajtóval vezérelt 8x8-as LED mátrixon, a módosított **LedControl** programkönyvtár segítségével
- **LedControl_8x8_scroll_4** – szöveg görgetése négy, felfűzött, **MAX7219** meghajtóval vezérelt 8x8-as LED mátrixon, a módosított **LedControl** programkönyvtár segítségével
- **LedControl_7seg** – szöveg és szám görgetése egy 8 számjegyű, **MAX7219** meghajtóval vezérelt 7-szegmenses kijelzőn, a módosított **LedControl** programkönyvtár segítségével

LED8x8_MAX7219.ino

- A kijelző modul meghajtásához három digitális kimenet kell

```
const int SPI_CS  = 10;                    // SPI Chip select    (LED CS bemenetre)
const int SPI_CLK = 11;                    // SPI CLK órajel     (LED CLK bemenetre)
const int SPI_DIN = 12;                    // SPI MOSI adatvonal (LED DIN bemenetre)
const unsigned char led1[]= {0xFF,0x00,0x30,0x48,0x90,0x90,0x48,0x30};  //Iv
const unsigned char led2[]= {0xFF,0x18,0x18,0xFF,0x00,0xF8,0xA8,0xA8};  //HE

void setup() {
  pinMode(SPI_CS,OUTPUT);                  // SPI_CS kimenet inicializálása
  digitalWrite(SPI_CS,HIGH);
  pinMode(SPI_CLK,OUTPUT);                 // SPI_CLK kimenet inicializálása
  digitalWrite(SPI_CLK,LOW);
  pinMode(SPI_DIN,OUTPUT);                 // SPI_DIN kimenet inicializálása
  digitalWrite(SPI_DIN,LOW);
  Init_MAX7219();                          // LED vezérlő bekonfigurálása
}

void loop (){
  for(int i=0; i<8; i++)                   // Első karakter (8 sor) kiírása
    SPI_Write2(i+1,led1[i]);
  delay(1000);                             // 1 s várakozása
  for(int i=0; i<8; i++)                   // Második karakter (8 sor) kiírása
    SPI_Write2(i+1,led2[i]);
  delay(1000);                             // 1 s várakozása
}
```

regisztercím eltolás

regisztercím eltolás

LED8x8_MAX7219.ino (folytatás)

```
/* A LED vezérlő IC konfigurálása */
void Init_MAX7219(void) {
  SPI_Write2(0x09, 0x00);       // Dekódolást kikapcsolni
  SPI_Write2(0x0A, 0x08);       // Közepes fényerő beállítása
  SPI_Write2(0x0B, 0x07);       // Pásztázási korlát beállítása
  SPI_Write2(0x0C, 0x01);       // Megjelenítési mód beállítása
  SPI_Write2(0x0F, 0x0F);       // Display teszt engedélyezés
  delay(500);
  SPI_Write2(0x01, 0x00);       // 0. sor törlése
  SPI_Write2(0x02, 0x00);       // 1. sor törlése
  SPI_Write2(0x03, 0x00);       // 2. sor törlése
  SPI_Write2(0x04, 0x00);       // 3. sor törlése
  SPI_Write2(0x05, 0x00);       // 4. sor törlése
  SPI_Write2(0x06, 0x00);       // 5. sor törlése
  SPI_Write2(0x07, 0x00);       // 6. sor törlése
  SPI_Write2(0x08, 0x00);       // 7. sor törlése
  SPI_Write2(0x0F, 0x00);       // Display teszt tiltás
  delay(500);
}

/* Két bájt kiküldése */
void SPI_Write2(unsigned char MSB, unsigned char LSB) {
  digitalWrite(SPI_CS,LOW);                // CS aktiválás
  shiftOut(SPI_DIN,SPI_CLK,MSBFIRST,MSB); // SW SPI adatküldés 1.
  shiftOut(SPI_DIN,SPI_CLK,MSBFIRST,LSB); // SW SPI adatküldés 2.
  digitalWrite(SPI_CS,HIGH);               // CS deaktiválás
}
```

Technikai részletek az inicializáláshoz

Table 2. Register Address Map

REGISTER	ADDRESS					HEX CODE
	D15–D12	D11	D10	D9	D8	
No-Op	X	0	0	0	0	0xX0
Digit 0	X	0	0	0	1	0xX1
Digit 1	X	0	0	1	0	0xX2
Digit 2	X	0	0	1	1	0xX3
Digit 3	X	0	1	0	0	0xX4
Digit 4	X	0	1	0	1	0xX5
Digit 5	X	0	1	1	0	0xX6
Digit 6	X	0	1	1	1	0xX7
Digit 7	X	1	0	0	0	0xX8
Decode Mode	X	1	0	0	1	0xX9
Intensity	X	1	0	1	0	0xXA
Scan Limit	X	1	0	1	1	0xXB
Shutdown	X	1	1	0	0	0xXC
Display Test	X	1	1	1	1	0xXF

Adat

Nem használ adatot
DP a b c d e f g

0: no decode 1: decode

0 – 0xF

0 – 7

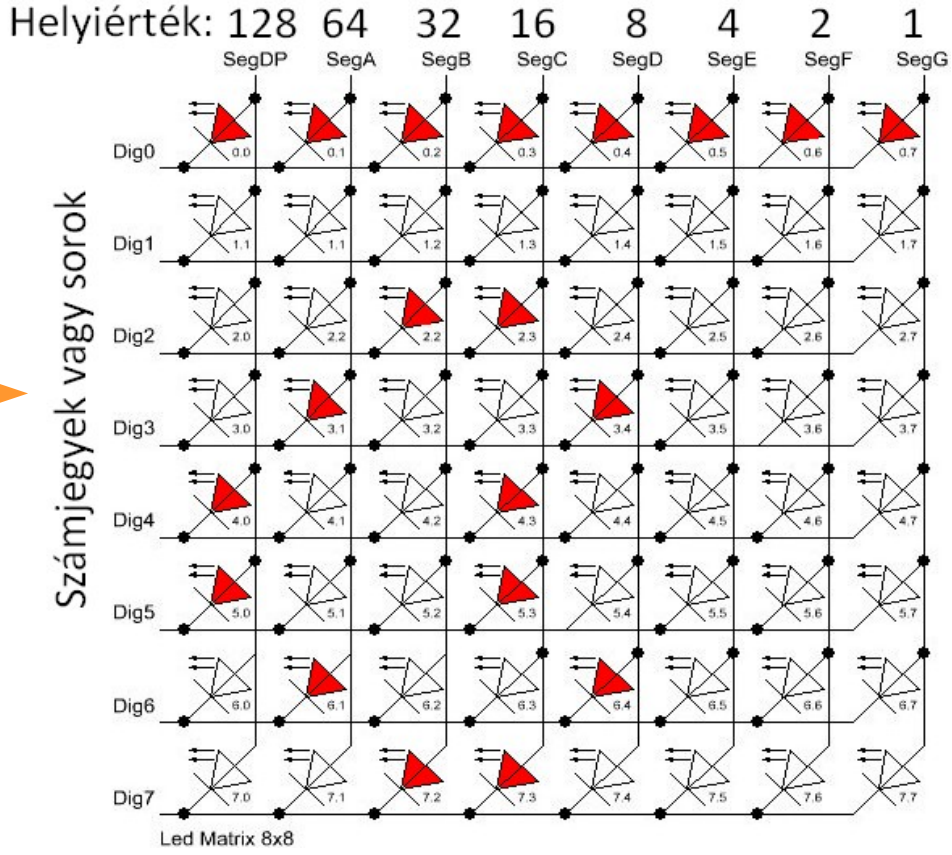
0: shutdown 1: normal mode

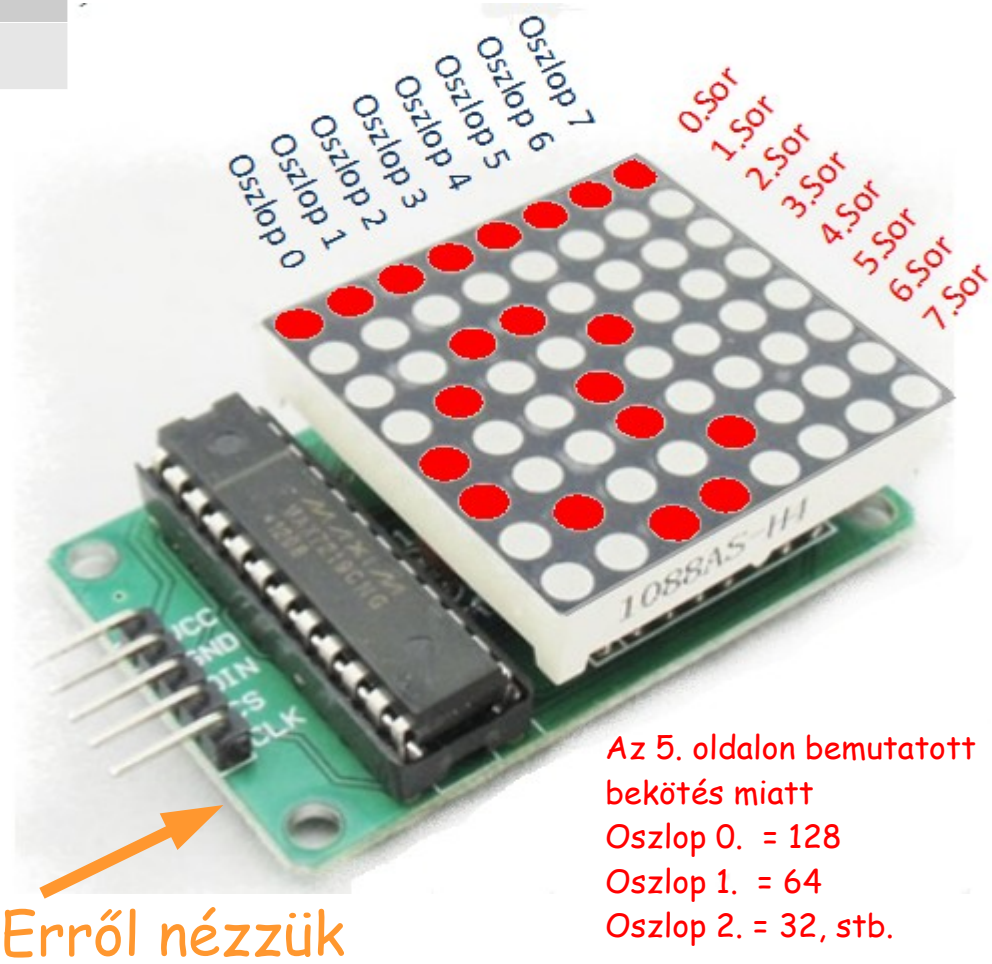
1: test mode 0: normal mode

Technikai részletek a kiíráshoz

Sor	Bin	Hex	Sor	Bin	Hex
0	11111111	0xFF	4	10010000	0x90
1	00000000	0x00	5	10010000	0x90
2	00110000	0x30	6	01001000	0x48
3	01001000	0x48	7	00110000	0x30

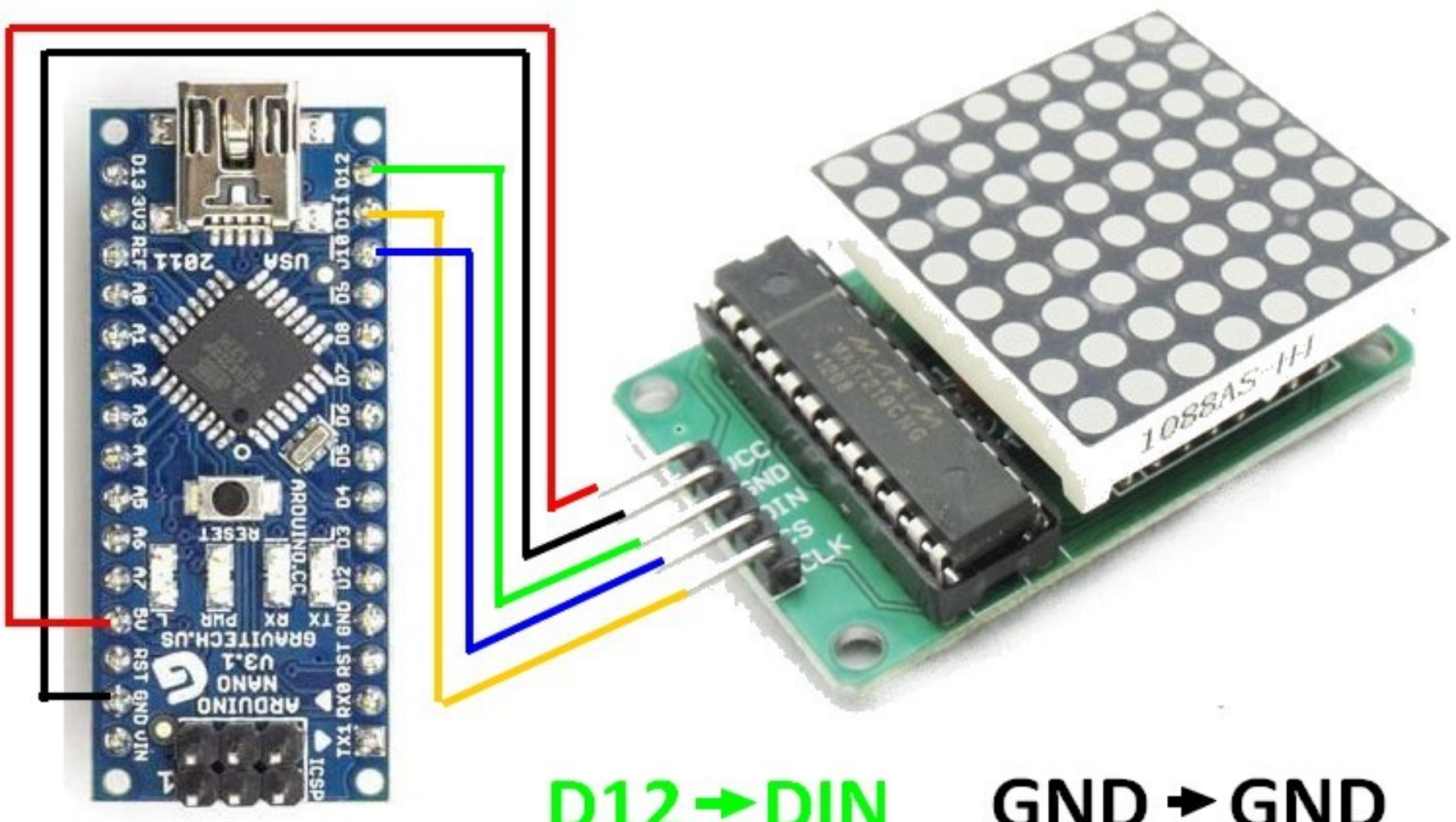
```
const unsigned char led1[]={
    0xFF, 0x00, 0x30, 0x48,
    0x90, 0x90, 0x48, 0x30
};  // I♥
```



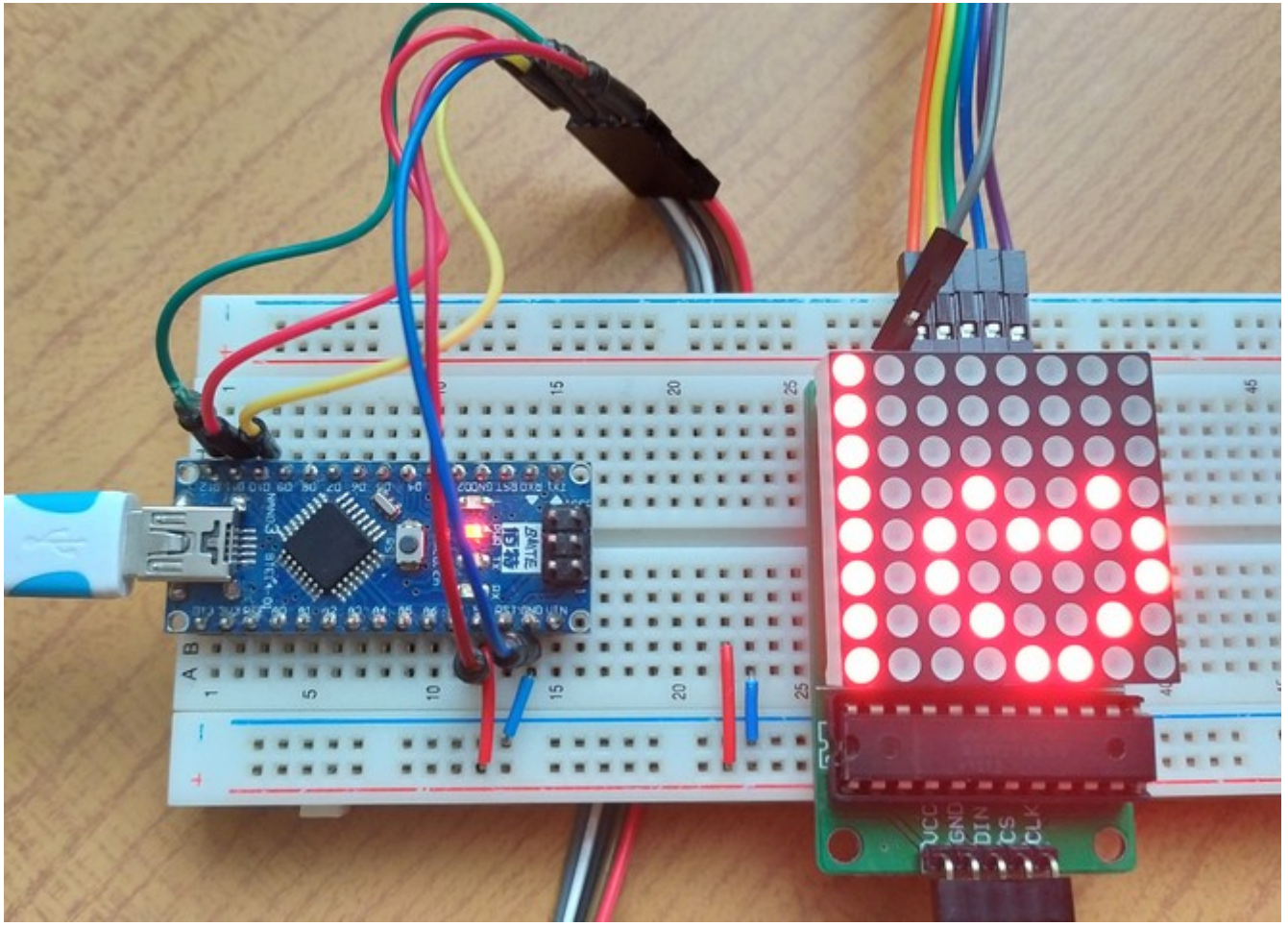


Az 5. oldalon bemutatott bekötés miatt
Oszlop 0. = 128
Oszlop 1. = 64
Oszlop 2. = 32, stb.

Bekötési vázlat



Futási eredmény



LedControl programkönyvtár

- **LedControl** programkönyvtár Arduinohoz/Energiához

  Alkalmas max 8 db. felfűzött **MAX7219** IC-vel meghajtott 8x8-as LED mátrix vagy 8 digites, közös katódú 7 szegmens kijelző vezérlésére

- 2007 – 2015 Eberhard Fahle - eredeti változat forráskód (github.com/wayoda/LedControl)
- LedControl programkönyvtár dokumentáció wayoda.github.io/LedControl/

További lehetőségek:

- **MAX7219** library - github.com/nickgammon/MAX7219
- **MD_MAX72XX** - arduinolibraries.info/libraries/md_max72-xx
- **MaxMatrix** library - github.com/AndreasBur/MaxMatrix

LedControl programkönyvtár

LedControl(int dataPin, int clkPin, int csPin, int numDevices); - Konstruktor

shutdown(int addr, bool status); - Kijelzés leállítása/engedélyezése (true/false)
setScanLimit(int addr, int limit); - Pontsorok vagy digitek számának beállítása (0-7)
setIntensity(int addr, int intensity); - Fényerő beállítása (0 – 15)
clearDisplay(int addr); - Kijelző törlése

8x8 LED mátrix:
setLed(int addr, int row, int col, boolean state); - egy LED ki-/bekapcsolása
getRow(int addr, int row); - Egy LEDsor állapotának kiolvasása (saját bővítés)
setRow(int addr, int row, byte value); - egy LED sor állapotának beállítása
setColumn(int addr, int col, byte value); - egy LED oszlop állapotának beállítása

7-szegmens kijelző:
setDigit(int addr, int digit, byte value, boolean dp); - egy számjegy kiírása (0..9)
setChar(int addr, int digit, char value, boolean dp); - egy karakter kiírása (A,b,c,d,E,F,H,L,P)

LedControl_8x8.ino

- Az előző program megvalósítása a LedControl programkönyvtárral

```
#include "LedControl.h"
LedControl lc = LedControl(12, 11, 10, 1);
const unsigned char data[2][8]= {
  {0xFF,0x00,0x30,0x48,0x90,0x90,0x48,0x30},  //Iv
  {0xFF,0x18,0x18,0xFF,0x00,0xF8,0xA8,0xA8}   //HE
};
/* we always wait a bit between updates of the display */
unsigned long delaytime = 1500;

void setup() {
  lc.shutdown(0, false);          // A kijelzés beindítása
  lc.setIntensity(0, 8);          // Közepes fényerő beállítása
  lc.clearDisplay(0);             // Képernyő törlés
}

void loop() {
  for(int j=0; j<2; j++) {        // Végigmegyünk a bitképeken
    for(int i=0; i<8; i++) {      // Egy 8y8-as bitkép kirajzolása soronként
      lc.setRow(0,i,data[j][i]);
      delay(100);                 // Egy kis lassítás a látványosság kedvéért...
    }
    delay(delaytime);
  }
}
```

MOSI, CLK, CS, felfűzött egységek száma

A programkönyvtár bővítése: getRow()

- Szöveg görgetéséhez vissza kell tudni olvasni a **LedControl** osztály privát adattömbjének (`byte status[64]`) bájtjait, amihez egy `LedControl::getRow(int addr, int row)` metódust defináltunk
- **LedControl.h** állományba be kell szúrni az alábbi sort, az osztály publikus metódusainak deklarációi közé:

```
byte getRow(int addr, int row);
```

- **LedControl.cpp** állományba be kell toldani az alábbi sorokat, például a `setROW()` függvény definíciója elé:

```
byte LedControl::getRow(int addr, int row) {
    int offset;
    if(addr<0 || addr>=maxDevices)
    return 0;
    if(row<0 || row>7)
    return 0;
    offset=addr*8;
    return status[offset+row];
}
```

LedControl_8x8_scroll.ino

```
// Fényújság egy MAX7219-cel vezérelt 8x8-as LED mátrixon, LedControl használatával
#include "LedControl.h"
#include "english_6x8_pixel.h"  // A libraries/Fonts mappában helyeztük el

#define DATA_PIN    12          // MOSI -> DIN
#define CLOCK_PIN   11          // CLK  -> CLK
#define LOAD_PIN    10          // CS   -> LOAD
const char line1[] = "LedControl_8x8_scroll example program ";

// Proto: LedControl(int dataPin, int clkPin, int csPin, int numDevices=1);
LedControl lc=LedControl(DATA_PIN, CLOCK_PIN, LOAD_PIN, 1);

void setup() {
  delay(1000);
  pinMode(DATA_PIN,OUTPUT);
  pinMode(CLOCK_PIN,OUTPUT);
  pinMode(LOAD_PIN,OUTPUT);
  digitalWrite(LOAD_PIN,HIGH);
  lc.shutdown(0,false);
  lc.setIntensity(0,8);
  lc.clearDisplay(0);
}

void loop() {
  showText((char*)line1);
}
```

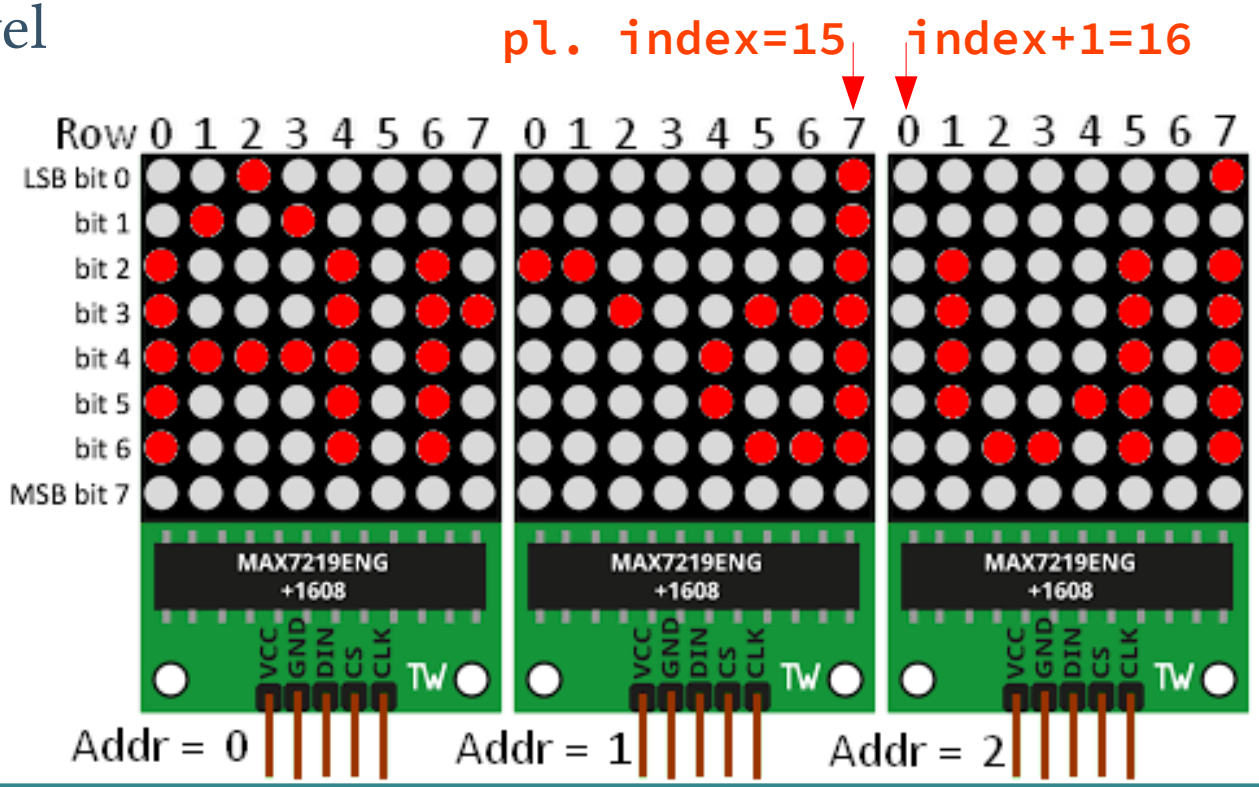
LedControl_8x8_scroll.ino (folytatás)

- Vegyük észre, hogy `showText()` és `writeChar()` megegyezik a előző előadásban mutatottakkal, `scrollinRow()` pedig alig különbözik...

```
void scrollinRow(int data) {
  for(int r=0; r<7; r++) {
    lc.setRow(0,r,lc.getRow(0,r+1));
  }
  lc.setRow(0,7,data);
}

void writeChar(char ch) {
  ch -= 32;               //Start numbering from space
  for (int line=0; line<6; line++) {
    scrollinRow(font6x8[ch][line]); //write ont row
    delay(100);
  }
}

void showText(const char *s) {
  while(*s) {
    writeChar(*s);
    s++;
  }
}
```

Szöveg görgetés több modulban

- A modul, és sorszám szerinti címzés miatt görgetésnél a futó indexből elő kell állítani a modul címét (`addr = index/8`, vagy `index>>3`), és a modulon belüli sorszámot (`row = index%8`, vagy `index & 0x07`)
- Ugyanezt ki kell számolni index + 1-re is
- Görgetés: az `index+1` által címzett *modul, sor* `getRow()`-val kiolvasott adatot az `index` által címzett *modul, sor*-ba tesszüka `setRow()` metódus segítségével



LedControl_8x8_scroll_4.ino

```
// Fényújság négy MAX7219/8x8-as LED mátrixon, LedControl használatával
#include "LedControl.h"
#include "english_6x8_pixel.h"
#define DATA_PIN    12          // MOSI -> DIN
#define CLOCK_PIN   11          // CLK  -> CLK
#define LOAD_PIN    10          // CS   -> LOAD
#define NDISP 4                 // A felfűzött modulok száma
#define NROWS 32                // A pixelsorok száma (8*NDISP)
const char line1[] = "LED8x8_MAX7219_LedControl_4 example program ";
LedControl lc=LedControl(DATA_PIN, CLOCK_PIN, LOAD_PIN, NDISP);

void setup() {
  pinMode(DATA_PIN,OUTPUT);
  pinMode(CLOCK_PIN,OUTPUT);
  pinMode(LOAD_PIN,OUTPUT);
  digitalWrite(LOAD_PIN,HIGH);
  for(int i=0; i<NDISP; i++) {
    lc.shutdown(i,false);
    lc.setIntensity(i,10);
    lc.clearDisplay(i);
  }
}

void loop() {
  showText((char*)line1);
}
```

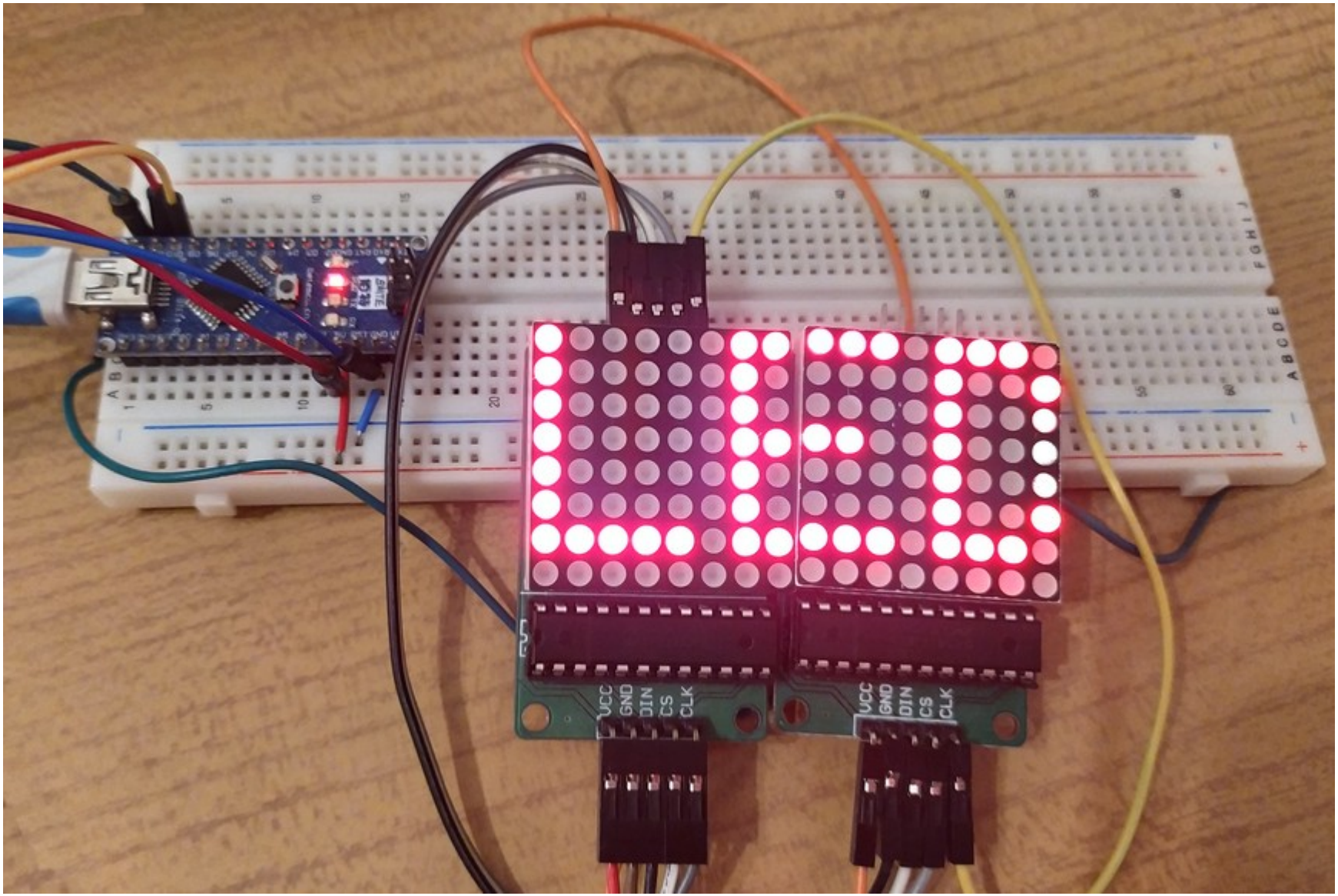
LedControl_8x8_scroll.ino_4 (folytatás)

```
void scrollinRow(int data) {
  int a0,a1,r0,r1;
  for(int r=0; r<NROWS-1; r++) {          // Itt r a futó index (0 - 30)
    a0 = r>>3;                            // a0 a cél modul száma: r/8)
    a1 = (r+1)>>3;                        // a1 s forrás modul száma: (r+1)/8
    r0 = r&0x07;                          // r0 a cél sor száma: r%8
    r1 = (r+1)&0x07;                      // r1 a forrás sor száma: (r+1)%8
    lc.setRow(a0,r0,lc.getRow(a1,r1)); // egy sor balra léptetése
  }
  lc.setRow(NDISP-1,7,data);              // az utolsó helyre beírjuk az új adatot
}

void writeChar(char ch) {
  ch -= 32;
  for (int line=0; line<6; line++) {
    scrollinRow(font6x8[ch][line]);
    delay(100);
  }
}

void showText(const char *s) {
  while(*s) {
    writeChar(*s);
    s++;
  }
}
```

Ezekben nincs változás

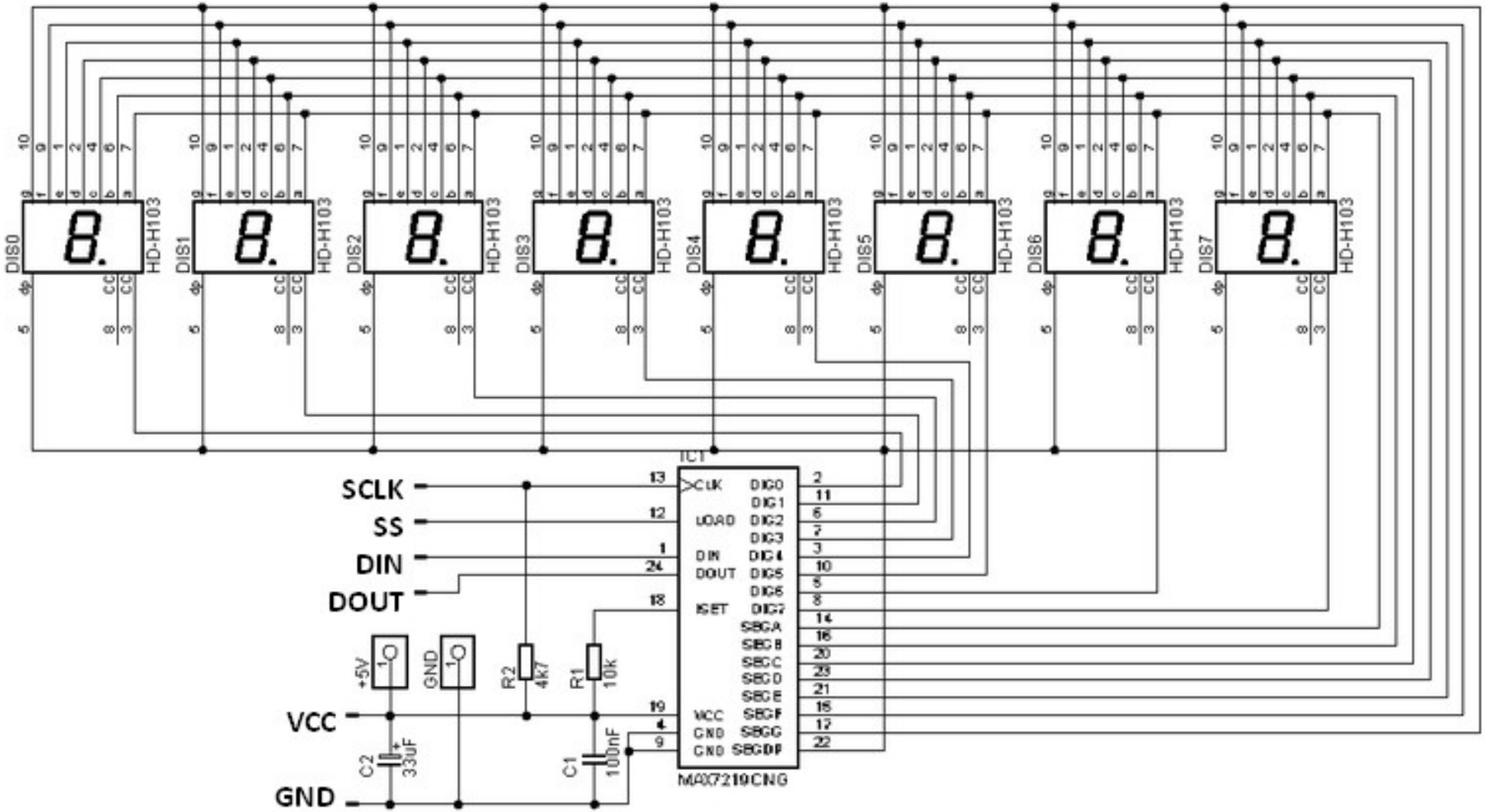
LedControl_8x8_scroll_4 futási eredmény

- Az ábrán két felfűzött modullal kapott futási eredményt láthatjuk



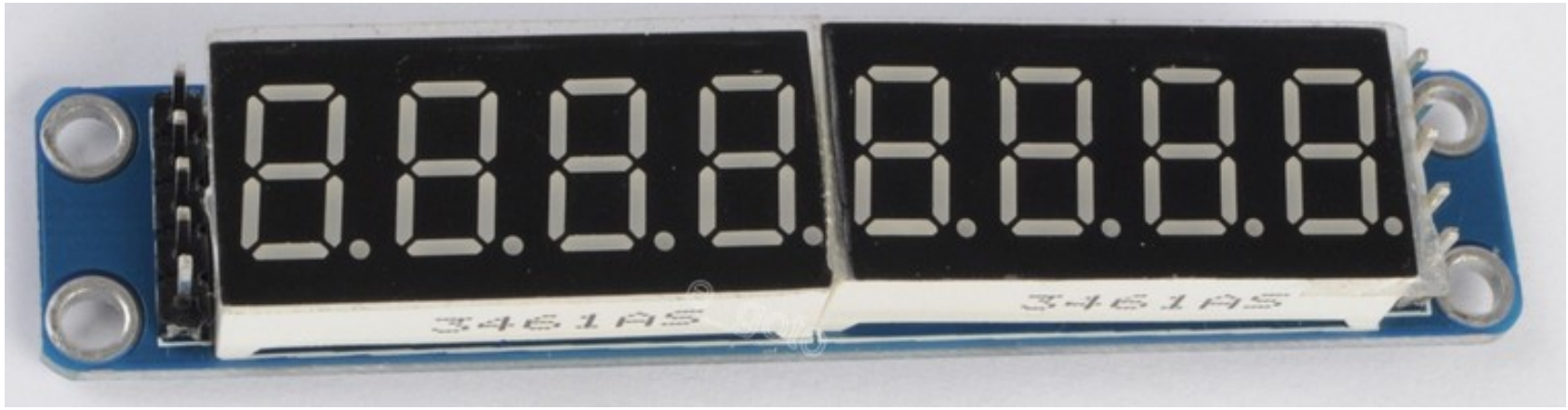
8 számjegyű hétszegmenses kijelző vezérlése

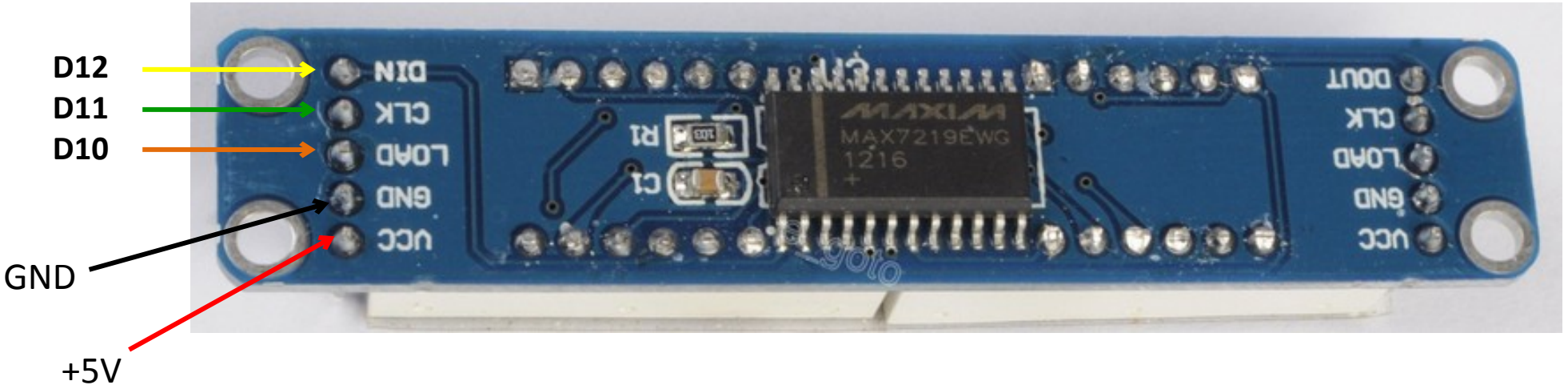
7 szegmens + DP vezérlése 8 db közös vezetéken, számjegykiválasztás másik 8 db vezetéken.
MAX7219 SPI interfésszel rendelkezik. Beépített LED áramkorlátozás és fényerőszabályozás .



Készen kapható kijelző modul

Így van bekötve: dig7 dig6 dig5 dig4 dig3 dig2 dig1 dig0





Bekötési vázlat a **LedControl_7seg.ino** példaprogramhoz (szoftveres SPI kezelés)

LedControl_7seg.ino

```
#include  <LedControl.h>
LedControl lc=LedControl(12,11,10,1); //DIN,CLK,CS és a modulok száma
unsigned long delaytime=250;
byte buffer[8];                       //Karakterek tárolója

void setup() {
    lc.shutdown(0,false);             //Modul „felébresztése”
    lc.setIntensity(0,8);             //Közepes fényerő
    lc.clearDisplay(0);               //Kijelző törlése
}

void loop() {
    writeArduinoOn7Segment();         //Arduino szöveg kiírása
    scrollDigits();                   //Számjegyek tologatása
    displayDate(2014, 6, 9);          //Adott dátum kijelzése
    for(int k=0; k<16; k++) {         //Futó idő kijelzése
      long time = millis()/1000;
      int s = time%60;
      int h = time/3600;
      int m = (time/60)%60;
      displayTime(h,m,s);
      delay(500);
    }
    lc.clearDisplay(0);               //Kijelző törlése
    delay(delaytime);
}
```

Folytatás a következő lapon. . .

LedControl_7seg.ino (folytatás)

```
void writeArduinoOn7Segment() {                    //Az Arduino szöveg kiírása
  lc.setChar(0,6,'a',false); delay(delaytime);  //a
  lc.setRow(0,5,0x05); delay(delaytime);         //r
  lc.setChar(0,4,'d',false); delay(delaytime); //d
  lc.setRow(0,3,0x1c);  delay(delaytime);        //u
  lc.setRow(0,2,B00010000); delay(delaytime);   //i
  lc.setRow(0,1,0x15);  delay(delaytime);        //n
  lc.setRow(0,0,0x1D);  delay(2000);             //o
  lc.clearDisplay(0);  delay(delaytime);         //Kijelző törlése
}

void scrollDigits() {
  int i,j;
  for(i=0; i<8; i++) buffer[i]=32;              //Buffer törlése (Space)
  for(j=0;j<16;j++) {
    for(i=0; i<7; i++) {
      buffer[i] = buffer[i+1];
      lc.setDigit(0,7-i,buffer[i],false);
    }
    buffer[7] = j;
    lc.setDigit(0,0,j,false);
    delay(delaytime);
  }
  lc.clearDisplay(0);
  delay(delaytime);
}
```

LedControl_7seg.ino (folytatás)

```
void displayDate(int year, int month, int day) {
  boolean dp;
  buffer[4]= year%10; year /=10;
  buffer[5]= year%10; year /=10;
  buffer[6]= year%10;
  buffer[7]= year/10;
  buffer[2]= month%10;
  buffer[3]= month/10;
  buffer[0]= day%10;
  buffer[1]= day/10;
  for(int i=0; i<8; i++) {
    dp = (i==2 || i==4);
    lc.setDigit(0,i,buffer[i],dp);
  }
  delay(2000);
  lc.clearDisplay(0);
  delay(delaytime);
}
```

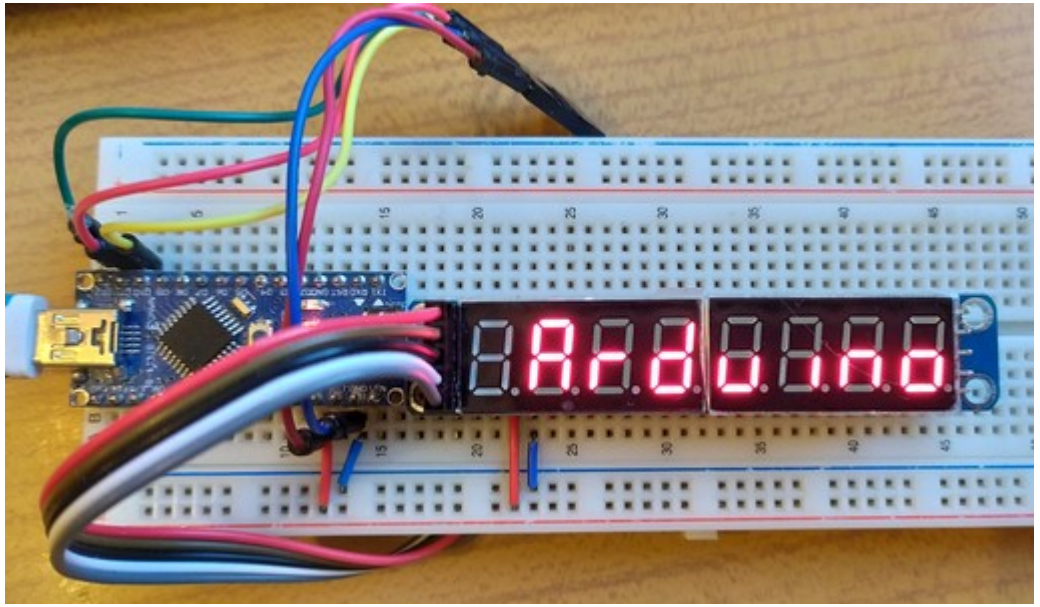
Adott dátum kiíratása **2014.06.09** formátumban.

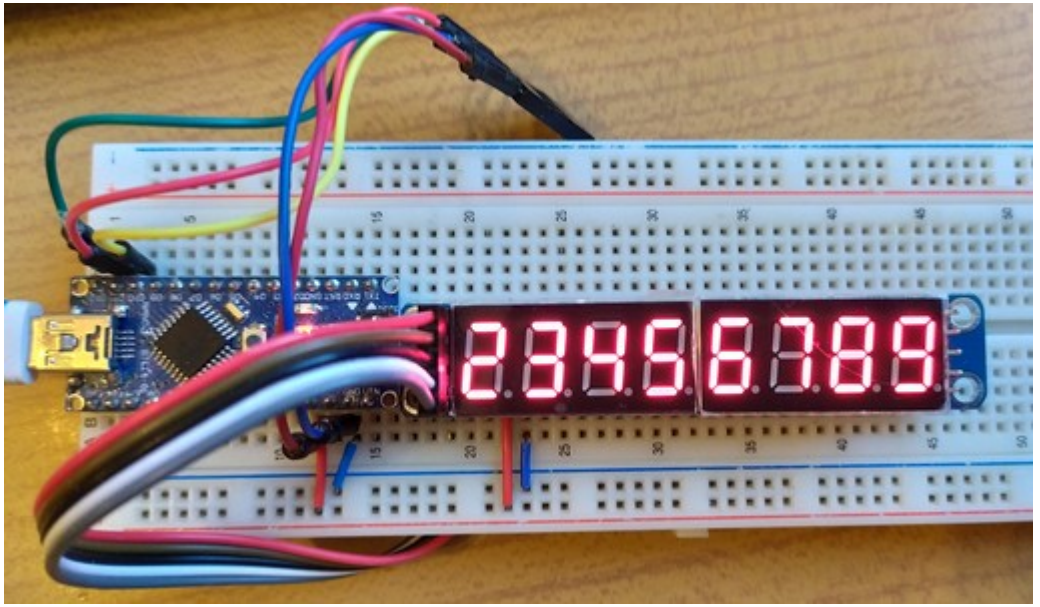
LedControl_7seg.ino (folytatás)

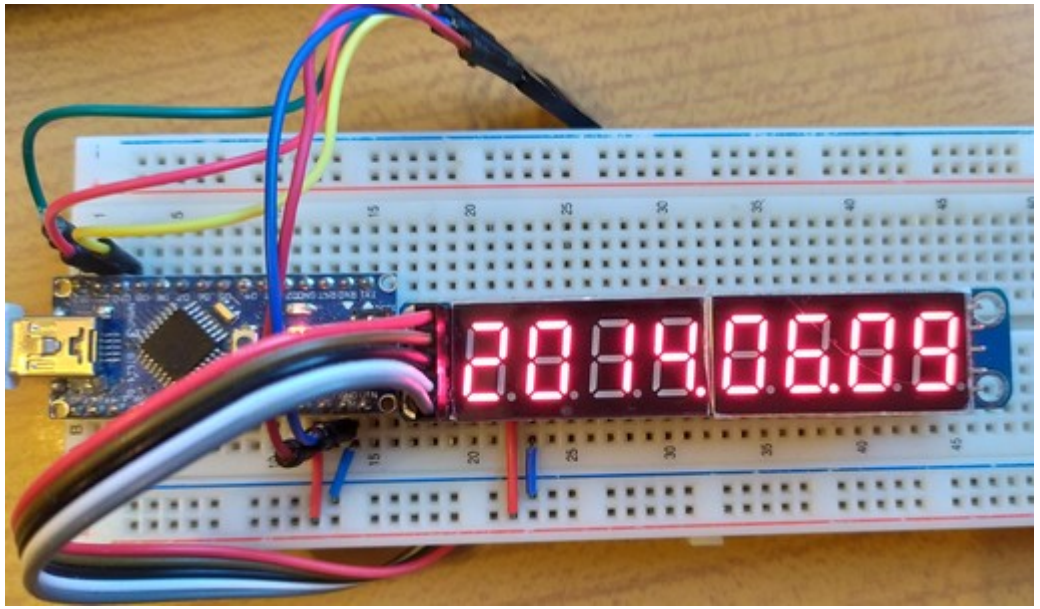
```
void displayTime(int hour, int mins, int sec) {
  buffer[7]= hour/10;
  buffer[6]= hour%10;
  buffer[5]= '-';
  buffer[4]= mins/10;
  buffer[3]= mins%10;
  buffer[2]= '-';
  buffer[1]= sec/10;
  buffer[0]= sec%10;
  for(int i=0; i<8; i++) {
    lc.setChar(0,i,buffer[i],false);
  }
  delay(delaytime);
}
```

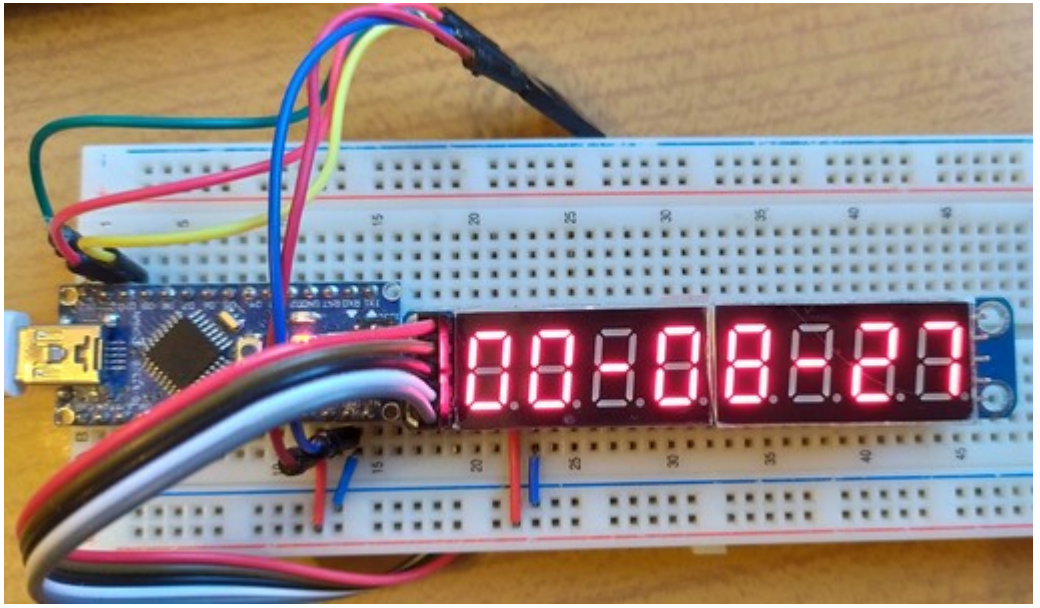
Adott idő kiíratása **HH – MM – SS** formátumban.

LedControl_7seg futási eredménye

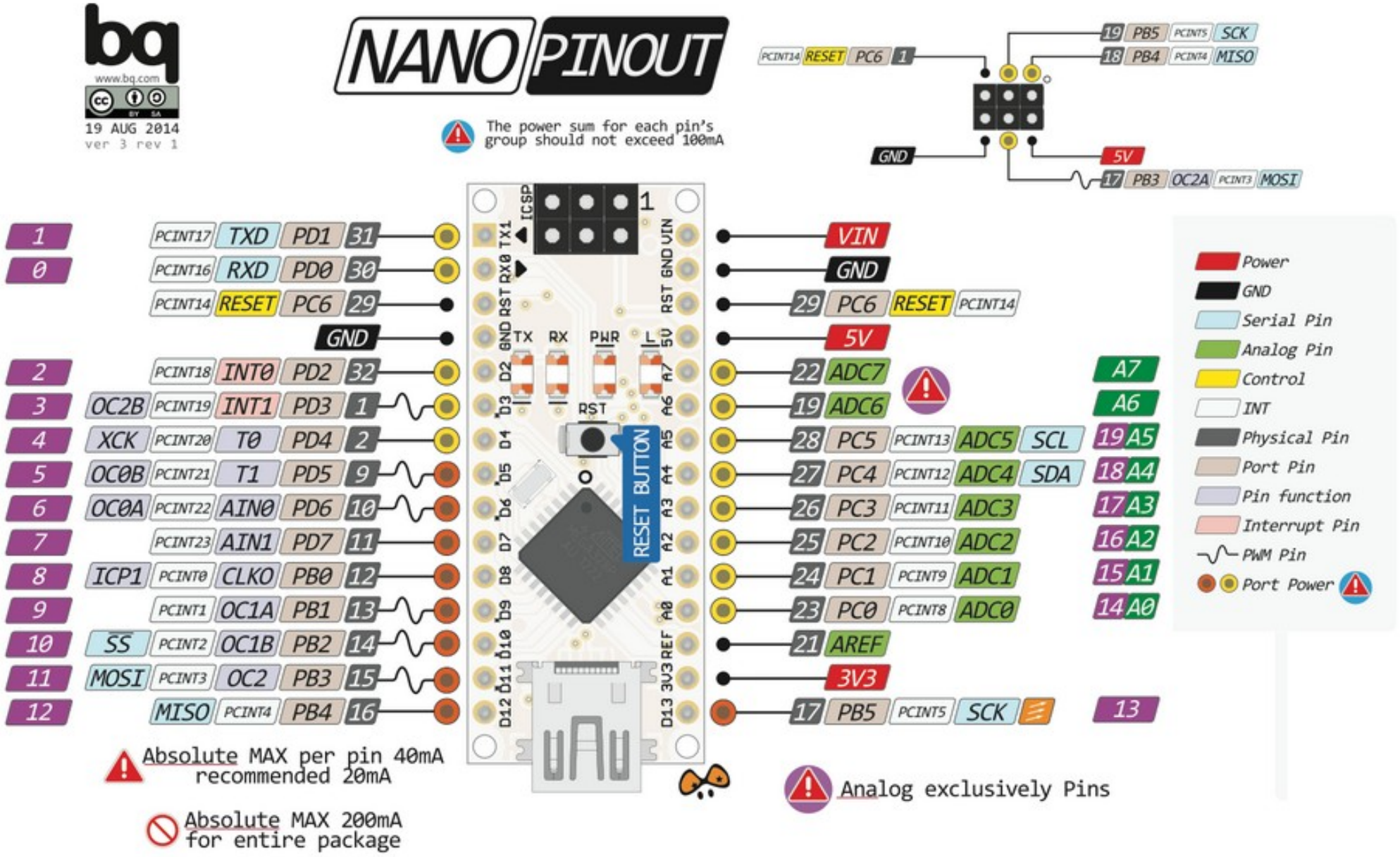








Az Arduino nano kártya kivezetései



Ellenállás színkódok