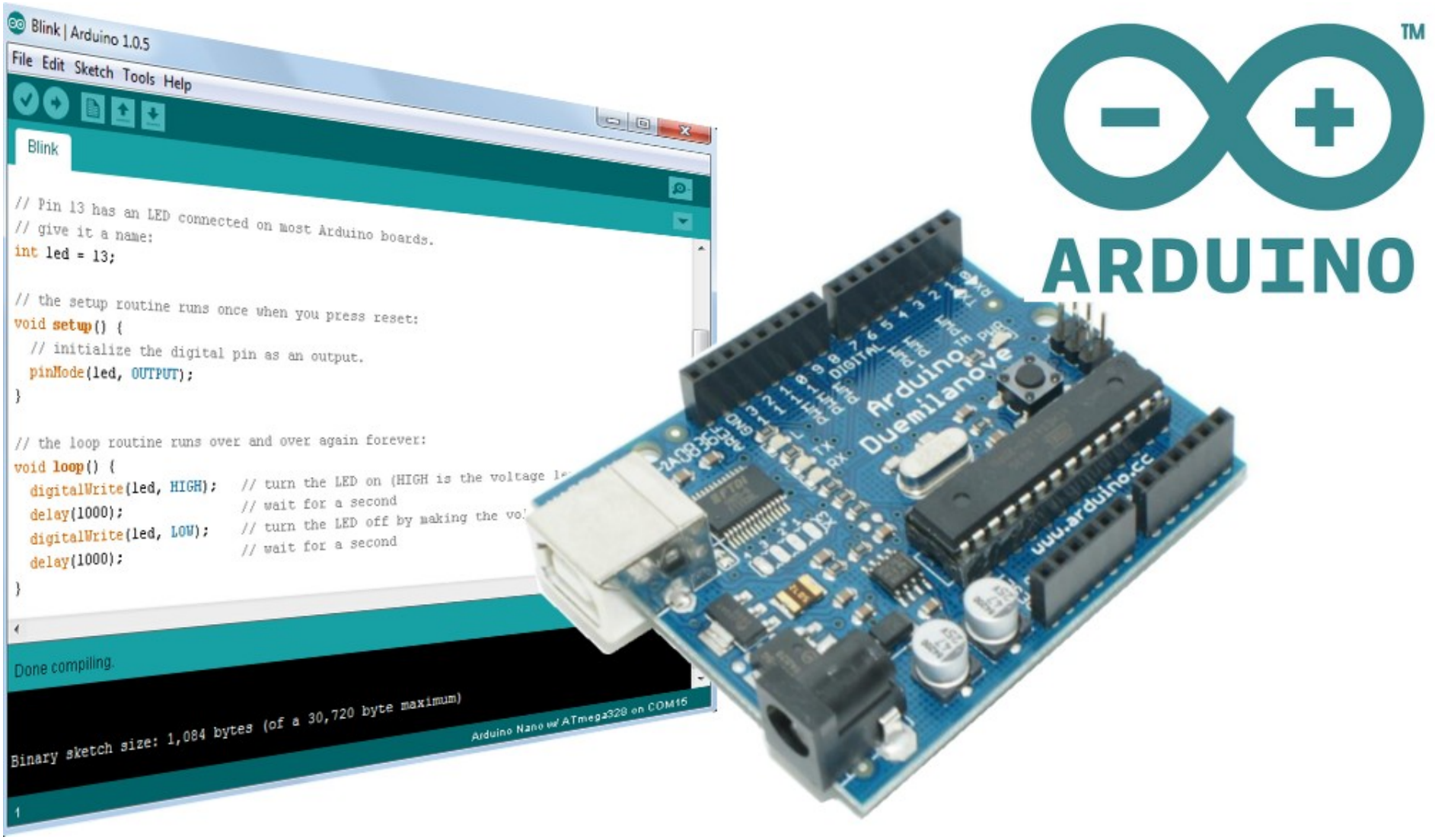
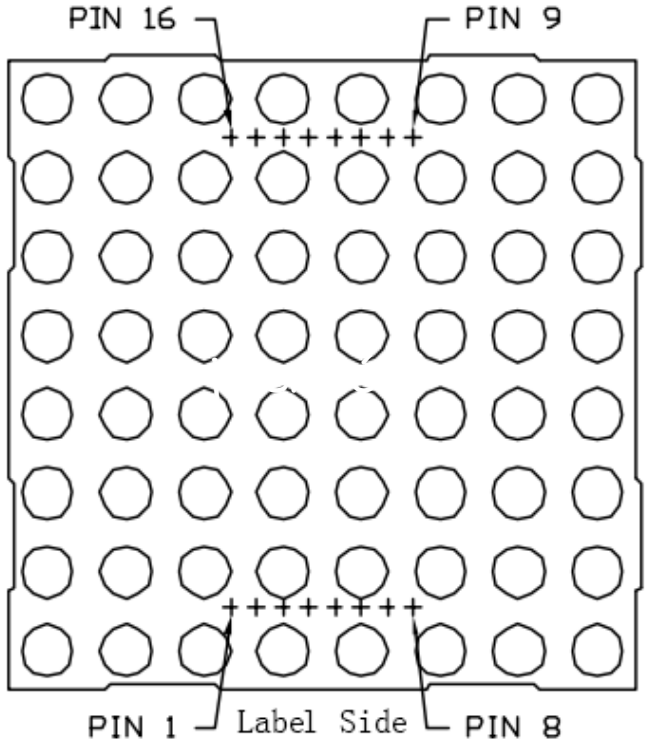
Arduino tanfolyam kezdőknek és haladóknak



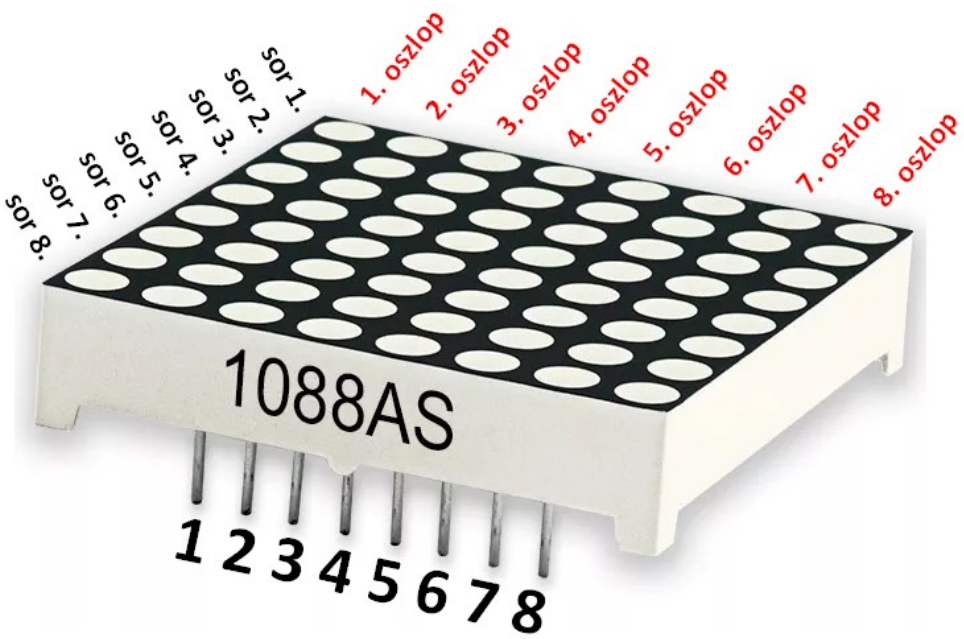
8. Ismerkedés a 8x8 LED mátrix használatával

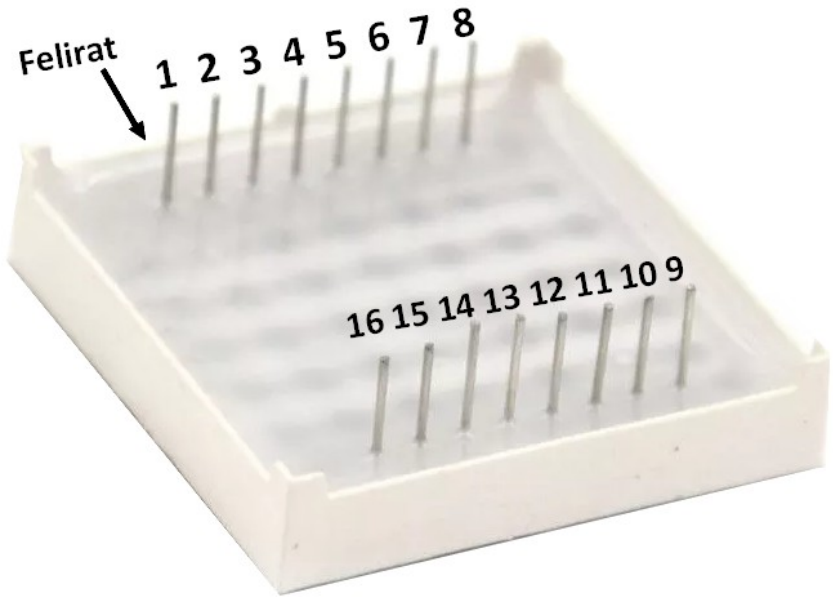
LED 8x8 mátrix

- A LED mátrix kijelző előnyei: alacsony tápfeszültség, jó fényerő és betekintési szög, hosszú élettartam, alacsony ár. Kapható egy-, ill. két színű és RGB kijelző is. Ma az egy színűvel foglalkozunk
- Többféle kivitelben kapható: a latin ABC-hez megfelel az 5x7-es mátrix, általános célra a 8x8-as mátrix, vagy annak többszörösei (16x16 vagy 32x32, főleg a távol-keleti karakterek megjelenítéséhez)
- A kivezetések két sorban helyezkednek el, számozásuk az IC-knél megszokott módon és irányban történik

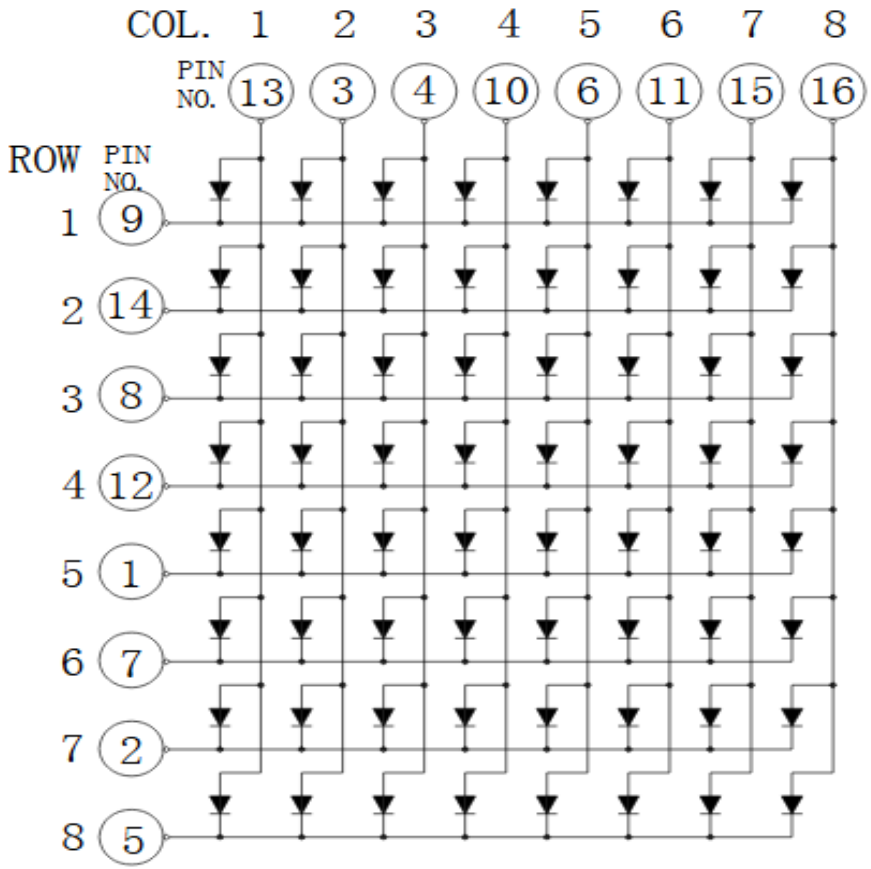


LED 8x8 mátrix



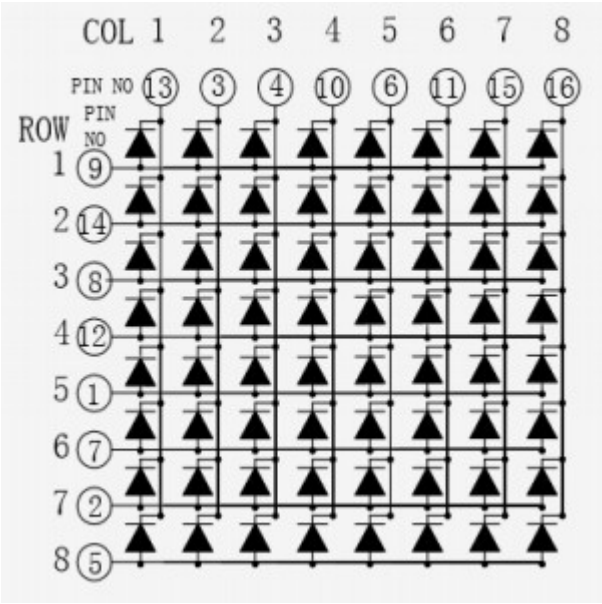


- Gyakori típus: **1088AS**
  sorok: katód, **oszlopok:** anód
- Egyidejűleg csak 1 sor vagy oszlop lehet aktív

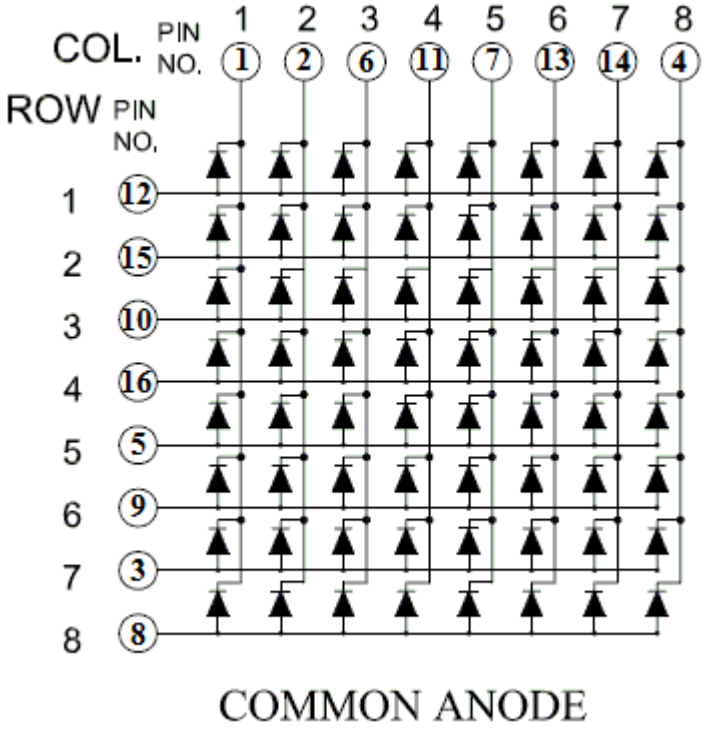


Létezik más bekötésű 8x8 LED is!

- **1088BS:** itt a sorok az anódok és az oszlopok a katódok



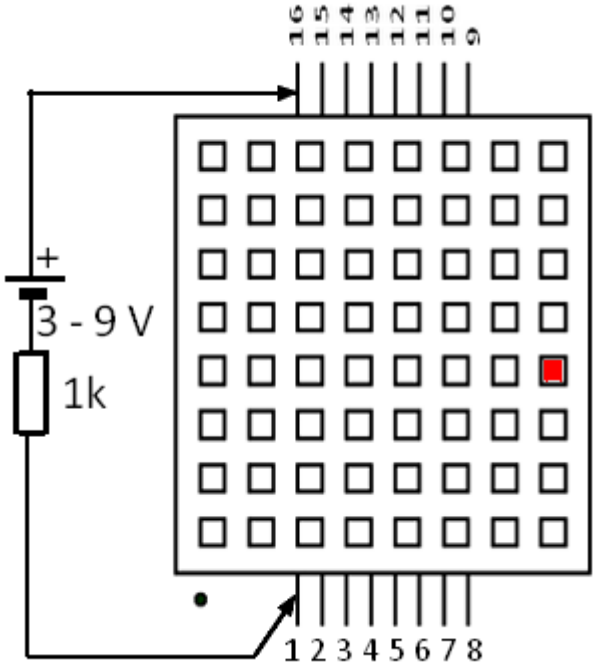
- **1388ASR:** itt is a sorok az anódok és az oszlopok a katódok, de más a lábkiosztás!

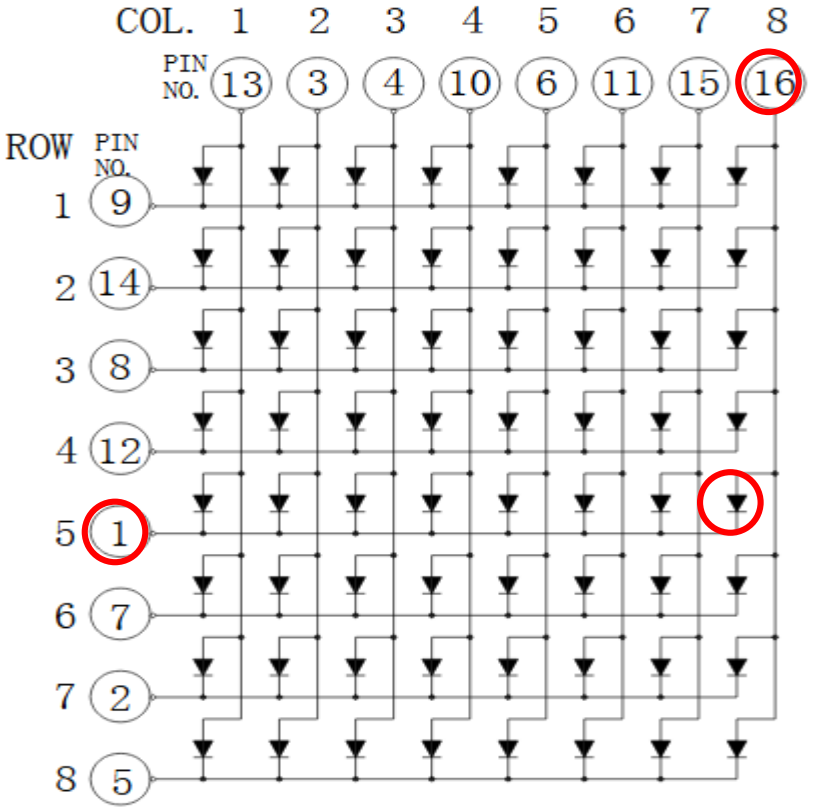


… és még sokan mások!

Hogyan ellenőrizhetjük a kivezetéseket?

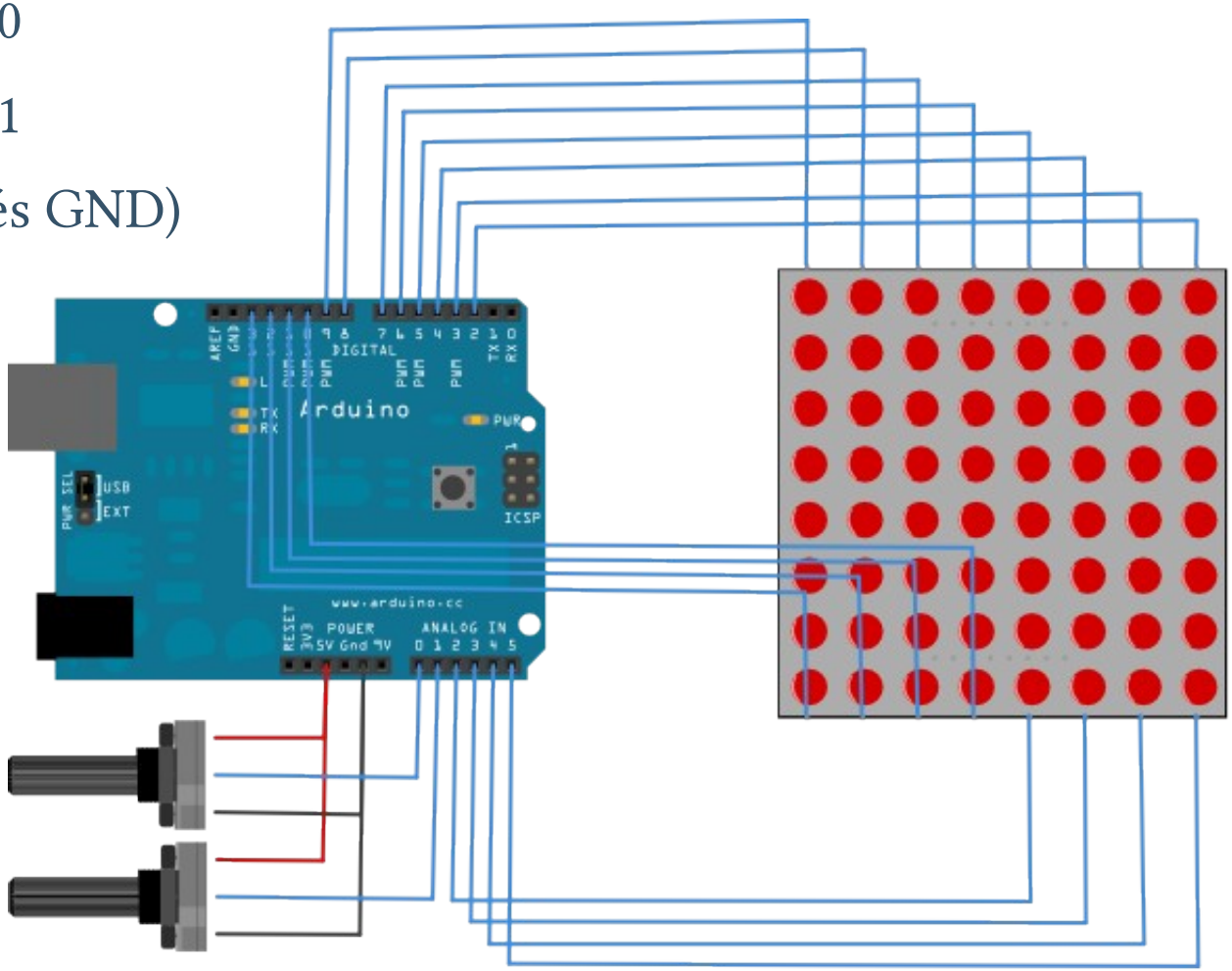
- Egy 3 – 9 V közötti feszültségű áramforrás és egy **sorbakötött 1 kΩ-os ellenállás** segítségével meghajthatjuk a LED-eket. Ha pl. a **1088AS** esetén a negatív pólust az 1-es lábhoz csatlakoztatjuk, a többi lábon „végigmenve” megtaláljuk az oszlopok kivezetéseit
- A maradék kivezetések a sorokhoz tartoznak, csak a sorrendjüket kell megtalálni





A kísérleti kapcsolás

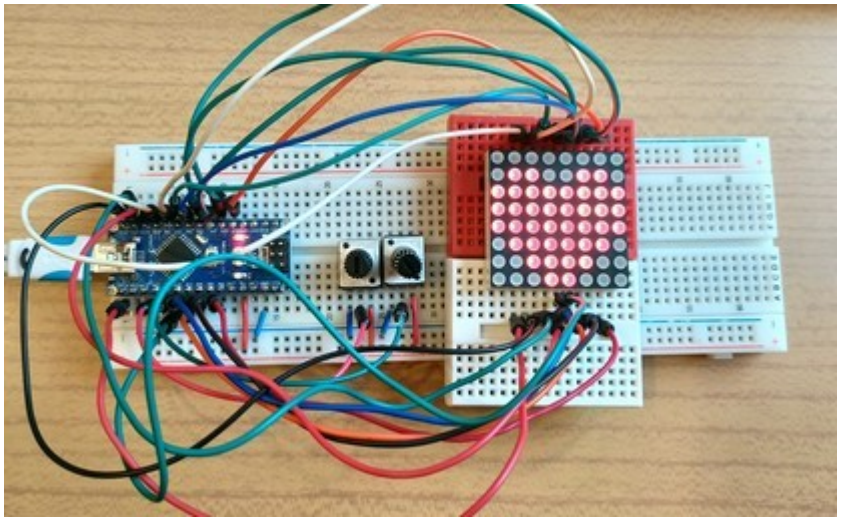
- **1088AS:** 1 – D13, 2 – D12, 3 – D11, 4 – D10, 5 – A2, 6 – A3, 7 – A4, 8 – A5, 9 – D2, 10 – D3, 11 – D4, 12 – D5, 13 – D6, 14 – D7, 15 – D8, 16 – D9
- 1. potméter csúszka – A0
- 2. potméter csúszka – A1
- (Potméterek végei +5V és GND)

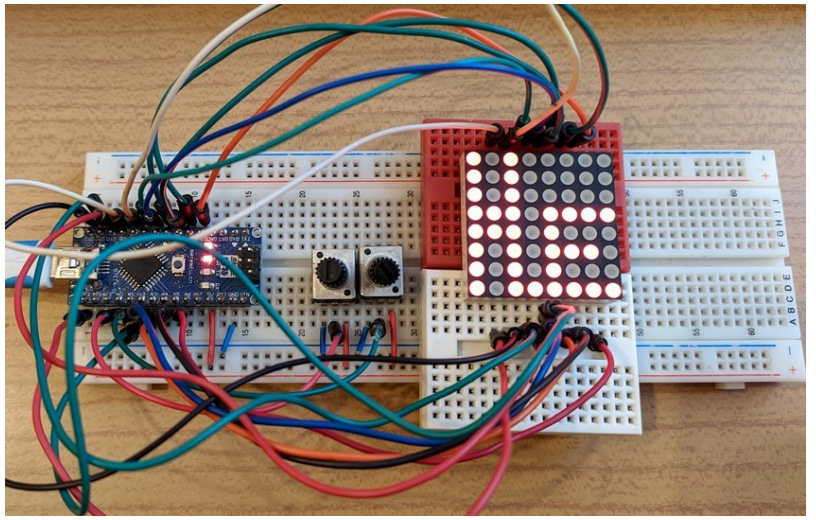


- Forrás: https://create.arduino.cc/projecthub/Hirusha234/controlling-8x8-matrix-0c36d8

Mintaprogramok

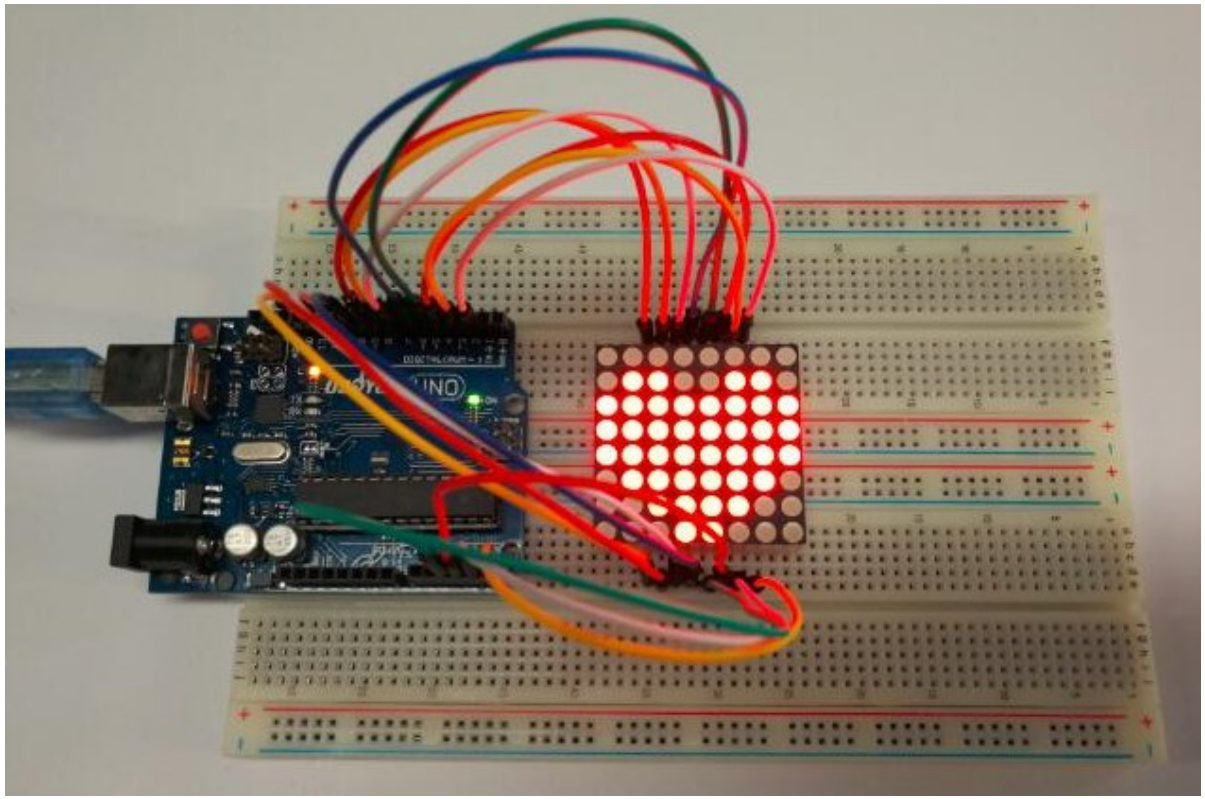
- **led8x8_blinkheart** – animált szívecske megjelenítése
- **led8x8_firka** – a potméterek segítségével „rajzolhatunk”
- **led8x8_scrolltext** – szöveg görgetése szoftveres időzítéssel
- **led8x8_scrolltext2** – szöveg görgetése **MsTimer2** időzítéssel





A led8x8_blinkheart projekt

- Az eredeti projekt az osoyoo.com honlapján található, de át kellett dolgozni, hogy a **1088AS** típusú kijelzőnkhöz jó legyen, s könnyen lehessen váltani a közös katódú, illetve közös anódú típusok között
- A kapcsolás megegyezik a 6. oldalon bemutatottal, de a potmétereket most nem használjuk



Kattintson a képre az animáció megnyitásához!

led8x8_blinkheart.ino

```
/* A program jelenlegi formájában az 1088AS 8x8 LED
 * kijelzőhöz készült. Fordított polaritású (pl. 1088BS)
 * kijelző esetén a rON, rOFF, CON, cOFF konstansokat
 * logikailag negálni kell (0 helyett 1, 1 helyett 0)
 */
#define rON  0      // sor aktiválás (1088AS esetén katód:0)
#define rOFF 1      // sor deaktiválás
#define cON  1      // oszlop aktiválás (1088AS esetén anód:1)
#define cOFF 0      // oszlop deaktiválás

int row[] = {2, 7, A5, 5, 13, A4, 12, A2}; // 9,14,8,12,1,7,2,5
int col[] = {6, 11, 10, 3, A3, 4, 8, 9};   // 13,3,4,10,6,11,15,16

//--- nagy "szív" bitminta ---
unsigned char biglove[8][8] =
{
  0, 0, 0, 0, 0, 0, 0, 0,
  0, 1, 1, 0, 0, 1, 1, 0,
  1, 1, 1, 1, 1, 1, 1, 1,
  1, 1, 1, 1, 1, 1, 1, 1,
  1, 1, 1, 1, 1, 1, 1, 1,
  0, 1, 1, 1, 1, 1, 1, 0,
  0, 0, 1, 1, 1, 1, 0, 0,
  0, 0, 0, 1, 1, 0, 0, 0,
};
```

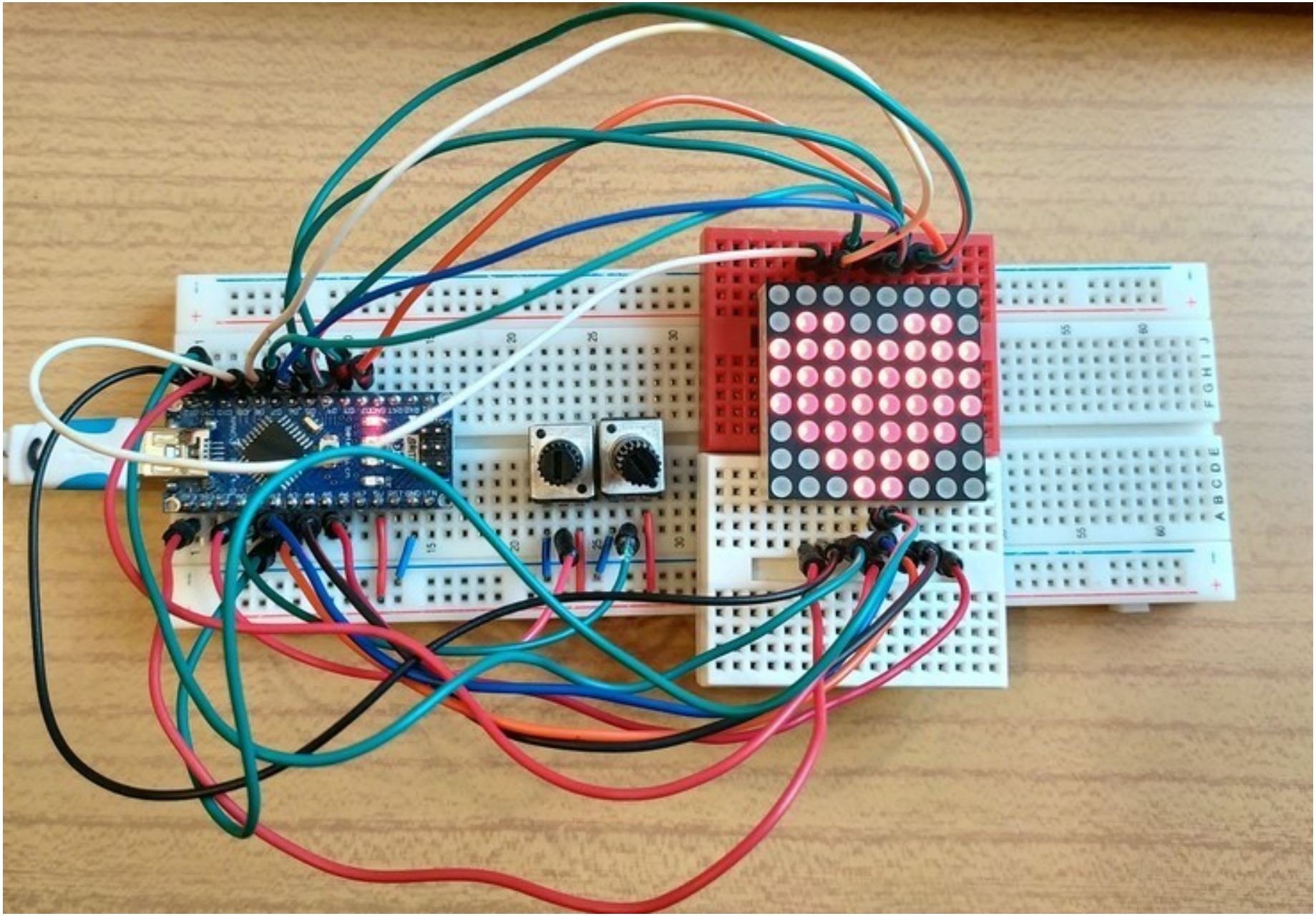
```
//--- kis "szív" bitminta ---
unsigned char biglove[8][8] =
{
  0, 0, 0, 0, 0, 0, 0, 0,
  0, 0, 0, 0, 0, 0, 0, 0,
  0, 0, 1, 0, 0, 1, 0, 0,
  0, 1, 1, 1, 1, 1, 1, 0,
  0, 1, 1, 1, 1, 1, 1, 0,
  0, 0, 1, 1, 1, 1, 0, 0,
  0, 0, 0, 1, 1, 0, 0, 0,
  0, 0, 0, 0, 0, 0, 0, 0,
};
```

led8x8_blinkheart.ino

```
void setup() {
  for (int i = 0; i < 8; i++) {
    pinMode(row[i], OUTPUT);              // sor vezérlő kimenetek
    pinMode(col[i], OUTPUT);              // oszlop vezérlő kimenetek
  }
}

void loop() {
  for (int i = 0 ; i < 1000 ; i++) {
    Display(biglove);                     //Display the "Big Heart"
  }
  for (int i = 0 ; i < 500 ; i++) {
    Display(smalllove);                   //Display the "small Heart"
  }
}

//--- Megjelenítés (a kép felvillantása) -------
void Display(unsigned char dat[8][8]) {
  for (int c = 0; c < 8; c++) {           // Sorra vesszük az oszlopokat
    digitalWrite(col[c], cON);            // Aktuális oszlop aktiválása
    for (int r = 0; r < 8; r++) {         // Sorra vesszük a sorokat
      digitalWrite(row[r],dat[r][c]^rOFF); // Képpont felvillantása
      digitalWrite(row[r],rOFF);
    }
    digitalWrite(col[c], cOFF);           // oszlop deaktiválása
  }
}
```

Negálás, ha rOFF = 1, hogy dat[r,c]=1 aktiváljon!

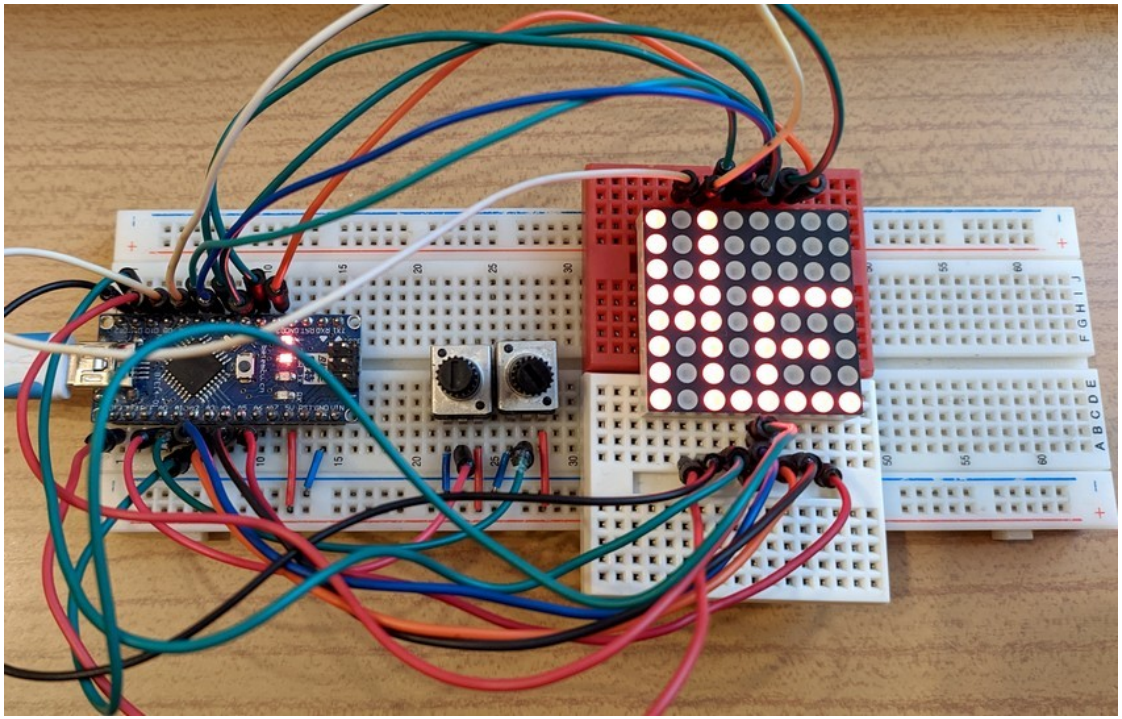
A led8x8_blinkheart projekt eredménye



A led8x8_firka projekt

- Az ötletadó eredeti projekt szintén az osoyoo.com honlapján található, de teljesen átdolgoztuk, hogy:
  - a **1088AS** típusú kijelzőnkhöz jó legyen
  - könnyen átszabható legyen az eltérő kijelzőkhöz
  - vonalakat rajzolhassunk a mágnestáblához hasonlóan (törlés újraindítással)
  - a potméterekkel vezérelt kurzor villogjon

- Ezen a képen egy futás eredménye látható, melynek során egy **Hobbielektronika** logót rajzoltunk



led8x8_firka.ino

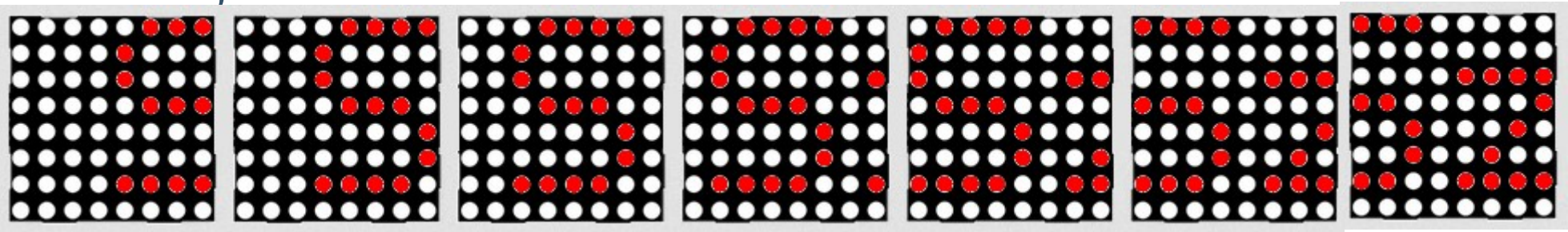
```
#define rON  0      // sor aktiválás (1088AS esetén katód:0)
#define rOFF 1      // sor deaktiválás
#define cON  1      // oszlop aktiválás (1088AS esetén anód:1)
#define cOFF 0      // oszlop deaktiválás

int row[] = {2, 7, A5, 5, 13, A4, 12, A2}; // 9,14,8,12,1,7,2,5
int col[] = {6, 11, 10, 3, A3, 4, 8, 9};   // 13,3,4,10,6,11,15,16
int x = 0;          // Kurzor pozíció
int y = 0;

unsigned char pixels[8][8] = {
  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
};

void setup() {
  for (int i = 0; i < 8; i++) {
    pinMode(row[i], OUTPUT);            // sor vezérlő kimenetek
    pinMode(col[i], OUTPUT);            // oszlop vezérlő kimenetek
  }
}
```

led8x8_firka.ino

```
void loop() {
  x = map(analogRead(A0),0,1023,0,7);    // X pozíció beolvasása (sor)
  y = map(analogRead(A1),0,1023,0,7);    // Y pozíció beolvasása (oszlop)
  pixels[x][y] = 0;                      // Pixel kioltása
  for (int i = 0 ; i < 100; i++) {
    Display(pixels);                     // Az aktuális kép megjelenítése
  }                                        A kurzor villogtatása
  pixels[x][y] = 1;                      // Pixel aktiválása
  for (int i = 0 ; i < 100; i++) {
    Display(pixels);                     // Az aktuális kép megjelenítése
  }
}

//--- Megjelenítés (a kép felvillantása) -------
void Display(unsigned char dat[8][8]) {    Ugyanaz, mint az előző programban!
  for (int c = 0; c < 8; c++) {          // Sorra vesszük az oszlopokat
    digitalWrite(col[c], cON);           // Aktuális oszlop aktiválása
    for (int r = 0; r < 8; r++) {        // Sorra vesszük a sorokat
      digitalWrite(row[r],dat[r][c]^rOFF); // Képpont felvillantása
      digitalWrite(row[r],rOFF);
    }
    digitalWrite(col[c], cOFF);          // oszlop deaktiválása
  }
}
```

Szöveg megjelenítése és görgetése

- A 8x8-as bitképet most nem pontonként kezeljük, hanem oszloponként 1-1 bájtként: `byte pixels[8];`
- A karakterek 6x8 pixeles képeit egy-egy bájtos tömbben tároljuk, s a fontkészletet a `font6x8[][6]` „kétdimenziós” tömbben (valójában a tömbök tömbje…) tároljuk
- A karakterek megjelenítését pixeloszloponként végezzük, közben kb. 250 ms várakozást (és képfrissítést!) iktatunk be
- A `pixels[]` tömb 1–7. elemét eggyel balra léptetjük, majd az üresen maradó oszlopba a karakter soron következő pixeloszlopát írjuk be
- Betűköz: a 6x8-as karaktermátrix első oszlopa általában nulla ez biztosítja a térközt a karakterek között



led8x8_scrolltext.ino

```
#include "font6x8.h"

#define rON  0      // sor aktiválás (1088AS esetén katód:0)
#define rOFF 1      // sor deaktiválás
#define cON  1      // oszlop aktiválás (1088AS esetén anód:1)
#define cOFF 0      // oszlop deaktiválás

int row[] = {2,7,A5,5,13,A4,12,A2};  // 9,14,8,12,1,7,2,5
int col[] = {6,11,10,3,A3,4,8,9};    // 13,3,4,10,6,11,15,16

byte pixels[8];                      // Ebben a tömbben egy byte egy oszlopot ír le

//--- Megjelenítés (a kép felvillantása) -------
void refreshScreen() {
  for (int c = 0; c < 8; c++) {      // Sorra vesszük az oszlopokat
    digitalWrite(col[c], cON);       // Aktuális oszlop aktiválása
    byte thisByte = pixels[c];       // Pixel oszlop elővétele
    for (int r = 0; r < 8; r++) {    // Sorra vesszük a sorokat
      if (bitRead(thisByte,r)) {     // Ha a pixel értéke = 1
        digitalWrite(row[r],rON);    // Képpont felvillantása
      }
      digitalWrite(row[r],rOFF);
    }
    digitalWrite(col[c],cOFF);       // oszlop deaktiválása
  }
}
```

bitRead - gyári makró, megadott bit vizsgálata:

#define bitRead(value, bit) (((value) >> (bit)) & 0x01)

led8x8_scrolltext.ino

```
//--- Mátrix görgetés balra --------
void scrollLeft(byte data) {
  for (byte i = 0; i < 7; i++) {      // Nem 8, hanem csak 7 !!!
    pixels[i] = pixels[i + 1];        // Minden pixeloszlop balra lép
  }
  pixels[7] = data;                   // Az utolsó oszlop az új adat
}

//--- Törli a pixel mátrixot -------
void clearMatrix() {
  for (byte i = 0; i < 8; i++) {
    pixels[i] = 0;
  }
}

//--- Egy karakter kiírása ---------
void writeChar(char ch) {
  ch -= 32;                           // szóköz (32) az első karakter a font6x8 tömbben
  for (int line = 0; line < 6; line++) {
    scrollLeft(font6x8[ch][line]);    // Egy pontoszlop beszúrása jobbról
    for(int i=0; i<250; i++) {
      refreshScreen();                // Képfrissítés 250-szer (várakozás)
    }
  }
}
```

led8x8_scrolltext.ino

- A szöveg kiírásánál addig írjuk ki egymás után a karaktereket, amíg a lezáró nulla kódig el nem jutunk

```
//--- Szöveg kiírása karakterenként ----
void showText(char *s) {                  // s egy mutató, az adat memóriacímét tartalmazza
  while (*s) {                            // s: karakterre mutat, *s: a mutatott karakter
    writeChar(*s);                        // Az ütemezés a writeChar függvényen belül
    s++;                                  // történik, itt nem kell foglalkoznunk vele
  }
}

void setup() {
  for (int i = 0; i < 8; i++) {
    pinMode(row[i], OUTPUT);              // sor vezérlő kimenetek
    pinMode(col[i], OUTPUT);              // oszlop vezérlő kimenetek
  }
  clearMatrix();
}

void loop() {
  showText("Szia, Pista vagyok. ");  // Az ilyen hívás egy mutatót ad át, ami a
  showText("LED8x8 scrolltext example  "); // nullával lezárt karakterfüzérre mutat
  clearMatrix();
}
```

Mutatók és szövegkonstansok

- **A mutatók** (pointers) olyan változók, amelyek nem az adatot, hanem az adat elérési címét tartalmazzák (egy memóriacímet)
- Hivatkozás operátor: &, a változó címét adja meg, pl. &i
- Hivatkozás feloldás operátor: *, az adott címen található adat

```
int *p;          // Egész típusú adatra mutató pointert deklarálunk
int i = 5, result = 0;  // Egész típusú változókat deklarálunk
p = &i;          // 'p' itt már az 'i' változó címét tartalmazza
result = *p;     // 'result' felveszi a 'p' által mutatott értéket (5-öt)
```

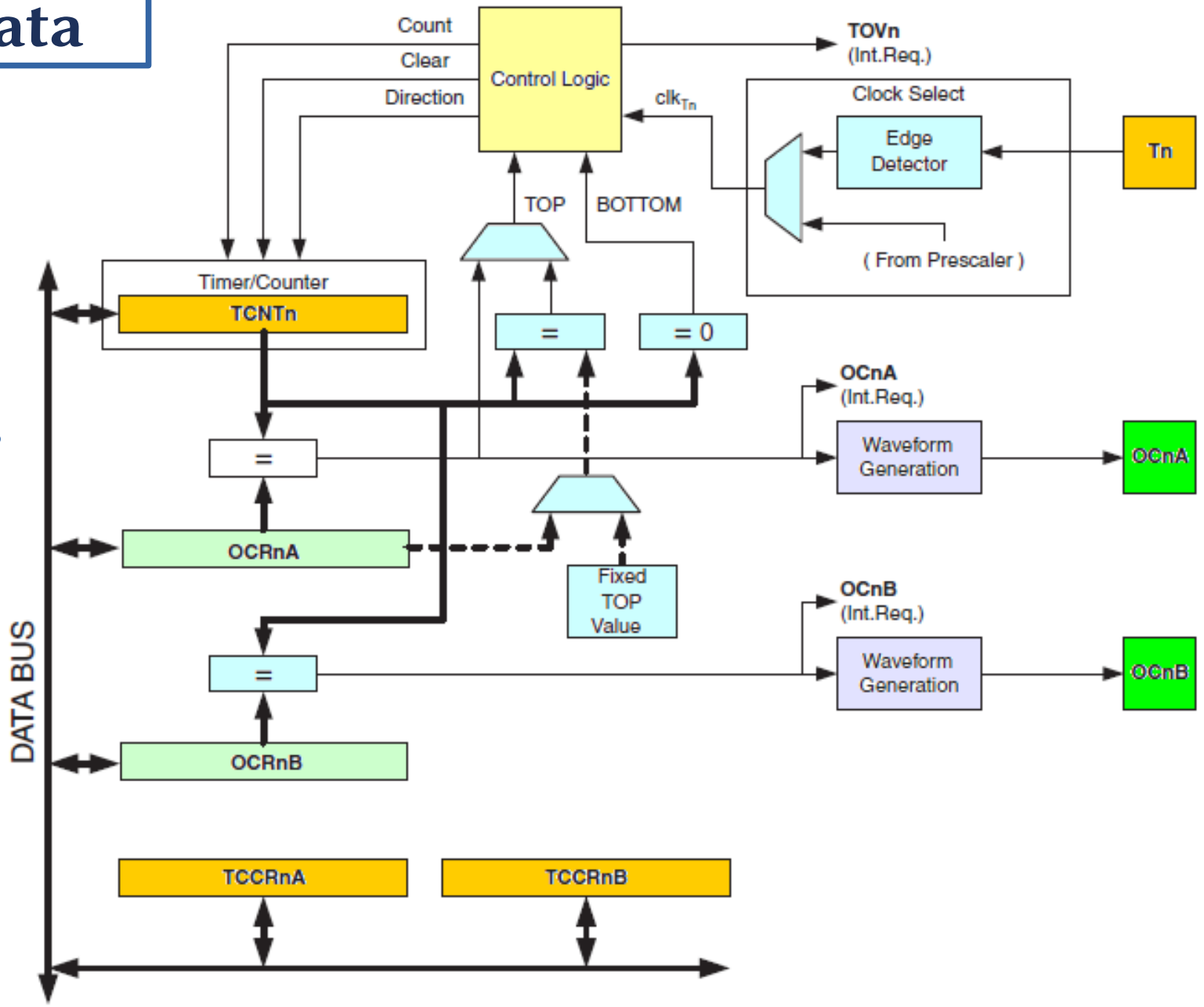
- **Szövegkonstans:** idézőjelek közé zárt karaktersorozat, melyhez a fordító letároláskor egy végjelző nullát hozzáfűz
- Az alábbi deklarációk azonos eredményre vezetnek:

```
char Str1[8] = {'a', 'r', 'd', 'u', 'i', 'n', 'o', '\0'};
char Str2[] = "arduino";
char Str3[8] = "arduino";
```

} Szövegkonstansok

Timer2 blokkvázlata

8-bites számláló
10-bites előosztó
két 8-bites Output Compare csatorna
Megszakítások:
TOVF, OCFA, OCFB
2 vezérlő regiszter:
TCCR2A, TCCR2B



Timer2 regiszterek

- **TCCR2A** – Timer/Counter2 vezérlő regiszter *A*

7	6	5	4	3	2	1	0	
COM2A1	COM2A0	COM2B1	COM2B0	–	–	WGM21	WGM20	TCCR2A
R/W	R/W	R/W	R/W	R	R	R/W	R/W	
0	0	0	0	0	0	0	0	

COM2A1	COM2A0	Description
0	0	Normal port operation, OC0A disconnected.
0	1	Toggle OC2A on Compare Match
1	0	Clear OC2A on Compare Match
1	1	Set OC2A on Compare Match

COM2B1	COM2B0	Description
0	0	Normal port operation, OC2B disconnected.
0	1	Toggle OC2B on Compare Match
1	0	Clear OC2B on Compare Match
1	1	Set OC2B on Compare Match

Timer2 regiszterek

- **TCCR2B** – Timer/Counter2 vezérlő regiszter ***B***

Bit	7	6	5	4	3	2	1	0	
(0xB1)	FOC2A	FOC2B	–	–	WGM22	CS22	CS21	CS20	TCCR2B
Read/Write	W	W	R	R	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

CS22	CS21	CS20	Description	
0	0	0	No clock source (Timer/Counter stopped).	
0	0	1	clk_{T2S}/(No prescaling)	16 MHz
0	1	0	clk_{T2S}/8 (From prescaler)	2 MHz
0	1	1	clk_{T2S}/32 (From prescaler)	**250 kHz**
1	0	0	clk_{T2S}/64 (From prescaler)	62.5 kHz
1	0	1	clk_{T2S}/128 (From prescaler)	15 625 Hz
1	1	0	clk_{T2S}/256 (From prescaler)	
1	1	1	clk_{T2S}/1024 (From prescaler)	

- ***CS22, CS21, CS20*** – az előosztási arányt itt állíthatjuk be például 16 MHz-es CPU frekvencia esetén 1/64 leosztás és 250-ig történő számolással **Timer2** pontosan 1 ms-onként csordul túl

Timer2 regiszterek

- **WGM2[2:0]** – Hullámforma generátor üzemmód bitek, megszabják a számlálás maximum értékét és a generált hullámformát

Mode	WGM22	WGM21	WGM20	Timer/Counter Mode of Operation	TOP	Update of OCRx at	TOV Flag Set on(1)(2)
0	0	0	0	Normal	0xFF	Immediate	MAX
1	0	0	1	PWM, Phase Correct	0xFF	TOP	BOTTOM
2	0	1	0	CTC	OCRA	Immediate	MAX
3	0	1	1	Fast PWM	0xFF	BOTTOM	MAX
4	1	0	0	Reserved	–	–	–
5	1	0	1	PWM, Phase Correct	OCRA	TOP	BOTTOM
6	1	1	0	Reserved	–	–	–
7	1	1	1	Fast PWM	OCRA	BOTTOM	TOP

Notes: 1. MAX= 0xFF
2. BOTTOM= 0x00

Timer2 regiszterek

TCNT2 – Timer/Counter Register

Timer2 számláló regiszter

Bit	7	6	5	4	3	2	1	0	
(0xB2)	TCNT2[7:0]								TCNT2
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	

OCR2A – Output Compare Register A

Bit	7	6	5	4	3	2	1	0	
(0xB3)	OCR2A[7:0]								OCR2A
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	

OCR2B – Output Compare Register B

Bit	7	6	5	4	3	2	1	0	
(0xB4)	OCR2B[7:0]								OCR2B
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	

TIMSK2 – Timer/Counter2 Interrupt Mask Register

Bit	7	6	5	4	3	2	1	0	
(0x70)	–	–	–	–	–	OCIE2B	OCIE2A	TOIE2	TIMSK2
Read/Write	R	R	R	R	R	R/W	R/W	R/W	

Timer2 megszakítás engedélyező bit

TIFR2 – Timer/Counter2 Interrupt Flag Register

Bit	7	6	5	4	3	2	1	0	
0x17 (0x37)	–	–	–	–	–	OCF2B	OCF2A	TOV2	TIFR2
Read/Write	R	R	R	R	R	R/W	R/W	R/W	

Timer2 túlcsordulásjelző bit

Az MsTimer2 programkönyvtár

- Az **MsTimer2** egy könnyen használható programkönyvtár, amely a **Timer2** időzítő felhasználásával *ms* felbontású időzítést tesz lehetővé. Link: playground.arduino.cc/Main/MsTimer2/
- Az **MsTimer2** programkönyvtár használata esetén **Timer2** átprogramozása miatt nem használhatjuk a **Tone()** függvényt!

```
#include <MsTimer2.h>
void flash() {
  static boolean output = HIGH;
  digitalWrite(13, output);
  output = !output;
}

void setup() {
  pinMode(13, OUTPUT);
  MsTimer2::set(500, flash); // 500ms periódus
  MsTimer2::start();
}

void loop() {

}
```

Példaprogram:

A beépített LED villogtatásának ütemezése **MsTimer2** segítségével

MsTimer2.cpp

MsTimer2 forráskódja
A miénktől eltérő CPU-kra vonatkozó feltételes fordításokat kihagytuk belőle

```
#include <MsTimer2.h>

unsigned long MsTimer2::msecs;
void (*MsTimer2::func)();
volatile unsigned long MsTimer2::count;
volatile char MsTimer2::overflowing;
volatile unsigned int MsTimer2::tcnt2;

void MsTimer2::set(unsigned long ms, void (*f)()) {
    float prescaler = 0.0;
    TIMSK2 &= ~(1<<TOIE2);
    TCCR2A &= ~((1<<WGM21) | (1<<WGM20));
    TCCR2B &= ~(1<<WGM22);
    ASSR &= ~(1<<AS2);
    TIMSK2 &= ~(1<<OCIE2A);
    TCCR2B |= (1<<CS22);
    TCCR2B &= ~((1<<CS21) | (1<<CS20));
    prescaler = 64.0;
    tcnt2 = 256 - (int)((float)F_CPU * 0.001 / prescaler);
    if (ms == 0) msecs = 1;
    else msecs = ms;
    func = f;
}
```

MsTimer2.cpp

```
void MsTimer2::start() {
    count = 0;
    overflowing = 0;
    TCNT2 = tcnt2;
    TIMSK2 |= (1<<TOIE2);
}

void MsTimer2::stop() {
    TIMSK2 &= ~(1<<TOIE2);
}

void MsTimer2::_overflow() {
    count += 1;
    if (count >= msecs && !overflowing) {
        overflowing = 1;
        count = 0;
        (*func)();
        overflowing = 0;
    }
}

ISR(TIMER2_OVF_vect) {
    TCNT2 = MsTimer2::tcnt2;
    MsTimer2::_overflow();
}
```

MsTimer2::set(unsigned long ms, void (*f)())
Ez a függvény beállítja a túlcsordulási intervallumot ms egységekben. Minden túlcsorduláskor az "f" függvény meghívásra kerül. A függvénynek nem lehet paramétere

MsTimer2::start() engedélyezi a megszakításokat

MsTimer2::stop() letiltja a megszakításokat

Szöveg görgetés MsTimer2 használatával

- A **led8x8_scrolltext2.ino** programban ugyanazt a szöveg-görgetést végezzük, amit az előző programban, de most a 8x8 LED mátrix képfrissítéstét az **MsTimer2** megszakításokkal ütemezzük
- Amint látni fogjuk, a program átírása mindössze néhány sor módosításával megoldható
- A fenti megoldás fő előnye az, hogy a kép előállítás és a megjelenítés külön válik, s a megjelenítés ütemezése nem függ a kép előállításának sebességétől
- A példaprogramban 5 ms-onként történik a képfrissítés, ami lehet, hogy túl halvány képet eredményez. Ebben ez esetben csökkentsük **Timer2** időzítésér 2 vagy 1 ms-ra

led8x8_scrolltext2.ino

```
#include <MsTimer2.h> // Ezen az oldalon csak ez a sor különbözik az előző programtól
#include "font6x8.h"

#define rON  0        // sor aktiválás (1088AS esetén katód:0)
#define rOFF 1        // sor deaktiválás
#define cON  1        // oszlop aktiválás (1088AS esetén anód:1)
#define cOFF 0        // oszlop deaktiválás

int row[] = {2,7,A5,5,13,A4,12,A2};  // 9,14,8,12,1,7,2,5
int col[] = {6,11,10,3,A3,4,8,9};    // 13,3,4,10,6,11,15,16
byte pixels[8];                      // Ebben a tömbben egy byte egy oszlopot ír le

//--- Megjelenítés (a kép felvillantása) -------
void refreshScreen() {
  for (int c = 0; c < 8; c++) {      // Sorra vesszük az oszlopokat
    digitalWrite(col[c], cON);       // Aktuális oszlop aktiválása
    byte thisByte = pixels[c];       // Pixel oszlop elővétele
    for (int r = 0; r < 8; r++) {    // Sorra vesszük a sorokat
      if (bitRead(thisByte,r)) {     // Ha a pixel értéke = 1
        digitalWrite(row[r],rON);    // Képpont felvillantása
      }
      digitalWrite(row[r],rOFF);
    }
    digitalWrite(col[c],cOFF);       // oszlop deaktiválása
  }
}
```

led8x8_scrolltext2.ino

```
//--- Mátrix görgetés balra --------
void scrollLeft(byte data) {
  for (byte i = 0; i < 7; i++) {      // Nem 8, hanem csak 7 !!!
    pixels[i] = pixels[i + 1];        // Minden pixeloszlop balra lép
  }
  pixels[7] = data;                   // Az utolsó oszlop az új adat
}

//--- Törli a pixel mátrixot -------
void clearMatrix() {
  for (byte i = 0; i < 8; i++) {
    pixels[i] = 0;
  }
}

//--- Egy karakter kiírása ---------
void writeChar(char ch) {
  ch -= 32;                           // szóköz (32) az első karakter a font6x8 tömbben
  for (int line = 0; line < 6; line++) {
    scrollLeft(font6x8[ch][line]);    // Egy pontoszlop beszúrása jobbról
    delay(250);                       // Várakozás: 250 ms
  }
}
```

Ezen az oldalon itt van csak különbség (a képfrissítést most nem itt hívjuk meg

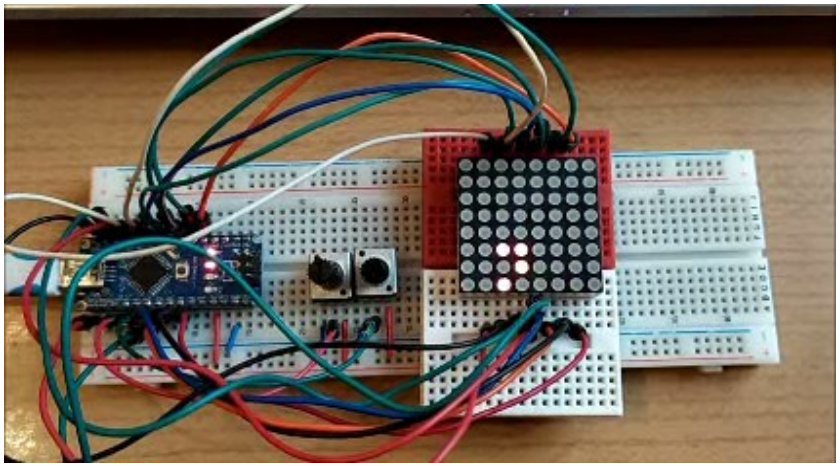
led8x8_scrolltext2.ino

```
//--- Szöveg kiírása karakterenként ----
void showText(char *s) {                  // s egy mutató, az adat memóriacímét tartalmazza
  while (*s) {                            // s: karakterre mutat, *s: a mutatott karakter
    writeChar(*s);                        // Az ütemezés a writeChar függvényen belül
    s++;                                  // történik, itt nem kell foglalkoznunk vele
  }
}

void setup() {
  for (int i = 0; i < 8; i++) {
    pinMode(row[i], OUTPUT);              // sor vezérlő kimenetek
    pinMode(col[i], OUTPUT);              // oszlop vezérlő kimenetek
  }
  clearMatrix();
 //--- Automatikus képfrissítés beállítása -----
  MsTimer2::set(5, refreshScreen);    // 5ms periódus
  MsTimer2::start();
}

void loop() {
  showText("MsTimer2 demo: ");
  showText("LED8x8 scrolltext2 example  ");
  clearMatrix();
}
```

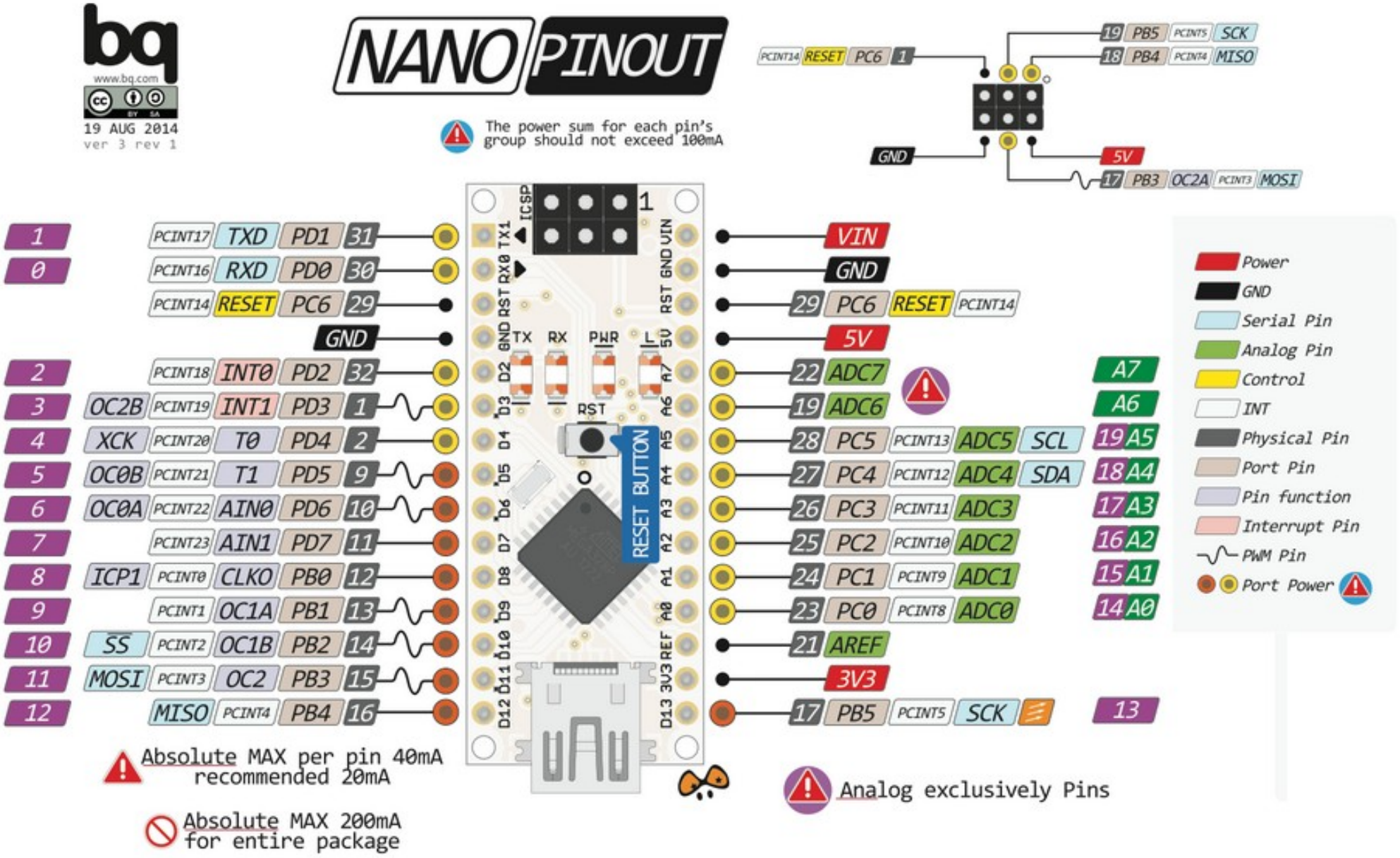
Itt van változás

Az 5 ms időzítés akár 1 ms-ra is csökkenthető, ha túl halvány a megjelenítés!

led8x8_scrolltext2.ino futási eredmény



Az Arduino nano kártya kivezetései



Ellenállás színkódok