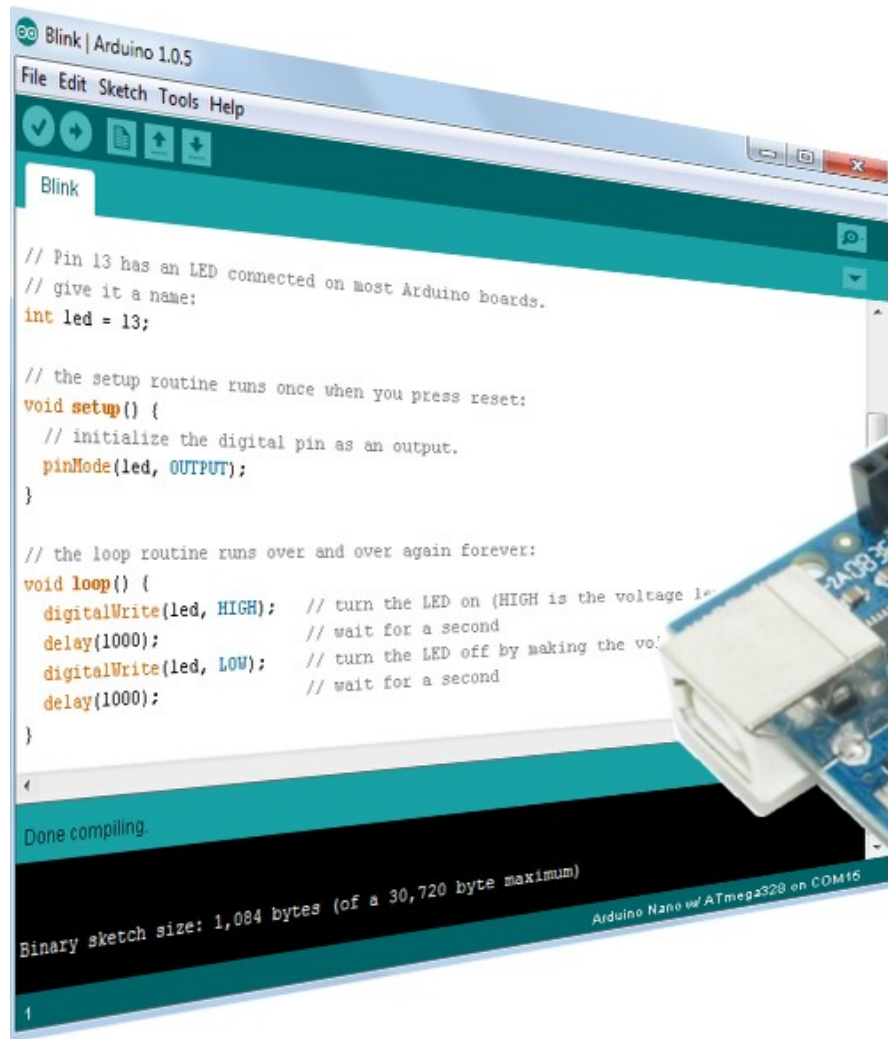


Bevezetés az elektronikába



```
Blink | Arduino 1.0.5
File Edit Sketch Tools Help
Blink
// Pin 13 has an LED connected on most Arduino boards.
// give it a name:
int led = 13;

// the setup routine runs once when you press reset:
void setup() {
  // initialize the digital pin as an output.
  pinMode(led, OUTPUT);
}

// the loop routine runs over and over again forever:
void loop() {
  digitalWrite(led, HIGH); // turn the LED on (HIGH is the voltage level)
  delay(1000);             // wait for a second
  digitalWrite(led, LOW);  // turn the LED off by making the voltage LOW
  delay(1000);             // wait for a second
}

Done compiling.
Binary sketch size: 1,084 bytes (of a 30,720 byte maximum)
Arduino Nano w/ ATmega328 on COM15
```

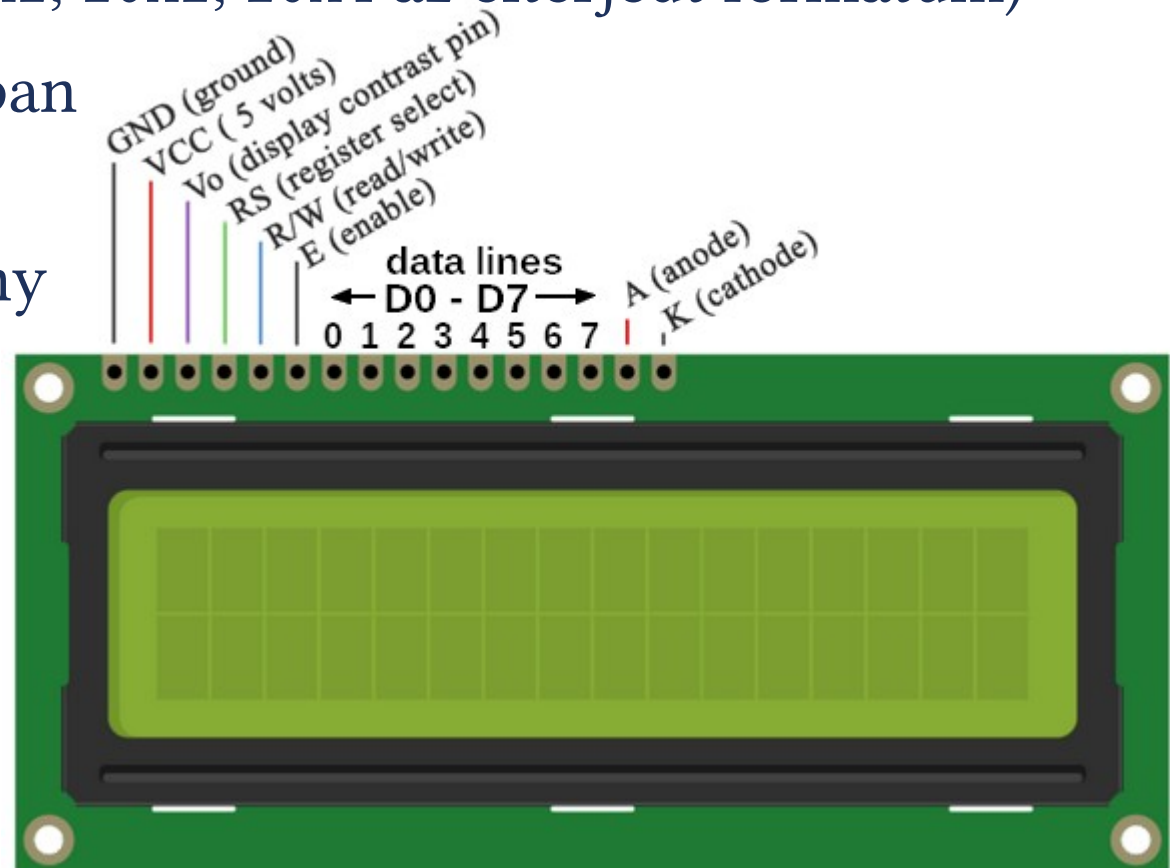


18. Arduino programozás

LCD kijelzők alkalmazása – II. rész

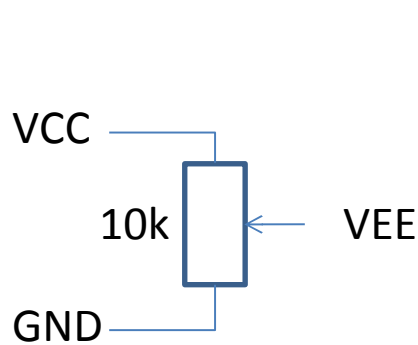
HD44780 kompatibilis kijelzők

- A HD44780 vagy kompatibilis vezérlővel ellátott alfanumerikus (csak rögzített karakterképet jeleníthetünk meg, pl. betű, szám, írásjel) kijelzők elterjedtségük miatt ipari szabványnak tekinthetők
- Tipikus a 16 kivezetéses, kétsoros, 2x16 karakteres modul, LED háttérvilágítással (8x2, 16x2, 20x2, 20x4 az elterjedt formátum)
- 8 és 4 adatvezetékes módban is használható
- R/W lehet mindig alacsony
- R/S adat/parancs váltó
0: adat 1: parancs küldés
- E – beíró jel (felfutó él) előtte a bemeneteket be kell állítani

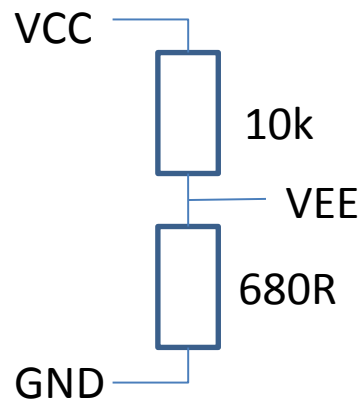


Kontraszt beállítása (variációk)

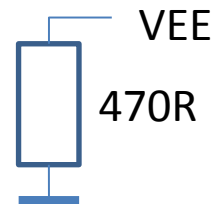
- Adatlap: 25 °C-on VEE számára VCC-4.5 V körüli érték az optimális
- A beállítást többféle módon is megoldhatjuk, legáltalánosabb módon egy 10 kΩ-os potméter segítségével. Sok esetben a GND-re kötés is működőképes megoldás



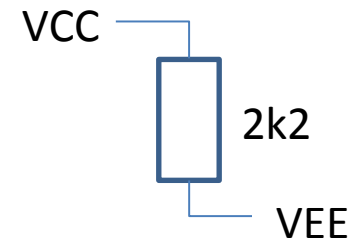
1. Az általánosan javasolt megoldás



2. Fixen beállított kontraszt



3. Egyszerűsített kontraszt beállítás (néhány 5 V-os kijelzőnél bevált Ez, vagy a direkt földre kötés is...)



4. Egyszerűsített kontraszt beállítás 3,3V-os kijelzőhöz

Bekötési vázlat

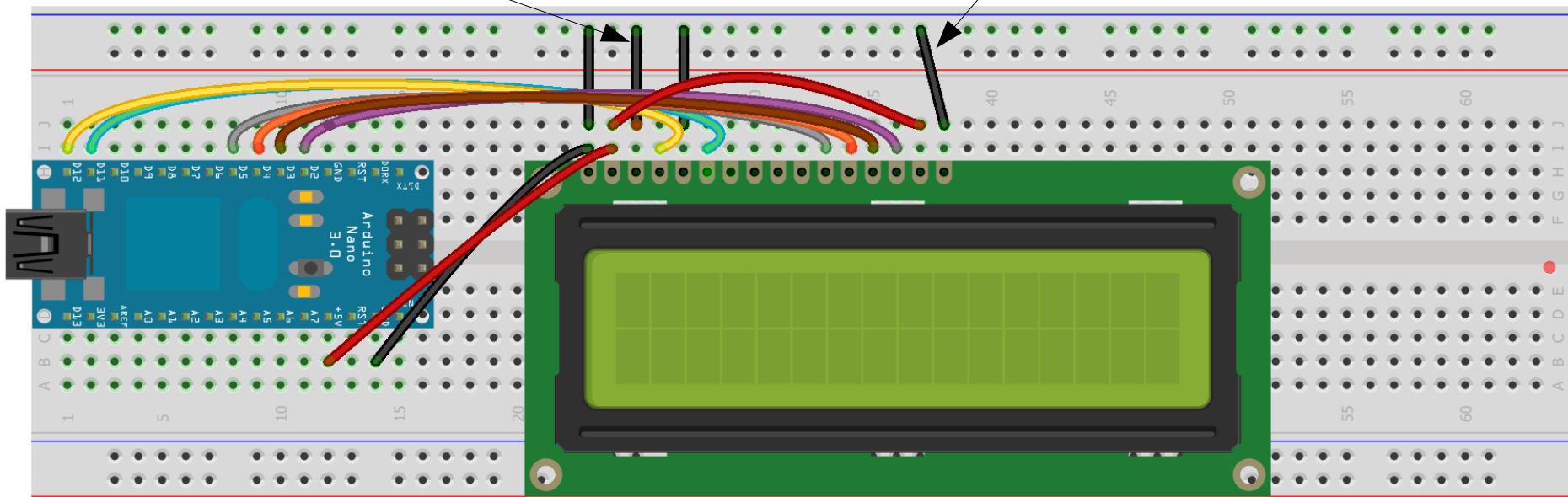
- Az LCD kijelzőt az Arduino beépített függvényei 4-bites módban használják (minden bájtot két részletben küldünk ki), ezért a **data0-data3** vonalakat nem kell bekötni!

1. GND – GND
2. VCC – +5V
3. VEE – GND
4. RS – D12
5. RW – GND
6. EN – D11

11. data4 – D5
12. data5 – D4
13. data6 – D3
14. data7 – D2
15. LED+ – +5V
16. LED- – GND

Kontrasztbeállítást lásd az előző oldalon!

Először kb. 100 Ω ellenálláson keresztül próbáljuk ki!



Made with  Fritzing.org

LiquidCrystal programkönyvtár

- Az LCD kijelzőt Arduino környezetben a **LiquidCrystal** beépített programkönyvtár segítségével kezelhetjük legegyszerűbben
- **A legfontosabb metódusok:**
 - ❖ **lcd.begin**(oszlop,sor) – a képernyő inicializálása és konfigurálása
 - ❖ **lcd.setCursor**(oszlop, sor) – pozíció beállítása a megadott helyre
 - ❖ **lcd.write**(karakterkód) – egy karakter kiírása
 - ❖ **lcd.print**("szöveg") – szöveg kiírása
 - ❖ **lcd.print**(kifejezés) – számérték kiírása
 - ❖ **lcd.createChar**(sorsz, adatok) – új karakter definiálása
ahol sorsz = 0 – 7, az adatok pedig egy 8 bájtos tömb, de csak 5 bit számít
- Bővebb információ: Arduino IDE Help/Referencia menüpontjában

Új karakterek definiálása

- Az LCD modulok beépített karakterkészlete nem alkalmas a magyar nyelvű szövegek ékezhelyes megjelenítésére
- A 128 fölötti kódok zöméhez általában japán vagy cirill betűk tartoznak, ahogy az alábbi ábrán is láthatjuk
- A 0 – 7 közötti kódú karaktereket azonban mi definiálhatjuk (5x8 pont)
- 8 – 15 közötti kódra ugyan-ezek a jelek jönnek elő!

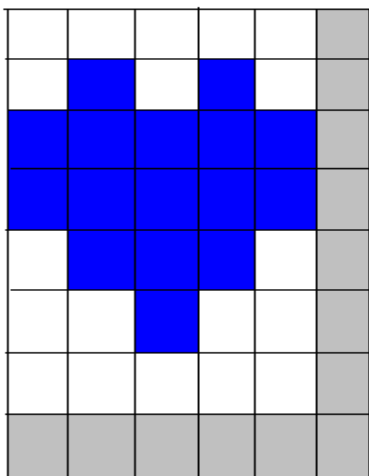
a 4 felső címbit

	0000	0001	0010	0011	0100	0101	0110	0111	1000	1001	1010	1011	1100	1101	1110	1111
xxxx0000	CG RAM (1)			0	@	P	`	P				-	9	3	ε	p
xxxx0001	(2)		!	1	A	Q	a	q			。	ア	チ	△	ä	g
xxxx0010	(3)		"	2	B	R	b	r			「	イ	ツ	×	ß	θ
xxxx0011	(4)		#	3	C	S	c	s			」	ウ	テ	モ	ε	ω
xxxx0100	(5)		\$	4	D	T	d	t			、	エ	ト	ヤ	μ	Ω
xxxx0101	(6)		%	5	E	U	e	u			・	オ	ナ	ユ	œ	Ü
xxxx0110	(7)		&	6	F	V	f	v			ヲ	カ	ニ	ヨ	ρ	Σ
xxxx0111	(8)		'	7	G	W	g	w			ア	キ	ヌ	ラ	g	π
xxxx1000	(1)		<	8	H	X	h	x			イ	ク	ネ	リ	γ	×
xxxx1001	(2)		>	9	I	Y	i	y			ウ	ケ	ル		ˆ	γ
xxxx1010	(3)		*	:	J	Z	j	z			エ	コ	ン	レ	j	キ
xxxx1011	(4)		+	;	K	C	k	c			オ	サ	ヒ	ロ	*	π
xxxx1100	(5)		,	<	L	¥	l	l			カ	シ	フ	ワ	φ	π
xxxx1101	(6)		-	=	M	J	m	j			ユ	ズ	ヘ	ン	ε	÷
xxxx1110	(7)		.	>	N	^	n	→			ヨ	セ	ホ	°	ñ	
xxxx1111	(8)		/	?	O	_	o	€			ッ	ソ	マ	°	ö	■

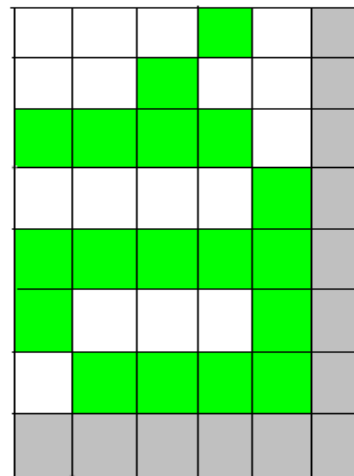
a 4 alsó címbit

Új karakterek definiálása

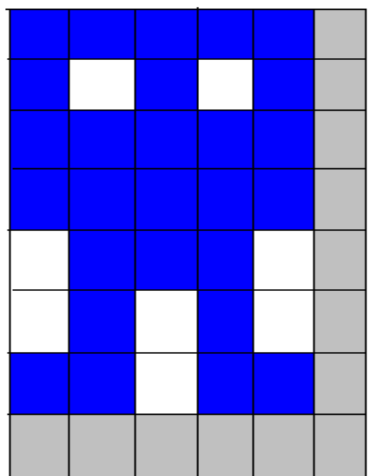
- A karakterek általában 5x7-es pontmátrixban helyezkednek el, a 8. sor elválasztó vonal. Különleges esetben (pl. teljes kitöltésű téglalap) a 8. sort is felhasználjuk



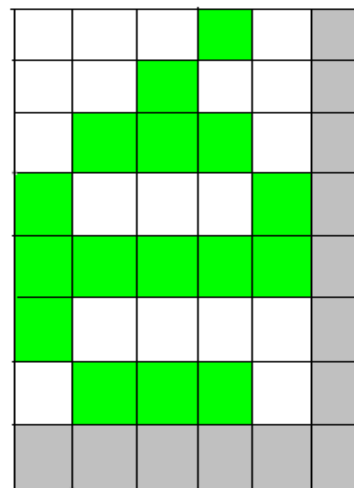
```
byte Heart[8] = {  
    0b00000,  
    0b01010,  
    0b11111,  
    0b11111,  
    0b01110,  
    0b00100,  
    0b00000,  
    0b00000 };
```



```
byte a1[8] = {  
    0b00010,  
    0b00100,  
    0b11110,  
    0b00001,  
    0b11111,  
    0b10001,  
    0b01111,  
    0b00000 };
```



```
byte Alien[8] = {  
    0b11111,  
    0b10101,  
    0b11111,  
    0b11111,  
    0b01110,  
    0b01010,  
    0b11011,  
    0b00000 };
```



```
byte e1[8] = {  
    0b00010,  
    0b00100,  
    0b01110,  
    0b10001,  
    0b11111,  
    0b10000,  
    0b01110,  
    0b00000 };
```

lcd_custom_chars.ino

Forrás: <https://lastminuteengineers.com/arduino-1602-character-lcd-tutorial/>

```
#include <LiquidCrystal.h>
LiquidCrystal lcd(12, 11, 5, 4, 3, 2);
```



```
byte Heart[8] = {
  0b00000,
  0b01010,
  0b11111,
  0b11111,
  0b01110,
  0b00100,
  0b00000,
  0b00000 };

byte Alien[8] = {
  0b11111,
  0b10101,
  0b11111,
  0b11111,
  0b01110,
  0b01010,
  0b11011,
  0b00000 };

byte Speaker[8] = {
  0b00001,
  0b00011,
  0b01111,
  0b01111,
  0b01111,
  0b00011,
  0b00001,
  0b00000 };

byte Skull[8] = {
  0b00000,
  0b01110,
  0b10101,
  0b11011,
  0b01110,
  0b01110,
  0b00000,
  0b00000 };

byte Bell[8] = {
  0b00100,
  0b01110,
  0b01110,
  0b01110,
  0b11111,
  0b00000,
  0b00100,
  0b00000 };

byte Check[8] = {
  0b00000,
  0b00001,
  0b00011,
  0b10110,
  0b11100,
  0b01000,
  0b00000,
  0b00000 };

byte Sound[8] = {
  0b00001,
  0b00011,
  0b00101,
  0b01001,
  0b01001,
  0b01011,
  0b11011,
  0b11000 };

byte Lock[8] = {
  0b01110,
  0b10001,
  0b10001,
  0b11111,
  0b11011,
  0b11011,
  0b11111,
  0b00000 };
```


lcd_custom_chars.ino

```
void setup() {  
  lcd.begin(16, 2);           // Inicializáljuk a kijelzőt  
  lcd.createChar(0, Heart);   // Grafikus karakterek feltöltése  
  lcd.createChar(1, Bell);  
  lcd.createChar(2, Alien);  
  lcd.createChar(3, Check);  
  lcd.createChar(4, Speaker);  
  lcd.createChar(5, Sound);  
  lcd.createChar(6, Skull);  
  lcd.createChar(7, Lock);  
  lcd.print("Custom Character");  
}
```

Az első sorba az `lcd.print` utasítás ír

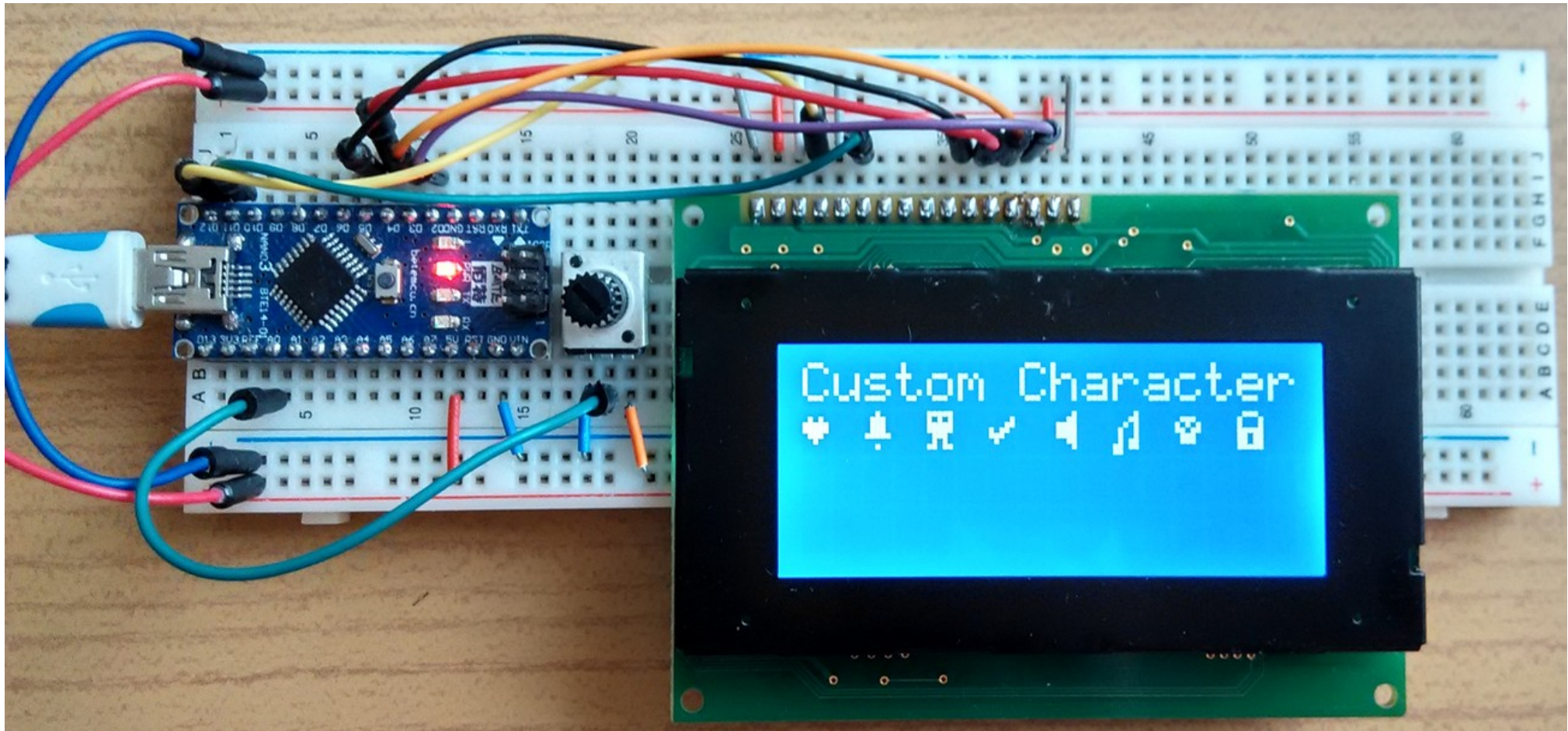
A második sort a `loop` függvény tölti fel
(bár az ismétlés szükségtelen....)



```
void loop() {  
  lcd.setCursor(0, 1); lcd.write(byte(0)); // Heart jel kiírása  
  lcd.setCursor(2, 1); lcd.write(byte(1)); // Bell jel kiírása  
  lcd.setCursor(4, 1); lcd.write(byte(2)); // Alien jel kiírása  
  lcd.setCursor(6, 1); lcd.write(byte(3)); // Check jel kiírása  
  lcd.setCursor(8, 1); lcd.write(byte(4)); // Speaker jel kiírása  
  lcd.setCursor(10, 1); lcd.write(byte(5)); // Sound jel kiírása  
  lcd.setCursor(12, 1); lcd.write(byte(6)); // Skull jel kiírása  
  lcd.setCursor(14, 1); lcd.write(byte(7)); // Lock jel kiírása  
}
```

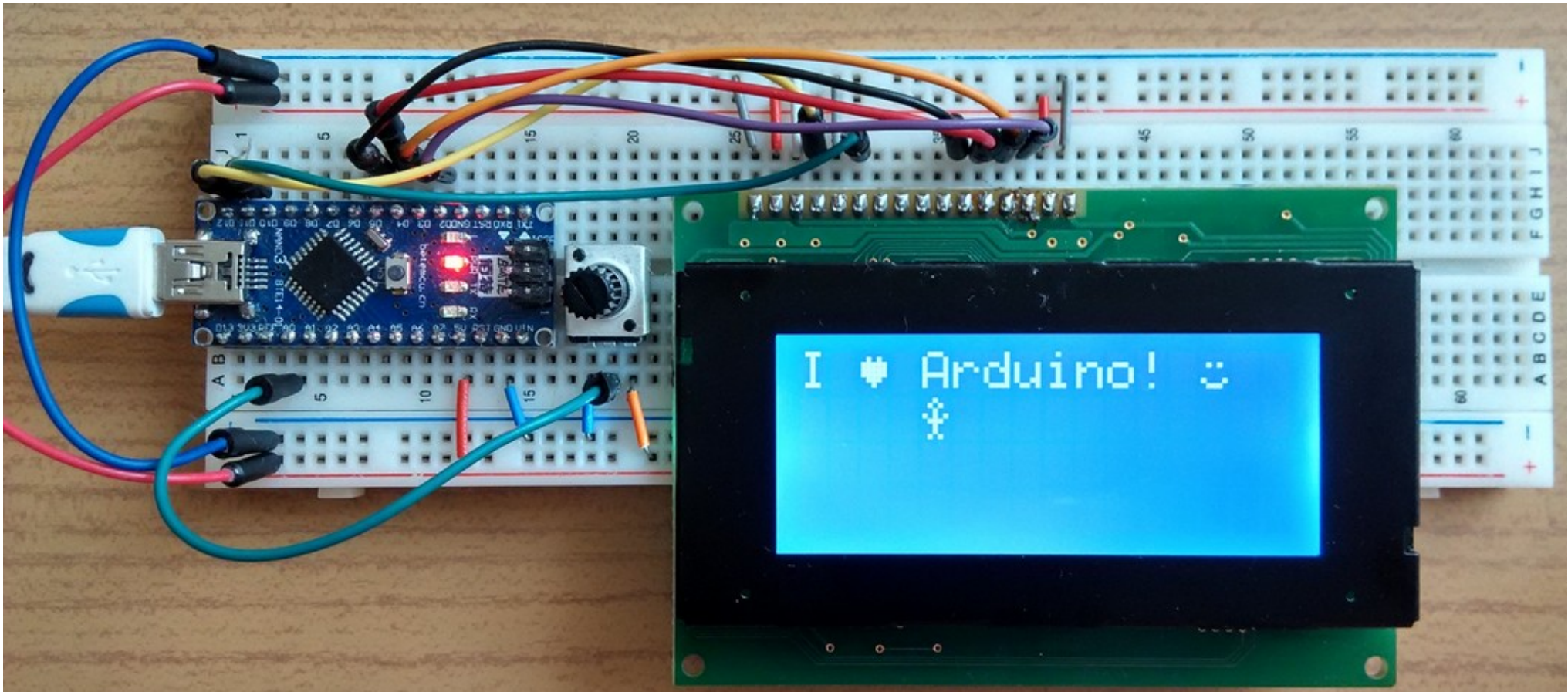
Lcd_custom_chars.ino

- A program futási eredménye az alábbi fényképen látható



CustomCharacter.ino

- Ez a program az Arduino IDE gyári mintapéldái közé tartozik
- Betöltés: **File/Examples/LiquidCrystal/CustomCharacter** menüpontra kattintsunk!
- Kapcsolási elrendezés: megegyezik a múlt órai feszültségmérővel



CustomCharacter.ino

```
#include <LiquidCrystal.h>
```

```
LiquidCrystal lcd(12, 11, 5, 4, 3, 2);
```

```
byte heart[8] = {
```

```
  0b00000,
```

```
  0b01010,
```

```
  0b11111,
```

```
  0b11111,
```

```
  0b11111,
```

```
  0b01110,
```

```
  0b00100,
```

```
  0b00000 };
```

```
byte smiley[8] = {
```

```
  0b00000,
```

```
  0b00000,
```

```
  0b01010,
```

```
  0b00000,
```

```
  0b00000,
```

```
  0b10001,
```

```
  0b01110,
```

```
  0b00000 };
```

```
byte frownie[8] = {
```

```
  0b00000,
```

```
  0b00000,
```

```
  0b01010,
```

```
  0b00000,
```

```
  0b00000,
```

```
  0b00000,
```

```
  0b01110,
```

```
  0b10001
```

```
};
```

```
byte armsDown[8] = {
```

```
  0b00100,
```

```
  0b01010,
```

```
  0b00100,
```

```
  0b00100,
```

```
  0b01110,
```

```
  0b10101,
```

```
  0b00100,
```

```
  0b01010 };
```

```
byte armsUp[8] = {
```

```
  0b00100,
```

```
  0b01010,
```

```
  0b00100,
```

```
  0b10101,
```

```
  0b01110,
```

```
  0b00100,
```

```
  0b00100,
```

```
  0b01010 };
```

CustomCharacter.ino

```
void setup() {
  lcd.begin(16, 2);           // Inicializáljuk a kijelzőt
  lcd.createChar(0, heart);
  lcd.createChar(1, smiley);
  lcd.createChar(2, frownie);
  lcd.createChar(3, armsDown);
  lcd.createChar(4, armsUp);
  lcd.setCursor(0, 0);
  lcd.print("I ");           // Szöveg kiírása az LCD-re
  lcd.write(byte(0));        // Szív
  lcd.print(" Arduino! ");
  lcd.write(byte(1));        // Szmájli
}

void loop() { //A potméter állásától függő sebességgel integet az emberke
  int sensorReading = analogRead(A0); //Potméter leolvasása
  int delayTime = map(sensorReading,0,1023,200,1000); //késleltetési idő
  lcd.setCursor(4, 1);       // Alsó sor 5. pozíció
  lcd.write(3);              // Emberke leengedett karral
  delay(delayTime);
  lcd.setCursor(4, 1);       // Alsó sor 5. pozíció
  lcd.write(4);              // Emberke felemelt karral
  delay(delayTime);
}
```

lcd_betvek.ino

- Ha magyarul akarunk írni, akkor be kell érünk az ékezetes kisbetűkkel (az ö betűnek így sem marad hely...)
- Az alábbi programot a „60 nap alatt Arduino tanfolyam” mintapéldájából vettük, és dolgoztuk át (<http://avr.tavir.hu>)

```
#include "LiquidCrystal.h"           //LCD kijelzőt használunk
LiquidCrystal lcd(12, 11, 5, 4, 3, 2); //bekötjük az LCD-t

//kis ékezetes betűk
byte a1[8] = {B10, B100, B1110, B1, B1111, B10001, B1111}; //á
byte e1[8] = {B10, B100, B1110, B10001, B11111, B10000, B1110}; //é
byte i1[8] = {B10, B100, B0, B1110, B100, B100, B1110}; //í
byte o1[8] = {B100, B100, B0, B1110, B10001, B10001, B1110}; //ó
byte o2[8] = {B1010, B0, B1110, B10001, B10001, B10001, B1110}; //ö
byte o3[8] = {B1010, B1010, B0000, B1110, B10001, B10001, B1110}; //ő
byte u1[8] = {B0010, B0100, B10001, B10001, B10001, B10011, B1101}; //ú
byte u2[8] = {B1010, B0, B0, B10001, B10001, B10011, B1101}; //ü
byte u3[8] = {B1010, B1010, B0, B10001, B10001, B10011, B1101}; //ű
```

Folytatás a következő oldalon ...

lcd_betwek.ino

```
byte b; //átmeneti tároló

void setup() {
  //Ékezetes karakterek definiálása (ö-nek nincs hely!
  lcd.createChar(0, a1); //á
  lcd.createChar(1, e1); //é
  lcd.createChar(2, i1); //í
  lcd.createChar(3, o1); //ó
  lcd.createChar(4, o3); //ö
  lcd.createChar(5, u1); //ú
  lcd.createChar(6, u2); //ü
  lcd.createChar(7, u3); //ű
  lcd.begin(16, 4);
  lcd.print("\010rv\002zt\007r\004");
  lcd.setCursor(0,1);
  lcd.print("\t\006k\357rf\005r\003g\001p"); //tükörfúrógép
  lcd.setCursor(0,2);
  Serial.begin(9600); //Soros port 9600 bps nyitása
}
```

2x16 kijelző esetén:
lcd.begin(16, 2);

2x16 kijelző esetén ezt a két sort el kell hagyni!

//4x16 karakteres LCD
//árvíztűrő

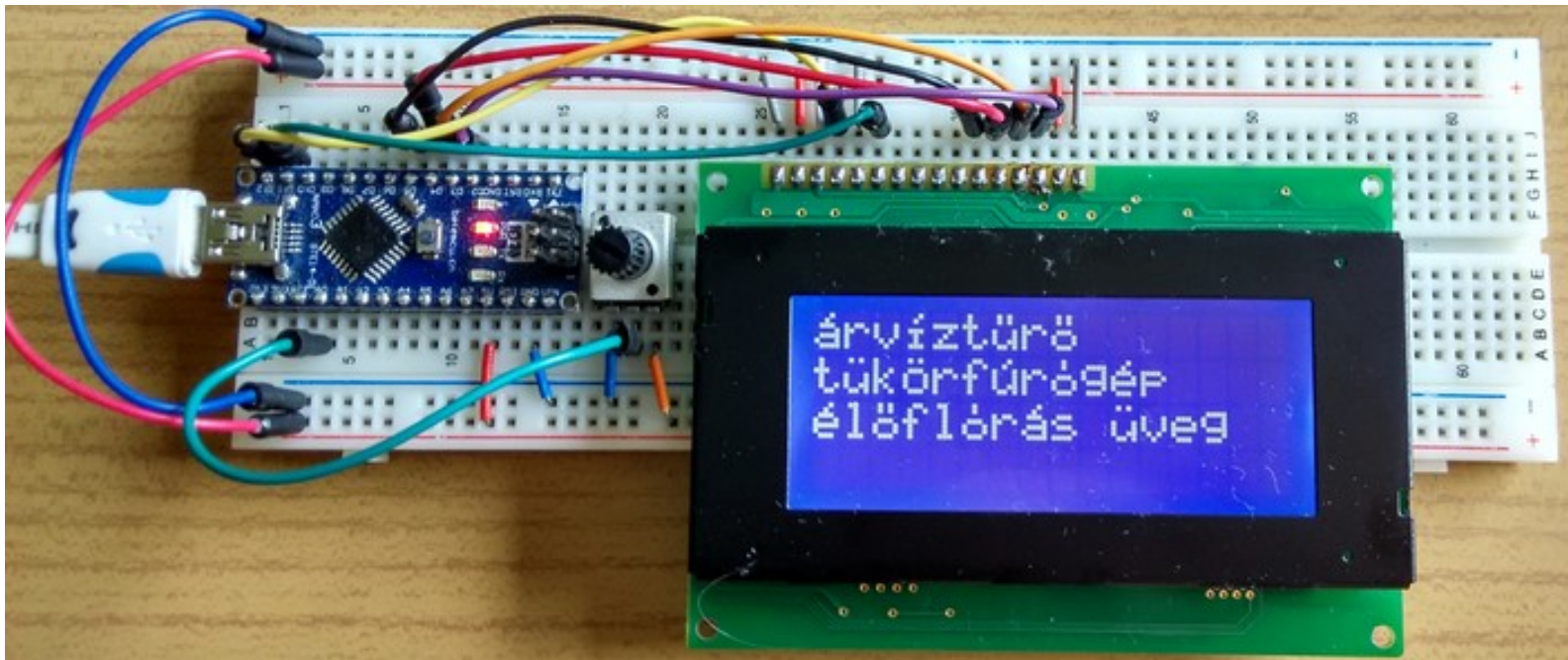
Folytatás a következő oldalon ...

Lcd_betwek.ino

```
void loop() {
  if (Serial.available()) {
    b = Serial.read();           //ha van karakter: beolvassuk
    Serial.print(b, DEC);       //kiírjuk a Windows karakterkódját
    Serial.print(": ");
    switch (b) {                //Helyettesítés karakterkód alapján
      case 225: b = 0; break;   //á
      case 233: b = 1; break;   //é
      case 237: b = 2; break;   //í
      case 243: b = 3; break;   //ó
      case 245: b = 4; break;   //ö
      case 246: b = 0xEF; break; //ö betű a ROM karaktergenerátorból
      case 250: b = 5; break;   //ú
      case 252: b = 6; break;   //ü
      case 251: b = 7; break;   //ű
    }
    Serial.println(b, DEC);     //kiírjuk az LCD karakterkódját
    lcd.write(b);              //írjuk ki az LCD-re
  }
}
```


Lcd_betwek.ino

- A program futtatásához nyissunk egy terminálablakot!
- A *setup()* függvényben megadott szöveg megjelenítése után karaktereket küldhetünk a programnak, s a kisbetűs szöveg ékezet helyesen jelenik meg a kijelzőn



Az Arduino nano kártya kivezetései



NANO PINOUT

The power sum for each pin's group should not exceed 100mA

