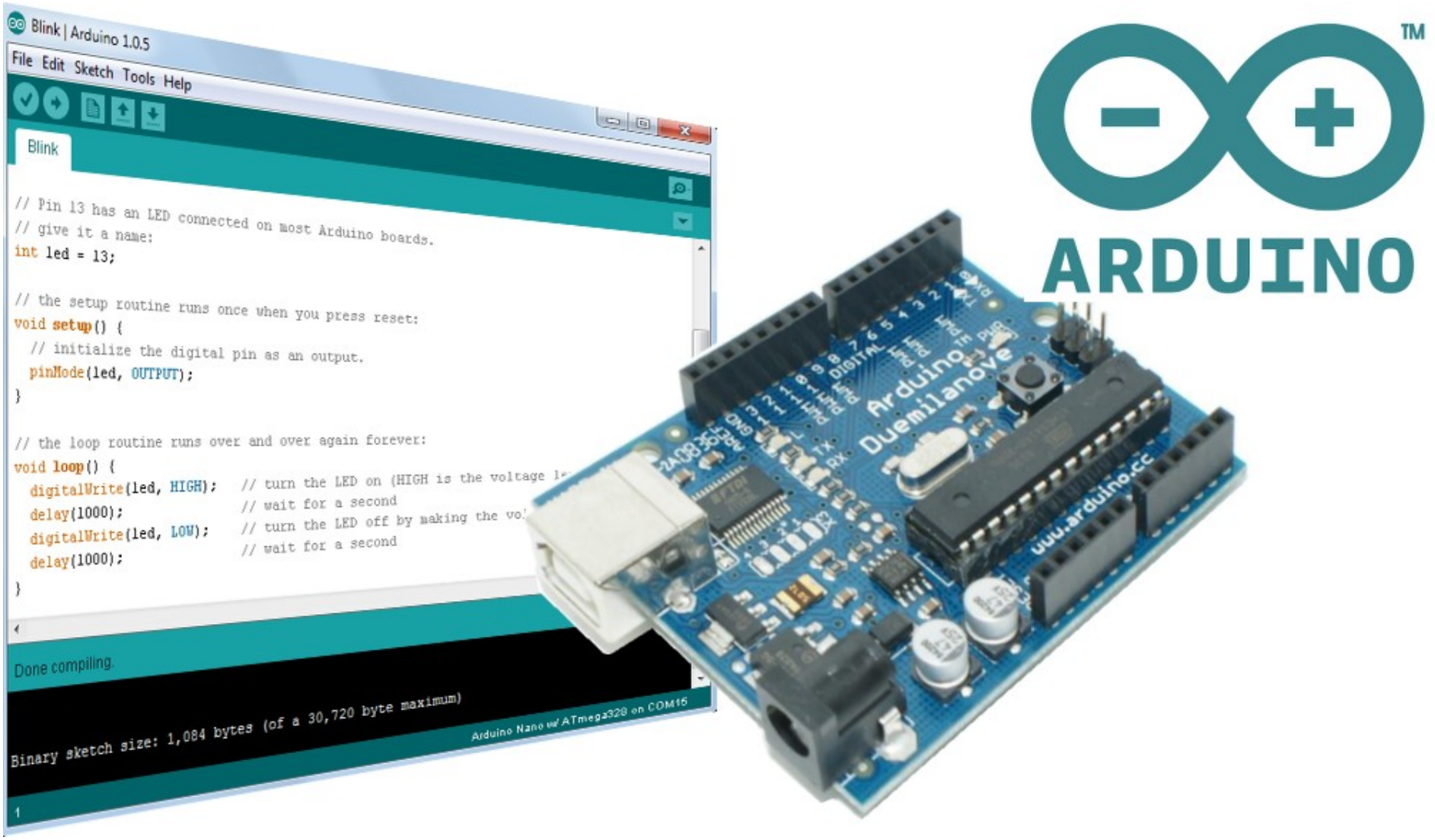
Arduino tanfolyam kezdőknek és haladóknak



11. Ultrahangos távolságmérés, OLED kijelzők

Ajánlott irodalom

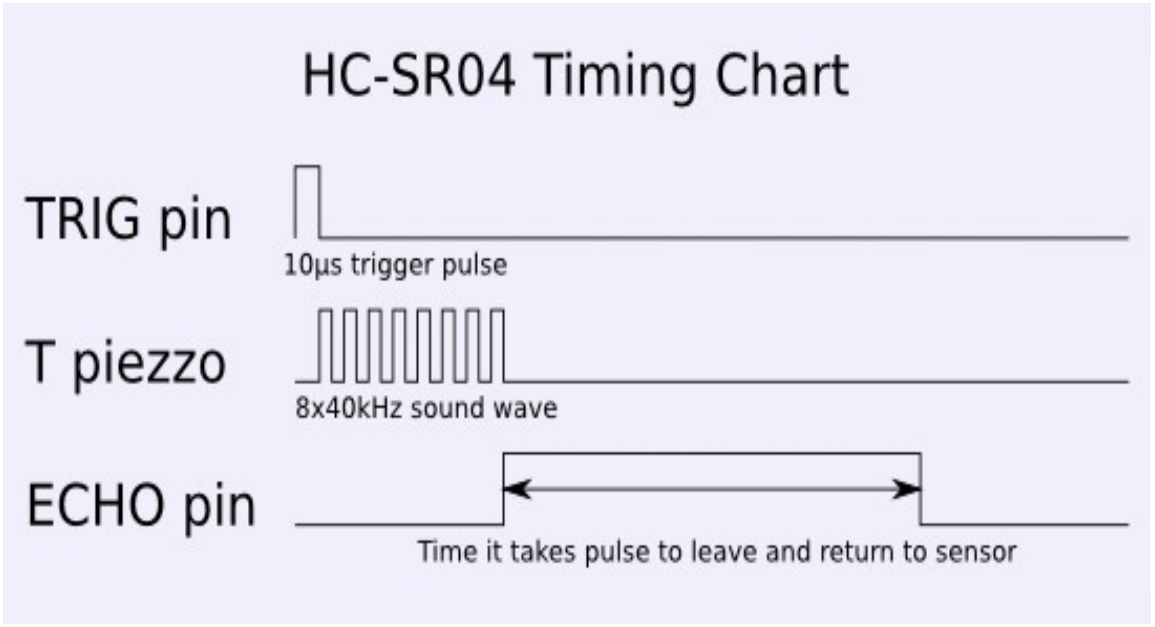
- Aduino LLC.: Arduino Language Reference
- ATMEL: ATmega328p mikrovezérlő adatlapja
- Brian W. Kernighan, Dennis Ritchie: A C programozási nyelv
- Cseh Róbert: Arduino programozási kézikönyv
- Harsányi Réka – Juhász Márton András:
  Fizikai számítástechnika: elektronikai alapok és Arduino programozás
- Ruzsinszki Gábor: Mikrovezérlős rendszerfejlesztés C/C++ nyelven I. – PIC mikrovezérlők
- Ruzsinszki Gábor: Mikrovezérlős rendszerfejlesztés C/C++ nyelven II. – Arduino

Arduino19_11 projektek

- **Sonar** – Ultrahangos távolságmérés
- **oled_text_demo** – egyszerű szövegkiírás OLED kijelzőn
- **oled_setfont** – szöveg kiírása többféle font használatával
- **oled_basic_drawing** – a rajzelemek használata

Ultrahangos távolságmérő

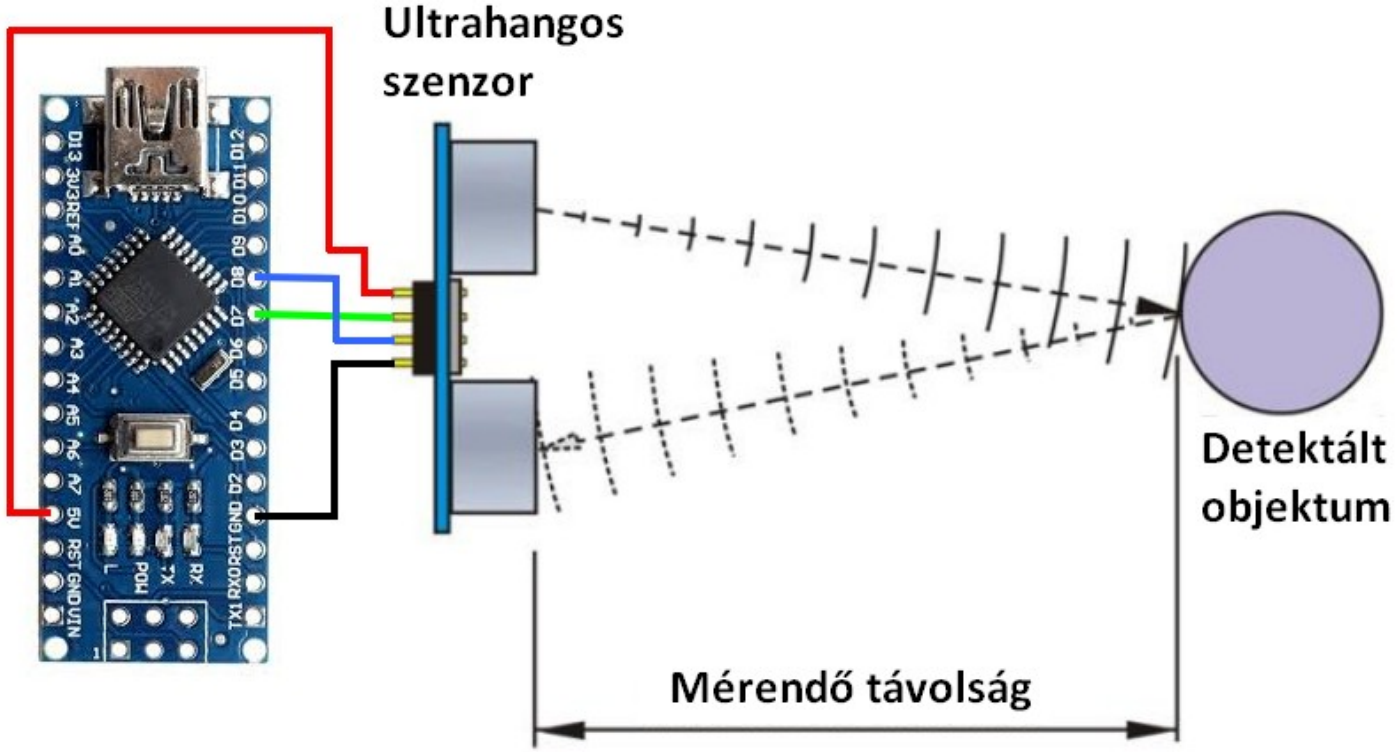
A **HC-SR04** modul piezo jeladója az indító impulzus hatására egy 40 kHz-es jelcsomagot sugároz ki. A modul digitális kimenő impulzusának szélessége megegyezik a visszaverődött hang terjedési idejével. Valójában ez tehát részben digitális, részben analóg szenzornak tekinthető…



Főbb paraméterek

- Tápfeszültség: 4.5 V – 5.5 V
- Mérési tartomány: 2 cm – 4 m (gyakorlatban inkább 2 m)
- Érzékelési szögtartomány: ~ 16 º

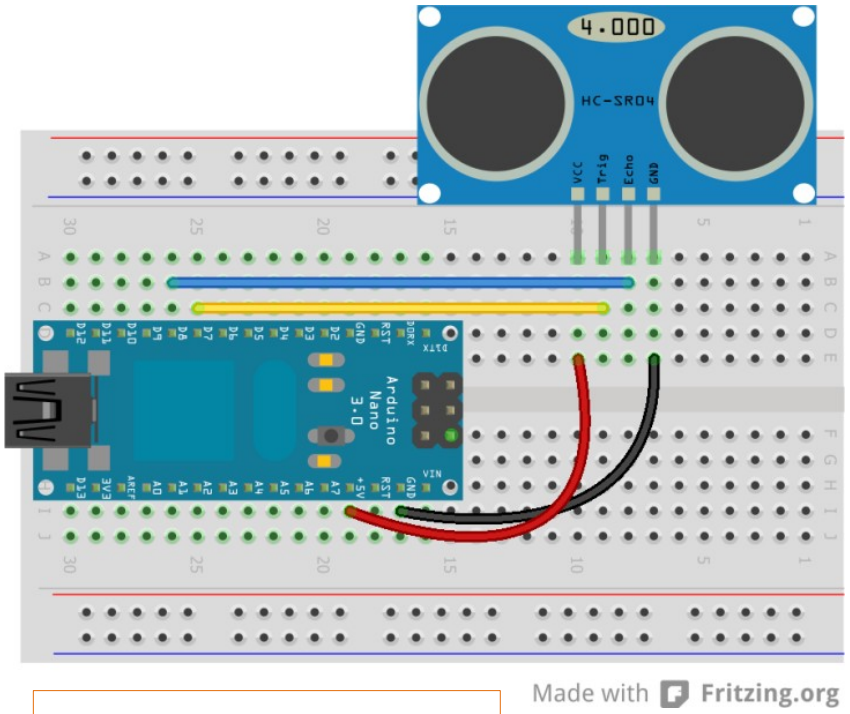
Az ultrahangos távolságmérés elve



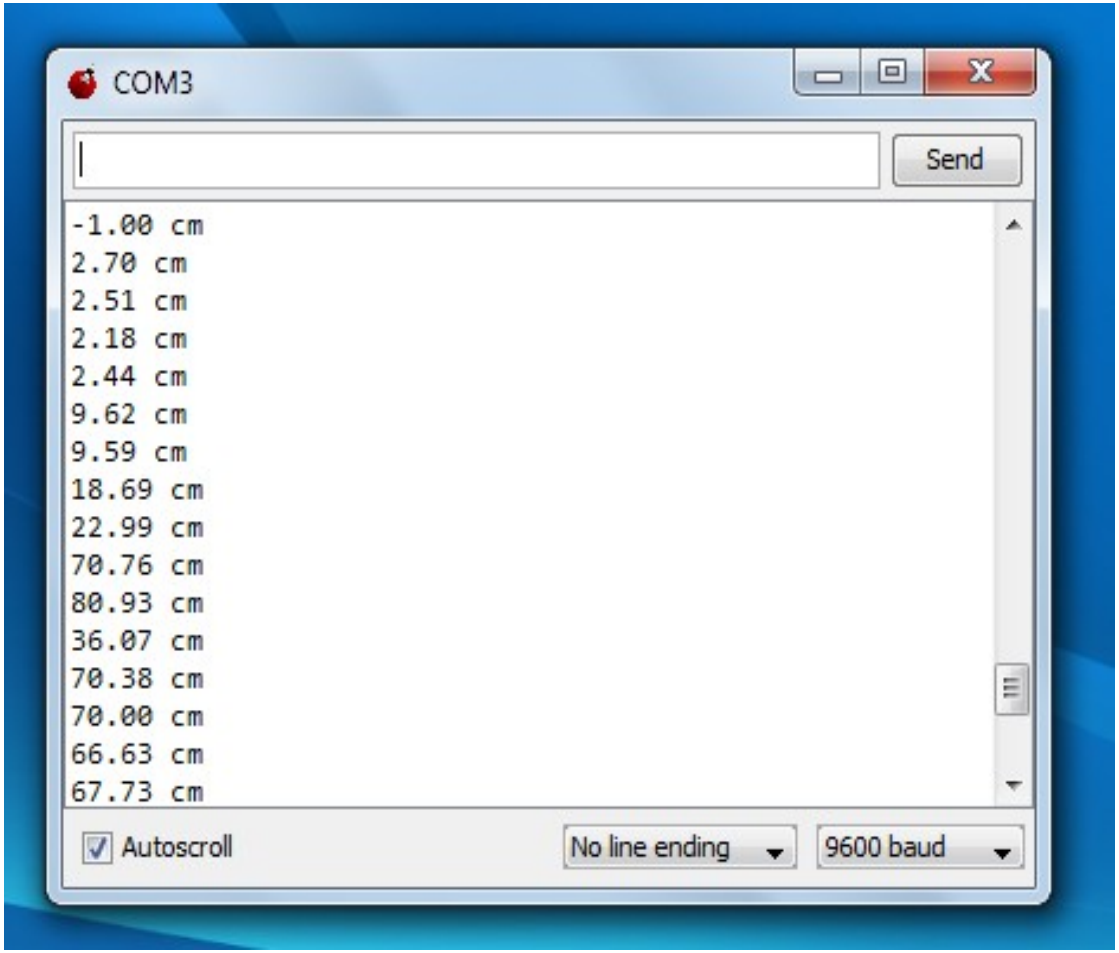
- Ha az ultrahang impulzus útjában egy tárgy található, akkor a hullámok egy része visszaverődik. Ez a visszhangot a vevő észlelheti
- Az adás és a vétel között eltelt t időből és a hang v terjedési sebességéből kiszámítható a szenzor és a tárgy között d távolság:

$$d = t \cdot v / 2$$ ahol v esetünkben kb. 340 m/s

Ultrahangos távolságmérő



Lábkiosztás
Trigger: D7
Echo: D8



unsigned long **pulseInt**(int *pin,* int *value*)

Meghatározza a beérkező impulzus szélességét (mikroszekundum egységekben).
pin – a vizsgált bemenet sorszáma
value – az vizsgálandó impulzus polaritása (HIGH vagy LOW)

Sonar.ino – 1/2. oldal

```
#define echoPin 8                 // Echo bemenet
#define trigPin 7                 // Trigger kimenet
#define RED_LED 13
```

Hardverfüggő rész

```
int maximumRange = 400;           // Legnagyobb távolság cm-ben
int minimumRange = 1;             // Minimális távolság cm-ben
long duration;                    // Időtartam [us]
float distance;                   // Távolság [cm]void

void setup() {
  Serial.begin(9600);             //Soros kapcsolat 9600 bit/s
  Serial.println("Sonar program");
  pinMode(echoPin, INPUT);        //Impulzus bemenet
  pinMode(trigPin, OUTPUT);       //Vezérlő kimenet
  digitalWrite(trigPin, LOW);     //Alaphelyzetben alacsony szint
  pinMode(RED_LED,OUTPUT);        //A beépített piros LED jelző funkciót lát el
  digitalWrite(RED_LED,HIGH);
}
```

Sonar.ino – 2/2. oldal

```
void loop() {
  delay(1000);                          //egy kis várakozás
  digitalWrite(RED_LED,HIGH);
  digitalWrite(trigPin, HIGH);
  delayMicroseconds(10);
  digitalWrite(trigPin, LOW);
  duration = pulseIn(echoPin, HIGH); //Impulzus szélességének meghatározása
  digitalWrite(RED_LED,LOW);
  distance = duration/58.82;    // Kiszámoljuk a távolságot
  if (distance >= maximumRange || distance < minimumRange) {
    distance = -1.0;
  }
  Serial.print(distance);       // A mért távolság kiíratása
  Serial.println(" cm");
}
```

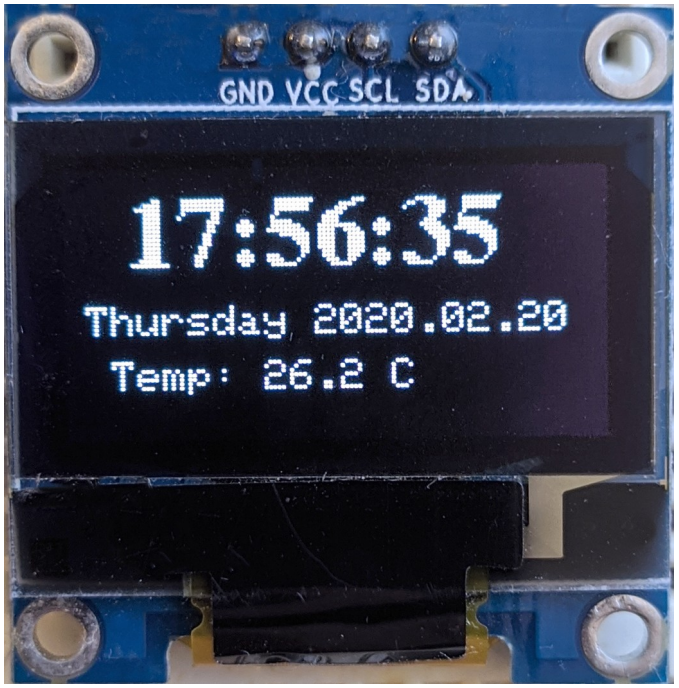
Trigger impulzus előállítása

$$d = \frac{t * v}{2}$$

Ahol *d* a távolság, *t* az impulzus hossza, *v* a hang terjedési sebessége (~340 m/s). Mivel *t* értéke µs-ban adott, d-t pedig cm-ben mérjük, így d = t * 34000/2000000, azaz **d = t/58.82**

OLED I2C kijelző SSD1306 vezérlővel

Jellemzők:



- OLED technológia
- 128x64 képpont
- 0,96” (2,4 cm) képátló
- I2C illesztő
- 2.5V-5.5V tápfeszültség
- SSD1306 vezérlő
- Monokróm (egyes változatoknál a felső harmad más színű)
- Grafikus megjelenítés
- Inverz mód
- Görgetés több irányba
- Dokumentáció: Solomon Systech SSD1306 adatlap

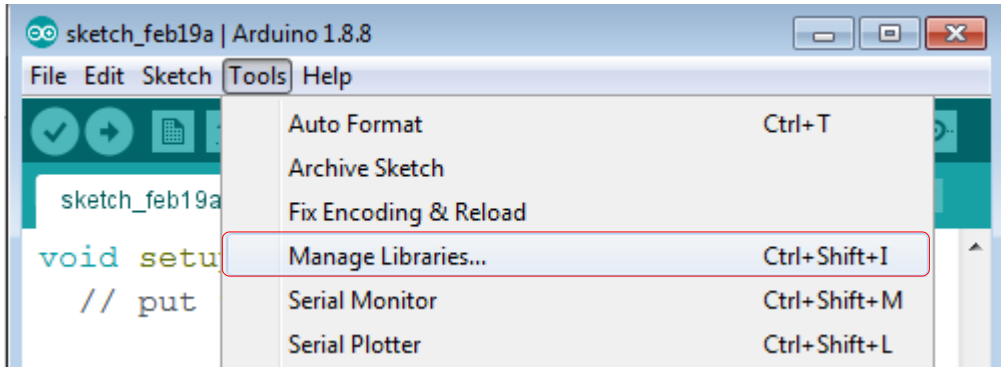
SSD1306 programkönyvtárak

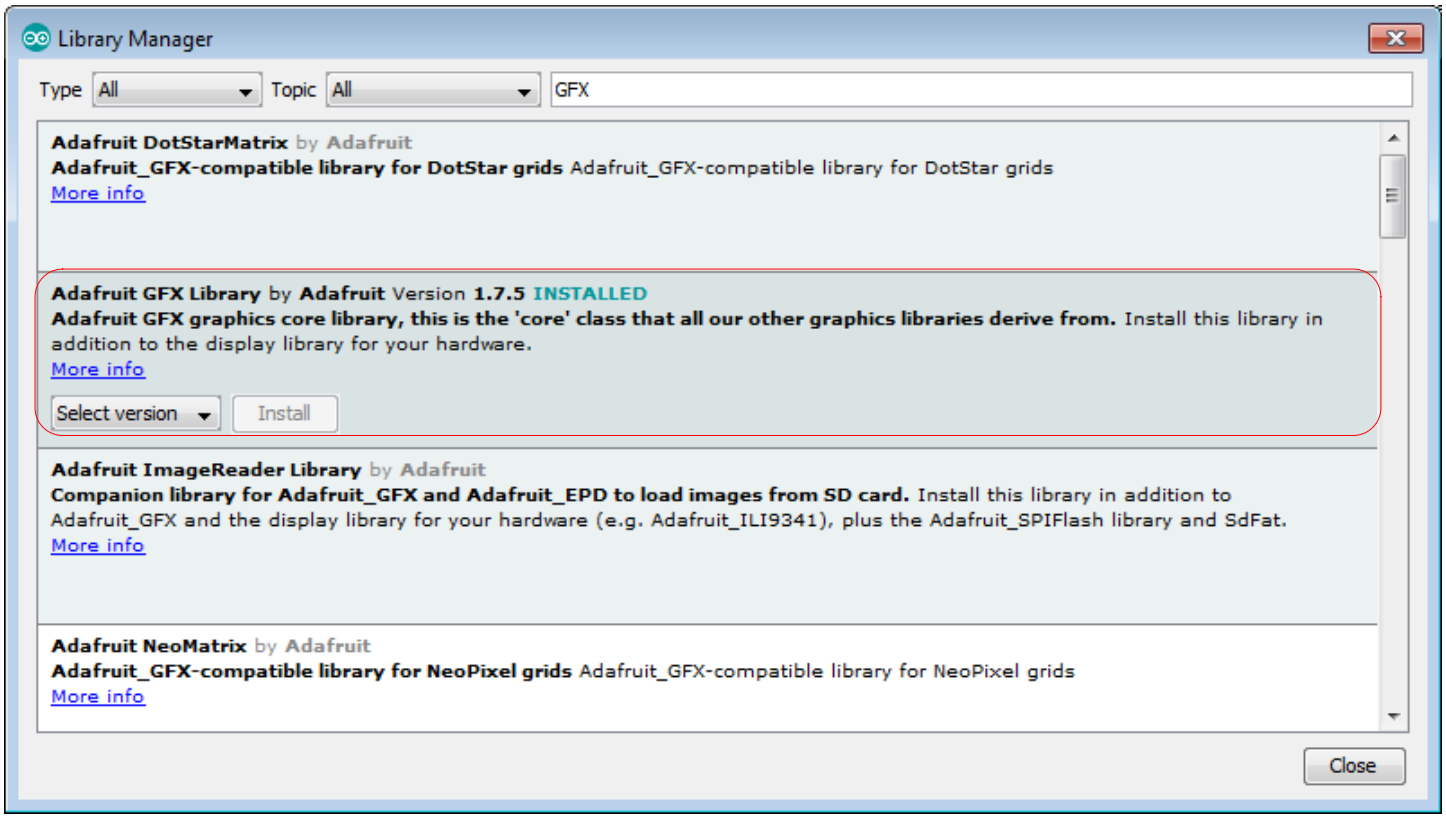
- A kijelző elvileg saját programmal is vezérelhető az adatlap alapján, de kényelmesebb a meglevő programkönyvtárak közül választani
- **Adafruit_SSD1306:** C++ nyelven írt osztály, amely az **Adafruit_GFX** és a **Print** osztályok leszármazottja, azok metódusait is örökli. Az **Adafruit_GFX** könyvtárat is telepíteni kell!
  forrás: github.com/adafruit/Adafruit_SSD1306
  és github.com/adafruit/Adafruit-GFX-Library
- **U8g2:** a korábbi **U8g** programkönyvtár továbbfejlesztett kiadása. Sokféle kijelzőhöz használható (pl. **SH1106** 1.3" OLED-hez is), de a fentinél bonyolultabb és terjedelmesebb
  Forrás: github.com/olikraus/U8g2_Arduino
- **OLED_I2C:** egyszerű, **SSD1306** I2C kijezőre optimalizált könyvtár
  Forrás: www.rinkydinkelectronics.com/library.php?id=79

Mi most csak az Adafruit_GFX + Adafruit_SSD1306 könyvtárakkal foglalkozunk!

A szükséges programkönyvtárak telepítése

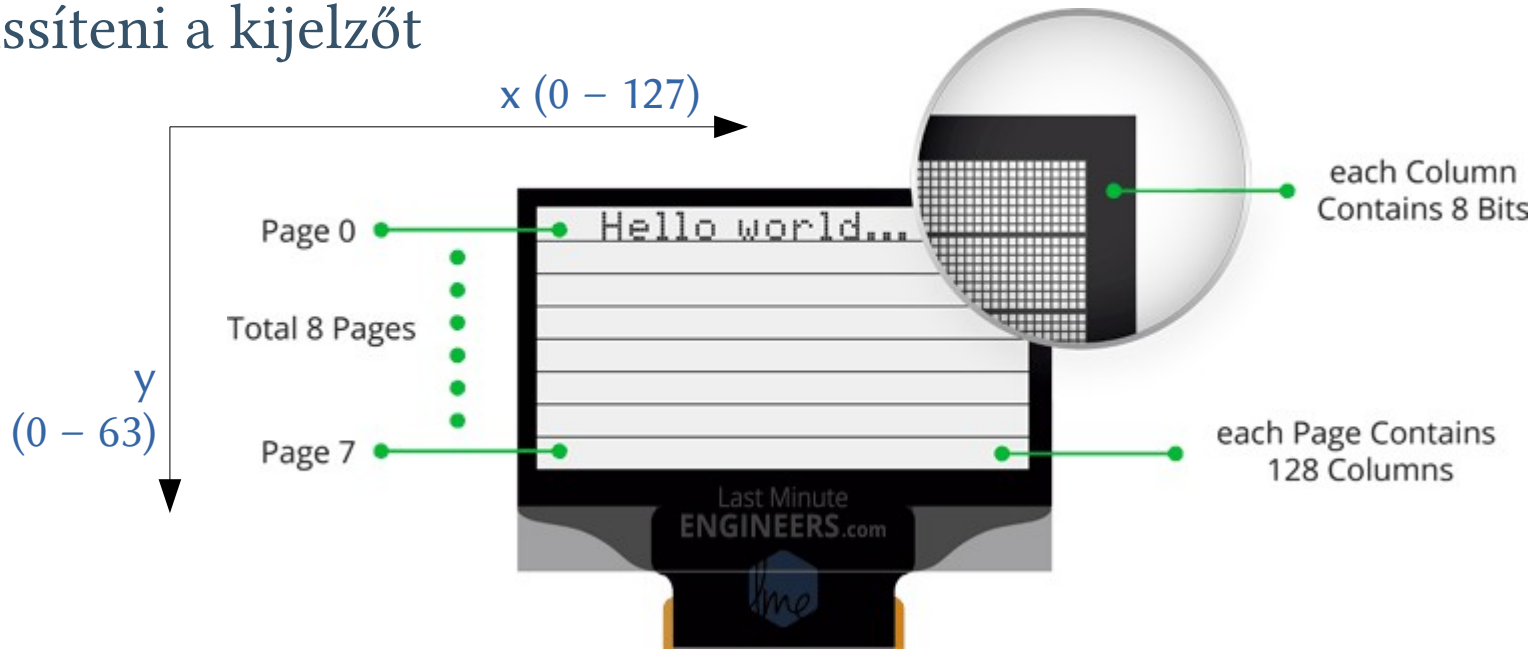
- Nyissuk meg a **Tools** menüben a **Manage Libraries** menüpontot!
- Keressük meg és telepítsük az **Adafruit GFX** könyvtárat!
- A keresősávba **GFX** helyett **ssd1306**-ot írva keressük meg és telepítsük az **Adafruit ssd1306** könyvtárat is!





SSD1306 használata Arduinoval

- Egy kitűnő bevezető tananyag található a Last Minute Engineers honlapján Interface OLED Graphic Display Module with Arduino ebből merítünk mi is. A tananyag az **Adafruit_SSD1306** könyvtárat használja
- A 128x64 képpontos kijelző 8 lapra van felosztva, amelyekben egy oszlop 8 képpontból áll, amelyeket egy bájt bitjei vezérelnek
- Minden lap 128 oszlopból áll, azaz 128 bájttal írható felül
- A teljes képet (1024 bájt) a memóriában kell összeállítani és innen kell frissíteni a kijelzőt



Az SSD1306+GFX könyvtár főbb metódusai

- **clearDisplay()** – töröl minden képpontot a bufferben
- **drawPixel(x,y, color)** – az x,y koordinátájú pixel beállítása
- **setTextColor(c[, bg])** – szöveg- és háttérszín megadása (0 vagy 1)
- **setTextSize(n)** – szövegméret megadása (n = 1 – 8)
- **setCursor(x,y)** – kurzor beállítása megadott pontba
- **print("message")** – szöveg kiírása
- **print(n,HEX)** – szám kiírása hexadecimális számrendszerben
- **print(n,DEC)** – szám kiírása tízes számrendszerben
- **display()** – képernyő frissítése (buffer kiírása)

Az SSD1306 osztály példányosítása

- Az **Adafruit_SSD1306** könyvtárat folyamatosan fejlesztik, ügyelve rá, hogy a régi programokkal kompatibilis maradjon, ezért többféle módon történhet a példányosítás:

Példányosítás régi módon

```
#include <Wire.h>
#include <Adafruit_SSD1306.h>
#include <Adafruit_GFX.h>
#define OLED_ADDR   0x3C

Adafruit_SSD1306 display(-1);

#if (SSD1306_LCDHEIGHT != 64)
  #error("Height incorrect, please fix\
          Adafruit_SSD1306.h!");
#endif

void setup() {
  display.begin(SSD1306_SWITCHCAPVCC,\
                OLED_ADDR);
  display.clearDisplay();
  ...
}
```

Új típusú példányosítás

```
#include <Wire.h>
#include <Adafruit_SSD1306.h>
#include <Adafruit_GFX.h>
#define OLED_ADDR   0x3C

#define SCREEN_WIDTH 128
#define SCREEN_HEIGHT 64
#define OLED_RESET    -1
Adafruit_SSD1306 display(SCREEN_WIDTH,
SCREEN_HEIGHT, &Wire, OLED_RESET);


void setup() {
  display.begin(SSD1306_SWITCHCAPVCC,\
                OLED_ADDR);
  display.clearDisplay();
  ...
}
```

oled_text_demo.ino

- A Last Minute Engineers honlapján található tananyagból vett mintaprogram segítségével mutatjuk be a kijelző használatát.
- **Egyszerű szövegkiírás: Hello world!** (az előző oldali példányosítás után folytatjuk a programot)


```
display.clearDisplay();
display.setTextSize(1);
display.setTextColor(WHITE);
display.setCursor(0,28);
display.println("Hello world!");
display.display();
delay(2000);
```


```
display.clearDisplay();
display.setTextColor(BLACK, WHITE);   //INVERTED
display.setCursor(0,28);
display.println("Hello world!");
display.display();
delay(2000);
```

oled_text_demo.ino

- **Szövegméret változtatása** – a **setTextSize(n)** függvénnyel megnagyobbíthatjuk a pixeleket, ezzel megnő a betűk mérete is


```
display.clearDisplay();
display.setTextColor(WHITE);
display.setCursor(0,24);
display.setTextSize(2);
display.println("Hello!");
display.display();
delay(2000);
```

Az alap font 7x10 pont
setTextSize(2) után
14x20 képpont méretű
lesz a fontméret

- **ASCII szimbólumok kiíratása** – a **print/println** hívásával szöveget, számot írhatunk ki, a **write()** metódussal pedig karakterkódot adhatunk meg, például **write(3)** egy szív karaktert ír ki


```
display.clearDisplay();
display.setCursor(0,24);
display.setTextSize(2);
display.write(3);          // Szívecske rajzolás
display.display();
delay(2000);
```

oled_text_demo.ino

- **Számok kiírása** – a print() metódussal nemcsak szöveget, hanem számokat is kiírathatunk (32 bites előjelt nélküli számokat adhatunk meg)


```
// Display Numbers
display.clearDisplay();
display.setTextSize(1);
display.setCursor(0,28);
display.println(123456789);  //Szám kiírása
display.display();
delay(2000);
```

- **Számrendszer megadása** – a **Print** osztály örökölt tulajdonságai miatt azt is megadhatjuk, hogy milyen számrendszerben írjuk ki a számot


```
display.clearDisplay();
display.setCursor(0,28);
display.print("0x"); display.print(0xFF, HEX);
display.print("(HEX) = ");
display.print(0xFF, DEC);
display.println("(DEC)");
display.display();
delay(2000);
```

További fontok használata

- A „beépített” 7x10 képpont méretű fonton kívül a **GFX** könyvtárral együtt települő további fontokat is használhatjuk:
  - Három fontcsalád (Serif, Sans, Mono)
  - négyféle méret (9, 12, 18 és 24 pont)
  - négyféle variáns (normál, italic/oblique, bold, bold italic/oblique)

- A használathoz be kell csatolni a használni kívánt fontok fejléc állományait, például:

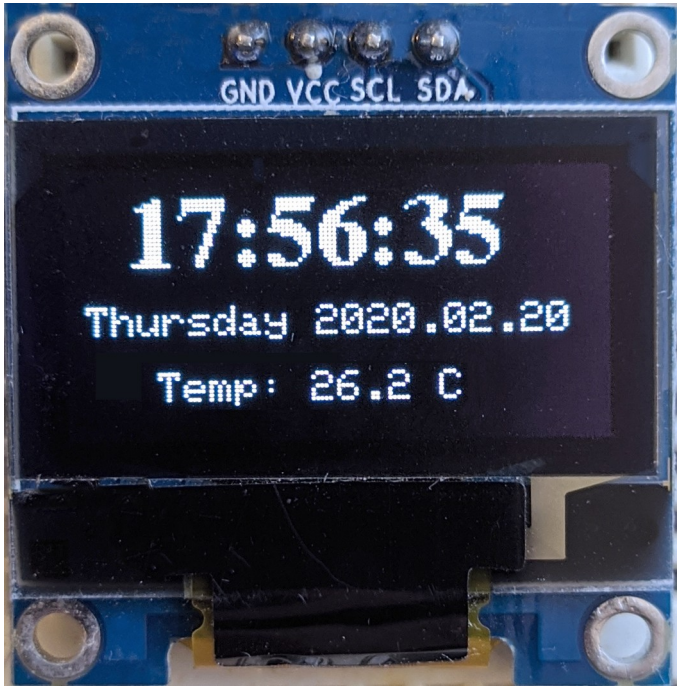
```
#include <Wire.h>
#include <Adafruit_GFX.h>
#include <Adafruit_SSD1306.h>
#include "Fonts/FreeSerifBold12pt7b.h"
```

- Ki kell választani a kívánt fontot, például:

```
display.setFont(&FreeSerifBold12pt7b);
```

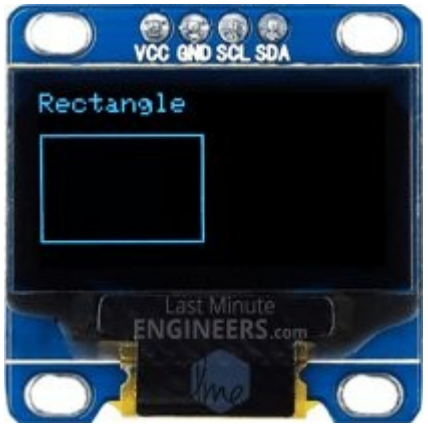
oled_setfont.ino

```
#include <Wire.h>
#include <Adafruit_GFX.h>
#include <Adafruit_SSD1306.h>
#include "Fonts/FreeSerifBold12pt7b.h"                    //Font betöltés
#define SCREEN_WIDTH 128 // OLED display width, in pixels
#define SCREEN_HEIGHT 64 // OLED display height, in pixels
Adafruit_SSD1306 display(SCREEN_WIDTH, SCREEN_HEIGHT, &Wire, -1);

void setup() {
  display.begin(SSD1306_SWITCHCAPVCC, 0x3C);
  display.clearDisplay();
  display.setFont(&FreeSerifBold12pt7b); //Font kiválasztása
  display.setTextSize(1);
  display.setTextColor(WHITE);
  display.setCursor(15,22);
  display.print("17:56:35");
  display.setFont();       //Vissza a beépített fonthoz
  display.setTextSize(1);
  display.setCursor(5,32);
  display.print("Thursday 2020.02.20");
  display.setCursor(20,45);
  display.print("Temp: 26.2 C");
  display.display();
}

void loop() {
}
```



Grafikus elemek: téglalap rajzolása

- **Üres téglalap rajzolása** – a **drawRect()** függvény öt paramétert kér: a bal felső sarok x,y koordinátái, szélesség, magasság és a tintaszín.




```
display.clearDisplay();
display.setTextSize(1);
display.setTextColor(WHITE);
display.setCursor(0,0);
display.println("Rectangle");
display.drawRect(0, 15, 60, 40, WHITE);
display.display(); delay(2000);
```

- **Kitöltött téglalap rajzolása** – a **fillRect()** metódussal történik, a paraméterek ugyanazok, mint az üres téglalap rajzolásnál


```
display.clearDisplay();
display.setTextSize(1);
display.setTextColor(WHITE);
display.setCursor(0,0);
display.println("Rectangle");
display.fillRect(0, 15, 60, 40, WHITE);
display.display(); delay(2000);
```

Grafikus elemek: lekerekített téglalap

- **Lekerekített téglalap rajzolása** – a **drawRoundRect()** függvény hat paramétert kér, melyből az 5. paraméter a lekerekítés sugara

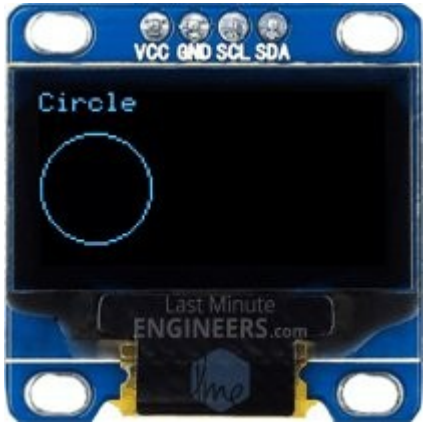

```
display.clearDisplay();
display.setTextSize(1);
display.setTextColor(WHITE);
display.setCursor(0,0);
display.println("Round Rectangle");
display.drawRoundRect(0, 15, 60, 40, 8, WHITE);
display.display(); delay(2000);
```

- **Kitöltött téglalap rajzolása** – a **fillRect()** metódussal történik, a paraméterek ugyanazok, mint az üres téglalap rajzolásnál


```
display.clearDisplay();
display.setTextSize(1);
display.setTextColor(WHITE);
display.setCursor(0,0);
display.println("Filled Round Rectangl");
display.fillRoundRect(0, 15, 60, 40, 8, WHITE);
display.display(); delay(2000);
```

Grafikus elemek: kör rajzolása

- **Üres kör rajzolása** – a **drawCircle()** függvény négy paramétert kér: a középpont x,y koordinátái, a sugár és a tintaszín.



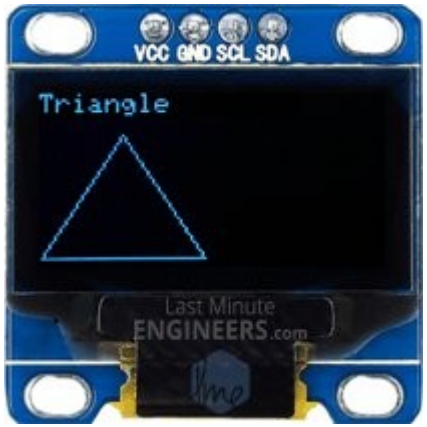

```
display.clearDisplay();
display.setTextSize(1);
display.setTextColor(WHITE);
display.setCursor(0,0);
display.println("Circle");
display.drawCircle(20, 35, 20, WHITE);
display.display(); delay(2000);
```

- **Kitöltött kör rajzolása** – a **fillCircle()** metódussal történik, a paraméterek ugyanazok, mint az üres kör rajzolásnál


```
display.clearDisplay();
display.setTextSize(1);
display.setTextColor(WHITE);
display.setCursor(0,0);
display.println("Filled Circle");
display.fillCircle(20, 35, 20, WHITE);
display.display(); delay(2000);
```

Grafikus elemek: háromszög rajzolása

- **Üres háromszög rajzolása** – a **drawTriangle()** függvény hét paramétert kér: a három csúcs x,y koordinátái, és a tintaszín.

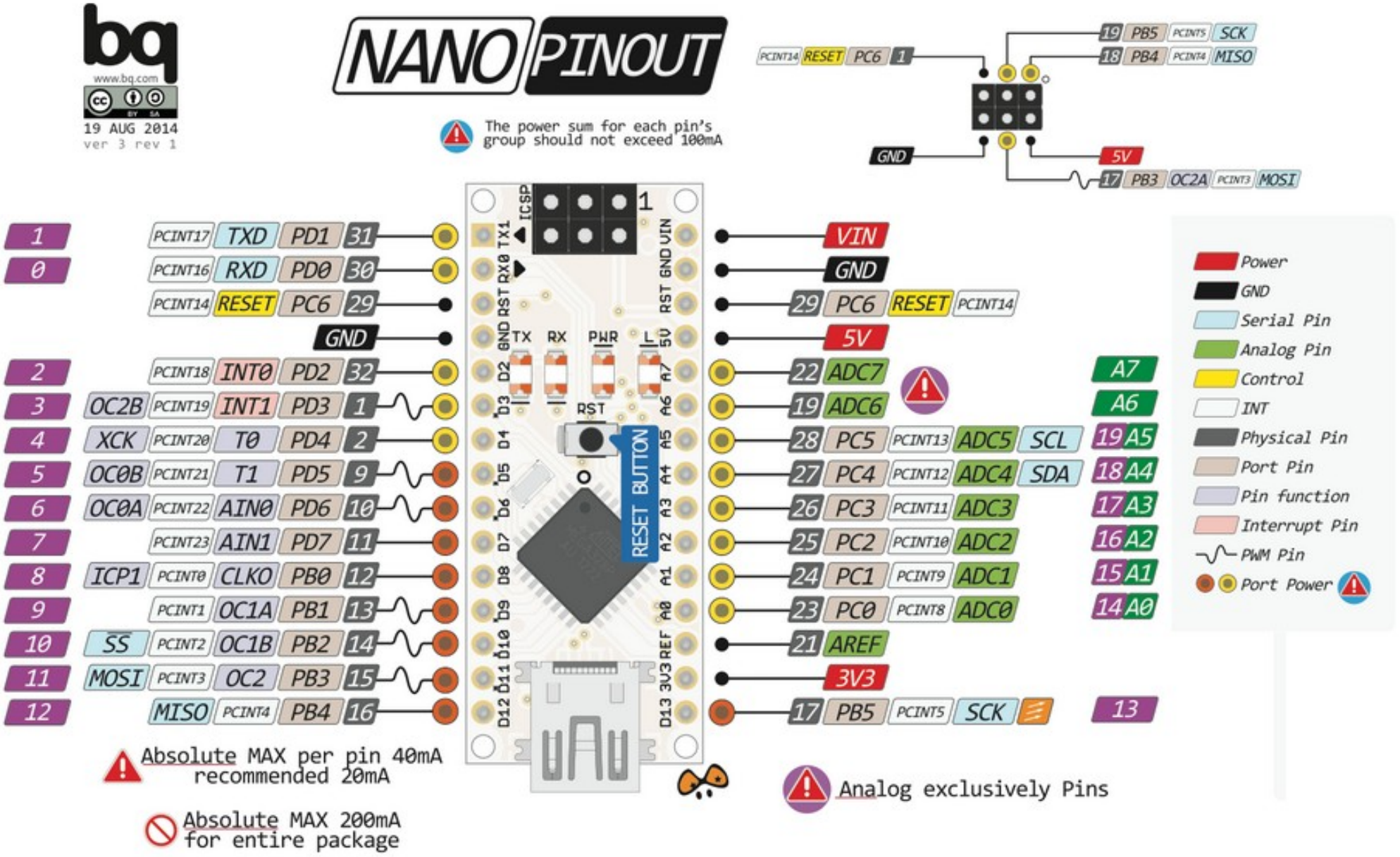



```
display.clearDisplay();
display.setTextSize(1);
display.setTextColor(WHITE);
display.setCursor(0,0);
display.println("Triangle");
display.drawTriangle(30,15, 0,60, 60,60, WHITE);
display.display(); delay(2000);
```

- **Kitöltött háromszög rajzolása** – a **fillTriangle()** metódussal történik, a paraméterek ugyanazok, mint az üres háromszög rajzolásnál


```
display.clearDisplay();
display.setTextSize(1);
display.setTextColor(WHITE);
display.setCursor(0,0);
display.println("Filled Triangle");
display.fillTriangle(30,15, 0,60, 60,60, WHITE);
display.display(); delay(2000);
```

Az Arduino nano kártya kivezetései



Ellenállás színkódok