Nyíregyházi Egyetem

Matematika és Informatika Intézet

Fényérzékelős Biztonsági Riasztórendszer Műszaki Dokumentáció

IoT Alapok (BMI1504L)

Biró Gergő, Biró Róbert Zsolt, Varga Tibor

Mérnökinformatika

Nyíregyháza, 2025.

Tartalom

1 Bevezetés

A projekt célja egy Arduino alapú, fényváltozást érzékelő riasztórendszer megvalósítása, amely behatolás esetén vizuális (LCD + RGB LED) és akusztikus (piezo buzzer) jelzést ad. A rendszer kezeli a némító funkciót, valamint képes automatikusan kalibrálni a környezeti fényviszonyokra.

1.1 Források:

- Tinkercad kód
- Tinkercad kapcsolási rajz
- Tinkercad fotók
- Arduino kód
- Arduino fotók

2 Rendszer működésének áttekintése

2.1 Fő funkciók

1. Környezeti fény kalibrálása indításkor
2. Folyamatos fényerő-figyelés (LDR szenzor)
3. Riasztás túl nagy fényváltozás esetén
4. Némítás / visszaállítás gomb segítségével
5. Vizuális visszajelzés LCD-n és RGB LED-en

2.2 Rendszerállapotok

Állapot	Megnevezés	Jelzés	Leírás
0	FIGYEL	zöld LED	Alap működés, fényérzékelés
1	RIASZTÁS	villogó piros + buzzer	Behatolás észlelve
2	NÉMÍTVA	piros LED	Riasztás letiltva gombnyomással

3 Fizikai felépítés

3.1 Anyagjegyzék

Név	Mennyiség	Összetevő
U1	1	Arduino Uno R3
D1	1	RCBG LED RGB
S1	1	Nyomógomb
U2	1	PCF8574-alapú, 39 (0x27) LCD 16 x 2 (I2C)
PIEZO1	1	Piezo
R3, R4, R5, R6, R2	5	10 kΩ Ellenállás
R1	1	Fotoellenállás

3.2 Arduino bekötések

3.2.1 Szenzorok és kezelőszervek

Eszköz	Arduino pin	Megjegyzés
Fotoellenállás (LDR) + 10 kΩ	A0	Feszültségosztóként kötve 10 kΩ ellenállással; az analogRead(A0) méri a fényerőt, indításkor ebből számol alap fényt (baseLight) és riasztási küszöböt (threshold)
Nyomógomb	D2	Külső 10 kΩ lehúzó ellenállással GND-re; rövid megnyomás vált a RIASZTÁS → NÉMÍTVA → FIGYEL állapotok között

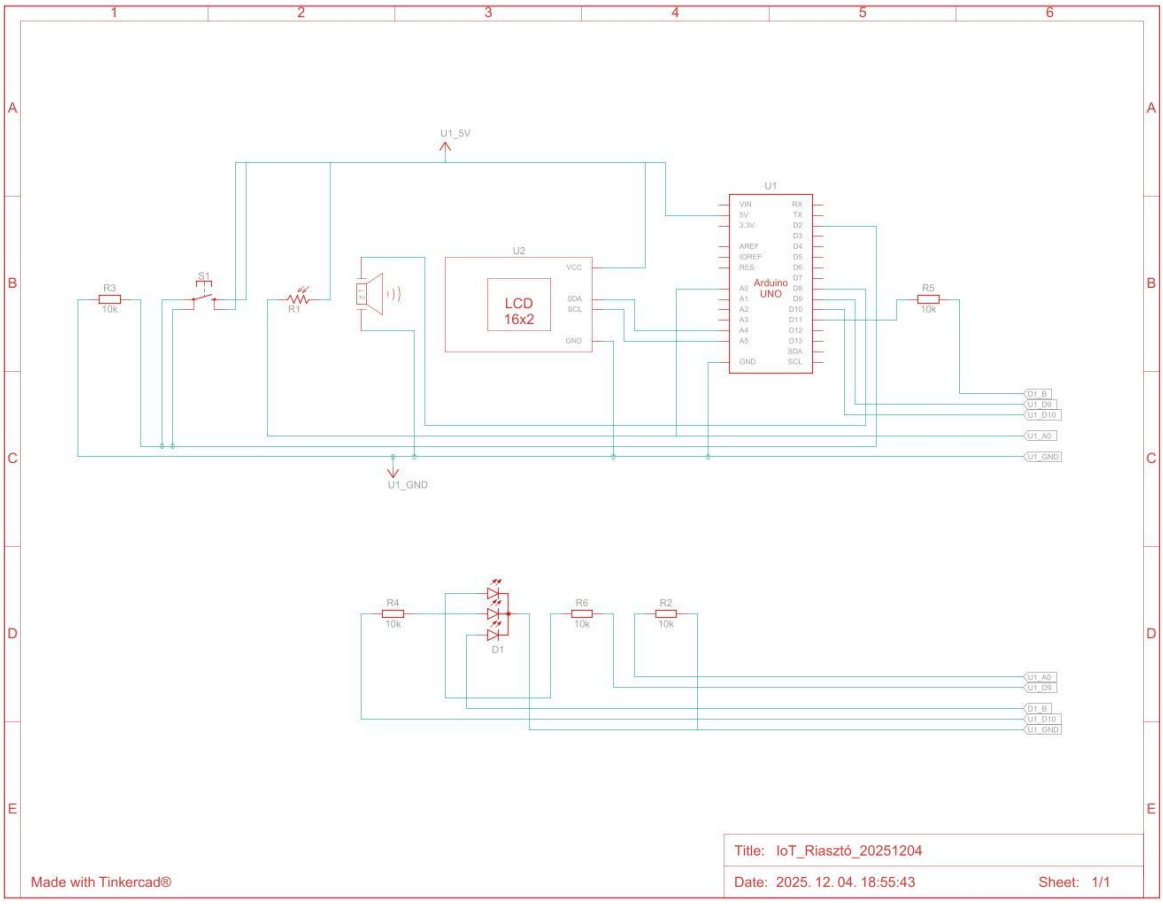
3.2.2 Vezérlő elemek

Eszköz	Arduino pin	Megjegyzés
Piezo buzzer	D8	5 V-os piezo; tone() függvénnyel riasztási hangot ad, villogó piros LED-del együtt működik
RGB LED – piros	D9	PWM kimenet; riasztási állapot jelzésére (villogó/folyamatos piros)
RGB LED – zöld	D10	PWM kimenet, Figyel (normál állapot) kijelzésére
RGB LED – kék	D11	PWM kimenet; jelen projektben nincs külön funkciója, de színkeveréshez használható

3.2.3 LCD 16x2 (I2C)

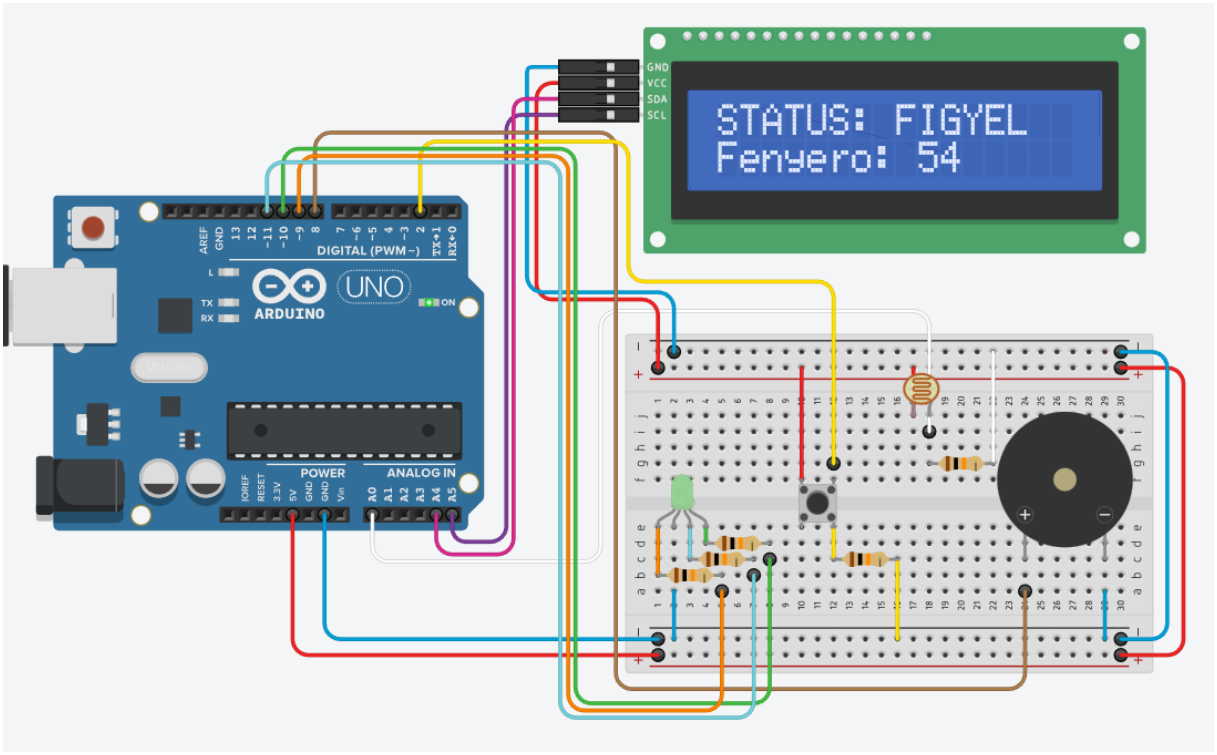
Csatlakozó / adat	Arduino / érték	Megjegyzés
GND	GND	Közös föld az LCD és a háttérvilágítás számára
VCC	5V	Tápellátás az LCD modulnak
SDA	A4	I2C adatvonal, a Wire könyvtár használja
SCL	A5	I2C órajelvonal

3.2.4 Kapcsolási rajz



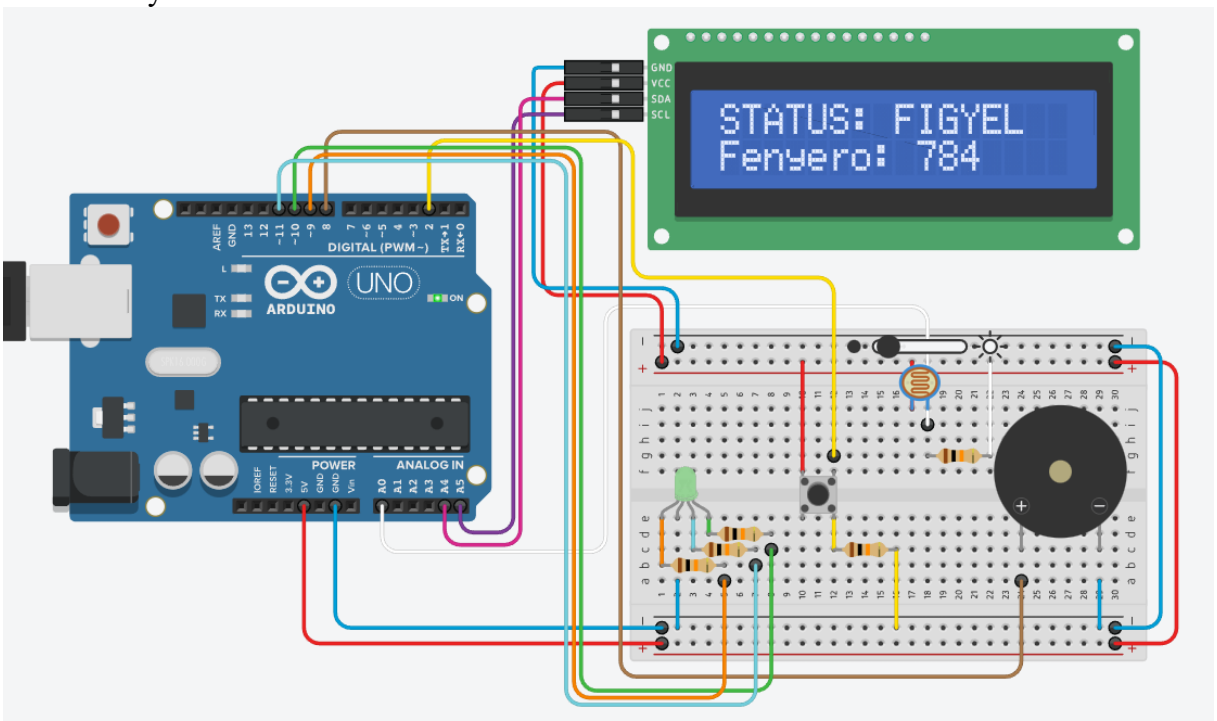
4 A rendszer működésének állapotdiagramja

1. Kiinduló helyzet



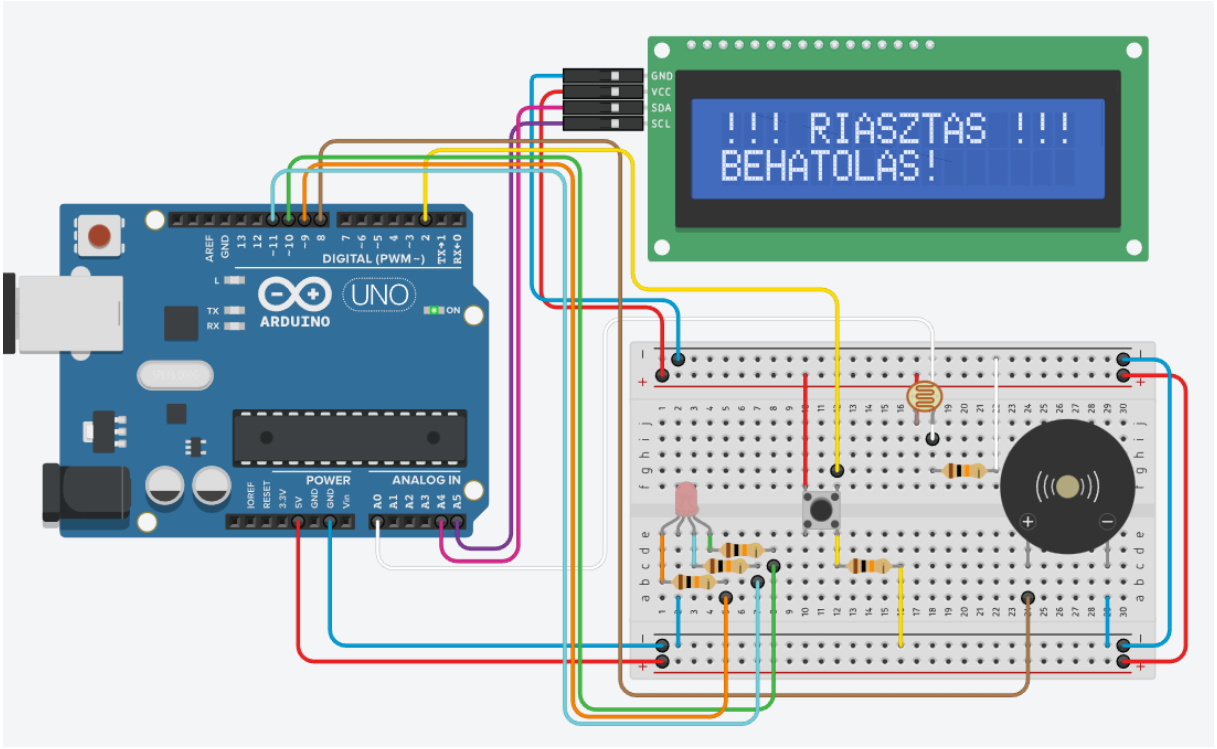
- LCD: „STATUS: FIGYEL”
- RGB LED: zöld
- Folyamatos fényérték kijelzés „Fenyero: <érték>”

2. Fénymérés



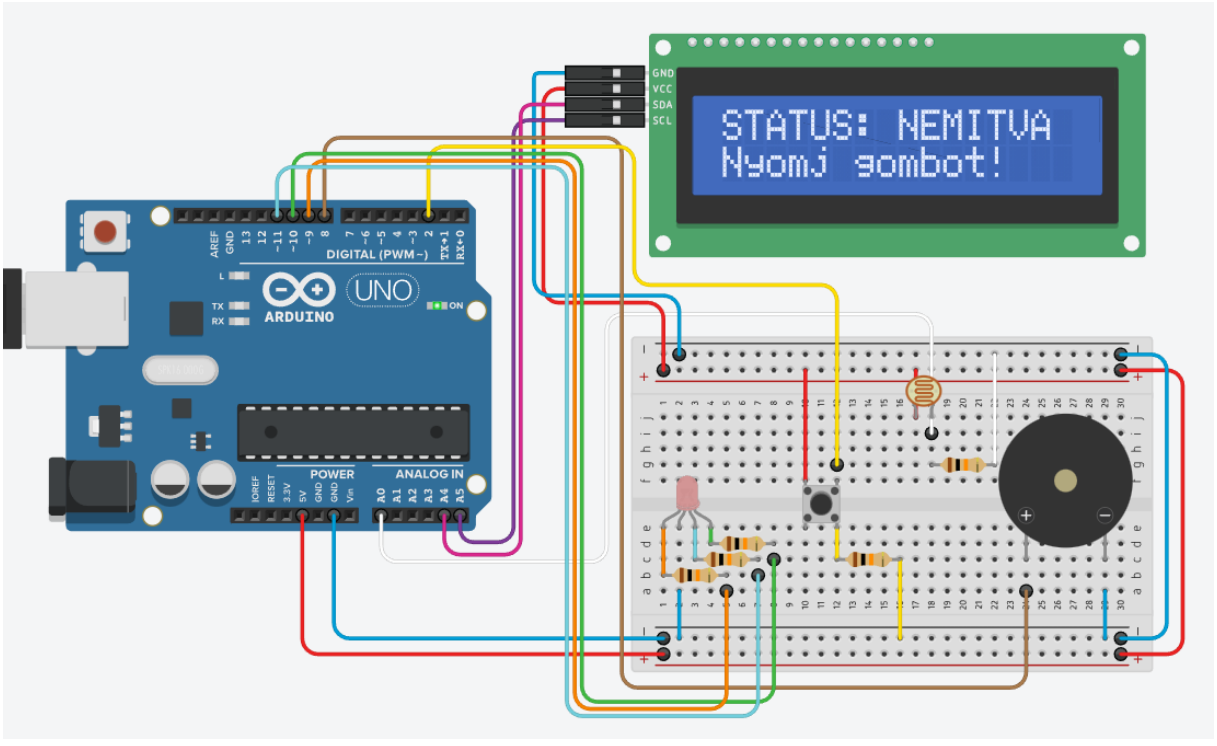
- A rendszer a „threshold” (küszöb) fölötti fényváltozást keresi.

3. Riasztás



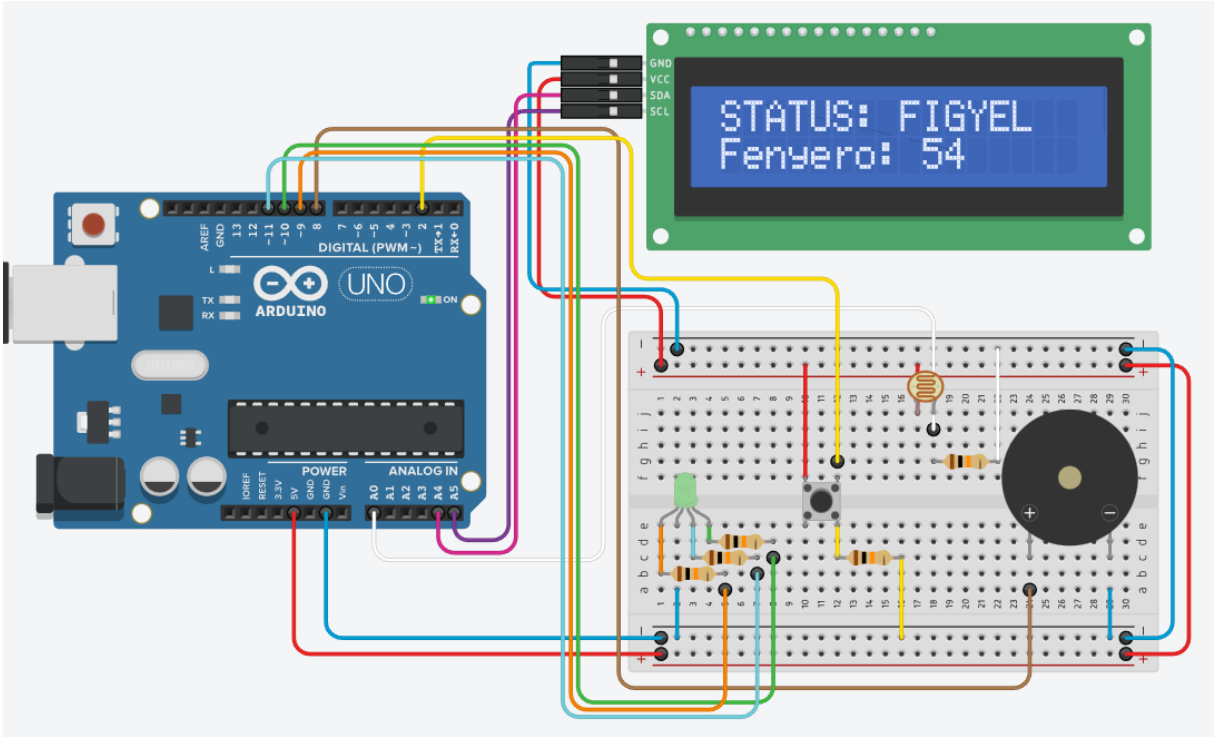
- Villogó piros LED
- Buzzer ki-be kapcsol
- LCD: „!!! RIASZTAS !!! BEHATOLAS!”

4. Némítás



- Buzzer elnémul
- Folyamatos piros LED
- LCD: „STATUS: NEMITVA – Nyomj gombot!”

5. Visszaállítás



Visszatérés FIGYEL módba.

5 TINKERCAD program működése

5.1 Programkód

5.1.1 Pin beállítások

```
//Fényérzékelő
const int sensorPin = A0;
//Gomb
const int buttonPin = 2;
//Csipogó
const int buzzerPin = 8;
//RGB LED
const int redPin   = 9;
const int greenPin = 10;
const int bluePin  = 11;
```

5.1.2 Változók

```
int lightLevel = 0;
int threshold = 700; //érzékenység kalibrálása
```

```
int baseLight = 0; //induláskori alap fényerő

int systemState = 0; // 0=Figyel, 1=Riaszt, 2=Némít
int lastButtonState = LOW;
unsigned long lastBlinkTime = 0;
unsigned long lastScreenUpdate = 0;
bool ledState = false;

void setup() {
  pinMode(redPin, OUTPUT);
  pinMode(greenPin, OUTPUT);
  pinMode(bluePin, OUTPUT);
  pinMode(buzzerPin, OUTPUT);
  pinMode(buttonPin, INPUT); //GND-re kötve

  Serial.begin(9600);

  lcd.init();
  lcd.backlight();

  lcd.setCursor(0,0);
  lcd.print("BIZT. RENDSZER");
  lcd.setCursor(0,1);
  lcd.print("Kalibralas...");
```

5.1.3 Fényerő kalibrálás

```
  long sum = 0;
  for (int i = 0; i < 100; i++) {
    int v = analogRead(sensorPin);
    sum += v;
    delay(10);
```

```
  }
  baseLight = sum / 100; // induló átlag
  threshold = baseLight + 800; // ennél erősebb fénynél riaszt

  Serial.print("Base: ");
  Serial.println(baseLight);
  Serial.print("Threshold: ");
  Serial.println(threshold);

  lcd.clear();
  lcd.setCursor(0,0);
  lcd.print("Alap feny: ");
  lcd.print(baseLight);
  lcd.setCursor(0,1);
  lcd.print("Kuszob: ");
  lcd.print(threshold);
  delay(1500);

  lcd.clear();
}

void loop() {
  lightLevel = analogRead(sensorPin);
  // Serial.println(lightLevel); // kijelzőn követhető mért érték
```

5.1.4 Gomb kezelés

```
  int currentButtonState = digitalRead(buttonPin);
  if (currentButtonState == HIGH && lastButtonState == LOW) {
    if (systemState == 1) systemState = 2; // Némítás
    else if (systemState == 2) systemState = 0; // Figyelés újbóli elindítása
    delay(300);
```

```
    lcd.clear();
  }
  lastButtonState = currentButtonState;
```

5.1.5 Kijelzés logika

```
// 0: FIGYEL (zöld LED)
  if (systemState == 0) {
    setColor(0, 255, 0);
    noTone(buzzerPin);

// Fényre riaszt: ha a fény meghaladja a beállított értéket
    if (lightLevel > threshold) {
      systemState = 1;
      lcd.clear();
    }

    if (millis() - lastScreenUpdate > 200) {
      lastScreenUpdate = millis();
      lcd.setCursor(0,0);
      lcd.print("STATUS: FIGYEL  ");
      lcd.setCursor(0,1);
      lcd.print("Fenyero: ");
      lcd.print(lightLevel);
      lcd.print("    ");
    }
  }

// 1: RIASZTÁS (villogó piros LED + csipogó)
  else if (systemState == 1) {
    lcd.setCursor(0,0);
    lcd.print("!!! RIASZTAS !!!");
```

```
    lcd.setCursor(0,1);
    lcd.print("BEHATOLAS!      ");

    if (millis() - lastBlinkTime > 150) {
      lastBlinkTime = millis();
      ledState = !ledState;
      if (ledState) {
        setColor(255, 0, 0);
        tone(buzzerPin, 1000);
      } else {
        setColor(0, 0, 0);
        noTone(buzzerPin);
      }
    }
  }

// 2: NEMÍTVA (folyamatos piros LED, nincs hang)
  else if (systemState == 2) {
    setColor(255, 0, 0);
    noTone(buzzerPin);

    lcd.setCursor(0,0);
    lcd.print("STATUS: NEMITVA ");
    lcd.setCursor(0,1);
    lcd.print("Nyomj gombot!   ");
  }

  delay(10);
}
```

5.1.6 RGB LED vezérlés

```
void setColor(int r, int g, int b) {
```

```
  analogWrite(redPin,   r);
  analogWrite(greenPin, g);
  analogWrite(bluePin,  b);
}
```

5.2 Állapotkezelés

0 – FIGYEL
- Zöld LED
- RIASZTAS állapotba vált ha: „lightLevel” nagyobb mint „threshold”

1 – RIASZTAS
- Villogó piros LED
- Buzzer hang
- LCD riasztási üzenet
- Gombnyomásra NEMITVA állapotba kerül

2 – NEMITVA
- Folyamatos piros LED
- Nincs buzzer
- Gombnyomásra visszatér FIGYEL módba

6 Felhasználói interfész (megjelenítés)

Állapot	Üzenet
FIGYEL	„STATUS: FIGYEL Fenyero: <érték>”
RIASZTÁS	„!!! RIASZTAS !!! BEHATOLAS!”
NÉMÍTVA	„STATUS: NEMITVA Nyomj gombot!”
INDÍTÁS	„BIZT. RENDSZER Kalibralas…”

7 ARDUINO Megvalósítás

7.1 Arduino kód

```
#include <Wire.h>
#include <LiquidCrystal_I2C.h>
LiquidCrystal_I2C lcd(0x27, 16, 2);
```

7.1.1 Pin beállítások

```
// Fényérzékelő
const int sensorPin = A0;
// Gomb
const int buttonPin = 2;
// Csipogó
const int buzzerPin = 8;
// RGB LED
const int redPin = 9;
const int greenPin = 10;
const int bluePin = 11;
```

7.1.2 Változók

```
int lightLevel = 0;
int threshold = 400; //érzékenység kalibrálása

int systemState = 0; // 0=Figyel, 1=Riaszt, 2=Némítva
int lastButtonState = LOW;
unsigned long lastBlinkTime = 0;
unsigned long lastScreenUpdate = 0;
bool ledState = false;

void setup() {
  pinMode(redPin, OUTPUT);
  pinMode(greenPin, OUTPUT);
  pinMode(bluePin, OUTPUT);
  pinMode(buzzerPin, OUTPUT);
  pinMode(buttonPin, INPUT);

  Serial.begin(9600);
```

7.1.3 LCD indítása

```
  lcd.init();
  lcd.backlight();

// Üdvözlés
  lcd.setCursor(0,0);
  lcd.print("BIZTONSAGI RDSR.");
  lcd.setCursor(0,1);
  lcd.print("Rendszer indul..");

// Teszt villanás
  setColor(255, 255, 255);
  delay(1500);
  setColor(0, 0, 0);
  lcd.clear();
}

void loop() {
  lightLevel = analogRead(sensorPin);
```

7.1.4 Gomb kezelése

```
  int currentButtonState = digitalRead(buttonPin);
  if (currentButtonState == HIGH && lastButtonState == LOW) {
    if (systemState == 1) systemState = 2; // Némítás
    else if (systemState == 2) systemState = 0; // Figyelés újbóli elindítása
    delay(300);
    lcd.clear();
  }
  lastButtonState = currentButtonState;
```

7.1.5 Kijelzés logika

```
// 0: FIGYEL (zöld LED)
  if (systemState == 0) {
    setColor(0, 255, 0); // Zöld
    noTone(buzzerPin);
// Fényre riaszt: ha a fény meghaladja a beállított értéket
   if (lightLevel < threshold) {
      systemState = 1;
      lcd.clear();
    }

// Kijelző frissítése
    if (millis() - lastScreenUpdate > 200) {
      lastScreenUpdate = millis();
      lcd.setCursor(0,0);
      lcd.print("STATUS: FIGYEL  ");
      lcd.setCursor(0,1);
      lcd.print("Fenyero: ");
      lcd.print(lightLevel);
      lcd.print("    ");
    }
  }

// 1: RIASZTÁS (villogó piros LED + csipogó)
  else if (systemState == 1) {
    lcd.setCursor(0,0);
    lcd.print("!!! RIASZTAS !!!");
    lcd.setCursor(0,1);
    lcd.print("BEHATOLAS!      ");
```

```
    if (millis() - lastBlinkTime > 150) {
      lastBlinkTime = millis();
      ledState = !ledState;
      if (ledState) { setColor(255, 0, 0); tone(buzzerPin, 1000); }
      else { setColor(0, 0, 0); noTone(buzzerPin); }
    }
  }

// 2: NEMÍTVA (folyamatos piros LED, nincs hang)
  else if (systemState == 2) {
    setColor(255, 0, 0);
    noTone(buzzerPin);

    lcd.setCursor(0,0);
    lcd.print("STATUS: NEMITVA ");
    lcd.setCursor(0,1);
    lcd.print("Nyomj gombot!   ");
  }

  delay(10);
}
```

7.1.6 RGB LED vezérlés

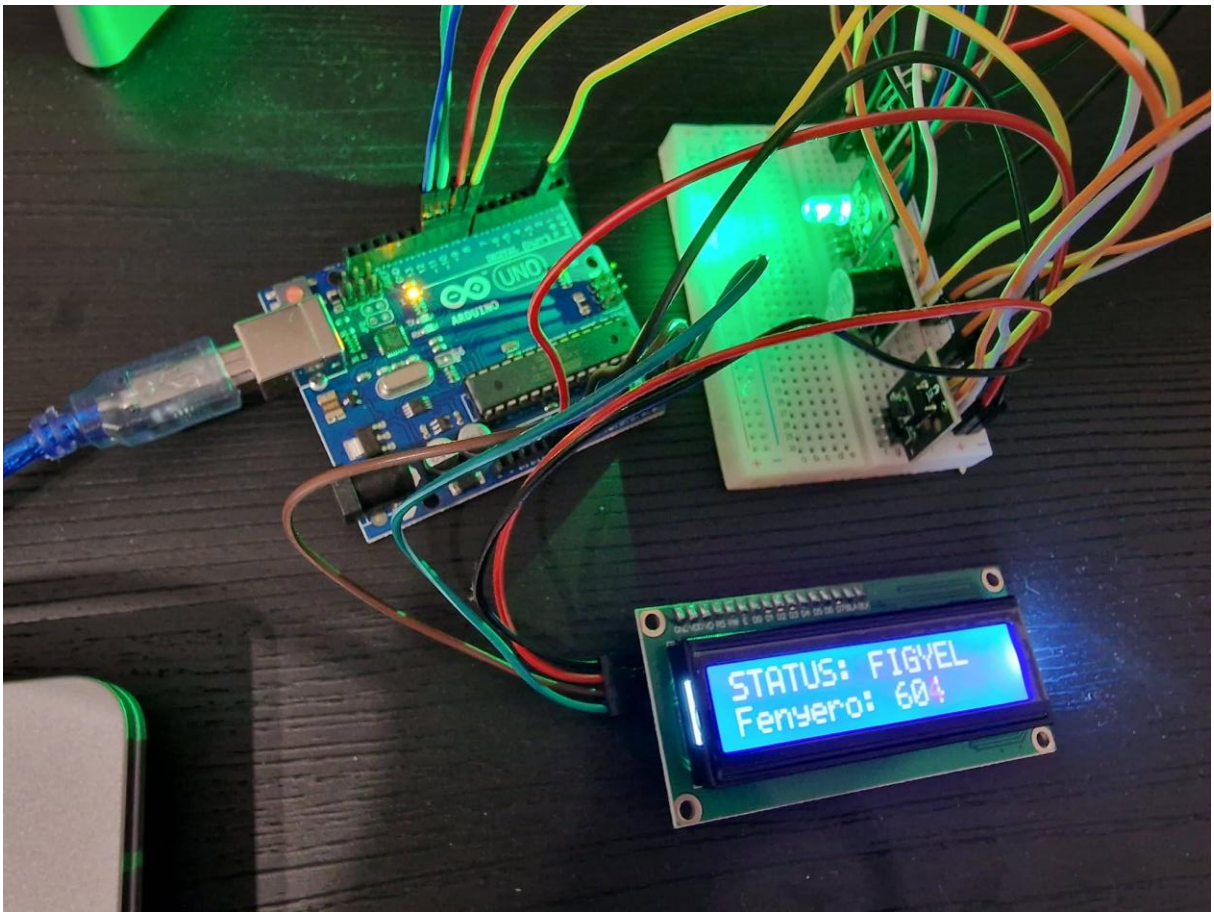
```
void setColor(int r, int g, int b) {
  analogWrite(redPin, r);
  analogWrite(greenPin, g);
  analogWrite(bluePin, b);
}
```

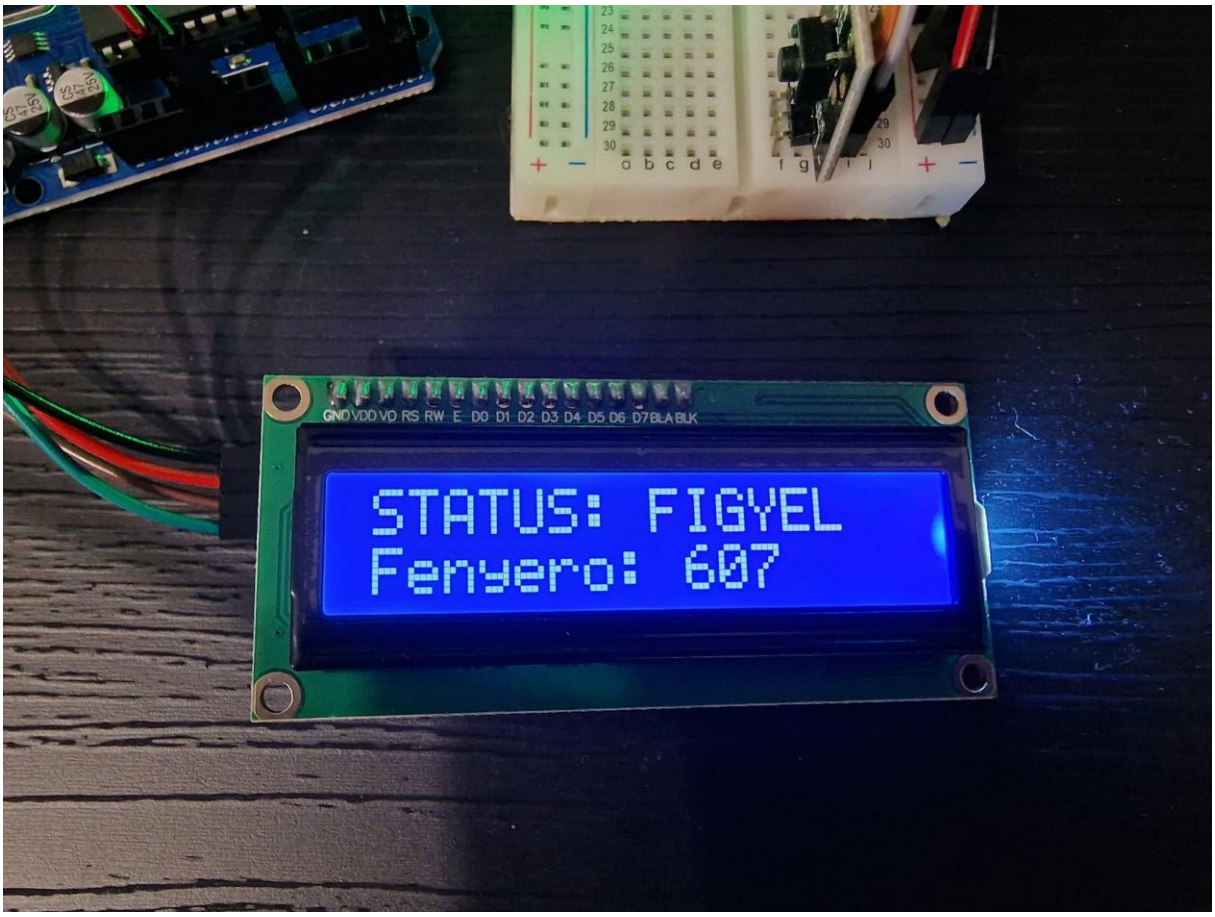
7.2 Bekötési mátrix

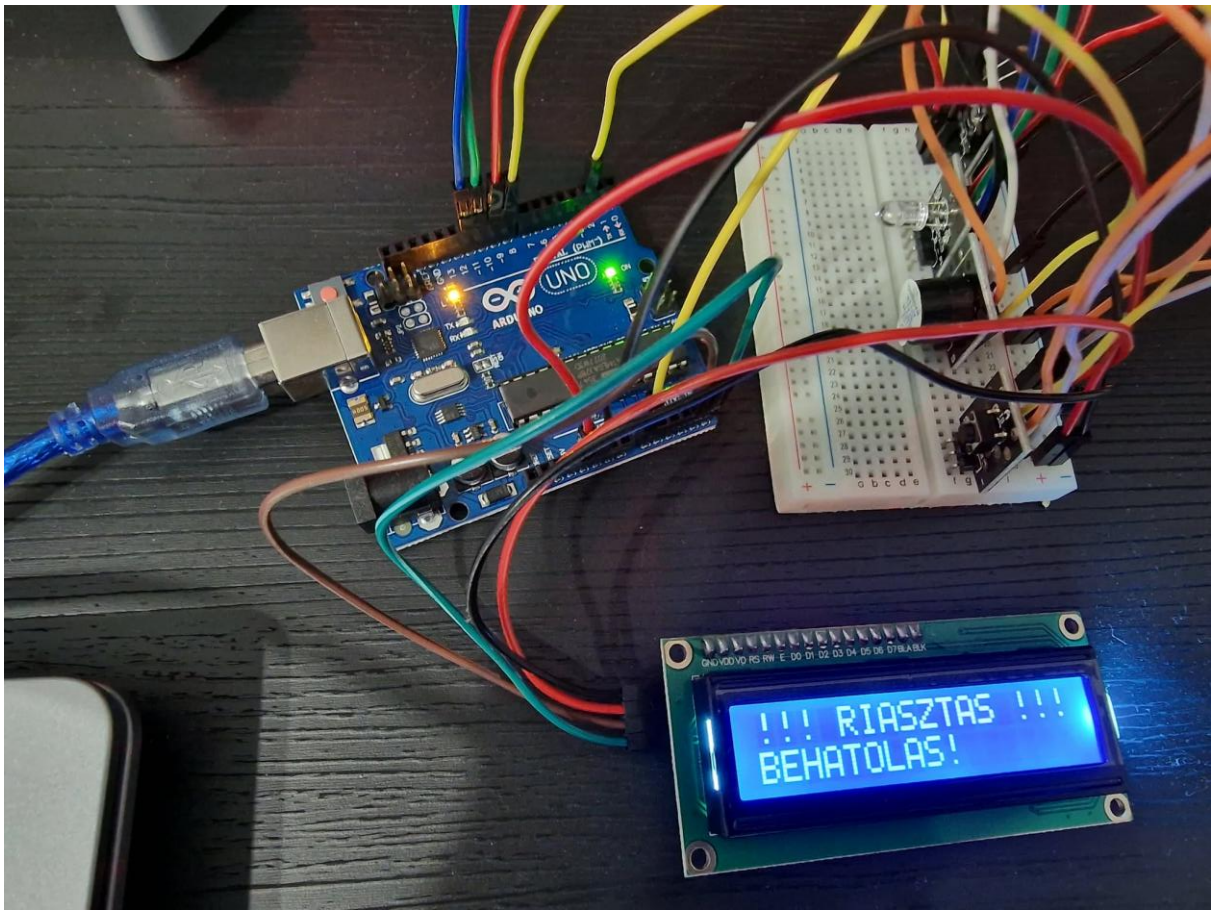
Komponens	Láb/ Funkció	Csatlakozás
Áramellátás	Arduino 5V	Breadboard PIROS (+) sáv
	Arduino GND	Breadboard KÉK (–) sáv

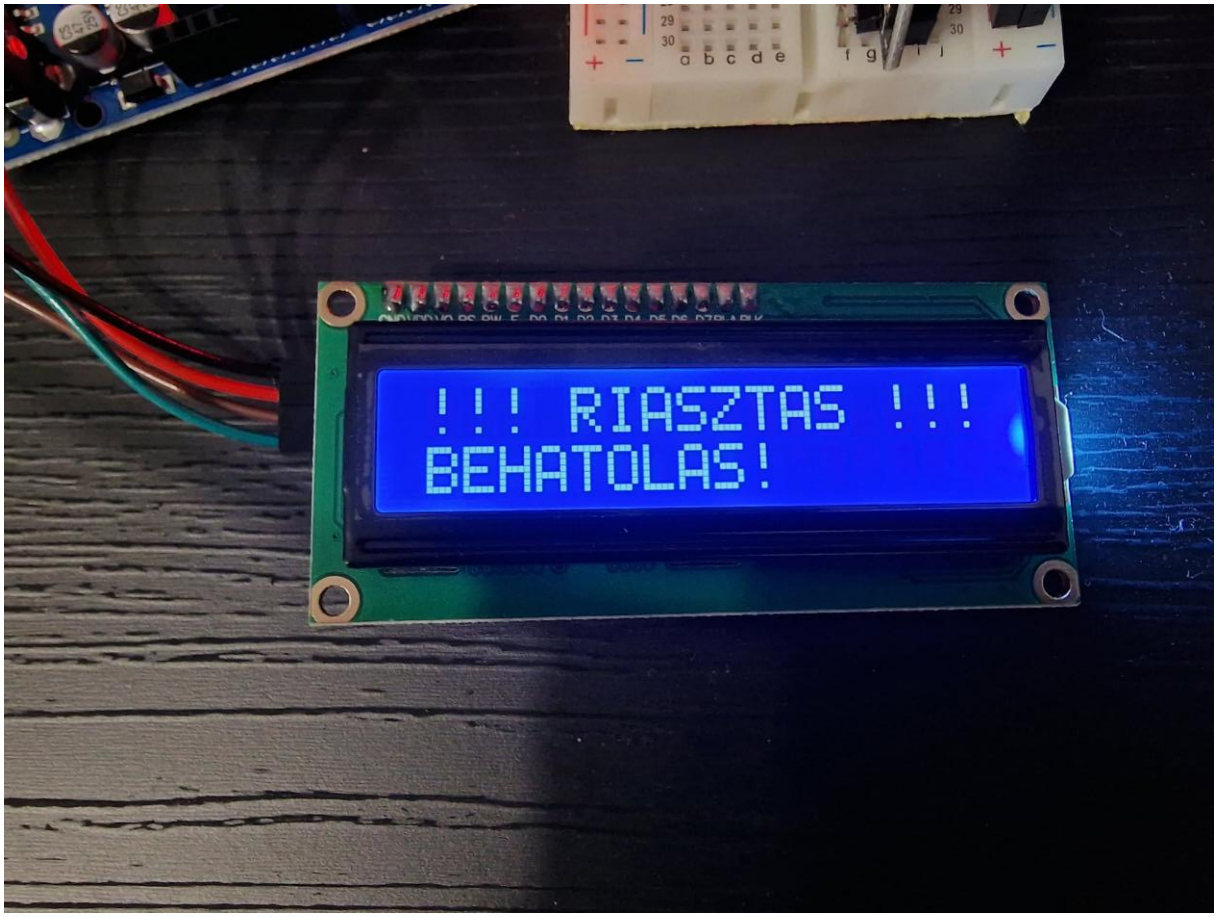
Fényellenállás (Photoresistor)	– (GND)	Breadboard KÉK sáv
	Középső (VCC)	Breadboard PIROS sáv
	S (Jel)	Arduino A0
RGB LED	– (GND)	Breadboard KÉK sáv
	R (Piros)	Arduino D9
	G (Zöld)	Arduino D10
	B (Kék)	Arduino D11
Nyomógomb (Button)	– (GND)	Breadboard KÉK sáv
	Középső (VCC)	Breadboard PIROS sáv
	S (Jel)	Arduino D2
Aktív Buzzer	– (GND)	Breadboard KÉK sáv
	S (Jel)	Arduino D8
LCD 1602 (I2C)	GND	Breadboard KÉK sáv
	VCC	Breadboard PIROS sáv
	SDA	Arduino A4
	SCL	Arduino A5

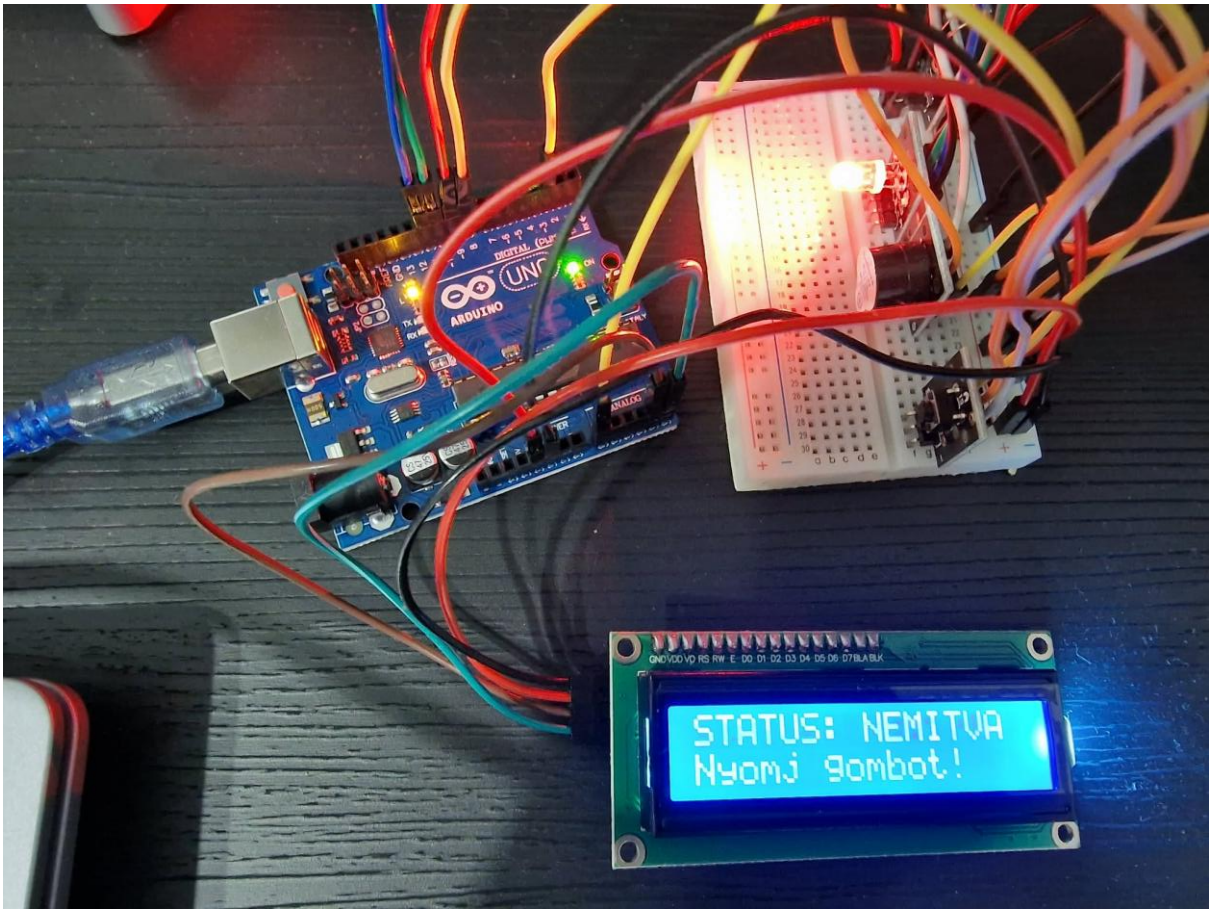
7.3 Rendszer működésének fizikai megvalósítása

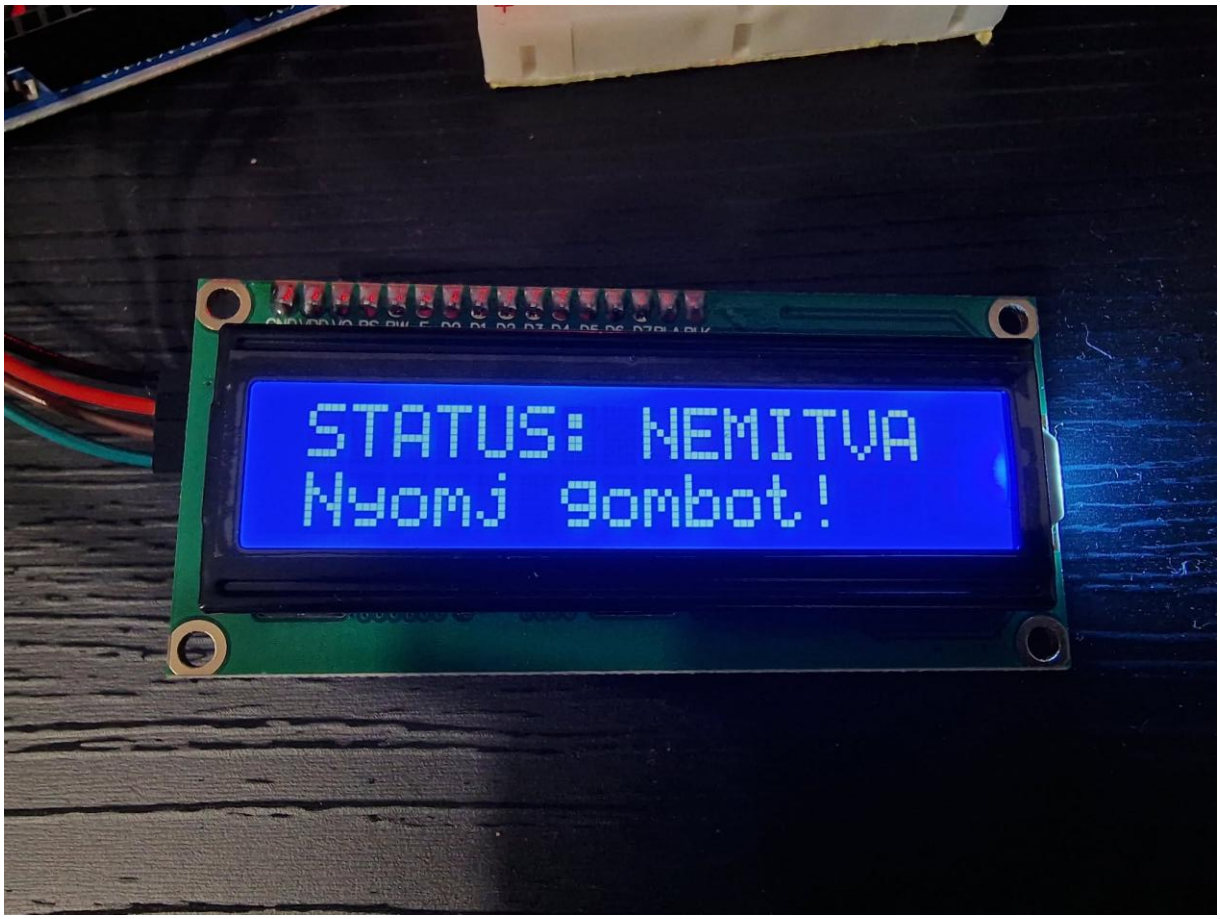



GND VDD VO RS RW E D0 D1 D2 D3 D4 D5 D6 D7 BLA BLK
STATUS: FIGYEL
Fenyero: 607


UNO
!!! RIASZTAS !!!
BEHATOLAS!


!!! RIASZTAS !!!
BEHATOLAS!


STATUS: NEMITVA
Nyomj gombot!


STATUS: NEMITVA
Nyomj gombot!