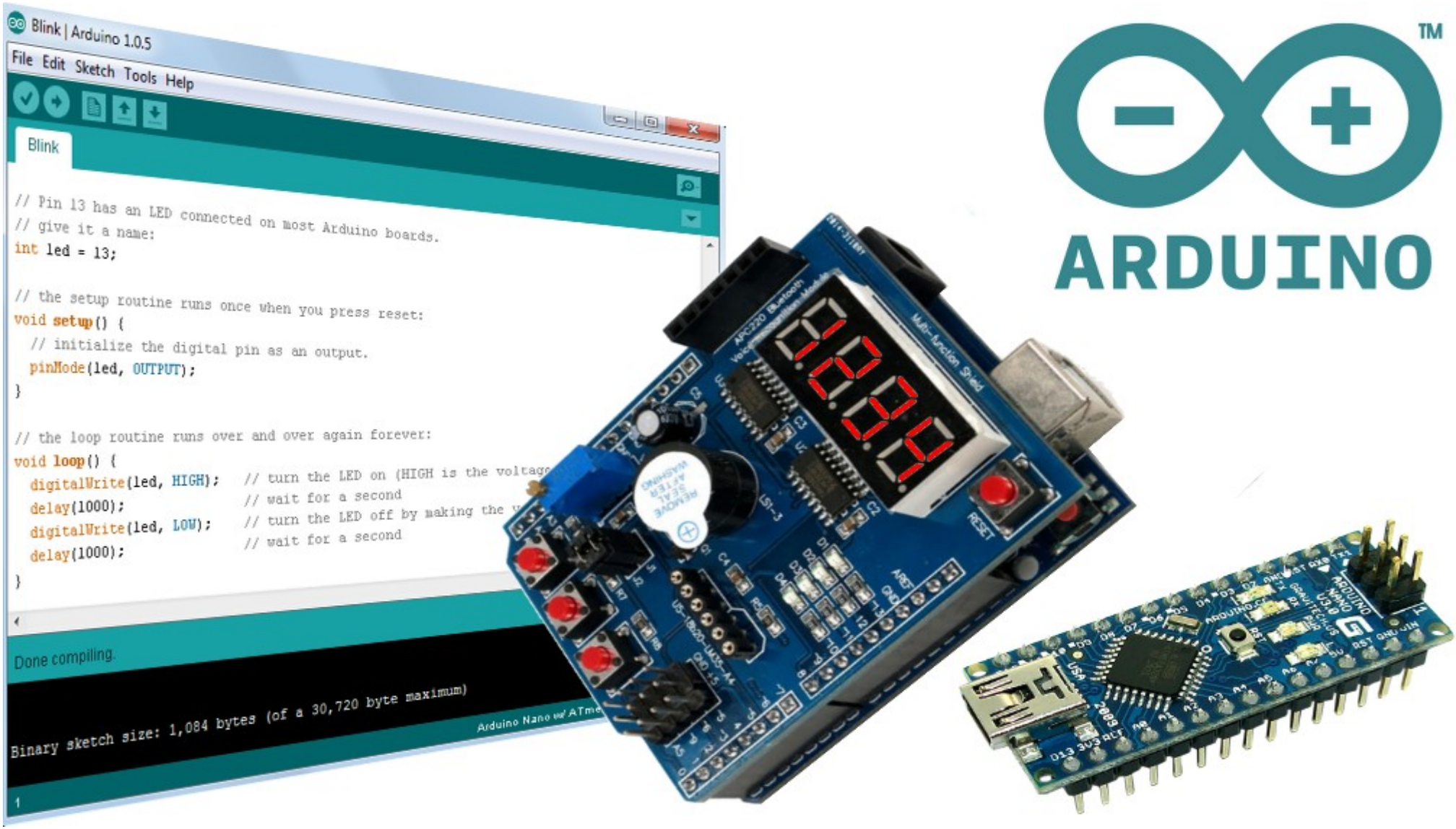
Arduino tanfolyam kezdőknek és haladóknak



11. Falkövető robot, I2C OLED kijelzők

Felhasznált és ajánlott irodalom

Ajánlott olvasnivalók:

- ALL ABOUT CIRCUITS : **How to Build a Wall-Following Robot**
- ENGINEERS GARAGE An EE World Online Resource : **Efficient Wall Following Robot ...**
- Last Minute ENGINEERS : **Interface L298N DC Motor Driver Module with Arduino**

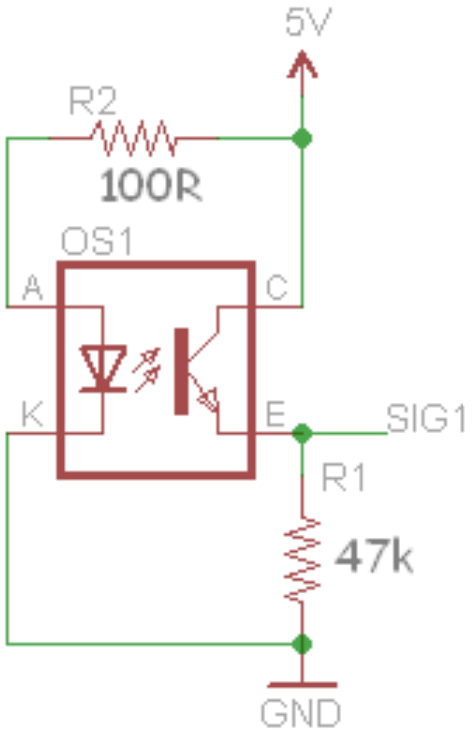
Korábbi előadások a témakörben:

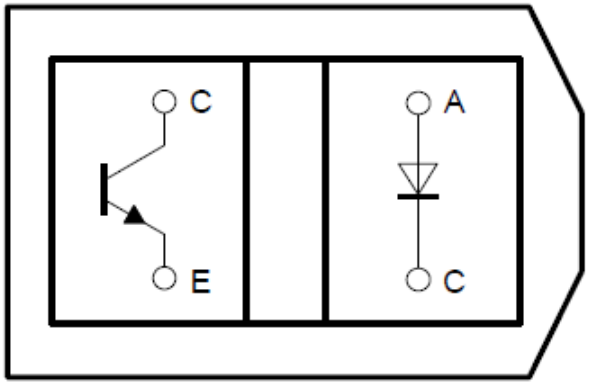
- Kisteljesítményű egyenáramú motorok vezérlése (2015. december 17.)
  előadásvázlat mintaprogramok
- Optoérzékelők, motorvezérlés, bevezetés a robotikába (2020. március 19.)
  előadásvázlat mintaprogramok video
- Motorvezérlés, bevezetés a robotikába (2021. január 28.)
  előadásvázlat mintaprogramok video
- Reflexiós fényérzékelők, akadályelkerülő robot (2021. február 11.)
  előadásvázlat mintaprogramok video

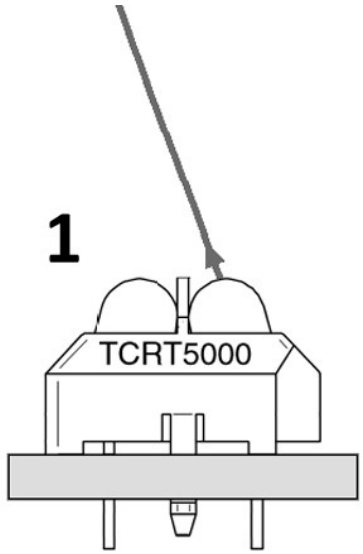
TCRT5000 reflektív optikai érzékelő

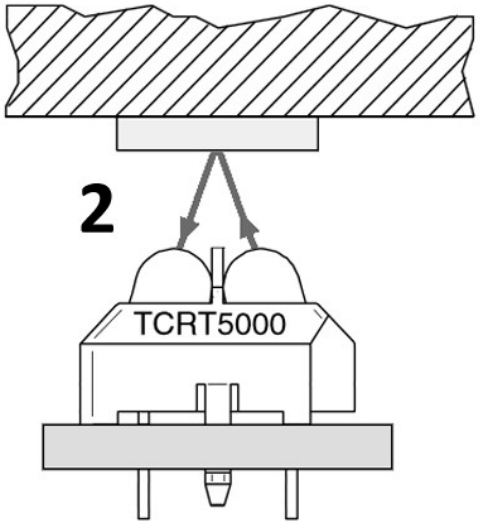
Egy IR LED-et és egy fototranzisztort tartalmaz.

Működési elv: ha nincs akadály, a fény nem verődik vissza, a tranzisztor nem érzékel fényt.
Visszaverődés esetén a távolságtól és a reflexiós tényezőtől függ a visszaverődési arány.





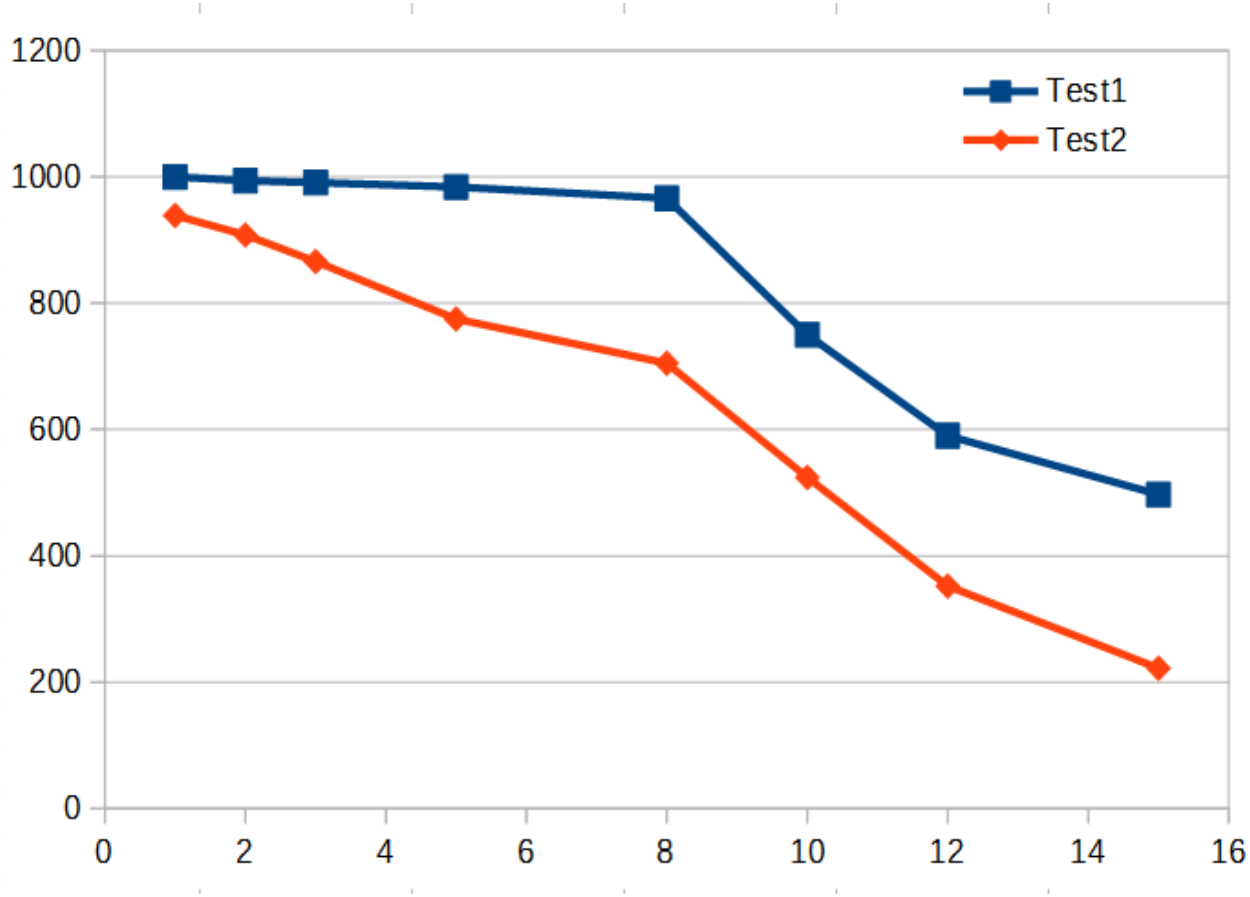




A beszűrődő fény hatása

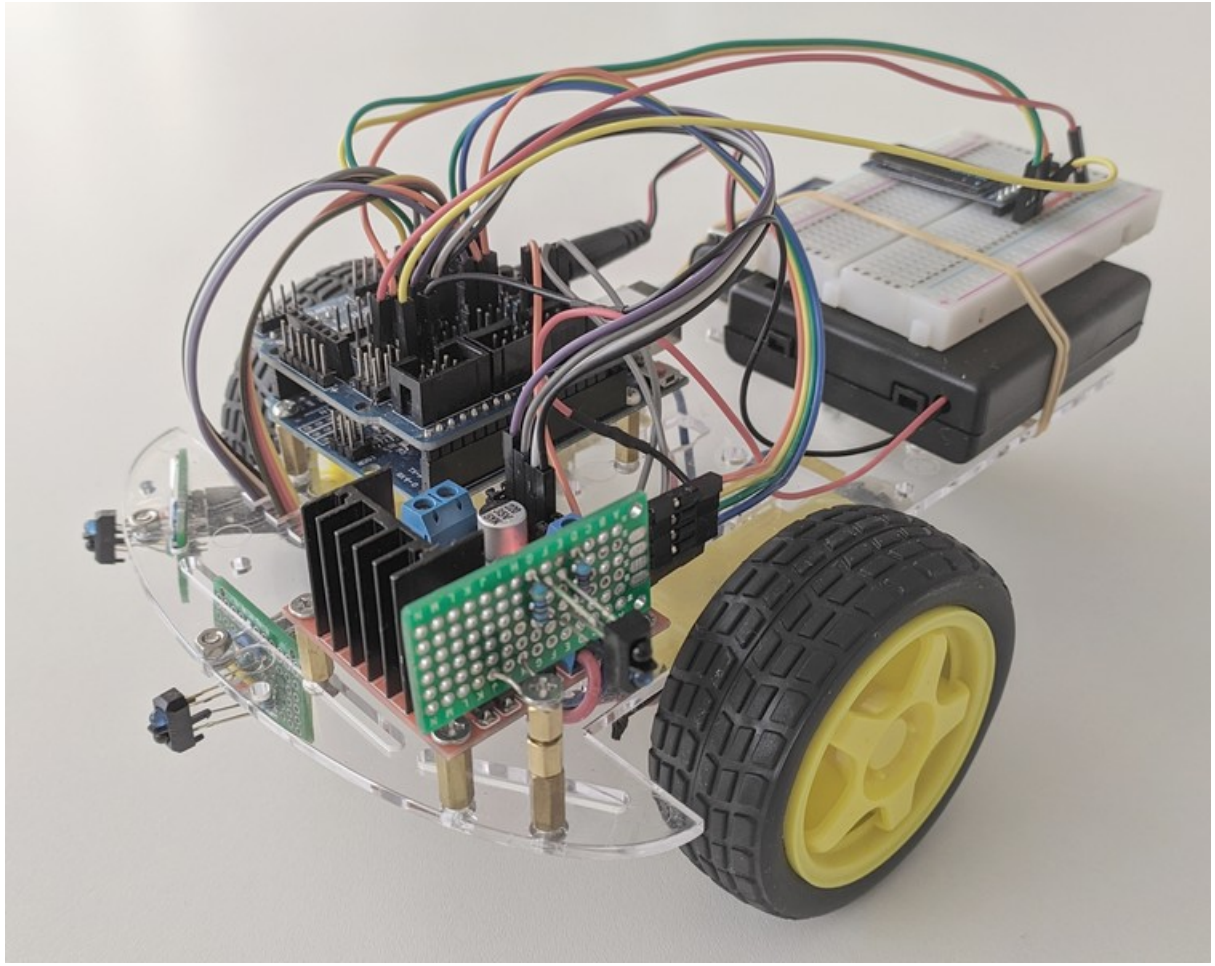
- Az alábbi táblázatban és ábrán összehasonlítottuk a beszűrődő fénnyel együtt mért (Test1) és a LED ki/bekapcsolásával a beszűrődő fény hatását korrigáló módszer (Test2) eredményét (a használt programokat az előző előadásban ismertettük).

Távolság [cm]	Test1	Test2
1	1000	939
2	994	908
3	991	866
5	984	775
8	966	705
10	750	524
12	590	352
15	497	222



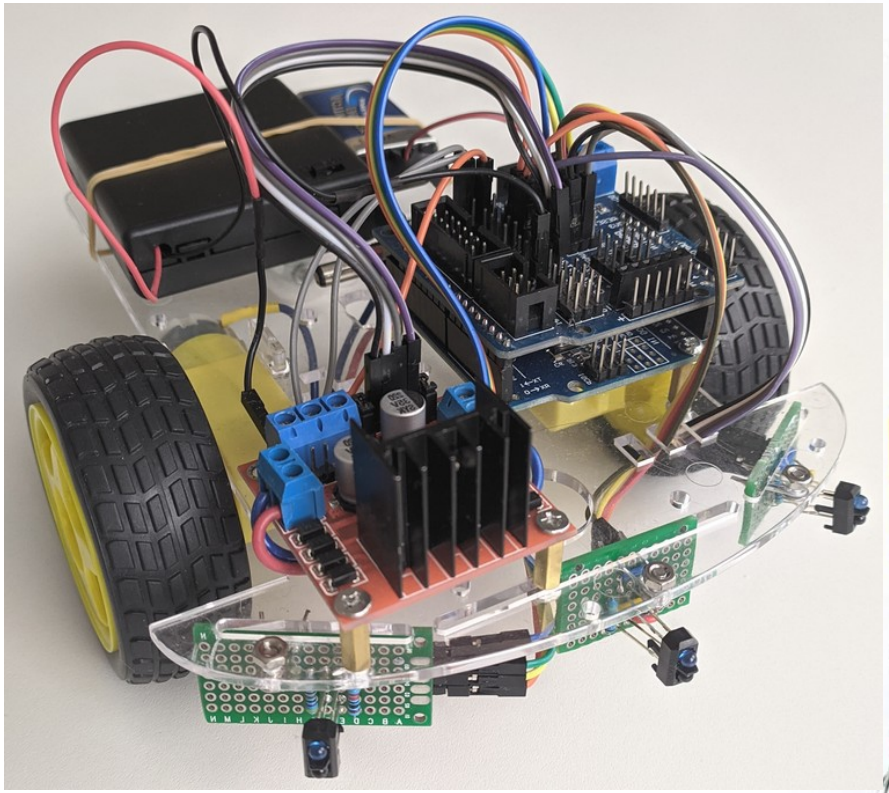
Falkövető robot

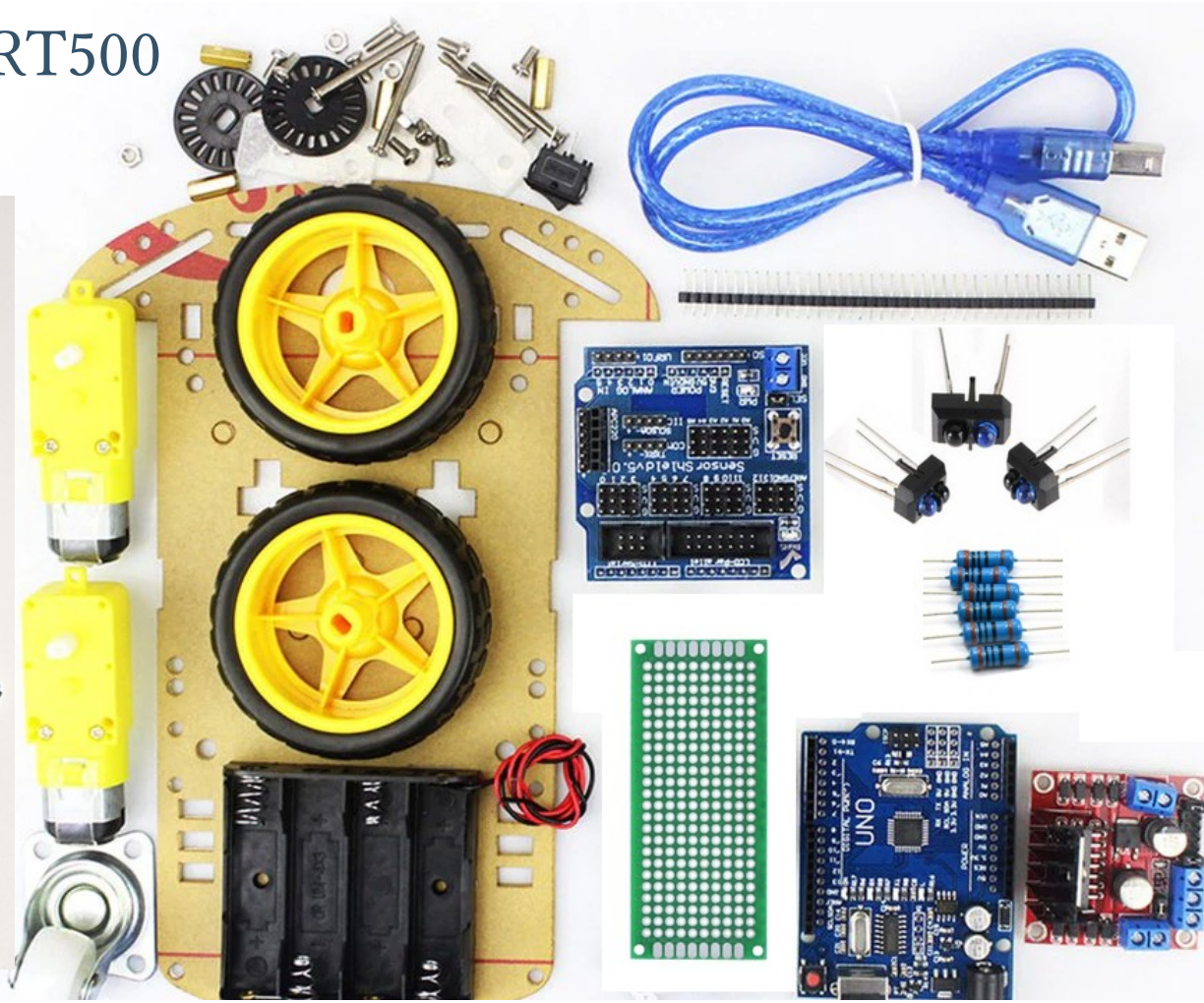
- Két kerék meghajtású (2WD) robotalváz
- Két, rögzített irányú, TCRT5000 szenzor (a harmadikat most nem használjuk)
- A reflexiós optikai szenzorok jó fényvisszaverő képességű anyagok esetén 1-30 cm távolságból észlelik az akadályokat
- A távolságtartásra és az észlelt akadályok kikerülésére alkalmas stratégiát kell választani



A hozzávalók

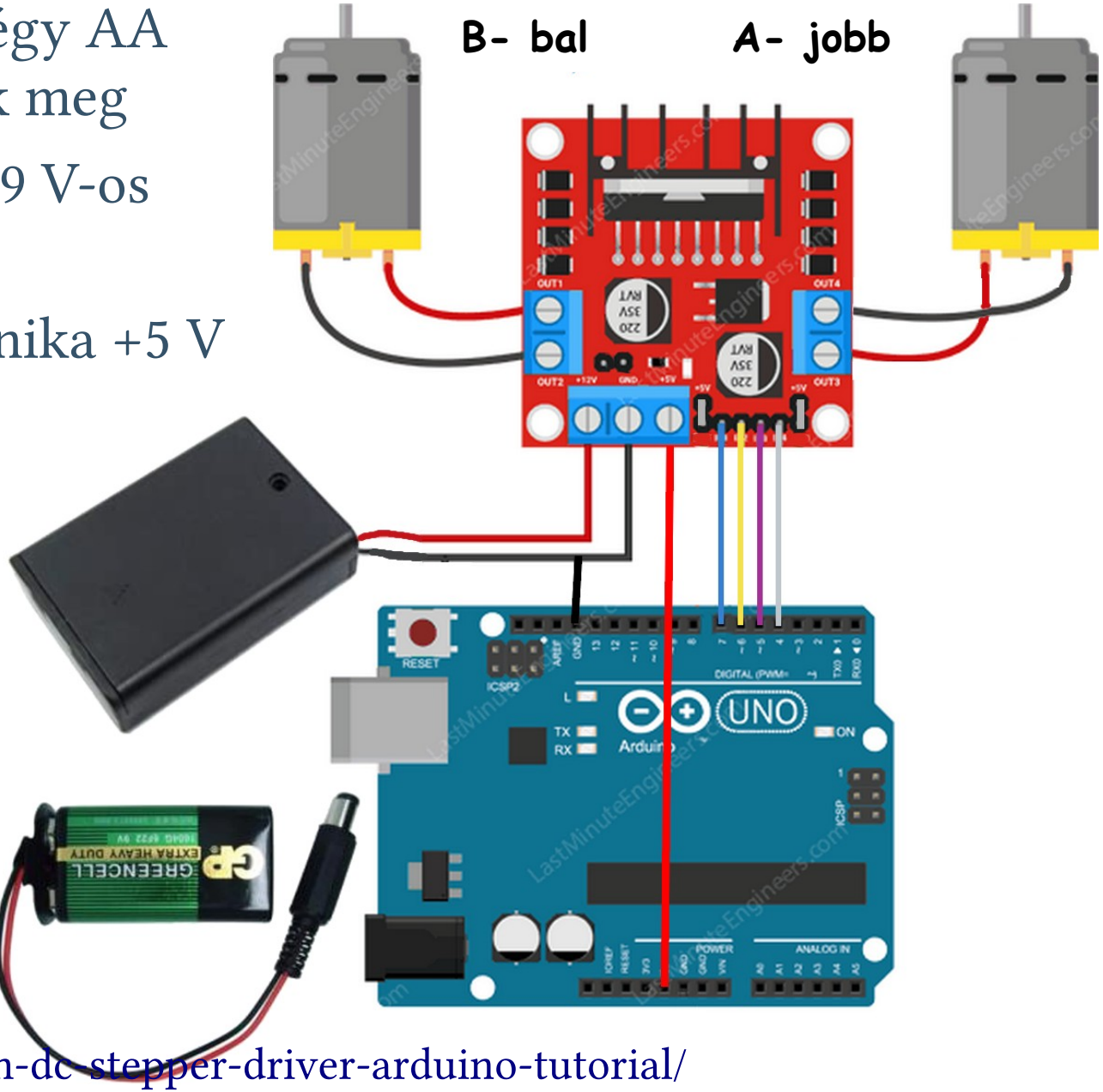
- Az előző előadásban már bemutatott 2WD robot alvázat most is egy *Arduino UNO* kártya, egy **L298N** motorvezérlő panel és egy szenzor fedlap segítségével hajtjuk meg
- Az érzékelő most 3 db TCRT500 szenzor lesz





A motorvezérlés kapcsolási vázlata

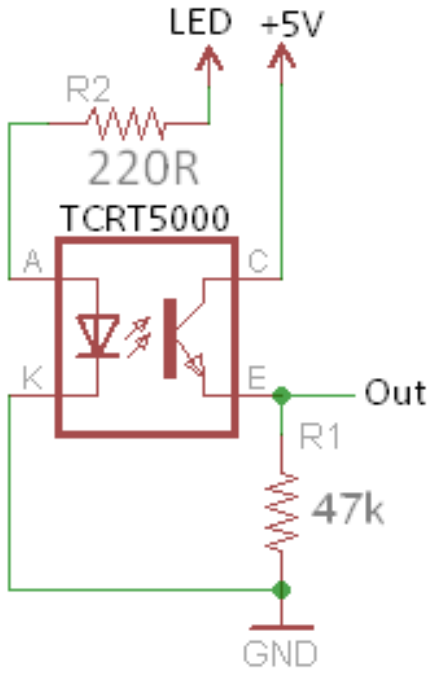
- A motorok táplálását négy AA ceruzaelemmel oldottuk meg
- Az Arduino kártya egy 9 V-os elemről kap táplálást és
- A motorvezérlő elektronika +5 V feszültségét az Arduino kártya biztosítja
- A motorok tükör-szimmetrikus bekötése hardveresen biztosítja az ellentétes irányú forgást a bal és jobb motornál

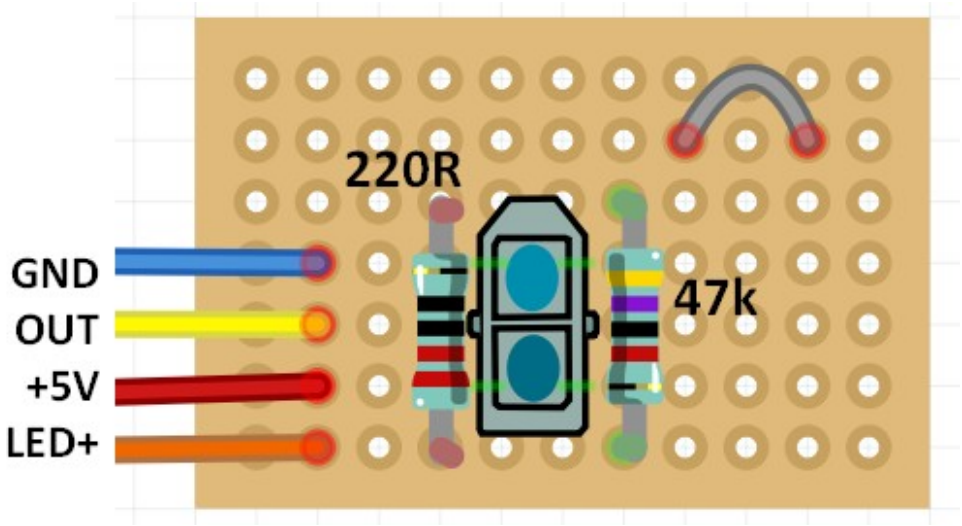


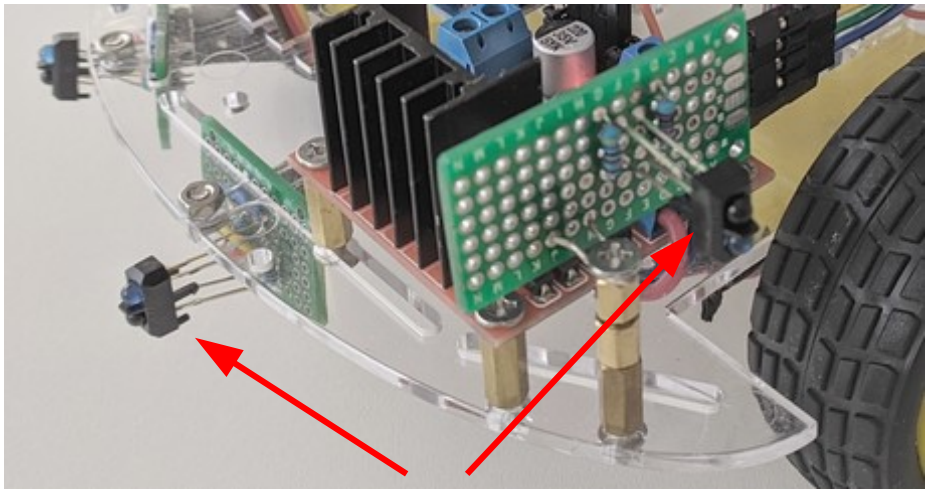
- Felhasznált forrás: lastminuteengineers.com/l298n-dc-stepper-driver-arduino-tutorial/

Az érzékelők kialakítása

- Barkácsoljunk két szenzort **TCRT5000** IR érzékelők felhasználásával és rögzítsük azokat a képen látható módon!
- A rögzítést egy, a panelbe forrasztott drót ”fül” (gémkapocs) és egy M3-as csavar segítségével oldottuk meg
- A kimeneteket az Arduino **A0** és **A1** bemeneteire, a LED vezérlést a **D9** és **D10** kimenetekre kötjük

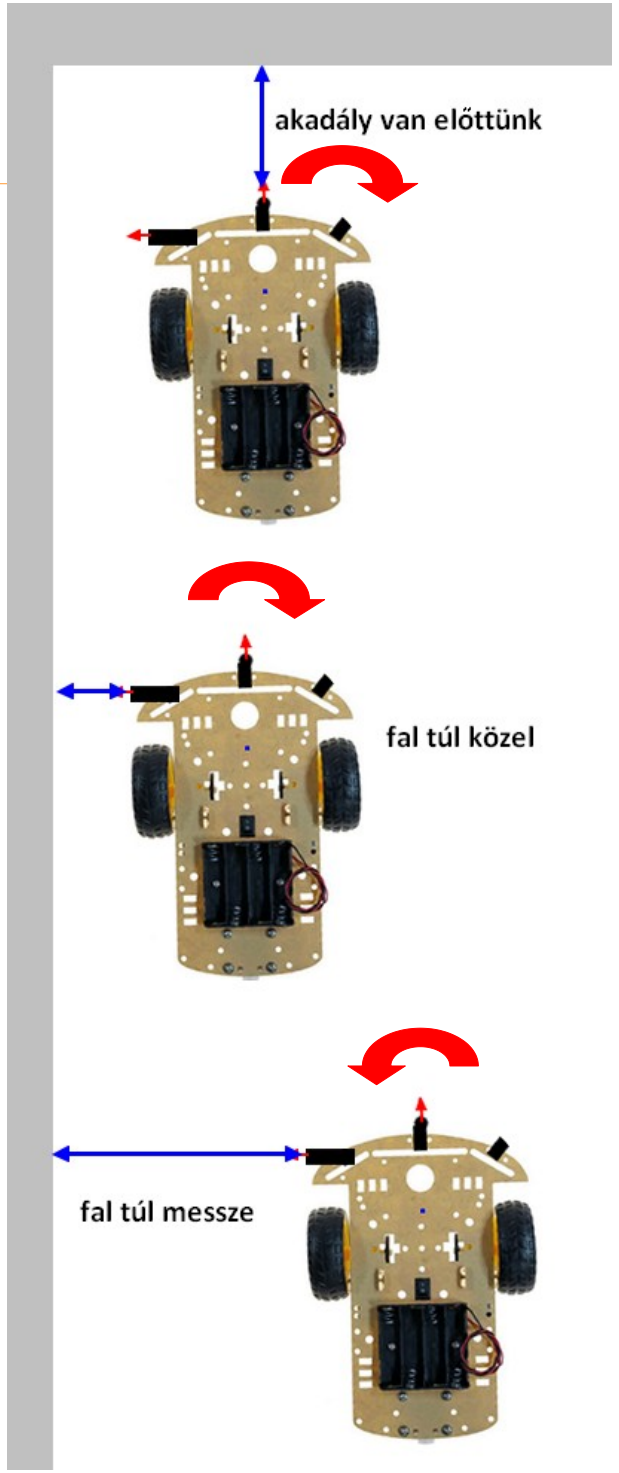




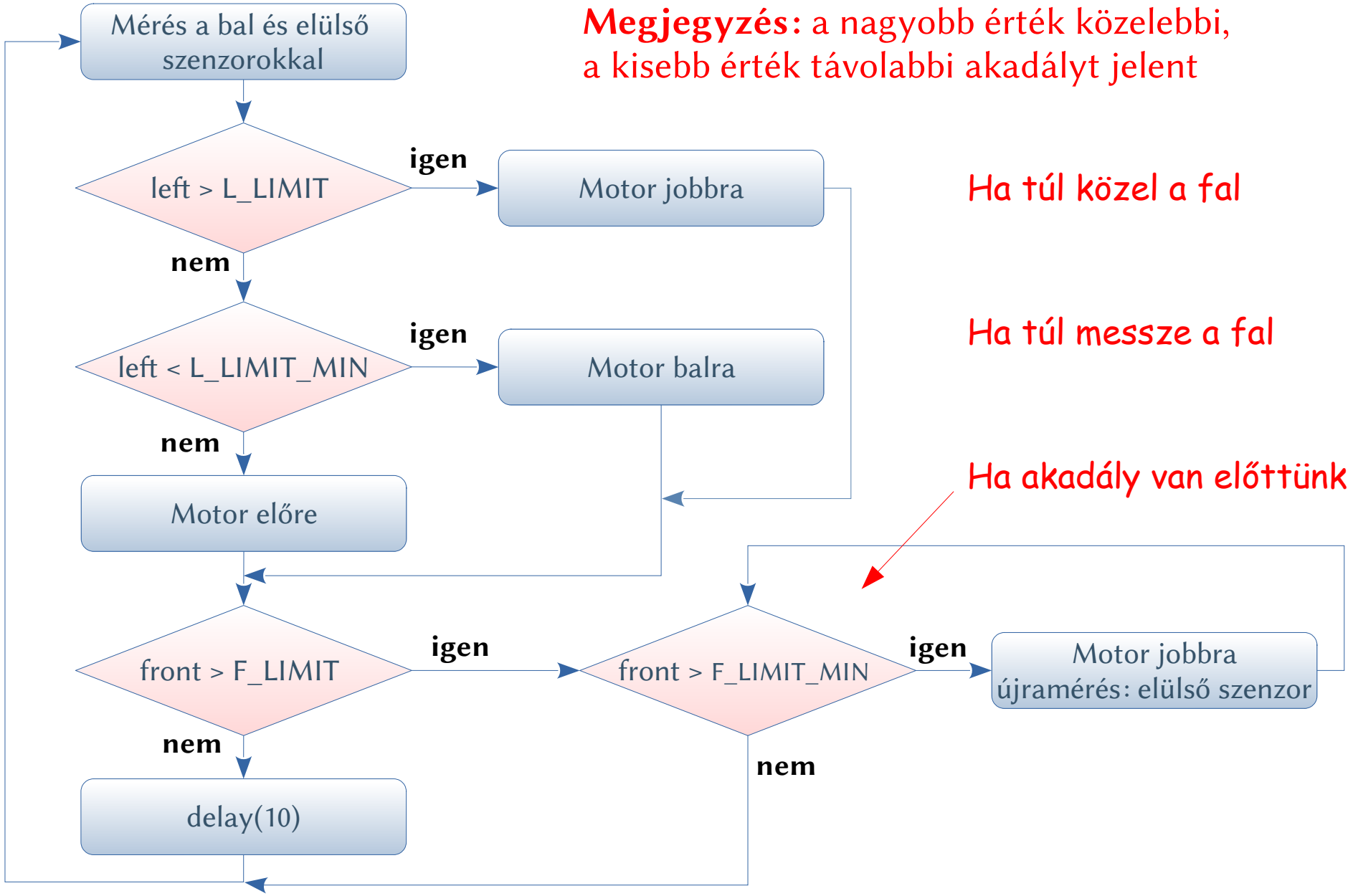


Akadálykikerülési stratégia

- A baloldali szenzorral a falat próbáljuk követni
- Ha túl messze van a fal, akkor balra fordulunk
- Ha túl közel van a fal, akkor jobbra fordulunk
- Ha akadály van előttünk, akkor addig fordulunk jobbra, amíg a front szenzor szabad utat nem jelez
- Minden más esetben előremenetbe kapcsolunk
- **Megjegyzések:**
  - A baloldali szenzor kiértékelése után a jobbra/balra „helyben fordulás” helyett a kanyarodás lenne a jobb megoldás, de ehhez lassúbb mozgású (nagyobb áttételű) motor kellene
  - A jobboldali szenzort most nem használjuk



A falkövető stratégia blokkvázlata



falkoveto_robot.ino 3/1.

- A program motorvezérléssel kapcsolatos része megegyezik az előző előadásban bemutatott robotvezérlő programmal
- Bekötés: **A0/D10** – bal szenzor, **A1/D9** – elülső szenzor, **D7/D6** – bal motor előre/hátra, **D5/D4** – jobb motor előre/hátra

```
#define B1A 7           // Bal motor előre
#define B1B 6           // Bal motor hátra
#define A1A 5           // Jobb motor előre
#define A1B 4           // Jobb motor hátra

#define L_sense A0      // Bal szenzor
#define L_LED   10
#define F_sense A1      // Elülső szenzor
#define F_LED    9
#define NMEAS   10      // Mérések száma

#define L_LIMIT_LOW 50 // Bal limit
#define L_LIMIT     80
#define M_LIMIT_LOW 30 // Elülső limit
#define M_LIMIT     50
```

```
100
150
 80
100
```

Az első próbánál ezeket a számokat használtuk

```
void setup() {
  analogReference(DEFAULT);

  //-- A TCRT5000 LED vezérlő kivezetések
  pinMode(L_LED, OUTPUT);   // bal
  pinMode(F_LED, OUTPUT);   // középső

  //-- Motorvezérlő lábak konfigurálása
  pinMode(B1A, OUTPUT);
  pinMode(B1B, OUTPUT);
  pinMode(A1A, OUTPUT);
  pinMode(A1B, OUTPUT);
  digitalWrite(B1A, LOW);
  digitalWrite(B1B, LOW);
  digitalWrite(A1A, LOW);
  digitalWrite(A1B, LOW);
}
```

falkoveto_robot.ino 3/2.

```
uint32_t meres(int p_sense, int p_LED) {
  uint32_t sum0, sum1;
  digitalWrite(p_LED, LOW);
  sum0 = 0;
  for (int i = 0; i < NMEAS; i++) {
    sum0 += analogRead(p_sense);
  }
  digitalWrite(p_LED, HIGH);
  sum1 = 0;
  for (int i = 0; i < NMEAS; i++) {
    sum1 += analogRead(p_sense);
  }
  if (sum0 > sum1) sum0 = sum1;
  return ((sum1 - sum0) / NMEAS);
}

void motor_forward() {
  digitalWrite(B1A, HIGH);
  digitalWrite(B1B, LOW);
  digitalWrite(A1A, HIGH);
  digitalWrite(A1B, LOW);
}
```

```
void motor_reverse() {
  digitalWrite(B1A, LOW);
  digitalWrite(B1B, HIGH);
  digitalWrite(A1A, LOW);
  digitalWrite(A1B, HIGH);
}

void motor_left() {
  digitalWrite(B1A, LOW);
  digitalWrite(B1B, HIGH);
  digitalWrite(A1A, HIGH);
  digitalWrite(A1B, LOW);
}

void motor_right() {
  digitalWrite(B1A, HIGH);
  digitalWrite(B1B, LOW);
  digitalWrite(A1A, LOW);
  digitalWrite(A1B, HIGH);
}

void motor_stop() {
  digitalWrite(B1A, LOW);
  digitalWrite(B1B, LOW);
  digitalWrite(A1A, LOW);
  digitalWrite(A1B, LOW);
}
```

falkoveto_robot.ino 3/3.

- A **loop()** függvény valósítja meg a falkövető stratégiát

```
void loop() {
  motor_forward();
  uint32_t left = meres(L_sense, L_LED);
  uint32_t front = meres(F_sense, F_LED);
  //--- ha közel a fal -------
  if (left > L_LIMIT) {
    motor_right();
  }
  //--- ha távol a fal -------
  else if (left < L_LIMIT_LOW) {
    motor_left();
  }
  else {
    motor_forward();
  }
  //--- ha akadály van elől -------
  if (front > F_LIMIT) {
    while (kozep > F_LIMIT_LOW) {
      front = meres(F_sense, F_LED);
      motor_right();
    }
  } else { delay(10); }
}
```

Gondolatkísérlet: Mit csinálna a robotunk egy ilyen labirintusban?

falkoveto_robot.ino: tapasztalatok 1.

- Az első kísérletnél túl közeli határokat szabtunk (gyakori ütközés)

Hogyan javíthatunk a robot viselkedésén?

- Csökkentsük a zavaró háttérfényt (beszűrődő napsütés, izzólámpa)!
- Optimalizáljuk és kalibráljuk az `L_LIMIT`, `L_LIMIT_MIN`, `F_LIMIT` és `F_LIMIT_MIN` paraméterek értékét!
  - Ehhez készítettünk egy új programot
  - Kiegészítettük a kapcsolást egy kijelzővel és egy nyomógombbal
  - A kijelző mutatja a soron következő paraméter nevét és a szenzor pillanatnyi értékét, így az akadálytól/faltól a kívánt távolságra elhelyezett robot kijelzőjéről közvetlenül leolvashatjuk a paraméter értékét és a nyomógombbal rögzíthetjük azt
- Kijelző gyanánt egy **I2C** buszra köthető, **SSD1306** vezérlővel ellátott **OLED** (organic LED) kijelzőt használunk – ennek megismeréséhez azonban tennünk kell egy kis kitérőt

Hol találhatunk bővebb információt?

- Itt most csak érintőlegesen foglalkozunk az **I2C** busz és az **OLED** kijelzők használatával, részletesebb információ a korábbi előadásokban található

Arduino témakörű előadások:

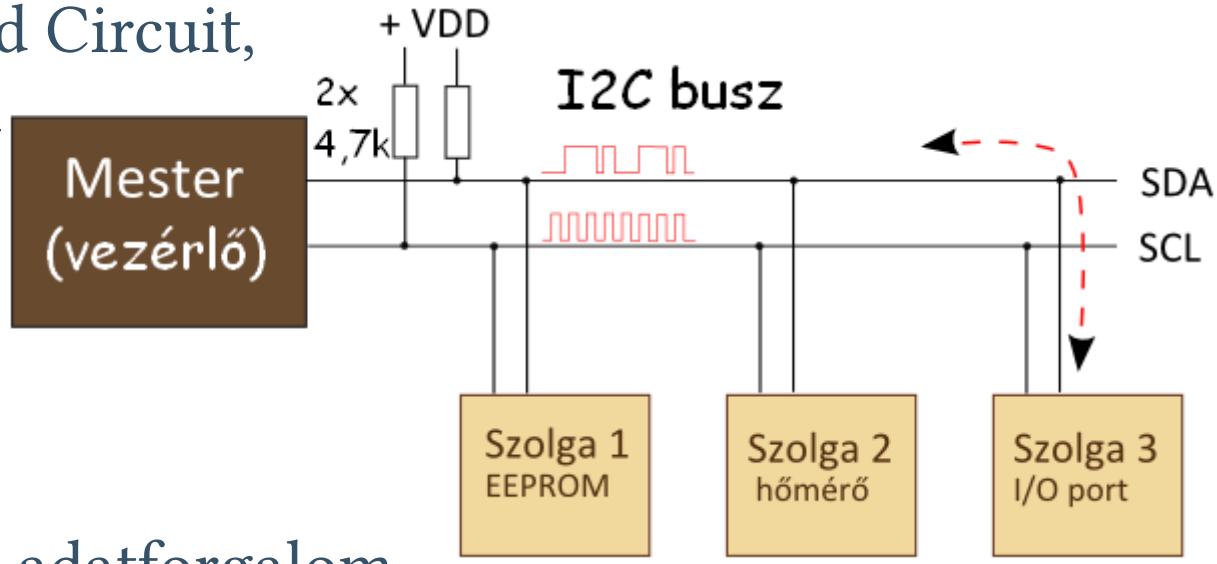
- Az I2C kommunikációs csatorna (2020. február 6.)
  előadásvázlat mintaprogramok
- Ultrahangos távolságmérés, OLED kijelzők (2020. február 6.)
  előadásvázlat mintaprogramok

STM32 mikrovezérlő témakörű előadások:

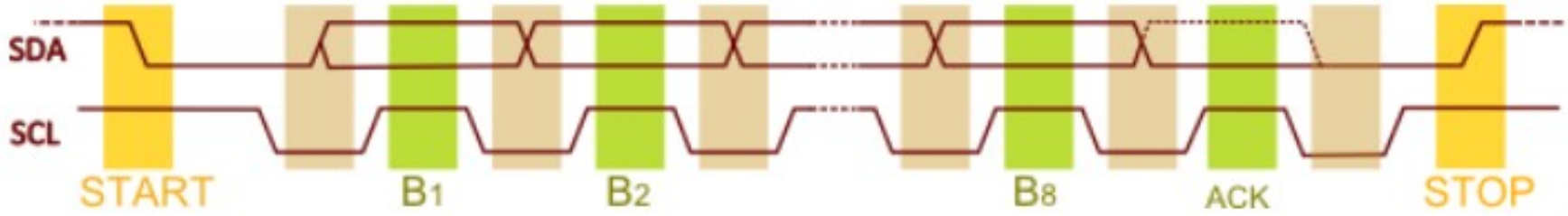
- Az I2C kommunikációs csatorna – 1. rész (2021. február 4.)
  előadásvázlat mintaprogramok video
- Az I2C kommunikációs csatorna – 2. rész (2021. február 18.)
  előadásvázlat mintaprogramok video

Az I2C busz

- I2C = IIC, Inter Integrated Circuit, azaz integrált áramkörök közötti kommunikáció
- A kétvezetékes buszt a PHILIPS fejlesztette ki 1995-ben



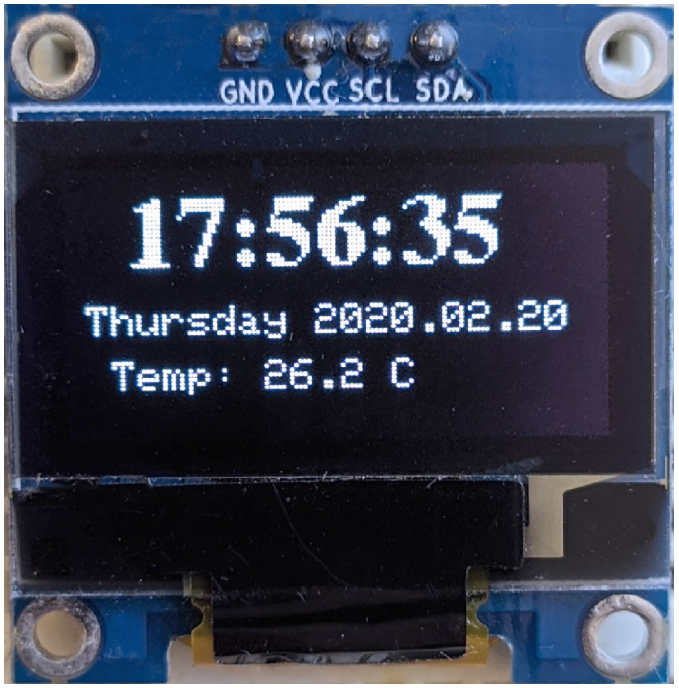
- Soros, 8-bit-es, kétirányú adatforgalom, sebessége 100 kbit/s (standard mode), vagy 400 kbit/s (fast mode)
- Az eszközök egy 7-bites címmel címezhetők (némi trükközéssel ez 10 bitre bővíthető)



Az I2C kommunikáció idődiagramja (egy bájt kiküldése)

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

OLED I2C kijelző SSD1306 vezérlővel

Jellemzők:

- **OLED** technológia
- **128x32** képpont
- 0,91” (2,3 cm) képátló
- I2C illesztő
- 2.5V-5.5V tápfeszültség
- **SSD1306** vezérlő
- Monokróm
- Grafikus megjelenítés
- Inverz mód
- Görgetés több irányba
- Dokumentáció: Solomon Systech SSD1306 adatlap

Kompatibilis az SSD1306 vezérlőjű 0.96” képátlójú, 128x64 felbontású kijelzővel, de az inicializálásban van néhány különbség

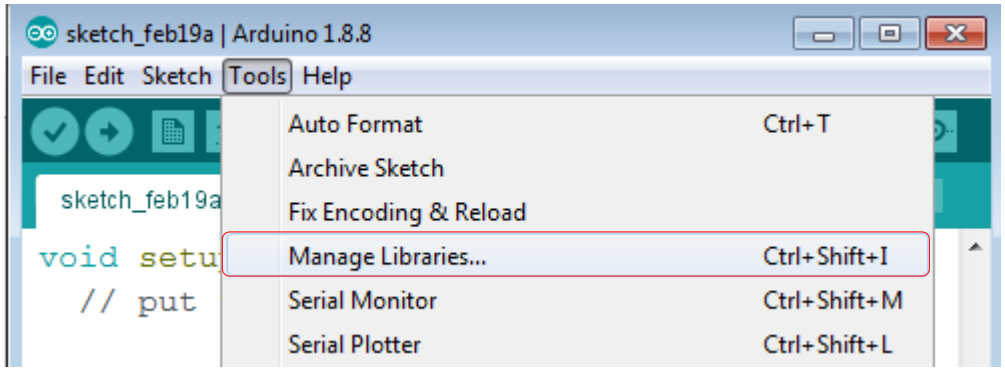
SSD1306 programkönyvtárak

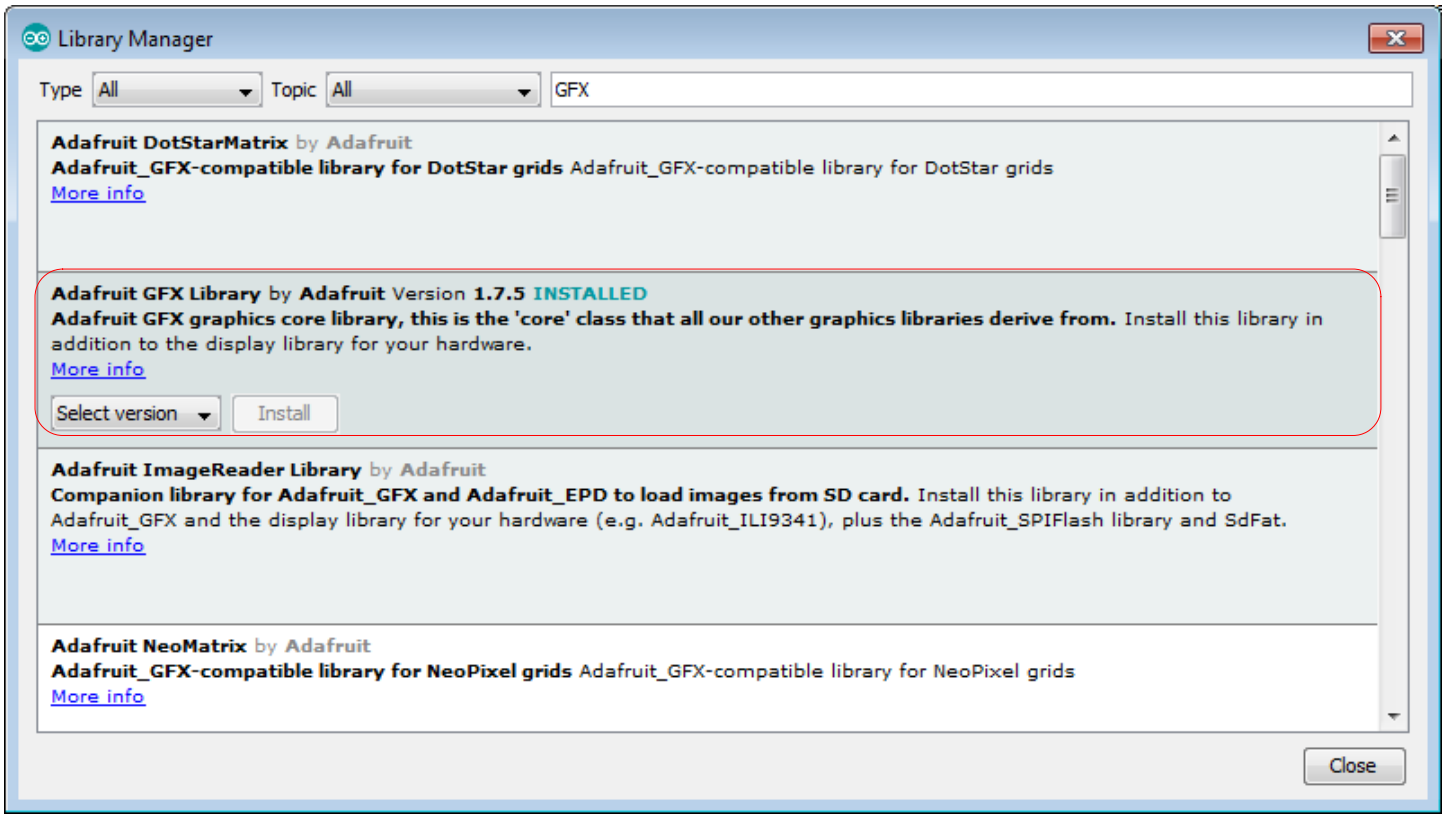
- A kijelző elvileg saját programmal is vezérelhető az adatlap alapján, de kényelmesebb a meglevő programkönyvtárak közül választani
- **Adafruit_SSD1306:** C++ nyelven írt osztály, amely az **Adafruit_GFX** és a **Print** osztályok leszármazottja, azok metódusait is örökli. Az **Adafruit_GFX** könyvtárat 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

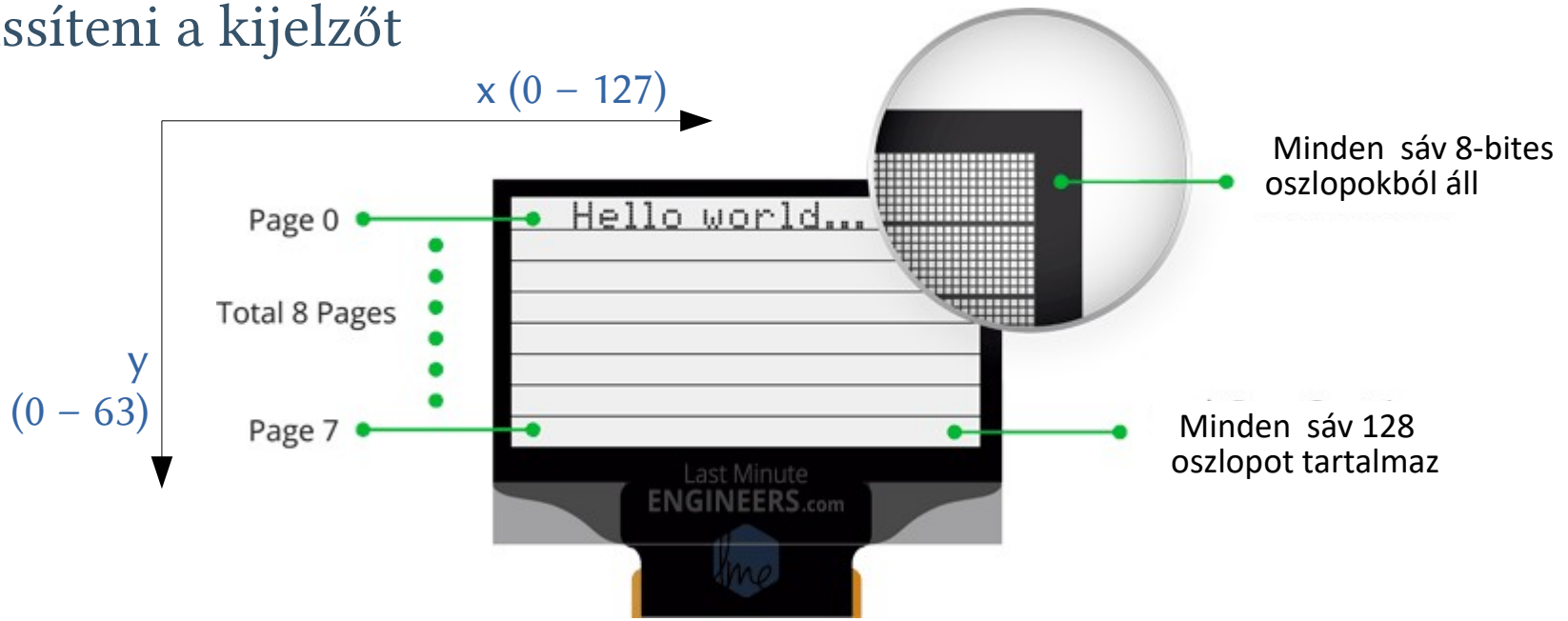
- Nyissuk meg a **Tools** menüben a **Manage Libraries** menüpontot!
- Keressük meg és telepítsük az **Adafruit GFX** könyvtárat!
- Majd a keresősávba **ssd1306**-ot írva keressük meg és telepítsük az **Adafruit ssd1306** könyvtárat is!





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 sávra van felosztva, amelyekben egy oszlop 8 képpontból áll, amelyeket egy bájt bitjei vezérelnek
- Minden sáv (page) 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_GFX.h>
#include <Adafruit_SSD1306.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();
  ...
}
```

Egyszerű szöveg kiírása

- 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 6x8 pont **setTextSize(2)** után 12x16 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);
```

Számok kiírása

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

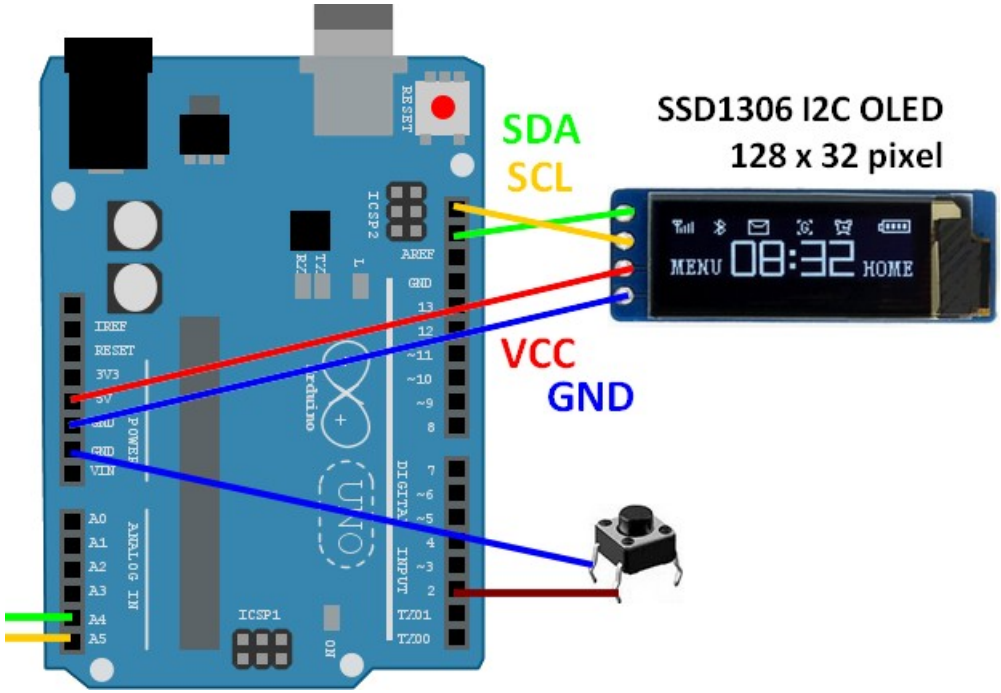

```
// Display Numbers
display.clearDisplay();
display.setTextSize(1);
display.setCursor(0,28);
display.println(123456789);  //Szám kiírása
display.display();
delay(2000);
```

- **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);
```

ssd1306_robothangolo.ino

- A robotunkat egészítsük ki egy **SSD1306** vezérlőjű **I2C** kijelzővel
- Az **I2C** busz az **AVREF** kivezetés mellett, vagy az **A4** (SDA) és **A5** (SCL) kivezetéseken érhető el
- Kössünk egy nyomógombot a **D2** bemenet és a **GND** közé! Ezzel a gombbal léptethetjük/rögzíthetjük majd a paramétereket



ssd1306_robothangolo.ino

```
#include <SPI.h>
#include <Wire.h>
#include <Adafruit_GFX.h>
#include <Adafruit_SSD1306.h>

#define SCREEN_WIDTH 128
#define SCREEN_HEIGHT 32
#define OLED_RESET     -1
#define SCREEN_ADDRESS 0x3C

Adafruit_SSD1306 display(SCREEN_WIDTH,
SCREEN_HEIGHT, &Wire, OLED_RESET);

#define PUSH2 2     // Nyomógomb
#define L_sense A0
#define L_LED   10
#define F_sense A1
#define F_LED    9
#define NMEAS   50

boolean waitforpress = true;
uint32_t l_limit = 100;
uint32_t l_limit_min = 80;
uint32_t f_limit = 100;
uint32_t f_limit_min = 80;
```

```
uint32_t meres(int p_sense, int p_LED) {
  uint32_t sum0, sum1;
  digitalWrite(p_LED, LOW);
  sum0 = 0;
  for (int i = 0; i < NMEAS; i++) {
    sum0 += analogRead(p_sense);
  }
  digitalWrite(p_LED, HIGH);
  sum1 = 0;
  for (int i = 0; i < NMEAS; i++) {
    sum1 += analogRead(p_sense);
  }
  if (sum0 > sum1) sum0 = sum1;
  return ((sum1 - sum0) / NMEAS);
}
```

ssd1306_robothangolo.ino

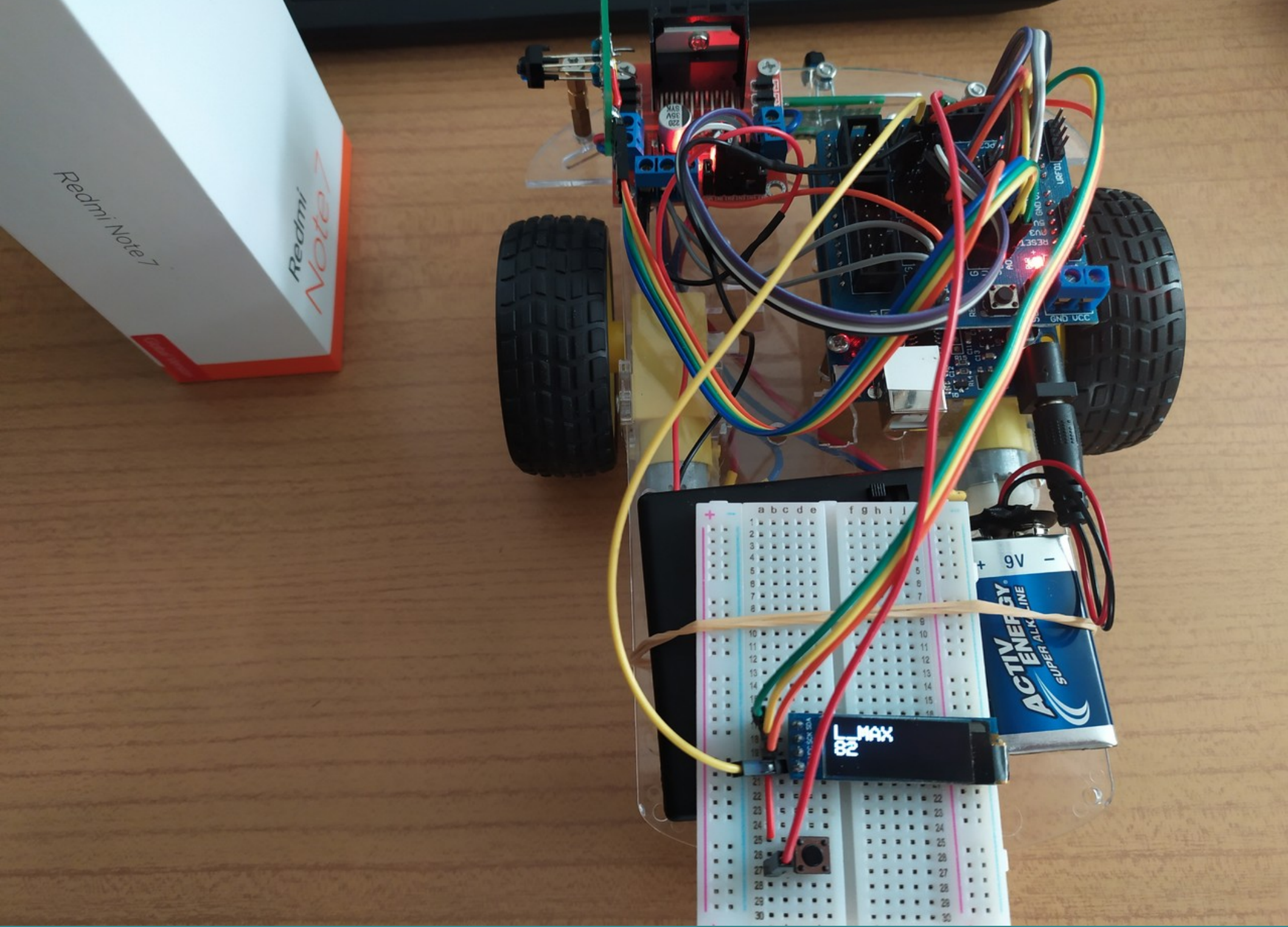
```
void setup() {
  Serial.begin(9600);
  display.begin(SSD1306_SWITCHCAPVCC, SCREEN_ADDRESS);
  display.display();
  pinMode(PUSH2, INPUT_PULLUP);
  pinMode(L_LED, OUTPUT);
  pinMode(F_LED, OUTPUT);
  delay(2000);
  l_limit = get_parameter("L_MAX", L_sense, L_LED);
  Serial.print("\r\nL_MAX = ");
  Serial.print(l_limit);
  l_limit_min = get_parameter("L_MIN", L_sense, L_LED);
  Serial.print("\r\nL_MIN = ");
  Serial.print(l_limit_min);
  f_limit = get_parameter("F_MAX", F_sense, F_LED);
  Serial.print("\r\nF_MAX = ");
  Serial.print(f_limit);
  f_limit_min = get_parameter("F_MIN", F_sense, F_LED);
  Serial.print("\r\nF_MIN = ");
  Serial.print(f_limit_min);
}

void loop() {

}
```

ssd1306_robothangolo.ino

```
uint32_t get_parameter(char* txt, int p_sense, int p_led) {
  uint32_t mypar;
  while (waitforpress) {            // Ha lenyomásra várunk és
    mypar = meres(p_sense, p_led);
    display.clearDisplay();
    display.setTextSize(2);             // Normal 2:1 pixel scale
    display.setTextColor(SSD1306_WHITE);        // Draw white text
    display.setCursor(0, 0);            // Start at top-left corner
    display.println(txt);
    display.setCursor(0, 16);            // Start at middle-left position
    display.println(mypar);
    display.display();
    if (!digitalRead(PUSH2)) {   // Ha lenyomás történt...
      waitforpress = false;      // Következő stáció: felengedésre várunk
    }
  }
  delay(20);                     // Pergésmentesítő késleltetés
  while (!waitforpress) {
    if (digitalRead(PUSH2)) {    // Ha felengedést észlelünk...
      waitforpress = true;       // Következő stáció: lenyomásra várunk
    }
  }
  delay(20);                     // Pergésmentesítő késleltetés
  return mypar;
}
```


Redmi Note 7
9V
GND VCC

falkoveto_robot.ino: tapasztalatok 2.

- A 2. kísérletnél lazább határokat szabtunk (nagyobb távolság)

falkoveto_robot_deluxe.ino 3/1.

```
#include <Adafruit_SSD1306.h>

#define B1A 7           // Bal motor előre
#define B1B 6           // Bal motor hátra
#define A1A 5           // Jobb motor előre
#define A1B 4           // Jobb motor hátra

#define PUSH2 2         // Nyomógomb
#define L_sense A0
#define L_LED   10
#define F_sense A1
#define F_LED    9
#define NMEAS   10

#define SCREEN_WIDTH 128    // OLED kijelző szélessége képpontokban
#define SCREEN_HEIGHT 32    // OLED kijelző magassága képpontokban
#define OLED_RESET     -1   // -1 jelentése: nincs RESET kivezetés
#define SCREEN_ADDRESS 0x3C // A kijelző I2C címe (jobbra igazítva)
Adafruit_SSD1306 display(SCREEN_WIDTH, SCREEN_HEIGHT, &Wire, OLED_RESET);

boolean waitforpress = true;
uint16_t l_limit = 80;
uint16_t l_limit_min = 50;
uint16_t f_limit = 50;
uint16_t f_limit_min = 30;
uint16_t chksum;
```

falkoveto_robot_deluxe.ino 3/2.

```
void setup() {
  analogReference(DEFAULT);
  pinMode(PUSH2, INPUT_PULLUP); pinMode(L_LED, OUTPUT); pinMode(F_LED, OUTPUT);
  pinMode(B1A, OUTPUT); pinMode(B1B, OUTPUT); pinMode(A1A, OUTPUT); pinMode(A1B, OUTPUT);
  digitalWrite(B1A, LOW); digitalWrite(B1B, LOW);
  digitalWrite(A1A, LOW); digitalWrite(A1B, LOW);
  display.begin(SSD1306_SWITCHCAPVCC, SCREEN_ADDRESS);
  display.display();
  if (!digitalRead(PUSH2)) {
    while (!digitalRead(PUSH2)); delay(20);
    l_limit = get_parameter("L_MAX", L_sense, L_LED);
    l_limit_min = get_parameter("L_MIN", L_sense, L_LED);
    f_limit = get_parameter("F_MAX", F_sense, F_LED);
    f_limit_min = get_parameter("F_MIN", F_sense, F_LED);
  }
  display.setTextSize(2); display.setTextColor(SSD1306_WHITE); display.clearDisplay();
  display.setCursor(0, 0);
  display.print(l_limit);
  display.setCursor(64, 0);
  display.print(f_limit);
  display.setCursor(0, 16);
  display.print(l_limit_min);
  display.setCursor(64, 16);
  display.print(f_limit_min);
  display.display();
  delay(5000);
}
```

falkoveto_robot_deluxe.ino 3/3.

- A **meres()**, **get_parameter()** és a motorvezérlés megegyezik a korábbiakkal

```
void loop() {
  motor_forward();
  uint32_t left = meres(L_sense, L_LED);
  uint32_t front = meres(F_sense, F_LED);

  if (left > l_limit) {
    motor_right();
  }
  else if (left < l_limit_min) {
    motor_left();
  }
  else {
    motor_forward();
  }

  if (front > f_limit) {
    while (front > f_limit_min) {
      front = meres(F_sense, F_LED);
      motor_right();
    }
  } else {
    delay(10);
  }
}
```

Ebben a függvényben csak átnevezések történtek

Az Arduino nano kártya kivezetései