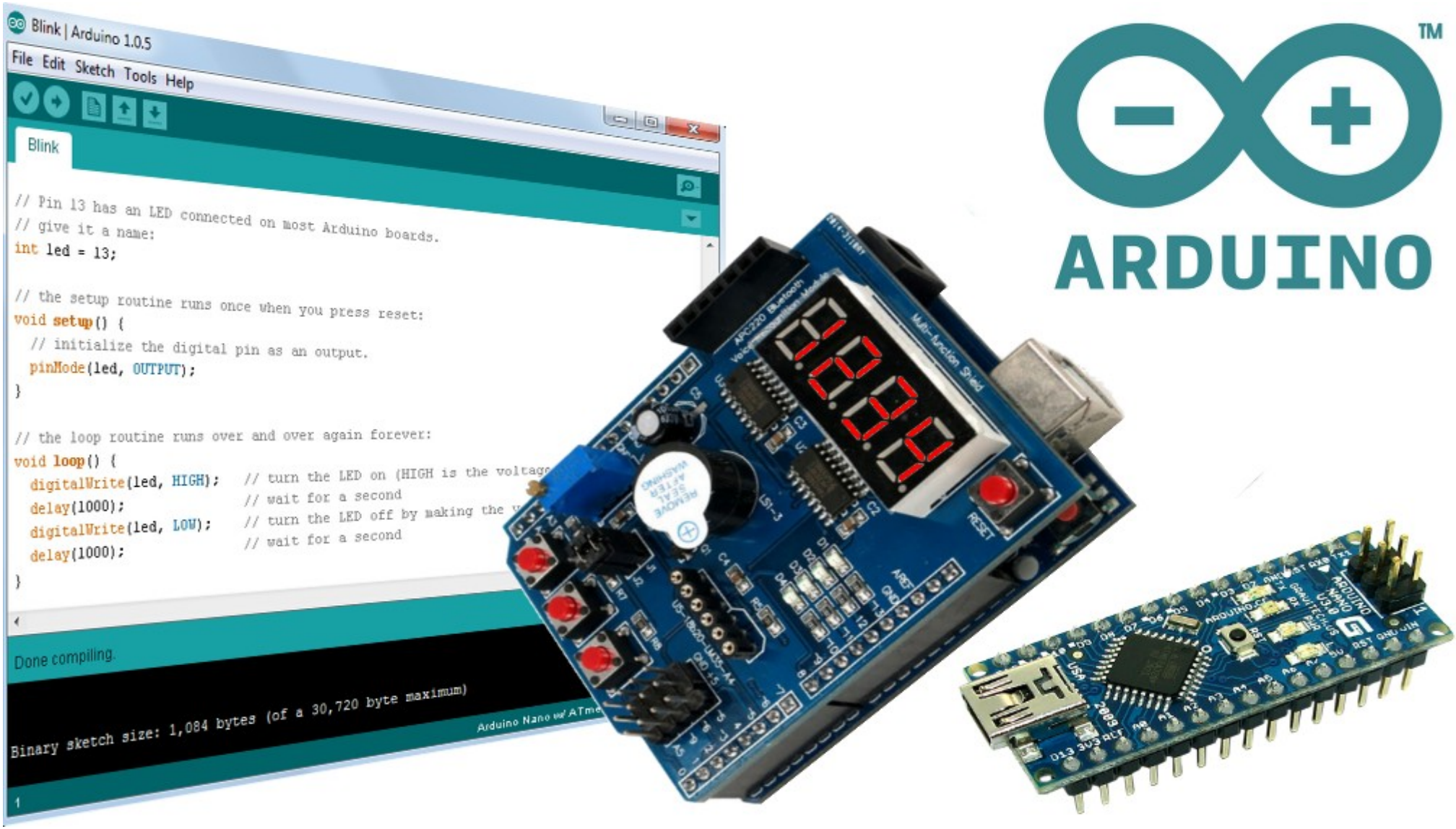
Arduino tanfolyam kezdőknek és haladóknak



9. Motorvezérlés, bevezetés a robotikába

Felhasznált és ajánlott irodalom

- Dovgy, Baryakhtar, Loktev: **Fizika 11. osztályosoknak**
- FizikusRobotBlog: **Akadálykikerülő Robot v1.0**
- FizikusRobotBlog: **Akadálykikerülő Robot v2.0**
- Sai Tat Sat Mishra: **How to make an obstacle avoiding robot..**.
- Abhiemanyu Pandit: **Obstacle Avoiding Robot using Arduino ...**

Korábbi előadások a témakörben

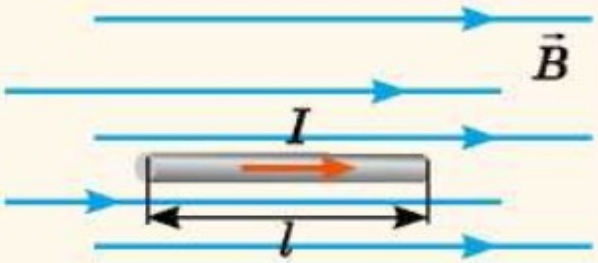
- Kisteljesítményű egyenáramú motorok vezérlése (2015. december 17.)
  előadásvázlat mintaprogramok
- Robotvezérlés sebességmérő szenzorral (2019. január 31.)
  előadásvázlat mintaprogramok
- Robotvezérlés WiFi kapcsolaton keresztül 1. rész (2019. február 14.)
  előadásvázlat mintaprogramok
- Robotvezérlés WiFi kapcsolaton keresztül 2. rész (2019. február 28.)
  előadásvázlat mintaprogramok
- Servo vezérlés, Processing keretrendszer, ultrahangos "radar"
  (2020. március 5.) előadásvázlat mintaprogramok
- Optoérzékelők, motorvezérlés, bevezetés a robotikába (2020. március 19.)
  előadásvázlat mintaprogramok video

ESP8266

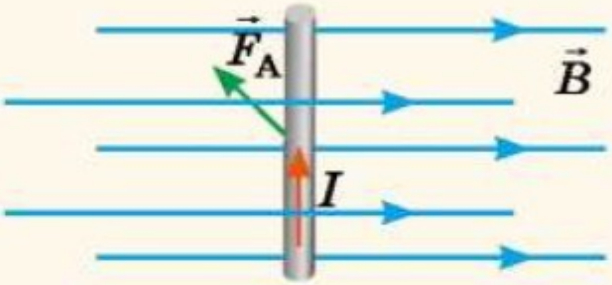
Ampére-féle erő

- Azt az erőt, amellyel a mágneses tér hat az áramjárta vezetőre, Ampére-féle erőnek nevezzük (Ampére 1820-as kísérletei nyomán)

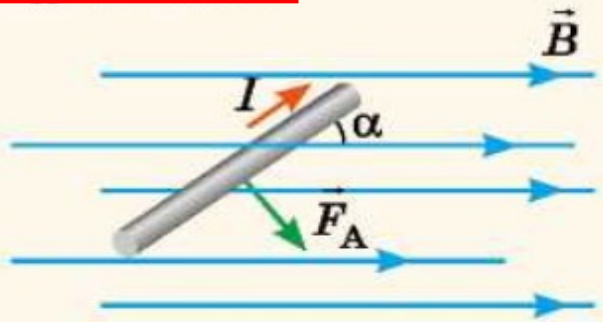
• A vezető párhuzamos az indukcióvektorokkal – a *mágneses tér nem hat a vezetőre*: $F_A=0$



• A vezető merőleges az indukcióvektorokra – az *Ampère-féle erő maximális értéket ér el*: $F_A=BIl$

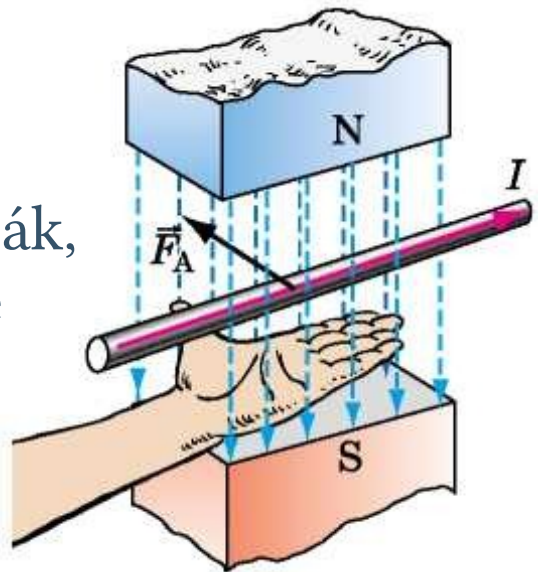


• Általános esetben az Ampère-féle erő a következő képlet segítségével határozható meg: $F_A=BIl\sin\alpha$



Forrás: uabooks.top/1044-11-ampre-fle-er.html

Az Ampére-féle erő iránya

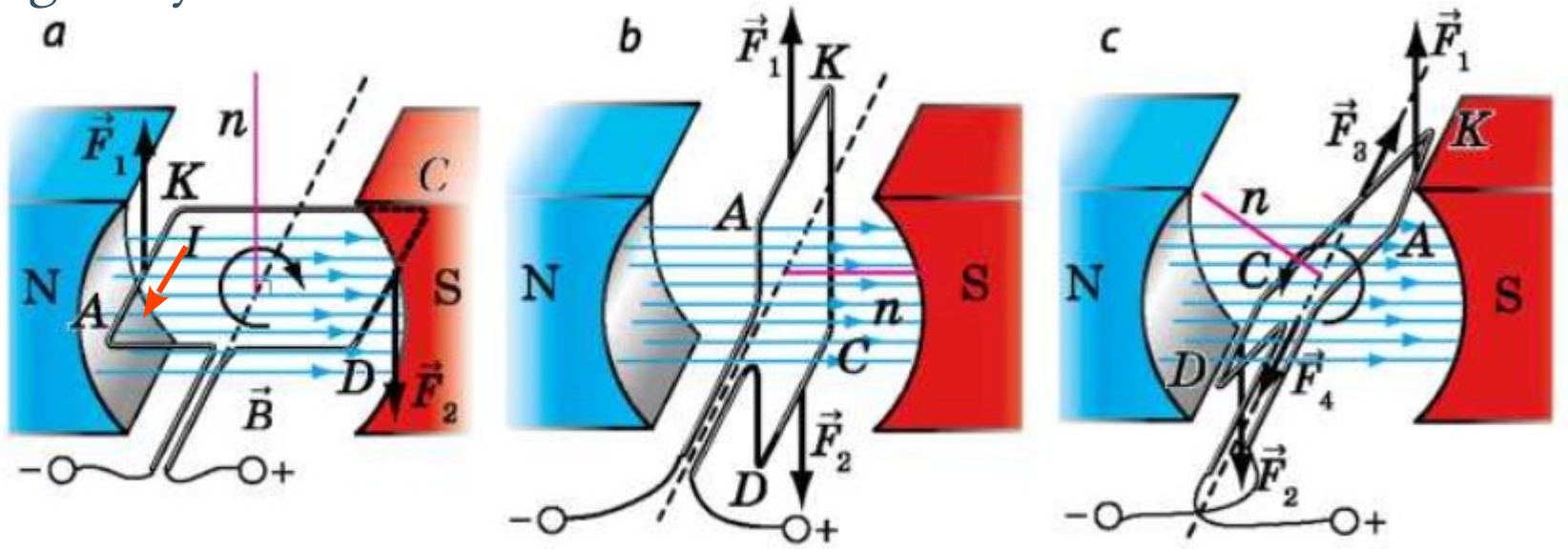


- Ha a mágneses tér indukcióvektorainak eredője a tenyerünkbe hatol, és ujjaink az áram irányát mutatják, akkor az oldalra kinyújtott hüvelykujjunk a vezetőre ható Ampére-féle erő irányát mutatja

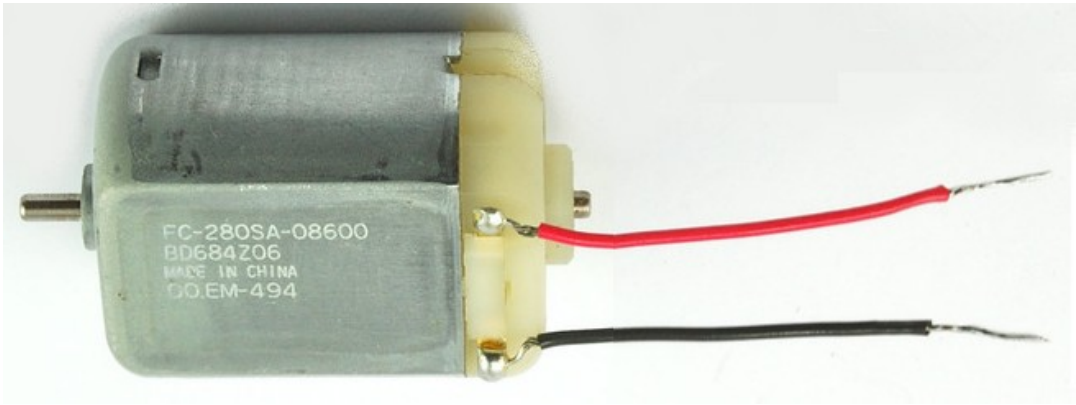
  a) Az áramvezető keretre óramutató járása szerinti irányú forgatónyomaték hat

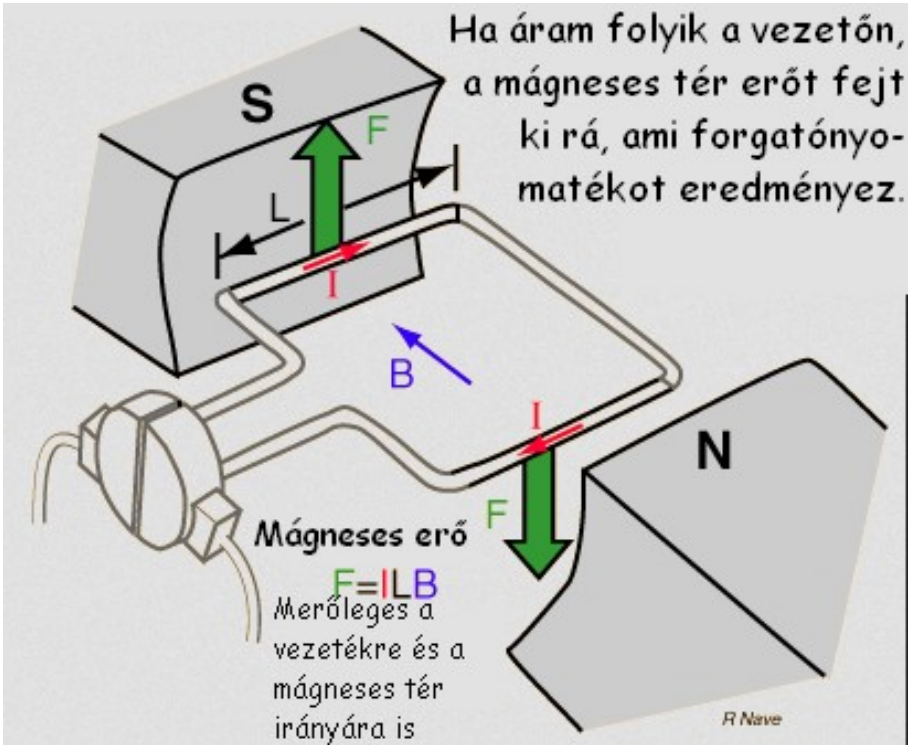
  b) Az áramvezető keretre nem hat forgatónyomaték

  c) Az áramvezető keretre óramutató járásával ellentétes irányú forgatónyomaték hat

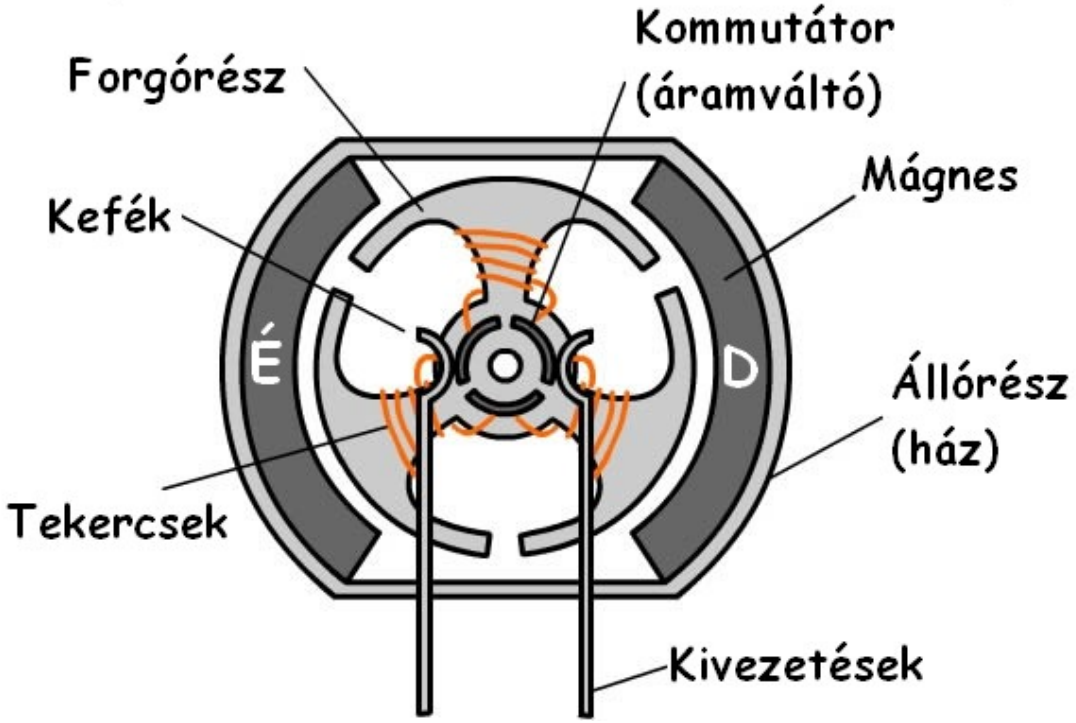


Kisteljesítményű DC motorok





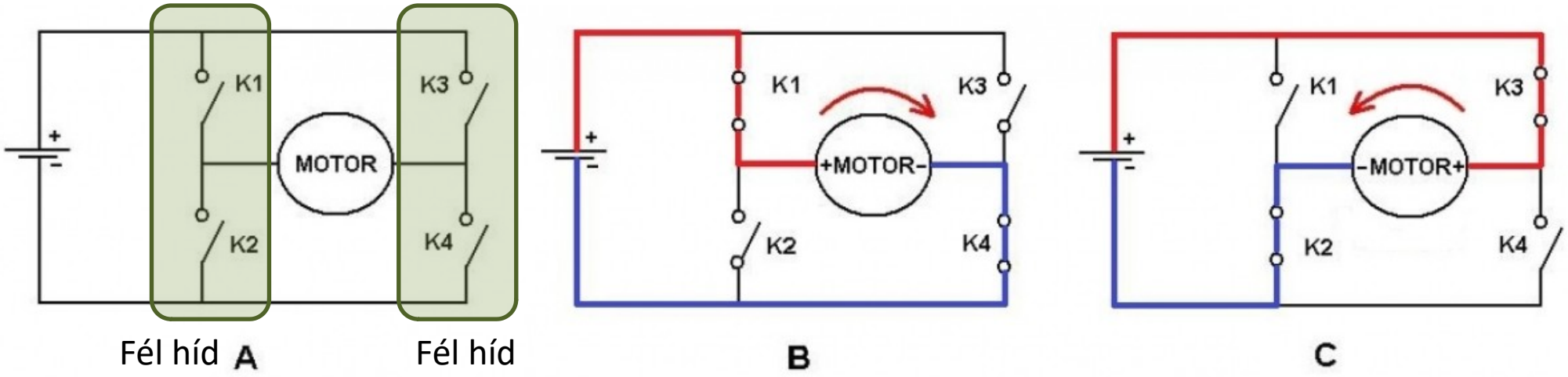
Tipikus kefés motor metszeti képe



Feszültség: 3 V – 6V
Áram: 0.4 A – 0.6 A
Forgásirány váltás: polaritásváltással

Motorvezérlés: H-híd

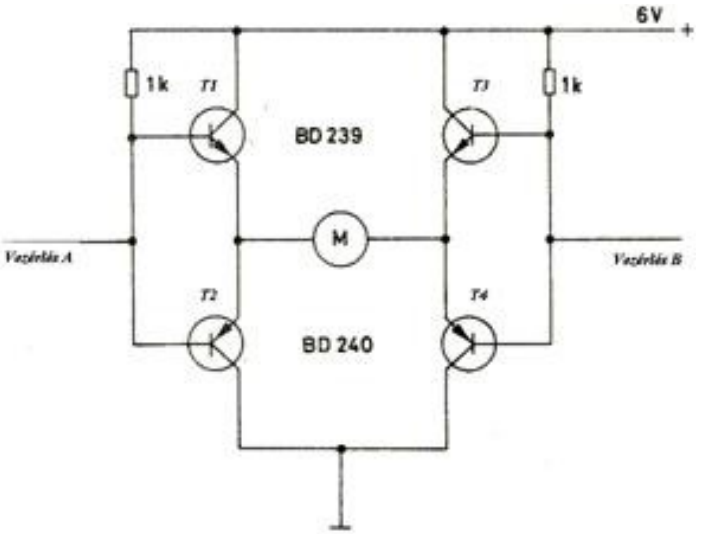
Ha a motor forgásirányát meg akarjuk változtatni, fel kell cserélni a polaritást.
Ezt négy kapcsolóval tudjuk megoldani. Az elrendezést H-hídnak nevezzük.



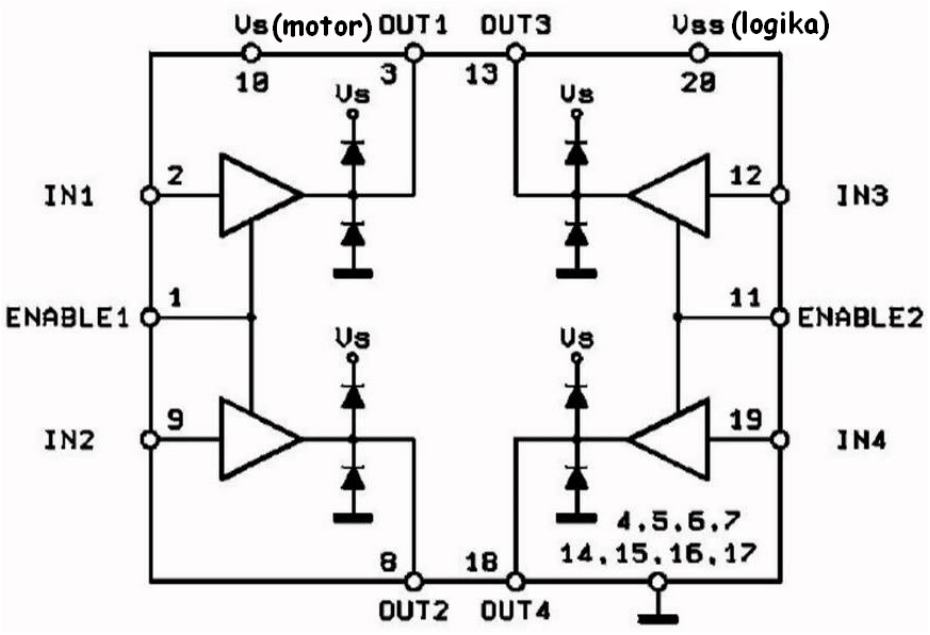
Forrás és leírás: hobbirobot.hu/content/vonalkoveto-i-robotika-kezdoknek

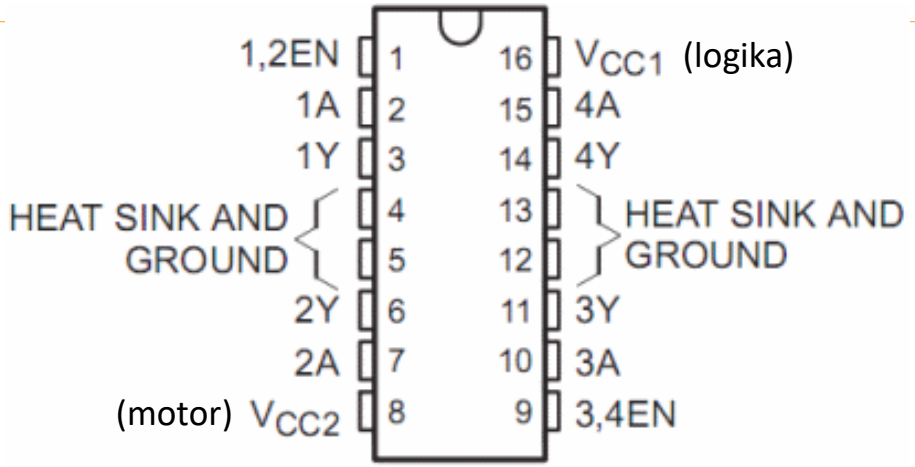
K1 és **K4** zárásakor a motor az egyik irányba forog.
K2 és **K3** zárásakor az ellenkező irányba forog.

Elektronikus vezérlés: tranzisztorokkal
Mivel **K1** és **K2,** s hasonlóan **K3** és **K4** sohasem lehet egyidejűleg zárva, elegendő félhidanként egy-egy vezérlőjel, mely a kapcsolókat ellenütemben zárja.



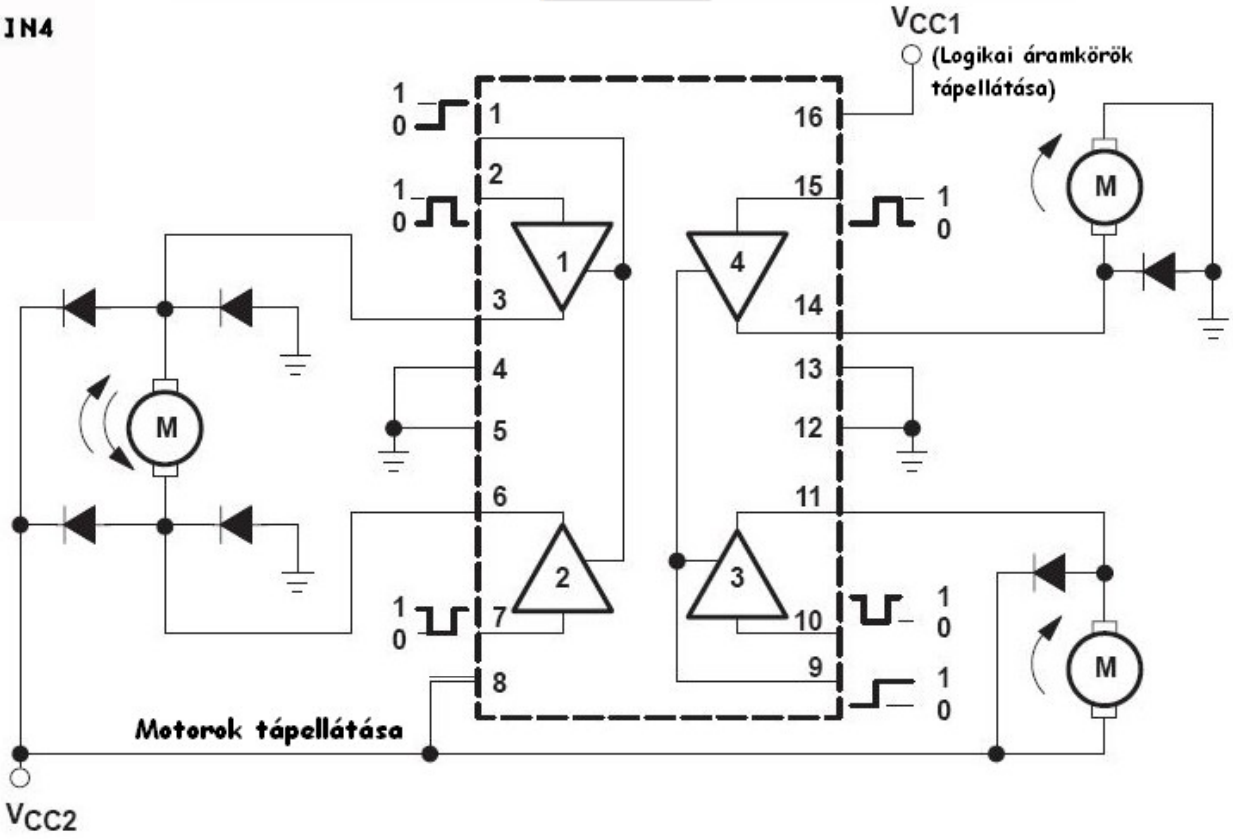
L293D 4db félhíd, vagy 2db H-híd





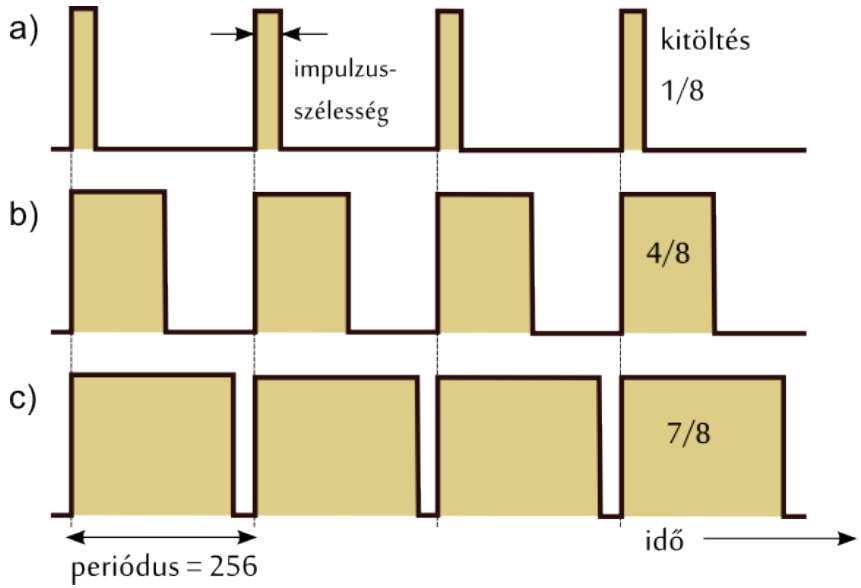
Félhíd: egyirányú forgás vezérlésére (pl. ventillátorok)
H-híd: polaritásváltást igénylő vezérlésekhez

V_{CC} = 4,5 – 36 V
V_A, V_{EN} = 2.3 – 7 V
I_{max} = 0,6 A
I_{peak} = 1,2 A (100 µs)



PWM: impulzus-szélesség moduláció

- PWM = pulse width modulation (impulzus-szélesség moduláció)
- A frekvencia állandó
- A kitöltés változtatható 0–255 között
- Az **`analogWrite()`** függvény csak bizonyos lábakra vonatkozóan használható
- Arduino Uno/Nano (**Atmega 328P**):



Időzítő	Csatorna	Kivezetés	Frekvencia
Timer0	OC0A, OC0B	6, 5	980 Hz
Timer1	OC1A, OC1B	9, 10	490 Hz
Timer2	OC2A, OC2B	11, 3	490 Hz

$$f_{PWM} = \frac{16\,MHz}{64 \cdot 256} = 976.56\,Hz$$

f_{CPU} — előosztás — periódus

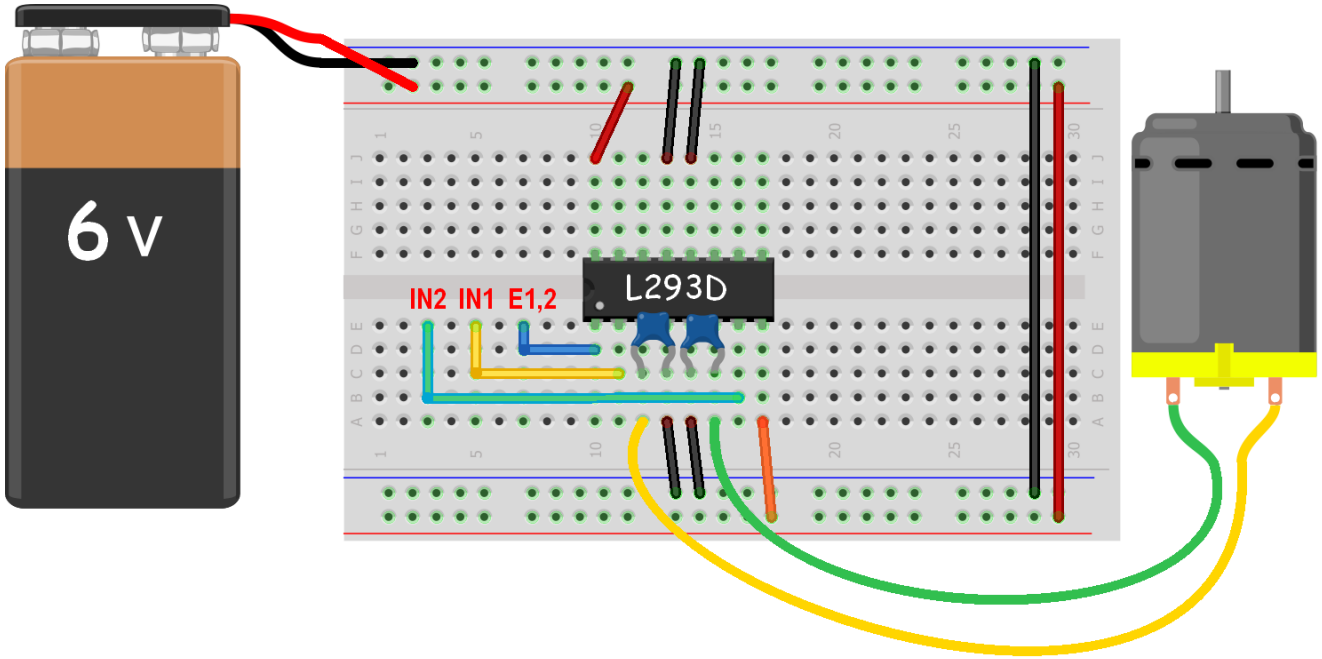
$$f_{PWM} = \frac{16\,MHz}{64 \cdot 510} = 490.19\,Hz$$

Egy motor vezérlése

Enable1,2	IN1	IN2	Motor
H	0	0	Stop
H	1	0	Előremenet
H	0	1	Hátramenet
H	1	1	Stop
L	X	X	Stop

E1,2: Letiltja vagy engedélyezi az 1. H-hidat

In1 és **In2** a forgás irányát adja meg.
Csak az **1,0** illetve **0,1** kombináció esetén van forgás.



Egyszerűsítési lehetőségek:
(ha sokalljuk a kivezetéseket)

1. Ha **E1,2** állandóan magas szintet kap, **In1** és **In2** önmagában elég a vezérléshez.
2. Ha **In2** = $\overline{\textbf{In1}}$ (inverter, IC-vel vagy tranzisztorral) akkor **E1,2** és **In1** elegendő a vezérléshez.

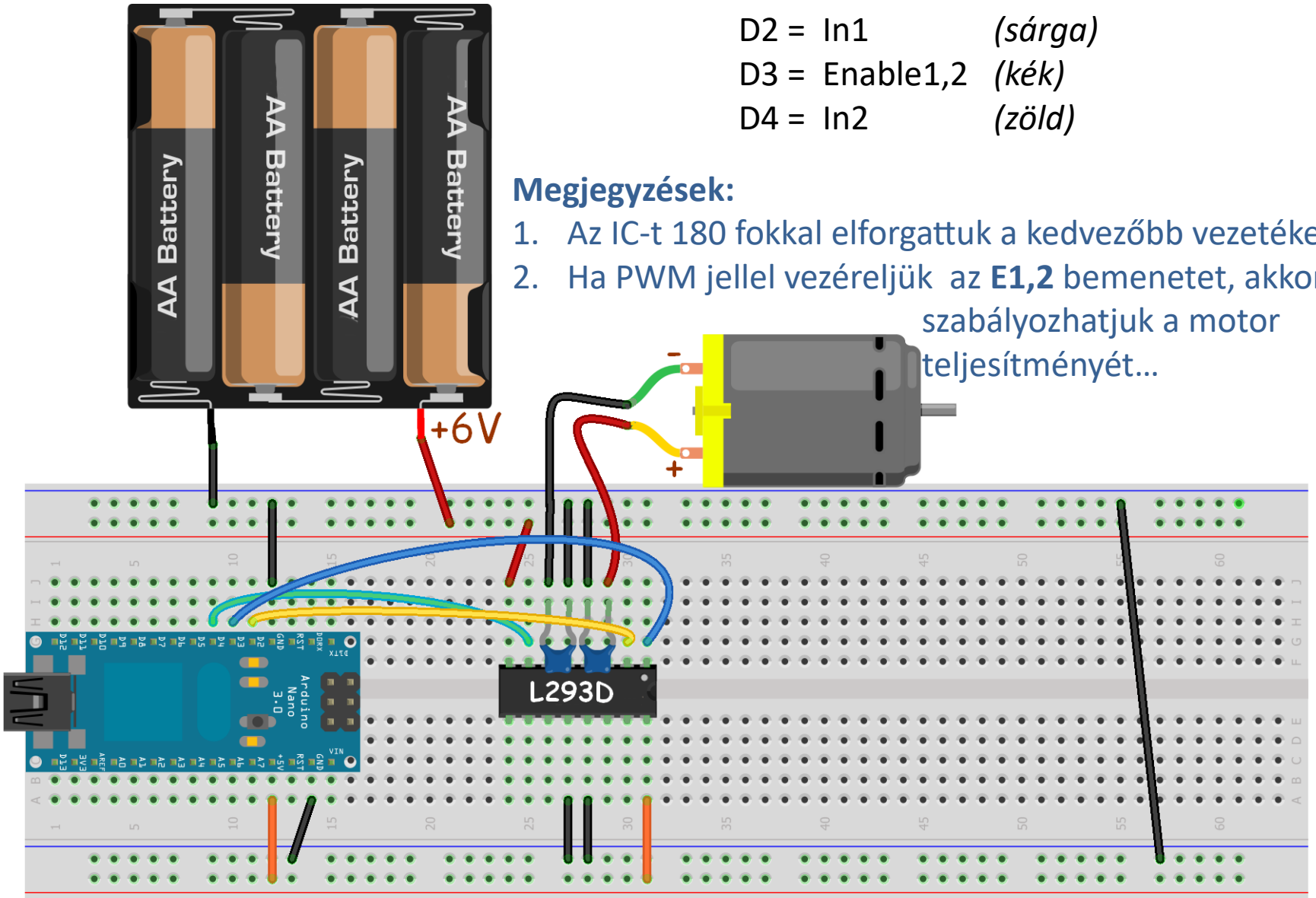
Egy motor vezérlése Arduinoval

Bekötési vázlat:

D2 = In1 *(sárga)*
D3 = Enable1,2 *(kék)*
D4 = In2 *(zöld)*

Megjegyzések:

1. Az IC-t 180 fokkal elforgattuk a kedvezőbb vezetékezéshez!
2. Ha PWM jellel vezéreljük az **E1,2** bemenetet, akkor szabályozhatjuk a motor teljesítményét…



Az **Arduino** itt az USB csatlakozón keresztül kap tápfeszültséget!

Made with Fritzing.org

L293D_test_1M.ino

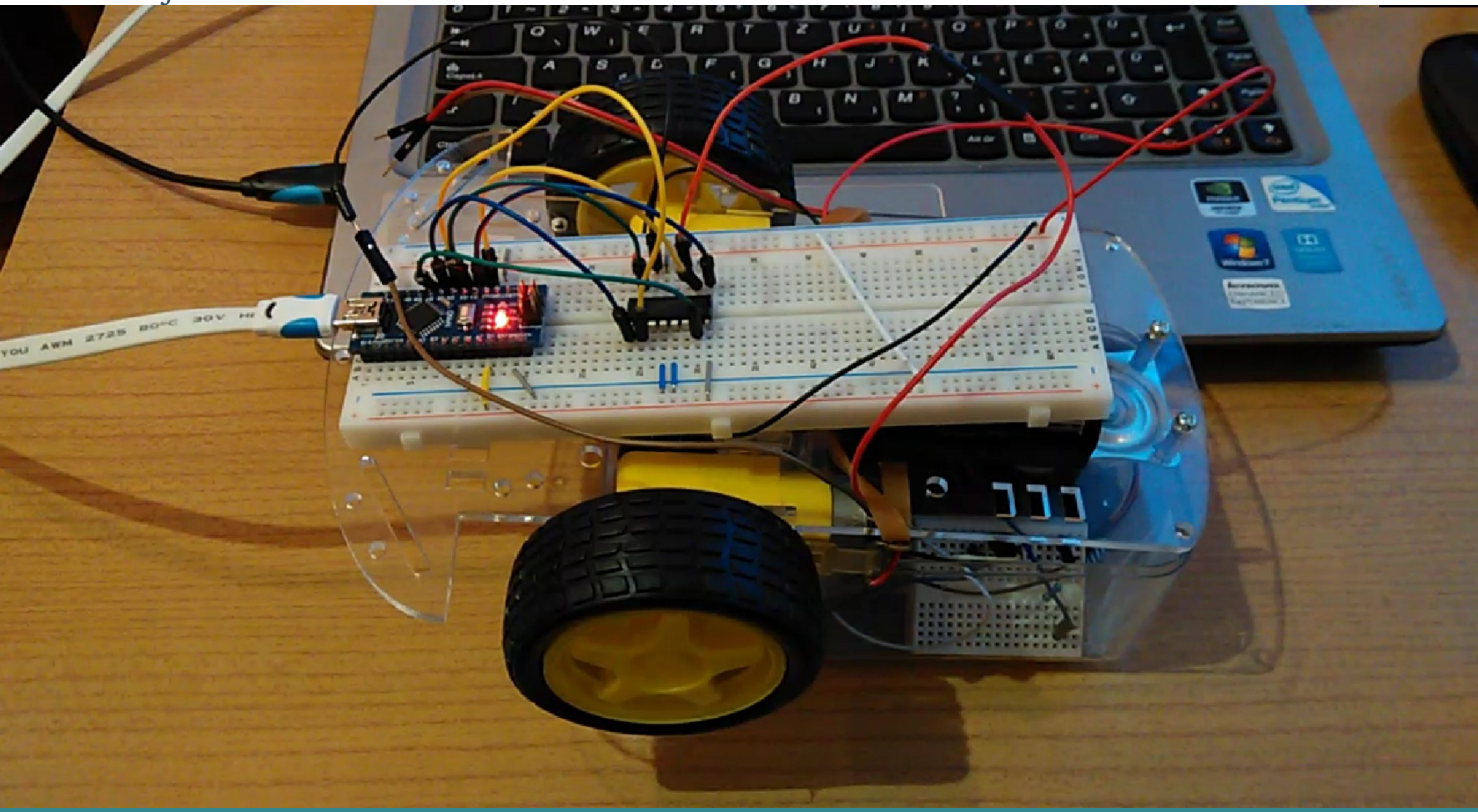
- Egy motor vezérlése. A végtelen ciklusban 5s várakozás után 2s előremenet, 1s várakozás, majd 2s hátramenet ismétlődik. Az előre- és hátramenet 75 %-os teljesítménnyel történik.

```
#define PWMA    3     //L293D pin1
#define D1A     2     //L293D pin2
#define D2A     4     //L293D pin7


void setMotor(int speed, boolean reverse)
{
  analogWrite(PWMA, speed);
  digitalWrite(D1A, !reverse);
  digitalWrite(D2A, reverse);
}


void setup()
{
  Serial.begin(9600);
  pinMode(PWMA, OUTPUT);
  pinMode(D1A, OUTPUT);
  pinMode(D2A, OUTPUT);
  Serial.println("L293D test program");
}
```

```
void loop()
{
  delay(5000);
  Serial.println("move forward 75%");
  setMotor(196, 0); // Forward 75%
  delay(2000);
  setMotor(0, 0);   // Stop motor
  delay(1000);
  Serial.println("move reverse 75%");
  setMotor(196, 1); // Reverse 75%
  delay(2000);
  setMotor(0, 0);   // Stop motor
}
```

L293D_test_1M.ino futási eredménye

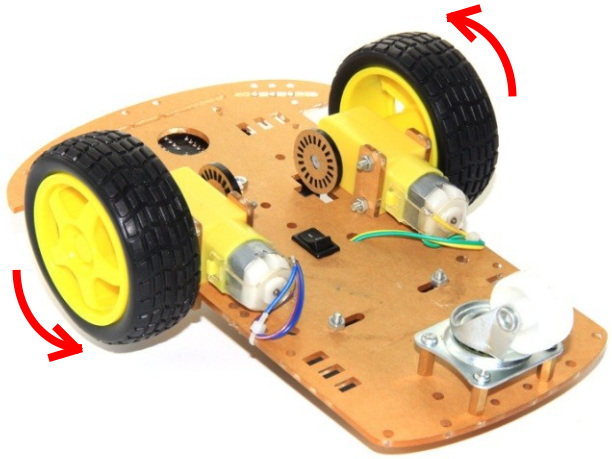
- A végtelen ciklusban 5s várakozás után 2s előremenet, 1s várakozás, majd 2s hátramenet ismétlődik.

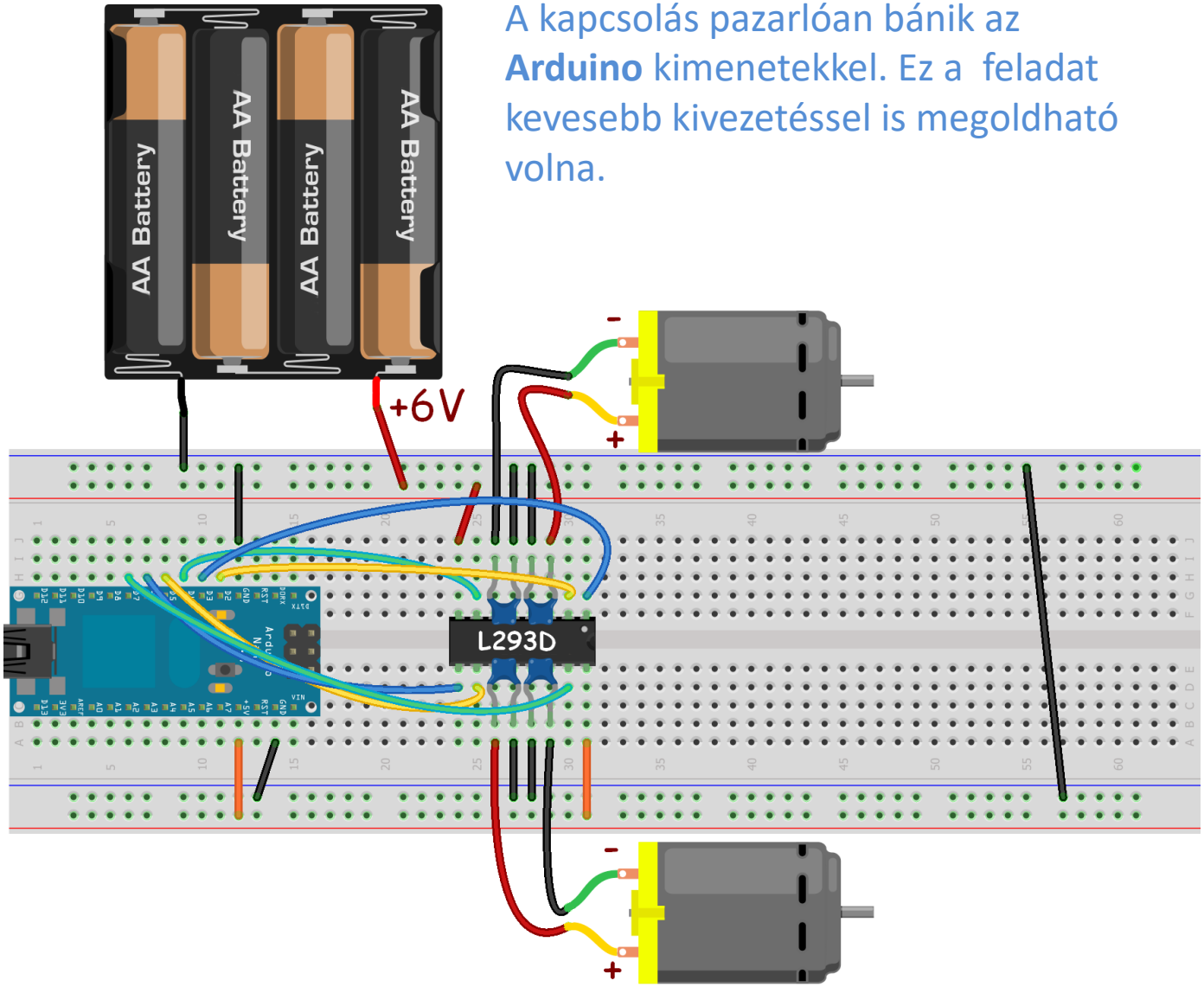


Két motor vezérlése Arduinoval

Megjegyzés:
A rajzon látható kapcsolásban azonos vezérlés mellett a két motor azonos irányba forog.

A fényképen látható független jobb és bal kerék meghajtású kocsihoz azonban az egyik motor polaritását fel kell cserélni!
Egyenesvonalú mozgásnál a két motor forgása ellentétes irányú!





A kapcsolás pazarlóan bánik az **Arduino** kimenetekkel. Ez a feladat kevesebb kivezetéssel is megoldható volna.

Az **Arduino** itt az USB csatlakozón keresztül kap tápfeszültséget!

Made with Fritzing.org

L293D_test_2M.ino

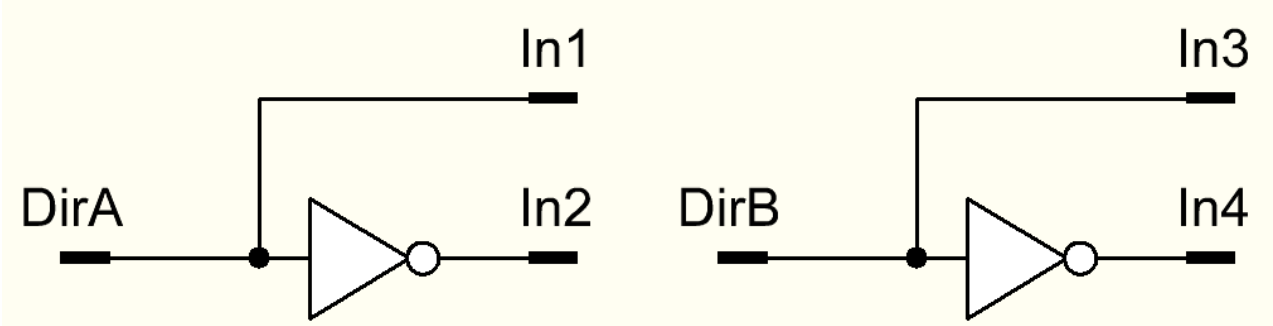
- Két motor vezérlése. A végtelen ciklusban 5s várakozás után 2s előremenet, 1s várakozás, majd 2s hátramenet ismétlődik.
- Az előre- és hátramenet 75 %-os teljesítménnyel történik.

```
#define PWMA    3     //L293D pin1
#define D1A     2     //L293D pin2
#define D2A     4     //L293D pin7
#define PWMB    6     //L293D pin9
#define D3B     5     //L293D pin10
#define D4B     7     //L293D pin15

void setup() {
  Serial.begin(9600);
  pinMode(PWMA, OUTPUT);
  pinMode(D1A, OUTPUT);
  pinMode(D2A, OUTPUT);
  pinMode(PWMB, OUTPUT);
  pinMode(D3B, OUTPUT);
  pinMode(D4B, OUTPUT);
  Serial.println("L293D test program");
}
```

```
void setMotor(int speed, boolean reverse)
{
  analogWrite(PWMA, speed);
  analogWrite(PWMB, speed);
  digitalWrite(D1A, !reverse);
  digitalWrite(D2A, reverse);
  digitalWrite(D3B, !reverse);
  digitalWrite(D4B, reverse);
}

void loop() {
  delay(5000);
  Serial.println("move forward 75%");
  setMotor(196, 0); // Forward 75%
  delay(2000);
  setMotor(0, 0);   // Stop motor
  delay(1000);
  Serial.println("move reverse 75%");
  setMotor(196, 1); // Reverse 75%
  delay(2000);
  setMotor(0, 0);   // Stop motor
}
```

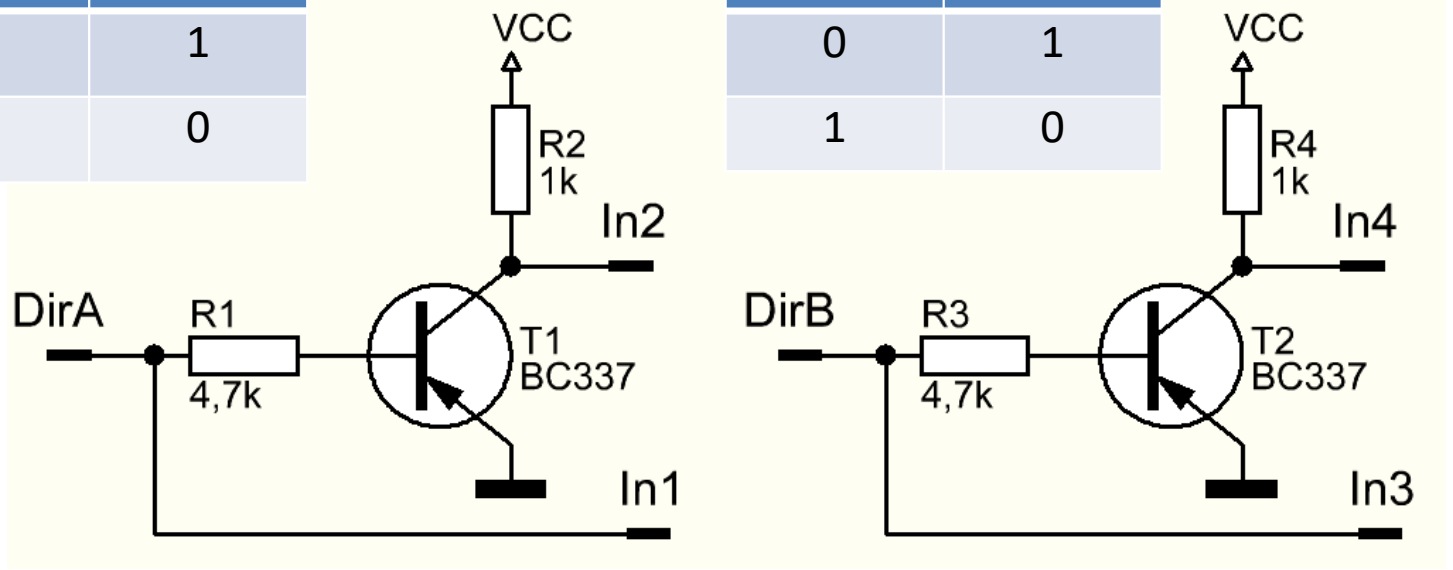
Egyszerűbb forgásirány-váltás

- Vegyük észre, hogy **In2** mindig **In1** inverze (az előző programban `D2A = !D1A`)
- Hasonlóan **In4** is mindig **In3** inverze (az előző programban `D4B = !D3B`).
- Ezeket az összefüggéseket elektronikusan is megvalósíthatjuk (logikai kapukkal, vagy tranzisztorokkal), s akkor kevesebb **Arduino** kivezetésre lesz szükségünk.



DirA	In2
0	1
1	0

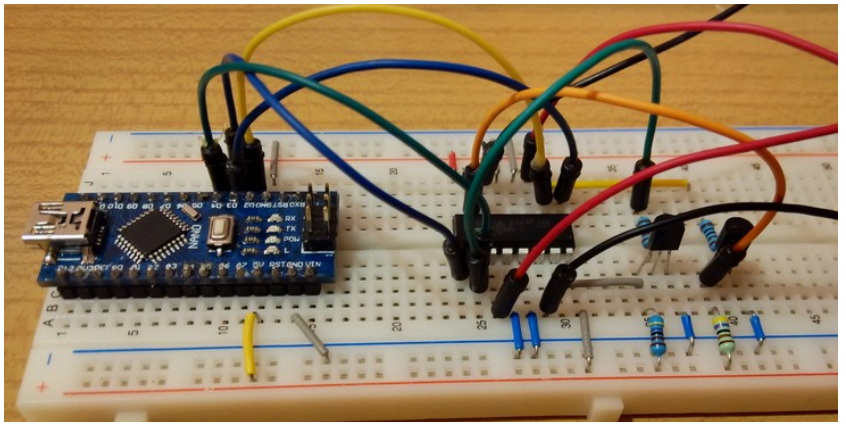
DirB	In4
0	1
1	0



Egyszerűbb forgásirány-váltás

Megjegyzés: Ebben a példában a motorok teljesítményét nem lehet függetlenül szabályozni (**PWMA = PWMB**), de ez nem mindig jó döntés!

Általában motoronként egy irányváltó és egy teljesítmény-szabályozó bemenetet szokás használni.



Az **Arduino** itt még az USB csatlakozón keresztül kap tápfeszültséget, de kipróbáláskor a **Vin** bemenetén kell táplálni (vagy a motor +6V-járól, vagy egy 9 V-os elemről)

Made with Fritzing.org

Egyszerűbb forgásirány-váltás

Két motor egyszerűsített vezérlése. A végtelen ciklusban 5s várakozás után 2s előremenet, 1s várakozás, majd 2s hátramenet ismétlődik. Az előre- és hátramenet 75 %-os teljesítménnyel történik.

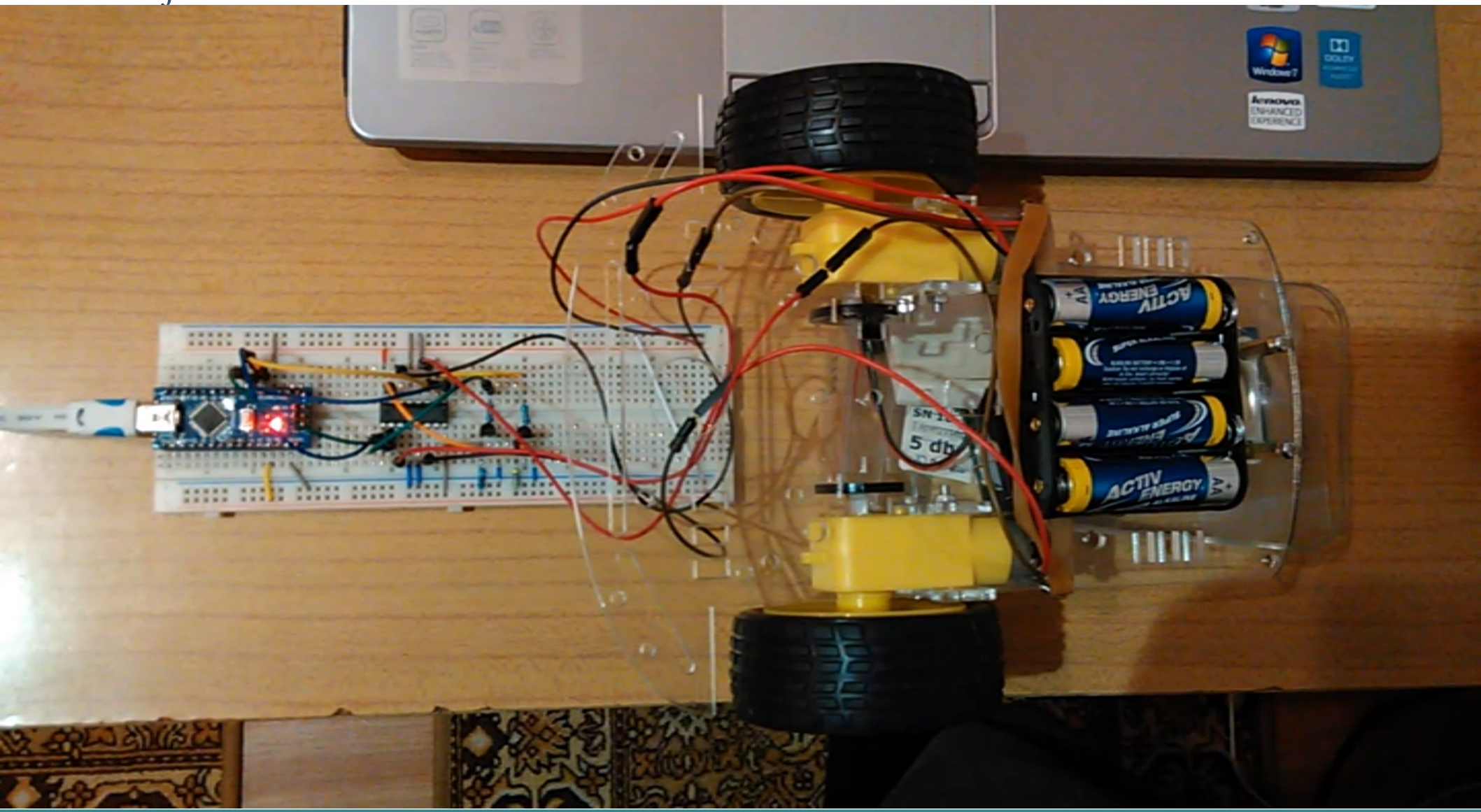
```
#define PWMA    3     //L293D pin1,pin9
#define DIRA    2     //L293D pin2
#define DIRB    4     //L293D pin10


void setMotor(int speed, boolean reverse)
{
  analogWrite(PWMA, speed);
  digitalWrite(DIRA, !reverse);
  digitalWrite(DIRB, reverse);
}


void setup()
{
  Serial.begin(9600);
  pinMode(PWMA, OUTPUT);
  pinMode(DIRA, OUTPUT);
  pinMode(DIRB, OUTPUT);
  Serial.println("L293D test program");
}
```

Itt szoftveresen fordítottuk meg az egyik motor forgásirányát!

```
void loop()
{
  delay(5000);
  Serial.println("move forward 75%");
  setMotor(196, 0); // Forward 75%
  delay(2000);
  setMotor(0, 0);   // Stop motor
  delay(1000);
  Serial.println("move reverse 75%");
  setMotor(196, 1); // Reverse 75%
  delay(2000);
  setMotor(0, 0);   // Stop motor
}
```

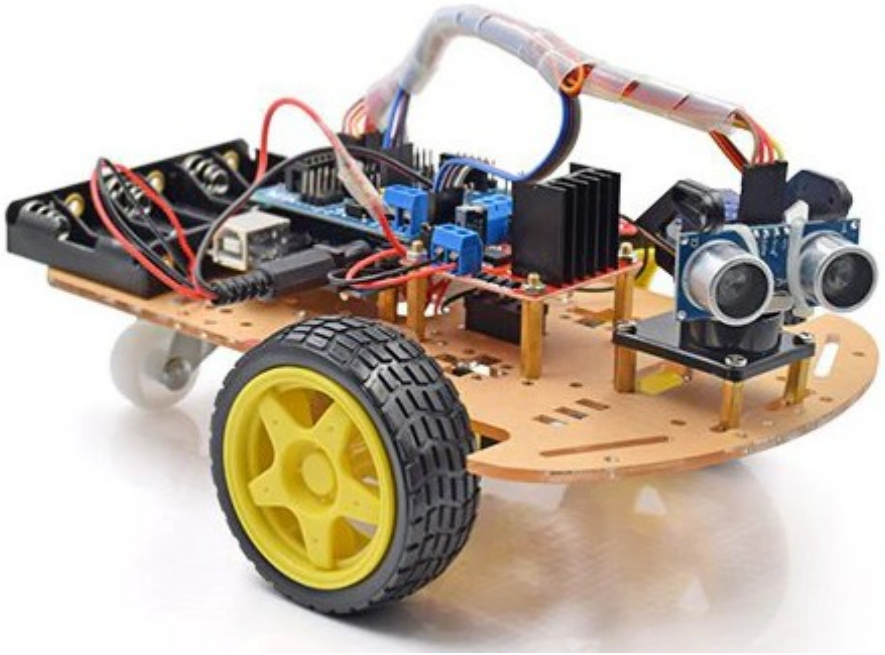
L293D_test2_2M.ino futási eredménye

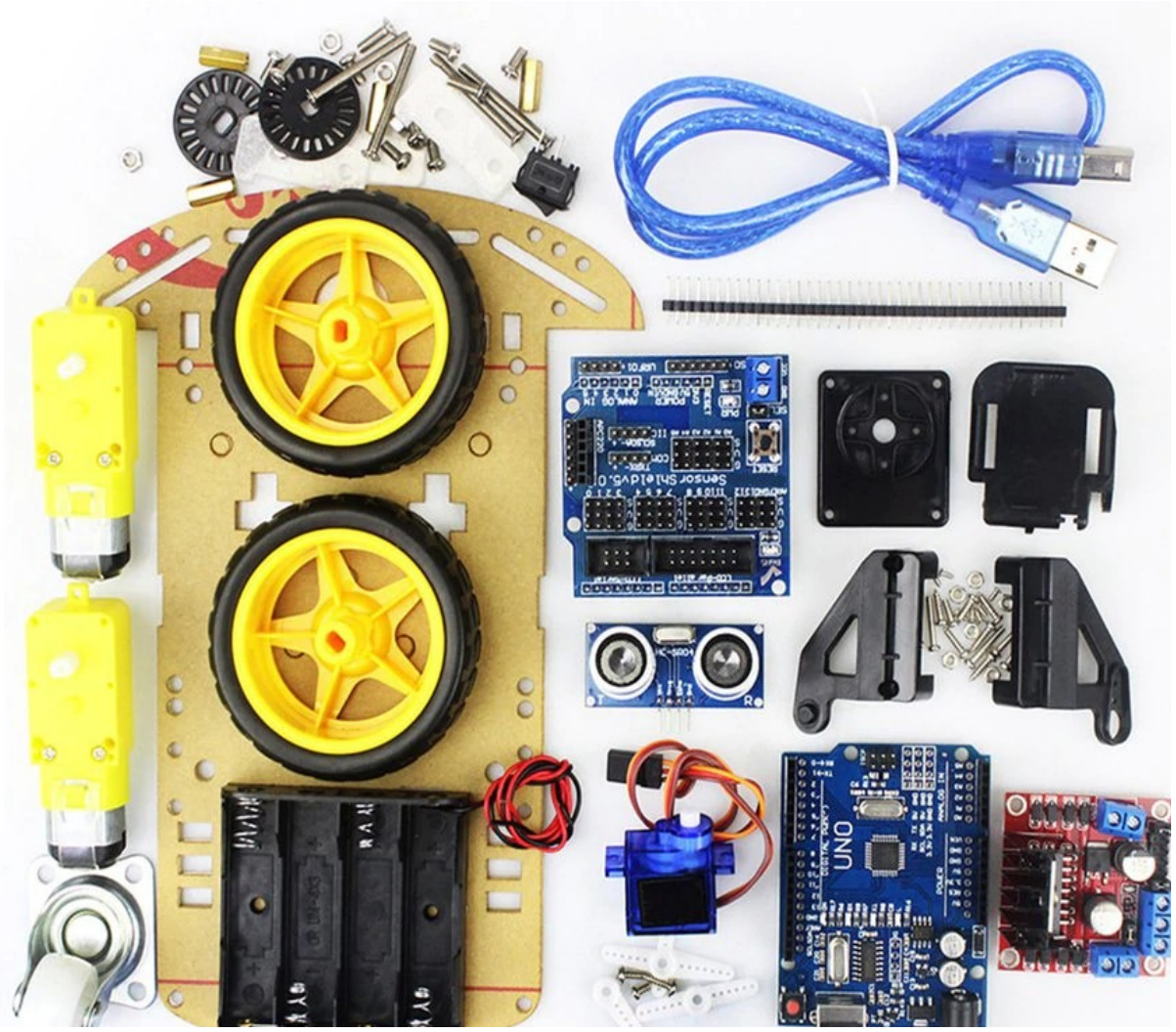
- A végtelen ciklusban 5s várakozás után 2s előremenet, 1s várakozás, majd 2s hátramenet ismétlődik.



Építsünk robotot!

- A képen látható 2WD (kétkerék meghajtású) robot az olcsóbbak közé tartozik, de nem volt szerencsés választás mert barkácsolás és kiegészítők nélkül nem lehet összerakni



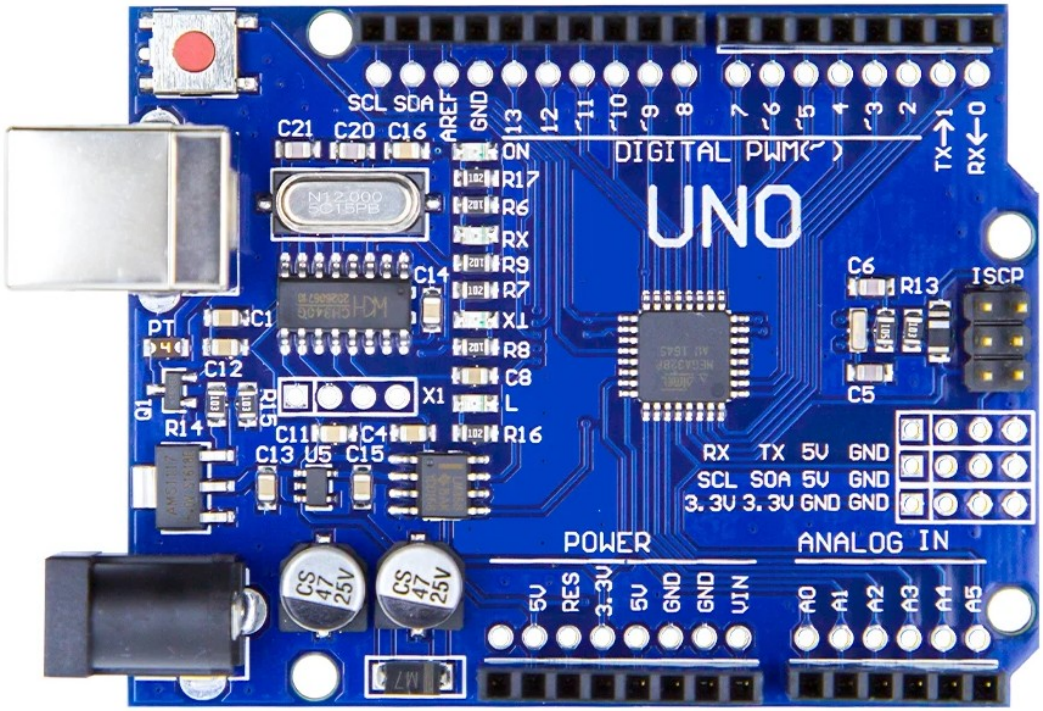


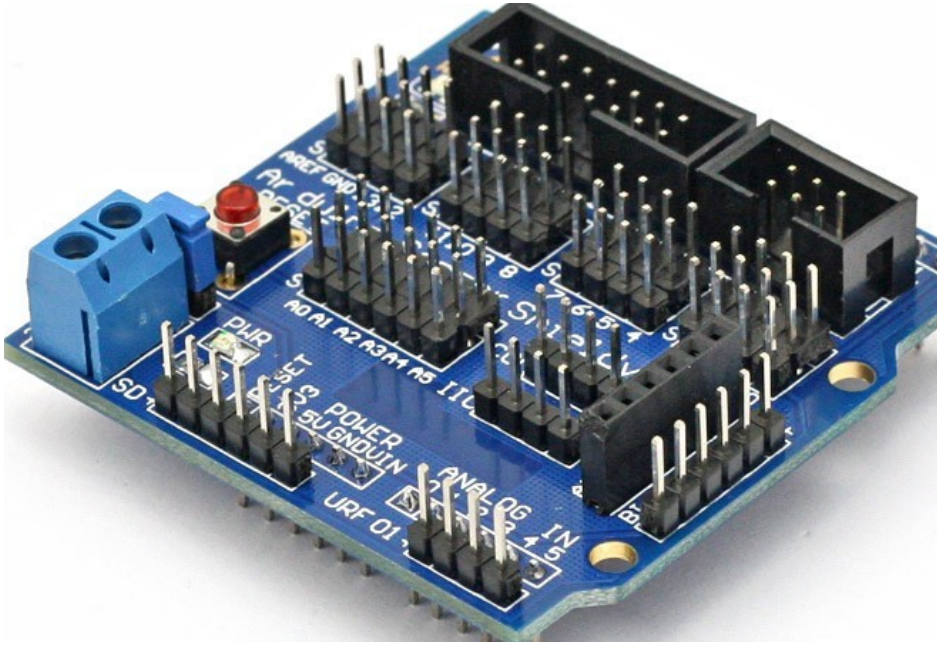
A robot alváz és a motorok

- A 2WD robotalváz (szerelőlap, bolygókerék, elemtartó, 2 db egyenáramú motor a műanyag kerekekkel) külön készletként is beszerezhető
- Könnyen összeszerelhető, de a szerelőlapon található furatok egyetlen általam ismert mikrovezérlő vagy motorvezérlő kártyához sem illeszkednek – jó, ha van otthon fúrógép, távtartók és csavarok

A mikrovezérlő kártya és kiegészítői

- Az Arduino kártya egy szokványos UNO klón, CH340 USB-soros átalakítóval
- Az Arduino IDE Tools menüjében az **Arduino/ Genuino Uno** kártyát kell kiválasztani

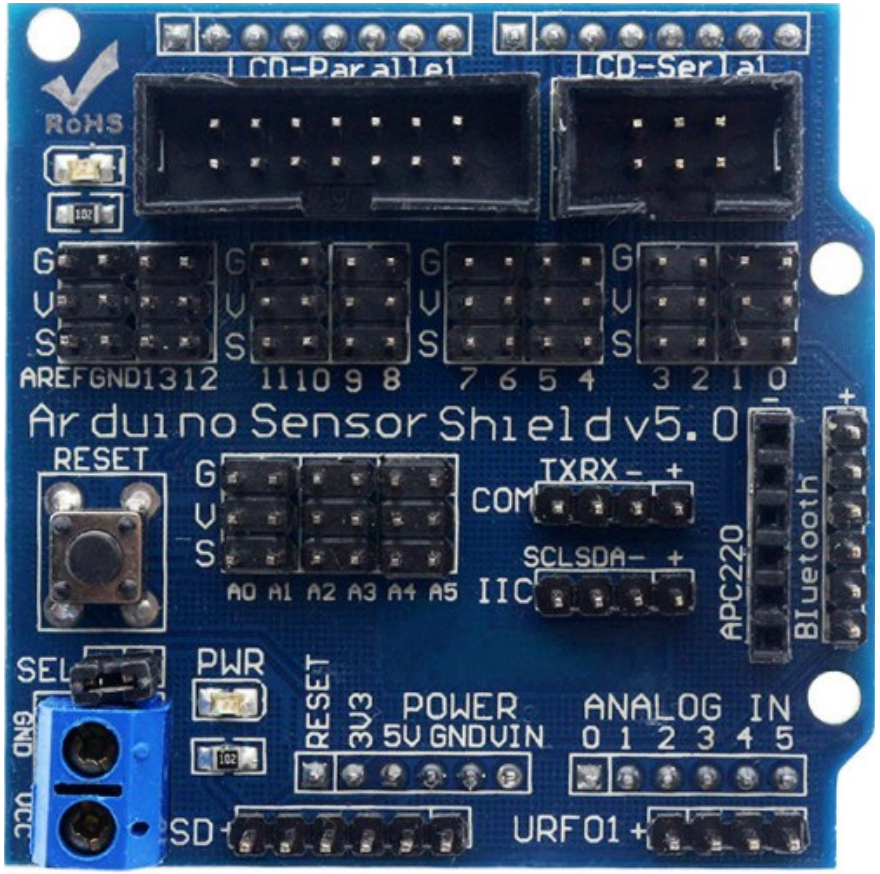


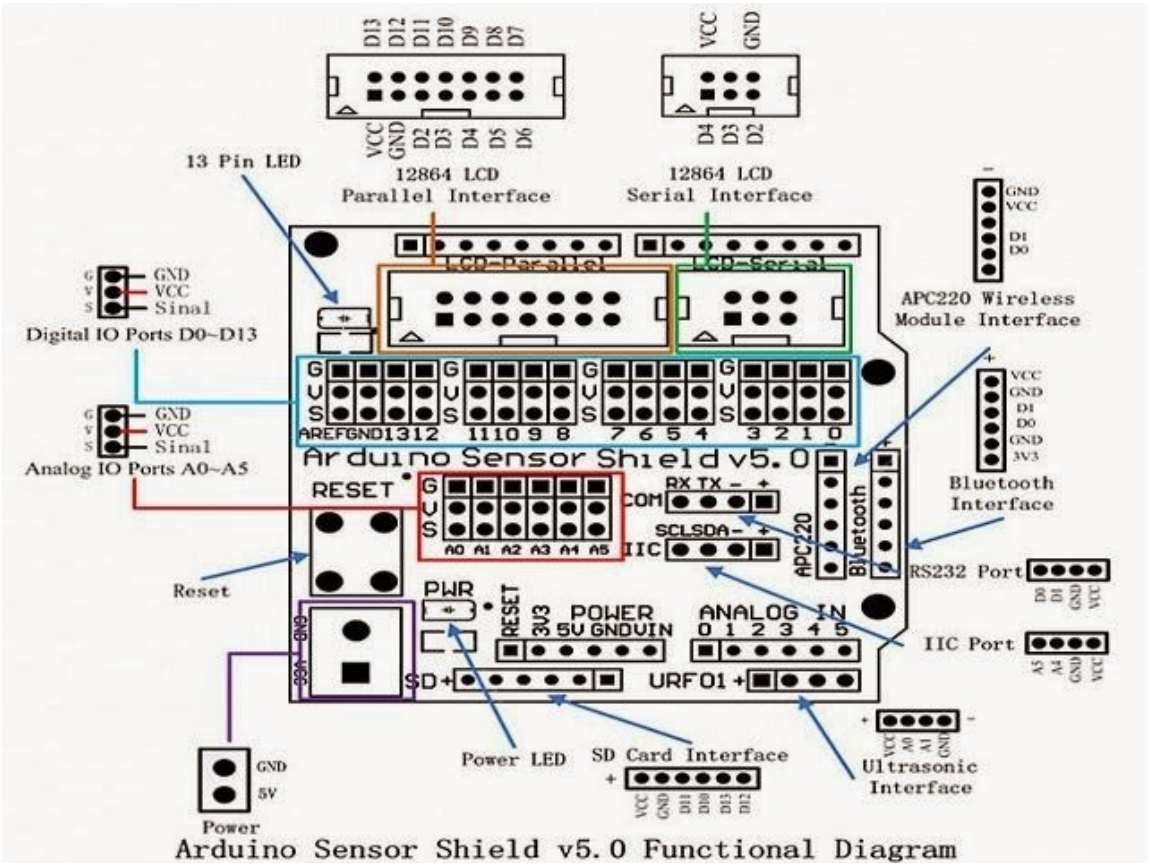


- A Sensor shield kártya csak az Arduino kártya kivezetéseinek kényelmesebb elérésére való, egyébként nélkülözhető

Arduino Sensor Shield

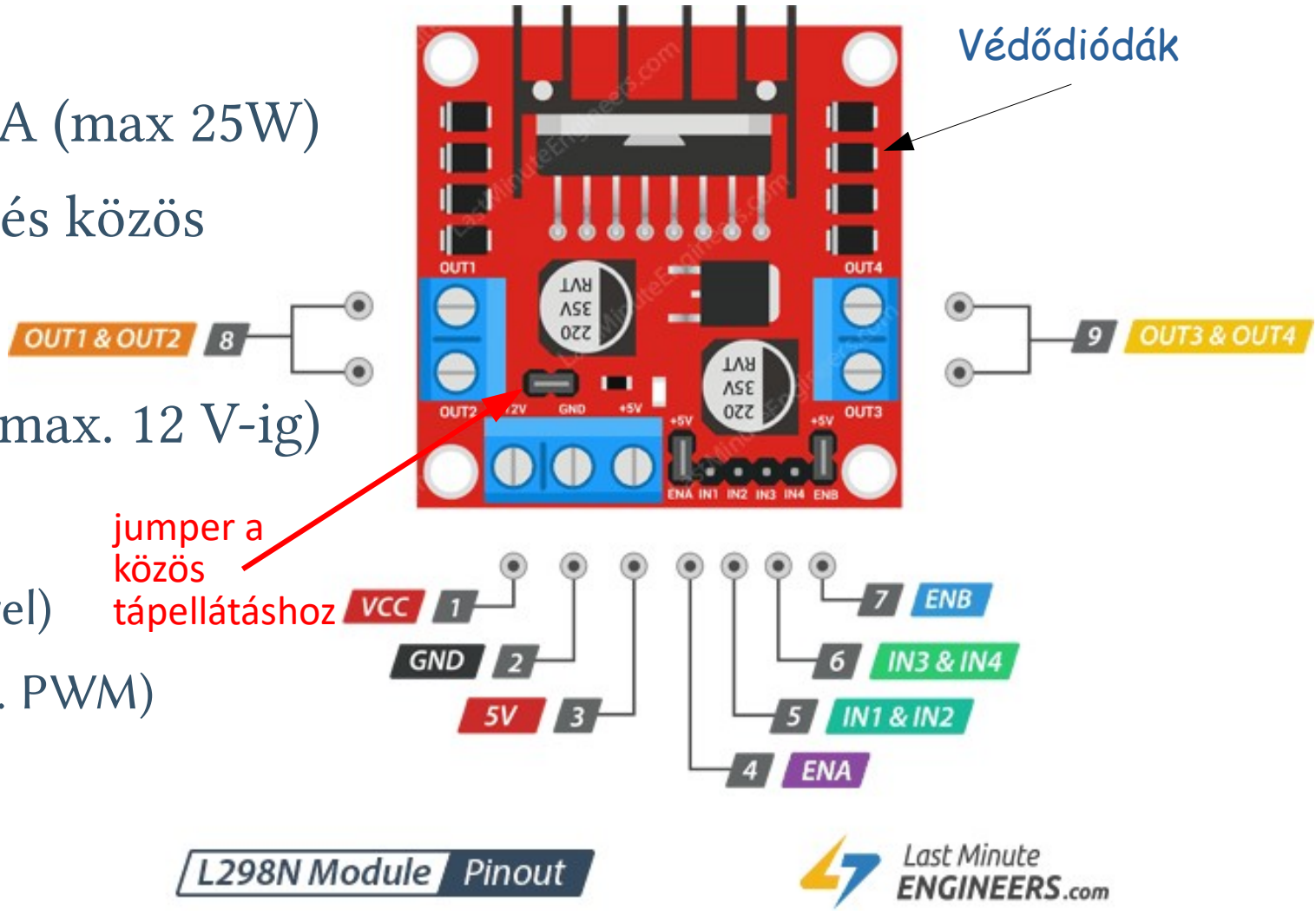
- Ez a fedlap a huzalozás, az eszközök csatlakoztatásának megkönnyítésére szolgál
- A hármas tüskesorok (GND, Vcc, Signal) a szenzorok vagy a szervók közvetlen csatlakoztatására szolgálnak





Az L298N motorvezérlő kártya

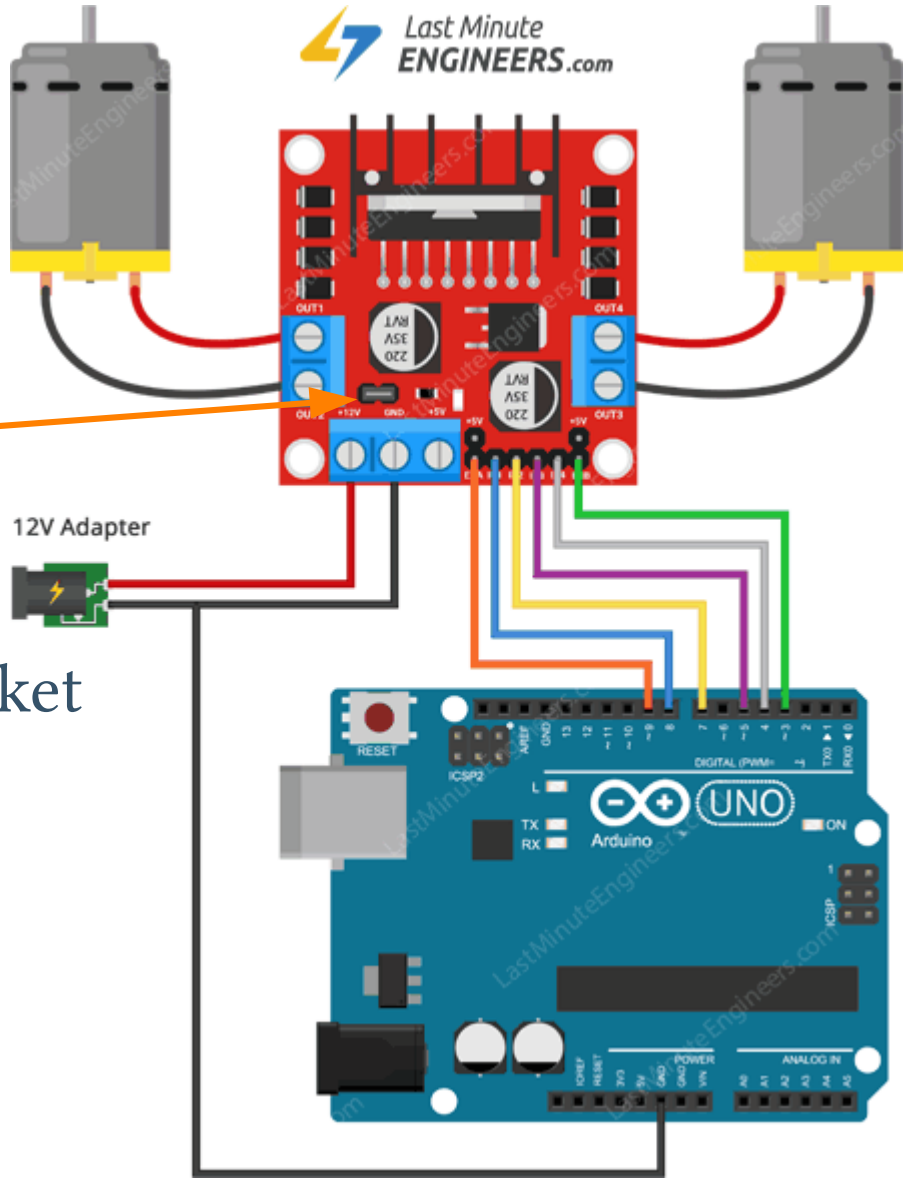
- A motorvezérlő kártya egy **L298N** IC-t tartalmaz (ST Microelectronics)
- Két teljes híd
- Max 46 V, ill. 2 A (max 25W)
- Motor és vezérlés közös vagy külön táplálással (közös táplálás max. 12 V-ig)
- Engedélyezés:
  - fixen (jumperrel)
  - Külső jellel (pl. PWM)



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

Egy lehetséges kapcsolás

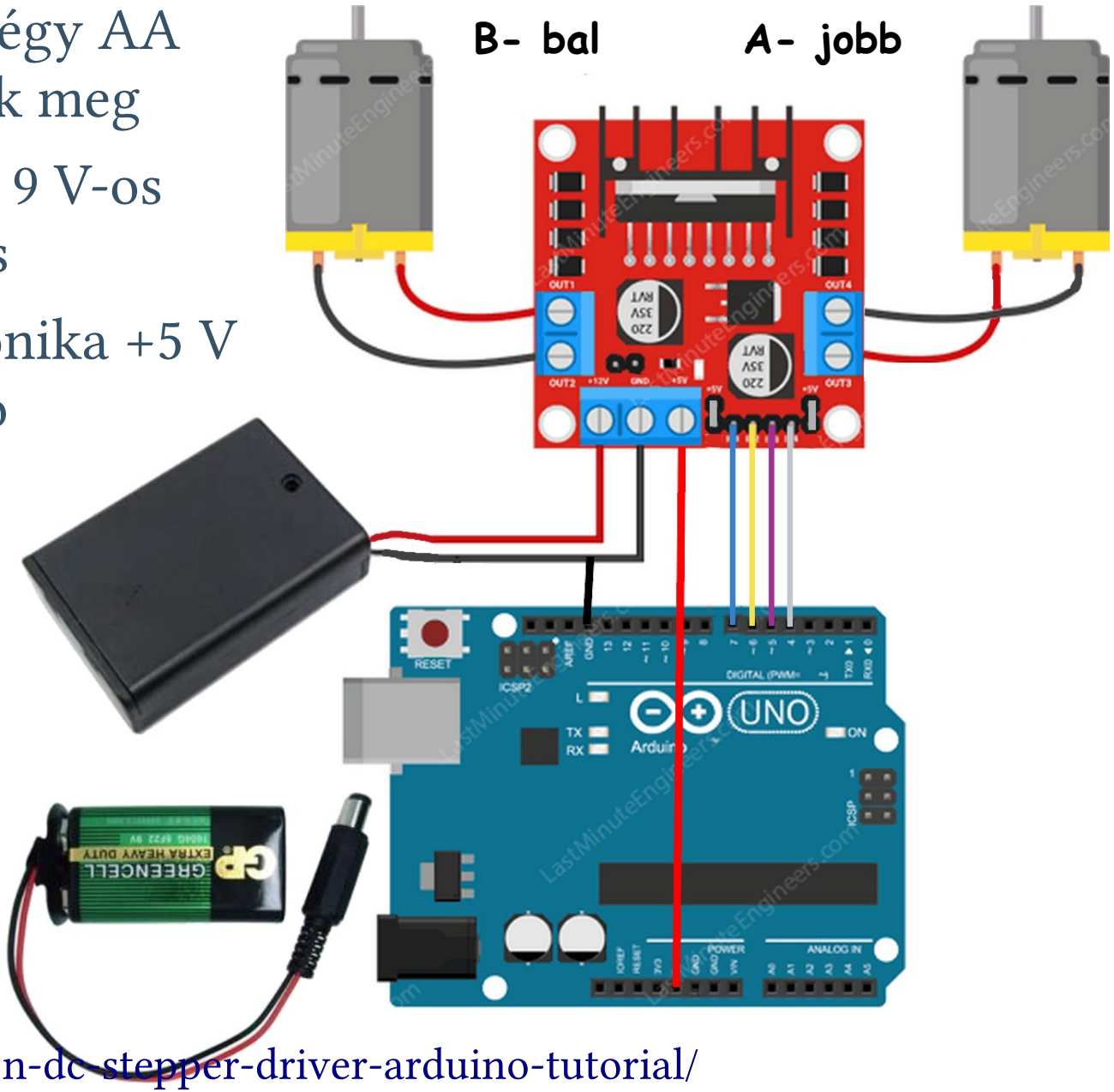
- Az Arduino kártya itt külön táplálást kap, csak a közös pont (GND) van közösítve, a motorvezérlő kártya pedig saját maga állítja elő az 5V tápfeszültséget (jumper van)
- Ha a motorok teljesítményét (a robot sebességét) is szabályozni akarjuk, akkor az Enable bemeneteket is az Arduino-val kell meghajtani



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

Az általunk megépített kapcsolás

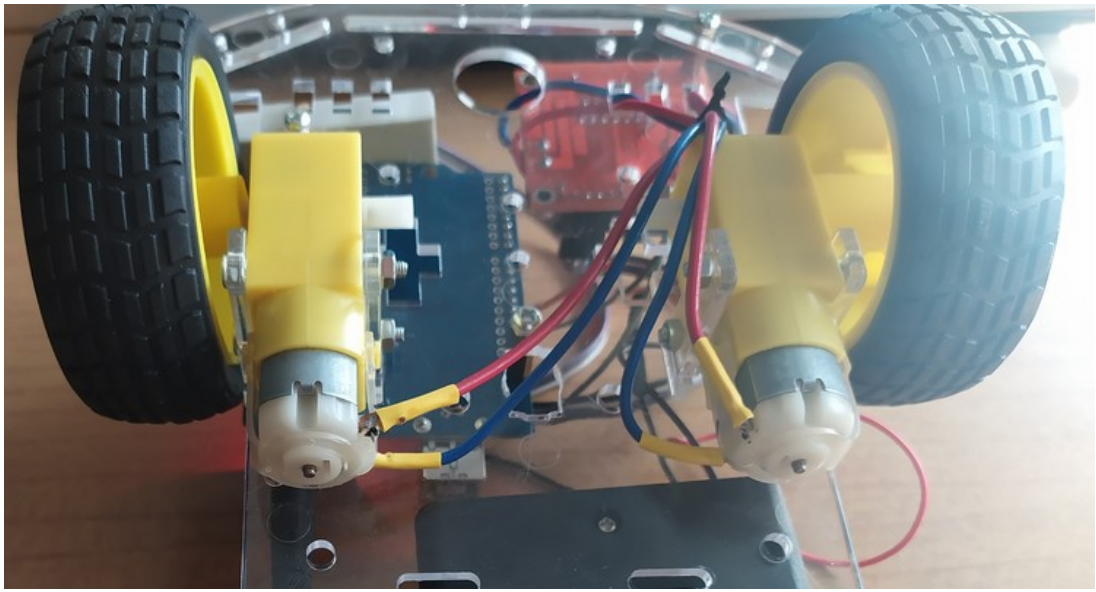
- A motorok táplálását négy AA ceruzaelemmel oldottuk meg
- Az Arduino kártya egy 9 V-os elemről kap táplálást és
- A motorvezérlő elektronika +5 V feszültségét az Arduino kártya biztosítja
- A motorok tükör-szimmetrikus bekötése hardveresen biztosítja az ellentétes irányú forgást a bal és jobb motornál

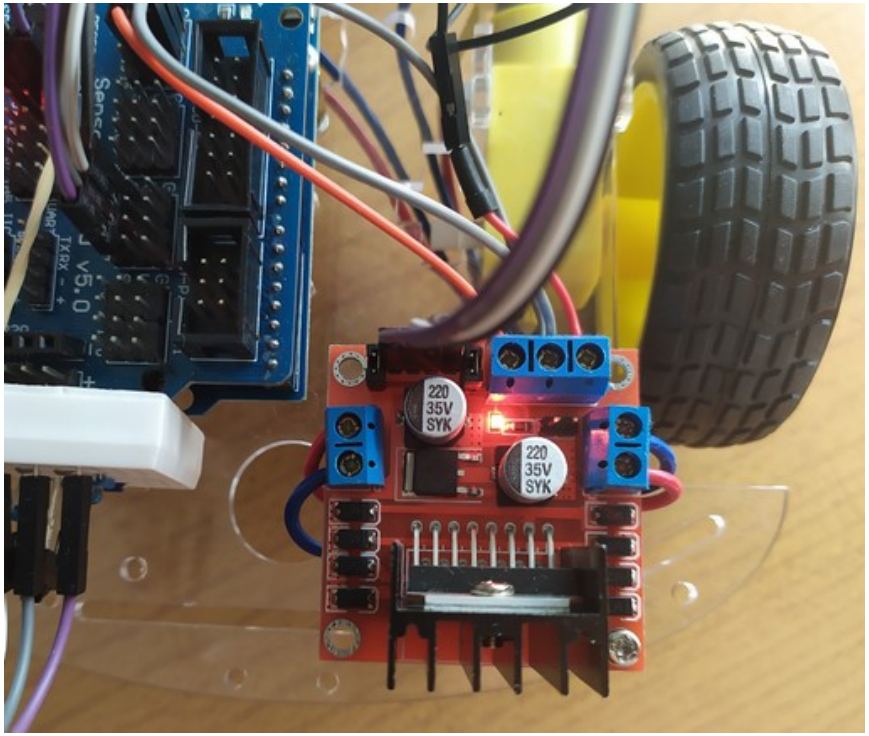


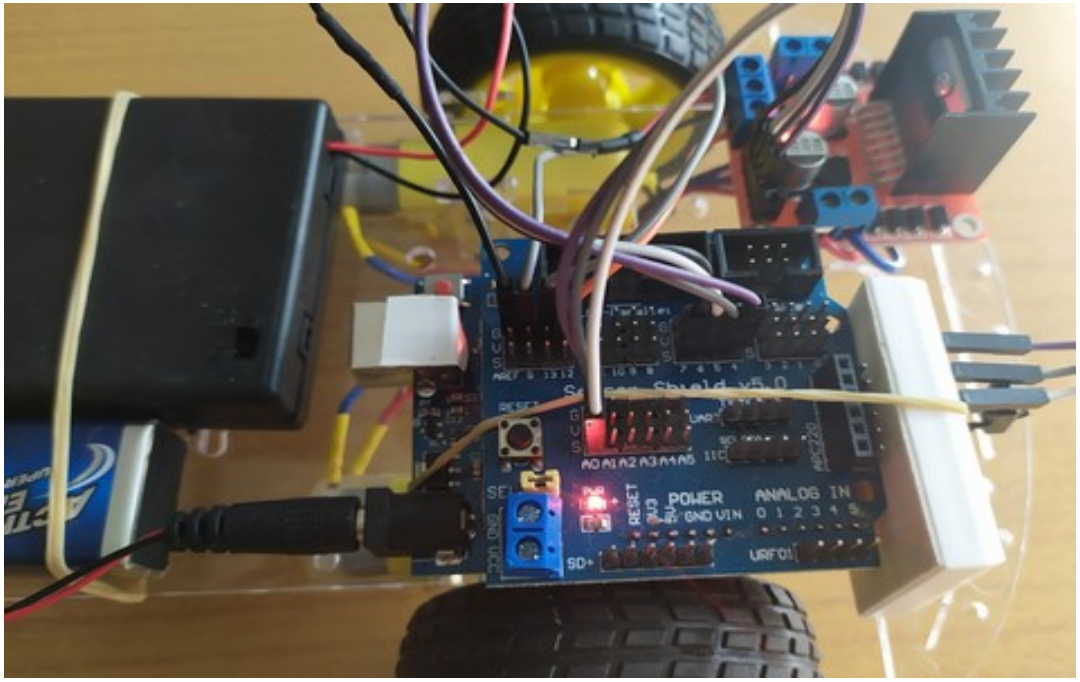
- Felhasznált forrás: lastminuteengineers.com/l298n-dc-stepper-driver-arduino-tutorial/

A megépített kapcsolás részletei

- A részletfotókon látható a motorok bekötése és a tápellátás megoldása







motor_test.ino

- Ez az egyszerű kis program csak arra szolgál, hogy a motorok működését és forgásirányát ellenőrizni tudjuk
- **Vezérlés:**

B1A	B1B	A1A	A1B	
0	0	0	0	Stop
1	0	1	0	Forward (Előre)
0	1	0	1	Reverse (Hátra)
1	0	0	1	Left (Balra)
0	1	1	0	Right (Jobbra)

- A program a soros porton kiírja, hogy éppen mit csinál (9600 bps)
- A program: Előre 3 mp, Hátra 3mp, Jobbra 3 mp, Balra 3 mp, közben 1-1 mp Stop

```
#define B1A 7   // Bal motor előre
#define B1B 6   // Bal motor hátra
#define A1A 5   // Jobb motor előre
#define A1B 4   // Jobb motor hátra
void setup() {
  Serial.begin(9600);
  Serial.println("Motor test");
  pinMode(B1A,OUTPUT);
  pinMode(B1B,OUTPUT);
  pinMode(A1A,OUTPUT);
  pinMode(A1B,OUTPUT);
  digitalWrite(B1A,LOW);
  digitalWrite(B1B,LOW);
  digitalWrite(A1A,LOW);
  digitalWrite(A1B,LOW);
}
```

Folytatás a következő oldalon...

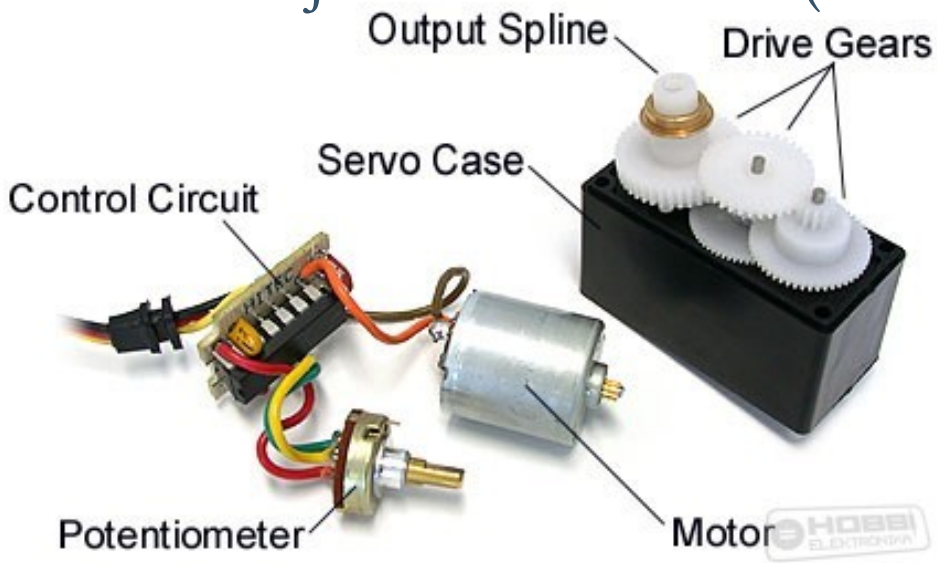
motor_test.ino

```
void loop() {
  Serial.println("Forward ALL");
  digitalWrite(B1A,HIGH);
  digitalWrite(B1B,LOW);
  digitalWrite(A1A,HIGH);
  digitalWrite(A1B,LOW);
  delay(3000);
  digitalWrite(B1A,LOW);
  digitalWrite(B1B,LOW);
  digitalWrite(A1A,LOW);
  digitalWrite(A1B,LOW);
  delay(1000);
  Serial.println("Reverse ALL");
  digitalWrite(B1A,LOW);
  digitalWrite(B1B,HIGH);
  digitalWrite(A1A,LOW);
  digitalWrite(A1B,HIGH);
  delay(3000);
  digitalWrite(B1A,LOW);
  digitalWrite(B1B,LOW);
  digitalWrite(A1A,LOW);
  digitalWrite(A1B,LOW);
  delay(1000);
```

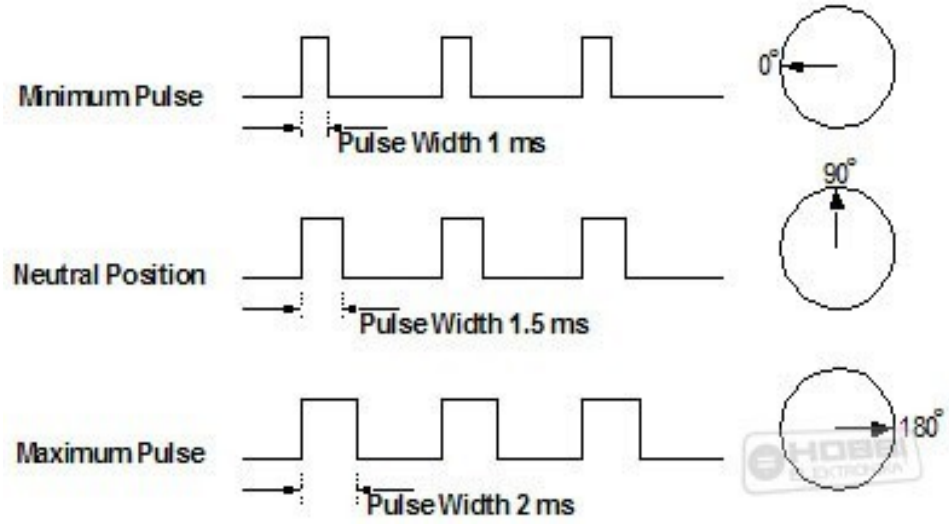
```
Serial.println("Turn RIGHT");
  digitalWrite(B1A,HIGH);
  digitalWrite(B1B,LOW);
  digitalWrite(A1A,LOW);
  digitalWrite(A1B,LOW);
  delay(3000);
  digitalWrite(B1A,LOW);
  digitalWrite(B1B,LOW);
  digitalWrite(A1A,LOW);
  digitalWrite(A1B,LOW);
  delay(1000);
  Serial.println("Turn LEFT");
  digitalWrite(B1A,LOW);
  digitalWrite(B1B,LOW);
  digitalWrite(A1A,HIGH);
  digitalWrite(A1B,LOW);
  delay(3000);
  digitalWrite(B1A,LOW);
  digitalWrite(B1B,LOW);
  digitalWrite(A1A,LOW);
  digitalWrite(A1B,LOW);
  delay(2000);
}
```

Emlékeztető: Szervo motorok

- A szervo egy pozícionálható motor, amely „ismeri” az aktuális pozícióját, és a cél pozíciót. Feladata, hogy az aktuális pozícióból a kívánt pozícióba álljon
- Felépítése: vezérlő áramkör, mechanikusan összekapcsolt motor és potenciométer (a potméterrel leosztott feszültség jelzi a pozíciót)
- Általában 180 °-os tartományban mozgatható, 1 – 2 ms szélességű, 50 Hz-es jellel vezérelhető (PWM)



Felépítés



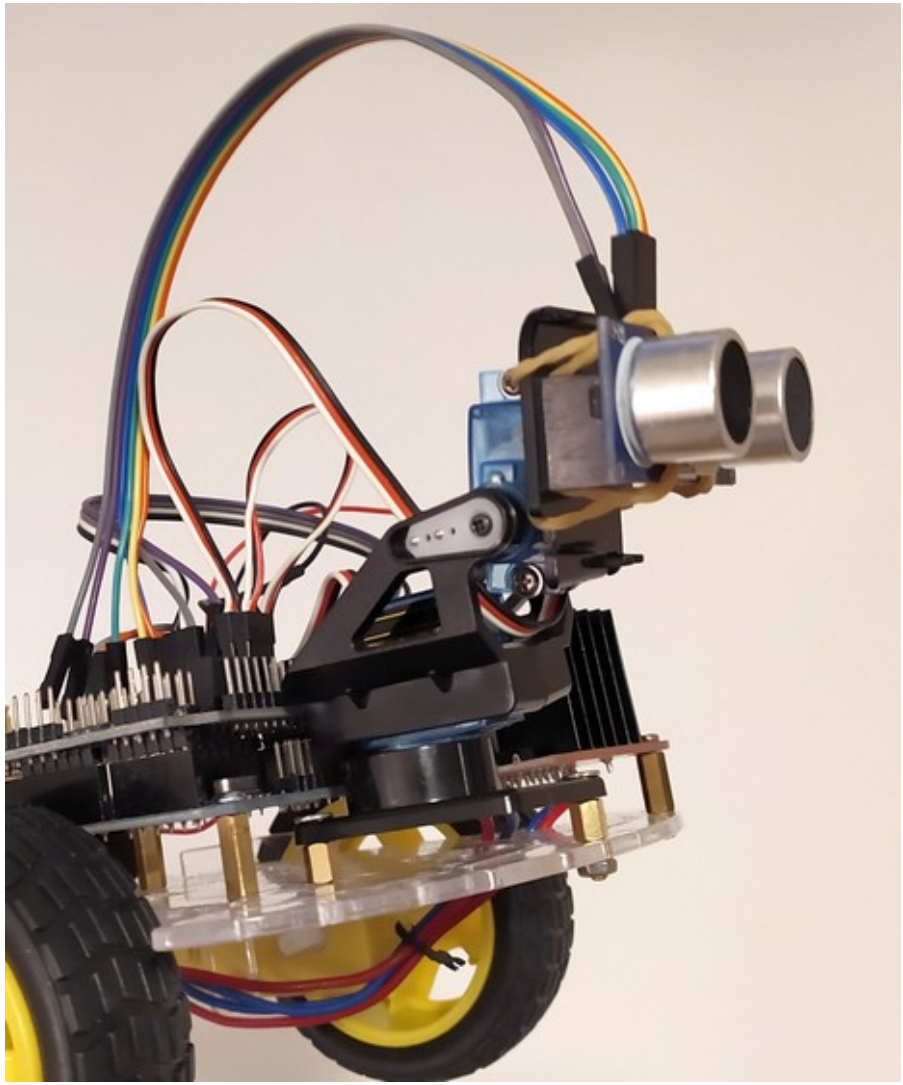
Jelalak

Servo programkönyvtár

- A beépített **Servo** programkönyvtár max. 8 db szervót kezel, s felhasználja a **Timer0** időzítőt, tehát az közben más célra nem használható.
- A programkönyvtár használatához be kell csatolni a **Servo.h** állományt és példányosítani kell a **Servo** osztályt (pl. `Servo myservo;`)
- **Servo.attach(***pin[, minvalue, maxvalue]***)** – hozzárendeli a **Servo** objektumot a megadott kivezetéshez. A mikroszekundumokban mért határértékek (minimális és maximális impulzusszélesség) megadása nem kötelező. Az alapértelmezett értékek: 544 és 2400 µs.
- **Servo.write(***adat***)** – a szervo beállítása (ha az adat < 200, akkor fokokban értendő, s a 0 – 180 tartományban vehet fel értéket, 200-nál nagyobb számok estén mikroszekundumban adható meg az impulzusszélesség)
- **Servo.writeMicroseconds(***adat***)** – a szervo beállítása (az adat itt mikroszekundumokban értendő, az inicializálásnál megadott *minvalue – maxvalue* tartományban)

servo_test: mozgatás két szervóval

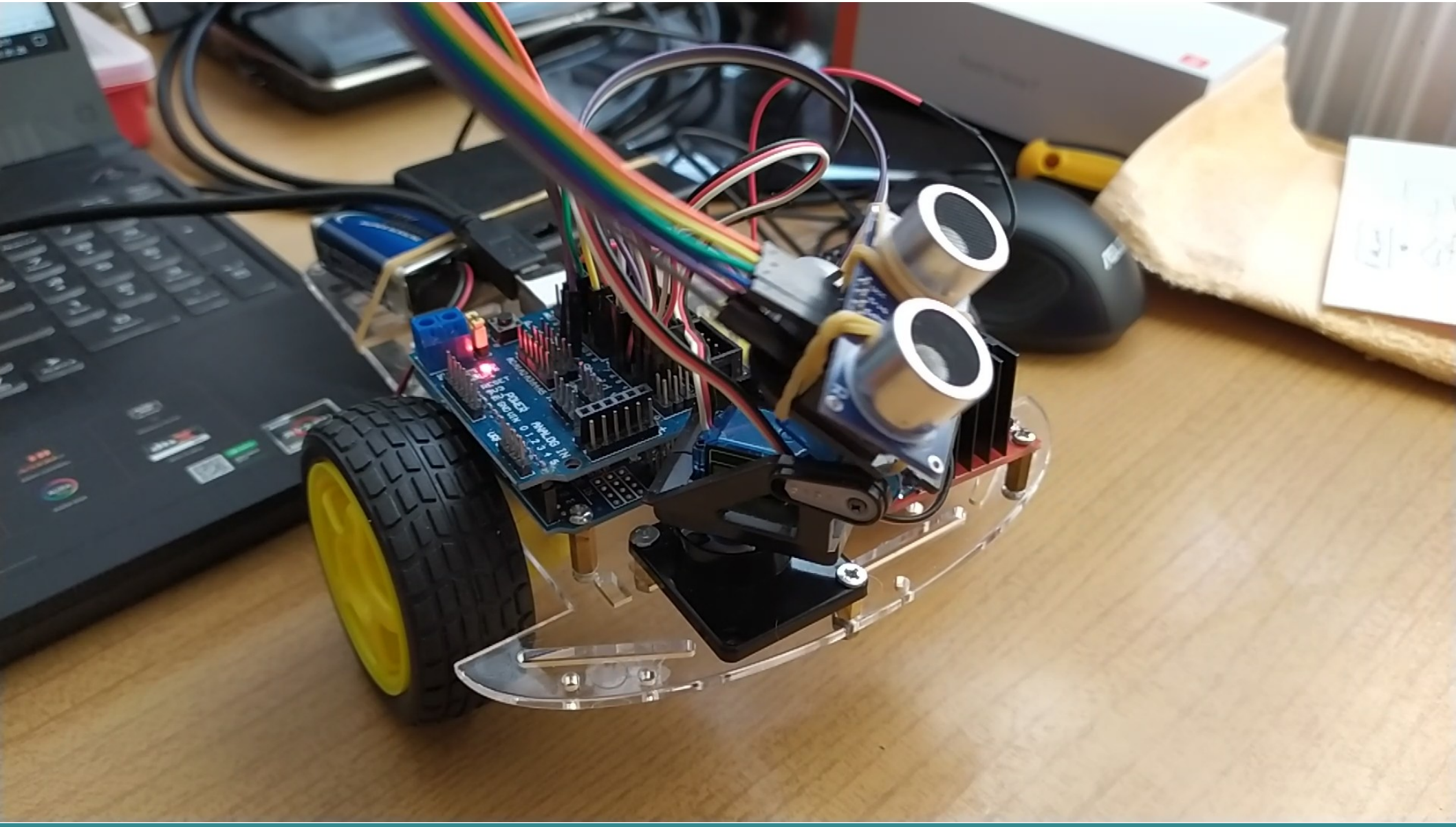
- Az előző előadásban **Timer1** két **PWM** csatornájának felhasználásával vezéreltük a két szervóval mozgatott (elforgatható és billegtethető) tartót
- Most a **Servo** programkönyvtár segítségével fogjuk mozgatni (ehhez fogjuk majd rögzíteni az ultrahangos távolságérzékelőt)
- A **Servo** objektumosztály példányosításánál két példányra lesz szükség:
  - servo1: a forgatáshoz
  - servo2: a billentéshez
- Pásztázási tartományok:
  servo1: 60° – 160°
  servo2: 90° – 160°



servo_test.ino

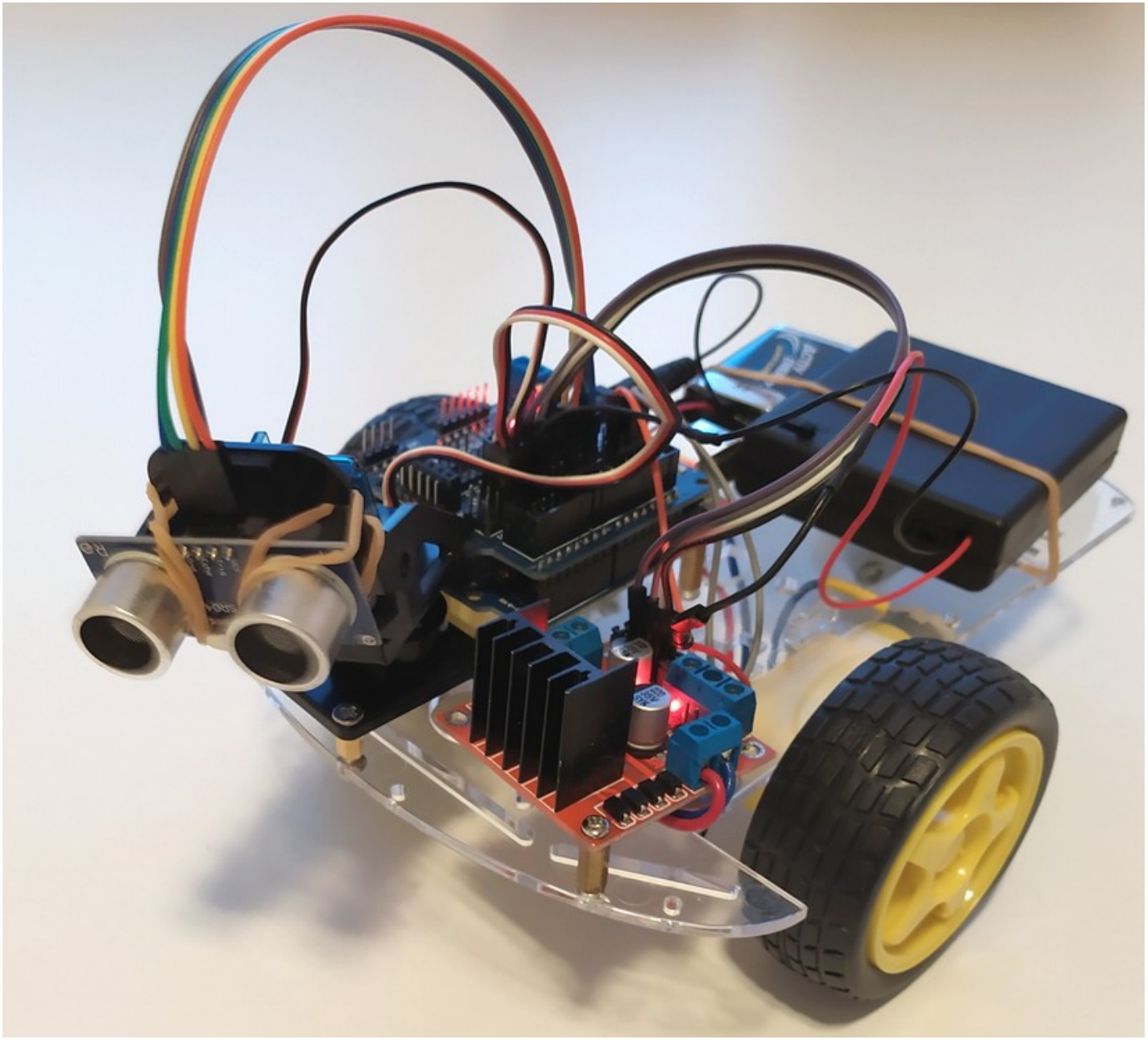
```
#include <Servo.h>
Servo servo1;                                   // Jobbr-balra forgatás
Servo servo2;                                   // le-fel forgatás
int pos = 0;                                    // a pozíciót tartalmazó változó

void setup() {
  servo1.attach(2);                             // Szervó 1. a D2 kimenetre van kötve
  servo2.attach(3);                             // Szervó 2. a D3 kimenetre van kötve
}

void loop() {
  for (pos = 60; pos <= 160; pos += 5) { // 60-tól 160 fokig forgatjuk, 5 fokonként
    servo1.write(pos);     delay(30);     // szervó beállítása az adott pozícióba
  }
  for (pos = 160; pos >= 60; pos -= 5) { // 160-tól 60 fokig lépked visszafelé
    servo1.write(pos);      delay(30);    // szervó beállítása az adott pozícióba
  }
  Delay(2000);

  for (pos = 90; pos <= 160; pos += 5) { // 90-től 160 fokig forgatjuk, 5 fokonként
    servo2.write(pos); delay(30);         // szervó beállítása az adott pozícióba
  }
  for (pos = 160; pos >= 90; pos -= 5) { // 160-tól 90 fokig lépked visszafelé
    servo2.write(pos);      delay(30);    // szervó beállítása az adott pozícióba
  }
  delay(2000);
}
```

servo_test.ino tapasztalatok



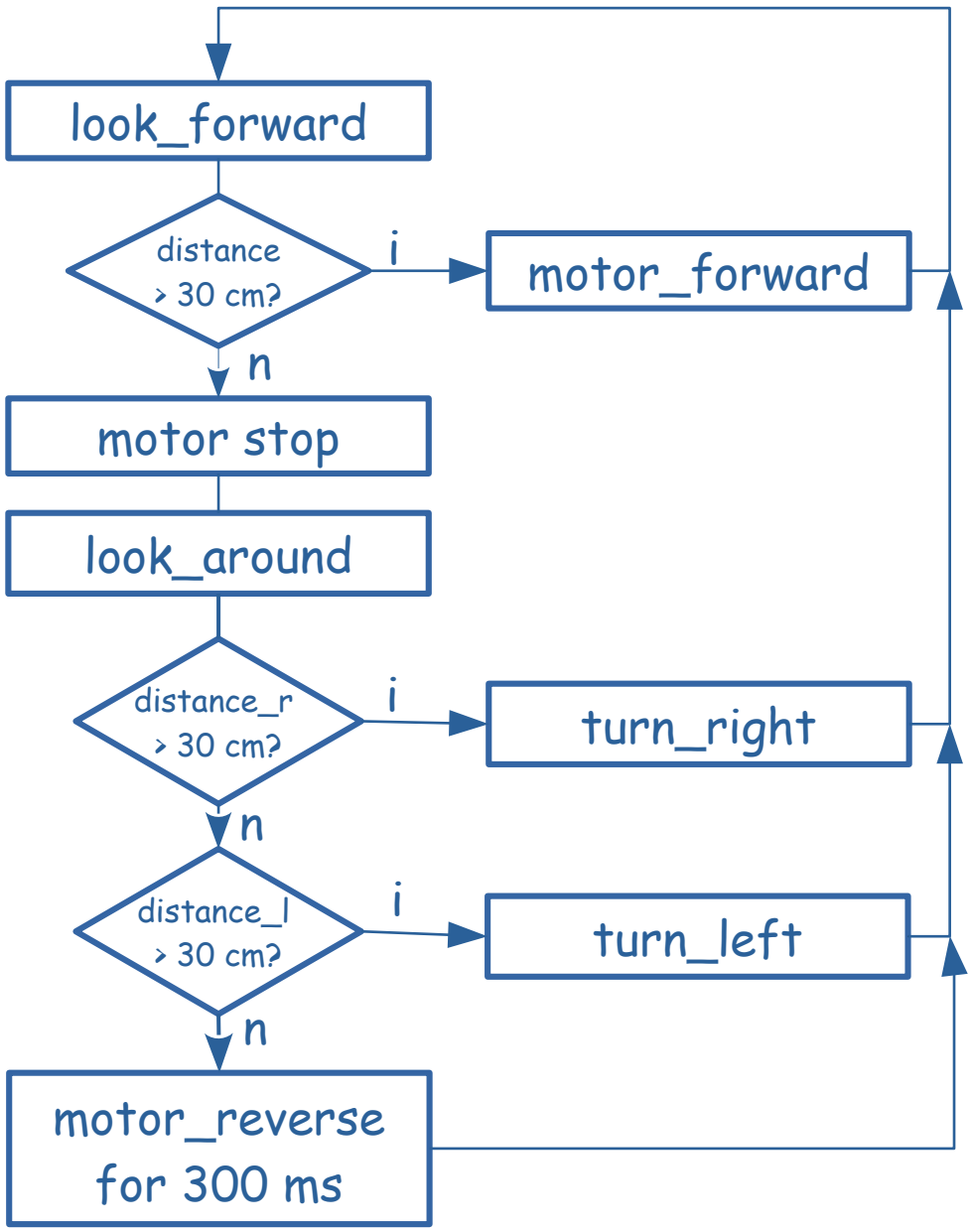
Akadálykikerülő robot

- Két kerék meghajtású (2WD) robotalváz
- Két szervó által mozgatott **HC-SR04** ultrahangos szenzor (az akadályok észlelésére)
- Az ultrahangos szenzor keskeny szögtartományban „lát”, ezért mozgatni kell (a robot forgatja a fejét)



Az akadálykikerülő algoritmus

- Ha elöl szabad az út, előre megyünk
- Ha elöl akadály van, akkor jobbra, vagy balra fordulunk (amerre hely van)
- Ha zsákutcába kerültünk, 300 ms-ig hátrafelé megyünk



akadalykikerulo_robot.ino 3/1.

```
#include <Servo.h>

Servo servo1;           // Jobbr-balra forgatás
Servo servo2;           // le-fel forgatás

#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 trigPin  8      // D8  kivezetés
#define echoPin 11      // D11 kivezetés
#define minDistance 30 // Legkisebb távolság cm-ben
int distance_f;         // szabad út elől
int distance_r;         // szabad út jobbra
int distance_l;         // szabad út balra

void setup() {
  // Szervók konfigurálása
  servo1.attach(2);             // Szervó1 a D2 kimenetre
  servo1.write(90);             // Középső állásban legyen
  servo2.attach(3);             // Szervó2 a D3 kimenetre
  servo2.write(70);             // Középső állásban legyen
  // Szonár konfigurálás
  pinMode(trigPin, OUTPUT);    // trigPin kimenet
  digitalWrite(trigPin, LOW);
  pinMode(echoPin, INPUT);     // echoPin bemenet
  . . .
```

```
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);
}
```

akadalykikerulo_robot.ino 3/2.

```
  pinMode(B1A, OUTPUT); // Motorvezérlő lábak
  pinMode(B1B, OUTPUT);
  pinMode(A1A, OUTPUT);
  pinMode(A1B, OUTPUT);
  digitalWrite(B1A, LOW);
  digitalWrite(B1B, LOW);
  digitalWrite(A1A, LOW);
  digitalWrite(A1B, LOW);
  delay(5000);
}

uint16_t ping(int angle) {
  servo1.write(angle);
  delay(100);
  digitalWrite(trigPin, HIGH);
  delayMicroseconds(10);
  digitalWrite(trigPin, LOW);
  long duration = pulseIn(echoPin, HIGH);
  uint16_t distance=(duration>0)? (duration/58): 200;
  return distance;
}

void look_around() {
  servo1.write(50); delay(200);
  distance_l = ping(50);
  servo1.write(150); delay(200);
  distance_r = ping(160);
}
```

```
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);
}
```

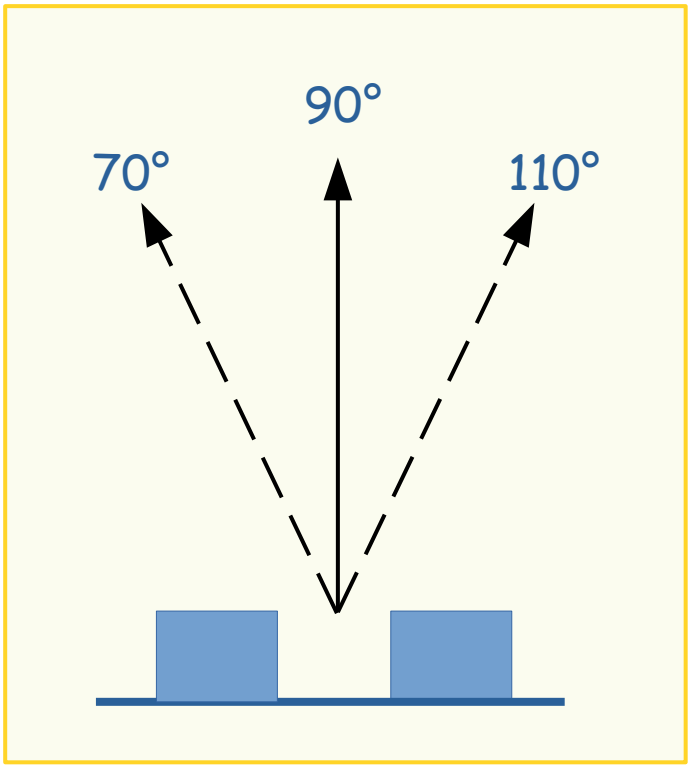
akadalykikerulo_robot.ino 3/3.

```
int look_forward() {
  int temp2 = ping(90);                          // Távolságmérés előre nézve
  return temp2;
}

void loop() {
  distance_f = look_forward();                   // Mekkora szabad táv van előttünk?
  if (distance_f > minDistance) {
    motor_forward();                             // Előre menet
  } else {
    motor_stop();                                // Hoppá!
    look_around();                               // Útkeresés jobbra-balra
    if (distance_r > minDistance) {              // Ha jobbra mehetünk
      motor_right();                             // Fordulj jobbra!
      delay(300);
      motor_stop();
    } else if (distance_l > minDistance) {       // Ha csak balra mehetünk
      motor_left();                              // Fordulj balra!
      delay(300);
      motor_stop();
    } else {                                     // Zsákutca esetén
      motor_reverse();                           // Menjünk vissza egy kicsit
      delay(300);
      motor_stop();
    }
  }
}
```

look_forward – kicsit másképp

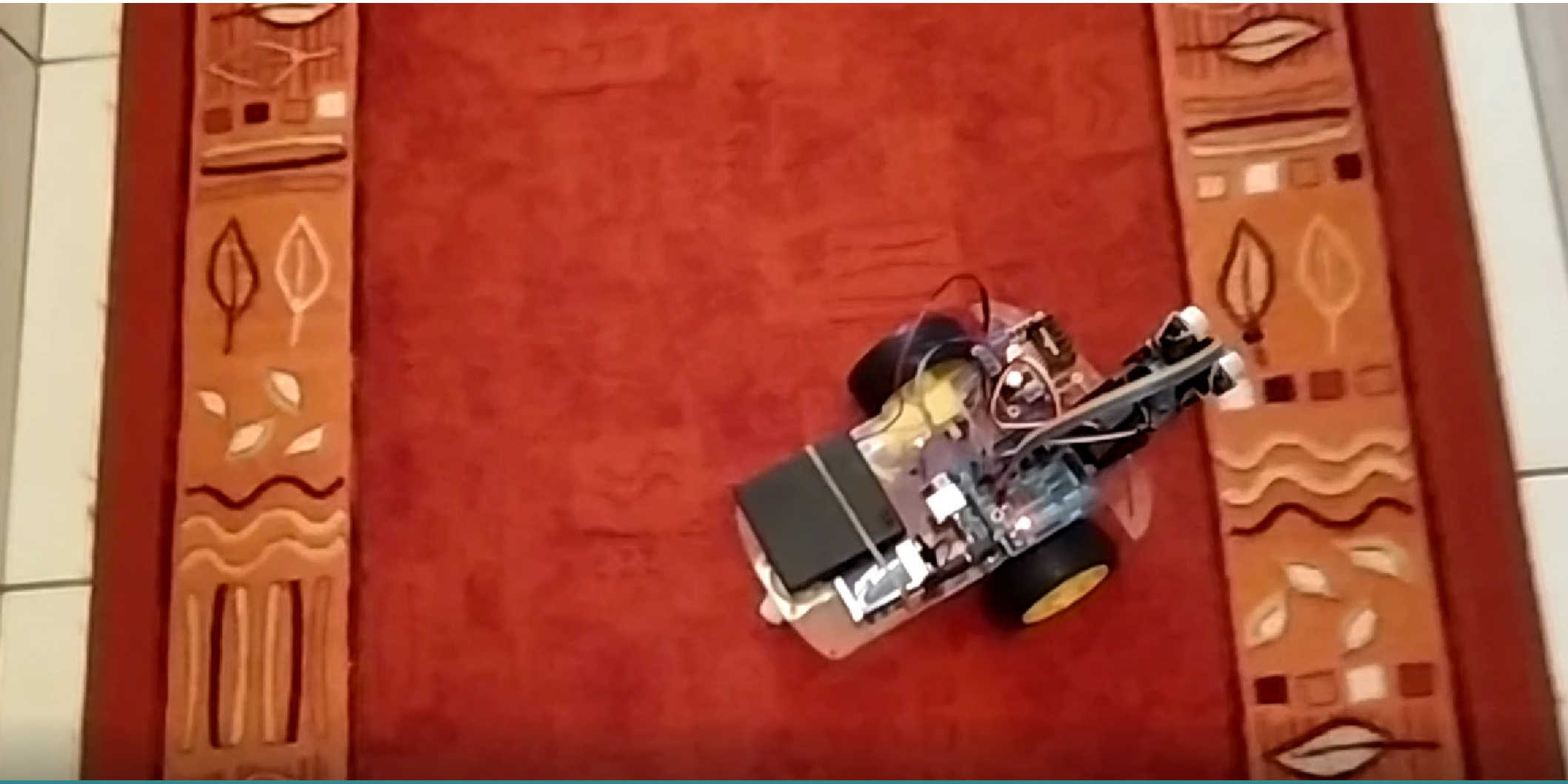
- A nem teljesen frontális akadályok „körültekintőbb” felmérésével próbálkoztunk az alábbi módon:
- Ez előre iránytól (90°-os szervó állás) jobbra és balra is mérünk távolságot, azaz kiszélesítjük a szenzor 16°-os észlelési szögtartományát)
- A három távolság minimumát tekintjünk mérvadónak
- Sajnos ez az algoritmus fölöslegesen sok fejmozgatással jár

```
int look_forward() {
  int temp1 = ping(70);
  int temp2 = ping(90);
  int temp3 = ping(110);
  int temp4 = min(temp1, temp2);
  return min(temp3, temp4);
}
```



akadalykikerulo_robot.ino tapasztalatok

- Hátramenet után körbe kellene nézni
- Előremenetnél a szenzor mozgatás helyett több szenzor kellene



Az Arduino nano kártya kivezetései