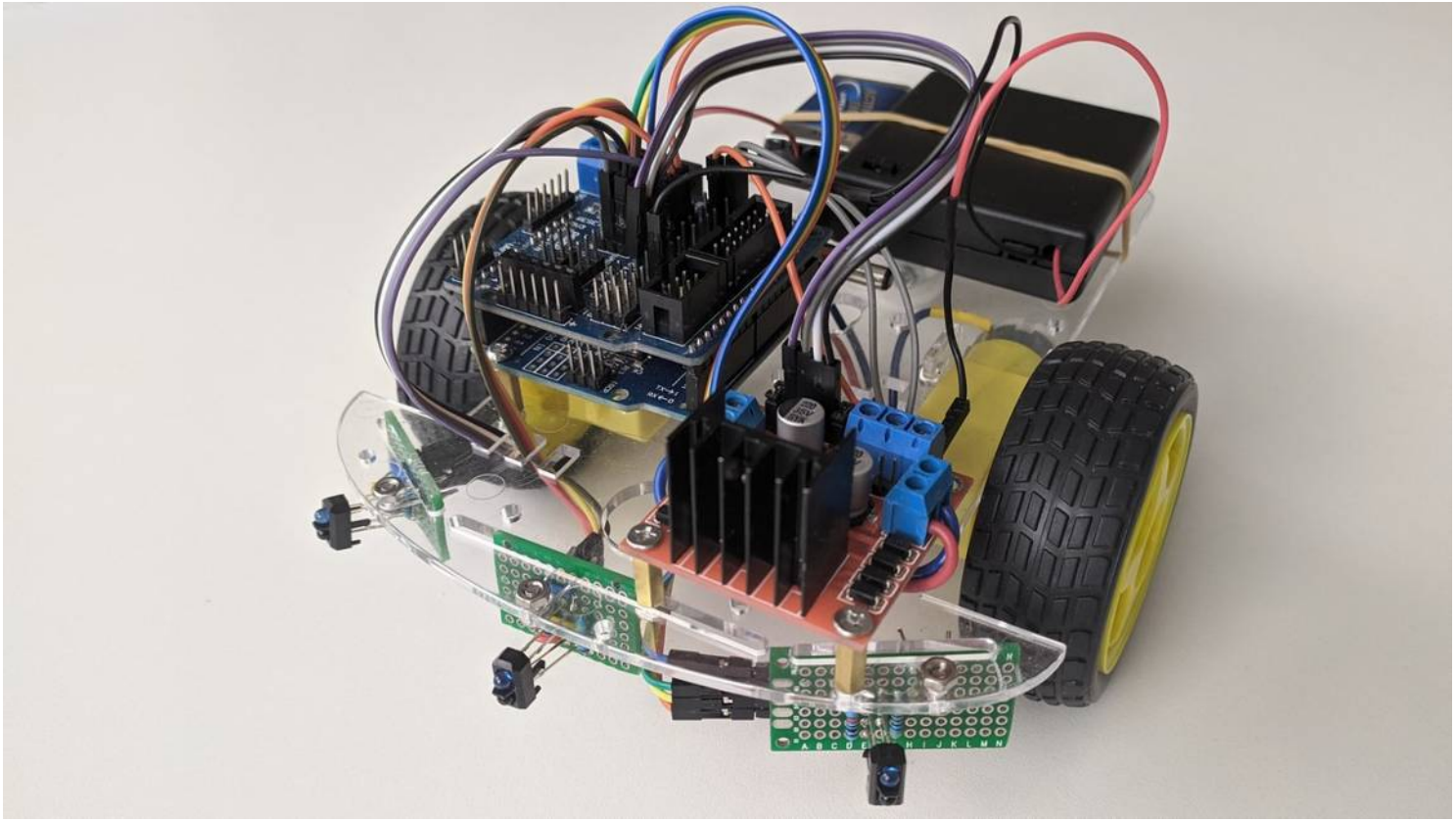
Vegyes témakörű előadások



Motorvezérlés - bevezetés a robotikába

A korábbi előadások jegyzéke

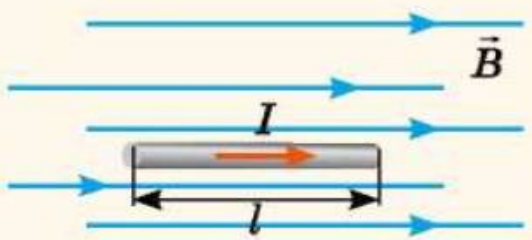
- Kisteljesítményű egyenáramú motorok vezérlése (2015. december 17.) előadásvázlat mintaprogramok
- Robotvezérlés sebességmérő szenzorral (2019. január 31.) előadásvázlat mintaprogramok
- Robotvezérlés WiFi kapcsolaton keresztül 1. rész (2019. február 14.) előadásvázlat mintaprogramok
- Robotvezérlés WiFi kapcsolaton keresztül 2. rész (2019. február 28.) előadásvázlat mintaprogramok
- Optoérzékelők, motorvezérlés, bevezetés a robotikába (2020. márc. 19.) előadásvázlat mintaprogramok video
- Motorvezérlés, bevezetés a robotikába (2021. január 28.) előadásvázlat mintaprogramok video
- Reflexiós fényérzékelők, akadályelkerülő robot (2021. február 11.) előadásvázlat mintaprogramok video
- Falkövető robot, I2C OLED kijelzők (2021. február 25.) előadásvázlat mintaprogramok video

A témának a technikai részletekre bővebben kitérő ismertetése a fentebb hivatkozott korábbi előadásokban található

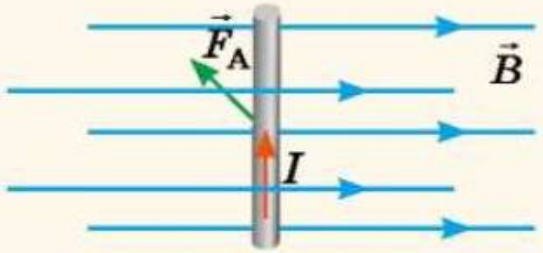
Ampére-féle erő

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

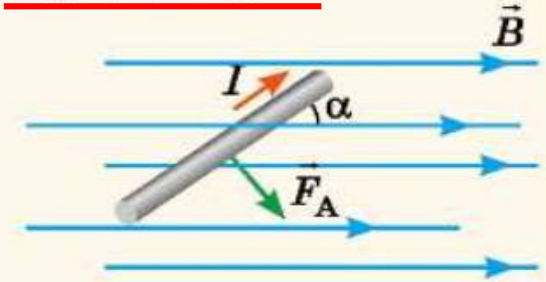
• A vezető párhuzamos az indukcióvektorokkal – a *mágneses tér nem hat a vezetőre*: $F_A=0$



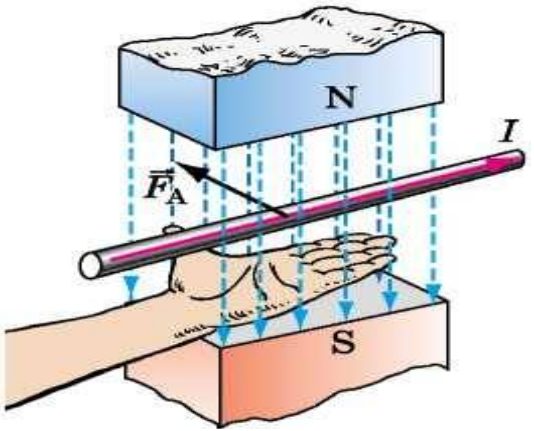
• A vezető merőleges az indukcióvektorokra – az *Ampère-féle erő maximális értéket ér el*: $F_A=BIl$



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



Az Ampére-féle erő iránya

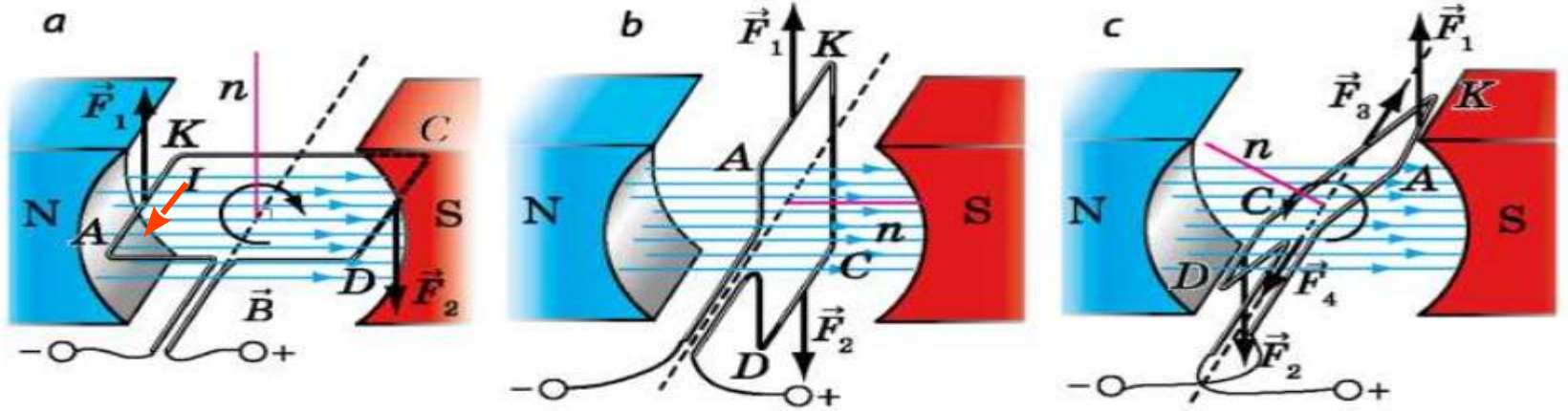


- Ha a mágneses tér indukcióvektorainak eredője a tenyerünkbe hatol, és ujjaink az áram irányát mutatják, akkor az oldalra kinyújtott hüvelykujjunk a vezetőre ható Ampére-féle erő irányát mutatja

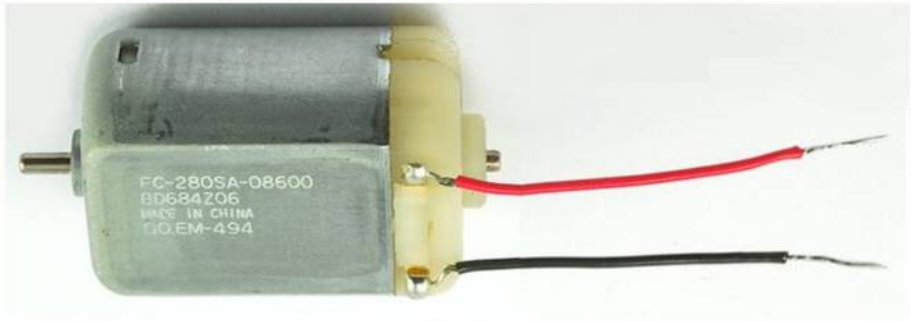
a) Az áramvezető keretre óramutató járása szerinti irányú forgatónyomaték hat

b) Az áramvezető keretre nem hat forgatónyomaték

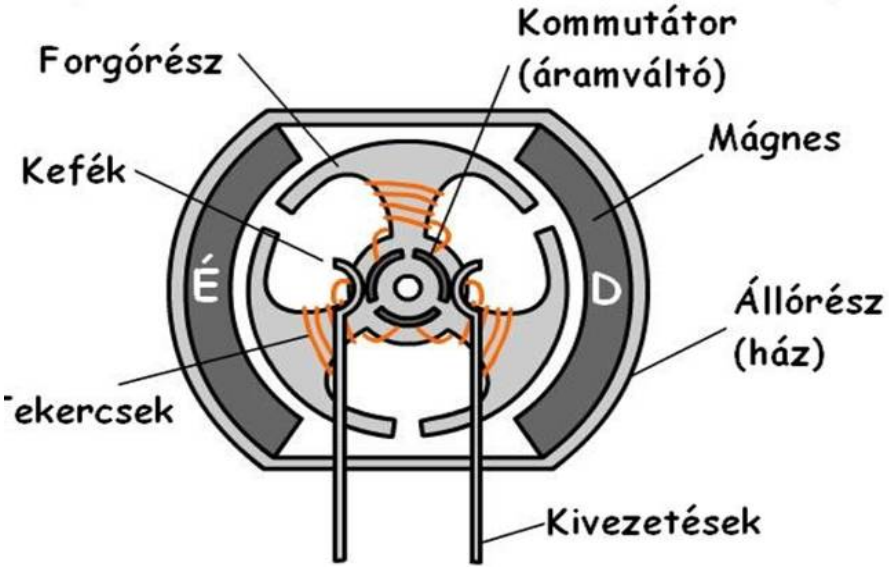
c) Az áramvezető keretre óramutató járásával ellentétes irányú forgatónyomaték hat

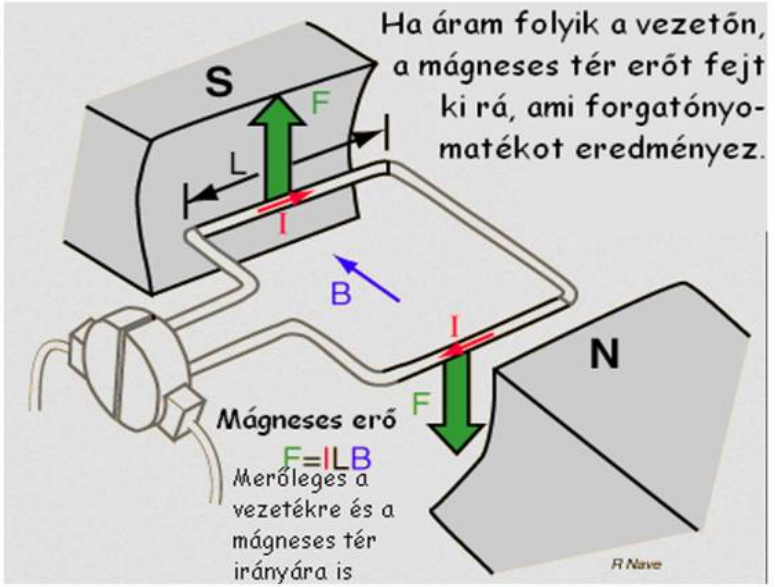


Kisteljesítményű DC motorok



Tipikus kefés motor metszeti képe

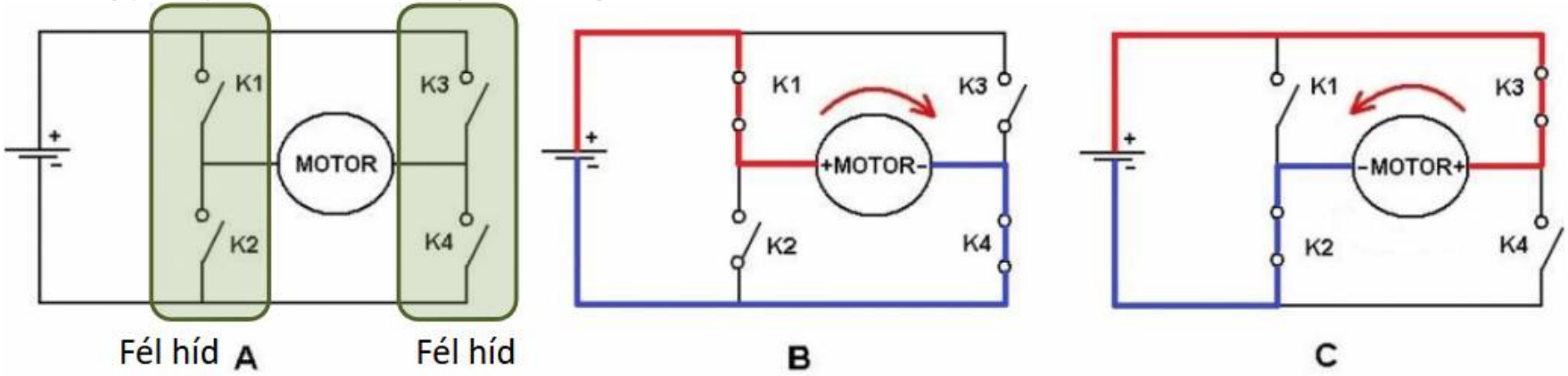




Feszültség: 3 V – 6V
Áram: 0.4 A – 0.6 A
Forgásirány váltás: polaritásváltással

Motorvezérlés: H-híd

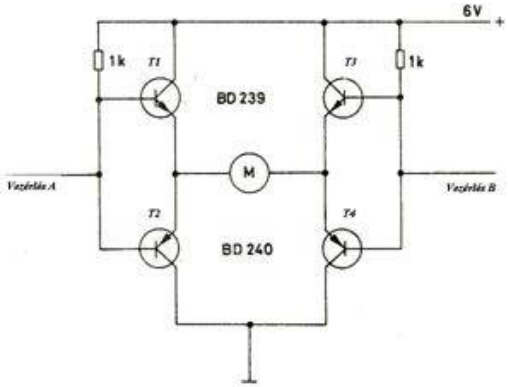
Ha a motor forgásirányát meg akarjuk változtatni, fel kell cserélni a polaritást.
Ezt négy kapcsolóval tudjuk megoldani. Az elrendezést H-hídnak nevezzük.



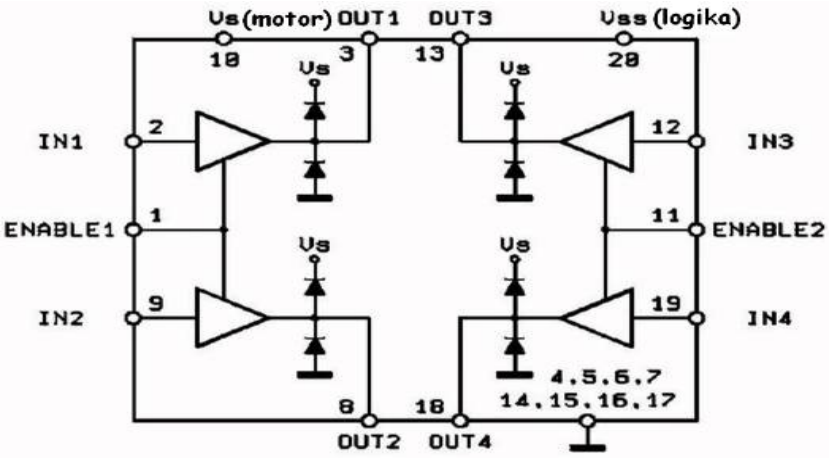
Forrás és leírás: hobbirobot.hu/content/vonalkoveto-i-robotika-kezdoknek

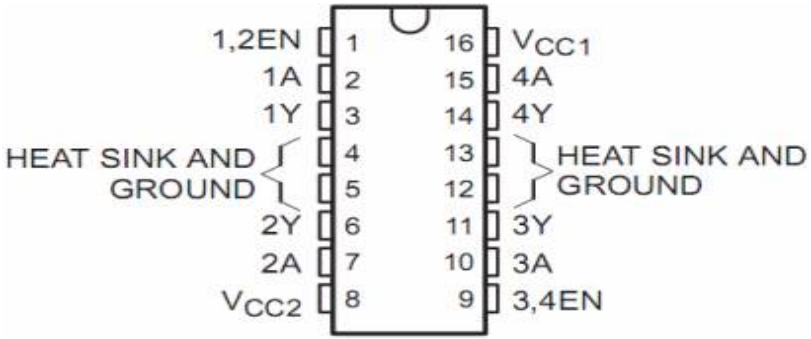
K1 és **K4** zárásakor a motor az egyik irányba forog.
K2 és **K3** zárásakor az ellenkező irányba forog.

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)

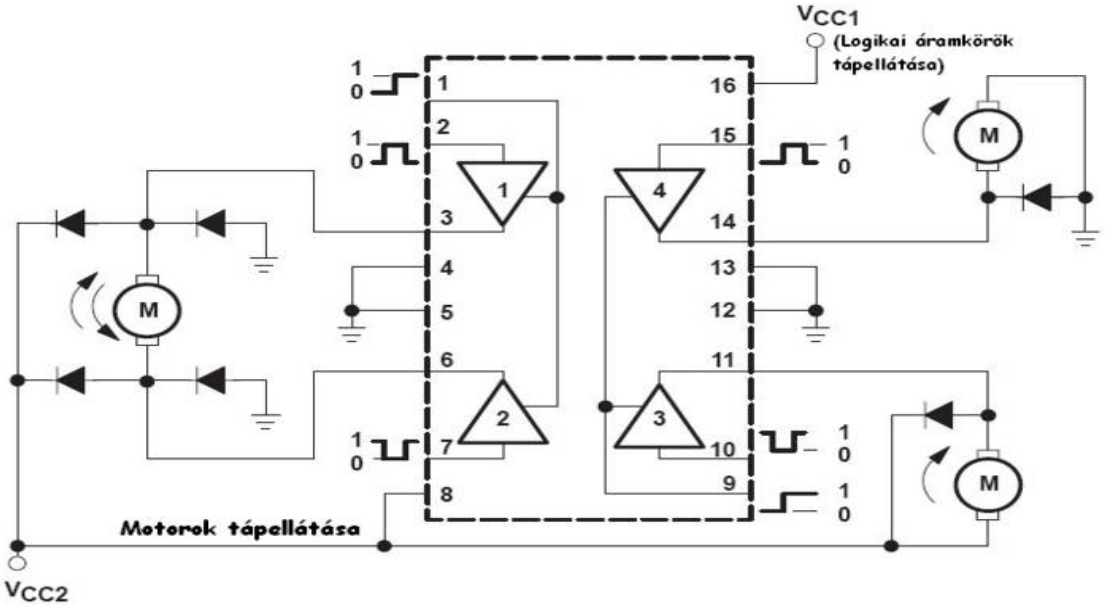
H-híd: polaritásváltást igénylő vezérlésekhez

V_{CC2} = 4,5 – 36 V

V_{CC1},V_A, V_{EN} = 2.3 – 7 V

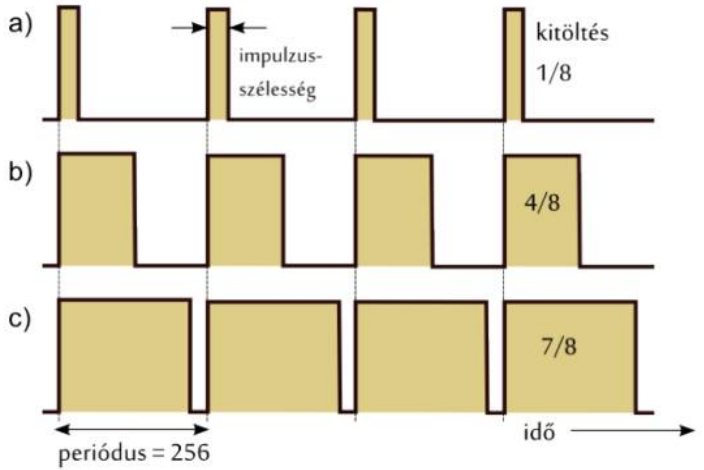
I_{max} = 0,6 A

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

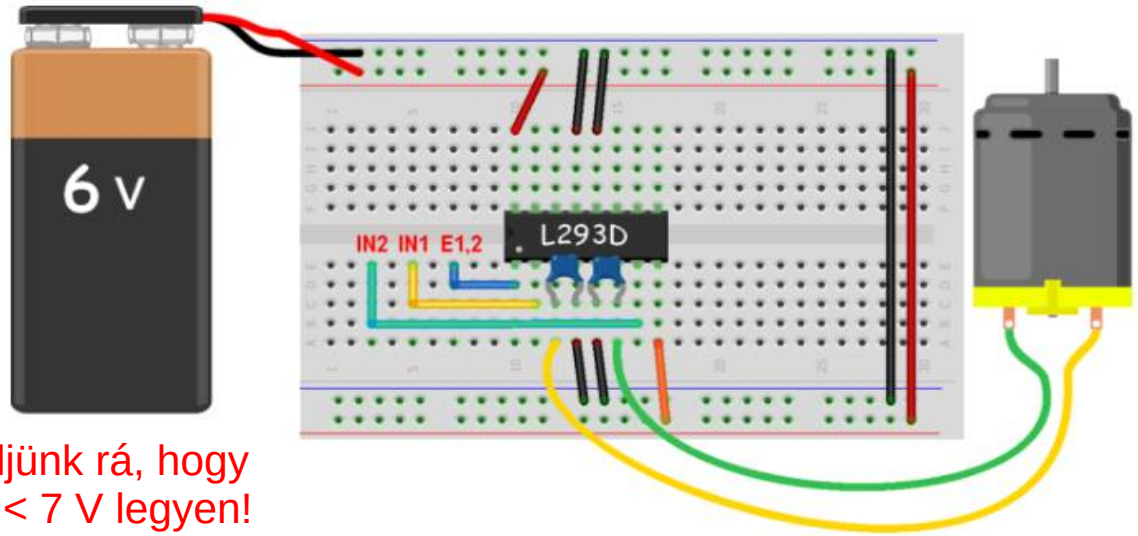
$$f_{PWM} = \frac{16\,MHz}{64 \cdot 510} = 490.19\,Hz$$

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.

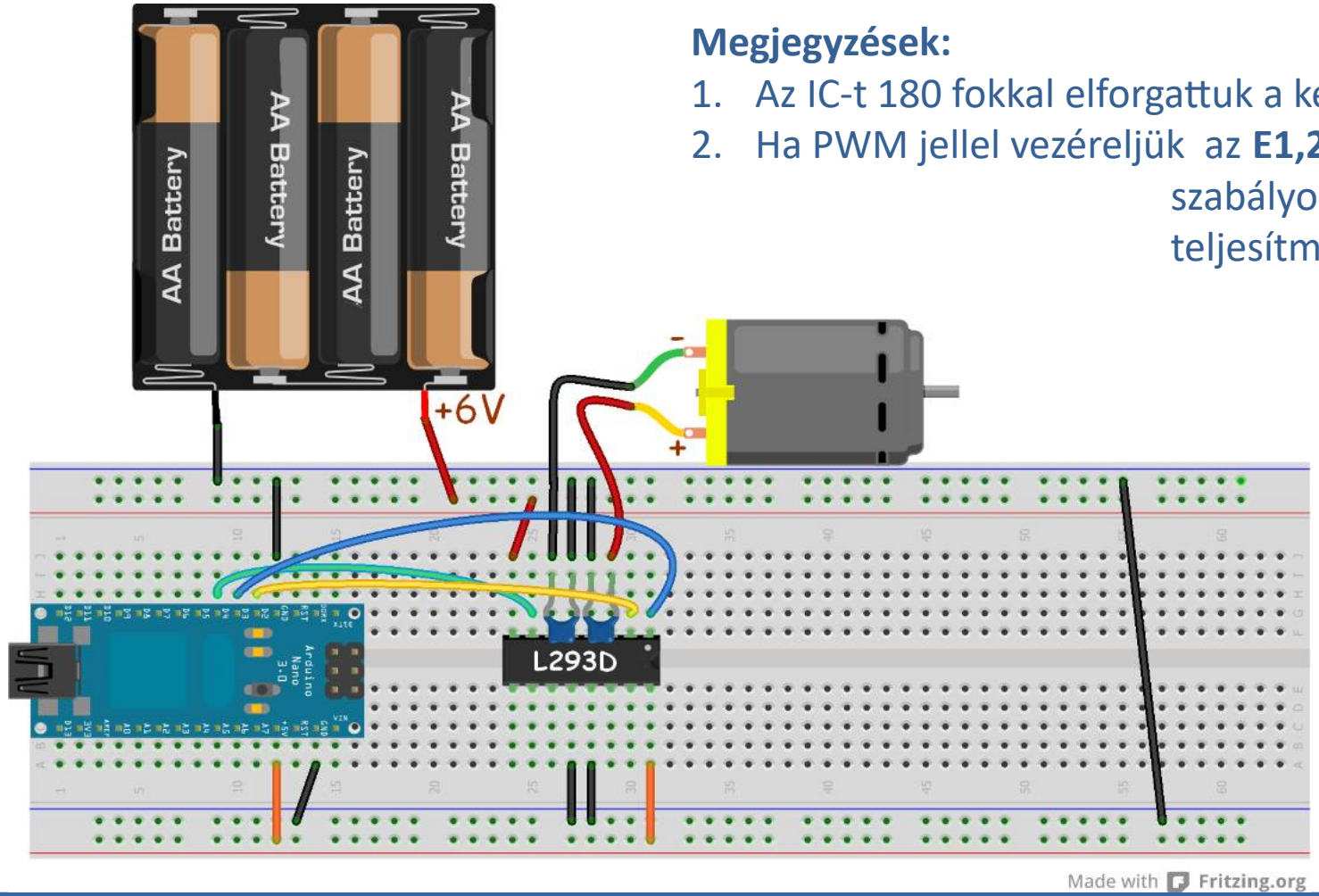


Ügyeljünk rá, hogy
Vcc1 < 7 V legyen!

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



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…

Bekötési vázlat:

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

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.

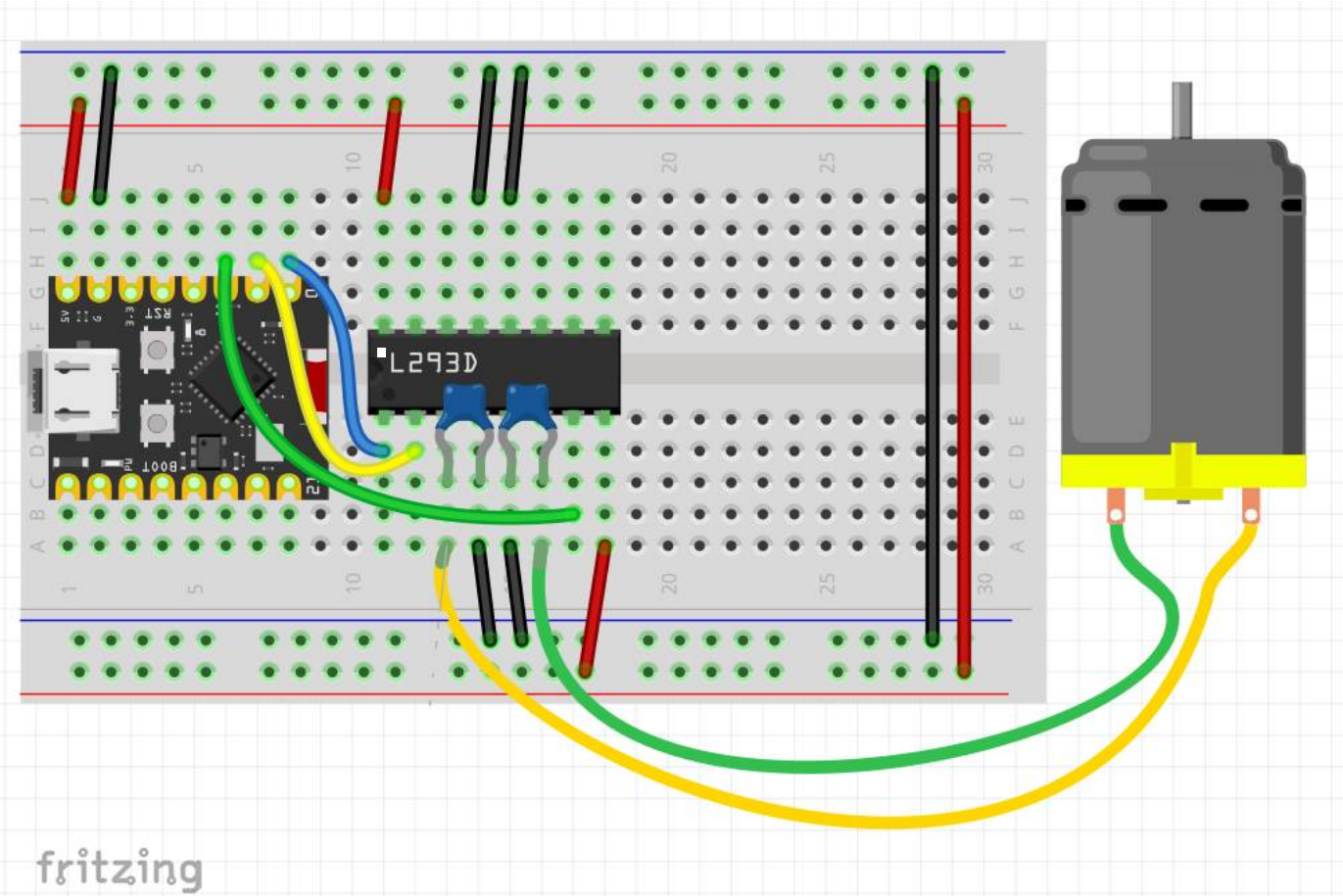
```
#define PWMA    3     //L293D pin1
#define D1A     2     //L293D pin2
#define D2A     4     //L293D pin7


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
}
```

Egy motor vezérlése ESP32-C3 kártyával

1,2E IO0	1A IO1	2A IO2	Motor
H	0	0	Stop
H	1	0	Előremenet
H	0	1	Hátramenet
H	1	1	Stop
L	X	X	Stop

A motorfeszültség (V_{CC2}, az IC pin8 lába) itt közösítve van az 5V-tal, de külön tápellátást is kaphat (4,5 – 36 V között)



L293D_test_1M.py – 2/1.

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
- **ESP32-C3** bármelyik kimenete **PWM** lehet
  A frekvenciát 1000 Hz-re állítottuk, a kitöltés 0 – 65 535 közötti lehet

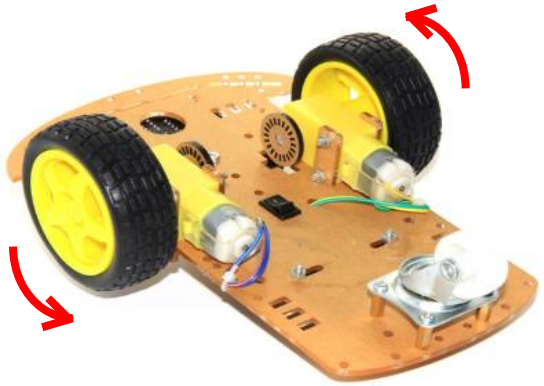
```
import time
import board
import pwmio
import digitalio

# L293D lábkiosztás - ESP32-C3 IO0, IO1, IO2
PWMA = board.IO0   # Enable1 -> PWM
D1A  = board.IO1   # Input1
D2A  = board.IO2   # Input2

# PWM kimenet létrehozása
pwm = pwmio.PWMOut(PWMA, frequency=1000, duty_cycle=0)

# Digitális kimenetek létrehozása
d1 = digitalio.DigitalInOut(D1A)
d1.direction = digitalio.Direction.OUTPUT

d2 = digitalio.DigitalInOut(D2A)
d2.direction = digitalio.Direction.OUTPUT
```

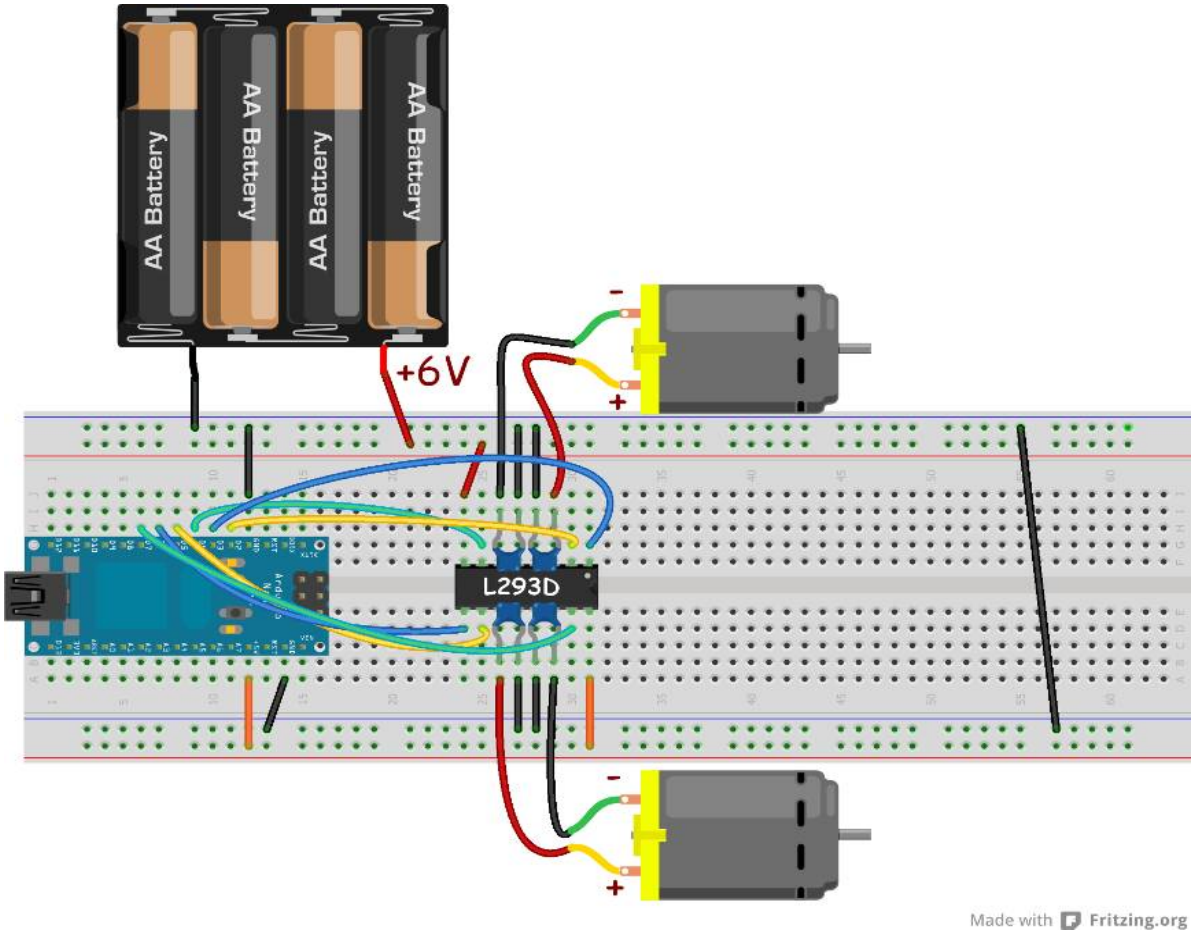
L293D_test_1M.py – 2/2.

```
def set_motor(speed_percent, reverse):
    """ speed_percent: 0-100 közötti érték (sebesség százalékban)
    reverse: True = hátra, False = előre    """
    # Skálázás 16 bites duty cycle-re
    duty = int((speed_percent / 100.0) * 65535)
    pwm.duty_cycle = duty
    d1.value = not reverse
    d2.value = reverse

while True:
    time.sleep(5)
    print("move forward 75%")
    set_motor(75, False)   # előre 75%
    time.sleep(2)
    set_motor(0, False)    # stop
    time.sleep(1)
    print("move reverse 75%")
    set_motor(75, True)    # hátra 75%
    time.sleep(2)
    set_motor(0, True)     # stop
```

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!





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

L293D_test_2M.ino

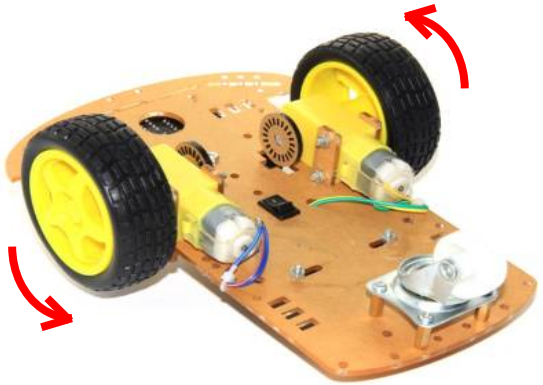
- Két motor vezérlése. A végtelen ciklusban 5s várakozás után 2s előremenet, 1s várakozás, majd 2s hátramenet ismétlődik
- Az előre- és hátramenet 75 %-os teljesítménnyel történik

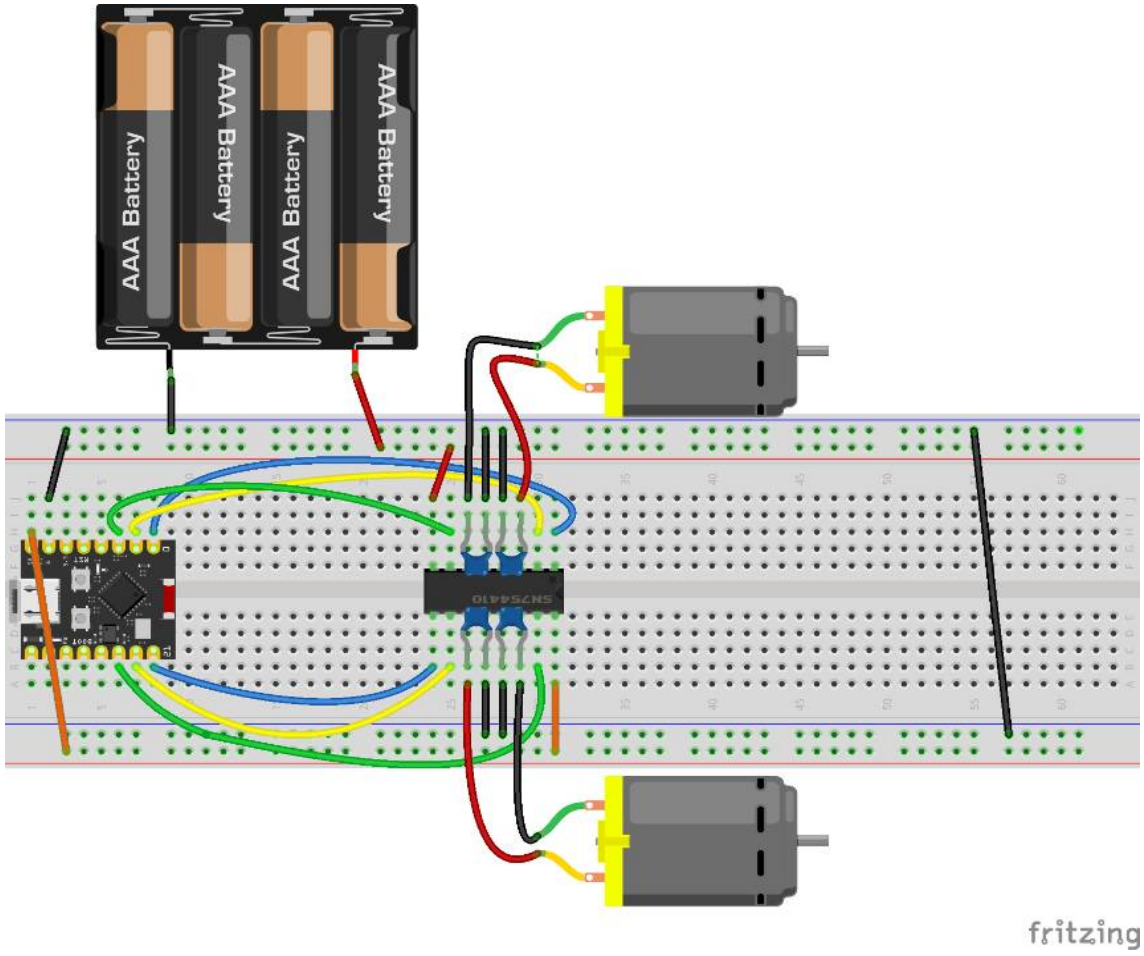
```
#define PWMA    3     //L293D pin1
#define D1A     2     //L293D pin2
#define D2A     4     //L293D pin7
#define PWMB    6     //L293D pin9
#define D3B     5     //L293D pin10
#define D4B     7     //L293D pin15

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
}
```

Két motor vezérlése ESP32-C3 kártyával

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!





L293D_test_2M.py – 2/1.

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
- **ESP32-C3** bármelyik kimenete **PWM** lehet
  A frekvenciát 1000 Hz-re állítottuk, a kitöltés 0 – 65 535 közötti lehet

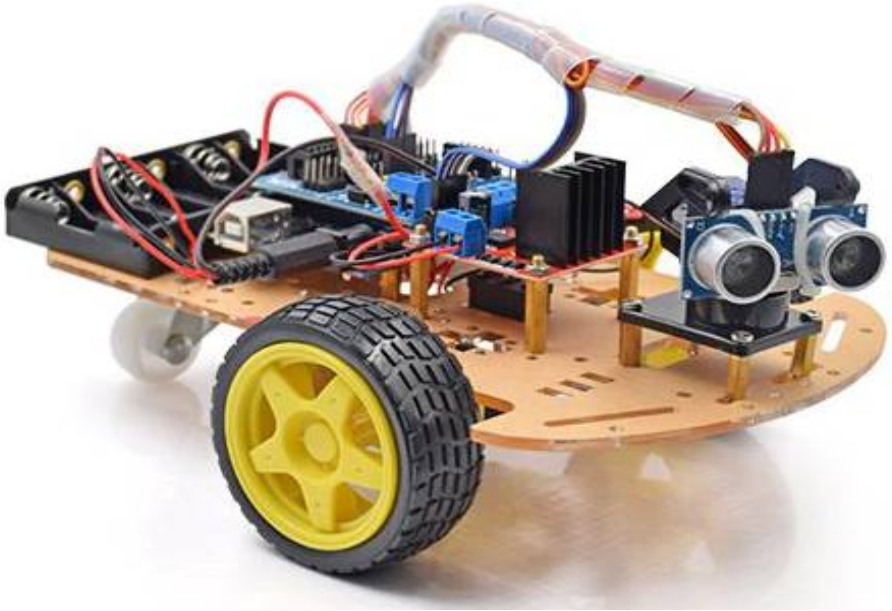
```
import time, board, pwmio, digitalio

# --- L293D lábkiosztás az ESP32-C3-on ---
PWMA = board.IO0    # L293D pin1
D1A  = board.IO1    # L293D pin2
D2A  = board.IO2    # L293D pin7
PWMB = board.IO21   # L293D pin9
D3B  = board.IO20   # L293D pin10
D4B  = board.IO10   # L293D pin15
# --- PWM inicializálás 1000 Hz frekvenciával ---
pwmA = pwmio.PWMOut(PWMA, frequency=1000, duty_cycle=0)
pwmB = pwmio.PWMOut(PWMB, frequency=1000, duty_cycle=0)
# --- Digitális irányvezérlő lábak ---
d1a = digitalio.DigitalInOut(D1A)
d1a.direction = digitalio.Direction.OUTPUT
d2a = digitalio.DigitalInOut(D2A)
d2a.direction = digitalio.Direction.OUTPUT
d3b = digitalio.DigitalInOut(D3B)
d3b.direction = digitalio.Direction.OUTPUT
d4b = digitalio.DigitalInOut(D4B)
d4b.direction = digitalio.Direction.OUTPUT
```

L293D_test_2M.py – 2/2.

```
def set_motor(speed: int, reverse: bool):
    """ speed: 0-100 közötti érték (százalék)
    reverse: True = hátramenet, False = előremenet   """
    # Skálázás 16 bites duty_cycle értékre (0-65535)
    duty = int((speed / 100.0) * 65535)
    pwmA.duty_cycle = duty
    pwmB.duty_cycle = duty
    d1a.value = not reverse
    d2a.value = reverse
    d3b.value = not reverse
    d4b.value = reverse
# --- Főprogram ---
print("\nL293D test 2M program (CircuitPython, ESP32-C3)")
while True:
    time.sleep(5)
    print("move forward 75%")
    set_motor(75, False)   # előre 75%
    time.sleep(2)
    set_motor(0, False)    # stop
    time.sleep(1)
    print("move reverse 75%")
    set_motor(75, True)    # hátra 75%
    time.sleep(2)
    set_motor(0, False)    # stop
```

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

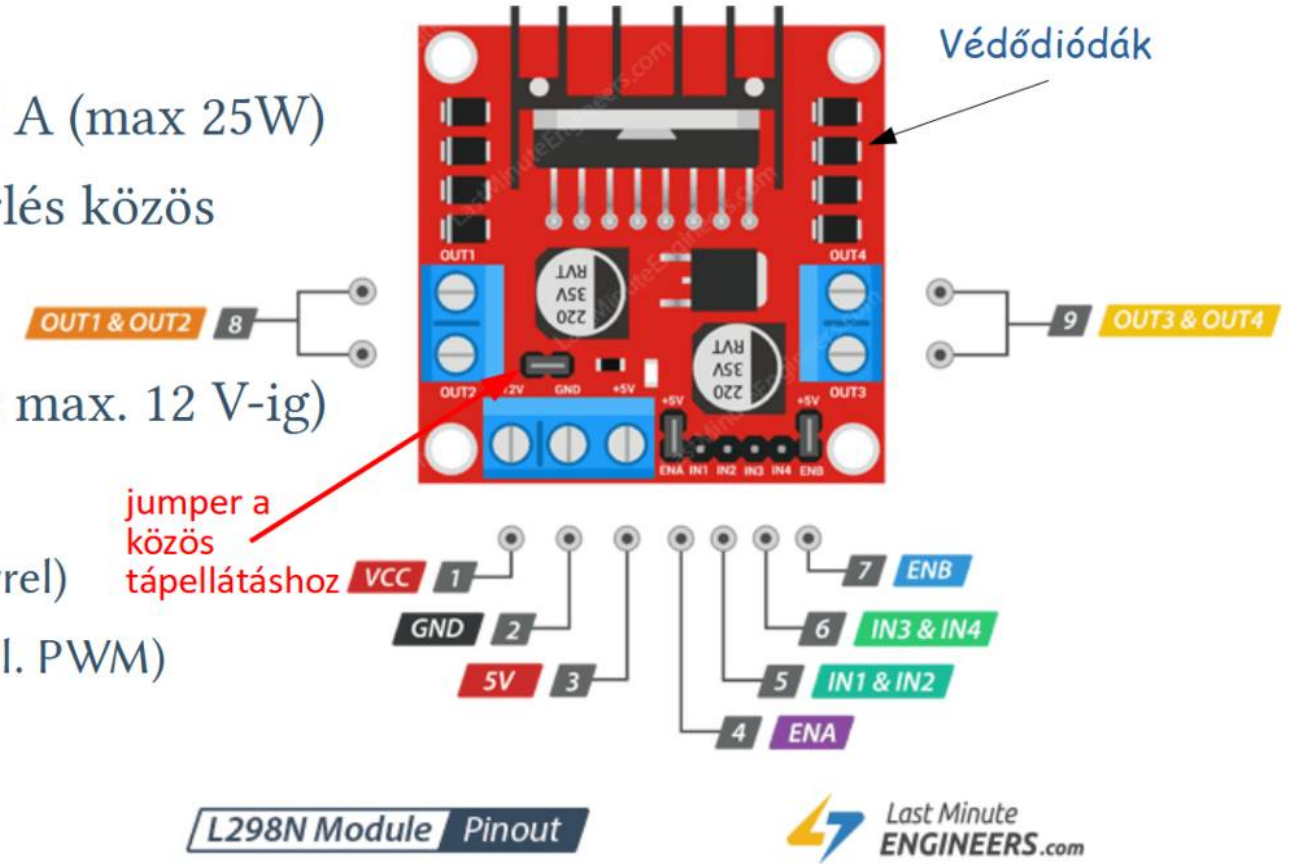


A robot alváz és a motorok

- 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

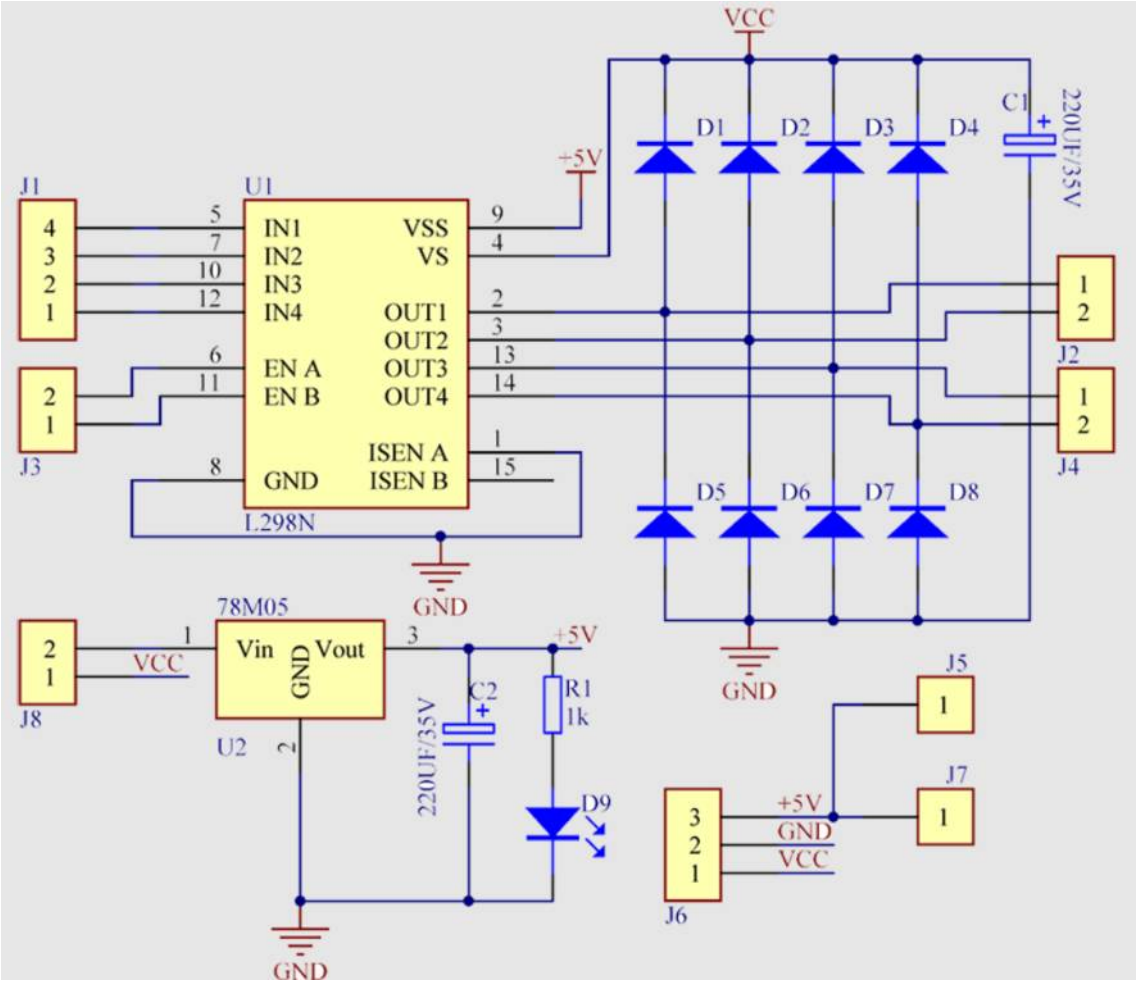
A motorvezérlő kártya

- 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)



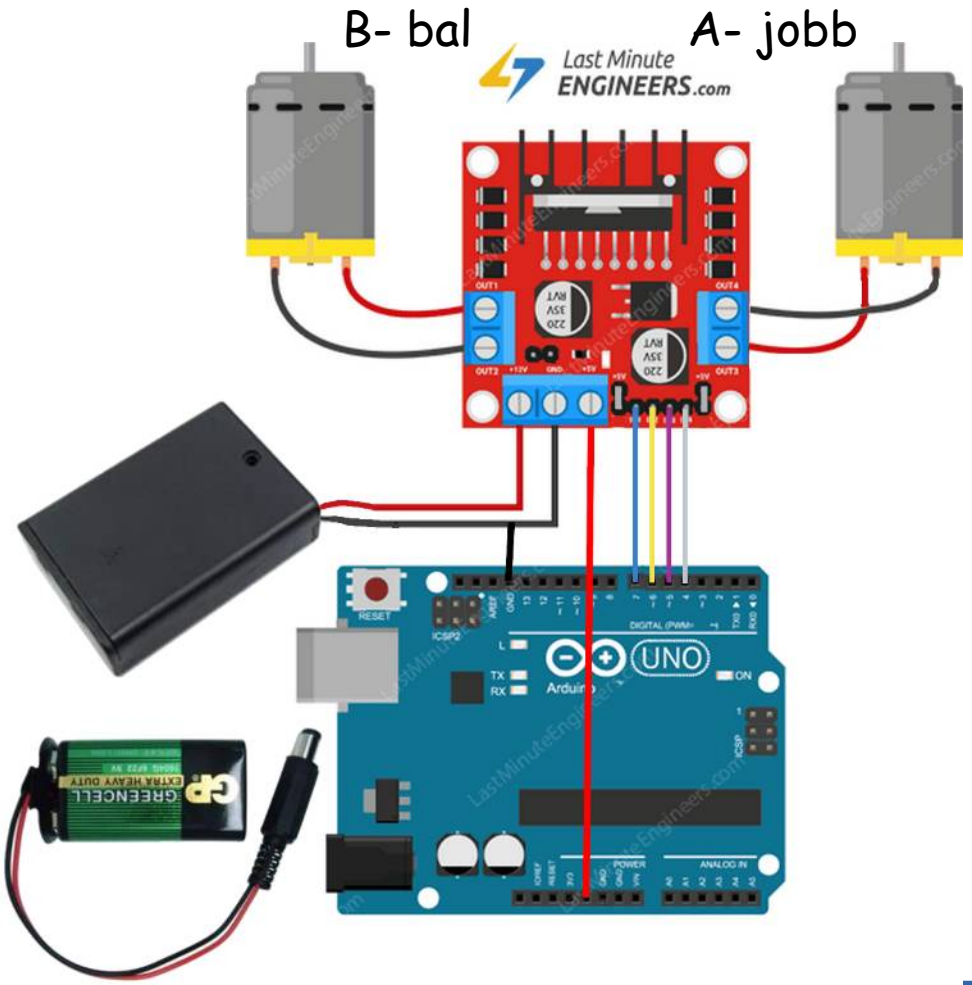
- Forrás: lastminuteengineers.com/l298n-dc-stepper-driver-arduino-tutorial/

A motorvezérlő kártya áramköri vázlata



Az általunk megépített kapcsolás

- A motorok táplálását négy AA ceruzaelemmel oldottuk meg
- Az Arduino kártya egy 9 V-os elemről kap táplálást és a motor-vezérlő elektronika +5 V feszültségét az Arduino kártya biztosítja
- A motorok tükörszimmetrikus bekötése hardveresen biztosítja az ellentétes irányú forgást a bal és jobb motornál
- Az egyszerűség kedvéért itt most nem használtunk PWM teljesítményvezérlést (ENA és ENB állandó magas szintet kapnak)



motor_test.ino – 2/1.

- Ez az egyszerű kis Arduino 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 – 2/2.

```
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);
}
```

motor_test.py – 2/1.

- A **motor_test** programot ESP32-C3 kártyára is átírtuk, CircuitPython nyelven
- Bekötés:

L298N		ESP32	Funkció
IN1	B1A	IO3	Bal motor előre
IN2	B1B	IO2	Bal motor hátra
IN3	A1A	IO1	Jobb motor előre
IN4	A1B	IO0	Jobb motor hátra

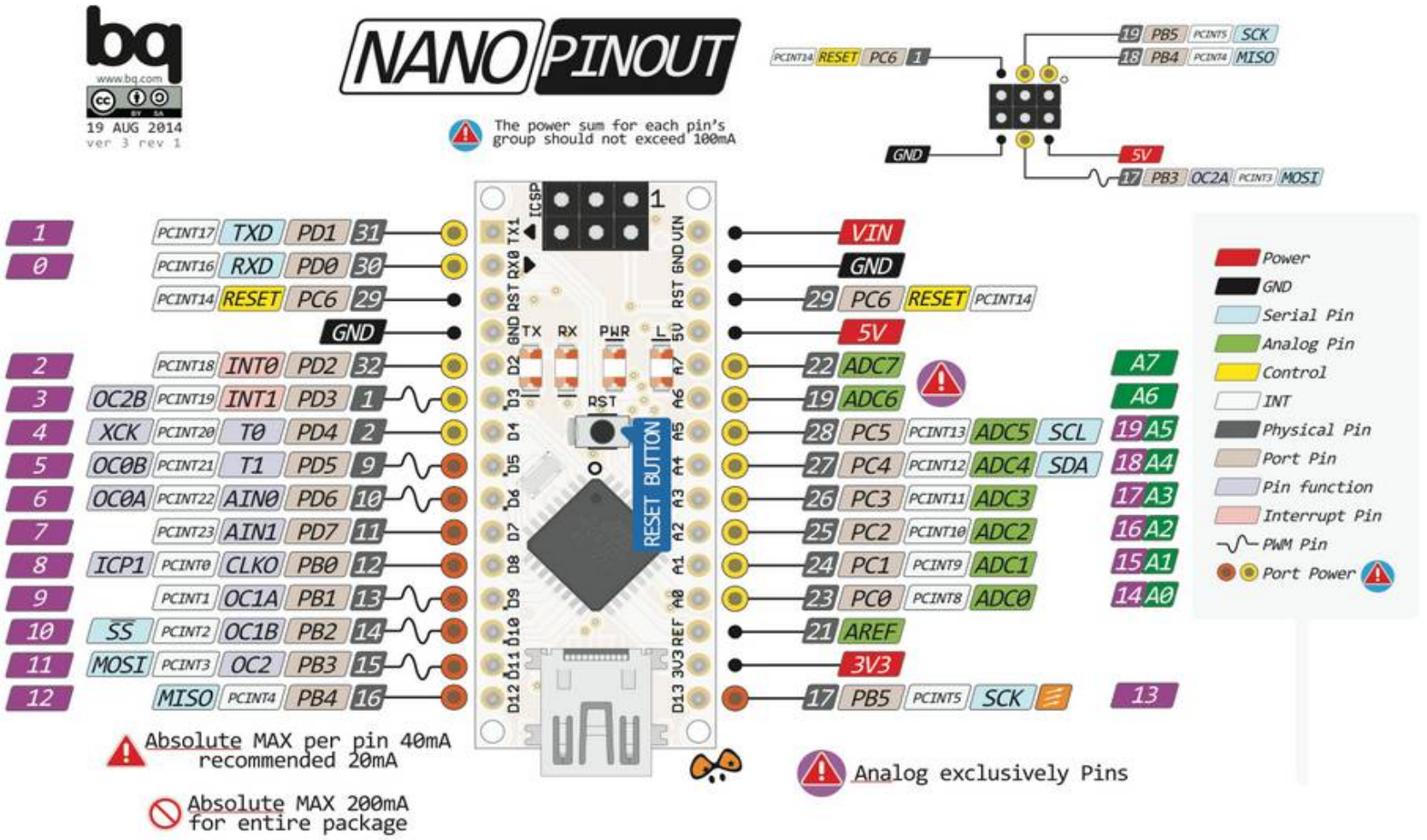
```
import time
import board
import digitalio
# --- L298N lábkiosztás az ESP32-C3-on ---
B1A = board.IO3   # Bal motor előre
B1B = board.IO2   # Bal motor hátra
A1A = board.IO1   # Jobb motor előre
A1B = board.IO0   # Jobb motor hátra
# --- Digitális kimenetek inicializálása ---
b1a = digitalio.DigitalInOut(B1A)
b1a.direction = digitalio.Direction.OUTPUT
b1b = digitalio.DigitalInOut(B1B)
b1b.direction = digitalio.Direction.OUTPUT
a1a = digitalio.DigitalInOut(A1A)
a1a.direction = digitalio.Direction.OUTPUT
a1b = digitalio.DigitalInOut(A1B)
a1b.direction = digitalio.Direction.OUTPUT
# --- Kezdeti állapot: minden motor kikapcsolva ---
b1a.value = False
b1b.value = False
a1a.value = False
a1b.value = False
print("Motor test (CircuitPython, ESP32-C3)")
```

motor_test.py – 2/2.

```
while True:
    print("Forward ALL")
    b1a.value = True
    b1b.value = False
    a1a.value = True
    a1b.value = False
    time.sleep(3)
    b1a.value = False
    b1b.value = False
    a1a.value = False
    a1b.value = False
    time.sleep(1)
    print("Reverse ALL")
    b1a.value = False
    b1b.value = True
    a1a.value = False
    a1b.value = True
    time.sleep(3)
    b1a.value = False
    b1b.value = False
    a1a.value = False
    a1b.value = False
    time.sleep(1)
```

```
    print("Turn RIGHT")
    b1a.value = True
    b1b.value = False
    a1a.value = False
    a1b.value = False
    time.sleep(3)
    b1a.value = False
    b1b.value = False
    a1a.value = False
    a1b.value = False
    time.sleep(1)
    print("Turn LEFT")
    b1a.value = False
    b1b.value = False
    a1a.value = True
    a1b.value = False
    time.sleep(3)
    b1a.value = False
    b1b.value = False
    a1a.value = False
    a1b.value = False
    time.sleep(2)
```

Az Arduino nano kártya kivezetései



Az ESP32 C3 Super Mini kártya kivezetései

Megjegyzés: Az A5 analóg bemenet (ADC2) nem használható, ha a WiFi használatban van!