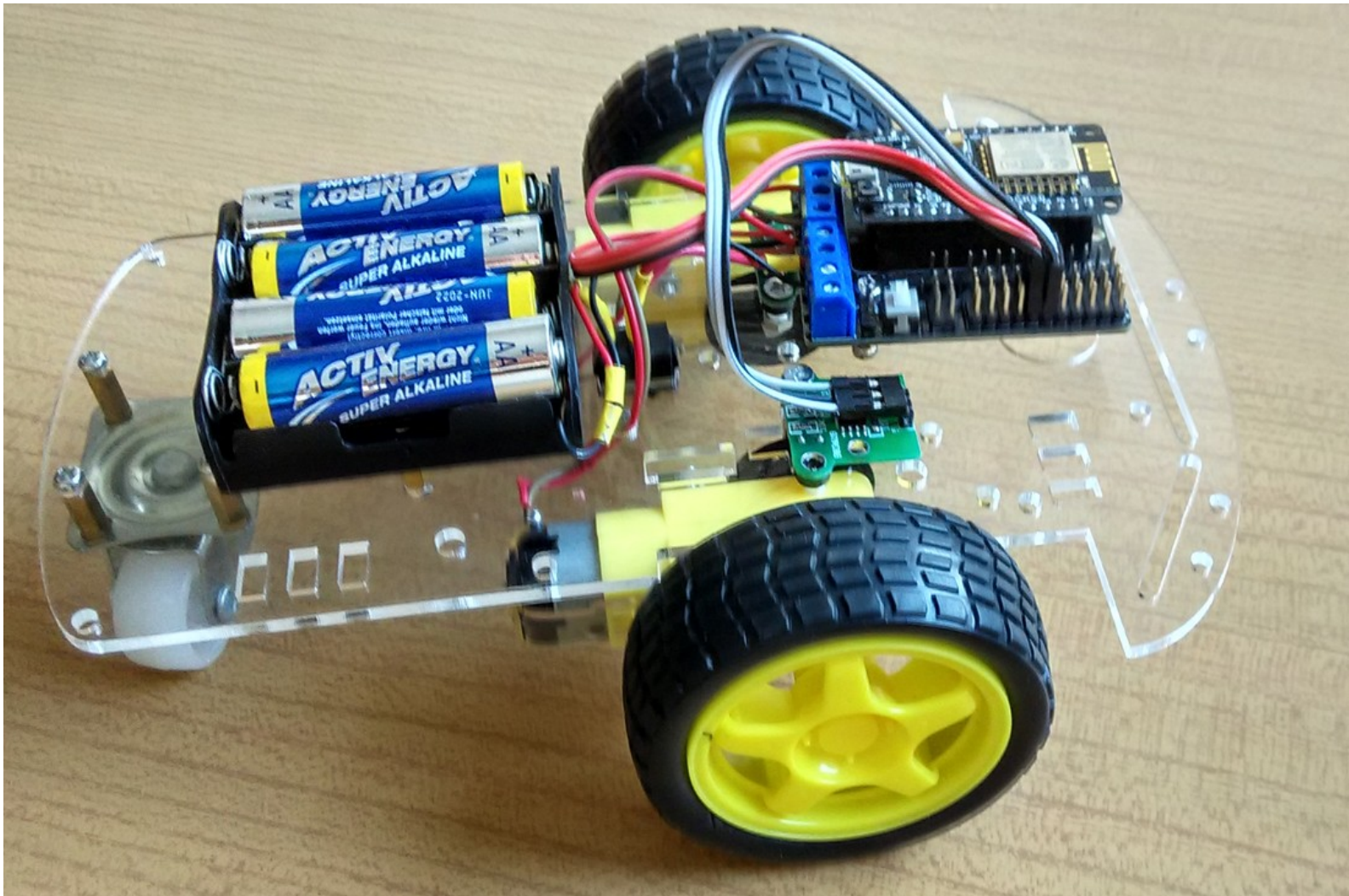


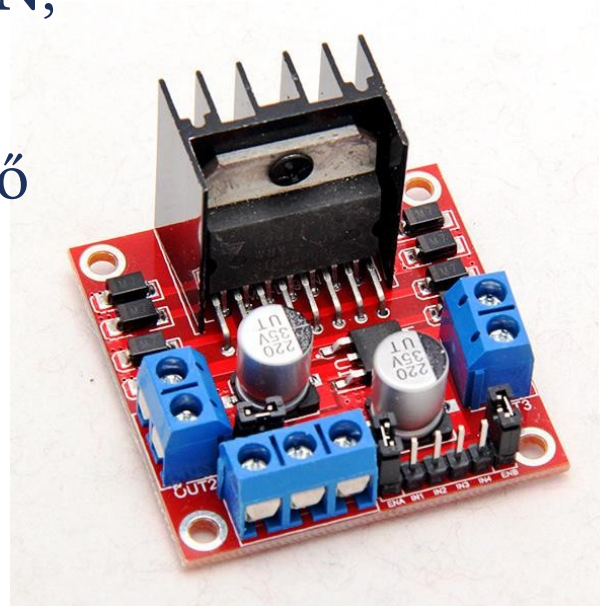
Vegyes témakörök



9. Robotvezérlés sebességmérő szenzorral

Ha robotot építünk, mennyi minden kell nekünk...

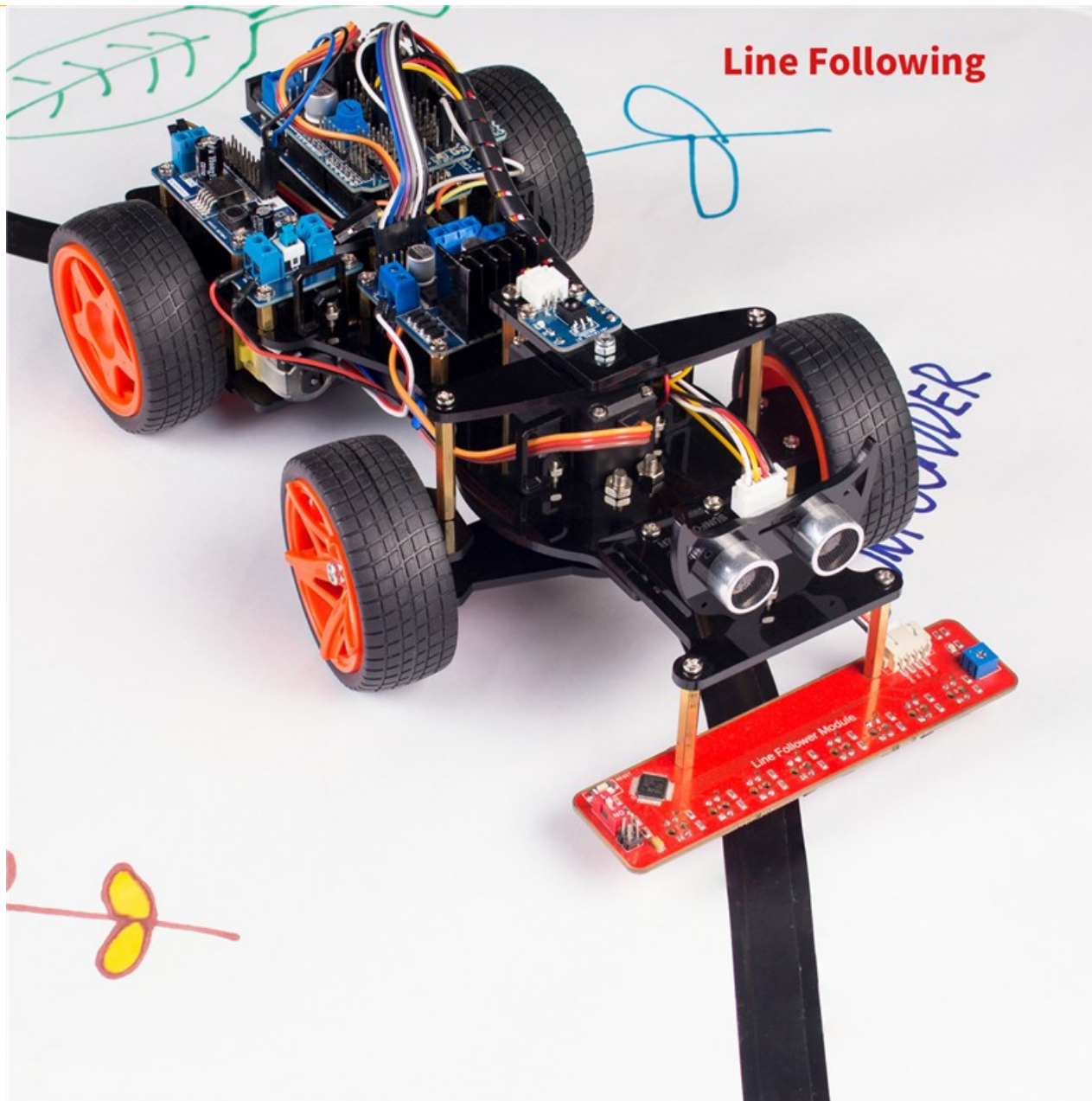
- 2WD robot alváz (2 db DC motor)
- Motorvezérlő panel (L298N, L293D, HG7881)
- Optikai elfordulás érzékelő szenzor
- MCU kártya
 - ❖ Arduino
 - ❖ NodeMCU
- További szenzorok, egyebek



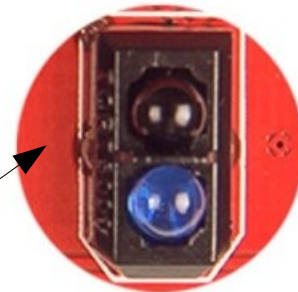
Példa: akadályelkerülő robot



Vonalkövető robot

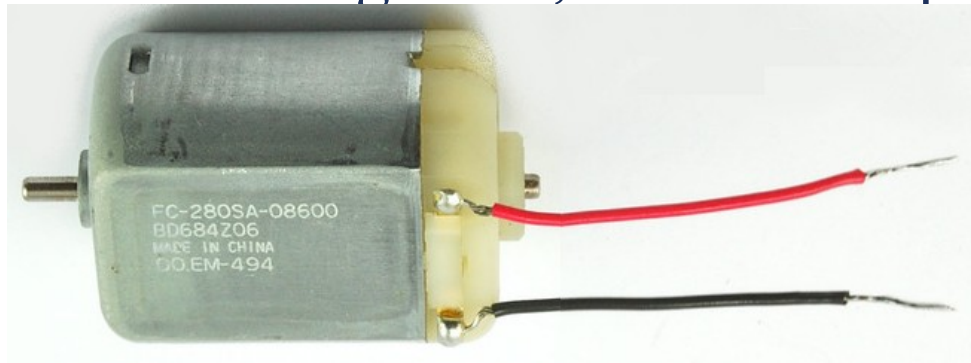


TCRT5000
Sensor

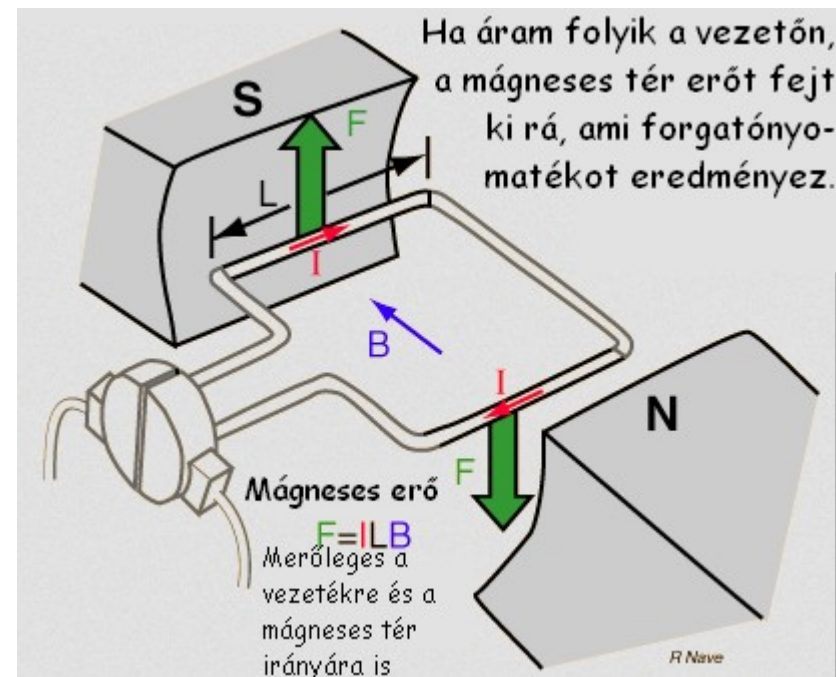
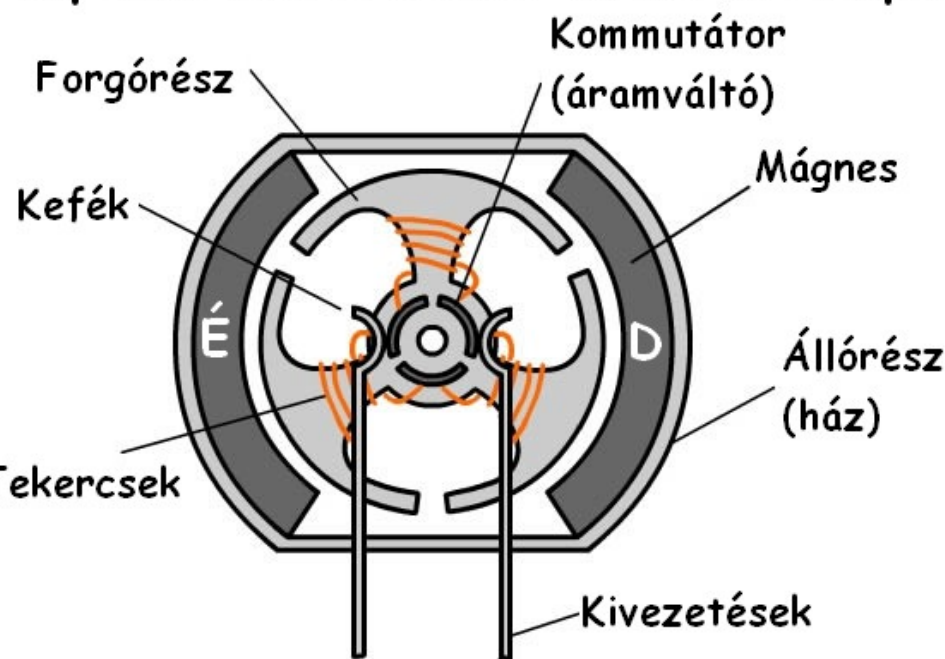


Kisteljesítményű DC motorok

- A hobbi célú motorok állandó mágnes terében forgó elektromágnesek, a tekercsek polaritását kommutátor váltja



Tipikus kefésc motor metszeti képe



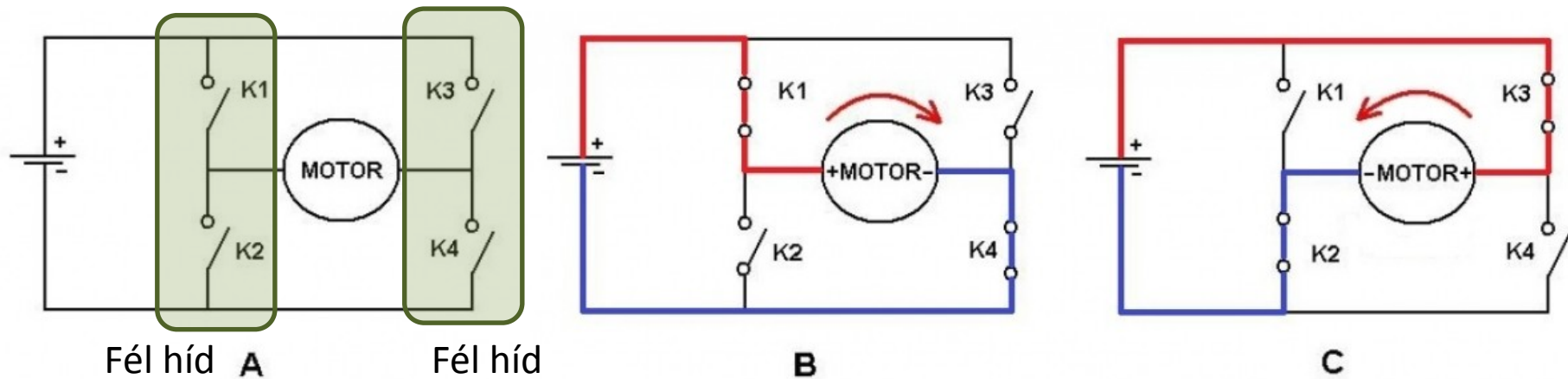
Feszültség: 3 V – 6V

Áram: 0.4 A – 0.6 A

Forgásirány váltás: polaritásváltással

Motorvezérlés: H-híd

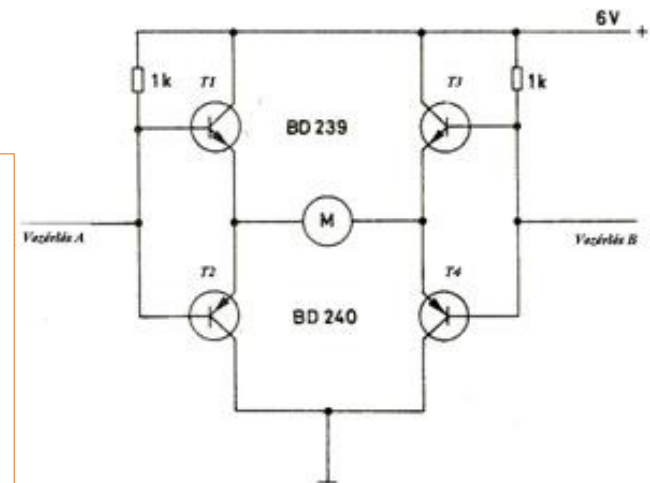
- Ha a motor forgásirányát meg akarjuk változtatni, fel kell cserélni a polaritást.
- Ezt négy kapcsolóval tudjuk megoldani. Az elrendezést H-hídnek nevezzük.



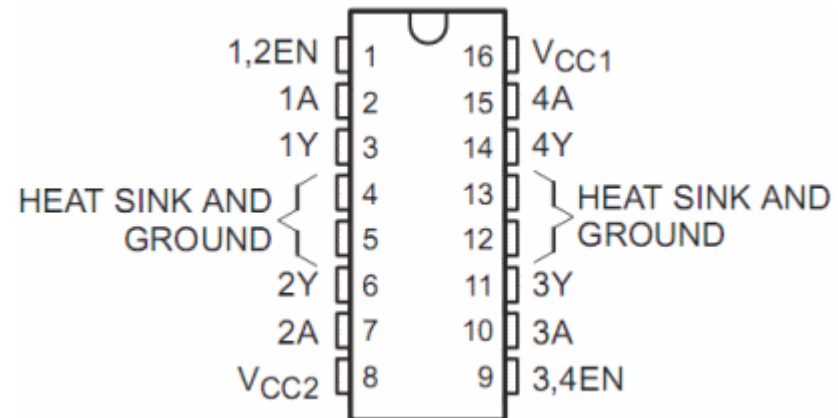
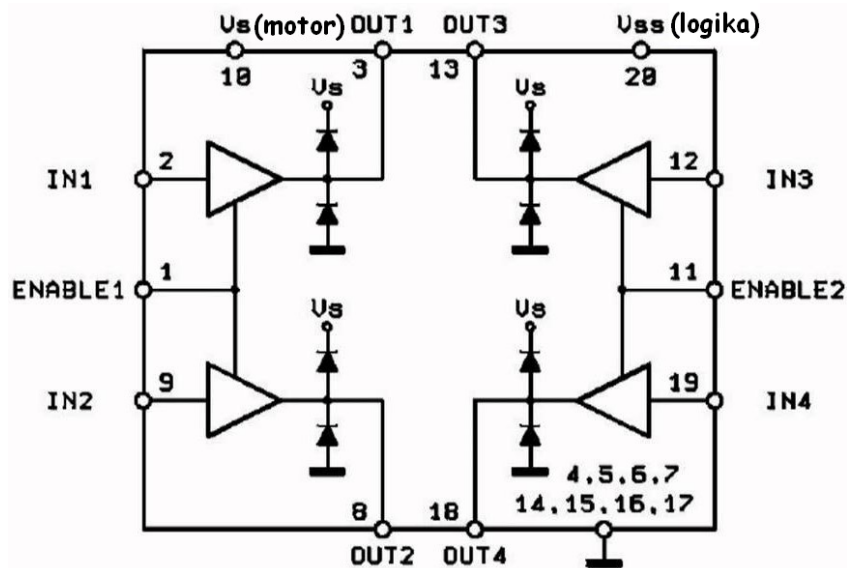
Forrás és leírás: hobbyrobot.hu/content/vonalkoveto-i-robotika-kezdoknek

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



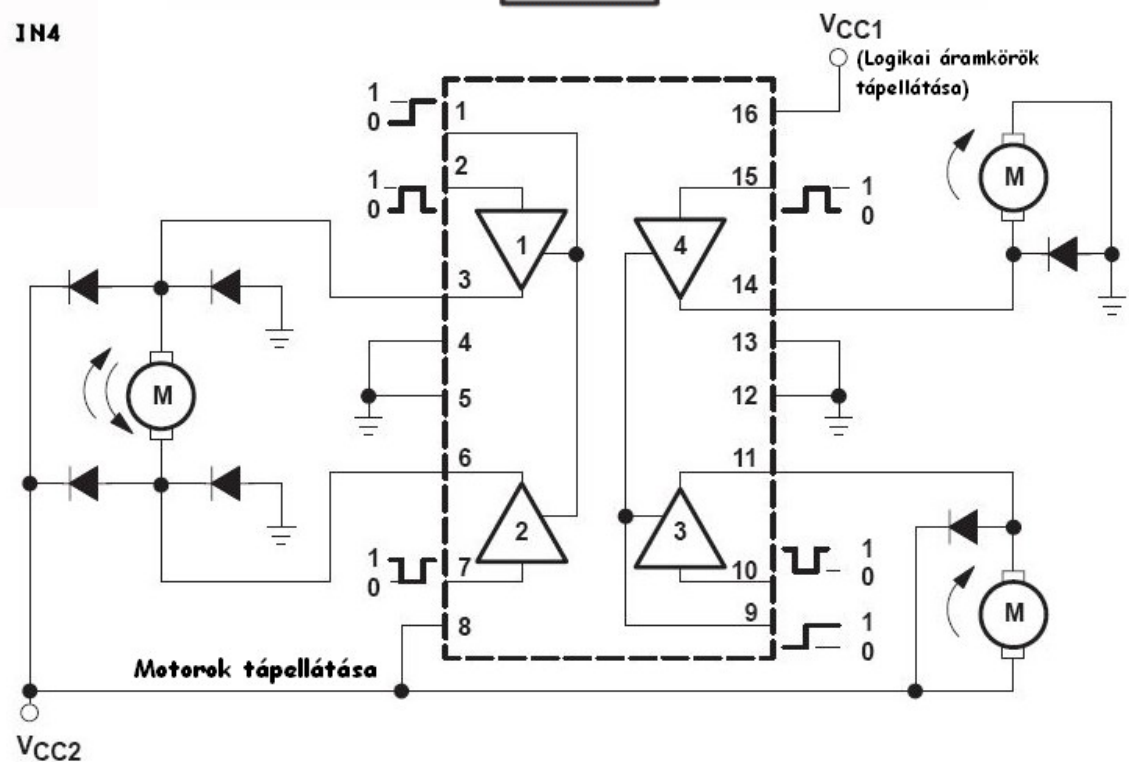
Félhíd: egyirányú forgás vezérlésére (pl. ventilátorok)
H-híd: polaritásváltást igénylő vezérlésekhez

$$V_{CC} = 4,5 - 36 \text{ V}$$

$$V_A, V_{EN} = 2,3 - 7 \text{ V}$$

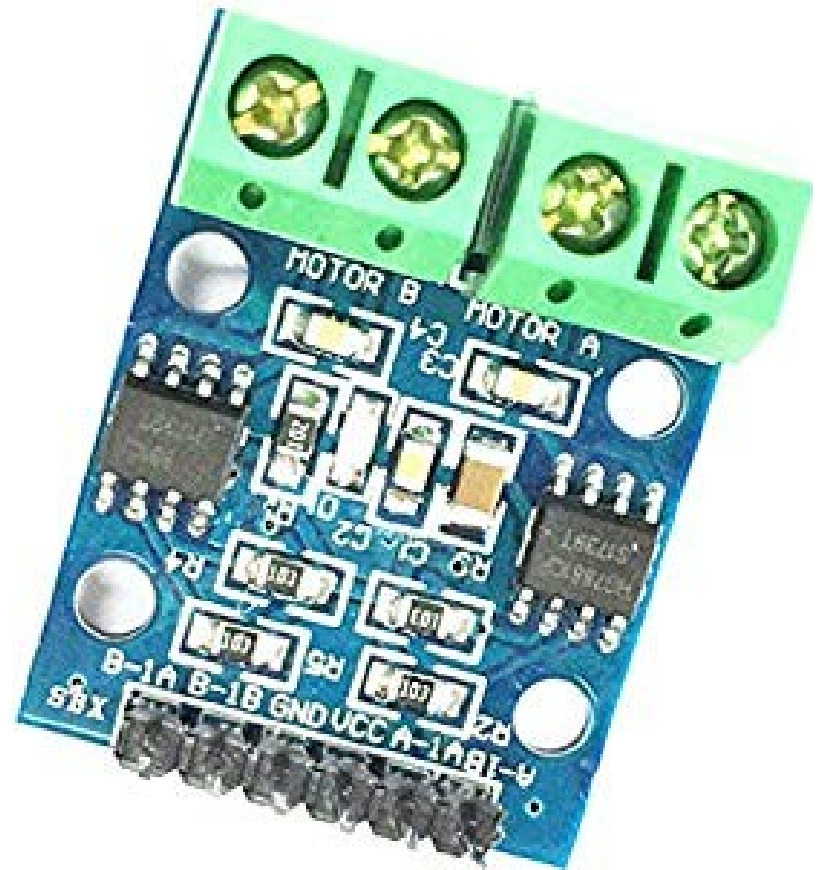
$$I_{max} = 0,6 \text{ A}$$

$$I_{peak} = 1,2 \text{ A (100 } \mu\text{s)}$$



HG7881 motorvezérlő kártya

- A **HG7881** (L9110) nagyon olcsó motorvezérlő kártya Arduinohoz
- Mindkét IC egy-egy H hidat vezérel
- Nincs **Enable** kivezetés, ezért a teljesítményvezérléshez az egyik bemenetet PWM-mel vezéreljük, a másik bemenet pedig az irányt váltja. Irányváltáskor a PWM jel kitöltését komplementálni kell!

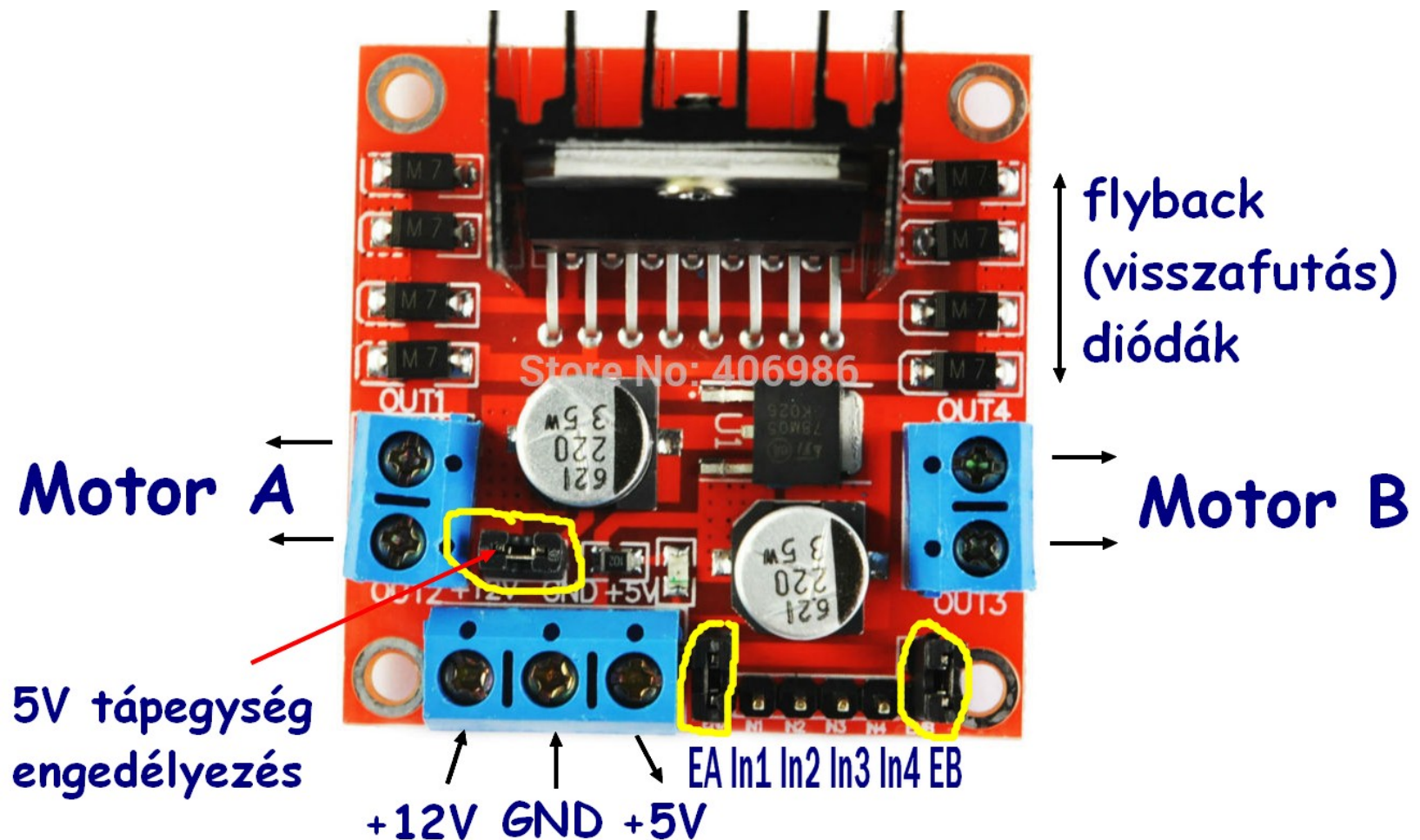


Input		Output		Description
IA	IB	OA	OB	
L	L	L	L	Off
H	L	H	L	Forward
L	H	L	H	Reverse
H	H	H	H	Off

Motor teljesítmény:
2.5-12V @ 800mA
Folyamatos üzemben

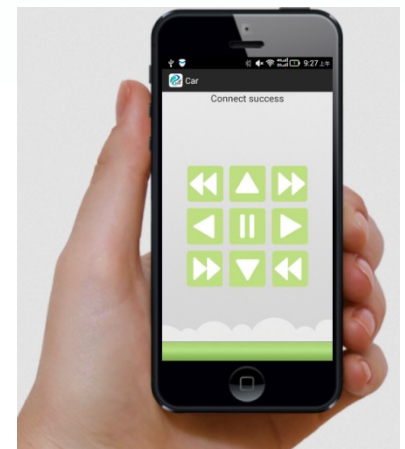
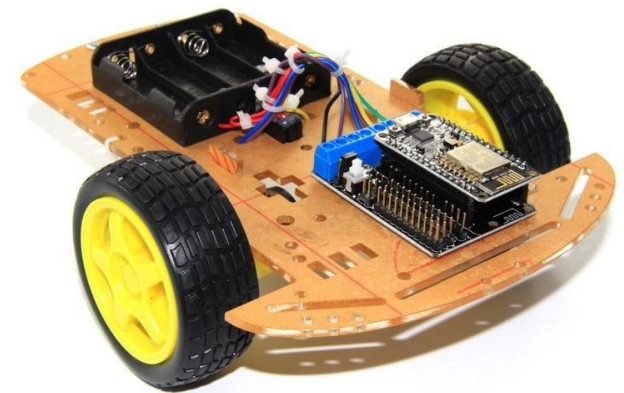
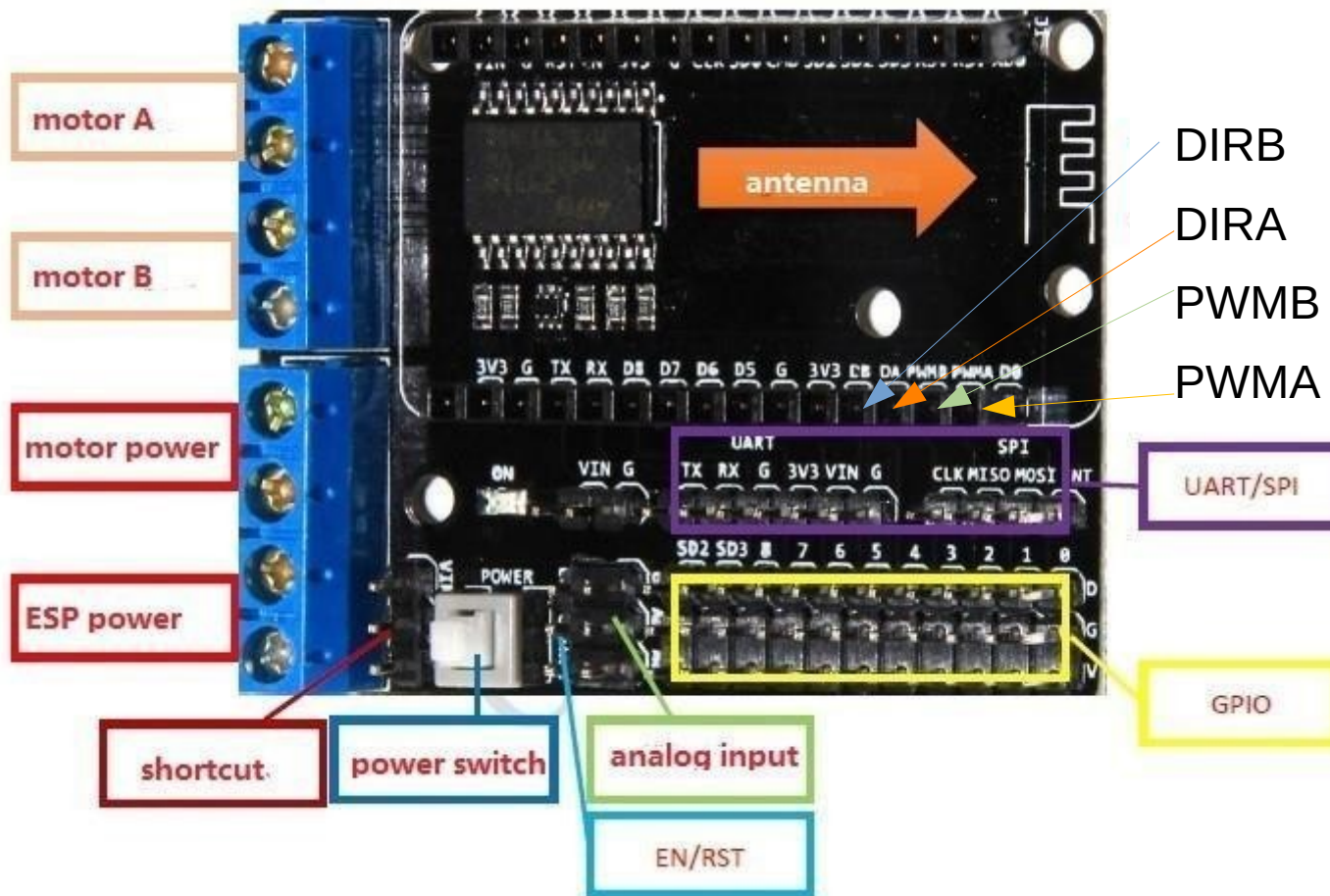
L298N motorvezérlő kártya

- A J1 átkötés engedélyezi az 5 V-os stabilizátort (tápfeszültség az Arduino kártya számára)
- A J2 és J3 átkötés fixen engedélyezi az E1, E2 bemeneteket



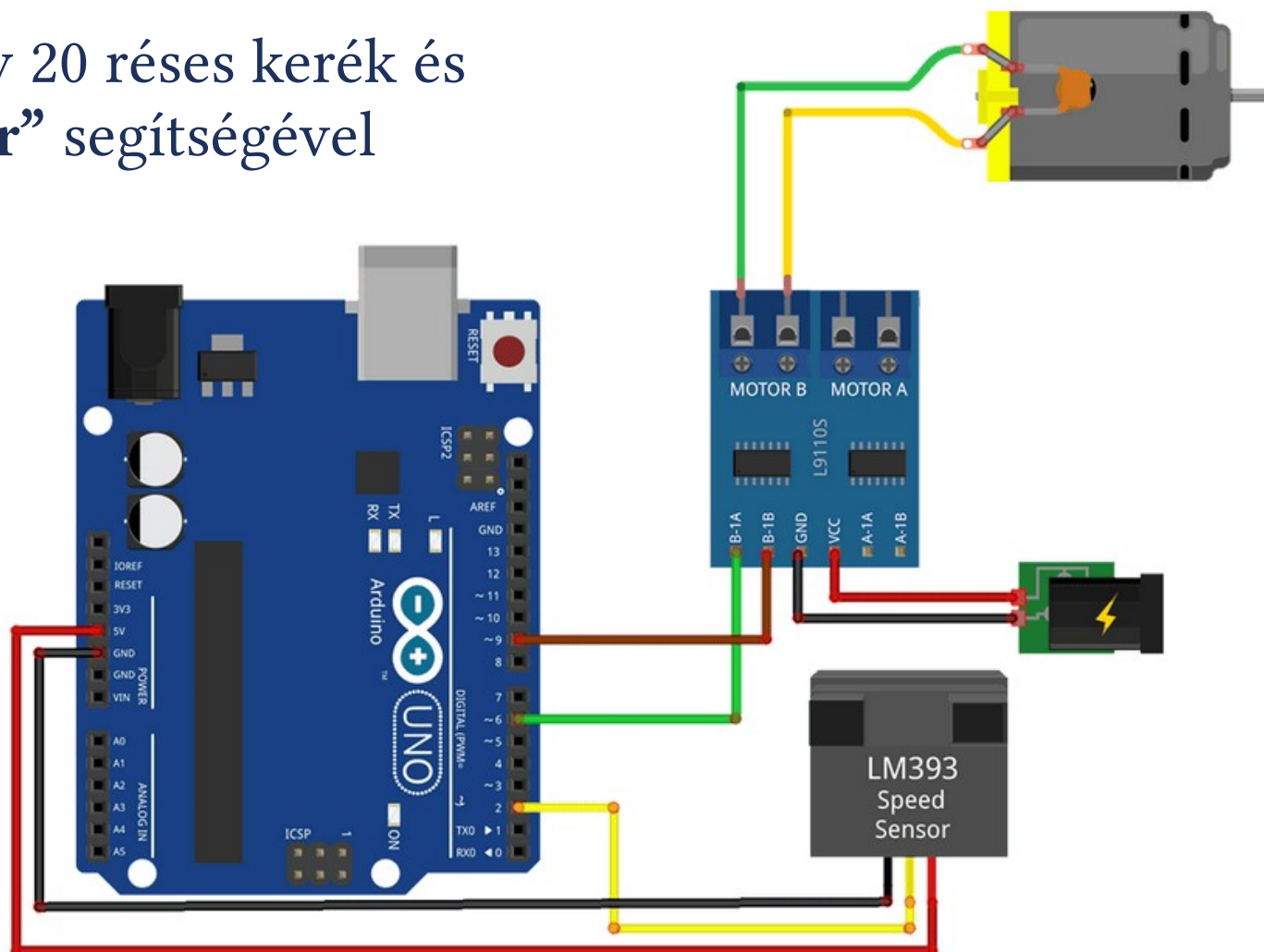
L293D motorvezérlő kártya (NodeMCU-hoz)

- A korábban már említett L293D motorvezérlő SMD változata
- A foglalatba illeszkedik a NodeMCU (ESP8266) kártya
- Például a **DOIT WiFi Car KIT** is ezt használja



Hogyan használjuk a sebességmérő szenzort?

- Egy Arduino mintapélda: How to use a Speed Sensor with Arduino
- A motor külön tápellátást kap
- A motor forgását egy 20 réses kerék és egy „**LM393 szenzor**” segítségével érzékeljük
- Az érzékelő jelével megszakítást kelthetünk, ha a **D2**, vagy **D3** bemenetre kötjük
- A mintapéldához kell a **TimerOne** programkönyvtár is



motor_speed_test.ino

- A1-re egy potmétert kötünk, D2-re az érzékelő kimenetét, D6 és D9-re a motorvezérlő B-1A és B-1B bemenete van rákötve

```
#include "TimerOne.h"
volatile unsigned int counter=0;

int b1a = 6;      // L9110 B-1A
int b1b = 9;      // L9110 B-1B

void docount() {
  counter++;      //számláló növelése
}

void timerIsr() {
  Timer1.detachInterrupt();      // timer stop
  Serial.print("Motor Speed: ");
  int rotation = (counter / 20);  // lyukak számával oszt
  Serial.print(rotation,DEC);
  Serial.println(" Rotation per seconds");
  counter=0; // reset counter to zero
  Timer1.attachInterrupt( timerIsr ); //timer enable
}
```

```
void setup() {
  Serial.begin(9600);
  pinMode(b1a, OUTPUT);
  pinMode(b1b, OUTPUT);
  attachInterrupt(0, docount, RISING);
  Timer1.initialize(1000000); // 1sec
  Timer1.attachInterrupt( timerIsr );
}

void loop() {
  int potvalue = analogRead(1);
  int motorspeed = map(potvalue, 0, 680, 255, 0);
  analogWrite(b1a, motorspeed); // Sebesség
  digitalWrite(b1b, 1);        // Irány
}
```

Forrás: [How to use a Speed Sensor with Arduino](#)

Sebesség, távolság és szög mérése

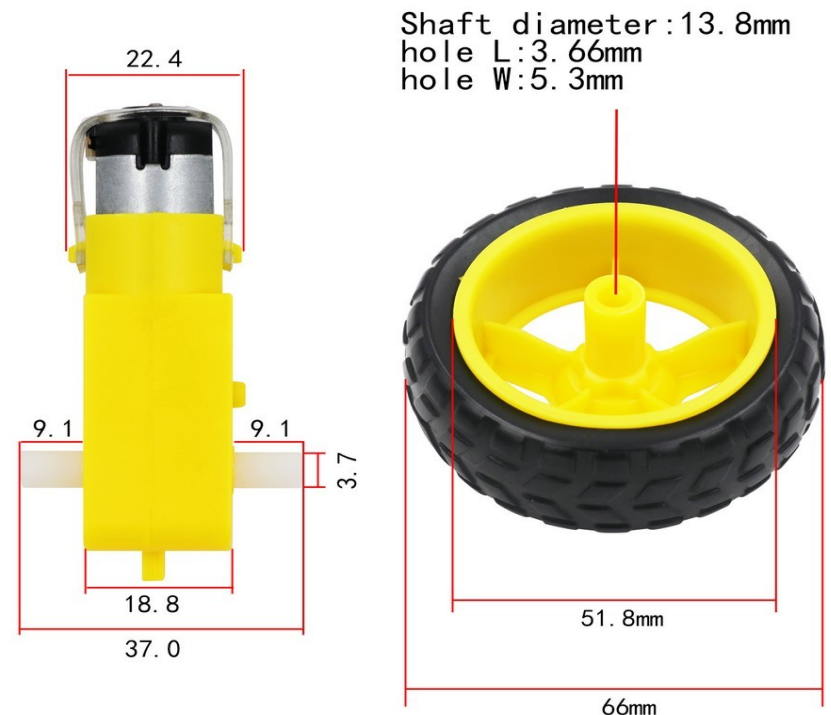
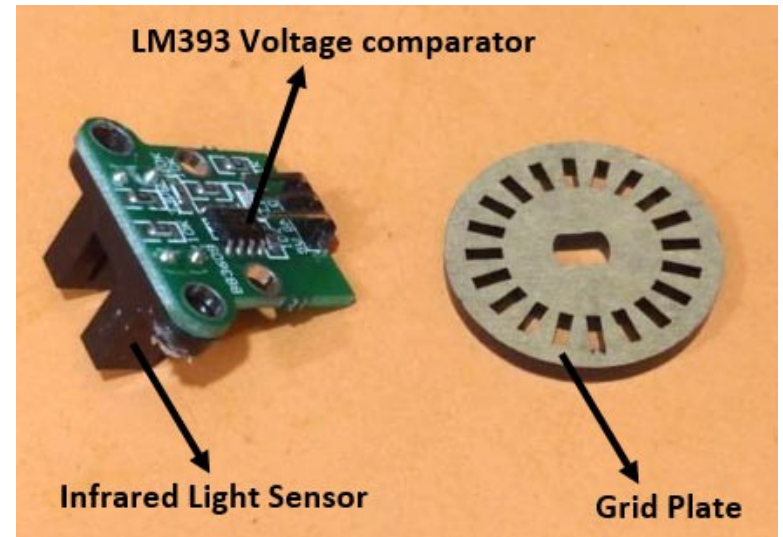
- $RPM = 60\,000 / \text{körülfordulási idő [ms]}$
- $\text{Sebesség} = d \cdot \pi \cdot RPS = d \cdot \pi \cdot RPM/60$
- $\text{Távolság} = d \cdot \pi \cdot \text{interruptszám}/40$
- $\text{Írányszög} = \text{angle_left} - \text{angle_right}$
amiket az alábbi programrész számol:

```
int angle_left=(left_intr%360)*(90/80);  
int angle_right=(right_intr%360)*(90/80);
```

- **Megjegyzés:**

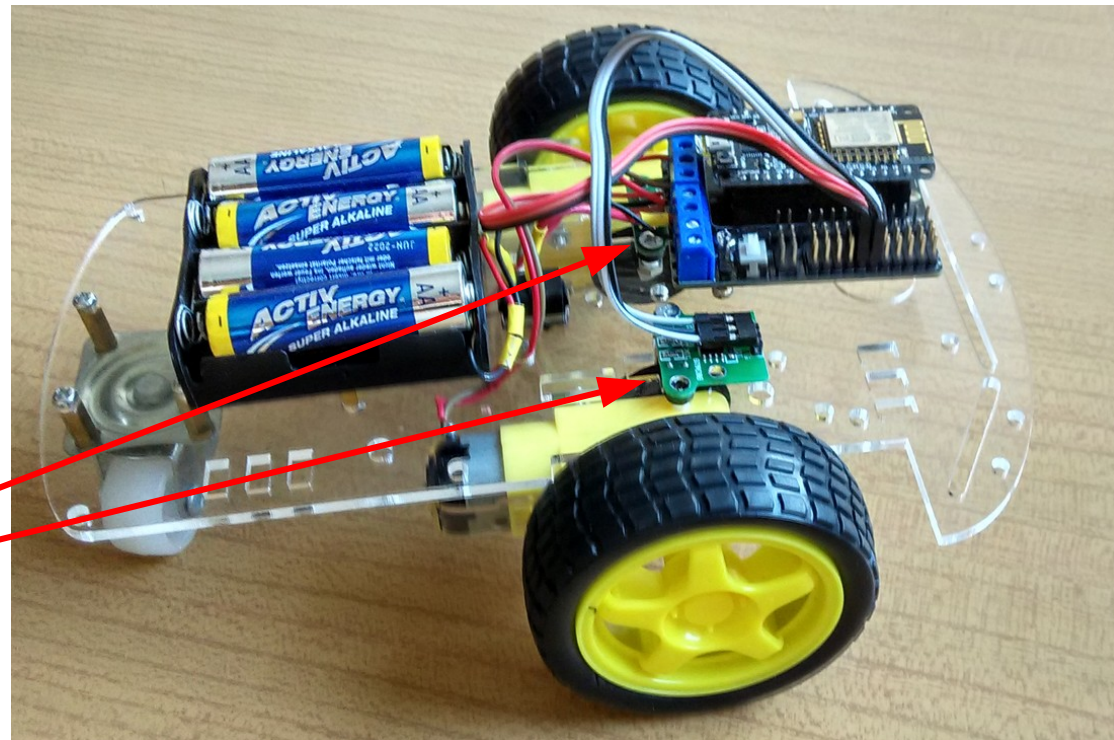
- ❖ A rácsozott keréken 20 db rés van, de a tapasztalat szerint körülfordulásonként 40 megszakítás keletkezik

Forrás: circuitdigest.com



ESP8266/microPython példák

- **Hardver:** a tavaszi kiállításon már bemutatott DOIT Wifi car (2WD robot alváz, NodeMCU kártya és L293D motorvezérlő panel
- **Kiegészítés:** 2 db HC-020K sebességmérő modul
- **Változás:** a változatosság kedvéért ezúttal **Lua** helyett **MicroPython** interpreter fut a kártyán
- A mikrovezérlő kivezetések hozzárendelése a hardver eleméhez a táblázatban látható



NodeMCU	Python	Motor board	Funkció
D1	GPIO5	PWMA	Motor A teljesítmény
D2	GPIO4	PWMB	Motor B teljesítmény
D3	GPIO0	DIRA	Motor A irány
D4	GPIO2	DIRB	Motor B irány
D5	GPIO14	optoB	Motor B forgásérzékelő
D6	GPIO12	optoA	Motor A forgásérzékelő

motor_speed_test.py

```
from machine import Pin, PWM
import time
pwma = PWM(Pin(5))
pwmb = PWM(Pin(4))
dira = Pin(0, Pin.OUT)
dirb = Pin(2, Pin.OUT)
optoa = Pin(12, Pin.IN, Pin.PULL_UP)
optob = Pin(14, Pin.IN, Pin.PULL_UP)
dira.value(0) " Motor A előre
dirb.value(0) " Motor B előre
pwma.duty(600) " Motor A indul
pwmb.duty(600) " Motor B indul
```

```
def callback(p): " Megszakításkor meghívandó függvény
    global counta, countb " Globális változóknban adjuk át az értéket
    if p == Pin(12): " Ha GPIO12 okozta a megszakítást,
        counta = counta + 1 " növeljük counta értékét
    if p == Pin(14): " Ha GPIO14 okozta a megszakítást,
        countb = countb + 1 " növeljük a countb számláló értékét
" Megszakítások engedélyezése és hozzárendelése a callback függvényhez
optoa.irq(trigger=Pin.IRQ_RISING, handler=callback)
optob.irq(trigger=Pin.IRQ_RISING, handler=callback)
```

```
" 10 másodpercig megy előre
" Másodpercenként kiíratjuk a
" számlálók állását
```

```
for i in range(10):
    counta = countb = 0
    time.sleep_ms(1000)
    print(counta, countb)
```

```
pwma.duty(0) " Motor A leállítás
pwmb.duty(0) " Motor B leállítás
```

Folytatás

motor_speed_test.py

```
motor_speed_test.py
1 from machine import Pin,PWM
2 import time
3 global counta
4 pwma = PWM(Pin(5))
5 pwmb = PWM(Pin(4))
6 dira = Pin(0, Pin.OUT)
7 dirb = Pin(2, Pin.OUT)
8 dira.value(0)
9 dirb.value(0)
10 pwmb.freq(100)
11 pwma.freq(100)
12 pwma.duty(600)
13 pwmb.duty(600)
14 optoa = Pin(12, Pin.IN, Pin.PULL_UP)
15 optob = Pin(14, Pin.IN, Pin.PULL_UP)
16
17 def callback(p):
18     global counta, countb
19     if p == Pin(12):
20         counta =counta+1
21     if p == Pin(14):
22         countb = countb +1
23
24 optoa.irq(trigger=Pin.IRQ_RISING, handler=callback)
25 optob.irq(trigger=Pin.IRQ_RISING, handler=callback)
26
27 for i in range(10):
28     counta = countb = 0
29     time.sleep_ms(1000)
30     print(counta, countb)
31
```

```

=== counta = countb = 0
=== time.sleep_ms(1000)
=== print(counta, countb)
===
=== pwma.duty(0)
=== pwmb.duty(0)
===
<IRQ>
<IRQ>
74 74
80 78
80 79
80 78
80 77
80 62
79 76
82 76
80 77
77 78
>>>
```

A két motor nem egyformán húz!

motor_forward.py

- Egyenesen haladáshoz az utat szakaszokra bontjuk, s egy-egy szakaszon belül a jobb- és baloldali előrehaladást szinkronizáljuk

```
from machine import Pin,PWM
import time
pwma = PWM(Pin(5))
pwmb = PWM(Pin(4))
dira = Pin(0, Pin.OUT)
dirb = Pin(2, Pin.OUT)
optoa = Pin(12, Pin.IN, Pin.PULL_UP)
optob = Pin(14, Pin.IN, Pin.PULL_UP)
counta = countb = 0
pwmb.freq(100)
pwma.freq(100)

def callback(p): " Ez a függvény megszakításkor fut le
    global counta, countb
    if p == Pin(12): counta = counta+1 " Számláló növelés
    if p == Pin(14): countb = countb +1

optoa.irq(trigger=Pin.IRQ_RISING, handler=callback)
optob.irq(trigger=Pin.IRQ_RISING, handler=callback)
```

Folytatás a
következő
oldalon

motor_forward.py

```
def forward(n):
    global counta,countb
    counta = countb =0
    dira.value(0)
    dirb.value(0)
    while n > counta or n > countb:
        if n>counta : pwma.duty(600)
        else: pwma.duty(0)
        if n>countb : pwmb.duty(600)
        else: pwmb.duty(0)
        time.sleep_ms(20)
        pwma.duty(0); pwmb.duty(0)

forward(20);
forward(20);
forward(20);
forward(20);
forward(20);
" Megszakítások letiltása
optoa.irq(trigger=0, handler=callback)
optob.irq(trigger=0, handler=callback)
```

```
" Előre megy n lépést
" Globális változók kellene
" Újra indul a számlálás
" Motor A előre
" Motor B előre

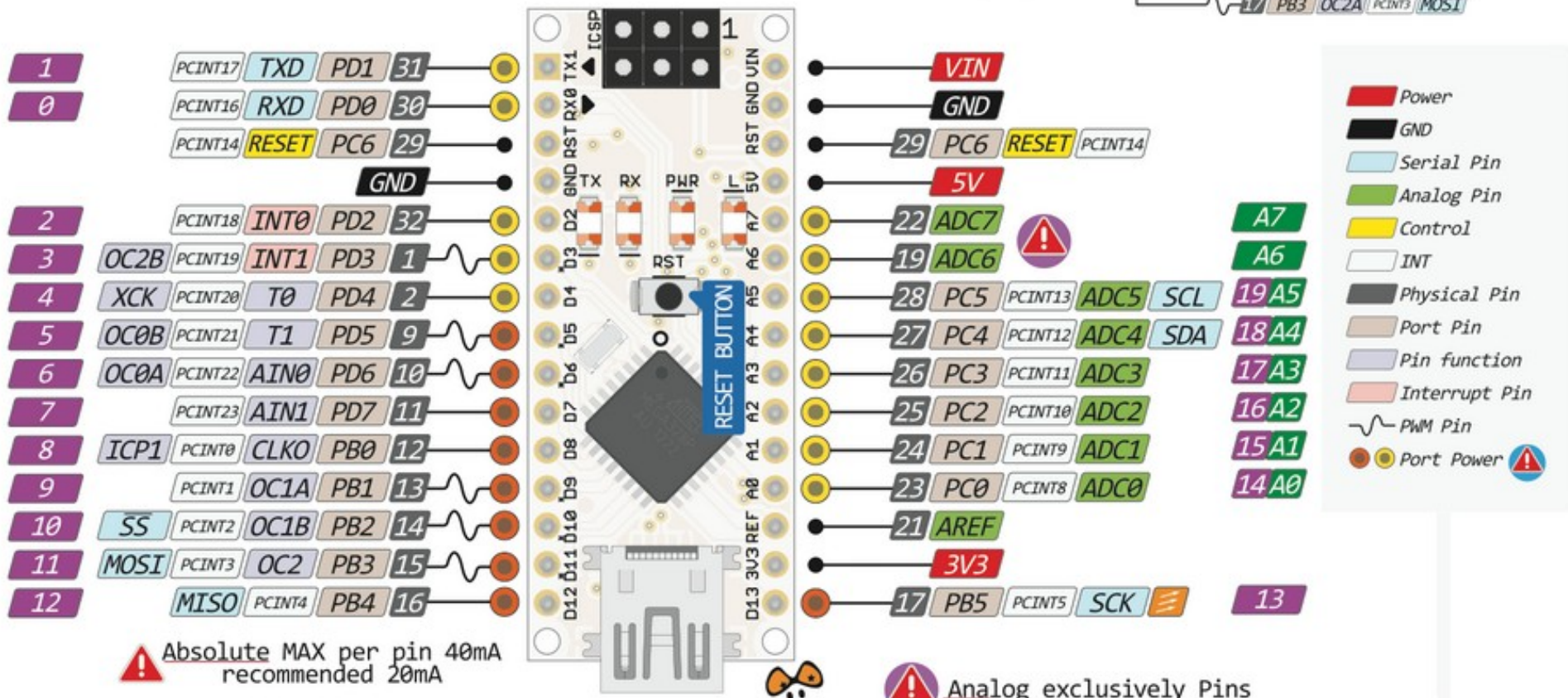
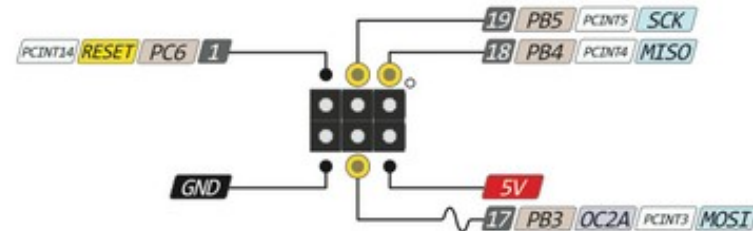
" Motor A haladjon, amíg kell
" Álljon le, ha célba ért
" Motor B haladjon, amíg kell
" Álljon le, ha célba ért
```

Az Arduino Nano kártya kivezetései



NANO PINOUT

The power sum for each pin's group should not exceed 100mA



Absolute MAX per pin 40mA recommended 20mA

Absolute MAX 200mA for entire package

Analog exclusively Pins

NodeMCU kártya MicroPythonnal

