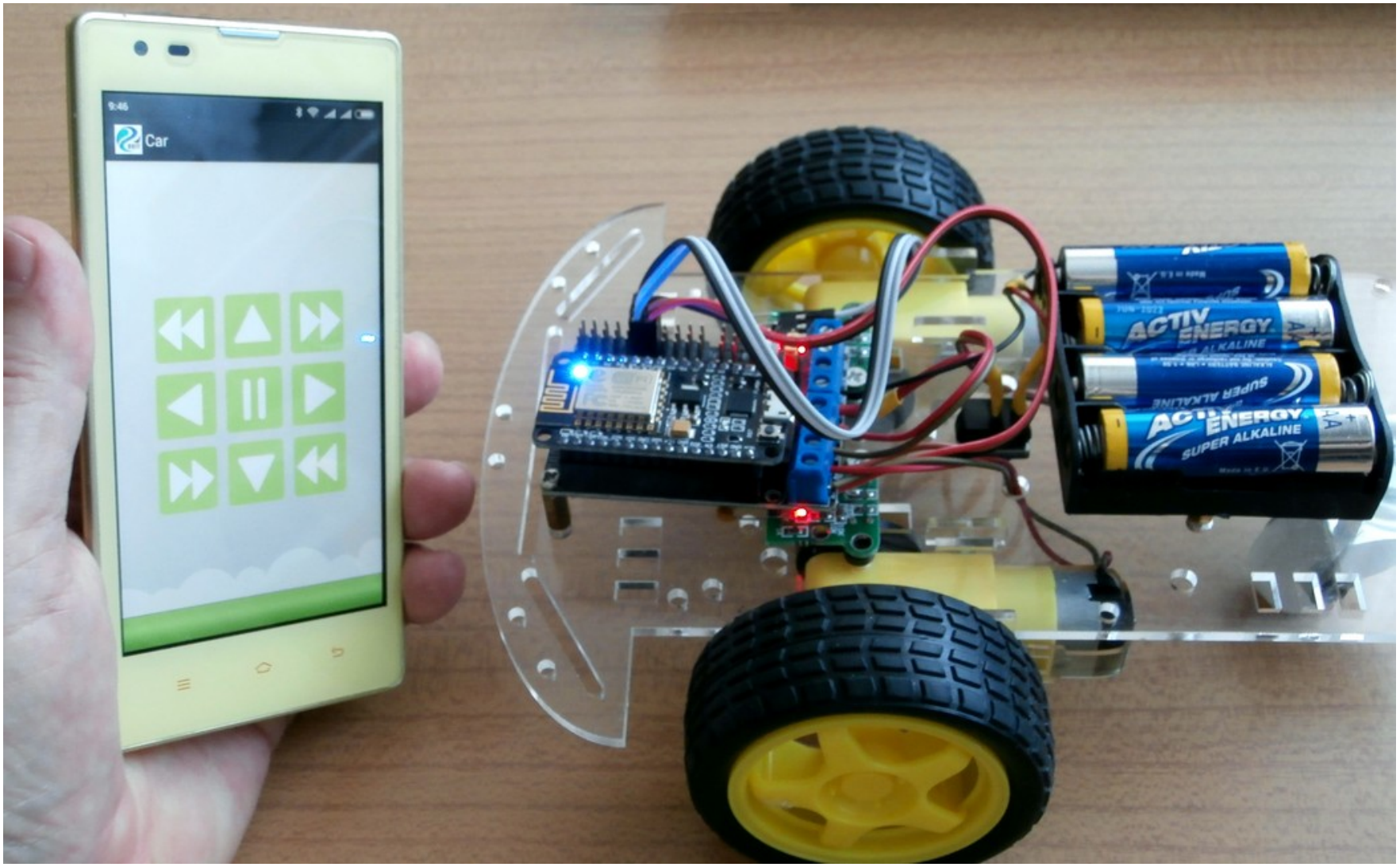
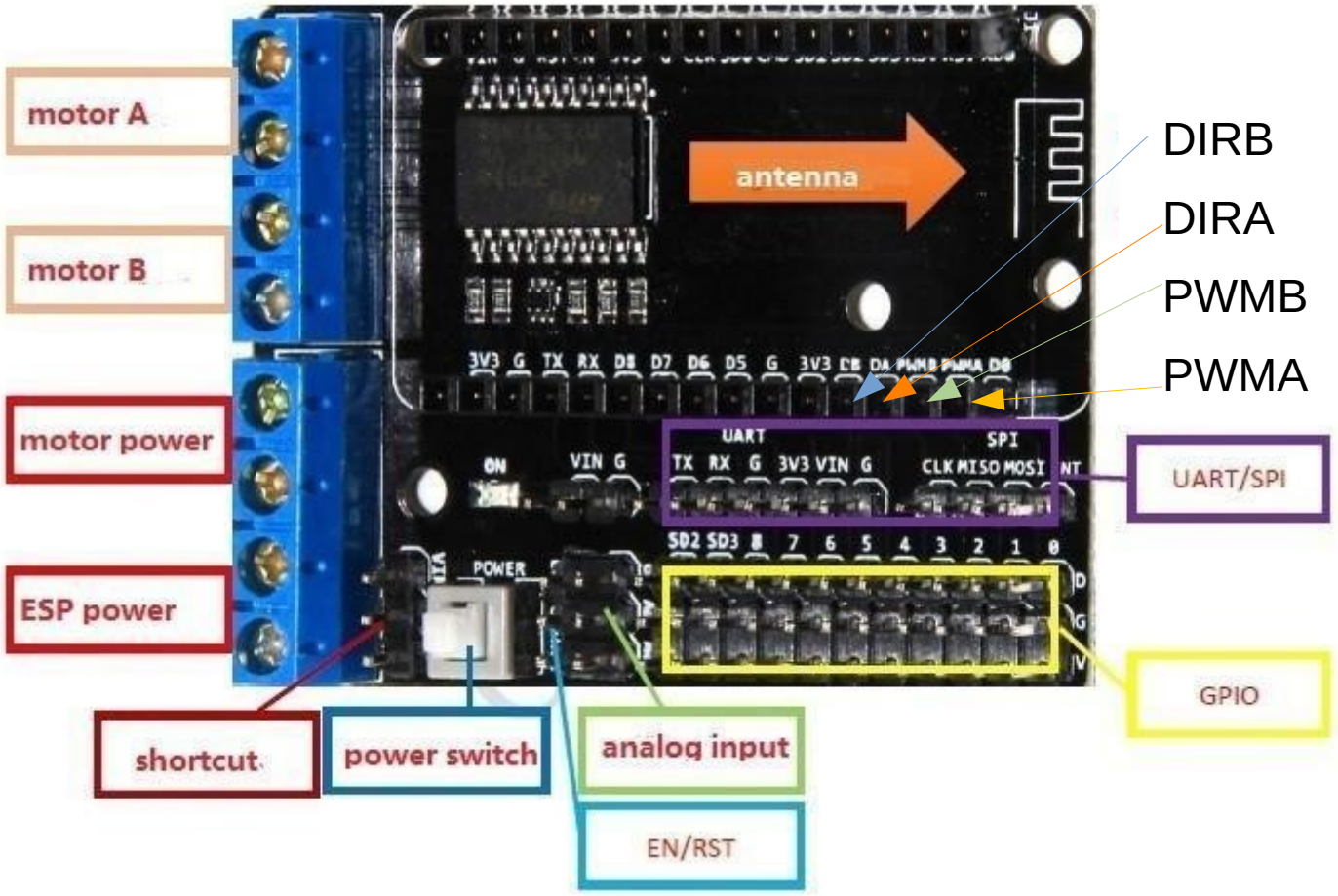
Vegyes témakörök

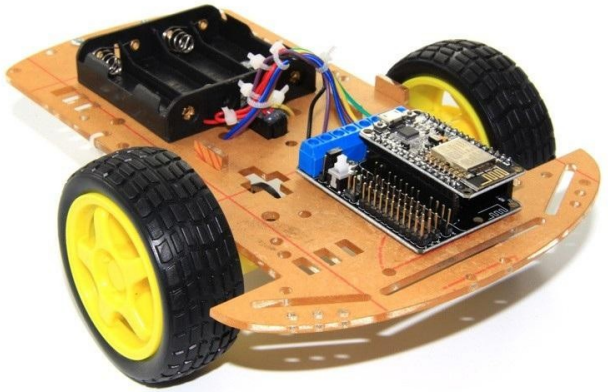


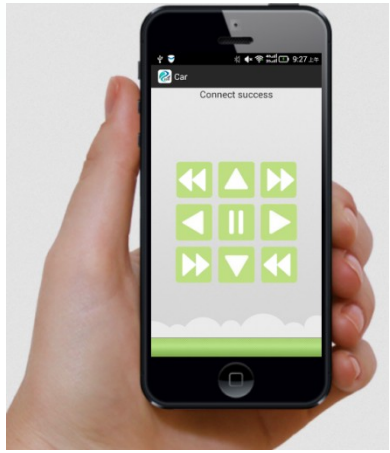
11. Robotvezérlés WiFi kapcsolaton keresztül – 2. rész

DOIT NodeMCU WiFi robot

- **2WD** (két kerék meghajtású) robot alváz elemtartóval
- **L293D** (két H-híd) motorvezérlő – SMD kivitel
- A foglalatba illeszkedő NodeMCU (ESP8266) k





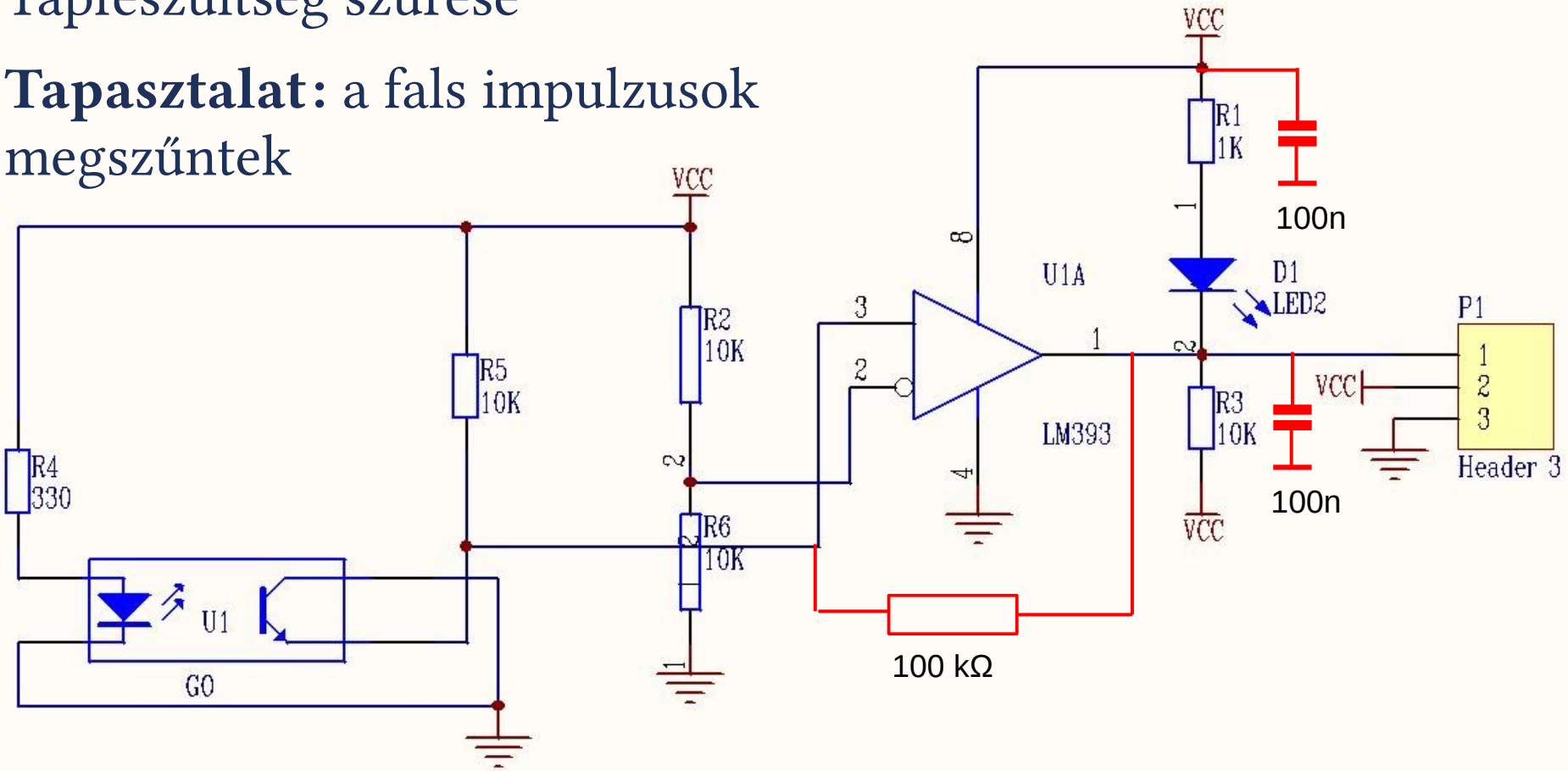


HC-020K optoérzékelő

- Az optoérzékelő egy IR fénysorompó
- A motor tengelyre húzható korongokon 20 rés van, így egy teljes körülfordulás során 20 impulzust ad a kimeneten – legalábbis elvileg…
- A fototranzisztor jelét egy LM393 feléből kialakított komparátor fogadja, de nincs hiszterézis!
- A kimenő jel oszcillál az átmeneteknél, emiatt több impulzust detektálunk (átlagosan 40 db-ot egy korülfordulás során)
- Célszerű tehát az áramkört feljavítani!

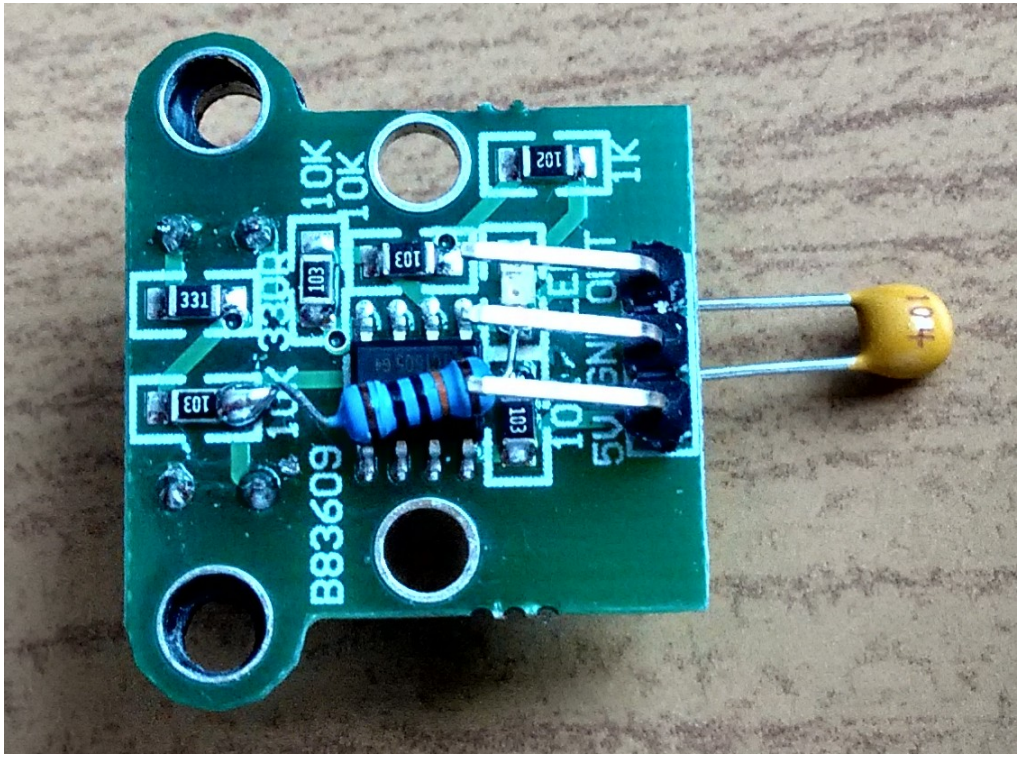
A HC-020K áramkörének korrekciója

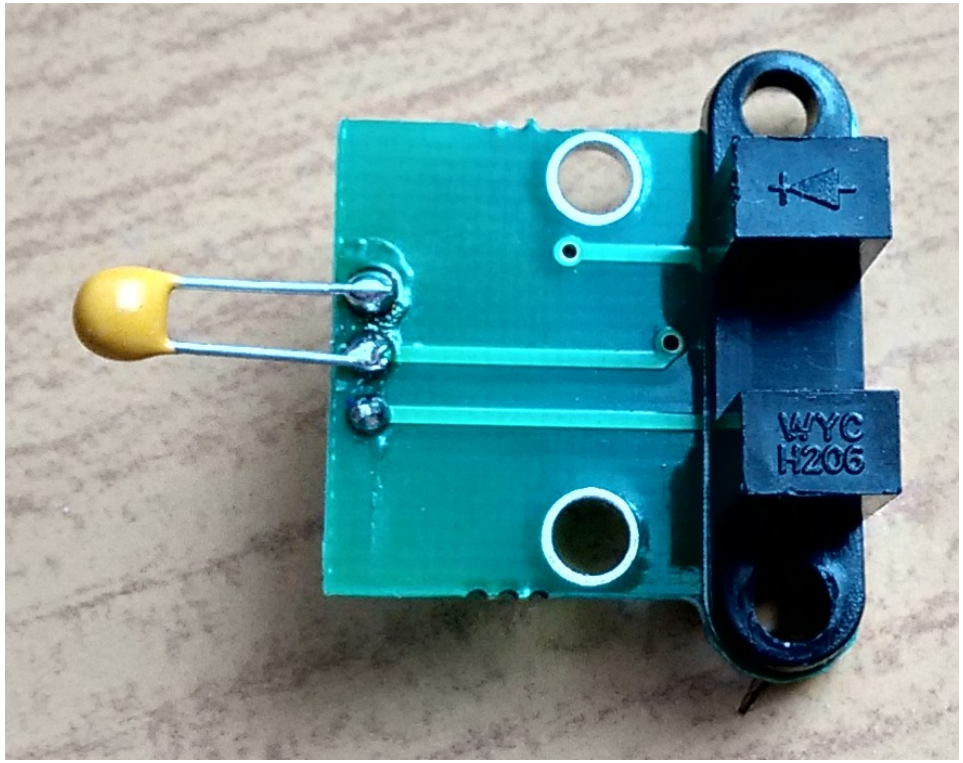
- Hiszterézis pozitív visszacsatolással
- Kimenő jel szűrése
- Tápfeszültség szűrése
- **Tapasztalat:** a fals impulzusok megszűntek



A HC-020K áramkörének korrekciója

- Az alábbi képeken a 100 kΩ-os visszacsatoló ellenállás és a kimenetet szűrő 100 nF-os kondenzátor beépítése látható
- Ezeken kívül még egy további 100 nF-os kondenzátort is beépítettünk a VCC és a GND lábak közé





Multitasking ESP8266 MicroPython alatt

- A **_thread** könyvtár által nyújtott **preemptív többfeladatúság** az **ESP8266** szűkös lehetőségei miatt nincs implementálva
- A **MicroPython** újabb kiadásaiban (1.9.10-től kezdve) van támogatás a **kooperatív többfeladatúságra** (itt az egyes feladatoknak önként kell időnként visszaadni a vezérlést), de elég bonyololult a használata. Forráskód és leírás itt található: Peter Hinch, github.com/peterhinch/micropython-async
- Speciális esetekben **egyszerűbb megközelítést** is használhatunk
  - Az előző előadásban említett, a **TCP** üzenetekre történő várakozásnál használt **socket.settimeout()** metódus is működik, de 0,5 s alatti időzítéseknél csomagvesztés lehet (nyomjuk a gombot, de a robot nem reagál)
  - A **Timer** könyvtár felhasználásával tetszőlegesen üzemezett visszahívásokat (callback) generálhatunk - a mai előadásban ezzel a módszerrel foglalkozunk

machine.Timer osztály

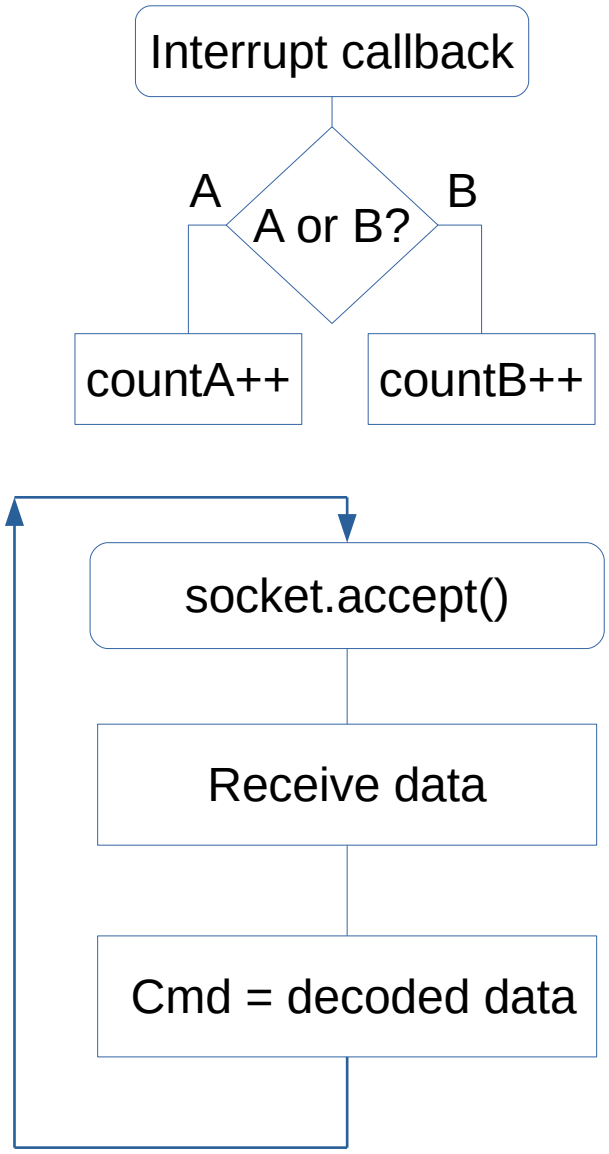
- A **machine** könyvtári modul **Timer** osztálya az időzítők kezelésére szolgál
- **ESP8266** esetén csak virtuális (szoftveresen implementált) **Timer** áll rendelkezésre, ennek azonosítója ***–1,*** például:

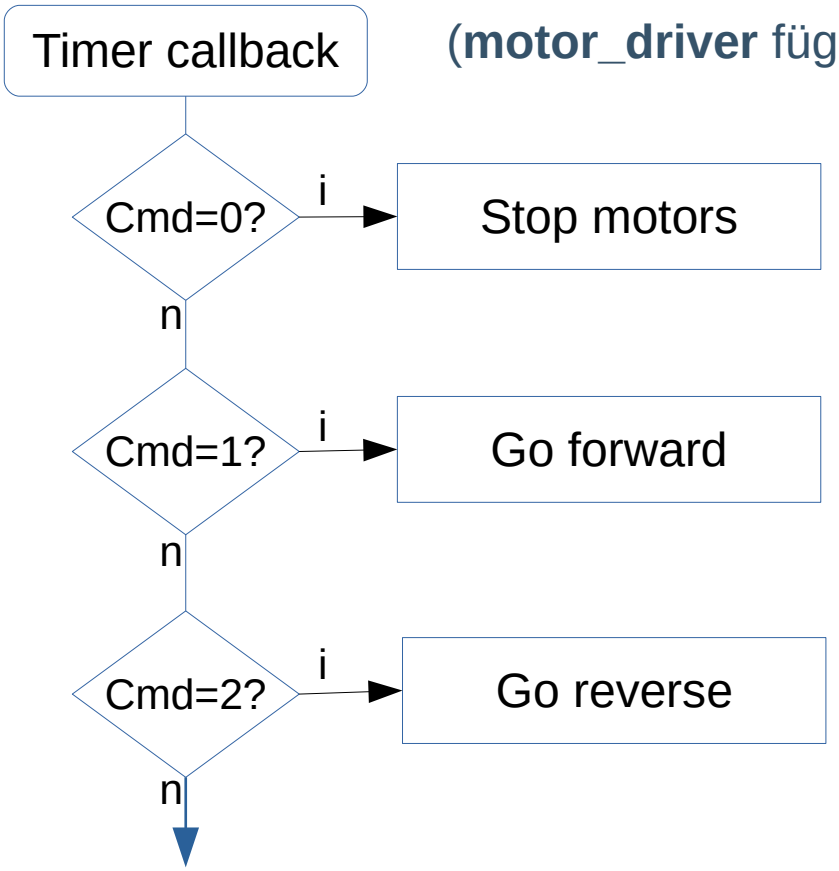
```
from machine import Timer

tim1 = Timer(-1)
tim2 = Timer(-1);
tim1.init(period=5000, mode=Timer.ONE_SHOT, callback=lambda t:print(1))
tim2.init(period=2000, mode=Timer.PERIODIC, callback=lambda t:print(2))
```

- **Metódusok**
  `Timer.init()` – az időzítő inicializálása (a periódus ms-ban értendő)
  `Timer.deinit()` – az időzítő leállítása
- **Működési módok:** `Timer.ONE_SHOT` és `Timer.PERIODIC`

A robotvezérlő program felépítése





(**motor_driver** függvény)

- az optoszenzor megszakításokat generál
- a motorvezérlést a Timer visszahívások,
- a socketet pedig a főprogram kezeli

Doit_car.py

```
import network
import usocket as socket
from machine import Pin, PWM, Timer
import time
time.sleep(2)  <-------------  # A leállíthatóság miatt kell
ap = network.WLAN(network.AP_IF)
ap.active(True)
ap.config(essid="ESP-AP")
ap.ifconfig(('192.168.1.1','255.255.255.0','192.168.1.1','192.168.1.1'))
#--- Kivezetések hozzárendelése -----
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)

#--- Globális változók ---------------
newcmd = lastcmd = counta = countb = 0
speed = 800          # A PWM kitöltés 0 - 1023 között érték lehet
```

Firmware a Doit Car Android alkalmazáshoz

Az alkalmazás fix IP címet feltételez

Ezen kivezetések kiosztása fix a motorvezérlő alaplap miatt

Doit_car.py

```
#--- Külső megszakítás visszahívási függvénye ---
def callback(p):
  global counta, countb
  if p == Pin(12): counta = counta +1
  if p == Pin(14): countb = countb +1

#--- Külső megszakítások engedélyezése felfutó élre ---
optoa.irq(trigger=Pin.IRQ_RISING, handler=callback)
optob.irq(trigger=Pin.IRQ_RISING, handler=callback)

#--- Szoftveres időzítő visszahívás konfigurálása
tim = Timer(-1)
tim.init(period=20, mode=Timer.PERIODIC, callback=motor_driver)
```

Megszakításkor léptetjük a megfelelő számlálót

Timer-rel ütemezetten periodikusan meghívatjuk a motorvezérlő függvényt

- A **GPIO12** és **GPIO14**-re kötött optoérzékelők megszakításokat keltenek, kiszolgálásukkor a megfelelő számlálót növeljük
- A virtuális **Timer** időzítővel periodikusan meghívjuk a motorvezérlő függvényt (**motor_driver**)

Doit_car.py

```
def motor_driver(tim) :                 # Ez a Timer callback függvény
  global counta, countb, newcmd, lastcmd, speed
  ca=counta; cb=countb; cmd=newcmd      # Ezeket csak egyszer olvassuk!
  if cmd == 0 :                         # --- STOP command? ---
    if cmd != lastcmd :
      pwma.duty(0)
      pwmb.duty(0)
  elif cmd == 1 :                       # --- GO FORWARD command? ---
    if (cmd != lastcmd) :               # Ha új a parancs ...
      counta = countb =0
      dira.value(0)                     # Előre menetbe kapcsolunk
      dirb.value(0)
      pwma.duty(speed)                  # Motor indítás
      pwmb.duty(speed)
    elif (ca == cb) :                   # Korábbi parancs esetén
      pwma.duty(speed)                  # csak az együttfutást kezeljük
    elif (ca > cb+2) :
      pwma.duty(speed-200)
    elif (cb > ca+2) :
      pwma.duty(speed+200)
  elif cmd == 2 :                       # --- GO REVERSE command ---
    #---  Ugyanaz mint a GO FORWARD parancs csak dira és dirb más ----
  lastcmd = cmd                         # Megjegyezzük a feldolgozott parancsot
```

Doit_car.py

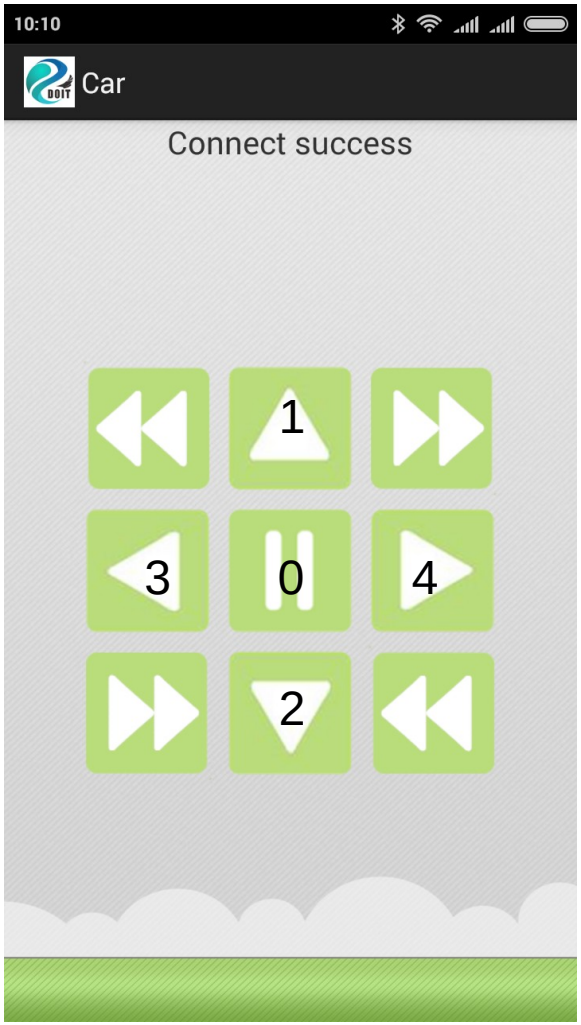
- Egy **TCP** socketet kell nyitnunk a **9003**-as porton és a kapott csomagok első karaktere lesz a parancs kódja ***(0 – 9)***
- Az üres üzenetcsomagot átlépjük (kapcsolat bontásakor jön ilyen)

```
s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
s.setsockopt(socket.SOL_SOCKET, socket.SO_REUSEADDR, 1)
s.bind(('', 9003))
s.listen(0)
# print("Listening, connect client to 192.168.1.1:9003")

while True:
    conn, addr = s.accept()
    data=conn.recv(1024)
#    print('received:',data,'from',addr)
    if data:
      newcmd = data[0]-48
    conn.close()
```

Doit Car.apk

- A **Doit NodeMcu** alapú robotjához készített firmware és applikáció (***sta*** és ***ap***) itt található: github.com/SmartArduino/DoitCar
  Leírás: smartarduino.gitbooks.io/user-manual-for-wifi-car-by-nodemcu-doitcar-/
- Mi az ***ap*** verziót használjuk, melynek előnye a közvetlen kapcsolat (nem kell router)
- Hátrány: fix IP címet (192.168.1.1) és port számot (9003) feltételez
- Rövid TCP üzenetekkel kommunikál, az első karakter a gomb kódja
- Kilépéskor egy üres üzenetet küld



NodeMCU kártya MicroPythonnal