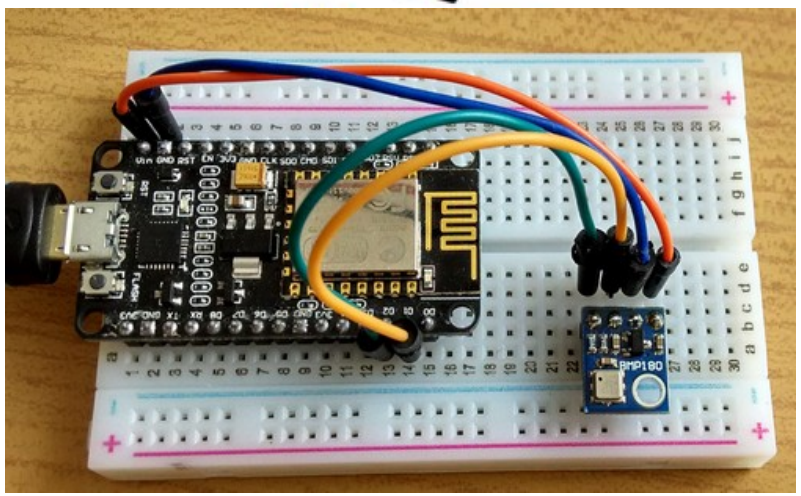
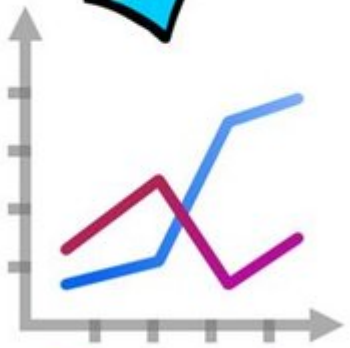
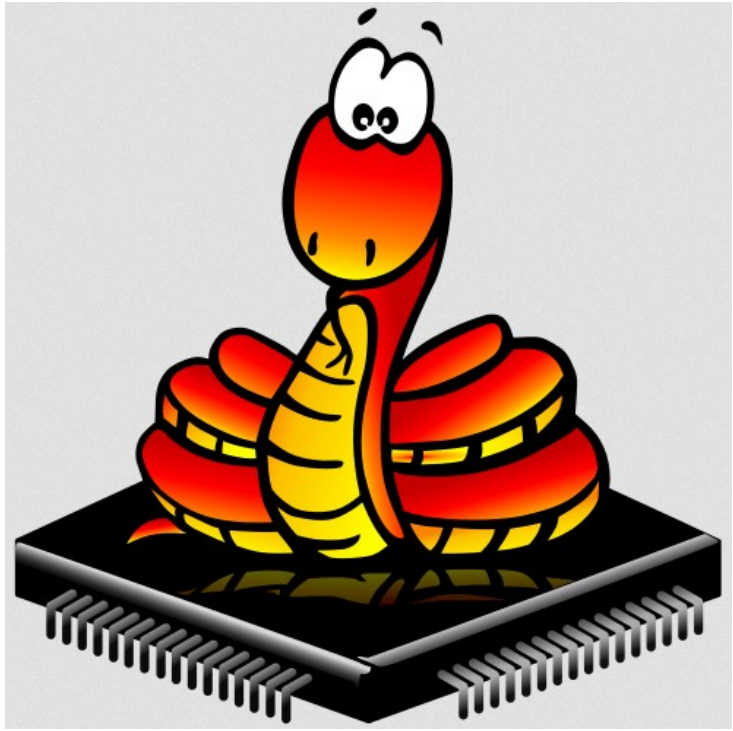
Vegyes témakörök



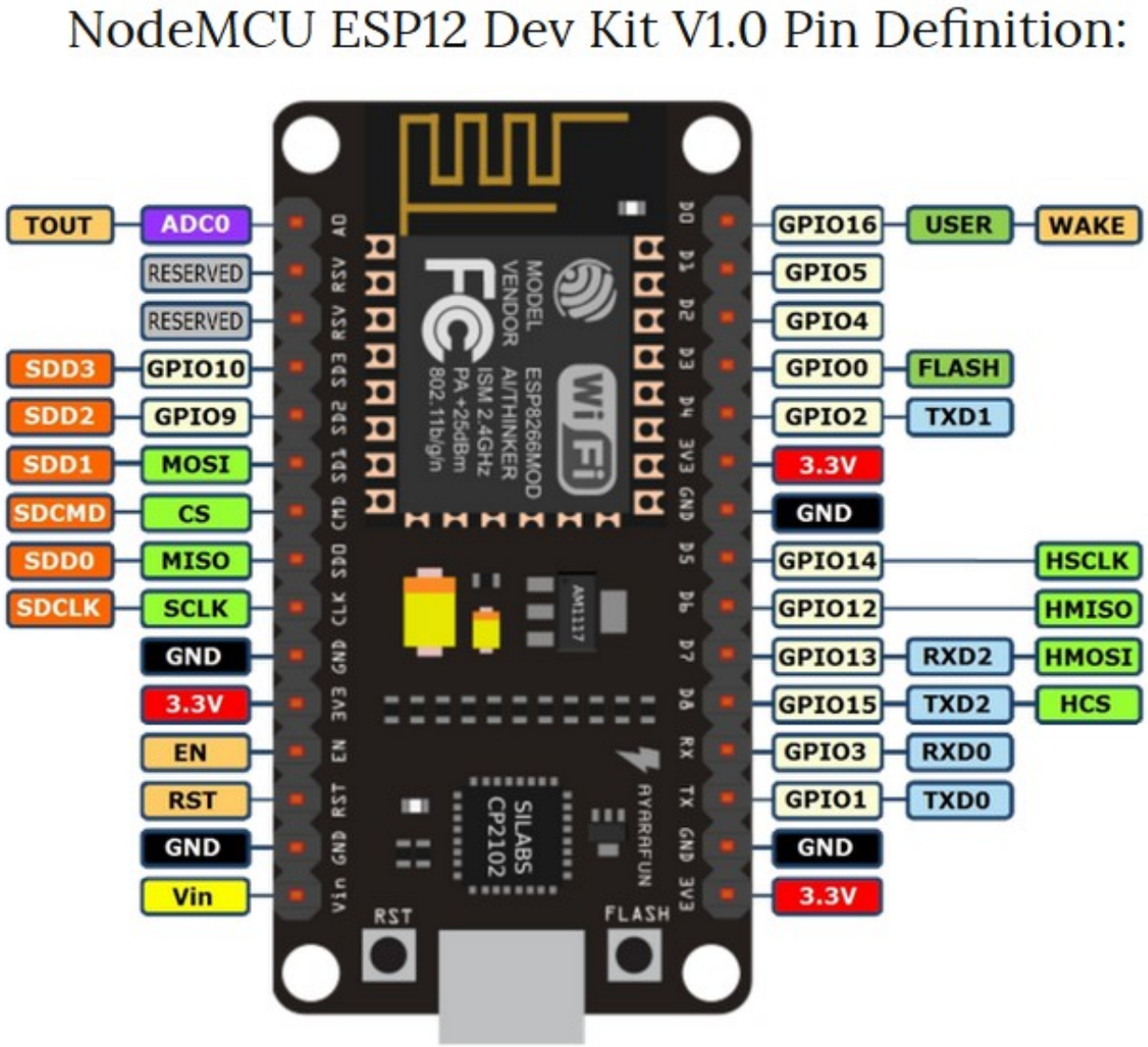




4. Hálózatkezelés microPythonnal, ESP8266 kártyán

NodeMCU kártya

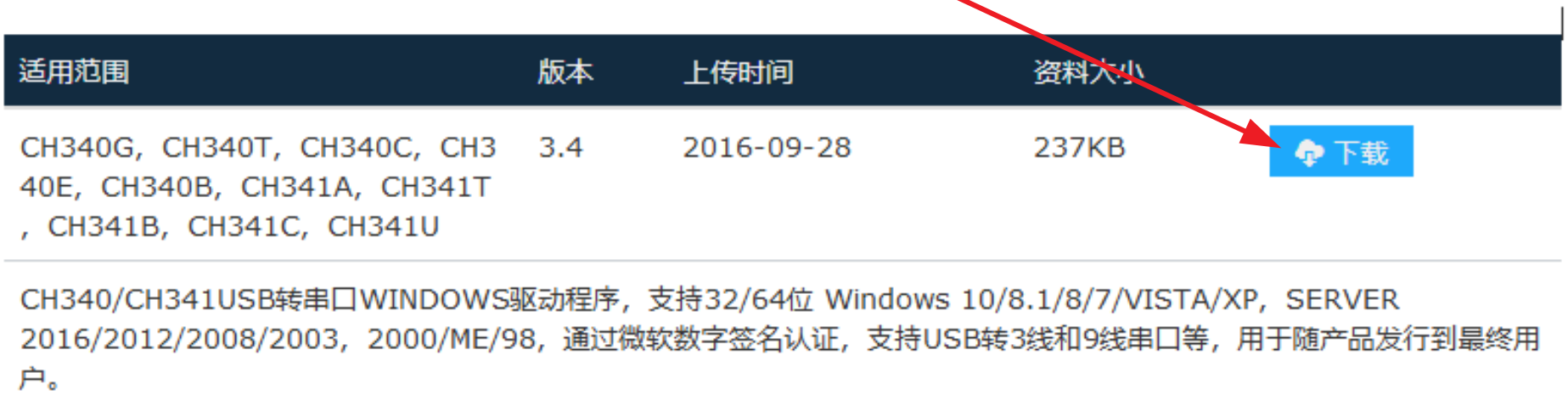
- ESP8266 CPU
  80 / 160 MHz
- 4 MB flash
- WiFi 2.4 GHz
- 1 analog input
- 13 digital I/O
  ebből az UART
  Rx és Tx foglalt
- 3,3 V jelszint és
  tápfeszültség
- MicroPythonban
  a GPIO számozásokat
  kell használni!



USB – soros kommunikáció

- A NodeMCU kártya egy USB-soros átalakítót is tartalmaz, melynek használatához a megfelelő meghajtó programot telepíteni kell
  - CP2012: Silabs usb-uart bridge VCP drivers
  - CH340: CH341SER_EXE letöltés
- Windows esetén a **Letöltés gombra** kell kattintani!

CH341SER.EXE



- Linux vagy MAC esetén a **CH341SER_LINUX.ZIP** vagy a **CH341SER_MAC.ZIP** állomány kiválasztása után kattintsunk!

MicroPython telepítése Esptool-lal

- A **MicroPython** firmware letöltése (esp8266-20180511-v1.9.4.bin)
- Az Esptool csomag telepítése (ha a gépünkön már van Python):

```
pip install esptool
```

- Feltöltés/törlés előtt kapcsoljunk flash módba a NodeMCU kártyát! (FLASH gomb lenyomva, miközben megnyomjuk a RESET gombot)
- A flash memória törlése (esetünkben COM9 portra került az eszköz)

```
esptool.py --port COM9 erase_flash
```

- A firmware feltöltése (egy sorba írjuk, csak itt nem fért ki!)

```
esptool.py --port COM9 --baud 460800 write_flash
--flash_size=detect 0 esp8266-20180511-v1.9.4.bin
```

- A feltöltés sebessége szükség esetén lehet kisebb (pl. 115200)
- A flash mérete (esetünkben 4 MB) konkrétan is megadható, pl. `--flash_size=4` formában

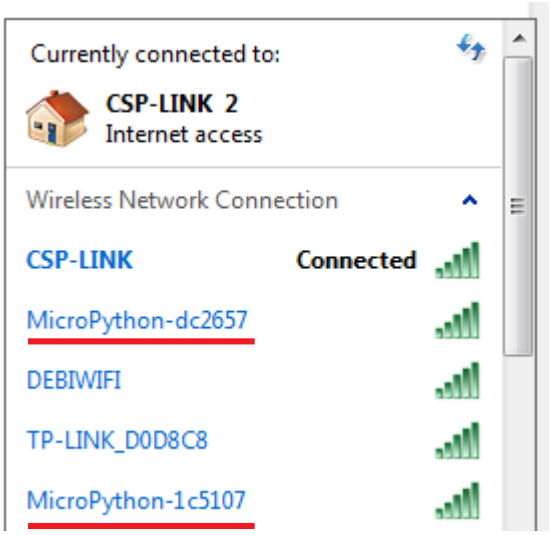
WebREPL – drótnélküli terminálkapcsolat

- **REPL** – Read, Evaluate, Print Loop (a soros kapcsolaton)
- **WebREPL** – ugyanez, WiFi kapcsolaton keresztül (csak érdekességként)
- Első alkalommal engedélyezni és konfigurálni kell:

```
import webrepl_setup
```

```
>>> import webrepl_setup
WebREPL daemon auto-start status: disabled

Would you like to (E)nable or (D)isable it running on boot?
(Empty line to quit)
> E
To enable WebREPL, you must set password for it
New password: python
Confirm password: python
Changes will be activated after reboot
Would you like to reboot now? (y/n) y
>>>
```



- A konfigurálás után csatlakozhatunk a kívánt AP eszközhöz (a **MicroPython-xxxxxx** nevekben az utolsó 6 karakter a MAC cím utolsó 6 karakterével egyezik meg.
  Az alapértelmezett jelszó: **micropythoN** (az utolsó karakter nagybetű)

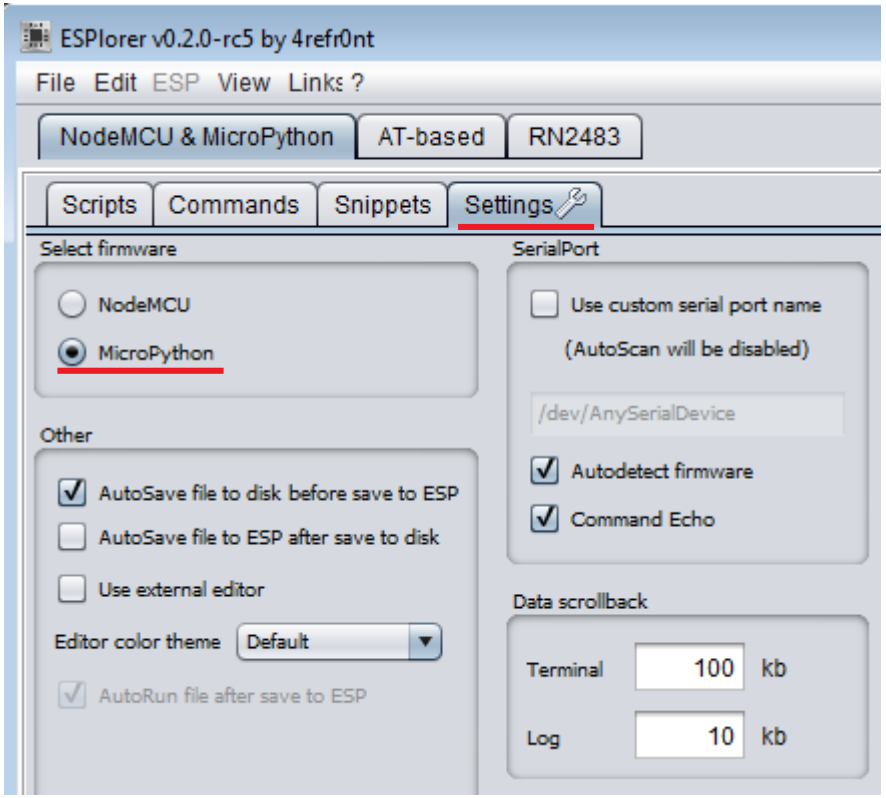
WebREPL – drótnélküli terminálkapcsolat

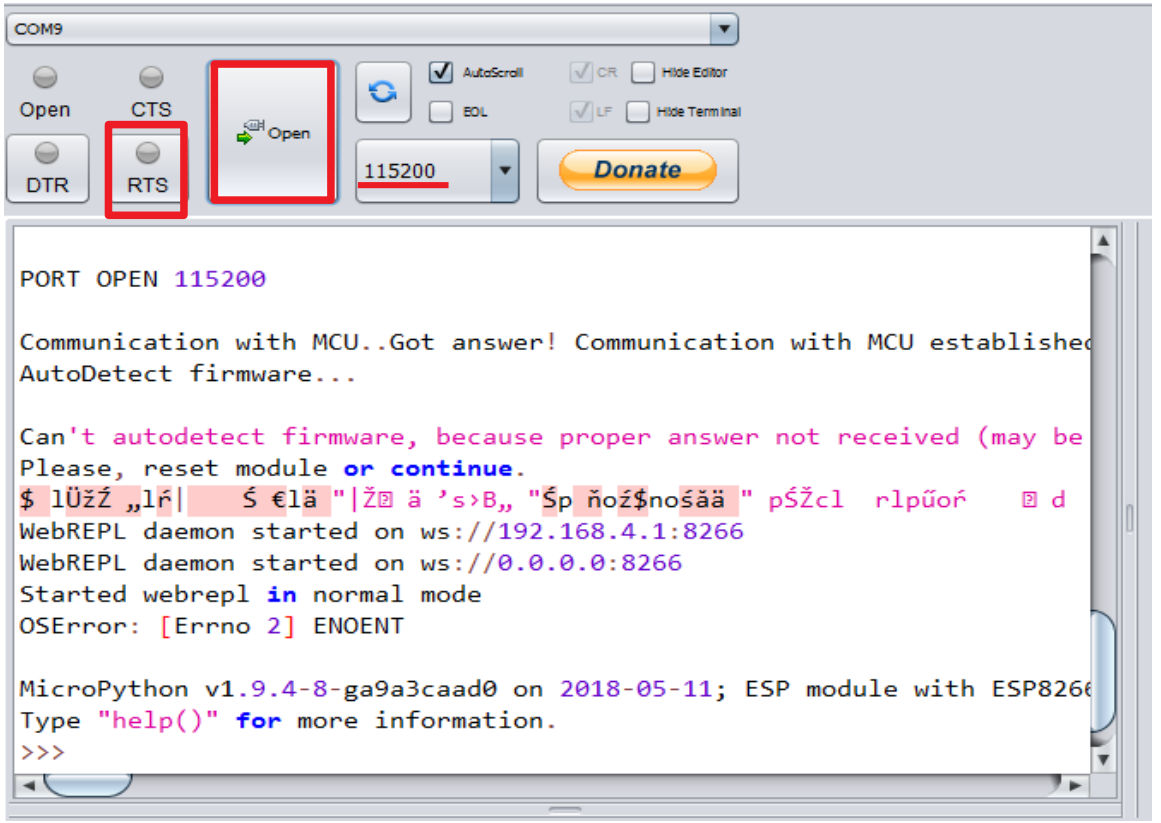
- Töltsük le a ***webrepl.html*** weblapot (offline használatra)!
  Link1: micropython.org/webrepl/
  Link2: github.com/micropython/webrepl/archive/master.zip
- Nyissul meg a webrepl.html lapot egy böngészővel!
- Ha a számítógépet már csatlakoztattuk az ESP8266-t aWiFi hálózatra akkor kattintsunk a **Connect** gombra és adjuk meg a korábban megadott felhasználói belépési jelszavunkat!
- Sikeres belépés után a böngésző ablakban parancsokat írhatunk be

forrás: learn.adafruit.com/micropython-basics-esp8266-webrepl/access-webrepl

Esplorer – az összkomfortos környezet

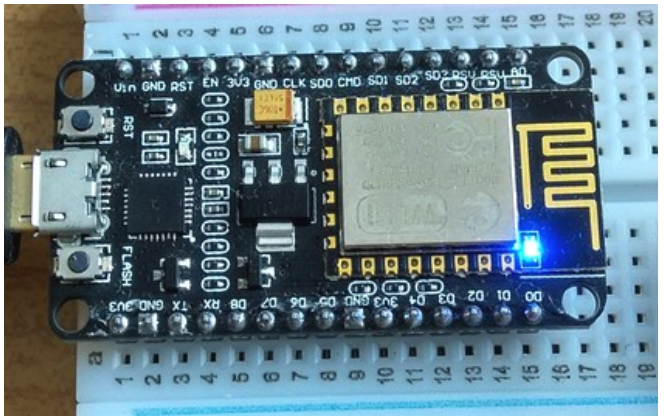
- **Esplorer:** Java-ban írt fejlesztői környezet ESP8266 fejlesztésekhez. vezetékes (USB – UART) kapcsolaton
- A **Settings** menüben választhatjuk ki a **MicroPython** módot
- Kattintsunk az **Open** gombra, majd kétszer az **RTS** gombra!





Esplorer - snippets

- Az **Esplorer** Snippet0 – Snippet15 gombjaihoz rövid (néhány soros) programokat rendelhetünk, melyekkel kipróbálhatjuk, vagy beállíthatjuk a kártyát
- Az alábbi kód (Blink) a **GPIO2** kivezetésre csatlakozó (beépített) LED-et villogtatja négyszer:



```
from machine import Pin          # GPIO kezeléshez
import time                      # Késleltetéshez
led = Pin(2, Pin.OUT)            # GPIO2 legyen kimenet
for i in range(4):               # i = 0,1,2,3
    print('LED ON')              # Kiírás
    led.value(0)                 # GPIO2 lehúz (LED ég)
    time.sleep(1)                # 1 s késleltetés
    print('LED OFF')             # kiírás
    led.value(1)                 # GPIO2 felhúz (LED nem ég)
    time.sleep(1)                # 1 s késleltetés
print("All done.")               # kiírás, for ciklus lejárt
```

Programfejlesztés a Scripts ablakban

- A beírt kód soronkén, blokkonként, vagy egészében átküldhető a mikrovezérlőnek, kipróbálhatjuk
- A kész kódot elmenthetjük, vagy a flash memóriába feltölthetjük
- Az alábbi példában egy WiFi hálózatpásztázás eredményét jelenítjük meg
  (forrás: github.com/casperp/esp8266_micropython_wifi_scan)

```
1  import network
2  import time
3  nic = network.WLAN(network.STA_IF)
4  try:
5    wlan_list = nic.scan()
6    time.sleep(5)
7  except:
8    wlan_list = [['NONE','NONE','NONE','NONE','NONE','NONE']]
9  for i in wlan_list:
10     name = str(i[0], 'utf8')
11     strength = str(i[3]) + ' dBm'
12     channel = 'Channel: ' + str(i[2])
13     status = i[4]
14     print(name, strength,channel,status)
```

```
apuci76 -80 dBm Channel: 1 4
DIGI-TJ9r -87 dBm Channel: 1 4
TP-LINK_D0D8C8 -59 dBm Channel: 6 4
szabina -83 dBm Channel: 6 3
CSP-LINK -69 dBm Channel: 6 4
MicroPython-dc2657 -42 dBm Channel: 6 4
Zold_Pont -90 dBm Channel: 6 3
DIRECT-UE-BRAVIA -73 dBm Channel: 6 3
Krassoi -88 dBm Channel: 7 3
DIGI-76aC -90 dBm Channel: 7 4
DEBIWIFI -68 dBm Channel: 11 4
DIGI-7Pe5 -74 dBm Channel: 11 4
Telekom-Y4FbWw -86 dBm Channel: 13 3
>>>
```

Csatlakozás a hálózathoz

- Az ESP8266 kártya **kliens végpont** (station) és szolgáltatást nyújtó **Access point** (hozzáférési pont) is lehet.
- A ***network modul*** függvényei és metódusai az alábbiak szerint használhatók (ssid és password helyére az aktuális értéket kell írni!)
- Az AP alapértelmezett IP címe: 192.168.4.1

```
import network

wlan = network.WLAN(network.STA_IF)    # create station interface
wlan.active(True)                      # activate the interface
wlan.scan()                            # scan for access points
wlan.isconnected()                     # check if the station is connected to an AP
wlan.connect('ssid', 'password')       # connect to an AP
wlan.config('mac')                     # get the interface's MAC adddress
wlan.ifconfig()                        # get IP/netmask/gw/DNS addresses

ap = network.WLAN(network.AP_IF)       # create access-point interface
ap.active(True)                        # activate the interface
ap.config(essid='ESP-AP')              # set the ESSID of the access point
```

Csatlakozás a helyi hálózatra

- Egy hasznos függvény a helyi hálózatra történő csatlakozásra (forrás: docs.micropython.org/en/latest/esp8266/quickref.html)

```
def do_connect():
    import network
    wlan = network.WLAN(network.STA_IF)
    wlan.active(True)
    if not wlan.isconnected():
        print('connecting to network...')
        wlan.connect('essid', 'password')
        while not wlan.isconnected():
            pass
    print('network config:', wlan.ifconfig())
```

- Az első csatlakozás után a flash memóriában rögzítésre kerülnek a csatlakozás paraméterei, legközelebb automatikusan csatlakozik az ESP8266!

ntptime – a pontos idő lekérdezése

- Az ntptime modul lehetővé teszi a pontos idő lekérdezését a megadott szerverről. Egyúttal az RTC-t is beállítja
- **ntptime.time()** a 2000.01.01-től eltelt UTC idő, másodpercekben
- **ntptime.settime()** dátum és idő tuplet, pl. (2018, 11, 14, 13, 28, 3, 2, 318)

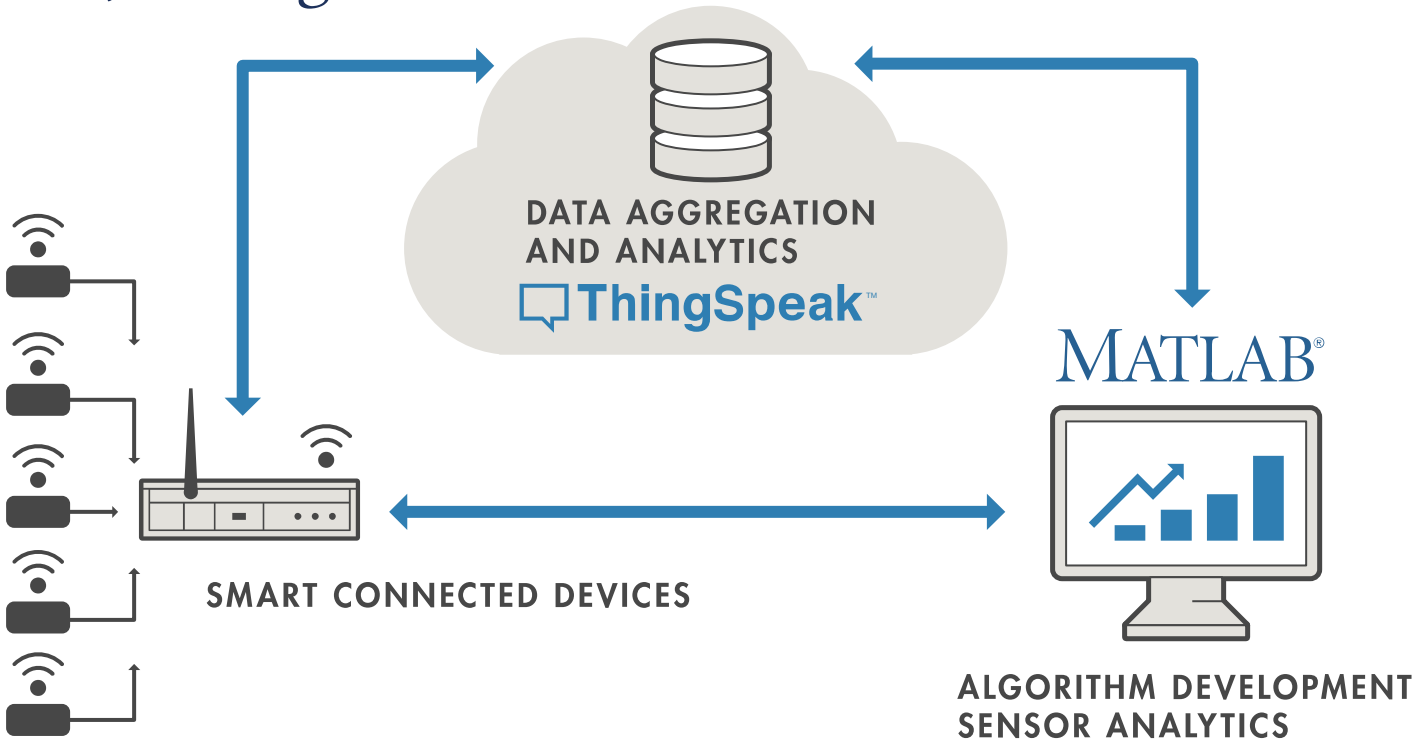
```
import ntptime
import time
from machine import RTC
rtc = RTC()
ntptime.host="hu.pool.ntp.org"
dummy=ntptime.settime() # timeserver lekérdezése
days = ['Hétfő','Kedd','Szerda','Csütörtök','Péntek',\
'Szombat','Vasárnap']
for i in range(20):
    yy,mm,dd,wd,h,m,s,q=rtc.datetime() #RTC lekérdezés
    print("%d-%02d-%02d  %02d:%02d:%02d %s" %\
    (yy,mm,dd,h+1,m,s,days[wd]))
    time.sleep(1)
```

```
2018-11-14  14:20:12 Szerda
2018-11-14  14:20:13 Szerda
2018-11-14  14:20:14 Szerda
2018-11-14  14:20:15 Szerda
2018-11-14  14:20:16 Szerda
...
```

- **RTC.datetime()** dátum és idő tuplet: *(yy,mm,dd,wd,h,m,s,ms)*

Thingspeak – IoT felhő

- Mi az IoT? Internet of Things, azaz a dolgok Internetje. Különféle szenzorok, adatgyűjtő eszközök küldhetnek adatokat privát csatornákba, melyeket a szerver tárol és megjelenít.
- Az adatok publikus vagy egyéni beállításban megtekinthetők, lekérdezhetők és akár MatLab-bal vagy számolótáblával elemezhetők, feldolgozhatók



Adatküldés HTTP protokollal

- Ha regisztráltunk, akkor létrehozhatunk csatornákat, amelyeknek egyedi azonosítója van. Például: thingspeak.com/channels/34 244
- Csatornánként 1 – 8 mezőt definiálhatunk (pl. egy DHT22 szenzor esetén field1 = hőmérséklet, field2 = páratartalom)
- Regisztráláskor kapunk egy vagy több API kulcsot. Adatot csak ennek birtokában tudunk beküldeni (xxxxxxx helyére az API kulcs kell!)

```
GET https://api.thingspeak.com/update?api_key=xxxxxxx&field1=adat1&filed2=adat2
```

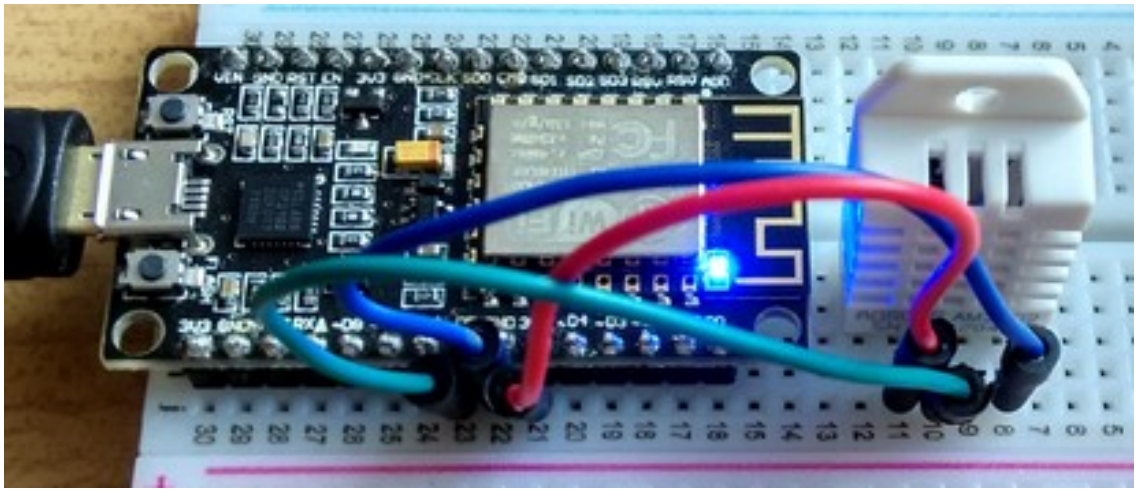
- A legegyszerűbben az *urequests* könyvtári modullal tölthetjük fel adatainkat, de az alábbi primitív példa nem védett a hibák ellen:

```
import urequests
r = urequests.get('https://api.thingspeak.com/update?\
api_key=DVDXXXXXXXETD&field1=22&field2=44')
print(r.text)
r.close()
```

„hasból” beírt adatok

Adatküldés HTTP protokollal

- Egy használhatóbb változat programlistája itt látható:

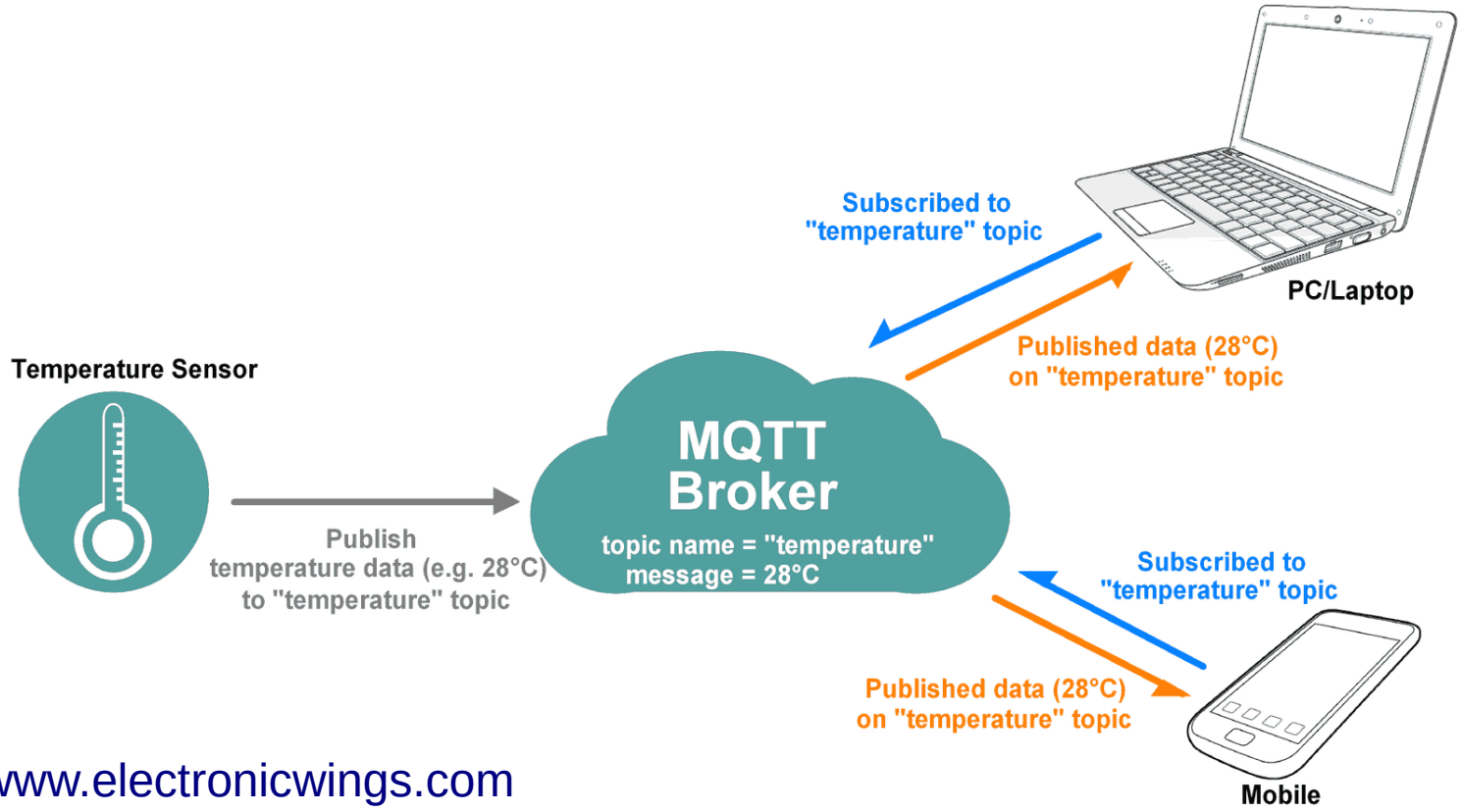


```
from machine import Pin
from dht import DHT22
import time
import urequests
DELAY = 80
d = DHT22(Pin(14)) # D5 láb
while True:
    d.measure()
    t = d.temperature()
    h = d.humidity()
    print('temperature = %.2f  humidity    = %.2f' % (t,h))
    payload = "&field1="+str(t)+"&field2="+str(h)
    topic = 'https://api.thingspeak.com/update?api_key=*******'
    try:
        r = urequests.get(topic+payload)
        print(r.text); r.close()
    except:
        print('failed')
    time.sleep(DELAY)
```

Saját API kód ide!

Adatküldés MQTT protokollal

- **MQTT** (Message Queuing Telemetry Transport) egy szabványos TCP/IP protokoll
- A kommunikációhoz kell egy lokális vagy globális szerver (a ”bróker”), amelyhez a kliensek adatközlésre (publish) vagy adatlekérésre (subscribe) iratkozhatnak fel



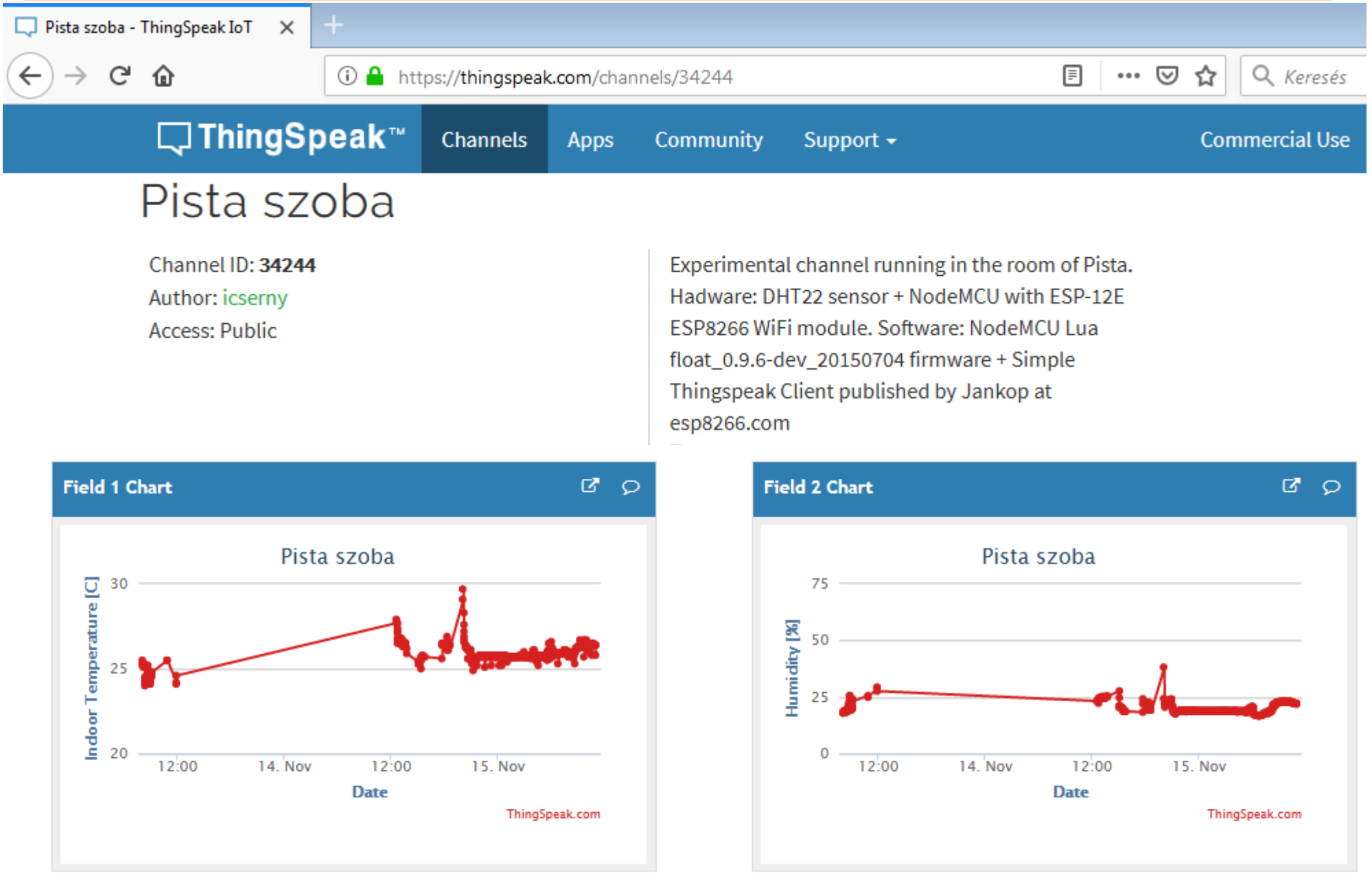
Kép forrása: www.electronicwings.com

Adatküldés MQTT protokollal

- Használjuk a ThingSpeak szervert MQTT módban! Ehhez az umqtt.simple modul MQTTClient objektumára lesz szükség.
- A küldendő adat a ***topic*** (az azonosítónk) és a ***payload*** (az adatok)

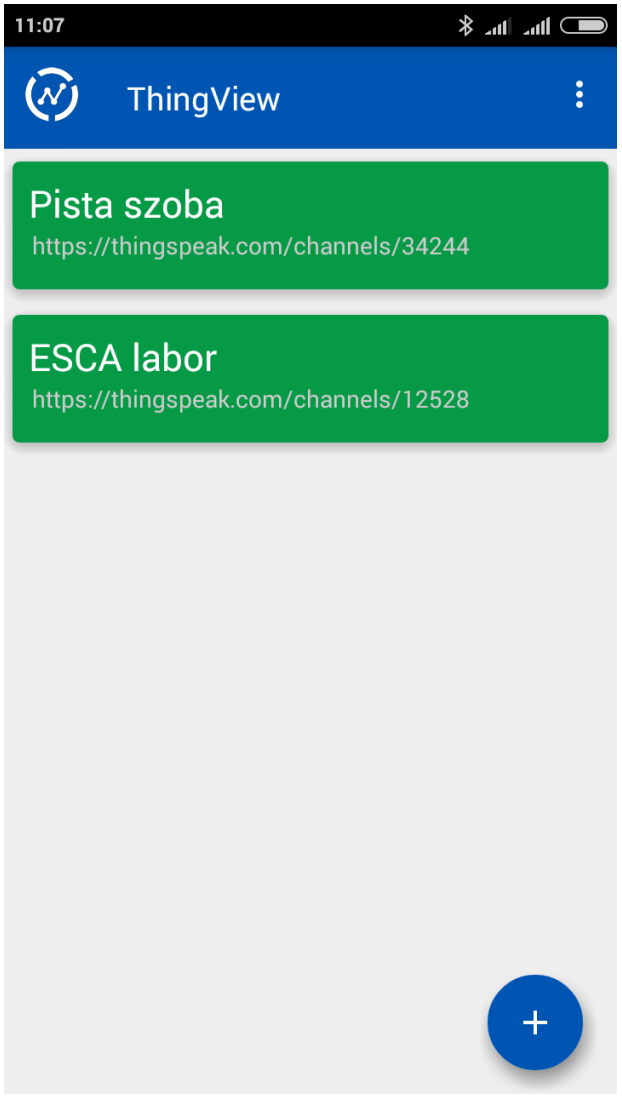
```
import dht, machine, time
from umqtt.simple import MQTTClient
DELAY = 80
DHT22_PIN = 14
SERVER = "mqtt.thingspeak.com"
client = MQTTClient("umqtt_client", SERVER)
CHANNEL_ID = "34244"
WRITE_API_KEY = "************"
topic = "channels/" + CHANNEL_ID + "/publish/" + WRITE_API_KEY
d = dht.DHT22(machine.Pin(DHT22_PIN))  # DHTT22 a GPIO14 (D5) lábon
while True:
    d.measure(); t = d.temperature(); h = d.humidity()
    print('temperature = %.2f' % t)
    print('humidity    = %.2f' % h)
    payload = "field1="+str(t)+"&field2="+str(h)
    client.connect()
    client.publish(topic, payload)
    client.disconnect()
    time.sleep(DELAY)
```

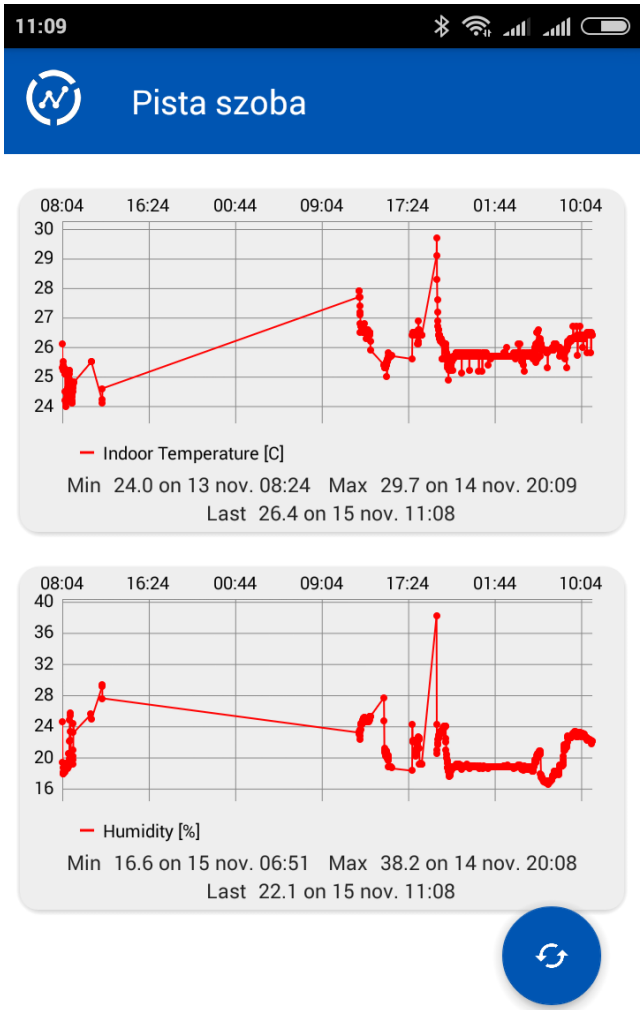
Az eredmény megtekintése böngészőben

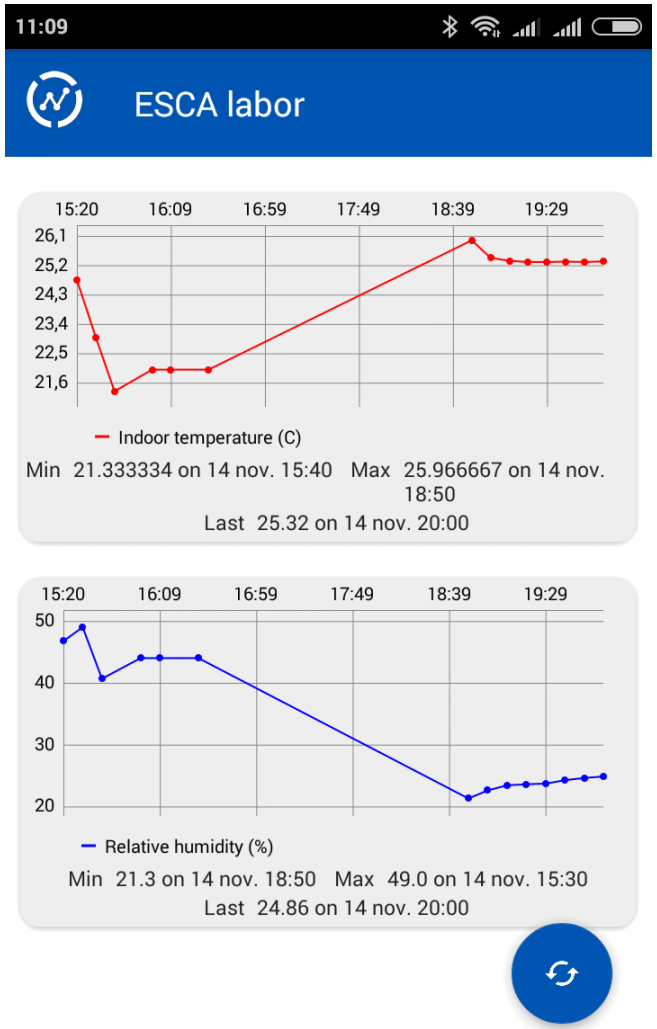


Az eredmény megtekintése applikációban

- **ThingView** - ThingSpeak megjelenítő (Google Play áruház)

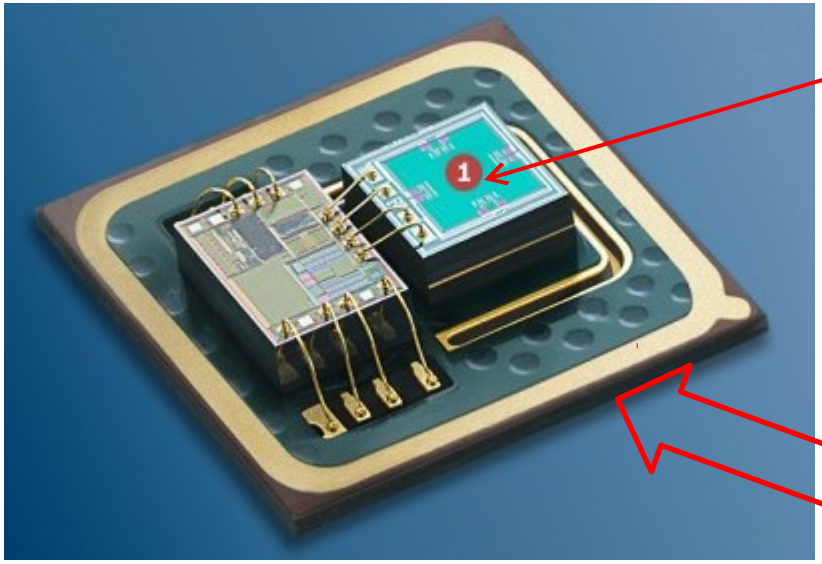






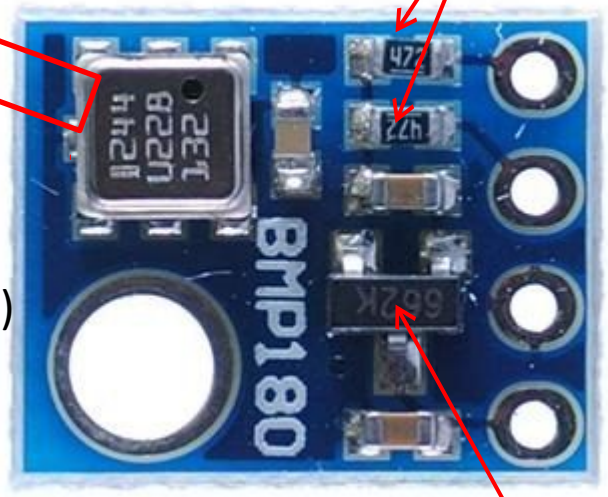
Légnyomás mérése BMP180 szenzorral

- A Bosch BMP180 szenzor egy I2C periféria, amely kész modul formájában is kapható (pl. Arduinohoz)



Piezorezisztív nyúlásmérő, amely a nyomás hatására bekövetkező deformációt érzékeli

Felhúzó ellenállások



SDA

SCL

GND

VIN (+5V)

Feszültségstabilizátor (3,3V)

Bosch SensorTec BMP180

Nyomásmérés: 300 – 1100 hPa (9000 - -300m)
Tápfeszültség: 1,8 – 3,6 V
Áramfelvétel: 5 µA (1 mintavétel/s esetén)
Kis zaj: 0.06hPa (0.5m) kisfogyasztású mód
0.02hPa (0.17m) nagyfelbontású mód
Jellemzők: Hőmérő, I2C felület, gyárilag kalibrált

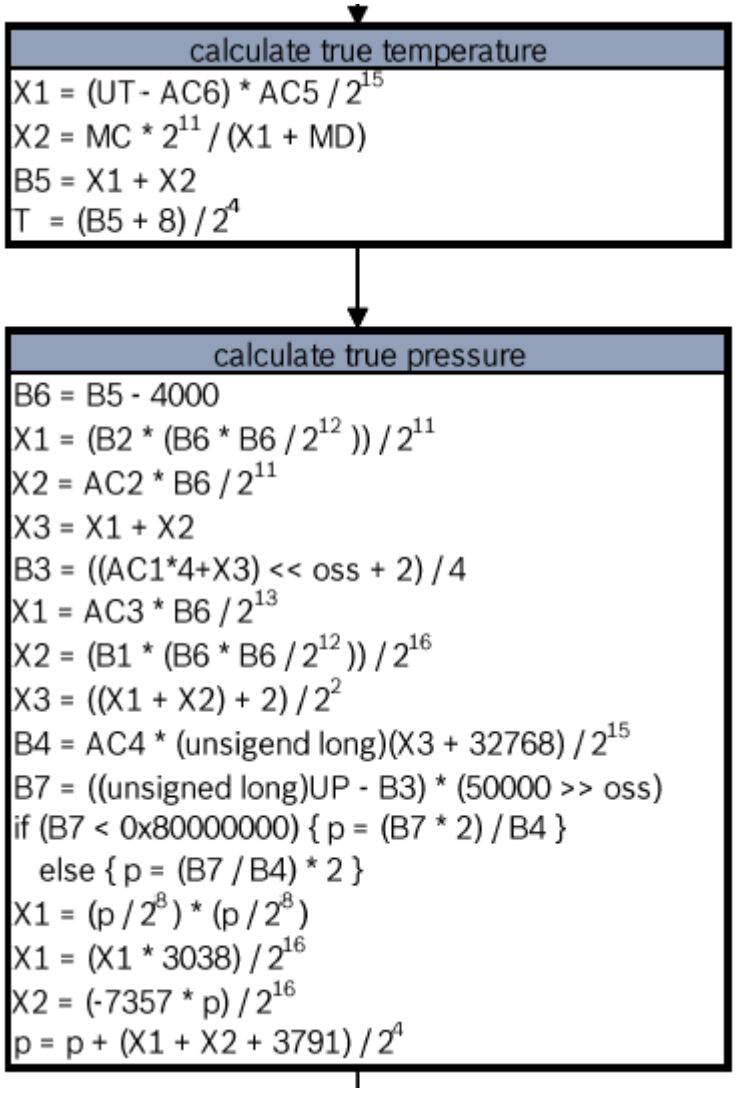
A kalibrált értékek meghatározása

- A nyers adatokból és a szenzor memóriájában tárolt kalibrációs adatokból e képletekkel számolhatjuk ki a nyomást és a hőmérsékletet

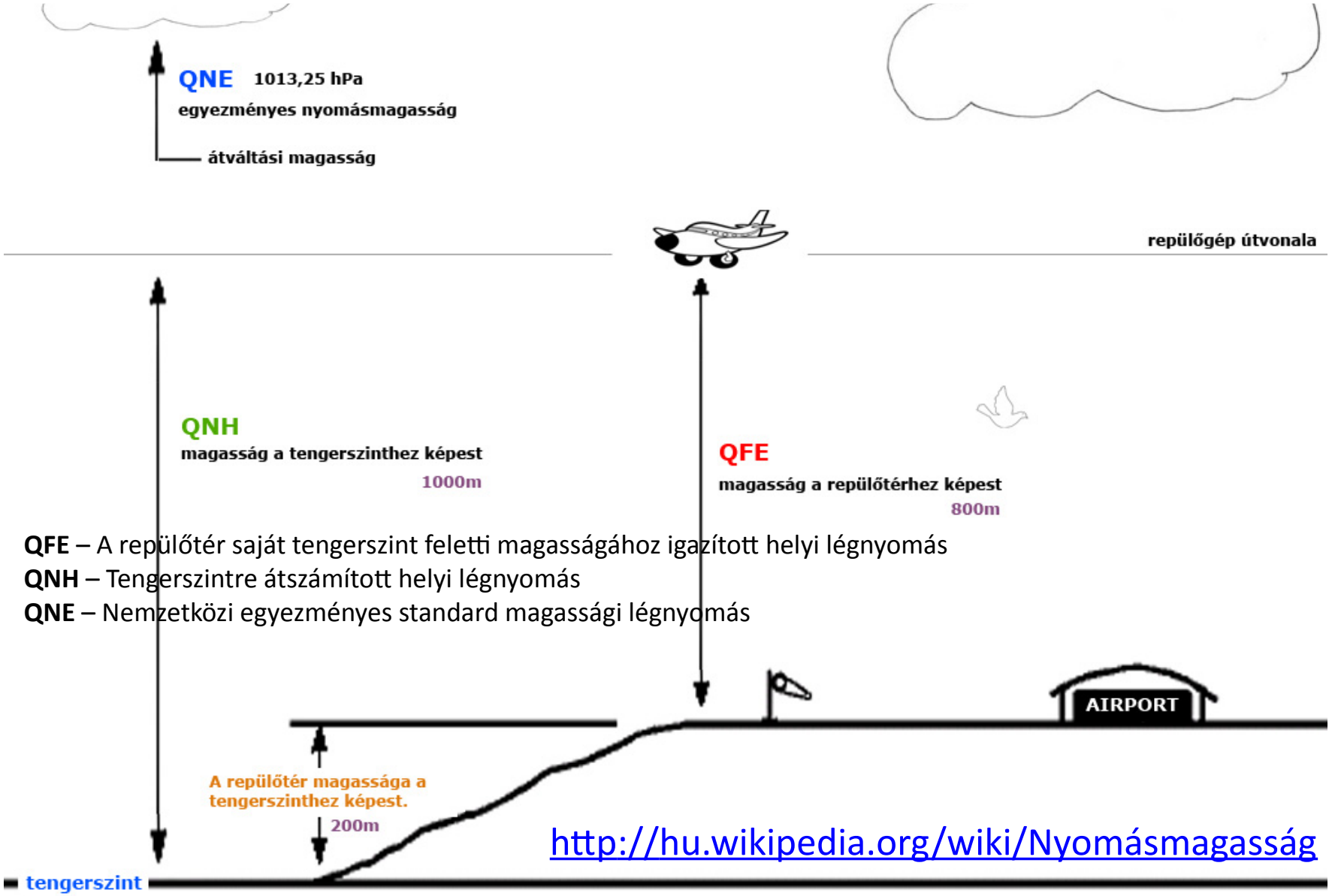
Table 5: Calibration coefficients

	BMP180 reg adr	
Parameter	**MSB**	**LSB**
AC1	0xAA	0xAB
AC2	0xAC	0xAD
AC3	0xAE	0xAF
AC4	0xB0	0xB1
AC5	0xB2	0xB3
AC6	0xB4	0xB5
B1	0xB6	0xB7
B2	0xB8	0xB9
MB	0xBA	0xBB
MC	0xBC	0xBD
MD	0xBE	0xBF

UT és **UP** a nyers hőmérséklet és nyomás adat
oss – oversampling (0 – 3)



– Milyen magasan repül a repülő?
– Mihez képest?



QFE – A repülőtér saját tengerszint feletti magasságához igazított helyi légnyomás
QNH – Tengerszintre átszámított helyi légnyomás
QNE – Nemzetközi egyezményes standard magassági légnyomás

http://hu.wikipedia.org/wiki/Nyomásmagasság

Helyi légnyomás átszámítása tengerszintre

- QFE ⟶ QNH átszámítás *(az alábbiakban p = QFE, P0 = QNH)*
- Tudnunk kell a helyi tengerszint feletti magasságot (altitude) és a szenzorral meg kell mérnünk az abszolút helyi nyomás értékét *(p)*.
- A tengerszintre átszámított légnyomás (p0) a következő képlettel számítható ki:

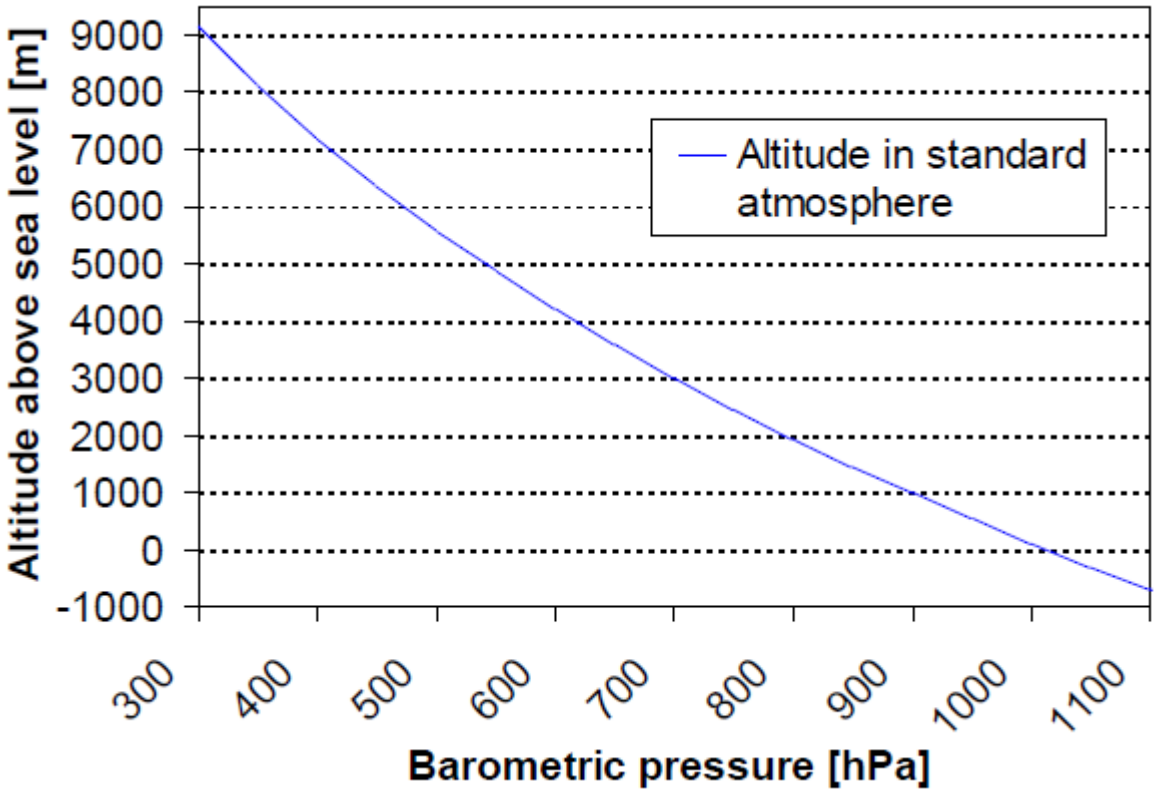
$$p_0 = \frac{p}{\left(1 - \frac{\text{altitude}}{44330}\right)^{5.255}}$$

- **Egyszerű(bb) közelítés:** Debrecenben (kb. 120 m) 1440 Pa-t hozzáadunk a mért légnyomáshoz. Vagy alkalmazzuk a *p0 = p*1.0153 114 576* összefüggést (altitude = 120 m esetre)

A magasság meghatározása

$$\text{magasság} = 44330 * \left(1 - \left(\frac{p}{p_0}\right)^{\frac{1}{5.255}}\right)$$

Ahol **p** az általunk mért nyomás *(QFE)*, $\mathbf{p_0}$ pedig a tengerszinti nyomás *(QNE)* (pl. 1013.25 hPa)



- A gyakorlatban a légnyomás nemcsak a magasságtól, hanem a meteorológiai viszonyoktól is függ (hőmérséklet, páratartalom, stb.)

A bmp180.py könyvtári modul

- **bmp180.py** egy micropython modul a Bosch **BMP180** szenzorhoz.
- A hőmérsékletet és a helyi nyomást méri, amiből az egyezményes nyomásmagasság is meghatározható
- Helyhiány miatt csak a tagfüggvények neveit soroljuk fel. A teljes forrásállomány itt található: github.com/micropython-IMU/micropython-bmp180
- Felhasználási engedély: The MIT License (MIT)
  Copyright (c) 2014 Sebastian Plamauer, oeplse@gmail.com

```
def compvaldump(self):              # a számítás részeredményeit adja meg
def makegauge(self):                # a nyers mérési adat kiolvasása
def blocking_read(self):            # Blokkoló olvasás
def oversample_sett(self, value): # túlmintavételezés beállítása
def oversample_sett(self):          # túlmintavételezési beállítás kiolvasása
def temperature(self):              # Hőmérséklet kiszámítása
def pressure(self):                 # nyomás kiszámítása
def altitude(self):                 # Egyezményes nyomásmagasság meghatározása
```

Fájlok feltöltése az ampy programmal

- Először installálnunk kell az **adafruit-ampy** csomagot (feltételezzük, hogy a **Python** már telepítve van a számítógépünkön)

  pip install adafruit-ampy

- Fájlok feltöltése (pl. az előző oldalon említett **bmp180.py**)

  ampy --port COM9 put bmp180.py

- Fájl kiolvasása és lementése:

  ampy --port COM9 get boot.py saved_boot.py

- Fájl eltávolítása:

  ampy --port COM9 rm main.py

- Fájlok listázása:

  ampy --port COM9 ls

bmp180_demo.py

- ESP8266 esetén szoftveres I2C vezérlést használunk
  SDA = GPIO4 (D1), SCL = GPIO5 (D2) választással
- Kiírjuk: hőmérséklet, helyi nyomás, magasság, tengerszintre korrigált nyomás

```
from bmp180 import BMP180
from machine import I2C, Pin
import time
i2c = I2C(scl=Pin(5), sda=Pin(4), freq=100000) #I2C busz konfiguálás
bmp180 = BMP180(i2c)                           #Példányosítás
bmp180.oversample_sett = 3                     #Túlmintavételezés (0, 1, 2, 3)
bmp180.baseline = 101325                       #Egyezményes tengerszinti nyomás
print("Temp Pressure Altitude Press. corrected")
while True:
    temp = bmp180.temperature
    p = bmp180.pressure
    p0 = p*1.0153114576                        #Korrekció 128 m-es magasságra
    altitude = bmp180.altitude
    print("T=%.1f C  QFE=%.2f hPa Alt=%.f m  QNH=%.2f hPa" \
    % (temp, p/100, altitude, p0/100))
    time.sleep(5)
```

Futási eredmény

- A futási eredményekből jól látható, hogy a meteorológiai viszonyoktól függően az egyezményes nyomásmagasság jelentősen eltér a tényleges magasságtól (10-11 m, 128m helyett)

```
Temp Pressure Altitude Press. corrected
T=26.7 C  QFE=1011.82 hPa Alt=11 m  QNH=1027.31 hPa
T=26.8 C  QFE=1011.88 hPa Alt=11 m  QNH=1027.37 hPa
T=26.8 C  QFE=1011.92 hPa Alt=11 m  QNH=1027.41 hPa
T=26.7 C  QFE=1011.83 hPa Alt=11 m  QNH=1027.32 hPa
T=26.7 C  QFE=1011.99 hPa Alt=10 m  QNH=1027.48 hPa
T=26.6 C  QFE=1011.71 hPa Alt=12 m  QNH=1027.20 hPa
T=26.6 C  QFE=1011.82 hPa Alt=11 m  QNH=1027.31 hPa
T=26.6 C  QFE=1011.80 hPa Alt=11 m  QNH=1027.29 hPa
```