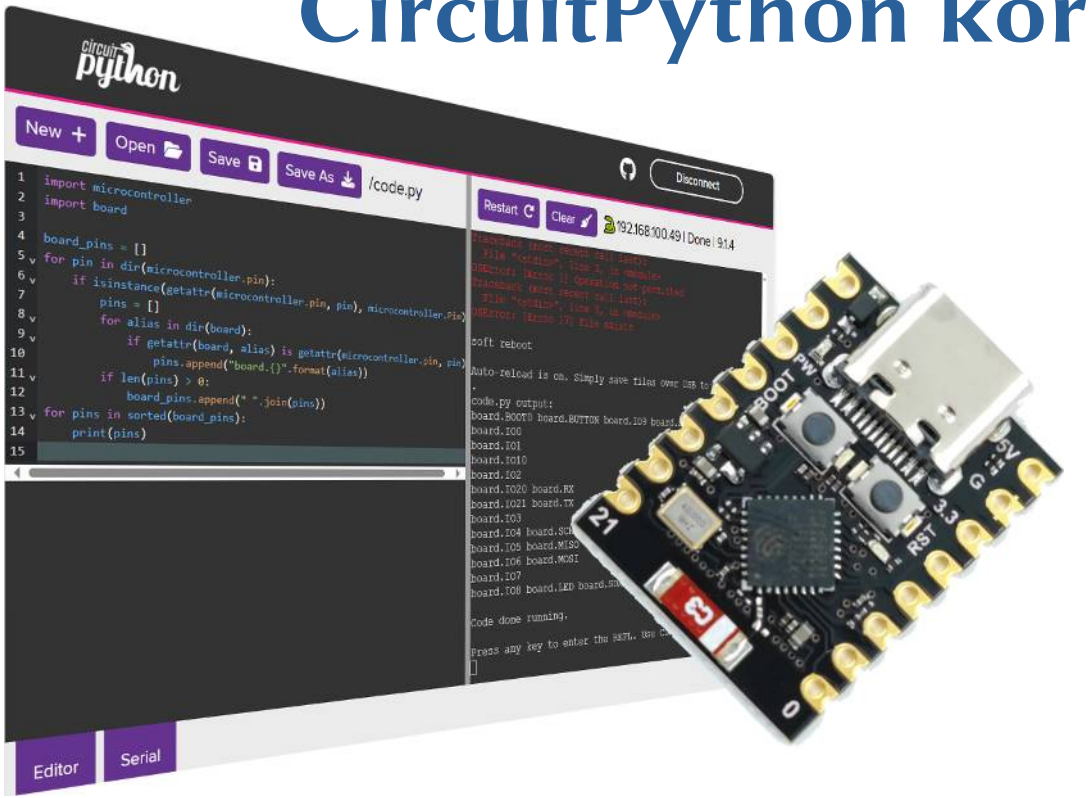
ESP32-C3 mikrovezérlők programozása CircuitPython környezetben



7. A WiFi modul használata

Felhasznált és ajánlott irodalom

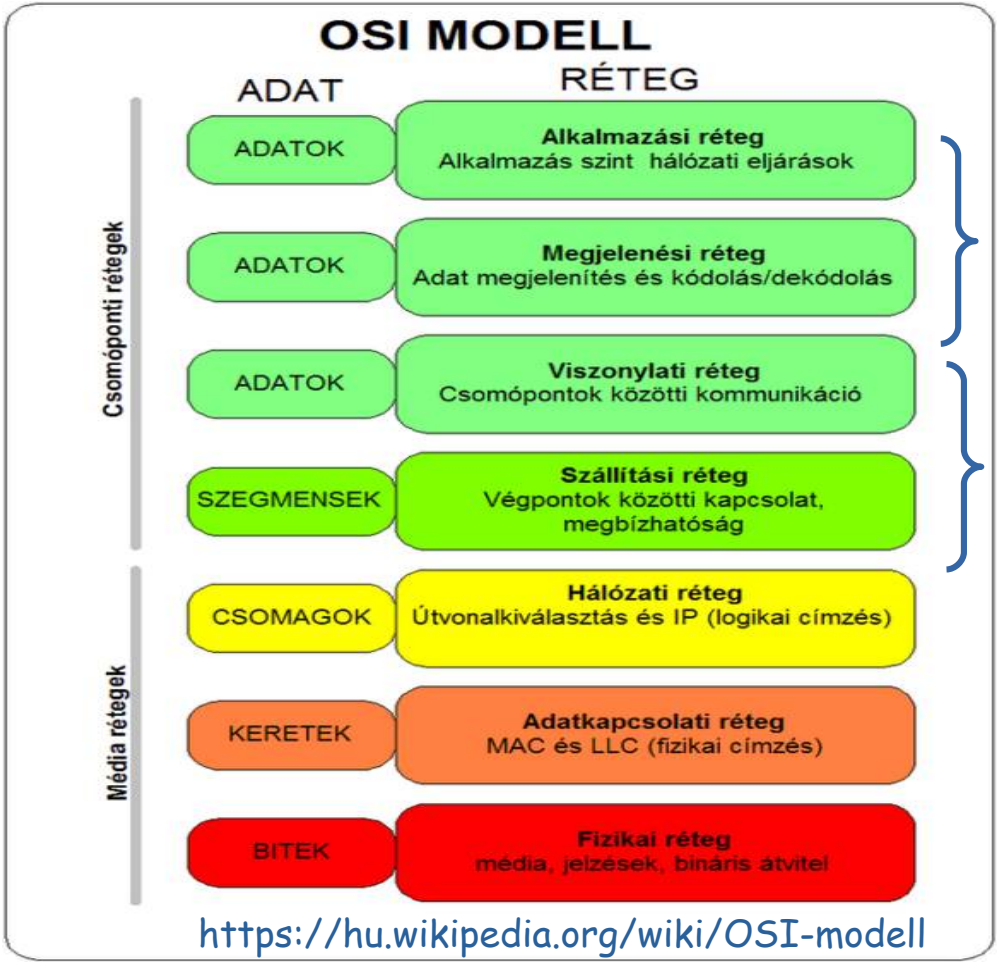
- **Python:**
  - Mark Pilgrim/Kelemen Gábor: Ugorj fejest a Python 3-ba!
  - P. Wentworth et al. (ford. Biró Piroska, Szeghalmy Szilvia és Varga Imre): Hogyan gondolkozz úgy, mint egy informatikus: Tanulás Python 3 segítségével
- **CircuitPython:**
  - Adafruit: https://circuitpython.org/downloads
  - Learn Adafruit: Welcome to CircuitPython
  - Learn Adafruit: CircuitPython Essentials
  - Adafruit: Adafruit CircuitPython API Reference
  - Adafruit: github.com/adafruit/Adafruit CircuitPython Bundle

- **Online eszközök és támogatás:**
  - Learn Adafruit: CircuitPython on ESP32 Quick Start
  - Adafruit: Adafruit Web Serial ESPTool
  - Adafruit: CircuitPython Code Editor

A hálózatkezelés alapfogalmai

- **MAC cím:** Egyedi eszközazonosító, amely az eszközt azonosítja a hálózaton (pl. 00:1A:2B:3C:4D:5E). Ezt a címet a gyártó rendeli az eszközhöz
- **IP címek:** Publikus cím az interneten való globális eléréshez (pl. 216.58.194.174), lokális cím a helyi hálózaton belüli kommunikációhoz (pl. 192.168.1.100)
- **DNS:** Domain név szerver. Tartományneveket fordít IP címekre, például: www.cspista.hu → 185.80.49.249. A DNS-ek globális hálózata teszi ezt lehetővé
- **Gateway:** Hálózati átjáró. Egyik hálózati szegmensből a hálózat többi részébe, vagy a világhálóba való adatkapcsolatot biztosító eszköz
- **Router:** Útvonalválasztó. A WAN (külső hálózat) és a LAN (helyi hálózat) között kezeli a forgalmat. Az otthoni router egyúttal DNS és DHCP szerver is…
- **Access Point (AP):** Hozzáférési pont, amely lehetővé teszi a WiFi (vezeték nélküli) eszközök, mint pl. az ESP32-C3, csatlakozását a helyi hálózathoz
- **UDP** és **TCP/IP csomagok:** Információcsomagok, amelyek az adatokat szállítják az interneten, Hosszabb üzenet csomagokra tördelve kerül továbbításra

Az OSI-modell (ISO 7498-1)



- Az **Open Systems Interconnection** (nyílt rendszerek összekapcsolása) referenciamodellje hét rétegbe szervezve írja le a hálózati kapcsolatot

HTTP, FTP, SMTP, Telnet, **NTP,** NFS

TCP, UDP

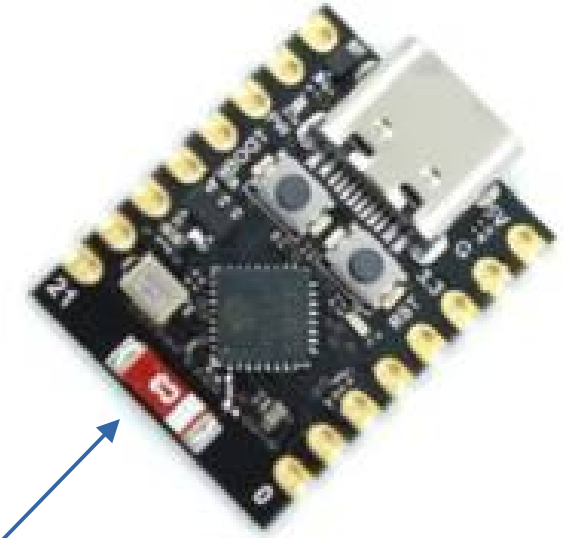
IP (**IPv4** – IPv6), ARP, IPSEC, ...

Ethernet, **WiFi,** PPP, ...

RS-232, V35, DSL, ISDN, 10BASE-T, 100BASE-TX stb.

ESP32-C3 Supermini: Wi-Fi modul

- **Protokoll támogatás:** 802.11 b/g/n a 2,4 GHz-es frekvenciasávon
- **Átviteli sebesség:** Maximális sebesség 150 Mbps (a 802.11n protokollal)
- **Támogatott üzemmódok:**
  - **Station (STA):** Csatlakozik egy meglévő hálózathoz
  - **Access Point (AP):** Hozzáférési pontként működik
  - **STA+AP:** Vegyes mód, ahol egyszerre csatlakozhat hálózathoz és nyújthat hálózati hozzáférést
- **Biztonság:** WPA/WPA2/WPA3 titkosítás, amely biztosítja a biztonságos kapcsolódást
- **Hatótáv:** Általában 30–50 méter beltéren



Hálózatkezelés CircuitPythonnal

- Bár a CircuitPythonnal kompatibilis kártyák száma folyamatosan nő, a hálózatkezelésre képes hardverek támogatása csak néhány gyártóra szorítkozik:
  - Expressif ESP32
  - Raspberry Pi Pico W
  - Adafruit Airlift (ESP32/SPI)
  - Wiznet W5000, W5500

} **wifi** modul

adafruit_esp32spi modul

adafruit_wiznet5k modul

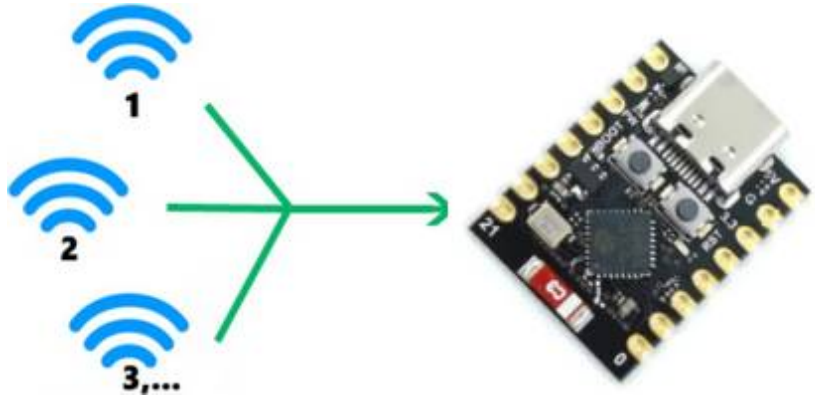
- A mai előadásban csak az ESP32 beépített WiFi moduljának kezelésével foglalkozunk, amelynek az elsődleges kezelője a CircuitPython firmware-be beépített **wifi** könyvtár
- Az alkalmazásainkban használt különféle protokollok kezelésére azonban kiegészítő könyvtárakat is kell majd telepíteni

A wifi modul használata

- A CircuitPython **wifi** modul az alapvető hálózati kapcsolódásra összpontosít, mint például a hálózathoz való csatlakozás és az IP-cím megszerzése

Mit csinálhatunk a wifi modullal, külső könyvtárak nélkül?

- **Wi-Fi hálózatok keresése**
  Lehetőséget nyújt a környező Wi-Fi hálózatok felderítésére és információik lekérdezésére
- **Csatlakozás Wi-Fi hálózathoz**
  Az alapvető Wi-Fi műveletek közé tartozik a vezeték nélküli hálózathoz való csatlakozás. Ehhez is a **wifi** modult használjuk



- **IP-címek, MAC-cím és egyéb információk lekérdezése**
  A **wifi** modullal különféle hálózati információkat is lekérdezhetünk, mint például az IP-cím, a MAC-cím vagy a csatlakozás állapota…

Wi-Fi hálózatok keresése

wifi_demo.py (részletek)

```
import wifi

print("Elérhető Wi-Fi hálózatok keresése...")
for network in wifi.radio.start_scanning_networks():
    print(f"SSID: {network.ssid}, Jelerősség: {network.rssi} dBm")

wifi.radio.stop_scanning_networks()  # Ne felejtsd el leállítani a keresést
```

- **start_scanning_networks():** Ez a funkció elindítja a hálózatkeresést, és visszaad egy listát az elérhető Wi-Fi hálózatokról
- **stop_scanning_networks():** Leállítja a hálózatkeresést, miután már megszerezted az információkat
- A **wifi.Network** osztály paraméterei:
  - **ssid** – a hálózat neve
  - **bssid** – az AP MAC címe
  - **country** – országkód
  - **authmode** – autentikációs mód
  - **rssi** – jelerősség

```
code.py output:
Elerheto Wi-Fi halozatok keresese...
SSID: HUAWEI-2.4G-9bQR, Jelerosseg: -73 dBm
SSID: DIRECT-UE-BRAVIA, Jelerosseg: -75 dBm
SSID: Telekom-1SrW9Z, Jelerosseg: -88 dBm
SSID: DIGI-8Usb, Jelerosseg: -90 dBm
SSID: T-E7CD88, Jelerosseg: -94 dBm
SSID: DIGI-vKHx, Jelerosseg: -92 dBm
SSID: DIGI_e5ac18, Jelerosseg: -83 dBm
SSID: DIGI-77fS, Jelerosseg: -95 dBm

Code done running.
```

A settings.toml állomány

- A „környezeti változók” – különösen a hálózati kapcsolódásokkal kapcsolatos adatokat a **settings.toml** állományban célszerű tárolni
- Egy tipikus **settings.toml** beállítás látható az alábbi ábrán:

```
CIRCUITPY_WIFI_SSID = "HUAWEI-2.4G-9bQR"
CIRCUITPY_WIFI_PASSWORD = "mypassword"
CIRCUITPY_WEB_API_PASSWORD = "mypass"
THINGSPEAK_API_KEY = "**************"
```

WiFi paraméterek

Web Browser jelszó

Thingspeak write key

- A környezeti változók az **os.getenv("név")** metódussal vehetők elő:

```
import os, wifi

ssid = os.getenv("CIRCUITPY_WIFI_SSID")
password = os.getenv("CIRCUITPY_WIFI_PASSWORD")
wifi.radio.connect(ssid, password)
```

Csatlakozás Wi-Fi hálózathoz

- A kapcsolódáshoz szükséges a hálózat neve (SSID) és a jelszó, mintapéldánkban ezeket a **settings.toml** fájlból olvassuk ki
- A sikeres kapcsolódás után lekérdezhetjük az ESP32-C3 által beszerzett IP-címet

wifi_demo.py (részletek)

```
import wifi
import os
# Wi-Fi hálózat elérése környezeti változókból
ssid = os.getenv("CIRCUITPY_WIFI_SSID")
password = os.getenv("CIRCUITPY_WIFI_PASSWORD")

# Csatlakozás a Wi-Fi hálózathoz
print(f"Csatlakozás a {ssid} hálózathoz...")
wifi.radio.connect(ssid, password)
# Ha sikeres a csatlakozás, kiírja az IP címet
print("Csatlakozás sikeres!")
print("IP cím: ", wifi.radio.ipv4_address)
```

```
Csatlakozás a HUAWEI-2.4G-9bQR hálózathoz...
Csatlakozás sikeres!
IP cím:  192.168.100.33
```

Hálózati információk lekérése

wifi_radio.py

A legnagyobb jelerősségű hálózat lesz elől

```
import os, wifi
networks = []
for network in wifi.radio.start_scanning_networks():
    networks.append(network)
wifi.radio.stop_scanning_networks()
networks = sorted(networks, key=lambda net: net.rssi, reverse=True)
for network in networks:
    print("ssid:",network.ssid, "ch:", network.channel, "rssi:",network.rssi)

print("connecting...")
wifi.radio.connect(ssid=os.getenv('CIRCUITPY_WIFI_SSID'),
                   password=os.getenv('CIRCUITPY_WIFI_PASSWORD'))
print("my IP addr:", wifi.radio.ipv4_address)
print("MAC addr:", wifi.radio.mac_address)
print("gateway:", wifi.radio.ipv4_gateway)
print("mgateway ap:", wifi.radio.ipv4_gateway_ap)
print("subnet:", wifi.radio.ipv4_subnet)
print("DNS", wifi.radio.ipv4_dns)
print("hostname:", wifi.radio.hostname)
print("tx_power:", wifi.radio.tx_power)
```

Hálózati információk lekérése

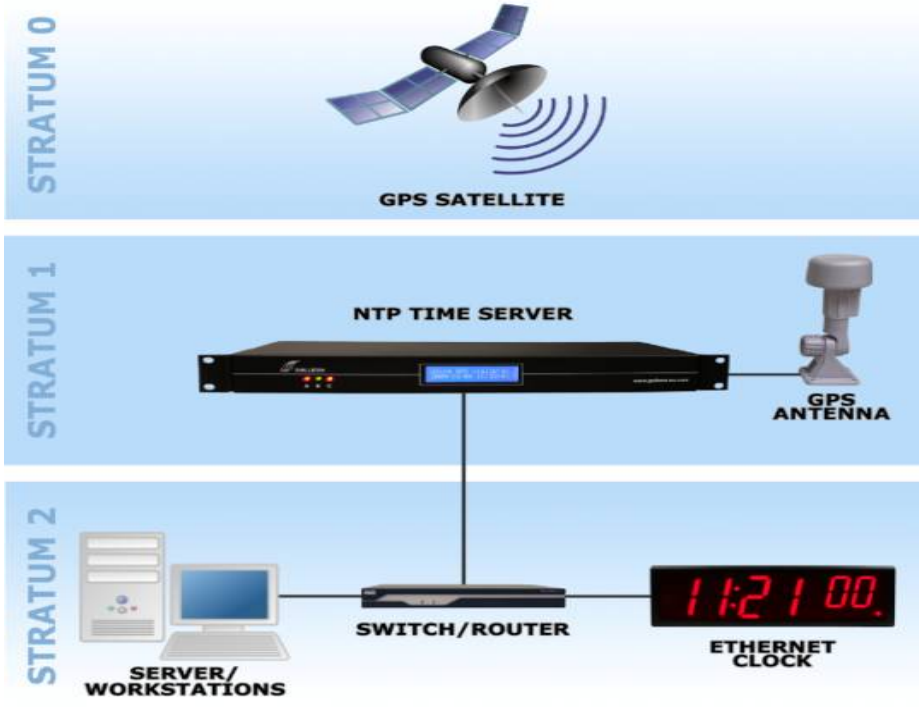
```
code.py output:
ssid: HUAWEI-2.4G-9bQR ch: 9 rssi: -74
ssid: DIGI_e5ac18 ch: 5 rssi: -83
ssid: DIGI-8Usb ch: 1 rssi: -90
ssid: DIGI-jX6H ch: 7 rssi: -90
ssid: Telekom-969415 ch: 7 rssi: -90
ssid: Telekom-qKomV6 ch: 1 rssi: -93
ssid: DIGI-vKHx ch: 4 rssi: -94
ssid: DIGI-77fS ch: 11 rssi: -94
ssid: Telekom-1SrW9Z ch: 1 rssi: -95
ssid: Digi46 ch: 4 rssi: -96
connecting...
my IP addr: 192.168.100.33
MAC addr: 9c9e6ec31a68
gateway: 192.168.100.1
mgateway ap: None
subnet: 255.255.255.0
DNS 192.168.100.1
hostname: cpy-32c3_supermini-9c9e6ec31a68
tx_power: 11.0
```

Mit NEM tudunk csinálni külső könyvtárak nélkül?

- **HTTP kérések:** Ha egy weboldallal vagy API-val szeretnénk kommunikálni (adatküldés/lekérés), szükségünk lesz az **adafruit_requests** könyvtárra, amely kezeli a **GET, POST, PUT, DELETE HTTP** kéréseket
- **SSL/TLS titkosított kapcsolatok:** Ha titkosított (HTTPS) kapcsolatot akarunk használni, szükségünk lesz egy SSL/TLS kezelésre képes könyvtárra, mint például az **ssl** vagy az **adafruit_requests.** Az alap **wifi** modul önmagában nem biztosít titkosított kapcsolatokat.
- **Speciális hálózati funkciók:** Az alap **wifi.radio** modul ugyan kezeli az IP-címek hozzárendelését, a hálózathoz való csatlakozást, alapvetően a domain nevek feloldása is működik, de speciális hálózati funkciók (pl. DHCP szerver, IPv6, VLAN) nem elérhetők beépített eszközökkel

Pontos idő lekérdezése NTP kéréssel

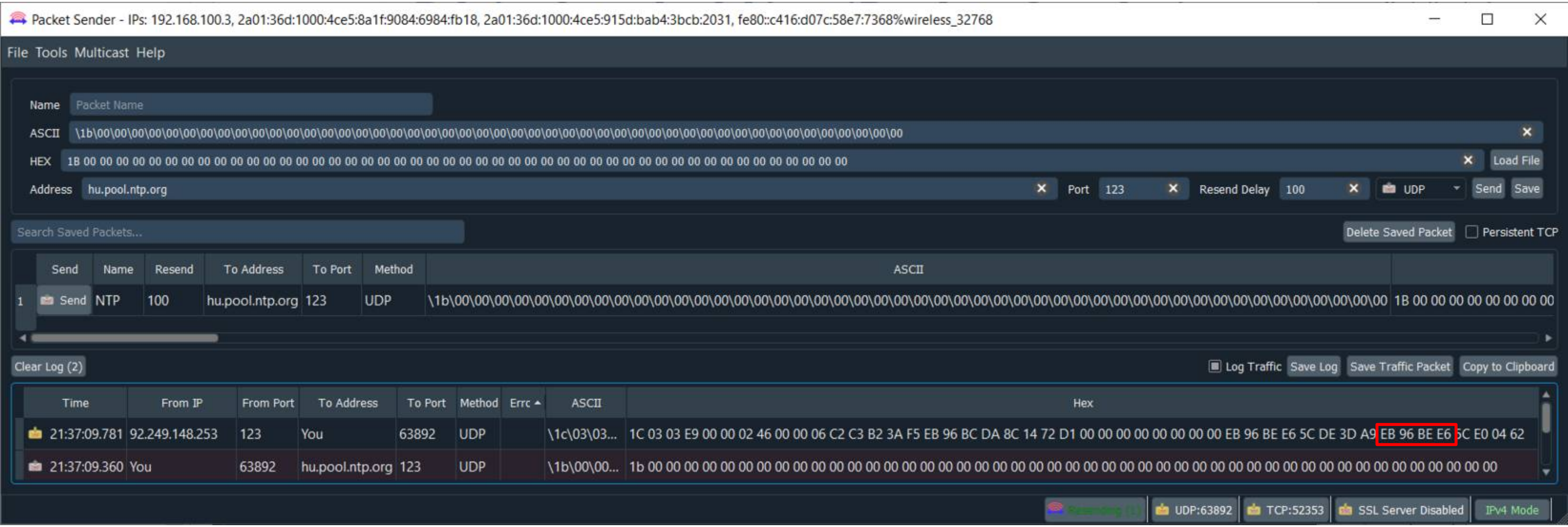
- A világot behálózó **NTP** (Network Time Protocol) szerverek UDP csomagokkal (User Datagram Protocol) kommunikálnak
- Az **NTP** szerver lehet helyi vagy távoli
- A nyilvános szervereket az **pool.ntp.org** fogja össze (lásd: https://www.ntppool.org/en/)
- Az **NTP** szerverek általában a 123-as portot használják
- Az üzenetcsomag formátumát (amely többnyire 48 bájt) és a protokollt az **RFC958** írja le
- A legegyszerűbb **NTP** kérelem: 0x1B és 47 db nulla



Kép forrása: www.galsys.co.uk/news/

NTP kérelem küldése (Packet Sender)

- A **Packet Sender** alkalmazás segítségével itt a számítógépről küldtünk NTP kérelmet a hu.pool.ntp.org címre, a kapott válasz 40.-41.-42.-43. bájtja az 1900. január 1. 0^h óta eltelt időt adja meg, másodpercekben. Például: 0xEB96BEE6 a 2025-04-01 19:37:10 időpontnak felel meg



Network Time Protocol (NTP) lekérések

- Bár NTP lekérést a **wifi** modullal is végezhetnénk, kényelmesebb utat választunk: az **adafruit_ntp** programkönyvtárral végezzük az NTP lekéréseket
- `class` **`adafruit_ntp.NTP`**`(socketpool, *, server: str = '0.adafruit.pool.ntp.org', port: int = 123, tz_offset: float = 0, socket_timeout: int = 10, cache_seconds: int = 0)`
- Properties: **datetime:** `struct_time` - current time from NTP server as structure
  utc_ns: `int` – current time from NTP server as integer in nanoseconds

ntp_simpletest.py

```
import os, time, socketpool, wifi
import adafruit_ntp
wifi_ssid = os.getenv("CIRCUITPY_WIFI_SSID")
wifi_password = os.getenv("CIRCUITPY_WIFI_PASSWORD")
wifi.radio.connect(wifi_ssid, wifi_password)
pool = socketpool.SocketPool(wifi.radio)
ntp = adafruit_ntp.NTP(pool, tz_offset=2, cache_seconds=3600)

while True:
    print(ntp.datetime)
    time.sleep(1)
```

CEST: Central European Summer Time (UTC + 2)

adafruit_ntp_demo.py futási eredmény

```
code.py output:
struct_time(tm_year=2025, tm_mon=4, tm_mday=1, tm_hour=22, tm_min=5,
tm_sec=10, tm_wday=1, tm_yday=91, tm_isdst=-1)
struct_time(tm_year=2025, tm_mon=4, tm_mday=1, tm_hour=22, tm_min=5,
tm_sec=11, tm_wday=1, tm_yday=91, tm_isdst=-1)
struct_time(tm_year=2025, tm_mon=4, tm_mday=1, tm_hour=22, tm_min=5,
tm_sec=12, tm_wday=1, tm_yday=91, tm_isdst=-1)
struct_time(tm_year=2025, tm_mon=4, tm_mday=1, tm_hour=22, tm_min=5,
tm_sec=13, tm_wday=1, tm_yday=91, tm_isdst=-1)
struct_time(tm_year=2025, tm_mon=4, tm_mday=1, tm_hour=22, tm_min=5,
tm_sec=14, tm_wday=1, tm_yday=91, tm_isdst=-1)
struct_time(tm_year=2025, tm_mon=4, tm_mday=1, tm_hour=22, tm_min=5,
tm_sec=15, tm_wday=1, tm_yday=91, tm_isdst=-1)
```

Az NTP idő formázott kiírása

- Apró módosítással olvashatóbb formátumban is kiírathatjuk a dátumot és az időt
- A :02, vagy :02d formátum azt jelenti, hogy a kétszámjegyű kiírásnál az „értéktelen” nullát is kiírjuk. Például 2025-04-03, 17:00:00

```
2025-04-02
14:49:09
2025-04-02
14:49:10
2025-04-02
14:49:11
2025-04-02
14:49:12
2025-04-02
14:49:13
2025-04-02
14:49:14
```

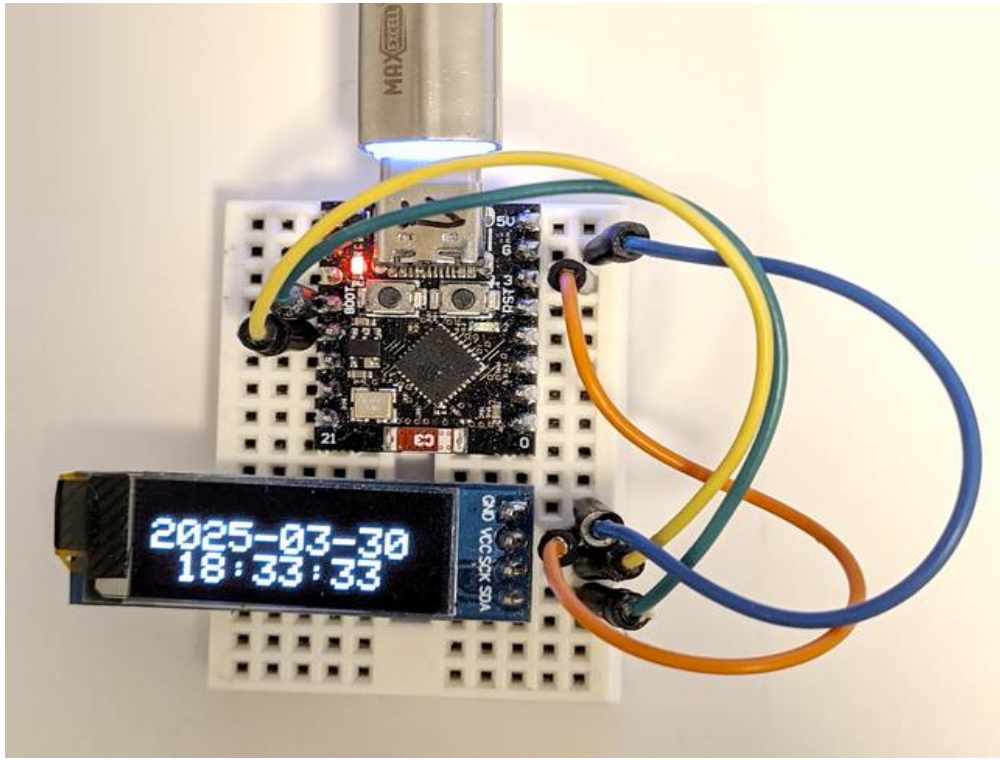
wifi_ntp_demo.py

```
import os, time, socketpool, wifi
import adafruit_ntp
wifi_ssid = os.getenv("CIRCUITPY_WIFI_SSID")
wifi_password = os.getenv("CIRCUITPY_WIFI_PASSWORD")
wifi.radio.connect(wifi_ssid, wifi_password)
pool = socketpool.SocketPool(wifi.radio)
ntp = adafruit_ntp.NTP(pool, tz_offset=2, cache_seconds=3600)

while True:
    t = ntp.datetime
    print(f"{t.tm_year}-{t.tm_mon:02}-{t.tm_mday:02}")
    print(f"{t.tm_hour:2d}:{t.tm_min:02d}:{t.tm_sec:02d}")
    time.sleep(1)
```

Készítsünk NTP órát!

- Ha az előző előadásban tanult képernyőkezelést ötvözzük a WiFi hálózatkezeléssel és az NTP időlekéréssel, akkor egy olyan órát készíthetünk, ami időnként az Internetről lekérdezi az időt, szinkronizálja a helyi időmérőt, és az OLED kijelzőn megjeleníti a dátumot, valamint az aktuális pontos időt
- A 128x32 képpont felbontású kijelzőn dupla fontmérettel 2 x 10 karaktert tudunk kiírni, így két sorba pont kifér a kiírás



ntp_clock.py

ntp_clock.py

```
import time, board, os, socketpool, wifi
import adafruit_ssd1306
from adafruit_ntp import NTP

wifi_ssid = os.getenv("CIRCUITPY_WIFI_SSID")
wifi_password = os.getenv("CIRCUITPY_WIFI_PASSWORD")
wifi.radio.connect(wifi_ssid, wifi_password)

pool = socketpool.SocketPool(wifi.radio)
ntp = NTP(pool, tz_offset=2, cache_seconds=3600)   # CEST timezone = UTC + 2 hours

i2c = board.I2C()
display = adafruit_ssd1306.SSD1306_I2C(128,32,i2c)
display.rotation = 2     # Optional rotation setting

while True:
    t = ntp.datetime     # current time
    display.fill(0)
    display.text(f"{t.tm_year}-{t.tm_mon:02}-{t.tm_mday:02}",0,0,1,size=2)    # Date
    display.text(f"{t.tm_hour:02}:{t.tm_min:02}:{t.tm_sec:02}",0,16,1,size=2) # Time
    display.show()
    time.sleep(1)
```

Kapcsolódás a helyi WiFi hálózathoz

OLED kijelző konfigurálás az előző előadás szerint
Itt a 128x32 felbontású kijelzőt használjuk

Az ESP32 C3 Super Mini kártya kivezetései