

# CircuitPython tanfolyam

The screenshot displays a Windows desktop environment. On the left, the Mu Python IDE (version 1.0.2) is open with a file named 'ledblink.py'. The code in the editor is as follows:

```
1 import board
2 import digitalio
3 import time
4
5 led = digitalio.DigitalInOut(board.LED)
6 led.direction = digitalio.Direction.OUTPUT
7
8 while True:
9     led.value = True
10    time.sleep(1.95)
11    led.value = False
12    time.sleep(0.05)
```

In the center, a physical STM32F4x1 development board is shown, featuring a USB-C connector, a push button, and various pins labeled with numbers and names like 'BOOT0', 'PWR', and 'SW'. On the right, a 'Windows Photo Viewer' window displays a 'Pinout Diagram' for the 'STM32F4x1Cx v2.0+'. The diagram includes a central image of the board with colored labels for each pin, a legend for pin functions (POWER, GROUND, CPU PIN, etc.), and a table of pin names and their corresponding functions. The legend includes categories like POWER, GROUND, CPU PIN, PIN NAME, CONTROL, ANALOG, TIMER & CHANNEL, USART, SPI / I2S, SDO (F411 Only), I2C, CAN BUS, USB, MISC, and BOARD HARDWARE. The diagram is titled 'WeAct Studio STM32F4x1Cx v2.0+ Pinout Diagram'.

## 10. Kiegészítő LCD kijelző a számítógéphez

# Felhasznált és ajánlott irodalom

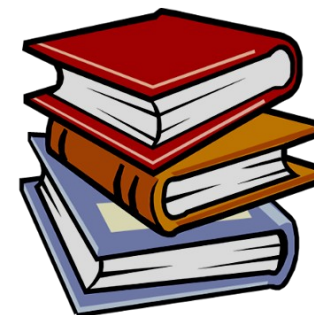
---

## 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>
- Adafruit: [Adafruit CircuitPython API Reference](#)
- Learn Adafruit: [CircuitPython Essentials](#)
- Adafruit: [github.com/adafruit/Adafruit CircuitPython Bundle](https://github.com/adafruit/Adafruit_CircuitPython_Bundle)



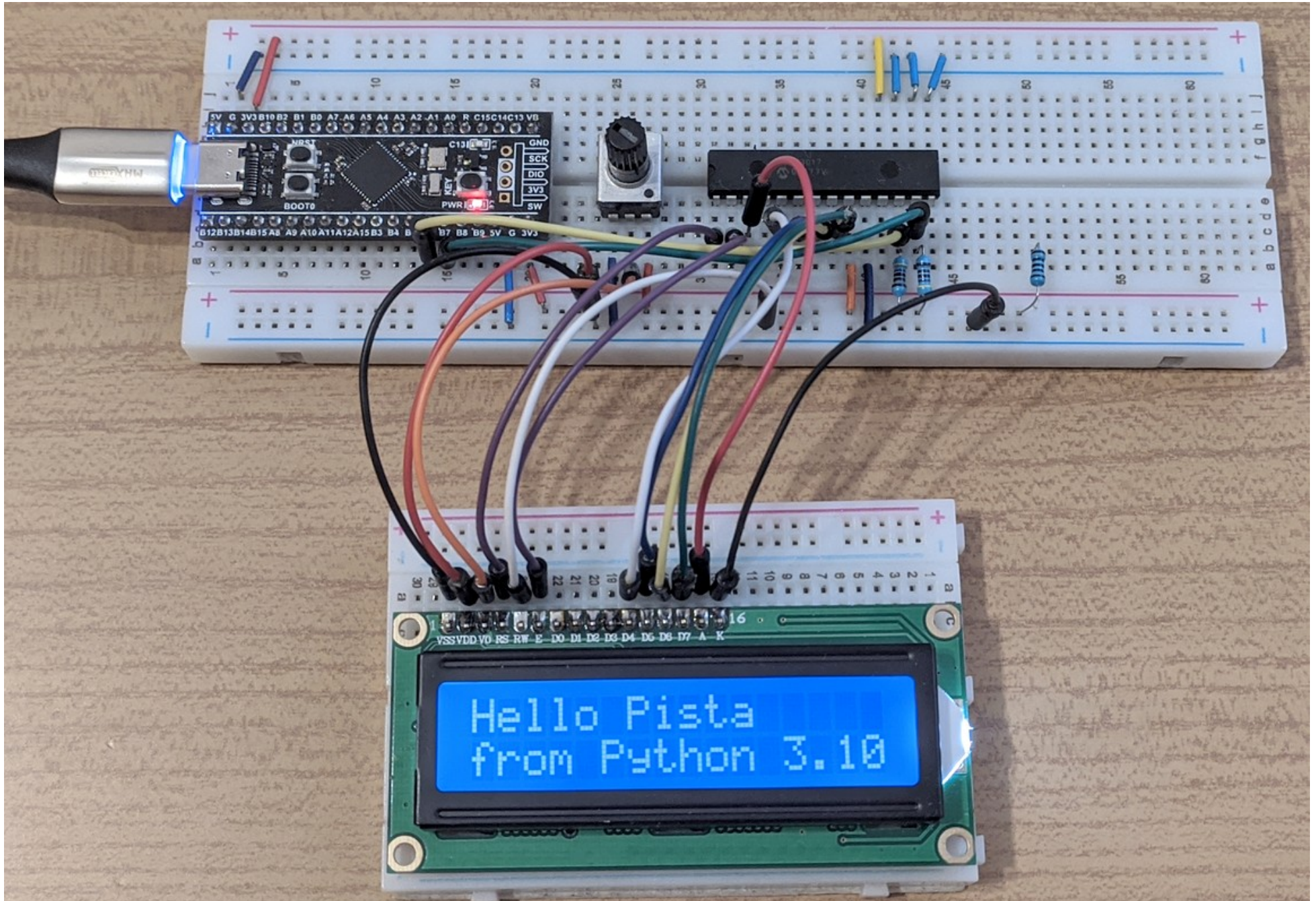
## Adatlapok és dokumentáció:

- MCP23017/MCP23S17: [adatlap és termékinfo](#)
- STM32F411CE [adatlap és termékinfo](#)
- STM32F411xC/E [Family Reference Manual](#)
- WeAct Studio: [STM32F4x1 MiniF4](#)





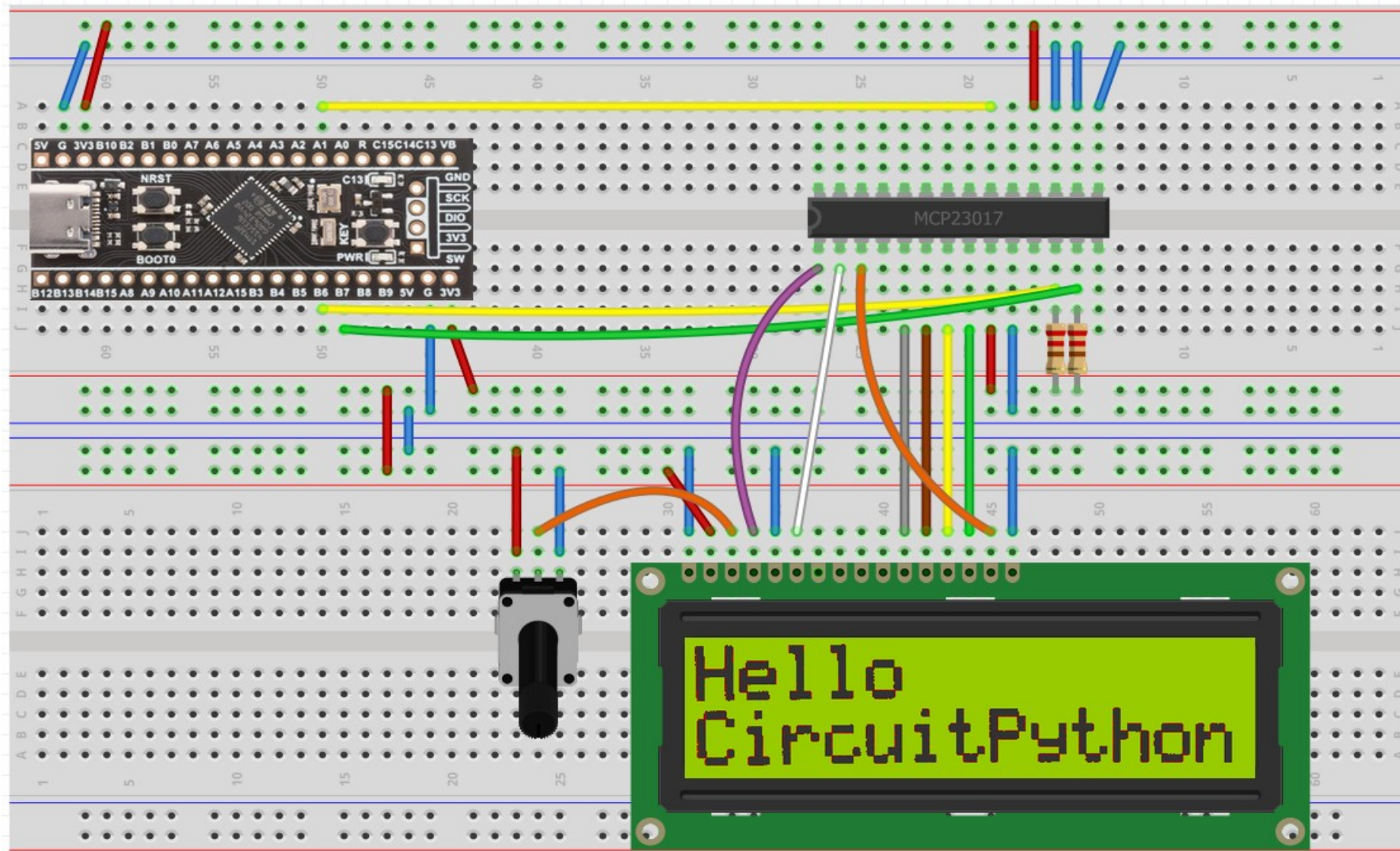
# 1. LCD kijelző vezérlése CircuitPythonban





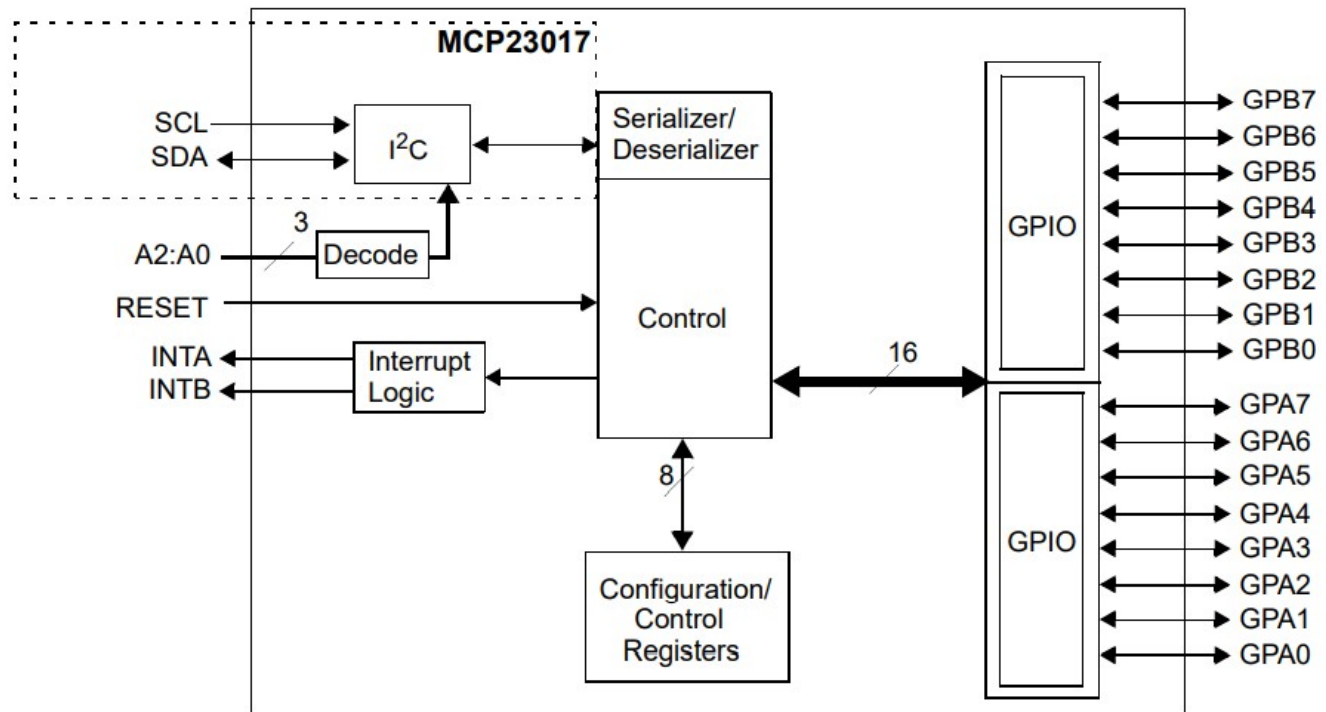
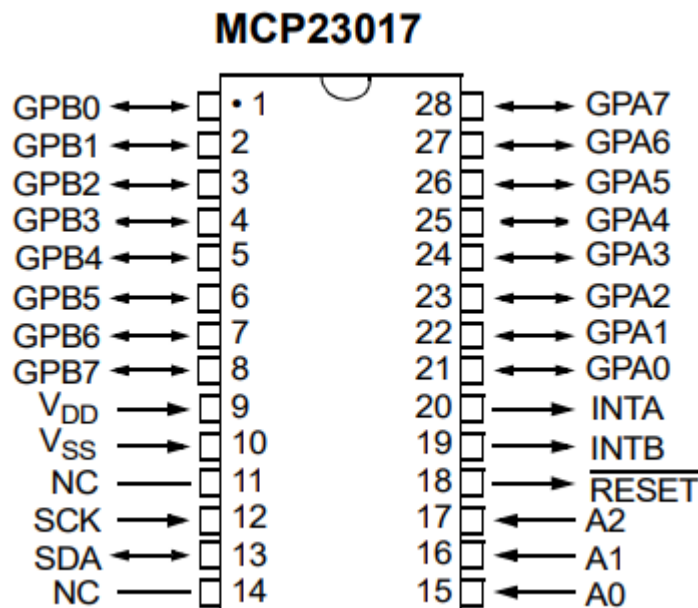
# Kapcsolási vázlat

- Lady Ada: Character LCDs leírását megtoldottuk azzal, hogy az LCD kijelzőt az MCP23017 I/O bővítőn keresztül vezéreljük. A hardver absztrakció miatt a kijelző kezelése transzparens



# Emlékeztető: Az MCP23017 periféria bővítő

- A Microchip gyártmányú **MC23 017** IC egy 16 bites kétirányú portbővítő, ami az **I2C** buszra csatlakozik (100, 400 és 1700 kHz)
- Három címvonallal nyolcféle cím állítható be (0x20-0x27)
- A 2x8 bites portok megszakításkérő jelet is adnak
- Konfigurálható a megszakításkérő jel forrása, a 8, vagy 16 bites mód, a bemenő jel polaritása és a belső felhúzás
- Tápfeszültség: 1.8–5.5 V
- Hardver RESET



# mcp23017\_lcd\_demo.py

```
import time
import board
import busio
from digitalio import *
import adafruit_character_lcd.character_lcd as characterlcd
from adafruit_mcp230xx.mcp23017 import MCP23017

i2c = busio.I2C(board.SCL, board.SDA)          # Initialize the I2C bus:
mcp = MCP23017(i2c, address=0x20, reset=True)  # MCP23017
# LCD is connected to the GPIOB port of the MCP23017 IC:
lcd_rs = mcp.get_pin(8)      # B0
lcd_en = mcp.get_pin(9)     # B1
lcd_d7 = mcp.get_pin(15)    # B7
lcd_d6 = mcp.get_pin(14)    # B6
lcd_d5 = mcp.get_pin(13)    # B5
lcd_d4 = mcp.get_pin(12)    # B4
lcd_backlight = mcp.get_pin(10) # B2

lcd = characterlcd.Character_LCD_Mono(
    lcd_rs, lcd_en, lcd_d4, lcd_d5, lcd_d6, lcd_d7, 16, 2, lcd_backlight
)

lcd.backlight = True          # Turn backlight on
lcd.clear()                  # Clear screen
lcd.message = "CircuitPython\nLCD display"
```

A bemutatott kapcsolás esetén így néz ki az I2C busz,  
a perifériabővítő és az LCD kijelző inicializálása

# lcd\_demo.py

- Ha elhagyjuk az **MCP23017** periféria-bővítőt, egyszerűbb lesz a kijelző inicializálása, bár így több MCU kivezetést használunk fel. A kijelzőt itt a **B4 – B9**, valamint az **A15** portkivezetésekre kötöttük

```
import board
import digitalio
import adafruit_character_lcd.character_lcd as characterlcd

lcd_rs = digitalio.DigitalInOut(board.B4)
lcd_en = digitalio.DigitalInOut(board.B5)
lcd_d7 = digitalio.DigitalInOut(board.B9)
lcd_d6 = digitalio.DigitalInOut(board.B8)
lcd_d5 = digitalio.DigitalInOut(board.B7)
lcd_d4 = digitalio.DigitalInOut(board.B6)
lcd_backlight = digitalio.DigitalInOut(board.A15)

lcd = characterlcd.Character_LCD_Mono(
    lcd_rs, lcd_en, lcd_d4, lcd_d5, lcd_d6, lcd_d7, 16, 2, lcd_backlight
)

lcd.backlight = True           # Turn backlight on
lcd.clear()                   # Clear screen
lcd.message = "CircuitPython\nLCD display"
```



# A PC kiegészítő LCD kijelző projekt

- Az előzőekben bemutatott LCD bekötéssel valósítjuk meg a PC kiegészítő kijelzőjét, ami az USB csatlakozón keresztül fogadja az adatokat (a PC felől virtuális soros portként kezelhető)
- **CircuitPython** oldalról az USB terminálkapcsolat **print** illetve **input** parancsokkal kezelhető, s a beépített **supervisor** modul segítségével ellenőrizhetjük, hogy érkezett-e beolvasandó adat, például ehhez hasonlóan:

```
import supervisor
# Main Loop
while True:
    if supervisor.runtime.serial_bytes_available:
        inText = input().strip() # Sorvége karakterek levágása
        if inText == "":
            continue
        print("received: ",text)
```

- A kiegészítő kijelző esetében természetesen majd az LCD kijelzőre fogjuk kiíratni a vett karaktereket



# mcp23017\_lcd\_display.py – 2/1.

```
import time
import board
import busio
import supervisor ←
from digitalio import *
from adafruit_mcp230xx.mcp23017 import MCP23017
import adafruit_character_lcd.character_lcd as characterlcd

# Initialize the I2C bus:
i2c = busio.I2C(board.SCL, board.SDA)
mcp = MCP23017(i2c, address=0x20, reset=True) # MCP23017

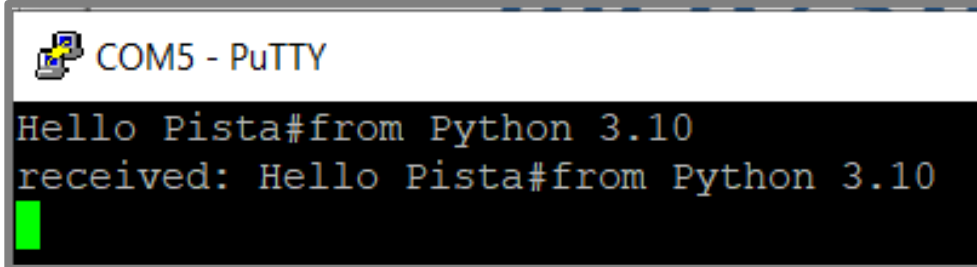
# LCD is connected to the GPIOB port of the MCP23017 IC:
lcd_rs = mcp.get_pin(8) # B0
lcd_en = mcp.get_pin(9) # B1
lcd_backlight = mcp.get_pin(10) # B2
lcd_d4 = mcp.get_pin(12) # B4
lcd_d5 = mcp.get_pin(13) # B5
lcd_d6 = mcp.get_pin(14) # B6
lcd_d7 = mcp.get_pin(15) # B7

lcd = characterlcd.Character_LCD_Mono(
    lcd_rs, lcd_en, lcd_d4, lcd_d5, lcd_d6, lcd_d7, 16, 2, lcd_backlight
)
```

A megjelölt sor kivételével az LCD inicializálása megegyezik a 6. oldalon bemutatottal

# mcp23017\_lcd\_display.py – 2/2.

```
text = "CircuitPython\nLCD display"
lcd.backlight = True # Turn backlight on
lcd.clear() # Clear screen
lcd.message = text
while True:
    if supervisor.runtime.serial_bytes_available: # If text is available
        text = input() # Read incoming bytes
        if text != "":
            lcd.clear()
            lcd.message = text.replace('#', '\n') # Write text (# line break)
```



- A # karaktert speciális jelentéssel ruháztuk fel: azt jelzi, hogy hol kell sort törni



# 2. Python webkliens alkalmazások

- Az LCD kijelző Python programokkal történő meghajtásához szükséges lépések:

- ❖ Soros port kezelése – **pyserial**
- ❖ NTP kliens – **socket** vagy **ntplib**
- ❖ HTTP kliens – **HTTP.client** vagy **requests**
- ❖ JSON adatok kezelése – **json**





# Python pyserial programkönyvtár

- A (virtuális) soros port kezeléséhez szükségünk lesz a **pyserial** programkönyvtárra, amelyet nekünk kell telepítenünk
  - Telepítése: `pip install pyserial`
  - Importálás: `import serial`
  - Inicializálás: `ser = serial.Serial(port='COM5', baudrate=9600)`
  - Szöveg kiküldése a kiegészítő LCD kijelzőre:  
`text = "Hello Pista#from Python 3.10\r"`  
`ser.write(text.encode())`
- Az `encode()` metódust azért kell meghívni, mert a **Python** által alapértelmezetten használt UTF-8 kódolású szöveget nem lehet közvetlenül a soros portra küldeni (hibajelzést kapnánk)
- Kimeneti buffer kiürítése: `ser.flush()`
  - Kimeneti csatorna lezárása: `ser.close()`

*A sebesség megadásának itt nincs jelentősége*



# write\_comport.py

- A soros porti adatküldést mutatja be az alábbi program, ami egy előre megadott szöveget küld ki az LCD kijelzőre

```
import serial
import time

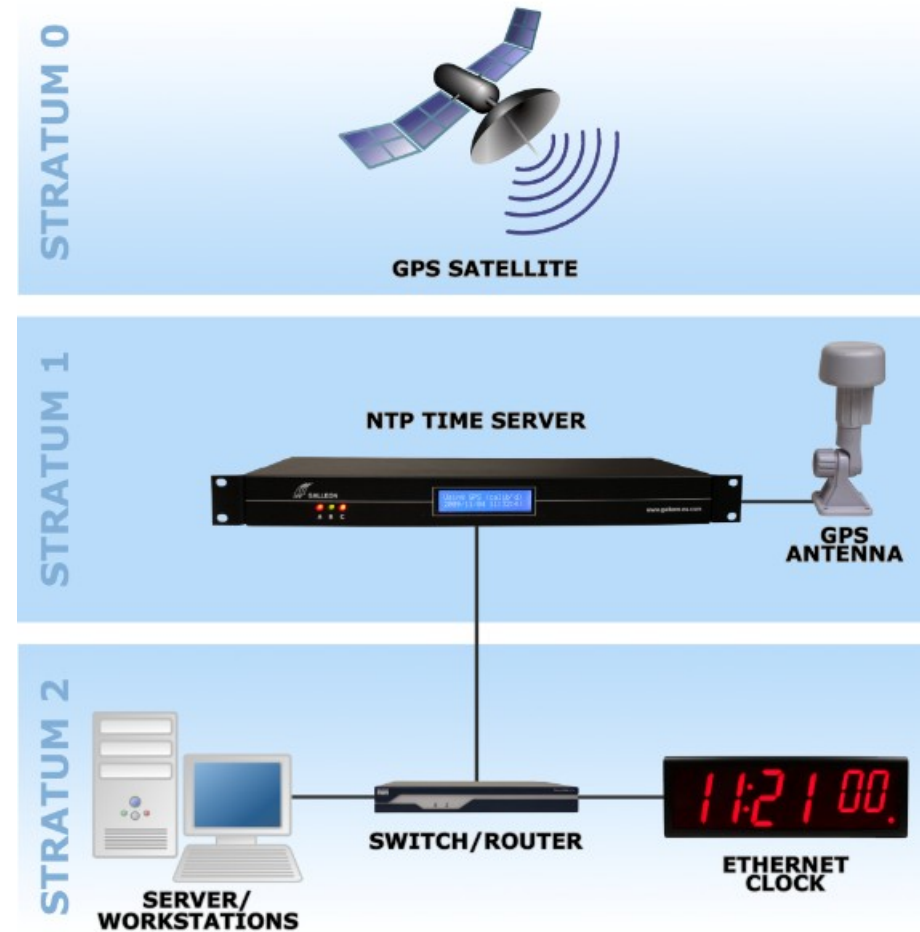
# configure the serial port
ser = serial.Serial(
    port='COM5',
    baudrate=9600,
    parity=serial.PARITY_NONE,
    stopbits=serial.STOPBITS_ONE,
    bytesize=serial.EIGHTBITS,
    timeout=1
)

# send the text to the serial port
text = "Hello Pista#from Python 3.10\r"
ser.write(text.encode())
ser.flush()
# wait for the text to be displayed on the LCD
time.sleep(1)
# close the serial port
ser.close()
```

A továbbiakban majd arra fogunk koncentrálni, hogy valami értelmesebb információ legyen a kiírandó szöveg

# Pontos idő lekérdezése NTP kéréssel

- A világot behálózó **NTP** (Network Time Protocol) szerverek ún. UDP csomagokkal kommunikálnak
- **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 eredetileg az **RFC958**, ma (NTP v4) az **RFC5905** írja le
- A legegyszerűbb **NTP** kérelem: 0x1B és 47 db nulla



Kép forrása: [www.galsys.co.uk/news/](http://www.galsys.co.uk/news/)



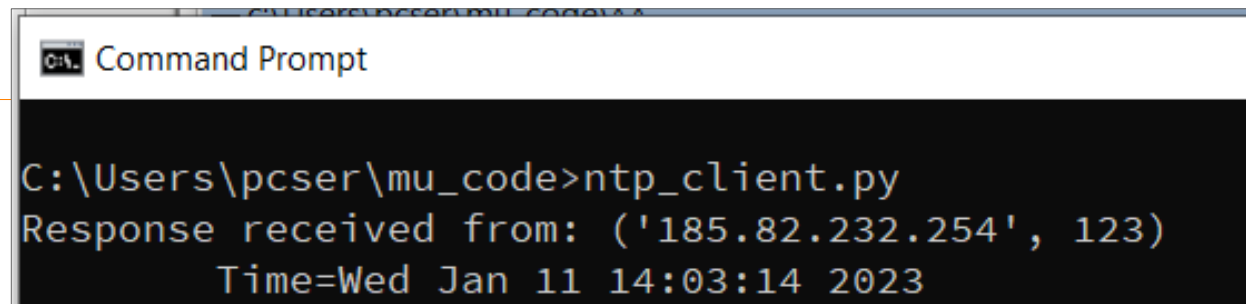
# ntp\_client.py

- Az alábbi módszer nem a legkényelmesebb út, de működőképes

```
import socket
import struct
import time
NTP_SERVER = "hu.pool.ntp.org"
TIME1970 = 2208988800 # Thanks to F.Lundh

def sntp_client():
    client = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
    data = '\x1b' + 47 * '\0'
    client.sendto(data.encode(), (NTP_SERVER, 123))
    data, address = client.recvfrom(1024)
    if data:
        print('Response received from:', address)
    t = struct.unpack('!12I', data)[10]
    t -= TIME1970
    print('\tTime=%s' % time.ctime(t))

if __name__ == '__main__':
    sntp_client()
```



```
Command Prompt
C:\Users\pcser\mu_code>ntp_client.py
Response received from: ('185.82.232.254', 123)
Time=Wed Jan 11 14:03:14 2023
```

# Python ntplib programkönyvtár

---

- Az NTP szerverek lekérdezése kényelmesebben megoldható az **ntplib** programkönyvtárral, amelyet nekünk kell telepítenünk
- Telepítése: `pip install ntplib`  
(A pip pampogni fog, hogy csak az elavult **setup.py** telepítéssel tudja felrakni, de ne foglalkozzunk vele!)
- Importálás: 

```
import ntplib
import time
```
- Inicializálás: `client = ntplib.NTPClient()`
- Pontos idő lekérése: `response = client.request('pool.ntp.org')`
- Az időadat konvertálása:  

```
timestamp = response.tx_time
local_time = time.localtime(timestamp)
text = time.strftime("%Y-%m-%d %H:%M:%S", local_time)
```
- Bővebben lásd [a time programkönyvtár leírásában!](#)

# strftime(): formátum megadása – 2/1.

Directive	Meaning	Notes
%a	Locale's abbreviated weekday name.	
%A	Locale's full weekday name.	
%b	Locale's abbreviated month name.	
%B	Locale's full month name.	
%c	Locale's appropriate date and time representation.	
%d	Day of the month as a decimal number [01,31].	
%H	Hour (24-hour clock) as a decimal number [00,23].	
%I	Hour (12-hour clock) as a decimal number [01,12].	
%j	Day of the year as a decimal number [001,366].	
%m	Month as a decimal number [01,12].	
%M	Minute as a decimal number [00,59].	
%p	Locale's equivalent of either AM or PM.	(1)
%S	Second as a decimal number [00,61].	(2)



# strftime(): formátum megadása – 2/2.

Directive	Meaning	Notes
%U	Week number of the year (Sunday as the first day of the week) as a decimal number [00,53]. All days in a new year preceding the first Sunday are considered to be in week 0.	(3)
%w	Weekday as a decimal number [0(Sunday),6].	
%W	Week number of the year (Monday as the first day of the week) as a decimal number [00,53]. All days in a new year preceding the first Monday are considered to be in week 0.	(3)
%x	Locale's appropriate date representation.	
%X	Locale's appropriate time representation.	
%y	Year without century as a decimal number [00,99].	
%Y	Year with century as a decimal number.	
%z	Time zone offset indicating a positive or negative time difference from UTC/GMT of the form +HHMM or -HHMM, where H represents decimal hour digits and M represents decimal minute digits [-23:59, +23:59]. [1]	
%Z	Time zone name (no characters if no time zone exists). Deprecated. [1]	
%%	A literal '%' character.	

# ntp\_localtime.py

```
import ntplib
import time
import serial
```

Az ntplib könyvtár felhasználásával kérdezzük le a pontos időt és írassuk ki az LCD kijelzőn!

```
# configure the serial port
```

```
ser = serial.Serial(port='COM5',baudrate=9600)
```

```
# Connect to the NTP server and get the current time
```

```
client = ntplib.NTPClient()
```

```
while True:
```

```
    response = client.request('pool.ntp.org')
```

```
    # Convert the time to a readable format
```

```
    timestamp = response.tx_time
```

```
    local_time = time.localtime(timestamp)
```

```
    # Use the strftime function to format the time and date
```

```
    formatted_time = time.strftime('%Y. %b. %d. #a %H:%M:%S\r\n', local_time)
```

```
    print(formatted_time)
```

```
    ser.write(formatted_time.encode())
```

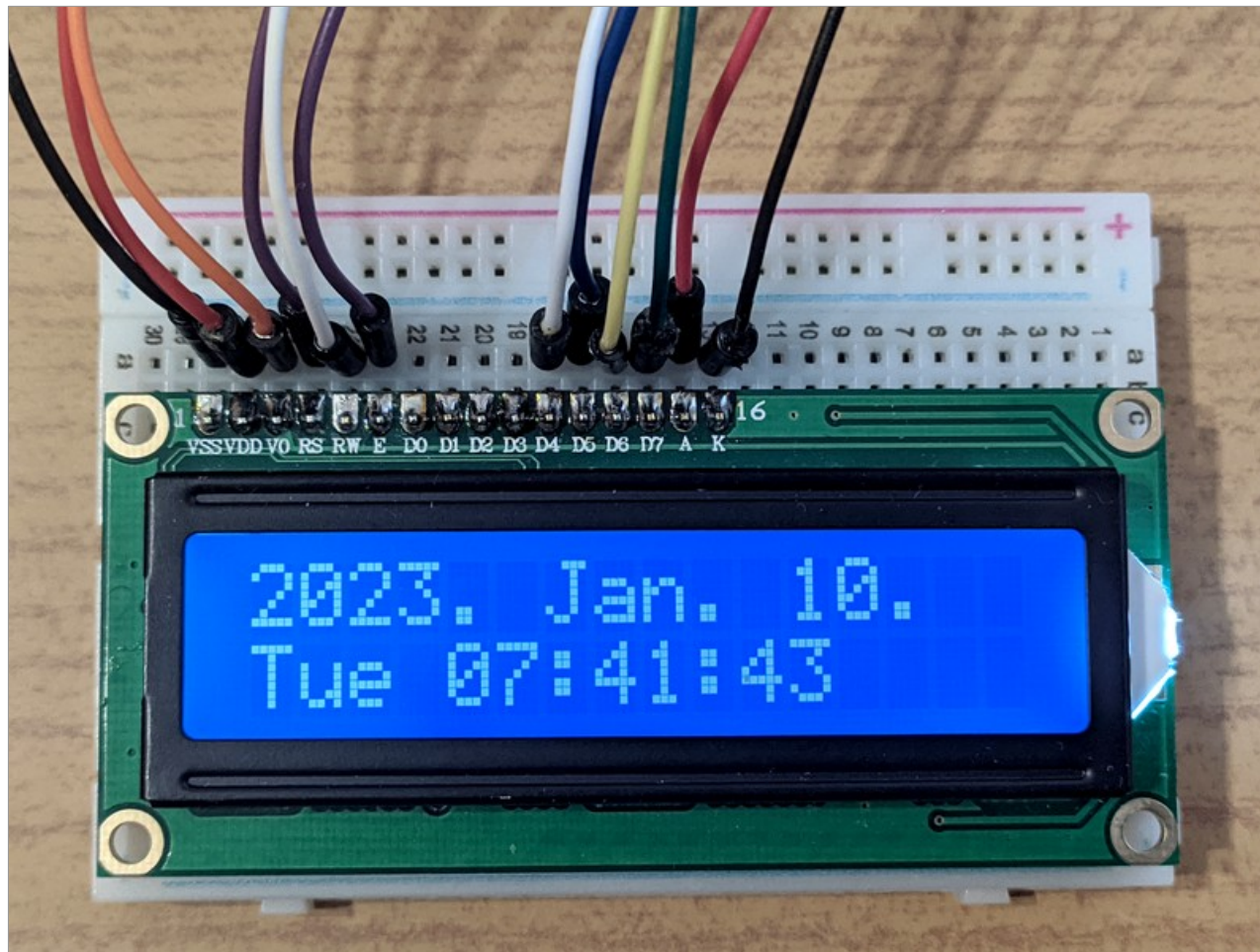
```
    ser.flush()
```

```
    # wait for the text to be displayed on the LCD
```

```
    time.sleep(10)
```

# ntp\_localtime.py futási eredménye

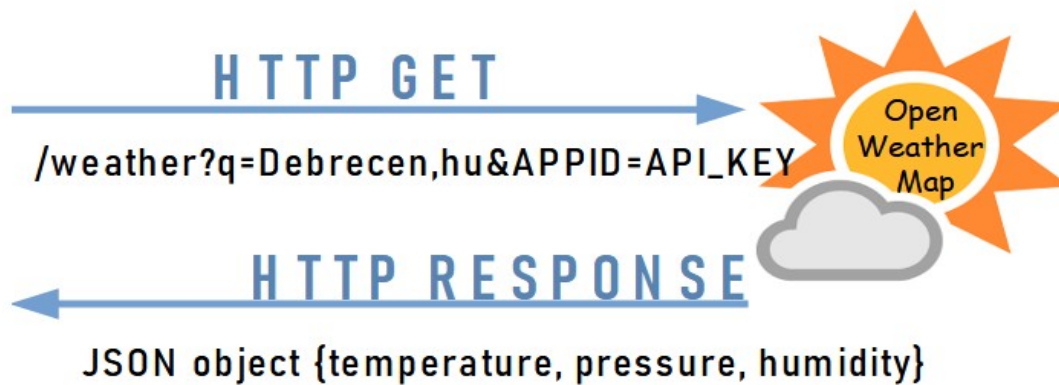
- Az '%Y. %b. %d. # %a %H:%M:%S' formátum hatására az ábrán látható módon lesz kiírva a dátum és az idő
- A # karakter most is a sortörés helyét jelzi...



# Időjárási adatok lekérése (OpenWeatherMap.org)

- Az aktuális időjárási adatokat és előrejelzéseket az [openweathermap.org](https://openweathermap.org) webszerverről tölthetjük le, HTTP kliensként
- Regisztráció (Sign in/Create Account) után az ingyenes szolgáltatásokat vehetjük igénybe (lásd Pricing, Free oszlop)
- Első bejelentkezéskor szerezzük meg és jegyezzük fel az API kulcsot!
- Mi most csak az aktuális adatok lekérdezésével foglalkozunk, például:  
`http://api.openweathermap.org/data/2.5/weather?q=Debrecen&appid=*****`
- A választ alapértelmezetten JSON formátumban kapjuk meg, pl.:  

```
{"coord":{"lon":21.6333,"lat":47.5333},"weather":[{"id":802,"main":"Clouds","description":"scattered clouds","icon":"03d"}],"base":"stations","main":{"temp":283.15,"feels_like":279.93,"temp_min":283.15,"temp_max":283.15,"pressure":1025,"humidity":46} . . .
```





# „Úr ír”, avagy az OpenAI chatGPT programot ír...

- **Q:** Write a Python program for downloading weather data for Debrecen from OpenWeatherMap
- **A:** You'll need to replace YOUR\_API\_KEY\_HERE with your own API key, which you can get by signing up for a free account at [https://home.openweathermap.org/users/sign\\_up](https://home.openweathermap.org/users/sign_up)
- Make sure that you have the **requests** library installed. You can install it by running `pip install requests` in your command line.
- This code will send a GET request to the OpenWeatherMap API, asking for the current weather data for Debrecen in Hungary, and parse the JSON data returned in the response. The parsed data will be printed on the console.

```
import requests

api_key = "YOUR_API_KEY_HERE"
city = "Debrecen"
country_code = "hu"

# API endpoint for current weather data
endpoint = f"http://api.openweathermap.org/data/2.5/weather?q={city},
{country_code}&appid={api_key}"

response = requests.get(endpoint) # Send GET request to the endpoint
data = response.json()          # Parse the JSON data from the response
print(data)                     # Print the weather data
```

**openweathermap\_with\_requests.py**

```
# API key for OpenWeatherMap
# City and country code for Debrecen
```

# A requests modul használata

- Python **requests** modul a webes kérés elküldésére és válaszfeldolgozására  
Támogatja az HTTP szabványokat, mint pl. a GET és POST kérés elküldését  
Telepítése: `pip install requests`
- A **GET kérés** elküldéséhez a **requests.get()** függvényt kell használni, amelynek át kell adni az URL-t, opcionálisan hozzáadva paramétereket, fejléceket.  
Eredményül egy **Response** objektumot kapunk, amely a szerver által visszaadott választ tartalmazza. Például:

```
import requests

response = requests.get("https://example.com/api/data")
print(response.status_code) # 200
print(response.text)       # szerver által visszaadott szöveg
```

- A **POST kérés** elküldéséhez a **requests.post()** függvényt kell használni.  
A kérés adatait a **data** paraméterben lehet átadni. Például:

```
import json, requests

data = {"name": "John", "age": 30}
headers = {"Content-type": "application/json"}
response = requests.post("https://example.com/api/submit",
data=json.dumps(data), headers=headers)
print(response.status_code) # 201
```

# Egy másik lehetőség: http.client

- Természetesen használhatjuk a beépített **http.client** könyvtárat is, itt a lekérés két lépésben történik (conn, conn.request)

```
import http.client
```

**openweathermap\_http\_client.py**

```
endpoint = 'api.openweathermap.org'
```

```
api_key = "YOUR_API_KEY_HERE"
```

```
city = "Debrecen" # Define the city
```

```
conn = http.client.HTTPSConnection(endpoint) # Create an HTTP connection
```

```
# Make a GET request to the OpenWeatherMap API
```

```
conn.request("GET", f"/data/2.5/weather?q={city}&appid={api_key}")
```

```
res = conn.getresponse() # Get the response from the server
```

```
data = res.read() # Read and print the response
```

```
print(data.decode()) # Get JSON data, output as text
```

```
conn.close() # Close the connection
```

```
C:\Users\pcser\mu_code>openweathermap_http_client.py
```

```
{"coord":{"lon":21.6333,"lat":47.5333},"weather":[{"id":804,"main":"Clouds",  
"description":"overcast clouds","icon":"04d"}],"base":"stations","main":  
{"temp":279.02,"feels_like":277.02,"temp_min":279.02,"temp_max":279.02,"pressure":1022,  
"humidity":93},"visibility":7000,"wind":{"speed":2.57,"deg":210},"clouds":{"all":100},  
"dt":1673518353,"sys":{"type":1,"id":6665,"country":"HU","sunrise":1673504318,  
"sunset":1673535847},"timezone":3600,"id":721472,"name":"Debrecen","cod":200}
```

# JSON adatok kezelése

- A **JSON** (JavaScript Object Notation) egy szöveges adatformátum, amelyet gyakran használnak az adatok cseréjére és tárolására webes alkalmazásokban. A Pythonban a **json** modul segítségével lehet kezelni a **JSON** adatokat.
- A **json.loads()** függvénnyel lehet a **JSON** formátumú szöveget Python objektummá alakítani (például listává vagy dict-é), a **json.dumps()** függvénnyel pedig Python objektumot lehet **JSON** formátumú szöveggé alakítani. Például:

```
import json
# JSON szöveg -> Python objektum
json_data = '{"name": "John", "age": 30}'
python_obj = json.loads(json_data)
print(python_obj) # {"name": "John", "age": 30}
print(python_obj['name'],python_obj['age']) # John 30

# Python objektum -> JSON szöveg
python_obj = {"name": "John", "age": 30}
json_data = json.dumps(python_obj)
print(json_data) # '{"name": "John", "age": 30}'
```



# openweathermap\_with\_display.py – 2/1.

- Az előzőekben bemutatott ismeretek birtokában könnyen összeállíthatunk egy Python alkalmazást, ami lekéri az időjárási adatokat az **OpenWeatherMap** szerverről és virtuális soros porton alkalmas formátumban kiküldi a kiegészítő LCD képernyőre

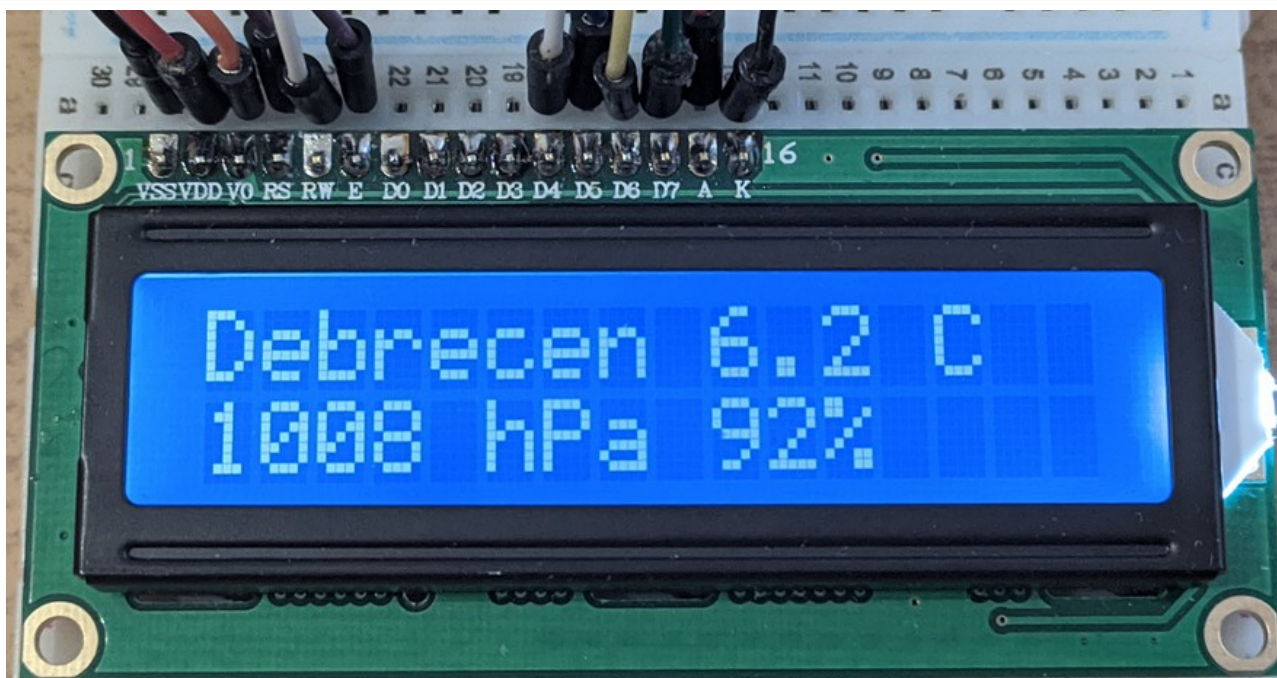
```
import requests                # HTTP kérések kezelése
import json                    # Json formátumú adatok kezelése
import io                      # Memóriában tárolt fájl kezelésére
import serial                  # Soros port kezelése
import time                    # Idő adatok és formátum konverzió kezelése

# A soros port konfigurálása
ser = serial.Serial(port='COM5', baudrate=9600)
# Az URL http vagy https elérésű is lehet...
API_ENDPOINT = "https://api.openweathermap.org/data/2.5/weather"

params = {
    "q": "Debrecen, Hungary",
    "lat": "47.53",
    "lon": "21.62",
    "units": "metric",
    "appid": "YOUR_API_KEY_HERE"
}                                     # Metrikus egységeknél °C skála van érvényben
```

# openweathermap\_with\_display.py – 2/2.

```
response = requests.get(API_ENDPOINT, params=params) # A HTTP kérés
dataJSON = json.loads(response.text) # Python objektummá alakítjuk
# print(json.dumps(dataJSON,indent=4)) # Csak ellenőrzéshez...
temp = float(dataJSON['main']['temp']) # Az adatok címezése így egyszerű
humidity = int(dataJSON['main']['humidity'])
pressure = int(dataJSON['main']['pressure'])
s=io.StringIO("") # A kiíratást ebbe irányítjuk át
print("Debrecen {0:.1f} C#{1} hPa {2}%\r\n".format(temp, pressure, humidity),
file=s)
print(s.getvalue()) # Ellenőrző kiíratás
ser.write(s.getvalue().encode()) # Kiküldjük az adatokat az LCD-re
s.close()
```

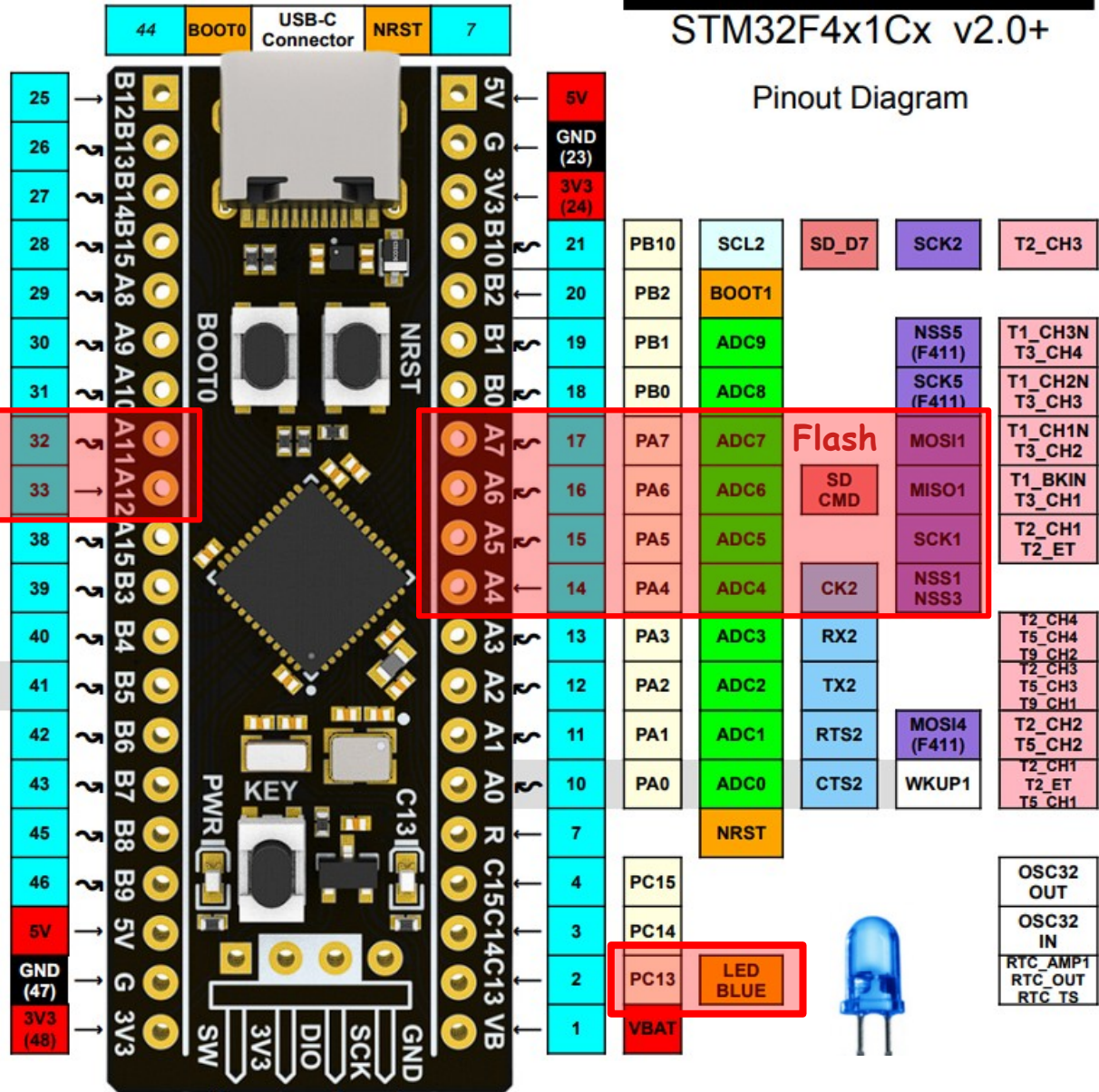




### Pinout Diagram

#### Legend

POWER
GROUND
CPU PIN
PIN NAME
CONTROL
ANALOG
TIMER & CHANNEL
USART
SPI / I2S
SDIO (F411 Only)
I2C
CAN BUS
USB
MISC
BOARD HARDWARE
← 5V → Tolerant
← 3.3V → (F411)
~ PWM ~ Pin



Notes:  
 TIM6 & 7 are only used by DAC and don't have any pins  
 All pins are 5V tolerant on F401  
 Pins 10 and 41 on F411 are 3.3V only.

Updated: 2020-03-16  
 Richard.Balint