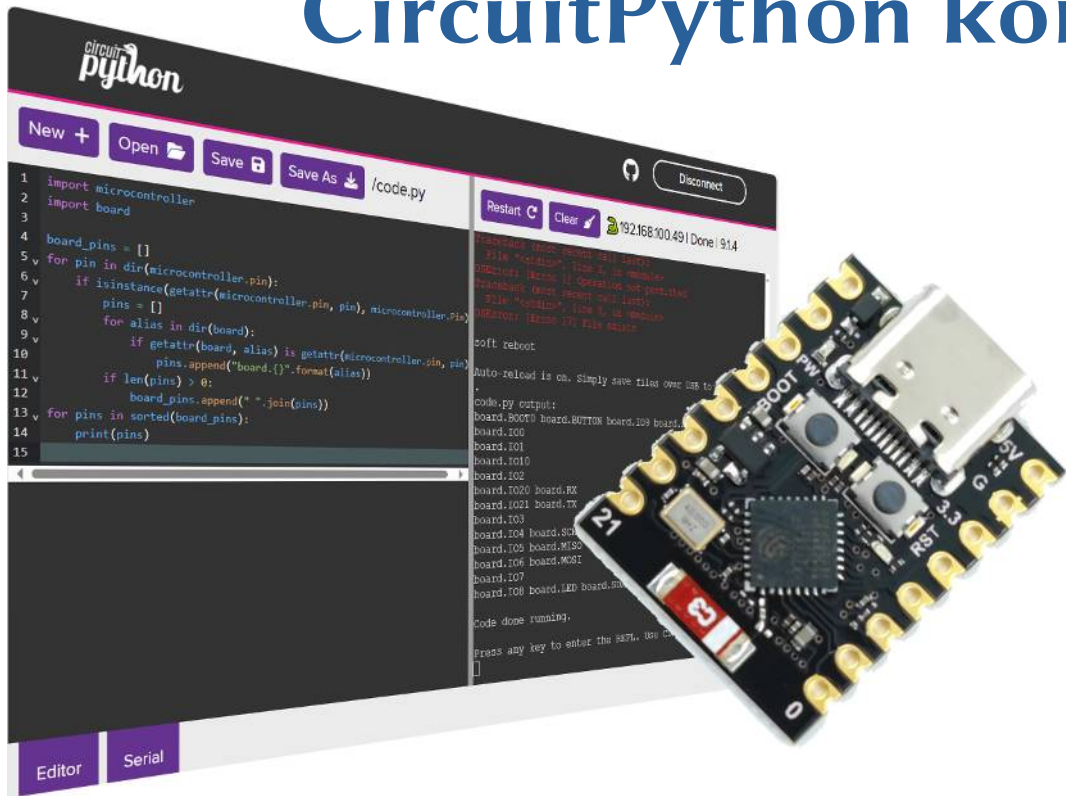


ESP32-C3 mikrovezérlők programozása CircuitPython környezetben



1. CircuitPython és az ESP32-C3 – a kezdő lépések

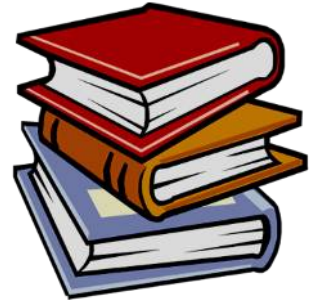
Felhasznált és ajánlott irodalom

❖ Python:

- Mark Pilgrim/Kelemen Gábor: [Ugorj fejest a Python 3-ba!](#)

❖ 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](https://github.com/adafruit/Adafruit_CircuitPython_Bundle)



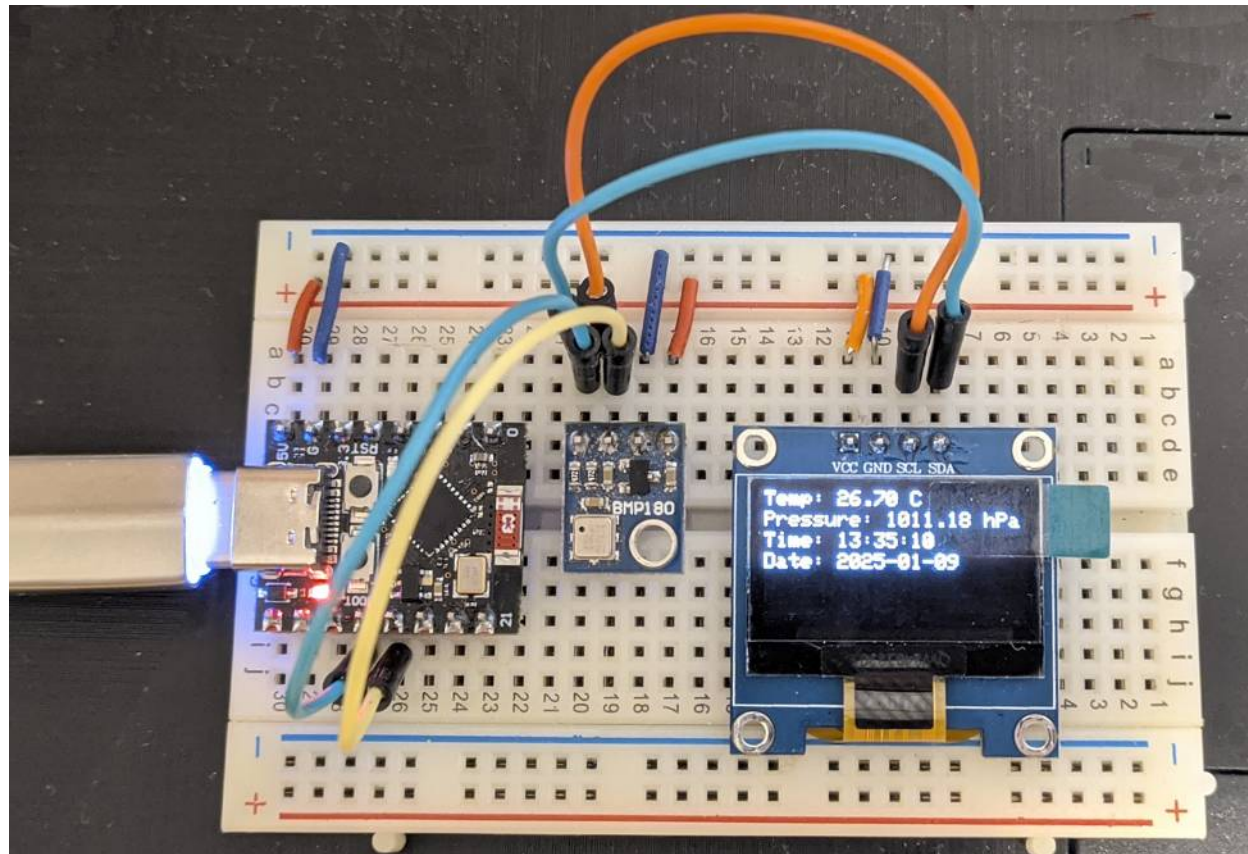
❖ Online eszközök és támogatás:

- Learn Adafruit: [CircuitPython on ESP32 Quick Start](#)
- Adafruit: [Adafruit Web Serial ESPTool](#)
- Adafruit: [CircuitPython Code Editor](#)



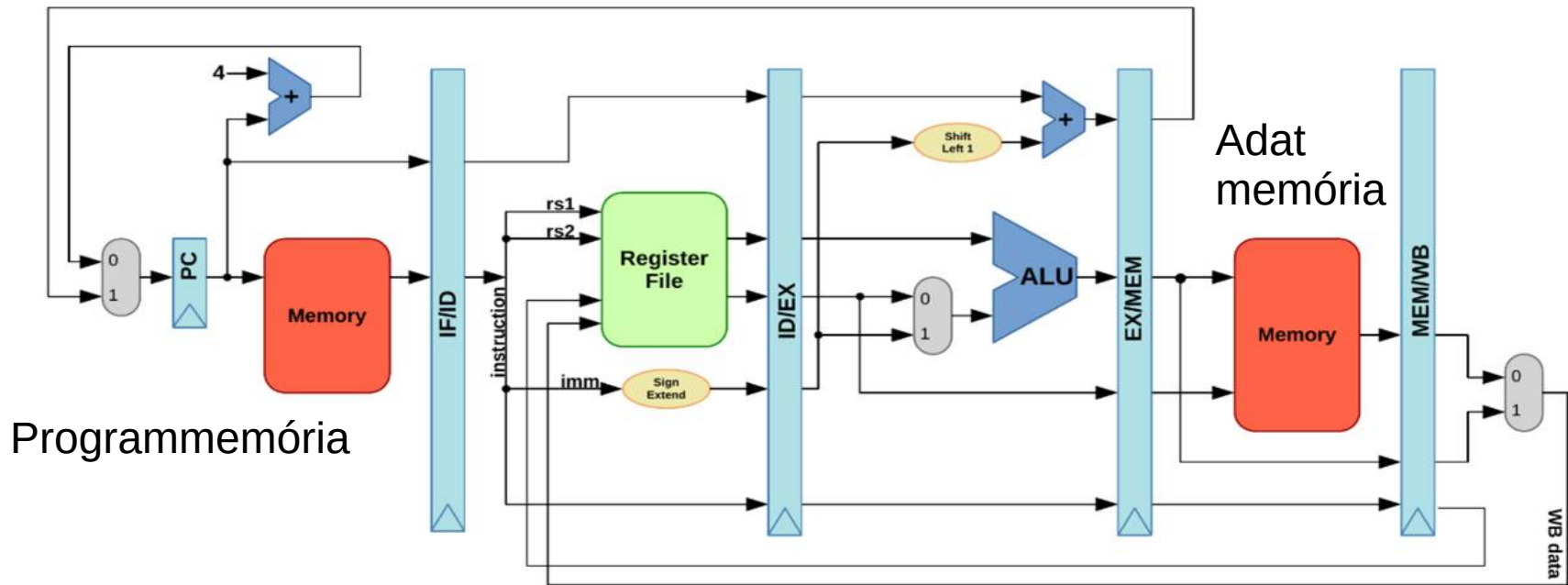
Miért ezekre esett a választás?

- ❖ Az **ESP32-C3** egy olcsó, de nagytudású mikrovezérlő (sok memóriával, WiFi és Bluetooth képességekkel)
- ❖ A **CircuitPython** pedig egy magasszintű, könnyen elsajátítható programnyelv, fejlett támogatással, sokféle célra és eszközhöz



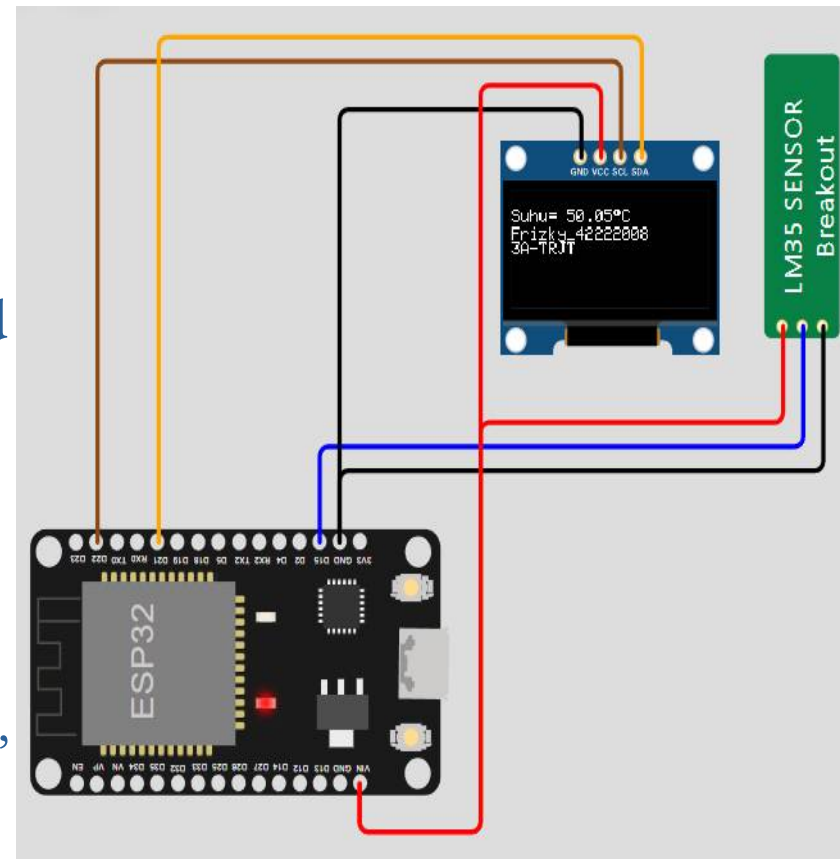
A programozható eszközök működése

- ❖ A CPU (Central Processing Unit) főbb részei a PC programszámláló, a programmemória, az utasításdekóder, az aritmetikai logikai egység (ALU), a regiszterkészlet, az adatmemória



Fizikai programozás

- ❖ A fizikai programozás (vagy fizikai számítástechnika) olyan interaktív rendszerek létrehozását jelenti hardverek és szoftverek segítségével, melyek képesek érzékelni a világban létrejövő jeleket és reagálni is tudnak rá
- ❖ A mi esetünkben ez saját készítésű csináld-magad projektek vagy alkotások megnevezésére szolgál, melyek mikrovezérlők (vezérlési feladatokra optimalizált extra kicsi számítógépek) be- és kimeneteire kötött különféle digitális és analóg érzékelők segítségével vezérelnek elektromechanikus eszközöket (jelfogó, motorok), világító vagy egyéb eszközöket, kijelzőket, de akár szoftvereket is



Mi az a CircuitPython?

- ❖ **CircuitPython:** programozási nyelv, arra tervezve, hogy segítse a tanulás és a kísérletezés folyamatát a mikrovezérlő kártyákkal
- ❖ Nem szükséges bonyolult keresztfordító rendszereket telepíteni a számítógépünkre. Amint csatlakoztattuk a kártyánkat, csak egy szövegszerkesztőre van szükség. Ilyen egyszerű...
- ❖ A **CircuitPython** a **Python** nyelven alapul. A korlátozott hardver erőforrások nyilvánvalóan korlátozott lehetőségeket biztosítanak, ugyanakkor a **CircuitPython** többletet is nyújt: hardvertámogatást a mikrovezérlő kártya perifériáinak elérésére és kezelésére, mellyel ún. „*fizikai programozást*” valósíthatunk meg.
- ❖ **Fizikai programozás:** olyan interaktív rendszerek létrehozását jelenti hardverek és szoftverek segítségével, melyek képesek érzékelni a világban létrejövő jeleket és reagálni is tudnak rá, azaz a programjaink hatására nemcsak a képernyőn történik valami, hanem a fizikai valóságban is...

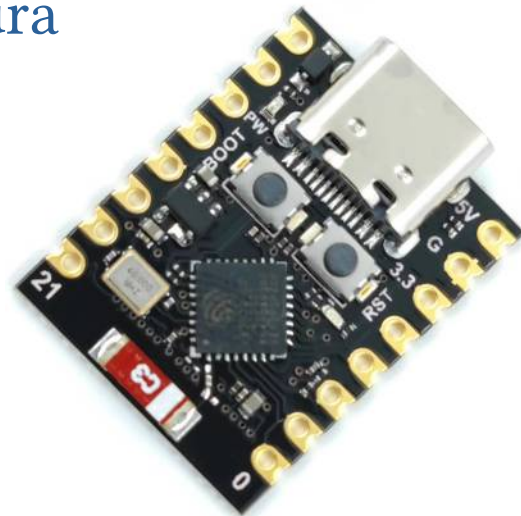
A támogatott kártyák

- ❖ A támogatott kártyák száma e sorok írásakor: ~~571~~ (számuk napról napra növekszik!) **618**
- ❖ Csak példa gyanánt néhány ismertebb típus:
Raspberry Pi Pico, Circuit Playgroung Express, Feather M4 express, **ESP32 Doit Devkit**, Nano RP2040 Connect, Metro ESP32-S2, nRF52 840 Dongle, Feather STM32F405 Express, **STM32F411CE blackpill**, Arduino Nano 33 IOT, STM32H743 Nucleo, BBC micro:bit v2, Pyboard, Raspberry Pi 4 model B és még sokan mások...
- ❖ A CircuitPython aktuális változatának letöltése innen: <https://circuitpython.org/downloads> (a kártya kiválasztása és rákattintás után a felbukkanó lapon a jobb felső sarokban elérhető a letöltési link)
- ❖ Mi a **Maker Go ESP32-C3 Super mini** kártyát fogjuk használni, a **CircuitPython** legfrissebb kiadásával (lásd: circuitpython.org/board/makergo_esp32c3_supermini/)

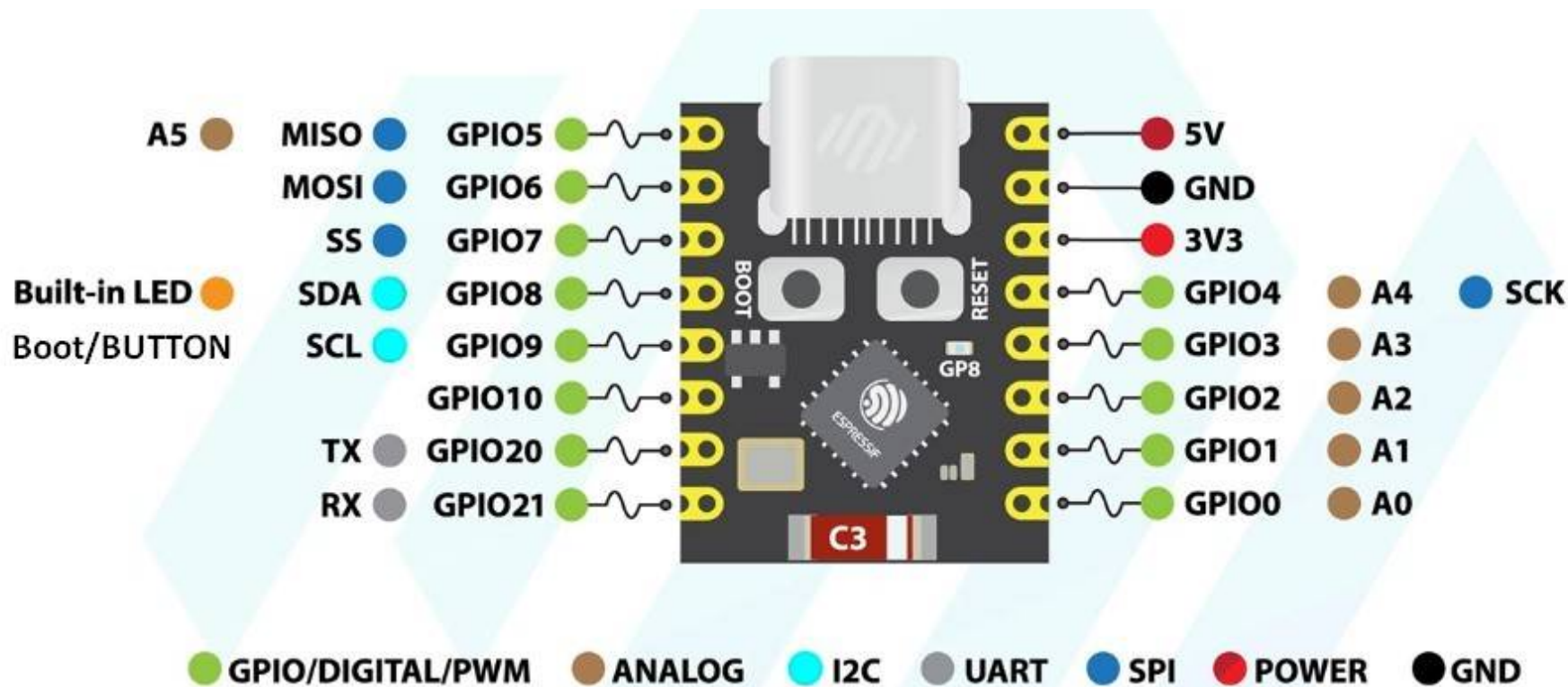


Az ESP32-C3 Super Mini kártya

- ❖ A **Maker Go ESP32-C3 Super Mini** egy kompakt fejlesztői kártya
- ❖ Mikrovezérlő: **ESP32-C3**, 32 bites **RISC-V** architektúra
- ❖ Órajel: Akár 160 MHz
- ❖ Memória: 400 KB SRAM, 384 KB ROM
- ❖ Tároló: 4 MB Flash memória
- ❖ Wi-Fi: 802.11 b/g/n (2.4 GHz)
- ❖ Bluetooth: Bluetooth 5.0 LE
- ❖ GPIO: Több mint 20 általános célú bemenet/kimenet (GPIO), melyek közül 13 van kivezetve
- ❖ Interfészek: SPI, I2C, UART, ADC, PWM



Az ESP32 C3 Super Mini kártya kivezetései



ESP32 C3 Super Mini

Megjegyzés: Az A5 analóg bemenet (ADC2) nem használható, ha a WiFi használatban van!

A CircuitPython firmware telepítése

- ❖ Mivel a kártyánk gyárilag nem tartalmazza a **CircuitPython** firmware-t, így bele kell töltenünk (csak egyszer kell végrehajtani) **10.0.3**
- ❖ Az **ESP32 Doit Devkit** kártyához a CircuitPython ~~9.2.1~~ változatát innen töltöttük le: circuitpython.org/board/makergo_esp32c3_supermini/
- ❖ A felprogramozást a mikrovezérlő gyárilag beégetett bootloadere segítségével végezhető el, ehhez vagy a fenti honlapon az **Open Installer** eszközt, vagy a **Web Serial ESPTool**, vagy a Pythonban írt parancssori **ESPTool** letöltő programot használhatjuk
 - Az **Open Installer** és a **Web Serial ESPTool** használata egyszerűbb, de Chrome alapú (*Chrome, Edge, Opera*) böngésző kell hozzá, amiben a soros portok elérése engedélyezhető (a <chrome://flags> URL megnyitása után az **Experimental Web Platform features** opció legyen **Enabled**-re állítva)
 - Az **ESPTool.py** letöltött **Python** környezetbe telepíteni kell a `pip install esptool` paranccsal

Az Open Installer használata

❖ URL: circuitpython.org/board/makergo_esp32c3_supermini/

CircuitPython 10.0.3

This is the latest **stable** release of CircuitPython that will work with the Maker Go ESP32C3 Supermini. **Use this release** if you are new to CircuitPython.

[Release Notes for 10.0.3](#)

ENGLISH (US)

DOWNLOAD .BIN NOW

OPEN INSTALLER

CircuitPython Installer for Maker Go ESP32C3 Supermini

[Full CircuitPython 10.0.3 Install](#)
[Install CircuitPython 10.0.3 Bin Only](#)
[Update WiFi credentials](#)

Close

Connect to Your Board

1. Plug your board into this computer. *Make sure the USB cable is good for data sync, and is not a charge-only cable.*
2. **Put your board into ROM bootloader mode**, by holding down the BOOT button (sometimes marked "B0"), and clicking the RESET button (sometimes marked "RST"). If your board doesn't have a BOOT button, just press RESET.
3. **Connect** Click this button to open the Web Serial connection menu and choose the serial port for this board.

There may be many devices listed, such as your remembered Bluetooth peripherals, anything else plugged into USB, etc. If you aren't sure which to choose, look for words like "USB", "UART", "JTAG", and "Bridge Controller". There may be more than one right option depending on your system configuration. Experiment if needed.

Previous

Next

circuitpython.org wants to connect to a serial port

JBL LIVE PRO+ TWS

Haylou GT3

USB JTAG/serial debug unit (COM9) - Paired

Connect

Cancel

Az Open Installer használata

Erase Flash

Itt törölhetjük a Flash memóriát
majd letöltődik a firmware

Now, optionally, erase everything on the Maker Go ESP32C3 Supermini.

Previous

Skip Erase

Continue

Reconnect to serial

Programletöltés után újraindítás és
újrapcsolódás után folytathatjuk

- **Connected** Click this button to open the Web Serial connection menu. If it is already connected, you can press it again if you need to select a different port.

There may be several devices listed. If you aren't sure which to choose, look for one that includes the name of your microcontroller.

Previous

Next

Fill in settings.toml

This step will write your network credentials to the settings.toml file on CIRCUITPY. Make sure your board is running CircuitPython.

If you want to skip this step and fill in settings.toml later, just close this dialog.

WiFi Network Name (SSID):

WiFi SSID

WiFi Password:

WiFi Password

Web Workflow API Password:

.....

Web Workflow API Port:

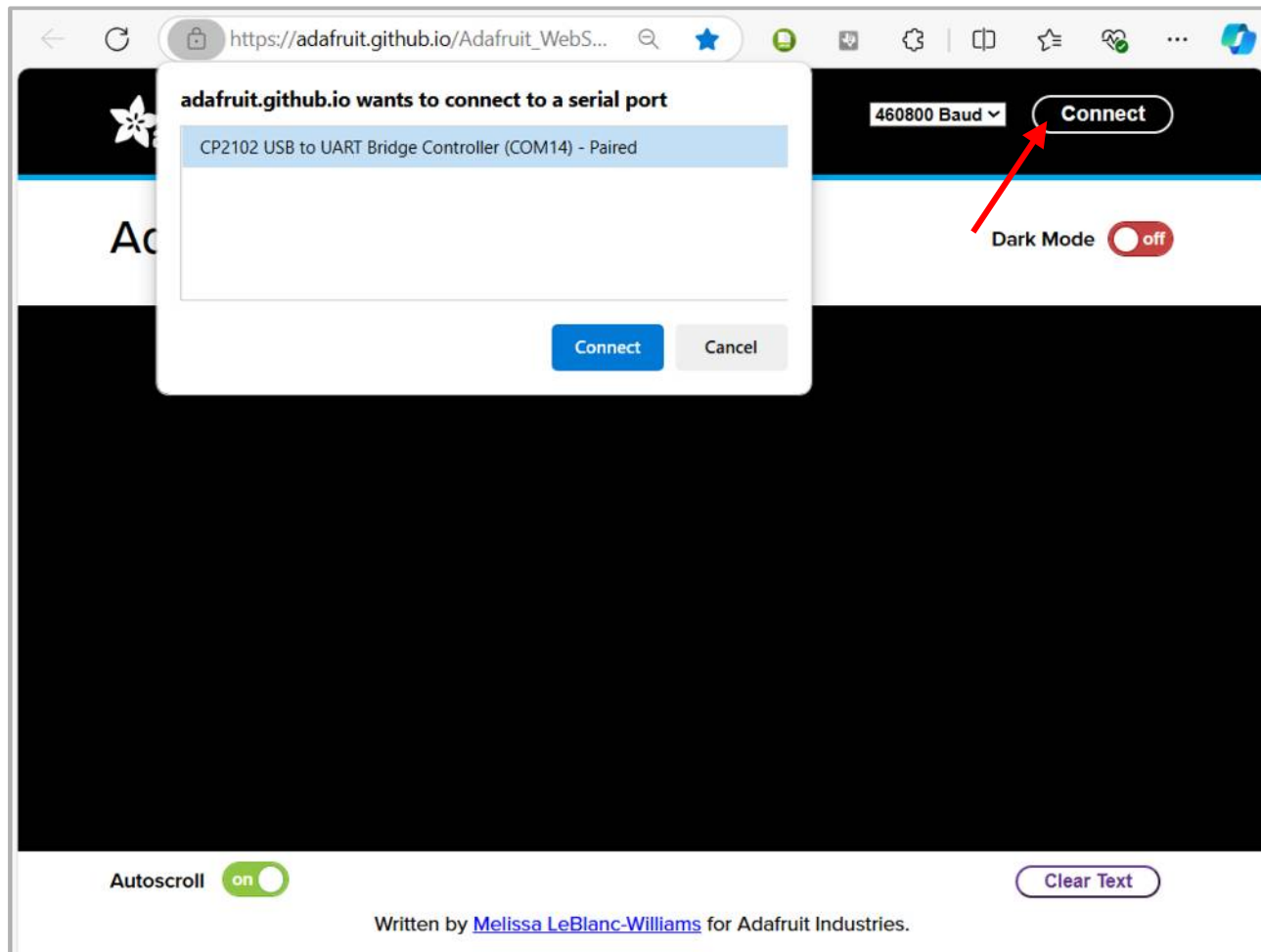
80

Previous

Next

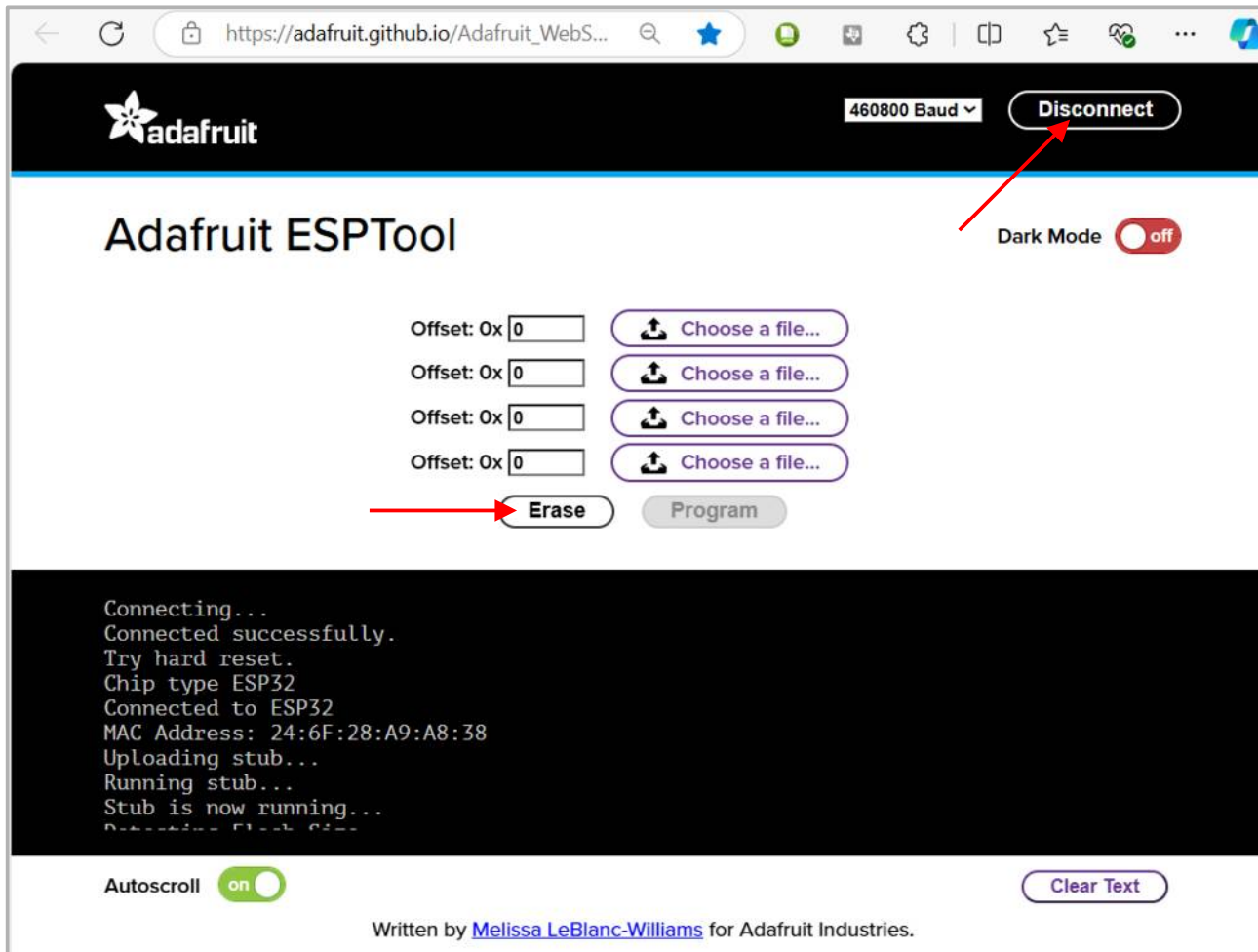
Kapcsolódás Web Serial ESPTool-hoz

- ❖ A böngészővel lépünk a Web Serial ESPTool nyitólapjára!
- ❖ Csatlakoztassuk BOOT módban a kártyánkat és tartsuk a BOOT gombot lenyomva!
- ❖ Kattintsunk a böngészőben a **CONNECT** gombra!
- ❖ Válasszuk ki az eszközünkhöz tartozó soros portot és folytassuk a kék **CONNECT** gombbal



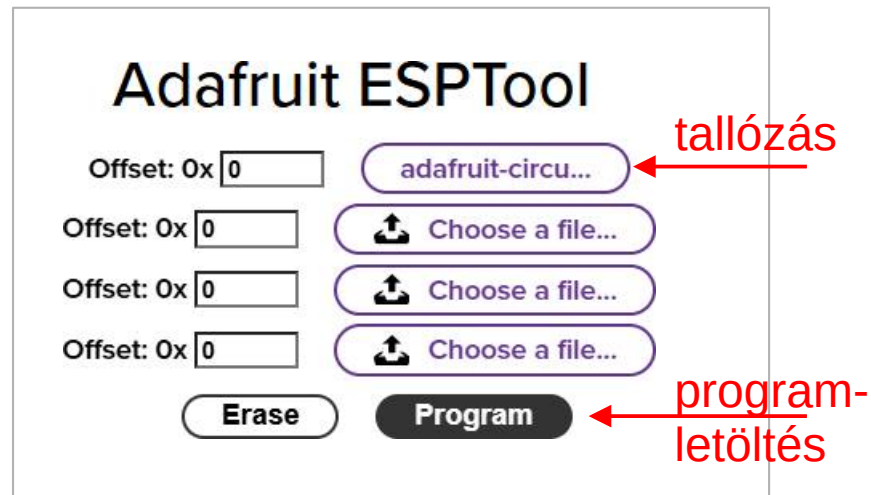
A Web Serial ESPTool használata

- ❖ Sikeres kapcsolódás esetén az ábrán látható kiírások jelennek meg a képernyőn
- ❖ Az **ERASE** gomb segítségével törölhetjük a flash memória tartalmát
- ❖ Az **DISCONNECT** gomb segítségével megszakíthatjuk a kártyával a soros porti kapcsolatot (most még ne nyomjuk!)



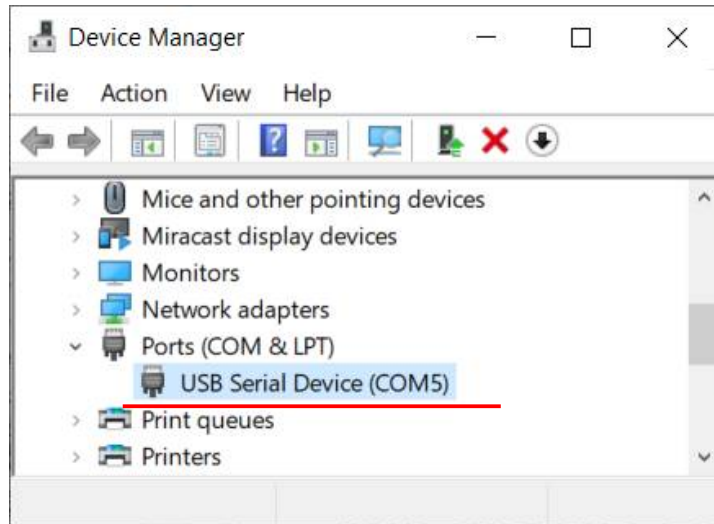
A Web Serial ESPTool használata

- ❖ A jobb felső gombra kattintva tallózzuk be az előzőleg letöltött **CircuitPython firmware**-t (adafruit-circuitpython-makergo_esp32c3_supermini-en_US-9.2.1.bin) ^{10.0.3}
- ❖ Ügyeljünk arra, hogy a betöltési kezdőcím **0x0** legyen!
- ❖ Az **ERASE** gombra kattintva a flash memória teljes tartalmát törölhetjük
- ❖ A **PROGRAM** gombra kattintva csak a felülírt flash memóriaterület törlődik, majd beírásra kerül a CircuitPython firmware
- ❖ A programletöltés után a **DISCONNECT** gombra kattintva szakítsuk meg a kapcsolatot, majd a kártya resetelése után kapcsolódjunk egy terminálablakhoz



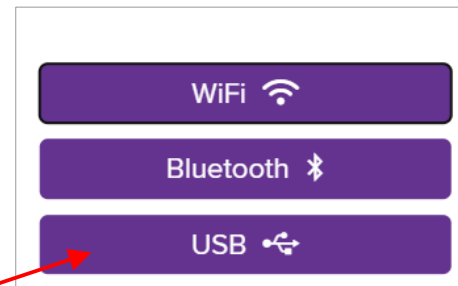
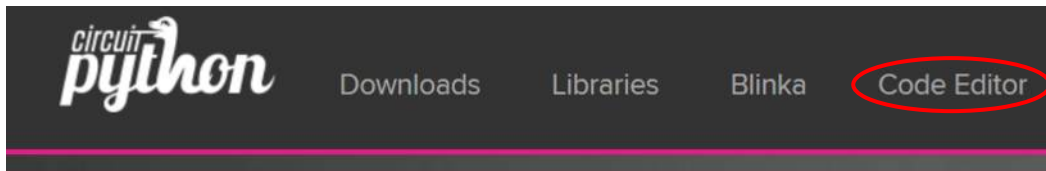
ESPTool.py – a parancssori letöltőprogram

- ❖ Programletöltésre **egy másik lehetőség** az offline, parancssori megoldás, az **esptool.py** (ehhez telepíteni kell a Python-t is, ha még nincs telepítve)
- ❖ A telepítéséhez adjuk ki az alábbi parancsot:
`pip install esptool`
- ❖ A **Device Manager**-ben keressük meg a kártyánkhoz tartozó port nevét (pl. **COM5**)
- ❖ A flash memória teljes törlése:
`esptool --port COM5 erase_flash`
- ❖ A **CircuitPython firmware** letöltése:
`esptool -p COM5 write_flash -z 0x0 firmware.bin`
ahol *firmware.bin* helyére az előzőleg letöltött firmware nevét, ill. elérési útvonalát kell megadni
- ❖ A parancsok kiadása után szükség lehet a **BOOT** gomb lenyomására!

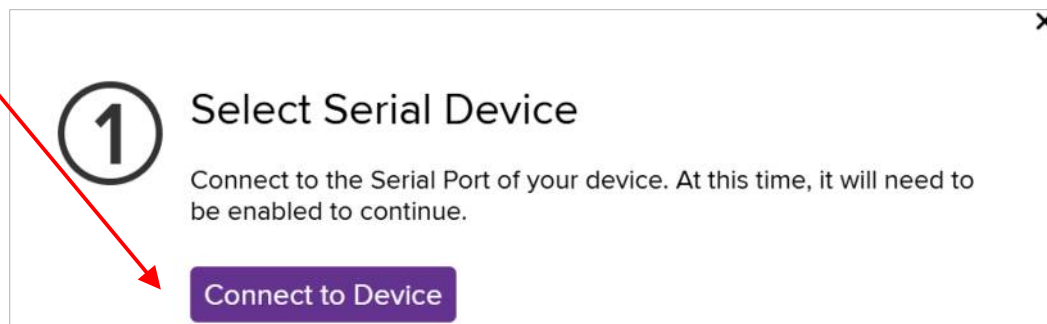


Webes munkakörnyezet

- ❖ A circuitpython.org honlapon a **Code Editor** gombra kattintva elindíthatjuk a webes munkakörnyezetet

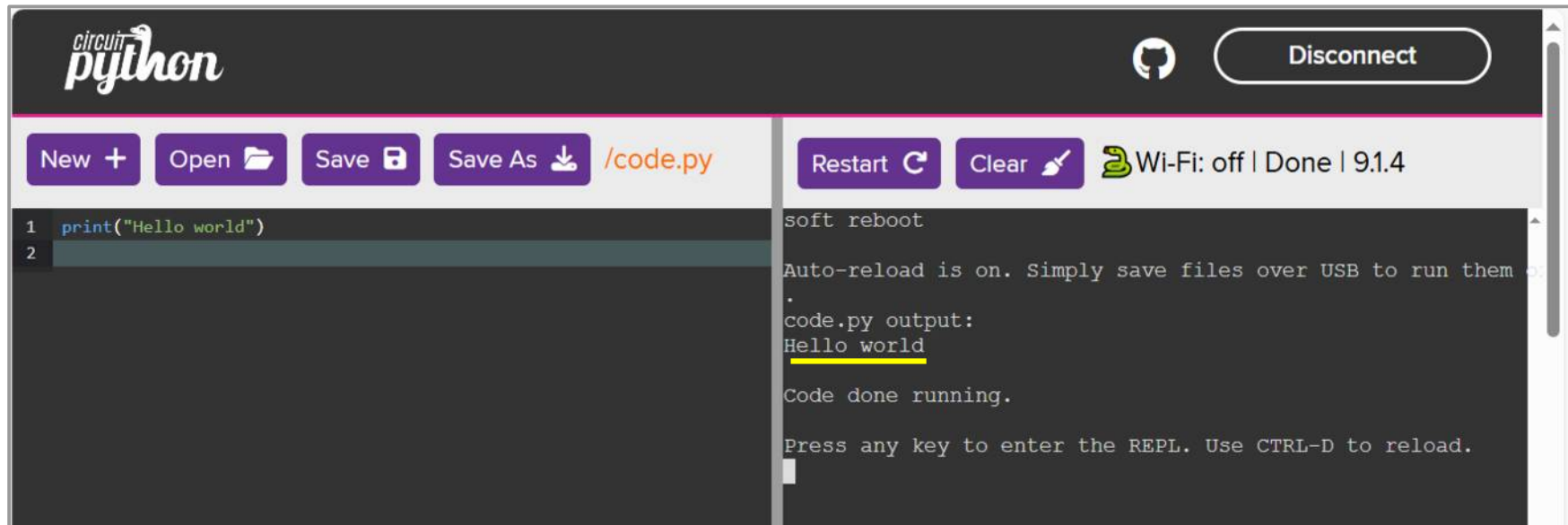


- ❖ A kapcsolódáshoz az USB opciót kell választanunk
- ❖ A **Connect to Device** gombra kattintva a szokásos módon ki kell választani a **Web Serial** kapcsolat számára a kártyához tartozó soros portot
- ❖ A kék **Connect** gombra kattintva megnyílik a kódszerkesztő és fájlkezelő ablak (USB port híján nincs CIRCUITPY: meghajtó!!!)



Helloworld.py

- ❖ Írjuk be a `print("Hello World!")` parancsot, vagy nyissuk meg a `code.py` állományt az **Open** gombra kattintva!
- ❖ A megszerkesztett programot `code.py` néven mentjük el
- ❖ A **Restart**, vagy **Save+Run** gombra kattintva lefut a program



A REPL mód használata

❖ A Serial (soros porti) ablakban **Enter** gomb nyomására beléphetünk az interaktív REPL módba (Read-Eval-Print Loop)

❖ Adjuk ki például az alábbi parancsokat

```
>>> print(5 + 8)
```

```
13
```

```
>>> help("modules")
```

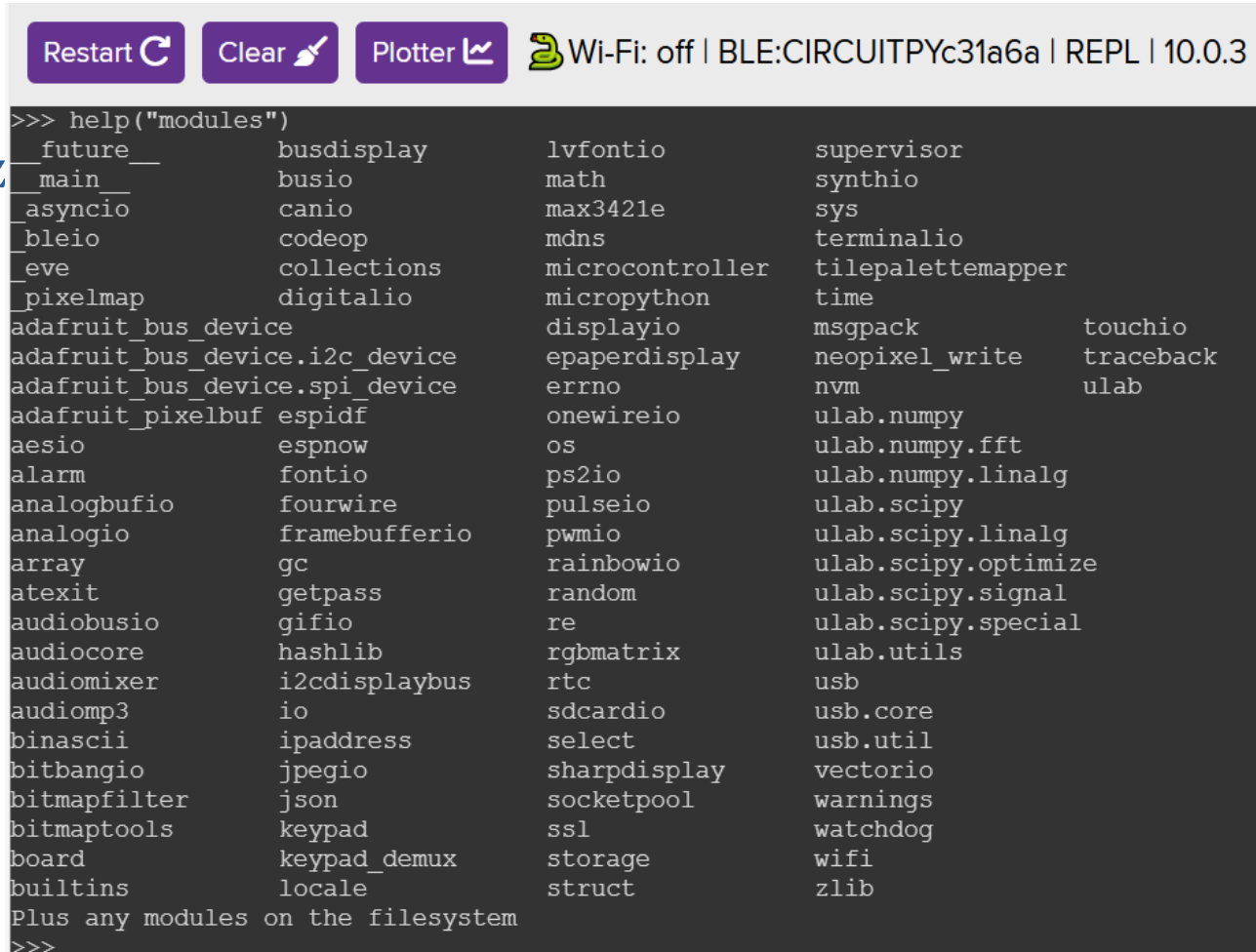
```
__future__
```

```
__main__
```

```
_asyncio
```

```
adafruit_bus_device
```

```
...
```



The screenshot shows the REPL interface with a dark background. At the top, there are three buttons: 'Restart' (with a circular arrow icon), 'Clear' (with a broom icon), and 'Plotter' (with a line graph icon). To the right of these buttons, the status bar displays: 'Wi-Fi: off | BLE:CIRCUITPYc31a6a | REPL | 10.0.3'. The main area shows the command prompt '>>>' followed by 'help("modules")'. The output is a list of modules arranged in four columns. The first column lists modules like __future__, __main__, _asyncio, bleio, _eve, _pixelmap, adafruit_bus_device, etc. The second column lists modules like busdisplay, busio, canio, codeop, collections, digitalio, etc. The third column lists modules like lvfontio, math, max3421e, mdns, microcontroller, micropython, displayio, etc. The fourth column lists modules like supervisor, synthio, sys, terminalio, tilepalettemapper, time, msgpack, touchio, etc. The list ends with 'Plus any modules on the filesystem' and another '>>>' prompt.

```
>>> help("modules")
__future__      busdisplay      lvfontio        supervisor
__main__        busio           math            synthio
_asyncio        canio          max3421e        sys
bleio          codeop         mdns            terminalio
_eve           collections    microcontroller tilepalettemapper
_pixelmap      digitalio      micropython     time
adafruit_bus_device
adafruit_bus_device.i2c_device
adafruit_bus_device.spi_device
adafruit_pixelbuf
aesio          espidf         onewireio       msgpack          touchio
alarm          espnow         os              ulab.numpy       ulab.traceback
analogbufio   fourwire       ps2io           ulab.numpy.fft   ulab
analogio      framebufferio  pulseio         ulab.numpy.linalg
array         gc             pwmio           ulab.scipy        ulab.scipy.linalg
atexit        getpass        random          ulab.scipy.optimize
audiobusio    gifio          re              ulab.scipy.signal
audiocore     hashlib        rgbmatrix       ulab.scipy.special
audiomixer    i2cdisplaybus rtc             ulab.utils        usb
audiomp3      io             sdcardio        usb.core
binascii     ipaddress     select          usb.util
bitbangio    jpegio        sharpdisplay    vectorio
bitmapfilter  json          socketpool      warnings
bitmaptools  keypad        ssl             watchdog
board        keypad_demux  storage         wifi
builtins     locale       struct          zlib
Plus any modules on the filesystem
>>>
```

A REPL mód használata

- ❖ Az első LED villogtató program megírásához derítsük ki, hogy hogyan hívják az egyes kivezetéseket!

- ❖ Adjuk ki az alábbi parancsokat

```
>>> import board
>>> dir(board)
['__class__', '__name__', 'BOOT0', 'BUTTON', 'I2C', 'IO0', 'IO1',
'IO10', 'IO2', 'IO20', 'IO21', 'IO3', 'IO4', 'IO5', 'IO6', 'IO7',
'IO8', 'IO9', 'LED', 'MISO', 'MOSI', 'RX', 'SCK', 'SCL', 'SDA',
'SPI', 'TX', 'UART', '__dict__', 'board_id']
>>> print(board.LED)
board.IO8
```

- ❖ A beépített LED tehát `board.IO8`, vagy `board.LED` néven érhető el

A beépített LED villogtatása: ledblink.py

- ❖ Írjuk be az alábbi programot!
- ❖ A **Save as** gombbal töltsük le a mikrovezérlőre **code.py** néven!
- ❖ **Restart**, vagy **Ctrl-D** nyomására a program elindul és végtelen ciklusban kétmásodpercenként felvillantja a beépített LED-et
- ❖ A **time.sleep()** függvény paraméterét másodpercben kell megadni
- ❖ **Megjegyzés:** a beépített LED katódját vezéreljük, ezért **a kimenet lehúzásakor** világít a LED!

```
import board
import digitalio
import time

led = digitalio.DigitalInOut(board.LED)
led.direction = digitalio.Direction.OUTPUT

while True:
    led.value = False
    time.sleep(0.25)
    led.value = True
    time.sleep(1.75)
```

ledblink.py

A kártya kivezetéseinek listázása: pin_list.py

- ❖ Listázzuk ki a kivezetések neveit és másodlagos neveit!

```
import microcontroller
import board

board_pins = []
for pin in dir(microcontroller.pin):
    if isinstance(getattr(microcontroller.pin, pin), microcontroller.Pin):
        pins = []
        for alias in dir(board):
            if getattr(board, alias) is getattr(microcontroller.pin, pin):
                pins.append("board.{}".format(alias))
        if len(pins) > 0:
            board_pins.append(" ".join(pins))
for pins in sorted(board_pins):
    print(pins)
```

pin_list.py

code.py output:

```
board.I00
board.I01
board.I02
board.I03
board.I04 board.SCK
board.I05 board.MISO
board.I06 board.MOSI
board.I07
board.I08 board.LED board.SDA
board.I09 board.SCL \
        board.BOOT0 board.BUTTON
board.I010
board.I020 board.RX
board.I021 board.TX
```

Digitális ki- és bemenetek kezelése

- ❖ A digitális ki- és bemenetek kezelése a **DigitalInOut** objektumok segítségével történik, például:

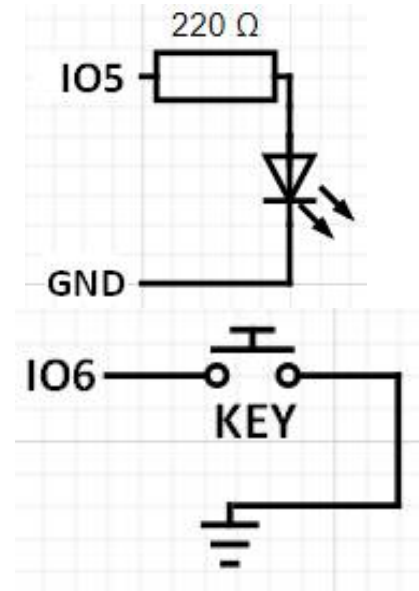
- ❖ **Kimenet konfigurálása:**

```
import digitalio, board
led = digitalio.DigitalInOut(board.IO5)
led.direction = digitalio.Direction.OUTPUT
led.drive_mode = digitalio.DriveMode.OPEN_DRAIN
```

- ❖ **Bemenet konfigurálása:**

```
import digitalio, board
button = digitalio.DigitalInOut(board.IO6)
button.direction = digitalio.Direction.INPUT
button.pull = digitalio.Pull.UP
```

- ❖ Megjegyzés: nem kell ilyen hosszú litániákat írni, ha így importálunk:
`from digitalio import DigitalInOut, Direction, DriveMode, Pull` vagy
`from digitalio import *`



LED vezérlése nyomógommbal: ledswitch.py

- ❖ Kapcsolgassuk egy LED-et (IO5) az IO6 és a GND közé kötött nyomógomb segítségével!

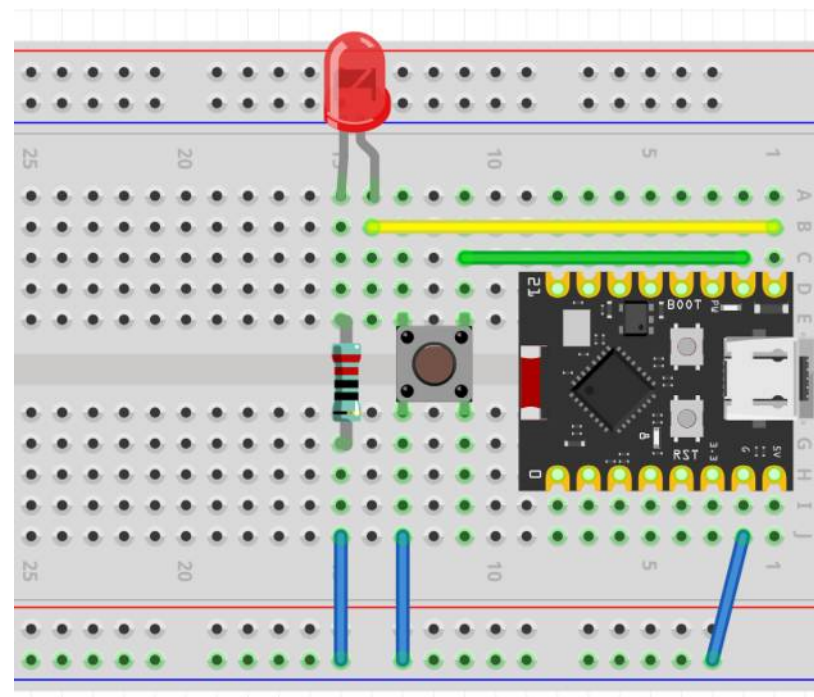
ledswitch.py

```
from digitalio import *           # DigitalInOut, Direction,
import time, board                # Pull, DriveMode

led = DigitalInOut(board.IO5)     # LED (IO5)
led.direction = Direction.OUTPUT
led.drive_mode = DriveMode.PUSH_PULL
led.value = False                # Initially OFF

button = DigitalInOut(board.IO6)  # Pushbutton (IO6)
button.direction = Direction.INPUT
button.pull = Pull.UP

while True:
    while button.value: pass      # Wait for keypress
    led.value = not led.value     # Flip LED state
    time.sleep(0.02)
    while not button.value:      # Wait for key release
        pass
    time.sleep(0.02)
```



LED vezérlés nyomógommbal másképp: ledswitch2.py

- ❖ Kapcsolgassuk a beépített LED-et (IO8) a beépített nyomógomb (IO9) segítségével!

ledswitch2.py

```
from digitalio import *           # DigitalInOut, Direction,
import time, board                # Pull, DriveMode

led = DigitalInOut(board.LED)     # LED (IO8)
led.direction = Direction.OUTPUT
led.drive_mode = DriveMode.PUSH_PULL
led.value = True                  # Initially OFF

button = DigitalInOut(board.BUTTON) # Pushbutton (IO9)
button.direction = Direction.INPUT
button.pull = Pull.UP

while True:
    while button.value: pass       # Wait for keypress
    led.value = not led.value      # Flip LED state
    time.sleep(0.02)
    while not button.value:       # Wait for key release
        pass
    time.sleep(0.02)
```

A beépített LED-nek a katódja van kivezetve az **IO8** lábon, ezért „fordított logikával” kell vezérelni

Ezért a `led.value = True` parancs hatására a LED kialszik

A nyomógomb helyes kezelése:

1. Lenyomásra várunk
2. Elvégezzük a tevékenységet
3. Pergésmentesítési várakozás
4. Felengedésre várunk
5. Pergésmentesítési várakozás

A Python nyelv elemei

- ❖ A **Python** objektumorientált nyelv, ami lehetővé teszi az adatok és a funkciók objektumokba szervezését
- ❖ **Osztályok és objektumok:**
 - **Osztály (class):** Az objektumok „tervrajza”
 - **Objektum:** Az osztály egy példánya
- ❖ Példa:

```
class Személy:  
    def __init__(self, név, kor):  
        self.név = név  
        self.kor = kor  
anna = Személy("Anna", 25)  # 'anna' egy Személy objektum
```

- ❖ A gyakorlatban a fentihez hasonló objektumosztályok különféle funkciókat is definiálnak, amely minden objektumpéldányban rendelkezésre áll

Változók létrehozása és használata

- ❖ A **változó** egy névvel ellátott objektum, amely egy értéket tárol
- ❖ Például: $x = 5$ - itt az **x** változó az 5 értéket tárolja
- ❖ **Változók létrehozása:** `változónév = érték`
- ❖ Példák:

```
név = "Anna"  
kor = 25
```
- ❖ **Változók típusai:**
 - **Szöveg** (string): `név = "Anna"`
 - **Egész szám** (integer): `kor = 25`
 - **Lebegőpontos szám** (float): `magasság = 1.75`

Összetett változók

- ❖ **Lista (list):** Több érték tárolása egy változóban.

Példa:

```
gyümölcsök = ["alma", "banán", "cseresznye"]
```

- ❖ **Tuple (tuple):** Több érték tárolása egy változóban, de az értékek nem módosíthatók.

Példa:

```
színek = ("piros", "zöld", "kék")
```

- ❖ **Szótár (dictionary):** Kulcs-érték párok tárolása.

Példa:

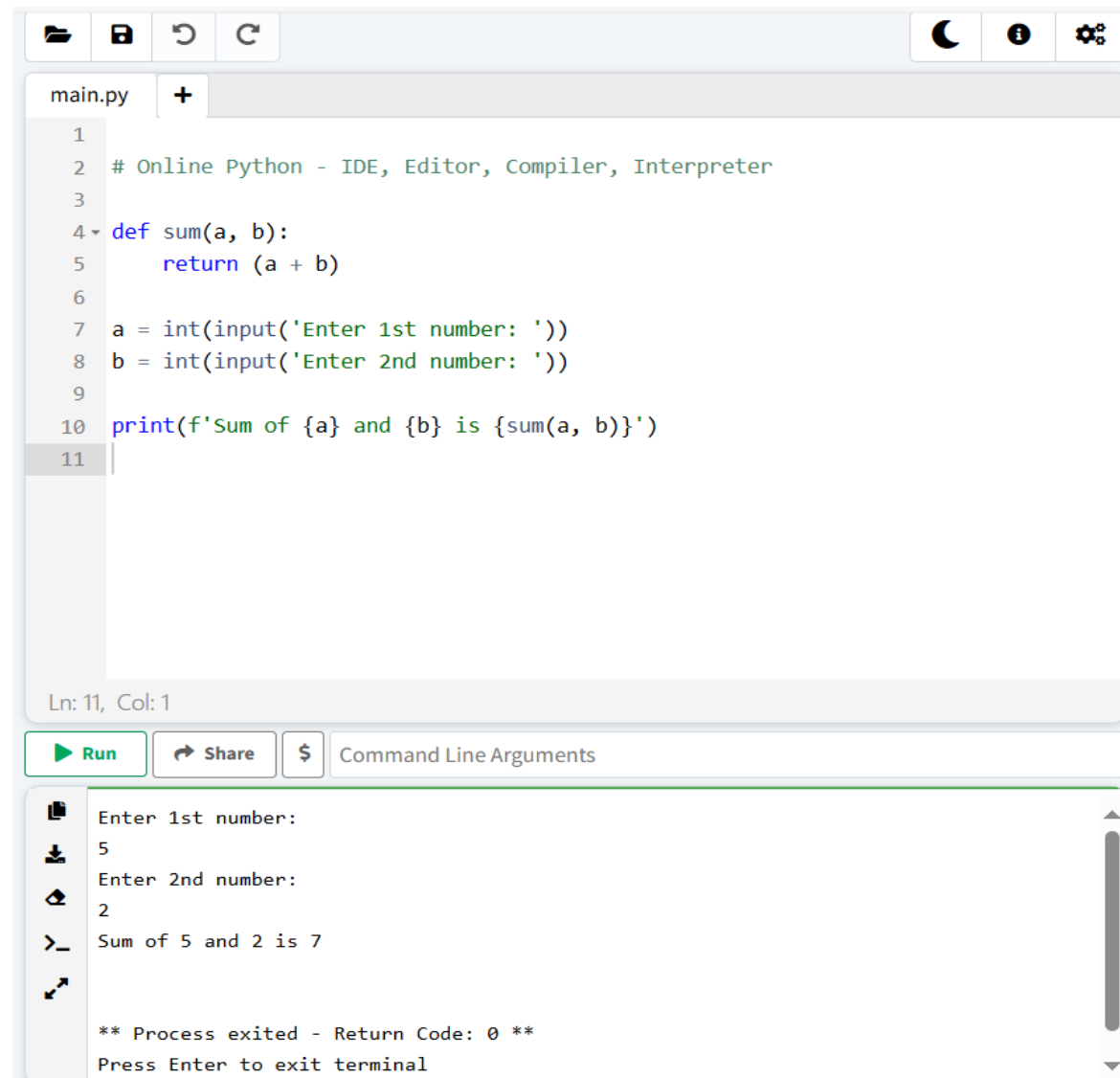
```
diák = {"név": "Anna", "kor": 25, "város": "Budapest"}
```

Alapvető műveletek

- ❖ A változókkal az értékedáson kívül műveleteket is végezhetünk.
Az alapvető operátorok a Pythonban: $+$, $-$, $*$, $/$, $//$, $\%$ és $**$, amelyek az összeadást, kivonást, szorzást, osztást, egész osztást, maradékot és hatványozást jelölik
- ❖ Legyen például $x = 5$, $y = 2$, akkor:
 - Összeadás: $x + y = 7$
 - Kivonás: $x - y = 3$
 - Szorzás: $x * y = 10$
 - Osztás: $x / y = 2.5$
 - Egészosztás: $x // y = 2$ (lefelé kerekít a legközelebbi egész számra)
 - Maradék: $x \% y = 1$ (az osztási maradékot adja vissza)
 - Hatványozás: $x ** y = 25$ (az 5 második hatványa)

Python szimulátor

- ❖ URL:
www.online-python.com/
- ❖ Itt az online felületen a Python nyelvet próbálhatjuk ki, ami bővebb a CircuitPythonnál, viszont fizikai programozásra nem tudjuk használni



The screenshot displays the Online Python IDE interface. At the top, there are icons for file operations (folder, save, undo, redo) and system settings (moon, info, gear). Below this is a tab labeled 'main.py' with a '+' icon to add more files. The main editor area contains a Python script with the following code:

```
1
2 # Online Python - IDE, Editor, Compiler, Interpreter
3
4 def sum(a, b):
5     return (a + b)
6
7 a = int(input('Enter 1st number: '))
8 b = int(input('Enter 2nd number: '))
9
10 print(f'Sum of {a} and {b} is {sum(a, b)}')
11
```

Below the editor, a status bar shows 'Ln: 11, Col: 1'. There are three buttons: 'Run' (green play icon), 'Share' (share icon), and a button with a '\$' icon for 'Command Line Arguments'. Below these buttons is a terminal window showing the execution output:

```
Enter 1st number:
5
Enter 2nd number:
2
Sum of 5 and 2 is 7

** Process exited - Return Code: 0 **
Press Enter to exit terminal
```