ESP32 mikrovezérlők programozása CircuitPython környezetben

CircuitPython és ESP32 – 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

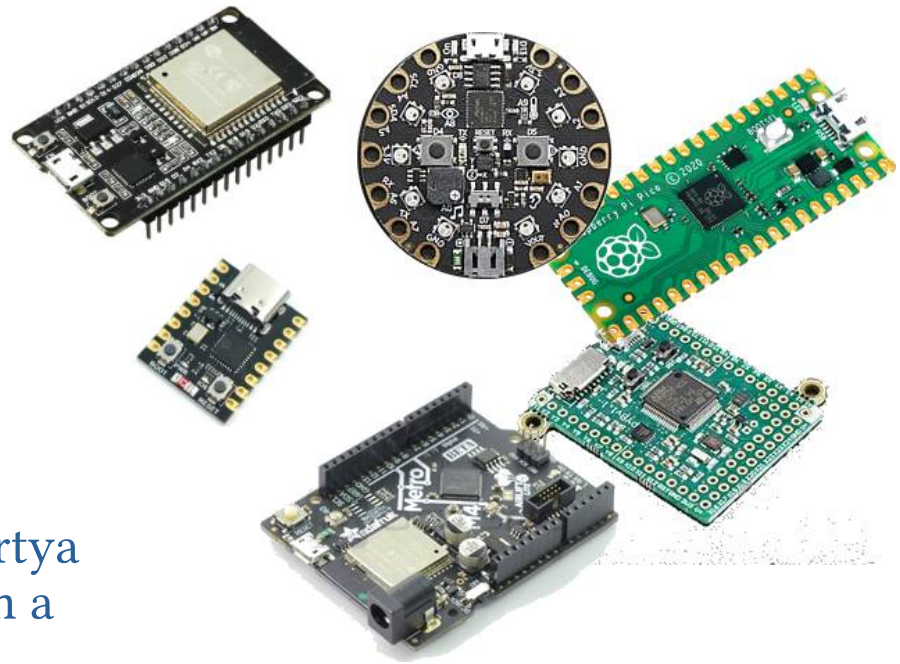
- **Online eszközök és támogatás:**
  - Learn Adafruit: CircuitPython on ESP32 Quick Start
  - Adafruit: Adafruit Web Serial ESPTool
  - Adafruit: CircuitPython Code Editor

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

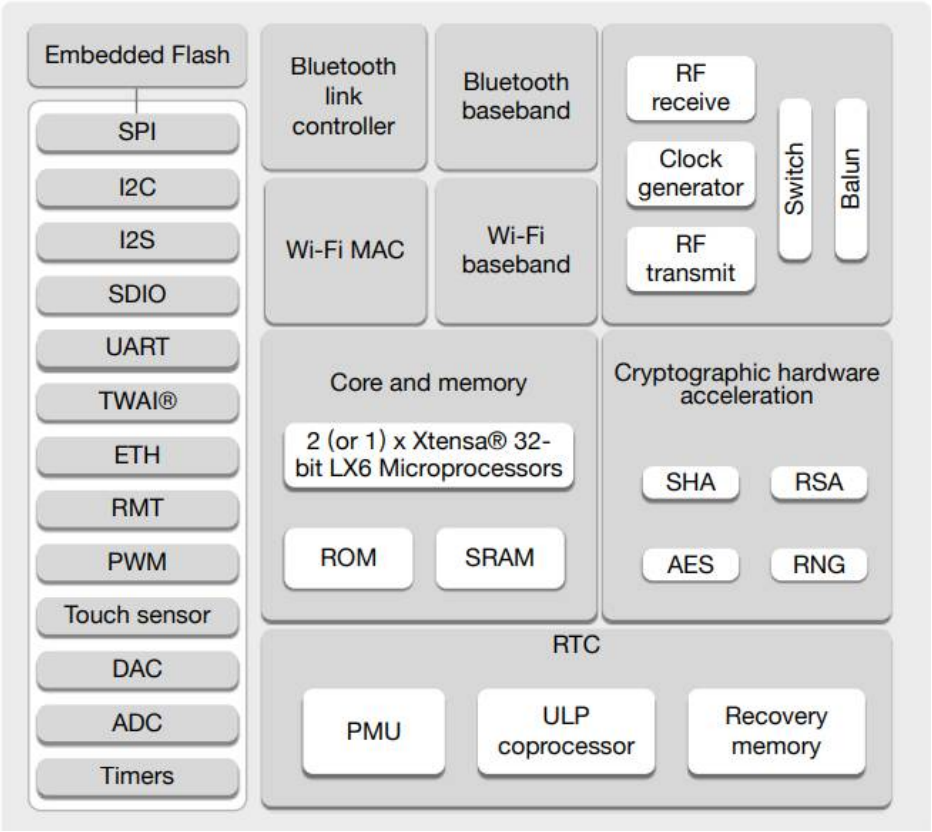
- A támogatott kártyák száma e sorok írásakor: **544** (ez 290-nel több, mint 3 éve!)
- Csak példa gyanánt néhány ismertebb típus: Raspberry Pi Pico, Circuit Playgroung Express, Feather M4 express, **ESP32 Doit Devkit,** Nano RP2040 Connect, Metro ESP32-S2, nRF52 840 Dongle, Feather STM32F405 Express, **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 az **ESP32 Doit Devkit** kártyát fogjuk használni, a **CircuitPython 9.1.4** változatával (lásd: circuitpython.org/board/doit_esp32_devkit_v1/)

Az ESP32 mikrovezérlő bemutatása

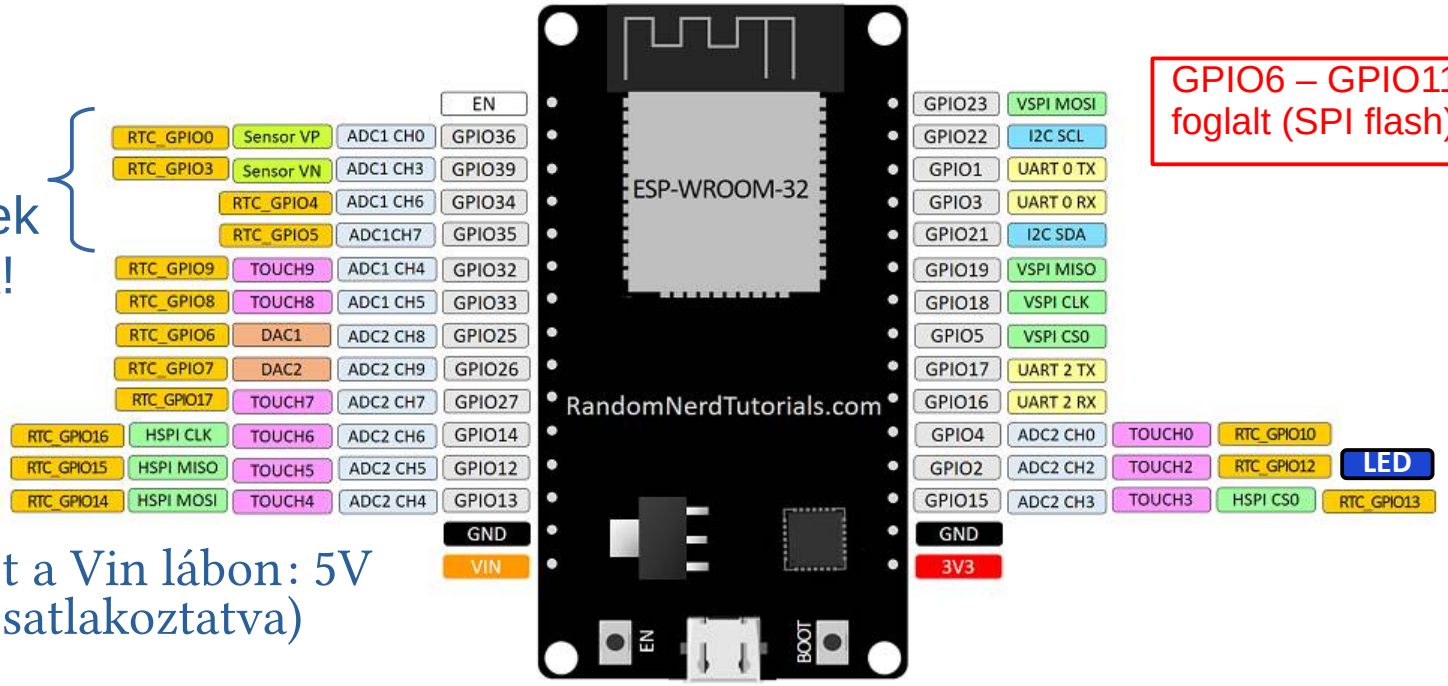
- Az **ESP32** az **Expressif** kétmagos mikro-vezérlője, beépített WiFi és Bluetooth/BLE kommunikációs képességgel
- CPU: 2x 32bites Xtensa LX6 mag
- ROM: 448 kB (firmware)
- RAM: 520 kB + 16 kB RTC RAM
- Flash: external QSPI
- 34 GPIO, 18 analog (ADC 12 bit),
  10 x touch sensor, 16 x PWM,
  2 x 8bit DAC, 3 x SPI, 2 x I2C, 2 x I2S,
  3 x UART, IR (Tx/Rx), TWAI, Hall sensor
  1 host (SD/eMMC/SDIO), 1 slave (SDIO/SPI),
  1024-bit OTP, up to 768-bit for customers,
  Cryptographic hardware acceleration



Az ESP32 Devkit-1 (DOIT) kártya kivezetései

- A Doit **ESP32 Devkit-1** kártya egy **ESP WROOM-32** modult (ESP32 + 4MB flash) és az alapkártyán egy **CP2102** USB-UART átalakítót, egy 3,3 V-os stabilizátort, egy **Reset** és egy **Boot** nyomógombot tartalmaz

Csak bemenetek lehetnek!



GPIO6 – GPIO11: foglalt (SPI flash)

- Tápellátás bemenet a Vin lábon: 5V (ha nincs USB-re csatlakoztatva)

Forrás:
randomnerdtutorials.com/getting-started-with-esp32/

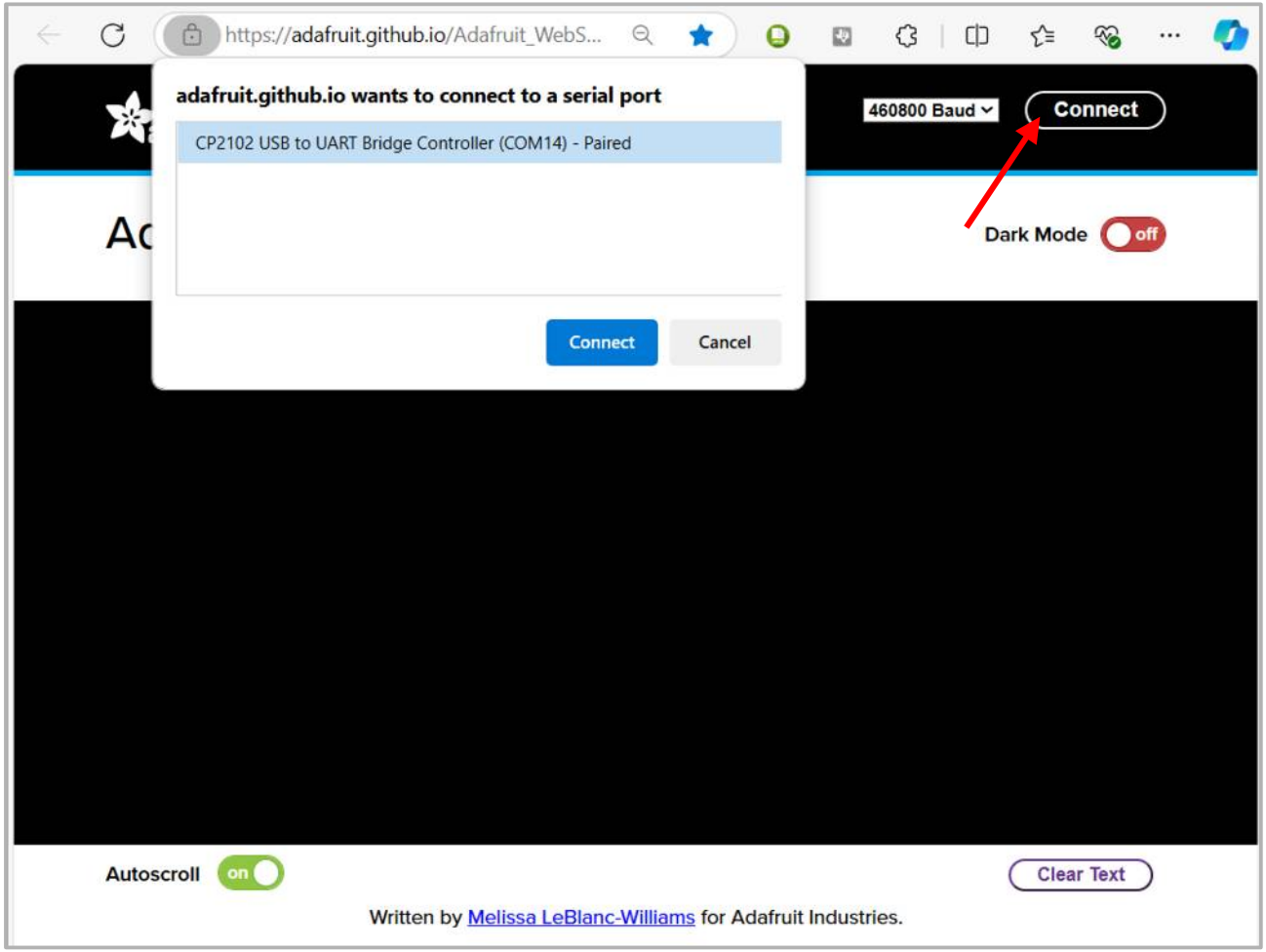
Pin 30 version

A CircuitPython firmware telepítése

- Mivel a kártyánk gyárilag nem tartalmazza a **CircuitPython** firmware-t, így bele kell töltenünk (csak egyszer kell végrehajtani)
- Az **ESP32 Doit Devkit** kártyához a CircuitPython 9.1.4 változatát innen töltöttük le: circuitpython.org/board/doit_esp32_devkit_v1/
- A felprogramozást a mikrovezérlő gyárilag beégetett bootloadere segítségével végezhető el, ehhez vagy a **Web Serial ESPTool,** vagy a Pythonban írt parancssori **ESPTool** letöltő programot használjuk
  - A **Web Serial ESPTool** használata egyszerűbb, de **Chrome,** vagy más, Chrome alapú (*Edge, Opera*) böngésző kell hozzá, amiben engedélyezhető a soros portok elérése (a **chrome://flags** URL megnyitása után az **Experimental Web Platform features** opció legyen **Enabled**-re állítva)
  - Az **ESPTool.py** letöltőt **Python** környezetbe telepíteni kell a `pip install esptool` paranccsal

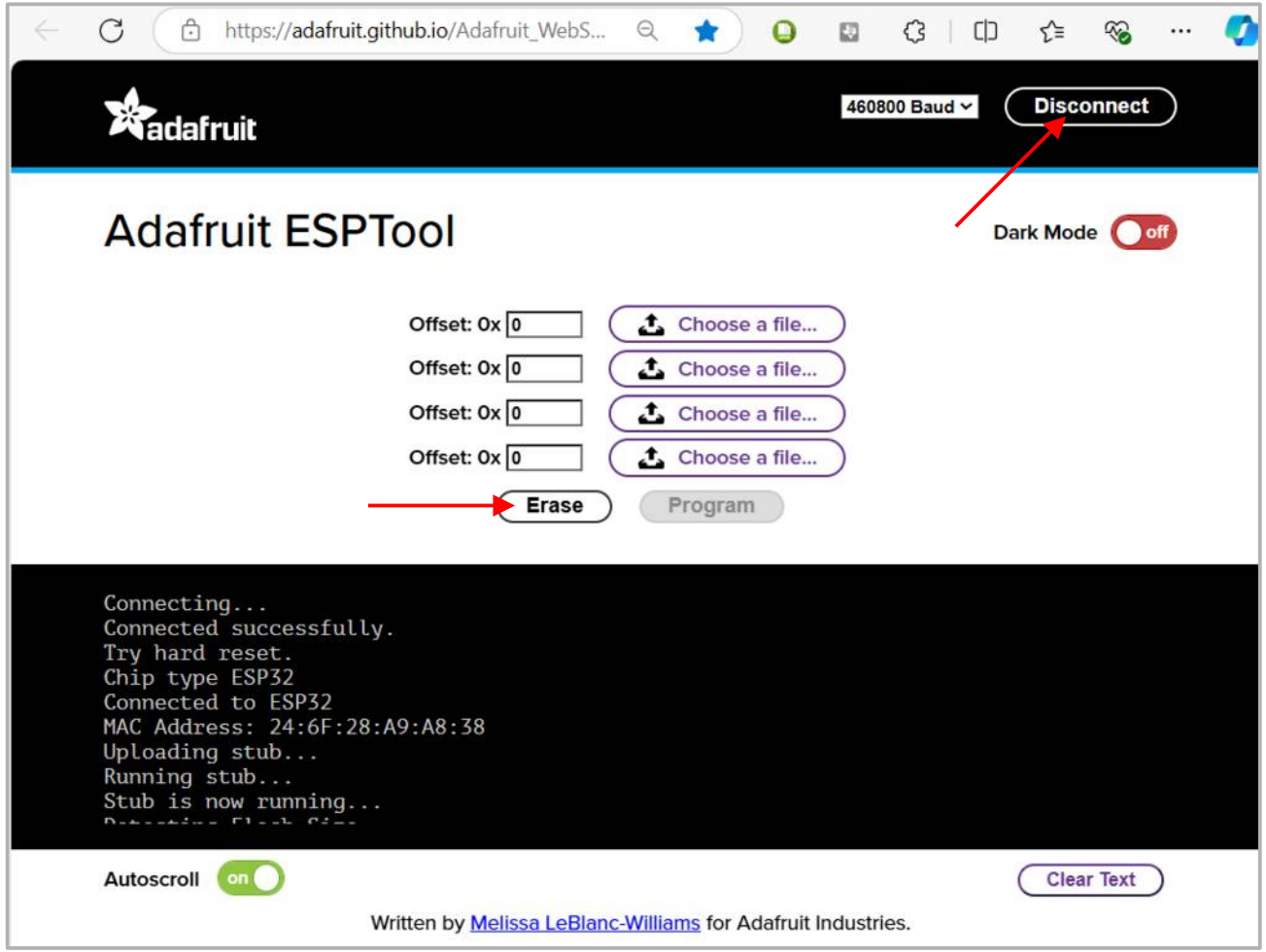
Kapcsolódás Web Serial ESPTool-hoz

- A böngészővel lépjünk 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



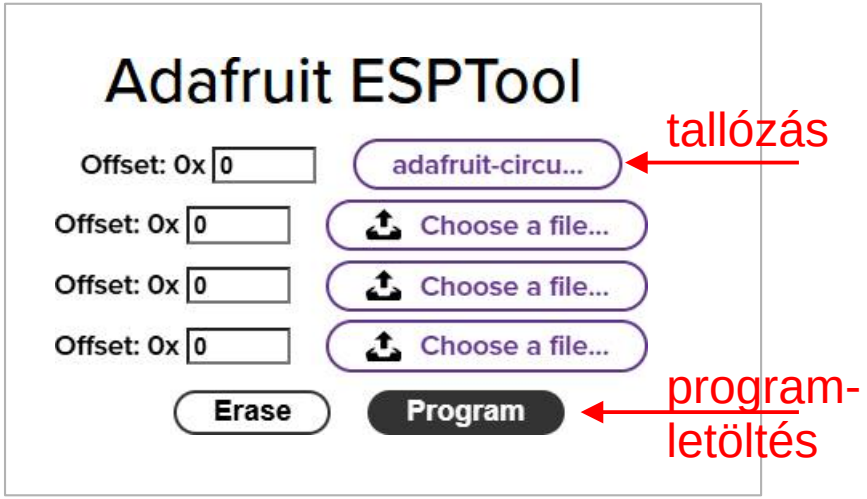
A Web Serial ESPTool használata

- 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



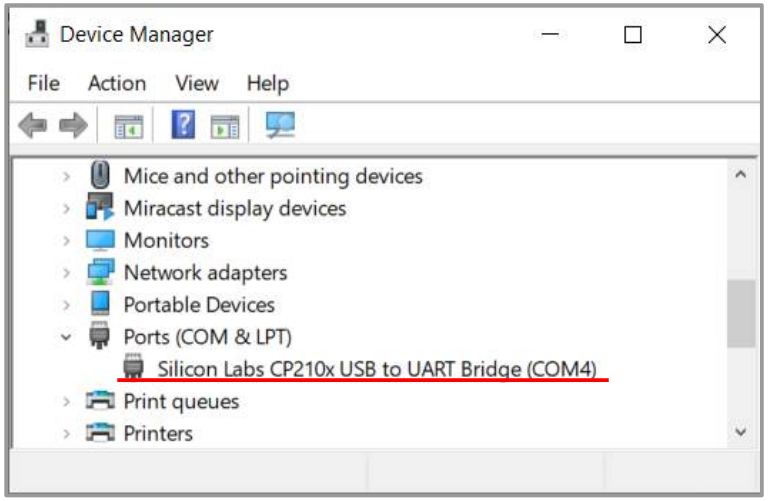
A Web Serial ESPTool használata

- A jobb felső gombra kattintva tallózzuk be az előzőleg letöltött **CircuitPython firmware**-t (adafruit-circuitpython-doit_esp32_devkit_v1-en_US-9.1.4.bin)
- 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

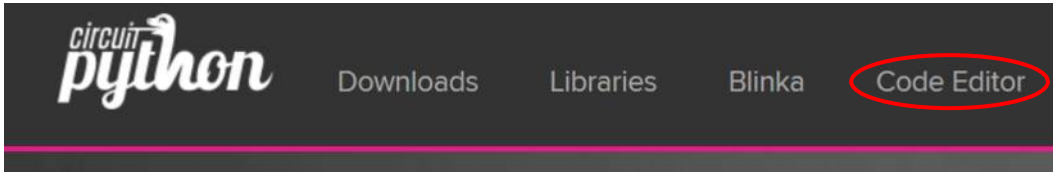
- Programletöltésre **egy másik lehetőség** az offline, parancssori megoldás, az **esptool.py** (ehhez telepíteni kell a Python-t is, ha még nincs telepítve)
- A telepítéséhez adjuk ki az alábbi parancsot:
  `pip install esptool`
- A **Device Manager**-ben keressük meg a kártyánkhoz tartozó port nevét (**COM4**)

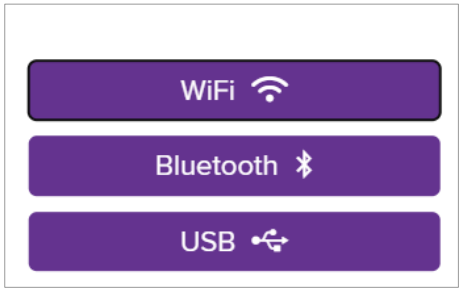


- A flash memória teljes törlése:
  `esptool --port COM4 erase_flash`
- A **CircuitPython firmware** letöltése:
  `esptool -p COM4 write_flash -z 0x0 firmware.bin`
  ahol *firmware.bin* helyére az előzőleg letöltött firmware nevét, ill. elérési útvonalát kell megadni
- A parancsok kiadása után szükség lehet a **BOOT** gomb lenyomására!

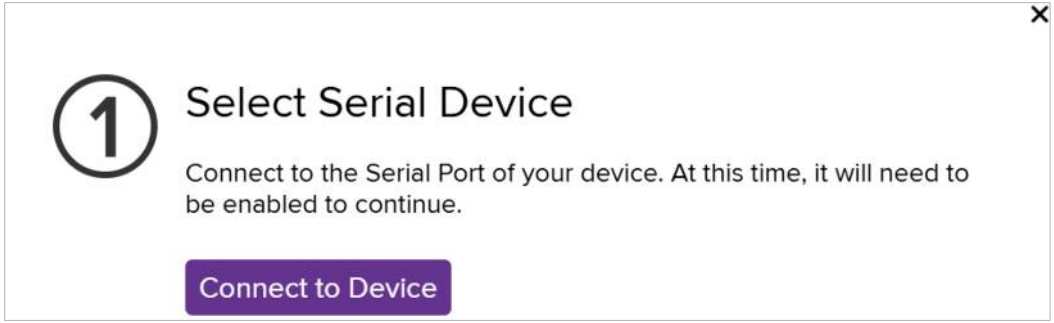
Webes munkakörnyezet

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





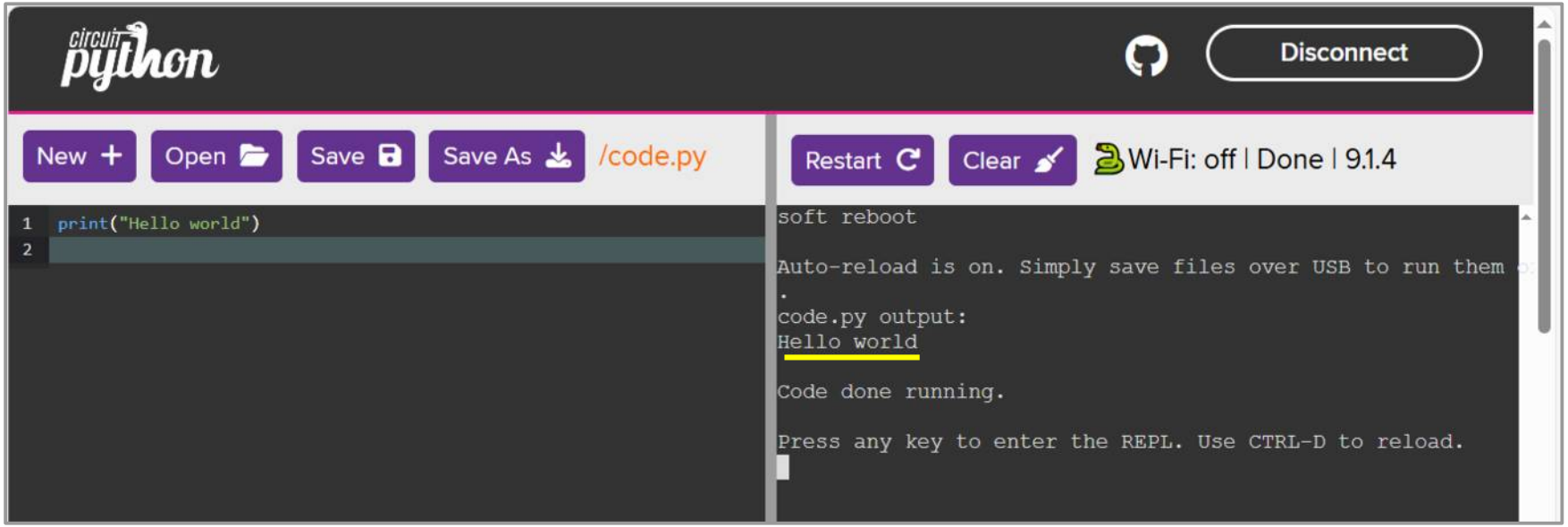
- A kapcsolódáshoz az USB opciót kell választanunk
- A **Connect to Device** gombra kattintva a szokásos módon ki kell választani a **Web Serial** kapcsolat számára a kártyához tartozó soros portot



- 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 mentsük el
- A **Restart,** vagy **Save+Run** gombra kattintva lefut a program

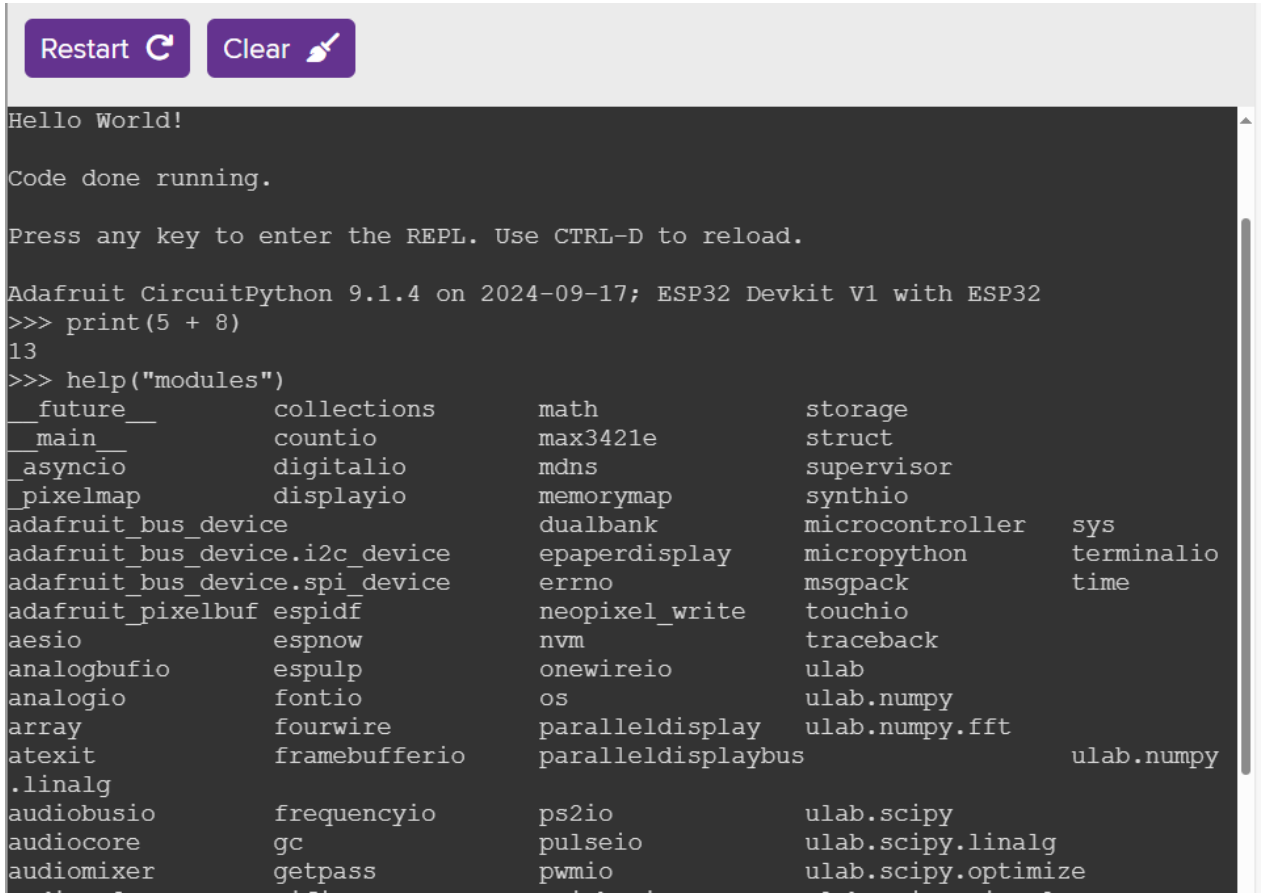


A REPL mód használata

- A 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

>>> print(”Hello World!”)
__future__
__main__
_asyncio
adafruit_bus_device
...
```



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__', 'D1', 'D12', 'D13', 'D14', 'D15', 'D16',
'D17', 'D18', 'D19', 'D2', 'D21', 'D22', 'D23', 'D25', 'D26', 'D27',
'D3', 'D32', 'D33', 'D34', 'D35', 'D4', 'D5', 'I2C', 'LED', 'MISO',
'MOSI', 'RX', 'RX0', 'RX2', 'SCK', 'SCL', 'SDA', 'SPI', 'TX', 'TX0',
'TX2', 'UART', 'VN', 'VP', '__dict__', 'board_id']
>>> print(board.LED)
board.D2
```

- A beépített LED tehát `board.D2`, 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

ledblink.py

```
import board
import digitalio
import time

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

while True:
    led.value = True
    time.sleep(0.05)
    led.value = False
    time.sleep(1.95)
```

A kártya kivezetéseinek listázása: pin_list.py

- Listázzuk ki a kivezetések neveit és másodlagos neveit!

pin_list.py

```
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)
```

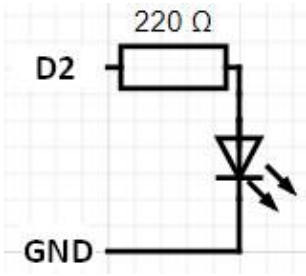
```
code.py output:
board.D1  board.RX0
board.D12
board.D13
board.D14
board.D15
board.D16 board.RX   board.RX2
board.D17 board.TX   board.TX2
board.D18 board.SCK
board.D19 board.MISO
board.D2  board.LED
board.D21 board.SDA
board.D22 board.SCL
board.D23 board.MOSI
board.D25
board.D26
board.D27
board.D3  board.TX0
board.D32
board.D33
board.D34
board.D35
board.D4
board.D5
board.VN
board.VP
```

- A **GPIO*n*** és **D*n*** jelölések számozása megegyezik
- A **GPIO6 – GPIO11** kivezetések a flash memória kezelésére fentartottak, ezért hiányoznak a listáról

Digitális ki- és bemenetek kezelése

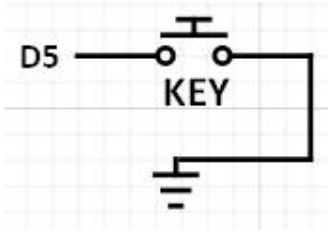
- A digitális ki- és bemenetek kezelése a DigitalInOut objektumok segítségével történik, például:
- **Kimenet konfigurálása:**

```
import digitalio, board
led = digitalio.DigitalInOut(board.D2)
led.direction = digitalio.Direction.OUTPUT
led.drive_mode = digitalio.DriveMode.OPEN_DRAIN
```



- **Bemenet konfigurálása:**

```
import digitalio, board
button = digitalio.DigitalInOut(board.D5)
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 nyomógombbal: ledswitch.py

- Kapcsolgassuk a beépített LED-et (**D2**) a **D5** és a GND közé kötött nyomógomb segítségével!

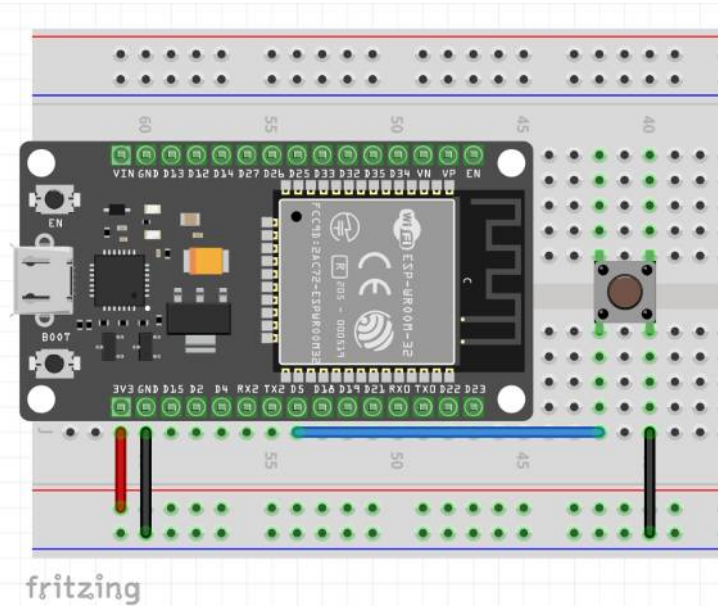
ledswitch.py

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

led = DigitalInOut(board.LED)           # Builtin LED (D2)
led.direction = Direction.OUTPUT
led.drive_mode = DriveMode.PUSH_PULL
led.value = False                       # Initially OFF

button = DigitalInOut(board.D5)         # Pusbutton (D5)
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)
```



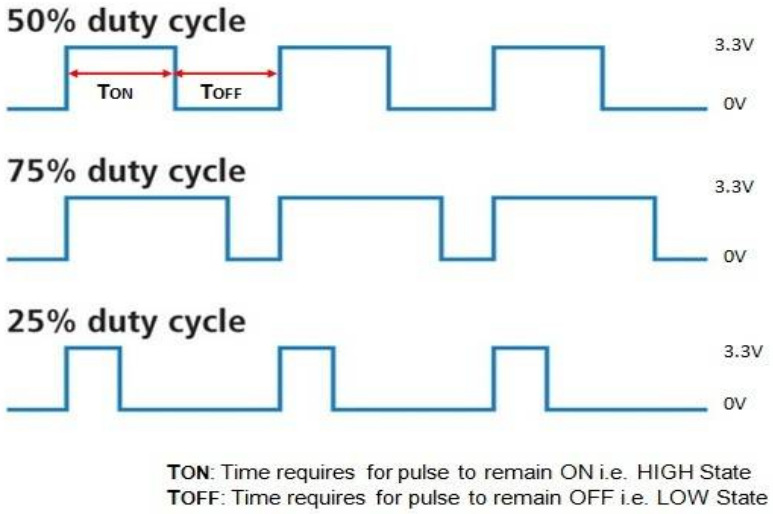
Impulzus-szélesség moduláció (PWM)

- Az impulzus-szélesség modulációs kimenetek konfigurálására és kezelésére a **pwmio** modul **PWMOut** osztálya szolgál
- A kitöltés (duty cycle) 0 – 65 535 közötti érték lehet
- A frekvencia lehet rögzített, vagy menet közben változtatható
- A konstruktor:

```
PWMOut(pin:Pin,frequency:int,duty_cycle:int,var_freq:bool)
```

ahol:

- ***pin*** – a választott kivezetés
- ***frequency*** – a frekvencia (Hz)
- ***duty_cycle*** – a kitöltés (0 – 65 535)
- ***var_freq*** – a változó frekvencia jelzése (ha True), s ez esetben korlátozott lehet a többi **PWMOut** kivezetés használata



Melyek a PWM-képes kivezetések?

- Az alábbi programmal ellenőrizhetjük, hogy mely kivezetéseket használhatjuk fel PWM kimenetként
- A **board** modulban található elemeket megpróbáljuk egyenként **PWMOut** kimenetként inicializálni, s ha ez sikerült, akkor az egy **PWM**-képes kivezetés

pwm_scan.py

```
import board
import pwmio

for pin_name in dir(board):
  pin = getattr(board, pin_name)
  try:
    p = pwmio.PWMOut(pin)
    p.deinit()
    print("PWM on:", pin_name)      # Prints the PWM-capable pins!
  except ValueError:                # Error: the pin is invalid
    print("No PWM on:", pin_name)   # Prints the invalid pins
  except RuntimeError:              # Timer conflict error
    print("Timers in use:", pin_name)  # Prints the timer conflict pins
  except TypeError:                 # Error: non-pin object
    pass                            # Passes non-pin objects
```

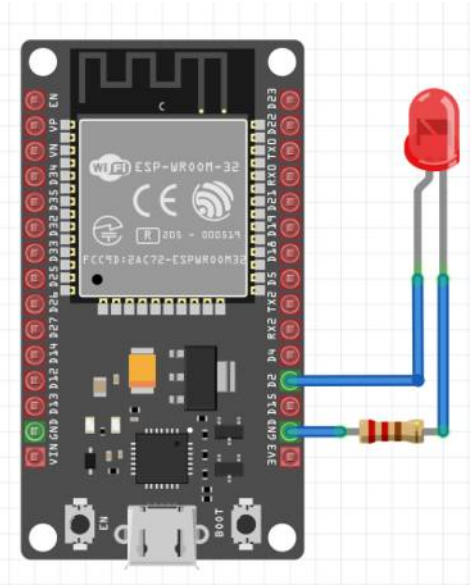
```
PWM on: D2
PWM on: D4
PWM on: D5
PWM on: D12
PWM on: D13
PWM on: D14
PWM on: D15
PWM on: D16
PWM on: D17
PWM on: D18
PWM on: D19
PWM on: D21
PWM on: D22
PWM on: D23
PWM on: D25
PWM on: D26
PWM on: D27
PWM on: D32
PWM on: D33
```

led_breathe.py

- LED fényerejének szabályozása impulzus-szélesség modulációval
- Mivel a szem érzékenysége nem lineáris, ezért a „lélegző” LED esetében a PWM kitöltés egy Gauss-függvény szerint szabályozzuk
- Leírás és Arduino mintaprogram itt található: makersportal.com/blog/2020/3/27/simple-breathing-led-in-arduino#gaussian
- A LED anódját itt az **A3** kimenetre kötöttük, a katódját pedig egy 220 Ω-os áramkorlátozó ellenálláson keresztül a **GND**-re

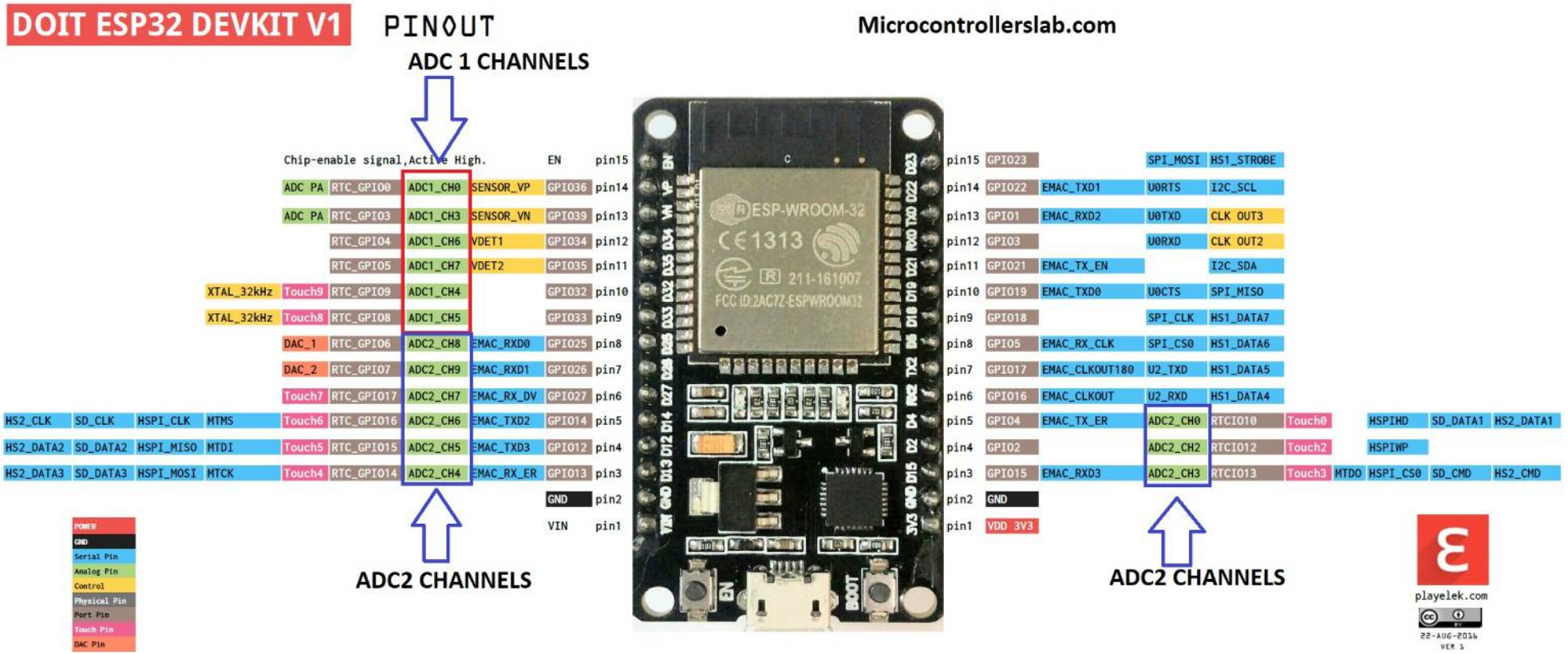
led_breathe.py

```
import time, board, pwmio
from math import exp, pow
led = pwmio.PWMOut(board.LED, frequency=5000, duty_cycle=0)
gamma = 0.14
beta = 0.5
n_pts = 500
while True:
    for i in range(n_pts):
        pwm_val =
65535*(exp(-(pow(((i/n_pts)-beta)/gamma,2.0))/2.0))
        led.duty_cycle = int(pwm_val)
        time.sleep(0.01)
```



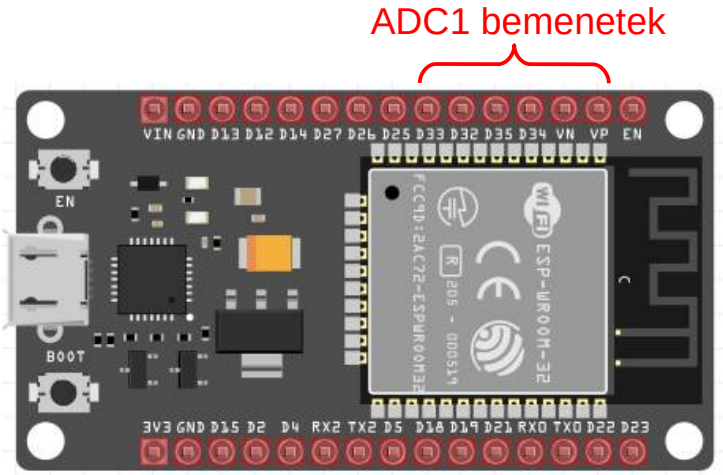
Analóg bemenetek

- Két ADC van, de nem mindegyik bemenet érhető el, s **ADC2** csak a **WiFi** letiltott állapotában használható



Analóg bemenetek kezelése

- Az analóg bemenetek kezeléséhez az **`analogio`** modult importáljuk
- A kívánt bemenethez az **`analogio.AnalogIn`** osztályt példányosítjuk
- **WiFi** használata esetén csak a **`VP, VN, D34, D35, D32, D33`** bemeneteket használhatjuk



- Például:

```
import board, analogio
analog_in = analogio.AnalogIn(board.D34)
```

- **`analog_in.value`** – a mért nyers érték (0 – 65 535)
- **`analog_in.reference_voltage`** – a névleges referencia feszültség értéke voltokban (esetünkben 3.3)

analogin_demo.py

- Vizsgáljuk egy analóg bemenet (**D34**) feszültségét! Az eredményt kiíratjuk illetve az Arduino IDE beépített **soros plotteren** is megjeleníthetjük
- A **get_voltage()** függvénnyel voltokra átszámolva kapjuk meg az eredményt

analogin_demo.py

```
import time
import board
from analogio import AnalogIn

analog_in = AnalogIn(board.D34)

def get_voltage(pin):
    return (pin.value * 3.3) / 65536

while True:
    print(get_voltage(analog_in))
    time.sleep(0.1)
```