

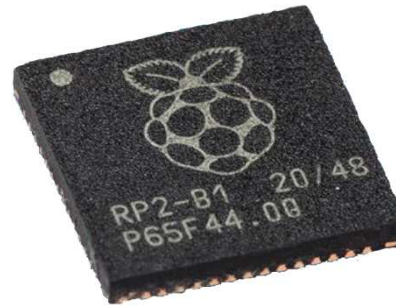
Mikrovezérlők

Raspberry Pi Pico

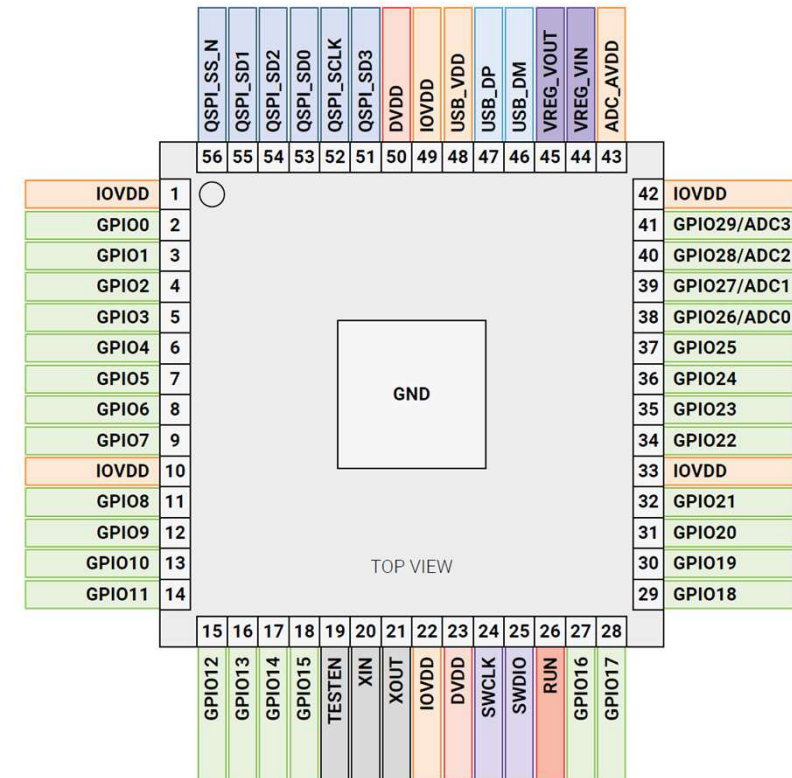
Dr. Hidvégi Timót
egyetemi docens

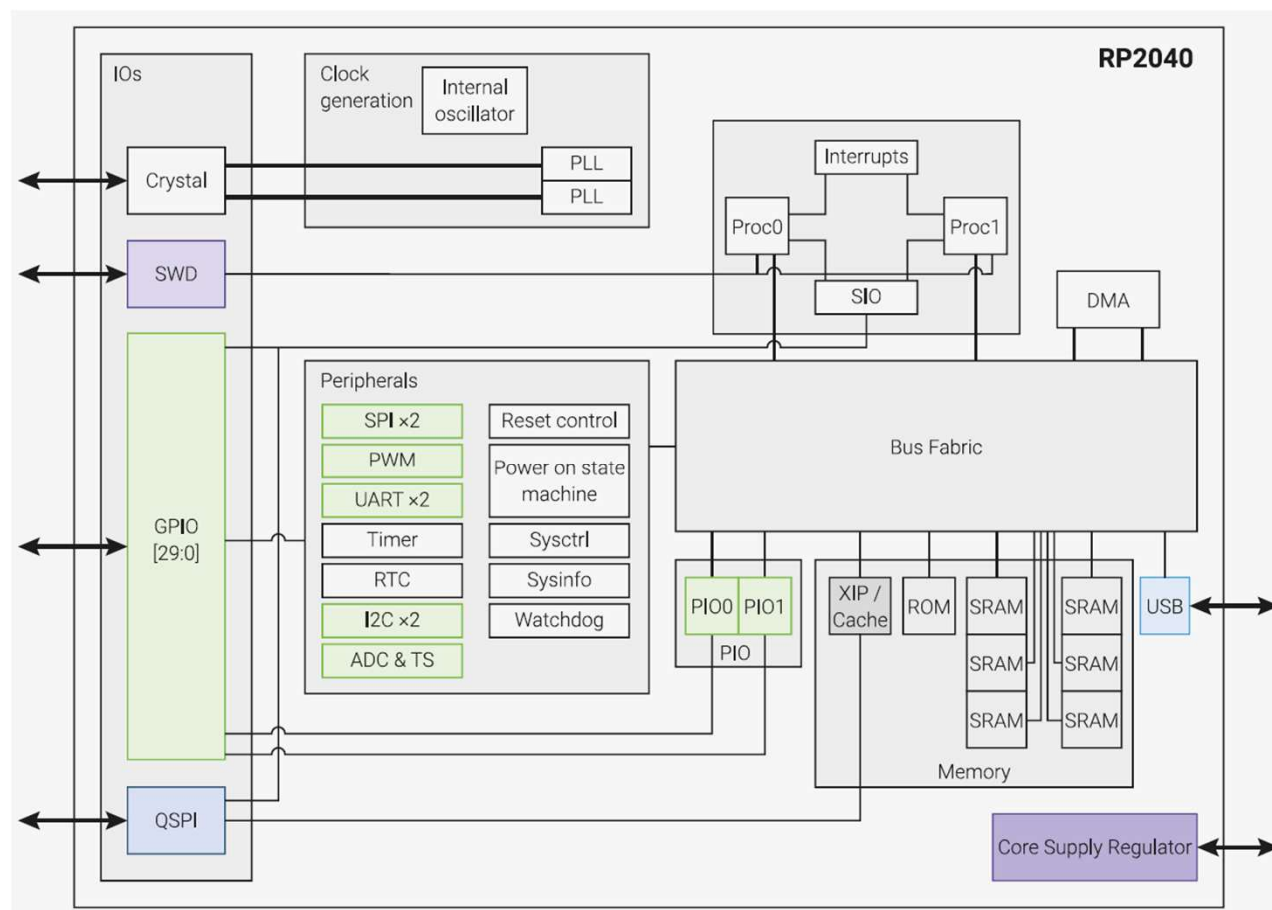
- Alapok
- Példák

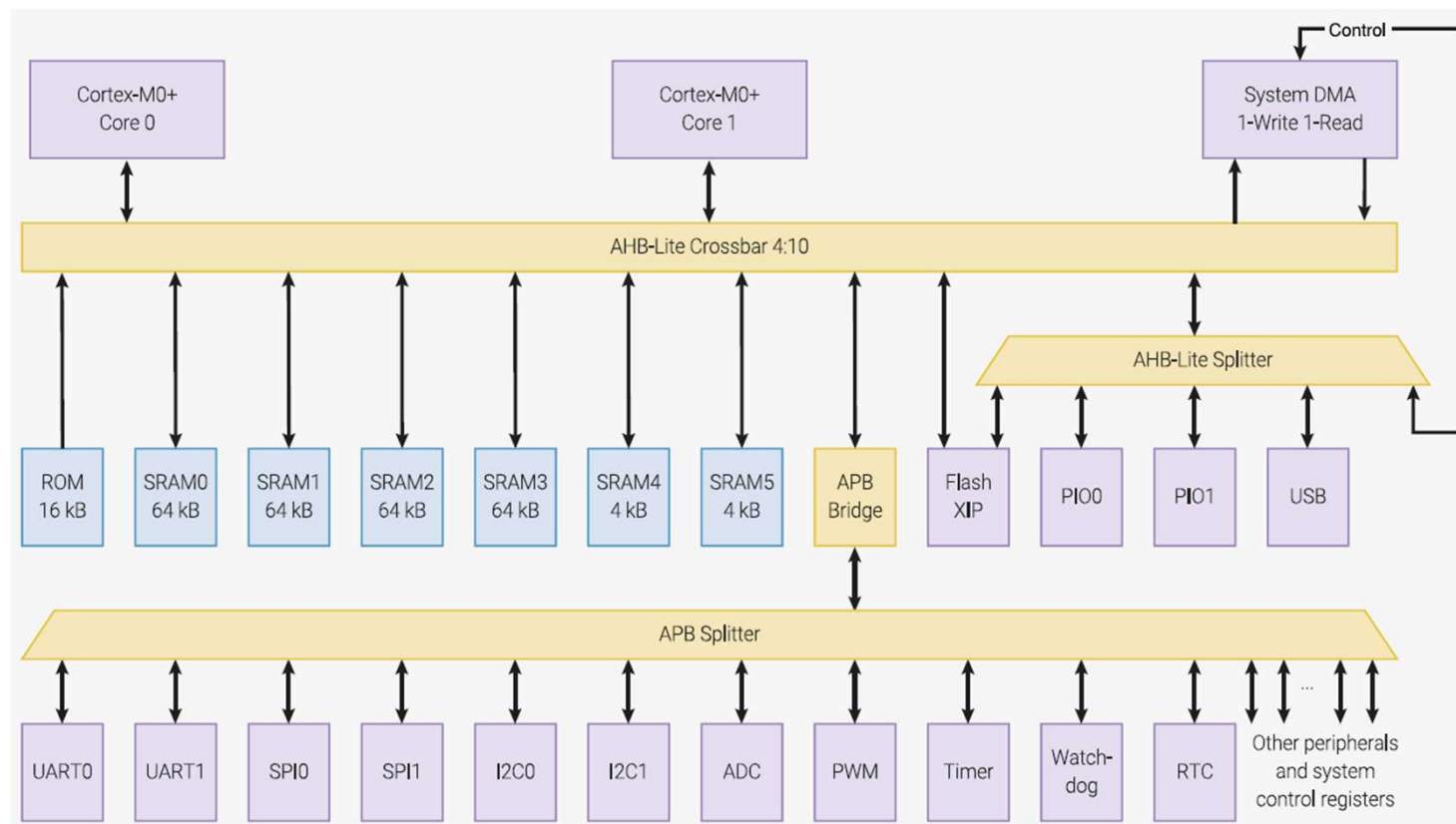
RP2040 főbb tulajdonságai



- CPU
 - Dual ARM Cortex-M0+ @ 133 MHz
- Memória
 - 264kB on-chip SRAM in six independent
- Architektúra
 - DMA controller
- Interfész
 - 30 GPIO, ebből 4 analóg bemenet is lehet
- Periféria
 - 2 × UART
 - 2 × SPI controller
 - 2 × I2C controller
 - 16 × PWM channel
 - 1 × USB



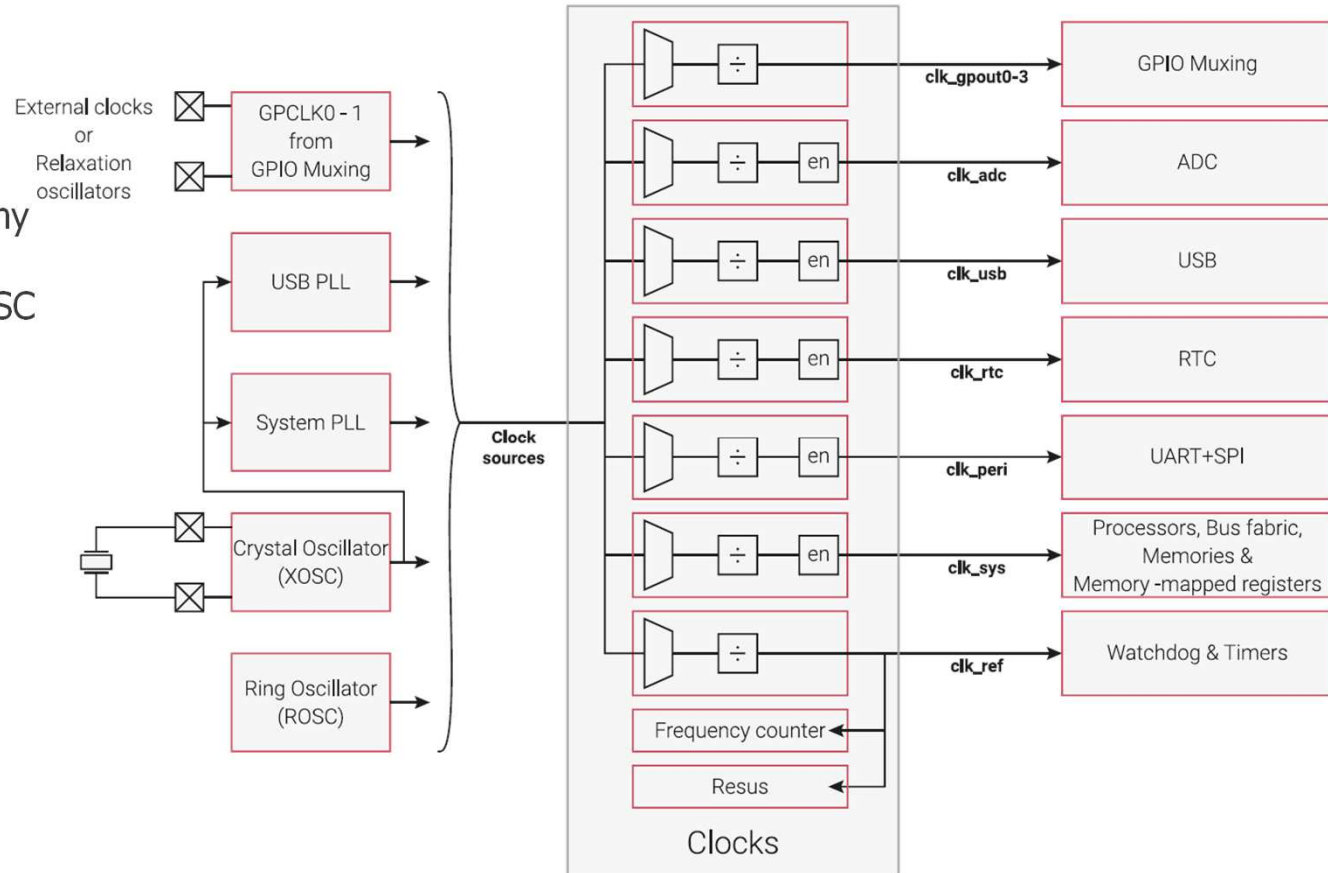




Órajel előállítás

■ ROSC

- Gyűrűs belső oszcillátor, nem pontos
- Indításkor kb 6 MHz, a teljes tartomány 1.2 MHz – 12 MHz
- Külső órajelforrás használatánál a ROSC letiltható -> energiatakarékosság



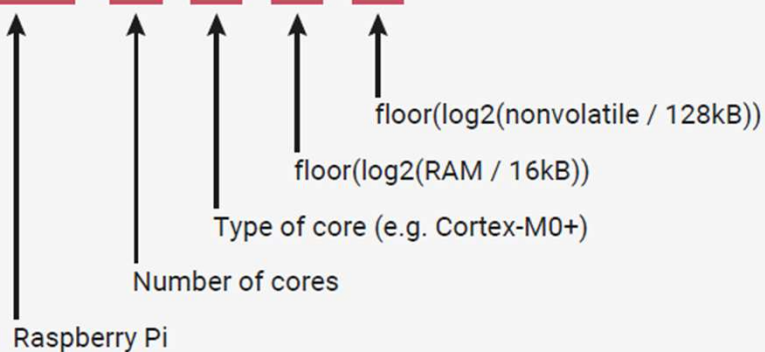
- Az RP2040 beágyazott ROM és SRAM memóriával rendelkezik.
- ROM
 - A 16 kB-os csak olvasható memória (ROM) a 0x0000000000 címtől található. A ROM tartalma a szilícium gyártásakor rögzített.
 - Tartalmazza a következőket:
 - *Kezdeti indítási kód*
 - *Flash boot szekvencia*
 - *Flash programozási rutinok*
 - *UF2 támogatással rendelkező USB eszköz*
 - *Segédprogram könyvtárak*
 - A ROM-ba való írás kísérlete nem generál hibát.

■ A chipen összesen 264 kB SRAM található.

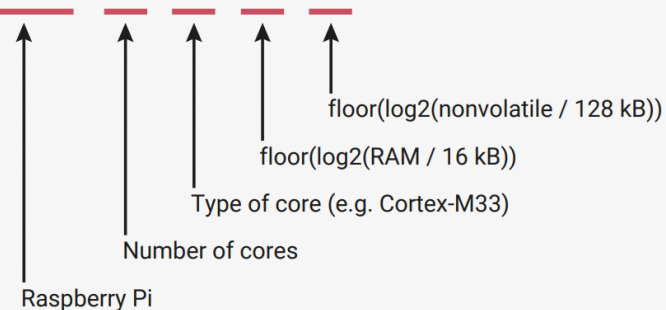
- Fizikailag ez hat bankra van felosztva, ez jelentősen javítja a memória sávszélességét több master esetén, de a szoftver egyetlen 264 kB-os memóriarégióként is kezelheti.
- Nincs korlátozás arra vonatkozóan, hogy mi tárolódik az egyes bankokban
 - *processzorkód, adatpufferek vagy ezek keveréke.*
- Négy 16k x 32 bites bank (egyenként 64kB) és két 1k x 32 bites bank (egyenként 4kB) van.
- A bankolás az SRAM fizikai particionálása, amely a teljesítményt javítja azáltal, hogy több egyidejű hozzáférést tesz lehetővé.
- Logikailag egyetlen 264kB összefüggő memória van. Minden egyes SRAM bankhoz egy dedikált AHB-Lite arbiteren keresztül lehet hozzáférni. Ez azt jelenti, hogy különböző buszmesterek párhuzamosan hozzáférhetnek különböző SRAM-bankokhoz, így minden rendszerórajel-ciklusban akár négy 32 bites SRAM-hozzáférés is történhet (mesterenként egy).
- Az SRAM a 0x20000000 címen kezdődik.

Jelölés, RP 2040, RP 2350

RP 2040



RP 2350



RP2350A

30 GPIO
7x7 QFN 60 pin

RP2350B

48 GPIO
10x10 QFN 80 pin

RP2354A

30 GPIO
7x7 QFN 60 pin
Stacked 2MB flash

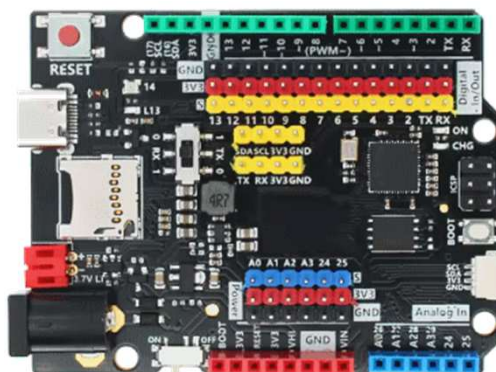
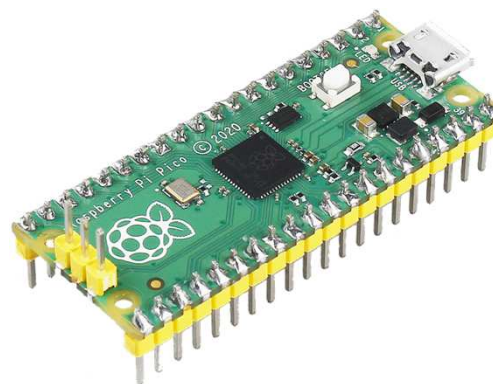
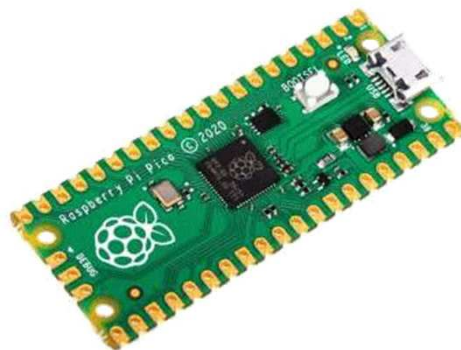
RP2354B

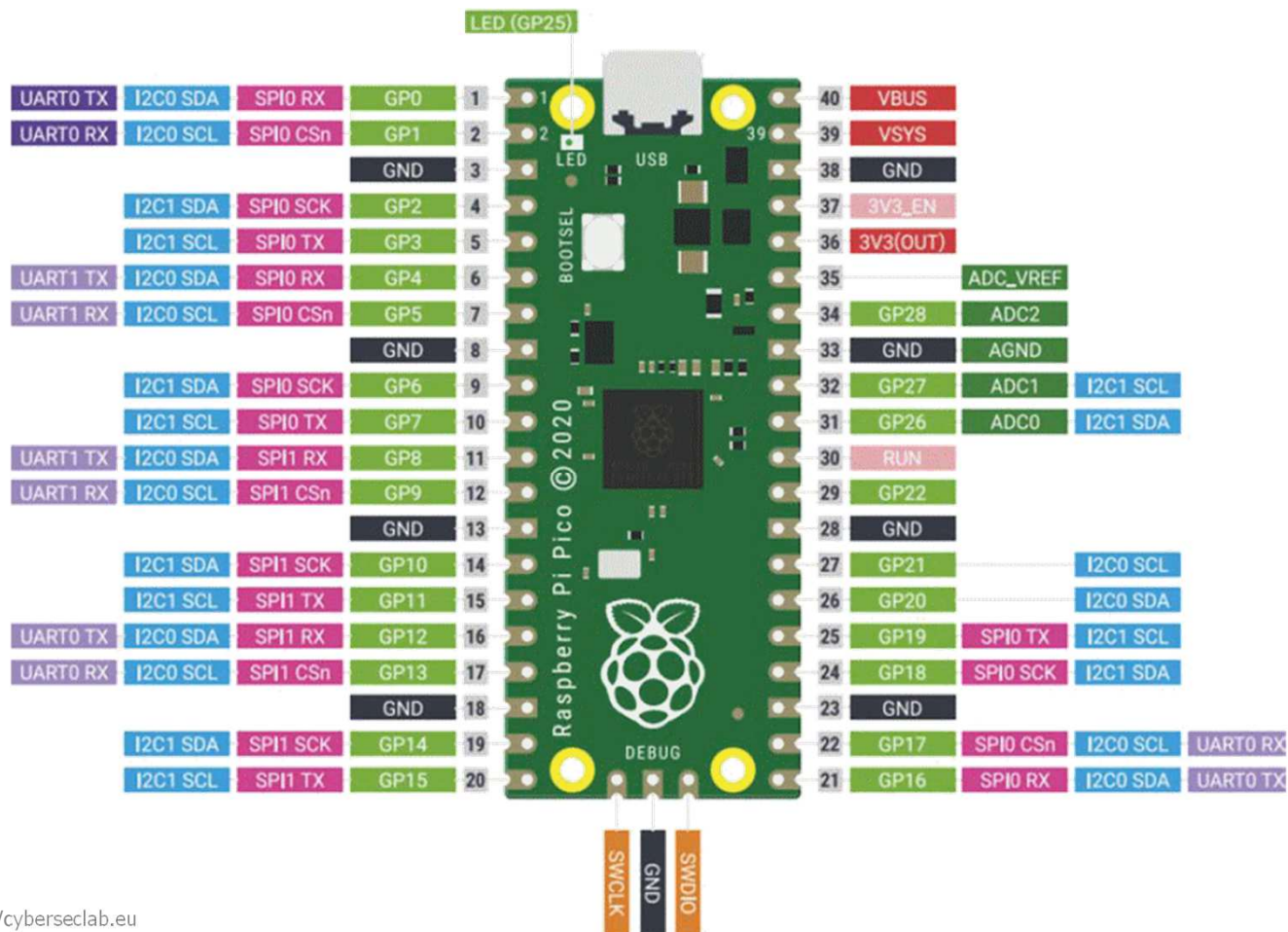
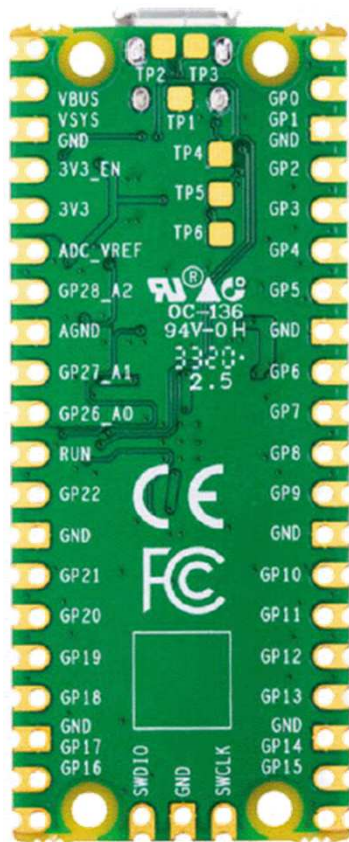
48 GPIO
10x10 QFN 80 pin
Stacked 2MB flash

- Raspberry Pi Pico a Raspberry Pi Foundation által kifejlesztett mikrokontrollerboard
 - RP2040 mikrovezérlőre épül
 - A Raspberry Pi Pico 2-n az RP2350-es eszköz található
- A Raspberry Pi Pico hasonló funkciókkal rendelkezik, mint az Arduino és az esp32-es eszközök
- A Raspberry Pi Pico jellemzően elektronikai projektekhez, IoT alkalmazásokhoz stb. használható.
- Programozásához jellemzően MicroPython alkalmazható, amely a Python lekicsinyített/egyszerűsített változata.

Raspberry Pi Pico fajtái

- Raspberry Pi Pico
 - Raspberry Pi Pico H
 - Raspberry Pi Pico W
 - Raspberry Pi Pico WH
-
- Egyéb implementációk





Összehasonlítás

	Raspberry Pi RP2040	Nordic nRF51822	Microchip SAM D21	NXP KL1x	STMicro STM32G0
Processor	Cortex-M0+	Cortex-M0	Cortex-M0+	Cortex-M0+	Cortex-M0+
Cores	2	1	1	1	1
Clock	133MHz	16MHz	48MHz	48MHz	64MHz
RAM	264KB	16 or 32KB	4 to 32 KB	4 to 32 KB	8 to 144 KB
Flash	Up to 16 MB ¹	128 or 256KB	16 to 256 KB	32 to 256 KB	16 to 512KB
PIO	Yes	-	-	-	-
GPIO	30	31 or 32	26 to 52	26 to 70 ^{5,13}	up to 94 ¹³
UART	2 + 4 ³	1	up to 6 ^{11,13}	2 to 4 ^{5,13}	up to 6 ¹³
I2C	2 + 8 ⁴	2 ⁸	up to 6 ^{11,13}	up to 3 ¹³	up to 3 ¹³
I2S	8 ²	-	1 ⁵	1 ⁵	up to 2 ¹³
SPI	2 + 8 ²	2 + 1 ⁹	up to 6 ^{11,13}	up to 3 ¹³	up to 3 ¹³
PWM	16 + 8 ²	via Timers	via Timers	up to 11 ¹³	via Timers
ADC	4 channels	8 channels	up to 20 channels ¹³	up to 20 channels ¹³	up to 16 channels ¹³
	12-bit	10-bit	12-bit	12-bit or 16-bit ⁵	12-bit
RTC ¹⁰	Yes	Yes	Yes	Yes	Yes
USB	Host + Device ⁶	-	Host + Device ^{5,7}	-	Host + Device ⁷

- „C”, C++ nyelv

- Python

- Thonny

- *Egyszerű és felhasználóbarát Python-szerkesztő, működik Windows, macOS és Linux platformokon. Beépített támogatással rendelkezik a Raspberry Pi Pico hardver/MicroPython firmware számára is.*

- <https://thonny.org>

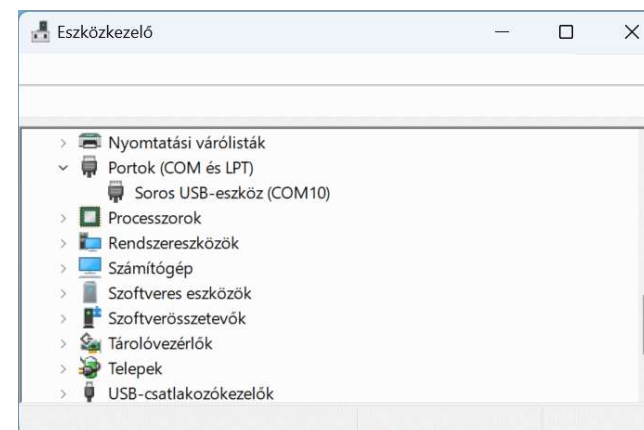
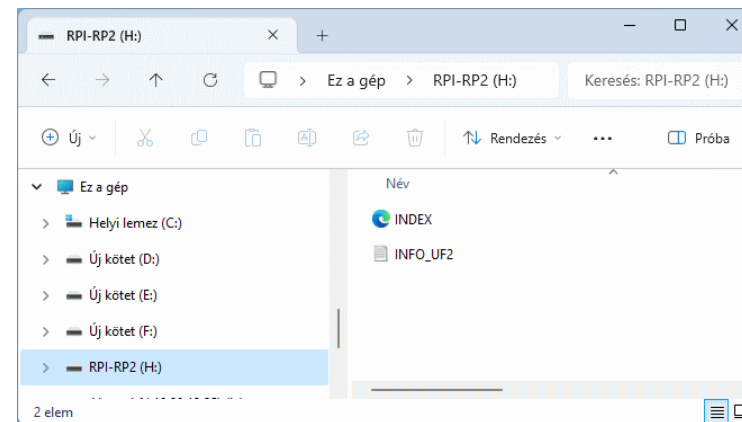
- Assembly

Operation	Description	Assembler	Cycles
Move	8-bit immediate	MOVS Rd, #<imm>	1
	Lo to Lo	MOVS Rd, Rm	1
	Any to Any	MOV Rd, Rm	1
	Any to PC	MOV PC, Rm	2
Add	3-bit immediate	ADDS Rd, Rn, #<imm>	1
	All registers Lo	ADDS Rd, Rn, Rm	1
	Any to Any	ADD Rd, Rd, Rm	1
	Any to PC	ADD PC, PC, Rm	2
	8-bit immediate	ADDS Rd, Rd, #<imm>	1
	With carry	ADCS Rd, Rd, Rm	1

- A MicroPython a Python „lekicsinyített” változata
- Tipikusan a mikrokontrollerekhez és a korlátozott rendszerekhez használják.
- Firmware
 - https://micropython.org/download/RPI_PICO/
 - <https://www.raspberrypi.com/documentation/microcontrollers/?version=E0C9125B0D9B>
- Könyvtárak
 - <https://docs.micropython.org/en/v1.16/library/>
 - machine könyvtár
 - class Pin – control I/O pins
 - class Signal – control and sense external I/O devices
 - class ADC – analog to digital conversion
 - class PWM – pulse width modulation
 - class UART – duplex serial communication bus
 - class SPI – a Serial Peripheral Interface bus protocol (master side)
 - class I2C – a two-wire serial protocol
 - class RTC – real time clock
 - class Timer – control hardware timers
 - class WDT – watchdog timer
 - class SD – secure digital memory card (cc3200 port only)
 - class SDCard – secure digital memory card

■ Telepítés menete

- Bootsel-t megnyomni
- Csatlakoztatás a laptophoz
- Bootsel elengedése
- Index.html-re kattintani -> navigálás
(<https://www.raspberrypi.com/documentation/microcontrollers/?version=E0C9125B0D9B>)
 - *Letöltés: RPI_PICO_W-20241129-v1.24.1.uf2*
- Átmásolni a letöltött uf2 file-t
- Thonny jobb alsó sarkában kiválasztani a MicroPython-t



MicroPython

Getting started with MicroPython

Download the correct MicroPython UF2 file for your board:

- Pico
- Pico W
- Pico 2
- Pico 2 W

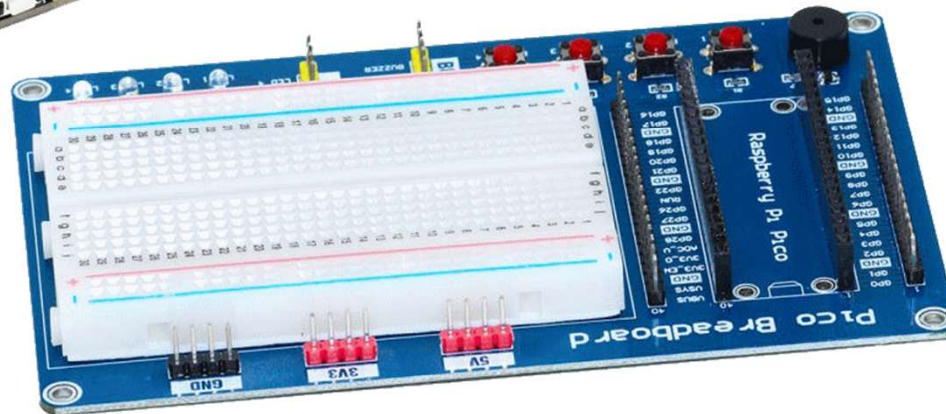


**SZÉCHENYI
EGYETEM**
UNIVERSITY OF GYŐR
GÉPESZMÉRNÖKI, INFORMATIKAI
ÉS VILLAMOSMÉRNÖKI KAR

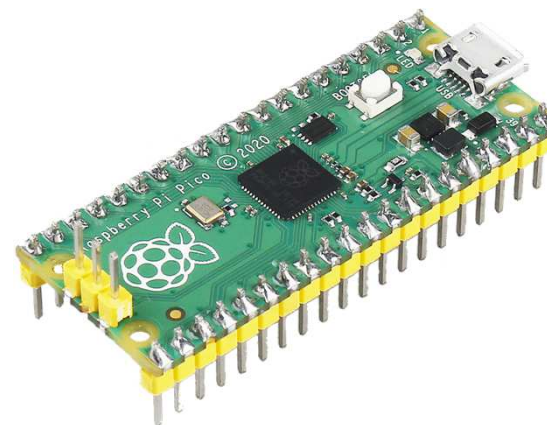
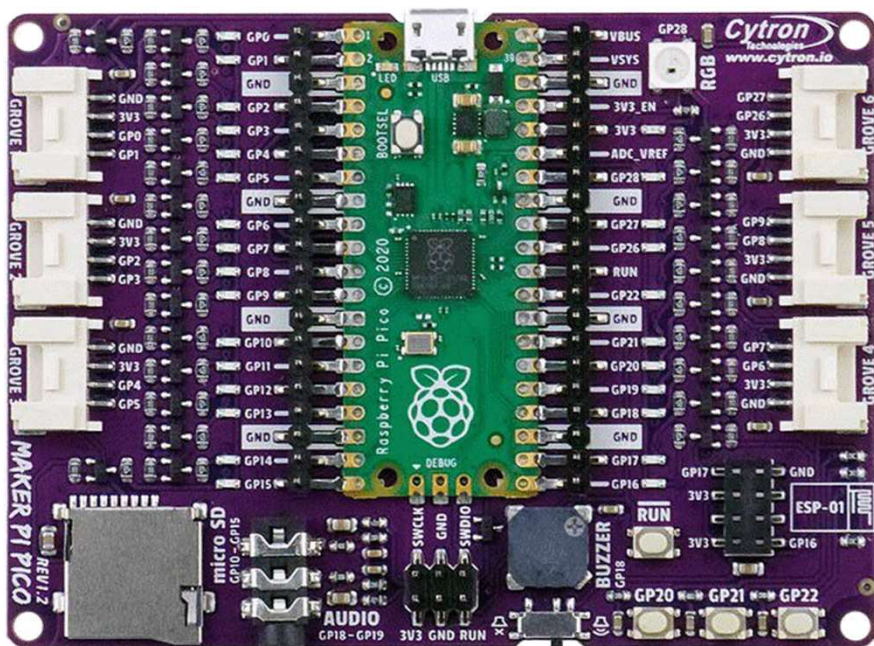


CYBERSECLAB

Fejlesztőpanelek



A felhasznált tesztpanel



<https://malnipc.hu/raspberry-pi-pico-wh>

<https://malnipc.hu/maker-pi-pico-base-pico-panel-nelkul> (a képen a fejlesztőpanel tartalmaz Pico board-ot)

Mi az a ThingSpeak? Vagy inkább saját felhő?

- Egy IoT-felhőplatform, ahol az érzékelők adatai a felhőbe küldhetők.
 - A regisztrációhoz új MathWorks-fiókot kell létrehozni (vagy a meglévőbe bejelentkezni).
 - A ThingSpeak szolgáltatást a MathWorks üzemelteti.
 - A ThingSpeak ingyenes a kisebb, nem kereskedelmi projektek számára.
 - A ThingSpeak tartalmaz egy webes szolgáltatást (REST API).

teszt

Channel ID: 1904080

Author: mwa00

Access: Private

Private View

Public View

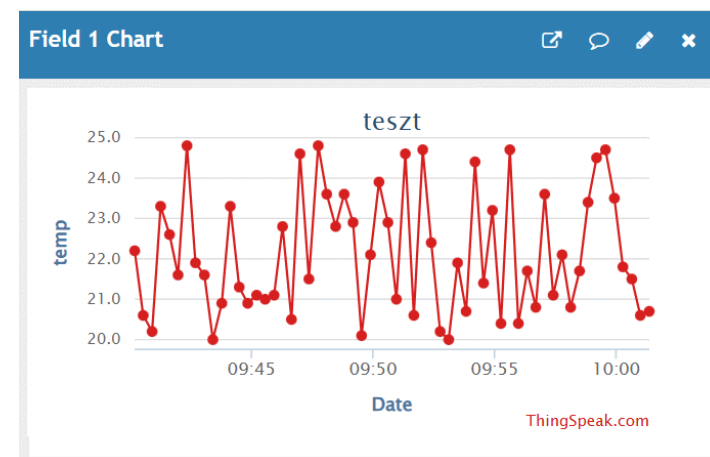
Channel Settings

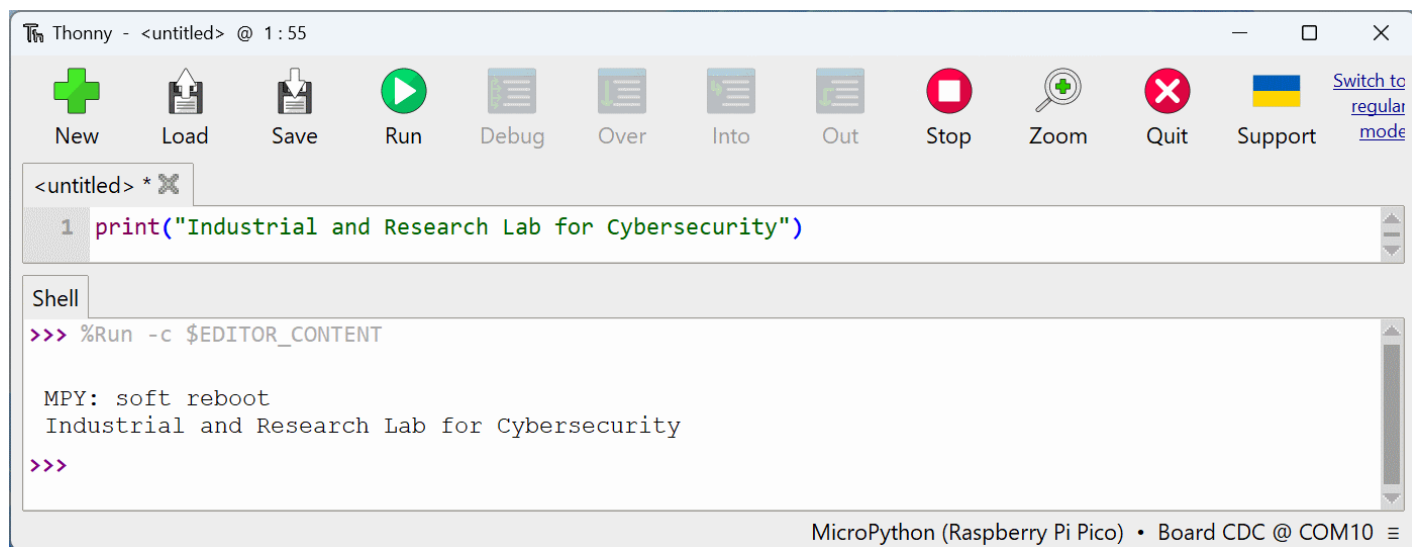
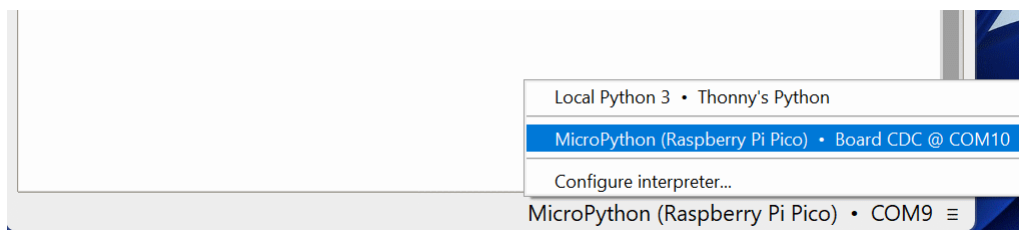
Sharing

API Keys

Data Import / Export

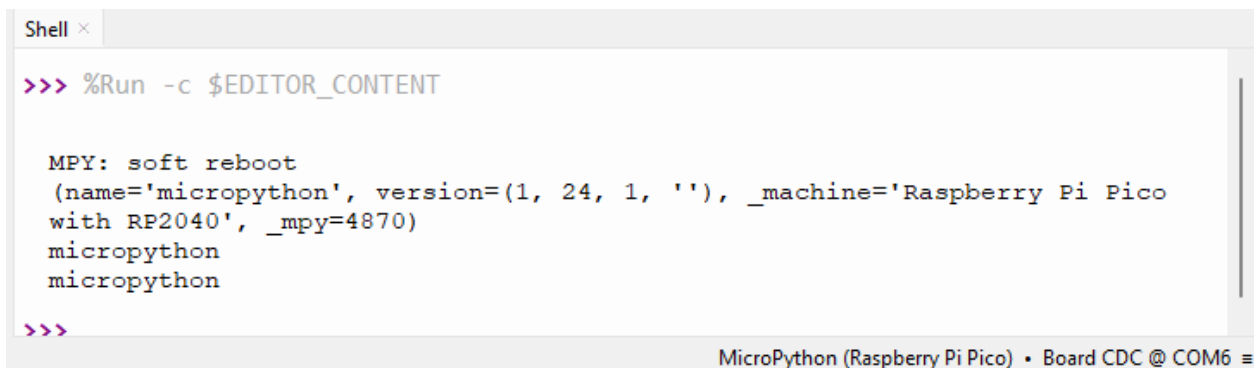
- Saját VPS állandó IP címmel
 - Egyedi fejlesztés





■ Eszközinformációk kiolvasása

```
import sys
print(sys.implementation)
print(sys.implementation.name)
print(sys.implementation[0])
```



```
Shell x
>>> %Run -c $EDITOR_CONTENT

MPY: soft reboot
(name='micropython', version=(1, 24, 1, ''), _machine='Raspberry Pi Pico
with RP2040', _mpy=4870)
micropython
micropython
>>>
```

MicroPython (Raspberry Pi Pico) • Board CDC @ COM6

Példák (portkezelés)

```
import machine  
import time
```

```
pin = 8  
led = machine.Pin(pin, machine.Pin.OUT)
```

```
while True:  
    led.value(1)  
    time.sleep(2)  
    led.value(0)  
    time.sleep(2)
```

```
import machine  
import time
```

```
pin = 7  
led = machine.Pin(pin, machine.Pin.OUT)
```

```
while True:  
    led.toggle()  
    time.sleep(1)
```

<https://docs.micropython.org/en/latest/library/machine.html>

<https://docs.micropython.org/en/latest/library/time.html>

<https://docs.micropython.org/en/latest/library/machine.Pin.html>

■ machine.freq

- Visszaadja a CPU frekvenciáját Hz-ben
- `machine.freq(MCU_frequency[, peripheral_frequency=48_000_000])`

```
import machine
```

```
print("CPU frekvencia:" + str(machine.freq()))  
machine.freq(240000000)
```


```
# CPU és az UART frekvenciája  
machine.freq(125000000, 125000000)  
print("CPU frekvencia:" + str(machine.freq()))
```

<https://docs.micropython.org/en/v1.16/library/machine.html>

Példák (adatbevitel)

```
import machine
import time
pin = 20
button = machine.Pin(pin, machine.Pin.IN, machine.Pin.PULL_UP)
while True:
    if button.value() == 1:
        print("Nyomd meg a gombot")
    else:
        print("A nyomógomb lenyomva")
    time.sleep(1)
```

```
if not button.value():
    print("A nyomógomb lenyomva ")
else:
    print("Nyomd meg a gombot")
time.sleep(1)
```



```
Shell
>>> %Run -c $EDITOR_CONTENT

MPY: soft reboot
Nyomd meg a gombot
Nyomd meg a gombot
Nyomd meg a gombot
Nyomd meg a gombot
A nyomógomb lenyomva
A nyomógomb lenyomva
A nyomógomb lenyomva
Nyomd meg a gombot
Nyomd meg a gombot

Traceback (most recent call last):
  File "<stdin>", line 10, in <module>
KeyboardInterrupt:
```


- A hőmérséklet-érzékelő egy előfeszített dióda, amely a V_{be} feszültséget méri.
 - AINSEL=4 csatorna
 - Jellemzően $V_{be} = 0,706V$ 27 C fokon, $-1,721mV/fok$ meredekséggel
 - $T = 27 - (ADC_voltage - 0.706)/0.001721$
 - Mivel a V_{be} és a V_{be} meredeksége a hőmérséklet-tartományban és eszközönként változhat, pontos méréshez kalibrálni kell

- A MicroPython jelenlegi verziója a Pi Pico számára nem teszi lehetővé a hardveres időzítők külön-külön történő használatát.
 - Ehelyett szinte korlátlan számú „szoftveres” időzítő létrehozására van lehetőség, amelyek mindegyike egyetlen hardveres időzítőre támaszkodik.

■ Projektváz

```
from machine import Timer
```

```
def interruptionHandler(timer):  
    ...
```

```
if __name__ == "__main__":  
    soft_timer = Timer(mode=Timer.PERIODIC, period=1000, callback=interruptionHandler)
```

```
mode=Timer., period=1000,  
deinit(  
init(  
mro(  
ONE_SHOT  
" + str(couPERIODIC
```

```
from machine import Pin, Timer
```

```
pin = 6
```

```
led = Pin(pin, Pin.OUT)
```

```
timer = Timer()
```

```
def blink(timer):
```

```
    led.toggle()
```

```
timer.init(freq=1, mode=Timer.PERIODIC, callback=blink)
```

- A rendszer újraindítása a feladata, ha az alkalmazás összeomlik.
- Elindulása után nem lehet leállítani vagy átkonfigurálni. Az engedélyezés után az alkalmazásnak rendszeresen törölni (feed()) kell a watchdogot, hogy megakadályozza a rendszer visszaállítását.
- Alkalmazás
 - WDT objektum létrehozása és elindítása.
 - Az időkorlátot milliszekundumban kell megadni.
 - Ha egyszer már fut, az időkorlát nem módosítható és a WDT-t sem lehet leállítani.
 - Az rp2040 eszközökön a maximális időkorlát 8388 ms.

```
from machine import WDT  
import time
```

```
print("WDT indul")  
wdt = WDT(timeout=5000)  
wdt.feed()  
print("WDT beállításra került")
```

```
while True:  
    print("időzítés")  
    time.sleep(1)
```

<https://docs.micropython.org/en/latest/library/machine.WDT.html>

- `RTC.datetime(év,hónap,nap, ,óra,
perc, másodperc, ,)`

```
from machine import RTC  
import time
```

```
rtc = RTC()  
rtc.datetime((2025, 4, 11, 0, 1, 30, 0, 0))  
print(rtc.datetime())
```

```
now = time.localtime()  
print(now)
```

```
print("Date: {}/{}{}".format(now[1], now[2], now[0]))  
print("Time: {}:{}".format(now[3], now[4]))
```

<https://docs.micropython.org/en/v1.16/library/machine.RTC.html>

```
from time import sleep
import _thread
```

```
# mind a két függvényben végtelen ciklus található
def core0Thread():
    szam0 = 0
    while True:
        szam0 += 2
        sleep(2)
        print("Core0: " + str(szam0))
```

```
def core1Thread():
    szam1 = 1
    while True:
        szam1 += 2
        sleep(2)
        print("Core1: " + str(szam1))
```

```
second_thread = _thread.start_new_thread(core1Thread, ())
core0Thread()
```

- Két processzor van az RP 2040 mikrovezérlőben
- A két függvényben egy-egy végtelen ciklus van
- Könyvtár
 - <https://docs.python.org/3/library/thread.html>

▪ „network” csomag alkalmazása

- Access Point (AP)
 - *Működik a Pico AP-ként, de Internethez nem tud csatlakozni.*
- Station (STA)
 - *Állomásként tud az Internethez csatlakozni egy routeren keresztül.*

```
import network
```

```
staIf = network.WLAN(network.STA_IF)  
print(staIf.active())
```

```
apIf = network.WLAN(network.AP_IF)  
print(apIf.active())
```

```
import network  
import requests
```

```
# Wi-Fi credentials  
ssid = 'XXXXXX'  
password = 'XXXXX'
```

```
# Connect to network  
wlan = network.WLAN(network.STA_IF)  
wlan.active(True)
```

```
wlan.connect(ssid, password)
```

<https://docs.micropython.org/en/v1.16/library/network.WLAN.html>

WiFi példa, HTTP kérés - válasz

```
import network
import requests

# Wi-Fi credentials
ssid = 'XXXXXX'
password = 'XXXXXX'

# Connect to network
wlan = network.WLAN(network.STA_IF)
wlan.active(True)

wlan.connect(ssid, password)

# GET request
response = requests.get("https://cyberseclab.eu")
responseCode = response.status_code
responseContent = response.content

print('Response HTTP code: ', responseCode)
print('Response content:', responseContent)
```

```
MPY: soft reboot
Response HTTP code: 200
Response content: b'\r\n\r\n\r\n<!doctype html>\r\n<html lang="zxx">\r ...
\r\n</h1>\r\n<div class="banner-btn wow animate__animated animate__fa ...
s 2023-ban hivatalosan is megalakult a G&eacute;p&eacute;szm&eacute;r&eacute;n&eacute;kar ...'
```


- A megszakításkérés egy sürgős beavatkozáskérés, amelyet a processzornak küldenek azért, hogy egy adott feladatot azonnal prioritásként kezeljen és hajtson végre. Ez megállítja azt a programrészletet, amelyet a processzor végrehajt.
- A megszakításkéréseket különböző események - például külső események (él-szintváltozások) – vagy belső perifériák okozhatják. Segítségükkel olyan feladatokat lehet végrehajtani, amelyek nem részei a főprogramnak, és a program többi részének a futásával egyidejűleg végezhetők (aszinkron végrehajtás).

■ Függvények

- `irq.init()` : A megszakítás újbóli inicializálása. Automatikusan újra aktiválódik.
- `irq.enable()` : A megszakítás engedélyezése.
- `irq.disable()` : A megszakítás letiltása.
- `irq()` : Kézzel indítja a megszakítási rutin hívását.
- `irq.flags()` a megszakítást kiváltó esemény típusának megismeréséhez. Csak az isr-ben használható.

■ Típusok

- Amikor a jel 0 V, a `Pin.IRQ_LOW_LEVEL` megszakítás lép működésbe.
- Amikor a jel 3.3 V, a `Pin.IRQ_HIGH_LEVEL` megszakítás lép működésbe.
- Amikor egy bemeneti jel LOW-ról (0 V) HIGH-ra (3,3 V) változik, a `Pin.IRQ_RISING` megszakítás aktiválódik.
- Amikor egy bemeneti jel HIGH-ról (3.3 V) LOW-ra (0 V) változik, a `Pin.IRQ_Falling` megszakítás aktiválódik.

<https://docs.micropython.org/en/latest/library/machine.Pin.html#machine.Pin.irq>

■ Megszakításkezelés váza

```
from machine import Pin

pinButton = Pin(20, mode=Pin.IN, pull=Pin.PULL_UP)

def interruptionHandler(pin):
    ...

pinButton.irq(trigger=Pin.IRQ_FALLING, handler=interruptionHandler)

while True:
    ...
```

- Adatlap

- <https://datasheets.raspberrypi.com/rp2040/rp2040-datasheet.pdf>

- MicroPython

- <https://datasheets.raspberrypi.com/pico/raspberry-pi-pico-python-sdk.pdf>

- Példák

- https://github.com/cyberseclabor/Mikrovezerlo/tree/main/RaspberryPiPICO/Eloadas_01
- https://webelektronika.com/tagcloud/rpi_pico

- Web
 - <https://cyberseclab.eu>
- Facebook
 - <https://www.facebook.com/IndustrialandResearchLab>
- Github
 - <https://github.com/cyberseclabor>
- LinkedIn
 - <https://www.linkedin.com/company/industrial-and-research-lab-for-cybersecurity>

enumeration ISO21434 MiTM
Artificial_Intelligence network
hacking education OT/ICS Android
car spoofing S7 forensics CyberSecLab
NIST800-82 training Purdue vehicle
HMI modell opc-ua PLC
OWASP pentest security NIS2 CAN
cyber Python C# OSINT
WiFi exploit linux AI OT nmap unit
scada sniffing kali online
modbus malware ethical
SDR Machine_Learning metasploit
vulnerability head Pentesting
Ethernet-IP