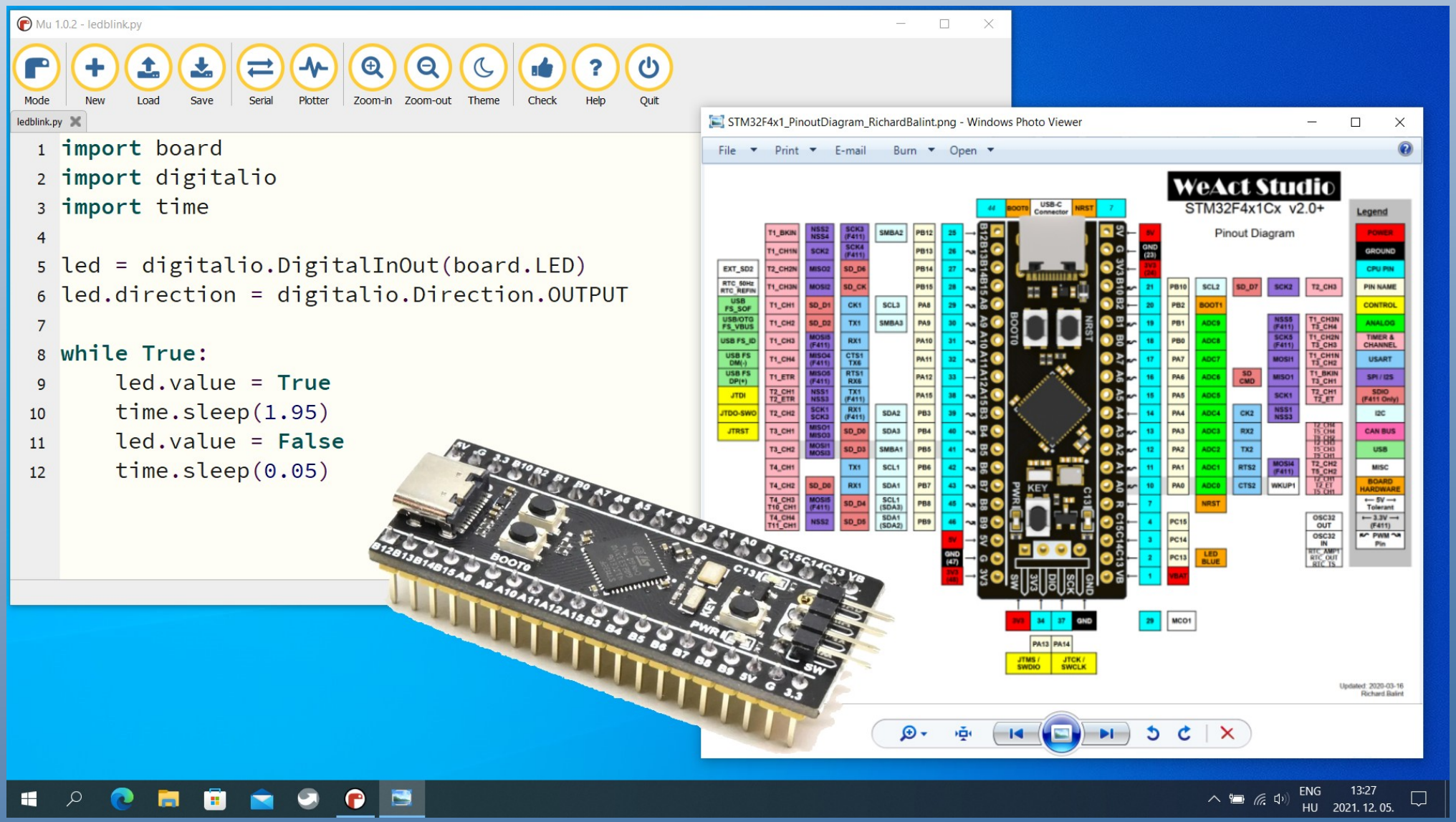
CircuitPython tanfolyam



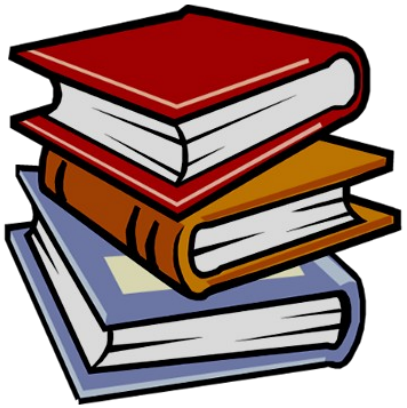
1. Az első lépések

Felhasznált és ajánlott irodalom

Python:

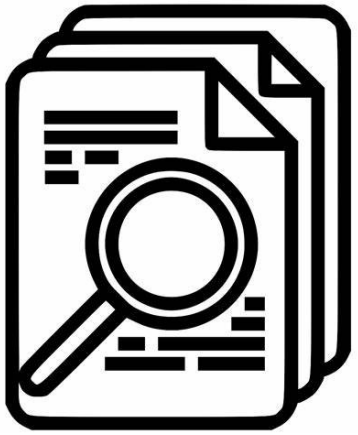
- Mark Pilgrim/Kelemen Gábor: Ugorj fejest a Python 3-ba!

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

Adatlapok és dokumentáció:



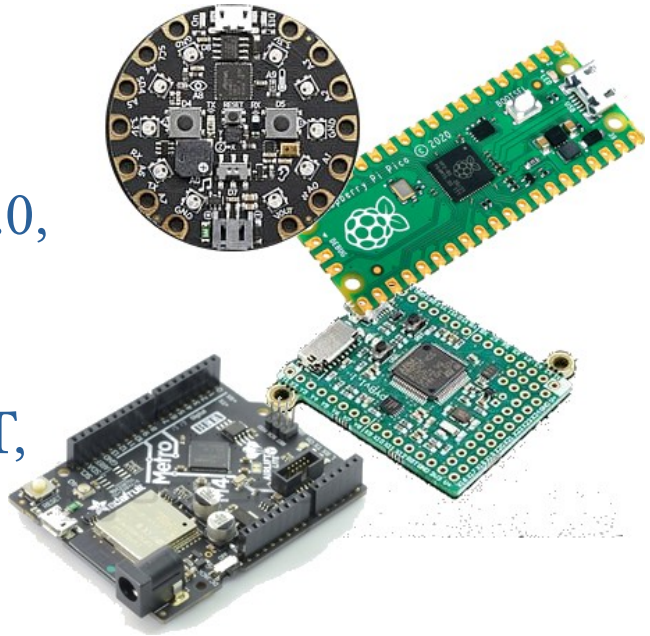
- **STM32F411CE** adatlap és termékinfo
- **STM32F411xC/E** Family Reference Manual
- **WeAct Studio:** STM32F4x1 MiniF4

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: 254
- Csak példa gyanánt néhány ismertebb típus: Raspberry Pi Pico, Circuit Playgroung Express, Feather M4 express, MagTag 2.9” WiFi, Teensy 4.0, 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…

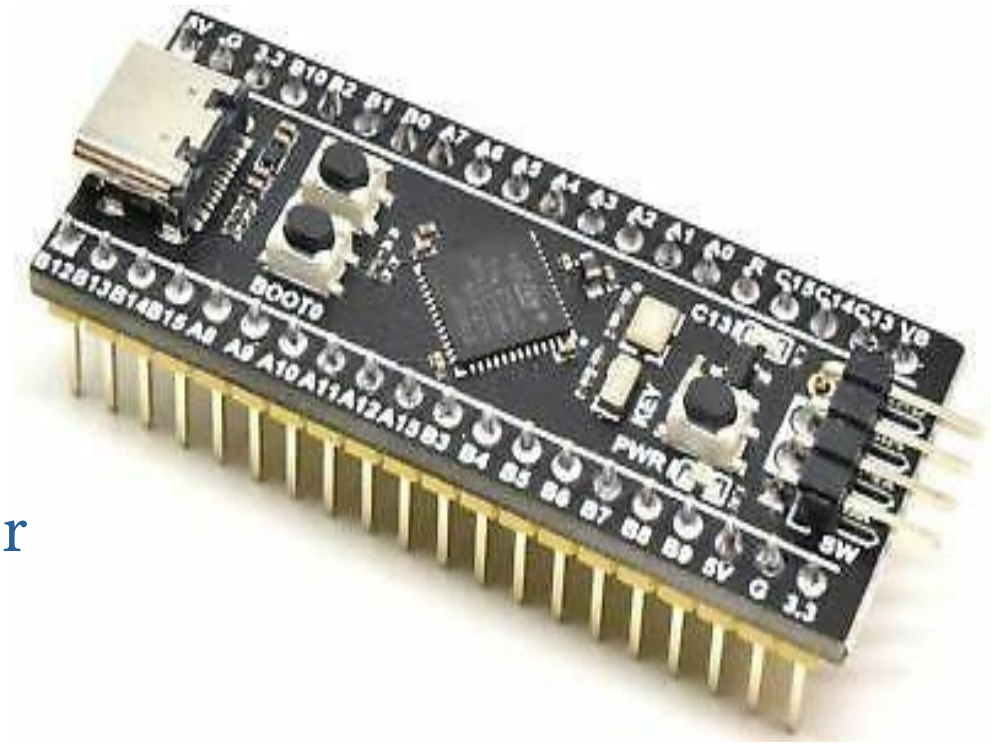


- A CircuitPython aktuális változatának letöltése innen: https://circuitpython.org/downloads (ki kell választani a kártyát és rákattintás után a felbukkanó lapon a jobb felső sarokban elérhető a letöltési link)
- Mi a WeAct Studio **STM32F411CE blackpill with flash** kártyáját használjuk, a CircuitPython 7.0.0 változatával (adafruit-circuitpython-stm32f411ce_blackpill_with_flash-en_US-7.0.0.bin)

STM32F411CE Black Pill (WeAct Studio)

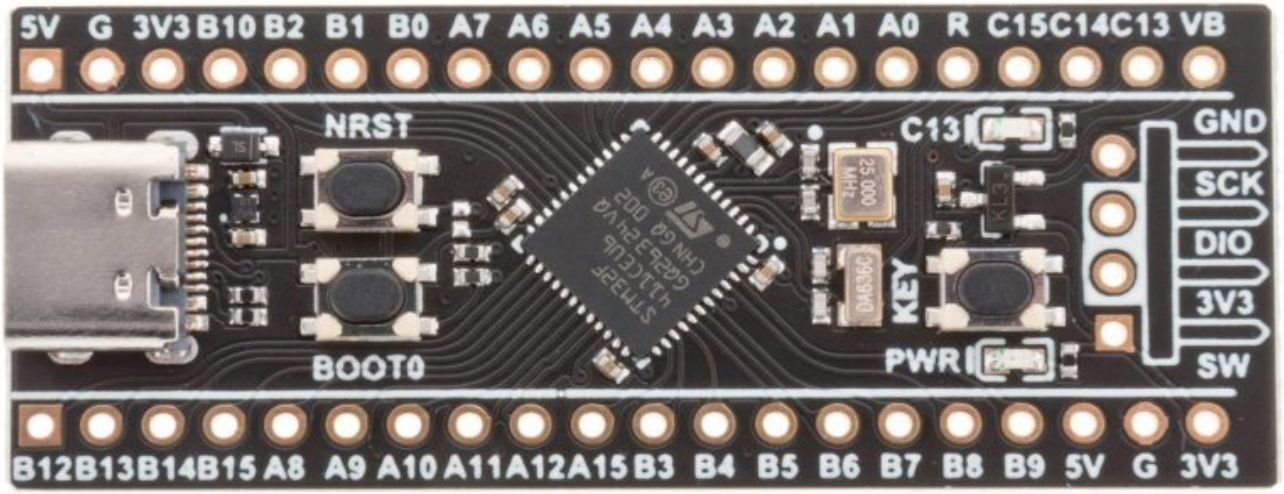
- STM32F411CEU6 100Mhz,

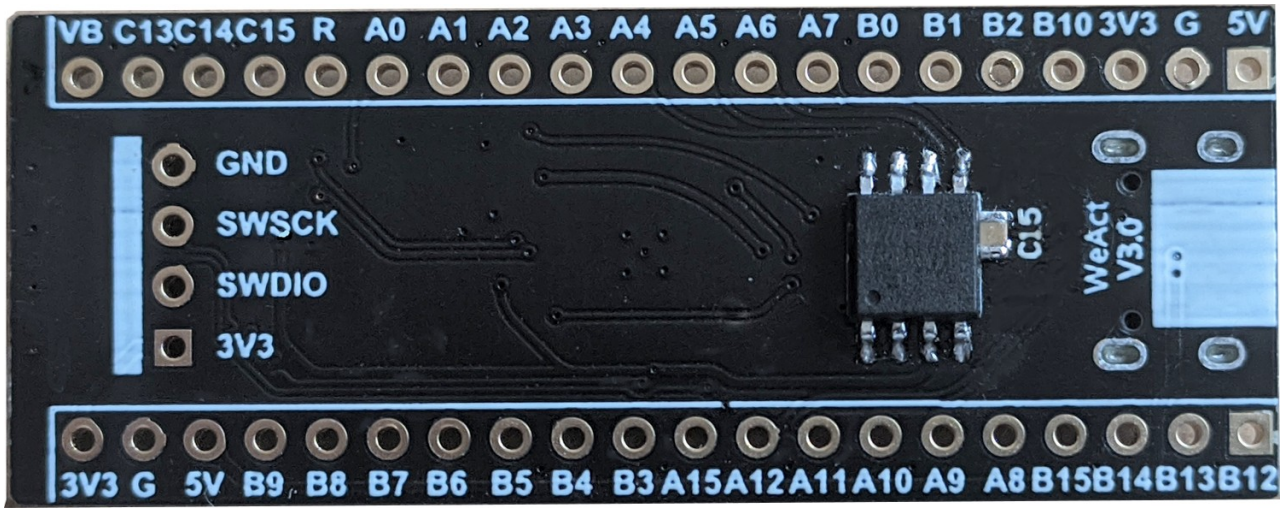
  128KB RAM, 512KB ROM
- 25MHZ HS crystal
  32.768khz LS crystal
- SPI Flash memóriával bővíthető
- 3 nyomógomb: Reset, Boot0, User
- DFU beépített bootloader
- CMSIS-DAP firmware
- Fejlesztői támogatás: C, Arduino,
  MicroPython, **CircuitPython,** ARM Mbed (nemhiv)
- USB Type-C csatlakozó, VDD = 3,3 V
- **Megjegyzés:** Flash bővítés nélkül nagyon korlátozott a Python használhatósága

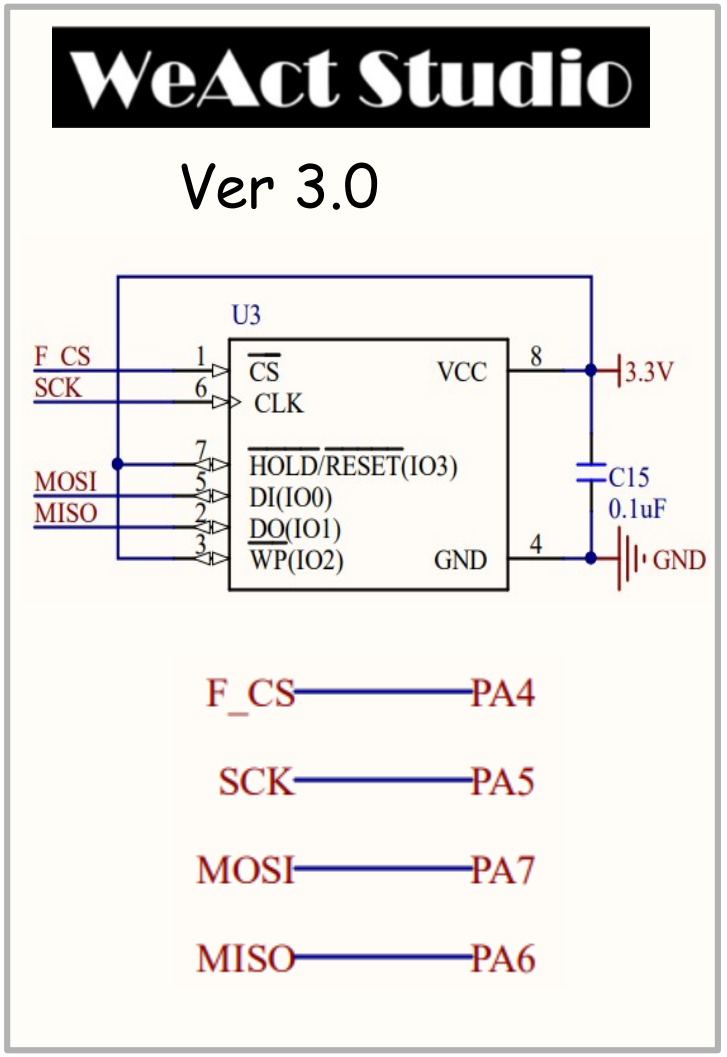


STM32F411CE Black Pill with Flash

- Flash memória: **W25Q64JVSIQ** 64 megabit, azaz 8 MB a Python forráskódok és programkönyvtárak tárolására
- A memória mellé egy 100 nF kondenzátort (C15) is kell forrasztani



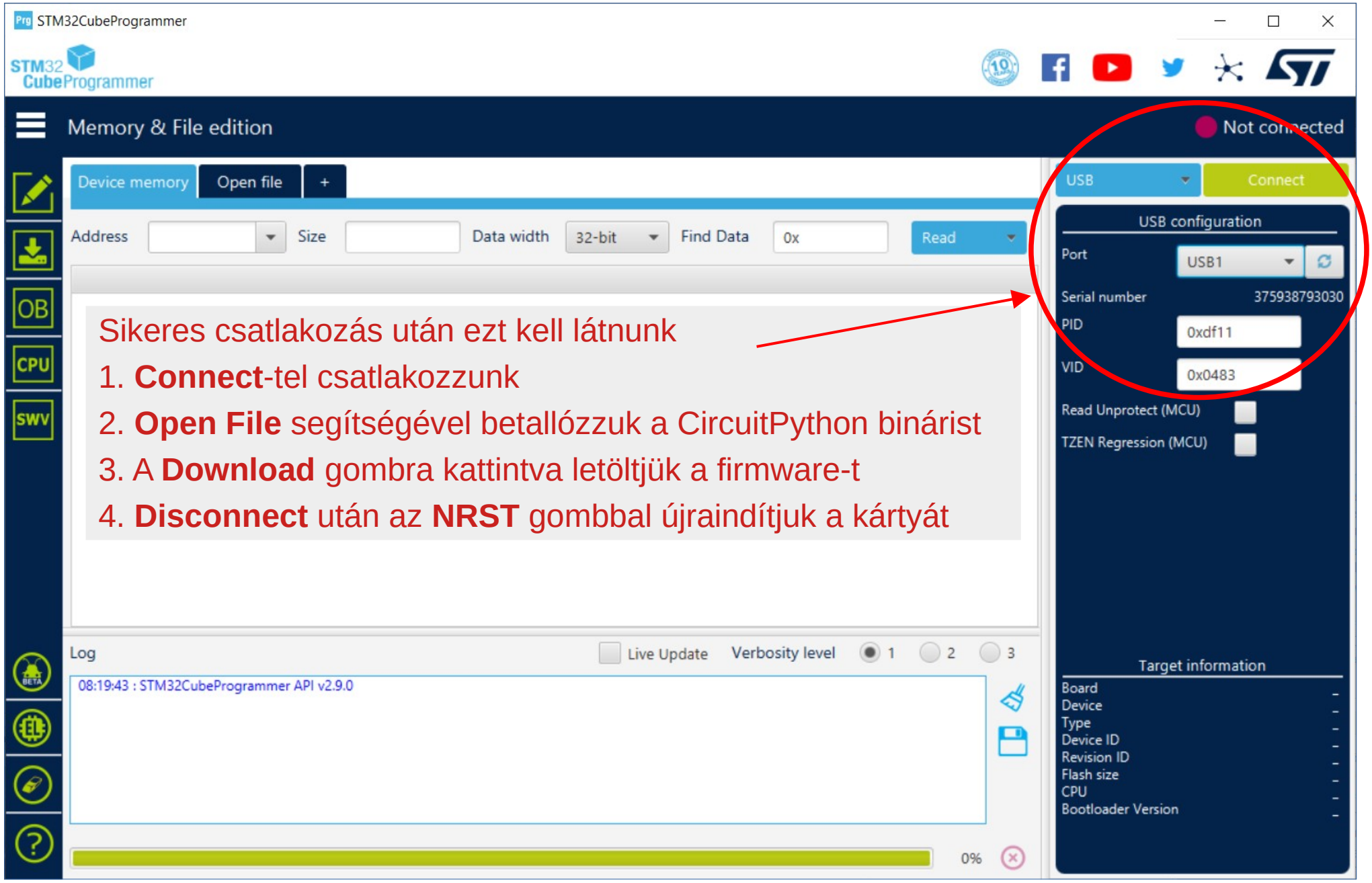




A CircuitPython firmware telepítése

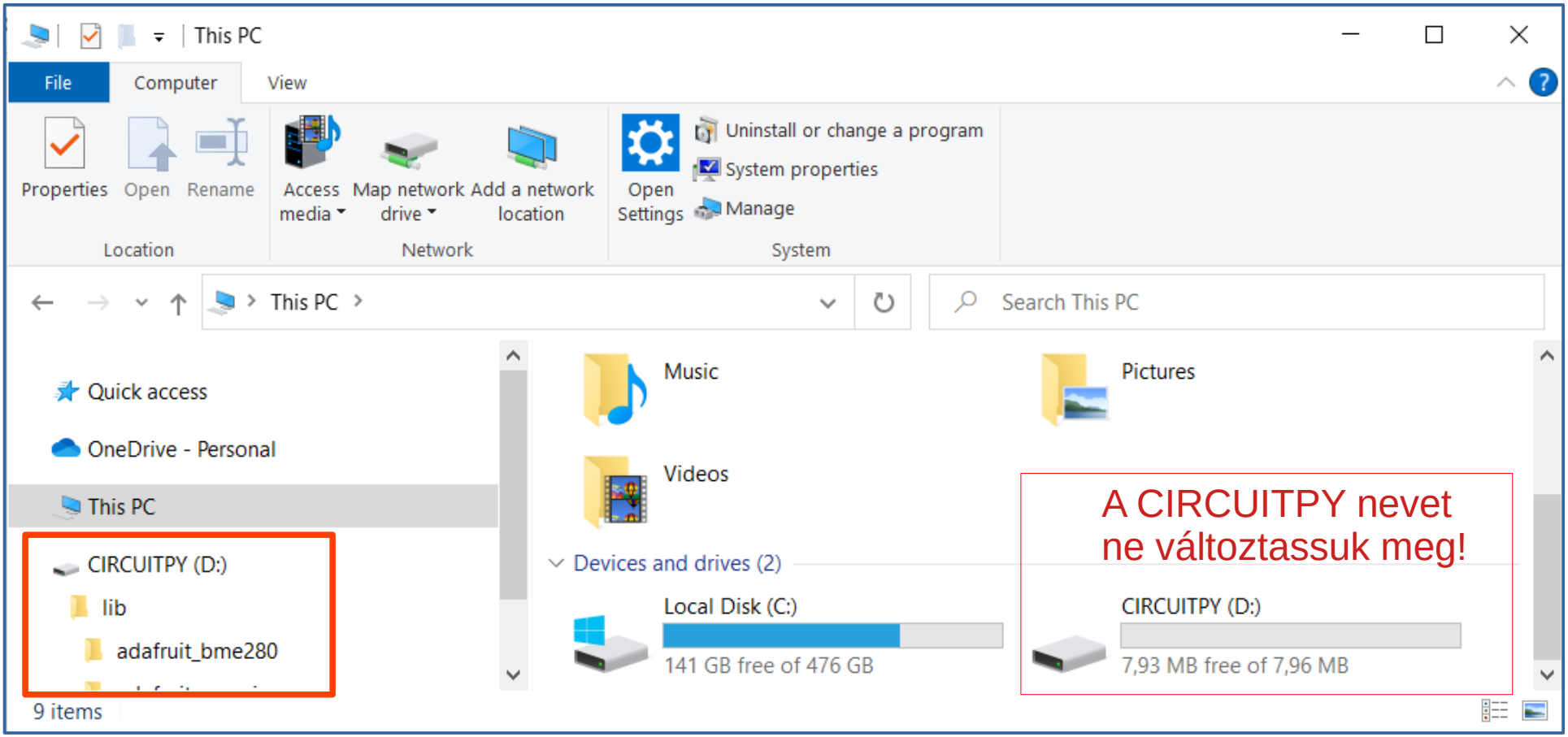
- Mivel a kártyánk gyárilag nem tartalmazza a **CircuitPython** firmware-t, így bele kell töltenünk (csak egyszer kell végrehajtani)
- Az **STM32F411CE blackpill with flash** kártyához a CircuitPython 7.0.0 változatát innen töltöttük le: adafruit-circuitpython-stm32f411ce_blackpill_with_flash-en_US-7.0.0.bin
  Megjegyzés: a flash memória nélküli kártyához másik firmware kell!
- A programozás legegyszerűbben a mikrovezérlő gyárilag beégetett **DFU loaderével** végezhető el, ehhez a számítógép felől az STM32CubeProgrammer letöltő programot használtuk
- **DFU módba** úgy léphetünk be, hogy csatlakoztatjuk a kártyánkat, lenyomjuk és lenyomva tartjuk a **BOOT0** gombot, miközben lenyomjuk majd felengedjük az **NRST** gombot. Ha sikeres volt az USB eszköz felismerés (nincs hibajelzés), akkor elengedhetjük a **BOOT0** gombot is
- Sajnos, a 25 MHz-es kristály miatt bizonytalan a DFU eszköz felismertetése, így kitartóan kell próbálkozni…

STM32Cube Programmer



A CIRCUITPY meghajtó

- A **CircuitPython** firmware egy fájlrendszert is létrehoz, ami egy **CIRCUITPY** nevű meghajtóként jelenik meg. Ebbe kell majd egyszerűen belemásolni a programokat és a **lib** mappába a könyvtári modulokat
- SPI flash memória kiegészítéssel a fájlrendszer 8 MB, enélkül csak néhány kilobájt lenne, ami egy komolyabb alkalmazáshoz már kevés



Állományok a CIRCUITPY meghajtón

- **boot_out.txt** – automatikusan keletkezik, a **CircuitPython** és a kártya nevét, illetve verzió azonosítóját tartalmazza
- **boot.py, boot.txt** – ha bekapcsoláskor, vagy hard reset-kor van ilyen állomány, akkor automatikusan végrehajtásra kerül
- **code.py, code.txt, main.py, main.txt** – ha van ilyen nevű állomány (keresés a fenti sorrendben), akkor a bootolás után, illetve szoft reset, vagy az állomány módosítása után automatikusan végrehajtódik
- **lib** mappa – ide helyezzük el az importálni kívánt kiegészítő modulokat

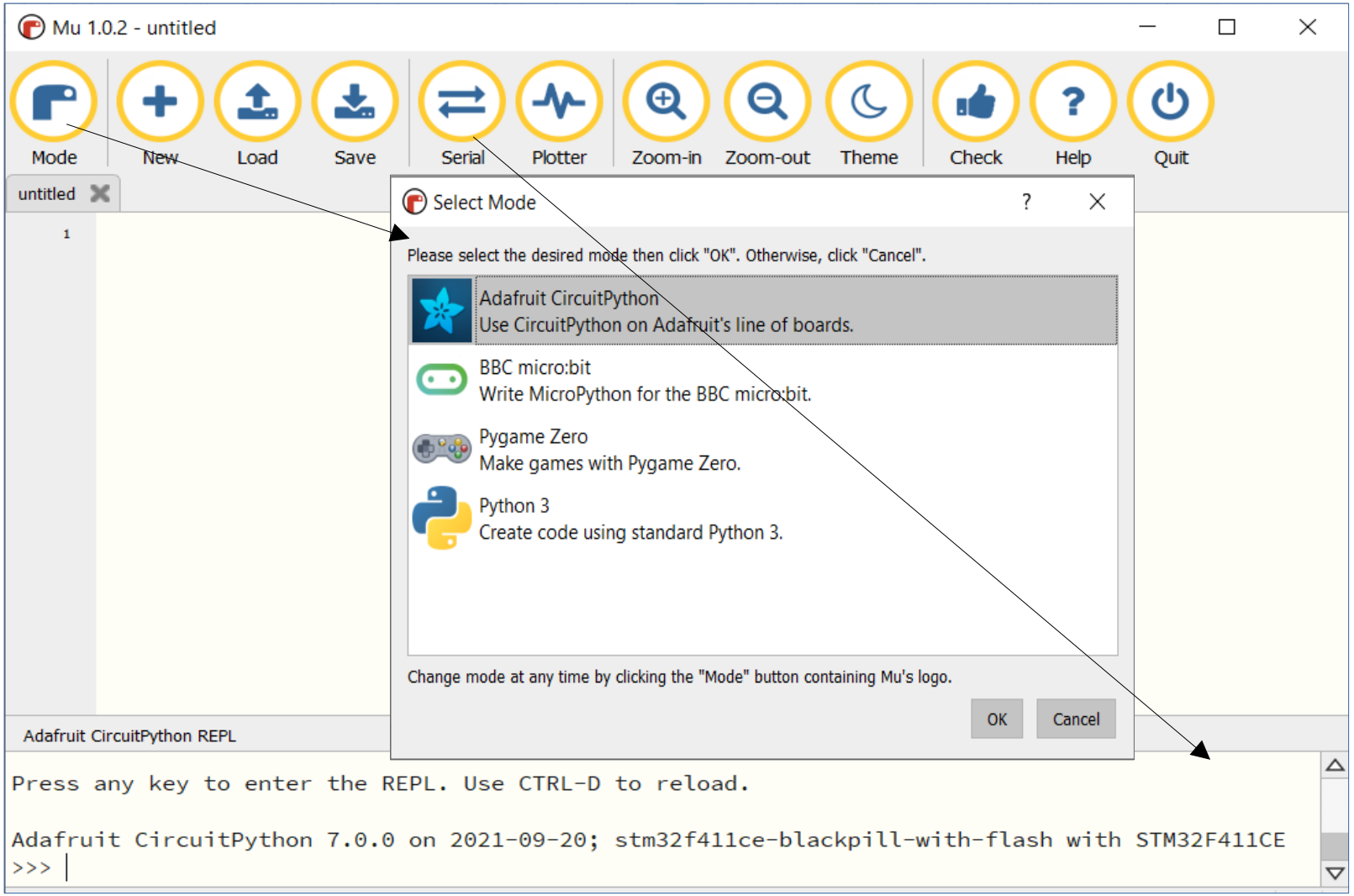
A mintaprogramok futtatása:

- Bemásoljuk a futtatni kívánt programot a **CIRCUITPY** meghajtóra **code.py** néven
- Tetszőleges nevű program futtatása (ha már a **CIRCUITPY** meghajtón van): REPL módban adjuk ki a következő parancsot

```
Adafruit CircuitPython 7.0.0 on stm32f411ce-blackpill-with-flash
>>> exec(open("pin_list.py").read())
```

A Mu editor telepítése

- A programok szerkesztéséhez akár a **Notepad Editort** is használhatjuk, de a **CircuitPythonhoz** készült **Mu editor** valamivel kényelmesebb…
- Letöltés innen: https://codewith.mu/ (van hordozható verzió is)



REPL mód

- REPL = Read Evaluate Print Loop, azaz interaktív mód

```
Press any key to enter the REPL. Use CTRL-D to reload.

Adafruit CircuitPython 7.0.0 on 2021-09-20; stm32f411ce-blackpill-with-flash with
STM32F411CE
>>> import board
>>> dir(board)
['__class__', '__name__', 'A0', 'A1', 'A10', 'A11', 'A12', 'A15', 'A2', 'A3', 'A4',
'A5', 'A6', 'A7', 'A8', 'A9', 'B0', 'B1', 'B10', 'B12', 'B13', 'B14', 'B15', 'B2',
'B3', 'B4', 'B5', 'B6', 'B7', 'B8', 'B9', 'C13', 'C14', 'C15', 'I2C', 'LED', 'SCL',
'SDA', 'board_id']
>>> help("modules")
__main__            board               math                storage
_bleio              builtins            microcontroller     struct
adafruit_bus_device                     busio               micropython       supervisor
adafruit_pixelbuf collections           msgpack             synthio
aesio               digitalio           neopixel_write      sys
analogio            displayio           onewireio           terminalio
array               errno               os                  time
atexit              fontio              pulseio             touchio
audiocore           framebufferio       pwmio               traceback
audiomp3            gc                  rainbowio           usb_cdc
audiopwmio          getpass             random              usb_hid
binascii            io                  re                  usb_midi
bitbangio           json                sdcardio            vectorio
bitmaptools         keypad              sharpdisplay
Plus any modules on the filesystem
```

parancsok

Ezek leírása a Core Modules szekcióban található

A beépített LED villogtatása: ledblink.py

- Írjuk be az alábbi programot!
- A **Save** gombbal töltsük le a **CIRCUITPY** eszközre *code.py* néven!

Mu 1.0.2 - ledblink.py

Mode | New | Load | Save | Serial | Plotter | Zoom-in | Zoom-out | Theme | Check | Help | Quit

ledblink.py

```
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)
```

Adafruit

A kártya kivezetéseinek listázása: pin_list.py

- Listázzuk ki a kivezetések neveit és másodlagos (alias) neveit!

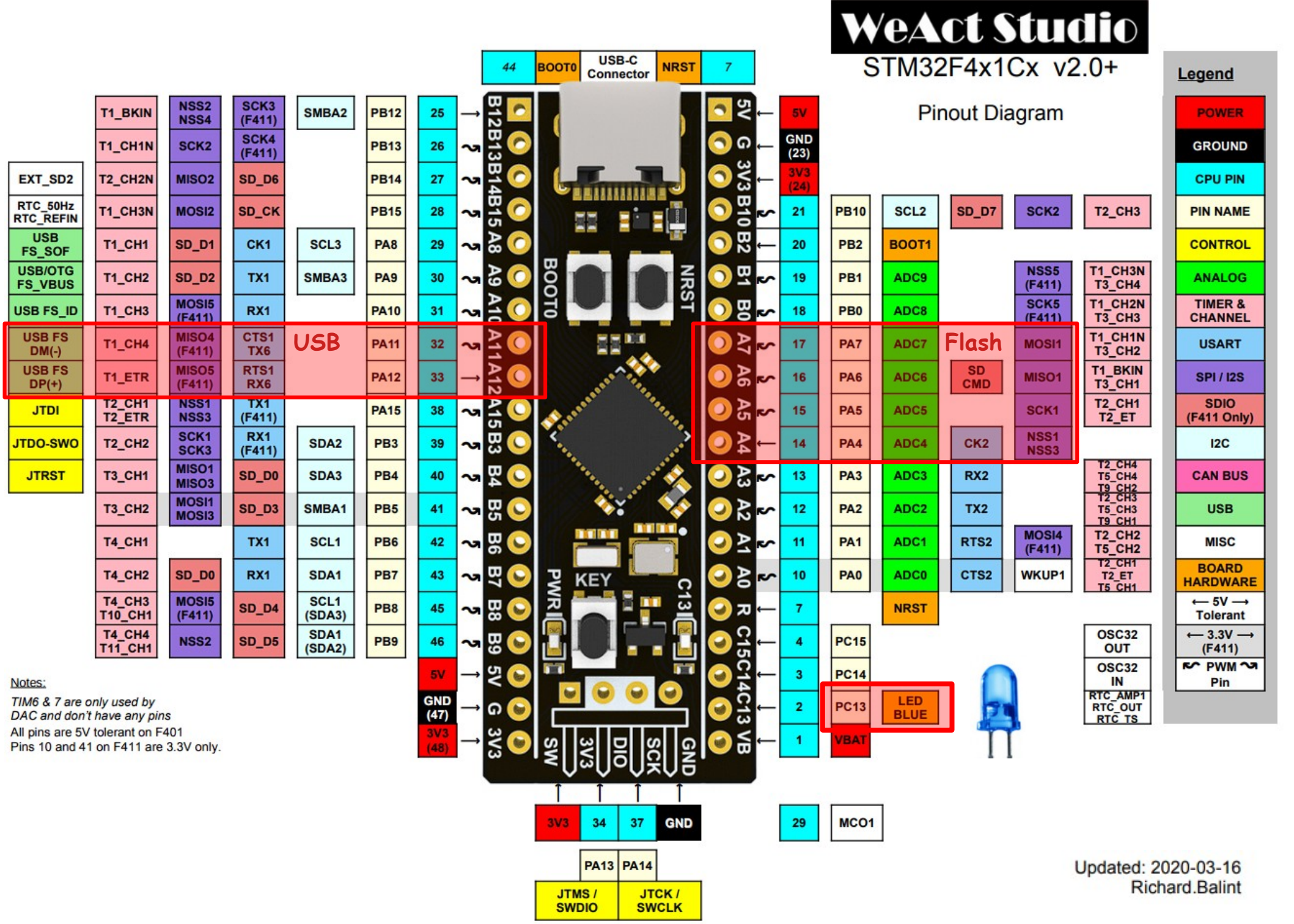
```
import microcontroller
import board

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)
```

```
board.A0
board.A1
board.A2
board.A3
board.A4
board.A5
board.A6
board.A7
```
FLASH

```
board.A7
board.A8
board.A9
board.A10
board.A11
board.A12
board.A15
```
USB

```
board.B0
board.B1
board.B2
board.B3
board.B4
board.B5
board.B6 board.SCL
board.B7 board.SDA
```

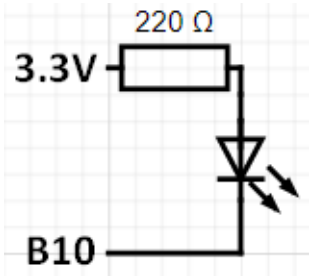
```
board.B8
board.B9
board.B10
board.B12
board.B13
board.B14
board.B15
board.C13 board.LED
```
SPI


WeAct Studio
STM32F4x1Cx v2.0+
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
USB
Flash
BOOT0
NRST
PWR
KEY
C13
SW
3V3
DIO
SCK
GND
LED BLUE
USB-C Connector
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

Digitális ki- és bemenetek kezelése

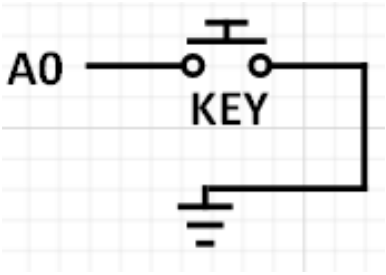
- A digitális ki- és bemenetek kezelése a **DigitalInOut** objektumok segítségével történik, például:
- **Kimenet konfigurálása:**

```
import digitalio, board
led = digitalio.DigitalInOut(board.B10)
led.direction = digitalio.Direction.OUTPUT
led.drive_mode = digitalio.DriveMode.OPEN_DRAIN
```



- **Bemenet konfigurálása:**

```
import digitalio, board
button = digitalio.DigitalInOut(board.A0)
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
```

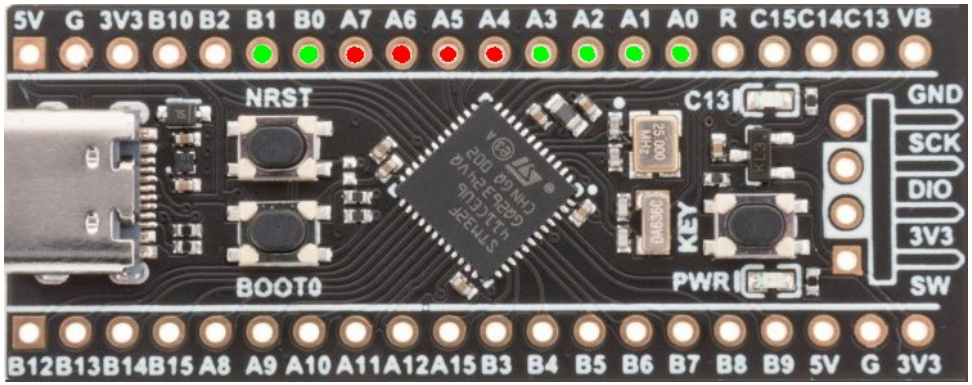
LED vezérlése nyomógombbal: ledswitch.py

- Kapcsolgassuk a beépített LED-et (**C13**) a **KEY** feliratú beépített nyomógomb (**A0**) segítségével!

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

led = DigitalInOut(board.LED)           # Builtin LED (C13)
led.direction = Direction.OUTPUT
led.drive_mode = DriveMode.PUSH_PULL
led.value = False                       # Initially OFF

button = DigitalInOut(board.A0)         # Builtin button (A0)
button.direction = Direction.INPUT
button.pull = Pull.UP

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

Analóg bemenetek kezelése

- Analóg bemenetek: **A0, A1, A2, A3, A8, A9, B0, B1**
  (A4, A5, A6 és A7 is lehetne analóg bemenet, de már foglaltak)



- Az analóg bemenetek kezeléséhez az `analogio` modult importáljuk
- Minden használni kívánt bemenethez az `analogio.AnalogIn` osztályt példányosítjuk
- Például:

```
import board, analogio
analog_in = analogio.AnalogIn(board.A1)
```

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

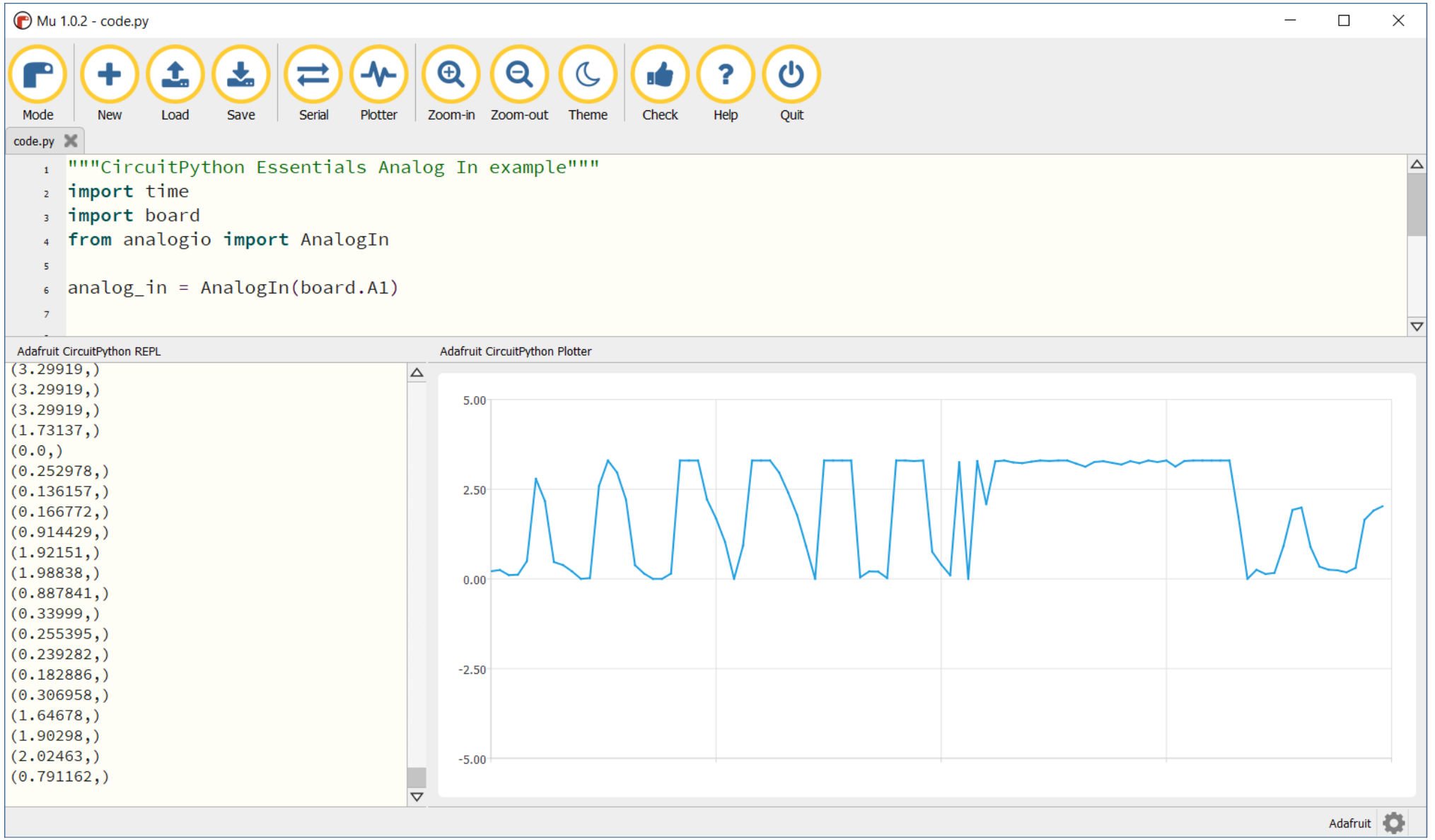
analogin_demo.py

- Vizsgáljuk egy analóg bemenet (A1) feszültségét! Az eredményt kiíratjuk illetve a **Mu editor** beépített **soros plotteren** is megjeleníthetjük
- A plotter *tuple* típusú adatokat vár. Pl. (x, y, z) típusú adatokkal három görbét jeleníthetünk meg
- A **get_voltage()** függvénnyel voltokra átszámolva kapjuk meg az eredményt

```
"""CircuitPython Essentials Analog In example"""
import time
import board
from analogio import AnalogIn

analog_in = AnalogIn(board.A1)


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


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

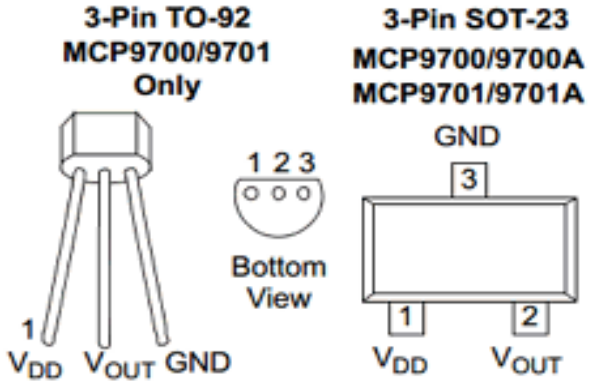
analogin_demo.py

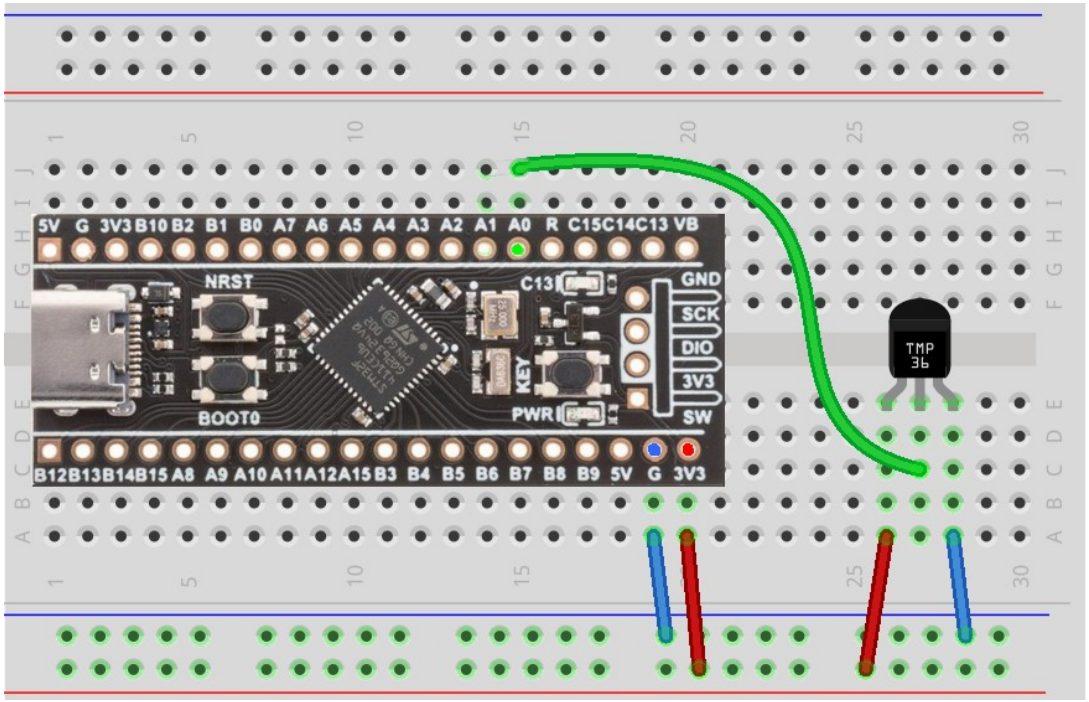
- A futási eredmény az ábrán látható. Itt csak zajt „mértünk”



Analóg hőmérő: analog_thermometer.py

- **MCP9700** (vagy **TMP36**)
  - VDD = 3,3 – 5 V
  - VOUT = 500 mV @ 0 ℃
  - slope = 10 mV/℃

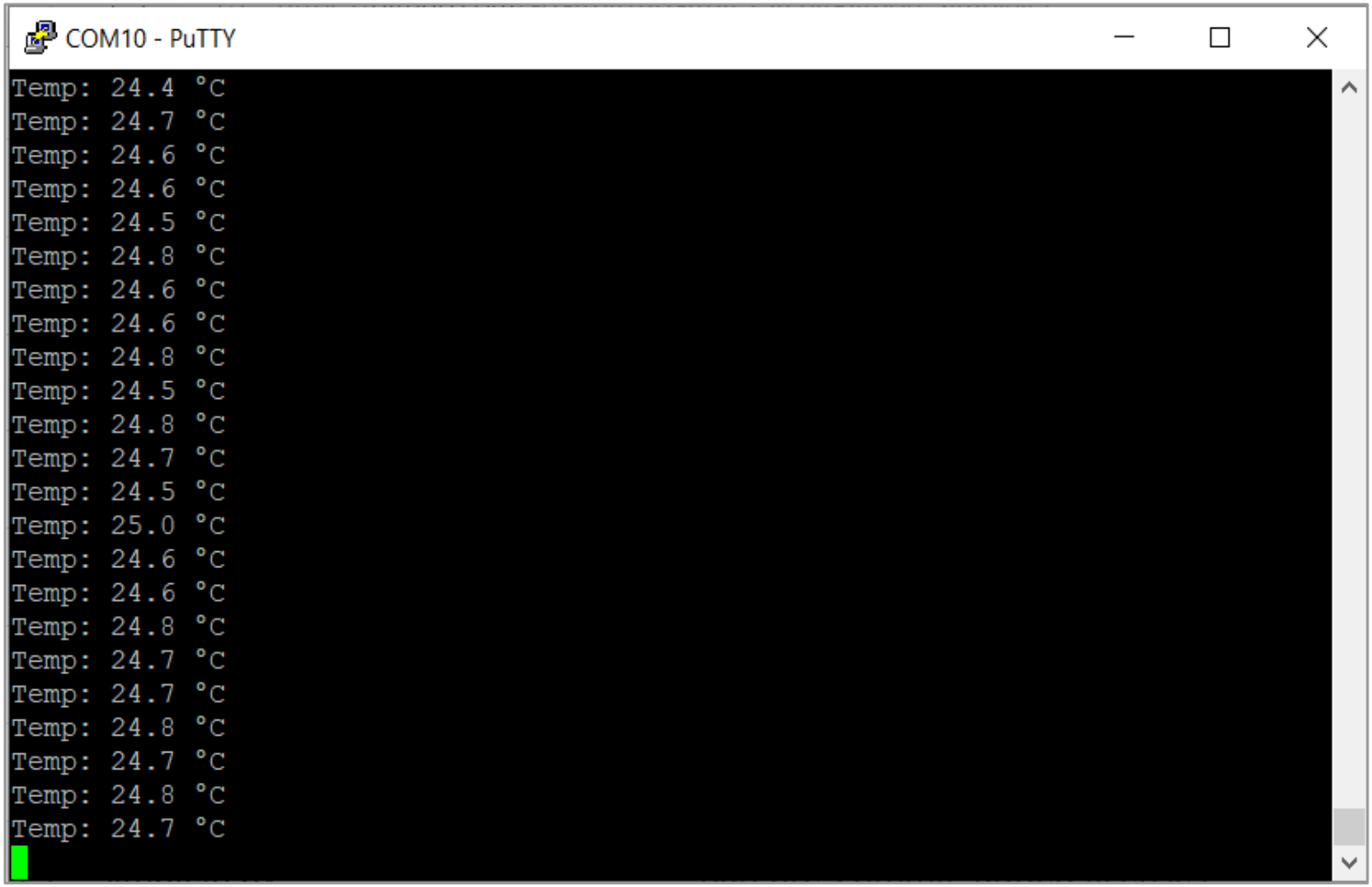


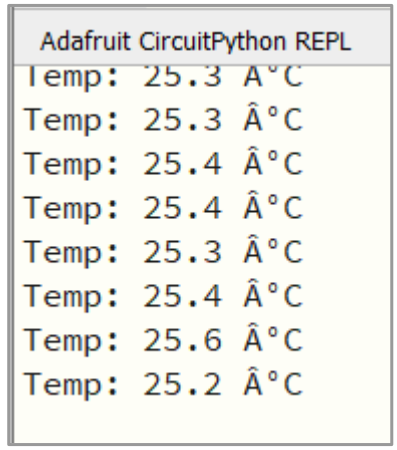



```
""" MCP9700 Analog thermometer"""
import time, board, analogio

thermometer = analogio.AnalogIn(board.A0)

def tempC(pin):
    return (((pin.value *pin.reference_voltage*1000) / 65536)-500)/10

while True:
    print("Temp: %0.1f °C" %(tempC(thermometer)))
    time.sleep(2)
```

analog_thermometer.py

- A **PuTTY.exe** ablakában helyesen jelenik meg a fok jel, a **Mu Editor** terminálablakában viszont nem



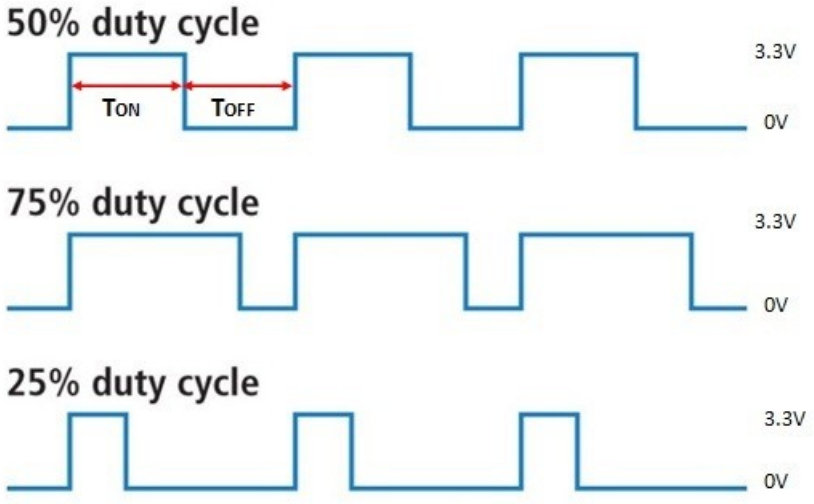


Impulzus-szélesség moduláció (PWM)

- 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



TON: Time requires for pulse to remain ON i.e. HIGH State
TOFF: Time requires for pulse to remain OFF i.e. LOW State

Melyek a PWM-képes kivezetések?

- Az alábbi **pwm_scan.py** 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

```
import board, 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 PWM-capable pins!
    except ValueError:  # When the pin is invalid.
        print("No PWM on:", pin_name)
    except RuntimeError:  # Timer conflict error.
        print("Timers in use:", pin_name)
    except TypeError:  # When checking a non-pin object
        pass  # Passes over non-pin objects in dir(board).
```

```
PWM on: A0
PWM on: A1
PWM on: A2
PWM on: A3
PWM on: A8
PWM on: A9
PWM on: A15
PWM on: B0
PWM on: B1
PWM on: B3
PWM on: B4
PWM on: B5
PWM on: B6
PWM on: B7
PWM on: B8
PWM on: B9
PWM on: B10
```

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

```
import time, board, pwmio
from math import exp, pow

led = pwmio.PWMOut(board.A3, 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)
```

Kiegészítő könyvtárak telepítése

- Azokat a könyvtári modulokat, amelyek az alkalmazáshoz szükségesek, de a **CircuitPython** firmware nem tartalmazza, a `CIRCUITPY` meghajtó ***lib*** mappájába kell bemásolni
- A kiegészítő könyvtárak lehetnek forráskódok (***.py*** kiterjesztés), vagy előfordított könyvtárak (***.mpy*** kiterjesztés)
- Az Adafruit CircuitPython Bundle már 300+ kiegészítő könyvtárat tartalmaz (dokumentáció), de számos más hasznos könyvtár is található az Interneten
- A következő programocskához az Adafruit SimpleIO modult fogjuk felhasználni, az egyszerű zenelejátszáshoz (változó frekvenciájú PWM-mel)
- A fenti linkről letölthető csomagból keressük ki a ***simpleio.py*** állományt és egyszerűen másoljuk be a `CIRCUITPY` meghajtó ***lib*** mappájába!

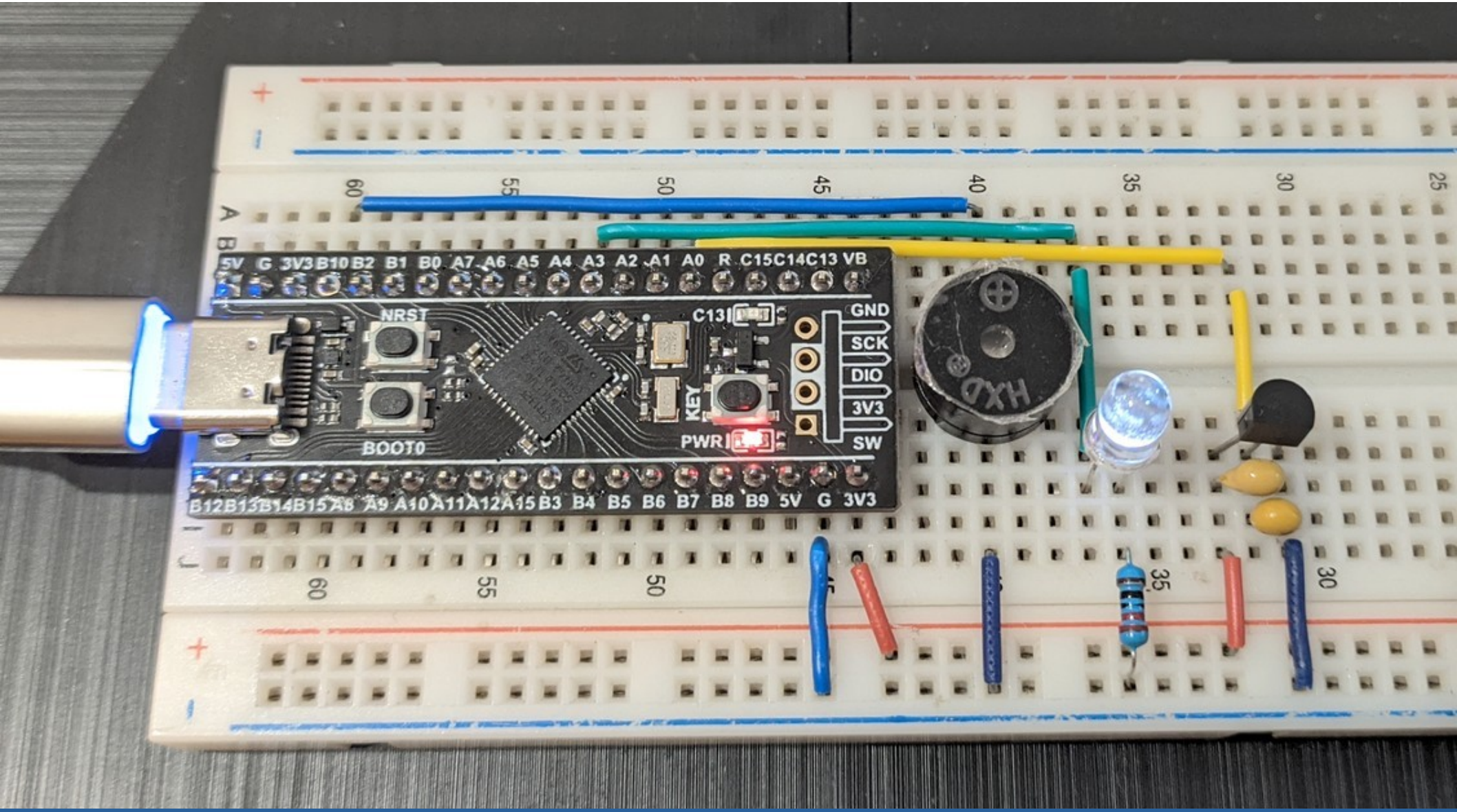
tone_demo.py

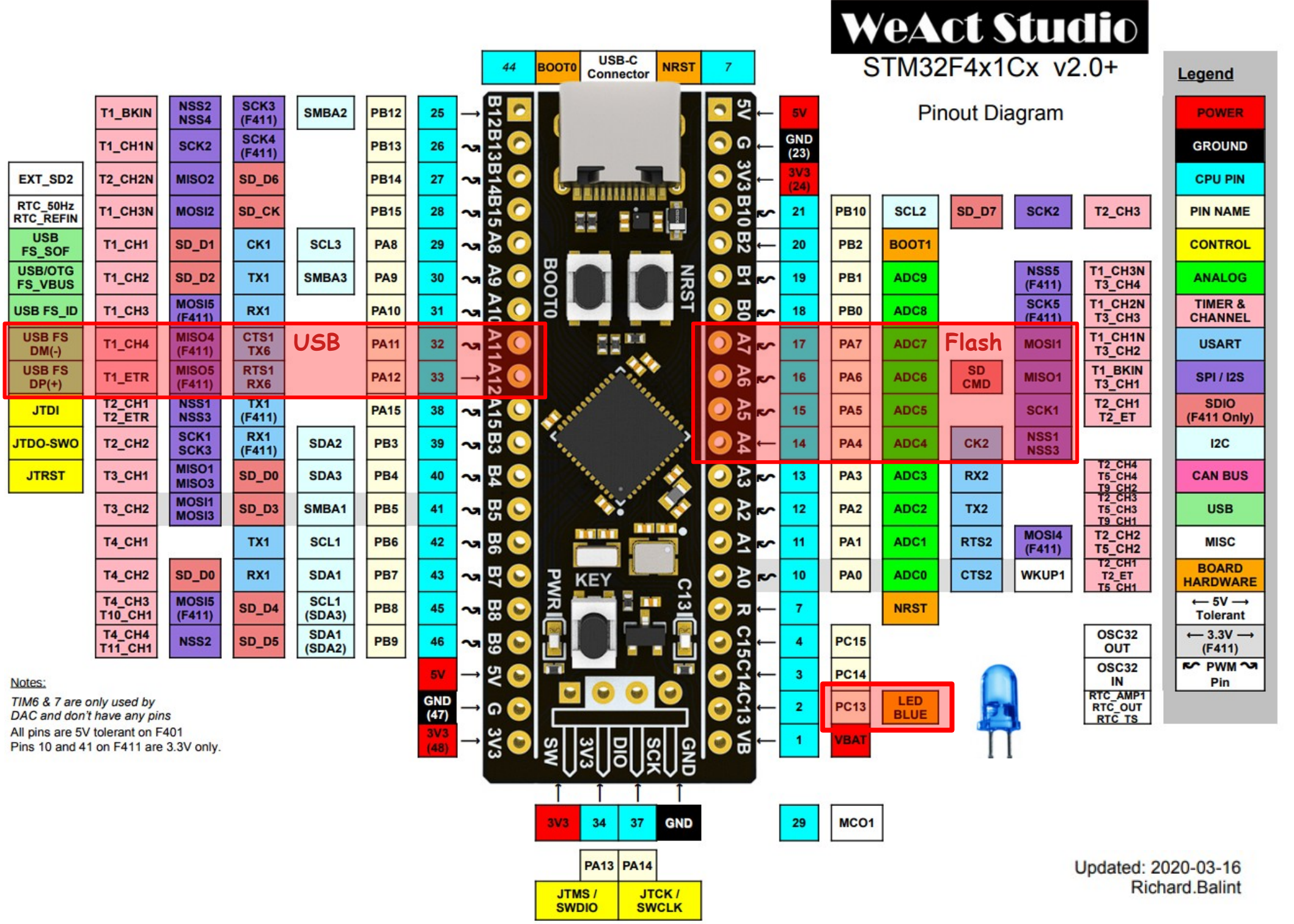
- Ez a program igényli a ***simpleio*** kiegészítő modult *(ld. előző oldal)*
- **Kapcsolás:** kössünk egy passzív piezo csipogót + jelű kivezetését kössük a B10 kimenetre, másik kivezetését pedig a GND-re!
- **Simpleio.tone**(*pin, frequency, duration*) – egy hang lejátszása
  pin – a felhasznált, PWM-képes kivezetés
  frequency – a hang frekvenciája (Hz)
  duration – időtartam (s)
- A **tone_demo.py** program egy oktávot (cdefgahc') játszik le

```
import time
import board
import simpleio

while True:
    for f in (262, 294, 330, 349, 392, 440, 494, 523):
        simpleio.tone(board.B10, f, 0.25)
    time.sleep(2)
```

Kapcsolási elrendezés

- A mai előadás mintapéldái az alábbi kapcsolással próbálhatók ki




WeAct Studio
STM32F4x1Cx v2.0+
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
USB
Flash
BOOT0
NRST
KEY
PWR
C13
SW
3V3
DIO
SCK
GND
LED BLUE
USB-C Connector
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