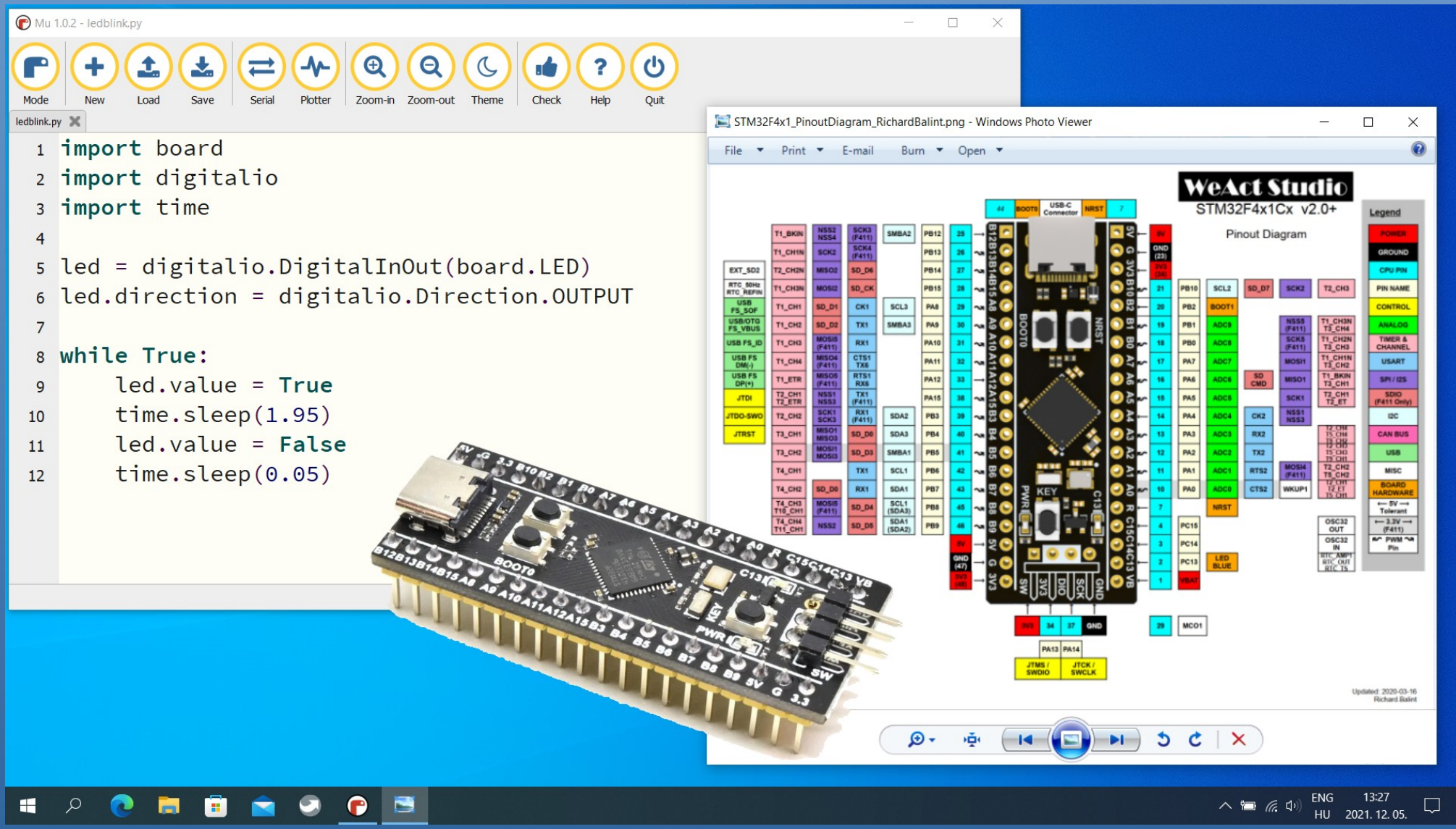
CircuitPython tanfolyam



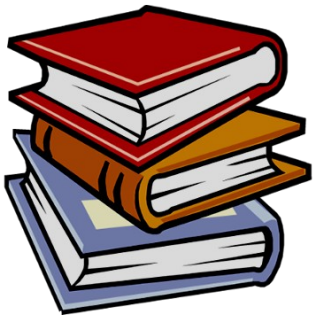
6. Fájlkezelés és az SD kártya használata

Felhasznált és ajánlott irodalom

Python:

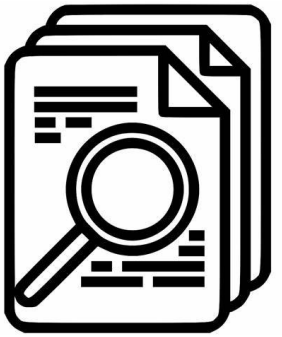
- Mark Pilgrim/Kelemen Gábor: Ugorj fejest a Python 3-ba!
- P. Wentworth et al. (ford. Biró Piroska, Szeghalmy Szilvia és Varga Imre): Hogyan gondolkozz úgy, mint egy informatikus: Tanulás Python 3 segítségével

CircuitPython:



- Adafruit: https://circuitpython.org/downloads
- Adafruit: Adafruit CircuitPython API Reference
- Learn Adafruit: CircuitPython Essentials
- Learn Adafruit: Thermistor
- Learn Adafruit: Adafruit MicroSD SPI or SDIO Card Breakout Board
- Adafruit: github.com/adafruit/Adafruit CircuitPython Bundle

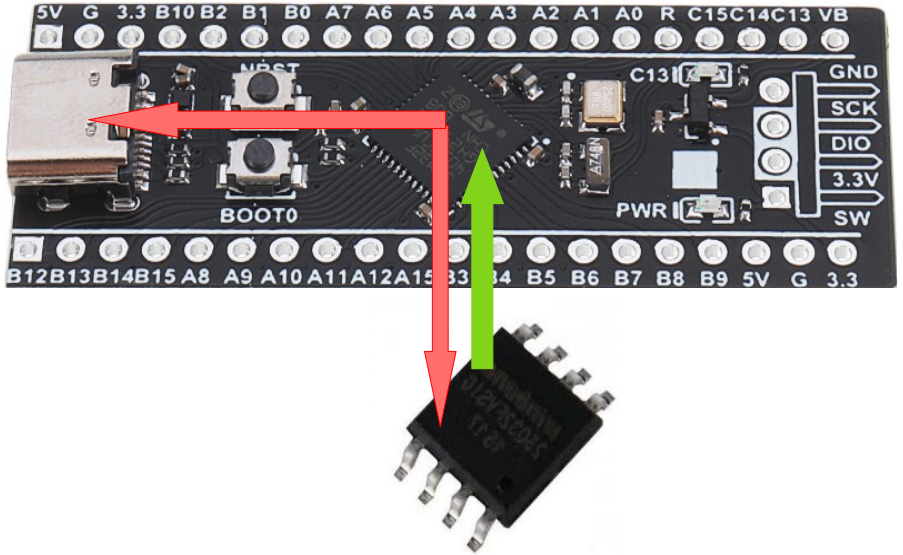
Adatlapok és dokumentáció:



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

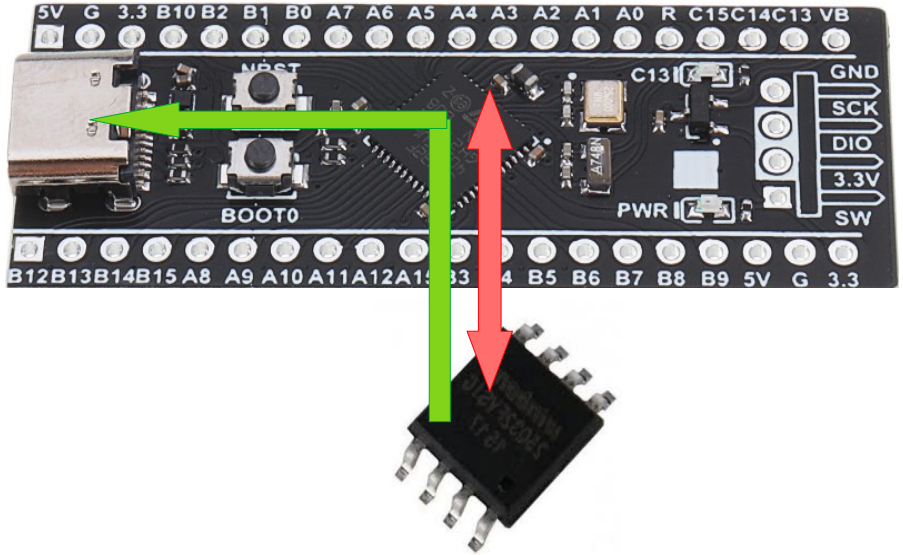
A CIRCUITPY fájlrendszer

- Az **STM32F411CE** blackpill kártya hátulján található flash memória egy kb. 8 MB fájlrendszert biztosít, amely USB-n keresztül a PC-ről is látható **CIRCUITPY** néven és a mikrovezérlő felől is használhatjuk programjainkban
- Az írás azonban egyidejűleg csak egyik oldalról engedélyezett
- Az átkapcsolás célszerűen a **boot.py** állományban végezhető




```
import storage
storage.remount("/", True)
```

(Ez az alapértelmezett beállítás)




```
import storage
storage.remount("/", False)
```

boot.py

- Az alábbi **boot.py** program segítségével egyszerűen kapcsolhatjuk át az írási jogot:
  - ha **RESET**-kor lenyomva tartjuk a **KEY** gombot, akkor a programjainkból írni tudjuk a flash memóriát
  - Ha **RESET**-kor nincs lenyomva a **KEY** gomb, akkor pedig a számítógép felől írható a flash memória

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

button = DigitalInOut(board.A0)        # Builtin KEY button (A0)
button.direction = Direction.INPUT     # Set pin as digital input
button.pull = Pull.UP                  # with internal pullup
time.sleep(0.25)
# CircuitPython can write to flash if button was pressed at RESET
storage.remount("/", button.value)
```

logger.py

- A CPU hőmérsékletét naplózzuk egy fájlba (5 s időközönként)

```
import time
import board
from digitalio import *
import microcontroller
led = DigitalInOut(board.LED)          # Builtin LED (C13)
led.direction = Direction.OUTPUT
led.drive_mode = DriveMode.OPEN_DRAIN
led.value = True                       # Initially OFF (negative logic)
try:
    with open("/temperature.txt", "a") as fp:
        while True:
            temp = microcontroller.cpu.temperature
            fp.write('{0:f}\n'.format(temp))
            fp.flush()
            led.value = not led.value
            time.sleep(5)
except OSError as e:      # Typically when the filesystem isn't writeable...
    delay = 0.5           # ...blink the LED every half second.
    if e.args[0] == 28:   # If the file system is full...
        delay = 0.25      # ...blink the LED faster!
    while True:
        led.value = not led.value
        time.sleep(delay)
```

A storage modul

- A **storage** modul a fájlrendszerek becsatolásval és lecsatolásával kapcsolatos feladatokat látja el, ami általában az operációs rendszerek feladata – de a CircuitPython esetében nincs OS

Tagfüggvényei:

- **mount**(*filesystem:VfsFat, path:str, *, readonly:bool=False*) – fájlrendszer becsatolása a megadott eszközön és elérési úttal (lásd: SD kártya)
- **umount**(*mount: Union[str, VfsFat]*) – a megadott fájlrendszer lecsatolása
- **remount**(*path: str, readonly: bool = False, *, disable_concurrent_write_protection: bool = False*) – fájlrendszer újracsatolása
  A konkurens írás engedélyezése a fájlrendszer sérülését okozhatja
- **getmount**(*mount_path: str*) – az útvonalhoz rendelt eszközt adja vissza
- **erase_filesystem**() – törli a fájlrendszert
- **VfsFat**(*block_device: str*) – új fájlrendszert hoz létre a megadott eszközön
- **disable_usb_drive**() – letiltja a **CIRCUITPY** fájlrendszer USB elérését
- **enable_usb_drive**() – megengedi a **CIRCUITPY** fájlrendszer USB elérését

Fájlok kezelése

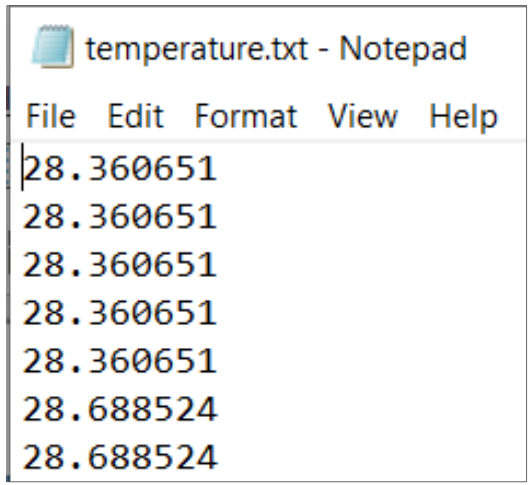
- Az alapvető fájlműveletek elérhetők beépített függvények formájában (további lehetőségek találhatók az **io** modulban)
- *f* = **open**(*name,mode*) – adott nevű fájl megnyitása adott módban
  mode: 'r' (read), 'w' (write), 'a' (append), 'b' (binary), 't' (text)

  Példa a 3. előadásból (**epd_bitmap.py**): `open(filename, "rb")`
- **f.read**(n) – adott számú bájt/karakter beolvasása (mind, ha *n* nincs megadva)
- **f.readline**() – egy sornyi szöveg beolvasása (ha nincs paraméter megadva)
- **f.write**(arg) – adatkiírás a fájlba (a paraméter típusa a fájl típusától függ)
- **f.seek**(n) – fájlmutató pozicionálása
- **f.flush**() – megvárja a kimeneti buffer kiürítését
- **f.close**() – fájl bezárása

Fájlok automatikus bezárása

- Az adatfolyam-objektumok rendelkeznek explicit **close**() metódussal, de mi történik, ha a kódod hibás, és még a **close**() hívása előtt összeomlik? A fájl nyitva maradhat
- A Python 2-nek volt erre egy megoldása: a **try...finally** blokk
- A Python 2.6-tól kezdve van egy még jobb megoldás: **with**, ami egy kontextust hoz létre és belépéskor, illetve kilépéskor értesíti az adatfolyam-objektumot, meghívva annak az **__enter__**, illetve **__exit__** metódusát, lehetőséget nyújtva kilépéskor a fájl bezárására (akkor is, ha egy nem kezelt kivételen át lépünk ki)
- Példa:

```
with open('/temperature.txt', 'rt') as f:
    f.seek(0)
    egy_karakter = f.read(1)
    print(egy_karakter)     # '2'
    egy_sor = f.readline() # '8.360651\n'
    print(egy_sor)
```



A beépített hőmérő kiolvasása

- A beépített hőmérő – néhány más hasznos információval együtt – a **microcontroller** modul **cpu** osztályának tulajdonságaként olvasható ki
- Nézzük meg az interaktív **REPL** módban (a terminálablakban)

```
Adafruit CircuitPython 7.2.5 on 2022-04-06; stm32f411ce-blackpill-with-flash with STM32F411CE
>>> import microcontroller
>>> dir(microcontroller)
['__class__', '__name__', 'Pin', 'Processor', 'ResetReason', 'RunMode', 'cpu', 'delay_us',
'disable_interrupts', 'enable_interrupts', 'nvm', 'on_next_reset', 'pin', 'reset', 'watchdog']
>>> dir(microcontroller.cpu)
['__class__', 'frequency', 'reset_reason', 'temperature', 'uid', 'voltage']
>>> print(microcontroller.cpu.temperature)
33.3931   (Celsius fokokban)
>>> print(microcontroller.cpu.voltage)
3.33086
>>> print(microcontroller.cpu.frequency)
96000000
>>>
```

- Megjegyzés: **Processor** és **cpu** ugyanaz, csak más néven

Az os modul

- Az **os** modul olyan függvényeket biztosít számunkra, amit általában az operációs rendszerek nyújtanak
- **os.chdir**(*path: str*) – az aktuális könyvtár átállítása
- **os.getcwd**() – az aktuális könyvtár nevének lekérése
- **os.listdir**(*dir: str*) – a megadott (ennek hiányában az aktuális) könyvtár bejegyzéseit adja vissza egy lista formájában
- **os.mkdir**(*path: str*) – új könyvtár létrehozása
- **os.rmdir**(path: str) – könyvtár törlése
- **os.remove**(path: str) – fájl törlése
- **os.rename**(old_path: str, new_path: str) – fájl/könyvtár átnevezése
- **os.stat**(path: str) – fájl/könyvtár státuszinformáció lekérése
- **os.uname**() – firmware és hardware információk lekérése
- **os.sep** – a szeparátor karakter (esetünkben: '/')

Az os modul

- Az alábbiakban előbb **REPL** módban próbáljuk ki az **os** modul függvényeit, majd egy program segítségével formázottan listázzuk ki a **CIRCUITPY** fájlrendszer állományait

```
Adafruit CircuitPython 7.2.5 on 2022-04-06; stm32f411ce-blackpill-with-flash
with STM32F411CE
>>> import os
>>> dir(os)
['__class__', '__name__', 'remove', 'sep', 'chdir', 'getcwd', 'listdir',
'mkdir', 'rename', 'rmdir', 'stat', 'statvfs', 'sync', 'uname', 'unlink',
'urandom']
>>> os.chdir('lib')
>>> os.getcwd()
'/lib'
>>> os.listdir()
['adafruit_displayio_ssd1306.mpy', 'adafruit_displayio_sh1106.mpy',
'adafruit_display_text', 'adafruit_bitmap_font', 'adafruit_display_shapes']
>>> os.stat('/boot.py')
(32768, 0, 0, 0, 0, 0, 598, 1650442008, 1650442008, 1650442008)
>>> os.stat('/lib')
(16384, 0, 0, 0, 0, 0, 0, 1480810702, 1480810702, 1480810702)
>>>
```

listdir.py

```
import os

def print_directory(path, tabs=0):
    for file in os.listdir(path):
        stats = os.stat(path + "/" + file)
        filesize = stats[6]
        isdir = stats[0] & 0x4000
        if filesize < 1000:
            sizestr = str(filesize) + " by"
        elif filesize < 1000000:
            sizestr = "%0.1f KB" % (filesize / 1000)
        else:
            sizestr = "%0.1f MB" % (filesize / 1000000)
        prettyprintname = ""
        for _ in range(tabs):
            prettyprintname += "   "
        prettyprintname += file
        if isdir:
            prettyprintname += "/"
        print('{0:<40} Size: {1:>10}'.format(prettyprintname, sizestr))
        if isdir:
            print_directory(path + "/" + file, tabs + 1)

print("Files on filesystem:")
print("====================")
print_directory("/")
while(True):
    pass
```

A **CIRCUITPY** fájlrendszer rekurzív kilistázása

listdir.py futási eredménye

- Az alábbi listán egy futtatás eredménye látható

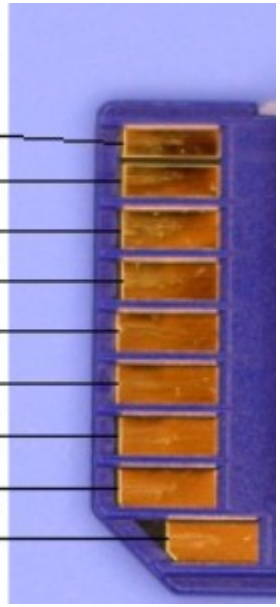
```
code.py output:
Files on filesystem:
====================
code.py                                  Size:     896 by
lib/                                     Size:       0 by
   adafruit_displayio_ssd1306.mpy        Size:     750 by
   adafruit_displayio_sh1106.mpy         Size:     664 by
   adafruit_display_text/                Size:       0 by
      bitmap_label.mpy                   Size:     3.9 KB
      label.mpy                          Size:     3.8 KB
      __init__.mpy                       Size:     4.6 KB
   adafruit_bitmap_font/                 Size:       0 by
      bdf.mpy                            Size:     2.2 KB
      bitmap_font.mpy                    Size:     452 by
      glyph_cache.mpy                    Size:     394 by
      pcf.mpy                            Size:     4.1 KB
      ttf.mpy                            Size:     688 by
      __init__.py                        Size:       0 by
   adafruit_display_shapes/              Size:       0 by
      circle.mpy                         Size:     528 by
      line.mpy                           Size:     398 by
      polygon.mpy                        Size:     1.0 KB
      rect.mpy                           Size:     884 by
      roundrect.mpy                      Size:     1.6 KB
      sparkline.mpy                      Size:     1.2 KB
      triangle.mpy                       Size:     1.1 KB
boot_out.txt                             Size:     138 by
System Volume Information/               Size:       0 by
   WPSettings.dat                        Size:      12 by
   IndexerVolumeGuid                     Size:      76 by
boot.py                                  Size:     598 by
```

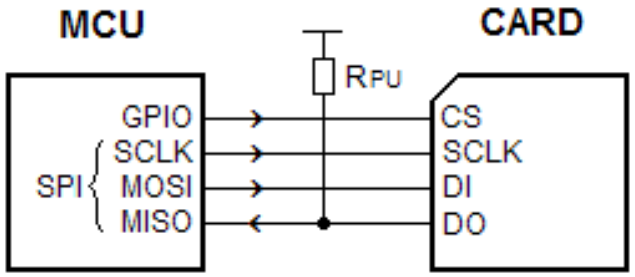
Secure Digital (SD) kártyák

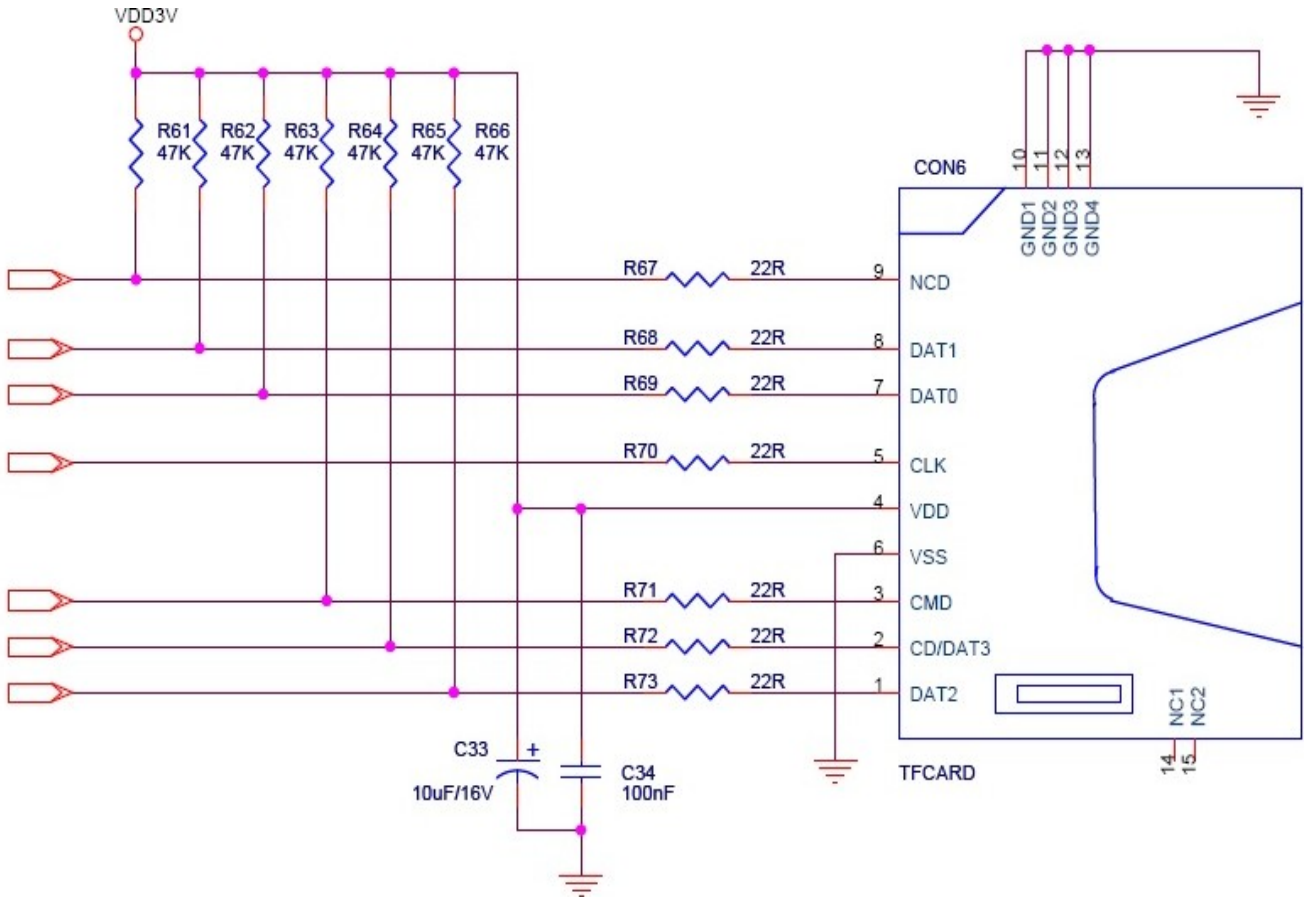
- A **Secure Digital Memory Card** *de facto* szabvány a hordozható eszközök memóriakártyái közül
- Meghajtható **SPI** módban, illetve 1, vagy 4 adatvonalas **SDIO** módban (pl. az **STM32F446RE** mikrovezérlő rendelkezik **SDIO** perifériával is), az adatok 512 bájtos blokkokba vannak szervezve

SDC

No	SD	SPI
8	DAT1	
7	DAT0	DO
6	Vss2	
5	CLK	SCLK
4	Vcc	
3	Vss1	
2	CMD	DI
1	CAT3	CS
9	DAT2	







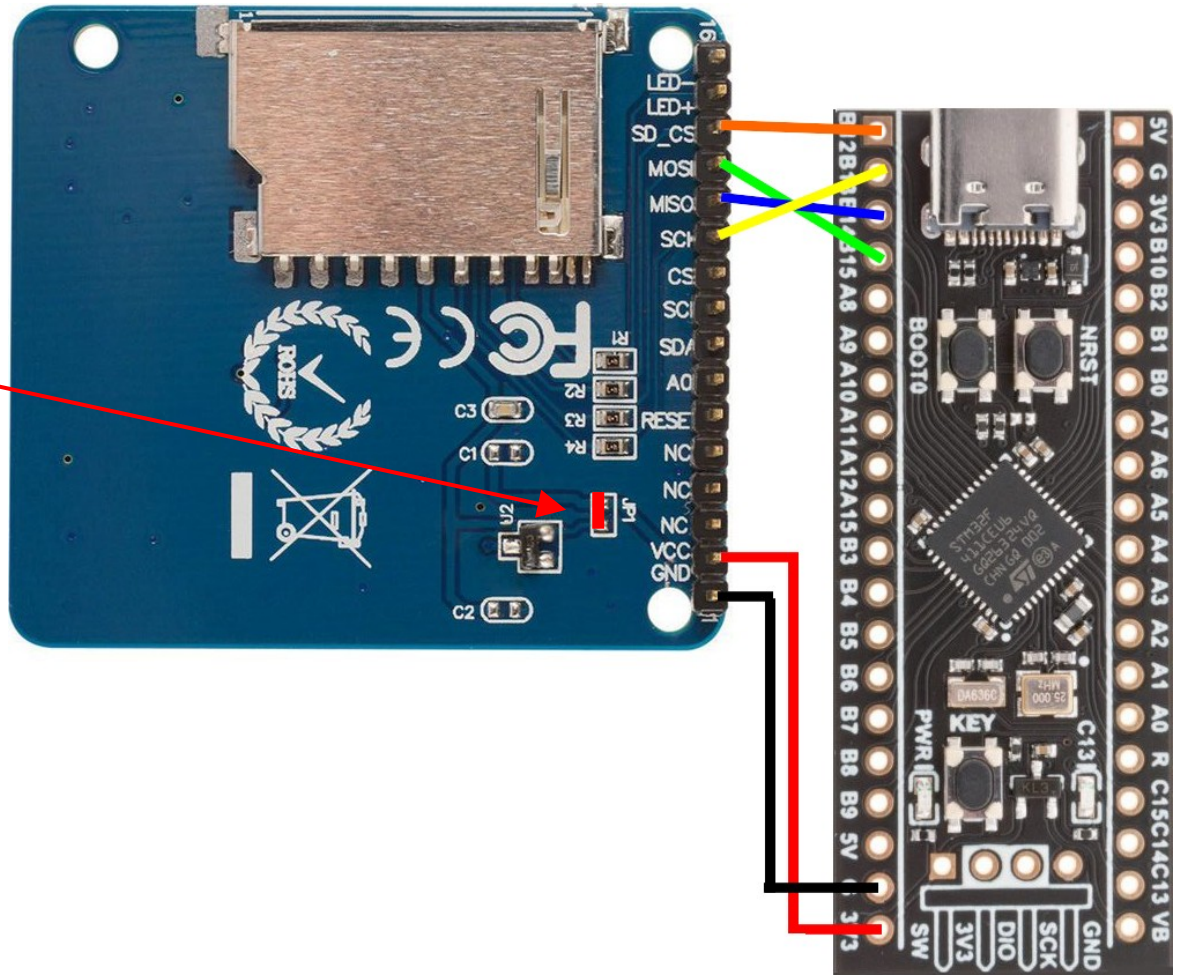
Secure Digital (SD) kártyák jelölései

- A kártya jelölései közlik a kártya kapacitását, típusát/kategóriáját, a maximális olvasási és a minimális írási sebességet
- **SD:** max. 2 GB, **SD HC:** max 32 GB (high capacity),
  SD XC: max 2 TB (extended capacity)

- Max. olvasási sebesség
- A kártya típusa (SD HC)
- UHS kategória (itt: UHS-I)
- SD sebességosztály (6 MB/s)
- UHS sebességosztály (itt nincs)
- Kártya kapacitása (itt 4GB)

Bekötési vázlat

- Egy 1.8” TFT kijelző modul hátulján található SD kártya foglalatot használtunk fel
- A rendelkezésre álló kivezetések csak az 1-bites mód használatát teszik lehetővé (SPI/SDIO)
- A tápfeszültség forrasztással választható (3,3V, vagy 5V)



SD board	STM32F411CE	
SD_CS	NSS2	B12
MOSI	MOSI2	B15
MISO	MISO2	B14
SCK	SCK2	B13
VCC	VCC	3V3
GND	GND	G

SD kártya szoftveres meghajtása

- A **CircuitPython** elvileg háromféle SD-kártya meghajtót nyújt:
  - Adafruit_sdcard – a kevesebb memóriújú kártyákhoz (SPI)
  - **sdcardio** – az **STM32F411CE** kártya esetén ez **beépített modul** (SPI)
  - **sdioio** – az **SDIO** perifériát használja, esetünkben nem támogatott
- Egy összehasonlítás a *Benchmarking SD card access* c. tutoriálban található, amelyben a fenti periféria könyvtárak teljesítményét méri meg egy tesztprogram segítségével

Programkönyvtár	Olvasási sebesség	Naplózási sebesség
adafruiit_sdcard	120 kB/s	160 sor/s
sdcardio	650 kB/s	700 sor/s
sdioio	1100 kB/s	1000 sor/s

Az SD-kártya becsatolása

- Az SD-kártyát az SPI 2. csatorna segítségével kezeljük
- A konfigurálás lépései:
  - Konfiguráljuk az SPI csatornát
  - Konfiguráljuk az **sdcardio** csatornát (SPI és CS megadásával)
  - **VfsFat** fájlrendszer létrehozása a megadott blokk-eszközhöz
  - A fájlrendszer becsatolása a megadott elérési úttal
- Az alábbi kódot mentsük el **mount_sd.py** néven a **lib** mappába!

```
import board
import busio
import sdcardio
import storage

spi = busio.SPI(board.B13, MOSI=board.B15, MISO=board.B14)
cs = board.B12
sdcard = sdcardio.SDCard(spi, cs)
vfs = storage.VfsFat(sdcard)
storage.mount(vfs, "/sd")
```

sdcardio_test.py

- Az alábbi programmal létrehozunk egy fájlt, beleírunk egy sort, hozzáfűzünk egy újabb sort, majd kilistázzuk az egészet

```
import board
import busio
import sdcardio
import storage
spi = busio.SPI(board.B13,MOSI=board.B15,MISO=board.B14)
cs = board.B12
sdcard = sdcardio.SDCard(spi, cs)
vfs = storage.VfsFat(sdcard)
storage.mount(vfs, "/sd")

with open("/sd/test.txt", "w") as f:
    f.write("Hello world!\r\n")

with open("/sd/test.txt", "a") as f:
    f.write("This is another line!\r\n")

with open("/sd/test.txt", "r") as f:
    print("Printing lines in file test.txt:")
    for line in f:
        print(line, end='')
```

Ehelyett
`import mount_sd`
is írható, ha előzőleg elmentettük

Figyeljük meg a soronkénti listázás technikáját!

sdcardio_test.py futási eredménye

Mu 1.0.2 - code.py

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

code.py

```
1  import mount_sd
2
3  with open("/sd/test.txt", "w") as f:
4      f.write("Hello world!\r\n")
5
6  with open("/sd/test.txt", "a") as f:
7      f.write("This is another line!\r\n")
8
9  with open("/sd/test.txt", "r") as f:
10     print("Printing lines in file test.txt:")
11     for line in f:
12         print(line, end='')
13
14 while(True):
15     pass
```

Adafruit CircuitPython REPL

```
Auto-reload is on. Simply save files over USB to run them or enter REPL to disable.
code.py output:
Printing lines in file test.txt:
Hello world!
This is another line!
```

Adafruit

sdcardio_listdir.py

```
import os
import mount_sd

def print_directory(path, tabs=0):
    for file in os.listdir(path):
        stats = os.stat(path + "/" + file)
        filesize = stats[6]
        isdir = stats[0] & 0x4000
        if filesize < 1000:
            sizestr = str(filesize) + " by"
        elif filesize < 1000000:
            sizestr = "%0.1f KB" % (filesize / 1000)
        else:
            sizestr = "%0.1f MB" % (filesize / 1000000)
        prettyprintname = ""
        for _ in range(tabs):
            prettyprintname += "   "
        prettyprintname += file
        if isdir:
            prettyprintname += "/"
        print('{0:<40} Size: {1:>10}'.format(prettyprintname, sizestr))
        if isdir:
            print_directory(path + "/" + file, tabs + 1)

print("Files on filesystem:")
print("====================")
print_directory("/sd")
```

Az SD-kártya fájlrendszerének rekurzív kilistázása

sdcardio_listdir.py futási eredménye

Mu 1.0.2 - code.py

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

code.py

```
1  import board
2  import busio
3  import sdcardio
4  import storage
5  import os
6
7  spi = busio.SPI(board.B13, MOSI=board.B15, MISO=board.B14)
8  cs = board.B12
9  sdcard = sdcardio.SDCard(spi, cs)
10 vfs = storage.VfsFat(sdcard)
11 storage.mount(vfs, "/sd")
12
```

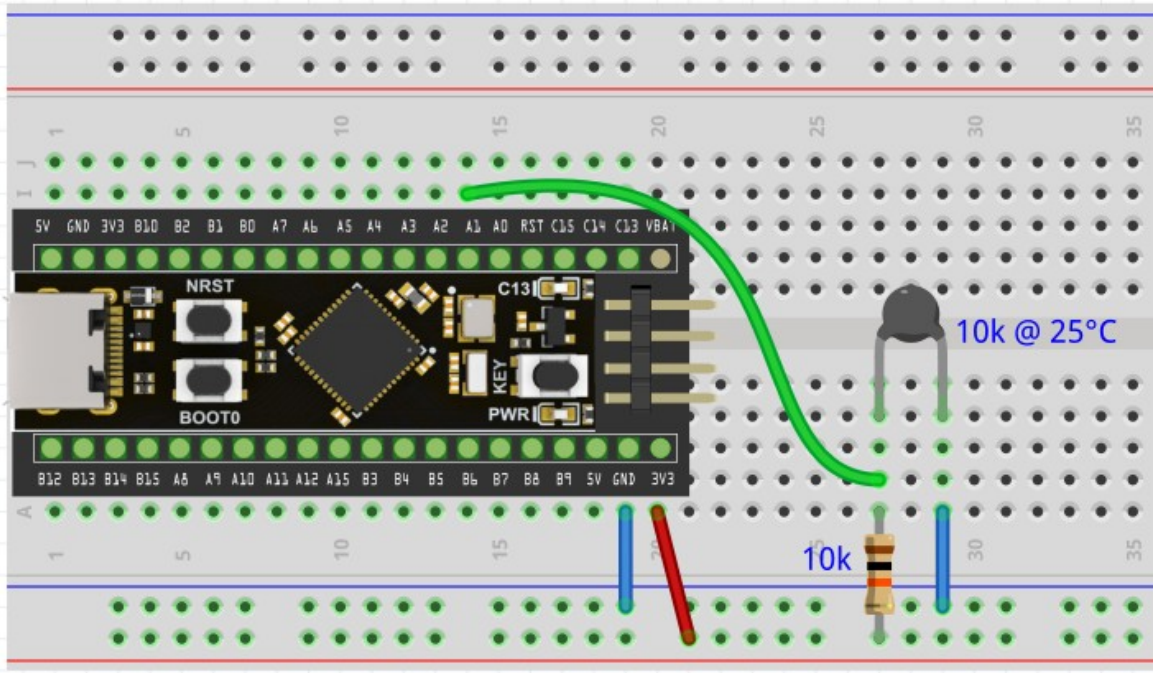
Adafruit CircuitPython REPL

```
Auto-reload is on. Simply save files over USB to run them or enter REPL to disable.
code.py output:
Files on filesystem:
====================
blinka.bmp                                Size:     1.8 MB
welcome-to-circuitpython.pdf              Size:    15.8 MB
circuitpython-essentials.pdf              Size:     8.3 MB
test.txt                                  Size:      37 by
```

Hőmérés termisztorral

- A termisztor hőérzékeny ellenállás, a kiindulási érték nálam 25°C-on 10 kΩ, a hümérséklet növekedésekor az ellenállás csökken
- A bekötés az alábbi ábrán látható, az osztó felső tagja egy fix 10 kΩ-os ellenállás. Az osztó középső pontját az A1 analóg bemenetre kötjük
- A leosztott feszültség, Vo = Vref x R / (R + 10k)
- Az ADC által mért szám, N_{ADC} = Vo * 65 536 / Vref
- N_{ADC} = R * 65 536 / (R + 10k)
- A fentiekből:
  R = 10k/(65 536/N_{ADC}−1)

```
import board
import analogio
term=analogio.AnalogIn(board.A1)
R = 10000/(65536/term.value - 1)
print(R)

9375.87        (26 és 27°C között)
```



Hőmérsékletté alakítás

- A termisztor ellenállásából a Steinhart-Hart egyenlettel kaphatjuk meg a hőmérsékletet, ám ez kissé komplikált:

$$\frac{1}{T} = A + B\ln R + C(\ln R)^3$$

- Ehelyett az egyszerűsítet B-paraméteres egyenletet oldjuk meg, ahol B = 3000 – 4000 közötti érték

$$\frac{1}{T} = \frac{1}{T_0} + \frac{1}{B}\left(\ln\frac{R}{R_0}\right)$$

- Az alábbi kód forrása: learn.adafruit.com/thermistor/circuitpython

```
def steinhart_temperature_C(r, Ro=10000.0, To=25.0, beta=3950.0):
    import math
    steinhart = math.log(r / Ro) / beta      # log(R/Ro) / beta
    steinhart += 1.0 / (To + 273.15)         # log(R/Ro) / beta + 1/To
    steinhart = (1.0 / steinhart) - 273.15   # Invert, convert to C
    return steinhart

>>> steinhart_temperature_C(R)
26.4574
```

thermistor.py

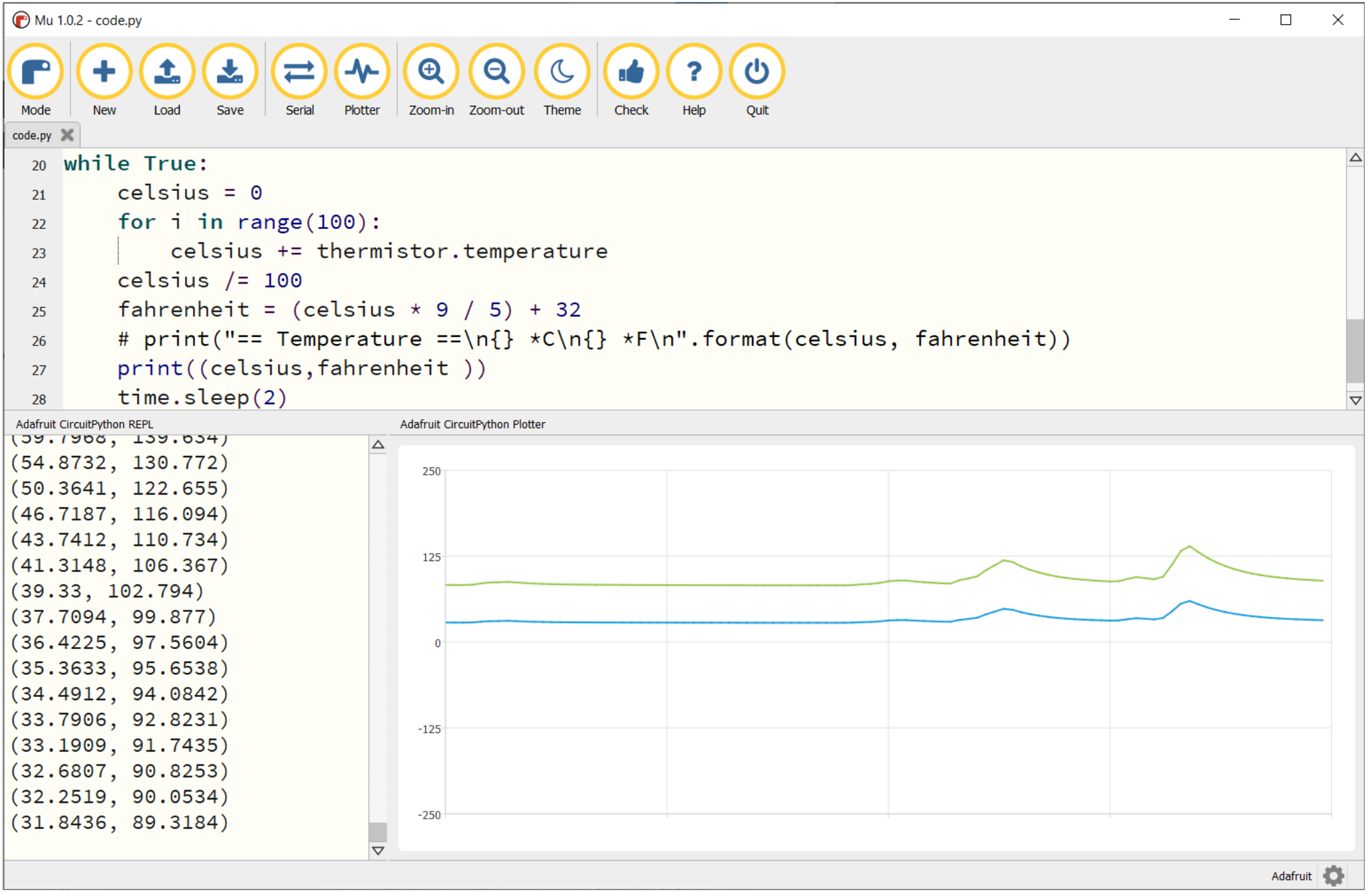
- A kényelmesebb út az Adafruit_thermistor programkönyvtár használata
- Az ingadozások csökkentése érdekében átlagolást is végezhetünk

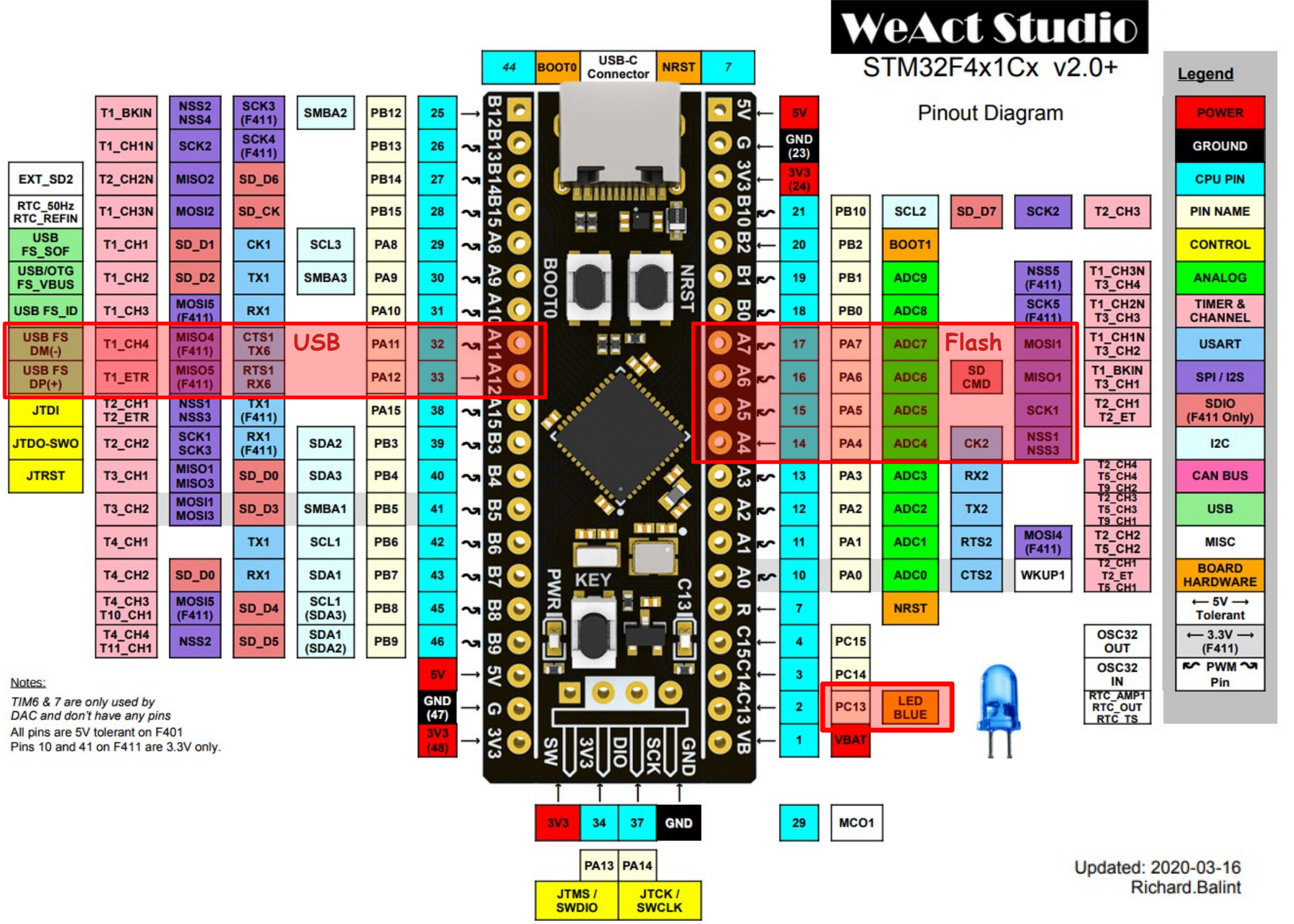
```
import time
import board
import adafruit_thermistor

pin = board.A1
resistor = 10000
resistance = 10000
nominal_temp = 25
b_coefficient = 3950
thermistor = adafruit_thermistor.Thermistor(pin, resistor, resistance,
nominal_temp, b_coefficient, high_side=False)

while True:
    celsius = 0
    for i in range(100):
        celsius += thermistor.temperature
    celsius /= 100
    fahrenheit = (celsius * 9 / 5) + 32
    print((celsius,fahrenheit ))
    time.sleep(2)
```

True: felső ágban a termisztor
False: alsó ágban a termisztor

Átlagolás

thermistor.py futási eredménye




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-C Connector
BOOT0
NRST
USB
Flash
KEY
PWR
C13
SW
3V3
DIO
SCK
GND
JTMS / SWDIO
JTCK / SWCLK
LED BLUE
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