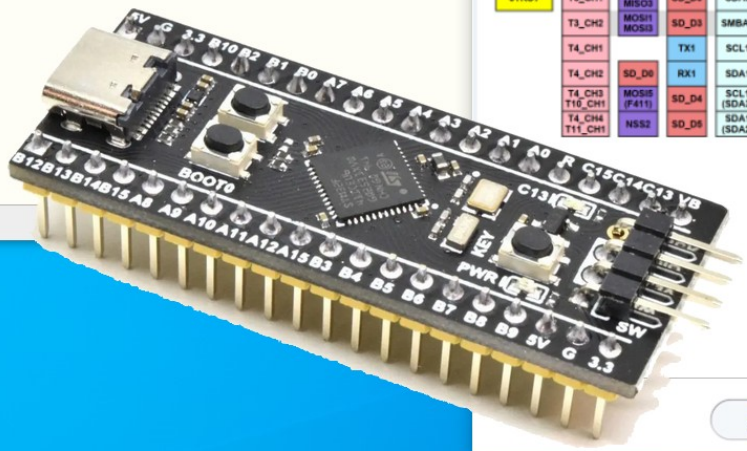
CircuitPython tanfolyam

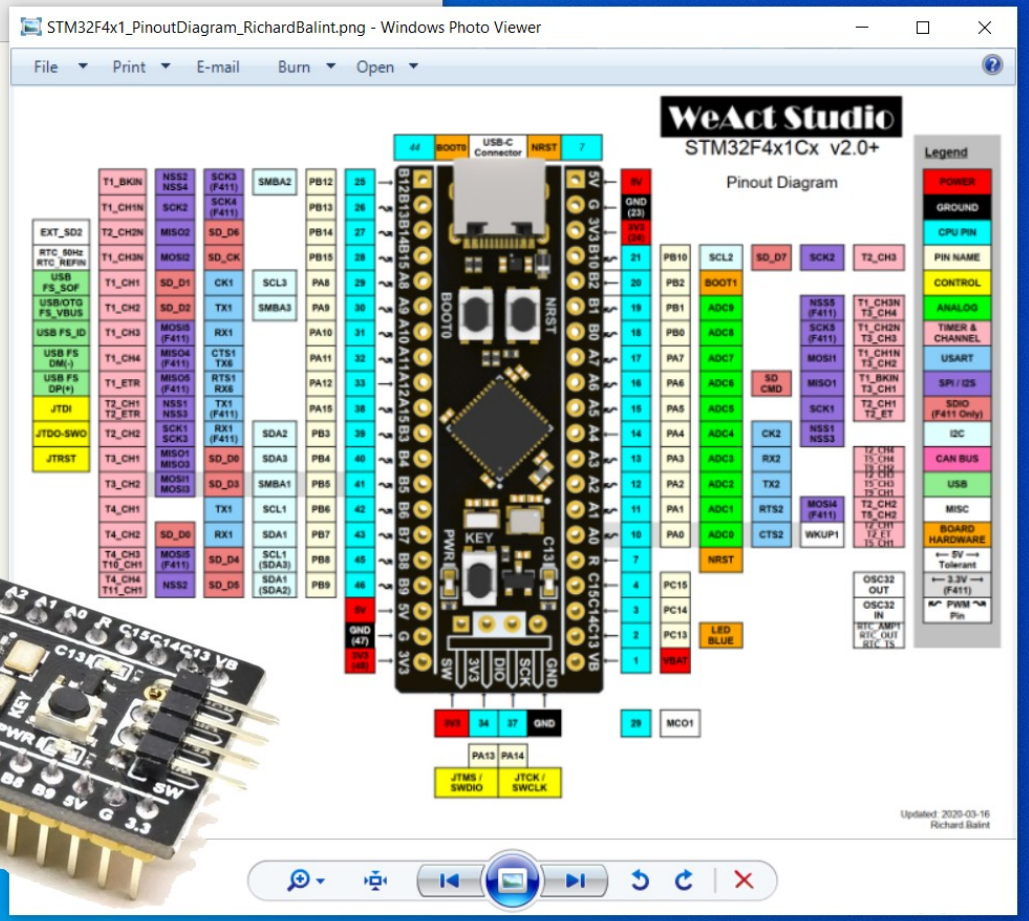
Mu 1.0.2 - ledblink.py

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)
```





ENG HU 13:27 2021. 12. 05.

7. LED mátrix és számkijelzők vezérlése

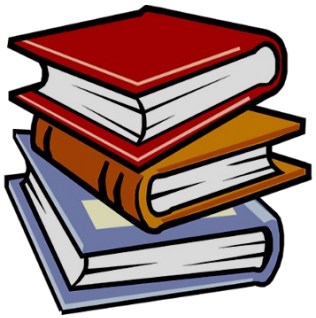
Felhasznált és ajánlott irodalom

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

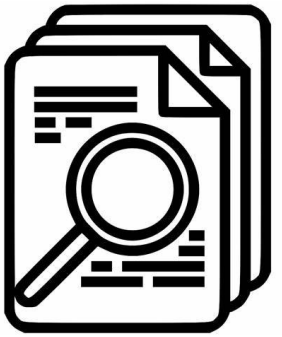
CircuitPython:

- Adafruit: https://circuitpython.org/downloads
- Adafruit: Adafruit CircuitPython API Reference
- Learn Adafruit: CircuitPython Essentials
- Adafruit: github.com/adafruit/Adafruit CircuitPython Bundle

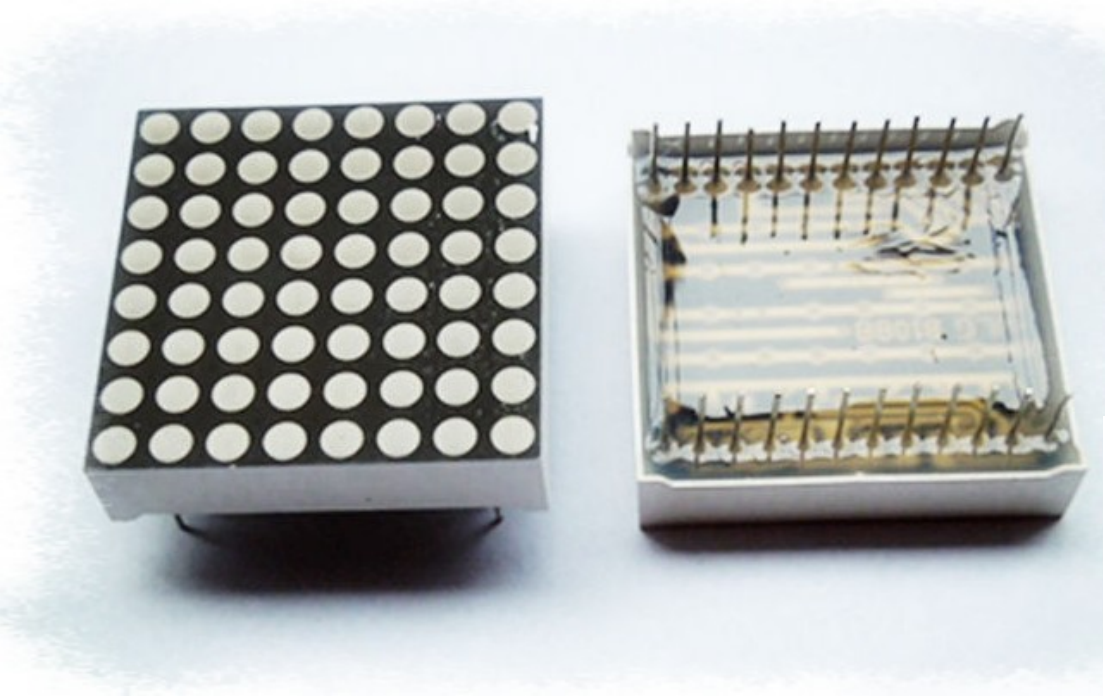


Adatlapok és dokumentáció:

- **MAX7219:** adatlap és termékinfo
- **TM1637:** adatlap
- **TM1638:** adatlap
- **STM32F411CE** adatlap és termékinfo
- **STM32F411xC/E** Family Reference Manual
- **WeAct Studio:** STM32F4x1 MiniF4

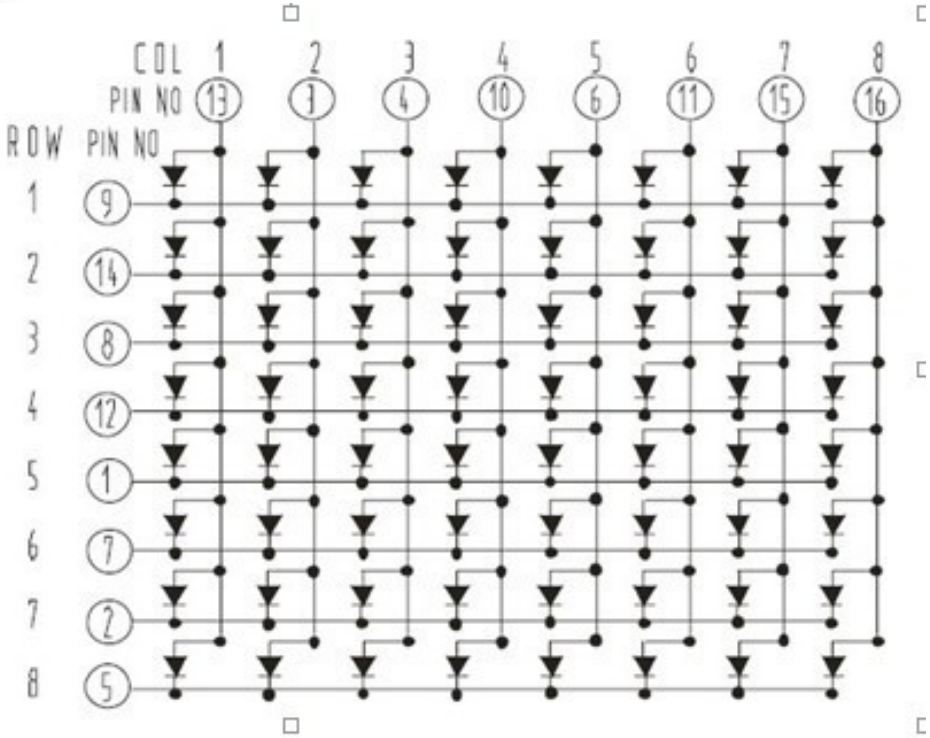


8x8 LED mátrix



3 mm-es piros LED-ek 8x8-as mátrixba szervezve

A **1088AS** típusnál a sorkiválasztó vonalak a katódokat közösítik



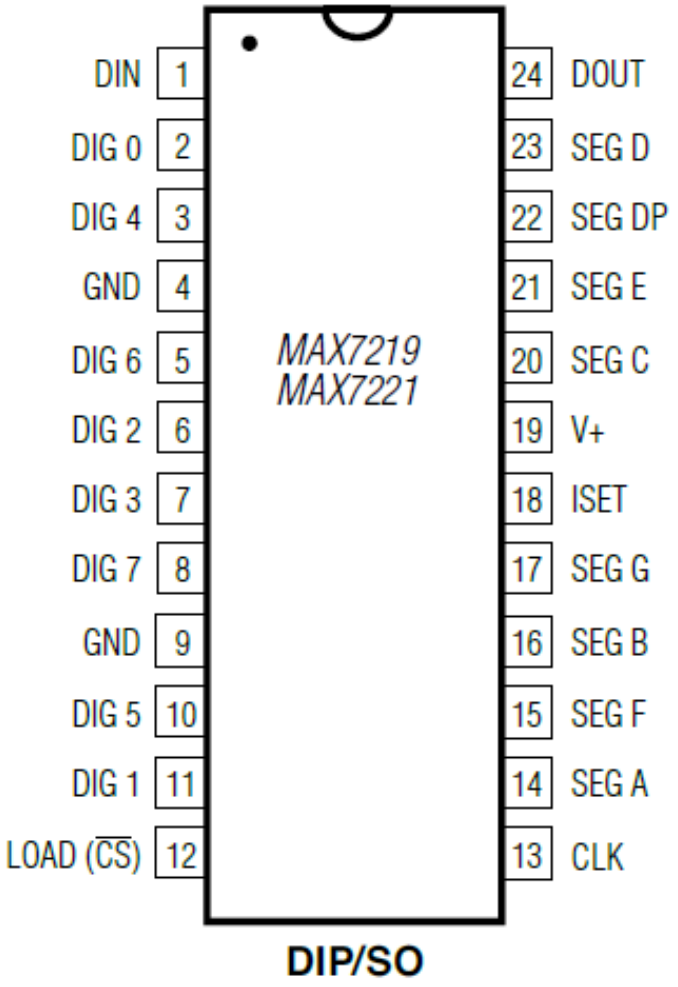
Multiplex kijelzés, egyidejűleg legfeljebb egy sor, vagy egy oszlop lehet aktív.

Kényelmes meghajtás:

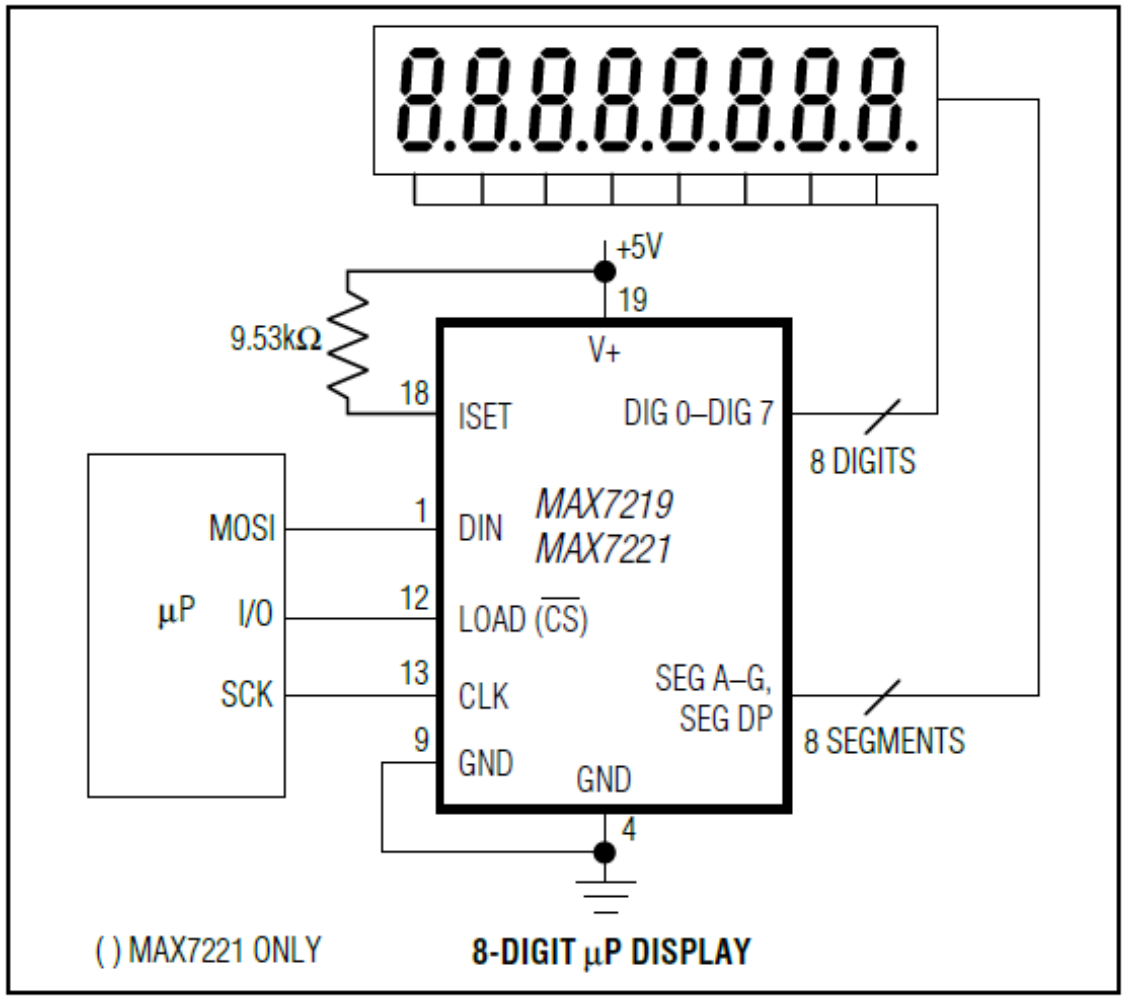
- 1 db **MAX7219**, vagy
- 2 db **74HC595** (+ meghajtó +áramkorlátozás)
- 1 db **MCP23017** (+ meghajtó +áramkorlátozás)

MAX7219

LED meghajtó IC, beépített áramkorlátozással. 7 szegmens kijelző esetén 1-8 db számjegy meghajtása (opcionálisan beépített dekódolással), vagy 8x8 LED mártix meghajtása. Az IC vezérlése SPI buszon történhet.

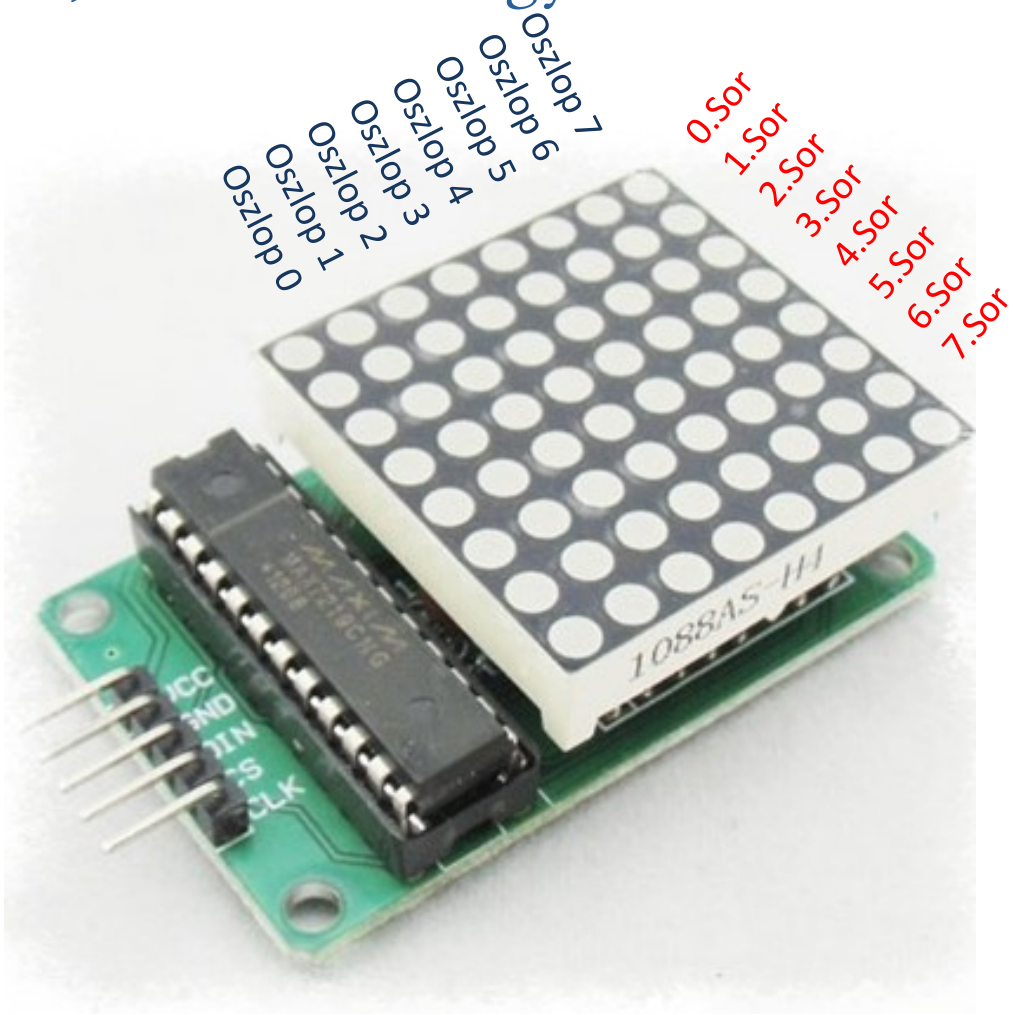


DIP/SO



Kijelző modul MAX7219 IC-vel

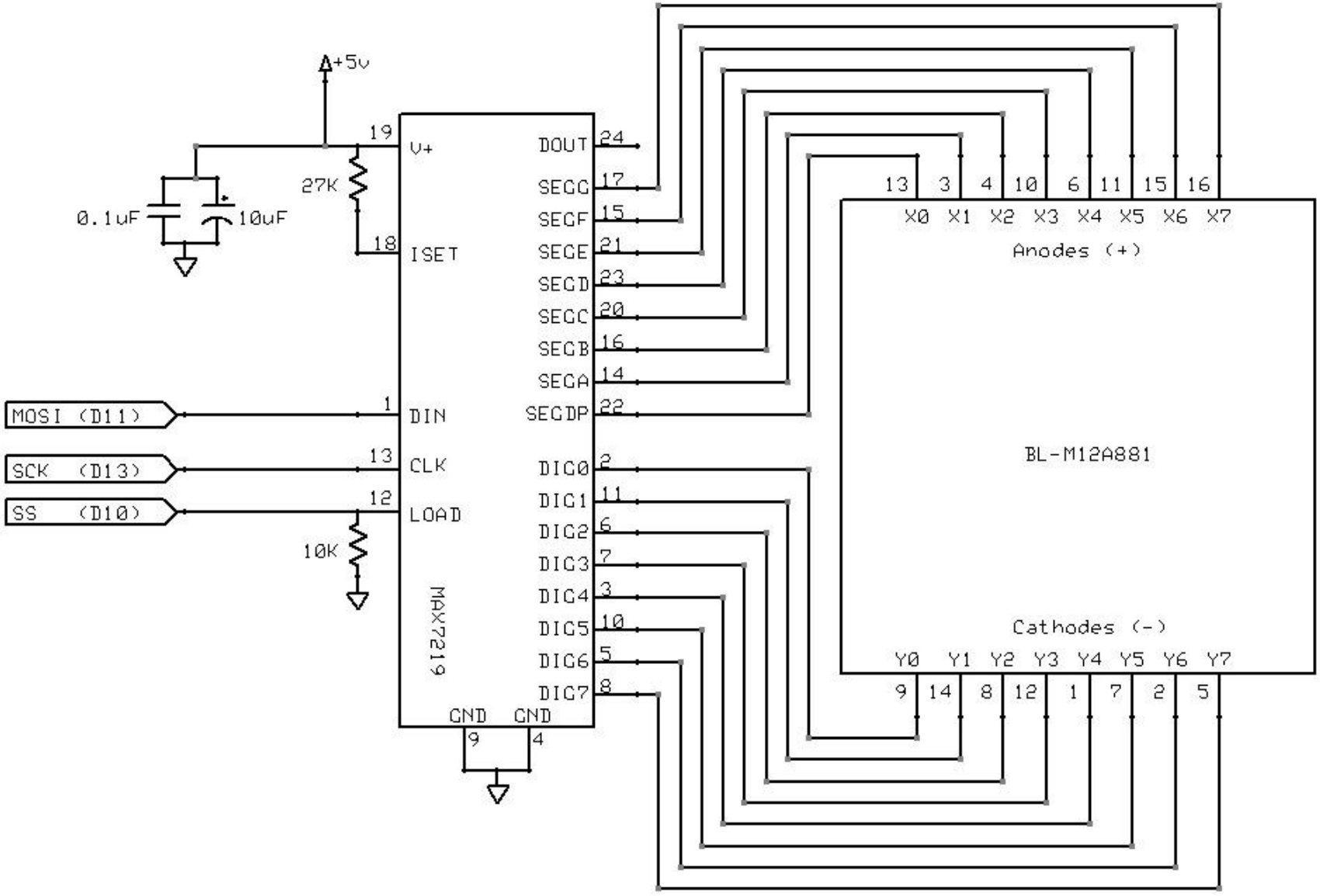
- Az E-bay kínálatában kapható, furatszerelt vagy felületszerelt kivitelben
  - 8x8 LED mátrix
  - MAX7219 vezérlő
  - Felfűzhető kivitel
  - Tápellátás: 3,5 – 5 V



Bemenetek	**Kimenetek**
1 VCC	1 VCC
2 GND	2 GND
3 DIN	3 DOUT
4 CS	4 CS
5 CLK	5 CLK

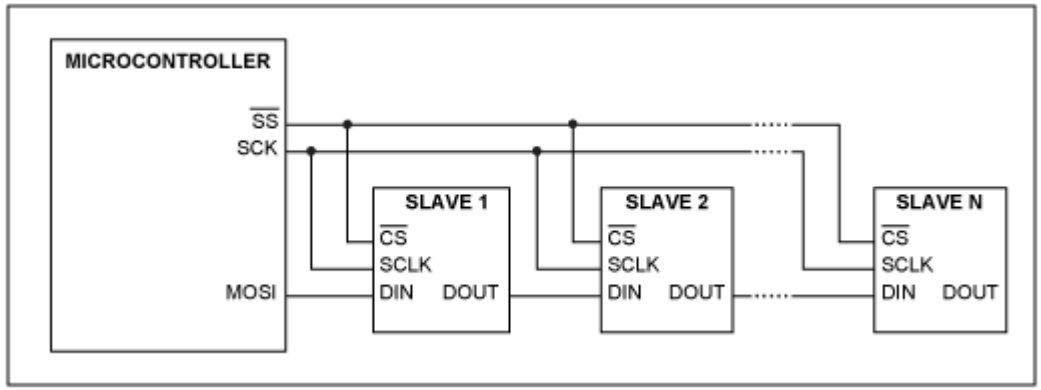
- **DIN/DOUT** – soros adat, **CLK** – szinkron órajel, **CS** – eszközkiválasztó jel, **VCC** – tápfeszültség, **GND** – a tápegység közös pontja („föld”)

Kapcsolási rajz

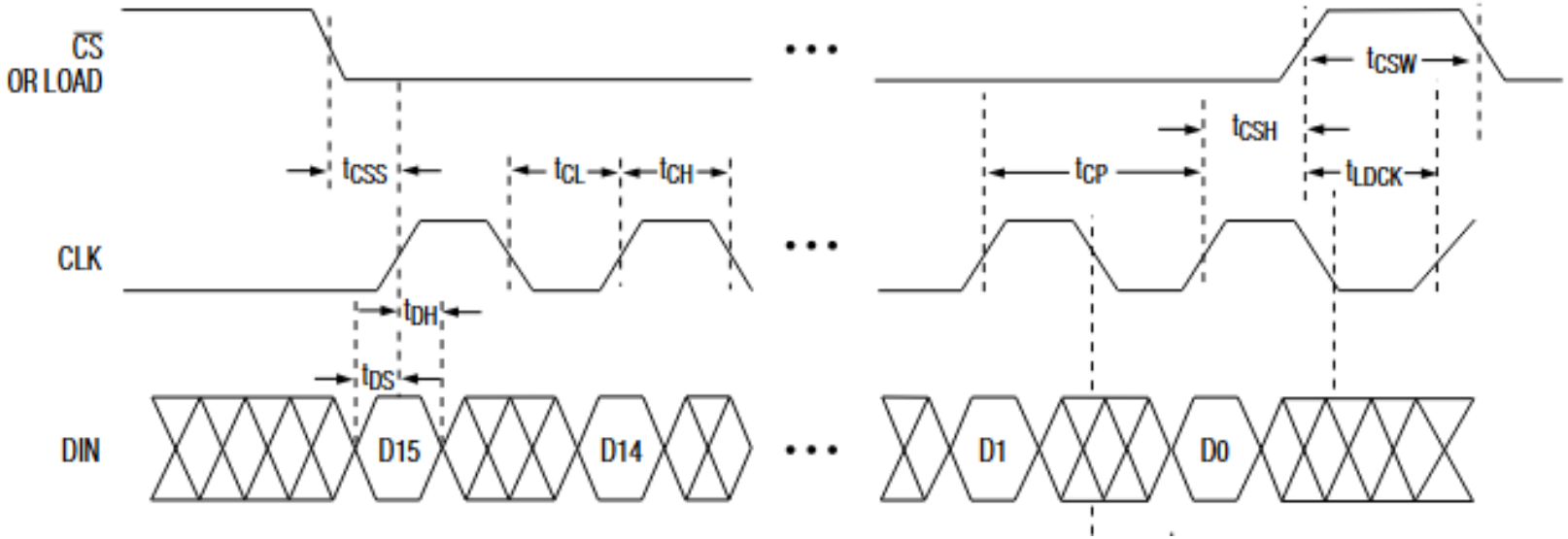


Hogy működik az SPI adatküldés?

- **SPI** – soros periféria illesztőt jelent, ahol az adatbitek nem több vezetéken párhuzamosan, hanem egy adatvezetéken, sorban, egymás után haladnak



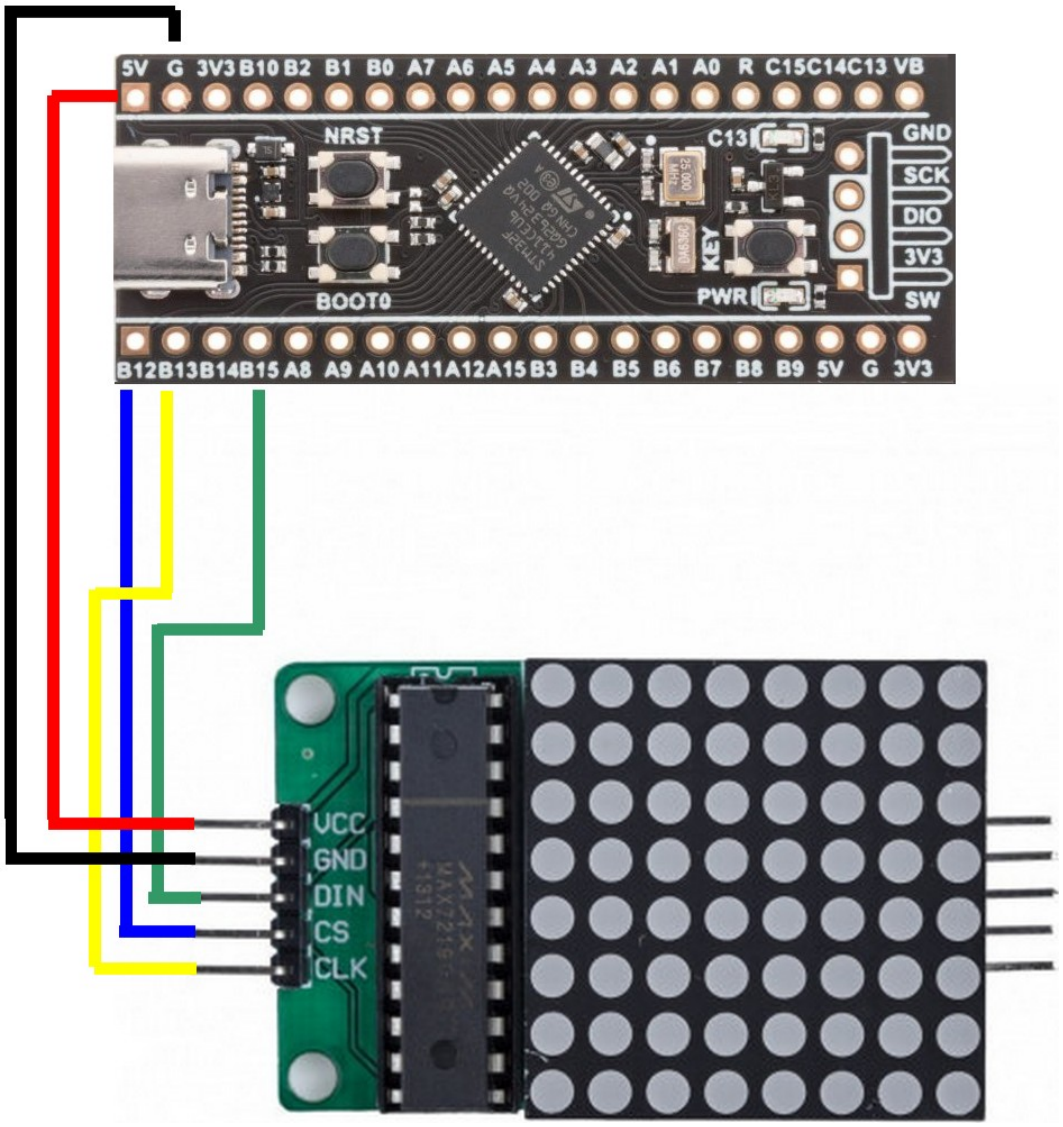
- Az adatbitek szinkronizálását egy külön vezetéken kiküldött órajel impulzusok végzik, a tranzakció szinkronizálását pedig a **LOAD** (**CS**) jel végzi (felfutó élére íródik be a 16 bites adat)



Bekötési rajz

- Az **STM32F411CE** mikrovezérlő **SPI2** csatornáját használjuk
- A tápfeszültség hivatalosan 4 – 5 V, de a tapasztalat szerint 3,3 V-tal is működőképes

MAX7219	STM32F411CE
VCC	5V vagy 3V3
GND	GND
DIN	B15 (MOSI2)
CS	B12 (NSS2)
CLK	B13 (SCK2)



Technikai részletek az inicializáláshoz

Table 2. Register Address Map

REGISTER	ADDRESS					HEX CODE
	D15–D12	D11	D10	D9	D8	
No-Op	X	0	0	0	0	0xX0
Digit 0	X	0	0	0	1	0xX1
Digit 1	X	0	0	1	0	0xX2
Digit 2	X	0	0	1	1	0xX3
Digit 3	X	0	1	0	0	0xX4
Digit 4	X	0	1	0	1	0xX5
Digit 5	X	0	1	1	0	0xX6
Digit 6	X	0	1	1	1	0xX7
Digit 7	X	1	0	0	0	0xX8
Decode Mode	X	1	0	0	1	0xX9
Intensity	X	1	0	1	0	0xXA
Scan Limit	X	1	0	1	1	0xXB
Shutdown	X	1	1	0	0	0xXC
Display Test	X	1	1	1	1	0xXF

Adatbájt

Nem használ adatot
Szegmensvezérlő bitek
. . .

. . .
Szegmensvezérlő bitek

0: no decode 1: decode

0 – 0xF

0 – 7

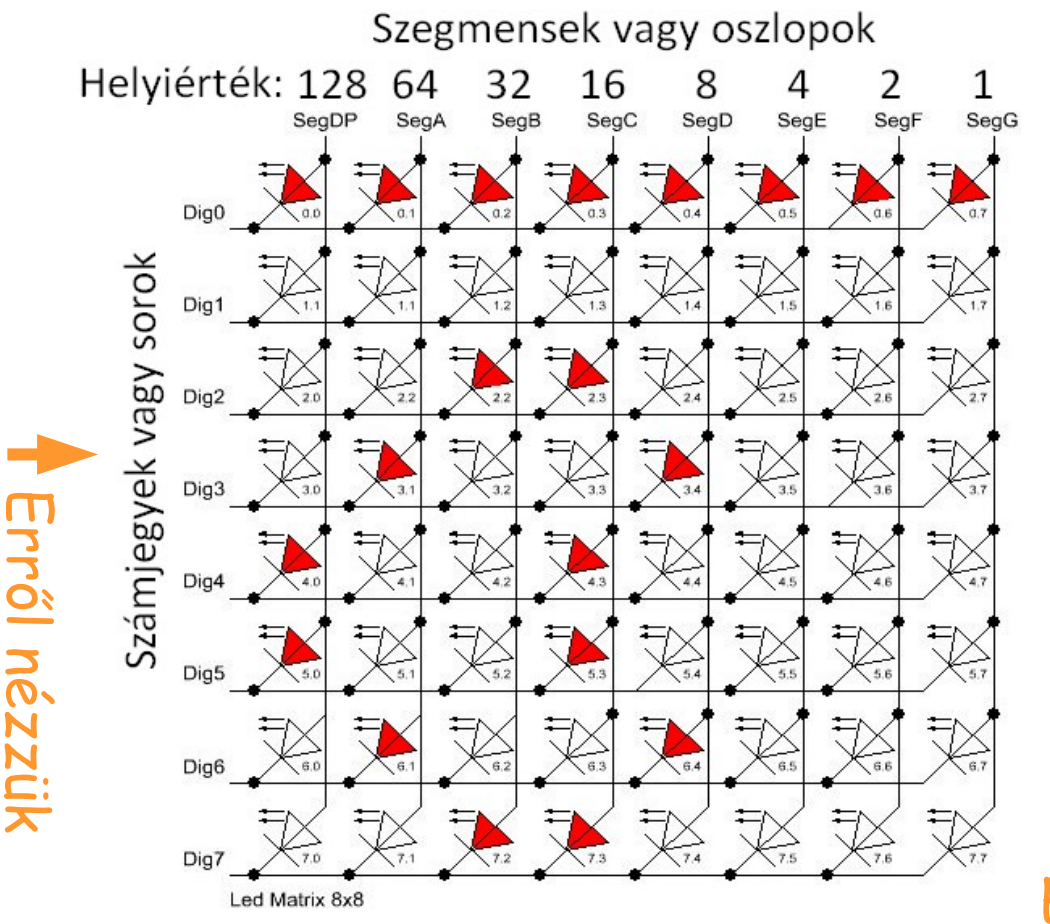
0: shutdown 1: normal mode

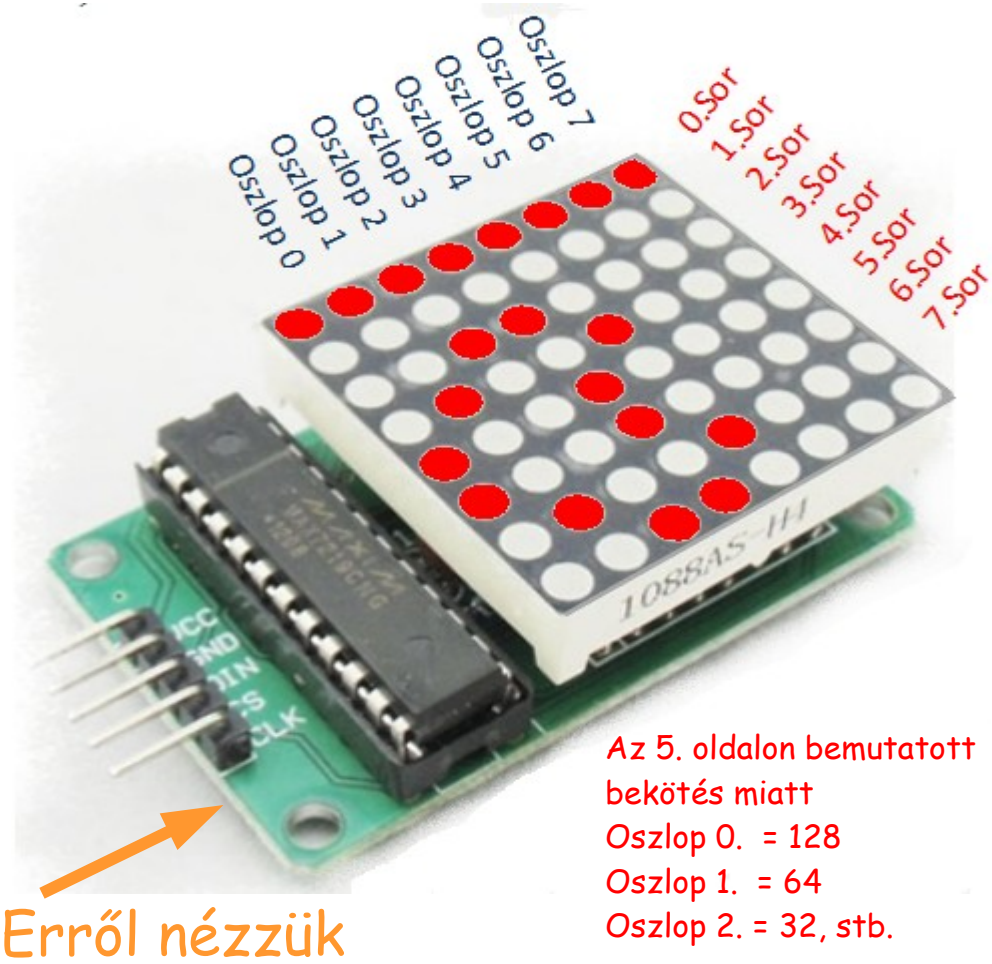
1: test mode 0: normal mode

Technikai részletek a kiíráshoz

Sor	Bin	Hex	Sor	Bin	Hex
0	11111111	0xFF	4	10010000	0x90
1	00000000	0x00	5	10010000	0x90
2	00110000	0x30	6	01001000	0x48
3	01001000	0x48	7	00110000	0x30

```
minta = [0xFF, 0x00,
    0x30, 0x48,
    0x90, 0x90,
    0x48, 0x30 ]   # I♥ mintázat
```





Az 5. oldalon bemutatott bekötés miatt
Oszlop 0. = 128
Oszlop 1. = 64
Oszlop 2. = 32, stb.

Adafruit MAX7219 programkönyvtár

- Az **Adafruiut CircuitPython Bundle** gyűjtemény része az **Adafruit CircrcuitPython MAX7219** programkönyvtár, amely három részből áll:
  - **max7219.py** – a MAX7219 IC inicializálására és kezelésére
  - **matrices.py** – MAX7219 osztály leszármazott, LED mátrix kezeléshez
    matrices.Matrix8x8 osztály: egy 8x8-as LED mátrix vezérlése
    matrices.CustomMatrix osztály: több, összekapcsolt 8x8-as LED vezérlése
  - **bcddigits.py** – MAX7219 osztály leszármazott, hétszegmens kijelzőkhöz
    bcddigits.BCDDigits osztály: max. 8-digites hétszegmenses kijelzőhöz
- **Függőség:** telepíteni kell az **adafruit_framebuffer.py** könyvtárat és a **font5x8.bin** állományt is
- Leírás: MAX7219 LED Matrix/Digit Display Driver
- Mintaprogramok:
  max7219_simpletest.py és max7219_showbcddigits.py

max7219.MAX7219 osztály

A **max7219.MAX7219** osztály tagfüggvényei:

- **init_display()** – a vezérlő és a kijelző inicializálása (virt. fv.)
- **brightness()** – fényerő szabályozása (0 – 15)
- **show()** – a buffer tartalmának megjelenítése a kijelzőn
- **fill**(*bit_value*) – a buffer feltöltése a megadott értékkel (0/1)
- **pixel**(*x, y, bit_value*) – a megcímzett képpont beállítása
- **scroll**(dx, dy) – a kép görgetése a megadott lépéssel (ha előtte nem töröljük a szélső sort, akkor léptetéskor duplikákódik!)
- **write_cmd**(*cmd, data*) – egy parancs kiküldése: a *cmd*-vel megadott sorszámú regiszter tartalma *data* lesz)

matrices.Matrix8x8 osztály

A **matrices.Matrix8x8** osztály tagfüggvényei:

- **init_display()** – a vezérlő és a kijelző inicializálása mátrix kijelzéshez (kikapcsolja a dekódolást)
- **clear_all()** – törli mindegyik pixelt a bufferben
- **text(***str, xpos, ypos, bit_value = 1***)** – szöveg megjelenítése
  str – a megadott szöveg
  xpos, ypos – a kiírás kezdő pozíciója
  bit_value – 0: inverz kiírás, 1: normál kiírás

bcddigits.BCDDigits osztály

A **bcddigits.BCDDigits** osztály tagfüggvényei:

- **init_display()** – a vezérlő és a kijelző inicializálása számok kijelzéséhez (bekapcsolja a BCD⟶7SEG dekódolást)
- **set_digit(***dpos, value***)** – egy számjegy (0-15) kiírása *pos* pozícióba
- **set_digits(***dpos, values***)** – számjegyek (0-15) kiírása *pos* pozíciótól, ahol *values* egy lista, elemei 0-15 közötti egész számok
- **show_dot**(*dpos, bit_value*) – tizedespont megjelenítése (1), vagy törlése (0) a megadott pozíción
- **clear_all()** – törli mindegyik számot és tizedespontot a bufferben
- **show_str**(*pos, str*) – numerikus karakterfüzér (0-9, - és .) kiírása
- **show_help**(*pos*) – HELP felirat kiírása adott pozíciótól

matrix8x8_simpletest.py

- A **matrix8x8_simpletest.py** mintaprogram az **Adafruit MAX7219** programkönyvtár **max7219_simpletest.py** mintapéldájának az *„STM32F411CE blackpill with flash”* kártyára adaptált változata
- Az adatküldéshez az **SPI2** csatornát használjuk a **busio.SPI** objektumosztály példányosításával
- A **CE** jel vezérléséhez ezt ki kell egészítenünk egy digitális kimenettel, melyhez a **board.B12** kivezetést használjuk fel
- A program inicializálás után az alábbi műveleteket ismételgeti:
  - Az összes LED kigyújtása, majd leoltása
  - Egy pixelsor kigyújtása és jobbra görgetése
  - Az ’Adafruit’ szöveg megjelenítése karakterenként
  - A legutoljára kiírt karakter kigörgetése jobbról balra a kijelzőről
  - Az ’Adafruit’ szöveg görgetése fényúság szerűen
- Az alábbi programlista helyett tanulságosabb lehet a **MAX7219_led8x8.ipynb** JUPYTER notebookot nézegetni

matrix8x8_simpletest.py - 2/1.

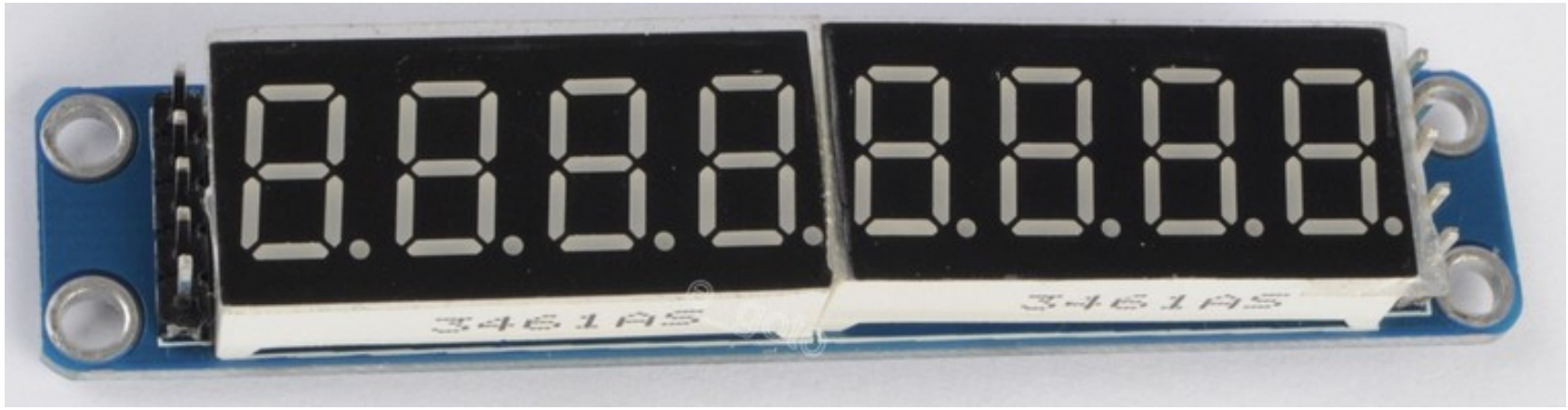
```
import time
import board
import digitalio
import busio
from adafruit_max7219 import matrices
spi = busio.SPI(board.B13, MOSI=board.B15)
cs = digitalio.DigitalInOut(board.B12)
matrix = matrices.Matrix8x8(spi, cs)
while True:
    print("Cycle start")
    # all lit up
    matrix.fill(True)
    matrix.show()
    time.sleep(0.5)

    # all off
    matrix.fill(False)
    matrix.show()
    time.sleep(0.5)

    # one column of leds lit
    for i in range(8):
        matrix.pixel(1, i, 1)
    matrix.show()
    time.sleep(0.5)
```

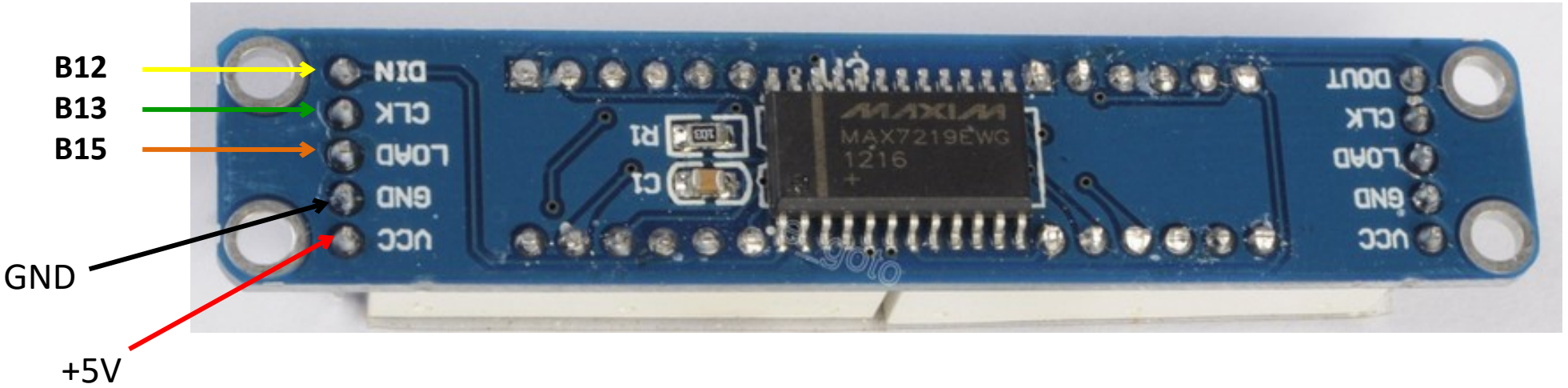
matrix8x8_simpletest.py - 2/2.

```
# now scroll the column to the right
for j in range(8):
    matrix.scroll(1, 0)
    matrix.show()
    time.sleep(0.5)

# show a string one character at a time
adafruit = "Adafruit"
for char in adafruit:
    matrix.fill(0)
    matrix.text(char, 0, 0)
    matrix.show()
    print(char)
    time.sleep(1.0)

# scroll the last character off the display
for i in range(8):
    matrix.scroll(-1, 0)
    matrix.show()
    time.sleep(0.5)

# scroll a string across the display
for pixel_position in range(len(adafruit) * 8):
    matrix.fill(0)
    matrix.text(adafruit, -pixel_position, 0)
    matrix.show()
    time.sleep(0.25)
```

8 számjegyű hétszegmenses kijelző vezérlése

Így van bekötve: dig7 dig6 dig5 dig4 dig3 dig2 dig1 dig0

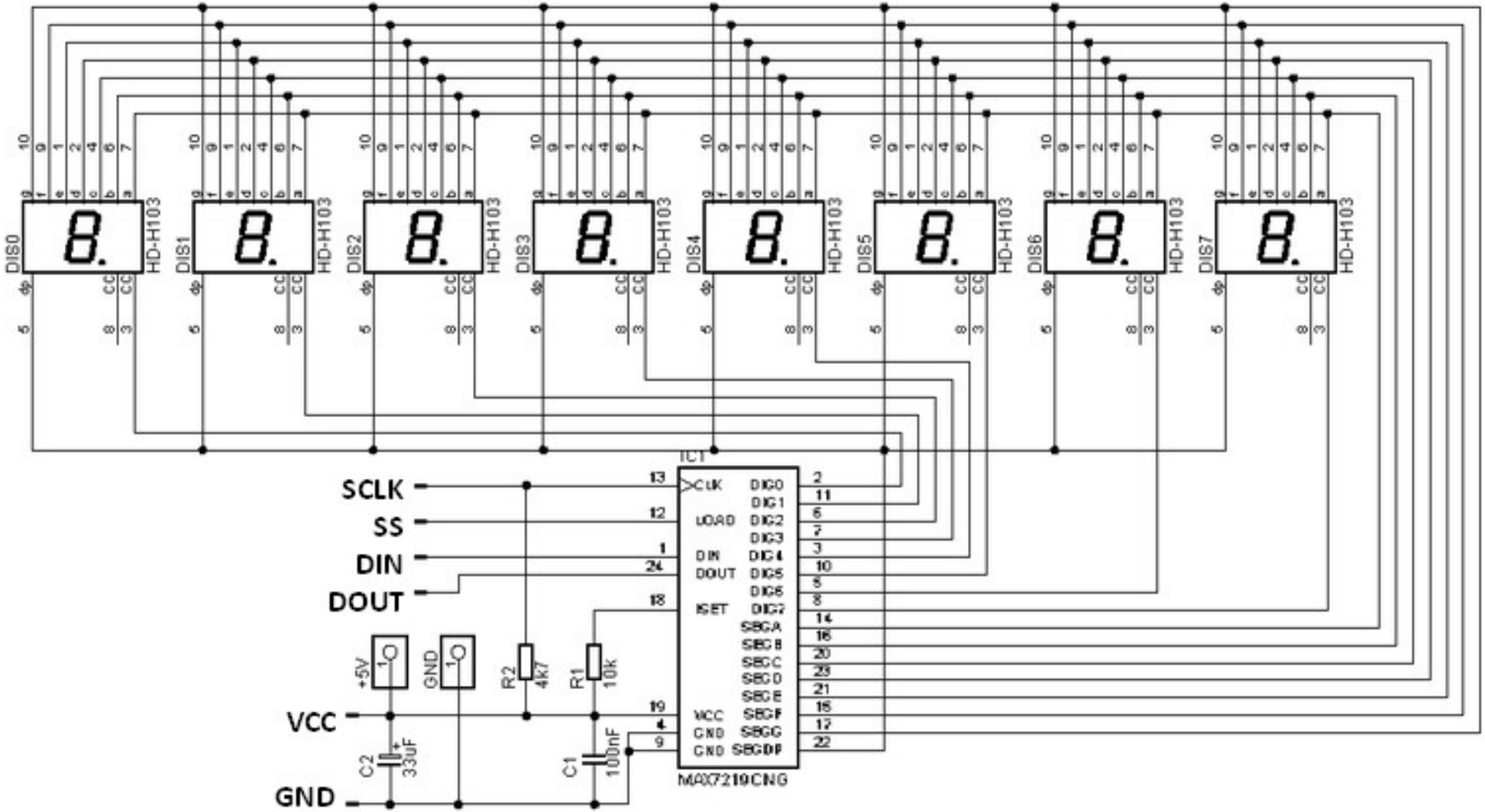




Bekötési vázlat a **max7219_showbcddigits.py** példaprogramhoz és **STM32F411CE** blackpill with flash kártyához (SPI2 csatorna)

8 számjegyű hétszegmenses kijelző vezérlése

7 szegmens + DP vezérlése 8 db közös vezetéken, számjegykiválasztás másik 8 db vezetéken.
MAX7219 SPI interfésszel rendelkezik. Beépített LED áramkorlátozás és fényerőszabályozás .



max7219_showbcddigits.py

- A **max7219_showbcddigits.py** mintaprogram az **Adafruit MAX7219** programkönyvtár **max7219_showbcddigits.py** mintapéldájának az *„STM32F411CE blackpill with flash”* kártyára adaptált változata
- Az adatküldéshez az **SPI2** csatornát használjuk a **busio.SPI** objektumosztály példányosításával
- A **CE** jel vezérléséhez ezt ki kell egészítenünk egy digitális kimenettel, melyhez a **board.B12** kivezetést használjuk fel
- A program inicializálás után az alábbi műveleteket ismételgeti:
  - Az 12 345 678 szám kiírása és a fényerőszabályozás bemutatása
  - A "-HELP-90" felirat megjelenítése
  - Két tizedespontot is tartalmazó szám kiírása (72.5 és –10.8)
  - A HELP szöveg kiíratása
  - Négyjegyű, nyolcjegyű és véletelen számok kiíratása
- Az alábbi programlista helyett tanulságosabb lehet a **max7219_showbcddigits.py** JUPYTER notebookot nézegetni

max7219_showbcddigits.py – 3/1.

```
import time
import board
import digitalio
import busio
import random
from adafruit_max7219 import bcddigits

spi = busio.SPI(board.B13, MOSI=board.B15)
cs = digitalio.DigitalInOut(board.B12)
leds = bcddigits.BCDDigits(spi, cs, nDigits=8)

while True:
    # clear display and dim 0
    leds.brightness(0)
    leds.clear_all()

    # place 8-digit number on display
    value = 12345678
    leds.show_str(0, "{:8}".format(value))
    leds.show()

    # increase the brightness slowly
    for i in range(16):
        leds.brightness(i)
        time.sleep(0.5)
```

max7219_showbcddigits.py – 3/2.

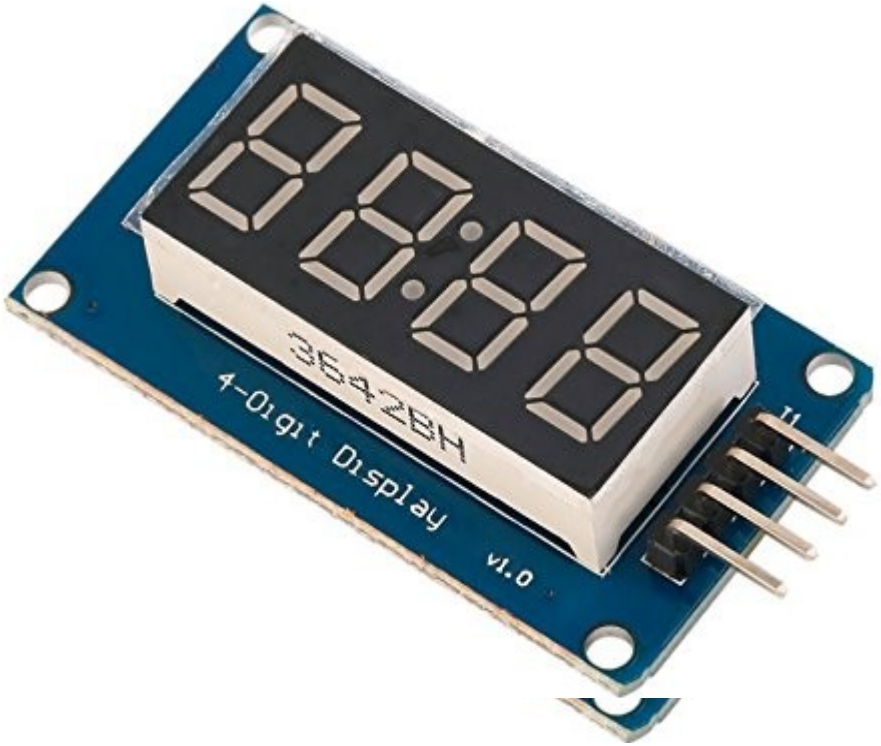
```
leds.brightness(5)
# show "-HELP-90" on display
leds.show_str(6, "90")  # show 90 starting at position 6
leds.set_digit(0, 10)  # show - at position 0
leds.set_digit(1, 12)  # show H at position 1
leds.set_digit(2, 11)  # show E at position 2
leds.set_digit(3, 13)  # show L at position 3
leds.set_digit(4, 14)  # show P at position 4
leds.set_digit(5, 10)  # show - at position 5
leds.show()
time.sleep(1.0)

leds.clear_all()
# set the two dots and two 4-digit numbers
leds.show_dot(2, 1)
leds.show_dot(6, 1)
leds.show_str(0, " 72.5")
leds.show_str(4, "-10.8")
leds.show()
time.sleep(5.0)

# show the help string
leds.clear_all()
leds.show_help(2)
leds.show()
```

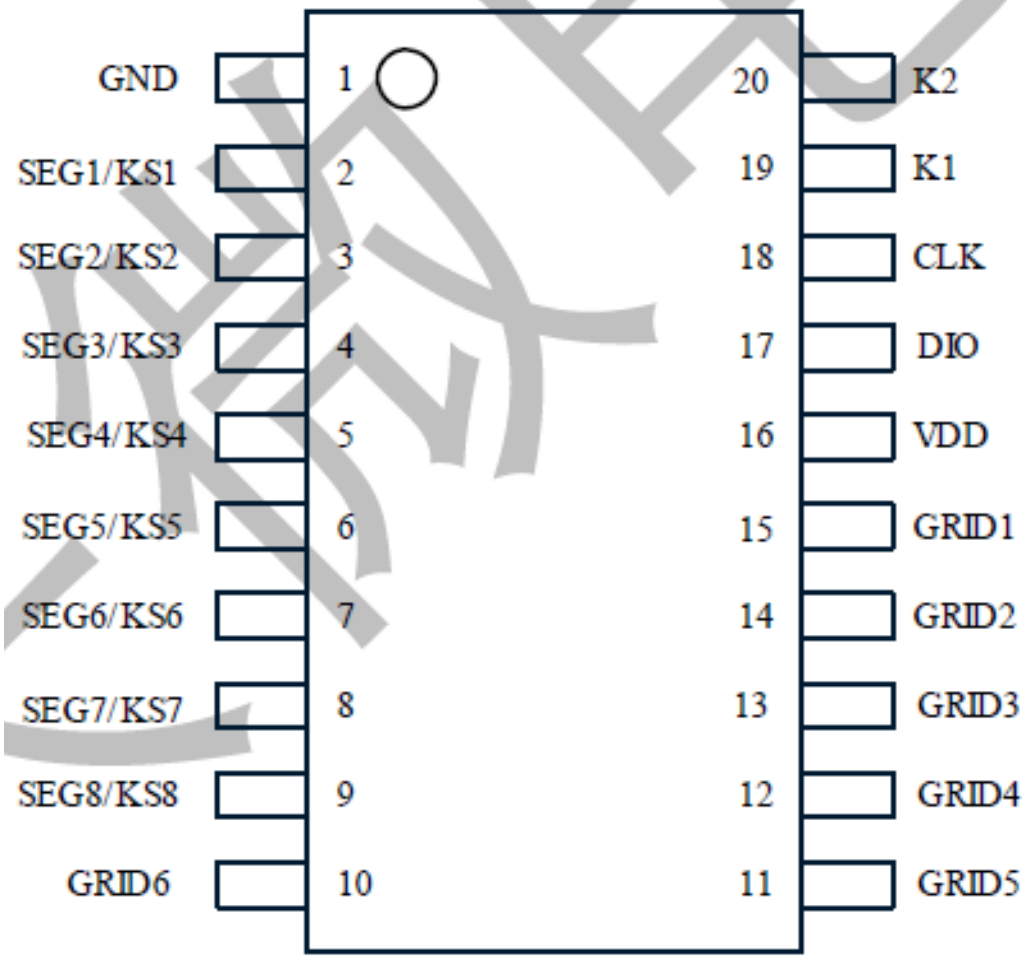
max7219_showbcddigits.py – 3/3.

```
leds.clear_all()
# show a 4 character numeric string
leds.show_str(0, "   0")
leds.show()
time.sleep(1.0)

leds.clear_all()
# show 0->8
for digit in range(8):
    leds.set_digit(digit, digit)
leds.show()
time.sleep(1.0)

# show random 8-digit numbers via show_str
for _ in range(10):
    number = random.uniform(-1.0, 1.0)
    number *= 10000.0
    number_string = "{:9.3f}".format(number)
    leds.clear_all()
    leds.show_str(0, number_string)
    leds.show()
    time.sleep(1.0)
```

Négyszámjegyű kijelző TM1637 vezérlővel

- Négyszámjegyű kijelző (közös anódú, **3642BH** típusú LED modul)
- Óra típusú kijelző (két pont középen)
- **TM1637** vezérlő kétvezetékes meghajtó (Titan Microelectronics)
- Részletes ismertető a 2019. május 2-i előadásban található
- 5 V tápfeszültség

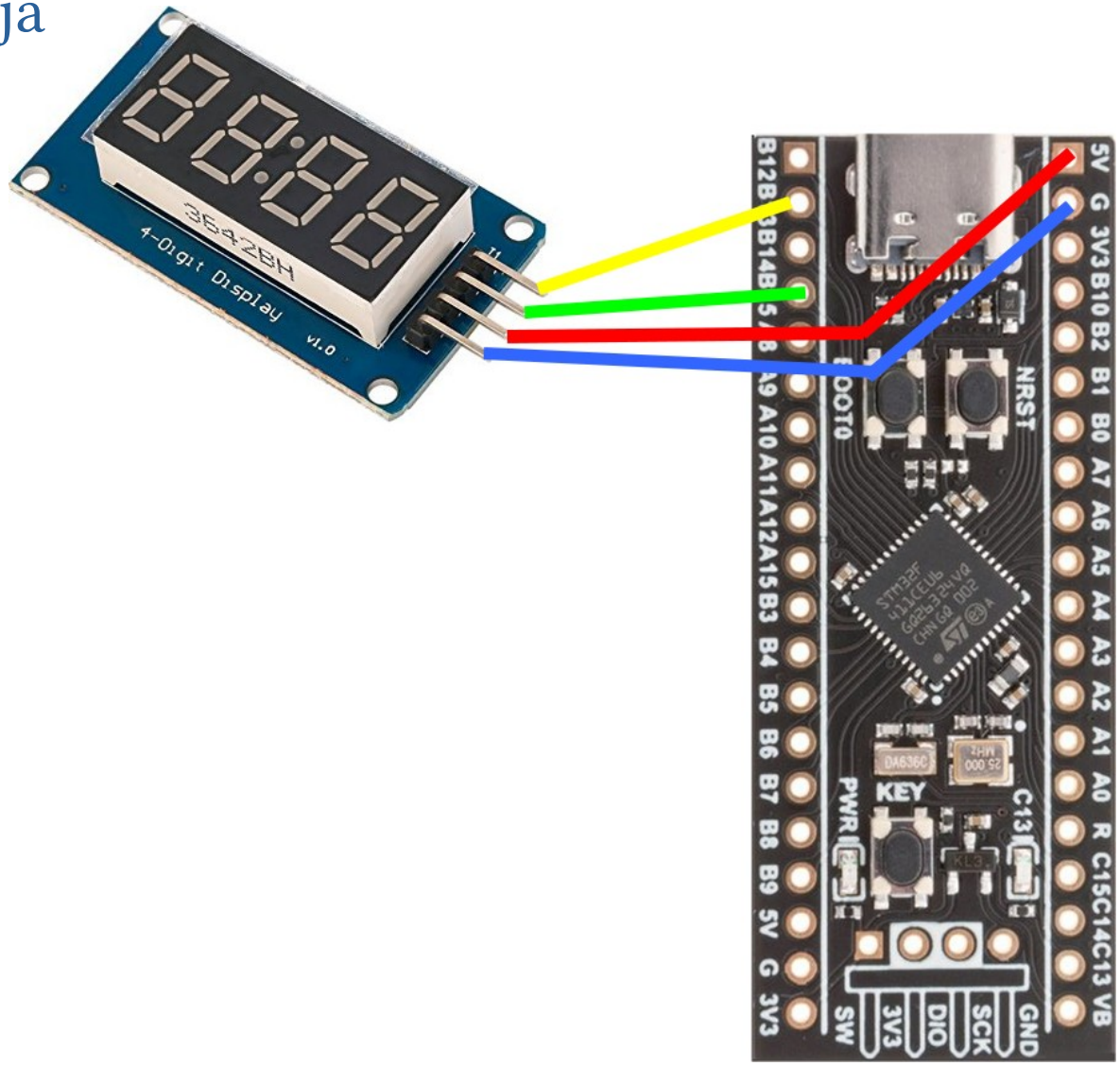




Bekötési vázlat

- A bekötésnél a softveres meghajtás miatt bármelyik digitális ki/bemenetet használhatjuk, de a bemutatott mintaprogram az alábbi bekötést használja

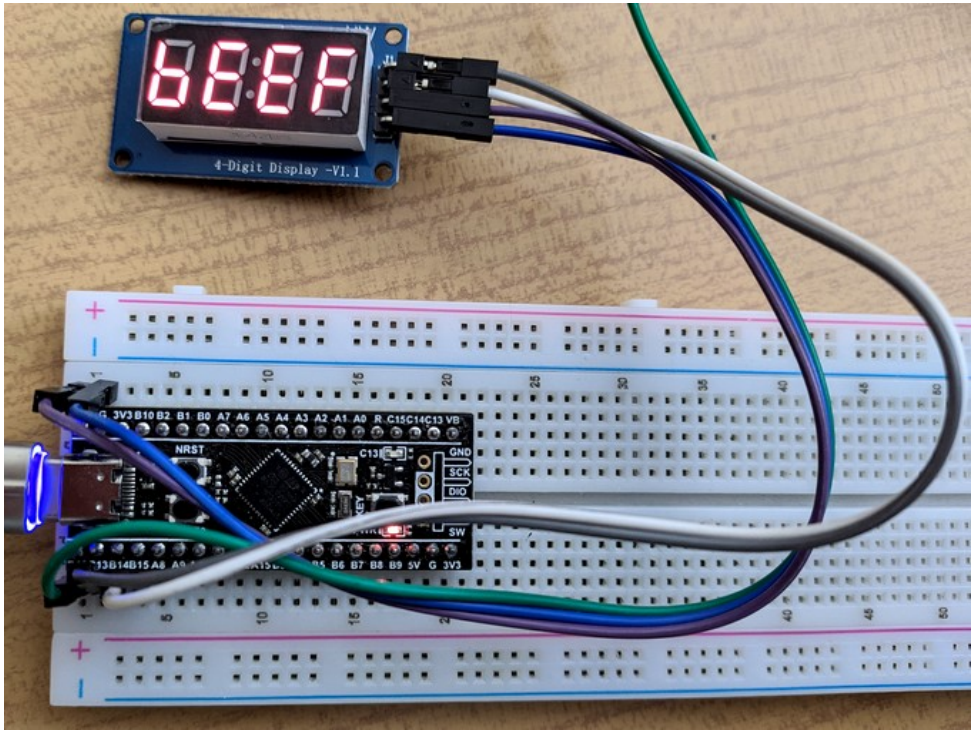
TM1637	STM32F411CE
VCC	5V vagy 3V3
GND	GND
DIO	B15
CLK	B13



tm1637_test.py

- Programkönyvtár: CircuitPython library für the TM1637 7-Segment Display
- Az alábbi egszerű mintapéldával csak a kijelző működését ellenőrizzük

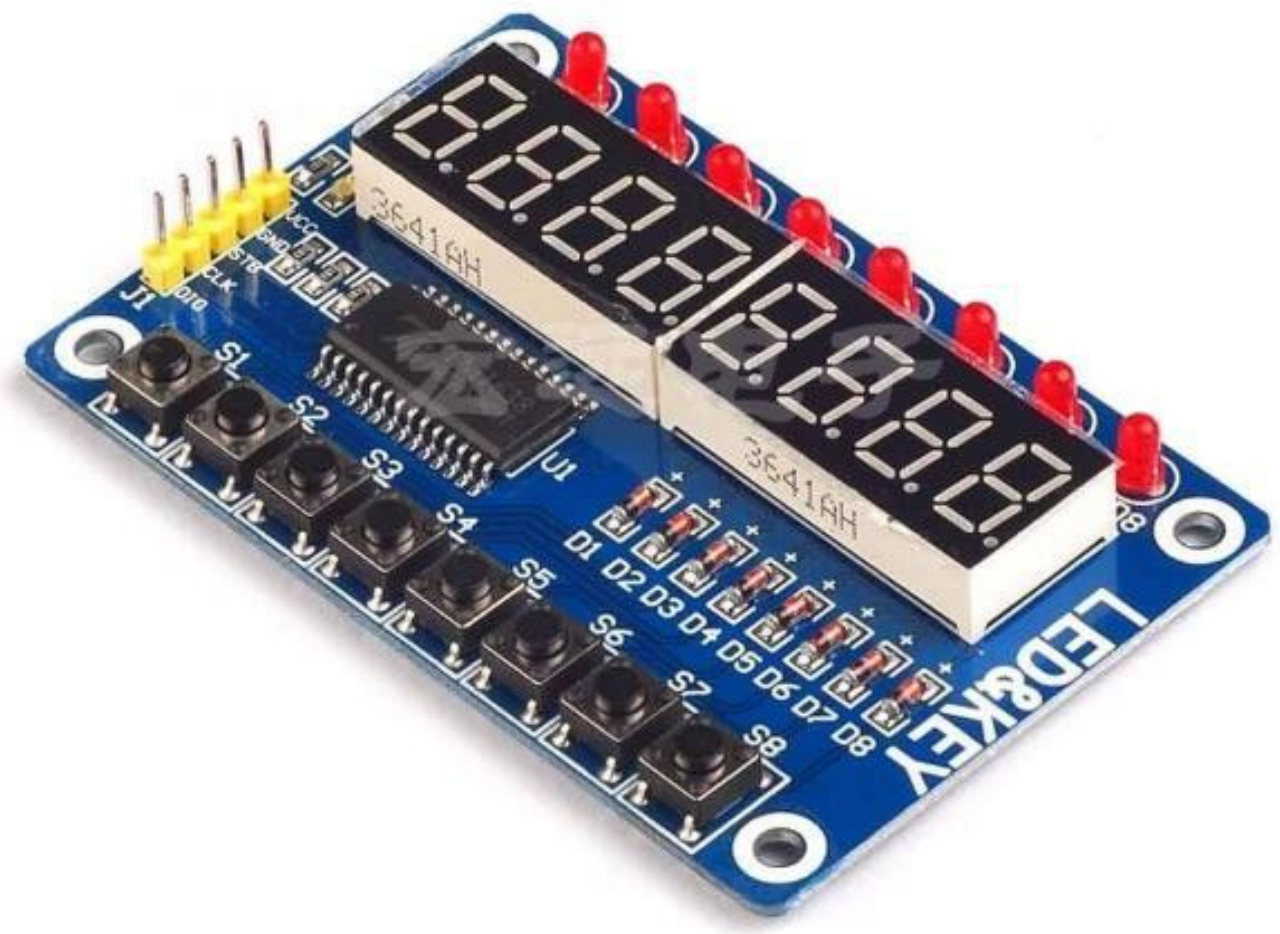
```
import time
import TM1637
import board


CLK = board.B13
DIO = board.B15
display = TM1637.TM1637(CLK, DIO)
display.hex(0xbeef)
time.sleep(5)
while True:
    pass
```



Nyolcszámjegyű kijelző TM1638 vezérlővel

- A LED & KEY kártya a továbbfejlesztett TM1638 vezérlőt használja, amely a 8-számjegyű 7-szegmenses kijelző mellett 8 db LED-et és 8 db nyomógombot is kezel
- Programkönyvtár: circuitpython-tm1638 (ez is szoftveres perifériakezelést végez)
- Bekötési vázlat:

TM1638	STM32F411CE
VCC	5V vagy 3V3
GND	GND
STB	B12
CLK	B13
DIO	B15



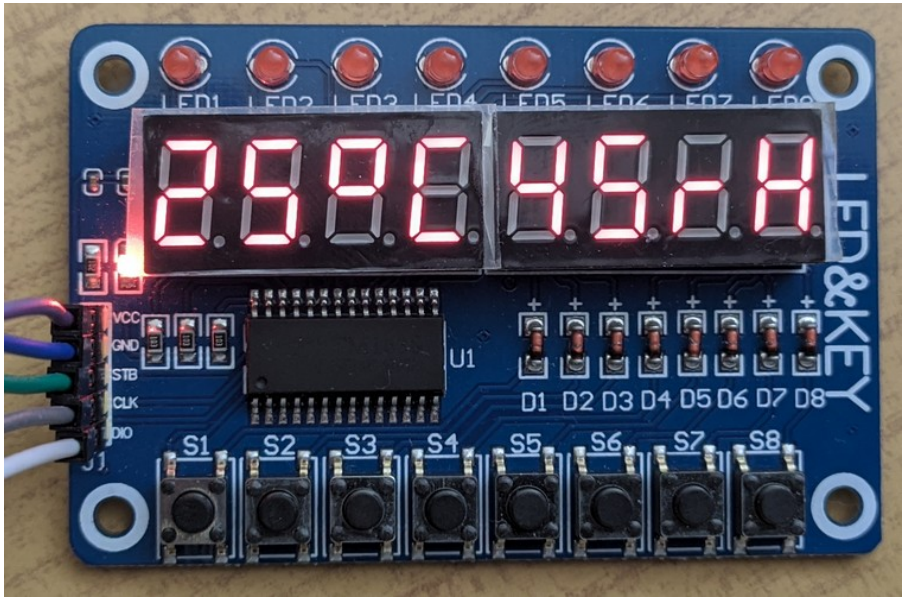
tm1638_test.py

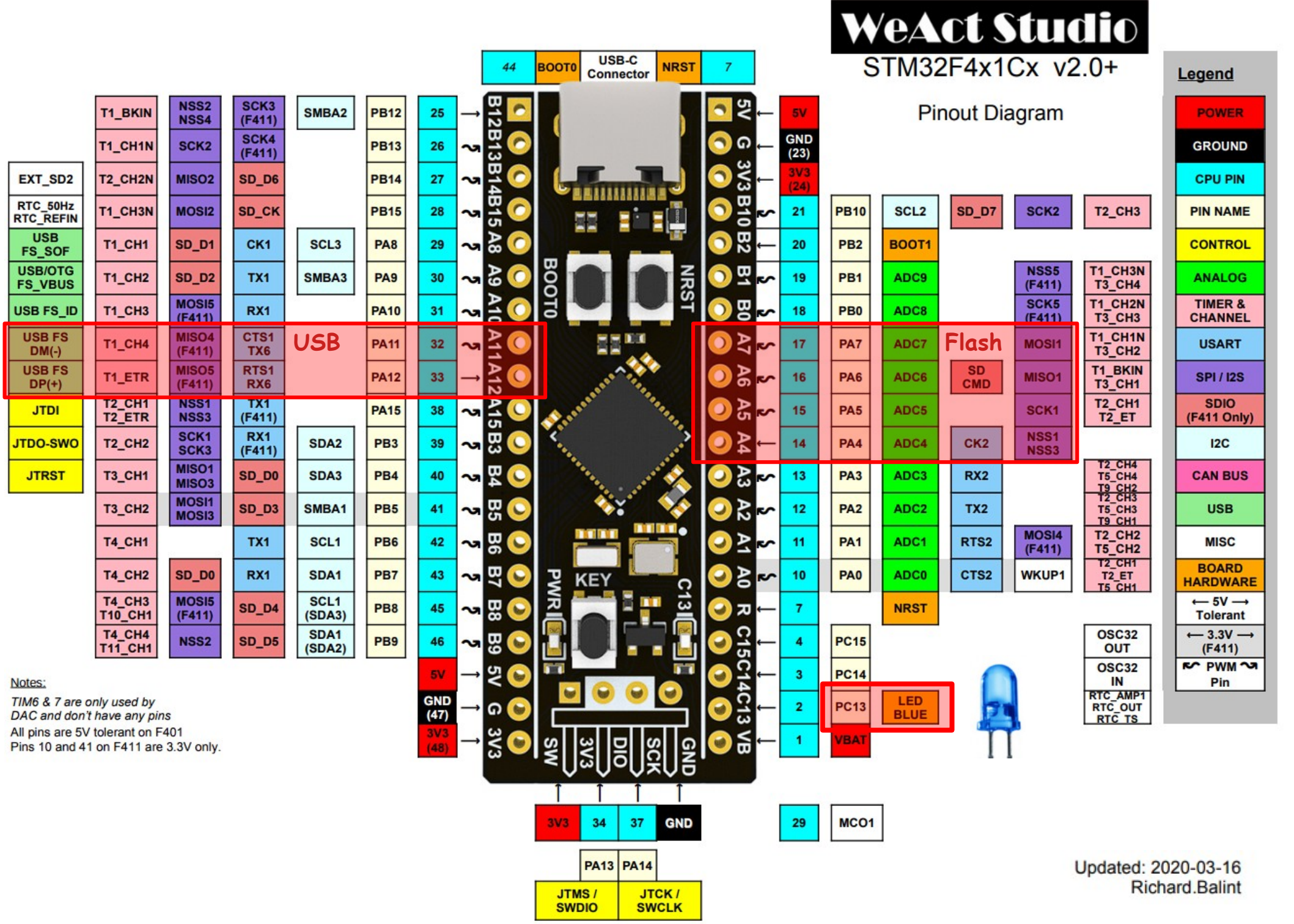
```
import tm1638
import time
import board
from digitalio import DigitalInOut, Direction

# TM1638 Keyboard & LED setup.
stb = DigitalInOut(board.B12)
stb.direction = Direction.OUTPUT
clk = DigitalInOut(board.B13)
clk.direction = Direction.OUTPUT
dio = DigitalInOut(board.B15)
dio.direction = Direction.OUTPUT

kled = tm1638.TM1638(stb, clk, dio, brightness=2)

# blink all leds
for i in range(8):
    kled.led(i, 1)
    time.sleep(0.5)
    kled.led(i, 0)

# display text
kled.show("PYTHON")
```

tm1638_test.py

```
# change brightness
for i in range(5):
    kled.brightness(i)
    time.sleep(0.5)
kled.brightness(2)
time.sleep(0.5)
kled.clear()

# display number
kled.number(-1234567)
time.sleep(1.5)
kled.clear()

# led combinations
kled.leds(170)
time.sleep(0.5)
kled.leds(85)
time.sleep(0.5)
kled.leds(170)
time.sleep(0.5)
kled.leds(85)
time.sleep(0.5)
kled.clear()
```

```
# display temperature and humidity
kled.temperature(25)
kled.humidity(45)
time.sleep(1.5)
kled.clear()

# wait for a key press and activate
# corresponding led
while True:
    kled.leds(kled.keys())
```




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
JTMS / SWDIO
JTCK / SWCLK
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