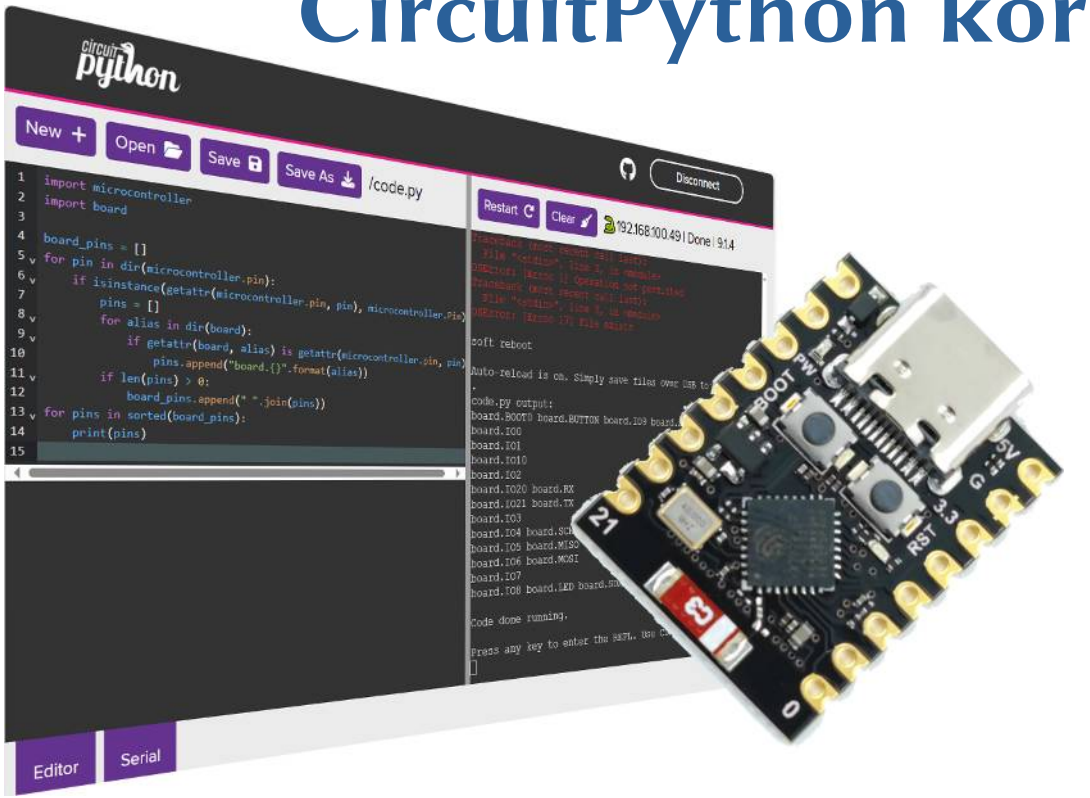
ESP32-C3 mikrovezérlők programozása CircuitPython környezetben



5. Digitális eszközök kezelése (1-wire, SPI)

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
  - Learn Adafruit: Welcome to CircuitPython
  - Learn Adafruit: CircuitPython Essentials
  - Adafruit: Adafruit CircuitPython API Reference
  - Adafruit: github.com/adafruit/Adafruit CircuitPython Bundle

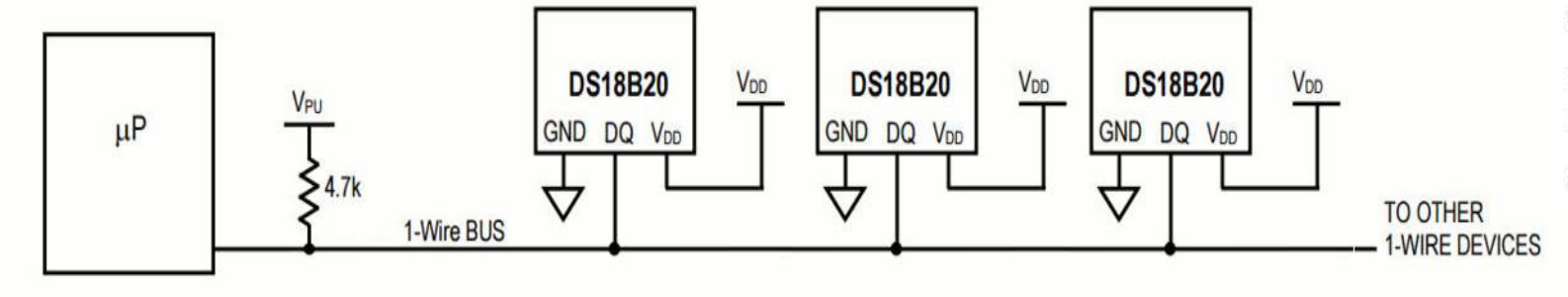
- **Online eszközök és támogatás:**
  - Learn Adafruit: CircuitPython on ESP32 Quick Start
  - Adafruit: Adafruit Web Serial ESPTool
  - Adafruit: CircuitPython Code Editor

A mai előadáshoz kapcsolódó segédanyagok

- **Adatlapok:**
- Analog Devices: DS18B20 adatlap
- Sitronix: ST7585 display controller
-
- **Programkönyvtárak:**
  - Adafruit_DS18x20 documentation
  - Adafruit_OneWire documentation
  - Adafruit_framebuf documentation
  - Adafruit_PCD8544 documentation

Az 1-wire busz

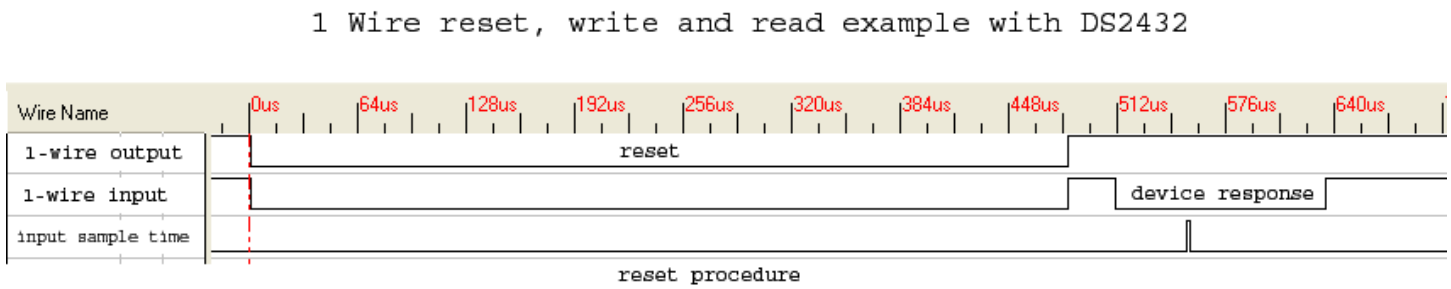
- A **1-Wire** a **Dallas Semiconductor** által tervezett vezetékes, fél-duplex soros busz, amely alacsony sebességű adatkommunikációt (16,3 kbit/s) biztosít egyetlen vezetéken keresztül. Tipikusan kisfogyasztású, olcsó eszközökhöz, például digitális hőmérőkhöz, elektronikus kulcsokhoz (iButton) használják
- Különlegessége, hogy az eszközök áramellátását az adatvezetéken keresztül is biztosítani tudja (aktív állapotában feltölti az eszközökbe beépített kapacitást)
- Az **1-wire** buszon mindig van egy **mester eszköz** (számítógép vagy mikrovezérlő), amely vezérli az adatforgalmat



1-wire eszközök működési feszültségtartományai

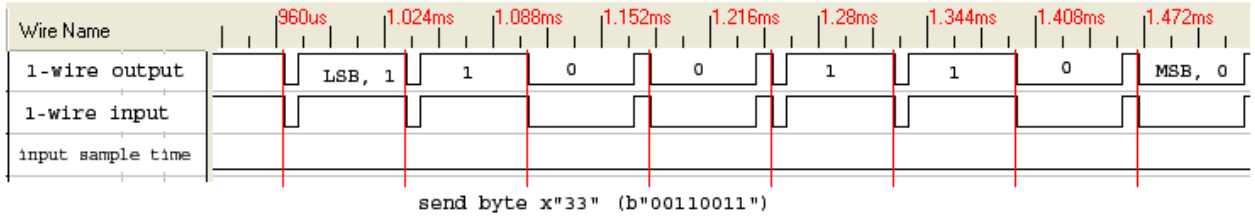
- 1.71V (min) to 1.89V (max)
- 1.71V (min) to 3.63V (max)
- 2.97V (min) to 3.63V (max)
- 2.8V (min) to 5.25V (max)

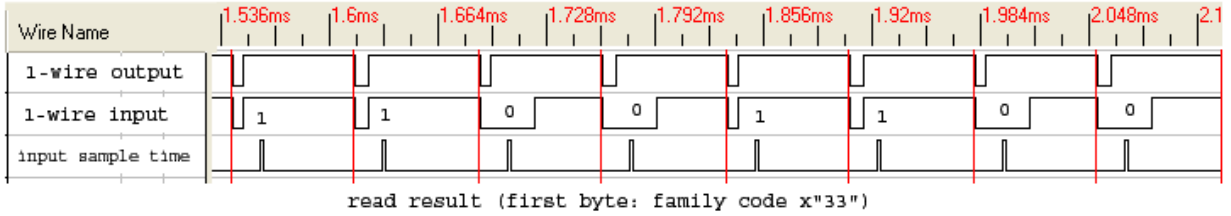
Hogyan zajlik az adatforgalom az 1-wire buszon?



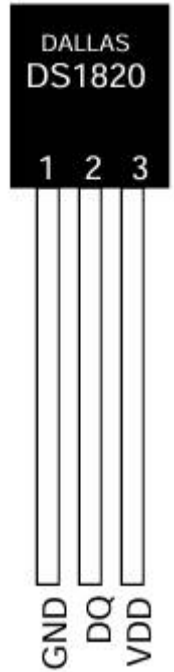
Az ábra forrása:

[Wikipedia/1-wire]

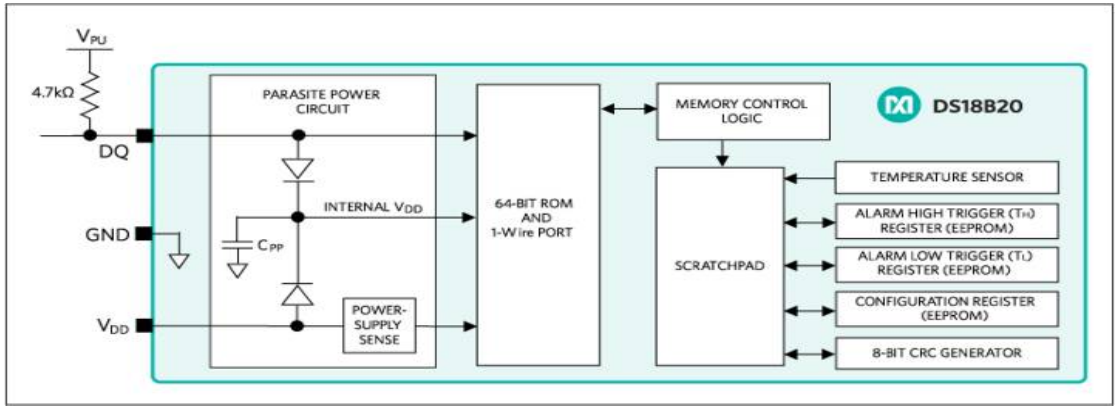


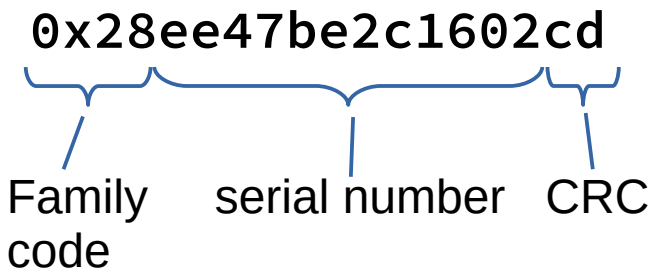


DS18B20 digitális hőmérő



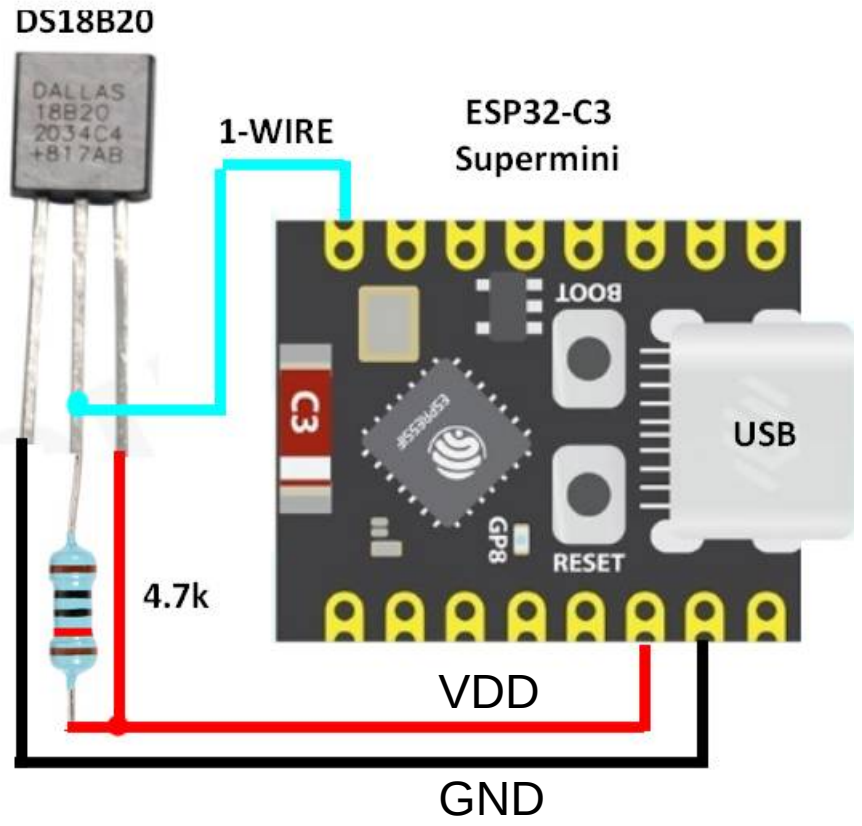
- A **DS18B20** a Dallas Semiconductor programozható felbontású 1-wire buszra fűzhető digitális hőmérője
  - **Mérési tartomány:** –55℃ és +125℃ között
  - **Felbontás:** programozhatóan 9 – 12 bit *(0,5℃ – 0.0625℃)*
  - **Pontosság:** ±0,5℃ a –10℃ és +85℃ közötti tartományban
  - **1-Wire kommunikáció:** Egyetlen adatvezetéken keresztül kommunikál
  - **Egyedi azonosító:** Minden érzékelő gyárilag egyedi azonosítóval rendelkezik, így több érzékelőt is felfűzhetünk egy buszveztékre
  - **Energiatakarékos:** Alacsony energiafogyasztású





Bekötési vázlat

- Első kísérletünkhöz az **IO21**-es kivezetésre kötünk egy, vagy több **DS18B20** hőmérőt, így ez a láb lesz az 1-wire busz vezérlője (master)
- Az **1-wire** buszvezeték felhúzására egy 4.7 kΩ-os ellenállást használunk
- A hőmérő VDD és GND lábait a mikro-vezérlő 3V3 és GND kivezetéseihez kötjük
- A hőmérő ún. parazita üzemmódban is táplálható, ekkor a VDD lábat is a GND-re kötjük



onewire_demo.py

- Felderítjük az **IO21** kivezetéshez csatlakozó **1-wire** buszt és kilistázzuk a megtalált eszközök azonosítóját

```
import board
from adafruit_onewire.bus import OneWireBus

ow_bus = OneWireBus(board.IO21)
print("Resetting bus...", end="")
if ow_bus.reset():
    print("OK.")
else: raise RuntimeError("Nothing found on bus.")

print("Scanning for devices...", end="")
devices = ow_bus.scan()
print("OK.")
print("Found {} device(s).".format(len(devices)))

# For each device found, print out some info
for i, d in enumerate(devices):
    print(" ")
    print("Device {:>3}".format(i))
    print("    ROM=0x{}    Family=0x{:02x} CRC=0x{:02x}".format(d.rom.hex(), d.family_code, d.crc))
    print("    Serial Number = ", end="")
    for byte in d.serial_number:
        print("0x{:02x} ".format(byte), end="")
```

```
Resetting bus...OK.
Scanning for devices...OK.
Found 2 device(s).

Device   0
    ROM=0x28ee47be2c1602cd    Family=0x28 CRC=0xcd
    Serial Number = 0xee 0x47 0xbe 0x2c 0x16 0x02
Device   1
    ROM=0x28eed98d2e1601b5    Family=0x28 CRC=0xb5
    Serial Number = 0xee 0xd9 0x8d 0x2e 0x16 0x01
```

ds18x20_asynctest.py

- Ha még nem ismerjük a hőmérőnk egyedi címét, akkor az **1-wire** buszon például a legelsőnek válaszoló hőmérőt így olvashatjuk ki

```
import time
import board
from adafruit_onewire.bus import OneWireBus
from adafruit_ds18x20 import DS18X20

ow_bus = OneWireBus(board.IO21)

# Scan for sensors and grab the first one found.
ds18 = DS18X20(ow_bus, ow_bus.scan()[0])
ds18.resolution = 12

while True:
    conversion_delay = ds18.start_temperature_read()
    conversion_ready_at = time.monotonic() + conversion_delay
    print("waiting", end="")
    while time.monotonic() < conversion_ready_at:
        print(".", end="")
        time.sleep(0.1)
    print("")
    print("Temperature: {0:0.3f}C".format(ds18.read_temperature()))
    time.sleep(1.0)
```

```
waiting........
Temperature: 24.250C
waiting........
Temperature: 24.250C
waiting........
Temperature: 24.250C
waiting........
Temperature: 24.250C
waiting........
Temperature: 24.250C
...
waiting........
Temperature: 24.313C
waiting........
Temperature: 24.313C
waiting........
```

ds18x20_addressed.py

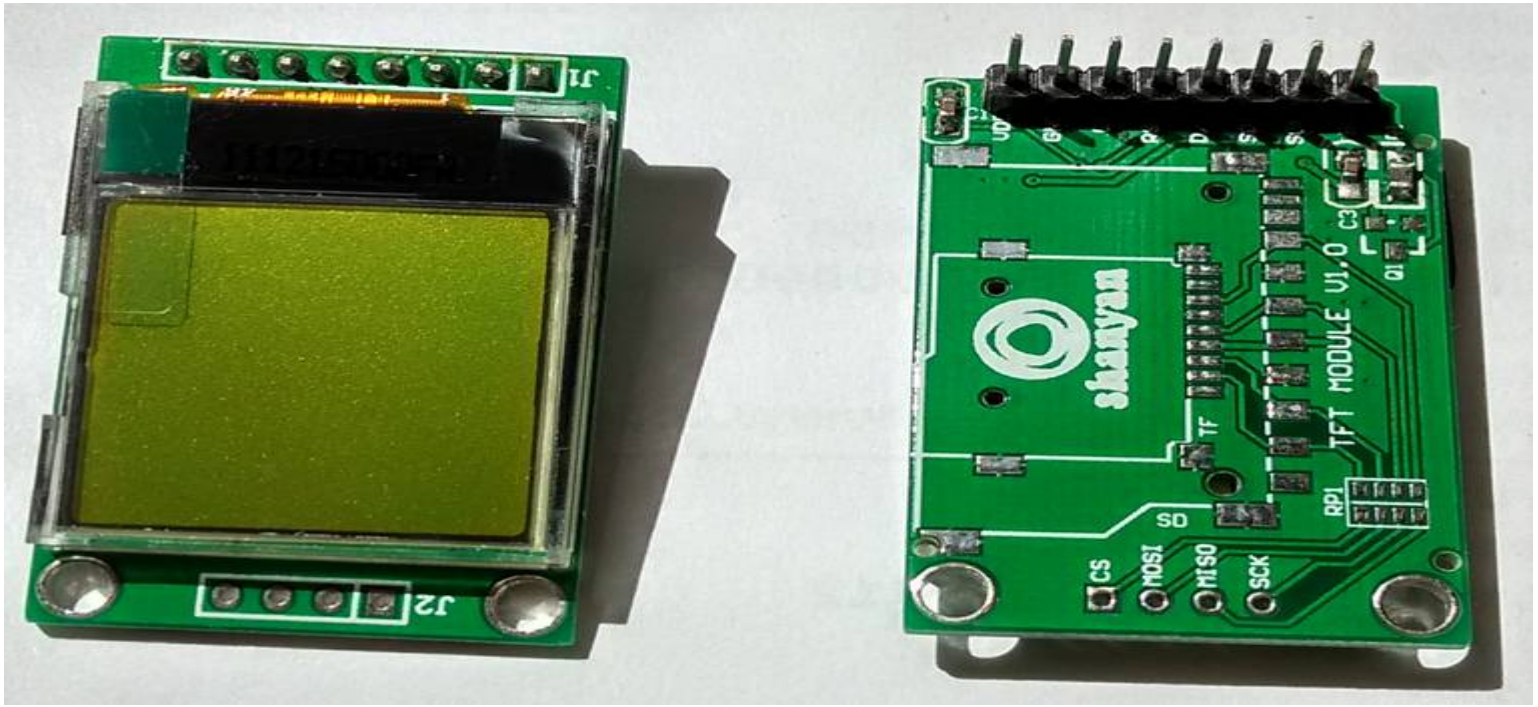
- Ha már ismerjük a hőmérőnk egyedi címét, akkor a cím felhasználásával célzottan megszólíthatjuk az 1-wire buszon

```
import time, board
from adafruit_onewire.bus import OneWireBus, OneWireAddress
from adafruit_ds18x20 import DS18X20

ow_bus = OneWireBus(board.IO21)
rom_address_bytes = bytearray([0x28,0xee,0xd9,0x8d,0x2e,0x16,0x01,0xb5])
rom_address = OneWireAddress(rom_address_bytes)
ds18 = DS18X20(ow_bus, rom_address)

while True:
    conversion_delay = ds18.start_temperature_read()
    conversion_ready_at = time.monotonic() + conversion_delay
    print("waiting", end="")
    while time.monotonic() < conversion_ready_at:
        print(".", end="")
        time.sleep(0.1)
    print("")
    print("Temperature: {0:0.3f}C".format(ds18.read_temperature()))
    time.sleep(1.0)
```

```
waiting........
Temperature: 25.625C
waiting........
Temperature: 25.625C
waiting........
Temperature: 25.625C
waiting........
Temperature: 25.625C
```

Shanyan TFT module 1.0

- Monokróm grafikus LCD 96*64 képpont + 1 ikonsor
- Sitronix ST7585 vezérlő, SPI vezérléssel, LED megvilágítással
- A modul SD kártyával kiegészíthető

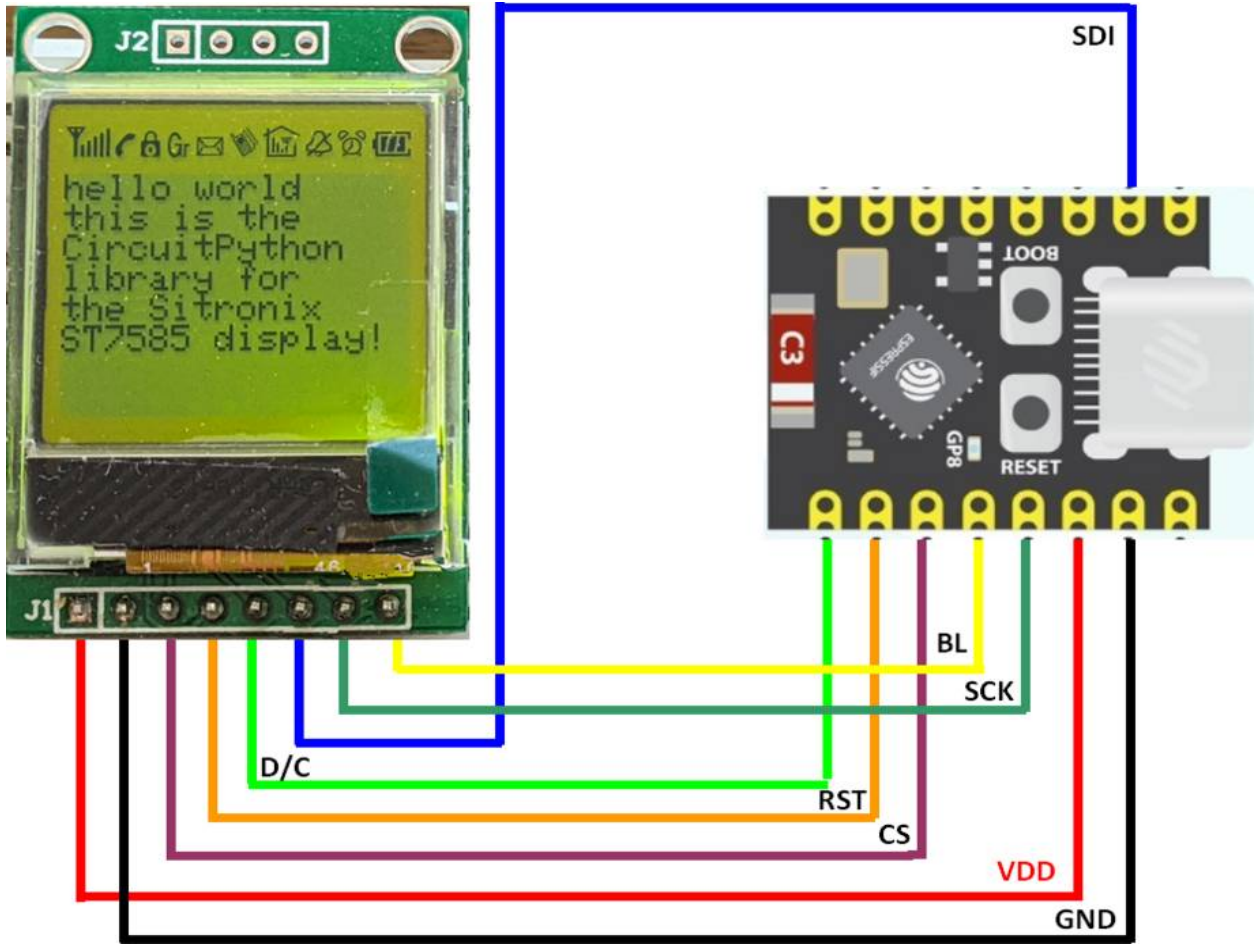


Shanyan TFT module 1.0 kivezetések

- VDD (Voltage Drain Drain) tápfeszültség (+3,3 V)
- GND (Ground) a tápegység közös pontja
- CS (Chip Select) modul aktiválásához alacsony szintre kell húzni
- RST (Reset) a modult alaphelyzetbe állítja
- D/C (Data/Command) adat vagy parancsküldés választó vonal
- SDI (Serial Data Input) SPI soros adatbemenet
- SCK (Serial Clock) SPI órajel
- BL (Backlight) LED megvilágítás (I_f = 20 mA, U_f = 2,1 V typ.)

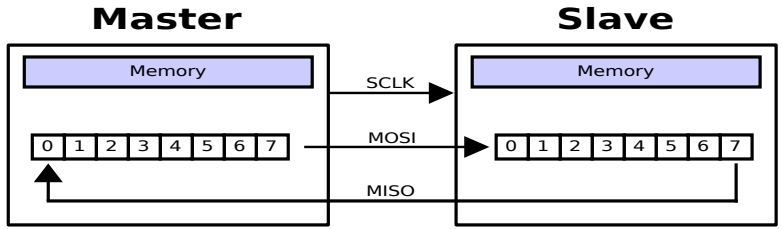
Sanyan ST7585 LCD bekötése

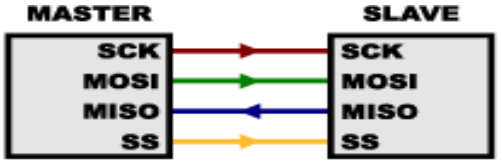
Display	ESP32-C3
VDD	3,3 V
GND	GND
CS	IO2
RST	IO1
D/C	IO0
SDI	IO6
SCK	IO4
BL	IO3

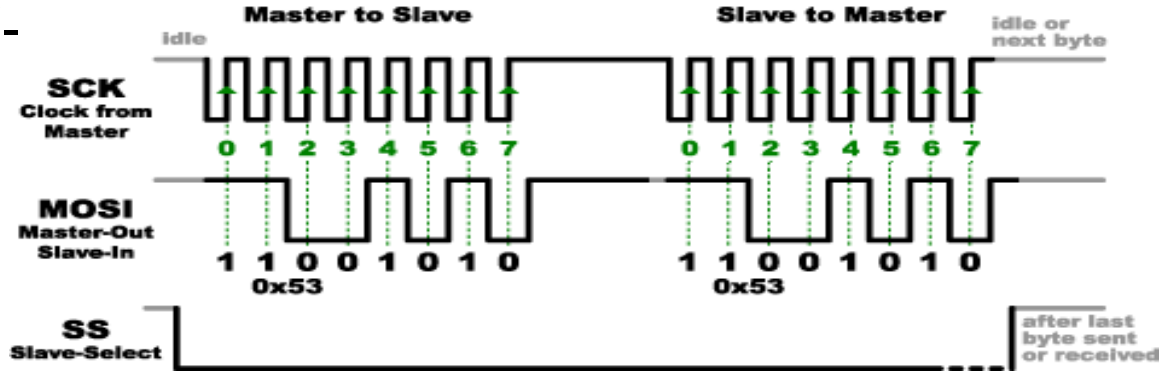


Az SPI kommunikációs csatorna

- A Soros Periféria Illesztő (**SPI**) bitsoros, kétirányú kommunikáció (mi most csak az egyik irányt használjuk)
- A Master (esetünkben a mikrovezérlő) állítja elő az órajelet, ez szinkronizál és ütemezi az adatküldés sebességét
- Az **SPI** csatorna jelei:
  SCK – az órajel
  MOSI – a master kimenő adatvonala
  MISO – a master adatbemenete
  CS, vagy **SS** – a slave kiválasztó jele
- A kommunikáció során két léptető-regiszter adatot cserél







Az SPI osztály tagfüggvényei

- A **busio** modul tartalmazza a különféle soros perifériákat kezelő osztályokat (I2C, SPI, UART)
- Leírásuk az Adafruit CircuitPython dokumentációban található

```
>>> import busio
>>> dir(busio)
['__class__', '__name__', 'I2C', 'SPI', 'UART']
>>> help(busio.SPI)
object <class 'SPI'> is of type type
  deinit -- <function>            # Periféria elengedése
  __enter__ -- <function>         # Context Manager (pl. with) segítő
  __exit__ -- <function>          # Context Manager (pl. with) segítő
  configure -- <function>         # Csatorna konfigurálása
  try_lock -- <function>          # Csatorna lefoglalása
  unlock -- <function>            # Csatorna felszabadítása
  readinto -- <function>          # Adatbeolvasás
  write -- <function>             # Adatkiírás
  write_readinto -- <function>    # Írás/olvasás (duplex mód)
  frequency -- <property>         # Adatsebesség beállítása
>>>
```

ST7585 vezérlő

- LCD 66 x 102 pontmátrix vezérlő/meghajtó (esetünkben 64x96)
- 8-bit, SPI, I2C kommunikáció (esetünkben SPI)
- **Esetünkben a 8 pixeles sávok Y címe fordítva működik!**
  A képernyő bal felső sarkában Y = 7, X = 0, s az adatbájt LSB bitje szabja meg a képpont láthatóságát.

Data Structure

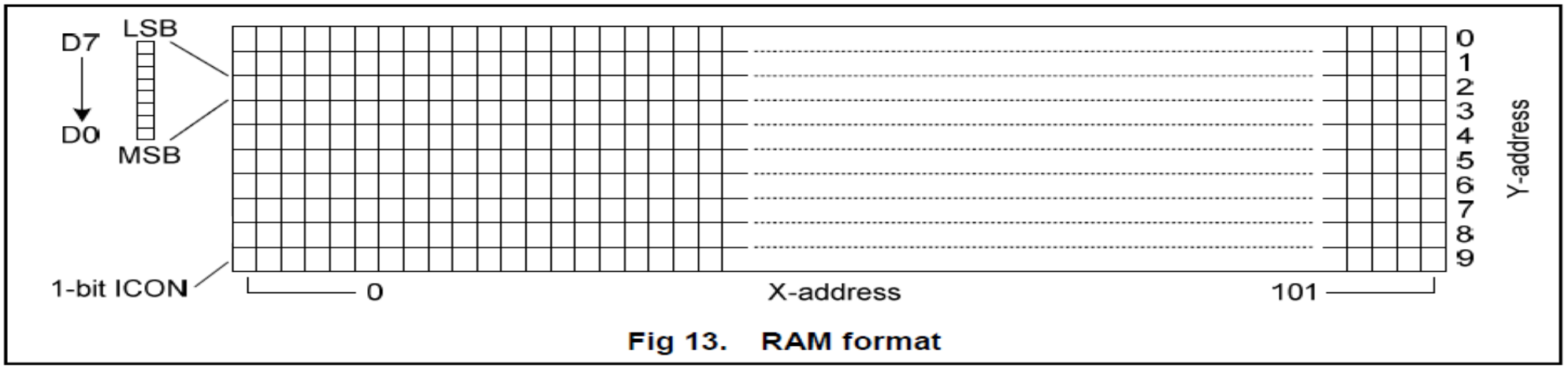


Fig 13. RAM format

ST7585 vezérlő

- Mi a ***Horizontal*** (vízszintes) ***címzési módot*** használjuk
- Ebben az egymás után kiküldött adatbájtok az Y címmel kiválasztott sáv X irányú szomszédos képpontjait adják meg.
  Tehát minden bájt 8 db egymás alatt felkvő pontot ír le, s minden adatbáit 1 képpontot lép X irányba.

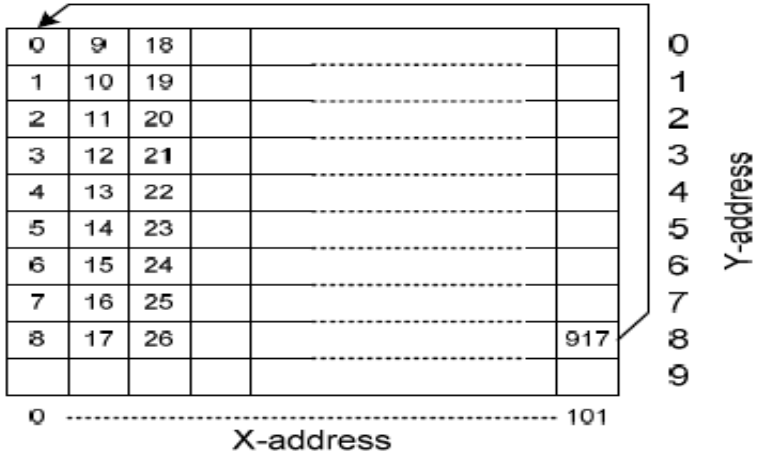


Fig 14. Addressing : Vertical Mode (V=1)

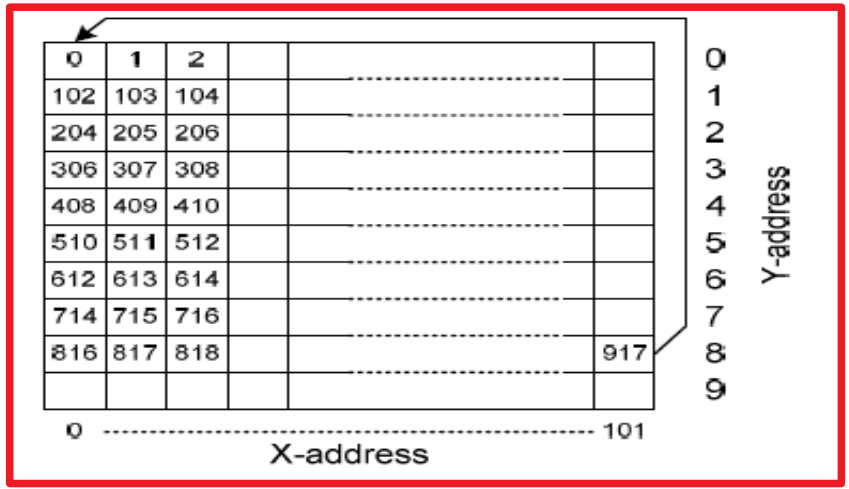


Fig 15. Addressing : Horizontal Mode (V=0)

ST7585 parancsok

- Az inicializálás és a vezérlés az alábbi parancsokkal történik

H=0 or 1 (H-Flag Independent)										
INSTRUCTION	A0	R/W (RWR)	COMMAND BYTE							DESCRIPTION
			D7	D6	D5	D4	D3	D2	D1	D0	
NOP	0	0	0	0	0	0	0	0	0	0	No operation
Function Set	0	0	0	0	1	0	0	PD	V	H	Power down; entry mode; Select instruction table
Write Data	1	0	D7	D6	D5	D4	D3	D2	D1	D0	Write data to RAM

H=0 (Basic Instruction)										
INSTRUCTION	A0	R/W (RWR)	COMMAND BYTE							DESCRIPTION
			D7	D6	D5	D4	D3	D2	D1	D0	
Display Control	0	0	0	0	0	0	1	D	0	E	Sets display configuration
Set Y Address of RAM	0	0	0	1	0	0	Y3	Y2	Y1	Y0	Sets Y address of RAM 0≤Y≤9
Set X Address of RAM	0	0	1	X6	X5	X4	X3	X2	X1	X0	Sets X address of RAM 0≤X≤101

H=1 (Extended Instruction)										
INSTRUCTION	A0	R/W (RWR)	COMMAND BYTE							DESCRIPTION
			D7	D6	D5	D4	D3	D2	D1	D0	
Set V0	0	0	1	V04	V03	V02	V01	V00	0	0	Set V_{OP} parameter to register
Set Test Mode	0	0	0	0	1	1	0	T1	T0	TEN	Select test mode

ST7585 parancsok – Function Set

- A **Function Set** paranccsal
  - Választhatunk az alap és a kibővített parancskészlet között (H bit)
  - Power Down módba állíthatjuk a vezérlőt (PD = 1 esetén)
  - Választhatunk a Horizontal (V=0) és a Vertical (V=1) mód között

Function Set

A0	R/W	D7	D6	D5	D4	D3	D2	D1	D0
0	0	0	0	1	0	0	PD	V	H

Flag	Description
PD	PD=0: chip is active PD=1: chip is in power down mode All LCD outputs at VSS (display off), bias generator and V0 generator off, VOUT can be disconnected, oscillator off (external clock possible), RAM contents not cleared; RAM data can be written.
V	Select addressing mode: V=0 for Horizontal Addressing; V=1 for Vertical Addressing.
H	H=0: Basic Instruction set; H=1: Extended instruction set. Data access can be used in both instruction blocks. Refer to the instruction table.

ST7585 parancsok - adatküldés

- Adatküldésnél a D/C bemenetet (itt A0) magas szintre kell húzni!
- **Az adatküldés** (Write Data) parancs H = 0 és H = 1 állapotban egyaránt kiadható (független a H bittől)!
- A kiküldött adatok az adat RAM soron következő címére kerülnek.
- Az adott 8 pixeles sávon belül LSB bit (itt D0) a legfelső, az MSB bit (itt D7) pedig a legalsó képpontot vezérli
- A bit 0 értéke törli (papír szín), az 1 érték pedig megjeleníti (tintaszín) az adott képpontot

A0	R/W	D7	D6	D5	D4	D3	D2	D1	D0
1	0	Write Data							

ST7585 parancsok- Display Control

- A ***Display vezérlés*** parancs *H = 0* esetén használható
- A **D** bit 0 értéke letiltja, az 1 érték pedig engedélyezi a RAM-ban tárolt értékek megjelenítését
- Az **E** bit 0 értéke a normál, 1 értéke pedig az inverz megjelenítés kiválasztására szolgál.

H=0 (Basic Instruction)

Display Control

This bits D and E selects the display mode.

A0	R/W	D7	D6	D5	D4	D3	D2	D1	D0
0	0	0	0	0	0	1	D	0	E

Flag	Description		
D,E	D	E	The bits D and E select the display mode.
	0	0	Display OFF
	0	1	All display segments on
	1	0	Normal mode
	1	1	Inverse video mode

ST7585 parancsok- X/Y címbeállítás

- Az **Y címbeírásnál** 0 – 9 közötti számot adhatunk meg egy-egy pixelsáv, illetve az ikonsor megcímzéséhez
- Y = 0 a legalsó, 7 pedig a legfelső pixelsávot címzi meg
- Y = 8, illetve Y = 9 esetén adatbeíráskor csak a D7 bit értéke számít!
- **X beíráskor** 0 – 101 közötti (esetünkben 0 – 95 közötti) számot adhatunk meg

H=0 (Basic Instruction)											
INSTRUCTION	A0	R/W (RWR)	COMMAND BYTE								DESCRIPTION
			D7	D6	D5	D4	D3	D2	D1	D0	
Display Control	0	0	0	0	0	0	1	D	0	E	Sets display configuration
Set Y Address of RAM	0	0	0	1	0	0	Y3	Y2	Y1	Y0	Sets Y address of RAM 0≤Y≤9
Set X Address of RAM	0	0	1	X6	X5	X4	X3	X2	X1	X0	Sets X address of RAM 0≤X≤101

ST7585 parancsok- kiterjesztett parancsok

- A **V0** beállítással az LCD-t működtető feszültséget szabályozhatjuk, ami a kontrasztot befolyásolja. A nagyobb érték sötétebb kijelzőt eredményez.
- Alapértéknek V[4:0] = 0-val próbálkozzunk! Gyenge kontraszt esetén növeljük 0x20-ra az értéket! Ennél nagyobb értéknél besötétülhet a képernyő

H=1 (Extended Instruction)										
INSTRUCTION	A0	R/W (RWR)	COMMAND BYTE							DESCRIPTION
			D7	D6	D5	D4	D3	D2	D1	D0	
Set V0	0	0	1	V04	V03	V02	V01	V00	0	0	Set V_{OP} parameter to register
Set Test Mode	0	0	0	0	1	1	0	T1	T0	TEN	Select test mode

- A teszt mód beállítása nem derült ki, hogy mire jó….

Az ST7585.py library

- Készítsünk könyvtárat az **adafruit_pcd8544.py** mintájára!
- A módosítások releváns részletei:

adafruit_pcd8544.py

```
_LCDWIDTH = const(84)
_LCDHEIGHT = const(48)
_PCD8544_POWERDOWN = const(0x04)
_PCD8544_ENTRYMODE = const(0x02)
_PCD8544_EXTENDEDINSTRUCTION = const(0x01)
_PCD8544_DISPLAYBLANK = const(0x0)
_PCD8544_DISPLAYNORMAL = const(0x4)
_PCD8544_DISPLAYALLON = const(0x1)
_PCD8544_DISPLAYINVERTED = const(0x5)
_PCD8544_FUNCTIONSET = const(0x20)
_PCD8544_DISPLAYCONTROL = const(0x08)
_PCD8544_SETYADDR = const(0x40)
_PCD8544_SETXADDR = const(0x80)
_PCD8544_SETTEMP = const(0x04)
_PCD8544_SETBIAS = const(0x10)
_PCD8544_SETVOP = const(0x80)
```

st7585.py

```
_LCDWIDTH = const(96)
_LCDHEIGHT = const(64)
_ST7585_POWERDOWN = const(0x04)
_ST7585_ENTRYMODE = const(0x02)
_ST7585_EXTENDEDINSTRUCTION = const(0x01)
_ST7585_DISPLAYBLANK = const(0x0)
_ST7585_DISPLAYNORMAL = const(0x4)
_ST7585_DISPLAYALLON = const(0x1)
_ST7585_DISPLAYINVERTED = const(0x5)
_ST7585_FUNCTIONSET = const(0x20)
_ST7585_DISPLAYCONTROL = const(0x08)
_ST7585_SETYADDR = const(0x40)
_ST7585_SETXADDR = const(0x80)
_ST7585_SETTEMP = const(0x04)
_ST7585_SETBIAS = const(0x10)
_ST7585_SETVOP = const(0x80)
```

A show() függvény átalakítása

- A legfontosabb módosítás az **ST7585** osztály ***show()*** tagfüggvényében a címzési mód átalakítása

adafruit_pcd8544.py →

```
class PCD8544(framebuf.FrameBuffer):

    def show(self) -> None:
        self.write_cmd(_PCD8544_SETYADDR)
        self.write_cmd(_PCD8544_SETXADDR)
        self._dc_pin.value = True # Set D/C
        with self.spi_device as spi:
            spi.write(self.buffer)
```

st7585.py

```
class ST7585(framebuf.FrameBuffer):

    def show(self) -> None:
        for page in range(_LCDHEIGHT // 8):
            self.write_cmd(_ST7585_SETYADDR | (7 - page))  # Set Y address for each page
            self.write_cmd(_ST7585_SETXADDR | 0)           # Set X address to 0
            self._dc_pin.value = True                      # Set D/C pin to data mode
            with self.spi_device as spi:
                spi.write(self.buffer[page * _LCDWIDTH : (page + 1) * _LCDWIDTH])
```

sávon-ként írunk

st7585_simpletest.py

- A kijelző kipróbálásához az Adafruit PCD8544 simpletes.py példaprogramot adaptáltuk, ami bemutatja a kijelző egyszerű funkcióit
- A program futtatásához a mikrovezérlő /lib alkönyvtárába telepítenünk kell az **adafruit_framebuf.mpy** és az **st7585.py** könyvtárakat, a programunk mellé pedig be kell másolnunk a **font5x8.bin** állományt
- A program elején definiálnunk kell a kijelzőt vezérlő kivezetéseket, példányosítjuk az SPI buszt (spi) és a konfiguráljuk kijelzőt (display)

```
import time, board, busio, digitalio, st7585

# Initialize SPI bus and control pins
spi = busio.SPI(board.SCK, MOSI=board.MOSI) # SCK=IO4, MOSI=IO6
dc = digitalio.DigitalInOut(board.IO0)      # data/command pin
cs = digitalio.DigitalInOut(board.IO2)      # Chip select pin
reset = digitalio.DigitalInOut(board.IO1)   # reset pin

display = st7585.ST7585(spi, dc, cs, reset)
```

st7585_simpletest.py

```
display.bias = 4
display.contrast = 40
# Turn on the Backlight LED
backlight = digitalio.DigitalInOut(board.IO3)
backlight.switch_to_output()
backlight.value = True

print("Pixel test")
display.fill(0)         # Clear the display.
display.show_icons()    # Ikonsor megjelenítése
display.show()          # call show after changing pixels

# Set a pixel in the origin 0,0 position.
display.pixel(0, 0, 1)
# Set a pixel in the middle position.
display.pixel(display.width // 2, display.height // 2, 1)
# Set a pixel in the opposite corner position.
display.pixel(display.width - 1, display.height - 1, 1)
display.show()
time.sleep(2)
```

Az IO3 kimenet felkapcsolásával bekapcsoljuk a háttérvilágítás LED-jeit

Töröljük a képernyőt

Képpontok rajzolása:

- Bal felső sarokba
- Középre
- Jobb alsó sarok

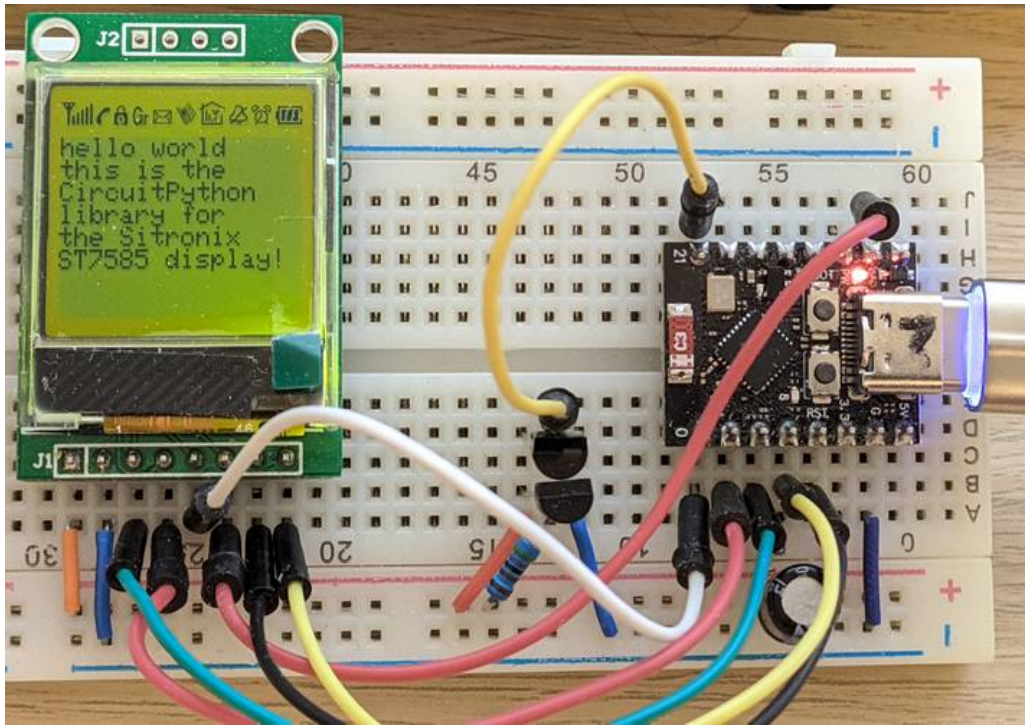
st7585_simpletest.py

```
print("Lines test")
Corners = ((0, 0), (0, display.height - 1), (display.width - 1, 0),
    (display.width - 1, display.height - 1),
)
display.fill(0)
for corner_from in corners:
    for corner_to in corners:
        display.line(corner_from[0], corner_from[1],
        corner_to[0], corner_to[1], 1)
display.show()
time.sleep(2)

print("Rectangle test")
display.fill(0)
w_delta = display.width / 10
h_delta = display.height / 10
for i in range(11):
    display.rect(0, 0,int(w_delta*i),int(h_delta*i),1)
display.show()
time.sleep(2)
```

sarokpontok koordinátái

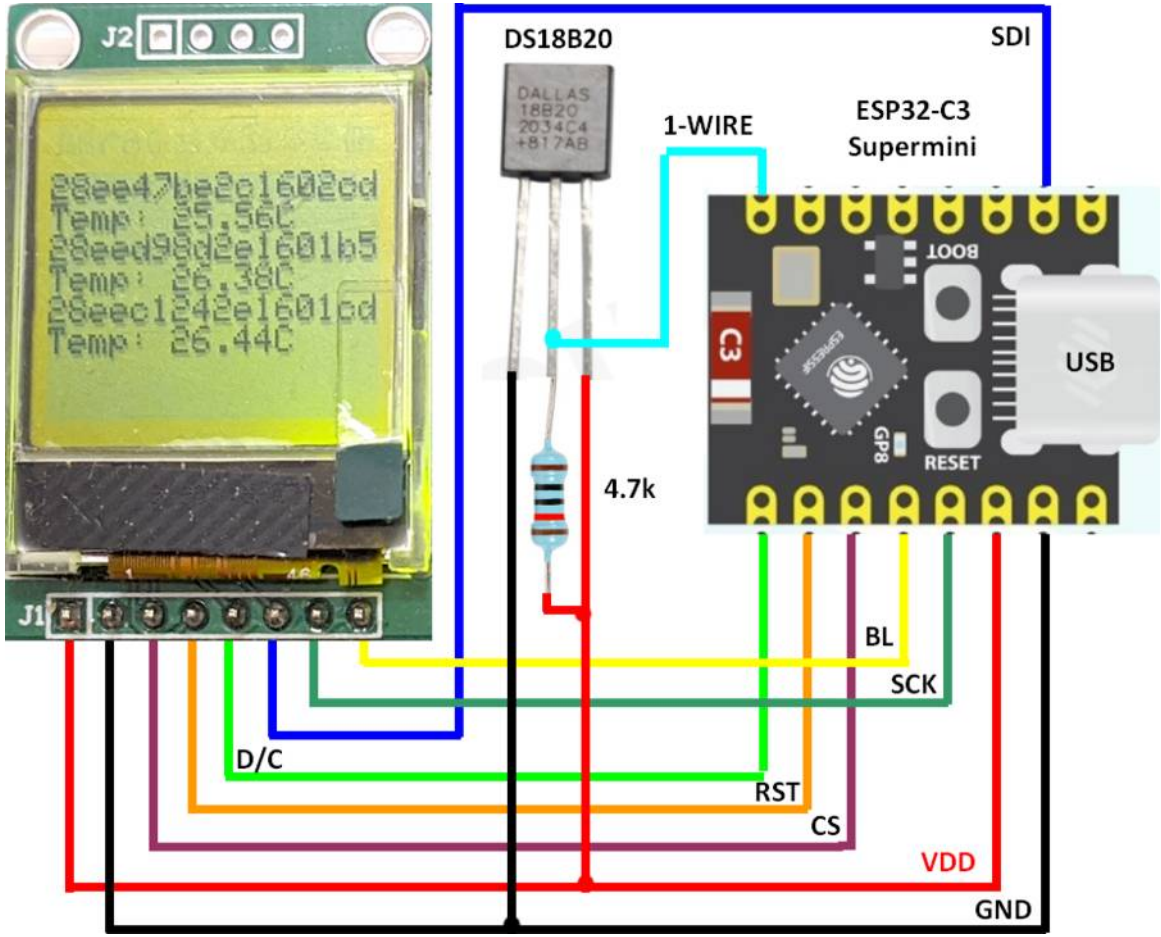
st7585_simpletest.py

```
print("Text test")
display.fill(0)
display.text("hello world", 0, 0, 1)
display.text("this is the", 0, 8, 1)
display.text("CircuitPython", 0, 16, 1)
display.text("library for", 0, 24, 1)
display.text("the Sitronix", 0, 32, 1)
display.text("ST7585 display!", 0, 40, 1)
display.show()
time.sleep(2)

while True:
    display.invert = True
    time.sleep(0.5)
    display.invert = False
    time.sleep(0.5)
```



Hőmérő alkalmazás

- Kössük össze a hőmérést a megjelenítéssel!
- Az alábbi kapcsolásban csak a korábban már bemutatott áramközi elrendezéseket egyesítettük
- Az **IO21** lábra 1 – 4 db **DS18B20** hőmérő kapcsolódhat, s a kijelzőn a ROM-ból kiolvasott azonosítójukkal együtt jelenítjük meg a pillanatnyi hőmérsékletet (hogy ellenőrizni lehessen, hogy melyik sor melyik kőmérőtől származó adat



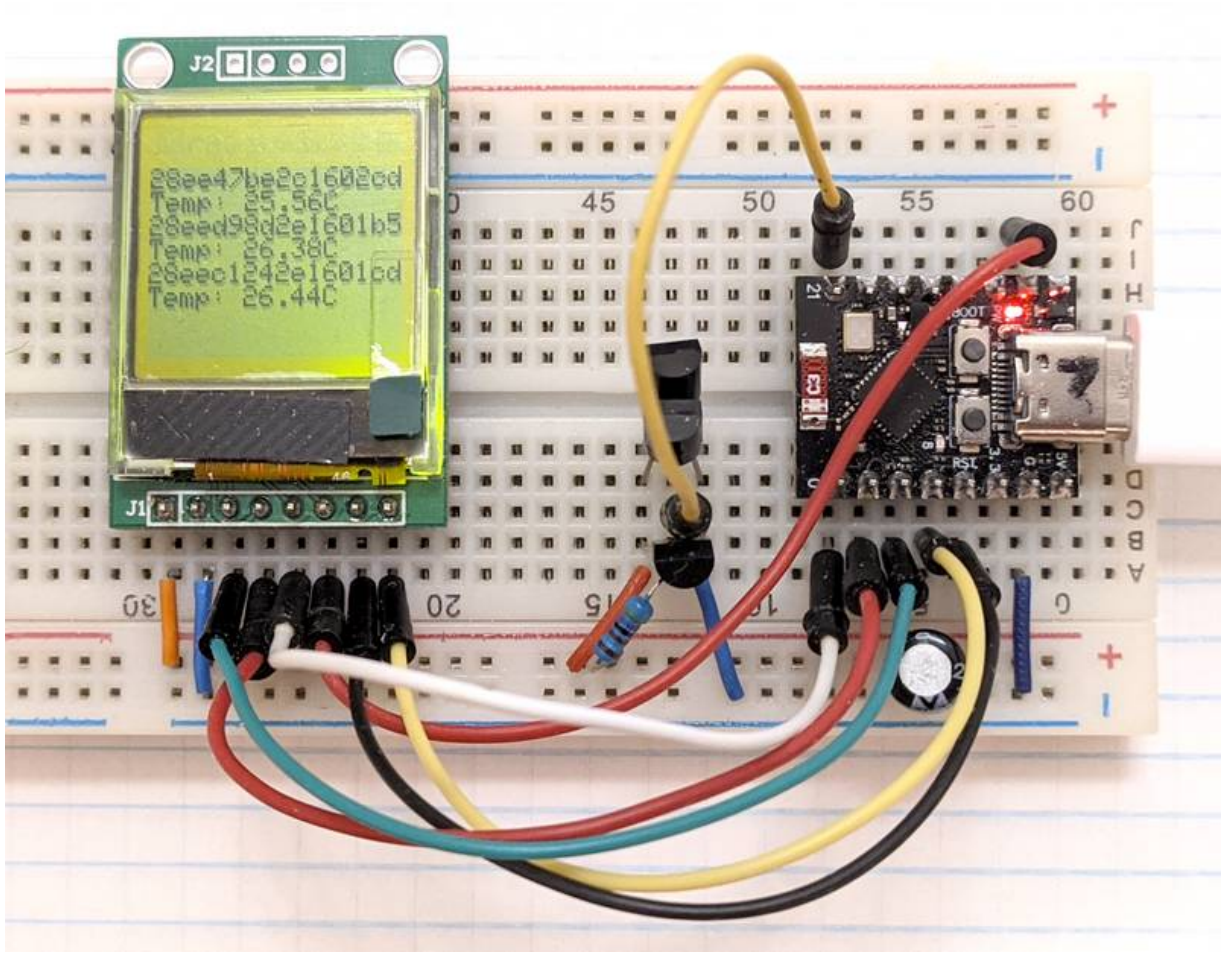
digital_thermometer.py – 2/1.

```
import time, board, busio, digitalio, st7585
from adafruit_onewire.bus import OneWireBus
from adafruit_ds18x20 import DS18X20

spi = busio.SPI(board.SCK, MOSI=board.MOSI) # Initialize SPI bus and control pins
dc = digitalio.DigitalInOut(board.IO0)      # data/command
cs = digitalio.DigitalInOut(board.IO2)      # Chip select
rst = digitalio.DigitalInOut(board.IO1)     # reset
display = st7585.ST7585(spi, dc, cs, rst)
display.bias = 4
display.contrast = 40
backlight = digitalio.DigitalInOut(board.IO3)
backlight.switch_to_output()
backlight.value = True                      # Turn on the Backlight LED
display.fill(0)                             # Clear display buffer
display.show()
display.show_icons()
```

digital_thermometer.py – 2/2.

```
ow_bus = OneWireBus(board.IO21)                 # Initialize 1-wire bus on IO21 pin
devices = ow_bus.scan()                         # Scan for devices on the bus
sensors = []                                    # Create a list of DS18X20 objects
for i, d in enumerate(devices):
    display.text(d.rom.hex(), 0, i*16, 1)       # Display the ROM ID
    sensor = DS18X20(ow_bus, d)                 # Create a new thermometer object
    sensor.resolution = 12
    sensors.append((i, sensor))                 # Append sensor to list of sensors
display.show()

while True:
    for i, sensor in sensors:
        conversion_delay = sensor.start_temperature_read()
        time.sleep(conversion_delay)
        temperature = sensor.temperature
        # Delete line before overwriting it
        display.fill_rect(0, i * 16 + 8, 96, 8, 0)
        display.text(f"Temp: {temperature:.2f}C  ", 0, i * 16 + 8, 1)
    display.show()
    time.sleep(5)
```

digital_thermometer.py – futási eredmény



Az ESP32 C3 Super Mini kártya kivezetései