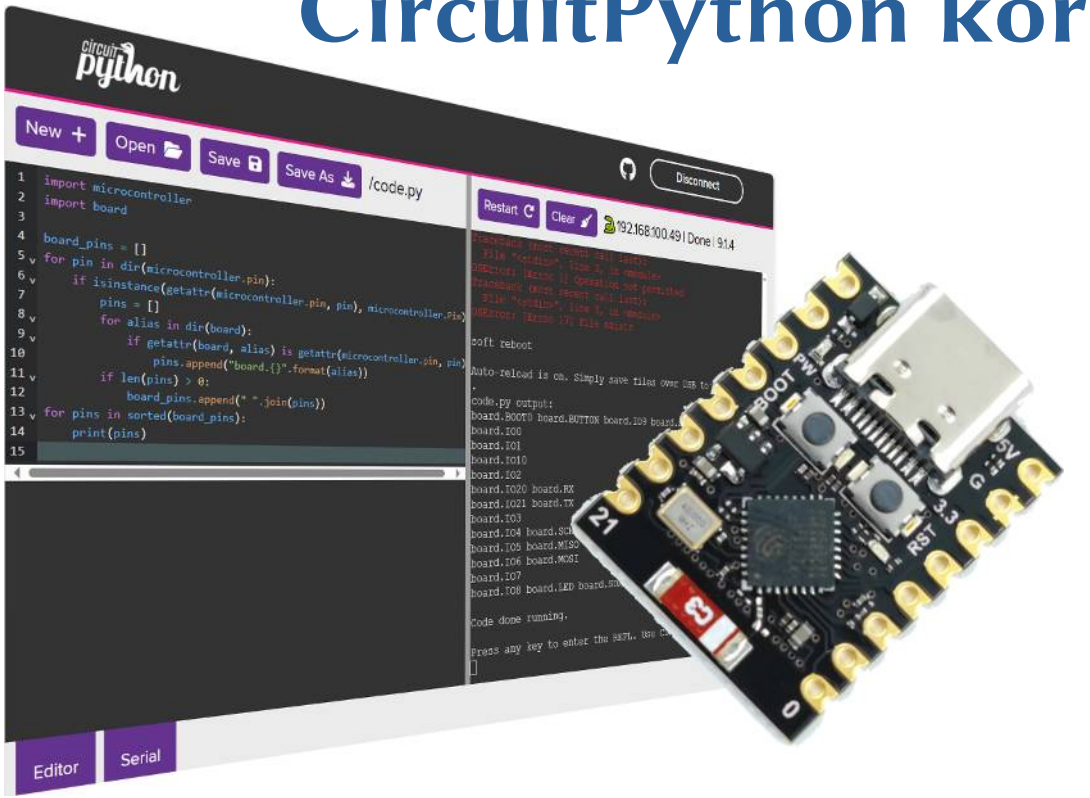
ESP32-C3 mikrovezérlők programozása CircuitPython környezetben



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

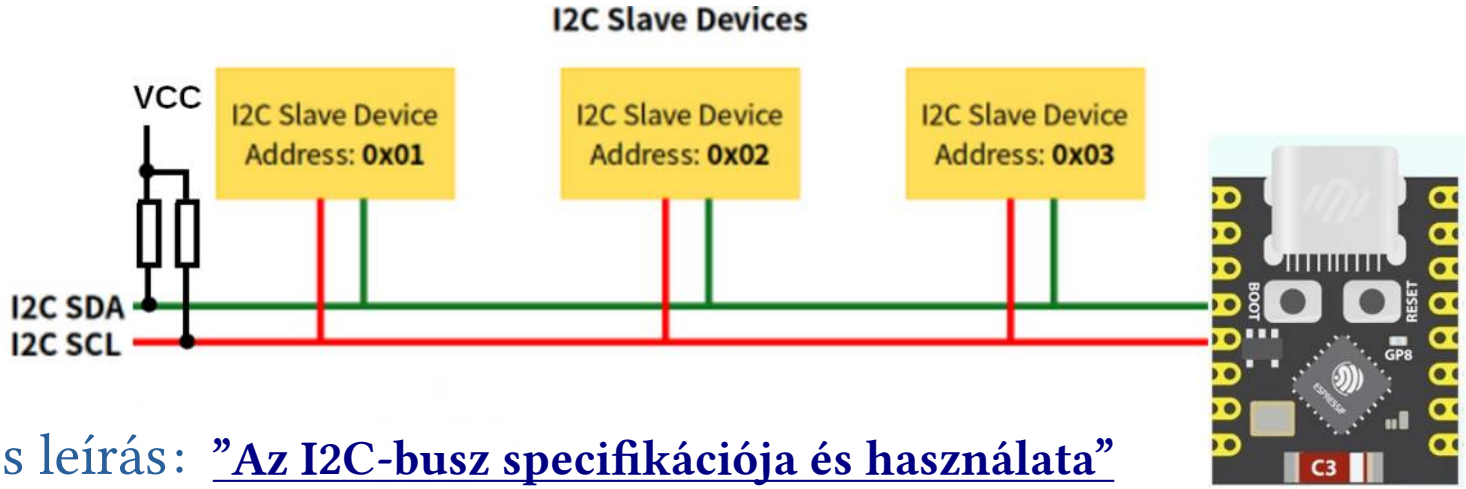
- **Online eszközök és támogatás:**
  - Learn Adafruit: CircuitPython on ESP32 Quick Start
  - Adafruit: Adafruit Web Serial ESPTool
  - Adafruit: CircuitPython Code Editor

CircuitPython könyvtárak

- A **CircuitPython könyvtárcsomag** (Adafruit CircuitPython Bundle) tartalmazza az összes jelenleg elérhető CircuitPython könyvtárat.
- A CircuitPython könyvtárak Python nyelven íródtak. További funkciókat biztosítanak és külső eszközöket is támogatnak a CircuitPythonon túl. A használni kívánt könyvtárakat a mikrovezérlő saját fájlrendszerében egy **lib** nevű mappában kell elhelyezni
- A **CircuitPython könyvtárcsomagnak** a firmware verziójához illeszkedő változatát kell letölteni és kibontani a számítógépünkre, innen tudjuk majd a mikrovezérlő kártyára áttölteni azt, amire éppen szükségünk van
  - Bundle for Version 9.x: adafruit-circuitpython-bundle-9.x-mpy-20251114.zip
  - Bundle for Version 10.x: adafruit-circuitpython-bundle-10.x-mpy-20251114.zip
  - Dokumentáció: Adafruit Sponsored Libraries and Drivers on GitHub

Az I2C kommunikációs csatorna

- Az **I2C** kommunikációs csatorna bit-soros, órajellel szinkronizált mester – szolga adatátvitel, ahol a címzés az első bájtban kiküldött címmel történik



- Részletes leírás: **”Az I2C-busz specifikációja és használata”**
- **A korábbi előadásainkban található ismertetők:**
  - Az I2C kommunikációs csatorna (2020. február 6.)
    🗎 előadásvázlat 🖫mintaprogramok
  - Az I2C kommunikációs csatorna – 1. rész (2021. február 4.)
    🗎 előadásvázlat 🖫mintaprogramok 📽video
  - Az I2C kommunikációs csatorna – 2. rész (2021. február 18.)
    🗎 előadásvázlat 🖫mintaprogramok 📽video

i2c_scan.py

- Az alábbi kis program felderíti és kilistázza az I2C buszon található eszközök címeit (a program forrása: CircuitPython Essentials)

```
import time
import board
i2c = board.I2C()           # To use default I2C bus (most boards)
# To create I2C bus on specific pins
# import busio
# i2c = busio.I2C(board.IO9, board.IO8)
                        ( SCL, SDA)

while not i2c.try_lock():
    pass
try:
    while True:
        print(
            "I2C addresses found:",
            [hex(device_address) for device_address in i2c.scan()],
        )
        time.sleep(2)
finally:  # unlock the i2c bus when ctrl-c'ing out of the loop
    i2c.unlock()
```

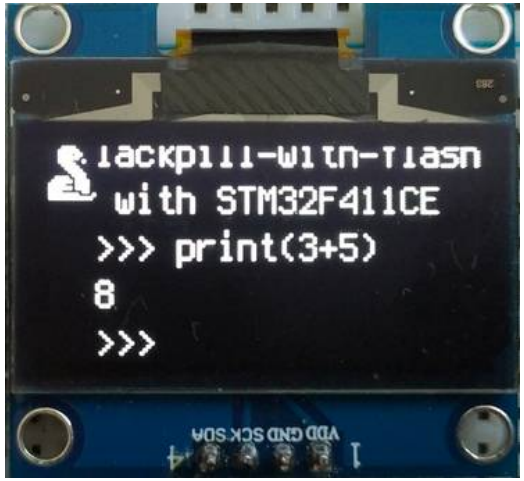
```
Adafruit CircuitPython REPL                    ( DS3231)

Auto-reload is on. Simply save files over
code.py output:
I2C addresses found: ['0x57', '0x68']
I2C addresses found: ['0x57', '0x68']
I2C addresses found: ['0x57', '0x68']
I2C addresses found: ['0x57', '0x68']
I2C addresses found: ['0x57', '0x68']
I2C addresses found: ['0x57', '0x68']
I2C addresses found: ['0x57', '0x68']
I2C addresses found: ['0x57', '0x68']
```

OLED kijelzők az I2C buszon

- Az **I2C OLED** grafikus kijelzőkben többnyire **SSD1306** vagy **SH1106** vezérlő található (ezek a kijelzők kaphatók SPI változatban is…)
- **OLED**-nek (*organic light-emitting diode*-nak) nevezzük a szerves fénykibocsátó diódát, amelyben a fénykibocsátásért felelős **elektrolumineszcens** réteg szerves félvezető vegyület, mely elektromos áram hatására világít
- 128x64 (128x32) felbontás
- 0.96” vagy 1.3” méret
- I2C cím **0x3C** (0x3D)
- 3,3V-5.0V tápfeszültség
- Normál/inverz mód
- Grafikus megjelenítés
- Szoftveresen állítható kontraszt
- Tükrözés/elforgatás

OLED kijelzők: „Két út van előttem…”

- Az egyszerűbb használati mód az **adafruit_ssd1306** meghajtó használata, amely a **busio** modul és az **adafruit_framebuf** (utóbbit telepíteni kell!) segítségével vezérli a kijelzőt (ami lehet akár **I2C,** akár **SPI** illesztőjű)
- Mivel **adafruit_sh1106** meghajtó nem állt rendelkezésre, ezért a 2021/22-es tanfolyam keretében az **adafruit_ssd1306** mintájára készítettünk egyet (csak **I2C** illesztőhöz!)
- A másik lehetőség a **displayio** modul használata, ehhez **adafruit_displayio_ssd1306** és **adafruit_displayio_sh1106** meghajtó is található (az utóbbi, sajnos, nem megfelelően kezeli a 2 pixeles eltolódást, ezért nekünk kell megoldanunk azt az alkalmazásokban - a legegyszerűbb megoldás 132x64 felbontásúnak definiálni a kijelzőt és csak a középső 128x64 képpontot használni)
- A **displayio** megjelenítés használatakor a soros porti üzenetek a kijelzőn is megjelennek, ami néha zavaró lehet!

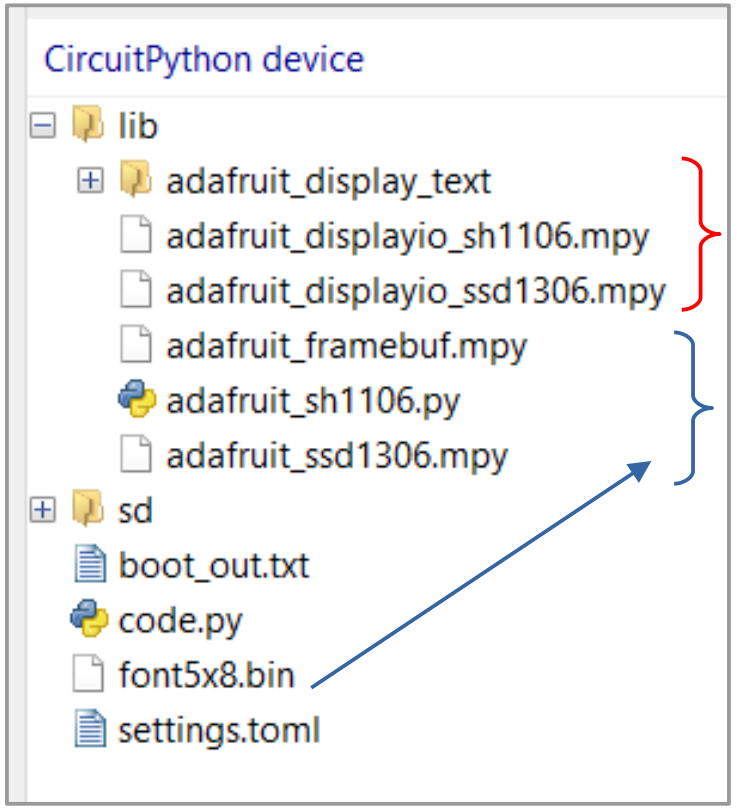


A kijelző vezérlésére használható függvények

Az **adafruit_ssd1306.SSD1306_I2C** osztály az alábbi tagfüggvényeket örökli a **FrameBuffer** osztálytól:

- **circle**(center_x, center_y, radius, color) – kör rajzolása
- **fill**(color) – a FrameBuffer kitöltése a megadott színnel
- **fill_rect**(x, y, width, height, color) – kitöltött téglalap rajzolása
- **hline**(x, y, width, color) – vízszintes vonal rajzolása adott hosszúságig
- **line**(x0, y0, x1, y1, color) – szakasz rajzolás Bresenham algoritmussal
- **pixel**(x, y, color=None) – pont rajzolása, vagy színének leolvasása
- **rect**(x, y, width, height, color, , fill=False) – téglalap rajzolása
- **rotation** – a kép forgatása 90 fokonként: 0, 1, 2 vagy 3
- **scroll**(delta_x, delta_y) – FrameBuffer eltolása X és Y irányba.
- **text**(string, x, y, color, *, font_name='font5x8.bin', size=1) – szöveg megjelenítése
- **vline**(x, y, height, color) – függőleges vonal rajzolása

Telepítendő könyvtárak és segédletek



- Az OLED kijelzők használatához az alábbi fájlokat kell telepíteni
  displayio típusú kijelző kezeléshez

  framebuf típusú kijelző kezeléshez

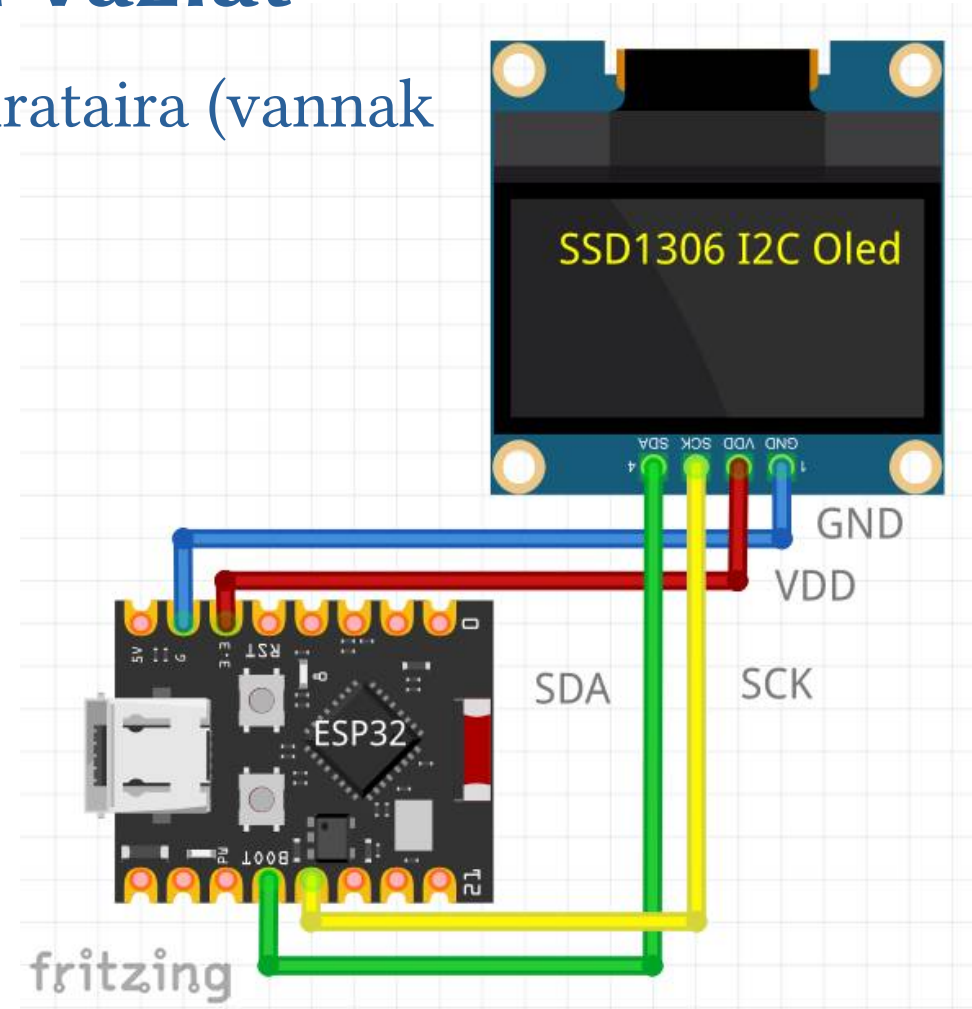
A **font5x8.bin** fontfájl a program mellett kell, hogy legyen, nem a **lib** mappában!

Az **adafruit_sh1106.py** állomány az előadás mellékletében, a többi könyvtár és segédlet az adafruit-circuitpython-bundle-10.x-mpy-20251114.zip csomagban található

Bekötési vázlat

- Bekötésnél ügyeljünk a kijelző feliratait (vannak más lábkiosztású kijelzők is)

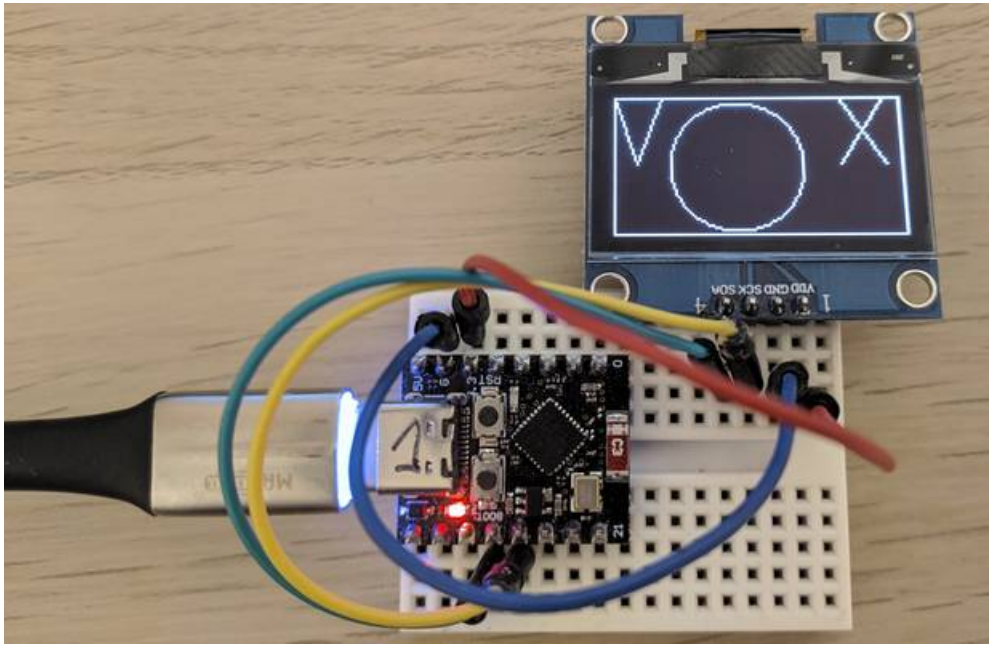
ESP32-C3	SSD1306
G	GND
3.3V	VDD
IO9	SCK
IO8	SDA



SSD1306_oled_simpletest.py

```
import board, busio, time
import adafruit_ssd1306
i2c = busio.I2C(board.SCL, board.SDA, frequency=400000)
oled = adafruit_ssd1306.SSD1306_I2C(128, 64, i2c, addr=0x3c)

oled.rotate(0)
oled.fill(0)
oled.line(0,0,10,31,1)
oled.line(10,31,21,0,1)
oled.line(100,0,120,31,1)
oled.line(100,31,120,0,1)
oled.circle(54,32,30,1)
oled.rect(0,0,128,64,1)
oled.show()
while True:
    oled.contrast(0x60)
    time.sleep(2)
    oled.invert(1)
    time.sleep(2)
    oled.invert(0)
    time.sleep(2)
     oled.contrast(0x90)
     time.sleep(2))
```

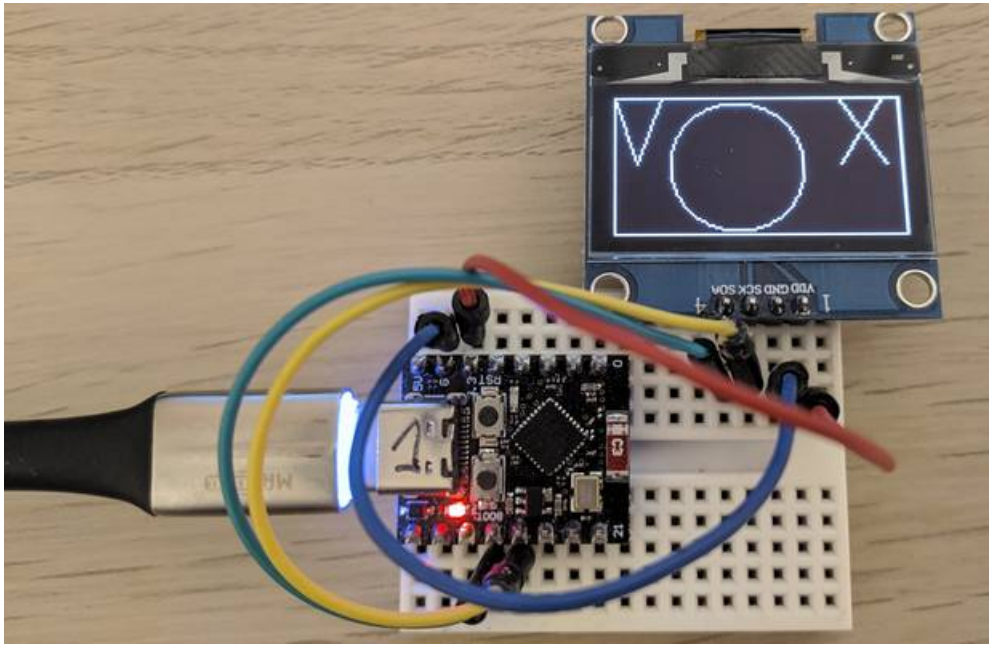
A line, circle, rect és hasonló objektumok rajzolását az adafruit_framebuf támogatja



SH1106_oled_simpletest.py

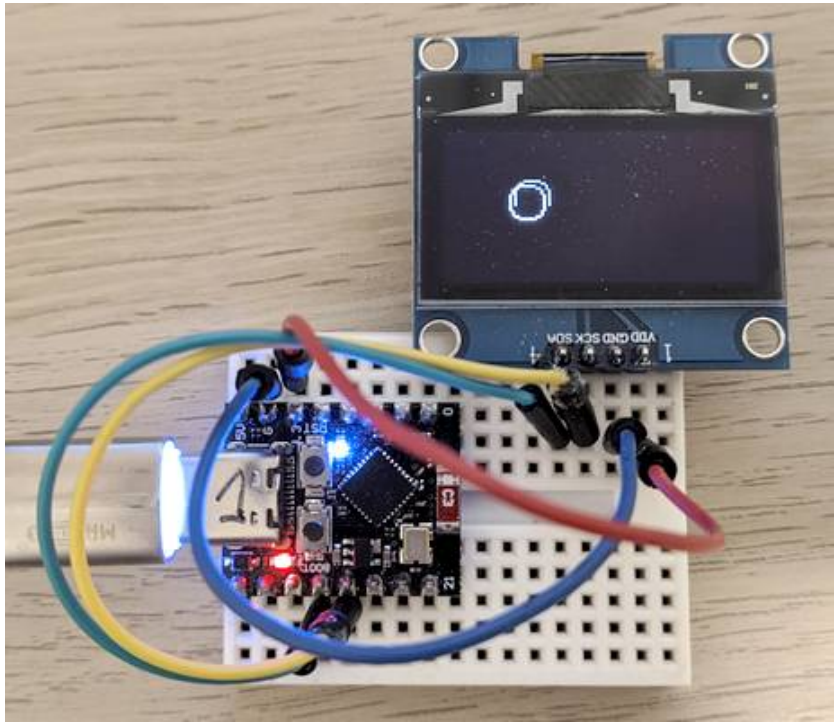
```
import board, busio, time
import adafruit_sh1106
i2c = busio.I2C(board.SCL, board.SDA, frequency=400000)
oled = adafruit_sh1106.SH1106_I2C(128, 64, i2c, addr=0x3c, external_vcc=False)

oled.rotate(0)
oled.fill(0)
oled.line(0,0,10,31,1)
oled.line(10,31,21,0,1)
oled.line(100,0,120,31,1)
oled.line(100,31,120,0,1)
oled.circle(54,32,30,1)
oled.rect(0,0,128,64,1)
oled.show()
while True:
    oled.contrast(0x60)
    time.sleep(2)
    oled.invert(1)
    time.sleep(2)
    oled.invert(0)
    time.sleep(2)
     oled.contrast(0x90)
     time.sleep(2))
```

A line, circle, rect és hasonló objektumok rajzolását az adafruit_framebuf támogatja



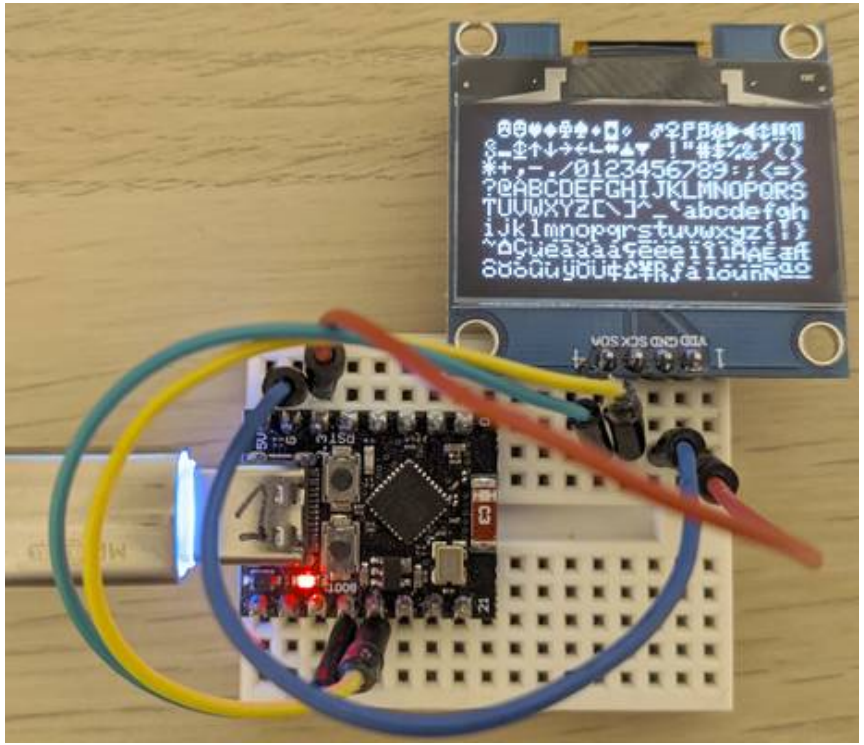
SH1106_bouncing_ball.py

- Az **Adafruit_CircutPython_SSD1306** programkönyvtár **ssd1306_bouncing_ball.py** mintapéldájának átdolgozott változatát nem akarom részletesen ismertetni, csupán azért teszem közzé, mert néhány hibát ki kellett javítani benne
- A javítások mellett a saját körrajzoló függvény helyett az **adafruit framebuf**-tól megörökölt körrajzoló függvényét használjuk
- Az **I2C** busz frekvenciájának megnövelésével és a lépésköz megduplázásával az animációt felgyorsítottak



sh1106_framebuffer_test.py

- Ehhez a programhoz az Adafruit_CircutPython_SSD1306 programkönyvtár **ssd1306_framebuftest.py** mintapéldáját használtuk fel, amely a rajzolási és szövegmegjelenítési funkciókat mutatja be
- **Követelmények:**
  adafruit_framebuf.mpy (a lib mappában)
  adafruit_ssd1306.mpy vagy
  adafruit_sh1106.py (a lib mappában)
  font5x8.bin (a code.py állomány mellett)



sh1106_framebuffer_test.py – 3/1.

```
import time, board, busio
from digitalio import DigitalInOut
import adafruit_sh1106
i2c = busio.I2C(board.SCL, board.SDA, frequency=400000)        # Create the I2C interface
display = adafruit_sh1106.SH1106_I2C(128, 64, i2c, addr=0x3C)
print("Pixel test")
display.rotate(0)    # Display rotation
display.fill(0)      # Clear display buffer
display.show()       # Copy buffer to screen

display.pixel(0, 0, 1)
display.pixel(display.width // 2, display.height // 2, 1) # Set a pixel in the middle position.
display.pixel(display.width - 1, display.height - 1, 1)   # Set a pixel in the opposite corner position
display.show()
time.sleep(0.1)

print("Lines test")
corners = (
    (0, 0),
    (0, display.height - 1),
    (display.width - 1, 0),
    (display.width - 1, display.height - 1),
)
```

sh1106_framebuffer_test.py – 3/2.

```
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(0.1)

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(0.1)

print("Text test")
display.fill(0)
try:
    display.text("hello world", 0, 0, 1)
    display.show()
    time.sleep(1)
    display.fill(0)
```

sh1106_framebuffer_test.py – 3/3.

```
    char_width = 6
    char_height = 8
    chars_per_line = display.width // 6
    for i in range(255):
        x = char_width * (i % chars_per_line)
        y = char_height * (i // chars_per_line)
        display.text(chr(i), x, y, 1)
    display.show()

except OSError:
    print(
        "To test the framebuf font setup, you'll need the font5x8.bin file from "
        + "https://github.com/adafruit/Adafruit_CircuitPython_framebuf/blob/main/examples/"
        + " in the same directory as this script"
    )
```

Megjegyzés: Az eredeti programban `except FileNotFoundError:` állt, de a CircuitPythonban nincs ilyen hiba definiálva, ezért helyette ezt kellett írni:
`except OSError:`

OLED kijelzők displayio támogatása

- A következő mintapéldában a **displayio** könyvtárat fogjuk használni, amely eleve része a CircuitPython firmware csomagnak
- A kijelző kezeléséhez azonban telepítenünk kell az **adafruit_displayio_ssd1306.mpy** vagy **adafruit_displayio_sh1106.mpy** könyvtárat (az Adafruit CircuitPython Bundle tartalmazza)
- **Megjegyzés:** ügyeljünk az elnevezésre, ne keverjük össze ezek nevét az előző programoknál használt **displayio** nélküliekkel!

- Telepítenünk kell az **adafruit_display_text** programkönyvtárat is
- A **SH1106_displayio_demo.py** példaprogram a kijelző inicializálása után kirajzol egy nagy fehér téglalapot, annak közepében egy fekete kitöltött téglalapot, s abban kiír egy szöveget
- A program forrása: az adafruit_displayio_ssd1306 könyvtár mintaprogramja, amelyen csak apróbb módosításokat eszközöltünk

SH1106_displayio_test.py – 2/1.

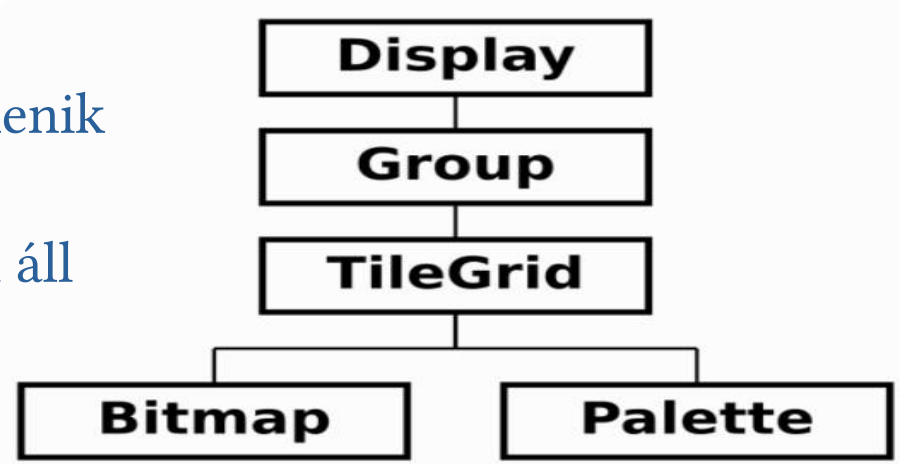
```
import board
import displayio
import terminalio
from adafruit_display_text import label
from i2cdisplaybus import I2CDisplayBus
import adafruit_displayio_sh1106  # ez a driver kell a lib mappába

displayio.release_displays()

# I2C inicializálás
i2c = board.I2C()  # uses board.SCL and board.SDA
display_bus = I2CDisplayBus(i2c, device_address=0x3C)

# SH1106 paraméterek
WIDTH = 128   # teljes fizikai szélesség
HEIGHT = 64
BORDER = 5

# colstart=2 → X irányú offset
display = adafruit_displayio_sh1106.SH1106(display_bus, width=WIDTH+4, height=HEIGHT, colstart=2)

# Megjelenítési kontextus
splash = displayio.Group()
display.root_group = splash
```

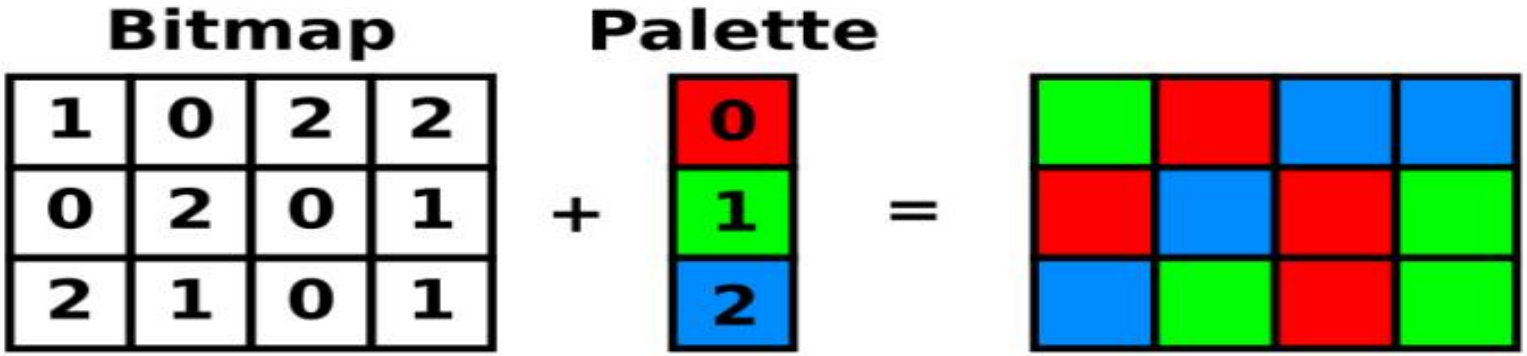
SH1106_displayio_test.py – 2/2.

```
# Háttér bitmap (fehér)
color_bitmap = displayio.Bitmap(WIDTH, HEIGHT, 1)
color_palette = displayio.Palette(1)
color_palette[0] = 0xFFFFFF
bg_sprite = displayio.TileGrid(color_bitmap, pixel_shader=color_palette, x=0, y=0)
splash.append(bg_sprite)

# Belső fekete téglalap
inner_bitmap = displayio.Bitmap(WIDTH - BORDER * 2, HEIGHT - BORDER * 2, 1)
inner_palette = displayio.Palette(1)
inner_palette[0] = 0x000000
inner_sprite = displayio.TileGrid(inner_bitmap, pixel_shader=inner_palette, x=BORDER, y=BORDER)
splash.append(inner_sprite)

# Szöveg
text = "   CircuitPython\r\n     tanfolyam\r\n Hobbielektronika"
text_area = label.Label(terminalio.FONT, text=text, color=0xFFFF00, x=10, y=15)
splash.append(text_area)

while True:
    pass
```

A displayio modul osztályai

- A **displayio** modul API dokumentációja
- A **Group** (vagy a legfelső Group) végeredményben az, ami a kijelzőn megjelenik a **show()** metódus meghívásakor
- A **TileGrid Bitmap** és **Palette** elemekből áll
- **Bitmap** és **Palette** kapcsolata:

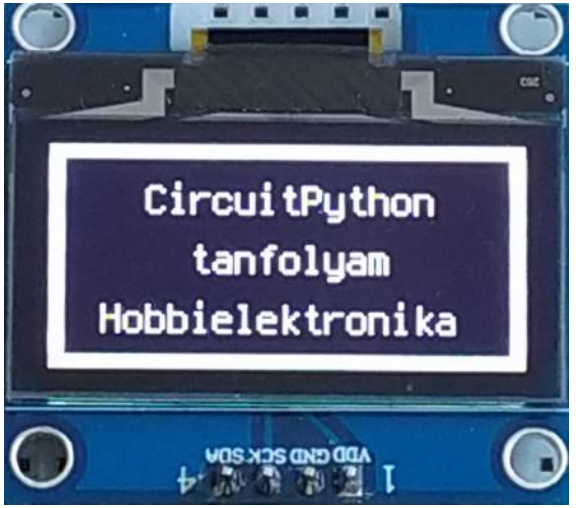




SH1106_displayio_test.py – hogy működik?

- Ez az általános gyűjtő objektum:

```
splash = displayio.Group()
display.show(splash)
```



- A befoglaló fehér téglalap:

```
color_bitmap = displayio.Bitmap(128, 64, 1)
color_palette = displayio.Palette(1)
color_palette[0] = 0xFFFFFF  # White
bg_sprite = displayio.TileGrid(color_bitmap,
pixel_shader=color_palette, x=2, y=0)
splash.append(bg_sprite)            SH1106 esetén eltolás kell
```

- A belső fekete téglalap:

```
inner_bitmap=displayio.Bitmap(120,56,1) # Draw smaller rectangle
inner_palette = displayio.Palette(1)
inner_palette[0] = 0x000000  # Black
inner_sprite = displayio.TileGrid(inner_bitmap,
pixel_shader=inner_palette, x=6, y=4)
splash.append(inner_sprite)            SH1106 esetén eltolás kell
```

SH1106_displayio_test.py – hogy működik?

- Szövegcímke (Text Label) rajzolása:

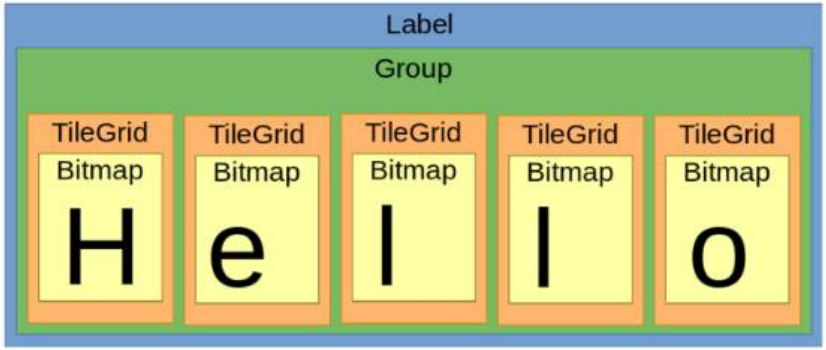
```
text = "   CircuitPython\r\n     tanfolyam\r\n Hobbielektronika"
text_area = label.Label(terminalio.FONT, text=text, color=0xFFFF00, x=10, y=15)
splash.append(text_area)
```

- A CircuitPython Display_Text Library tananyag szerint az **adafruit_display_text** modul kétféle szövegcímke típust kezel, közülük úgy lehet választani, hogy a

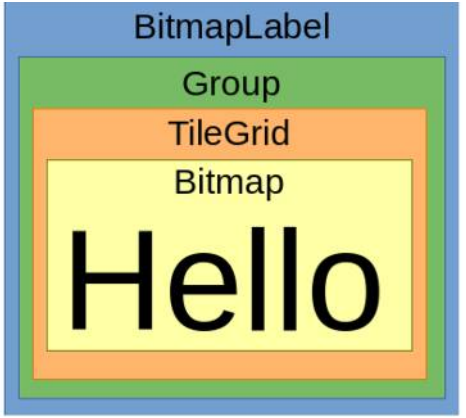
```
from adafruit_display_text import label
```
sor helyett

```
from adafruit_display_text import bitmap_label
```
sort írunk

Label (hagyományos)



BitmapLabel



SH1106_displayio_test_bitmaplabel.py

- A **BitmapLabel**-re történő áttéréshez csak minimális változtatást kell tennünk az előző **SH1106_displayio_test.py** programon:
- A program elején a

```
from adafruit_display_text import label
```

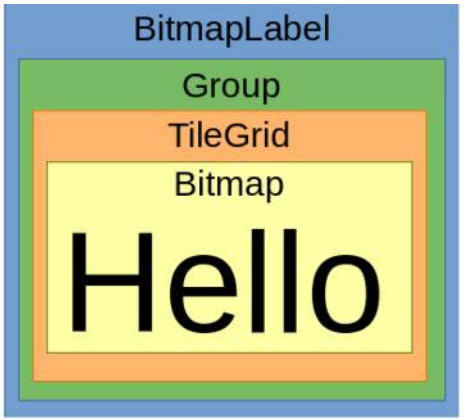
sor helyett

```
from adafruit_display_text import bitmap_label
```

sort írunk

- A szövegcímke (BitmapLabel) rajzolása pedig így módosul:

```
text = "   CircuitPython\n     tanfolyam\n Hobbielektronika"
text_area = bitmap_label.Label(terminalio.FONT, text=text, color=0xFFFF00, x=10, y=15)
splash.append(text_area)
```



Az ESP32 C3 Super Mini kártya kivezetései