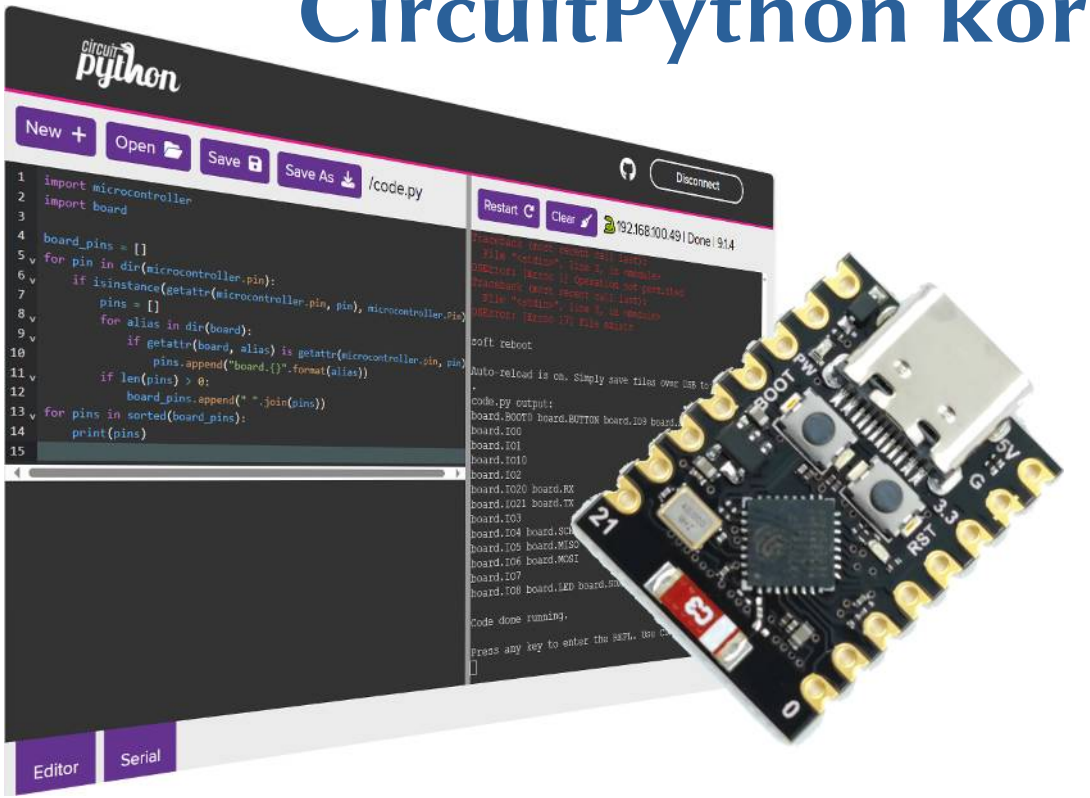
ESP32-C3 mikrovezérlők programozása CircuitPython környezetben



4. Az ST7735 színes TFT kijelző

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

Felhasznált és ajánlott irodalom

- **Tananyagok, útmutatók:**
  - Carter Nelson: CircuitPython Display Support Using displayio
  - Tim C.: CircuitPython Display_Text Library
  - Ruiz Brothers: Custom Fonts for CircuitPython Displays
  - Anna Barela: Creating Slideshows in CircuitPython

- **CircuitPython kiegészítő programkönyvtárak:**
  - Adafruit_ST7735R – displayio driver for the ST7735R 1.8” TFT
  - Adafruit_CircuitPython_Display_Text – text labels
  - Adafruit_CircuitPython_Bitmap_Font – custom font support
  - Adafruit_CircuitPython_ImageLoad – load image files
  - Adafruit_CircuitPython_Display_Shapes – lines, circles, triangles etc.
  - Adafruit_Slideshow – helper library for displaying a slideshow

- **Adatlapok és dokumentáció:**
  - Sitronix: ST7735 adatlap

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

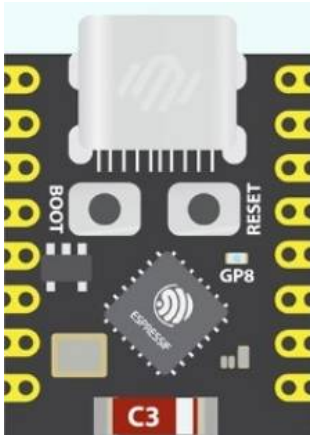
Display és Display bus

DISPLAY BUS

I2CDisplayBus

FourWire

ParallelBus



- A kijelzők meghajtása két külön rétegből áll:
  - a **Display Bus** felel azért, hogy a mikrokontroller fizikailag kommunikáljon a kijelzővel (SPI, I²C, párhuzamos busz stb.)
  - a **Display** objektum pedig a megjelenítést kezeli (méret, inicializálás, root group, frissítés), ehhez többnyire *meghajtó programkönyvtárt* kell telepíteni
- A folyamat mindig ugyanaz: először létre kell hozni a Display Bus-t, majd ezt át kell adni a Display példánynak.

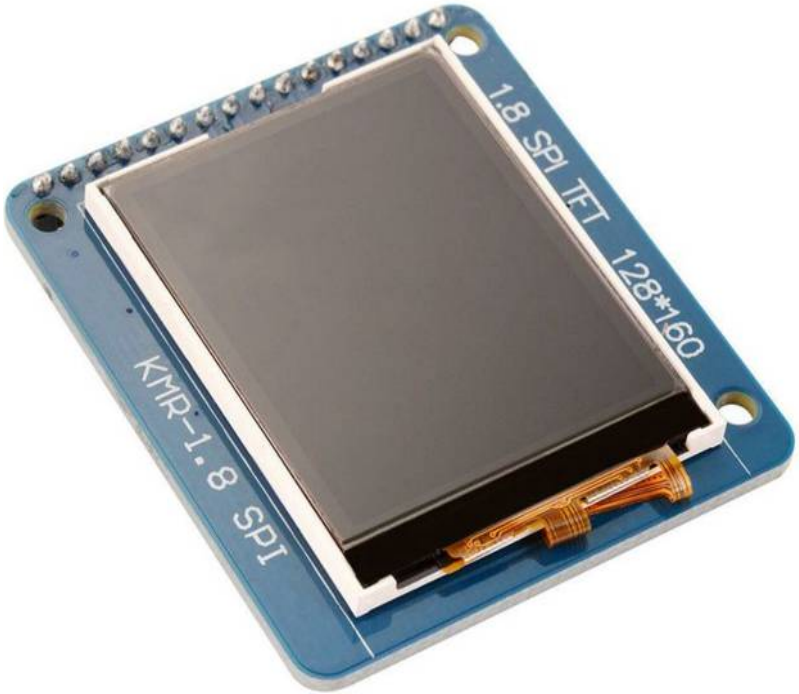
Változások a CircuitPython-ban

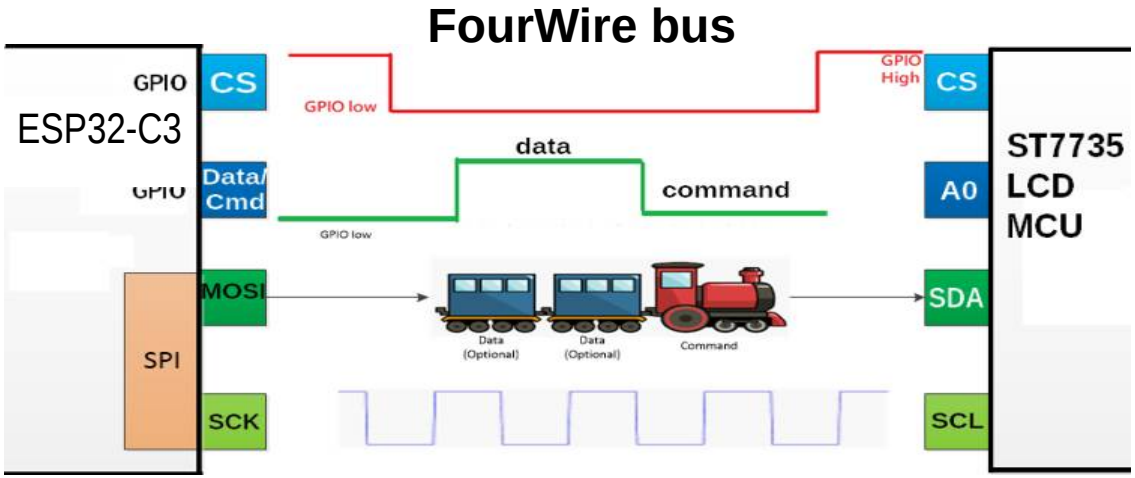
- A korábbi években már foglalkoztunk **CircuitPythonnal** és a **displayio** modullal, de azóta változtatások és átszervezések történtek a firmware-ben, így a régi mintaprogramok nem kompatibilisek a **CiruitPython** újabb verzióival
- A CircuitPython 10-től kezdve a felhasználói kódot frissíteni kell, hogy az új kijelző- és kijelzőbusz könyvtárakat használja, amelyek kikerültek a **displayio**-ból
  - `displayio.Display → busdisplay.BusDisplay`
  - `displayio.FourWire → fourwire.FourWire`
  - `displayio.EPaperDisplay → epaperdisplay.EPaperDisplay`
  - `displayio.I2CDisplay → i2cdisplaybus.I2CDisplayBus`
- Egy másik fontos változás a **displayio**-ban, hogy legfelső szintű **Group** megjelenítése már nem a **Display.Show()** metódussal történik:
  - `display.show(some_group) → display.root_group = some_group`

ST7735 1.8” színes TFT kijelző

- Vezérlő: **Sitronix ST7735**
- Interfész: SPI, csak írás (max. 10 – 15 MHz)
- Data/Command választás külön vonalon
- Kijelző: TFT, 65 536 szín (16 bit RGB, 5-6-5)
- Felbontás: **128 x 160** pixel
- SD kártya foglalat (SPI mód)
- Háttérvilágítás (LED anód/katód)

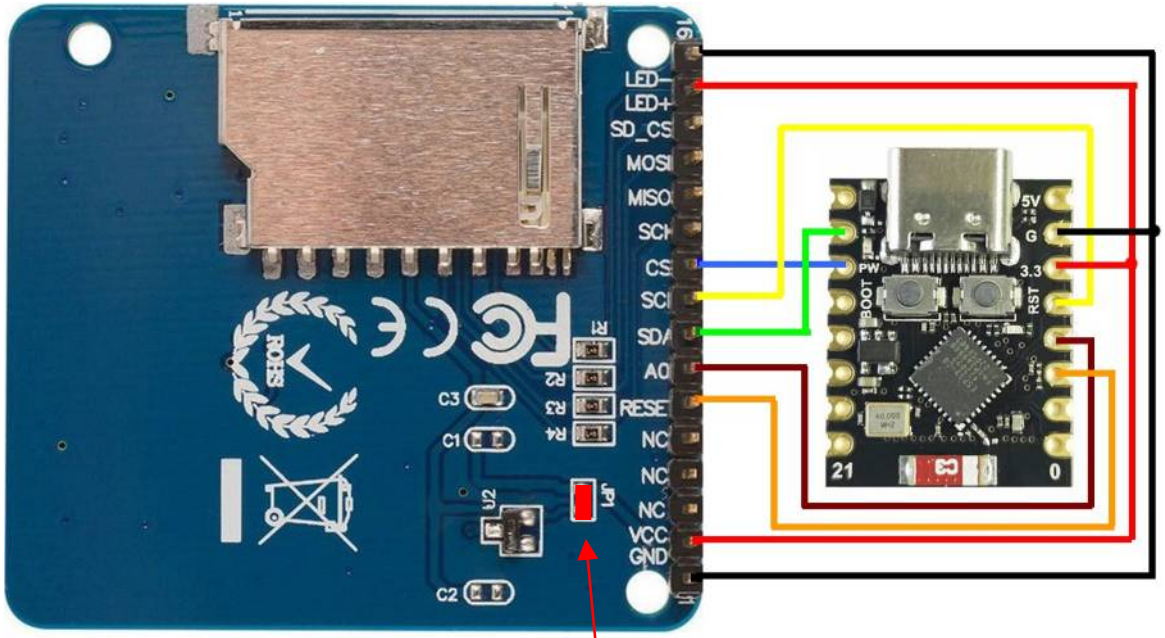
12-bites, 16 bites és 18 bites módban is használható





ST7735 kijelző bekötési vázlat

- **GND** – a tápegység közös pontja
- **VCC** – tápfeszültség 5V / 3.3V (JP1)
- **NC** – nincs bekötve
- **RESET** – HW reset ← **IO2**
- **A0** – adat/parancs ← **IO3**
- **SDA** – SPI adatvonal ← **IO6**
- **SCL** – SPI órajel ← **IO4**
- **CS** – SPI eszközválasztó jel ← **IO7**
- **SCK** – SD kártya SPI órajel
- **MISO** – SD kártya adatkimenete
- **MOSI** – SD kártya adatbemenete
- **SD_CS** – SD kártya eszközválasztó jel (0: aktív)
- **LED+** – kijelző háttérvilágítás (anód) ← **3V3**
- **LED–** – kijelző háttérvilágítás (katód) ← **GND**

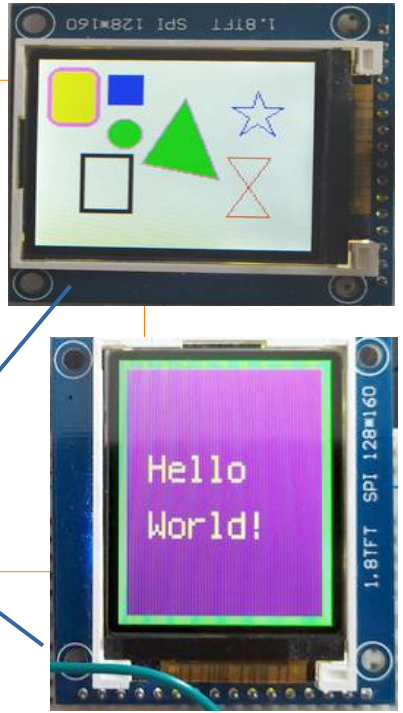


JP1 összekötése esetén 3,3 V tápfeszültség kell, **nyitott JP1** esetén pedig 5 V

Az Adafruit_ST7735R programkönyvtár

- Az Adafruit_ST7735R programkönyvtár elnevezése nem következetes, valójában ez egy ***displayio*** meghajtó
  (**R** betű nélküli verziója is van, de az az **ST7735B** vezérlőhöz való)
- Dokumentáció: displayio driver for ST7735R TFT-LCD displays
- Inicializálás:

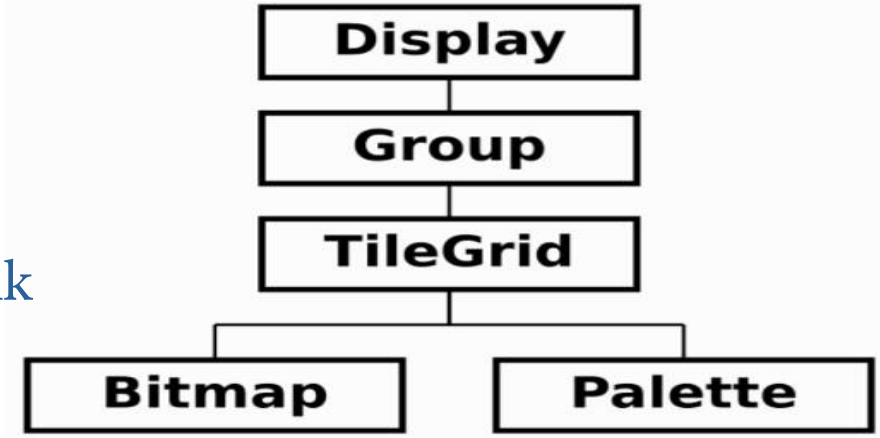
```
import board, displayio
from fourwire import FourWire
from adafruit_st7735r import ST7735R
# Release any resources currently in use for the displays
displayio.release_displays()
# create the spi device and pins we will need
spi = busio.SPI()
display_bus = FourWire(spi,command=board.IO3,chip_select=board.IO7,reset=board.IO2)
display=ST7735R(display_bus,width=160,height=128,rotation=90,bgr=True) # Landscape

# Vagy álló formátumhoz:
display=ST7735R(display_bus,width=128,height=160,rotation=0,bgr=True)  # Portrait
```

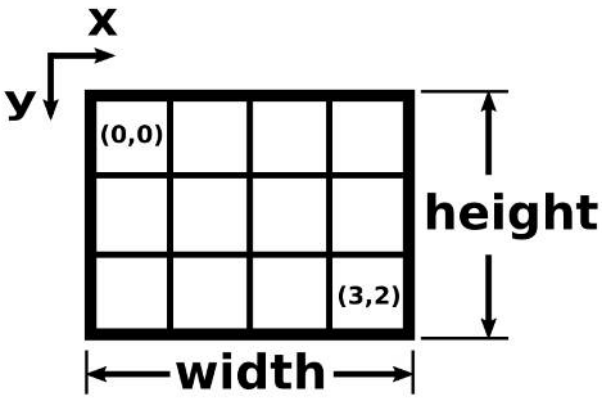


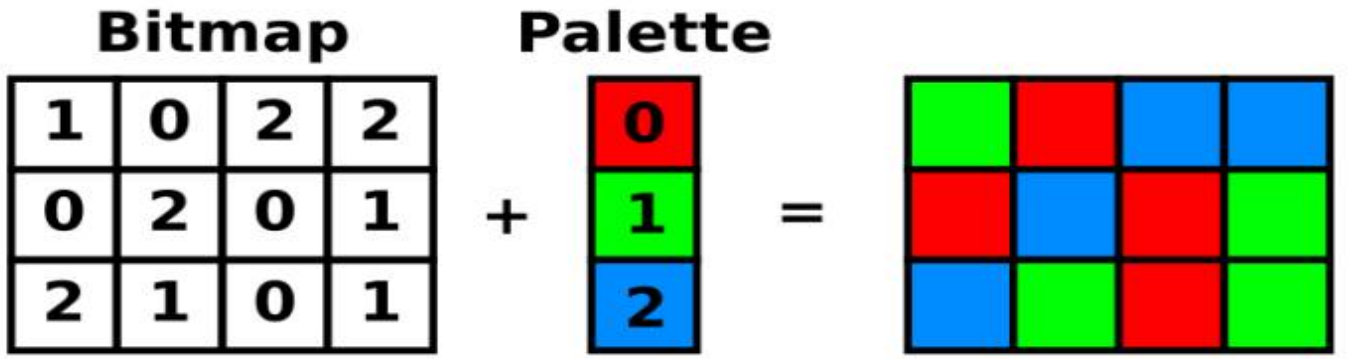
- Az **Adafruit_ST7735R** könyvtár nem beépített könyvtár, nekünk kell telepíteni

Emlékeztető: A displayio modul

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

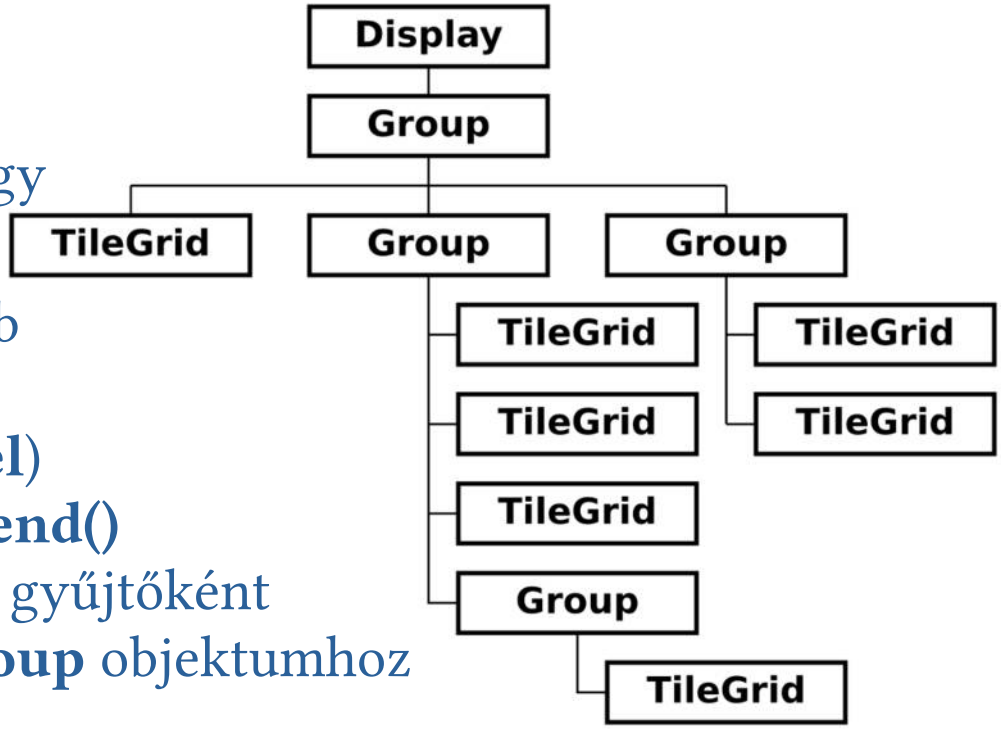






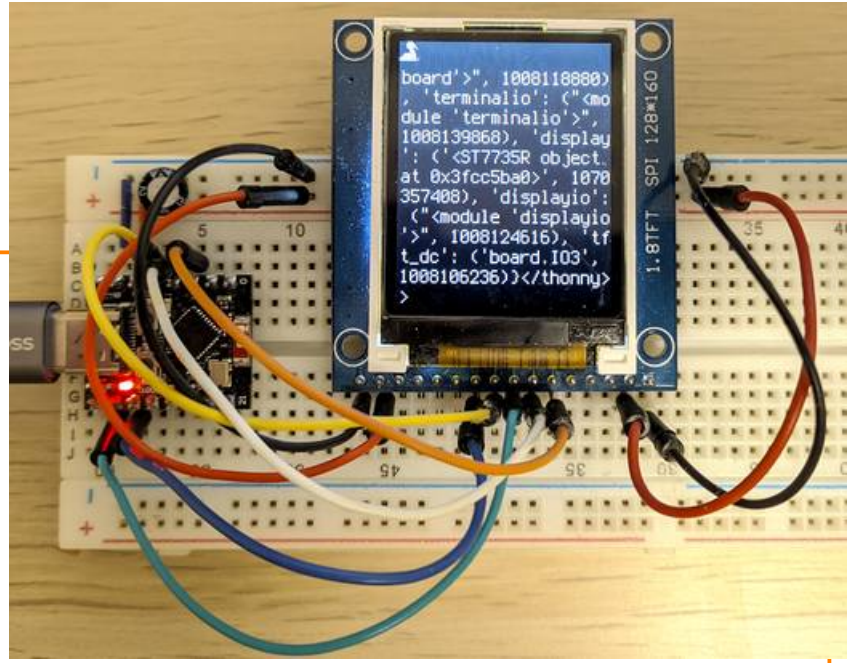
Hierarchia és összeillesztés

- A **displayio** modul osztályaiból példányosított objektumokból komplex ábrákat is összerakhatunk de be kell tartanunk az alábbi szabályokat:
- A **Display** objektum csak **Group** típusú objektumot tud megjeleníteni
- A **Display** objektum egyidejűleg csak egy **Group** objektumot tud megjeleníteni
- A **Group** típusú objektum egy vagy több **TileGrid** és/vagy **Group** típusú (illetve a **Group**-ból származtatott **Shape, Label**) objektumot tartalmazhat, ezeket az **append()** metódussal fűzhetjük hozzá az általános gyűjtőként funkcionáló, a hierarchia tetején ülő **Group** objektumhoz



ST7735R_simpletest.py – 2/1.

- Az ESP32-C3-ra adaptált Adafruit mintapélda elején importáljuk a felhasználni kívánt modulokat és inicializáljuk az SPI csatornát, majd **displayio** kijelzőként inicializáljuk az **ST7735R** képernyőt is




```
import board
import displayio
import terminalio
from adafruit_display_text import label
from fourwire import FourWire
from adafruit_st7735r import ST7735R

# Release any resources currently in use for the displays
displayio.release_displays()

spi = board.SPI()
tft_cs = board.IO7
tft_dc = board.IO3

display_bus = FourWire(spi, command=tft_dc, chip_select=tft_cs, reset=board.IO2)
display = ST7735R(display_bus, width=128, height=160, rotation=0, bgr=True)  # Portrait format
```

Innen kapjuk a terminalio.FONT-ot

ST7735R_simpletest.py – 2/2.

```
splash = displayio.Group()                  # Make the display context
display.root_group = splash                 # For CircuitPython versions prior to 8.0, use
                                            # display.show(my_display_group) instead.

# Draw a big (full screen) rectangle
color_bitmap = displayio.Bitmap(128, 160, 1)
color_palette = displayio.Palette(1)
color_palette[0] = 0x00FF00  # Bright Green
bg_sprite = displayio.TileGrid(color_bitmap, pixel_shader=color_palette, x=0, y=0)
splash.append(bg_sprite)

# Draw a smaller inner rectangle
inner_bitmap = displayio.Bitmap(118, 150, 1)
inner_palette = displayio.Palette(1)
inner_palette[0] = 0xAA0088  # Purple
inner_sprite = displayio.TileGrid(inner_bitmap, pixel_shader=inner_palette, x=5, y=5)
splash.append(inner_sprite)

# Draw a text label ("Hello World!") scaled ×2
# Text is placed inside a subgroup so the entire label can be scaled uniformly.
text_group = displayio.Group(scale=2, x=18, y=64)
text = "Hello \nWorld!"
text_area = label.Label(terminalio.FONT, text=text, color=0xFFFF00) #Yellow
text_group.append(text_area)  # Subgroup for text scaling
splash.append(text_group)
```

ST7735R_colored_labels.py – 2/1.

- Az Adafruit mintapéldája színes feliratokat ír ki ***displayio*** kijelzőn. A programot **ESP32-C3**-ra adaptáltuk és átírtuk, hogy a címkék pozicionálását is bemutassuk

```
import board
import displayio
import terminalio
from adafruit_display_text import label
from fourwire import FourWire
from adafruit_st7735r import ST7735R

# Release any resources currently in use for the displays
displayio.release_displays()

spi = board.SPI()
tft_cs = board.IO7
tft_dc = board.IO3

display_bus = FourWire(spi,command=tft_dc,chip_select=tft_cs,reset=board.IO2)
display = ST7735R(display_bus, width=128, height=160, rotation=0, bgr=True)

# Make the display context
splash = displayio.Group()
display.root_group = splash
```

Innen kapjuk a terminalio.FONT-ot

ST7735R_colored_labels.py – 2/2.

```
# Left column: RGBW
text_group_left = displayio.Group(x=10, y=20)
text_area_red = label.Label(terminalio.FONT, text="RED", color=0xFF0000, x=0, y=0)
text_group_left.append(text_area_red)
text_area_green = label.Label(terminalio.FONT, text="GREEN", color=0x00FF00, x=0, y=12)
text_group_left.append(text_area_green)
text_area_blue = label.Label(terminalio.FONT, text="BLUE", color=0x0000FF, x=0, y=24)
text_group_left.append(text_area_blue)
text_area_white = label.Label(terminalio.FONT, text="WHITE", color=0xFFFFFF, x=0, y=36)
text_group_left.append(text_area_white)
splash.append(text_group_left)
# Right column: CMYK + Grey
text_group_right = displayio.Group(x=74, y=20)
text_area_cyan = label.Label(terminalio.FONT, text="CYAN", color=0x00FFFF, x=0, y=0)
text_group_right.append(text_area_cyan)
text_area_magenta = label.Label(terminalio.FONT, text="MAGENTA", color=0xFF00FF, x=0, y=12)
text_group_right.append(text_area_magenta)
text_area_yellow = label.Label(terminalio.FONT, text="YELLOW", color=0xFFFF00, x=0, y=24)
text_group_right.append(text_area_yellow)
text_area_grey = label.Label(terminalio.FONT, text="GREY", color=0x808080, x=0, y=36)
text_group_right.append(text_area_grey)
splash.append(text_group_right)
while True:
    pass
```

A Group pozíciójához képest relatívak a Label koordinátái

Képek megjelenítése displayio-val

- A ***displayio*** modullal kezelt kijelzőkön bitmap képeket is betölthetünk és megjeleníthetünk, amire kétféle lehetőség áll rendelkezésünkre
- **ImageLoad** (az ***adafruit_imageload*** könyvtár kell hozzá)
  - A képet RAM-ba tölti (emiatt nagyobb a memóriaigény)
  - A megjelenítés gyors, a betöltött kép módosítható
  - Többféle képformátumot is kezel (BMP, GIF, PNG, JPG)
  - Az **adafruit_imageload** könyvtár mappát a /**lib** mappába be kell másolni
- **OnDiskBitmap** (a ***displayio*** része)
  - A képet közvetlenül a fájlból tölti át a kijelzőre
  - A megjelenítés lassabb, menet közben kell a pixel színezést konvertálni
  - Nem kell hozzá kiegészítő könyvtárat telepíteni
  - Kevesebb RAM kell hozzá, de a kép nem módosítható
- Mindkét esetben a kép **TileGrid** objektumként kerül a kijelzőre.

Az ImageLoad programkönyvtár használata

- **Imageload** az adafruit_imageload könyvtár fő osztálya, amely egyszerű módot kínál a bitképek dekódolására és megjelenítésére
- Használatához először importáljuk az adafruit_imageload könyvtárat

```
import adafruit_imageload
```

- Töltsük be a képfájlt a memóriába, mint Bitmap és Palette objektumok

```
my_bitmap, my_palette = adafruit_imageload.load("/image.bmp",
  bitmap=displayio.Bitmap, palette=displayio.Palette)
```

- Készítsünk TileGrid-et a `my_bitmap` és `my_palette` objektumokból

```
my_tilegrid = displayio.TileGrid(my_bitmap, pixel_shader=my_palette)
```

- Adjuk hozzá TileGrid-et az aktuális displayio.Group példányhoz

```
my_display_group.append(my_tilegrid)
```

ST7735R_imageload_demo.py

- Az **ImageLoad** programkönyvtárral betöltjük és *portrait* módban megjelenítjük a **buek.jpg** képet


```
import board, displayio
from fourwire import FourWire
from adafruit_st7735r import ST7735R
from adafruit_imageload import load

displayio.release_displays()
spi = board.SPI()
display_bus = FourWire(spi, command=board.IO3, chip_select=board.IO7, reset=board.IO2)
display = ST7735R(display_bus, width=128, height=160, rotation=0, bgr=True) # Portrait mode

splash = displayio.Group()
display.root_group = splash
bitmap, palette = load("buek.jpg")
tile_grid = displayio.TileGrid(bitmap, pixel_shader=palette)
splash.append(tile_grid)

while True:
    pass
```

Az OnDiskBitmap programkönyvtár használata

- Az **OnDiskBitmap** a ***displayio*** modul része és nagyon egyszerűen használható bitképek dekódolására és megjelenítésére
- Első lépésként létrehozunk egy OnDiskBitmap-objektumot, amely olvasási elérést biztosít egy fájlban tárolt bitmap képhez anélkül, hogy azt RAM-ba töltené

```
my_bitmap = displayio.OnDiskBitmap("my_bitmap.bmp")
```

- A második lépésben ebből az objektumból hozunk létre egy TileGrid-et, az automatikus színkonvertálást használva:

```
my_tilegrid = displayio.TileGrid(my_bitmap,
        pixel_shader=my_bitmap.pixel_shader)
```

- Adjuk hozzá TileGrid-et az aktuális displayio.Group példányhoz

```
my_display_group.append(my_tilegrid)
```

ST7735R_bitmap_demo.py

- Az **OnDiskBitmap** módszerrel betöltjük és *landscape* módban megjelenítjük a **bitmap_demo.bmp** képet


```
import board, displayio
from fourwire import FourWire
from adafruit_st7735r import ST7735R

displayio.release_displays()
spi = board.SPI()
display_bus = FourWire(spi, command=board.IO3, chip_select=board.IO7, reset=board.IO2)
display = ST7735R(display_bus, width=160, height=128, rotation=90, bgr=True) # Landscape mode

splash = displayio.Group()
display.root_group = splash

bitmap = displayio.OnDiskBitmap("bitmap_demo.bmp")
tile_grid = displayio.TileGrid(bitmap, pixel_shader=getattr(bitmap, "pixel_shader", None))
splash.append(tile_grid)

while True:
    pass
```

Itt egy biztonságosabb **pixel_shader** kezelést használunk, amivel elkerüljük a hibára futást, ha a BMP-nek nincs **pixel_shader** attribútuma (pl. a 24 bites színmélységű képeknél)

ST7735R_slideshow.py

- Készítsünk diabemutatót (slideshow) egyszerűen:
  - Telepítsük az **adafruit_slideshow** könyvtárat!
  - Helyezzük el a képeket (.bmp) az /**images** nevű könyvtárba
  - Futtassuk az alábbi programot:

```
import board, displayio, time
from fourwire import FourWire
from adafruit_st7735r import ST7735R
from adafruit_slideshow import PlayBackOrder, SlideShow

displayio.release_displays()
spi = board.SPI()
display_bus = FourWire(spi, command=board.IO3, chip_select=board.IO7, reset=board.IO2)
display = ST7735R(display_bus, width=128, height=160, rotation=0, bgr=True) # Portrait mode

# Create the slideshow object that plays through once alphabetically.
slideshow = SlideShow(display,folder="/images",loop=True,
                      order=PlayBackOrder.ALPHABETICAL, dwell=5)

while slideshow.update():
    pass
```

Az ESP32 C3 Super Mini kártya kivezetései