Az RP2040 mikrovezérlő programozása Arduino IDE környezetben – 4. rész

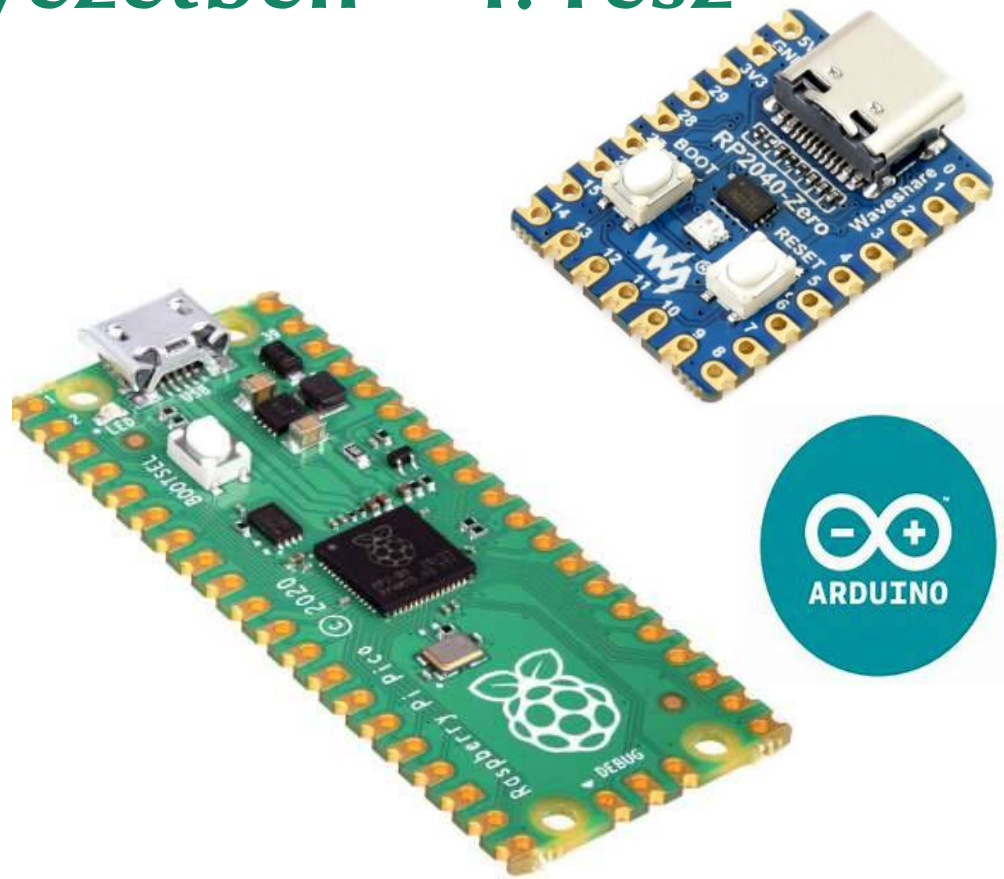
Bink2 | Arduino 1.8.19 (Windows Store 1.8.57.0)

File Edit Sketch Tools Help

Bink2 §

```
#include <mbed.h>

mbed::DigitalOut led(LED1);

void setup() {
}

void loop() {
  led.write(1);                       // Switch the LED on
  rtos::ThisThread::sleep_for(1000); // Wait for a second
  led.write(0);                       // Switch the LED off
  rtos::ThisThread::sleep_for(1000); // Wait for a second
}
```

15 Raspberry Pi Pico on COM14



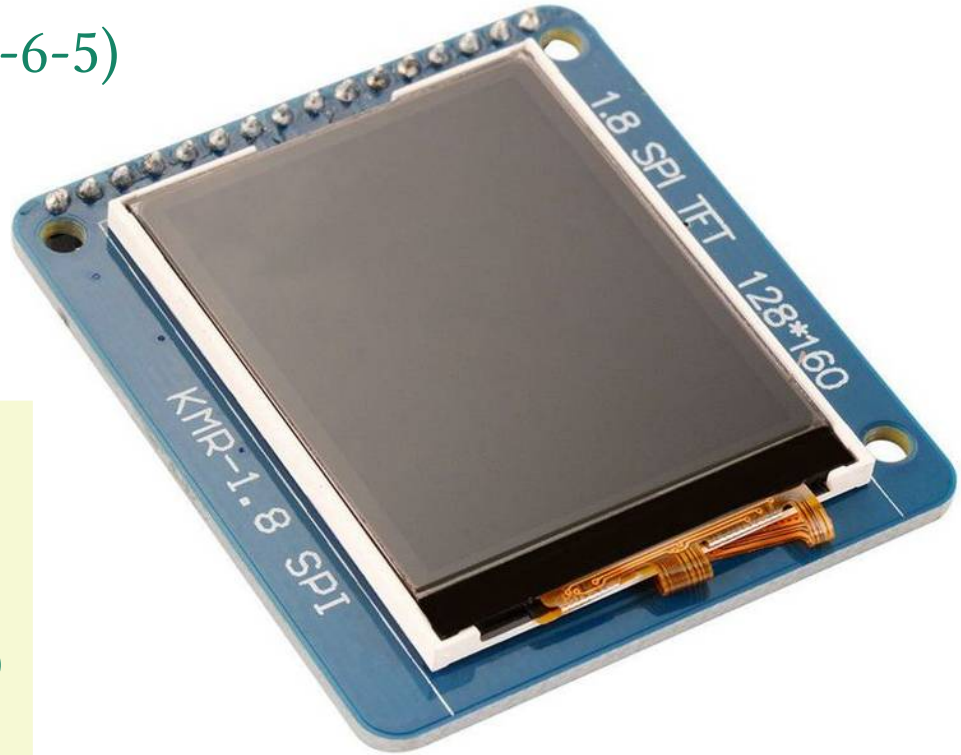
Felhasznált és ajánlott irodalom

- Raspberry Pi: Pico-series Microcontrollers
- Raspberry Pi: RP2040 adatlap (PDF)
- Raspberry Pi: Raspberry Pi Pico Datasheet
- Raspberry Pi: Raspberry Pi Pico-series C/C++ SDK
- Raspberry Pi: Getting started with Raspberry Pi Pico-series Microcontrollers
- Waveshare: RP2040-Zero, a Pico-like MCU Board

- Ralphjy: Program RPi Pico using Mbed library with Arduino IDE
- Learn Embedded Systems: Basic Multicore Pico Project
- BigG: Four Multicore C programs for Raspberry Pi Pico using Arduino IDE
- Valentin Milea: DHT sensor library for the Raspberry Pi Pico
- Alan Yorinks: NeoPixelConnect

ST7735 1.8” színes LCD kijelző

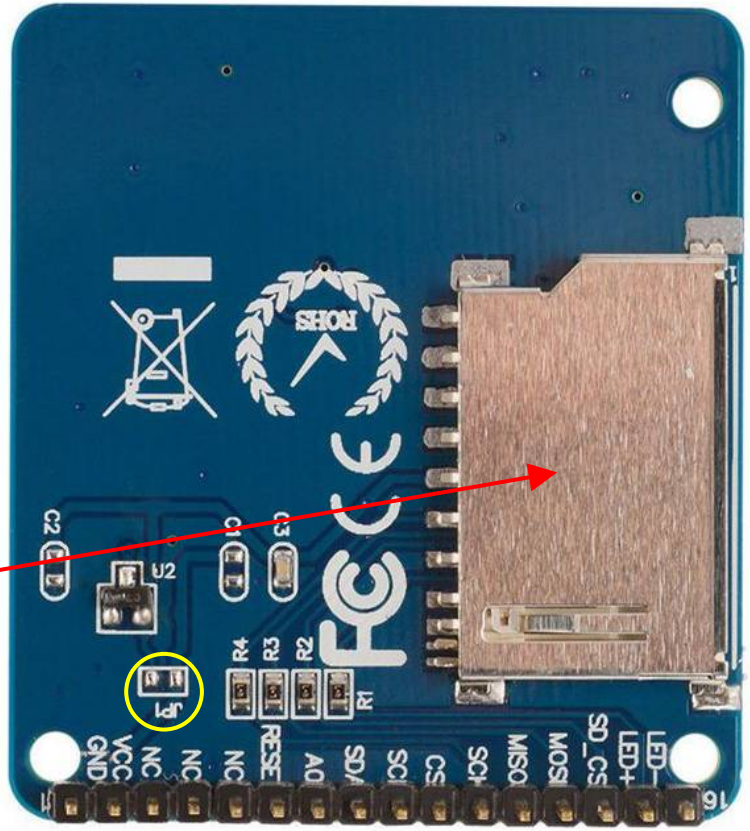
- Vezérlő: **Sitronix ST7735**
- Interfész: SPI, csak írás (max. 10 – 15 MHz)
- Data/Command választás külön vonalon
- Kijelző: TFT, 65 536 szín (16 bit RGB, 5-6-5)
- Felbontás: **128 x 160** pixel
- SD kártyahely: csak SPI módban használható, a kivezetések: **SD_CS, MOSI, MISO, SCK**



- A kijelzővel a 2019/2020-as évadban már részletesen foglalkoztunk:
  - 2020. május 28. (Arduino tanfolyam)
  - 2020. május 21. (STM32/ARM Keil IDE)
  - 2020. június 6. (STM32/ARM Keil IDE)

A kivezetések

- **GND** – a tápegység közös pontja
- **VCC** – tápfeszültség (5V, vagy **JP1** zárása után 3.3V)
- **NC** – nincs bekötve
- **RESET** – kijelző reset jel (0: aktív, 1: inaktív)
- **A0** – kijelző regisztertválasztó jel (RS, DC)
- **SDA** – kijelző SPI adatbemenet (MOSI jel)
- **SCL** – kijelző SPI órajel
- **CS** – kijelző SPI eszközválasztó jel (0: aktív)
- **SCK** – SD kártya SPI órajel
- **MISO** – SD kártya adatkimenete
- **MOSI** – SD kártya adatbemenete
- **SD_CS** – SD kártya eszközválasztó jel
- **LED+** – kijelző háttérvilágítás (anód)
- **LED–** – kijelző háttérvilágítás (katód)

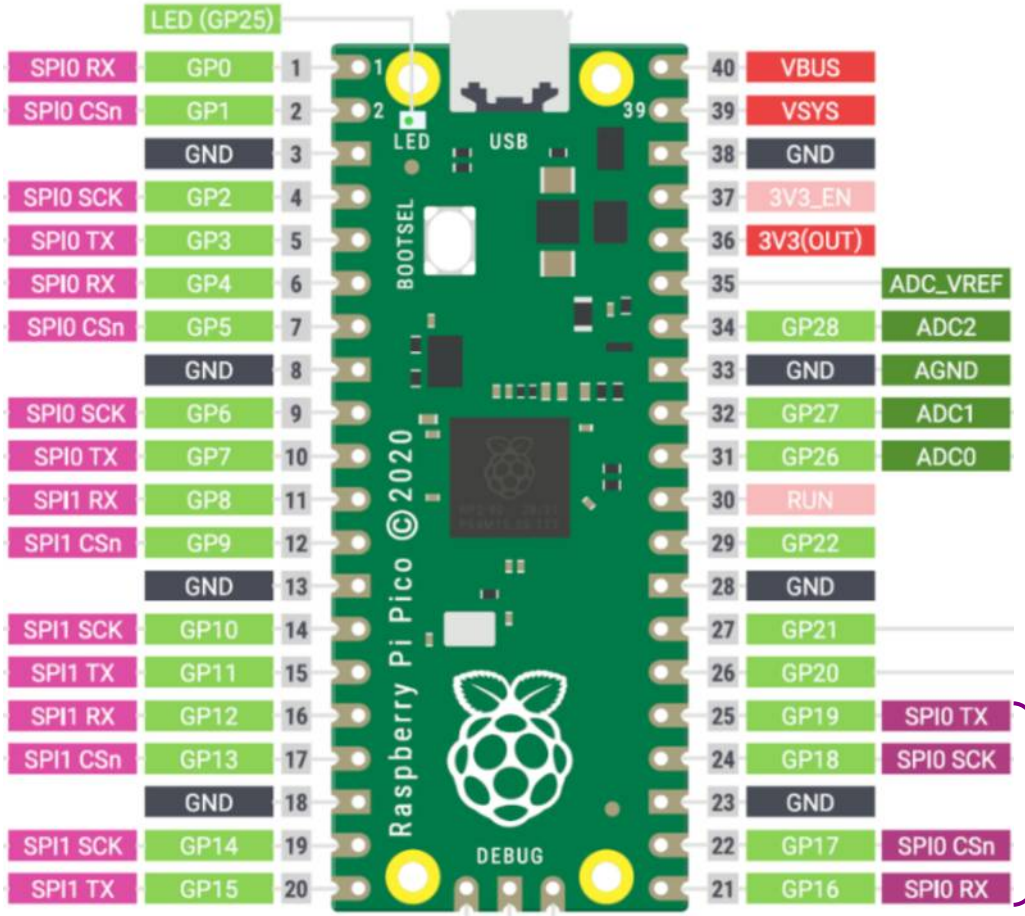


Bekötési vázlat

- Ebben az elrendezésben az **SPI0** csatorna alapértelmezett kivezetéseit használjuk.
- **RESET, A0** és **CS** bármelyik **GPIO** lábra rákötheto

ST7735	RP2040
GND	GND
VCC	3V3
RESET	GPIO1
A0	GPIO0
SDA	GPIO19
SCL	GPIO18
CS	GPIO17
LED+	3V3
LED-	GND

Az SD kártya foglalaltot most nem kötjük be



Az Adafruit-ST7735 programkönyvtár

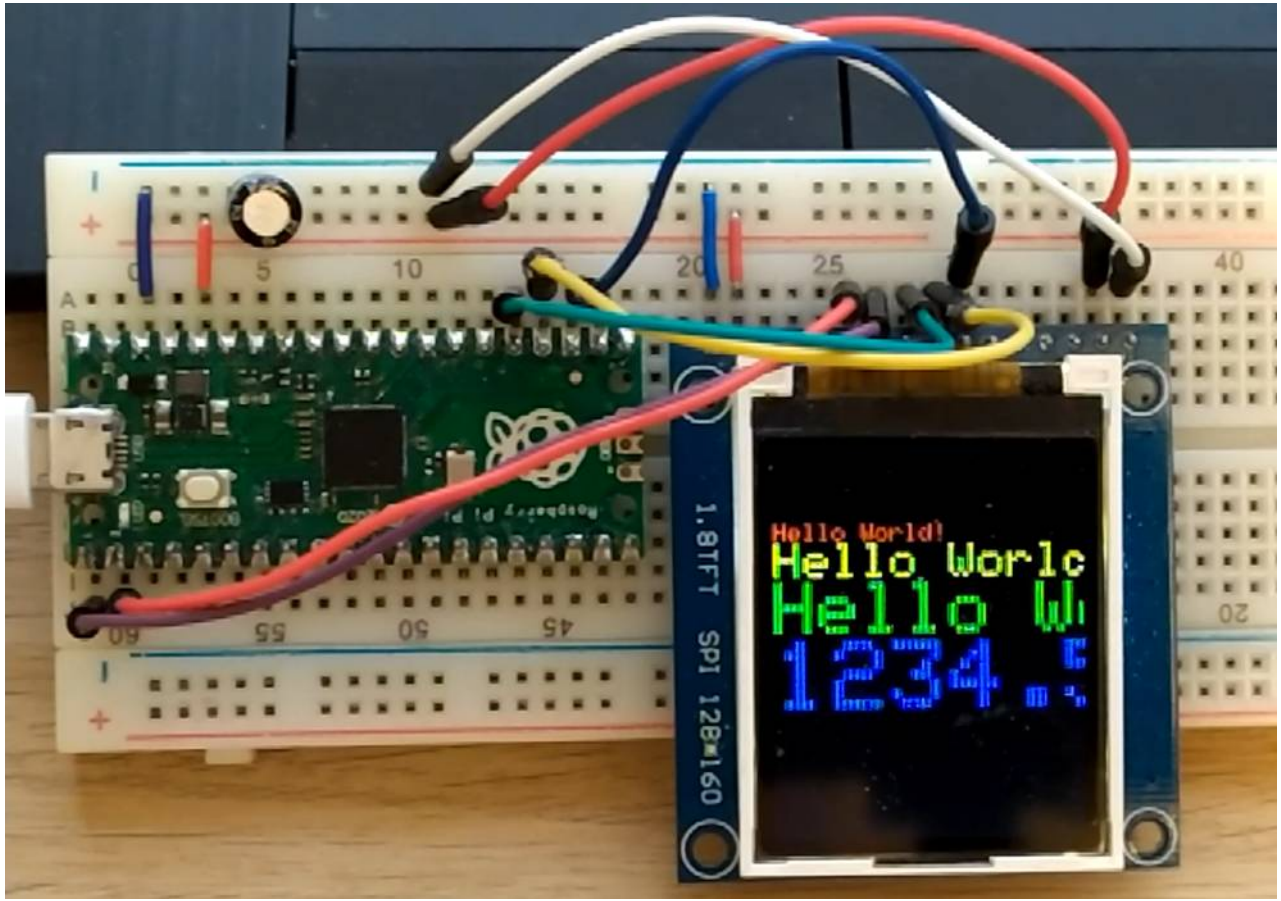
- Bár az **Arduino IDE** telepítőcsomagjában van egy beépített **TFT** nevű könyvtár, ami az **ST7735** kijelzőkhöz használható, de az **RP2040** mikrovezérlőhöz azt nem lehet lefordítani, ezért telepítsük az **Adafruit_ST7735_and_ST7789** programkönyvtárat
- A telepítéshez a **Tools** menüben a **Library Manager** menüpontban adjuk meg az **ST7735** keresőszót és válasszuk ki, majd telepítsük a fenti csomagot
- Telepítenünk kell az **Adafruit_GFX_Library**-t is
- A programkönyvtár számos mintapéldát is tartalmaz, ezek közül kettőt adaptáltunk a **Raspberry Pi Pico** kártyához, az előző oldalon bemutatott kapcsolást figyelembe véve:
  - **graphicstest.ino** - különféle geometriai alakzatokat és rövid szövegeket jelenít meg, bemutatva a kijelző képességeit, egyúttal ellenőrizhetjük a kijelző konfigurálását is. Ennek a programnak egy másik változatát is elkészítettük, amelyben megmutatjuk, hogy hogyan lehet az alapértelmezettől eltérő SPI lábkiosztást használni.
  - **displayOnOffTest.ino** – egy GPIO kivezetést nyomógomb bemenetnek használ, amellyel ki-bekapcsolhatjuk a kijelzőt. A program méri és a kijelzőn kiírja a futó időt

ST7735_demo.ino (részlet)

```
#include <Adafruit_GFX.h>    // Core graphics library
#include <Adafruit_ST7735.h> // Hardware-specific library for ST7735
#include <SPI.h>             // The default pre-initialized SPI channel
#define TFT_CS        17     // Chip select pin
#define TFT_RST        1     // RESET pin
#define TFT_DC         0     // A0 (data/command selector) pin
Adafruit_ST7735 tft = Adafruit_ST7735(TFT_CS, TFT_DC, TFT_RST);

void setup(void) {
  Serial.begin(9600);
  tft.initR(INITR_BLACKTAB); // Init ST7735S chip, black tab
  tft.setSPISpeed(10000000); // You may safely use 10 - 15 MHz
  tft.setRotation(2);        // Orientation: portrait, upside down

  uint16_t time = millis();
  tft.fillScreen(ST77XX_BLACK);
  time = millis() - time;
  Serial.print("Fill time = ");
  Serial.println(time, DEC);
  . . .
}
```

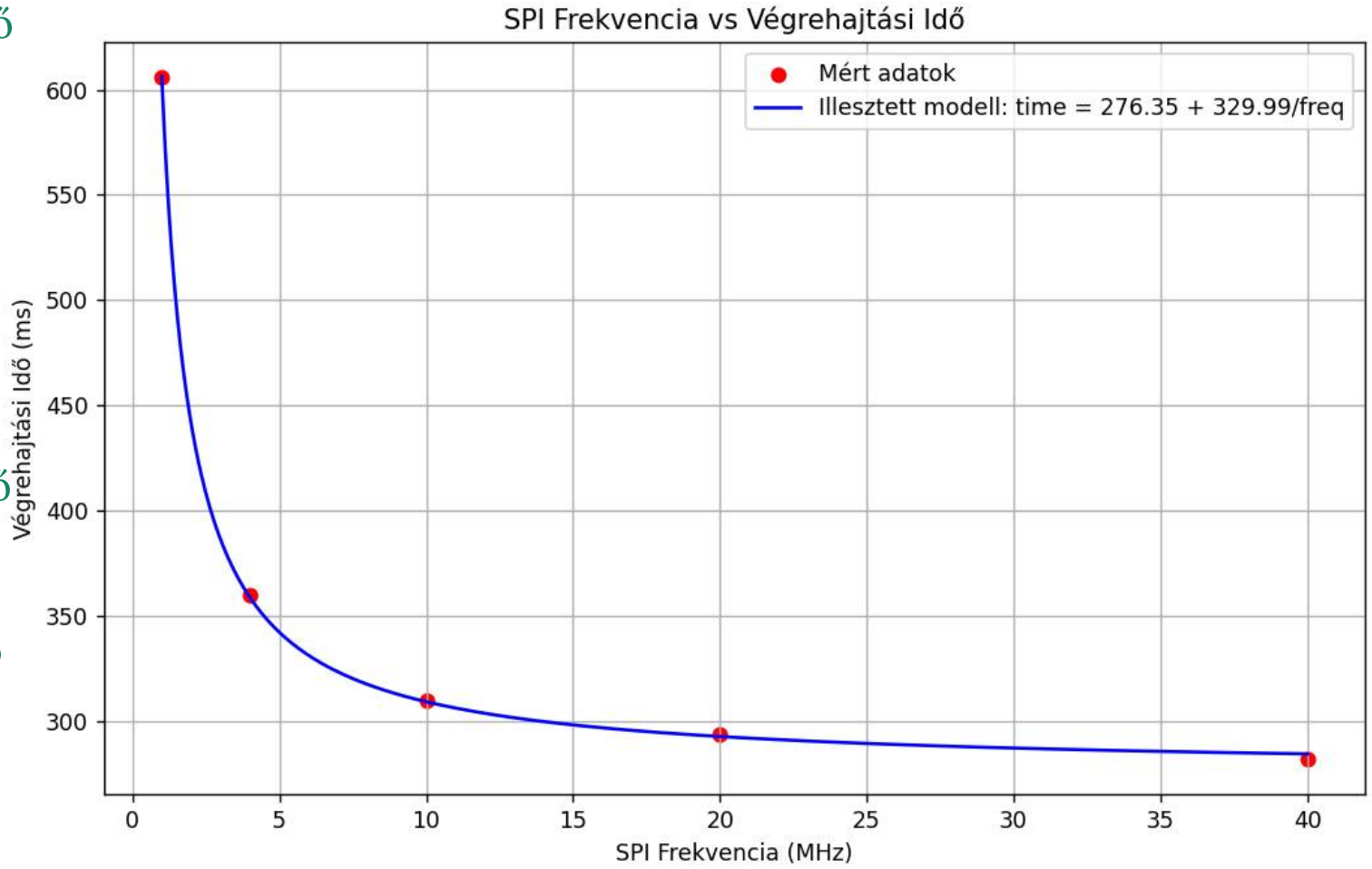
A **Graphicstest.ino** program adaptálásához csak a program elejét kellett módosítani, a fentiek szerint, hogy az **ST7735 kijelző** az előzőekben bemutatott bekötésnek megfelelően legyen inicializálva

ST7735_demo.ino – futási eredmény



ST7735_demo – a futási idő elemzése

- A program méri az első képernyőtörlési parancs végrehajtási idejét, így meg tudjuk nézni, hogyan változik a futási idő az SPI frekvencia függvényében
- Az illesztett modell szerint a frekvenciafüggő tag mellet egy állandó többletidő is jelentkezik, melynek hatása nagyobb frekvenciákon igen jelentős



ST7735_demo2.ino (részlet)

```
#include <Adafruit_GFX.h>
#include <Adafruit_ST7735.h>
#include <SPI.h>

#define MOSI_PIN 19
#define MISO_PIN 16
#define SCLK_PIN 18
#define TFT_CS   17
#define TFT_RST   1
#define TFT_DC    0
Adafruit_ST7735 tft = Adafruit_ST7735(&SPI,TFT_CS,TFT_DC,TFT_RST);

void setup(void) {
  gpio_set_function(SCLK_PIN, GPIO_FUNC_SPI);
  gpio_set_function(MOSI_PIN, GPIO_FUNC_SPI);
  gpio_set_function(MISO_PIN, GPIO_FUNC_SPI);
  SPI.begin();
  tft.initR(INITR_BLACKTAB);   // Init ST7735S chip, black tab
  tft.setSPISpeed(10000000);   // You may safely use 10 - 15 MHz
  tft.setRotation(2);          // Orientation: portrait, upside down
  . . .
}
```

Ebben a változatban megmutatjuk, hogyan lehet megadni azt, hogy mely SPI kivezetéseket kívánjuk használni

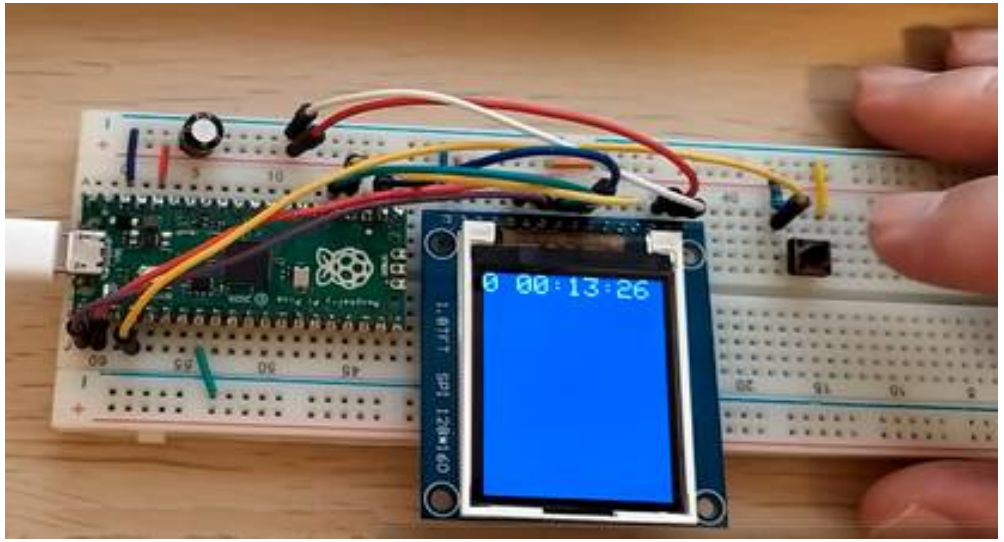
Itt adhatjuk meg SPI alternatív kivezetéseit

Ebben a változatban hivatkoznunk kell az általunk (át)konfigurált SPI objektumra

SPI módba kell állítanunk a kiszemelt GPIO lábakat és inicializáljuk az SPI eszközt

ST7735_on-off.ino

- Ez a **displayOnOffTest.ino** adaptációja, ami bemutatja, hogyan lehet futásidőben a kijelzőt ki- és bekapcsolni a képernyő beégésének elkerülésére
- A program azt is szemlélteti, hogyan lehet egy korábbi értéket törölni úgy, hogy azt a képernyő háttérszínével rajzoljuk újra, mielőtt az új értéket ugyanarra a helyre írnánk. Ezzel elkerülhető a **fillScreen()** hívása, amely az egész képernyő törlését és annak teljes újrarajzolását igényelné
- A program egy GPIO kivezetést nyomógomb bemenetként használ. A nyomógomb lenyomásai programmegszakítást keltenek, amelyek a kijelző ki- és bekapcsolását vezérlik futásidőben
- Az SPI adatátvitelre az alapértelmezett és előre konfigurált SPI csatornát használjuk

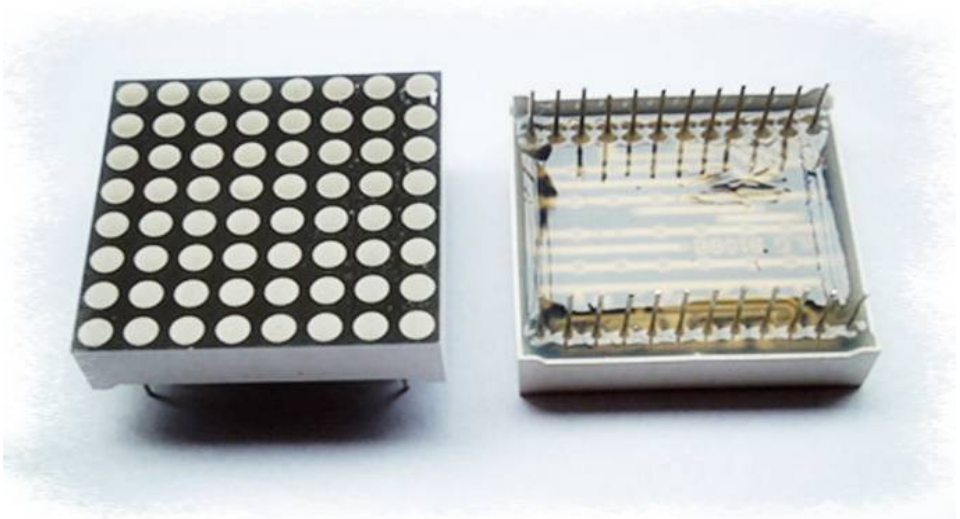


ST7735_on-off.ino (az inicializálás részletei)

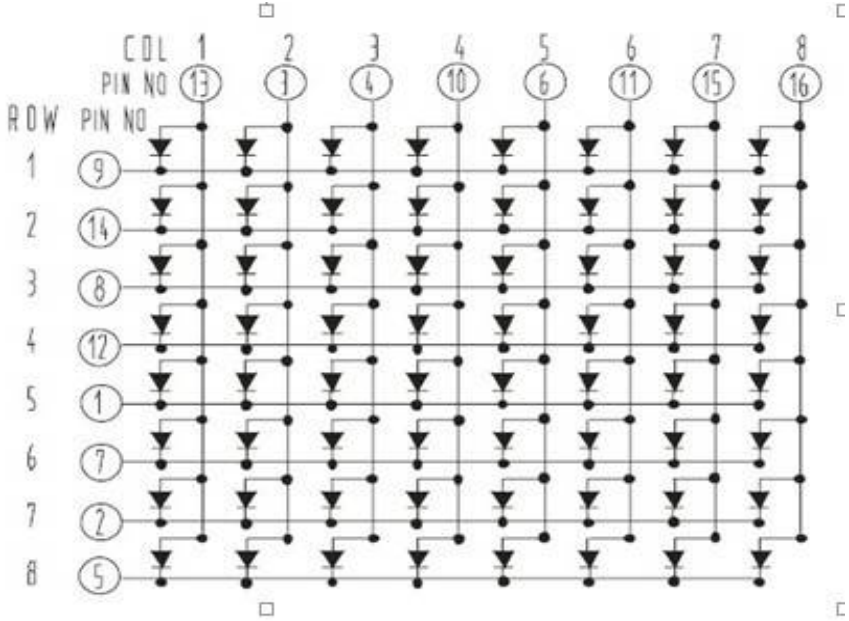
```
#include <Adafruit_GFX.h>    // Core graphics library
#include <Adafruit_ST7735.h> // Hardware-specific library for ST7735
#include <SPI.h>
#define TFT_CS        17
#define TFT_RST        1
#define TFT_DC         0
Adafruit_ST7735 tft = Adafruit_ST7735(TFT_CS, TFT_DC, TFT_RST);
const uint8_t   Button_pin              = 2; // GPIO2

void setup() {
  pinMode(Button_pin, INPUT_PULLUP);
  attachInterrupt(digitalPinToInterrupt(Button_pin), senseButtonPressed, FALLING);
  tft.initR(INITR_BLACKTAB);      // Init ST7735S chip, black tab
  tft.setSPISpeed(1000000);
  tft.setRotation(2);             // Orientation: portrait, upside down
  tft.setFont();
  tft.fillScreen(Display_Backround_Color);
  tft.setTextColor(Display_Text_Color);
  tft.setTextSize(2);
  isDisplayVisible = true;   isButtonPressed = false;
}
```

A program többi részét változatlanul hagytuk...

8x8 LED mátrix



3 mm-es piros LED-ek 8x8-as mátrixba szervezve
A **1088AS** típusnál a sorkiválasztó vonalak a katódokat közösítik



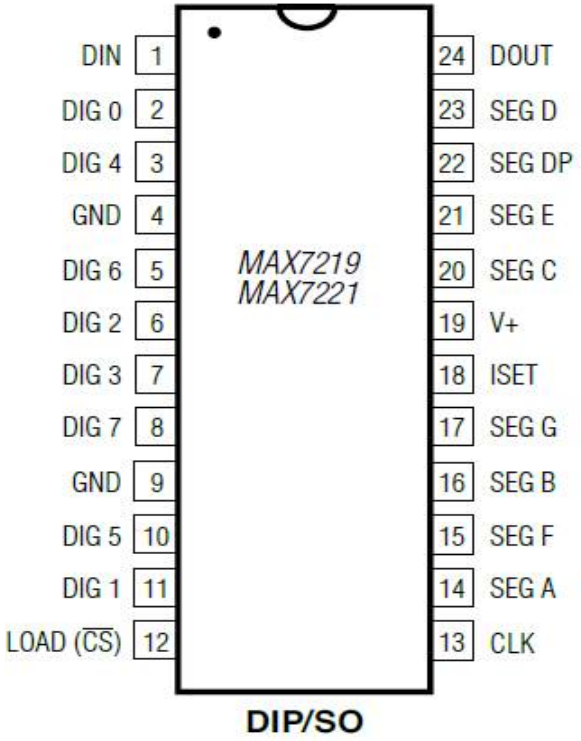
Multiplex kijelzés, egyidejűleg legfeljebb egy sor, vagy egy oszlop lehet aktív.

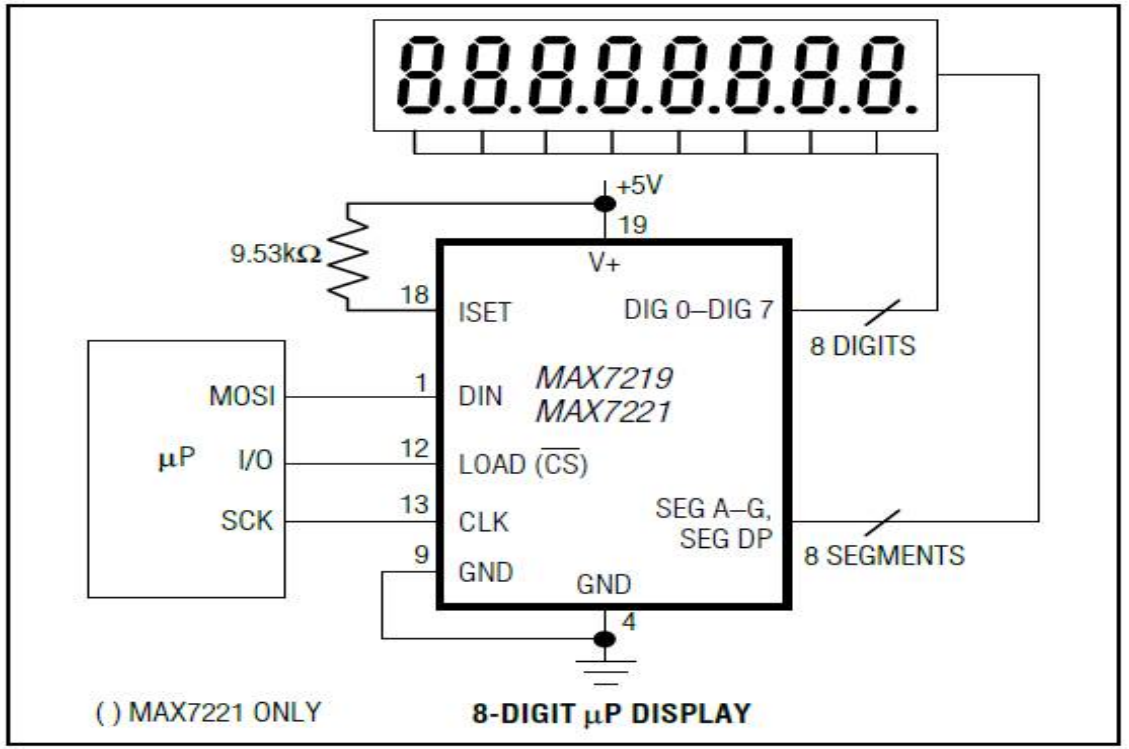
Kényelmes meghajtás:

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

MAX7219

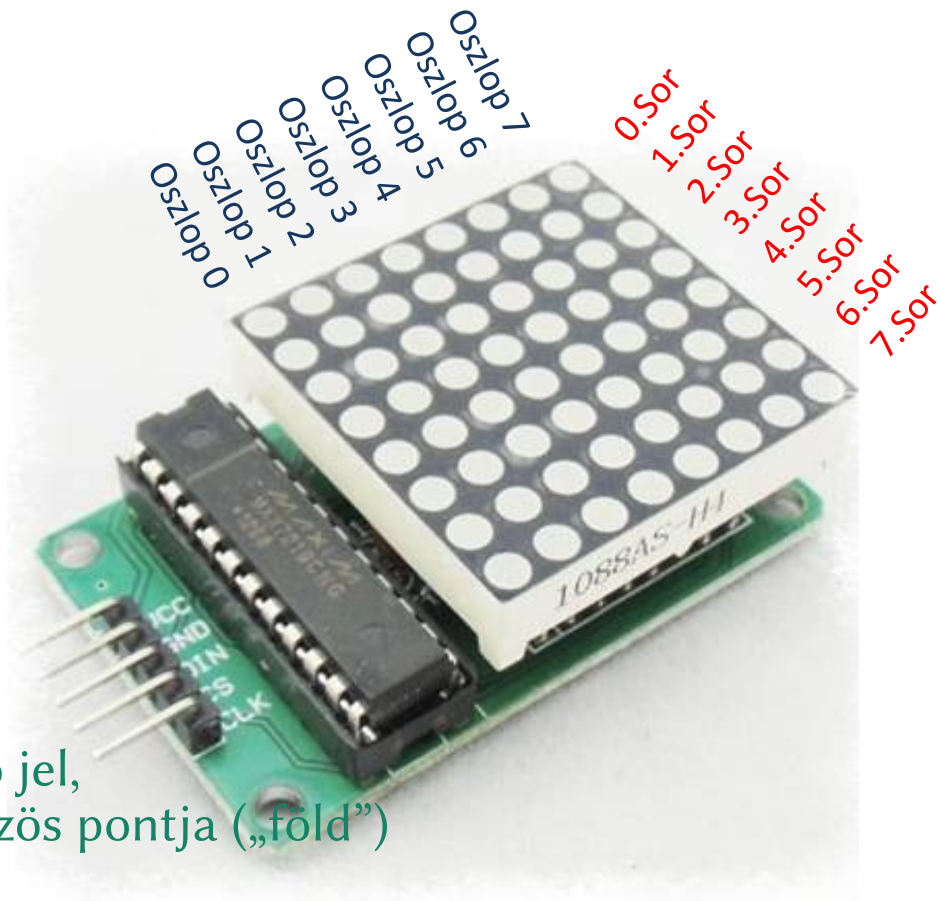
LED meghajtó IC, beépített áramkorlátozással. 7 szegmens kijelző esetén 1-8 db számjegy meghajtása (opcionálisan beépített dekódolással), vagy 8x8 LED mártix meghajtása. Az IC vezérlése SPI buszon történhet.





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

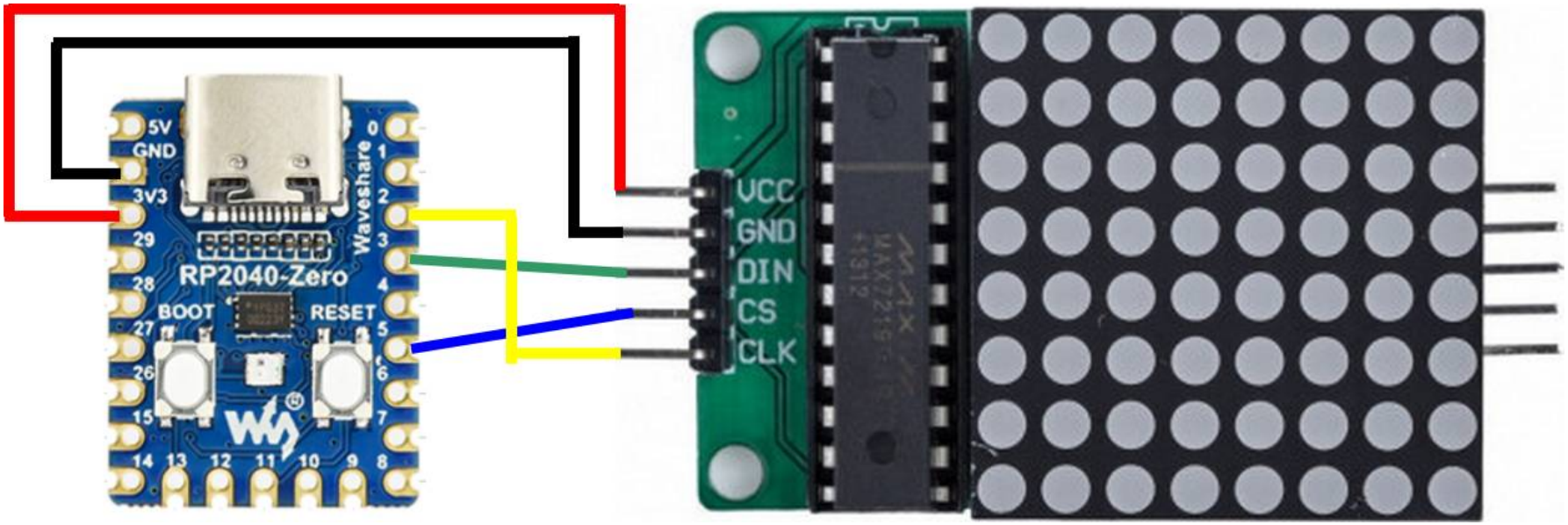


Bemenetek	**Kimenetek**
1 VCC	1 VCC
2 GND	2 GND
3 DIN	3 DOUT
4 CS	4 CS
5 CLK	5 CLK

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

Bekötési rajz

- Az **RP2040** mikrovezérlő **SPI** csatornáját használjuk, de nem az alapértelmezett kivezetésekkel
- A tápfeszültség hivatalosan 4 – 5 V, de a tapasztalat szerint 3,3 V-tal is működőképes a LED mátrix



Technikai részletek az adatlapból

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

SPI kezelés Mbed módra

- A **MAX7219_demo.ino** program az RP2040 mikrovezérlővel és **MAX7219** IC-vel egy 8x8-as LED mátrixot vezérel, az mbed::SPI objektumosztály használatával
- A LED mátrixon felváltva jeleníti meg a **H** és **W** betűket: a "H" betűt a led1[], míg a "W" betűt a led2[] tömb tartalmazza

A program lépései:

- **SPI inicializálása:** A **setup()** függvény beállítja az SPI kommunikációt (1 MHz sebesség, 8-bites formátum) és a chip select (CS) lábat
- **MAX7219 inicializálása:** Az **Init_MAX7219()** gondoskodik a kijelző alapvető beállításáról (például dekódolás kikapcsolása, fényerő szabályozása, tesztmód aktiválása)
- **Adatok küldése:** A **SPI_Write2()** küldi a mátrix sorainak vezérléséhez szükséges adatokat.
- **Karakterek megjelenítése:** A **loop()** függvény időzítve írja ki a két betűt a LED mátrixra, 1 másodperces késleltetéssel váltogatva a megjelenítést.

MAX7219_demo.ino – 2/1.

```
#include <mbed.h>
#define MOSI_PIN p3
#define MISO_PIN p4
#define SCLK_PIN p2
#define   CS_PIN p5
const unsigned char led1[] = {0xFF, 0x18, 0x18, 0x18, 0x18, 0x18, 0x18, 0xFF}; // H
const unsigned char led2[] = {0x1F, 0x60, 0x80, 0x40, 0x40, 0x80, 0x60, 0x1F}; // W
mbed::SPI spi(MOSI_PIN, MISO_PIN, SCLK_PIN); // SPI inicializálás
mbed::DigitalOut cs(CS_PIN);    // Chip Select pin

void setup() {
  cs = 1;                       // CS initially High
  spi.format(8, 0);             // 8-bit format, 0 mode
  spi.frequency(1000000);       // 1 MHz SCLK frequency
  Init_MAX7219();               // MAX7219 initialization
}

void loop() {
  for (int i = 1; i < 9; i++) { SPI_Write2(i, led1[i - 1]); } // Display letter H
  delay(1000);                  // 1 másodperc késleltetés
  for (int i = 1; i < 9; i++) { SPI_Write2(i, led2[i - 1]); } // Display letter W
  delay(1000);                  // 1 másodperc késleltetés
}
```

Nem az alapértelmezett SPI kivezetéseket használjuk

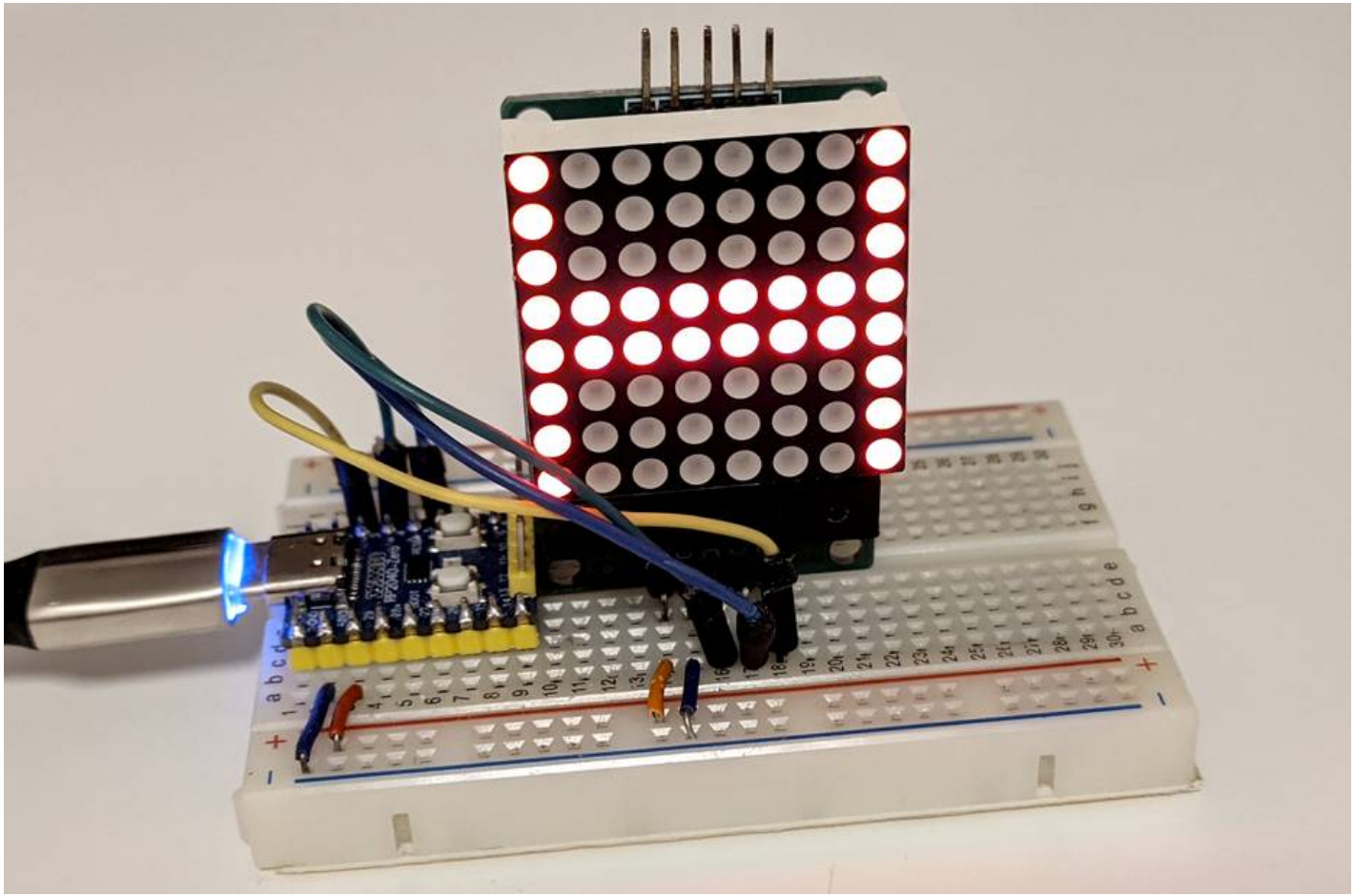
MAX7219_demo.ino – 2/2.

```
//--- Sending commands to SPI ----------
void SPI_Write2(unsigned char MSB, unsigned char LSB) {
  cs = 0;                           // CS Low
  spi.write(MSB);                   // Sending first byte
  spi.write(LSB);                   // Sending second byte
  cs = 1;                           // CS High
}

//--- MAX7219 initialization ------------
void Init_MAX7219() {
  SPI_Write2(0x09, 0x00);           // Decoding off
  SPI_Write2(0x0A, 0x08);           // Normal brightness
  SPI_Write2(0x0B, 0x07);           // Scan limit = 7
  SPI_Write2(0x0C, 0x01);           // Normal mode
  SPI_Write2(0x0F, 0x0F);           // Enable test mode
  delay(500);                       // Wait for 500 ms
  for (int i=1; i<=8; i++) {
    SPI_Write2(i, 0x00);            // Delete rows
  }
  SPI_Write2(0x0F, 0x00);           // Test mode off
  delay(500);                       // Wait for 500 ms
}
```

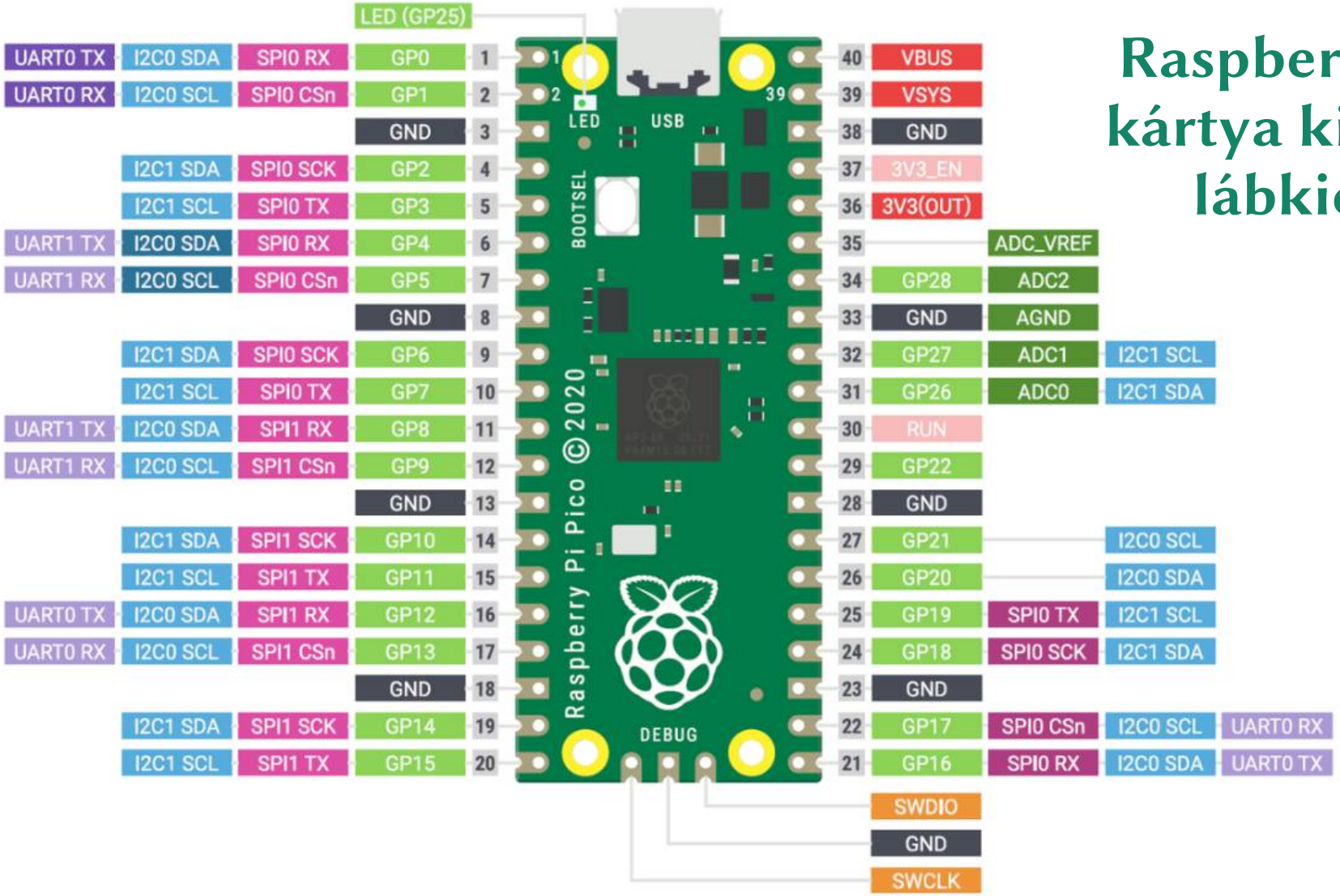
A vezérlő parancsok mindig két bájtból állnak, ezért a küldés mindig 16 bites...

A kijelzővezérlő inicializálása az adatlap szerint

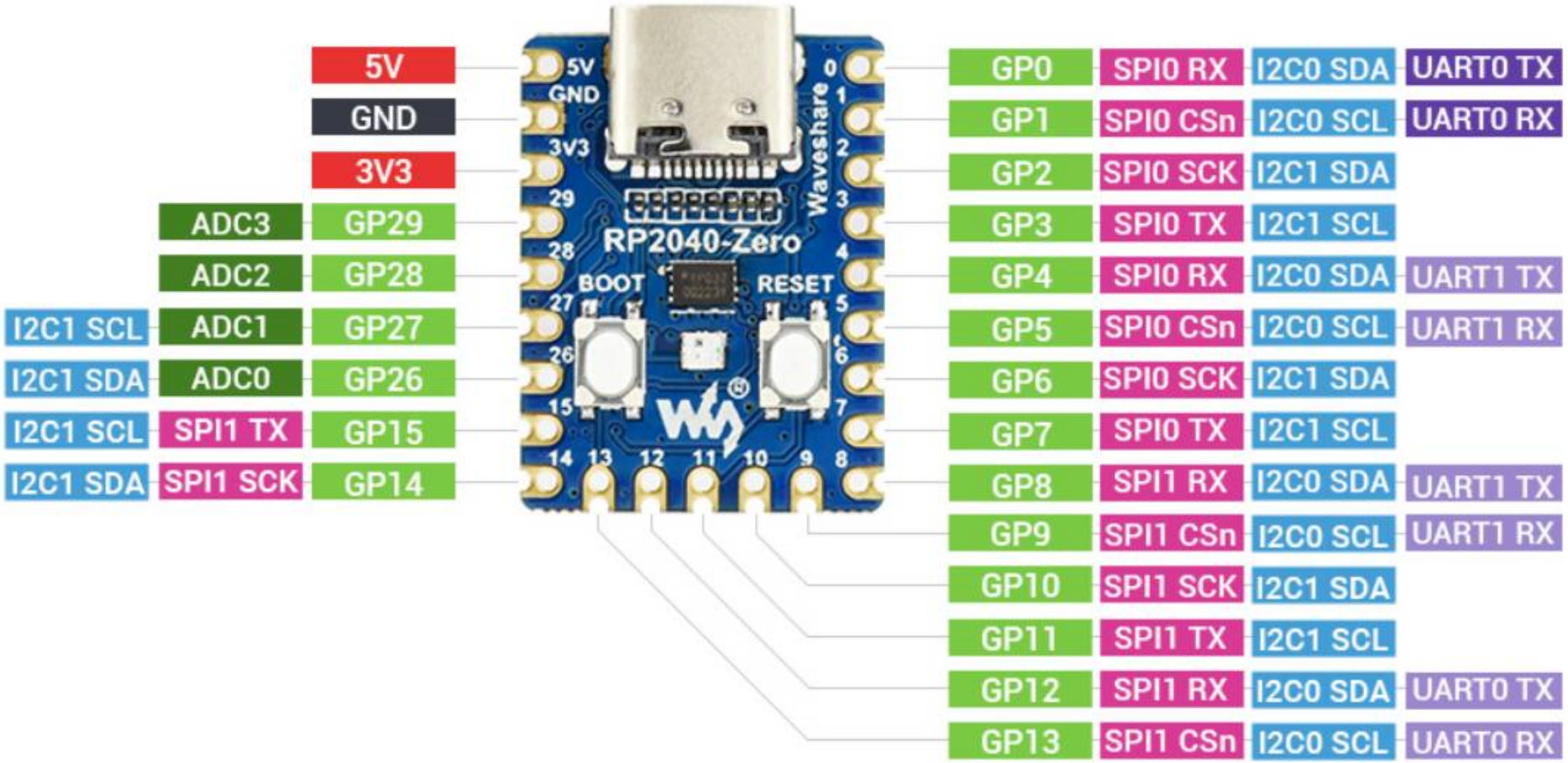
MAX7219_demo.ino – futási eredmény



Raspberry Pi Pico kártya kivezetések lábkiosztása



A kivezetések lábkiosztása



További kivezetések

- Néhány kivezetés a kártya hátoldalán van kivezetve