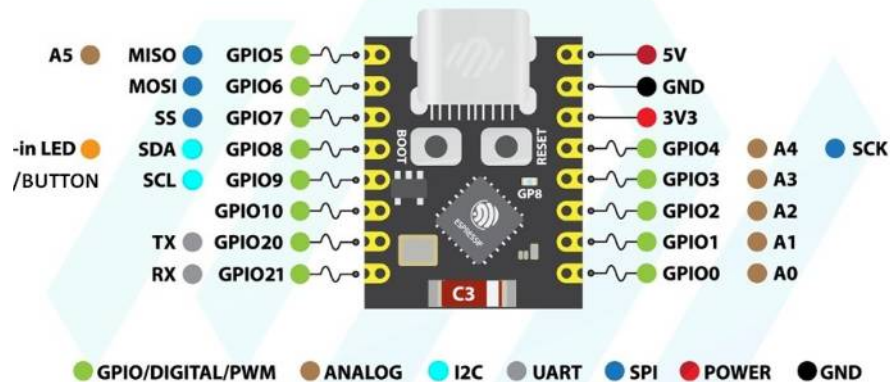
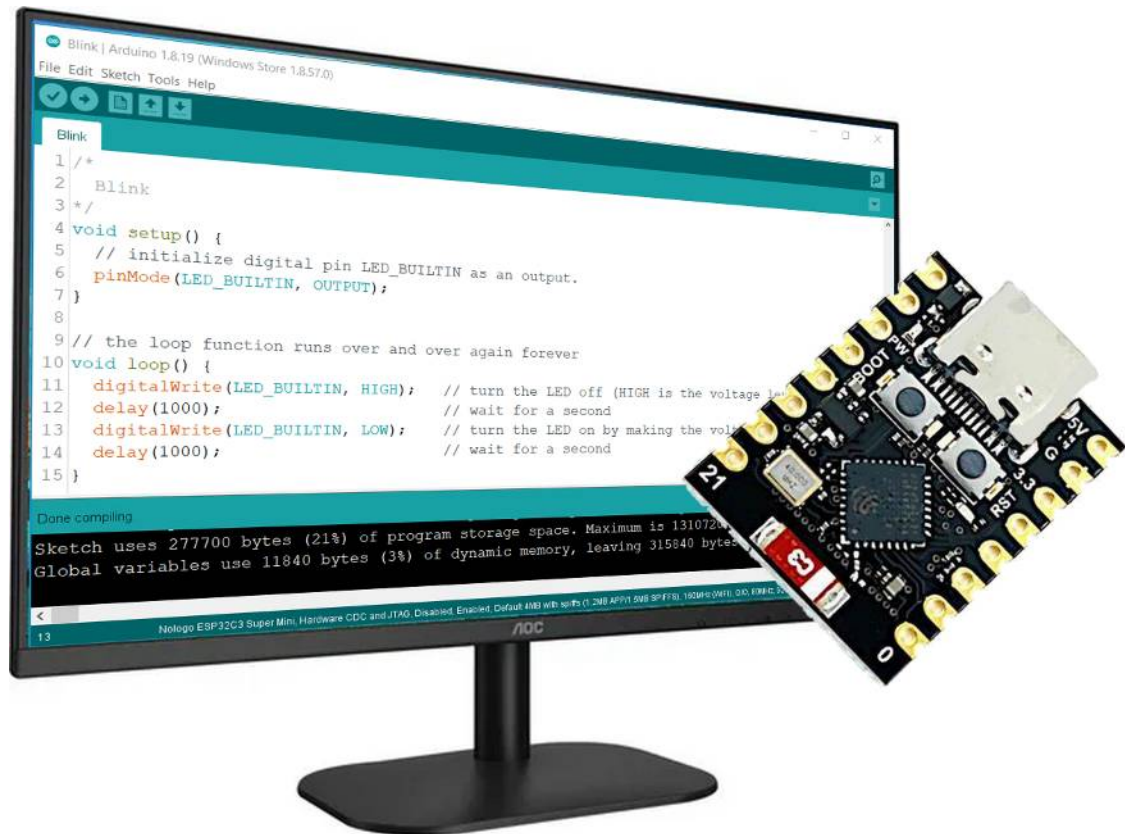
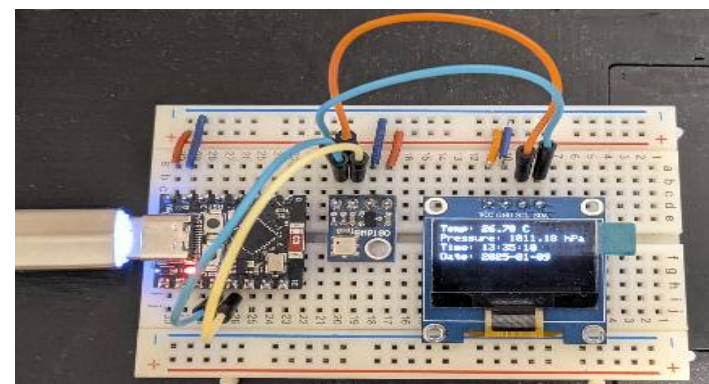


ESP32-C3 mikrovezérlők programozása Arduino IDE környezetben



ESP32 C3 Super Mini



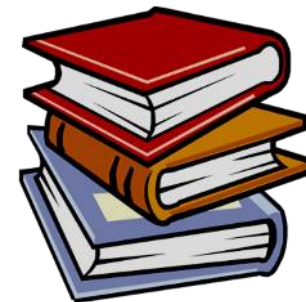
Felhasznált és ajánlott irodalom

❖ Arduino:

- Arduino IDE letöltés: arduino.cc/en/software

❖ Espressif:

- [Arduino core for the ESP32](#)
- [ESP32 Arduino core documentation](#)
- [ESP32-C3 Datasheet](#)
- [ESP32-C3 Technical Reference Manual](#)



❖ Online tutorials:

- Sidharth Mohan Nair: [ESP32-C3 Super Mini Tutorial](#)
- TIAGOTECH: [ESP32-C3 Super Mini: Arduino IDE Quick Start Guide](#)
- LinuxHaxor: [Unleashing ESP32-C3 with Arduino: A Definitive Guide](#)
- Michiel van der Wulp: [ESP32-C3 SuperMini and expansion board](#)

Mi az Arduino?

❖ Az **Arduino** egy szabad szoftveres, nyílt forráskódú elektronikai fejlesztőplatform, vagy ökoszisztéma az elektronikus eszközök könnyen elsajátítható kezeléséhez, ami az alábbi összetevőket jelenti:

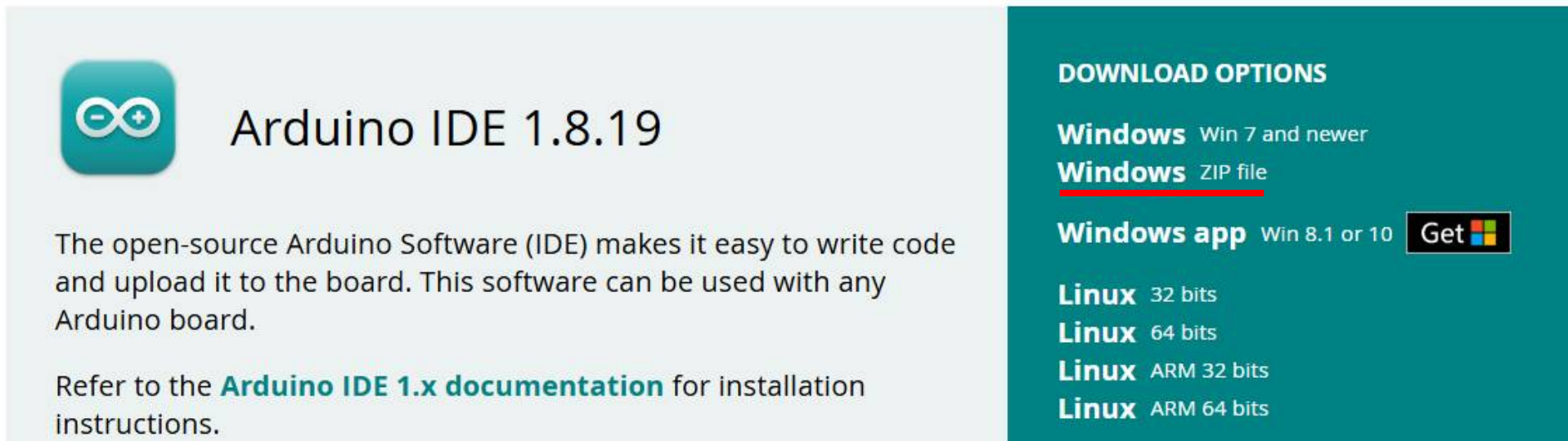
- **Arduino IDE** (integrált fejlesztői környezet): Java alapú, keresztplatformos fejlesztői környezet (szerkesztő, fordító, programletöltő stb.)
- **Arduino kártya**: eredetileg **ATmega** mikrovezérlőn alapuló hardver, amely önállóan vagy a számítógéppel összekapcsolva is működhet, de a támogatott kártyák száma rohamosan bővül, és a keretrendszer bővítőcsomagok telepítésével kiegészíthető
Ebben az előadásban az **ESP32-C3 Supermini** kártyához mutatjuk be példaprogramokat, ehhez az Arduino IDE-t az ESP32 Arduino Core csomaggal bővítjük
- **Arduino programnyelv és programkönyvtár-gyűjtemény**: amely lehetővé teszi, hogy a mikrovezérlő részleteinek pontos ismerete nélkül, egyszerűen írassunk programot




```
void setup() {  
  pinMode(LED_BUILTIN, OUTPUT);  
}  
  
void loop() {  
  digitalWrite(LED_BUILTIN, HIGH);  
  delay(1000);  
  digitalWrite(LED_BUILTIN, LOW);  
  delay(1000);  
}
```

Az Arduino IDE telepítése

- ❖ **Windows 10** esetén a **Microsoft Store**-ban is rendelkezésre áll az **Arduino IDE**. Más esetben az arduino.cc/en/software oldalról töltsük le a valamelyik **Arduino** kiadást –
- ❖ A bemutatott programokhoz az **Arduino IDE Legacy 1.8.19** verzióját használtuk (általában a ZIP verziót szoktam telepíteni)




 **Arduino IDE 1.8.19**

The open-source Arduino Software (IDE) makes it easy to write code and upload it to the board. This software can be used with any Arduino board.

Refer to the [Arduino IDE 1.x documentation](#) for installation instructions.

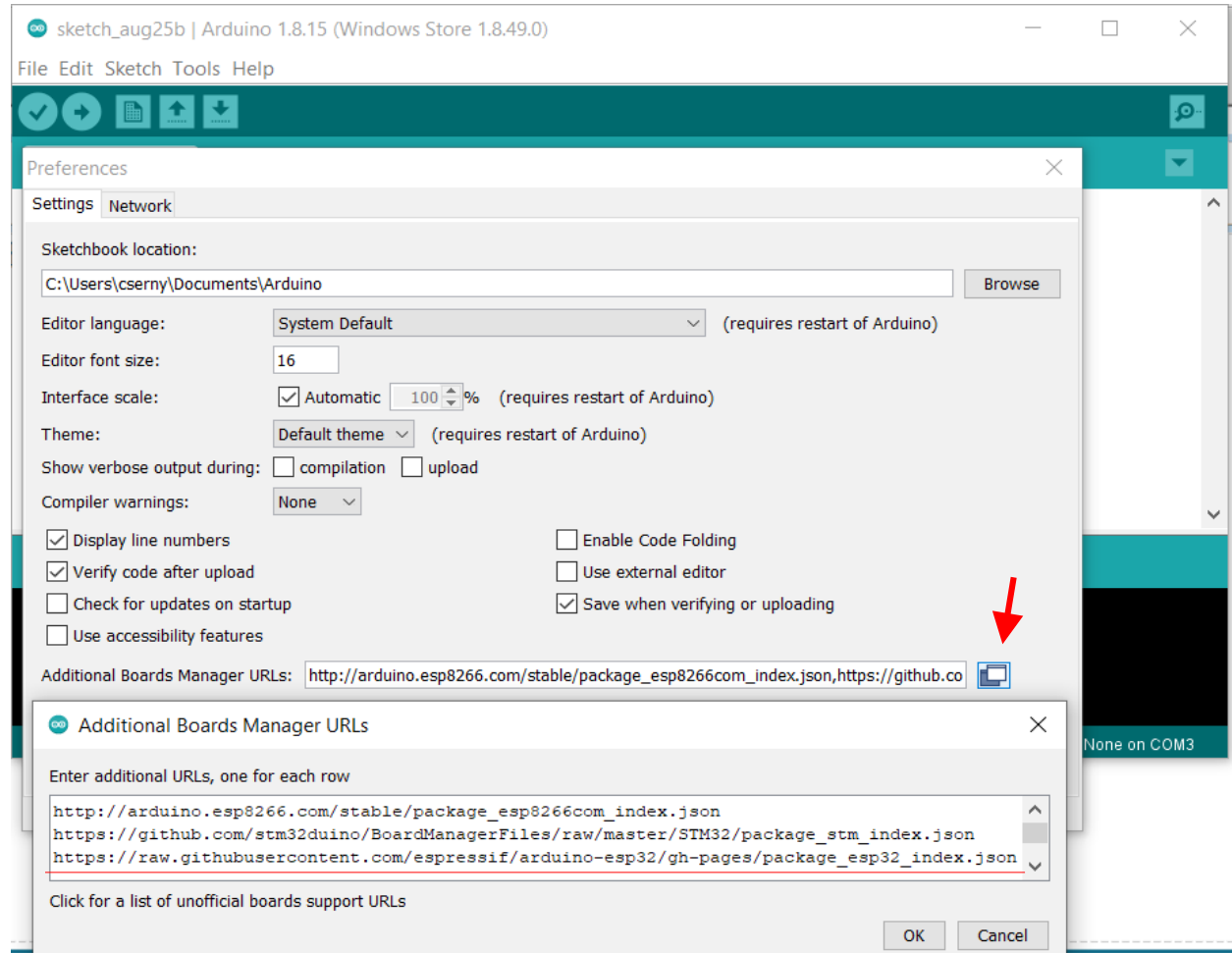
DOWNLOAD OPTIONS

- Windows** Win 7 and newer
- Windows** ZIP file
- Windows app** Win 8.1 or 10 
- Linux** 32 bits
- Linux** 64 bits
- Linux** ARM 32 bits
- Linux** ARM 64 bits

- ❖ A letöltés és telepítés után előfordulhat, hogy a kártyához való meghajtó programot is telepíteni kell, a soros illesztő típusától függően

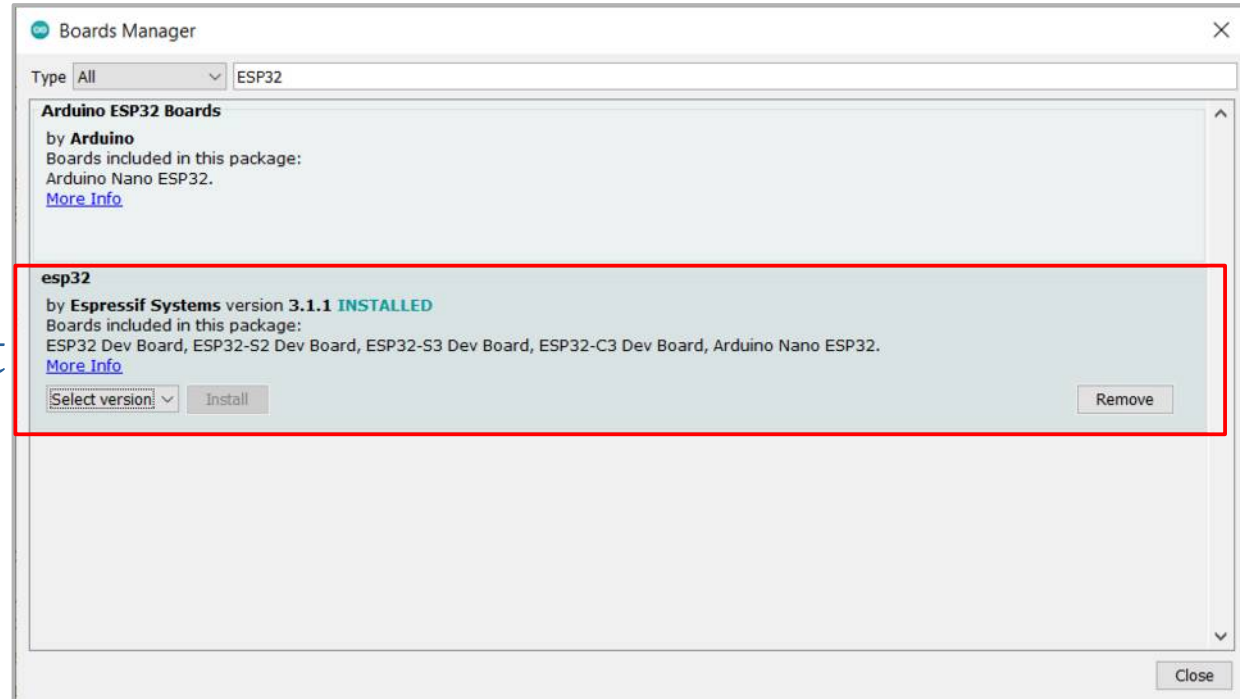
Az ESP32 hardver támogatás telepítése

- ❖ Az Arduino **File/Preferences** menüpontjára kattintunk
- ❖ A felbukkanó lapon az **Additional Boards Manager** URLs rovatba másoljuk be (vagy a mellette levő ikonra kattintva szerkesszük bele a listába) az alábbi sort:
https://raw.githubusercontent.com/espressif/arduino-esp32/gh-pages/package_esp32_index.json
- ❖ Ezután a **Tools** menü **Boards Manager** pontjában a felbukkanó listában választható és telepíthető az **esp32** kártyát támogató programcsomag



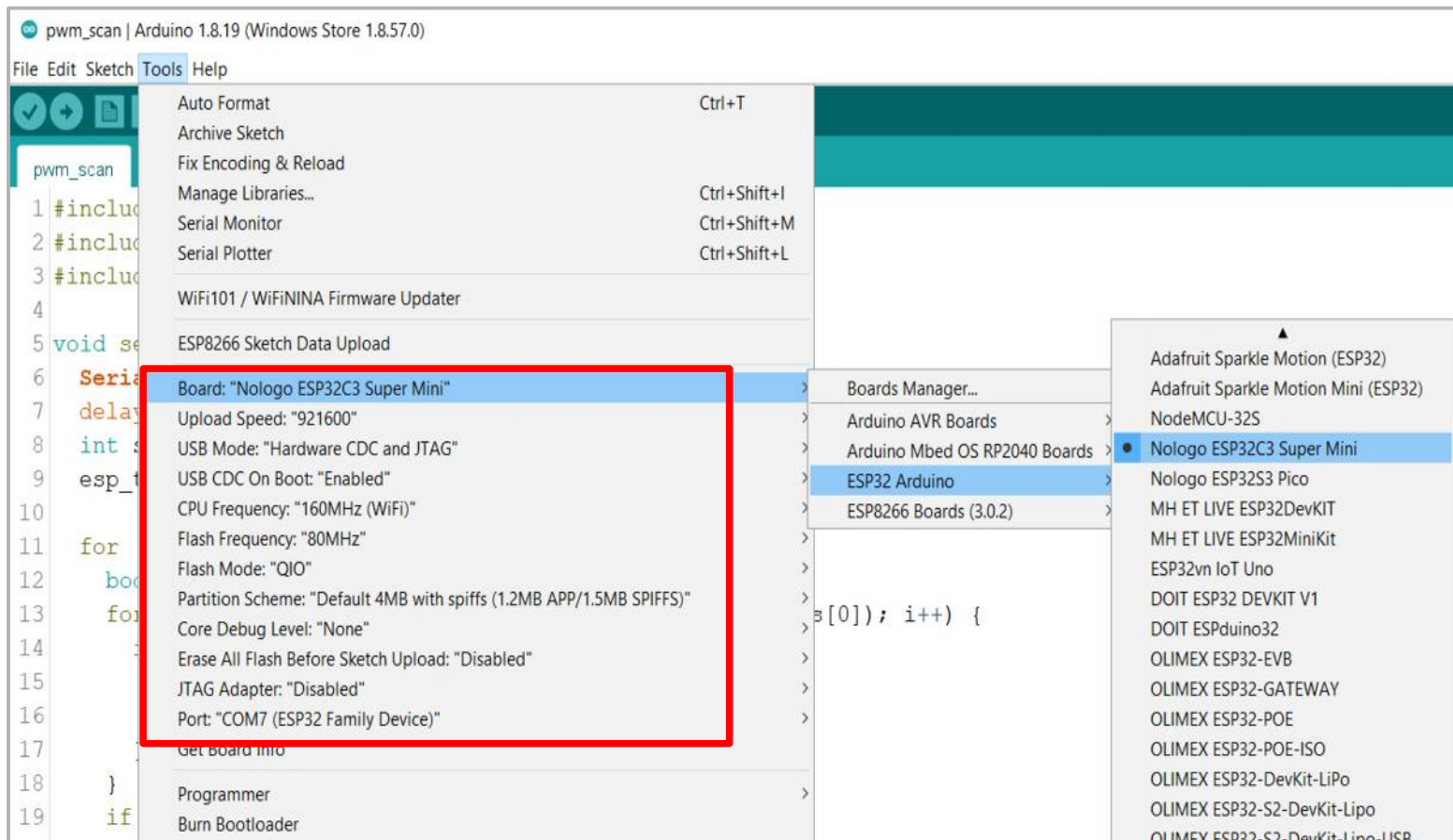
Az ESP32 hardver támogatás telepítése

- ❖ A **Tools** menü **Boards Manager** pontjában a felbukkanó listában keressük meg az **ESP32** kártyát támogató csomagot (a **by Espressif System** változatot) és kattintsunk az **INSTALL** gombra!
- ❖ E sorok írásakor a 3.1.1 verzió a legfrissebb, ami néhány vonatkozásban különbözik az általunk 3 évvel korábban használt 2.x verziótól



Az Arduino IDE beállítása

- ❖ A **Tools** menüben az **ESP32** kártyák közül válasszuk a **Nologo ESP32-C3 Supermini** kártyát és konfiguráljuk az ábra szerint!
- ❖ Csatlakoztatás után válasszuk ki a kártyához csatlakozó soros portot! (pl. **COM7**)

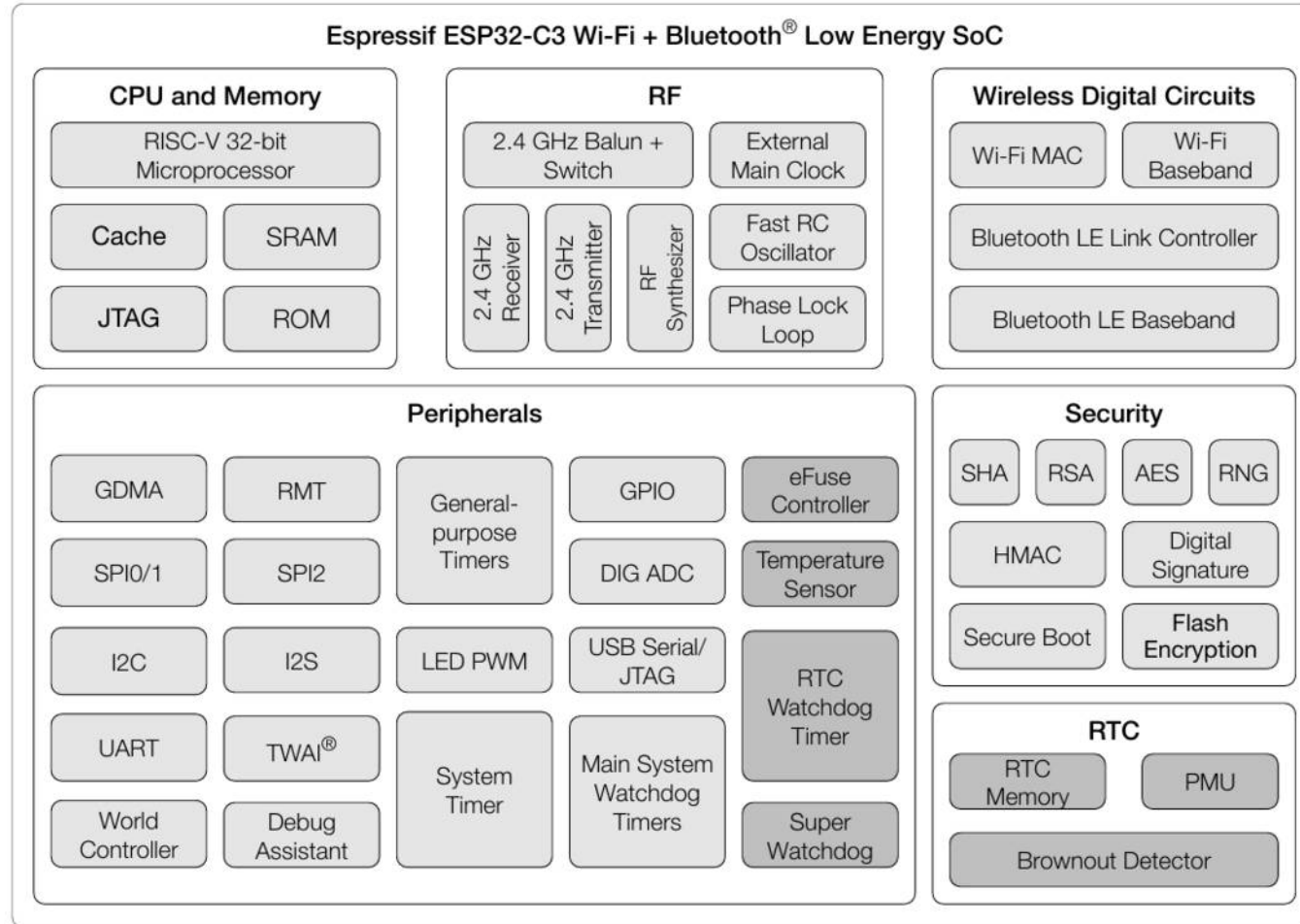


Az ESP32-C3 Super Mini kártya

- ❖ A **Maker Go ESP32-C3 Super Mini** egy kompakt fejlesztői kártya
- ❖ Mikrovezérlő: **ESP32-C3**, 32 bites **RISC-V** architektúra
- ❖ Órajel: Akár 160 MHz
- ❖ Memória: 400 KB SRAM, 384 KB ROM
- ❖ Programtároló: 4 MB Flash memória
(a képen látható **ESP32-C3 FH4x** feliratú MCU már az IC tokban tartalmazza a 4 MB flash memóriát)
- ❖ Wi-Fi: 802.11 b/g/n (2.4 GHz)
- ❖ Bluetooth: Bluetooth 5.0 LE
- ❖ GPIO: 16 általános célú bemenet/kimenet (GPIO), melyek közül 13 van kivezelve
- ❖ Interfészek: SPI, I2C, UART, ADC, PWM



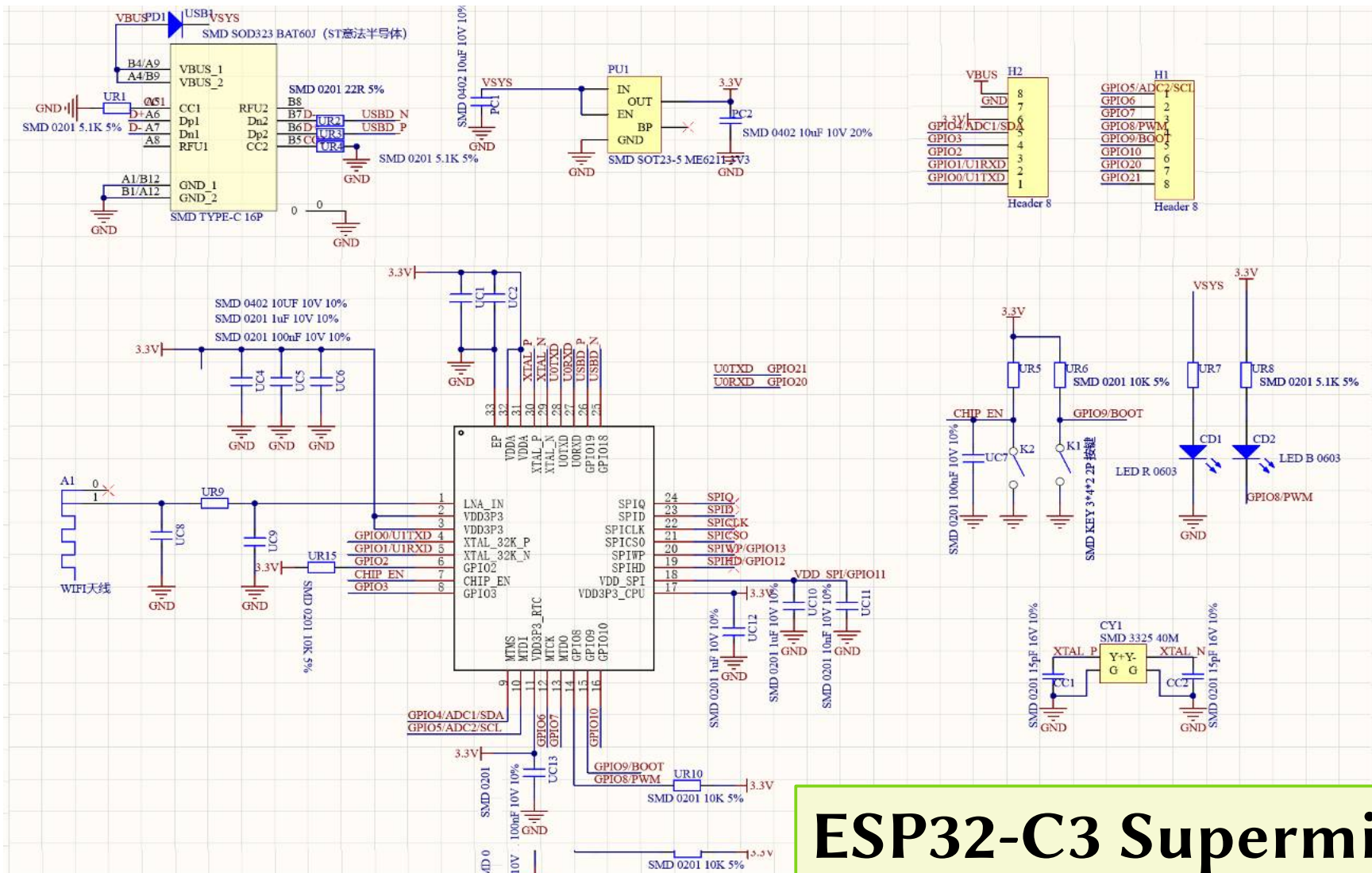
ESP32-C3 funkcionális blokkdiagram



Támogatott perifériák

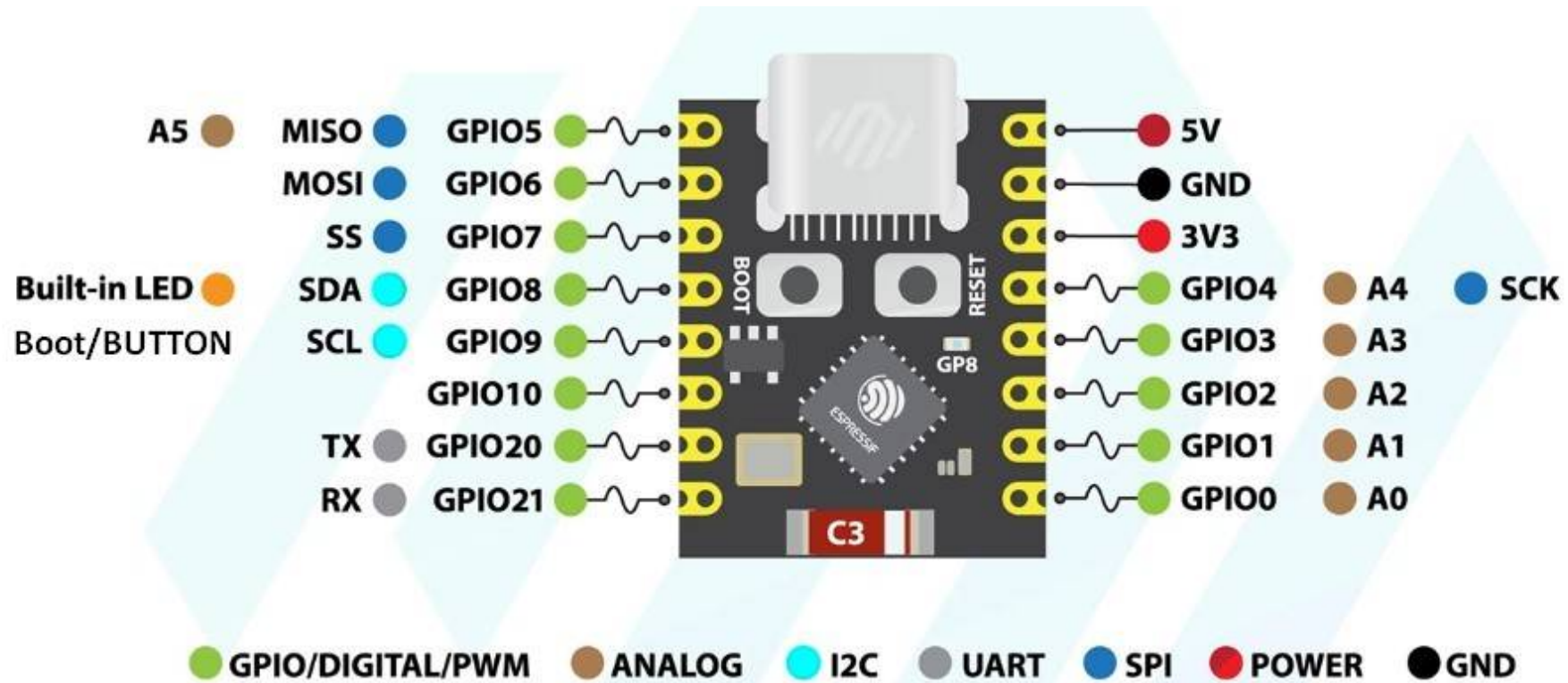
- ❖ Az ESP32 Arduino Core 3.1.1 az alábbi perifériákat támogatja
- ❖ Az **ESP WROOM-32**-höz képest eltérés a CPU, nincs hagyományos Bluetooth, Hall szenzor, DAC, SDMMC és érintésérzékelés periféria
- ❖ Az **ESP32-C3** előnye a kis fogyasztás és a beépített USB CDC/JTAG periféria

Peripheral	ESP32	ESP32-S2	ESP32-C3	ESP32-S3
ADC	Yes	Yes	Yes 12 bit	Yes
Bluetooth	Yes	Not Supported	Not Supported	Not Supported
BLE	Yes	Not Supported	Yes	Yes
DAC	Yes	Yes	Not Supported	Not Supported
Ethernet	Yes	Not Supported	Not Supported	Not Supported
GPIO	Yes	Yes	Yes	Yes
Hall Sensor	Yes	Not Supported	Not Supported	Not Supported
I2C	Yes	Yes	Yes	Yes
I2S	Yes	Yes	Yes	Yes
LEDC	Yes	Yes	Yes	Yes
Motor PWM	No	Not Supported	Not Supported	Not Supported
Pulse Counter	No	No	No	No
RMT	Yes	Yes	Yes	Yes
SDIO	No	No	No	No
SDMMC	Yes	Not Supported	Not Supported	Yes
Timer	Yes	Yes	Yes	Yes
Temp. Sensor	Not Supported	Yes	Yes	Yes
Touch	Yes	Yes	Not Supported	Yes
TWAI	No	No	No	No
UART	Yes	Yes	Yes	Yes
USB	Not Supported	Yes	Yes Only CDC/JTAG	Yes
Wi-Fi	Yes	Yes	Yes	Yes



ESP32-C3 Supermini

Az ESP32 C3 Super Mini kártya kivezetései



ESP32 C3 Super Mini

Megjegyzés: Az A5 analóg bemenet (ADC2) nem használható, ha a WiFi használatban van!

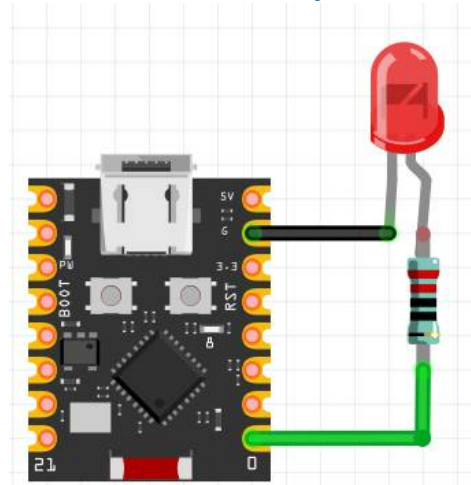
Digitális I/O

- ❖ **pinMode(*pin*, *mode*)** - beállítja a megnevezett kivezetés üzemmódját, ahol:
pin – a kiválasztott GPIO kivezetés sorszáma (0 – 39, de nem mindegyik elérhető)
mode – üzemmód: **INPUT**, **INPUT_PULLUP**, **INPUT_PULLDOWN**, vagy **OUTPUT**
az **INPUT_PULLUP** belső felhúzást, az **INPUT_PULLDOWN** belső lehúzást jelent
Ezt a függvényt többnyire a program **setup()** szekciójában használjuk, a kezdeti beállításoknál
- ❖ **digitalRead(*pin*)** – beolvassa a megadott sorszámú kivezetésen a pillanatnyi jelszintet
pin – a kiválasztott GPIO kivezetés sorszáma, a visszatérési érték pedig a pillanatnyi jelszint, ami 0 (alacsony), vagy 1 (magas) értékű lehet
- ❖ **digitalWrite(*pin*, *level*)** – beállítja a korábban kimenetnek állított kivezetésen a jelszintet
pin – a kiválasztott GPIO kivezetés sorszáma
level – a kimeneti szint, ami **LOW** (= 0, alacsony), vagy **HIGH** (=1, magas) értékű lehet

ESP32_ledblink.ino – egyszerű LED villogtatás

- ❖ Villogtassuk a **GPIO0** kimenetre kötött LED-et! A LED áramát egy soros ellenállással (pl. 220 Ω) korlátozhatjuk
- ❖ Ha a LED katódját a GND-re kötjük, az anódját pedig az áramkorlátozó ellenálláson keresztül a **GPIO0** kivezetésre, akkor a kimenet magas szintje gyújtja ki a LED-et
- ❖ A késleltetéshez a beépített **delay()** függvényt használjuk, a késleltetés idejét milliszekundumokban kell megadni
- ❖ **Megjegyzés: GPIO8, vagy LED_BUILTIN** használatával a beépített LED villog, de a katód vezérlése miatt fordított logikával (lásd: **Blinky.ino**)

```
void setup() {  
  pinMode(0, OUTPUT);    // GPIO0 legyen digitális kimenet  
}  
  
void loop() {  
  digitalWrite(0, HIGH); // GPIO0 aktív magas  
  delay(1000);           // egy másodperc késleltetés  
  digitalWrite(0, LOW);  // GPIO0 aktív alacsony  
  delay(1000);           // egy másodperc késleltetés  
}
```



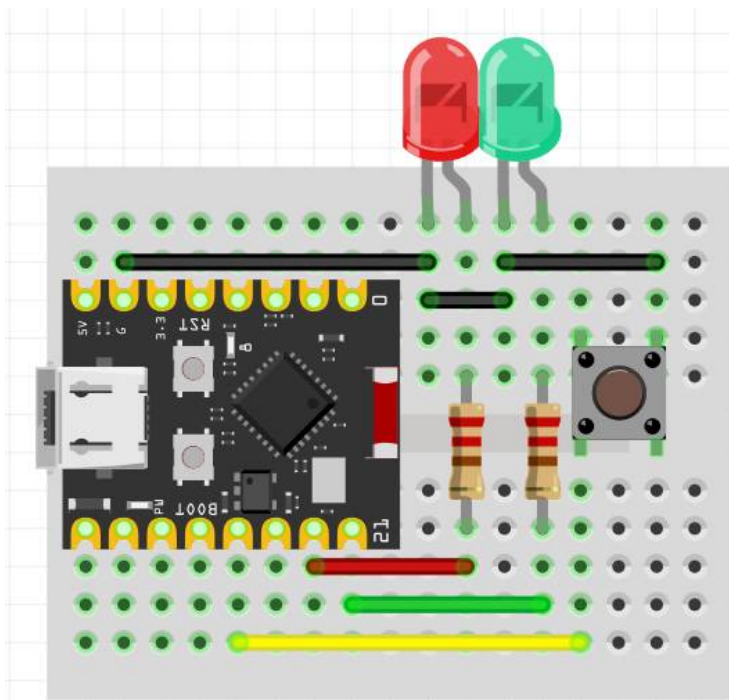
ESP32_button2led.ino – nyomógomb állapotának beolvasása

- ❖ **Feladat:** A két LED a kapcsoló állásától függően világítson:
 - Ha a kapcsoló nyitva van, a piros LED világítson!
 - Ha a kapcsoló zárva van, a zöld LED világítson!
- ❖ A nyomógomb állapotát a **digitalRead()** függvénnyel vizsgáljuk!

```
#define RED_LED    20
#define GREEN_LED 21
#define BUTTON    9

void setup() {
  pinMode(RED_LED,OUTPUT);    // legyen kimenet
  pinMode(GREEN_LED,OUTPUT); // legyen kimenet
  pinMode(BUTTON,INPUT_PULLUP); // Bemenet belső felhúzással
}

void loop()
  int state = digitalRead(BUTTON)
  digitalWrite(RED_LED, state); // világít, ha state = HIGH
  digitalWrite(GREEN_LED,!state); // világít, ha state = LOW
  delay(20); // pergésmentesítő késleltetés
}
```



ADC – Analóg-digitális átalakító

❖ Az **ADC** feladata az, hogy diszkrét kódokká alakítsa a bejövő analóg jelet

❖ A konverzió digitális értéke (N_{ADC}):

■ Végkiterés: $N_{ADC} = 4095$, ha a felbontás 12 bit
bemenő jel $\geq V_{R+} - 1.5 * LSB$

■ Nulla: $N_{ADC} = 0$, ha a bemenő
jel $\leq V_{R-} + 0.5 LSB$

V_{R+} és V_{R-} a referencia-
forrás két sarka

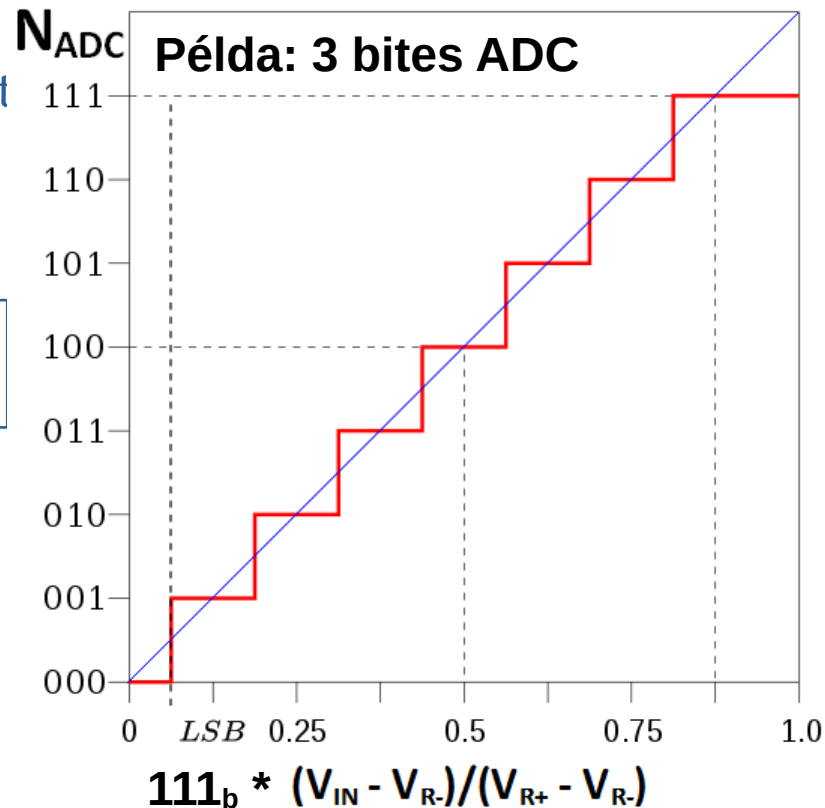
❖ Közbeeső értékekre:

$$N_{ADC} = 4096 * (V_{IN} - V_{R-}) / (V_{R+} - V_{R-})$$

❖ A fenti képletből V_{IN} -t kifejezve ezt kapjuk:

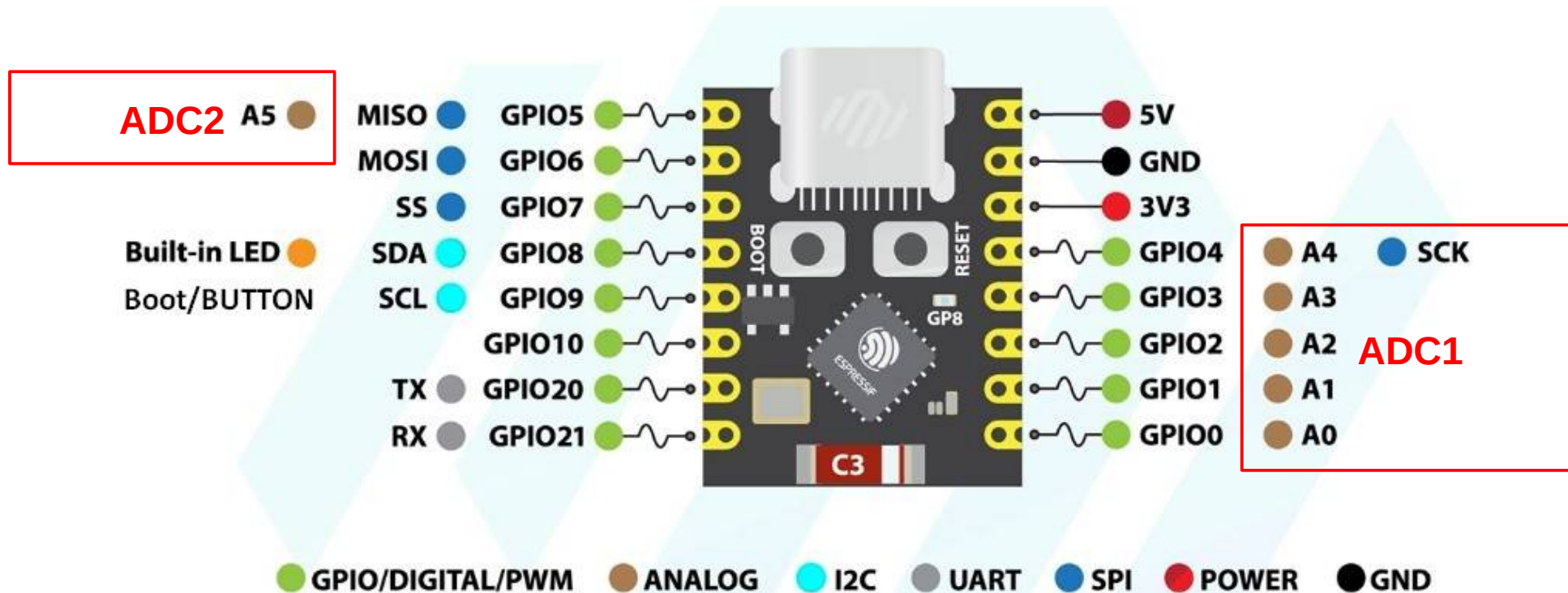
$$V_{IN} = (V_{R+} - V_{R-}) * N_{ADC} / 4096 + V_{R-}$$

❖ V_{R-} általában = 0



Analóg bemenetek

❖ Két ADC van, de ADC2 csak a WiFi letiltott állapotában használható



ESP32 C3 Super Mini

Az analóg-digitális átalakítót kezelő függvények

Az ADC-vel analóg jeleket mérhetünk meg, ami hasznos egy potméterrel leosztott feszültség vagy egy analóg szenzor jele számszerű értékének meghatározására

Az ADC „egylövetű” (OneShot) módjához használható API függvények:

- ❖ **analogRead(*pin*)** – megméri a megnevezett bemeneten a feszültséget és visszaad egy számot (alapértelmezetten 12 bit a felbontás és kb. 2,5 V a méréshatár)
- **analogReadMillivolts(*pin*)** – megméri a megnevezett bemeneten a feszültséget és visszaadja a millivoltokban mért feszültség értékét
- ❖ **analogReadResolution(*resolution*)** – beállítja a felbontást (9 –12 bit, default: 12)
- ❖ **analogSetAttenuation(*attenuation*)** – méréshatár beállítása az összes bemenetre vonatkozóan (ADC_ATTEN_DB_0: 750 mV, ADC_ATTEN_DB_2_5: 1055 mV, ADC_ATTEN_DB_6: 1300 mV, ADC_ATTEN_DB_11: 2500 mV)
- ❖ **analogSetPinAttenuation(*pin*, *attenuation*)** – a méréshatár beállítása egy adott lábra vonatkozóan

ADC_OneShot.ino

- ❖ Az alábbi programocska a OneShot mód használatát mutatja be:
A **GPIO2 (A2)** bemenetre kapcsolt feszültséget mérjük meg

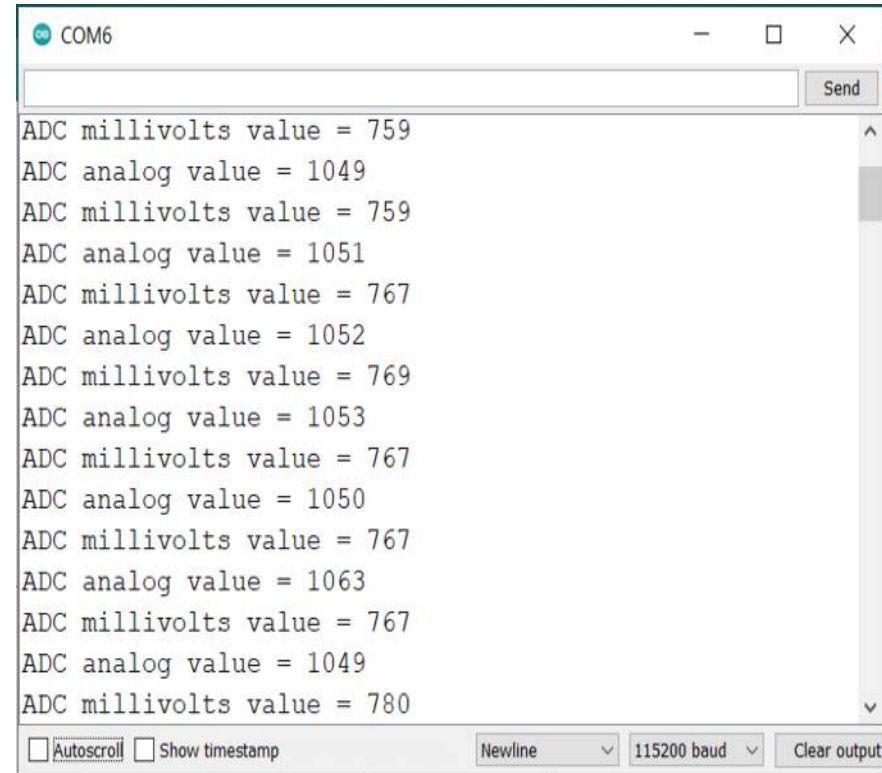
```
const int AN2 = 2 // GPIO2 (A2) bemenet

void setup() {
  Serial.begin(115200);
  analogReadResolution(12); // 12-bit felbontás
  analogSetAttenuation(ADC_11db); // Méréshatár 0-2.5V
}

void loop() {
  // read the analog / millivolts value for pin 2:
  int analogValue = analogRead(AN2);
  int analogVolts = analogReadMilliVolts(AN2);

  // print out the values you read:
  Serial.printf("ADC analog value = %d\n", analogValue);
  Serial.printf("ADC millivolts value = %d\n", analogVolts);

  delay(2000); // delay between reads
}
```



The screenshot shows a serial terminal window titled 'COM6'. The window displays a series of ADC readings, alternating between millivolts and raw analog values. The readings are as follows:

ADC millivolts value	ADC analog value
759	1049
759	1051
767	1052
769	1053
767	1050
767	1063
767	1049
780	

The terminal window also shows control options at the bottom: Autoscroll, Show timestamp, Newline, 115200 baud, and Clear output.

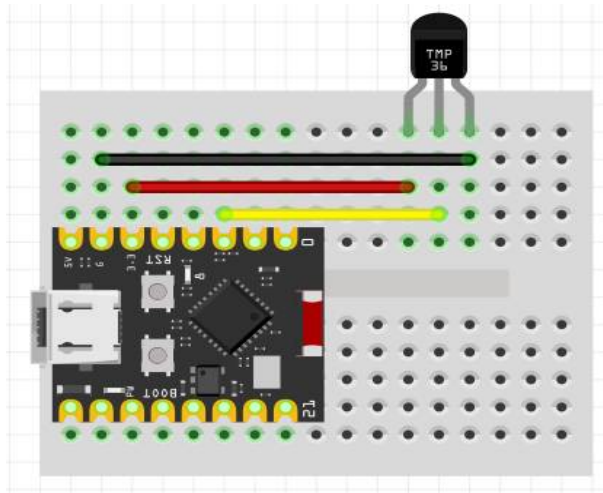
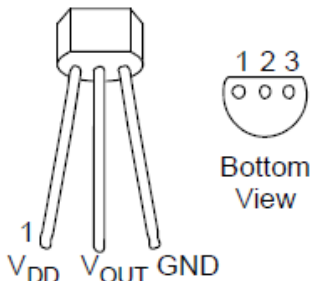
ESP32_MCP9700.ino – analóg hőmérő használata

Microchip MCP9700

- VDD = 2,5 – 5,5 V
- Mérési tart.: -40 – 150 °C
- Érzékenység: 10 mV / °C
- Nullapont: 500 mV @ 0 °C

Kössük a hőmérő kimenetét az ESP32 GPIO2 bemenetére!

3-Pin TO-92
MCP9700/9701
Only

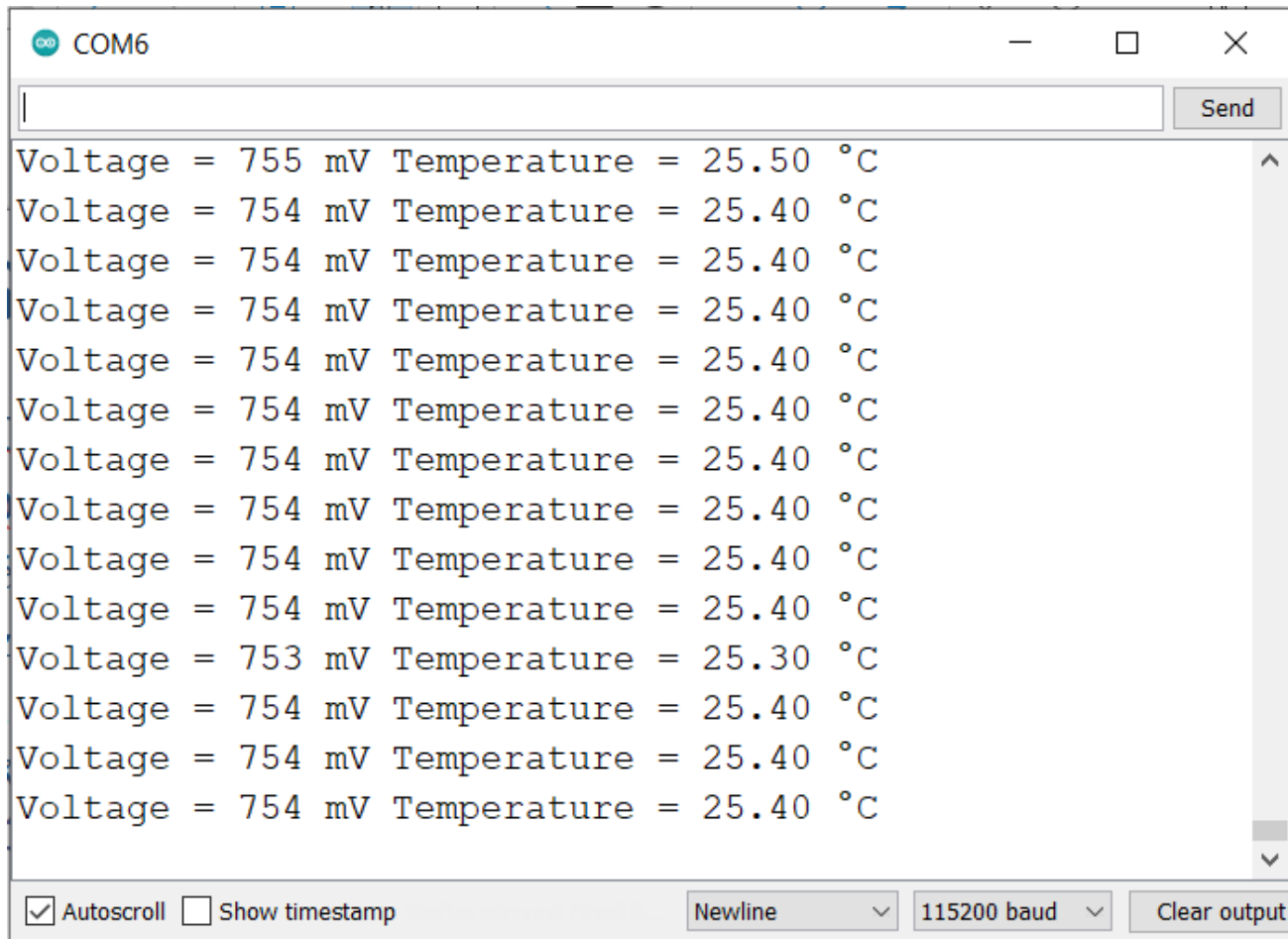


```
const int analog_pin = 2;

void setup() {
  Serial.begin(115200);
  analogReadResolution(12); // 12-bit felbontás
  analogSetAttenuation(ADC_11db); // Méréshatár 0-2,5 V
}

void loop() {
  uint32_t millivolts = 0;
  for(int i=0; i<1024; i++) {
    millivolts += analogReadMilliVolts(AN2);
  }
  millivolts = millivolts>>10;
  float tempC = (millivolts - 500) / 10.0;
  Serial.print("Voltage = ");
  Serial.print(millivolts);
  Serial.print(" mV Temperature = ");
  Serial.print(tempC);
  Serial.println(" °C");
  delay(2000);
}
```

ESP32_MCP9700.ino – futási eredmény



The screenshot shows a serial terminal window titled "COM6". The window contains a list of sensor readings. Each line displays the voltage and temperature. The voltage values are mostly 754 mV, with one instance of 755 mV and one of 753 mV. The temperature values are mostly 25.40 °C, with one instance of 25.50 °C and one of 25.30 °C. The window has a "Send" button at the top right and a "Clear output" button at the bottom right. The bottom status bar includes checkboxes for "Autoscroll" (checked) and "Show timestamp" (unchecked), a "Newline" dropdown menu, a baud rate dropdown menu set to "115200 baud", and a "Clear output" button.

```
COM6
Voltage = 755 mV Temperature = 25.50 °C
Voltage = 754 mV Temperature = 25.40 °C
Voltage = 754 mV Temperature = 25.40 °C
Voltage = 754 mV Temperature = 25.40 °C
Voltage = 754 mV Temperature = 25.40 °C
Voltage = 754 mV Temperature = 25.40 °C
Voltage = 754 mV Temperature = 25.40 °C
Voltage = 754 mV Temperature = 25.40 °C
Voltage = 754 mV Temperature = 25.40 °C
Voltage = 754 mV Temperature = 25.40 °C
Voltage = 753 mV Temperature = 25.30 °C
Voltage = 754 mV Temperature = 25.40 °C
Voltage = 754 mV Temperature = 25.40 °C
Voltage = 754 mV Temperature = 25.40 °C
```

Autoscroll Show timestamp Newline 115200 baud Clear output

wifi_scan.ino

- ❖ Felderítjük a WiFi hálózatot (forrás: <https://linuxhaxor.net/code/arduino-support-esp32-c3.html>)

```
#include "WiFi.h"

void setup() {
  Serial.begin(115200);
  Serial.println("Scanning available networks...");
  WiFi.mode(WIFI_STA); // station mode
  WiFi.disconnect(); delay(100);
}

void loop() {
  Serial.print("Scan start ... "); int n = WiFi.scanNetworks(); Serial.print(n);
  Serial.println(" network(s) found");
  for (int i = 0; i < n; i++) { Serial.printf("%d: %s, Ch:%d (%ddBm) %s ", i+1,
    WiFi.SSID(i).c_str(), WiFi.channel(i), WiFi.RSSI(i), WiFi.encryptionType(i) ==
    WIFI_AUTH_OPEN ? "open" : "");
    uint8_t* cc = WiFi.BSSID(i);
    for (int k = 0; k < 6; k++) { Serial.print(*cc, HEX);
      if (k != 5) Serial.print(":");
      cc++;
    }
    Serial.println("");
  }
  Serial.println(""); delay(5000); // 5 sec delay
}
```

wifi_scan.ino futási eredménye

```
COM7

Scan start ... 11 network(s) found
1: CSP-RN7, Ch:13 (-29dBm)  6E:BF:23:5:16:7F
2: HUAWEI-2.4G-9bQR, Ch:9 (-68dBm)  A:2F:E9:E9:F9:B0
3: DIGI-jX6H, Ch:11 (-85dBm)  50:1D:93:C6:9E:84
4: DIGI_e5ac18, Ch:5 (-88dBm)  0:67:62:E5:AC:18
5: Digi46, Ch:4 (-90dBm)  CC:32:E5:23:DA:83
6: DIGI-vKHx, Ch:2 (-91dBm)  3C:E8:24:E3:FE:F0
7: DIGI-8Usb, Ch:9 (-92dBm)  3C:E8:24:E3:1C:F8
8: Telekom-735285, Ch:13 (-92dBm)  2:34:AF:90:15:95
9: DIGI-77fS, Ch:1 (-93dBm)  14:9:DC:D1:FC:F0
10: Telekom-969415, Ch:12 (-94dBm)  B0:5B:99:1F:AA:E6
11: Telekom-1SrW9Z, Ch:1 (-95dBm)  B8:EE:E:55:3A:E5

Scan start ... 14 network(s) found
1: CSP-RN7, Ch:13 (-30dBm)  6E:BF:23:5:16:7F
2: Digi46, Ch:4 (-79dBm)  CC:32:E5:23:DA:83
```

Itthoni eszközök

Autoscroll Show timestamp

Newline 115200 baud Clear output

Csatlakozás a WiFi hálózathoz

- ❖ Az **ESP32 Wi-Fi** perifériája és az **ESP32 Arduino Core** részét képező WiFi programkönyvtár lehetővé teszi, hogy a helyi hálózatra csatlakozzunk (kliensként **STA** módban, vagy elérési pontként **AP** módban)
- ❖ Most a kliens (**STA**) módot fogjuk használni

```
#include <WiFi.h>
#include "secrets.h"

void setup_wifi() {
  delay(2000);
  WiFi.mode(WIFI_STA);
  Serial.print("\r\nConnecting to ");
  Serial.println(WIFI_SSID);
  WiFi.begin(WIFI_SSID, WIFI_PASS);
  while (WiFi.status() != WL_CONNECTED) {
    delay(500);
    Serial.print(".");
  }
  Serial.println("WiFi connected");
  Serial.print("IP address: ");
  Serial.println(WiFi.localIP());
}
```

- ❖ A személyes adatokat kiszerveztük egy fejléc állományba, amelyet a **Vázlatfüzet** (Sketchbook) mappa **libraries/secrets** almappjában helyeztünk el

secrets.h

```
#define WIFI_SSID MY_SSID
#define WIFI_PASS MY_PASSWORD
```


ESP32_sntp.ino

```
#include <WiFi.h>
#include "time.h"
#include "secrets.h"

// TimeZone rule for Europe/Budapest including daylight adjustment rules (optional)
// See at: https://leo.leung.xyz/wiki/Timezone
const char* time_zone = "CET-1CEST,M3.5.0,M10.5.0/3";
const char* ntpServer = "hu.pool.ntp.org";

struct tm timeinfo;
void setup() {
  Serial.begin(115200);
  setup_wifi();           // Connecting to WiFi AP
  configTzTime(time_zone, ntpServer);
}

void loop() {
  delay(5000);
  if (getLocalTime(&timeinfo)) {
    Serial.println(&timeinfo, "%A, %B %d %Y %H:%M:%S");
  }
  else {
    Serial.println("*** Failed to obtain time ***");
  }
}
```

Lekérjük a pontos időt a Közép-Európai Időzónára vonatkozóan (a téli/nyári időszámítás automatikus figyelembevételével)

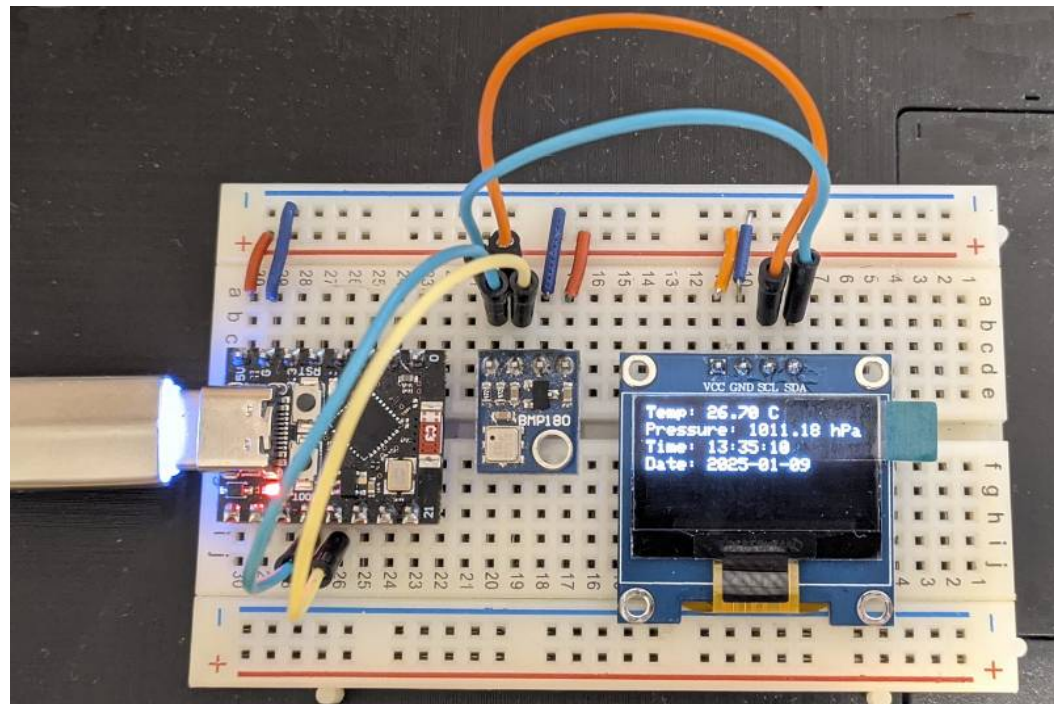
ESP32_sntp.ino futási eredménye

```
COM6
Connecting to HUAWEI-2.4G-9bQR
.....WiFi connected
IP address: 192.168.100.50
Wednesday, January 15 2025 15:05:44
Wednesday, January 15 2025 15:05:49
Wednesday, January 15 2025 15:05:54
Wednesday, January 15 2025 15:05:59
Wednesday, January 15 2025 15:06:04
Wednesday, January 15 2025 15:06:09
Wednesday, January 15 2025 15:06:14
Wednesday, January 15 2025 15:06:19
Wednesday, January 15 2025 15:06:24
Wednesday, January 15 2025 15:06:29
Wednesday, January 15 2025 15:06:34
Wednesday, January 15 2025 15:06:39
```

sntp_bmp180_oled.ino

A következő példaprogramban a következő tevékenységek zajlanak:

- Inicializáljuk az **SSD1306** kijelzőt és a **BMP180** szenzort
- Bejelentkezünk a **WiFi** hálózatba
- Konfiguráljuk az **SNTP** klienst az **ESP32**-n, hogy az eszköz az aktuális időt az **NTP** szerverről szinkronizálja a megadott **CET** időzónában
- A **loop()** függvényben periodikusan kiolvassuk a **BMP180** szenzorról a hőmérséklet és a légnyomás értékét,
- majd kiíratjuk az OLED kijelzőre
- A kijelző 3. és 4. sorába pedig kiíratjuk a helyi időt és a dátumot



sntp_bmp180_oled.ino – 3/1.

```
#include <Wire.h>
#include <Adafruit_BMP085.h>
#include <Adafruit_SSD1306.h>
#include <WiFi.h>
#include <time.h>
#include "secrets.h"

// TimeZone rule for Europe/Budapest including daylight adjustment rules (optional)
// See at: https://leo.leung.xyz/wiki/Timezone
const char* time_zone = "CET-1CEST,M3.5.0,M10.5.0/3";
const char* ntpServer = "hu.pool.ntp.org";
struct tm timeinfo;

// OLED kijelző beállításai
#define SCREEN_WIDTH 128
#define SCREEN_HEIGHT 64
#define OLED_RESET -1
Adafruit_SSD1306 display(SCREEN_WIDTH, SCREEN_HEIGHT, &Wire, OLED_RESET);

// BMP180 szenzor beállításai
Adafruit_BMP085 bmp;
```

sntp_bmp180_oled.ino – 3/2.

```
void setup() {
  Serial.begin(115200);

  // OLED kijelző inicializálása
  if (!display.begin(SSD1306_SWITCHCAPVCC, 0x3C)) {
    Serial.println(F("SSD1306 allocation failed"));
    for (;;)
  }
  display.display();
  delay(2000);
  display.clearDisplay();

  // BMP180 szenzor inicializálása
  if (!bmp.begin()) {
    Serial.print("Could not find a valid BMP085 sensor, check wiring!");
    while (1);
  }

  setup_wifi();
  configTzTime(time_zone, ntpServer);
}
```

sntp_bmp180_oled.ino – 3/3.

```
void loop() {
  char buffer[128];
  float temperature = bmp.readTemperature();
  float pressure = bmp.readPressure() / 100.0F;

  display.clearDisplay();
  display.setTextSize(1);
  display.setTextColor(SSD1306_WHITE);
  display.setCursor(0, 0);
  display.print("Temperature: ");
  display.print(temperature);
  display.println(" C");
  display.print("Pressure: ");
  display.print(pressure);
  display.println(" hPa");

  // Idő és dátum kiírása
  struct tm timeinfo;
  if (!getLocalTime(&timeinfo)) {
    Serial.println("Failed to obtain time");
    return;
  }
}
```

```
display.print("Time: ");
sprintf(buffer, sizeof(buffer),
"%02d:%02d:%02d", timeinfo.tm_hour,
timeinfo.tm_min, timeinfo.tm_sec);

display.println(buffer);
display.print("Date: ");
sprintf(buffer, sizeof(buffer),
"%04d-%02d-%02d", timeinfo.tm_year + 1900,
timeinfo.tm_mon + 1, timeinfo.tm_mday);

display.println(buffer);
display.display();
delay(2000);
}
```