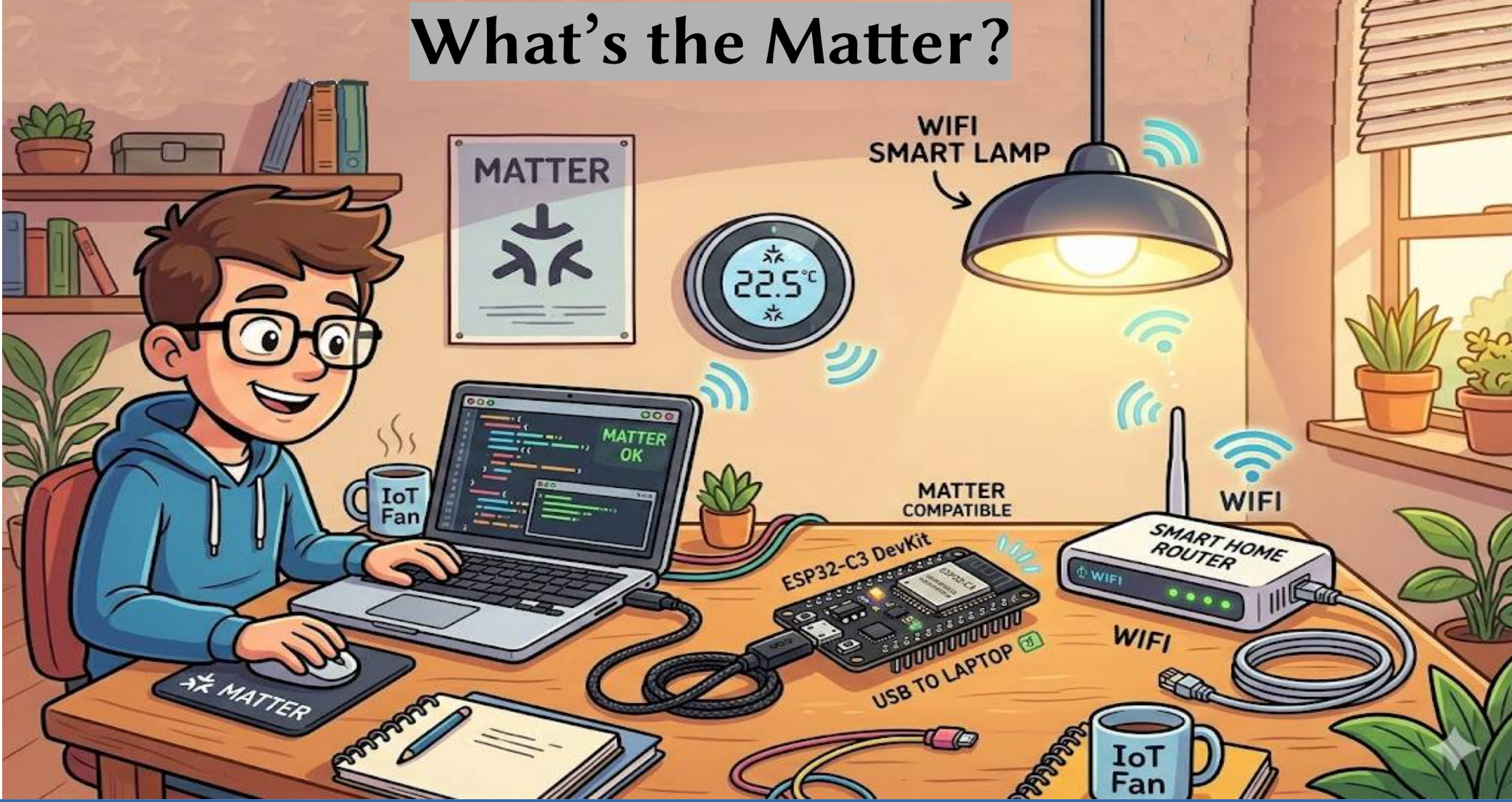
What's the Matter?



Felhasznált és ajánlott irodalom

- Espressif: Espressif's Solution for Matter
- Silicon Labs: How Matter is Unifying the Smart Home
- Matter fejlesztők: Matter Handbook
- Connectivity Standards Alliance: Matter: The Foundation for Connected Things
- Espressif: Espressif Zerocode
- Espressif: Arduino core for the ESP32 family of SoCs
- Espressif: ESP32 Arduino Core's documentation
- Espressif: ESP-IDF Programming Guide
- Espressif: Espressif's SDK for Matter Programming Guide
- IKEA: DIRIGERA Hub okos termékekhez

Az okosotthon dilemma

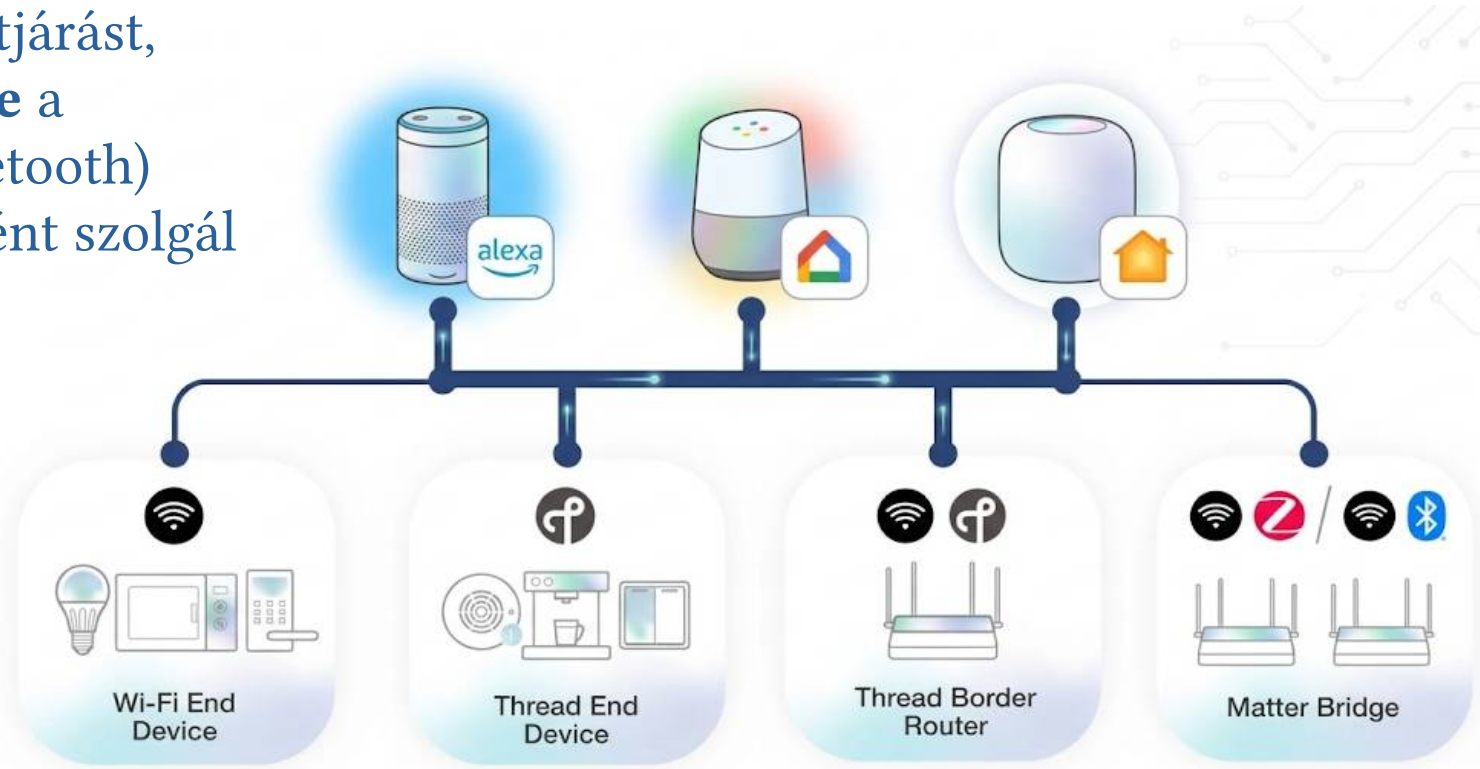
- **Alapprobléma:**
  Túl sok vezeték nélküli technológia
  Zárt ökoszisztémák
  Sokféle app, hub és átjáró kell
- **Gyártók:**
  Választaniuk kell a rendszerek között; ugyanabból a termékből több verziót (SKU) kell gyártaniuk
- **Kereskedők:**
  Duplikált készletek a polcokon; nehéz szaktanácsot adni; sok a kompatibilitás miatti visszaküldés
- **Fogyasztók:**
  Vásárlási bizonytalanság; az eszközök nem kombinálhatóak szabadon; nehézkes a váltás a rendszerek között
- **Megoldás: Matter –** az egységes szabvány

Mi a Matter?

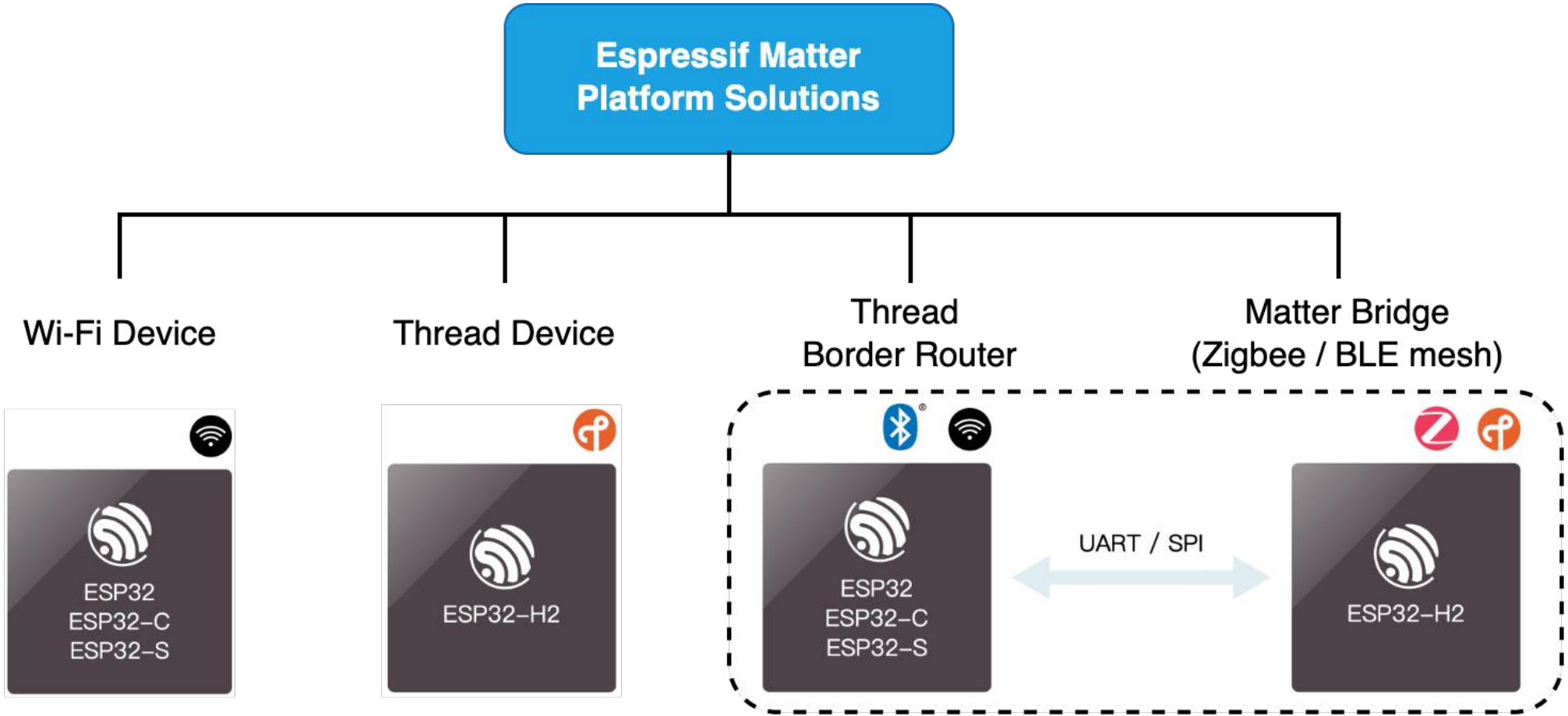
- A **Matter** egységes, IP-alapú okosotthon-szabvány, amely biztosítja, hogy különböző gyártók okosotthon eszközei és a különböző szolgáltatók IOT platformjai együttműködjenek
- 2019-ben indult, 1.0 verziója 2022-ben jelent meg; létrehozója és fenntartója a **Connectivity Standards Alliance** (CSA), több mint 500 iparági szereplővel
- A **Matter-tagság** lehetővé teszi, hogy a gyártók eszközei hivatalos digitális tanúsítványt kapjanak. Ez a tanúsítvány a **Matter-eszköz** között titkosított, biztonságos kommunikáció alapja
- A **Matter kommunikáció** a nagy sávszélességet igénylő eszközöknél a **Wi-Fi-t,** az energiatakarékos kiegészítőknél pedig a **Thread** technológiát használja, amely a **Zigbee** által is alkalmazott IEEE 802.15.4 rádiós réteget ruházza fel natív IP-képességgel. A folyamatba a **BLE** is bekapcsolódik, amely az első találkozáskor az eszközök biztonságos hitelesítéséért és a hálózathoz való titkosított csatlakozásáért felel
- A „hagyományos” **Zigbee** vagy **Z-Wave** eszközök egy **Matter-képes** központi egység (Bridge/Hub) segítségével integrálhatók. Ez a központ „tolmácsként” működik: a saját rádiós szabványán (pl. Zigbee) fogadja a régi eszközök jeleit, majd azokat **Matter**-protokollá fordítva továbbítja a hálózat többi része felé

Matter hálózati architektúra

- A Matter ökoszisztémában a Wi-Fi és a Thread végpontok alkotják a hálózat gerincét. A **Thread Border Router** biztosítja a két technológia közötti zökkenőmentes IP-átjárást, míg a **Matter Bridge** a régebbi (Zigbee/Bluetooth) eszközök tolmácsaként szolgál

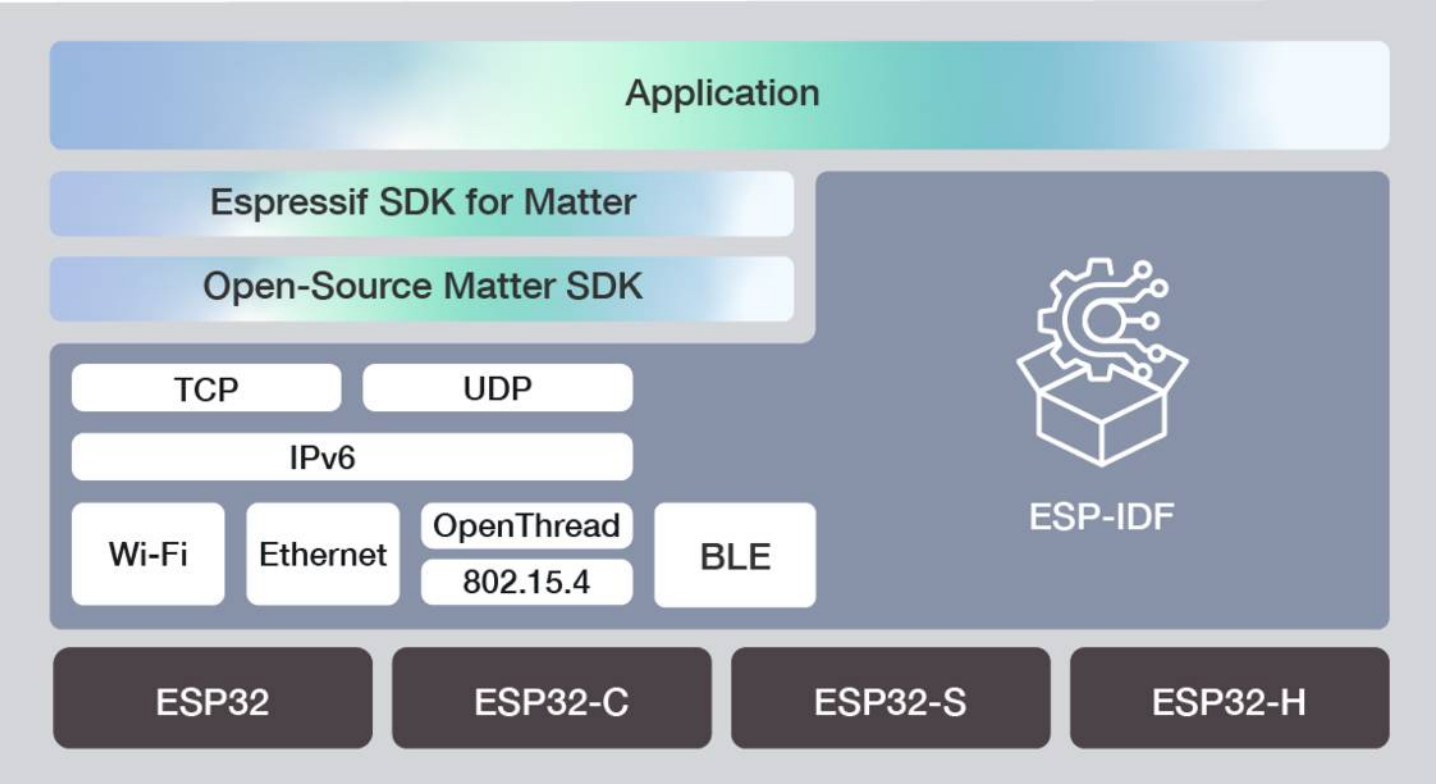


Espressif Matter HW platformok



Espressif Matter szoftverplatform

- **Open-Source Matter SDK:** a közösségi összefogással fejlesztett hivatalos alap
- **Espressif Matter SDK** ennek az **ESP32** chipekre optimalizált kiegészítése



ESP32 Matter Fejlesztési Keretrendszerek

1. ESP-Zero Code Online

Előnyök:
+ Kattintásalapú fejlesztés
+ Cloud-alapú konfiguráció

Hátrányok:
- Korlátozott testreszabás
- Nem nyílt forráskódú

2. Arduino IDE (Core & Matter)

Előnyök:
+ Ismert, könnyű környezet
+ Hatalmas közösségi támogatás

Hátrányok:
- Korlátozott mélység
- Fapadosabb API-k

3. ESPRESSIF SDK (IDF)

Előnyök:
+ Teljes kontrol és mélység
+ Professzionális, skálázható

Hátrányok:
- Meredek tanulási görbe
- Komplex konfiguráció

ESP Zerocode – a királyi út

- Az ESP ZeroCode lehetővé teszi, hogy „kulcsrakész”, hitelesített Matter-kompatibilis firmware-t kapjunk csupán konfigurálással, kódírás és tanúsítvány beszerzése nélkül

A fejlesztés lépései:

- **Eszköz létrehozása névvel:** a gyártó (vagy hobbista) megadja az eszköz nevét
- **Eszköztípus kiválasztása:** pl. WS2812 LED Strip, relé, dimmer, ventilátor stb. – ez határozza meg a Matter-profil működését
- **Modul kiválasztása:** pl. ESP32-C3 Devkit1 – ez adja a hardveres alapot, amelyre a firmware épül
- **Device konfigurálása:** a GPIO-k és funkciók beállítása (melyik láb mit vezérel, gomb-logika, LED-visszajelzés stb.)
- **Test Mode:** a konfiguráció kipróbálása és validálása, mielőtt végleges firmware vagy modulok rendelése készül belőle

WiFi Matter On/Off kapcsoló

- Első mintapéldánkat („okoskonnektor”) az Espressif Zerocode online fejlesztői platformján készítettük el. Konfigurálás után letölthetjük a firmware-t és kapunk mellé egy számot (manual pairing code) és egy QR kódot

CONFIGURATION

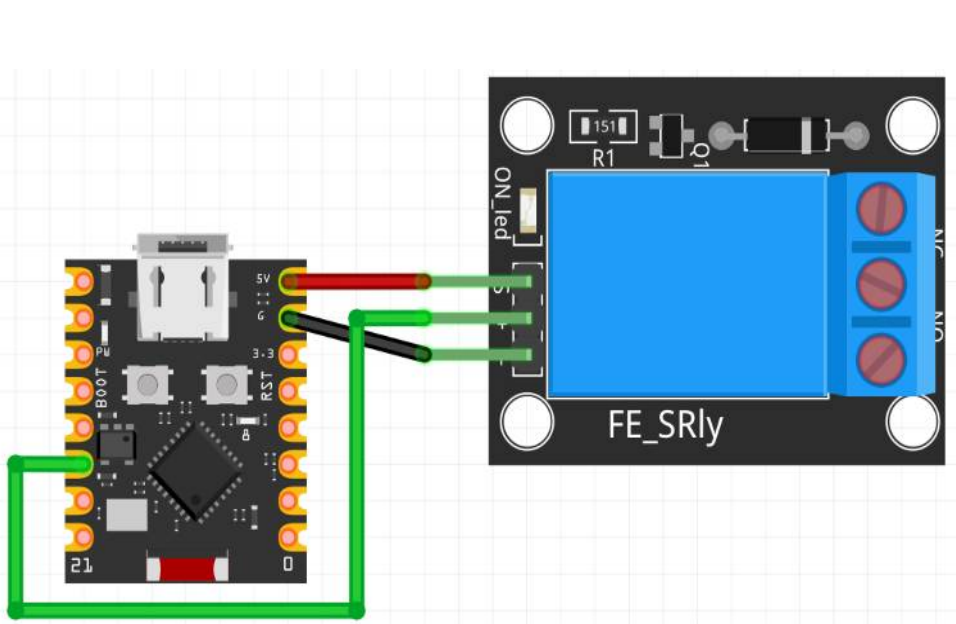
Eszköz neve: **csp_onoff**

Device: **Plug**
Module: **ESP32-C3**

Input button: **GPIO9**
Active High: **no**

Output relay: **GPIO10**
Active high: **yes**

LED: **GPIO8**
Active high: **no**



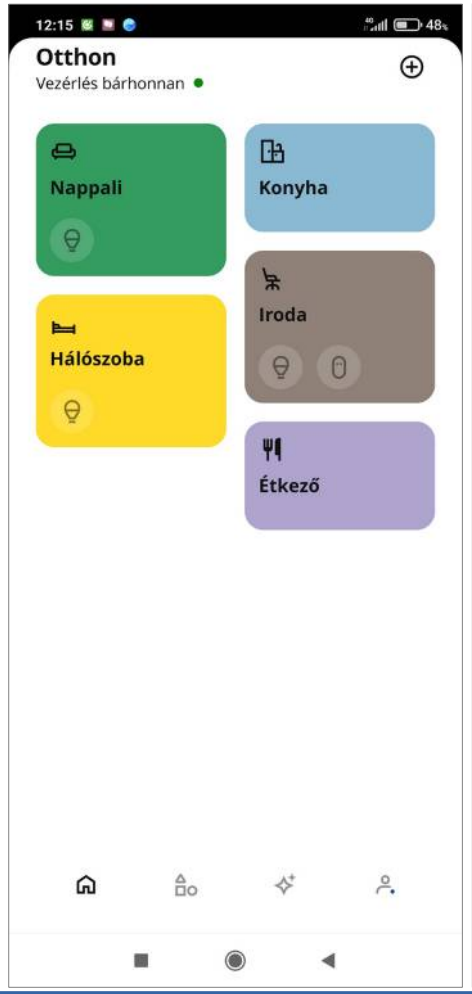
matter

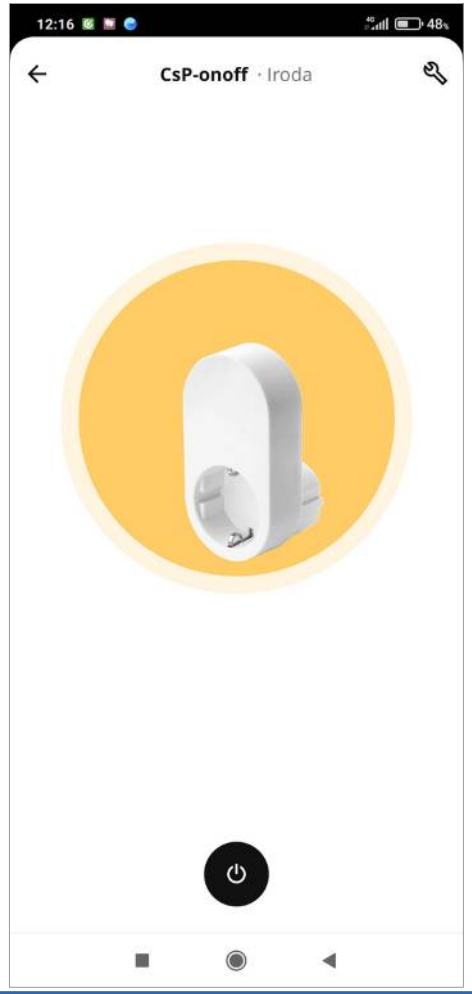
1118-544-7754

Kapcsolódás IKEA DIRIGERA-hoz

- Az **IKEA Home** app és a **DIRIGERA** hub teljes körű Matter-támogatással rendelkezik
- **Hirdetés** (Beaconing): Az **ESP ZeroCode**-dal készített eszköz BLE-n hirdeti magát
  Az **IKEA Home** app automatikusan jelzi: "Új eszköz a közelben, csatlakozunk?"
- **Azonosítás:** A felhasználó beolvassa a QR-kódot vagy beírja a 11 jegyű párosítási kódot
- **Biztonsági kézfogás** (PASE): Titkosított csatorna épül ki a telefon és az ESP32 között
- **Wi-Fi provisioning:** Az **IKEA Home** app Bluetooth-on átadja a helyi Wi-Fi hitelesítő adatait az eszköznek (ne felejtsük el a telefont a 2.4 GHz-es hálózathoz rendelni)
- **Hálózati váltás:** Az ESP32 lekapcsolja a BLE-t, és feljelentkezik a Wi-Fi hálózatra
- **Üzemi hitelesítés:** A DIRIGERA hub egyedi digitális tanúsítványt (NOC) állít ki az eszköznek
- **Vezérlés:** Az eszköz megjelenik az **IKEA Home**-ban és készen áll a csoportosításra/automatizálásra.
- **Multi-Admin:** Az eszköz ezután párhuzamosan más Matter-appokhoz (pl. Apple Home) is hozzáadható

A Matter eszköz az IKEA Home-ban





Szabályozható fényű lámpa

- Második mintapéldánkat is az Espressif Zerocode online fejlesztői platformján készítettük el. Itt egy PWM szabályozású LED fényforrást választottunk.
  A hőmérő nem tartozik a kapcsoláshoz.

CONFIGURATION

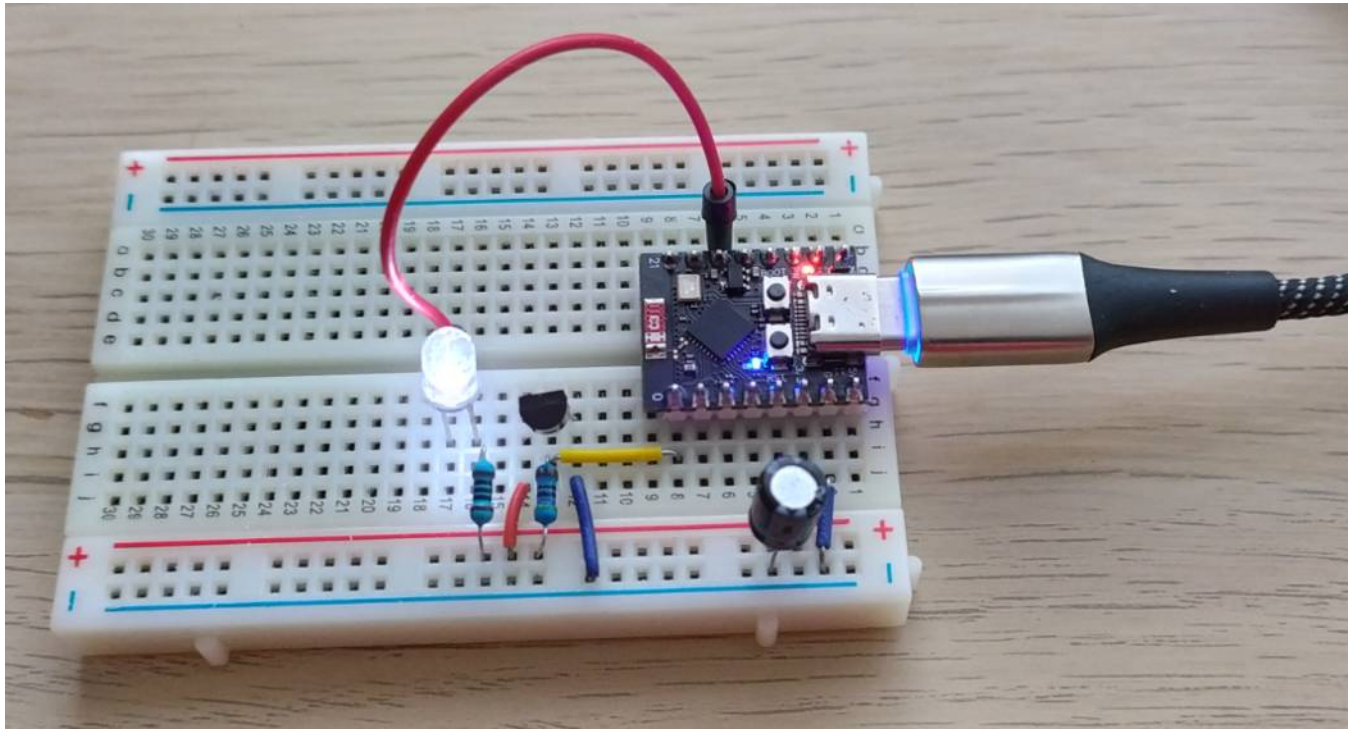
Eszköz neve: **csp_dimmable**

Device: **Lighting Fixture**

Driver: **PWM | Brightness**
Module: **ESP32-C3**

Input button: **GPIO9**
Active High: **no**

LED Output: **GPIO4**
PWM freq: **4000 Hz**



Az ESP Zerocode kötöttségei

- Az **ESP Zerocode** fő előnye, hogy egyedi egyedi **DAC** (Device Attestation Certificate) és egyedi QR-kód/Manual Pairing Code azonosítókkal ellátott, gyárilag hitelesített firmware-t ad, így kódolás nélkül is piackész termék hozható létre
- **Kötöttségek:** csak az előre definiált eszközprofilok és hardveres meghajtók használhatók, egyedi programozásra vagy ismeretlen perifériák kezelésére nincs lehetőség
- **Eszköztípusok:** Air Conditioner, Blinds, Contact Sensor, Dimmer, Laundry Washer, Light, multi_sensor, Multi-Purpose Switch, Occupancy Sensor, Plug, Refrigerator, Socket, temperature_sensor, Thermostat
- **Driverek:** aht20, BP1658CJ, BP5758D, GPIO, I2C, Physical Contact, PIR, PWM, sht30, SM2335EGH, UART, Ultrasonic, WS2812

Matter fejlesztés Arduino IDE-ben

Matter fejlesztés Arduino IDE-ben

- **Feltételek:** Arduino IDE (pl. 1.8.19), ESP32 Core 3.1.1 és min. 4MB Flash memória
- **Beállítás:** Partition Scheme -> RainMaker vagy Huge APP (min. 3MB APP hely kell)
- Kezdésnek az ESP32 Matter library mintapéldáit nézzük (pl. MatterTemperatureSensor)
- **Munkafolyamat:** A mintakód inicializálja a Matter stack-et, nekünk a fizikai I/O-t kell a Matter klaszterekhez rendelni
- **Párosítás:** Manuális kóddal (11 jegyű PIN); a QR-kód Arduinónál nálam nem működött
- **Előnyök:**
  - Ingyenes, és bármely Arduino könyvtár (szenzorok) beépíthető
  - Teljes kontroll a helyi logika felett (pl. offline gombkezelés)
- **Hátrányok:**
  - Extrém hosszú fordítási idő és hatalmas memóriaigény
  - Csak az alapvető Matter eszköztípusok érhetők el (korlátozott mélység)
  - Nincs egyedi DAC tanúsítvány; az eszköz „nem hitelesített” marad
  - Az egyedi azonosítók hiánya miatt egyszerre csak egy példány párosítható stabilan

Mi történik az NVM memóriában?

- Az ESP32 Flash memóriájának egy elkülönített, „nem felejtő” részébe kerülnek a Wi-Fi adatok, a biztonsági kulcsok és a hálózati azonosítók a párosítás során
- **Firmware frissítés:** Az új kód rátöltése csak az alkalmazás-partíciót írja felül, az NVS tartalmát nem bántja. Az eszköz újraindítás után azonnal felismeri a hálózatot, nem kell újra párosítani
- **Decommissioning:** Ez a folyamat törli a Matter-specifikus adatokat és kulcsokat az NVS memóriából. Ekkor az eszköz „elfelejti” az okosotthon központot (pl. DIRIGERA) és a Wi-Fi hozzáférést
- **Visszaállítás (Reset):** Csak a **Matter.decommission()** hívás vagy a teljes Flash-memória törlés után válik az eszköz újra párosíthatóvá
- **Kódból vezérelve:** A 5 másodperces gombnyomásra programozott törlés a legbiztosabb módszer a manuális resetre

MatterTemperatureSensor.ino

- Az ESP32 Matter library mintapéldája valójában csak szimulált adatokat küld ki, de arra jó, hogy a felismertetést és az adatküldést kipróbáljuk vele
- A „gyári” programban csak a helyi WiFi bejelentkezési adatait kell beírni vagy becsatolni

```
#include <Matter.h>
#include <WiFi.h>
#include "secrets.h"                        // WiFi credentials

// List of Matter Endpoints for this Node
// Matter Temperature Sensor Endpoint
MatterTemperatureSensor SimulatedTemperatureSensor;

// WiFi is manually set and started
const char *ssid = WIFI_SSID;               // Change this to your WiFi SSID
const char *password = WIFI_PASS;           // Change this to your WiFi password
```

```
ESP-ROM:esp32c3-api1-20210207
Pairing code:
34970112332
QR code URL:
https://project-chip.github.io/connectedhomeip/qrcode.html?data=MT:Y.K9042C00KA0648G00
```

A Matter kezelés váza

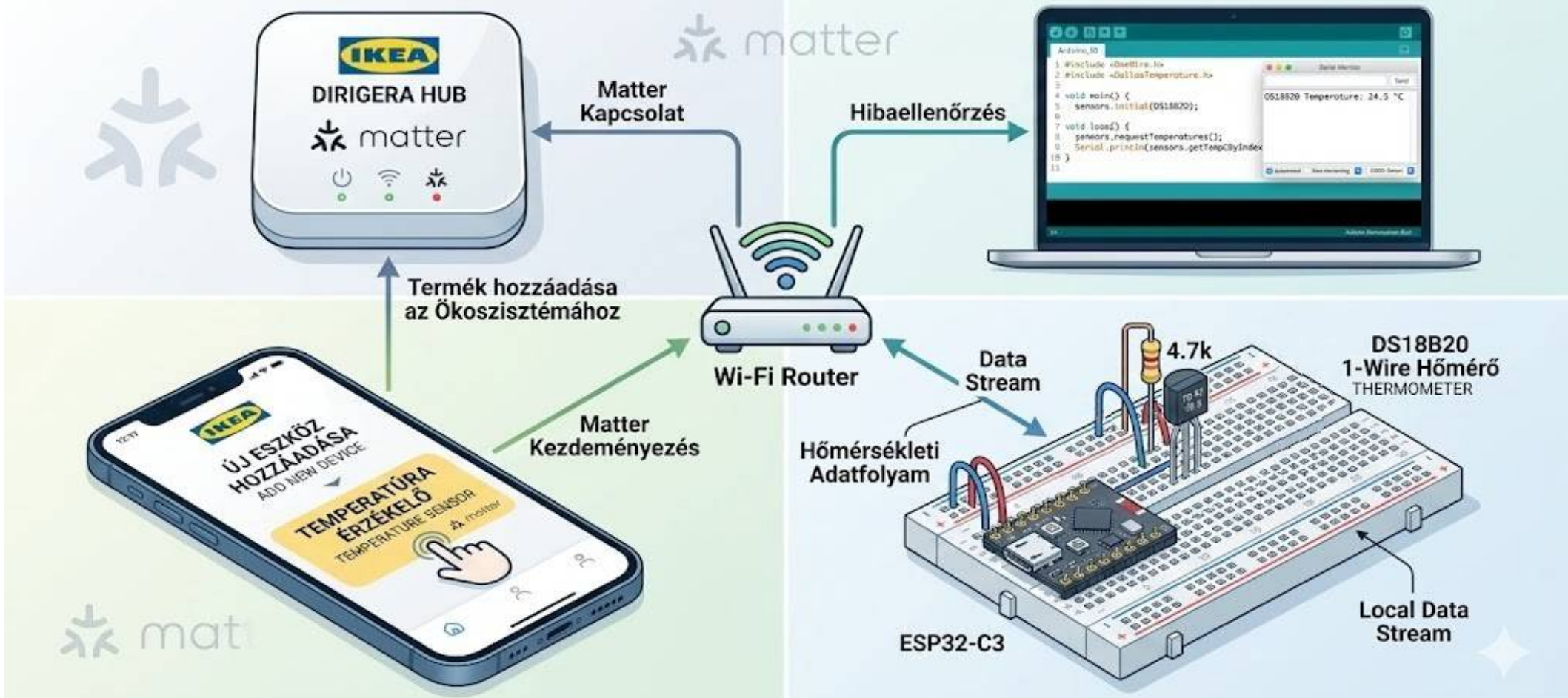
```
#include <Matter.h>
MatterTemperatureSensor TemperatureSensor;              // 1. Az eszköz típusának deklarálása

void setup() {
    TemperatureSensor.begin(20.0);                      // 2. Kezdőérték megadása és inicializálás
    if (!Matter.isDeviceCommissioned()) {
       String pin = Matter.getManualPairingCode();      // 3. Párosítási adatok lekérése
       String url = Matter.getOnboardingQRCodeUrl();
    }
    Matter.begin();                                     // 4. A Matter stack indítása
    while (!Matter.isDeviceCommissioned()) {
        delay(100);                                     // 5. Várunk, amíg a párosítás be nem fejeződik
    }
}

void loop() {
    float currentTemp = 25.5;                           // 6. Adatok (beküldés a Matter hálózatba)
    TemperatureSensor.setTemperature(currentTemp);
    if (/* gombnyomás */) {
        Matter.decommission();                          // 7. Gyári visszaállítás (Decommission)
    }
}
```

Matter Hőmérséklet Érzékelő Integráció (DS18B20 & IKEA)

Matter Temperature Sensor Integration (DS18B20 & IKEA)



Matter_DS18B20.ino

- Az előző programot kiegészítettük valódi hőmérsékletméréssel, amihez egy **DS18B20** 1-wire hőmérőt választottunk. Ezzel sikerült darászfészekbe nyúlni, mert a szokásos Onewire/DallasTemperature könyvtárak **ESP32-C3**-on nem működnek. A megoldás az **OnewireNg** programkönyvtár használata
- A hőmérsékletet 2 másodpercenként mérjük, de csak 10 másodpercenként publikáljuk, s közben gördülő átlagot számolunk
- A beépített LED a WiFi csatlakozásra várva villog, a comissionigra várva pedig folyamatosan világít, utána pedig elalszik
- Induláskor a program ellenőrzi, hogy történt-e már csatlakoztatás. Ha még nem (első bekapcsolás, vagy a decomissioning gomb min. 5 mp lenyomása után) akkor comissioningra vár és a soros porton kiírja a kézi párosítási kódot
- A program fő hátránya, hogy fixen beírt WiFi azonosítást használ

Matter_DS18B20.ino – 3/1.

```
#include <WiFi.h>
#include <Matter.h>
#include "secrets.h"
#include "OneWireNg_CurrentPlatform.h"
#include "drivers/DSTherm.h"
#include "utils/Placeholder.h"
#define OW_PIN 0
#define LED_PIN 8
static Placeholder<OneWireNg_CurrentPlatform> ow;

MatterTemperatureSensor TemperatureSensor;
float measureBuffer[5], lastSentTemp = 20.0, alpha = 0.8;
int measureIndex = 0;
unsigned long lastMeasureMs = 0, lastSendMs = 0, lastBlinkMs = 0, button_time_stamp = 0;
bool ledState = false, button_state = false;

void setup() {
    pinMode(LED_PIN, OUTPUT); digitalWrite(LED_PIN, HIGH);
    pinMode(BOOT_PIN, INPUT_PULLUP); Serial.begin(115200);
    new (&ow) OneWireNg_CurrentPlatform(OW_PIN, false);

    if (!Matter.isDeviceCommissioned()) {
        delay(200);
        Serial.printf("Pairing code: %s\n", Matter.getManualPairingCode().c_str());
        Serial.printf("QR code URL: %s\n", Matter.getOnboardingQRCodeUrl().c_str());
    }
```

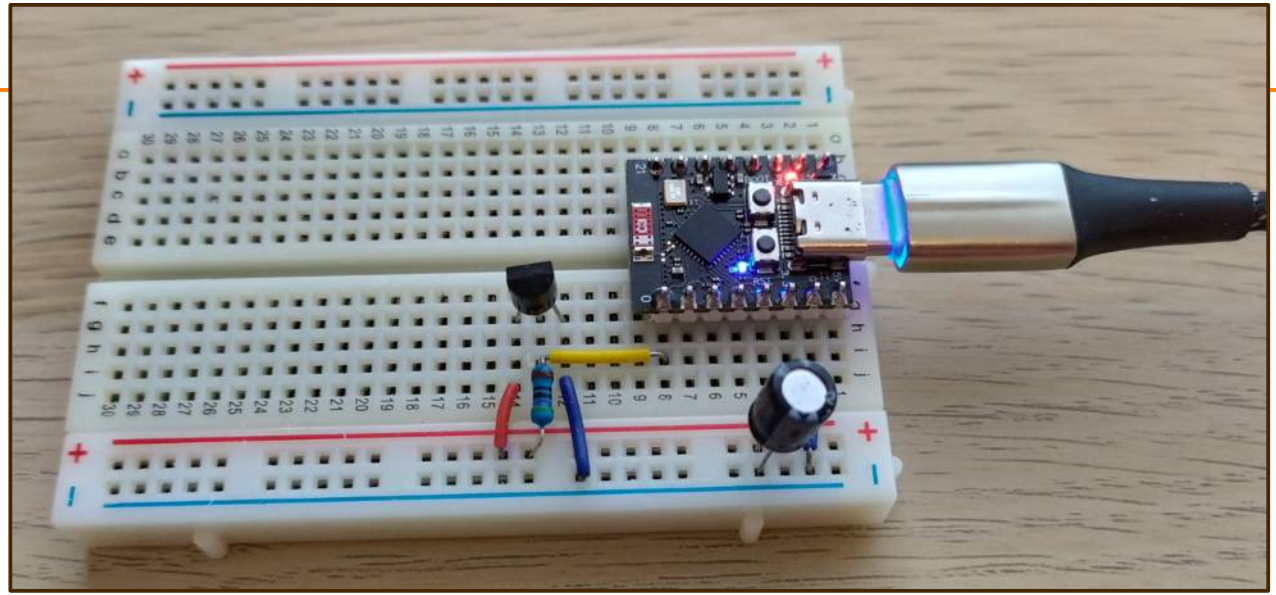
Matter_DS18B20.ino – 3/2.

```
    WiFi.begin(WIFI_SSID, WIFI_PASS);
    while (WiFi.status() != WL_CONNECTED) {
        if (millis() - lastBlinkMs > 300) {
            lastBlinkMs = millis(); ledState = !ledState;
            digitalWrite(LED_PIN, ledState ? LOW : HIGH);
        }
        delay(10);
    }
    digitalWrite(LED_PIN, LOW);
    TemperatureSensor.begin(lastSentTemp);
    Matter.begin();
    while (!Matter.isDeviceCommissioned()) delay(50);
    digitalWrite(LED_PIN, HIGH);
}

void loop() {
    unsigned long now = millis();
    if (now - lastMeasureMs >= 2000) {
        lastMeasureMs = now;
        DSTherm drv(ow); drv.convertTempAll(DSTherm::MAX_CONV_TIME, false);
        Placeholder<DSTherm::Scratchpad> scrpd;
        if (drv.readScratchpadSingle(scrpd) == OneWireNg::EC_SUCCESS) {
            measureBuffer[measureIndex] = (float)scrpd->getTemp2() / 16.0;
            measureIndex = (measureIndex + 1) % 5;
        }
    }
```

Matter_DS18B20.ino – 3/3.

```
    if (now - lastSendMs >= 10000) {
        lastSendMs = now; float sum = 0;
        for (int i = 0; i < 5; i++) sum += measureBuffer[i];
        lastSentTemp = alpha * lastSentTemp + (1.0 - alpha) * (sum / 5.0);
        TemperatureSensor.setTemperature(lastSentTemp);
    }
    bool bp = digitalRead(BOOT_PIN) == LOW;
    if (bp && !button_state) { button_time_stamp = now; button_state = true; }
    if (!bp && button_state) button_state = false;
    if (button_state && (now - button_time_stamp > 5000)) {
        Matter.decommission(); button_time_stamp = now;
    }
}
```



A hőmérő használata az IKEA Home rendszerben