
Matematikai és Informatikai Intézet

# Moduláris Plug & Play IoT / okos otthon rendszer

**Készítette**
Farkas Levente
Program tervező informatikus BSc

**Témavezető**
Dr. Geda Gábor
egyetemi docens

Eger, 2025

# Tartalomjegyzék

# Bevezetés

Az emberi faj a létezésétől kezdve lusta volt és lesz. Minden nap valamivel könnyebbé, egyszerűbbé akarjuk tenni a mindennapokat, hogy minél kevesebbel keljen foglalkoznunk és több időt tölthessünk más tevékenységgel. Az okos otthon rendszerek pontosan ezt teszik lehetővé, ezek régen drágák voltak így csak a tehetősebb emberek engedhették meg maguknak. Ma azonban olcsóbb ökoszisztémák is léteznek, de még így sem tudják sokan megfizetni. A projekt amiről ezen dolgozat szól egy olcsóbb, közösség alapú ökoszisztéma létrehozását tűzte ki célul.

Számomra elképesztő belegondolni abba, hogy pár sor kóddal ki tudunk hatni a környezetünkre. Főképp azért, mert a programozás szempontjából én mindig adatkezelésként tekintettem a munkánkra. Még a szakdolgozat témájának kiválasztása előtt kapcsolatba kerültem a mikrokontrollerek világával és attól a ponttól nem tudtam elengedni ezen eszközök varázsát. Tekintettel arra, hogy az áramkörök és elektromosság foglalkoztatott hobbi szinten már régóta, ez egy nagyszerű lehetőség volt, hogy kettő nagy szenvedélyemet, az informatikát és az áramköröket ötvözzem a szakdolgozatomban. Számomra hihetetlen volt akár csak egy LED[1] villanása is.

Szakdolgozatom termék-orientált. Célja az, hogy a kevésbé tehetős emberek számára is elérhetővé tegye az okos otthonok varázsát. Logikám szerint az okos eszközök árát nagyrészt a programozható mikrochipek teszik ki, ezeknek egy részről a hozzá tartozó áramkört kell vezérelniük, amely nem egy nehéz feladat, azonban az adatnak amely alapján ez működni fog valahogy el kell jutnia az eszközhöz, ez WiFi-n keresztül történik minden esetben, azonban így jelentősen megugrik ezek ára. A gondolatmenetemben egy egy eszközt képzeltem el, amely ezt a drága technológiát központosítja, ezzel a rendszer szummázott árát csökkentve.

Az eszköz önmagában nem képes sok mindenre, azonban a hozzá kapcsolódó „buta” modulok használatával lehetőség nyílik arra, hogy egy eszköz négy modult, azaz működésében különálló okos eszközt kezeljen. A fejlesztés idejében az eszköz négy modult képes befogadni, azonban ez áramkör és program módosítással növelhető.

Számomra fontos volt még az általam nem ismert, vagy nem használt technológiák használata, így próbáltam minél több számomra új technológiát használni, ezekről a technológiákról bővebben a 1. fejezetben olvashatnak.

[1] Light Emitting Diode, avagy fényt kibocsájtó dióda.

# 1. fejezet

# Technológiák

## 1.1. Eszköz oldal

### 1.1.1. Programozási nyelv

Ahogy később a a 2. fejezet a 2.1. szekciójában olvashatja az ESP32 eszközöket C/C++ nyelveken lehet főképp programozni, ezek mellett script nyelveket is lehet használni.

A megoldásban első sorban C++ nyelvet akartam használni, azonban ez a megközelítés felvetett számos problémát a megvalósítás közben (ezekről bővebben a a 2.2.2. alszekcióban olvashat). A problémák hatására elhagytam a C++ programozási nyelvet és MicroPython-ra váltottam.

A ESP32_GENERIC_S3-SPIRAM_OCT-20241129-v1.24.1 firmware-en történt a fejlesztés a váltás eredményeként. Ez elérhető a `https://micropython.org/download/` linken keresztül.

### 1.1.2. IDE

Alapvetően a Visual Studio Code volt a választott IDE a fejlesztéshez, azonban ez bármely extensiont próbáltam használni egyik sem tudta sikeresen feltölteni a programot az ESP-re. Így a fejlesztés első szakaszában a VSCode-ban megírt programot az Arduino IDE segítségével töltöttem fel.

A programozási nyelv váltása után a VSCode továbbra sem volt működőképes. Miután a probléma megoldására általam ismert módszerek tárháza kifogyott elkezdtem egy másik IDE után keresni. A keresésem eredményeként rátaláltam egy weblapra[1] ahol több fejlesztői környezetet említenek. A keresés folyamatában számos GitHub és StackOverflow beszélgetést néztem át válasz után kutatva. Ezekben a beszélgetésekben többször is említették a ThonyIDE-t. Ennek eredményéül kezdtem el használni az említett környezetet.

[1] `https://randomnerdtutorials.com/micropython-ides-esp32-esp8266/`

## 1.2. API

Szerencsére az API-hoz használt technológiával nem volt probléma, így ez Node.js-ben íródott Visual Studio Code felhasználásával. Ezt már használtam így nem tartalmazott különösebb meglepetést a fejlesztés.

## 1.3. Adatbázis

A MongoDB egy nyílt forráskódú, dokumentumorientált NoSQL adatbázis, amelyet skálázható és nagy teljesítményű alkalmazások fejlesztésére terveztek. A hagyományos relációs adatbázisokkal (pl. MySQL, PostgreSQL) ellentétben a MongoDB nem táblákban, hanem rugalmas, JSON-szerű dokumentumokban tárolja az adatokat. Ez lehetővé teszi a gyors és hatékony adatkezelést, különösen olyan esetekben, amikor az adatok szerkezete dinamikus vagy változó.

Fő jellemzői a MongoDB-nek:

1. Dokumentumorientált adatmodell:
   - Az adatokat BSON (Binary JSON) formátumban tárolja, ami egy bináris reprezentációja a JSON-nek.
   - Egy dokumentum egy kulcs-érték párokból álló struktúra, amely hasonlít a programozási nyelvekben használt objektumokhoz.
   - Példa egy dokumentumra:

```
{
    "_id": {
        "$oid": "67d6c6e5e3c0cb995623d001"
    },
    "owner": "testOwner",
    "location": "Unknown",
    "device_name": "Unnamed",
    "aditionalInfo": {
        "canBeControlledBy": [
            "testOwner2"
        ]
    }
}
```

Listing 1.1. Egy teszt dokumentum az adatbázisból.

2. Rugalmas séma:

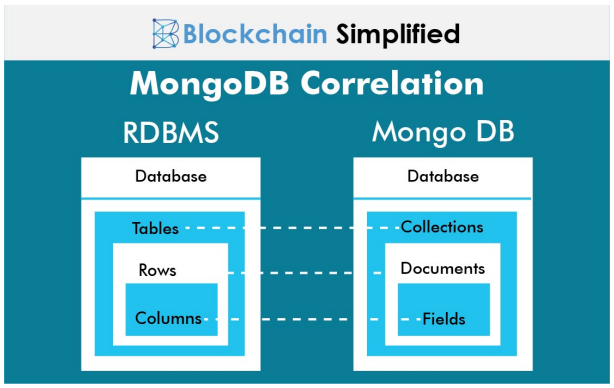
- A MongoDB nem követeli meg, hogy minden dokumentumnak ugyanaz a szerkezete legyen. Ez lehetővé teszi, hogy különböző típusú adatokat tároljunk ugyanabban a gyűjteményben (collection).
- Például egy users gyűjteményben lehetnek olyan dokumentumok, amelyeknek nincs age mezője, vagy más mezők is lehetnek.

3. Skálázhatóság:
   - A MongoDB horizontálisan skálázható, ami azt jelenti, hogy az adatbázis több szerverre (shard) osztható szét, hogy kezelni tudja a nagy mennyiségű adatot és a magas terhelést.
   - Támogatja a replikációt is, ami magas rendelkezésre állást és hibaelállást biztosít.
4. Teljesítmény:
   - A MongoDB gyors adatelérést biztosít indexek használatával. Többféle indexet támogat, beleértve az egyszerű, összetett, szöveges és geospatial indexeket.
   - A memóriában történő adatkezelés (in-memory storage) tovább növeli a teljesítményt.
5. Aggregáció és lekérdezés:
   - A MongoDB hatékony lekérdezési nyelvet (Query Language) és aggregációs keretrendszert (Aggregation Framework) biztosít az adatok elemzéséhez és feldolgozásához.

A MongoDB továbbá támogatja a CRUD műveleteket (Create, Read, Update, Delete), valamint a tranzakciókat is (a 4.0 verziótól kezdve).[1]

### 1.3.1. MongoDB struktúrája

A MongoDB az adatbázison belül táblák helyett collection-ök vannak, ezen belül rekordok, avagy sorokkal ellentétben dokumentumokat találunk, ahol oszlopok helyett mezők vannak, az összehasonlítást egy hagyományos SQL adatbázissal a 1.1. ábrán lehet látni. [1]




1.1. ábra. SQL adatbázis és MongoDB szerkezeti hasonlóságai.

# 2. fejezet

# ESP32, az eszköz

## 2.1. ESP32 bemutatása

Az ESP32 egy rendkívül sokoldalú és hatékony mikrokontroller, amely számos területen alkalmazható. Duális magos processzora, integrált Wi-Fi és Bluetooth képességei, valamint gazdag I/O lehetőségei miatt kiváló választás mind IoT, mind ipari vagy okos otthon alkalmazásokhoz. Az ESP32 fejlesztése egyszerű és gyors, köszönhetően a széles körben elérhető fejlesztői eszközöknek és könyvtáraknak. Ezek a tulajdonságok teszik az ESP32-t az egyik legnépszerűbb mikrokontrollerré a beágyazott rendszerek világában.

Az általam a 2.1 használt modell az ESP32-S3-DevKitC-1 N32R8V, amely egy ESP32-S3-WROOM-2 chipet tartalmaz, ezt az Espressif System fejlesztette ki.


2.1. ábra. Általam használt ESP32 modell

Ezen eszközök számos bemeneti és kimeneti perifériákkal rendelkeznek:

– Digitális I/O pin-ek

– Analóg bemenetek (ADC[1])

– Digitális-analóg átalakítók (DAC[2])

– PWM (Pulse Width Modulation[3]) kimenetek

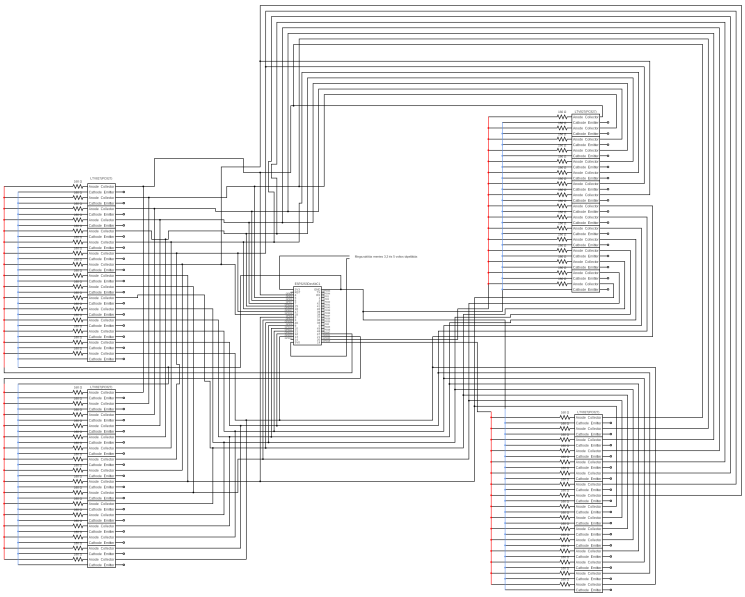
– SPI, I2C, UART kommunikációs interfészek

Ezek a csatlakozók lehetővé teszik különböző érzékelők, kijelzők és egyéb perifériák csatlakoztatását. Ezeket az eszközöket nagyrészt C és C++ nyelven lehet programozni, azonban van lehetőség a MicroPython és Lua script nyelvek használatára.

## 2.2. Fejlesztés

### 2.2.1. Hardver szintű fejlesztés

A projekt megvalósításához szükség volt egy egyedi áramkörre, amivel maximalizálhatom az elérhető portok számát. A 2.2-es ábrán látható az említett áramkör, amely



2.2. ábra. Egyedi áramkör

segítségével elérem az általam elfogadható eredményt. Ebben a fő alkatrész az optocsatoló, amellyel igyekeztem az esetleges interferenciát csökkenteni.

Ez a diagram a Circuit Diagram[4] segítségével készült. Az ábra nem tartalmazza az érzékelő vezetékeket, amivel az eszköz érzékeli, hogy csatlakoztatva van-e a modul.

[1] Analog to Digital Converter, lehetővé teszi az áramkörben érzékelt analóg jelek feldolgozását.
[2] Digital to Analog Converter, lehetővé teszi az áramkörre kibocsájtott feszültség modulálását.
[3] Ezen technológia teszi lehetővé a DAC működését.
[4] `https://www.circuit-diagram.org/`

### 2.2.2. Szoftver fejlesztése és feltárt problémák

# 3. fejezet

# API

# 4. fejezet

# Front end

# Irodalomjegyzék

[1] A szekció a Blockchain Simplified alapján készült. [Letöltve/látogatva:2025.03.21]