

Budapesti Műszaki és Gazdaságtudományi Egyetem

Méréstechnika és Információs Rendszerek Tanszék

**IoT alapú otthonautomatizálás megvalósítása ESP8266 használatával**

Diplomaterv

**Murányi Bence**

II. évf mérnökinformatikus szakos hallgató

MSc Intelligens rendszerek szakirány/ágazat

Konzulens:

Dr. Györke Péter

(Méréstechnika és Információs Rendszerek Tanszék)

Tartalomjegyzék

[ 1](#_Toc480969819)

[1 Abstract 6](#_Toc480969820)

[Bevezetés 7](#_Toc480969821)

[Az IoT-ről 7](#_Toc480969822)

[Intelligens otthon 7](#_Toc480969823)

[2 Irodalomkutatás 9](#_Toc480969824)

[2.1 Otthonautomatizálási rendszerek 9](#_Toc480969825)

[2.1.1 Crestron 9](#_Toc480969826)

[2.2 Alkalmazott technológiák 9](#_Toc480969827)

[2.2.1 JSON 9](#_Toc480969828)

[2.2.2 HighCharts 10](#_Toc480969829)

[3 Feladat 11](#_Toc480969830)

[3.1 A feladat célja 11](#_Toc480969831)

[3.2 Szenzor működési módok 11](#_Toc480969832)

[4 Irodalomkutatás 12](#_Toc480969833)

[4.1 Az ESP8266-ról 12](#_Toc480969834)

[4.2 ESP-12 tesztpanel 13](#_Toc480969835)

[4.3 Fejlesztőkörnyezetek 13](#_Toc480969836)

[4.3.1 AT-Command 13](#_Toc480969837)

[4.3.2 NodeMCU 14](#_Toc480969838)

[4.3.3 ESP SDK 14](#_Toc480969839)

[4.3.4 Arduino 14](#_Toc480969840)

[5 Tervezés 15](#_Toc480969841)

[5.1 A rendszer topológiája 15](#_Toc480969842)

[5.2 Működési módok 15](#_Toc480969843)

[5.2.1 Szenzor 15](#_Toc480969844)

[5.3 Működési módok 16](#_Toc480969845)

[5.3.1 Lokális szerver 16](#_Toc480969846)

[5.3.2 Beavatkozó 17](#_Toc480969847)

[5.3.3 Csatlakozás a hálózathoz 17](#_Toc480969848)

[5.4 Felhasználói interakciók 18](#_Toc480969849)

[5.4.1 Eszköz 20](#_Toc480969850)

[5.5 A központi szerver 22](#_Toc480969851)

[6 A projekt állapota 23](#_Toc480969852)

[6.1 Tervek a jövőre nézve 23](#_Toc480969853)

# Abstract

# Összefoglaló

Az elmúlt néhány évben egyre több otthonban megjelentek az épület működésének automatizálását célzó megoldások. Ezek a rendszerek folyamatosan fejlődnek, évről évre megjelennek felhasználói kényelemben és funkciókban előre lépést mutató fejlesztések. Ugyanakkor az is egyértelműen látszik, hogy a gyors elterjedés egyik legfőbb akadálya a termékek magas ára, a legolcsóbb üzleti megoldások is gyakran túl drágának bizonyulnak a hétköznapi ember számára.

Diplomatervem témája egy olyan okosotthon rendszer megtervezése és elkészítése, mely az üzleti megoldásokhoz viszonyítva töredékébe kerül, és képes kihelyezett mérőeszközök segítségével a felhasználók által megadott szabályokon keresztül automatikusan beavatkozni a környezetének működésébe.

A dolgozat első részében irodalomkutatást végzek, ahol néhány üzleti és nyílt forráskódú okosotthon rendszer bemutatásán keresztül képet kapunk az alapvető elvárásokról és lehetőségekről. A fejezet második részében bemutatom a szakdolgozat elkészítéséhez használt technológiákat és könyvtárakat.

Ezt követően előzetesen megfogalmazott elvárásokat szem előtt tartva megtervezek, majd meg is valósítok egy okosotthon rendszert, annak minden részére kitérve: elkészítem a szenzor és beavatkozó eszközök implementációját, és készítek egy webes felületet, ahol a felhasználó különösebb programozói tudás nélkül, grafikus felületen készíthet egyszerű szabályokat a saját otthona automatizálásához. A dolgozat utolsó részében egy hosszabb távú tesztüzem eredményeit figyelembe véve értékelem az elkészült rendszert, valamint támpontokat adok a továbbfejleszthetőségére vonatkozóan.

# Bevezetés

## Az IoT-ről

Az IoT (Internet of Things), mint fogalom, az elmúlt tíz évben kezdett el megjelenni, amikor is a végfelhasználót tekintve az internetre csatlakozó eszközök mennyisége lassan elkezdte meghaladni az internetezők mennyiségét. Ezt úgy kell érteni, hogy míg a hagyományos értelemben vett interneten a lánc végén valamilyen valós személy generálja a tartalmat (pl. egy email üzenet), addig az IoT lényegében arról szól, hogy egy központi szerverrel kapcsolatot fenntartva, a csomópontokban automatikus módon, emberi közreműködés igénybevétele nélkül gyűjtsünk adatokat (amiket aztán továbbítunk a szervernek), vagy végezzünk beavatkozásokat a környezetünkben. A dolgok internetében az eszközeink kapcsolatban vannak egymással, kommunikációs csatornának az internetet használva.

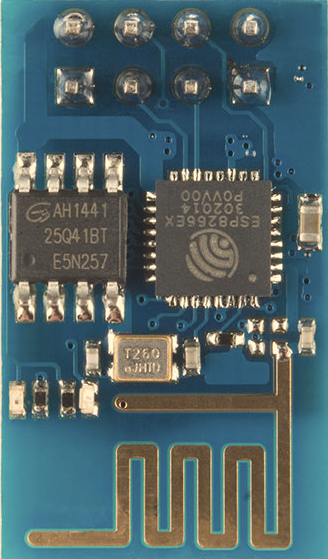
Látható, hogy a környezetünkben külön ráfordítás nélkül is már most rengeteg potenciális IoT eszköz van (okostelefon, autó, stb.), a benne rejlő lehetőségek miatt a téma jelenleg rendkívül aktuális.

## Intelligens otthon

Az intelligens otthon kifejezés a lakás automatizáltságát takarja, a fogalomra sokféle definíció létezik, hiszen rengeteg, önmagukban véve opcionális funkció tartozhat alá. Az otthonunkban megtalálható eszközök automatizáltsága már viszonylag régi jelenség, például az adott időpontban automatikusan bekapcsoló öntözőrendszer vagy az előre programozható mosógép már sok évvel ezelőtt sem voltak elképzelhetetlenek a lakásokban. Az otthon intelligens irányítása egy magasabb szintben gondolkozik, célja, hogy az önmagukban is akár bizonyos mértékben automatizált eszközeinket egy közös rendszer alá rendeljük, és ezen keresztül egységesen kezelhessük őket, hogy aztán a beállításainknak megfelelően automatikus működésre legyenek képesek. Az intelligens otthon célja, hogy a modern infokommunikációs technológiák révén az otthonban végzett hagyományos és újabb tevékenységeket kényelmesebbé, gyorsabbá és nem utolsósorban energiatakarékosabbá tegye. Példának okáért, amikor az utolsó személy is távozik a lakásból, a rendszer a beállításoknak megfelelően automatikusan lekapcsolná az égve felejtett villanyokat, bekapcsolná a riasztót és kikapcsolná vagy lejjebb venné a fűtést. Ugyanígy nem elképzelhetetlen, hogy az okostelefonos applikációnkkal (a telefon GPS [[1]](#footnote-1)helyadatai alapján akár automatikusan) munkából hazaindulás előtt jelezzük az otthonunknak, hogy készüljön fel a hazaérkezésünkre, így otthon már nem a hideg lakás fogad minket. Ha a munkából rendszeres időpontban megyünk haza, ez a működés akár automatizálható is, és ha túlóra vagy bevásárlás miatt kicsit megcsúszunk, ráérünk akkor jelezni ezt a rendszernek, ami alkalmazkodik a későbbi érkezéshez. A lakásban, ha szükség van rá (például ha egy filmet akarunk megnézni a tévénken), automatikus módon leereszthetőek a redőnyök a médialejátszó elindításával. Az öntözőrendszer beállításainknak megfelelően éjszaka bekapcsol, kivéve, ha nap közben jelentősebb mennyiségű csapadék esett, amit a rendszer például saját érzékelői és internetről letöltött adatok segítségével érzékel. A hűtőszekrény automatikusan értesít, ha elfogyott a tej, és akár a beállításoknak megfelelően maga is megrendelheti a szükséges dolgokat az interneten. Látható, hogy a lehetséges funkcióknak lényegében csak a képzeletünk és az anyagi lehetőségeink szabnak határt.

## ESP8266

Az ESP8266 egy az Espressif Systems által fejlesztett és gyártott chip, mely elsősorban vezeték nélküli (WIFI) internetkapcsolat létesítésére szolgál.



1. ábra ESP-01

Az eszköz legalapvetőbb tulajdonságai:

|  |  |
| --- | --- |
| Processzor |  |
| Órajel | 80 MHz |
| Támogatott vezetéknélküli szabványok |  |
| RAM |  |
|  |  |
|  |  |

* 32 bites ARM processzor
* 80 MHz órajel
* 96 KB data RAM
* 802.11 b/g/n szabványok támogatása
* WIFI 2.4 GHz (WPA/WPA2)
* SPI, I2C, ADC, PWM interfészek
* Fogyasztás
  + Alap: <300mA
  + Sleep[[2]](#footnote-2): <100µA

További pozitív tulajdonsága az ára: a pontos típustól függően már 1.5$ körüli áron hozzájuthatunk, ami azt tekintve, hogy ezt megelőzően egy hasonló igényeket kiszolgálni képes Wifi modul nagyságrendileg 25-40$ körül volt kapható, így ez rendkívül kedvező. A Láthatjuk, hogy az ESP8266 saját ARM processzorral rendelkezik, ennek köszönhetően lehetővé válik a standalone működés is. A fenti tulajdonságok (saját számítókapacitás, kivezetések, kis méret, stb.) alkalmassá teszik arra a feladatra, hogy szenzorhálózat csomópontjaiban, egy központi szerverrel kommunikálva elvégezze a szükséges méréseket és a kinyert adatok továbbítását.

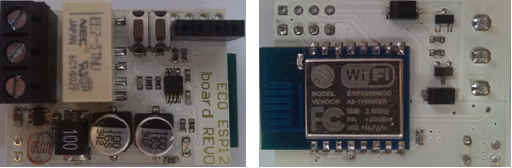
Megvalósítását tekintve az eddigiek során két modullal foglalkoztam. A fenti képen is megtekinthető ESP-01 a gyártó legegyszerűbb és egyben legolcsóbb panele, mely ugyan némileg korlátozza az IC képességeinek kiaknázási lehetőségeit (bizonyos lábak, pl. a GPIO lábak némelyike nincs kivezetve, stb.), de az eszközzel való megismerkedéshez, a fejlesztés megkezdéséhez jó alapot ad.

A másik megismert modul az ESP-12, az ESP8266 IC modul változata, mely lényegében az IC összes lehetőségéhez hozzáférést biztosít, így ez a változat a fejlesztési folyamatot követően az éles hálózatban is használható. Konzulensem, Györke Péter készített ehhez a modulhoz egy tesztpanelt, melyet a következőkben röviden be is mutatok.

## ESP-12 tesztpanel

A tesztpanel célja, hogy olyan általános célú kiegészítéseket adjon a modul mellé, amely a fejlesztés és tesztelés során a legtöbb felmerülhető funkció használatára lehetőséget ad.

A fejlesztés során szükséges kommunikációs csatornát az ESP8266 és a fejlesztés platformja között USB-UART átalakító biztosítja, ennél fogva adja magát, hogy a tápfeszültséget is innen nyerjük. Ugyanakkor, míg a szabványos USB 5V-ot szolgáltat, az IC-nek a működéshez 3.3V-ra van szüksége. AZ ESP-01 esetén az USB-UART átalakító végezte a tápfeszültség transzformálását is, a tesztpanelnél már erre nincs szükség erre, mivel az már saját maga elvégzi ezt.



2. ábra Tesztpanel

Kivezetések tekintetében így értelemszerűen a táp, föld, illetve a kommunikációhoz szükséges UART (RXD és TXD) lábakhoz tudunk csatlakozni, valamint a GPIO0 lábhoz, amelynél követelmény, hogy amennyiben updatelni akarjuk a firmwaret (és csak akkor!), ez a láb földre legyen kötve. A panelen a megfelelő interfészek teszteléséhez egy I2C hőmérsékletmérő és egy analóg fényerősségmérő szenzor is el lett helyezve. Helyet kapott ezen kívül még néhány további kiegészítés is, mint például nyomógombok, egy bistabil relé és néhány LED is.

# Irodalomkutatás

A fejezet első felében bemutatok néhány piaci és nyílt forráskódú megvalósítást az otthonautomatizálási rendszerek témakörében. Ezt követően röviden ismertetem a saját rendszerem elkészítésénél használt technológiákat és külső könyvtárakat.

## Otthonautomatizálási rendszerek

Otthonautomatizálási rendszerek tekintetében a piacon mára számos üzleti megoldás jelent meg. Ezen rendszerek nagy része csak a gyártó saját eszközeivel működik együtt, de láthatjuk majd, hogy ez alól akadnak kivételek. Léteznek moduláris megoldások is, ezek nemcsak hogy már meglévő házba telepíthetőek, hanem a rendszerbe bármikor könnyen becsatlakoztathatunk egy új eszközt is. Az alfejezetben áttekintünk két nyílt forráskódú megoldást is, a rendszerek tekintetében az alapvető működésükre, funkcionalitására és az alkalmazott technológiákra koncentrálva.

### Control4

Az amerikai székhelyű Control4 a személyre szabott otthonautomatizálási és vezérlési megoldások egyik piacvezető szereplője

### IContrall

A magyar érdekeltségű IContrall az otthon teljes automatizálását egy az egyben a saját eszközeivel kívánja megoldani. A gyártó a legtöbb háztartási eszköz számára - mint például a riasztó vagy az árnyékoló - speciális vezérlőket kínál, de ezen felül vannak univerzális vezérlő moduljaik is. Az, hogy az IContrall külső eszközöket nem enged be a rendszerébe, az egyértelmű hátrány mellett előnyökkel is szolgál: a kommunikációs csatorna titkos volta miatt a rendszer kevésbé sebezhető külső támadásokkal szemben. A modulok telepíthetőek már meglévő lakásba is: a kommunikáció a vezeték nélküli hálózaton keresztül történik, ez lehetővé teszi az új eszközök gyors csatlakoztatását a már meglévő rendszerhez .

A felhasználói vezérlésre okostelefonos és táblagépes felület szolgál, a gyártó ígérete szerint ez könnyen kezelhető, logikus szerkezetű, így az új felhasználók is hamar beletanulhatnak a rendszer üzemeltetésébe.



3. ábra Az IContrall felhasználói interface tableten

### Amazon Echo

Némileg kilóg a sorból az Amazon Echo, hiszen nem, vagy legalábbis nem kizárólag a hagyományos értelemben vett otthonautomatizálási rendszer. Ugyanakkor technológiailag a Google Home-al versenyezve véleményem szerint azt az irányt mutatja, amit a jövő okosotthon rendszerei követni fognak, így mindenképp érdemes röviden áttekinteni.



4. ábra Amazon Echo Dot és Amazon Echo

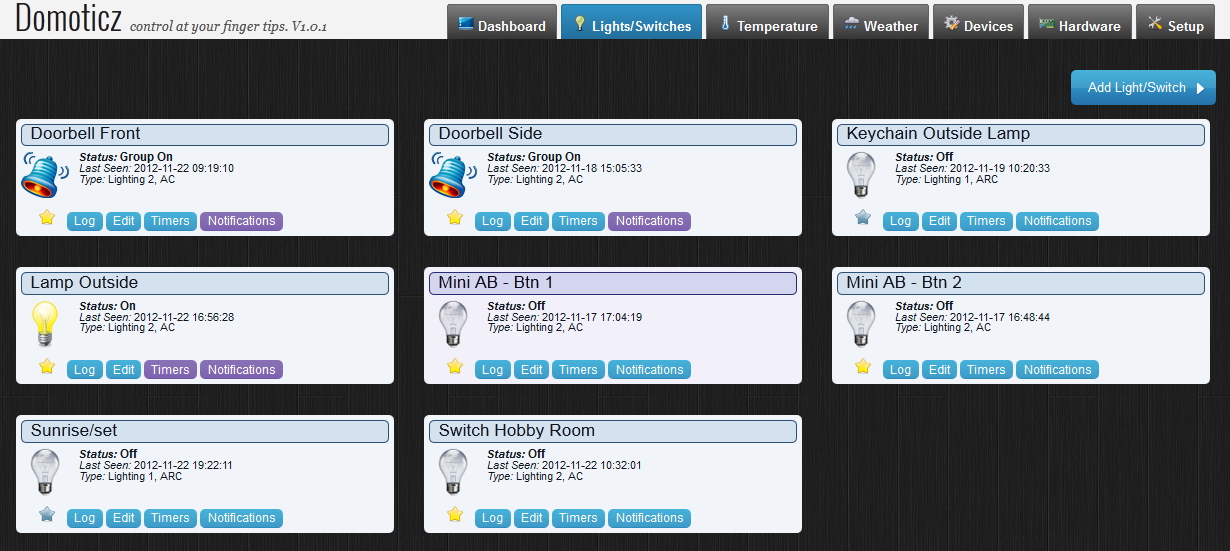
Az Amazon Echo legfőbb különlegessége a hangvezérlésben rejlik: az eszköz mai viszonylatban fejlett beszédfelismerő rendszere nem csak az összekötött okos eszközeink vezérlését teszi lehetővé, hanem készségesen válaszol az időjárásra vonatkozó kérdésünkre, sőt, egyszerűbb matematikai problémákat is meg tud oldani nekünk. Ehhez pedig csak kérdeznünk kell.

Ár tekintetében abszolút versenyképesnek érzem az Amazon Echo-t: a gyártó oldaláról rendelve 150$-ért már miénk lehet, de aki megelégszik a kisebb változattal, már 50$-ért beszerezheti. Az Amazon Echo önmagában még nem alkot egy okos otthon rendszert, de a rendszer magjaként képes működni, ha további eszközöket csatolunk hozzá.

### Domoticz

Az open source megvalósítású, C++ nyelven írt Domoticz többféle platformon is elérhető: telepíthető a három nagy (Windows, Linux, MacOS) asztali operációs rendszerre, de minden további nélkül futtatható például egy Raspberry PI-ról vagy a Synology gyártó NAS[[3]](#footnote-3) készülékeiről is. Az első, 2012-ben kiadott verzió óta folyamatosan fejlesztik, mind a hozzáadható eszközök tekintetében, mind az elérhető funkciók tekintetében. A felhasználók egy HTML5 webes interfészen keresztül férnek hozzá a rendszerhez, mely a saját, beágyazott webszerverén fut. A viselkedés beállítására többféle megoldást kínál a Domoticz: többek között LUA szkript vagy Blocky[[4]](#footnote-4) használatával is automatizálhatjuk az eseményekre történő reakciókat.

Az integrálható eszközök választéka széles, a nagy gyártók megoldásai (pl. Z-Wave eszközök, Phillips Hue) mellett számos más eszköz használata is lehetséges. A rendszer felhasználói értesítéseket is képes küldeni mobil eszközeinkre, legyenek azok Android, iOS vagy Windows Phone rendszerűek.



5. ábra A Domoticz webes konfigurációs felülete

## Alkalmazott technológiák és keretrendszerek

A fejezetben egyenként ismertetem a dolgozatban felhasznált technológiákat, terjedelmi okokból elsősorban a lényegesebb, szakmai döntést igénylőkre kitérve. A fejlesztés során az ESP8266 üzleti logikájának megírása során többszöri API váltás történt, indokolt így ezekről a platformokról is rövid áttekintést adni, kiemelve a tapasztalt előnyöket és hátrányokat. A szerver oldal teljes mértékben a LAMP szoftver csomagként is ismert megoldásgyűjteményre épít, mely betűszó a mögötte rejlő technológiákat takarja: (esetünkben Debian) Linux operációs rendszerre telepített Apache HTTP szerver, melyhez tartozik egy MySQL adatbázis, az oldalak megvalósítása pedig elsődlegesen PHP szkriptnyelven történik. Az alfejezet végén további technológiákról esik szó, mint a platformok közötti kommunikáció megkönnyítésére használt JSON formátum, a szenzoradatok grafikus megjelenítésére szolgáló HighStock API, illetve a felhasználó autentiákációs keretrendszerként használt Master Login System.

### JSON

A JSON (JavaScript Object Notation) egy szöveges adatcsere-formátum, mely eredetileg a JavaScript programozási nyelv részeként jelent meg 1999-ben. Mára a nyelvtől lényegében függetlenné vált, szinte minden jelentős programozási nyelvhez készült JSON generáló és feldolgozó programcsomag. A JSON-t számos helyen alkalmazzák az XML alternatívájaként, mivel amellett, hogy egyes feladatokra az ember számára sokkal áttekinthetőbb struktúrát kínál, egyszerűbb is, ami gyorsabb feldolgozást tesz lehetővé. Létezik sémaleíró nyelve is, amivel a JSON struktúrák egyszerűen validálhatóak.

A JSON gyakorlatilag két féle struktúrára épít:

* Név-érték párok gyűjteménye. Ez nyelvtől függően reprezentálhat például egy objektumot, struktúrát, hash táblát vagy asszociatív tömböt[[5]](#footnote-5)
* Értékek listájaként reprezentálja például a tömböt, vektort, listát

Példaként megnézhetünk a dolgozatban használt, egy szabályt leíró struktúrát:

**{**

"description"**:** "Ez a szabály felkapcsolja a ledet, ha a fényerősség adott érték alá csökken"**,**

"triggers"**:** **[**

**{**

"id"**:** 1**,**

"rule\_id"**:** 1**,**

"sensor\_id"**:** 12**,**

"operation"**:** "<"**,**

"value"**:** "400"

**}**

**],**

"actions"**:** **[**

**{**

"action\_id"**:** 1**,**

"actuatorid"**:** 1**,**

"rule\_id"**:** 1**,**

"value"**:** "ON"

**}**

**],**

"device\_id"**:** 1**,**

"name"**:** "lampa1"

**}**

A hozzá tartozó séma pedig:

**{**

"title"**:** "Szabály"**,**

"type"**:** "object"**,**

"properties"**:** **{**

"user\_id"**:** **{**

"type"**:** "integer"

**},**

"id"**:** **{**

"type"**:** "integer"

**},**

"description"**:** **{**

"type"**:** "string"

**},**

"triggers"**:** **{**

"type"**:** "array"**,**

"uniqueItems"**:** **true,**

"items"**:** **{**

"type"**:** "object"**,**

"properties"**:** **{**

"id"**:** **{**

"type"**:** "integer"

**},**

"rule\_id"**:** **{**

"type"**:** "integer"

**},**

"sensor\_id"**:** **{**

"type"**:** "integer"

**},**

"operation"**:** **{**

"type"**:** "string"

**},**

"value"**:** **{**

"type"**:** "string"

**}**

**}**

**}**

**},**

"actions"**:** **{**

"type"**:** "array"**,**

"uniqueItems"**:** **true,**

"items"**:** **{**

"type"**:** "object"**,**

"properties"**:** **{**

"action\_id"**:** **{**

"type"**:** "integer"

**},**

"actuatorid"**:** **{**

"type"**:** "integer"

**},**

"rule\_id"**:** **{**

"type"**:** "integer"

**},**

"value"**:** **{**

"type"**:** "string"

**}**

**}**

**}**

**}**

**}**

**}**

### HighStock

A HighCharts egy tisztán JavaScript nyelven írt könyvtár, melynek segítségével webes környezetben készíthetünk grafikonokat.

A programcsomag része az eredetileg tőzsdei adatsorok megjelenítésére készült HighStock, de rugalmasságánál fogva szenzoradatok megjelenítésére is alkalmas. Az oldalon részletes API, számos tutorial és demo diagram segíti a megértést és a könnyű integrálhatóságot. Az eredetileg meglehetősen drága API-nak nem kereskedelmi célú felhasználásra létezik ingyenes licensze is, ami lehetővé tette a dolgozatban történő felhasználást is.



6. ábra A Highstock dolgozatban is használt grafikonja

### Bootstrap

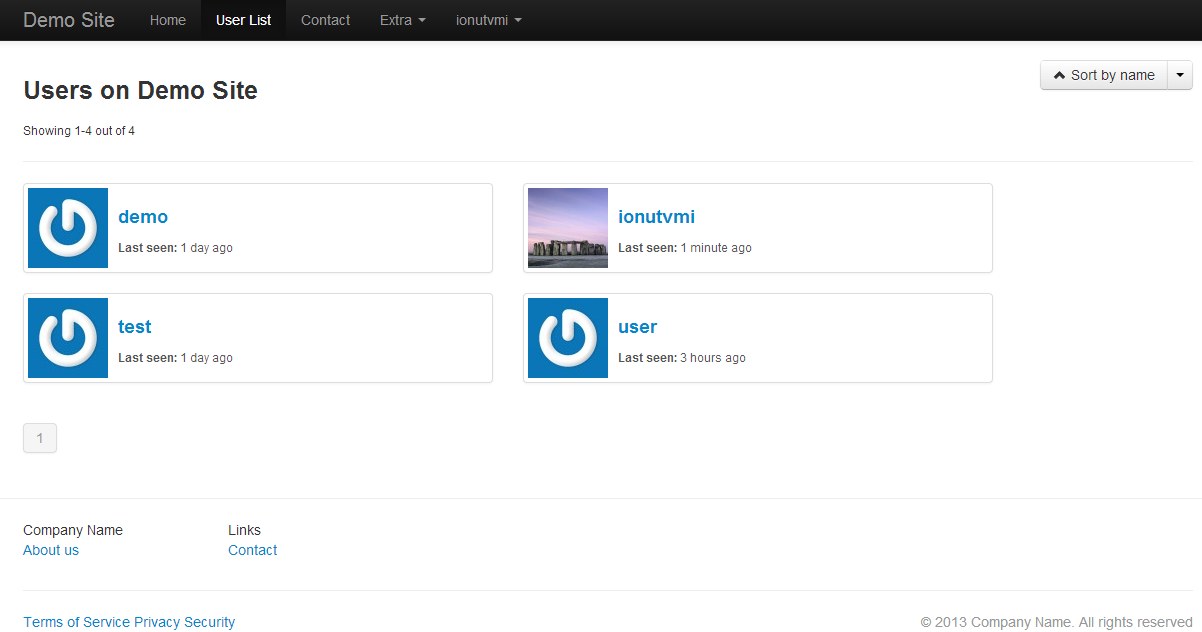
A Bootstrap manapság a világon legtöbbet használt CSS[[6]](#footnote-6) keretrendszer, melynek első verzióját a Twitter fejlesztői adták ki 2011-ben, de mára a reszponzív oldalfejlesztés egyik legfőbb eszközévé vált. Az open source megvalósítás, illetve hogy 2017-ben már a negyedik fő verzió megjelenésére számíthatunk, együttesen járulnak hozzá az egyre növekvő népszerűségéhez. Rendkívül sokféle, az oldalba beépíthető elemet tartalmaz, kezdve az egyszerű szövegformázási lehetőségektől egészen a táblázatokig vagy a tartalom oldalon belüli fülekbe szervezéséig.

A Bootstrap használata egyszerű, segítségével minimális erőfeszítéssel készíthetünk reszponzív oldalakat. Ez kiemelt jelentőséggel bír, hiszen abban a korban, amikor a weboldalakat az okostelefonok kijelzőjétől kezdve a PC monitorokig egyaránt kényelmesen szeretnénk használni, erre már nem plusz szolgáltatásként, hanem valós elvárásként tekinthetünk.

### Master Login System

A Master Login System egy bootstrap alapú user management keretrendszer, mely a szerver oldal alapjául szolgált. A php nyelven íródott rendszer a felhasználók kezelésére biztosít felületet, az ilyen rendszerektől elvárt alapvető igényeket kiszolgálva. A rendszer funkciói:

* Felhasználói alapműveletek (be -és kijelentkezés, regisztrálás, profil szerkesztése)
* Jogosultságkezelés (csoport alapú, admin, user, custom felhasználói csoportok)
* Admin panel (userek listázása, admin funkciók)
* Kapcsolattartási űrlapok



7. ábra Master Login System

## Beágyazott fejlesztőkörnyezetek

Az ESP8266 megjelenése (2014 második fele) óta komoly népszerűségre tett szert, így érdemes időt szánni arra, hogy megvizsgáljuk a fejlesztés platformja tekintetében a fejlesztők számára milyen alternatív fejlesztőrendszereket képes nyújtani. Ennek már csak azért is jelentősége van, mert a diplomaterv készítése során magam is többször váltottam platformot.

### AT-Command

Az IC-n gyárilag is feltöltött AT command nyelvet használó firmware használata indokolt lehet, ha az eszközt csak a wifi hálózathoz csatlakozáshoz miatt használjuk, de komplexebb alkalmazásokhoz (pl. szenzorhálózatban történő felhasználásnál) érdemes valami magasabb szintű nyelvet keresni a fejlesztéshez.

### NodeMCU

A fejlesztés kezdetén használt firmware a NodeMCU nevet viseli, mely egy C nyelven írt Lua interpreter. A választás mellett elsődlegesen az szólt, hogy az ESP-t használó közösség körében a korai szakaszban az egyik legnépszerűbb platform volt az ESP8266-on belül, így biztosítva a jól dokumentáltságot, a sok példakódot, valamint azt is, hogy a firmware-t a fejlesztők folyamatosan fejlesztik és karban is tartják. A NodeMCU kezdeti lendülete mára kissé erejét vesztette, de továbbra is a nagyobb támogatottságú környezetek közé tartozik. Egyszerűbb üzleti logikájú működés megvalósítására minden további nélkül ajánlott a használata. A rendszer fejlesztése során azonban nálam stabilitási gondok léptek fel: a NodeMCU-nál sokszor okozott gondot a memóriakorlátok betartása, egy egyszerű webes felület létrehozása is már komoly megszorításokat igényelt, ezek sokszor valószínűleg a firmware apróbb hibáiból eredő pazarló memóriahasználatra voltak visszavezethetőek.

### ESP SDK

A szakdolgozat készítésének kezdetén a firmware váltás mellett döntöttem, az egyik kipróbált új rendszer az ESP8266-ot gyártó Espressif által fejlesztett, így lényegében hivatalosnak tekinthető SDK lett. Ennek két alapvető változata az OS nélküli, és az RTOS[[7]](#footnote-7) alapú megvalósítás, a fő különbség, hogy míg előbbiben az eseménykezelés elsősorban timer és call-back hívások segítségével valósul meg, utóbbi a FreeRTOS[[8]](#footnote-8) ütemezőjére épít. Én a feladat jellegéhez való jobb illeszkedés miatt a NONOS verziót használtam. A platform mellett egyértelmű előny volt a C nyelv, magamhoz közelebbinek éreztem és felhasználható függvénykönyvtárak széles választékának köszönhetően több funkciót is jelentősen kisebb munkával sikerült újraimplementálnom. Sajnos a működés implementálása során többször rá kellett jönnöm, hogy a kiadott SDK bár folyamatos fejlesztés alatt van, sajnos még mindig sok hibát tartalmaz, ami nehézkessé tette az eszközben rejlő lehetőségek kiaknázását. Például bár hivatalosan az API tartalmazta az SSDP[[9]](#footnote-9) implementációját, a valóságban ezt nem sikerült életre keltenem, az interneten az SDK fórumjait böngészve pedig azt tapasztaltam, hogy ez egy javításra váró hiba.

### Arduino

Az Arduino egy az Atmel AVR mikrovezérlő családra épülő, szintén C nyelvű szabad szoftveres elektronikai fejlesztőplatform, mely alapvetően a gyártó saját eszközeihez készült és „easy to use” szemlélete révén hamar elterjedt főleg a hobbi felhasználók körében. Ez a folyamat azzal járt, hogy hamarosan más több más eszközre is megjelent a platform implementációja, a kereskedelmi forgalomban való megjelenését követően erre az ESP8266 esetében sem kellett sokat várni. Előnyei közé sorolható az egyszerű használhatósága, széles körű elterjedtsége, a funkciók jól dokumentáltsága, rengeteg online példakód. Ezen felül az ESP SDK-val tapasztaltakkal szemben az Arduino esetében azt tapasztaltam, hogy a leírt funkciók valóban működnek is, használat közben sokkal megbízhatóbbnak éreztem, így az ESP modulokon végzett fejlesztést teljes egészében erre a platformra építettem fel.

### Egyéb fejlesztőrendszerek

Fentieken felül természetesen léteznek más alternatívák is, például a Python fejlesztési felületet nyújtó MicroPython keretrendszernek is van implementációja ESP8266-ra, de ezekkel nem foglalkoztam részletesebben a dolgozat elkészítése során.

# Tervezés

Az előző fejezetben az üzleti és nyílt forráskódú megoldások megismerése során egyaránt azt láttam, hogy az újabb rendszerek már szinte kivétel nélkül előnyben részesítik a vezeték nélküli megoldásokat, hiszen ez biztosítja a könnyebb beépíthetőséget (már meglévő lakásba is), valamint a rendszer elemeinek cserélhetőségét és bővíthetőségét is. Kritikus tényező azonban a rendszerek ára, mely sok potenciális felhasználó számára jelenleg elérhetetlenné teszi az okosotthon rendszerek használatát. A tanulságok figyelembevételével ez a fejezet a korábban ismertetett technológiákra alapozva egy teljes okosotthon rendszer tervezési folyamatát mutatja meg, melyet aztán a későbbiekben meg is valósítok.

## A feladat célja

A feladatban egy otthonautomatizálási projekt részeként szeretnék a kommunikáció támogatására létrehozni egy hálózatot, melynek csomópontjaiban az internetre kapcsolódni képes eszközök egy központi szerverrel kommunikálnának.

A csomópontokban lévő internetre közvetetten kapcsolódó eszközök alapvetően két funkciót valósítanak meg. Egyrészt képesnek kell lennie arra, hogy a környezetükben található szenzoroktól előre meghatározott rendszerességgel adatokat gyűjtsenek és továbbítsák a szerver számára, amely aztán képes feldolgozni és megjeleníteni ezeket. Az eszközök másik felelőssége, hogy a szerverrel kommunikációs csatornát legyenek képesek felépíteni, és a központtól kapott utasításokat fogadva beavatkozzanak a környezetükben lévő eszközök működésébe.

Ugyancsak megvalósítandó feladat a központi szerver implementációja. Ennek több, a feladat szempontjából lényeges funkciója is van. Egyrészt képesnek kell lennie a szenzorok számára valamilyen felületet biztosítani, amin keresztül azok feltölthetik a mért adataikat. Ezeket természetesen a rendszer más nélkülözhetetlen adataival együtt tárolni is szeretnénk, így szükséges, hogy valamilyen adatbázisunk is legyen. A központi szervernek biztosítania kell valamilyen, célszerűen webes felületet az ügyfelek számára. Természetesen egy weboldal tetszőleges számú otthont kiszolgálhat, a felhasználói jogosultságkezelésnek köszönhetően az egyes rendszerek kezelőfelülete el van választva egymástól, a felhasználók csak a saját rendszereikhez kapnak hozzáférést.

Már a tervezési szakaszban érdemes foglalkozni az esetlegesen felmerülő problémákkal, amelyek a rendszer működését befolyásolják. Az interneten keresztül adatokat továbbító eszközök esetében például fontos kérdés az, hogy mi történik akkor, ha az internetkapcsolatunk valamilyen oknál fogva rövidebb, vagy akár hosszabb időre megszakad. Egy egyszerű architektúrában, ahol a szenzorok és beavatkozók a központi szerverrel kapcsolatot tartva csak üzeneteket küldenek és fogadnak, ez a probléma megszűnéséig a rendszer teljes időleges működésképtelenségét okozná, hiszen egyrészt a szenzoradatok nem érkeznek meg a szerverhez, másrészt a beavatkozónak küldött utasítások sem érnének célba. A probléma áthidalására érdemes olyan megoldást találni, ahol az internetkapcsolat megszakadása esetén is képesek a lakáson belüli eszközök működni, legfeljebb időlegesen a szerverről történő konfigurálási lehetőségeinket veszítjük el.

## ESP

### Működési módok

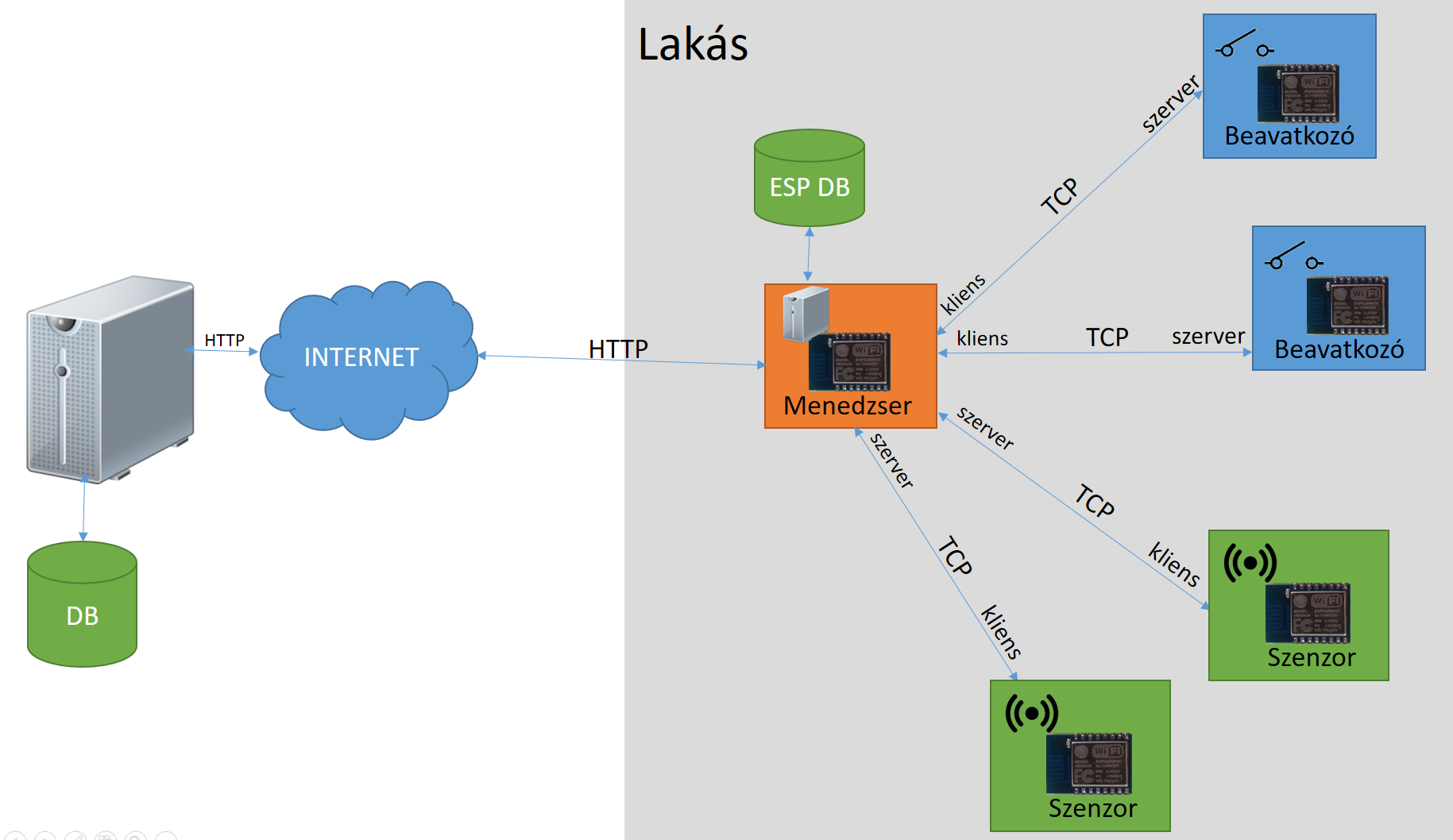
A fentiekben vázlatosan ismertetett feladatok megvalósításához több különböző működési módot implementálunk.

A három fő, különálló feladatkör:

* Szenzor
* Beavatkozó
* Menedzser

Ezeket a közvetkezőkben egyenként, részletesebben is ismertetem.

## A rendszer topológiája

A rendszer alapvető elemeit az alábbi ábrán követhetjük figyelemmel: 

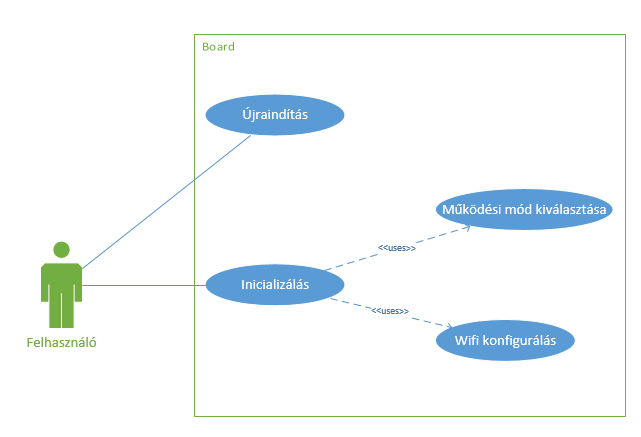
7. ábra A rendszer topológiája

A központi szerver az ábra bal oldalán látható. A szürke négyzet egy, fizikailag és logikailag is az egy helyen lévő egységeket takar, nevezzük lakásnak. A lakás alapvetően három féle elemet tartalmaz, szenzorokat, beavatkozókat, illetve kötelező elemként mindenhova tartozik egy helyi menedzser is. Az természetesen nincs kizárva, hogy utóbbinak fizikailag valamelyik szenzor vagy beavatkozó eszköz lássa el a feladatát, ahogy egy modul is lehet beavatkozó és szenzor is egyszerre, de a logikai szétválasztás miatt érdemes külön is gondolni rájuk. A központi szerverrel a helyi, lokális menedzserünk kommunikál, a többi modul rajta keresztül tartja a kapcsolatot, azaz a szenzorok ide küldik az adataikat, a beavatkozókat pedig közvetlen módon a lokális menedzser utasítja, a saját központi szerverről frissített adatbázisa alapján.

## ESP eszközök működése

### Felhasználói interakciók

A rendszer tervezése során figyelemmel kell lenni arra, hogy a rendszer legfeljebb átlagos informatikai ismeretekkel rendelkező felhasználók számára is minél könnyebben kezelhető legyen. Ennek jegyében a felhasználó rendszerbe való beavatkozási lehetőségeit az ESP boardon igyekeztem minimalizálni. A 8. ábrán láthatjuk, hogy a felhasználók feladata lényegében kimerül egy egyszeri inicializálásban, amit majd egy eszköz által szolgáltatott webes felületen tehetnek meg. Univerzális eszközök esetén a felhasználó ugyanezen a felületen a működési módot is kiválaszthatja, de ez a lépés előre meghatározott feladattal rendelkező ESP-k használata esetén kihagyható. A boardon található gomb segítségével probléma esetén bármikor újra lehet indítani, áramkimaradás (vagy lemerült elem) esetén a probléma megoldását követően ez automatikusan megtörténik. Minden más műveletet a felhasználó a szerver később bemutatott webes felületén tud végrehajtani.



8. ábra Eszköz oldali felhasználói use casek

## Hálózati hozzáférés inicializálása

Bármilyen ESP eszköz esetén a kommunikáció egyetlen csatornája a vezeték nélküli hálózat, így az eszköz indulását követően első feladatként szükséges, hogy a modul felcsatlakozzon valamilyen előre meghatározott vezeték nélküli hálózatra, ez például lehet a lakás otthoni WiFi hálózata.

Ennek jegyében az eszköz első indításának úgy kell megtörténnie, hogy a felhasználó megadhassa a Wifi-re történő csatlakozásához szükséges SSID és password adatokat. Ennek biztosítására az első alkalommal Acces Pointként induló eszköz által szolgáltatott weboldal fog szolgálni. Ha ez megtörtént, a modul automatikusan station módba vált, majd a saját működési módjának megfelelően már bármikor tud csatlakozni a hálózathoz, megnyitva az utat a többi eszközzel és a szerverrel történő kommunikációhoz.

Természetesen sokféle okból előfordulhat, hogy a megadott vezeték nélküli csatlakozási adatokat módosítani kell, így a felhasználó számára bármikor lehetővé kell tenni, hogy újrainicializálja az eszközt.

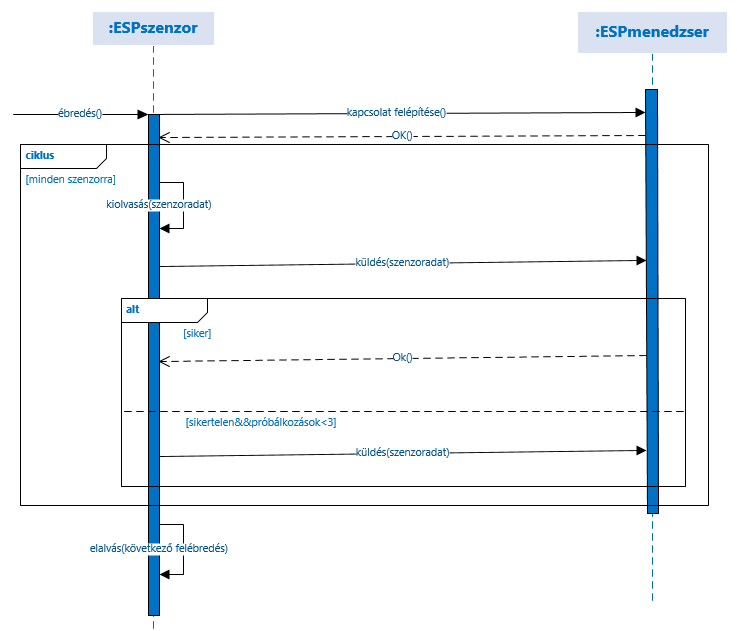
## Működési módok

A fentiekből láthatjuk, hogy az ESP eszközök több, felelősségét tekintve jól elkülöníthető funkciókat biztosítanak. A szenzor, beavatkozó és menedzser eszközöket a lakás különböző pontjain lehet elhelyezni igény szerint. Az elkülönített működési módok nem jelentik azt, hogy egy eszköz csak egy feladatot képes ellátni: a lakásnak lehetnek olyan pontjai, ahol egy helyen van szükségünk szenzor és beavatkozó eszközökre is. Ilyenkor cél lehet, hogy ne kelljen két modult telepítenünk, de figyelembe kell venni, hogy a működések és megszorításaik eltérhetnek, ezekre figyelni kell az implementációnál. Például az olyan beavatkozó eszköz, mely szenzor kiolvasási és továbbítási műveleteket végez, nem aludhat el a szenzoradatok kiolvasását követően, hiszen a beavatkozó feladatai folyamatos kapcsolatot feltételeznek a menedzser eszközzel. Helyette a szenzor életciklus egyszerűen egy ütemezett feladat, ami bizonyos időközönként lefut.

### Szenzor

A szenzorhálózat csomópontjaiban az eszköz fő feladata, hogy előre meghatározott időközönként érzékelőivel méréseket végezzen, majd a kiolvasott adatokat továbbítsa a menedzser eszköznek. Ebben az üzemmódban az eszköz az idő legnagyobb részében valójában nem csinál semmit, hisz két mérés között az elküldést követően a modulnak semmilyen feladata sincs. Emiatt a fogyasztás optimalizálása érdekében ebben az időszakban célszerű alvó módban üzemeltetni az eszközt megmondva, hogy mennyi idő elteltével ébredjen fel újra az új mérés elvégzéséhez. Ha ilyen módon készítjük el a programunkat, akkor optimális esetben az eszköz akár elemről is működtethetővé válik (hosszú ideig, hónapokig, vagy akár évekig működhet elemcsere nélkül), ami kihelyezett szenzorhálózati csomópontoknál egy fontos szempont lehet.

A fejlesztés eredményeként a beállított szenzor eszköz egy egyszerűsített ciklusa a következőképp vázolható.



8. ábra Egy szenzor életciklusa

A folyamat mindig úgy kezdődik, hogy a szenzor eszköz elindul, ami legtöbb esetben a deep sleep állapotból való felébredést jelenti, de a már beállított eszköz tápfeszültséget kapva (például elemcserénél) szintén ezt a folyamatot indítja el.

Kliensként a TCP hálózaton keresztül kapcsolódik a menedzser eszközhöz, a kapcsolat felépítésének sikeressége esetén pedig végigmegy a saját, regisztrált szenzorain, egyenként elvégzi az értékek kiolvasását és rögtön továbbítja is a menedzser eszközöknek. Ha az adatok továbbítása nem sikerül, akkor meghatározott számban újra próbálkozik a művelettel. Az összes szenzoradat átküldését követően az eszköz visszatér alvó üzemmódba, a következő mérési ciklus kezdetéig nincs feladata.

A beküldésnél a szenzornak az alábbiakat kötelezően meg kell adnia:

|  |  |
| --- | --- |
| **Változó neve** | **Változó jelentése** |
| deviceName | a szerver által történő azonosításra szolgál a csomópontban szereplő eszköz neve, ez nyílván egy egyedi azonosító, gyakorlatilag a szenzor eszköz MAC címe |
| devicePassword | az azonosításhoz szükséges, a deviceNamehez rendelt jelszó, amit |
| dataType | a mért szenzorérték típusát jelöli |
| data | az elküldeni kívánt adat |

### Menedzser

A menedzser működése egy adott lakás környezetében kritikus, hisz feladata többrétű. Egyrészről rendszeresen figyelnie kell, hogy új eszközök próbálnak-e felcsatlakozni a hálózatra, ha igen, akkor fel kell vennie velük a kapcsolatot (persze csak akkor, ha ezek az eszközök megfelelő módon hitelesítették magukat). A már csatlakozott szenzorok számára biztosítania kell, hogy azok elküldhessék neki a mért szenzoradatokat, illetve ezek alapján és a saját lokális szabályadatbázisa alapján utasítania kell a beavatkozókat a cselekvésre. Ahogy az ábrán is láthattuk, további feladata még a központi szerver irányába történő kommunikáció: egyrészt rendszeresen továbbítania kell a szenzorok által beküldött adatokat, valamint frissen kell tartania a saját környezetéhez tartozó szabályadatbázisát is és ha egy szabály aktívvá válik, akkor értesíti a beavatkozót.

### Beavatkozó

Beavatkozó üzemmódban az eszköz célja, hogy a környezetének (a csomópont) működésében változásokat eszközöljön a szervertől kapott utasítások alapján. Ennek biztosítására értelemszerűen folyamatos kapcsolatot kell fenntartani a menedzserrel, figyelve a kommunikációs csatornát.

A beavatkozó működés folyamatának lépései:

1. TCP kapcsolat felépítése a szerverrel

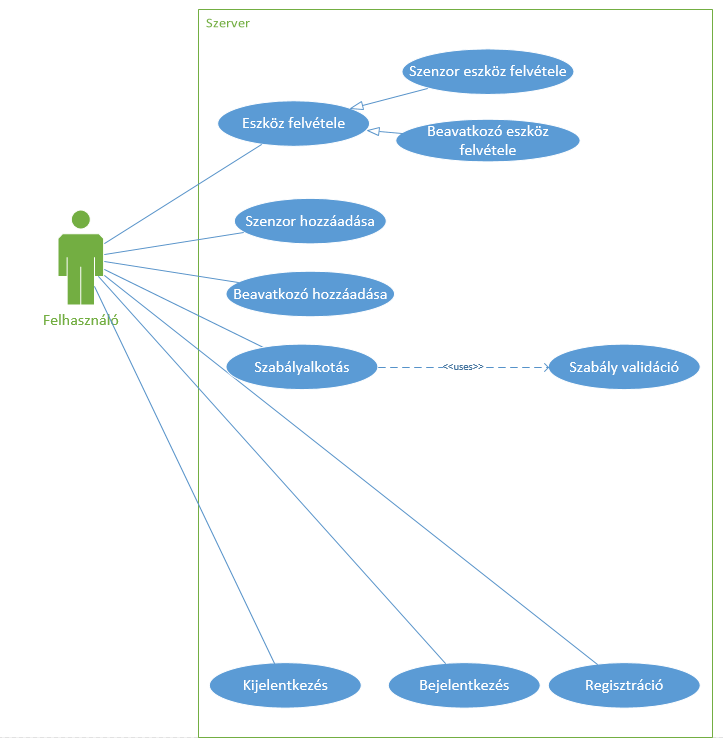
* UDP broadcast segítségével közli a környezetével, ki is ő, majd várja a lokális szerver csatlakozási igényét

1. Kommunikációs csatorna figyelése
   * Ha üzenet érkezik, annak tartalmától függően cselekszik

A fenti módban például egy „R1 ON” üzenet hatására az eszköz a megfelelő GPIO lábának 1-be tételével az R1-es relét átkapcsolhatja másik állásba. Másik lehetőség például, hogy a PWM interfészén keresztül végzi egy villanymotor vezérlését a szervertől kapott utasítások függvényében.

## Felhasználói interakciók (use case)

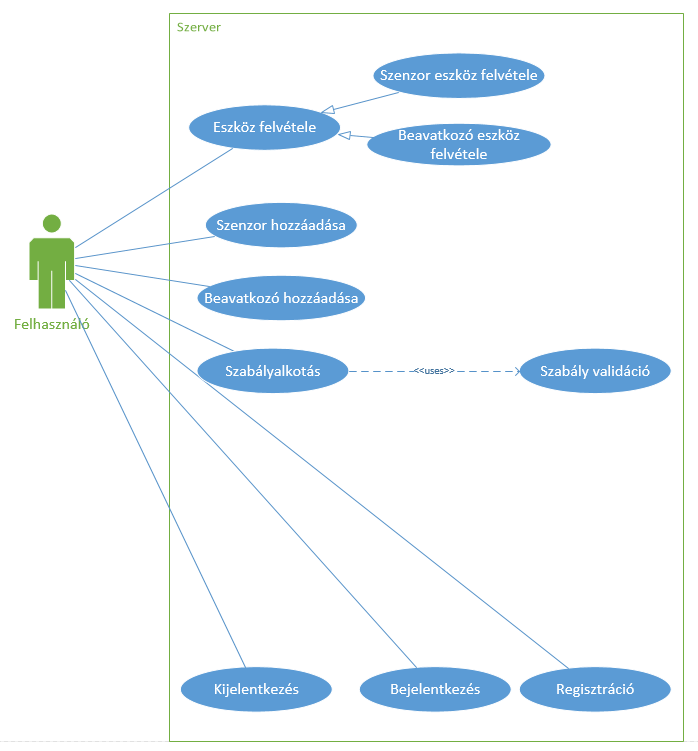
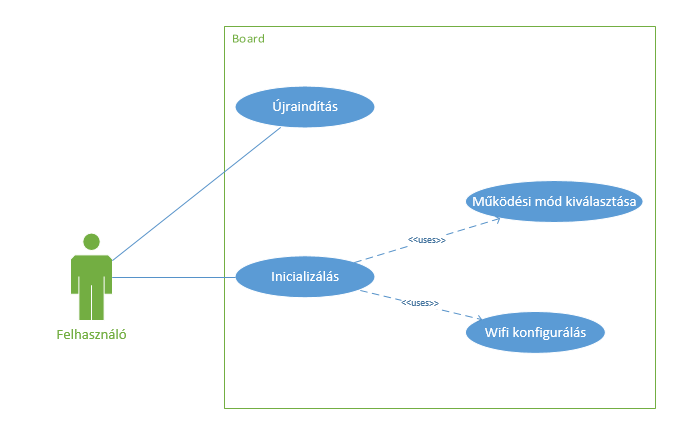
A rendszer topol



## Sleep

## Titkosítás

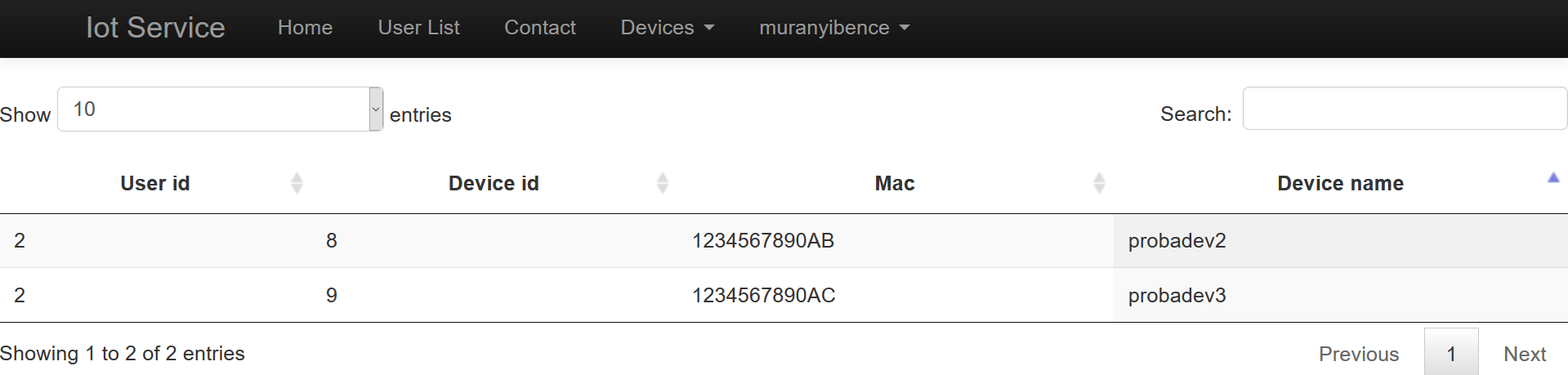
### Eszköz



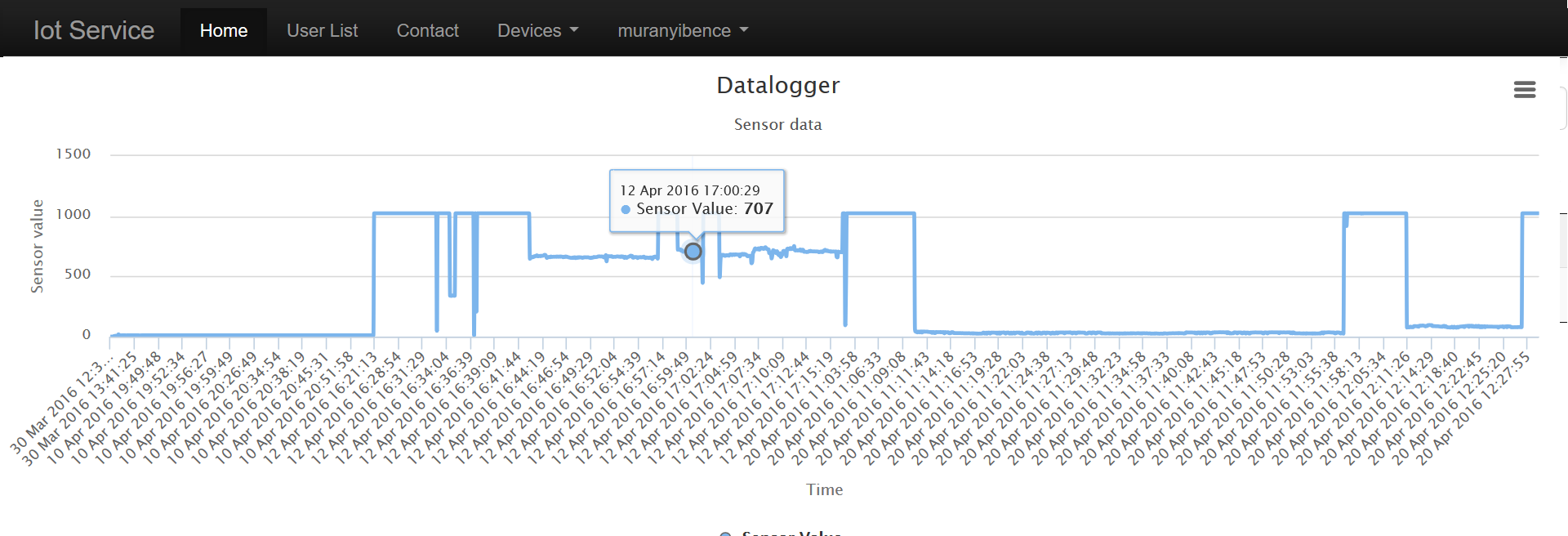
## A központi szerver

A központi szerver működése az egész rendszer számára kritikus, hiszen rajta keresztül vesszük fel az eszközeinket és szabályainkat. Ennek ellenére a fenti topológiában a feladatköre egyszerűsödik, hisz lényegében csak a lokális szerverekkel kell kommunikálnia, azt is csak a lokális szerver által kezdeményezett HTTP lekérdezéseken keresztül. Ennek megfelelően alapvetően két fő feladatkört tudhatunk a központi szervernek.

Egyrészt felületet kell nyújtania ahhoz, hogy a felhasználók az eszközeiket regisztrálhassák, szenzorokat, beavatkozókat hozhassanak létre, valamint ehhez szabályokat készítsenek. Ezekhez megfelelő, lehetőség szerint minél egyszerűbb űrlapokat kell létrehoznunk, amelyeken a fenti műveletek elvégezhetőek és követhetőek. A lenti ábrán látunk például egy oldalt, ahol a felhasználó megtekintheti a saját, beregisztrált eszközeit.



Fontos az is, hogy biztosíthassuk a felhasználók számára, hogy a saját hálózatukon belül a mért adatokat megjeleníthessék. A központi szerver erre is lehetőséget nyújt, a Highcharts API segítségével minden szenzor által beküldött érték grafikonos formában is megtekinthető.



# A projekt állapota

A Diplomaterv I. tárgy keretén belül az új fejlesztőrendszer megismerésével és kipróbálásával töltöttem, miközben elkezdtem a saját, webes szerver megoldásom implementálását.

Utóbbi már használható, a legtöbb, tőle kívánt funkció működőképes. A szerveren képesek vagyunk a szükséges adminisztrációs feladatok elvégzésére, mint amilyen például az új eszközök, vagy azokhoz tartozó szabályok felvétele a rendszerbe. Működik továbbá a vizualizáció is, a szenzoradatokat a szerver felületén grafikus formában is megtekinthetjük. Az eddigiek nem jelentik azt, hogy a központi szerver fejlesztése befejezettnek tekinthető: a felület sok ponton esetlen és nem túl felhasználóbarát, a kényelmesebbé tételre még szeretnék időt fordítani a jövőben.

Az eszközök működési módjaik közül a szenzor áll a legközelebb a kész változathoz: ez funkcionálisát nézve közel teljesnek nevezhető, néhány apróbb javításon, illetve a hosszú tesztelést követő esetlegesen felmerülő feladatokon kívül tulajdonképpen komolyabb munka már remélhetőleg nem adódik vele.

A beavatkozók és lokális szerver működésének implementációja egyelőre készülő fázisban van: eddig elsősorban a rendszer elemei közötti kommunikáció megvalósításán dolgoztam, továbbá olyan megvalósítandó feladatokon, mint a lokális szerver hitelesítése a központi szerver felé.

## Kitekintés

A jövőre nézve az elsődleges feladat azon részfeladatok befejezése, mely még nem, vagy nem teljes mértékben készült el. Ez elsősorban a menedzser és a beavatkozók megvalósítását és az egész rendszer alapos, több eszközre kiterjedő tesztelését jelenti.

A szenzor üzemmódnál az időszakos működési igény miatt érdekes, és lényeges kérdés az, hogy megoldható-e (gazdaságosan és ésszerű karbantartási igénnyel) a csomópont elemmel való működtetése. Ennek kiderítéséhez szükségessé válik majd az eszköz fogyasztásának mérése (mind alvó, mind aktív üzemmódban), és az eredmények kiértékelése.

Ha a szoftver fejlesztési folyamatának a végére érünk, a következő lépés az lehet, hogy a már éles használathoz specializált hardvert tervezzünk az eszközhöz, mely(ek) már a valódi, nem tesztelési célú csomópontokban is működhetnek.

Távlatibb, de nem elhanyagolható lehetőség a szenzorhálózat csomópontjaiban heterogén hálózatok összekapcsolása, így esetleg nem csak IoT, hanem pl. Bluetooth vagy rádiófrekvenciás eszközöket is bevonhatnánk a rendszer működésébe. A lehetőségek vizsgálata önmagában is egy elég jelentős méretű feladat lehet a jövőben.

Az IoT projekt témájánál maradva ugyancsak érdekes feladat lehetne valamilyen okostelefonos alkalmazást tervezni a rendszerhez, amelyben további beavatkozásokat is végezhetnénk.

# Irodalomjegyzék

**Nincsenek források az aktuális dokumentumban.**

1. Global Positioning System, műhold alapú navigációs rendszer, a szolgáltatás a világ minden pontjáról a nap 24 órájában ingyenesen elérhető. [↑](#footnote-ref-1)
2. Az ESP8266 eszközök képesek deep sleep üzemmódra. Látható, hogy ha nincs igényünk az eszköz folyamatos működésére (mint például a periodikus mérési üzemmódban), akkor a fogyasztás drasztikusan csökkenthető. A dolgozat a későbbiekben még foglalkozik ezzel a funkcióval. [↑](#footnote-ref-2)
3. Network Attached Storage, azaz a hálózatra csatolt tároló. A NAS egy fájl szintű adattároló eszköz, amely a számítógépes hálózathoz csatlakoztatva lehetővé teszi távoli felhasználók között adatok menedzselését, hozzáférési jogosultságok kezelését. Internetkapcsolat segítségével bárhonnan lehet rá csatlakozni. [↑](#footnote-ref-3)
4. A Blocky egy vizuális fejlesztőeszköz, melyet a Google készített. Használói vezérlési szerkezeteket reprezentáló blokkok egymáshoz illesztésével tudnak algoritmusokat, és végső soron programokat írni. [↑](#footnote-ref-4)
5. Az asszociatív tömb (angolul map vagy dictionary) adatszerkezet kulcs-érték párokat tartalmaz, ahol egy kulcshoz pontosan egy érték tartozik. [↑](#footnote-ref-5)
6. Cascading Style Sheets, a számítástechnikában egy [stílusleíró nyelv](https://hu.wikipedia.org/w/index.php?title=St%C3%ADlusle%C3%ADr%C3%B3_nyelv&action=edit&redlink=1), mely a [HTML](https://hu.wikipedia.org/wiki/HTML) vagy [XHTML](https://hu.wikipedia.org/wiki/XHTML) típusú strukturált dokumentumok megjelenését írja le. [↑](#footnote-ref-6)
7. Real Time Operation System, olyan operációs rendszer architektúra, ahol az egyes eseménykezelőknek meghatározott időn belül kötelezően le kell futnia. [↑](#footnote-ref-7)
8. A FreeRTOS egy népszerű RTOS kernel beágyazott eszközökre írva [↑](#footnote-ref-8)
9. Simple Service Discovery Protocoll, egy hálózati protokoll hálózati szolgáltatások felderítésére és jelenlétének közzétételére [↑](#footnote-ref-9)