# 

Budapesti Műszaki és Gazdaságtudományi Egyetem

Méréstechnika és Információs Rendszerek Tanszék

**IoT alapú otthonautomatizálás megvalósítása ESP8266 használatával**

Szakdolgozat

**Murányi Bence**

II. évf mérnökinformatikus szakos hallgató

MSc Intelligens rendszerek szakirány/ágazat

Konzulens:

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

# Bevezetés

## Az IoT-ről

Az IoT (Internet of Things), mint fogalom, a 21. század első évtizedének végén kezdett el megjelenni, amikor is a végfelhasználót tekintve az internetre csatlakozó tárgyak mennyisége lassan elkezdte meghaladni a csatlakozó személyek 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 a dolgok internete 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 nagyon aktuális.

## Intelligens otthon

Az intelligens otthon kifejezés a lakás automatizáltságát takarja, maga a fogalom pontosan talán nem is definiálható, hiszen rengeteg, önmagukban véve opcionális funkció tartozhat alá. Az otthonunkban megtalálható eszközös 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 bizonyos mértékben automatizált eszközeinket egy közös rendszer alá rendeljük, és ezen keresztül egységesen kezelhessük őket, határozhassuk meg a beállításaikat. 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 munkából hazaindulás előtt (ha ez szokatlan időpontban történik, egyéb esetben ez akár automatikusan is megtörténhet) jelezzük az otthonunknak, hogy készüljön fel a hazaérkezésünkre, így otthon már nem a hideg lakás fogad minket. A lakásban, ha szükség van rá (például ha egy filmet akarunk megnézni a tévénken), automatikus módon leereszthetőek a redőnyök. A hűtőszekrény automatikusan érzékelheti, 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.

# Irodalomkutatás

## Otthonautomatizálási rendszerek

Otthonoautomatizálási rendszerek tekintetében a piacon mára számos üzleti megoldás jelent meg. Ezen rendszerek többsége csak a gyártó saját eszközeivel működik együtt, de láthatjuk majd, hogy ez alól akadnak kivételek. A fejezetben áttekintünk néhány open source megoldást is, a rendszerek tekintetében az alapvető működésükre, funkcinonalitására és az alkalmazott technológiákra koncentrálva.

### Crestron

## Alkalmazott technológiák

### 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 adattárolás esetén emberi szemmel sokkal áttekinthetőbb struktúrát kínál, valamint egyszerűbb is, ami gyorsabb feldolgozást tesz lehetővé.

A JSON gyakorlatilag két féle struktúrára épít:

* Név-érték párok gyűjteménye. Ez nyelvtől függően reprezentálhat egy objektumot, struktúrát, hash táblát vagy asszociatív tömböt[[1]](#footnote-1)
* Értékek rendezett listájaként reprezentálja például a tömböt, vektort, listát

Példa JSON struktúrára:

{

"id": 1,

"name": "A green door",

"price": 12.5,

"tags": [

"home",

"green"

]

}

### 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 eredetileg meglehetősen drága API-nak nem kereskedelmi célú felhasználásra létezik ingyenes változata is, ami lehetővé tette a dolgozatban történő felhasználást.

### MySQL

A MySQL napjaink egyik legnépszerűbb adatbázis kezelő rendszere. Ennek egyik oka lehet, hogy a teljesen [nyílt forráskódú](https://hu.wikipedia.org/wiki/Ny%C3%ADlt_forr%C3%A1sk%C3%B3d) [LAMP](https://hu.wikipedia.org/wiki/LAMP_%28szoftvercsomag%29) ([Linux](https://hu.wikipedia.org/wiki/Linux)–[Apache](https://hu.wikipedia.org/wiki/Apache_HTTP_Server)–MySQL–[PHP](https://hu.wikipedia.org/wiki/PHP)) összeállítás részeként költséghatékony és egyszerűen beállítható megoldást ad [dinamikus](https://hu.wikipedia.org/wiki/Dinamikus_weboldal) webhelyek szolgáltatására.

Egyedi illesztőfelületekkel az adatbázis-kezelő elérhető C, C++ , C#, Delphi, Eiffel, Smalltalk, Java, Lisp, Perl, PHP, Python, Ruby és Tcl programozási nyelvvel. Egy *MyODBC* nevű ODBC interfész további, ODBC-t kezelő nyelvek számára is hozzáférhetővé teszi az adatbázis-kezelőt. A MySQL számára az ANSI C a natív nyelv.

## ESP8266

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

* 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

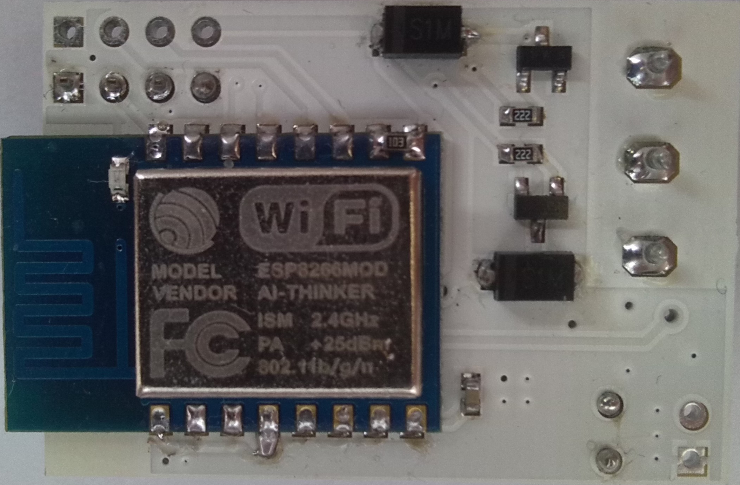
További pozitív tulajdonsága az ára: a pontos típustól függően már 3$ körüli áron hozzájuthatunk, ami azt tekintve, hogy ezt megelőzően egy hasonló igényeket kiszolgálni képes Wifi modul nagyságrendileg 25-40$ körül volt kapható, rendkívül kedvező. Megemlítendő, hogy a saját processzornak köszönhetően lehetővé válik a standalone működés. A fenti tulajdonságok (saját számítókapacitás, kivezetések, kis méret, stb.) nagyon 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 foglalkoztunk. 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 nyákra építhető 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 jobb oldalon is látható 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 port biztosítja, ennél fogva adja magát, hogy a tápfeszültséget is innen nyerjük. Ugyanakkor, míg a szabványos USB 5V-ot ad le, 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.

Kivezetések tekintetében így értelemszerűen a táp, föld, illetve a kommunikációhoz szükséges RXD és TXD lábakhoz tudunk csatlakozni, valamint a GPIO0 lábhoz, amelynél követelmény, hogy amennyiben updatelni akarjuk a firmwaret (és csak akkor!), 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.

## 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 a modul milyen alternatívákat képes nyújtani. Ennek már csak azért is jelentősége van, mert a szakdolgozat készítése során magam is többször váltottam platformot

### AT-Command

Az IC-n gyárilag is helyet kapott AT command nyelvet használó firmware használata indokolt lehet, ha az eszközt csak a Wifi hálózatban betöltött szerepe miatt használjuk, de komplexebb alkalmazásokhoz (pl. szenzorhálózatban történő felhasználásnál) érdemes valami magasabb szintű nyelvet keresni a fejlesztéshez.

### NodeMCU

A fejlesztés kezdetén használt platform a NodeMCU nevet viseli, mely egy C nyelven írt lua interpreter. Előnye az open source megvalósítás, illetve hogy a community 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 firmwaret a fejlesztők egyelőre folyamatosan fejlesztik és karbantartják. A NodeMCU kezdeti lendülete mára kissé erejét vesztette, de továbbra is a komolyabb támogatottságú környezetek közé tartozik.

### ESP SDK

A szakdolgozat készítésének kezdetén a firmware váltás mellett döntöttem, az 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 alapú megvalósítás, a feladatban én az elsőt használom. A platformváltás több lényeges szempontra vezethető vissza. Az egyik legfőbb szempontot stabilitási indokok nyújtják: a NodeMCU-nál sokszor okozott gondot a memóriakorlátok betartása, egy egyszerű webes felület létrehozása is már komoly megszorításokat igényelt, ezek sokszor valószínűleg a firmware hibáira voltak visszavezethetőek. Szempont volt még, hogy a C nyelvet egyébként is magamhoz közelebbinek éreztem, magasabb kifejezőerejének köszönhetően több funkciót is jóval kisebb munkával sikerült újraimplementálnom.

### Arduino

Az Arduino egy az Atmel AVR mikrovezérlő családra épülő, szabad szoftveres elektronikai fejlesztőplatform, arra tervezve, hogy a különböző projektekben az elektronikus eszközök könnyebben hozzáférhetőek, kezelhetőek legyenek. A keretrendszeren belül használható nyelv szintén a mikrokontrolleres környezetekben egyébként is népszerű C.

A fentieken felül természetesen léteznek más alternatívák is, például a python fejlesztési felületet nyújtó MicroPython keretrendszernek is van implementációja ESP8266-ra, de ezekkel már nem foglalkoztam részletesebben.

# Tervezés

## A feladat célja

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

A csomópontokban lévő internetre kapcsolódó eszközök alapvetően két funkciót valósítanak meg. Egyrészt képesnek kell lenniük 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.

## Szenzor működési módok

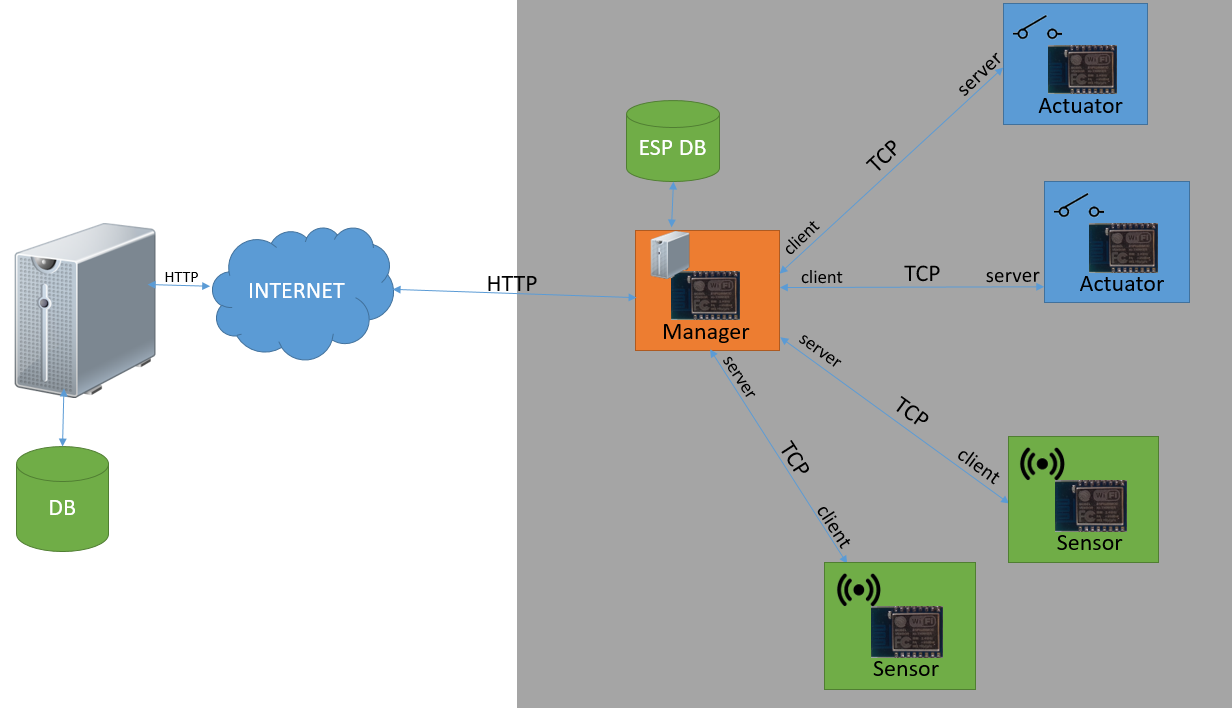
A fentiekben vázlatosan ismertetett feladatok megvalósításához több különböző működési módot implementálunk.

A három fő, különálló fázis:

* Csatlakozás a vezeték nélküli hálózathoz
* Szenzor
* Beavatkozó
* Lokális szerver

Ezeket a későbbiekben egyenként, részletesebben is ismertetni fogjuk.

## A rendszer topológiája

A rendszer alapvető elemeit az alábbi ábrán követhetjük figyelemmel: 

A központi szerver az ábra bal oldalán látható. A szürke négyzet egy, fizikailag és logikailag is többé-kevésbé egy helyen lévő egységet takar, nevezzük Locationnek. A Location alapvetően három féle elemet tartalmaz, szenzorokat, beavatkozókat, illetve mindenhova tartozik egy helyi szerver is. Az természetesen nincs kizárva, hogy utóbbinak fizikailag valamelyik 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 alapvetően a helyi, lokális szerverünk kommunikál, a többi modul meg rajta keresztül tartja a kapcsolatot, azaz a szenzorok ide küldik az adataikat, a beavatkozókat pedig közvetlen módon a lokális szerver utasítja, a saját központi szerverről frissített adatbázisa alapján.

## Működési módok

### Szenzor

A szenzorhálózat csomópontjaiban az eszköz egyik fő felelőssége, hogy előre meghatározott időközönként rámérjen az érzékelőkre, majd a kiolvasott adatokat továbbítsa a helyi szervernek. 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 sleep 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 ilyen módban működő 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 szenzor működési mód egy ciklusa a következőképp vázolható.

1. Alapállapot: Sleep
2. Sleepből automatikus felébredés meghatározott időközönként vagy külső esemény hatására
   * Szenzorok (ADC, I2C) mért értékeinek lekérdezése
   * Lekérdezett adatok esetleges konvertálása
   * Adatok beküldése a lokális szerverre a TCP csatornán keresztül.
3. Vissza Sleep állapotba (időzítve a következő felébredést)

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 |

## Működési módok

### Manager

A manager működése egy adott Location 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 lehetőség szerint rendszeresen továbbítania kell a szenzorok által beküldött adatokat, valamint frissen kell tartania a saját környezetéhez tartozó szabályadatbázisát is.

### 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 szerverrel, 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

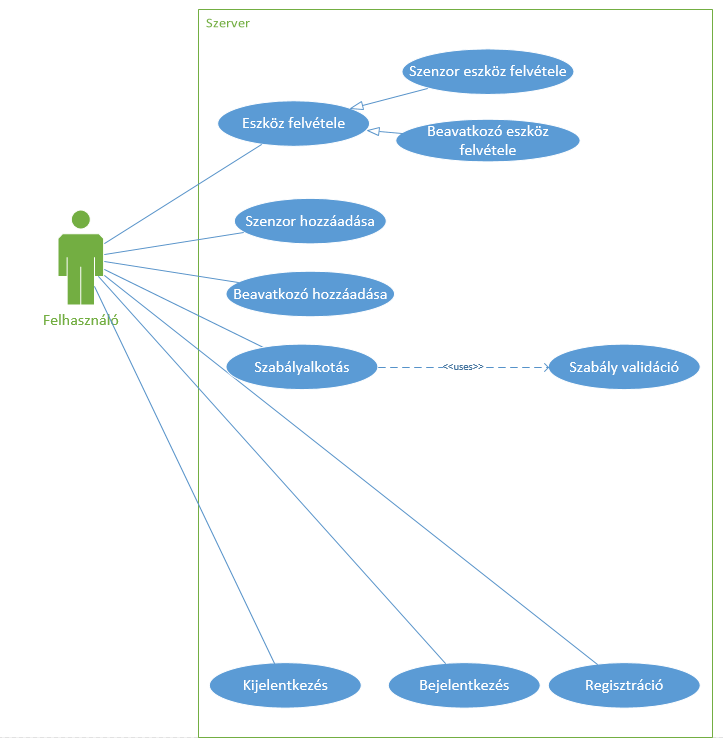
A fenti módban például egy „R1 ON” üzenet hatására az eszköz a megfelelő GPIO lábának 1-be tételével az R1-es relét átkapcsolhatja másik állásba. Másik lehetőség például, hogy a PWM interfészén keresztül végzi egy villanymotor vezérlését a szervertől kapott utasítások függvényében.

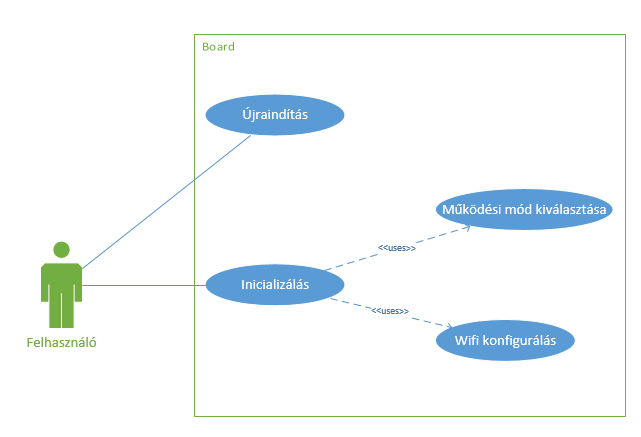
### Csatlakozás a hálózathoz

Az első mód működéséhez szükséges, hogy ténylegesen is legyen az eszköznek hozzáférése valamely vezeték nélküli hálózathoz. Ezért az eszköz indulását követően mielőtt bármi más történne, szükséges, hogy a modul felcsatlakozzon valamilyen (alapvetően előre megmondott) vezeték nélküli hálózatra.

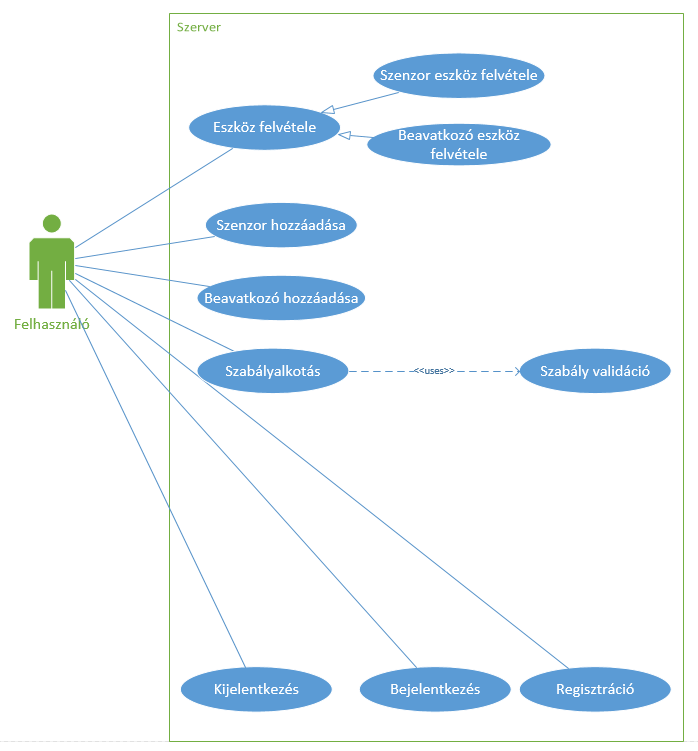
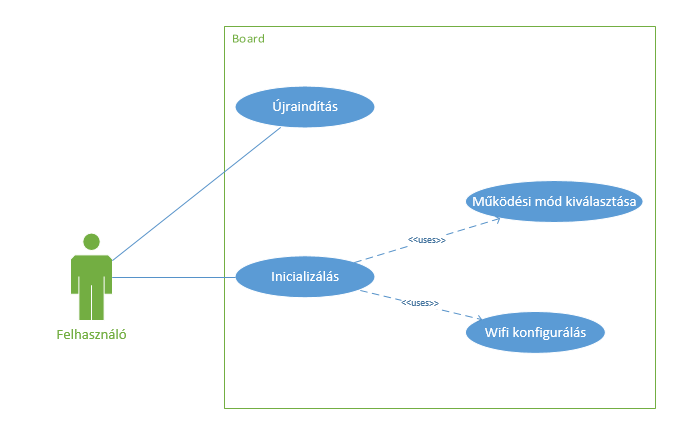
Az indítást követően így az ESP8266 elsőként ebben a módban indul, Acces Pointként (AP üzemmód), a rá csatlakozó eszköznek egy webes felületet nyújtva, amin lehetővé teszi, hogy a felhasználó megadja neki a Wifi-re történő csatlakozásához szükséges SSID és Password mezőket. Amikor ezek megtörténtek, a modul automatikusan Station módba vált, majd csatlakozik az előbb megadott hálózathoz a megadott jelszóval, amit követően az előzetes beállításának megfelelően válthat Szenzor vagy Beavatkozó üzemmódba.

## Felhasználói interakciók





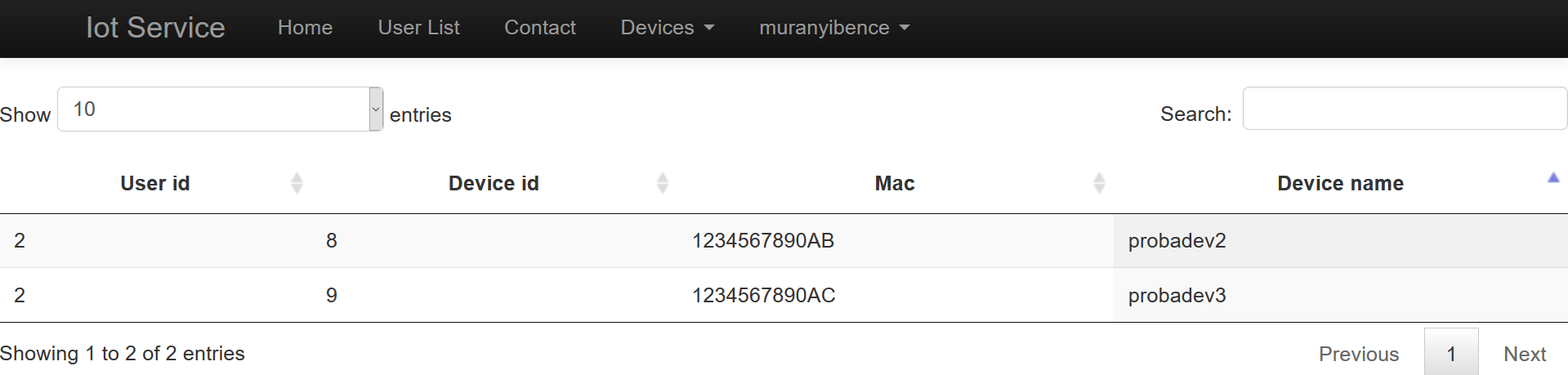
### Eszköz



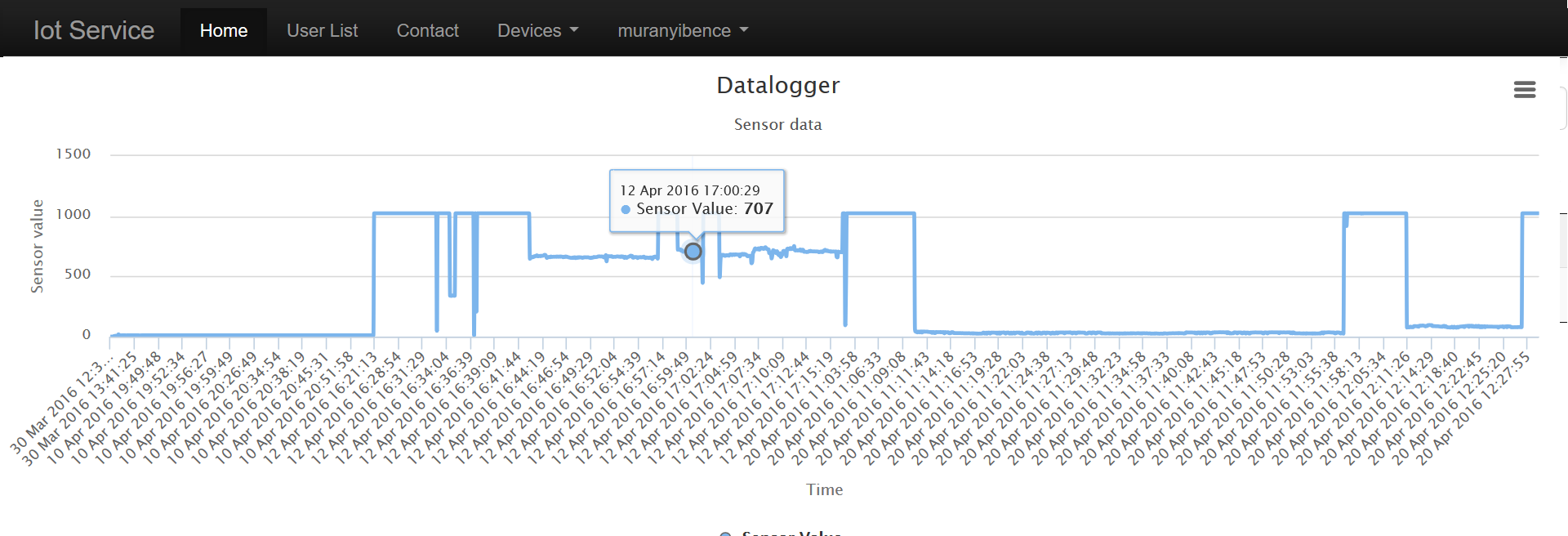
## A központi szerver

A központi szerver működése az egész rendszer számára kritikus, hiszen rajta keresztül vesszük fel az eszközeinket és szabályainkat. Ennek ellenére a fenti topológiban 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 regisztrálhasssák, szenzorokat, beavatkozókat hozhassanak létre, valamint ehhez szabályokat definiálhassanak. Ezekhez megfelelő, lehetőség szerint minél egyszerűbb formokat kell létrehoznunk, amelyeken a fenti műveletek elvégezhetőek és követhetőek. A lrnti 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 félév első részét az új firmware megismerésével és kipróbálásával töltöttem, miközben elkezdtem a saját, webes szerver megoldásom implementálását.

Utóbbi nagyrészt már használható, a legtöbb, tőle kívánt funkció már 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 mgtekinthetjük. 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 elsősosorban a rendszer elemei közötti kommunikáció megvalósításán dolgoztam, továbbá olyan megvalósítandó feladatokon, mint a lokális szerver hitelesítése a központi szerver felé.

## Tervek a jövőre nézve

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. Az asszociatív tömb (angolul map vagy dictionary) adatszerkezet kulcs-érték párokat tartalmaz, ahol egy kulcshoz pontosan egy érték tartozik. [↑](#footnote-ref-1)
2. Az ESP8266 eszközök képesek deep sleep üzemmódra. Látható, hogy ha nincs igényünk az eszköz folyamatos működésére (mint például a szenzor üzemmódnál), akkor a fogyasztás drasztikusan csökkenthető. A dolgozat a későbbiekben még foglalkozik ezzel a funkcióval. [↑](#footnote-ref-2)