UNIVERSITATEA „SAPIENTIA” DIN CLUJ-NAPOCA

FACULTATEA DE ȘTIINȚE TEHNICE ȘI UMANISTE,

TÎRGU-MUREȘ

SPECIALIZAREA CALCULATOARE

Proiectarea și dezvoltarea unui sistem de control al casei inteligente

PROIECT DE DIPLOMĂ

Coordonator științific:

Conf. dr. ing. Kutasi Dénes Nimród,

Ș.l. dr. ing. Szabó László Zsolt

Absolvent:

Bács Bernát

2023

UNIVERSITATEA „SAPIENTIA” din CLUJ-NAPOCA **Viza facultății:**
Facultatea de Științe Tehnice și Umaniste din Târgu Mureș
Specializarea: **Calculatoare**

LUCRARE DE DIPLOMĂ

Coordonator științific:	Candidat: **Bács Bernát**
Conf. dr. ing. Kutasi Dénes Nimród	Anul absolvirii: **2023**
Ș.L.dr.ing. Szabó László Zsolt	

a) Tema lucrării de licență:
PROIECTAREA ȘI DEZVOLTAREA UNUI SISTEM DE CONTROL AL CASEI INTELIGENTE

b) Problemele principale tratate:
- Studiu bibliografic privind sistemele de control dezvoltate pentru case inteligente
- Proiectarea unui sistem de măsură și control pentru casa inteligentă. Alegerea componentelor, alegerea tehnologiilor de programare.
- Metode soft computing aplicabile pentru casa inteligentă.
- Realizarea practică.

c) Desene obligatorii:
- Schema bloc al aplicației
- Schema bloc al sistemului de comunicații realizat
- Realizarea practică.fotografie.

d) Softuri obligatorii:
-Programul de comandă și control realizat pe unitatea centrală
- Modului de comunicație cu senzorii inteligenți
- Funcția soft computing realizat

e) Bibliografia recomandată:

[1] Nitin Naik, „Choice of Effective Messaging Protocols for IoT Systems: MQTT, CoAP, AMQP and HTTP,” Defence School of Communications and Information Systems Ministry of Defence, United Kingdom.
[2] Andy Stanford-Clark, Hong Linh Truong, „MQTT For Sensor Networks (MQTT-SN) Protocol Specification,” 2013
[3] Amezen, N. M., & Al-Ameri, J. A. M., “IoT-Based Shutter Movement Simulation for Smart Greenhouse Using Fuzzy-Logic Control. ”, 2019

f) Termene obligatorii de consultații: săptămânal
g) Locul și durata practicii: Universitatea „Sapientia” din Cluj-Napoca,
Facultatea de Științe Tehnice și Umaniste din Târgu Mureș
Primit tema la data de: 31.03.2022
Termen de predare: 27.06.2023

Semnătura Director Departament

Semnătura coordonatorului

Semnătura responsabilului programului de studiu

Semnătura candidatului

Declarație

Subsemnatul Bács Bernát absolvent a specializării Calculatoare., promoția 2023 cunoscând prevederile Legii Educației Naționale 1/2011 și a Codului de etică și deontologie profesională a Universității Sapientia cu privire la furt intelectual declar pe propria răspundere că prezenta lucrare de licență/proiect de diplomă/disertație se bazează pe activitatea personală, cercetarea/proiectarea este efectuată de mine, informațiile și datele preluate din literatura de specialitate sunt citate în mod corespunzător.

Tg-Mureș,

Data: 27.06.2023

Bács Bernát

Semnătura……

Proiectarea și dezvoltarea unui sistem de control al casei inteligente

Extras

Această lucrare prezintă implementarea unui sistem inteligent de control al casei. Sistemul îndeplinește în principal sarcini de automatizare a locuinței, cum ar fi controlul temperaturii, udarea plantelor și mișcarea automată a jaluzelelor, facilitând astfel viața de zi cu zi a utilizatorului. În prezent, dispozitivele IoT are un rol din ce în ce mai important pentru oameni și nu ne putem imagina viața noastră fără ele. Deoarece majoritatea oamenilor au acasă o conexiune la internet, construirea unui astfel de sistem IoT este mai ușoară, mai ieftină și mai eficientă, deoarece nu este nevoie de fire pentru a comunica între dispozitive.

Sistemul pe care l-am dezvoltat și construit constă într-o unitate centrală la care este conectat un ecran tactil, unde puteți urmări și interveni în procesele de automatizare, puteți vedea statistici despre temperatura din camere. La această parte centrală sunt conectați mai mulți senzori prin intermediul protocoalelor de comunicare Bluetooth sau MQTT, care furnizează date și îndeplinesc sarcinile de automatizare. În lucrare mea de diplomă documentez în detaliu proiectarea și construcția acestui sistem, ilustrând fiecare parte cu diagrame.

Cuvinte cheie: IoT, Home-automation, Raspberry Pi, MQTT

SAPIENTIA ERDÉLYI MAGYAR TUDOMÁNYEGYETEM
MAROSVÁSÁRHELYI KAR
SZÁMÍTÁSTECHNIKA SZAK

Okosotthon vezérlő rendszer tervezése és fejlesztése

DIPLOMADOLGOZAT

Témavezetők:
Dr. Kutasi Dénes Nimród,
egyetemi előadótanár
Dr. Szabó László Zsolt,
egyetemi adjunktus

Végzős hallgató:
Bács Bernát

2023

Kivonat

A dolgozat egy okosotthon vezérlőrendszer megvalósítását mutatja be. A rendszer elsősorban otthon automatizálási feladatokat lát el, mint például a hőmérséklet szabályzása, a növények öntözése és a redőnyök automatikus mozgatása, ezzel megkönnyítve a felhasználó mindennapjait. Napjainkban az IoT eszközök egyre fontosabb szerepet töltenek be az ember számára, már el sem tudjuk képzelni ezek nélkül az életünket. Mivel az emberek nagy része rendelkezik otthoni internetkapcsolattal, ezért egy ilyen IoT rendszer kiépítése egyszerűbb, olcsóbb és hatékonyabb, mert nincs szükség vezetékekre az eszközök közötti kommunikáció megvalósítására.

Az általam fejlesztett és megépített rendszer egy központi részből áll, amihez egy érintőképernyős kijelző van csatlakoztatva, itt lehet követni, illetve beavatkozni az automatizálási folyamatokba, statisztikákat lehet megnézni például a szobában lévő hőmérsékletekről. Ehhez a központi részhez több érzékelő kapcsolódik Bluetooth, illetve MQTT kommunikációs protokollok segítségével, amik adatokat szolgáltatnak, valamint ellátják az egyes automatizálási feladatokat. A dolgozatomban ennek a rendszernek a tervezését és megépítését dokumentálom le részletesen, az egyes részeket ábrákkal szemléltetve.

Kulcsszavak: IoT, Home-automation, Raspberry Pi, MQTT

Abstract

This thesis presents the implementation of a smart home control system. The system mainly performs home automation tasks such as temperature control, watering of plants and automatic movement of shutter, thus facilitating the user's daily life. Nowadays, IoT devices are playing an increasingly important role for people, and we can't imagine our lives without them. Since most people have an internet connection at home, building such an IoT system is easier, cheaper and more efficient because there is no need for wires to communicate between devices.

The system I developed and built consists of a central computer with a touch screen display connected to it, where you can follow and intervene in the automation processes, you can see statistics about the temperature in the room, for example. Several sensors are connected to this central part via Bluetooth or MQTT communication protocols, which provide data and perform the automation tasks. In my thesis I document the design and construction of this system in detail, illustrating each part with diagrams.

Keywords: IoT, Home-automation, Raspberry Pi, MQTT

Tartalomjegyzék

Ábrák jegyzéke

Táblázatok jegyzéke

1. Bevezető

Napjainkban az okosotthonok egyre népszerűbbek a világon, és egyre többen választják ezt a modern, innovatív és kényelmes megoldást otthonaik vezérlésére és automatizálására, ha van rá lehetőségük. Az okosotthon-technológiáknak köszönhetően a lakásokban számos eszköz, mint például a világítás, a fűtés, a klíma, a biztonsági rendszerek és az audiovizuális berendezések, könnyedén irányíthatóak és programozhatóak, akár távolról is a legtöbb esetben. Az okosotthonok vezérlésére használt rendszerek általában egy központi egységből állnak, amely lehetővé teszi a felhasználók számára, hogy az okostelefonjuk, a táblagépük vagy egy beágyazott kijelzőhöz kapcsolt, valamilyen szoftveren keresztül kommunikáljanak a rendszerrel. Egy okosotthon-vezérlőrendszer segítségével az otthoni berendezések könnyedén, egyszerűen és intuitívan kezelhetőek, ami jelentősen növeli az otthon kényelmét. Vannak olyan esetek is, amikor nem pusztán az otthon kényelméről, hanem annak biztonságáról is szó van. Például egy olyan személynek, aki tüdőbeteg és nagyon érzékeny a magas hőmérsékletre, illetve a magas páratartalomra, egy ilyen rendszer egy biztonságosabb lakást tud teremteni azzal, hogy a páratartalmat és a hőmérsékletet megfelelően szabályozza. Ezen kívül az okosotthon-technológiák energiahatékonyabbá tehetik az otthonokat, mivel a vezérlőrendszer lehetővé teszi a fogyasztás optimalizálását, így csökkentve a felesleges energiafelhasználást.

Az okosotthon-technológia már ma is számos ember életét megkönnyíti, és a jövőben valószínűleg még szélesebb körben lesz elérhető. Az okosotthon-vezérlőrendszerek egyre intelligensebbek és sokoldalúabbak lesznek, így az otthonok irányítása és automatizálása egyre egyszerűbb és hatékonyabb lesz.

1.1. Dolgozat célja

A dolgozatom célja egy olyan rendszer létrehozása, amellyel megvalósítható az okosotthon koncepció. Egy okosotthon vezérlőrendszere rengeteg funkciót tartalmazhat, én a dolgozatomban csak néhány funkciót valósítok meg. A rendszer egy Raspberry Pi-n alapszik, amihez csatlakoztatva van egy LCD érintőképernyő. Erről a képernyőről lehet konfigurálni, illetve irányítani a rendszert. A funkciókhoz tartozik a hőmérséklet és a páratartalom figyelése,

illetve az ehhez történő statisztikák követése hónapokra lebontva, a növények öntözése és a fűtés szabályozása. A Raspberry-hez a képernyőn kívül más eszköz vezetékesen nem csatlakozik, az volt a célom, hogy az érzékelők, illetve a vezérlőrendszer közötti kommunikáció vezeték nélkül történjen. Ennek egy lakásban az a nagy előnye, hogy az érzékelők bárhol elhelyezhetőek függetlenül attól, hogy hol található a vezérlőrendszer.

1.2. Do It Yourself – DIY

A "Do It Yourself" (magyarul: Csináld magad) angol kifejezés arra utal, hogy egy egyén bizonyos célját saját maga próbálja megoldani szakszerű segítség kérése nélkül. A DIY tevékenységek nagyon széles skálája tartozik ide, beleértve mindenféle otthoni barkácsolást, festést stb. Napjainkban már egyre olcsóbbak az elektronikai eszközök, érzékelők, relék és ennek köszönhetően a DIY mozgalom kiterjedt az elektronikára is. Rengeteg dokumentáció érhető el az interneten ezeknek az eszközöknek a működéséről és így viszonylag egyszerűen elő lehet állítani bármilyen eszközt a lakásban, amit egy személy megálmodott. Az általam készített rendszer szerintem beillik ebbe a DIY mozgalomba, ez egy olyan rendszer, amit a továbbiakban a lakásomban használni tudok.

1.3. Internet of Things – IoT

Az IoT (magyarul: Dolgok internete) egy technológiai trend, amely lényege, hogy különböző eszközök összekapcsolhatóak az internet segítségével, tehát egy IoT eszköz lényegében egy internetre csatlakoztatható egység. Több IoT eszköz képes kommunikálni egymással, adatokat gyűjteni a környezetből, önálló feladatokat ellátni. Ezeknek az eszközöknek egyik nagy előnye, hogy tetszőleges távolságra lehetnek egymástól, így a kommunikációhoz nem szükséges kábel. Az IoT-nek hála akár távolról is szabályozhatjuk az otthonunkban lévő eszközöket, erre egy klasszikus példa, amikor hosszabb ideig nem vagyunk otthon és hazaindulva indítjuk el a fűtést a lakásunkban azért, hogy mire hazaérünk, meleg fogadjon az otthonunkban.

2. Elméleti megalapozás és szakirodalmi tanulmány

2.1. Szakirodalmi tanulmány

A szakirodalmi tanulmány során először a Raspberry Pi-k felépítését és működését vizsgáltam, ezután pedig a napjainkban használt otthon automatizálási rendszereket néztem meg kicsit közelebbről. A Raspberry Pi-ről, annak felépítéséről és a választott modellről, majd a 2.3.1 pontban fogok részletesebben írni.

Ha a szakirodalomban rákeresünk a *home automation* kulcsszóra, akkor nagyon sok anyagot találhatunk, ugyanis napjainkban nagyon elterjedt az ilyen rendszerek készítése.

Ahogyan azt az [1] tanulmányban olvashatjuk, a vezeték nélküli technológián (főleg Bluetooth) alapuló otthon automatizálási rendszerek egyre népszerűbbek, ezért az én dolgozatomban is igyekeztem a minél kevesebb kábelt használni a projekt elkészítéséhez. Ebben a tanulmányban említett alkalmazás központi vezérlőegység szintén egy beágyazott weboldal, amit a felhasználó egy érintőképernyőről tud kezelni.

Összességében az otthonunk kényelmének érdekében, egy jól működő és kényelmes otthon automatizálási rendszer minden eszköz vezérélését egyetlen helyre kell gyűjtse, azaz egy helyről irányítható kell legyen a teljes rendszer.

2.2. Ismert hasonló alkalmazások

A mai világban már számtalan otthon automatizálási rendszer ismeretes. Legtöbbjük néhány olyan funkciót is tartalmaz, amit egy átlagember nem nagyon használ, gondolok itt például egy jacuzzi vagy medence vezérlésére. A továbbiakban bemutatok három otthon automatizálási rendszert, amelyek nagyon népszerűek a piacon, és véleményem szerint ezek az alkalmazások a legjobbak között vannak. Nyilván ezek a rendszerek sokkal fejlettebbek, mint az általam készített rendszer, ami részben annak köszönhető, hogy vannak olyan funkcionalitások a nagyobb rendszerekben, amelyet egy diplomamunka keretein belül nem lehet megvalósítani, mert ahhoz számos nagyon drága alkatrészre, valamint sokkal több időre lenne szükség.

2.2.1. LOGO!

A LOGO! [2] a Siemens által gyártott költséghatékony otthoni automatizálási rendszer egy programozható vezérlőegységből és hozzá kapcsolódó szenzorokból és kimenetekből áll. Az egység programozása a LOGO!Soft Comfort szoftverrel történik, amely egy grafikus programozási környezetet biztosít a felhasználók számára. A felhasználók létrehozhatnak programokat, amelyek különféle szenzorok és kimenetek felhasználásával vezérlik az otthoni berendezéseket és rendszereket. A rendszerben található szenzorok a környezet különféle paramétereit mérik, például a hőmérsékletet, a páratartalmat, a világítást, a mozgást és az ajtók és ablakok állapotát. Az egység kimenetei vezérelik az otthoni berendezéseket. A készülékeket bármilyen okos eszközről tudjuk vezérelni Wi-Fi-n keresztül.

2.2.2. Eaton xComfort RF

Az Eaton xComfort RF [3] rendszer egy vezeték nélküli épületautomatizálási rendszer. Az xComfort alkalmazás lehetővé teszi az otthoni rendszerek távoli vezérlését és monitorozását, és lehetővé teszi az ütemezési funkciók és a szabályok beállítását az otthoni berendezések működésének automatizálásához. Az alkalmazás segítségével a felhasználók az otthonuk bármely pontjáról vezérelhetik az xComfort rendszer elemeit, vagy akár távolról is, az interneten keresztül. Az Eaton xComfort RF rendszer biztonságos és megbízható, mivel a vezeték nélküli kommunikáció AES-128 titkosítással van védve, így biztosítva a személyes adatok és a vezérlési információk biztonságát.

2.2.3. Google Home

A Google [4] által fejlesztett okosotthon nagyon hasonló a fentiekben leírt két rendszerhez, viszont nagy különlegeség ebben a rendszerben, hogy támogat hang alapú vezérlést. A rendszernek a Google Asszisztensen keresztül lehet utasításokat adni vagy akár kérdéseket is fel lehet tenni, amire ő majd válaszol. A hang alapján bármilyen kérdésre tud válaszolni, mert össze van kötve a Google-el, ezért ha kérdezünk tőle valamit, olyan mintha a Google-on keresnénk rá. Ez az egyik nagy előnye a többi rendszerrel szemben, viszont ugyanakkor a hátránya is, mert az állandó hallgatózás egy kicsivel több elektromos energiát igényel.

2.3. Felhasznált technológiák

2.3.1. Raspberry Pi 3 Model B+

A Raspberry Pi nem más, mint egy számítógép, pontosabban egy egykártyás számítógép, mérete alig nagyobb, mint egy bankkártya, egyszerű architektúrával rendelkezik. Az ára viszonylag alacsony, elsősorban tanulási célokra fejlesztették ki és elég népszerű lett. Mivel egy számítógépről beszélünk, ezért a Raspberry Pi-n egy operációs rendszernek is futnia kell, ahhoz, hogy használni tudjuk. Az ajánlott operációs rendszer az eszközhöz a Raspbian, ami nem más, mint a Linux Debian kifejezetten Raspberry Pi-re optimalizált változata. Ez az operációs rendszer rendelkezik előre telepített Python fordítóval, ami számomra hasznos, mert ezt a programozási nyelvet használtam a diplomamunka elkészítéséhez. Az újabb modellek rendelkeznek Ethernet csatlakozóval, illetve integrált Wi-Fi-vel és Bluetooth-al, ami egy ilyen témájú diplomamunka elkészítéséhez nélkülözhetetlen.

Az általam választott eszköz egy Raspberry Pi 3 Model B+ [5]. Ez a Raspberry Pi a legutolsó számítógép a hármas szériából. A számítógép központi feldolgozóegységét egy 64 bites, négy magos, ARMv8 processzor alkotja, amely 1,4 GHz-en működik és rendelkezik 1 GB LPDDR2 SDRAM-al. Található rajta egy Ethernet csatlakozó, 4 USB port, 40 darab GPIO pin, HDMI csatlakozó, 4 pólusú 3,5 mm-es jack csatlakozó, CSI kamera port, DSI kijelző port és egy microSD kártyahely, amely a Raspberry rendszerindításért és az adatok tárolásáért felel. Fontos megemlíteni, hogy ez a típus már rendelkezik dual-band 802.11ac vezeték nélküli hálózattal (vagyis Wi-Fi-vel), illetve Bluetooth 4.2-vel és BLE-vel. A tápellátása 5V/2,5A egy microUSB-n keresztül.

A választásom azért esett erre a típusra, mert egy IoT alkalmazás esetén elengedhetetlen a BLE technológia és a vezeték nélküli internet elérés, továbbá a rendszer specifikáció megfelelőek ahhoz, hogy egy weboldalt lehessen fejleszteni és működtetni rajta.

2.3.2. NodeMCU-32S

Az ESP32 egy nagyon sokoldalú és hatékony fejlesztői mikrovezérlő, amelyet az Espressif Systems fejlesztett ki. Az ESP32 rendkívül kisméretű, energiatakarékos és költséghatékony, ezért nagyon népszerűvé vált a fejlesztők körében.

A NodeMCU-32S egy ESP32-alapú fejlesztői mikrovezérlő, amely nagyon hasznos IoT alkalmazások fejlesztésére. Erős és hatékony processzora egy kétmagos Tensilica LX6, 240

MHz-es frekvencián működik, továbbá rendelkezik 512 KB RAM-mal és 4 MB Flash memóriával, amely egy mikrovezérlőhöz képest nagy kapacitásnak számít, ugyanis ez a memória csak a rá megírt programokat kell tárolja. Ahogyan a használt Raspberry Pi, úgy a NodeMCU-32S is rendelkezik vezeték nélküli internet hozzáféréssel, Bluetooth 4.2-vel és BLE-vel. A tápellátása szintén microUSB csatlakozóval történik, illetve a mikrovezérlő rendelkezik 26 darab GPIO pinnel, ami kimenet esetén maximum 3,3V feszültséget tud szolgáltatni, annak ellenére, hogy az eszköz 5V-al működik.

2.3.3. Raspbian

A Linux egy nyílt forráskódú, bárki számára ingyenesen elérhető, operációs rendszer mag. Az operációs rendszer kernele (vagyis a magja) felelős a hardver és a szoftver közötti kommunikációért, és ez teszi lehetővé az alkalmazások futtatását a számítógépen. A Linux a UNIX operációs rendszerre épül, és számos disztribúciója létezik már, mint például a Fedora, az Ubuntu vagy a Debian. Az én esetemben a Raspbiant használtam, ami egy Debian alapú, kifejezetten Raspberry Pi számítógépre optimalizált operációs rendszer.

2.3.4. Django keretrendszer

A Django egy ingyenes, nyílt forráskódú, Python nyelven írt webkeretrendszer, amely lehetővé teszi a fejlesztők számára, hogy gyorsan és hatékonyan készítsenek dinamikus, adatbázisokkal összekötött webalkalmazásokat. A Django model-template-view (MTV) architektúrára épül, amely elválasztja az adatbázis-kezelést, a logikát és a megjelenítést egymástól. A Django keretrendszerben a modell reprezentálja az adatbázisban tárolt adatokat, a nézet felelős a felhasználói felület megjelenítéséért és kezeléséért, és a vezérlő összeköti a két réteget. Az ORM (Object-Relational Mapping) segítségével az adatbázis kezelése egyszerűbbé válik, mivel az objektumokat adatbázis példányokká alakítja, így egy adatbázist felfoghatunk úgy, mint egy osztály, a benne lévő mezőket úgy, mint az osztály adattagjai. A Djangoban a megjelenítési réteg úgynevezett sablonokból (templates) épül fel, amelyek HTML kódban beágyazott Python kódot tartalmaz. Az alkalmazások URL-jeit az urls.py fájlban definiáljuk, amelyek a nézeteket hívják meg, és kezelik a felhasználói interakciókat. A static mappában helyezkednek el a weboldalhoz tartozó CSS és JavaScript fájlok. A keretrendszer rendelkezik

egy admin felülettel, ahol kezelni lehet az adatbázisban levő táblákat, adatokat lehet beszúrni, illetve a már meglévő adatokat lehet módosítani.

Ha egy weboldal elkészítéséhez használunk egy keretrendszert, annak számos előnye van: felgyorsítja a fejlesztést, mert nem kell "újra feltalálni a kereket", azaz sok beépített funkció használható, és ha szükségünk van adatbázisra, akkor annak a kezelése nagyon egyszerű és hatékony.

2.3.5. Bluetooth Low Energy

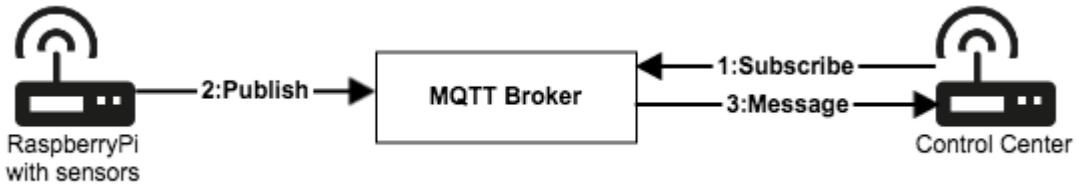
A Bluetooth [6] egy olyan vezeték nélküli kommunikációs protokoll, amely segítségével a különböző eszközök képesek adatokat cserélni valamint kommunikálni egymással rövid távolságon keresztül, általában 10 méteren belül. A Bluetooth lehetővé teszi egy személyes hálózat kialakítását a közeli eszközök között, így az eszközök vezeték nélkül tudnak egymással adatot cserélni. A Bluetooth egyik hátránya, hogy viszonylag magas az energiafogyasztása, ha állandóan be van kapcsolva, például ha egész nap bekapcsolva hagyjuk a Bluetooth-ot a telefonunkon, akkor észrevehetjük, hogy valamennyivel gyorsabban fog merülni a készülék akkumlátora. Ez a korlátozott teljesítményű, és kicsi akkumlátorokat vagy elemeket használó eszközöknél (például IoT eszközöknél) egy felmerülő probléma. Ezt a problémát hivatott megoldani a BLE, vagyis a Bluetooth Low Energy.

A BLE protokoll kifejezetten kis energiafelhasználású eszközök, például szenzorok és IoT eszközök kommunikációjára van tervezve. Az alacsony energiafogyasztás elérése érdekében a BLE eszközöket az idő nagy részében alvó állapotban tartják. A kapcsolatok időtartama nem haladja meg a néhány milliszekundumot, ami jelentősen rövidebb a hagyományos Bluetooth kapcsolatok körülbelüli száz milliszekundumos időtartamánál [6]. Mivel általában az érzékelők és az IoT eszközök csak nagyon kis adatmennyiséget küldenek, így ez a néhány milliszekundumnyi kapcsolatidő is bőven elegendő. Ennek köszönhető a BLE alacsonyabb energiafogyasztása.

2.3.6. MQTT

Az MQTT (Message Queuing Telemetry Transport) [7] egy nyílt kommunikációs protokoll, amelyet direkt az IoT eszközök közötti kommunikációra terveztek. Az MQTT kifejezetten az olyan helyzetekre lett kialakítva, ahol a kommunikációs eszközöknek korlátozott

energiaforrásai vannak. Ez a protokoll alapvetően egy üzenetküldő protokoll és lehetővé teszi a hatékony és biztonságos adatátvitelt. Az MQTT protokollban a kommunikáció témák (topics) és üzenetek (messages) segítségével történik. Az eszközök egy adott témára iratkozhatnak fel (subscribe), majd azok az üzenetek, amelyeket más eszközök a témára küldtek, elérhetővé válnak a feliratkozók számára. A kommunikáció lehet kétoldalú is, egy eszköz küldhet is üzenetet és fogadhat is, valamint egy témára több kliens is feliratkozhat. A rendszer kommunikációját az 1. ábra mutatja be.



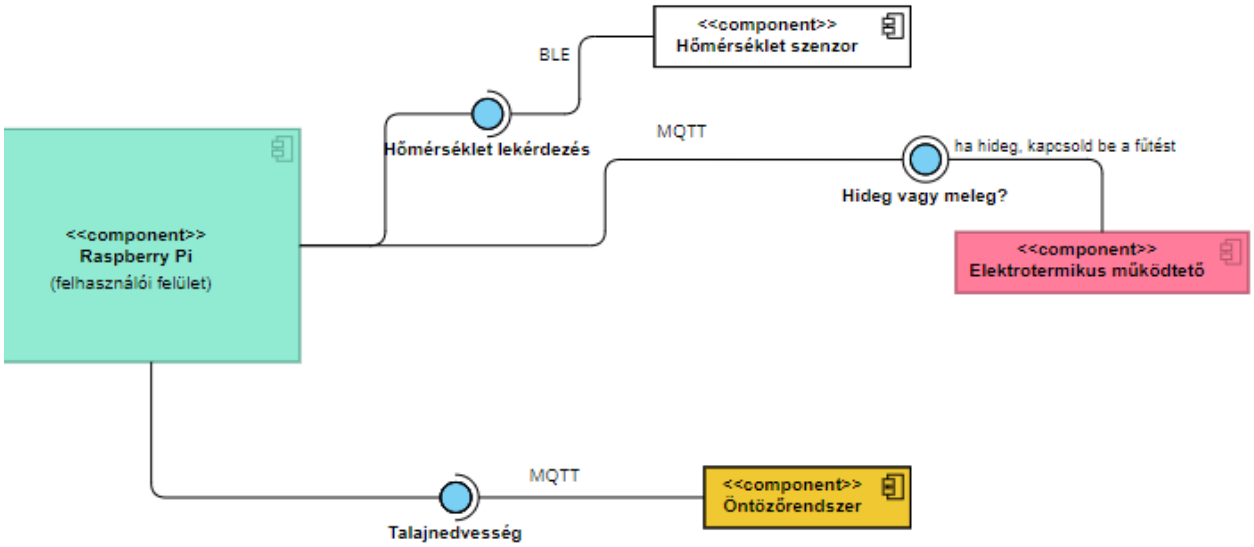
1. ábra Kommunikáció MQTT-vel

Annak érdekében, hogy az üzenet kézbesítéséről meg lehessen győződni, úgynevezett szolgáltatási minőségi szinteket vezetett be az MQTT (angolul Quality of Service). Ezek tartalmazzák azt a tényt, hogy a fogadó fél valóban megkapta-e a küldött üzenetet [8]. Az említett szintek a következők:

0. QoS 0: a protokollnak ez a szintje a legalacsonyabb, itt konkrétan nincs visszajelzés arról, hogy az üzenetet a fogadó felek megkapták vagy nem. Mivel az MQTT TCP/IP protokollra épít, ezért ebben az esetben csak ezen múlik, hogy az üzenet célba ért vagy sem [9].
1. QoS 1: ezen a szinten az biztos, hogy az üzenet egyszer megérkezik a fogadó felekhez. Ezt a fogadó egy PUBACK [9] csomag küldésével hitelesíti azt, hogy megkapta a csomagot. Ha a küldő fél egy bizonyos idő után nem kapja meg a PUBACK csomagot, akkor az üzenetet ismét el fogja küldeni.
2. QoS 2: ez a küldés a legmegbízhatóbb és együtt a leglassabb is. Itt a fogadó az üzenetet pontosan egyszer fogja megkapni egy négyszeri kézfogásos út segítségével. Ebben az esetben a küldő és a fogadó többször fog üzenetet váltani egymással a nagyobb biztonság érdekében. Ha valahol ez az oda-vissza küldés elakad, akkor az üzenet küldése elölről fog kezdődni [10].

3. A rendszer specifikációi és architektúrája

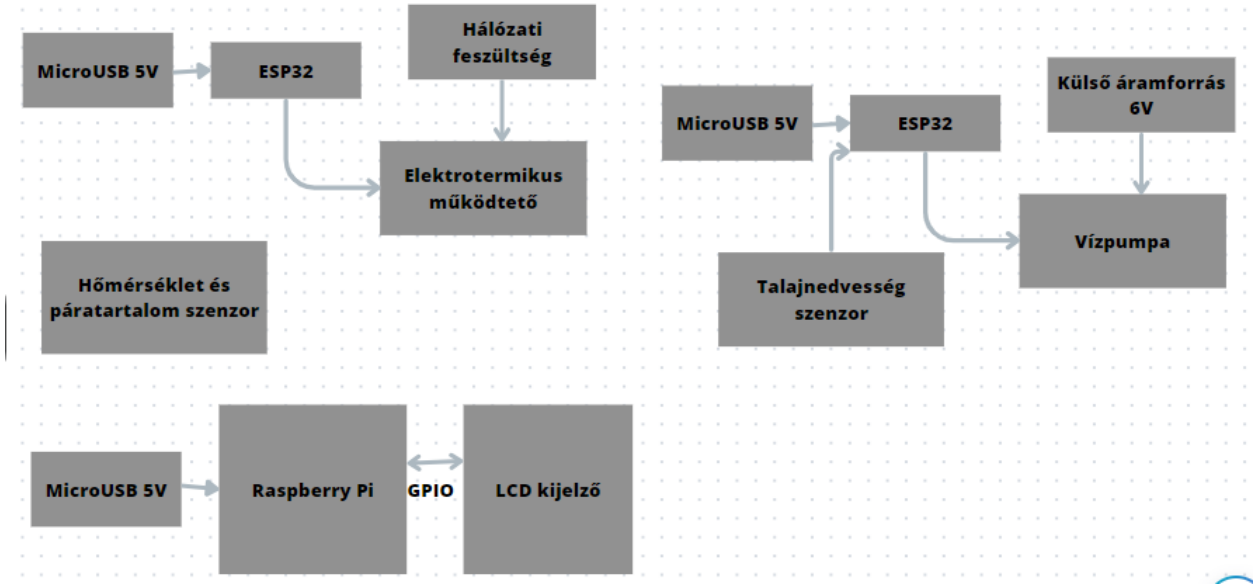
A rendszer felépítése több fő komponensből tevődik össze: egy Raspberry Pi számítógép, ami a központi vezérlőegység szerepét tölti be, valamint a hozzá kapcsolódó mikrovezérlők, amik a kliensek. A rendszer teljes architektúráját a 2. ábra mutatja be.



2. ábra A rendszer architektúrája

3.1. Hardver architektúra

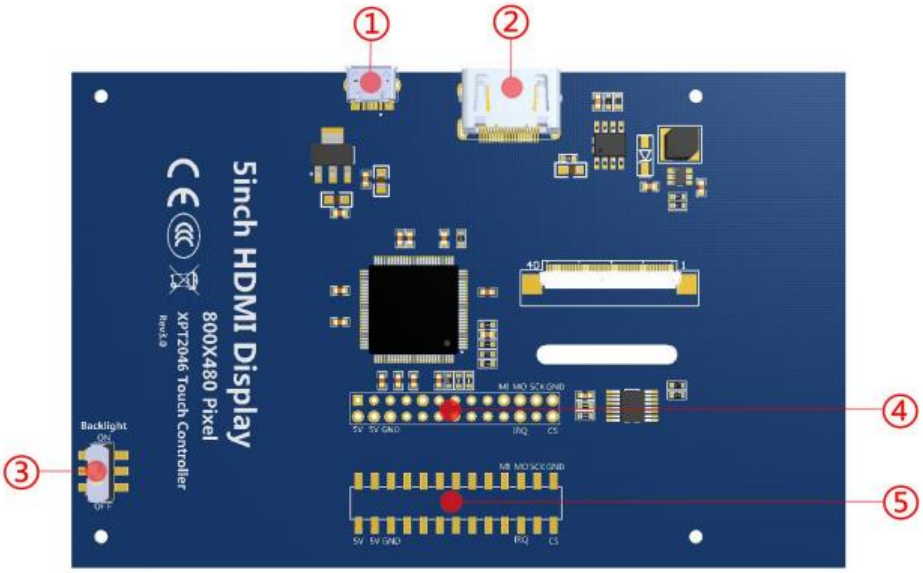
A Raspberry Pi 3 Model B+ számítógép szolgáltatja a rendszer magját, amihez egy 5 inches átmérőjű kijelző van csatlakoztatva a GPIO pineken keresztül. Ez a kijelző szolgáltatja a rendszer kezelőfelületét. A Raspberry Pi-hez vezeték nélkül, BLE segítségével, kapcsolódik a hőmérséklet és talajnedvesség szenzor, valamint szintén vezeték nélkül kapcsolódik, MQTT kommunikációs protokoll segítségével több ESP32 típusú mikrovezérlő, amik ellátják a különféle feladatokat. A teljes hardver tömbvázlatát a 3. ábra mutatja be. A lentiekben részletesen bemutatom ezeket a feladatokat.



3. ábra Hardver tömbvázlat

3.1.1. LCD kijelző

Az általam használt kijelző egy 5 inch átmérőjű, 800x480 felbontású érintőképernyő, amely támogatja a HDMI bemenetet. Azért esett erre a kijelzőre a választás, mert kifejezetten Raspberry Pi-re készített kijelző és érintéssel vezérelhető, ami azért fontos, mert a rendszert erre a kijelzőről lehet irányítani érintésekkel.



4. ábra HDMI kijelző háta

A *4. ábra* mutatja be a kijelző hátsó részét, az ábrán láthatóak számmal való jelölések, amik a lényegesebb komponenseket jelölik, amelyek a következők:

1. microUSB interface: normál esetben ez szolgáltatja a kijelző áramellátását. Amennyiben a 4. pontban említett 13*2 pin socket csatlakoztatva van, (egy Raspberry Pi csatlakozott a kijelzőhöz) akkor ezt a csatlakozót nem kell, illetve nem is lehet használni.
2. HDMI interface: ezen keresztül kommunikál a Raspberry Pi a kijelzővel
3. ki/be kapcsol: ezen a kapcsolón ki illetve be tudjuk kapcsolni a kijelző háttérvilágítását
4. 13*2 pin socket: összesen 26 pin, aminek a segítségével a kijelző kapja az 5V tápellátást a Raspberry Pi-től, illetve szintén ezeknek a pineknek segítségével küldi az érintéseknek az információját a kijelző a számítógép felé.
5. kiterjesztett interface: a 13*2 pin socket kiterjesztése.

Miután az alkatrészek összeillesztése sikerült, telepíteni kellett a szükséges drivereket annak érdekében, hogy az érintések ténylegesen működjenek a kijelzőn, azaz legyen meg az interakció egy érintésre. Ehhez be kell jelentkezni a számítógépre és a terminálban le kell futtatni a következő parancsokat egymás után:

```
git clone https://github.com/goodtft/LCD-show.git
chmod -R 755 LCD-show
cd LCD-show/
sudo ./LCD5-show
```

3.1.2. Hőmérséklet és páratartalom szenzor

A hőmérséklet és a páratartalom mérésére egy Xiaomi által gyártott Mi Temperature and Humidity Monitor 2 LYWSD03MMC típusú eszközt használtam, amit a 5. ábra mutat be. Azért esett erre az eszközre a választás, mert támogat BLE-t és ennek a segítségével könnyedén lehet az adatokat vezeték nélkül, Bluetooth-on keresztül kommunikálni a Raspberry Pi felé. A szenzor egy CR2032 típusú 3V-os lapos elemmel működik, amivel akár több, mint egy évet is képes elem csere nélkül adatokat küldeni, továbbá rendelkezik egy kis LCD kijelzővel, amin látható a jelenlegi hőmérséklet, páratartalom és egy szimbólum, ami jelzi, hogy mennyire komfortos az adott térben a levegő.

Ahhoz, hogy az érzékelőből Bluetooth segítségével ki tudjuk olvasni a mért adatokat, szükségünk lesz a szenzor MAC címére. Ezt könnyen ki lehet deríteni a Raspberry Pi-ről, ha a terminálba lefuttatjuk a `sudo hcitool lescan` parancsot, ez ki fogja listázni az összes

közelben lévő BLE-s eszközt névvel ellátva. Ezek után, MAC cím szerint figyelhetjük a küldött adatokat a szenzortól. Alapértelmezetten az érzékelő minden 10 másodpercben küldi a mért adatokat hexadecimális értékek formájában a 0x0036 karakterisztikán, ami a következőképpen néz ki:

```
Notification handle = 0x0036 value: 0b 09 26 c8 0b
```

Ebben a részben a value rész az, ami minket érdekel, itt kódolva van három érték: a hőmérséklet, a páratartalom és az elem feszültségszintje. Láthatjuk, hogy az értékek hexadecimális alakban vannak, ezeket dekódolnunk kell és át kell alakítanunk tízes számrendszerbe, a következők szerint:

1. Hőmérséklet: az első négy számjegy adja meg a hőmérsékletet. Az értékek sorrendjét fel kell cserélni, a hexadecimális értéket át kell alakítani tízes számrendszerbe és a kapott számot el kell osztani 100-al, így megkapjuk a hőmérséklet értékét két tizedes pontossággal. Ez a számítás a fenti értékekre: `0b 09`, ezt megfordítjuk és a `09 0b` értéket kapjuk, amit ha átalakítunk tízes számrendszerbe `2315`, és elosztjuk 100-al, azaz a szenzor által küldött hőmérséklet `23,15 °C`.
2. Páratartalom: az ötödik és a hatodik számjegy jelzi a páratartalmat, ezt a hexadecimális értéket át kell alakítani tízes számrendszerbe és megkapjuk a páratartalmat százalékban. A fenti példában az érték `26`, ezt átalakítva `38-ot` kapunk, azaz a mért páratartalom `38%`.
3. Elem feszültségszintje: az utolsó négy számjegy jelzi az elem feszültségszintjét, az átalakítás pontosan úgy működik, mint a hőmérséklet esetében, annyi különbséggel, hogy itt az átalakított értéket 1000-el kell osztani. A fenti példában az utolsó négy számjegy `c8 0b`, ami megfordítva `0b c8`, átalakítva `3016`, elosztva 1000-el a 3,016 értéket kapjuk, ami azt jelenti, hogy az elem feszültségszintje `3,016V`.

5. ábra Hőmérséklet és páratartalom szenzor

3.1.3. Öntözőrendszer és talajnedvesség szenzor

A legtöbb esetben, legyen szó otthon automatizálási rendszerről vagy csak egy sima öntözőrendszerről, az automatizálási folyamat kimerül abban, hogy egy öntözőrendszert időzíteni lehet, hogy milyen időközönként vagy esetleg napszakonként kapcsoljon be, és, hogy mennyi időt működjön. Ez nem minden esetben megfelelő, mert az ilyen rendszer nem veszi figyelembe a talaj nedvességét, azaz ha éppen esik az eső vagy a talaj elég nedves az adott növénynek, az öntözés akkor is el fog indulni. Az általam tervezett és elkészített rendszer ezt a problémát hivatott megoldani, azzal, hogy egy talajnedvességet mérő szenzorral rendelkezik, így az öntözés csak akkor fog megvalósulni, amikor arra ténylegesen szükség van.

Az öntöző központi részét egy ESP32 típusú mikrovezérlő alkotja. Alapjáraton ez a mikrovezérlő végzi az öntözést: beolvassa a szenzor adatait, amennyiben szükséges az öntözés elindítja a vízpumpát, és amikor a talaj eléri a megfelelő nedvességet, akkor leállítja a pumpát. Ezen kívül külsőleg is be lehet avatkozni az öntözés folyamatába. Amennyiben a felhasználói felületen a felhasználó kikapcsolja az automatikus öntözést, lehetősége van manuálisan irányítani a pumpát.

A fent említett vízpumpa egy 3-6V feszültségű vízálló pumpa, ami ha vízben van képes a vizet egy csövön keresztül eljuttatni a növényhez. Ennek a működtetéséhez egy külső 6V-os áramforrást használok, mert az ESP32 kimeneti pinje nem képes akkora áramot szolgáltatni, ami meghajtaná ezt a motort.

A talaj nedvességének a mérésére egy kapacitív talajnedvesség szenzort használtam. A szenzor működésének az alapelve a kapacitás változásának érzékelése a talajban. Ha a talajhoz vizet adunk hozzá, a talaj kapacitása megváltozik. A víznek magas dielektromos állandója, ami azt jelenti, hogy jó vezetője a kapacitív hatásnak, azaz amikor a talajban a víztartalom növekszik, a kapacitív érzékelőnek a kapacitása is növekszik. Ezen az elven tudjuk mérni a talaj nedvességét ennek a szenzornak a segítségével. A szenzor három kábel segítségével csatlakozik az ESP32-höz, az egyik szál a plusz 5V, a másik a GND, a harmadik az, amelyiken az analóg adatot olvassa az ESP32. Fontos azt tudni, hogy a szenzort használat előtt kalibrálni kell annak érdekében, hogy az értékeket helyesen tudjuk kezelni. A szenzor által adott érték csak egy szám, ami önmagában nem ér túl sokat, szükség van egy viszonyítási alapra, amit úgy kaphatunk meg, hogy először a szenzort teljesen száraz talajba dugjuk és leolvassuk a mért értéket. Miután ez megvan, ugyanezt meg kell csinálnunk egy teljesen nedves talajban is. Ha megvan ez a két érték, akkor akár egy

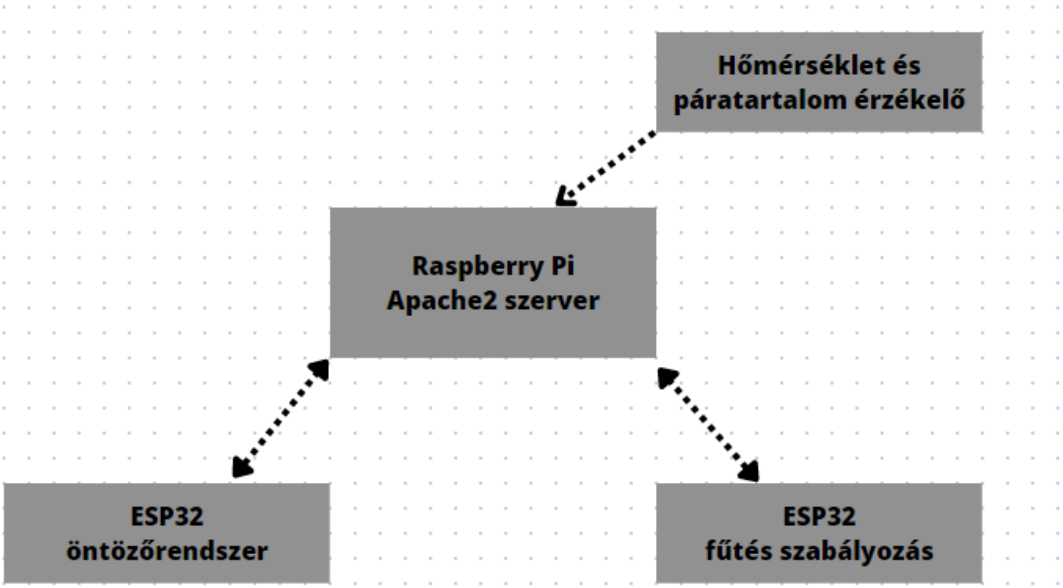
százalékos skálát is ki lehet alakítani ennek a két értéknek a segítségével, így meg lehet állapítani a talajnedvességet százalékban.

3.1.4. Zárt elektrotermikus működtető

A hőmérséklet szabályozásához egy Salus zárt elektrotermikus működtetőt használok. Ez az eszköz 230V váltófeszültséggel működő, 2W teljesítményű szabályzó eszköz, és egy radiátorra lehet szerelni. A szabályzó lehet nyitva, illetve zárva. Nyitott állapotban engedi a fűtést, zárt állapotban viszont nem. Az eszköz alapállapotban zárva van (NC - normally closed), ami azt jelenti, hogy hogyha nincs áram alatt, akkor zárva van és amint áramot adunk az eszköznek elkezd kinyitni. A nyitási illetve zárási idő kicsit hosszabb, körülbelül 2-3 percet vesz igénybe. A leírtakból látható, hogy ennek az eszköznek szüksége van valamilyen vezérlőre, azért, hogy a hőmérséklet függvényében nyisson, illetve zárjon. Erre szintén egy ESP32 típusú mikrovezérlőt használok, mint az öntözőrendszernél és az elv is hasonló. Mivel az ESP32 működéséhez 5V egyenfeszültséghez és az elektrotermikus működtető működéséhez 230V váltófeszültséghez van szükség, ezért kell egy relé, ami „kapcsolatot teremt” a két eszköz között.

3.2. Szoftver architektúra

Ahogyan azt a fentiekben is említettem, a felhasználói felületet egy beágyazott weboldal fogja biztosítani, ami egy helyi szerveren fut a Raspberry Pi-n. Innen lehetőség van irányítani a folyamatokat, beavatkozni a különféle irányításokba. Ezen kívül szükség volt az ESP32 mikrovezérlőkre is szoftvert fejleszteni annak érdekében, hogy üzeneteket tudjanak fogadni, illetve küldeni a központi vezérlőegységnek, valamint, hogy irányítani tudjuk a folyamatokat. Ahhoz, hogy a vezeték nélküli kommunikáció működjön, az eszközök közös hálózatra kell csatlakozzanak. Ahogy azt az 6. ábra is bemutatja, az eszközök vezeték nélkül kommunikálnak a központi egységgel, ami a Raspberry Pi. Az ESP32 mikrovezérlők és a Raspberry Pi között a kommunikáció kétoldalú, azaz mind a két fél küldhet üzenetet és fogadhat is üzenetet. A hőmérséklet és páratartalom szenzor csak adatokat küld a Raspberry Pi felé, nem szükséges adatokat fogadni a központi résztől.



6. ábra Szoftver tömbvázlata

3.3. Követelmény specifikáció

Az alábbiakban be fogom mutatni az alkalmazás pontos követelményeit, a funkcionális követelményeket, illetve a nem funkcionális követelményeket. A rendszer irányítható és elérhető egy beágyazott weboldalról, ami böngészőn keresztül érhető el.

3.3.1. Funkcionális követelmények

A funkcionális követelmények pontosan leírják az adott rendszer pontos működését, valamint azt, hogy ez a rendszer milyen funkciókat biztosít a felhasználók számára. Az általam készített alkalmazásnak csak egy típusú felhasználója van. Az alkalmazásban a következő funkcionalitások vannak:

- Hőmérséklet és páratartalom követése – a kezdőoldalon a felhasználónak lehetősége kell legyen az éppen aktuális hőmérsékleti és páratartalmi adatok követésére, amit tíz másodpercenként kell frissíteni, azáltal, hogy lekérjük az adatokat a szenzortól. A hőmérséklet és páratartalom szenzortól szolgáltatott adatokat minden órában menteni kell az adatbázisba.
- Statisztikák követése – amikor a felhasználó kiválaszt egy hónapot a weboldalon, akkor egy diagramot kell megjeleníteni az adott hónap hőmérsékleteivel. A hőmérséklet adatok naponként átlagolni kell és úgy kell megjeleníteni a diagramon.

- Növény öntözése automatikusan – a növények automatikus öntözését a felületen ki-illetve be lehet kapcsolni. Ha az automatikus öntözés be van kapcsolva, akkor a felhasználó kiválaszthatja, hogy éppen milyen növényt szeretne öntözni. Ilyenkor a vízpumpa automatikusan kapcsol be és ki.
- Vízpumpa ki-bekapcsolása – a felhasználónak lehetősége kell legyen csak a vízpumpát kapcsolni ki és be, manuálisan a felületről.
- Fűtés vezérlése – a felhasználó beállíthat egy hőmérsékletet 14 és 28 °C között, 0,5 °C-os lépésekkel. Ha a beállított érték meghaladja 0,2 °C-kal az aktuális hőmérsékletet, akkor a fűtés le kell álljon, ellenkező esetben el kell induljon.
- Redőny állapotának követése – a felületen kell legyen egy oldal, ahol a redőny aktuális állapotát lehet követni egy animáció segítségével. A redőny állapotát az aktuális időjárási körülmények befolyásolják az aktuális tartózkodási hely függvényében.

3.3.2. Nem funkcionális követelmények

A rendszer működéséhez szükség van egy Raspberry Pi számítógépre a szerver futtatásához, valamint a felhasználói felület megjelenítéséhez. A nem funkcionális követelmények ehhez a részhez a következőek:

- Raspbian operációs rendszer
- Python 3.5+
- Django 3.2+
- bluepy könyvtár
- paho-mqtt könyvtár
- pymongo könyvtár
- skfuzzy könyvtár

A fent említett nem funkcionális követelmények szükségesek ahhoz, hogy a Raspberry Pi-n futtatni tudjuk a rendszert működtető szervert. Ezek mellett még szükséges az is, hogy a rendszerhez csatlakozó összes eszköz egy hálózaton (azaz Wi-Fi-n) legyen, annak érdekében, hogy kommunikálni tudjanak egymással.

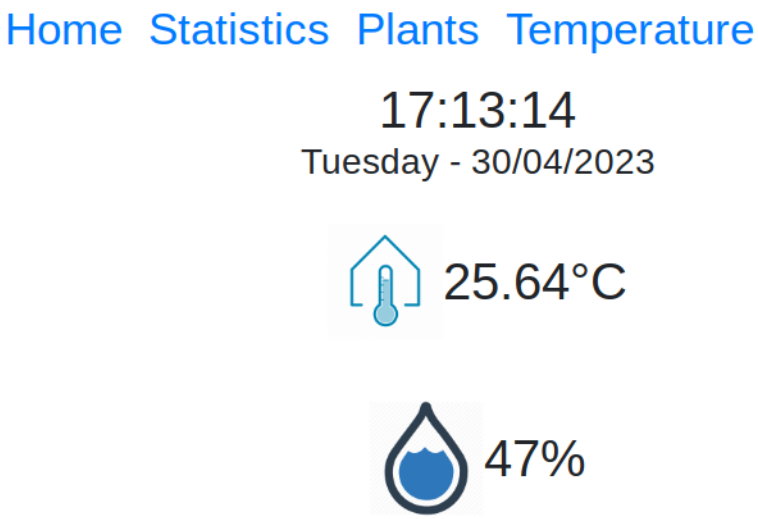
3.4. Webes felület bemutatása

A rendszer a felhasználóval a webes felületen keresztül kommunikál. Itt adatokat tudhat meg a felhasználó a rendszerrel kapcsolatban, mint például a hőmérséklet egy helyiségben vagy

a páratartalom szintén egy helyiségben. A weboldal elkészítéséhez a Django keretrendszert használtam és a fejlesztés a Raspberry Pi-n történt. A felület rendelkezik egy felső navigációs sávval, ahol a felhasználó tud navigálni az oldalak között érintések segítségével.

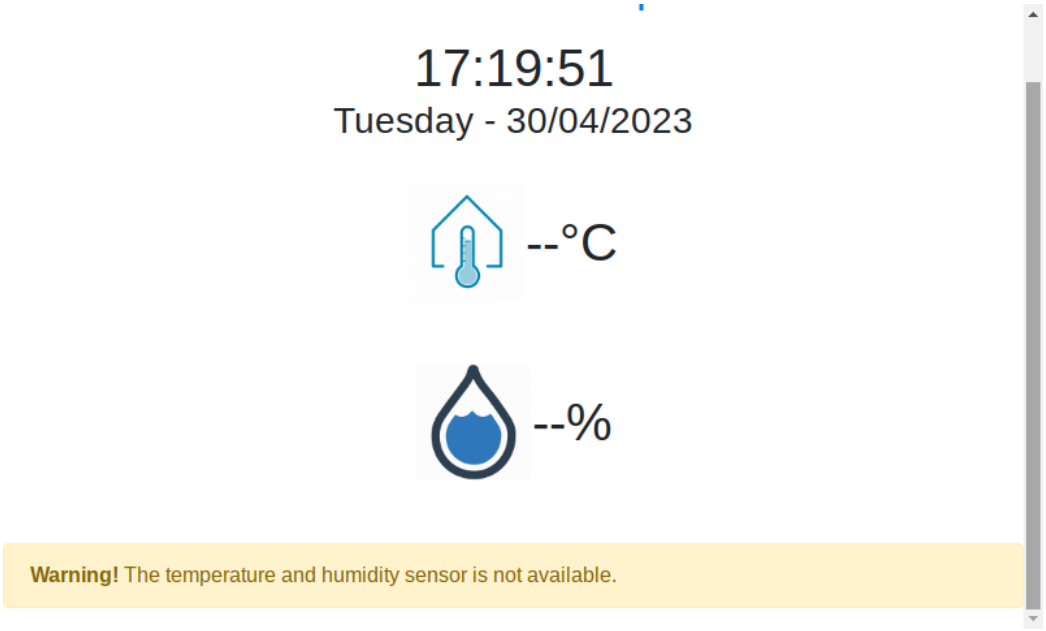
3.4.1. Home oldal

A Home oldal az alap oldal, amin megjelenik a jelenlegi dátum, a pontos idő, a jelenlegi hőmérséklet a lakásban és a páratartalom. Ez az oldal információközlő, nincs lehetőség interakcióba lépni a rendszerrel csak az adatokat tudjuk nyomon követni. Az adatokat BLE-n keresztül kapjuk a hőmérséklet és páratartalom szenzortól minden 10 másodpercben. Előfordulhat, hogy a szenzor éppen inaktív, mert lemerült az eleme vagy hatótávolságon kívül van (ami nagyjából 10 méter, de ez függ a helységtől és a falaktól). Amikor az eszköz nem érhető el és nem küld adatot, akkor a Home oldalon megjelenik egy sárga figyelmeztető üzenet erről a tényről. A Home oldalt az 7. ábra mutatja be.



7. ábra Home oldal

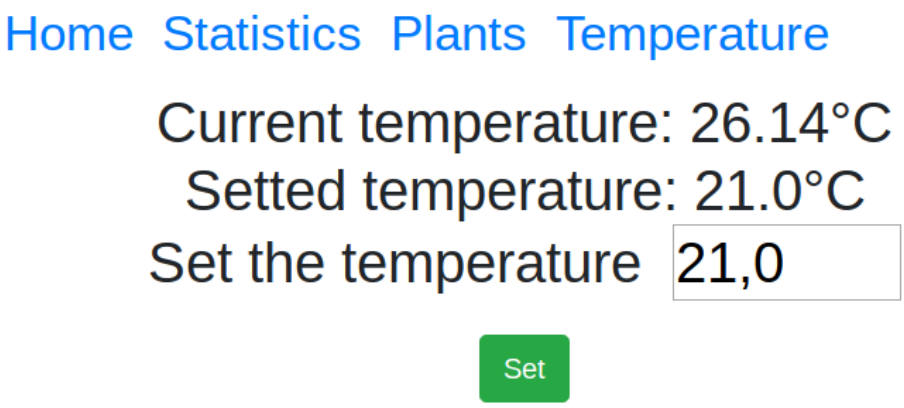
A 8. ábran látható a fent említett figyelmeztető jelzés. Ez azt jelzi, hogy az eszköz éppen nem tud adatokat szolgáltatni a központi rész felé valamilyen okból kifolyólag. Ilyenkor a hőmérséklet és a páratartalom helyén két vonal jelenik meg, amint a szenzor újból elérhető lesz, a felirat eltűnik és bekerülnek az adatok a vonalak helyére.



8. *ábra Home oldal figyelmeztetéssel*

3.4.2. Hőmérséklet szabályozás oldal

A hőmérséklet szabályozása oldalon lehetőség van egy kívánt hőmérsékletet beállítani a helyiségbe. Amikor beállítunk egy hőmérsékletet, akkor az elmentődik az adatbázisba. A rendszer folyamatosan figyeli a hőmérséklet szenzor által küldött adatokat. Amennyiben a hőmérséklet alacsonyabb, mint a beállított érték, a Raspberry Pi üzenetet küld az ESP32-nek, hogy indítsa el a fűtést. Amikor a hőmérséklet meghaladta a beállított értéket, akkor újabb üzenetet küld a Raspberry Pi az ESP32-nek. hogy állítsa le a fűtést. Ahogy az a 9. ábran látszik, az oldalon lehet látni az éppen aktuálisan beállított értéket és a jelenlegi értéket. Amennyiben módosítani szeretnék a beállított értéket, a beviteli mezőbe tudjuk állítani ezt 0,5 °C-os lépésekkel. Ha beállítottuk az értéket, a Set gombra kattintva el is mentettük a módosításokat.



9. ábra Hőmérséklet szabályzás

3.4.3. Növények öntözése oldal

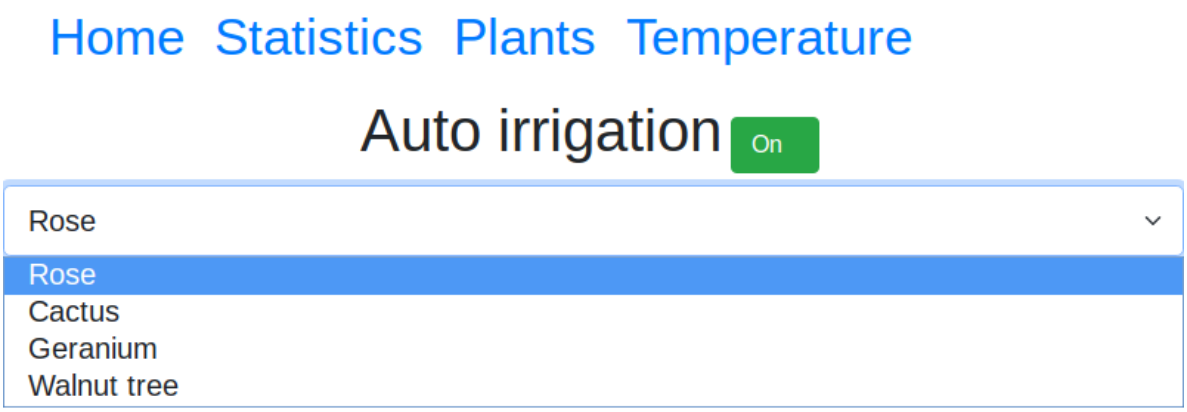
A növények öntözéséhez két opció közül választhatunk. Az egyik opció az, hogy a rendszerre bízzuk az öntözést, azaz bekapcsoljuk az automatikus öntözést. Ekkor a rendszer a talajnedvesség függvényében fogja öntözni a növényt. Mivel minden növénynek más és más mennyiségű vízre van szüksége öntözés során, ezért egy legördülő listából ki lehet választani a növény típusát. Ahogyan azt már említettem, a talajnedvesség szenzort kalibrálni kell. A dolgozat készítése során előre elvégeztem négy mérést négy különböző növényre. A mérés menete a következő volt: a talajnedvesség szenzort a teljesen száraz talajba tettem, amikor látszott, hogy a növénynek szüksége van az öntözésre és a szenzor által mért értéket leolvastam. Ezután megöntöztem a növényt és a szenzor által mért értéket ismét leolvastam. Ezeket a száraz, illetve nedves értékeket egy adatbázisba bevittem, és ha a felhasználó kiválaszt egy növényt a listából, az ESP32-nek egy üzenetet küld ezekkel az értékekkel és beállítódnak az öntözéshez a száraz, illetve nedves értékek. A mért értékek 1. táblázat soraiban és oszlopaiban találhatóak:

Növény neve	Száraz talajban mért érték (mV)	Nedves talajban mért érték (mV)
Rózsa	1785	1402
Kaktusz	2002	1280
Muskátli	1912	1296
Diófa csemete	1900	1325

1. táblázat Mért feszültségértékek

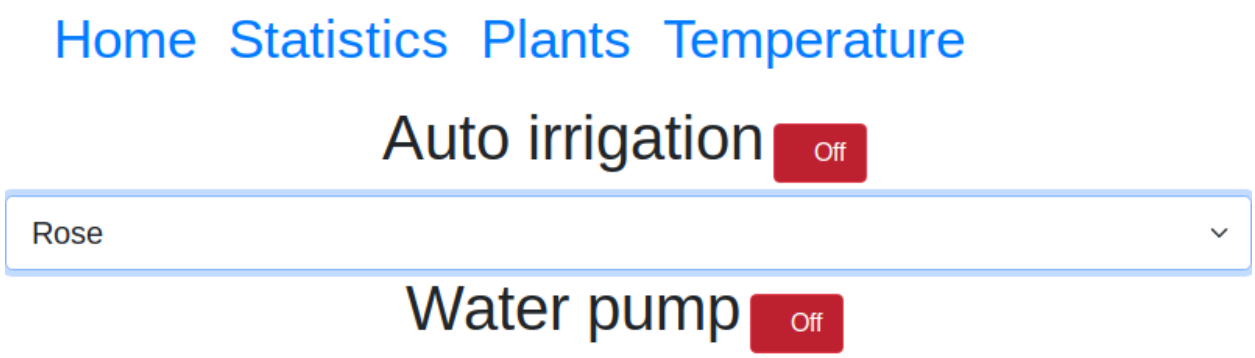
A szenzor analóg értékeket ad vissza, azaz a táblázatban lévő értékek nem mások, mint feszültségértékek mV-ban kifejezve.

A felhasználói felületen a felhasználónak lehetősége van a fenti növényekből választani egyet, amelynek az értékei beállítódnak az öntözéshez, ezeket lehet változtatni bármikor, ha az automatikus öntözés be van kapcsolva. Ezt a 10. ábra mutatja be.



10. ábra Automatikus öntözés

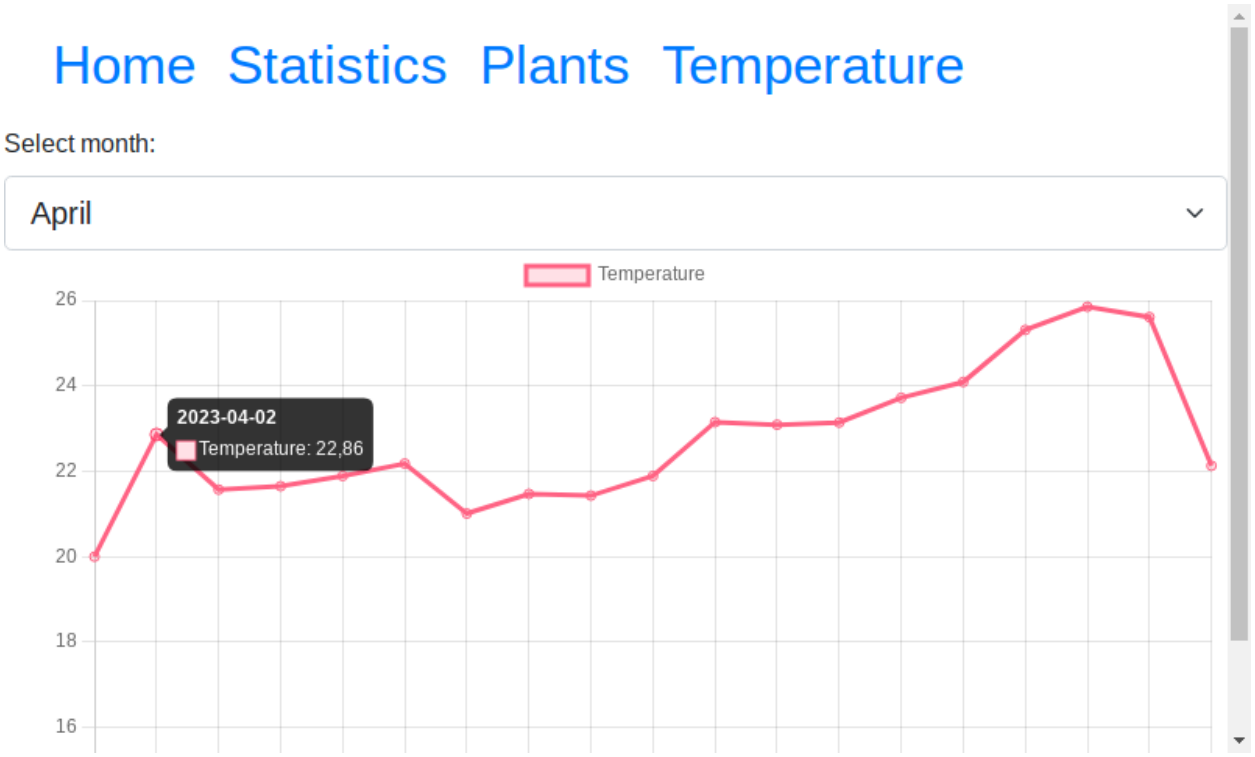
A második opció az öntözésre az a manuális öntözés. Ez annyiból áll, hogy a felhasználónak lehetősége van a felületről ki, illetve bekapcsolni a vízpumpát. Ekkora nem számít a talaj nedvessége, a pumpa addig fog öntözni, amíg a felhasználó a felületen keresztül le nem állítja azt. Ez a funkció egy szabadságot ad a felhasználónak, kezében érezheti az irányítást, illetve ha egy olyan növényt szeretne öntözni, ami nem szerepel a listán, akkor ennek a segítségével erre is lehetőség van. Ezt a felületet a 11. ábra mutatja be.



11. ábra Manuális öntözés

3.4.4. Statisztikák oldal

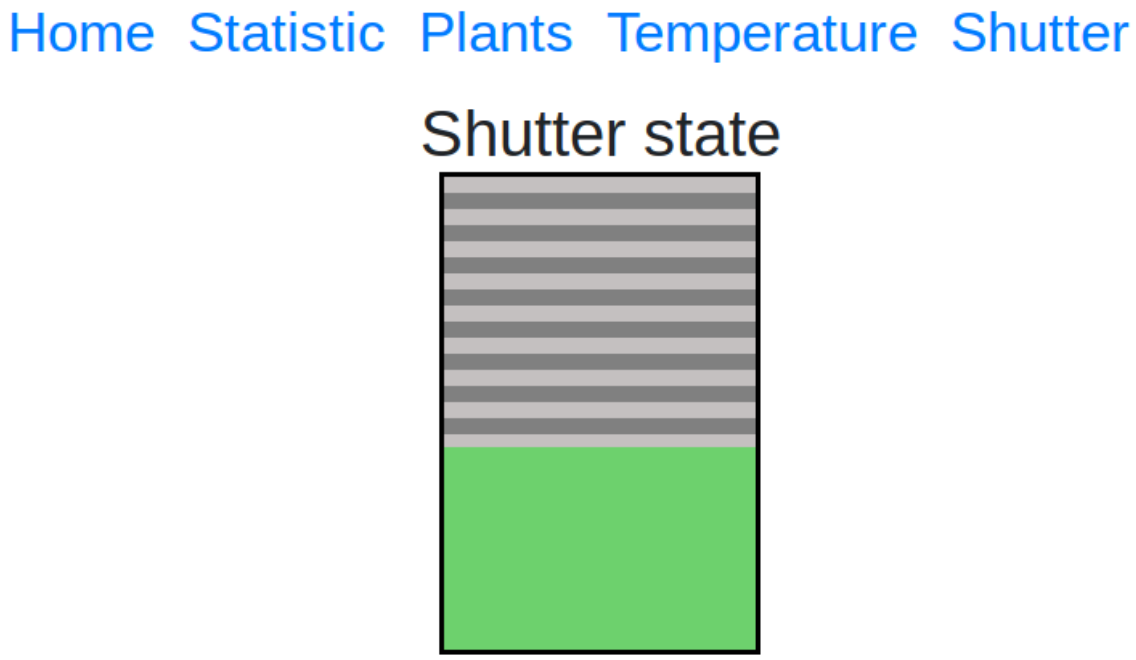
Az oldalon lehetőség van havi szinten figyelemmel követni a helységben lévő naponkénti átlaghőmérsékletet. Itt is látható egy legördülő lista a hónapokkal, itt ha kiválasztjuk a kívánt hónapot, akkor megjelenik egy diagram az adatokkal. A hőmérséklet minden órában rögzítődik automatikusan és elmentődik az adatbázisba. A diagram kirajzolásánál ezekből az elmentett értékekből számítódik egy átlag, ami megjelenik a diagramon. Ezt az oldalt a 12. ábra mutatja be.



12. ábra Statisztika oldal

3.4.5. Redőny állapota oldal

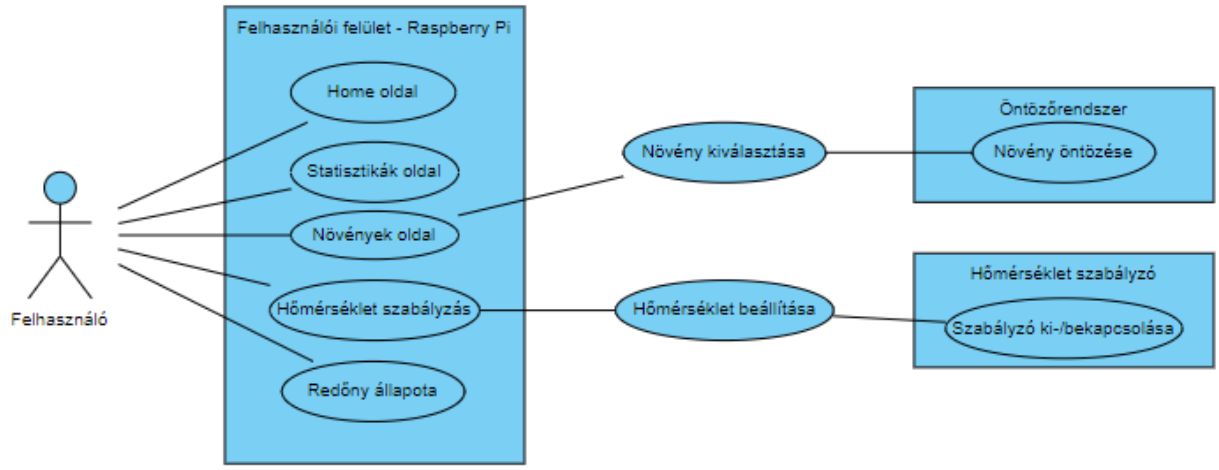
Ezen az oldalon a felhasználónak lehetősége van a redőny állapotát követni egy animáción keresztül, amit az 13. ábra mutat be. Az oldal a szervertől kap egy százalékos értéket, amit a fuzzy logika segítségével számol ki a szerver a benti hőmérséklet, a kinti hőmérséklet és az időjárás előrejelzés függvényében. Ez a százalék jelenti a redőny állapotát, azaz, hogy a redőny pontosan hány százalékra kell legyen leengedve az ablakon. Ezt szemlélteti az animáció is, ezt lehet figyelemmel követni az oldalon. Az 13. ábra esetében a fuzzy logika által kiszámolt érték 57%. Ezt szemlélteti az animáció, hogy a redőny ennyi százalékban kell leengedve legyen.



13. ábra Redőny állapotát jelző oldal

4. Részletes tervezés

Első lépésben, alkatrészek hiánya miatt, a tervezést a szoftverrel kezdtem, azon belül is a felhasználói felülettel, ami a weboldalt jelenti. Ezt követte a szükséges áramkörök tervezése, majd megépítése. A 14. ábra mutatja be a rendszer használati eset diagramját. Ezen megfigyelhető a rendszer egyszerű felhasználása a felhasználói felületen keresztül.



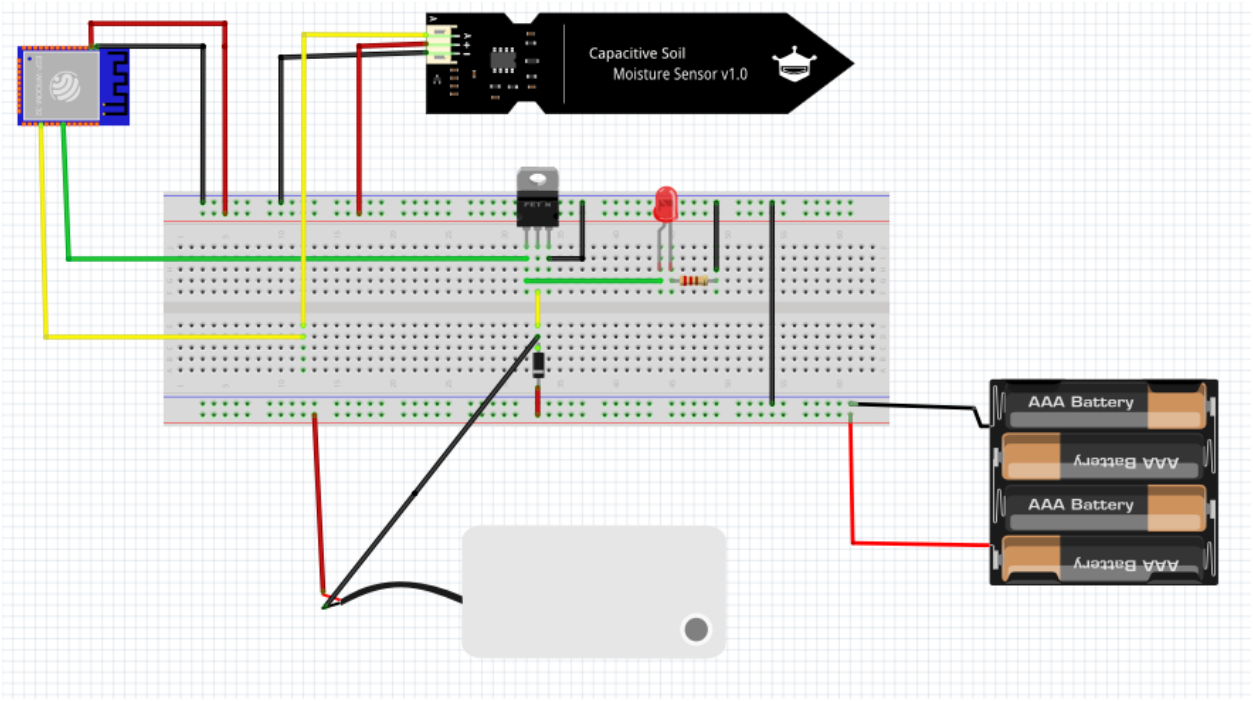
14. ábra A rendszer használati eset diagramja

4.1. Hardver tervezés

A hardver tervezésnél első lépésként Fritzingbe terveztem meg az áramkört, majd csak ezt követően építettem meg fizikailag őket.

4.1.1. Öntöző megvalósítása ESP32-vel

Az öntözőrendszernek a fő részét egyértelműen a mikrovezérlő alkotja, aminek a tápellátása 5V és microUSB-n keresztül történik. Mivel ennek a mikrovezérlőnek a kimeneti pinje nem képesek akkora áramot szolgáltatni, ami elindítaná a vízpumpát, ezért szükség van egy külső áramforrás használatára egy MOSFET tranzisztoron keresztül. Külső áramforrásnak négy darab 1,5V feszültségű elemet használok sorba kötve. Ez körülbelül 6V feszültséget tud szolgáltatni a pumpa működéséhez. Az öntözőrendszer kapcsolási rajza Fritzingbe a 15. ábran látható.

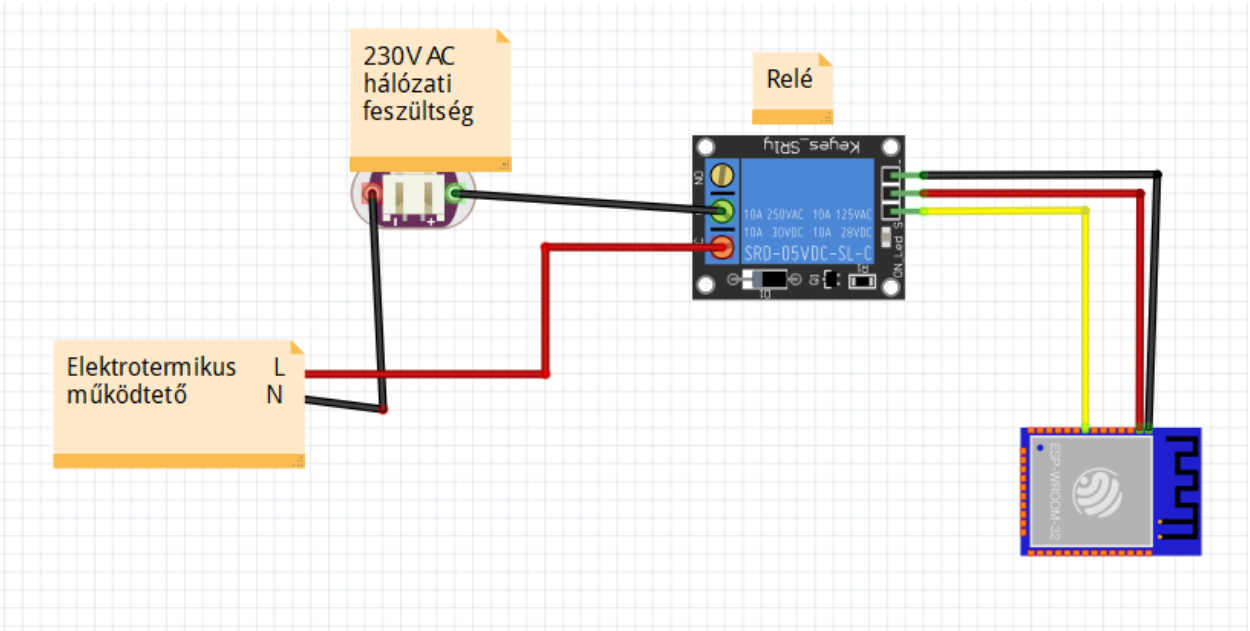


15. ábra Öntöző kapcsolási rajza

A vízpumpa ki és bekapcsolásához egy IRLZ44N MOSFET-et használtam. Azért esett erre a választás, mert ennek a MOSFET-nek a Gate feszültsége megfelelő ahhoz, hogy az ESP32 viszonylag alacsony 3,3V kimeneti értékre is nyisson és működtetni tudja a vízpumpát. A tranzisztor Gate lábát az ESP32 kimenetére kell kötni, ahonnan a vezérlőjelet kapjuk, a Drain lábát az áramforrás pozitív részére egy diódán keresztül, a Source lábat pedig az ESP32 negatív részére. A Drain lábnál szükség van egy diódára a motorral párhuzamosan. A dióda az áramot csak az egyik irányba engedi, ezzel egy védelemként szolgál az áramkörben. A 15. ábran látható egy piros led, ez akkor világít, amikor a motor működik. Tesztelésnél volt ez nagyon hasznos, mert így láttam, hogy a mikrovezérlőre feltöltött forráskóddal van probléma vagy esetleg valami mással.

4.1.2. Zárt elektrotermikus működtető megvalósítása

A fűtés szabályzására egy 230V hálózati feszültséggel működő eszközt használtam. Ebben az esetben az áramkört úgy kellett kialakítani, hogy egy vezérlőjel szerint működjön a zárt elektrotermikus működtető. Ehhez egy relét kellett használnom. Ennek a rendszernek a kapcsolási rajza a 16. ábran látható.



16. ábra Zárt elektrotermikus működtető

4.2. Szoftver tervezés

A szoftver tervezése során különböző programozási nyelvekben kellett elmélyülnöm, amelyeket már részben ismertem. A fejlesztéshez használtam Python, C++, JavaScript, HTML nyelveket. Külön szoftverrel rendelkezik mind a két ESP32, illetve külön szoftvere van a Raspberry Pi-n futtatott felhasználói felületnek.

4.2.1. Webes felület megvalósítása

Ebben a fejezetben egy oldal megvalósítását fogom bemutatni Django keretrendszerben, az oldalak elkészítésénél ezt a struktúrát követtem. A Django MVT architektúrára épül, azaz különválasztja az adatbázist, a logikát valamint a megjelenítést egymástól. Ez a következőképpen néz ki:

- Model – ez felel az adatok reprezentációjáért, valamint az adatbázis kezelésért. A modellek az adatbázis tábláinak felelnek meg. A hőmérsékletek tárolására a következő Django modellt hoztam létre:

```
class Temperature(models.Model):
    id = models.AutoField(primary_key=True)
    value = models.FloatField()
    unit = models.CharField(max_length=3)
    recorded_at = models.DateTimeField(auto_now_add=True)
```

- View – a view kezeli a kéréseket és a válaszokat, igazából ez valósítja meg az oldal logikáját. Ezek kis függvények, amelyek adatokat dolgoznak fel. A lenti kódrészlet a növényeket kéri le az adatbázisból, majd adja tovább a template-nek.

```
def plants(request):
    plants = Plant.objects.all()
    return render(request, 'plants.html', {'plants': plants})
```

- Template – ezek sablonok, amelyekben a HTML kódok találhatóak. A felhasználói felület megjelenítésért felel. A fenti kódrészletben a 'plants.html' a sablon, ami megjelenítődik az oldalon az átadott paraméterekkel együtt.

Ha egy oldalt szeretnénk megjeleníteni, akkor a fenti lépéseket kell követnünk, ezek után az elérési URL-t kell beállítani, amelyhez hozzá kell rendelnünk a view-t. Minden URL-hez hozzárendelhető egy view függvény, amely feldolgozza az adatokat a kérés után. Ezt a struktúrát követtem minden oldal megvalósításánál.

4.2.2. Hőmérséklet és páratartalom kiolvasása

Ahogyan azt már a fentiekben írtam, a kommunikációra, ennek a szenzornak az esetében, BLE-t használok. A Függelék 1. pontban megtalálható a teljes kódrészlet, ahol látszik, hogy készítettem egy XiaomiTempHumidity nevű osztályt és ennek a konstruktorában állítom be a kiolvasáshoz szükséges adatokat, mint például az eszköz MAC címét. Itt a bluepy csomagból használom a Peripheral osztályt. Ez a bluepy csomag lehetővé teszi a Python fejlesztők számára, hogy kapcsolódni tudjanak BLE eszközökhöz. A Függelék 1. pontban látható a read_temp() metódus, itt történik a küldött adatoknak a dekódolása, aszerint, ahogy azt a 3.1.2 fejezetben már leírtam. Az eszközhöz való csatlakozást szükséges egy try except blokkba rakni, mert előfordulhat, hogy a csatlakozás az eszközhöz nem sikerült. Ekkor nem kellene leálljon a program egy hibával, ezért szükséges a try except blokk használata a csatlakozáshoz.

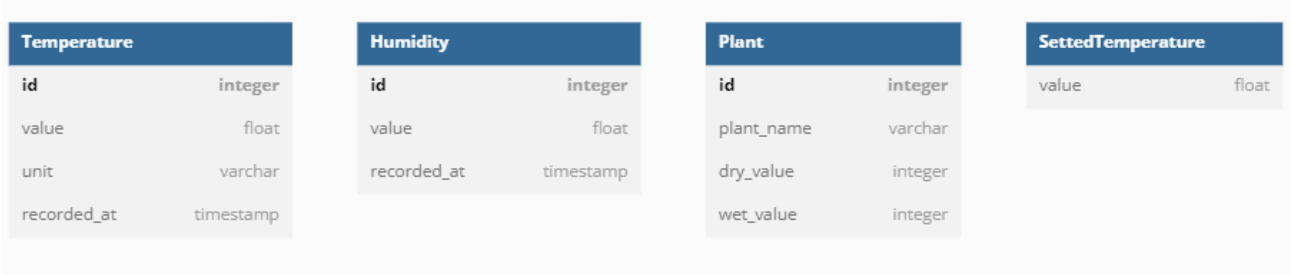
Az említett értékeket egy view függvényen keresztül továbbítjuk a Home oldalnak, ami majd megjeleníti őket, ahogy azt a Függelék 2. pontba látható. Amennyiben a kapcsolódás valamilyen okból kifolyólag nem sikerül, az oldalra „--“ érték jelenik meg. Erre azért van szükség, mert nem állíthatjuk le a program futását, ha nem sikerült csatlakozni a szenzorhoz.

Mivel a Home oldalon ezek az értékek folyamatosan kell frissüljenek, anélkül, hogy a felhasználó frissítené az oldalt, ezért Ajax kérést használok ennek a megoldására. Az Ajax (Asynchronous JavaScript and XML) egy webfejlesztési technika, amely lehetővé teszi a böngésző és a szerver közötti aszinkron kommunikációt. Ez az Ajax kérés JavaScript nyelven

íródott és a Függelék 3. pont mutatja be. Ez a kérés automatikusan hívódik minden 10 másodpercben, amit a következő sor hajt végre: setInterval(refreshTempAndHumidity, 10000);

4.2.3. Adatbázis

A rendszernek több adatot is el kell tárolnia, ezért szükséges volt egy adatbázist használtam. Első sorban, a Django keretrendszer által szolgáltatott adatbázist használtam, ahol négy darab táblám van: egy tábla a hőmérsékletek tárolására, egy tábla a páratartalom tárolására, egy tábla a rögzített növények tárolására és egy tábla a beállított hőmérséklet tárolására. A táblák mezőit az 17. ábra mutatja be.



17. ábra Adatbázisban lévő táblák

A hőmérséklet és a páratartalom esetében fontos eltárolni a rögzítés idejét, annak érdekében, hogy majd a statisztikák megjelenítésénél tudjunk havi statisztikákat készíteni. A növények esetében a mért dry és wet érték van eltárolva, amik az 1. táblázatban láthatóak. Ezeket a táblákat Django Modelek segítségével implementáltam.

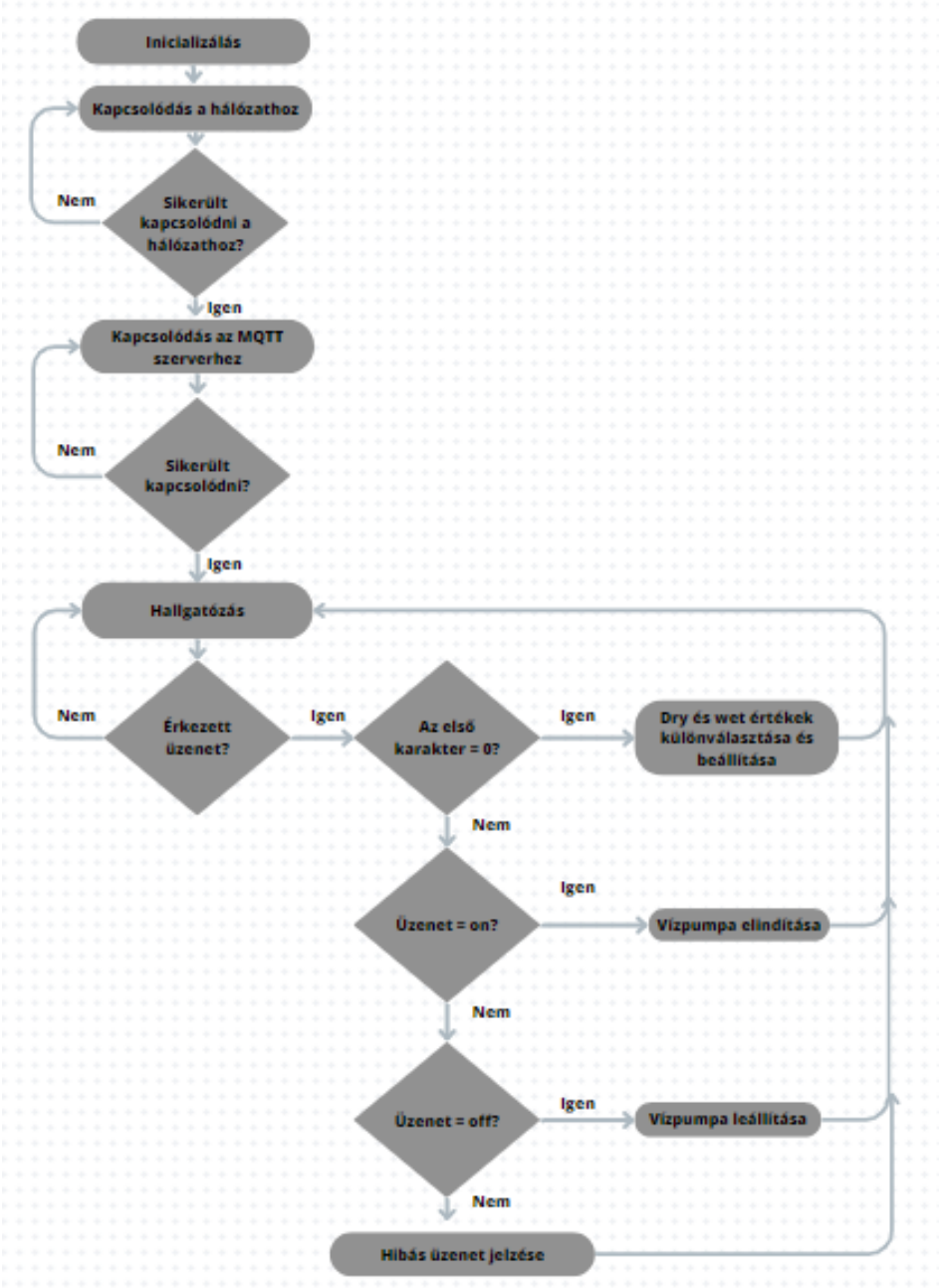
A fent említett adatbázist használtam a fejlesztés első felében, majd utána áttértem egy felhőalapú adatbázisra. Itt MongoDB-t használtam, ami lehetővé teszi, hogy az adatbázist ne lokálisan, hanem a felhőben tároljuk. Ennek egyik nagy előnye, hogy így a rögzített adatok bárhonnan elérhetőek. A MongoDB egy dokumentumorientált adatbázist (NoSQL adatbázis) jelent. A nem relációs adatbázisok olyan adattárolási megoldásokat jelentenek, amelyek eltérnek a hagyományos relációs adatbázisok által alkalmazott táblázatos struktúrától, az adatokat JSON formátumban tárolják. Azért, hogy adatokat tudjunk feltölteni, illetve lekérni a felhőből, telepíteni kell a Raspberry Pi-re a PyMongo-t, ezután egy ugynevezett Connection Stringre van szükségünk, ami a felhővel való kapcsolódásért felel, és a következőképpen néz ki:

```
CONNECTION_STRING="mongodb+srv://user:pass@cluster.mongodb.net/SmartHome"
```

Ezek után, a 17. ábran látható struktúra szerint építettem fel a dokumentumokat.

4.2.4. Öntözőrendszerhez fejlesztett szoftver

Az öntözőrendszer esetében az ESP32 mikrovezérlőhöz kellett szoftvert fejleszteni. Ez C++ nyelven és Arduino IDE fejlesztői környezetben történt.



18. ábra Az öntözőrendszer szoftveres aktivitás diagramja

A 18. ábra mutatja be a mikrovezérlőre írt szoftver aktivitás diagramját. A központ résztől kapott üzenetek tartalmazzák a szükséges utasításokat. A hálózathoz való kapcsolódásra a *WiFi.h* header állományban található *WiFiClient* osztályt használta, az MQTT kliens kialakításához a *PubSubClient.h* headerben található *PubSubClient* osztályt.

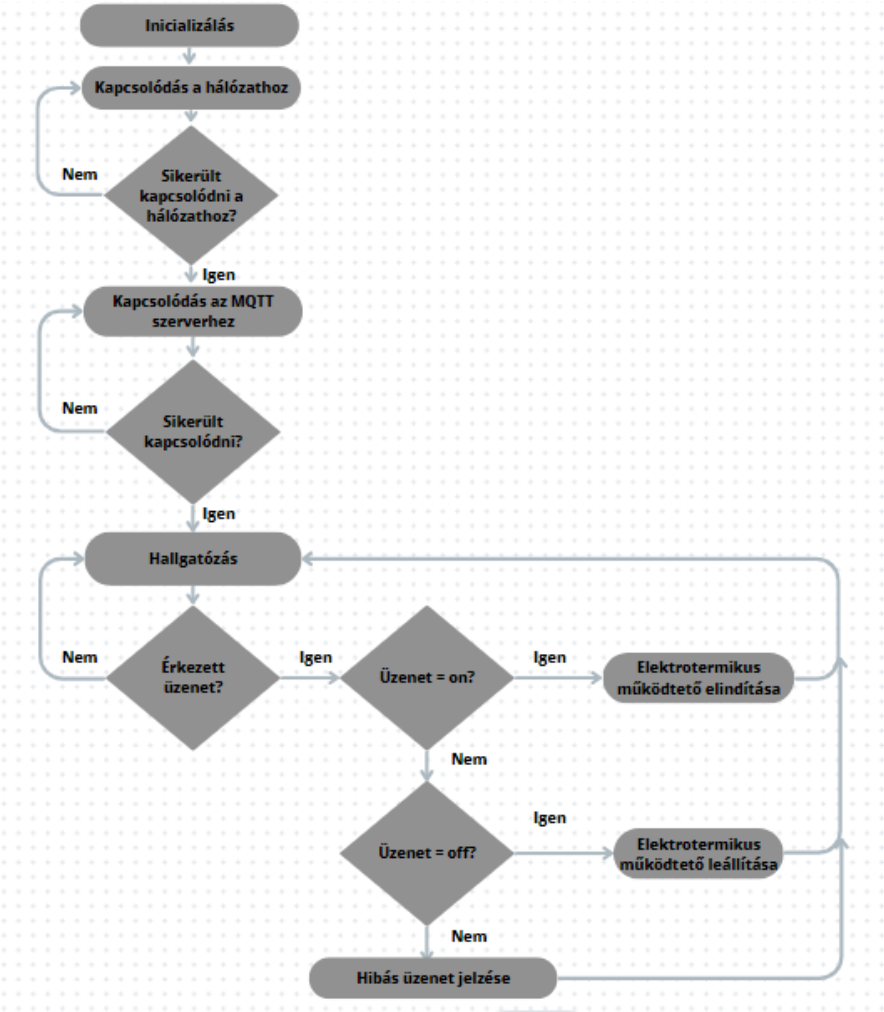
A mikrovezérlő többféle üzeneteket kaphat, és ezek a következőek:

- *on:* ez azt jelezi a mikrovezérlőnek, hogy a vízpumpát el kell indítani
- *off:* ez azt jelezi a mikrovezérlőnek, hogy a vízpumpát le kell állítani
- *auto:* ebben az esetben az *autoIrrigation* változót be kell állítani 1-es értékre. Ha ez a változó értéke 1, akkor a folyamatos figyelés közben a mikrovezérlő tudja, hogy a talajnedvesség szenzortól kapott érték függvényében kell kezelni a vízpumpát
- *manual:* az *autoIrrigation* változó értékét 0-ra kell állítani, ezzel jelezve, hogy az öntözés folyamatát a felhasználó kezeli
- *0[dry_ertek]|[wet_ertek]:* ha ez az üzenet érkezik, akkor a felhasználó kiválasztott a felületen egy növényt. Az ilyen típusú üzenetek 0 karakterrel kezdődnek, jelezve, hogy növénykiválasztás történt. Ebben az esetben az üzenetről a 0 karaktert le kell vágni és a maradék szöveget két részre kell vágni a | karakter szerint. Például, ha az üzenet 01785|1402, akkor a beállítandó dry érték 1785 és a beállítandó wet érték 1402.

A kód e részét a Függelék 4. pont mutatja be. A fent említett üzenetek két topicon keresztül érkeznek, az egyik a *water_pump* topic, a másik a *plant_type* topic.

4.2.5. Hőmérséklet szabályzáshoz fejlesztett szoftver

A hőmérséklet szabályzására szintén egy másik ESP32 típusú mikrovezérlőhöz kellett szoftvert fejleszteni, ami csatlakozik a hálózathoz és az MQTT szerverhez annak érdekében, hogy vezérelje az elektrotermikus működtetőt. Ennek a szoftvernek az aktivitás diagramját a 19. ábra mutatja be. A csatlakozást az MQTT szerverhez a Függelék 5. pontba található, ahol látható a kódrészlet. A csatlakozás folyamatosan történik, amíg nem jött létre a kapcsolat. Amennyiben a csatlakozás nem sikeres, 2 másodpercenként az ESP32 újra fog próbálkozni.



19. ábra A hőmérséklet szabályozás aktivitás diagramja

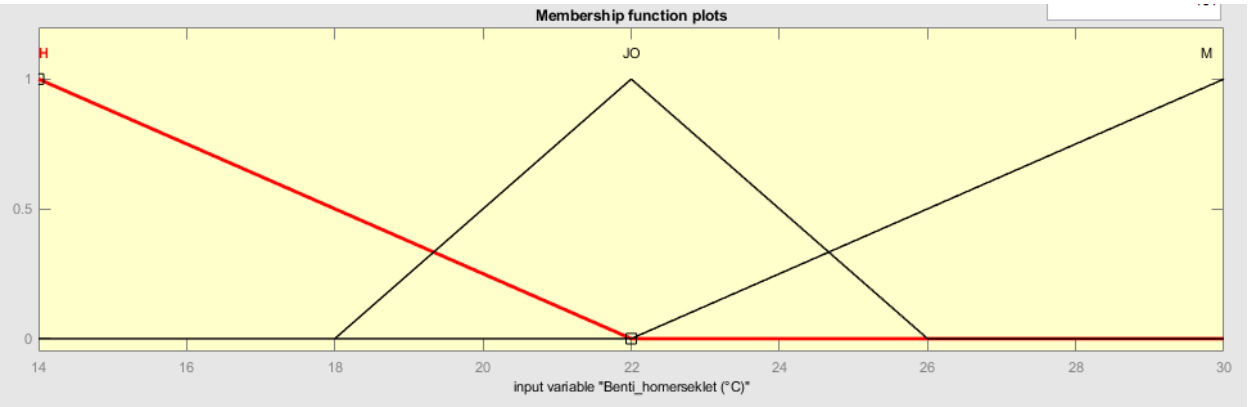
4.2.6. Redőnyök szabályzása fuzzy logikával

A rendszer rendelkezik egy fuzzy döntéshozóval, ami a redőnyök mozgatásáért felel. A redőnyt egy PWM jellel vezérelhetjük, a következőképpen: az említett PWM jel kitöltési tényezője 0%, akkor a redőny teljesen felhúzott állapotban van, ha a kitöltési tényező 100%, akkor a redőny teljesen leengedett állapotban van. Tehát ez lesz a fuzzy rendszer kimenete, illetve a rendszernek három bemenete van:

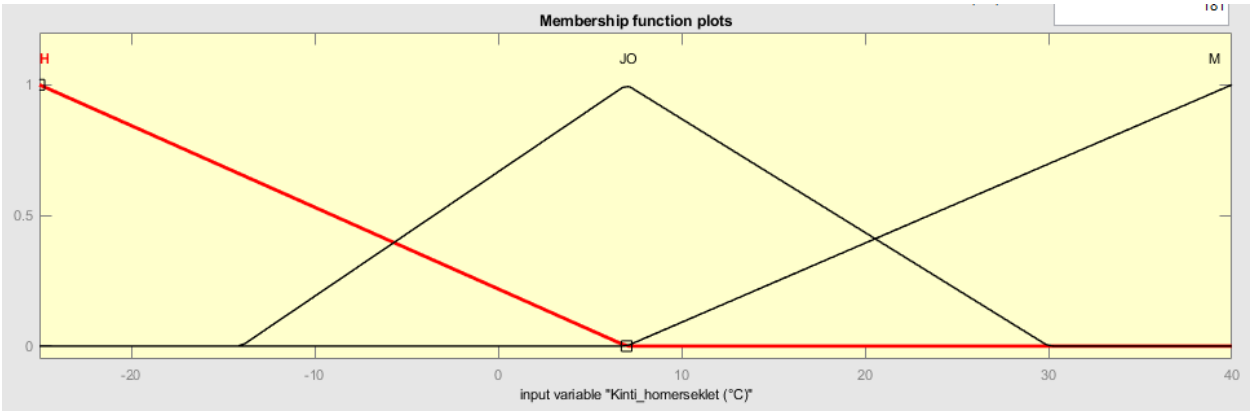
- a helységben lévő belső hőmérséklet: ezt az adatot a fent említett hőmérséklet és páratartalom érzékelő fogja szolgáltatni. Az eszköz 0-50°C tartományban képes mérni.
- kinti hőmérséklet: ezt az adatot az OpenMeteo API fogja szolgáltatni

- időjárás előrejelzés: szintén az OpenMeteo fogja szolgáltatni, az értéke a WMO (Weather interpretation codes) [11] kód szerint fog változni, ami az időjárási körülményeket egy 0-tól 99-ig terjedő skálán osztályozza.

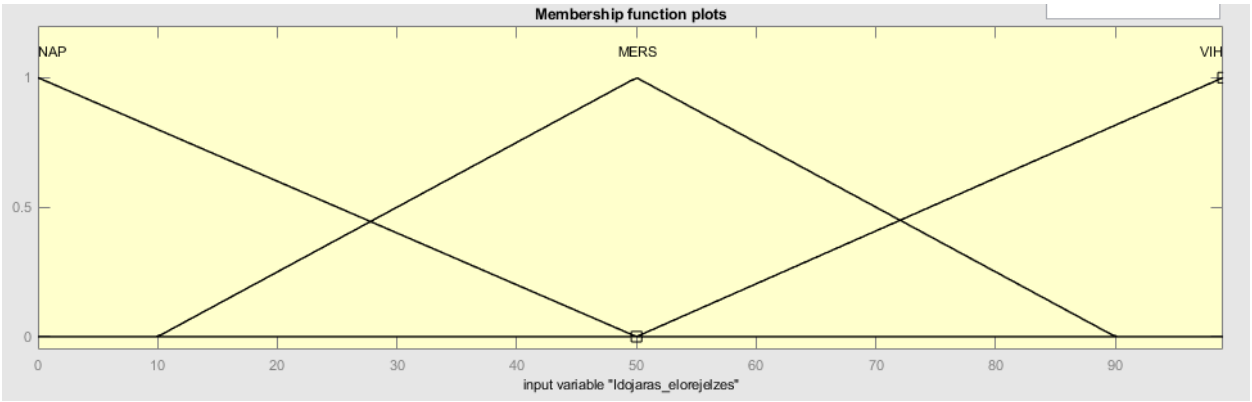
A fent említett bemeneti univerzumokat le kell fedni tagsági függvényekkel, amelyeket a 20. ábra 21. ábra 22. ábra, mutatja be, illetve a kimenetnek a tagfüggvényét a 23. ábra mutatja be.



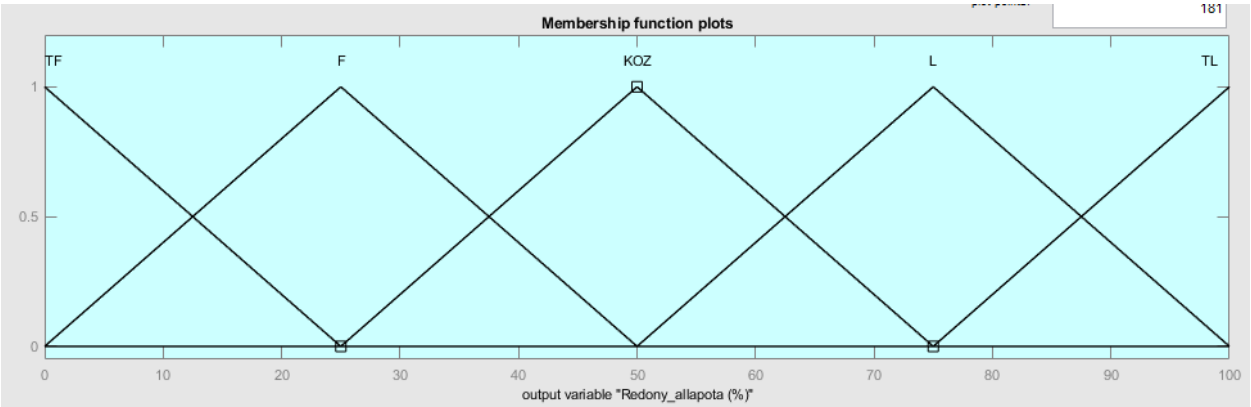
20. ábra Benti hőmérséklet tagfüggvénye



21. ábra Kinti hőmérséklet tagfüggvénye



22. ábra Időjárás előrejelzés tagfüggvénye



23. ábra Redőny állapotának tagfüggvénye

A fenti ábrákon alkalmazott nyelvi változóknak a következő a jelentésük az egyes ábrák esetében:

- Benti, illetve kinti hőmérséklet nyelvi változói (*20. ábra*, *21. ábra*):
  - H – hideg
  - JO – megfelelő hőmérséklet
  - M – meleg
- Időjárás előrejelzés esetében alkalmazott nyelvi változók (*22. ábra*)
  - NAP – alacsony
  - MERS – mérsékelt
  - VIH – viharos
- A kimenet esetében a redőny állapotára vonatkozó nyelvi változók (*23. ábra*):
  - TF – teljesen felhúzva
  - F – felhúzva
  - KOZ – középen
  - L – leengedve
  - TL – teljesen leengedve

A szabálybázist a 2.táblázat mutatja be, ami meghatározza a szabályokat a tagsági függvények alapján.

Benti hőmérséklet	Kinti hőmérséklet	Időjárás előrejelzés	Redőny állapota
H	H	NAP	TF
H	H	MERS	TF
H	H	VIH	TF
H	JO	NAP	F
H	JO	MERS	F
H	JO	VIH	TF
H	M	NAP	F
H	M	MERS	F
H	M	VIH	F
JO	H	NAP	KOZ
JO	H	MERS	F
JO	H	VIH	F
JO	JO	NAP	KOZ
JO	JO	MERS	KOZ
JO	JO	VIH	F
JO	M	NAP	L
JO	M	MERS	L
JO	M	VIH	F
M	H	NAP	F

M	H	MERS	F
M	H	VIH	F
M	JO	NAP	L
M	JO	MERS	L
M	JO	VIH	F
M	M	NAP	TL
M	M	MERS	TL
M	M	VIH	KOZ

2. *táblázat Szabálybázis*

A fenti ábrák alapján felírhatóak a bemeneti univerzumra a tagsági függvények egyenletei:

A benti hőmérséklet esetében:

$$\mu H(x) = \begin{cases} 1, & ha\ x \le 14 \\ \dfrac{22-x}{22-14}, & ha\ 14 < x < 22 \\ 0, & ha\ x \ge 22 \end{cases}$$

$$\mu JO(x) = \begin{cases} 0, & ha\ x \le 18 \\ \frac{x-18}{22-18}, & ha\ 18 < x \le 22 \\ \frac{x-22}{26-22}, & ha\ 22 < x \le 26 \\ 0, & ha\ x > 26 \end{cases}$$

$$\mu M(x) = \begin{cases} 0, & ha\ x \le 22 \\ \dfrac{30-x}{30-22}, & ha\ 22 < x < 30 \\ 1, & ha\ x \ge 30 \end{cases}$$

A kinti hőmérsékletek esetében az egyenletek:

$$\mu H(x) = \begin{cases} 1, & ha\ x \le -25 \\ \dfrac{7-x}{7-(-25)}, & ha\ -25 < x < 7 \\ 0, & ha\ x \ge 7 \end{cases}$$

$$\mu JO(x) = \begin{cases} 0, & ha\ x \leq -14 \\ \frac{x-(-14)}{7-(-14)}, & ha -14 < x \leq 7 \\ \frac{x-7}{30-7}, & ha\ 7 < x \leq 30 \\ 0, & ha\ x > 30 \end{cases}$$

$$\mu M(x) = \begin{cases} 0, & ha\ x \leq 7 \\ \frac{40-x}{40-7}, & ha\ 7 < x < 40 \\ 1, & ha\ x \geq 40 \end{cases}$$

Az időjárás előrejelzés esetében az egyenletek:

$$\mu NAP(x) = \begin{cases} 1, & ha\ x \leq 0 \\ \frac{50-x}{50-0}, & ha\ 0 < x < 50 \\ 0, & ha\ x \geq 50 \end{cases}$$

$$\mu MERS\ (x) = \begin{cases} 0, & ha\ x \leq 10 \\ \frac{x-10}{50-10}, & ha\ 10 < x \leq 50 \\ \frac{x-50}{90-50}, & ha\ 50 < x \leq 90 \\ 0, & ha\ x > 90 \end{cases}$$

$$\mu VIH(x) = \begin{cases} 0, & ha\ x \leq 50 \\ \frac{99-x}{99-50}, & ha\ 50 < x < 99 \\ 1, & ha\ x \geq 99 \end{cases}$$

A kimenet esetében a következő egyenleteket írhatjuk fel:

$$\mu TF(x) = \begin{cases} 1, & ha\ x \leq 0 \\ \frac{25-x}{25-0}, & ha\ 0 < x < 25 \\ 0, & ha\ x \geq 25 \end{cases}$$

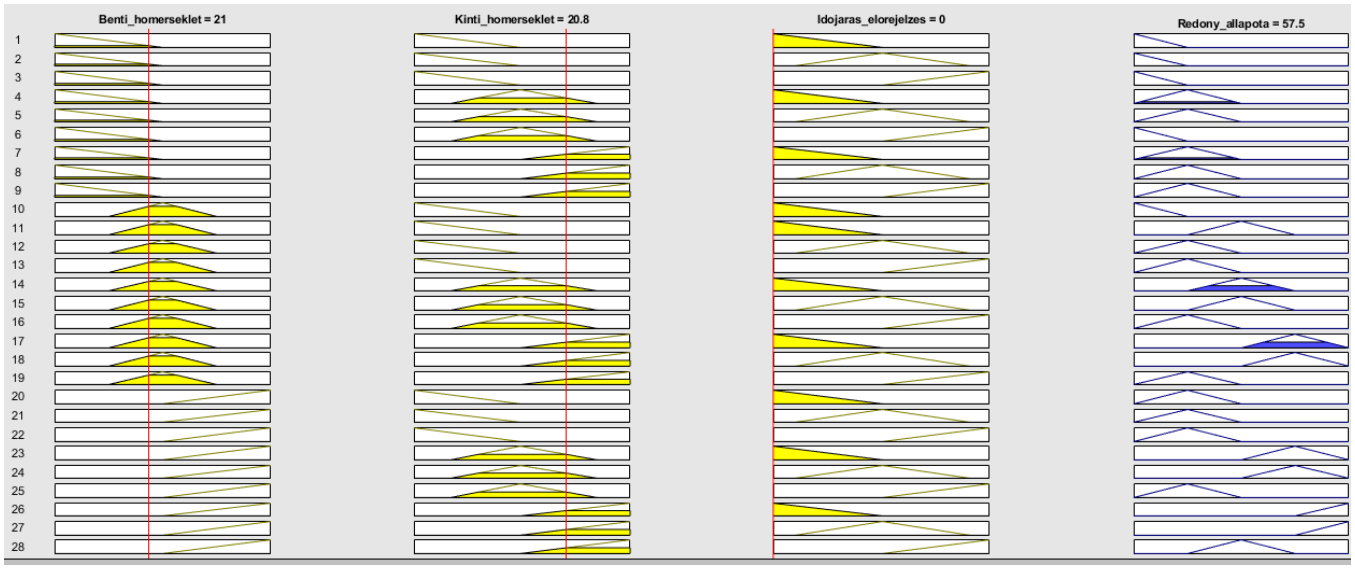
$$\mu F\ (x) = \begin{cases} 0, & ha\ x \leq 0 \\ \frac{x-0}{25-0}, & ha\ 0 < x \leq 25 \\ \frac{x-25}{50-25}, & ha\ 25 < x \leq 50 \\ 0, & ha\ x > 50 \end{cases}$$

$$\mu KOZ\ (x) = \begin{cases} 0, & ha\ x\ \leq 25 \\ \frac{x-25}{50-25}, & ha\ 25 < x\ \leq 50 \\ \frac{x-50}{75-50}, & ha\ 50\ < x\ \leq 75 \\ 0, & ha\ x\ > 75 \end{cases}$$

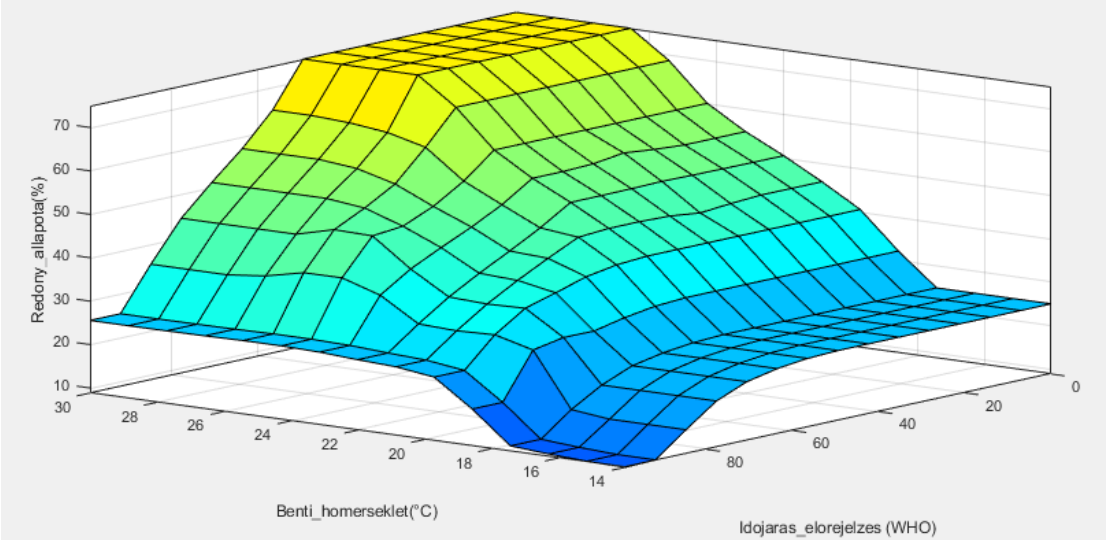
$$\mu L\ (x) = \begin{cases} 0, & ha\ x\ \leq 50 \\ \frac{x-50}{75-50}, & ha\ 50 < x\ \leq 75 \\ \frac{x-75}{100-75}, & ha\ 75\ < x\ \leq 100 \\ 0, & ha\ x\ > 100 \end{cases}$$

$$\mu TL(x) = \begin{cases} 0, & ha\ x\ \leq 75 \\ \frac{100-x}{100-75}, & ha\ 75\ < x\ < 100 \\ 1, & ha\ x\ \geq 100 \end{cases}$$

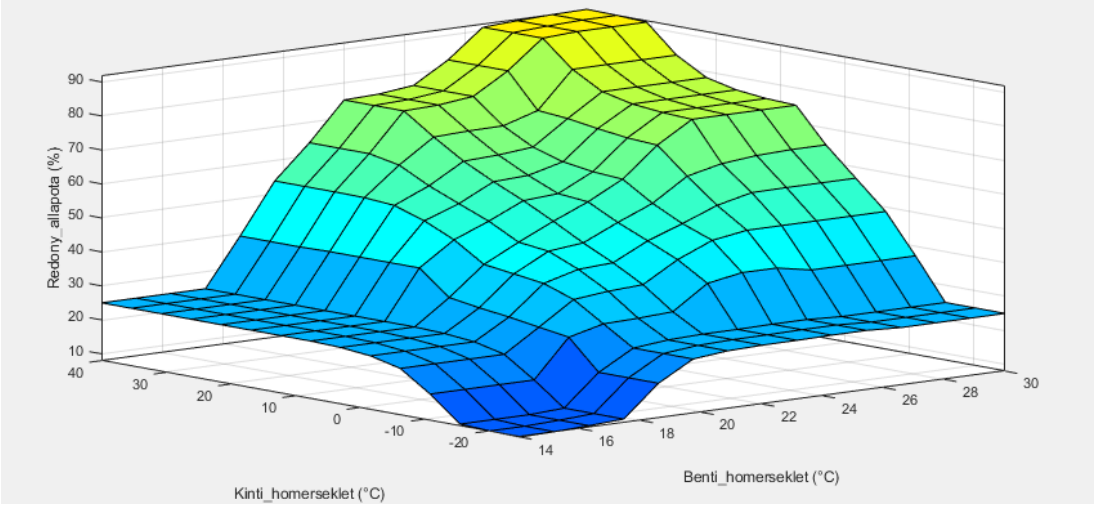
A fent tervezett rendszert Matlab-ban készítettem el, a Fuzzy Toolbox segítségével. Az 24. ábran láthatóak a meghatározott szabályok. Itt minden egyes bemenet külön állítható és figyelni lehet a rendszer által adott kimenetet az éppen adott bemenetekre. Az 24. ábran a kinti hőmérséklet 20,8°C, a benti 21°C, az időjárás előrejelzés kódja 0, ami azt jelenti, hogy tiszta és napos az égbolt. Ezekre a bemenetekre a rendszer válasza 57,5%, ami azt jelenti, hogy a redőnyt kicsivel lennebb kell engedni, mint félig.



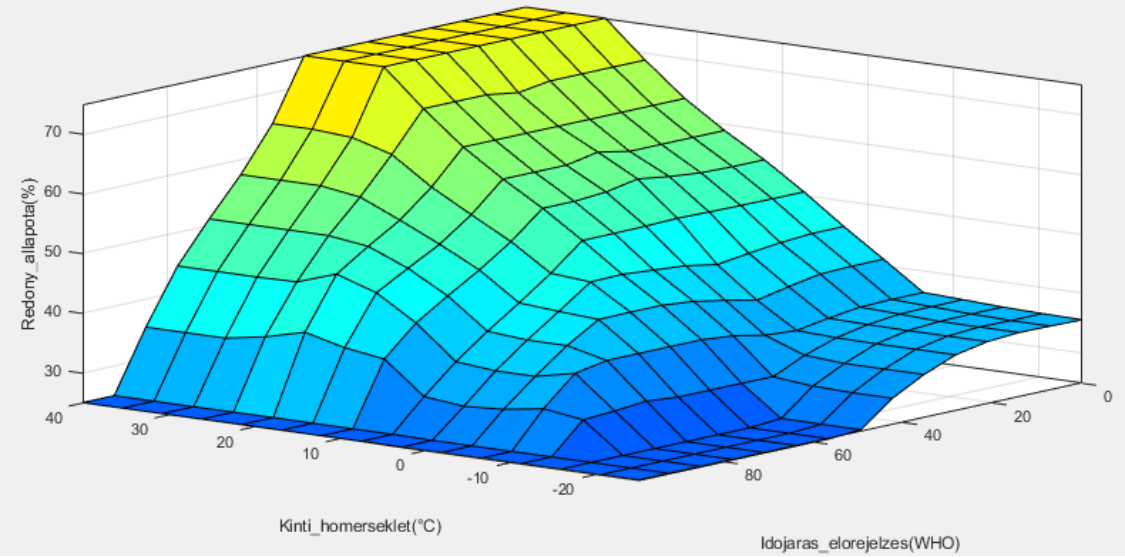
24. ábra Példa a rendszer kimenetére és bemenetére



25. ábra A redőny állapota a benti hőmérséklet és az időjárás előrejelzés függvényében



26. ábra A redőny állapota a kinti hőmérséklet és a benti hőmérséklet függvényében



27. ábra A redőny állapota a kinti hőmérséklet és az időjárás előrejelzés függvényében

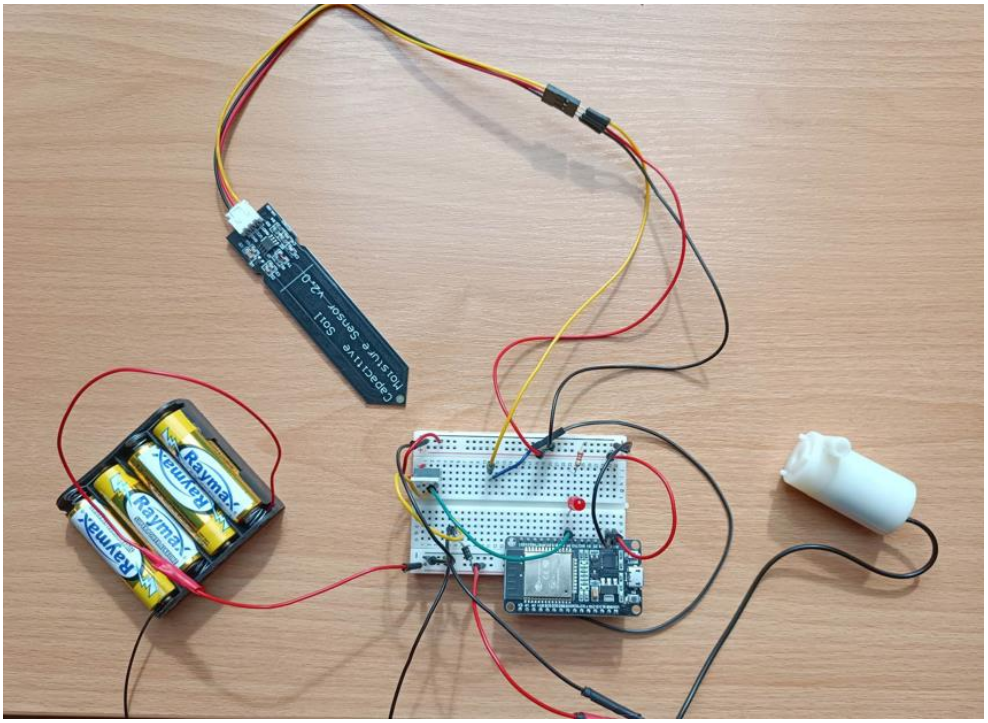
Az 25. ábra 26. ábra 27. ábra szemlélteti a rendszer kimenetét két bemenet függvényében. Ezekről az ábrákról leolvasható, hogy a megírt szabályok helyesek és a rendszer elvárt módon működik [12][13].

A Matlaban való tervezés és szimulációk után, szükség volt a fuzzy rendszer implementálása Python nyelven is, azért, hogy a fuzzy által kiszámolt redőny állapotát jelző értéket tudjuk használni a felhasználói felületen is.

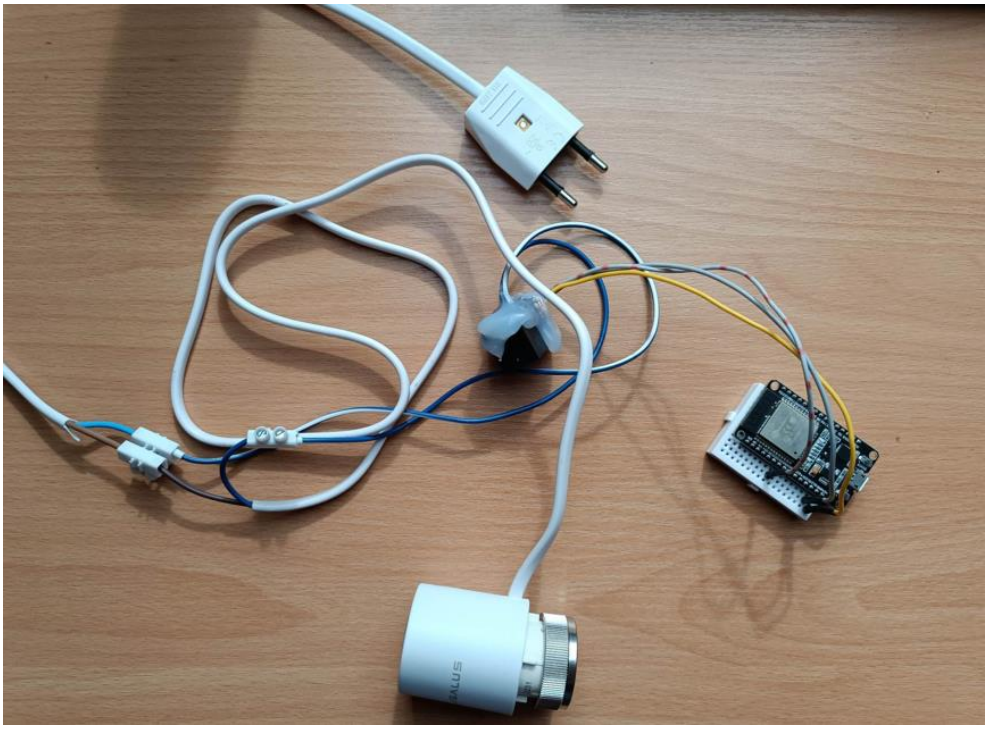
5. Üzembe helyezés

A rendszer üzembe helyezésének érdekében szükség volt a tervezett részeknek a megépítésére. Mivel minden rész vezeték nélkül kommunikál egymással, szabadon elhelyezhetőek egy lakásban, azonban külön áramforrást kell biztosítani a Raspberry-nek és külön-külön az ESP32-nek is microUSB-n keresztül. Fontos az, hogy minden eszköz egy Wi-Fi hálózaton legyen, ugyanis csak így tudnak kommunikálni az eszközök egymással.

28. ábra Központi egység



29. *ábra Öntözőrendszer*



30. *ábra Hőmérséklet szabályzó*

A 28. ábra 29. ábra 30. ábra mutatja be a különálló részegységeknek a fizikai megvalósítását.

6. Továbbfejlesztési lehetőségek

Egy okosotthon vezérlő rendszer esetében számos továbbfejlesztési lehetőség adódik, rengetek lehetőségünk van a továbbfejlesztésre. A rendszerhez hozzáadható bármennyi érzékelő, hardveres és szoftveres szempontból is továbbfejleszthető. Ezek mellett vannak olyan hibalehetőségek is amik a rendszernek a nem megfelelő használatakor jöhetnek elő. Az alábbiakban felsorolok néhány továbbfejlesztési lehetőséget:

- a felhasználói felület bővítése, a design finomítása
- a vízpumpához tartozó víztartályba érzékelő beépítése, ami jelez és leállítja a vízpumpát, ha a tartályból kifogyott a víz
- több érzékelő hozzáadása a rendszerhez
- napelem integrálása a rendszerbe és az adatai kezelése
- redőny tényleges beépítése a rendszerbe

7. A projekt fájl tartalma

A projekthez mellékelt forráskódok, illetve mellékelt fájlok tartalma a következő:

- *home_automation*: a webes alkalmazás forráskódját tartalmazó könyvtár
- *esp32/ontozo.ino:* az ESP32 mikrovezérlőre fejlesztett szoftver forráskódja, ami az öntözőrendszerhez készült – ez egy Arduino fájl
- *esp32/homerseklet.ino:* az ESP32 mikrovezérlőre fejlesztett szoftver forráskódja, ami a fűtés szabályozásához készült – ez is egy Arduino fájl
- *fuzzy_matlab/redony.fis:* a redőny mozgatásához szükséges fuzzy logika szimulációját tartalmazó fájl, ami megnyitható Matlabban a Fuzzy Toolbox használatával

8. Irodalomjegyzék

[1] K. Venkatesh, P. Rajkumar, S. Hemaswathi, B.Rajalingam, “IoT Based Home Automation Using Raspberry Pi”, Jour of Adv Research in Dynamical & Control Systems, Vol. 10, 07-Special Issue, 2018
https://www.researchgate.net/profile/Rajalingam-Balakrishnan/publication/IoT_Based_Home_Automation_Using_Raspberry_Pi

[2] LOGO!
https://new.siemens.com/global/en/products/automation/systems/industrial/plc/logo/home-automation.html

[3] Eaton
https://www.eaton.com/hu/hu-hu/products/residential/wireless-smart-home-solution-xcomfort.html

[4] Google Home
https://home.google.com/what-is-google-home/

[5] Raspberry Pi 3 Model B+ adatlap
https://static.raspberrypi.org/files/product-briefs/Raspberry-Pi-Model-Bplus-Product-Brief.pdf

[6] Bluetooth és BLE
https://www.makeuseof.com/what-is-ble-bluetooth-low-energy/

[7] MQTT
https://www.hivemq.com/blog/how-to-get-started-with-mqtt/

[8] Nitin Naik, „Choice of Effective Messaging Protocols for IoT Systems: MQTT, CoAP, AMQP and HTTP,” Defence School of Communications and Information Systems Ministry of Defence, United Kingdom.

[9] Andy Stanford-Clark, Hong Linh Truong, „MQTT For Sensor Networks (MQTT-SN) Protocol Specification,” 2013.

[10] Kakucs Gergő disszertációja, “Indukciós edzőgépek felkészítése IIoT kompatibilitásra MQTT protokoll és Siemens PLC-k alkalmazásával”, 2020

[11] Open-Meteo dokumentáció
https://open-meteo.com/en/docs

[12] Brassai Sándor Tihamér, “Neurális hálózatok és Fuzzy logika”, Scientia kiadó, Kolozsvár, 2019
http://real.mtak.hu/122603/1/BrassaiSandor_Neuralis%20halozatok_REAL.pdf

[13] Amezen, N. M., & Al-Ameri, J. A. M., “IoT-Based Shutter Movement Simulation for Smart Greenhouse Using Fuzzy-Logic Control. ”, 2019

9. Függelék

Függelék 1. pont: Hőmérséklet és páratartalom kiolvasása

```
from bluepy import btle

class XiaomiTempHumidity:
    def __init__(self, mac):
        self.mac = mac.lower()
        self.peripheral = btle.Peripheral()

    def connect(self):
        try:
            self.peripheral.connect(self.mac)
        except:
            print("Nem sikerult kapcsolodni")

    def disconnect(self):
        self.peripheral.disconnect()

    def read_temp(self):
        data = self.peripheral.readCharacteristic(0x0036)
        humidity = data[2]
        temp = (data[1] << 8 | data[0]) / 100
        return temp, humidity
```

Függelék 2. pont: Értékek küldése a Home oldalra

```
def index(request):
    try:
        xi = XiaomiTempHumidity('A4:C1:38:35:F1:35')
        xi.connect()
        temp, hum = xi.read_temp()
    except:
        temp = '--'
        hum = '--'

    return render(request, 'home.html', {'temp': temp, 'hum':
hum})
```

Függelék 3. pont: Ajax kérés az adatok frissítésére

```
    function refreshTempAndHumidity(){
      $.ajax({
          url: 'refresh/',
          type: 'GET',
          dataType: 'json',
          success: function(response){
              var temp = response.temperature;
              var hum = response.humidity;
              showOrHideWarning(temp, hum);
              $('#temperature').text(temp);
              $('#humidity').text(hum);
          }
      });
  }
  …
  setInterval(refreshTempAndHumidity, 10000);
```

Függelék 4. pont: Mikrovezérlőn kezelt üzenetek

```
if(messageTemp[0] == '0'){
    // cut the first charachter, wich is 0
    int charachter = messageTemp.indexOf('|');
    if(charachter != -1){
        dry = messageTemp.substring(1, charachter).toInt();
        wet = messageTemp.substring(charachter + 1).toInt();
        Serial.print(dry);
        Serial.print(wet);
      }
}
if(messageTemp == "auto"){
    autoIrrigation = 1;
    digitalWrite(waterPump, LOW);
}

if(messageTemp == "manual"){
    autoIrrigation = 0;
    digitalWrite(waterPump, LOW);
}


if(messageTemp == "on"){
    digitalWrite(waterPump, HIGH);
}
if(messageTemp == "off"){
    digitalWrite(waterPump, LOW);
}
```

Függelék 5. pont: Kapcsolódás az MQTT szerverhez

```
void connect_mqttServer() {
    while (!client.connected()) {
        if(WiFi.status() != WL_CONNECTED){
            setup_wifi();
        }

        Serial.print("Attempting MQTT connection...");

        if (client.connect("ESP32_2")) {
            //sikeres csatlakozas
            Serial.println("connected");
            // feliratkozas a topicra
            client.subscribe("temperature");
        }
        else {
            //sikertelen csatlakozas
            Serial.print("failed, rc=");
            Serial.print(client.state());
            Serial.println(" trying again in 2 seconds");
            delay(2000);
        }
    }

}
```

UNIVERSITATEA SAPIENTIA DIN CLUJ-NAPOCA

FACULTATEA DE ȘTIINȚE TEHNICE ȘI UMANISTE, TÎRGU-MUREȘ

SPECIALIZAREA CALCULATOARE

Vizat decan

Conf. dr. ing. Domokos József

Vizat director departament

Ș.l. dr. ing. Szabó László Zsolt