Budapesti Műszaki és Gazdaságtudományi Egyetem
Villamosmérnöki és Informatikai Kar
Hálózati Rendszerek és Szolgáltatások Tanszék

Nagy Borbála

FENNTARTHATÓ MEZŐGAZDASÁGOT TÁMOGATÓ OKOSÜVEGHÁZ FEJLESZTÉSE IOT TECHNOLÓGIÁKKAL

KONZULENS

Gódor Győző

BUDAPEST, 2023

Tartalomjegyzék

HALLGATÓI NYILATKOZAT

Alulírott **Nagy Borbála**, szigorló hallgató kijelentem, hogy ezt a szakdolgozatot meg nem engedett segítség nélkül, saját magam készítettem, csak a megadott forrásokat (szakirodalom, eszközök stb.) használtam fel. Minden olyan részt, melyet szó szerint, vagy azonos értelemben, de átfogalmazva más forrásból átvettem, egyértelműen, a forrás megadásával megjelöltem.

Hozzájárulok, hogy a jelen munkám alapadatait (szerző(k), cím, angol és magyar nyelvű tartalmi kivonat, készítés éve, konzulens(ek) neve) a BME VIK nyilvánosan hozzáférhető elektronikus formában, a munka teljes szövegét pedig az egyetem belső hálózatán keresztül (vagy hitelesített felhasználók számára) közzétegye. Kijelentem, hogy a benyújtott munka és annak elektronikus verziója megegyezik. Dékáni engedéllyel titkosított diplomatervek esetén a dolgozat szövege csak 3 év eltelte után válik hozzáférhetővé.

Kelt: Budapest, 2023. 12. 16.

...…….
Nagy Borbála

Összefoglaló

Manapság egyre elterjedtebbek az intelligens technológiák és a fenntartható megoldások az élet minden területén, okosotthonok, okos városok jönnek létre. Ezen a vonalon integrálódik a mezőgazdaságba is az informatika és az automatizáció, okosüvegházakat, okos farmokat és beavatkozó robotokat kezdtek gyártani az utóbbi években.

Az IoT eszközök jelenléte sokoldalú segítséget nyújt a folyamatos monitorozásban, emberi beavatkozás nélküli vezérlésben, és nem utolsó sorban az erőforrások hatékony kihasználásában. A növénytermesztés fenntarthatósága napjaink egyre égetőbb problémája, az emberek száma a Földön csak növekszik, a megművelhető terület viszont ugyanaz. Emiatt szükség van új technológiákra, innovációkra, amikkel effektívebben használjuk ki a rendelkezésünkre álló erőforrásokat, készleteket.

A dolgozatban bemutatom egy IoT alapokra épülő okosüvegház fejlesztését, ami a fenntartható mezőgazdaság támogatására jött létre. A ház valójában egy makett, rendelkezik külön vezérelhető részegységekkel, megvalósításra kerül benne az állandó monitorozás egy szenzorrendszer által, az automatikus beavatkozás és a távoli vezérlés. Emellett szó esik egy kliensoldali Android alkalmazásról, amit az üvegház aktuális állapotainak megfigyelésére és módosítására készítettem.

Abstract

Nowadays, intelligent technologies and sustainable solutions are becoming increasingly prevalent in every aspect of life; smart homes and smart cities are emerging. Following this trend, agriculture is also integrating information technology and automation, with the production of smart greenhouses, smart farms, and intervention robots in recent years.

The presence of IoT devices provides significant assistance in continuous monitoring, human intervention-free control, and, not least, efficient resource utilization. The sustainability of crop cultivation is an increasingly pressing issue in our times; the global population is only increasing, while cultivable land remains the same. This necessitates new technologies and innovations to more effectively utilize the available resources and stocks.

In this thesis, I present the development of a smart greenhouse based on IoT principles, designed to support sustainable agriculture. The greenhouse is essentially a model, equipped with separately controllable subsystems. It incorporates continuous monitoring through a sensor system, automatic intervention, and remote control. Additionally, I discuss a client-side Android application that I developed for observing and modifying the current state of the greenhouse.

1 Bevezetés

Napjaink felkapott jelzője a *fenntartható* és az *okos,* minden modern technológiától elvárt, hogy igazak legyenek rá ezek a kifejezések. Nekünk, informatikusoknak nagy szerencsénk van, hogy a társadalomnak erre igénye van, ugyanis remekül kamatoztathatjuk tudásunkat ezekben a projektekben. Egy ilyen terület a mezőgazdaság is, ami évezredes múltra tekint vissza, az utóbbi néhány évben viszont jelentősen megváltozott, különösen is kibővült a tartalma. Már nem csak a kézzel vagy nagy munkagépekkel való földművelést értjük mezőgazdaság alatt, hiszen egyre inkább integrálódtak az informatikai technológiák is ebbe a szektorba.

Egy okosüvegház kiváló példája ennek az integrálódásnak, célja a fenntarthatóság, optimalizálás, a produktivitás növelése, nemcsak rövid, hanem hosszútávon is. Az erőforrások effektív kihasználása az egyik legfontosabb cél lett, az erre való törekvés egyre nagyobb. Szenzorrendszereket és különböző automatizált technikákat használnak ennek megteremtése érdekében, ezzel a termelést hatékonyabbá és minőségibbé téve.

Ez a cél és terület számomra is fontos, nagy haszna van a társadalom és a gazdaság szempontjából is, és bizonyosan nem idejétmúlt. Ezen okokból választottam szakdolgozatom témájának, hogy fejlesszek egy, a fenntartható mezőgazdaságot támogató okosüvegházat, ami IoT alapokon nyugszik. Az IoT alapú technológiák számos előnyt nyújtanak egy ilyen rendszer esetén. A szenzorok segítségével folyamatos monitorozást hajthatunk végre az üvegház paraméterein, a beavatkozók által pedig automatikus locsolást, szellőztetést és egyéb módosításokat hajthatunk végre emberi beavatkozás nélkül. Elérhetővé válhat a távoli megfigyelés és vezérlés, a termelés optimalizálása, a környezettudatosság.

A dolgozatban az általam fejlesztett okosüvegház megvalósításának a lépéseit fogom bemutatni, összesen nyolc fejezetre osztva.

A második fejezetben piackutatást végzek, amiben áttekintem az automatizálás fejlődését, az informatika jelenlétét a növénytermesztésben, és beszélek a növények különböző igényeiről. Ezt követően mai korunk néhány meglévő okosüvegház-megvalósítását részletezem.

A harmadik fejezetet a használt eszközök leírásával folytatom, ideértve az érzékelőket, beavatkozókat és vezérlőket, amiket beépítettem a rendszerbe.

A negyedik fejezetben olvashatunk a megvalósított rendszerről, annak logikai és fizikai felépítéséről. Szó esik a különböző részegységekről, interfészekről, vezérlési mechanizmusokról. Bemutatom az összeállított üvegházat fizikai, hardveres oldalról.

Az ötödik fejezetben megismertetem a szoftverarchitektúrát és implementációt. Kifejtem a használt operációs rendszer előnyeit, beszélek a Python-ban megírt kódról és a különböző kihívásokról, amikkel az implementálás során találkoztam. A rész végén írok az MQTT kliensoldali részéről, valamint a program automatikus elindításáról.

A hatodik fejezetben betekintést nyújtok az üvegház által használt kommunikációs technológiákba, egész pontosan az I2C és az MQTT rejtelmeibe, ezután pedig feltérképezem a kommunikációs struktúrát is.

A hetedik fejezetben bemutatom az általam készített kliensoldali alkalmazást, amit egy Android alkalmazásban valósítottam meg. Beszélek az architektúráról és azokról a fő technológiákról, amik alapján a programot felépítettem, amik az alkalmazás jellegét különösen meghatározzák. Végül a felhasználói felületen vezetem végig az olvasót, egyfajta felhasználói tájékoztató formájában.

Az utolsó, nyolcadik fejezetben értékelem az elkészült munkát és szót ejtek az esetleges továbbfejlesztési lehetőségekről.

2 Piackutatás

Bármilyen innovatív ötlet haszontalanná tud válni, hogyha nem mérjük fel előtte a piaci igényeket, nem tudjuk, hogy a felhasználóknak mik az elvárásaik egy adott rendszerrel szemben. Az okosüvegházak célközönsége a mezőgazdaságban dolgozóktól, akik professzionális termelést szeretnének végrehajtani az átlagemberekig terjed, akik csak kis mennyiségben szeretnének saját maguk számára növényeket termeszteni. A következőkben át fogom tekinteni a növénytermesztésben az automatizálás fejlődését, a jelenlegi technológiákat, a már meglévő okosüvegház megvalósításokat, tendenciákat, illetve kitérek a növények igényeire a környezeti tényezőkkel szemben.

2.1 Az automatizálás fejlődése

A növénytermesztésben az automatizálás folyamatos fejlődése hosszú történettel rendelkezik, amit a technológiai fejlődés, az ipari forradalmak és az informatikai forradalom együttesen alakítottak. Néhány mérföldkövet érdemes megemlíteni, amik elvezettek minket a modern mezőgazdasági gyakorlatokig.

Az első nagyobb áttörést a traktorok és egyéb mezőgazdasági gépek megjelenése jelentette a 19. században. Ezek tették lehetővé a talajművelés, vetés és betakarítás folyamatainak mechanizálását, ezzel jelentősen növelve a termelékenységet és csökkentve a munkaerőigényt. [1]

A 20. század közepén jelentek meg az automatizált öntözési rendszerek, mint például a csepegtető öntözés, amik által hatékonyabb lett a vízfelhasználás, a vízforrások felhasználása pedig fenntarthatóbb. [2] Ebben az időszakban már nagyon gyors ütemben nőtt az emberek száma, akik nem a maguk kis kertjében termesztett növényekből élnek meg, hanem a boltban vásárolt terményekből, így különösen is nagy szükség lett a fenntarthatóság és hatékonyság növelésére.

A következő nagy mérföldkő a precíziós mezőgazdaság megjelenése az 1980-as években, ami a GPS és a szenzorok bevezetésével tette lehetővé a pontosabb növényápolási gyakorlatokat. A gazdák egyre pontosabban tudták irányítani traktoraikat és egyéb gépeiket, optimalizálva ezzel a talajművelést, vetést és permetezést. Ezzel már nem csupán a gépészet, hanem az informatika is megkezdte térnyerését a mezőgazdaságban, aminek automatizálása így rohamos ütemben kezdhetett fejlődni. [3]

A 21. század elején jelentek meg az automatizált betakarítási rendszerek, értve ezalatt drónokat, robotokat, akik „agyában” különböző képfeldolgozási rendszerek futnak, amik alapján képesek megállapítani, melyik az érett termés, és azt milyen erővel kell eltávolítani a tövéről. Ez a módszer már egyre inkább kezdte kiváltani az emberi beavatkozás szükségességét, amire egyre nagyobb az igény, tekintve, hogy a mezőgazdasággal foglalkozni kívánó emberek száma egyre inkább csökken. [4]

Pár évvel később az okosüvegházak és hidroponikus rendszerek is bekerültek a köztudatba, amikben a magasabb szintű automatizálási technikák lehetővé teszik a környezeti paraméterek, mint hőmérséklet, páratartalom, világítás automatikus és szükség szerinti szabályozását. A hidroponikus rendszer lényege, hogy a növény gyökérzete nem a talajjal érintkezik, hanem egy speciális összetételű oldattal, amiből közvetlenül fel tudja venni a tápanyagot, ezzel kikerülve a termőföldet, mint közvetítő közeget. [5]

2.2 Az informatika a növénytermesztésben

Ahogy említettem is, egyre inkább szivárognak be az informatikai rendszerek a növénytermesztésbe, ennek részleteibe szeretnék egy kis betekintést nyújtani.

Egyre elterjedtebbek a különféle szenzorok, és ezzel együtt az IoT (Internet of Things) alapú technológiák, amik lehetővé teszik a mezőgazdászok számára, hogy valós időben monitorozzák a növények számára fontos paramétereket. Ezt egészíti ki a mesterséges intelligencia és a gépi tanulás, aminek alkalmazásával a gépek tanulhatnak az adatokból, és hozzáigazíthatják működésüket az adott növény számára optimális eredmény eléréséhez. Ez a rendszer még nem kiforrott, folyamatosan tud fejlődni, és használatával van a legnagyobb esély arra, hogy az emberi beavatkozásokat ki lehessen kerülni a mezőgazdaságban. [6] Az, hogy ez jó-e vagy sem, inkább erkölcsi kérdés, informatikai szempontból inkább egy kihívásnak mondanám.

Az automatizált rendszerek és robotok segítenek a monoton és ismétlődő feladatokban, például a gyomlálásban, permetezésben és betakarításban, ami szintén növeli a hatékonyságot és csökkenti a munkaerőigényt, ami ezen a területen manapság előny. Robotok és gépek által vezérelt rendszerben sokkal könnyebb a termesztési folyamatokat pontosan irányítani, például a víz- és tápanyagfelhasználást. [4]

A mi feladatunk tehát, hogy a lehető legkevesebb erőforrással, fenntartható és hatékony módszerekkel tudjuk a lehető legoptimálisabb szintre emelni a növénytermesztést.

2.3 Növényi igények

A növények számára sok fontos környezeti változó van, amik meghatározzák fejlődésüket, és amiknek szabályzására figyelmet kell fordítanunk. A következőkben bemutatok néhány alapvető paramétert ezek közül.

2.3.1 Fény

A fény jelenléte elengedhetetlen az asszimilációhoz, amely során a növények a szervetlen sókból és CO_2-ből a klorofill segítségével szerves vegyületeket képeznek, ezáltal megkötve a fény energiáját. A lekötött energia a légzés folyamán szabadul fel, ezt használja a növény saját testének a felépítésére. Az asszimiláció egy széles kategória, amely magában foglalja a fotoszintézist, de kiterjed más tápanyagok feldolgozására is a növények szervezetében.

A növények 100-200 lux fényerősség mellett kezdenek asszimilálni, viszont ez a kis fénymennyiség még nem elég ahhoz, hogy új tápanyagokat tudjanak előállítani, így csak tartalékaikból élnek. Átlagosan 500-800 lux szükséges ahhoz, hogy a szervesanyag-tevékenység és a légzés egyensúlyba kerüljön, de természetesen ez függ a növény fényigényétől és a környezet hőmérsékletétől. Az ideális állapot eléréséhez általában ettől nagyobb megvilágításra van szükség, néhány szobanövény akár 10-20000 luxot is igényelhet.

Azonban nem csak az számít, milyen erősségű fényt kapnak növényeink, hanem annak időtartama is. Amennyiben természetes fényt is kapnak, úgy pótmegvilágításról beszélünk, amit átlagosan 150 W/m^2 erősséggel, 6-10 órán keresztül kell alkalmazni. Hogyha a növények teljes sötétben állnak, akkor mesterséges megvilágításról vagy műfényben nevelésről van szó, ebben az esetben a megvilágítás erőssége 400-500 W/m^2, időtartama pedig 12-18 óra. A hagyományos izzók viszont nem megfelelőek erre a célra, ugyanis több hőt sugároznak, mint amennyi fényt adnak, ami miatt a növény nyurga hajtásokat hozna. [7]

Mesterséges megvilágításnál arra is figyelnünk kell, hogy milyen hullámhosszú fényt kapnak növényeink, ugyanis a fehér fény különböző komponensei más-más

területeken táplálják és fejlesztik őket. A fotoszintézishez szükséges hullámhossz a 440 és 660 nm. A fejlődéshez, növekedéshez a 660 nm-es vörös és 735 nm-es távoli vörös fény az ideális, a levélképződéshez pedig a 435 nm-es kék. A 440 nm emellett a növények fény irányába történő mozgásáért is felel. [8]

2.3.2 Víz

Mint minden élőlény, a növények sem tudnának víz nélkül élni, hiszen az a működtetője számos fiziológiai és biokémiai folyamatnak. Szükség van rá a fotoszintézis során is, hiszen annak a szén-dioxid mellett a víz a másik elengedhetetlen összetevője. A víz emellett hordozza a talajban oldott tápanyagokat és ásványi anyagokat is, ami által a gyökerek képesek felvenni azokat.

A víz részt vesz a sejtfalak turgor-nyomásának fenntartásában is, ami segít megtartani a növényeknek a formájukat, tartani a leveleiket és megakadályozni a lelapulást vagy hajlást. Gyakran azonban nem a vízhiány miatt kókadtak a növény levelei, hanem pont, hogy a túlöntözés miatt, ami szintén egy elterjedt probléma.

A hűtésben is lényeges szerepet játszik a víz, hiszen a növények hűtési mechanizmusként használják a leveleiken keresztül való párologtatást. Emellett víz nélkül az anyagcserét sem tudják végrehajtani, ideértve az anyagok szállítását a sejtek között és a szerves anyagok lebontását energiává.

Természetesen a növények vízigénye függ a fajtól, növekedési stádiumtól és a környezeti feltételektől. Az optimális növekedéshez és fejlődéshez pontos megfigyeléseket kell tenni a konkrét növénnyel kapcsolatban, általánosságokat nehezen tudunk levonni. [9]

2.3.3 Hőmérséklet

Minden növénynek van egy optimális hőmérséklet-tartománya, amiben a legjobban nő és fejlődik. Alapvetően azt gondolnánk, hogy az állandó klíma tesz nekik a legjobbat, viszont bizonyos növények számára fontos a nappali és éjszakai hőmérséklet-különbség. Sok zöldségnek, gyümölcsnek és dísznövénynek szüksége van különféle hőmérsékletbeli ingadozásokra a megfelelő gyümölcsképzés és virágzás érdekében.

Emellett az egyes növények stressztűrése is fontos szempont, hiszen a túl alacsony vagy túl magas hőmérséklet stresszt jelenthet a növények számára. A téli és tavaszi fagy okozza a legtöbb gondot hazánkban, a termések, virágok könnyedén lefagyhatnak, illetve

a nagy hidegben méhek sem jelennek meg. A túl magas hőmérséklet viszont például már gátolja az asszimilációt és növeli a párolgást, ami által az elvesztett vízmennyiséget a növény már nem tudja pótolni. [10]

2.3.4 Páratartalom

Az optimális páratartalom, az eddigiekhez hasonlóan szintén nem tekinthető statikus értéknek, sok tényezőtől függ és nem is lehet egy értékkel jellemezni, azonban néhány általános következtetést le lehet vonni.

Az ideálisnál magasabb páratartalom elősegíti a gombás és baktériumos betegségek terjedését, vízkórságot idéz elő, rontja a terméskötődést és lassítja a transzspirációt – ami magában foglalja a vízfelvételt, vízszállítást és párologtatást –, ezáltal a tápanyagok felvételét.

Hasonlóan azonban a túl alacsony páratartalom is okozhat zavarokat és nem várt viselkedést, szintén rontja a terméskötődést – tehát nem tapad a bibéhez és nem csírázik a pollen –, gyengébb a transzspiráció, mert lezárnak a gáznyílások, pórusok, védve ezzel a növényt a túlpárologtatástól, fénytelenné válnak a termések, vagy apró, hajszálvékony repedések keletkezhetnek a termés felületén. Bár alapvetően a nedves levegő ad jó táptalajt a fertőzéseknek, van olyan betegség is, ami kifejezetten gyorsabban terjed és jobban fertőz a száraz levegőben. [11]

2.4 Meglévő okosüvegház-megvalósítások

A piacon már pár éve jelen vannak különféle okosüvegház-megvalósítások, amik kisebb-nagyobb sikert arattak. Bár mára természetesnek tűnik, hogy minden eszközünk okos, és hogy az intelligens technológiák életünk minden területének részévé válnak, a mezőgazdaság automatizálása még mindig egy olyan terület, ahol elférnek a fejlesztések, innovatív megoldások. A jelenleg népszerű megoldásokból és termékekből szeretnék a következőkben néhányat bemutatni.

2.4.1 SMARTKAS

Az elmúlt évek egyik legnagyobb úttörő megoldását nyújtotta a SMARTKAS, ami egy magyar fiatal, Mészáros Dávid vállalkozása. 2020 májusában indult a cég, amely megújuló okosüvegházakat, beltéri farmokat tervez és hoz létre. A vállalat Hollandiában

került megalapításra azzal a céllal, hogy segítsen megoldani az éhezés problémáját a világban. Logójukat a 2.1-es ábrán láthatjuk.

2.1. ábra: A SMARTKAS logója

Sikerük egyik titka, hogy megkeresik a legjobb stratégiai helyszíneket, ahol megújuló energiaforrásokkal tudnak gondoskodni az okosfarmok energiaellátásáról. Mikrodózisokat használva adagolják a növényeknek a megfelelő mennyiségű szén-dioxidot, így nagy mértékben csökkentik biológiai lábnyomukat. A mesterséges intelligencia, a robotika és az adatvezérelt növekedési folyamatok jóvoltából évszaktól függetlenül, személyzet nélkül, teljesen automatizáltan tudják működtetni okosfarmjaikat. Öntözőrendszerük vízpozitív, mivel a keringő vízellátás újrahasznosítása mellett további vizet állítanak elő az esőből és a levegő páratartalmából.

2.2. ábra: A SMARTKAS egyik üvegháza

A termelési folyamat egy légmentesen zárt térben megy végbe, ahogy azt a 2.2. ábrán láthatjuk, valamint nem használnak növényvédő szereket, így az általuk termesztett növényeket még csak megmosni sem kell fogyasztás előtt. Több, mint ötven fajta gyümölcs, zöldség és fűszernövény termelésére képesek, emellett termesztenek kozmetikai cégeknek algákat, valamint orvosi célokra marihuánát is.

Egyik legnagyobb projektjük Hollandiában található, ami Európa legnagyobb, magánkézben levő üzleti parkja. Ezen kívül megtalálhatóak az Egyesült Királyságban, Magyarországon és Brazíliában is. A cég értéke 2022-ben már elérte a négyszázmillió eurót, azóta ez csak növekedett. [12]

2.4.2 Plantee

A Plantee egy prágai székhelyű startup, ami egy kicsi, beltéri üvegházat valósít meg, teljesen hétköznapi, nem professzionális használatra. A terméket a 2.3. ábrán láthatjuk.

2.3. ábra: A Plantee [13]

2019-ben alapította a céget két cseh fiatal, azzal a céllal, hogy megoldást nyújtsanak azoknak a hétköznapi embereknek, akik nem tudják, hogy kell gondozni egy növényt, viszont szívesen tartják őket. Emellett azoknak is megfelelő, akik a saját otthonukban szeretnének egy különlegesebb, klímához nem illeszkedő növényt termeszteni.

A termékben megvalósításra kerül a növény és a talaj fűtése, a növény számára ideális mennyiségű és összetételű LED világítás, a talajnedvesség monitorozása és az alapján locsolás egy beépített tartályból, valamint kártevőmonitorozás. A Plantee jelenleg 1399 euróba kerül, ez mostani árfolyam szerint 532 055 forint. [13]

3 Használt eszközök

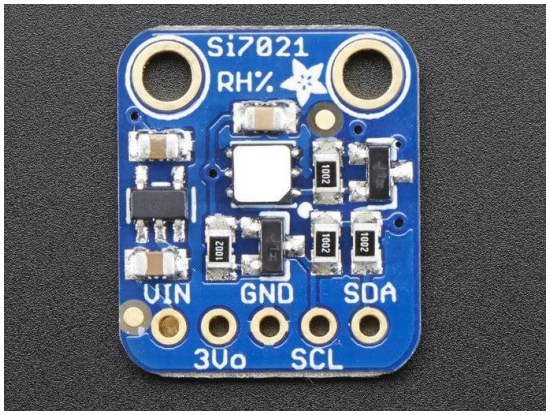
3.1 Érzékelők

Elsőkörben megállapítható, hogy egy okosüvegháznak szüksége van egy szenzorrendszerre, amivel valós időben tudjuk monitorozni a benne uralkodó aktuális állapotokat. A növények számára fontos paraméterek szerencsére már köztudottak, így könnyen találhatunk olyan érzékelőket a piacon, amiket érdemes beépítenünk a rendszerbe.

3.1.1 Hőmérséklet és páratartalom

Sajnos az utóbbi évek tavaszán szélsőségesen változó időjárással, hőmérséklettel kellett szembenézniük a növényeknek, ami sokszor a termelést törtrészeire tudja csökkenteni. Egy ilyen zárt rendszer nagy előnye az is, hogy a hőmérsékletet könnyen tudjuk állandósítani, így nem kell tartani az esetleges hirtelen lehűlésektől, melegedésektől.

A rendszerben a 3.1. ábrán látható Adafruit Si7021 hőmérséklet- és páratartalom-mérő szenzort használtam, ami -10 és +85 C° között képes mérni hőmérsékletet, tehát széles tartományú, és emellett nagy pontosságú is. A relatív páratartalmat 0 – 80% között tudja precízen jelezni, de felette is használható, valamivel kisebb pontossággal. A szenzor I2C kommunikációt használ. [13]



3.1. ábra: Adafruit si7021 [13]

3.1.2 Fényerősség

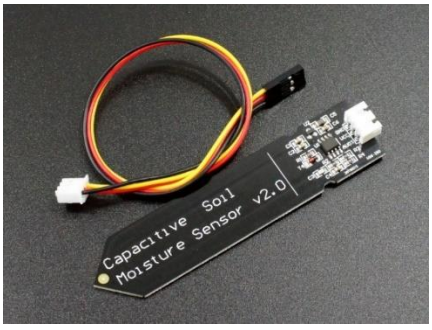
A fény mennyisége szintén létfontosságú, ehhez a 3.2. ábrán látható Adafruit TSL2591 fényérzékelő szenzort használtam. Látható és infravörös tartományban is képes mérni, 188 mikrolux és 88000 lux között. Összehasonlításképpen a holdtalan, borús éjszakai égbolt fényintenzitása 10^{-4} lux, egy lakószoba átlag megvilágítása 500 lux, a közvetlen napfény pedig 32000-130000 lux között van. Ez a szenzor is I2C-n keresztül kommunikál. [15]

3.2. ábra: Adafruit tsl2591 [15]

3.1.3 Talajnedvesség

Az egyik legalapvetőbb paraméter, amit mérnünk kell, a talajnedvesség. Erre a célra egy kapacitív talajnedvesség-mérő szenzort használtam (3.3. ábra), aminek előnyös tulajdonsága a hosszú élettartam, ami kapacitív mivoltának köszönhető.

A szenzor kimenete analóg, a kimeneti feszültség fordítottan arányos a talajnedvesség szintjével. A nedves talaj jobb elektromos vezető, nagyobb a kapacitása, így a kimeneti feszültség csökken. Száraz talaj esetén ennek az ellenkezője történik, a vezetőképesség csökken, a kimeneti feszültség pedig megnő. [16] Az általam mért két szélsőérték teljesen száraz szenzor esetén – 45%-os relatív páratartalommal a szobában – 24649 volt, vízbe mártva pedig 6253. Az elvi maximum 32768, a minimum pedig 0 lenne.



3.3. ábra: Kapacitív talajnedvesség-mérő szenzor [16]

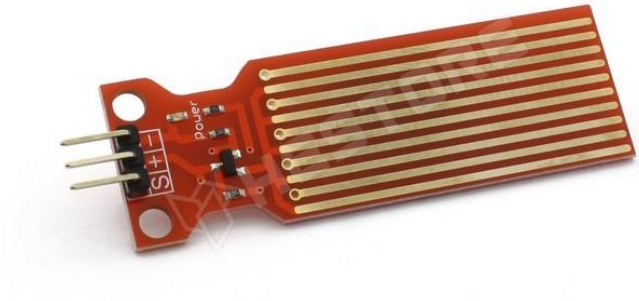
3.1.4 Szélerősség

A szélerősséget természetesen nem bent, hanem az üvegház mellett, azon kívül mérjük, amire azért van szükség, hogy tudjuk, mikor nyithatjuk ki az ablakot és mikor nem ajánlott. A méréshez a 3.4. ábrán látható A1733 kanalas anemométert használtam, ami maximum 70 m/s szélerősségig mér, 1 m/s pontossággal. Az előzőhöz hasonlóan ez az érzékelő is analóg kimenettel rendelkezik, viszont ennek üzemi feszültsége 12V. [17]

3.4. ábra: A1733 kanalas anemométer

3.1.5 Vízszint

Az üvegház alatt helyezkedik el egy víztartály, amiből pumpák segítségével locsolunk, így azok megfelelő működéséhez tudnunk kell, mennyi víz van még a tartályban. Erre a célra a 3.5. ábrán látható WLD-75 vízszintmérő szenzort használtam, aminek kimenete a vízszinttel arányos feszültség. [18] Nagyjából 8000-es értékre emelkedik hirtelen – ami a mérési tartomány negyedének felel meg, ha beletesszük a vízbe a végét, és ugyanerre az értékre csökken le gyorsan, amikor kiemeljük a vízből, így ezt a határértéket állapítottam meg.



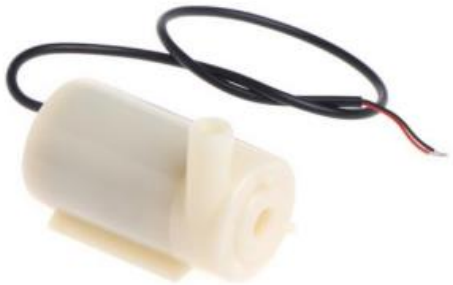
3.5. ábra: WLD-75 vízszintmérő szenzor [18]

3.2 Beavatkozók

Az érzékelők mellett fontos pont az automatizálás is, így szükségünk van különböző beavatkozókra, hogy a könnyen változtatható paraméterek esetén elkerülhető legyen az emberi beavatkozás. Ezeknél elsődleges szempont, hogy a céljuk az, hogy minél kevesebb erőforrást fogyasszunk, minél optimálisabb legyen az energiafelhasználás, ezzel is javítva a hatékonyságot.

3.2.1 Vízpumpa

A locsolás a legalapvetőbb eleme az üvegháznak, ehhez vízpumpákat volt szükséges beiktatni a rendszerbe. A használt vízpumpa (3.6. ábra) a merülő vízszivattyúkra hasonlít, tehát víz alá kell helyezni, és onnan fogja kipumpálni a vizet. Működéséhez elengedhetetlen, hogy valóban víz alatt legyen használat közben, emiatt van szükség a fentebb említett vízszintmérő alkalmazására. Mivel On/Off alapú, így a rendszerhez egy relén keresztül kapcsoljuk, amivel ki/bekapcsolni tudjuk. [19]



3.6. ábra: Vízpumpa [19]

Hátránya ennek az eszköznek, hogy nem a legjobb minőségű, a hozzá adott vezetékekben nagyon vékonyak a drótok. Mivel nem túl hosszú a kábel, így muszáj összeforrasztani vagy kötni valahogy egy másik vezetékkel, hogy elérjen a vezérlés helyétől a víztartályig, ez okozott nehézségeket.

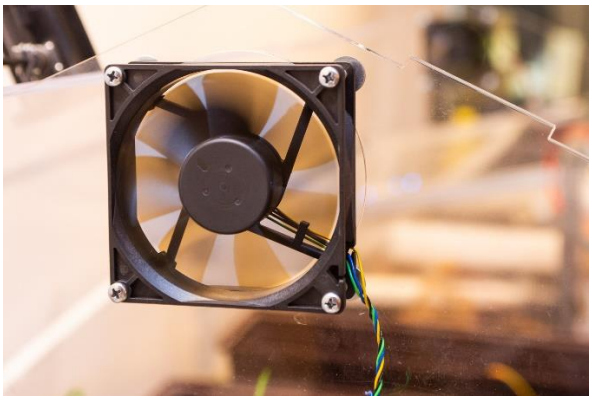
3.2.2 Ablaknyitó motor

Amennyiben a kinti környezeti változók elősegítik, hogy az üvegházon belül elérjük a kívánt állapotokat, egy léptetőmotor segítségével ki tudjuk nyitni az ablakot. A motor a 3.7. ábrán látható, üzemi feszültsége 12V, és egy mikrokontroller használata is szükséges működéséhez. [20]

3.7. ábra: Ablaknyitó motor [20]

3.2.3 Ventilátor

A hőmérséklet, és főként a páratartalom változtatására egy OEM PC ventilátort használtam (3.8. ábra). Alapvetően az üzemi feszültsége ennek is 12V, de fordulatszáma arányos a rá adott feszültséggel, tehát kisebb feszültségen is működik lassabb fordulatszámmal. [21] A jelenlegi rendszerben egy relén keresztül vezéreljük, ahol a maximális, 12V-ot kapja.



3.8. ábra: A ventilátor

3.2.4 LED szalag

A fény növelésére és paramétereinek állítására RGB LED szalagot alkalmaztam (3.9. ábra), ami által kielégíthetjük a növények különböző hullámhosszok iránti igényüket, ahogy arról fentebb szó is esett. A LED szalag működtetéséhez 5V szükséges.

3.9. ábra: RGB LED szalag [22]

3.3 Egyéb eszközök

3.3.1 Relé

Ahogyan fentebb már említésre került, néhány beavatkozó vezérlése relén keresztül történik. Erre a célra a 3.10. ábrán látható Robofun 5V egycsatornás relé modult használtam. [22] A vezérlése fordított, tehát amikor a hozzá kötött GPIO lábra 1-es értéket állítunk be, akkor kapcsol ki, amikor pedig 0-t, akkor be.

3.10. ábra: Robofun 5V relé modul [22]

3.3.2 Analóg-digitális átalakító

Mivel az általam használt vezérlő nem képes az analóg bemenet kezelésére, a szenzorok egy része viszont analóg kimenetű, így szükséges volt az AD konverter használata. A rendszerben az Adafruit ADS1115 16 bites, 4 csatornás átalakítót használtam (3.11. ábra), aminek előnye a magas felbontás, valamint az I2C interfész, ami könnyű interfészt biztosít. [24] Ahogy a többi Adafruit termékhez, ehhez is pontos dokumentáció található, sok példakóddal, amik megkönnyítik a fejlesztés menetét.

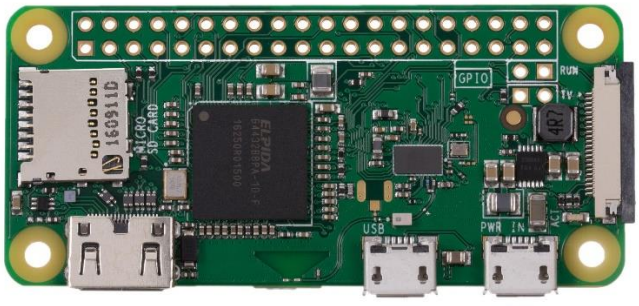
3.11. ábra: Adafruit ADS1115 16-bit ADC [24]

3.4 Vezérlés

Az érzékelők és beavatkozók összekötéséhez szükségünk van vezérlő elemekre, amiben a rendszer logikáját tudjuk elkészíteni. Erre az alábbi eszközöket használtam.

3.4.1 Raspberry Pi Zero W

A Raspberry Pi Zero (3.12. ábra) egy kisméretű, alacsony költségű számítógép, amit kifejezetten olyan alkalmazásokhoz készítettek, ahol a kis méret, alacsony ár és alacsony energiafogyasztás fontos szempont.



3.12. ábra: Raspberry Pi Zero [25]

Egy egymagos, 1 GHz-es processzort és 512 MB RAM-ot tartalmaz, ami bőven elegendő annak az egyszerű programnak, amit futtatni szeretnénk rajta. Memóriáját a többi raspberry-hez hasonlóan egy behelyezett SD kártya adja, ez a jelenlegi rendszerben egy Kingston 16 GB-os memóriakártya. Rendelkezik egy mini HDMI porttal, ami által monitort tudunk hozzá csatlakoztatni, két micro USB porttal az áramellátáshoz és a külső perifériákhoz – egér, billentyűzet –, valamint GPIO lábakkal. Beépített Wi-Fi és Bluetooth támogatást is tartalmaz, ami lehetővé teszi a vezeték nélküli kapcsolatot. [25]

3.4.2 Raspberry Pi 4 Model B

A Raspberry Pi 4 (3.13. ábra) egy erősebb modell, szélesebb körben használható, akár nagyobb projektekre is.

3.13. ábra: Raspberry Pi 4 Model B [26]

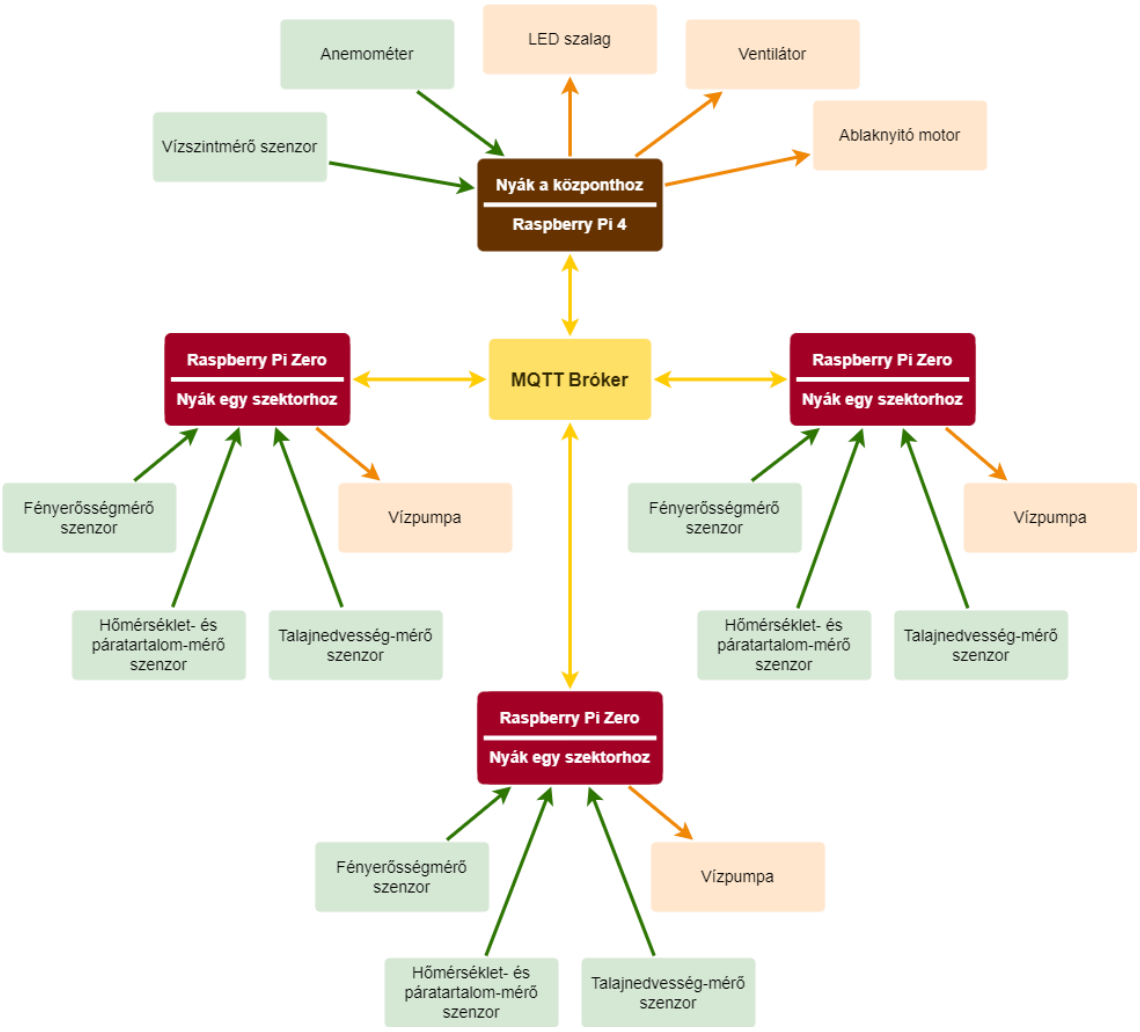
Processzora egy négymagos ARM Cortex-A72, ami a korábbi modellekhez képest jelentős teljesítményjavulást hozott, 4 GB RAM-ot tartalmaz. Számos csatlakozóval rendelkezik, ideértve két 3.0 és két 2.0 USB portot, egy Gigabit Ethernet csatlakozót, két micro HDMI kimenetet, egy 3,5 mm-es hangkimenetet, egy USB C-s csatlakozót az áramellátás biztosítására, valamint GPIO lábakat. Memóriáját szintén egy SD kártya adja, jelen rendszerben egy SanDisk 16 GB-os memóriakártya. Szintén tartalmaz beépített Bluetooth és Wi-Fi támogatást. [26]

4 A megvalósított rendszer

Az okosüvegház témájával előző félévben, önálló laboratórium keretein belül kezdtem el foglalkozni. Ekkor ez egy félbemaradt projekt volt, korábbi félévekben elkezdték vele a munkálatokat, majd én a már meglévő eszközöket felhasználva gondoltam újra és valósítottam meg a jelenlegi rendszert.

4.1 Logikai felépítés

A 4.1. ábrán láthatjuk a logikai felépítését a rendszernek, itt az érzékelőket zöld, a beavatkozókat narancssárga színnel jelöltem, a nyilak irányítása pedig a kommunikáció irányát mutatja.



4.1. ábra: A rendszer logikai felépítése

Az üvegház három szektorra lett osztva, amiknek a vezérléséért egy-egy Raspberry Pi Zero felel. Minden szektorban használunk fényérzékelő, hőmérséklet- és páratartalom-mérő, valamint talajnedvesség-mérő szenzort a környezeti tényezők megfigyelésére. A beavatkozók közül a vízpumpa kapott helyet minden szektorban, így külön tudjuk locsolni az összes szektort a talajnedvesség és az igények alapján.

A központi egységhez tartozik az anemométer és a vízszintmérő szenzor, ezen kívül ide van kötve a LED szalag, a ventilátor és az ablaknyitó motor is, ezek vezérlése a Raspberry Pi 4 feladata. Felmerülhet a kérdés, hogy miért mérjük szektoronként a páratartalmat, fényerősséget, hogyha ezek változtatását csak központilag tudjuk irányítani. Ennek a jelenlegi rendszerben költséghatékonysági okai voltak, illetve egy ilyen kis üvegházban természetesen nincs különbség a szektorok mellett mért értékekben, mivel nagyon közel helyezkednek el egymáshoz, így a nyákok is ehhez lettek tervezve. A hőmérsékletet is mérjük, azonban ehhez sem tartozik hűtő-fűtő berendezés, egy nagyobb költségvetésű projekt esetén természetesen azt is tartalmazná a rendszer.

4.2 Fizikai részegységek

Ahogy korábban is volt már róla szó, három szektorra és egy központi egységre lett osztva az üvegház, amik mindegyikéhez tartozik egy logikai vezérlő. Annak érdekében, hogy a beavatkozók és szenzorok jól strukturáltak legyenek, minden szektor vezérlőjéhez és a központi egységhez is tartozik egy nyák, ami megfelelő interfészt biztosít a használt eszközök és a Raspberry-k között.

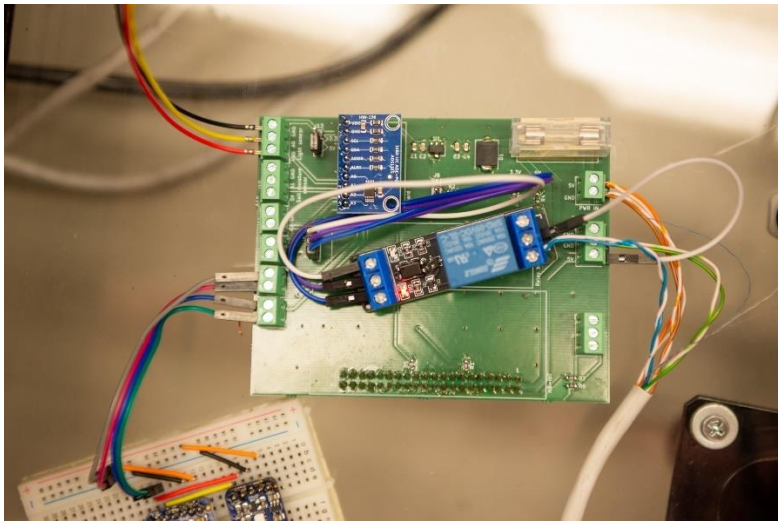
4.2.1 Szektornyák

Mivel a szektorok logikai és fizikai felépítése megegyezik, így a hozzájuk tartozó nyákok is. Ezek tervezése és elkészítése is megvalósult már, mielőtt hozzám került volna a projekt, a következőkben a szektorokhoz tartozó áramkört szeretném bemutatni.

A használt logikai vezérlő, a Raspberry Pi Zero a nyákhoz annak alján egy dupla tüskesorral csatlakozik. Nem igényel külön tápellátást, hanem azt is a boardon lévő tápból kapja.

A tesztelés során a szektornyák számára egy labortáp biztosítja a megfelelő feszültséget, azonban egy 5V, 3A-es hálózati adapterrel is működőképes lenne. Ahogy a 4.2. ábrán láthatjuk, a nyákon kapott helyet az ADC is, amire a talajnedvesség-mérő szenzor miatt volt szükség, mivel az csak analóg kimenettel rendelkezik. Kimeneti

feszültségtartománya 1.2-3V, így az ADC konstrukcióban lévő 3.3V-os táplálása megfelelő, ezzel pontosabb mérési eredményeket adva.



4.2. ábra: Szektornyák

Ahogy a képen is látjuk, több ki- és bemenettel is rendelkezik a nyák, ezek közül a felső hármat használom a talajnedvesség-mérő szenzorhoz, valamint az alsó négyet az I2C-n kommunikáló hőmérséklet- és páratartalom-mérő és a fényerősségmérő szenzorhoz. Ezek mellett a vízpumpához szükséges reléhez is kapunk egy interfészt, annak kimenetét pedig a J7-es, jobb oldalt középen található blokk egészíti ki.

Láthatunk egy lineáris regulátort is, ami 3.3V-ot állít elő, ez az I2C-s szenzorok és az ADC bemenetére van vezetve. A voltage selectort minden szektornál 5V-ra állítottam, ez adja a talajnedvesség-mérőnek a tápot. Találhatunk még egy address jumpert és helyet egy NB-IoT modulnak, ezeket a jelenlegi rendszerben nem használom.

Ezen kívül egy olvadóbiztosítékkal is rendelkezik a nyák, ami a túláram és rövidzár ellen véd.

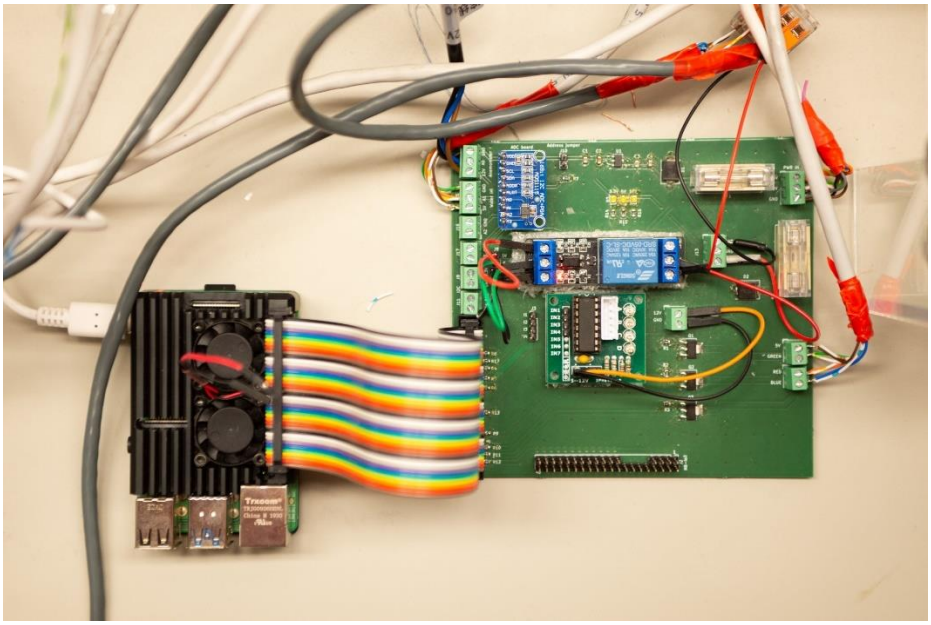
4.2.2 Központi vezérlőegység

A központi egységhez tartozó nyák is hasonló feladatokat lát el, mint a szektorokhoz tartozó, azonban más ki-és bemenetekkel rendelkezik, más eszközöket csatlakoztatunk hozzá. (4.3. ábra)

A vezérlő Raspberry Pi 4 egy negyven pines szalagkábelen keresztül csatlakozik a boardhoz. Mivel ennek nagyobb az áramfelvétele, így külön tápellátást igényel, a saját, USB-C-s csatlakozójával tudjuk áram alá helyezni.

A nyák bemenetére 5 és 12V-ot köthetünk, amit szintén a labortáp biztosít, mindkettőre szükség van a különböző eszközök miatt. Az 5V-ból egy feszültségregulátor itt is szolgáltat 3.3V-ot, amit az ADC használ. A központi egységben két analóg kimenetű szenzort is használunk, az anemométert és a vízszintmérőt, ezek mindketten az ADC bemenetére vannak kötve.

A LED szalagot, a léptetőmotort és a ventilátort a relén keresztül is innen tudjuk vezérelni, ezek megvalósításáért is a központi nyák felel.



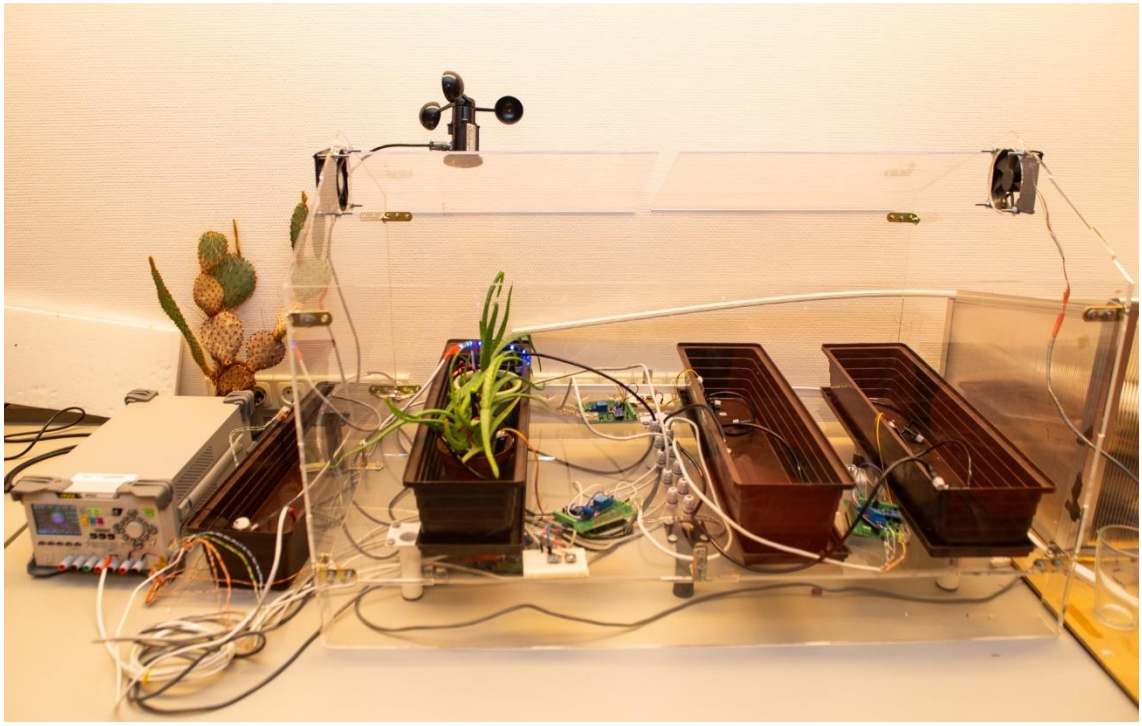
4.3. ábra: Központi vezérlőegység

4.3 Az üvegház összeállítása

A leglátványosabb és izgalmasabb rész az üvegház fizikai összeállítása volt. Ekkor minden összeállt és egyszerre kellett működnie, így természetesen ez is okozott kihívásokat, és rávilágított bizonyos hibákra.

4.3.1 A ház fizikailag

A projekt hozzám kerülésekor már készen állt egy üvegház váza a használatra. Ekkor már a keret össze volt állítva, ki volt fúrva, így ezzel már nem kellett foglalkoznom. A 4.4. képen láthatjuk jelenlegi kinézetét, anyaga plexi, méretei 100x60x60 cm, alakja valóban egy kis házra emlékeztet.



4.4. ábra: Az üvegház

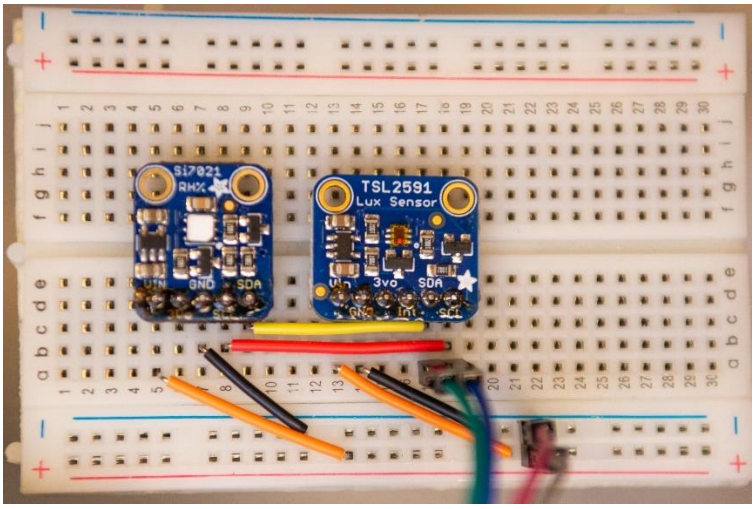
A két oldalfalon felül egy-egy lyuk található a két ventilátornak, alul pedig tömbszelencéket láthatunk, hogy a vezetékek bevezetése egyszerű és biztonságos lehessen. A vezetékek könnyebb elvezetésének érdekében az üvegház megemelt, 10 cm magas tartólábakon áll, a szélesebb oldalain pedig a plexi leér a „földig” – ez jelenleg az asztalt jelenti.

Ezen kívül láthatjuk még a barna kaspókat is, amiből három az üvegházon belül a három szektornak felel meg, egy pedig kívül a víztartály feladatát látja el. Ezekhez járt egy-egy tálca is, ezzel biztosítva, hogy a bennük lévő víz és föld ne koszolja össze a mellette lévő területet és eszközöket. A kaspókból egyébként pontosan hat fér az üvegházba, így lehet még bővíteni a rendszert.

4.3.2 Eszközök behelyezése

Annak érdekében, hogy minden eszköz a neki rendelt helyen lehessen, megfelelően hosszú kábeleket kellett biztosítani. A szektorok vezérlőegysége az üvegházon belül helyezkedik el, illetve a hozzá tartozó szenzorok is, így ezeknek megfelelőek voltak a korábban is használt, rövidebb kis kábelek. A vízpumpa és a táp eléréséhez egy UTP kábelt használtam, párosával bekötve az ereket. Az UTP kábel és a locsoláshoz szükséges csövek az üvegház közepén elhelyezkedő tömszelencéken keresztül kerülnek kivezetésre.

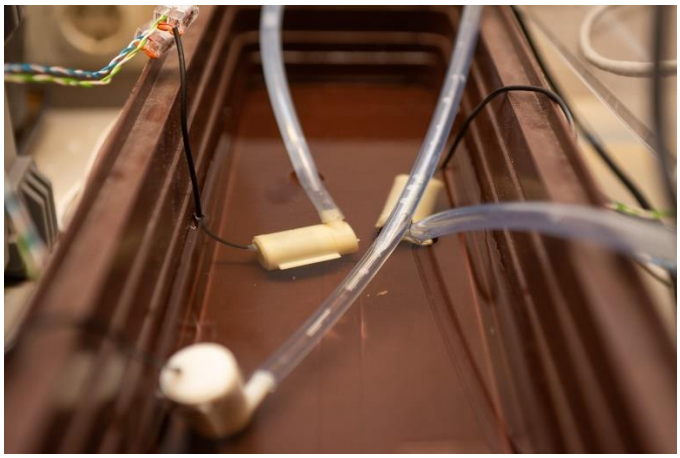
A hőmérséklet- és páratartalom-mérő, valamint a fényérzékelő szenzort egy breadboardon keresztül kapcsoltam a rendszerhez, ahogyan azt a 4.5. képen is láthatjuk, mivel a szektornyák csak egy SDA/SCL csatlakozóval rendelkezik. A két szenzornak más-más sorrendben vannak a pinjei, úgyhogy jumpereket használtam azok összekötésére.



4.5. ábra: Szenzorok a breadboardon

Az eszközök kézhez kapásakor mindkét szenzornak a túloldalán voltak a lábai, ami különösen a fényérzékelőnél volt probléma, mivel így csatlakoztatáskor pont lefelé nézett az érzékelője. Egy fényszenzor számára ez természetesen nem az ideális állapot, így mindegyik szenzoron át kellet forrasztani a pineket a túloldalra. Ez akkor is szükséges lett volna, hogyha nem használunk hozzá breadboardot, mivel a vezetékek valamennyire akkor is takartak volna.

A víztartály, benne a pumpákkal és a vízszintmérő szenzorral az üvegház mellett kapott elhelyezést, ahogyan azt a 4.4. képen is láthattuk. Alacsony víztartályra volt szükség, mivel a vízszintmérő sem túl hosszú, és a pumpákhoz tartozó vezeték sem, ezeknek a csatlakozója pedig nem érhet vízbe. Így sikerült kivezetni a csatlakozókat a tartályon kívülre, hogy ne essen bántódásuk.



4.6. ábra: A víztartály

A pumpa vezetékeinek csatlakozóját wagok segítségével sikerült megoldani, mivel, ahogy korábban említettem is, nagyon vékony kábelekről van szó, amiket nem igazán lehetett volna forrasztani sem, így ez a megoldás született. A kipumpált víz először a 4.6. képen látható átlátszó, vastagabb csőbe kerül, majd ebből vezetjük tovább abba, amivel konkrétan a locsolás történik. A használt locsolóval csepegtető öntözőrendszert valósítottunk meg, amit működés közben láthatunk a 4.7. képen.

4.7. ábra: Locsolás

A központi egységet az üvegházon kívül, az alá helyeztem el, mivel ennek több komponense szintén kívül van, így könnyebb volt megoldani a vezetékezést. A vízszintmérő szenzor elhelyezését kissé láthatjuk a 4.4. képen, a víztartály oldalához van rögzítve. Érdekesség, hogy ahogy több órán keresztül vízben volt, nagyon jól látszott, ahogy megindult az elektrolízis.

A LED szalagot az egyes szektor köré tettem, mivel egy valós rendszerben is szektoronként lenne megvalósítva a világítás. Ennek kék fényét láthatjuk a 4.7. képen megvillanni a háttérben.

Az anemométer az üvegház tetejére került, mivel ott semmi nem árnyékolja le a szelet, illetve így könnyen el is lehet vezetni a hozzá tartozó kábelt alulra, a vezérléshez.

A ventilátorokat a ház falára, a neki tervezett helyekre erősítettem fel. Az egyik ventilátor kifelé fújja a levegőt, a másik pedig befelé, így könnyebben és gyorsabban tud szellőzni. Működés közben láthatjuk a 4.8. képen. A vezetékeket az üvegház oldalán és az alatt sikerült elvezetni, így nem kellett a tömbszelencéken keresztülvinni.

4.8. ábra: A fújó ventilátor

5 Szoftverarchitektúra és implementáció

A Raspberry Pi által vezérelt rendszer megteremti a szenzorok és beavatkozók közötti összeköttetést, azonban ehhez elengedhetetlen egy erre a célra megfelelő kód implementálása. A következőkben bemutatom az üvegház szoftveres oldalát, használt programozási nyelveket, esetleges problémákat, amikbe az implementálás során belefutottam.

5.1 Operációs rendszer

A használt Raspberry-ket a Raspberry Pi Imager használatával setupoltam fel, ami elsősorban a Debian 12 használatát javasolja. Ennek nagyon örültem, mivel többször is dolgoztam már Debianos rendszeren, illetve számos előnnyel is rendelkezik.

Ez a legújabb Debian, 2023 júniusában dobták piacra, így még hosszú ideig támogatott. Rengeteg dokumentációt lehet találni hozzá, ami segít a fejlesztésben és a problémák megoldásában. Emellett az is szempont, hogy a Linux hatékonyan működik kevés erőforrással is, ami kifejezetten hasznos esetünkben, hiszen a Raspberry Pi Zero nem bővelkedik ezekben. Nagy mértékben és egyszerűen testreszabható, amire itt is szükség volt a különböző interfészek használatához. Stabil és biztonságos, ami különösen fontos egy ilyen projekt esetén, ahol valós környezetben a kihelyezés után ritkán lehet és kell hozzáférni az eszközökhöz. [27]

Egyetlen apró buktató volt, a Raspberry Pi Imager a 64 bites Debiant javasolja, viszont a zero csak 32 bites rendszeren működik, így végül második körben azt kellett rá feltelepíteni. Ettől eltekintve viszont nem adódtak problémák a telepítés során, az imager megengedi, hogy már ekkor beállítsunk bizonyos paramétereket, az engedélyezett interfészeket, ssh kapcsolatot, hálózatokat, amikre automatikusan csatlakozik az eszköz.

Az egyik zero-t még előző félévben, önálló labor keretein belül üzemeltem be, így azon Debian 11 fut, hiszen akkor még az volt az aktuális legújabb verzió. Végül nem cseréltem le, mivel még ez az operációs rendszer is sokáig támogatott, használatában nincs különbség a 12-től.

5.2 Implementáció

Egy ilyen projekt esetén, valamint Raspberry használata mellett valószínűleg mindenkinek az lenne a logikus lépés, hogy pythonban kódoljon, így természetesen én is erre a következtetésre jutottam.

5.2.1 A Python előnyei

Bár egyetemi keretek között nem tanultunk python-ban programozni, könnyű olvashatósága és egyszerűsége miatt nagyon gyorsan tanulható. Magas szintű nyelvi elemeket tartalmaz, ami meggyorsítja a fejlesztés menetét és megkönnyíti a karbantartást. Platformfüggetlen, tehát ugyanazt a kódot futtathatjuk Linux és Windows alapú rendszereken is. Nagy és aktív fejlesztői közösséggel rendelkezik, ami rengeteg támogatást jelent, ha most kezdünk belemerülni a nyelvbe jobban. [28]

Ezenkívül rengeteg külső könyvtárat és modult kínál a széleskörű funkciók eléréséhez, ami magában foglalja az IoT szenzorok kezelését is. Ezen könyvtárak közé tartozik többek között az RPi.GPIO is, amivel a Raspberry GPIO lábait tudjuk könnyedén beállítani, valamint az Adafruit-Blinka is, ami az adafruitos szenzorok olvasásához elengedhetetlen.

5.2.2 Csatlakoztatott eszközök elérése

Az első feladat az volt, hogy a kódból le tudjuk kérni a különböző szenzorok aktuális értékeit. Szerencsére minden szenzorhoz találtam dokumentációt példakóddal, beimportálható könyvtárral, ideértve az AD átalakítót is. Ennek következtében ezen eszközök inicializálásához csupán ez a pár sor kellett [29]:

```
import adafruit_tsl2591
import adafruit_si7021
import Adafruit_ADS1x15

i2c = board.I2C()
lightsensor = adafruit_tsl2591.TSL2591(i2c)
thsensor = adafruit_si7021.SI7021(i2c)
adc = Adafruit_ADS1x15.ADS1115()
```

Okozott egy kis problémát, hogy eleinte nem találta az I2C eszközt, mivel nem tudtam, hogy nem a default, 0x48-as címen volt, hanem a 0x49-esen. Az *i2cdetect* paranccsal végül könnyen fel tudtam térképezni, melyik eszköz milyen címen van, így már az ADC által küldött értékeket is ki tudtam olvasni, ami az analóg szenzoroktól származott. Végül kiderült, hogy a nyákon az address jumper összeköttetésben maradt az

ADC-vel, és az okozta ezt a problémát, így azóta eltávolítottam, és már a 0x48-as címet használom.

A fő egységnél jött elő az a probléma, hogy eleinte nem találtam, a relé melyik pinre van kötve. Ennek kiderítésére multiméterrel megmértem a relé bemenetén található feszültséget, valamint az egyes lábakét is, így hamar megtaláltam a szükséges lábat.

Emellett a LED vezérléséhez is külön módszer kellett, mert a GPIO lábak csak két értéket tudnak felvenni, vagy küldünk rájuk áramot, vagy nem, ez azonban nem megfelelő arra, hogy bármilyen RGB színkombinációt elő tudjunk állítani. Ennek a megoldására használtam a PWM, azaz Pulse Width Modulation technikát, ami lehetővé teszi, hogy egy digitális jellel közelítőleg utánozzunk egy analóg jelet. Segítségével megadhatjuk, hogy milyen frekvenciával és milyen kitöltési tényezővel küldünk áramot az adott pinre. Így tehát hogyha megadjuk, hogy a piros színre kötött láb 100%-ban legyen aktív, a zöld 60%-ban, a kék pedig 10%-ban, ami az *on 100 60 10* MQTT üzenetben érkezik, akkor az RGB(255, 153, 25) narancssárga színt kapjuk a LED szalagon. A vezérlésből egy kódrészlet [30]:

```
red = 16
GPIO.setup(red, GPIO.OUT)
r = GPIO.PWM(red, 100)
…
def on_message(client, userdata, msg):
    if (msg.topic).find("light") != -1:
        if (msg.payload.decode()).find("on") != -1:
            print("turn on light")
            rgb = msg.payload.decode().split(" ")
            r.start(int(rgb[1]))
            …
        if (msg.payload.decode()).find("off") != -1:
            print("turn off light")
            r.stop()
            …
```

5.2.3 MQTT kliens

A kódban a Paho MQTT klienst használtam, ami egyszerű interfésszel rendelkezik, és tökéletesen megfelel a célnak. [31] Az újrafelhasználhatóság érdekében külön fájlba került a kliens inicializálása, a feliratkozás és a publikálás is, így ezek mind importálva vannak a fő kódba.

Az eszközöket jelenleg 3 másodpercenként olvassa a kód, és ugyanennyi időként publikálja ezeket az értékeket MQTT üzenetben a brókernek. A feliratkozás a szükséges topicokra a futás elején történik, ahol az eredeti üzenetkezelő függvényt felül is írjuk egy

sajáttal. Ennek a feladata az, hogy bizonyos üzenetekre helyesen, személyre szabottan tudjon reagálni a program, amire egyik példa a szektorokban a locsolás vezérlése. Ennek során feliratkozunk a *FIM3VE/general/waterlevel* topicra, amire a fő egység 500-as vízszint alatt *danger* üzenetet küld, felette pedig *enough*-t. Amennyiben a legutolsó üzenet *danger* volt, akkor, ha akarjuk sem tudjuk bekapcsolni a locsolást, ugyanis leégne a locsoló motorja elegendő víz hiányában. Amennyiben megérkezik az *enough* üzenet, ismét engedélyezve lesz a locsolás, amire az utasítást a *FIM3VE/sector<szektor_száma>/water* topicon kapjuk.

```
global KEVES_VIZ
KEVES_VIZ = True

def on_message(client, userdata, msg):
    global KEVES_VIZ
    print(f"Recieved `{msg.payload.decode()}` from `{msg.topic}` topic")
    if (msg.topic).find("sector3/water") != -1:
        if not KEVES_VIZ:
            print("locsol")
            GPIO.output(relay, False)
            time.sleep(5.0)
            GPIO.output(relay, True)

    if (msg.topic).find("general/waterlevel") != -1:
        if (msg.payload.decode()).find("danger") != -1:
            KEVES_VIZ = True
        elif (msg.payload.decode()).find("enough") != -1:
            KEVES_VIZ = False

subscriber.subscribe(client, "FIM3VE/general/waterlevel", on_message)
subscriber.subscribe(client, "FIM3VE/sector3/water", on_message)
subscriber.run(client)
```

5.3 Automatikus indítás

Fontos, hogy amint áram alá kerül a Raspberry, elinduljon rajta magától a program, ne kelljen semmilyen emberi beavatkozás hozzá. Erre a célra először a crontabot használtam, azonban az nem hozta a kívánt eredményt, így más eszközt kerestem. A használt operációs rendszer systemd-t használ, ami átfogó és hatékony kezelést biztosít a rendszernek indulásától a leállításig, valamint a futó szolgáltatások számára. Konfigurációs fájlokban tárolja az indítási és szolgáltatási beállításokat, elkerülve ezzel a hagyományos parancsfájlok használatát. Az általa vezérelt rendszerek közös naplófájlrendszert használnak, ami lehetővé teszi a különböző szolgáltatások naplóbejegyzéseinek csoportosítását és könnyű kezelését. Célja az indítási folyamatok párhuzamosítása, hogy ezzel meggyorsítsa a rendszerindítást. [32]

Szerettem volna, hogy az általam írt python script is elinduljon a rendszerrel együtt, így létrehoztam az alábbi systemd service konfigurációs fájlt.

```
[Unit]
Description=Start Monitoring Service
After=network.target

[Service]
ExecStart=/usr/bin/python3 /home/uveghaz/kod/main.py
WorkingDirectory=/home/uveghaz/kod/
Restart=always
User=uveghaz
Group=users
Environment=PATH=/usr/bin:/usr/local/bin
Environment=NODE_ENV=production
RestartSec=3
StandardOutput=syslog
StandardError=syslog
SyslogIdentifier=start_monitoring

[Install]
WantedBy=multi-user.target
```

A szolgáltatás neve *Start Monitoring* lett, és mivel a kód helyes futásához szükség van arra, hogy hálózatra kapcsolódjon az eszköz, így az *After* paraméterben megadtam, hogy csak a hálózat beállítása után futtassa ezt a szolgáltatást. A *Restart=always* sor is egyértelműen indokolt, hogyha valami esetleg hibát okozna a programban, nem tudna wifire csatlakozni egyből, minél kevésbé legyen ez a felhasználó számára észrevehető, próbálkozzon újra a program. [33]

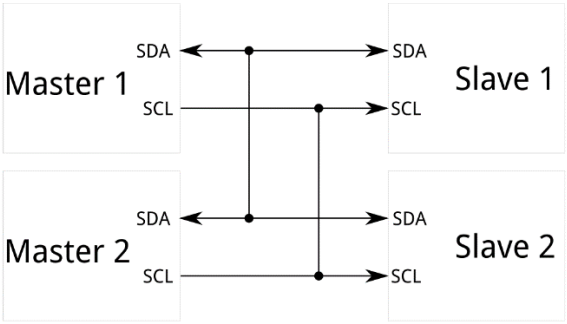
Ezek után már csak arra volt szükség, hogy a *main.py* fájlnak helyesen állítsam be a jogosultságait, a *users* group tudja futtatni, illetve az *uveghaz* felhasználó birtokolja azt. A szolgáltatás indítása és engedélyezése után elindult a kód, és minden újraindításnál ezt sikeresen meg is teszi.

6 Kommunikáció

A rendszer fontos pontja a kommunikáció a szenzorok, beavatkozók és a vezérlés között, amire az eszközöktől függően több különböző technológiát ismerhettem meg, amik közül az egyik legfontosabb az I2C kommunikáció. Ezek mellett lényeges pont a szektorok és a központi egység közötti, valamint az ezek és a felhasználó közötti adatátvitel, aki számára szükséges, hogy lássa a mért adatokat. Olyan technológiára van szükségünk, ami hatékonyan tudja kezelni a több forrásból érkező és több címzett számára küldött üzeneteket, erre a célra pedig ideális megoldást nyújt az MQTT (Message Queuing Telemetry Transport).

6.1 I2C

Az Inter-Integrated Circuit protokoll olyan kommunikációs forma, amely lehetővé teszi, hogy több slave IC kommunikáljon egy vagy több masterrel. A master felel a kommunikációért, ő a főnök, a slave pedig – hiszen jelentése szolga – az, akitől lekérjük, vagy neki küldjük az információt. A többszereplős kommunikációt úgy valósítja meg, hogy minden eszköz egyedi azonosítóval rendelkezik, így nem keverednek össze az adataik. A működés vázlatos rajzát a 6.1. ábrán láthatjuk.

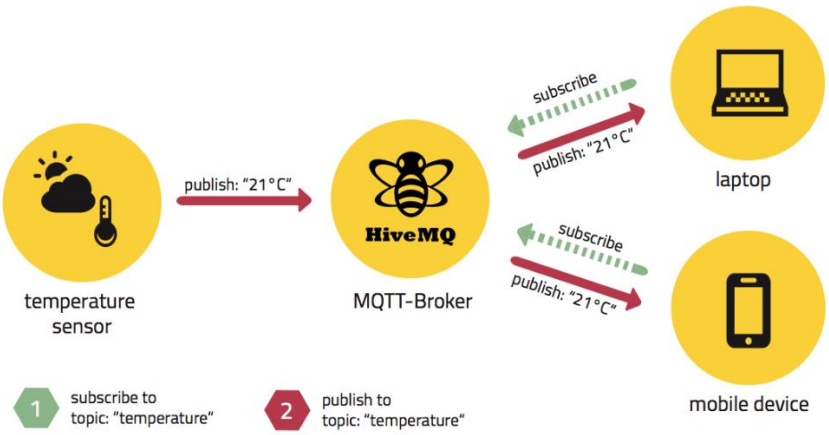


6.1. ábra: I2C kommunikáció [34]

Kétvezetékes soros adatátvitelt tesz lehetővé, ahol a két vezeték az SDA – Serial Data Line –, amin az adatok továbbítódnak, és az SCL – Serial Clock Line –, amin az órajel pulzusokat küldi az aktuális master, aki generálta. Általában alacsony adatátviteli sebességgel működik, 100 kHz-en vagy 400 kHz-en. [34]

6.2 MQTT

Az MQTT egy nyílt, ingyenes protokoll, amit arra terveztek, hogy megbízható üzenetküldést valósítson meg a lehető legegyszerűbb üzenetformátumban. A kommunikációban három típusú fél vesz részt, broker, publisher és subscriber. A 6.2. ábrán láthatunk egy példát az MQTT alapú kommunikációra.



6.2. ábra: Az MQTT működése [35]

6.2.1 Topicok

Az üzenetek megkülönböztetésére, osztályzására az úgynevezett topicok szolgálnak, ezek definiálják az üzenet tartalmát. Általában hierarchikusan szervezettek, a „/” karakter használatával tudunk létrehozni al-topicokat. Egy „/” karakter utáni „#” jellel tudunk feliratkozni az adott topic összes al-topicjára.

6.2.2 Résztvevők feladatai

A subscriberek, nevükből adódóan, feliratkozhatnak a különböző topicokra, ami után az összes, arra a topicra publikált üzenetet megkapják. Nem ismerik a publishereket, csak a brókert. A publisherek feladata, hogy az általuk meghatározott topicra publikáljanak. Nekik nem szükséges tudni, hányan és kik iratkoztak fel a témáikra, csak a bróker kilétéről van tudomásuk. Egy kliens lehet egyszerre publisher és subscriber is, ez a kettő nem zárja ki egymást, sőt egy kliens több témára is nyugodtan felirakozhat. A bróker feladata, hogy menedzselje a kliensek közti üzenetküldést, hogy minden subscriber megkapja az általa rendelt üzeneteket. A jelenlegi rendszerben egy online elérhető publikus brókert használtam a HiveMQ oldaláról. [35]

6.2.3 Szolgáltatási szintek

Három szolgáltatási szintet definiáltak az MQTT-ben. A magasabb szolgáltatásminőség nagyobb megbízhatóságú üzenet célba juttatást valósít meg, de természetesen ennek ára is van, ami a nagyobb sávszélesség és/vagy késleltetés.

A szerver megtartja az utoljára elküldött üzenetet, és egy új feliratkozó esetén egyből elküldi ezt is a kliensnek. A szolgáltatási szintek abban különböznek egymástól, hogy QoS = 0 esetén a szerver legfeljebb egyszer, QoS = 1 esetén legalább egyszer, QoS = 2 esetén pedig pontosan egyszer küldi el a megőrzött üzeneteket. [35]

6.2.4 Kliensek lehetőségei

A klienseket egy 23 bájtos egyedi string azonosítja. Amikor csatlakozik egy kliens a szerverhez, beállíthat egy clean-session flaget, aminek 1-es értéke esetén a kliens összes feliratkozása törlődni fog, ha az eszköz lekapcsolódik a szerverről. Nulla érték esetén a kliens előfizetése egészen addig élő marad, amíg vissza nem kapcsolódik, és ekkor az összes addigi üzenet elküldésre kerül neki.

Ezek mellett egy végrendeletet (will) is megadhatnak, ami által, ha a kliens váratlanul lecsatlakozik, akkor a szerver egy üzenetet küld a kliens által előre meghatározott topicra. Ilyen lehet akár egy riasztás, ha egy érzékelő lecsatlakozott. [35]

6.2.5 MQTT a jelenlegi rendszerben

Az üvegház fő kommunikációja MQTT alapon megy, így lényeges tudnunk, milyen információt milyen topicon keresztül találunk meg, kik iratkoztak fel rá és ki küldi az üzeneteket. A 6.3. ábrán láthatjuk a rendszerben használt topicokat és minden fontos információt róluk.

FIM3VE/

- **general/**
  - **waterlevel**: A vízszintmérő szenzor értéke, a központi egység a publisher, az összes szektor és az android app a subscriber, kritikus érték alá csökkenéskor „danger”, felé emelkedéskor „enough” üzenet
  - **windlevel**: Az anemométer értéke, a központi egység a publisher, az android app a subscriber
  - **ventilator**: Ide érkezik a jelzés, hogy induljon be a ventilátor, az android app a publisher, a központi egység a subscriber
  - **window**: Ide érkezik a jelzés, hogy nyíljon ki vagy csukódjon be az ablak, az android app a publisher, a központi egység a subscriber. Értéke „open” vagy „close”
  - **light**: Ide érkezik a jelzés, hogy kapcsolódjon le vagy fel az égősor, az android app a publisher, a központi egység a subscriber. Értéke lekapcsoláskor „off”, felkapcsoláskor pedig „on <red> <green> <blue>”, ahol a red, green, blue értékek az adott színnek megfelelő égő erősségét adják meg 0-tól 100-ig.
- **sector1/**
  - **temperature**: Hőmérsékletmérő szenzor értéke, az egyes szektor a publisher, az android app a subscriber
  - **humidity**: Páratartalom-mérő szenzor értéke, az egyes szektor a publisher, az android app a subscriber
  - **soilmoisture**: A talajnedvesség-mérő szenzor értéke, az egyes szektor a publisher, az android app a subscriber
  - **lightness**: Fényerősségmérő szenzor értéke, az egyes szektor a publisher, az android app a subscriber
  - **water**: Ide érkezik a jelzés, hogy beinduljon a vízpumpa az egyes szektorban, az android app a publisher, egyes szektor a subscriber
- **sector2/**
  - **temperature**: Hőmérsékletmérő szenzor értéke, a kettes szektor a publisher, az android app a subscriber
  - …
- **sector3/**
  - **temperature**: Hőmérsékletmérő szenzor értéke, a hármas szektor a publisher, az android app a subscriber
  - …

6.3. ábra: A használt MQTT topicok és annak tulajdonságai

7 Android alkalmazás

Felhasználói oldalról természetesen akkor van igazán haszna egy okosüvegházнak, hogyha távolról is tudjuk figyelni az aktuális állapotokat és esetleg be is tudunk avatkozni. Erre a célra két lehetőség fordult meg a fejemben, egy telefonos alkalmazás vagy egy weboldal. Mindkettőnek megvannak az előnyei és a hátrányai, viszont az Androidalapú szoftverfejlesztés tárgy keretein belül jobban meg tudtam ismerni az Androidra való fejlesztést, így végül azt választottam. Az alkalmazás ikonját a 7.1. ábrán láthatjuk.

7.1. ábra: A GreenHouse alkalmazás ikonja

7.1 Architektúra és technológiai háttér

A fejlesztés az Android hivatalos nyelvén, Kotlinban történt, ami egy modern, statikusan típusos programozási nyelv. Teljes mértékben kompatibilis a Java-val, illetve elég hasonlóak is, így nem nehéz megtanulni a szintaktikáját a Java után. Támogatja a funkcionális és az objektumorientált programozást is, a kód nagyon szépen olvasható, értelmezhető.

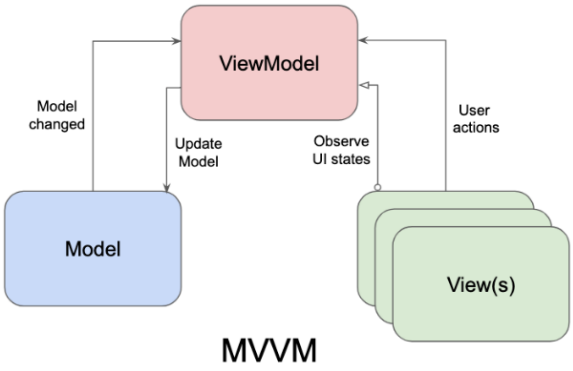
7.1.1 Android Room

Az adatbázis kezelésére az Android Roomot használtam, ami a háttérben SQLite adatbázist használ, és ahhoz biztosít absztrakt réteget, ami által az alkalmazásban könnyen és hatékonyan kezelhetjük a lokális adatokat. Együttműködik az ORM, azaz Object-Relational Mapping koncepcióval, ami alapján minden osztály egy tábla, minden objektum egy sor a táblában és minden attribútum egy mező. [37]

Bár a legtöbb adatot az MQTT brókertől kapja az alkalmazás, a használt topicokat érdemes elmenteni, hogy ne kelljen minden alkalommal újra beírni a felhasználónak. A mentett adatok tehát minden szektorhoz egy név, egy MQTT topic, illetve a benne termesztett növények, amiket megadhat a felhasználó. Ezeket természetesen szerkeszteni és törölni is lehet.

7.1.2 MVVM architektúra

Az alkalmazást az MVVM, azaz Model-View-ViewModel architektúra alapján építettem fel (7.2. ábra). A Model a legalsó réteg, ami az alkalmazás üzleti logikáját és adatmodelljét tartalmazza, független a felhasználói interfésztől. A View a legfelső réteg, ami a felhasználói interfészt reprezentálja, passzív, és csak a ViewModel-től kapott adatokat jeleníti meg. Köztük áll a ViewModel, ami a Model által szolgáltatott adatokat fordítja olyan formába, ami könnyen megjeleníthető a View számára. Ő felelős a felhasználói interakciók kezeléséért és a megfelelő műveletek elindításáért. A kommunikáció tehát a szomszédos rétegek között kétirányú. [38]



7.2. ábra: MVVM architektúra [39]

7.1.3 Paho MQTT kliens

Több olyan csomag is elérhető az interneten, aminek segítségével androidos alkalmazásban tudunk MQTT üzeneteket kezelni, ezek közül a Paho MQTT klienst választottam, mivel ehhez találtam a legtöbb dokumentációt, példakódot, segédletet. A példakódok miatt a rendszerbe való integrálása eleinte nagyon könnyen ment, viszont hamar abba a problémába ütköztem, hogy az egyik fájlban egy olyan flaget használnak, ami Android 12-től már nem támogatott. Mivel ezt a kódot nem én írtam, hanem az importált csomag használja, így nem tudtam csak egyszerűen átírni. Szerencsére azonban

nyílt forráskódú, githubon megtalálható, és már több pull request is érkezett ennek a problémának a kijavítására, így tudtam találni egy olyan változatot, amit már egy felhasználó kijavított, és a jitpack segítségével az ő kódját tudtam használni. [40]

7.1.4 Navigáció

A programon belüli navigációhoz az Android navigation component könyvtárát használtam, ami tartalmaz egy NavHost-ot és egy NavController-t. Első lépésként screen-eket, képernyőket kell definiálnunk, amiket odaadhatunk a NavHost-nak. A NavHost egy compose komponens, amiben megadhatjuk, melyik screennel akarjuk indítani az alkalmazást, illetve az adott screenekben melyik navigációs gombra nyomva melyik képernyőre navigáljunk. Összességében elég egyszerű és átlátható navigációt tudunk készíteni ezzel a technológiával, ami könnyen módosítható és megbízható. [41] A képernyők között így még információt is tudunk továbbítani, ami jelen programban is hasznos, amikor kiválasztunk a szektorok listájából egyet, hiszen akkor a következő képernyőnek szüksége van arra az információra, hogy melyik konkrét szektor adatait töltse be.

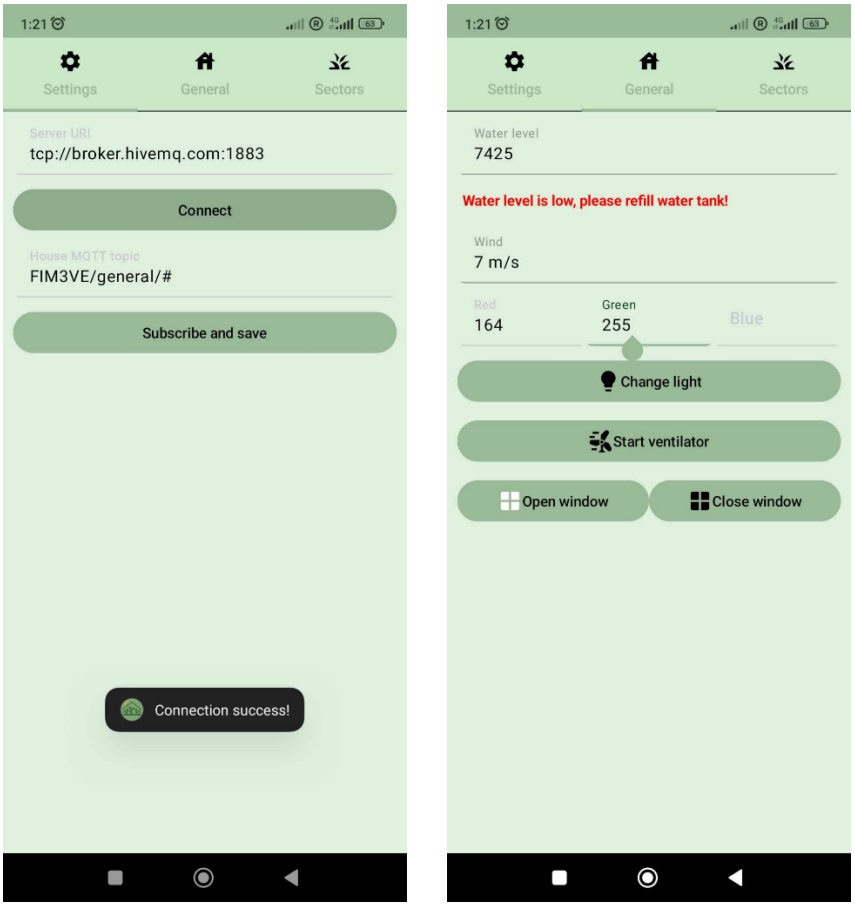
7.1.5 Jetpack compose

A Jetpack Compose egy modern toolkit UI készítéséhez, általa a Kotlin kódban írhatjuk le a UI paramétereit, ami alapján a compose motor generálja a felületet. Előnyei, hogy kevesebb kóddal tudjuk ugyanazt a nézetet elérni, nincs szükség hozzá a korábban megszokott XML layoutra. Könnyebben újrafelhasználható, hatékonyabb, illetve sokkal egyszerűbb így eljuttatni a megjelenítendő adatokat a felülethez. Készíthetünk hozzá preview-kat, aminek köszönhetően nem kell lefuttatni a programot ahhoz, hogy egy adott nézet elrendezését megtekinthessük, ami különösen a fejlesztés korai szakaszában nagyon hasznos. [42] A fentebb említett Androidalapú szoftverfejlesztés tárgy keretein belül jobban megismerhettem ezt a technológiát, így emiatt és sok előnyös tulajdonsága miatt természetes volt, hogy most is ezt fogom választani.

7.2 Felhasználói felület

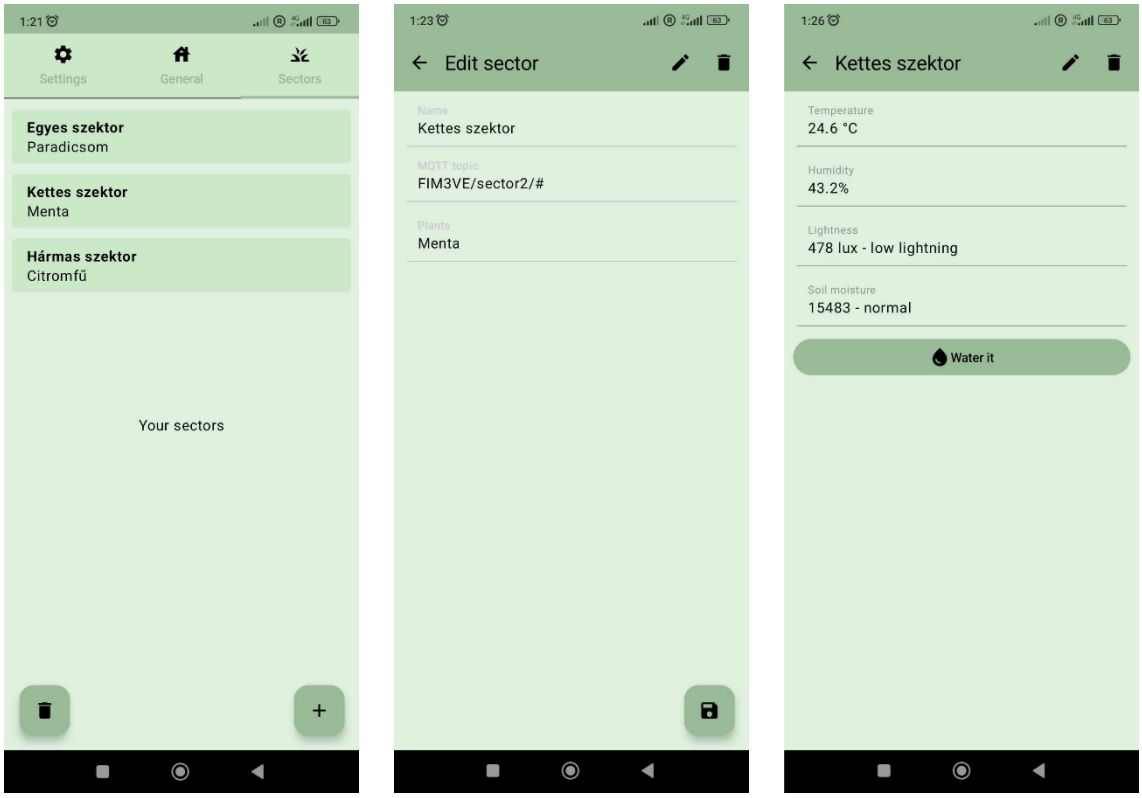
Az alkalmazás három fő képernyővel rendelkezik. Megnyitáskor a *Settings* képernyőn (7.3. ábra, bal oldali képernyőkép) találjuk magunkat, ahol beállíthatjuk, melyik MQTT brókerhez akarunk csatlakozni, illetve megadhatjuk az üvegház fő egységének az MQTT topicját. Amennyiben sikeres volt a kapcsolódás, egy Toast üzenetet láthatunk, ami ezt megerősíti.

Átlépve a *General* fülre találhatjuk meg a központi egységhez csatolt érzékelőket és beavatkozókat (7.3. ábra jobb oldali képernyőkép). Amennyiben 8000 alatt van a *Water level* érték, az alkalmazás egy figyelmeztető üzenettel jelzi, hogy kevés víz van a tartályban, töltsük fel azt. Hogyha elegendő a víz, az üzenet eltűnik. A *Red, Green, Blue* mezőkben 0-255-ig adhatjuk meg, milyen RGB értéket vegyen fel a LED világítás, ami akkor kerül beállításra, ha a *Change light* gombra rányomunk, aminek feladata elküldeni az MQTT üzenetet a megfelelő topicra. A *Start ventilator* gombbal a ventilátort indíthatjuk el 30 másodpercre, az *Open window* gombbal az ablakot kinyithatjuk, a *Close window* gombbal pedig bezárhatjuk.



7.3. ábra: Settings és General képernyők

A *Sectors* fülre kattintva láthatjuk a szektorokat felsorolva a listában (7.4. ábra, bal oldali képernyőkép). Hogyha szeretnénk újat hozzáadni, azt a jobb alsó sarokban található plusz gombbal tehetjük meg, ekkor egy hasonló képernyő jelenik meg, mint ami a 7.4. ábrán a középső képernyőképen látható. Mentés után ismét a szektorok listájához kerülünk, ahol egy adott szektorra kattintva megjelenik a 7.4. ábra jobb oldali képernyőképén látható ablak, ahol az adott szektorhoz tartozó szenzorok értékeit láthatjuk. A *Lightness* blokkban 700 lux alatt a szöveg *low lightning,* 700-2000 lux között *normal lightning,* 2000 lux felett pedig *strong lightning*. A talajnedvesség értéke 10000 alatt *wet*, 10000 és 20000 között *normal*, 20000 felett pedig *dry*. A *Water it* gombra kattintva elindíthatjuk a locsolót, ami adott 30 másodpercig vizet pumpál a szektorhoz tartozó földbe. A szektort a jobb felső sarokban lévő ceruza ikonra kattintva tudjuk szerkeszteni, a kuka ikonnal pedig törölni. Az összes szektor együttes törlésére is lehetőség van a szektorok listájának a képernyőjén, a bal alsó sarokban elhelyezkedő kuka lebegő gombbal.



7.4. ábra: Sectors képernyő változatai

8 Összefoglaló és fejlesztési lehetőségek

Összességében úgy gondolom, nagyon jól döntöttem a témaválasztáskor, hiszen valódi eszközökkel foglalkozhattam, a végén kialakult egy fizikai rendszer, az eredmény szabad szemmel látható lett. A növénytermesztés hatékonyabbá tétele egy valós probléma, amire egy jó megoldás az okosüvegházak használata, így ténylegesen hasznosnak éreztem a munkát.

A megvalósított rendszerrel elégedett vagyok, a tervezett egységek szinte mindegyikét tartalmazza, az elgondolt funkciókat sikerült beépíteni a rendszerbe. A használt eszközök beváltnak bizonyulnak, különösen az I2C-s szenzorokkal vagyok nagyon megelégedve. A Raspberry is rendkívül megbízhatóan működik, nagyon örültem, hogy ennyi internetes segédanyagot sikerült találni ebben a témában.

Az üvegház összerakása különösen izgalmas volt, hiszen mi, informatikusok nem gyakran végzünk olyan munkát, aminek ilyen látványos és valóban kézzel fogható kimenete van. Mindig örömmel töltött el, amikor egy-egy rész elkezdett működni, bevált az általam írt kód, az elképzelt elrendezés.

Néhány olyan eszköz még rendelkezésemre állt, amit végül nem integráltam a rendszerbe, ezek használatával lehetne fejleszteni az üvegházat. Az egyik ilyen az ablaknyitó motor, amit terveztem beletenni, így az alkalmazásban létre is hoztam az *open* és *close* gombokat az ablak számára. A Raspberry jelenleg csak kinaplózza, hogy mikor történne az ablaknyitás, viszont valójában nem kapcsolódik a mikrokontrollerhez.

A másik ilyen eszköz az NB-IoT modul, ami eredetileg minden szektor és a központi egység részét képezte volna, hogy azon keresztül történjen a kommunikáció. Ennek előnye, hogy nagyobb a hatótávolsága, mint a wifinek, illetve így nem igényelt volna a rendszer internethozzáférést. Végül maradt a wifis megoldás, mivel mindegyik Raspberry képes rá, ezenkívül pedig az MQTT kommunikációt is egyszerűbb volt így megvalósítani, mivel interneten keresztül el lehet érni a publikus brókert, amit az Android alkalmazással is könnyedén megtalálunk.

Egy másik továbbfejlesztési lehetőség pont az MQTT-hez kapcsolódik, és nem más, mint egy saját MQTT bróker létrehozása. A mostani rendszer így sérülékenyebb, bárki küldhet ezekre a topicokra bármit. Egy valós rendszerben természetesen ezt nem lehetne megtenni, semmilyen biztonságot nem tudnánk így garantálni. Több oldal is

létezik, ahol saját brókert tudunk készíteni, ebben az esetben a kliensek is egyedi azonosítóval rendelkeznek, így minden szempontból hasznos fejlesztés lenne.

Ezeken kívül természetesen sokféle automatizációt tudna tartalmazni a rendszer, el lehetne menteni, milyen növény milyen igényekkel rendelkezik, ehhez milyen beavatkozások kellenek a szenzorértékektől függően. Integrálhatnánk a rendszerbe mesterséges intelligenciát, ami az adatokból tanulva kiszámolja, mi az ideális a termesztett növényeknek, alkalmazhatnánk hűtő-fűtő, kártevőmentesítő rendszereket. A lehetőségek tárháza tehát végtelen, ez egy olyan terület és egy olyan projekt, amiben határ a csillagos ég a fejlesztések terén, így remélem, a jövőben is lehetőségem nyílik majd arra, hogy ehhez hasonló projektekkel foglalkozhassak.

Irodalomjegyzék

[1] Mezőgazdasági gépek első megjelenése: https://versatile.hu/mezogazdasagi-gepek-elso-megjelenese/ (2023. 11. 28.)

[2] Az öntözés története: http://www.ontozesmuzeum.hu/az-ontozes-tortenete/ (2023. 11. 28.)

[3] Mi az a precíziós gazdálkodás és hogyan kezdjünk hozzá? https://agrarkozosseg.hu/mi-az-a-precizios-gazdalkodas-es-hogyan-kezdjunk-hozza/ (2023. 11. 28.)

[4] Robotok és mezőgazdasági technológia: automatizált betakarítás és feldolgozás, https://geldoblog.com/robotok-es-mezogazdasagi-technologia-automatizalt-betakaritas-es-feldolgozas/ (2023. 11. 28.)

[5] A hidroponika előnyei és a hidroponikus rendszerek alapjai: https://geoperlit.hu/hidroponika-elonyei-es-a-hidroponikus-rendszerek-alapjai/ (2023. 11. 28.)

[6] National geographic: Mesterséges intelligencia a mezőgazdaságban, https://ng.24.hu/tudomany/2022/02/25/mesterseges-intelligencia-a-mezogazdasagban/ (2023. 11. 28.)

[7] Szobanövények természetes és mesterséges fényigényei: https://www.kertelunk.hu/szobanovenyek-fenyigenyei.html (2023. 11. 29.)

[8] A szobanövények téli művi megvilágítása: https://ujszo.com/a-szobanovenyek-teli-muvi-megvilagitasa (2023. 11. 29.)

[9] Öntözés – Mikor? Miért? Hogyan? https://plantebudapest.com/novenygondozas/ontozes-mikor-miert-hogyan/ (2023. 11. 29.)

[10] Hőmérséklet a növénytermesztésben: https://www.tuja.hu/kerteszeti-lexikon/homerseklet.html (2023. 11. 29.)

[11] A páratartalom és szabályozása: https://agroforum.hu/szakcikkek/zoldseg/a-paratartalom-es-szabalyozasa/ (2023. 11. 29.)

[12] SMARTKAS: https://smartkas.com/hu (2023. 12. 03.)

[13] Plantee: https://plant.ee/products/plantee (2023. 12. 04.)

[14] Adafruit Si7021 Temperature + Humidity Sensor: https://learn.adafruit.com/adafruit-si7021-temperature-plus-humidity-sensor/overview (2023. 11. 22.)

[15] Adafruit TSL2591 High Dynamic Range Digital Light Sensor - STEMMA QT: https://www.adafruit.com/product/1980 (2023. 11. 22.)

[16] Talajnedvesség-szenzor (kapacitív elvű, v1.2): https://shop.tavir.hu/termek/shop/modulok/paratartalom/talajnedvesseg-szenzor-kapacitiv-elvu/ (2023. 11. 22.)

[17] Anemometer - Kanalas szélsebességmérő szenzor Analóg feszültség kimenettel: https://www.rpibolt.hu/Anemometer-Kanalas-Szelsebessegmero-szenzor-Analog (2023. 11. 22.)

[18] WLD-75 Water-level detection sensor: https://www.hestore.hu/prod_10035547.html?gross_price_view=1&source=gads&gclid=CjwKCAiAvJarBhA1EiwAGgZl0IMCQzMqlmMBPgZjxgXyo_TqNToJvospQtwJivBJKyRnPby6Urxh5hoCEmEQAvD_BwE (2023. 11. 22.)

[19] Kis vízpumpa: https://techfun.sk/hu/produkt/mala-vodna-pumpa/?lang=hu&currency=HUF&gad_source=1&gclid=CjwKCAiAvJarBhA1EiwAGgZl0BRF2Ero8VwJTq2A1-kJINvrB7iaMnjc3gJdfyfztAVq2HaNYvLRGRoCpUMQAvD_BwE (2023. 11. 22.)

[20] Léptetőmotor 28BYJ-48 ULN2003 modul: https://techfun.sk/hu/produkt/krokovy-motor-28byj-48-modul-uln2003/?lang=hu&currency=HUF&gad_source=1&gclid=CjwKCAiAvJarBhA1EiwAGgZl0M7Sts0kw7UUTPDk1oefc31a4BSb-QQhe0kwLVWb_VARA8V2t9ZRihoCzEcQAvD_BwE (2023. 11. 22.)

[21] 12V 2 tűs ventilátor: https://techfun.sk/hu/produkt/ventilator-12v-2-pinovy/?attribute_pa_variant=4010-40-x-40-x-10-mm&lang=hu&currency=HUF&gad_source=1&gclid=CjwKCAiAvJarBhA1EiwAGgZl0MKqNtUhGbr6xkDznpq6UyzONewxI-kYeS19-W_90eaR_4KJHvMhAhoCFdEQAvD_BwE (2023. 11. 22.)

[22] LED szalag: https://nz.pcpartpicker.com/product/ZBLdnQ/placeholder (2023. 11. 22.)

[23] Univerzális relé modul: https://www.hestore.hu/prod_10035513.html?lang=hu (2023. 11. 22.)

[24] ADS1115 16-Bit ADC - 4 Channel with Programmable Gain Amplifier - STEMMA QT: https://www.adafruit.com/product/1085 (2023. 11. 22.)

[25] Raspberry Pi Zero W: https://www.rpibolt.hu/Raspberry-Pi-Zero-W (2023. 11. 24.)

[26] Raspberry Pi 4 Model B: https://www.rpibolt.hu/raspberry-pi-4-model-b-4gb (2023. 11. 24.)

[27] Linux Mint Magyar Közösség: Megjelent a Debian 12 „Bookworm”, https://linuxmint.hu/hir/2023/06/megjelent-a-debian-12-bookworm (2023. 11. 24.)

[28] Why Python is a Good Programming Language: https://learnpython.com/blog/python-is-good-programming-language/ (2023. 11. 24.)

[29] Python & CircuitPython: https://learn.adafruit.com/adafruit-4-channel-adc-breakouts/python-circuitpython (2023. 11. 25.)

[30] Pulse Width Modulation: https://sourceforge.net/p/raspberry-gpio-python/wiki/PWM/ (2023. 11. 25.)

[31] How to Use MQTT in Python with Paho Client: https://www.emqx.com/en/blog/how-to-use-mqtt-in-python (2023. 11. 25.)

[32] Systemd – minimális alapok: https://magyarlinux.hu/systemd-minimalis-alapok/ (2023. 11. 25.)

[33] Red Hat: Chapter 10. Managing Services with systemd, https://access.redhat.com/documentation/en-us/red_hat_enterprise_linux/7/html/system_administrators_guide/chap-managing_services_with_systemd (2023. 11. 25.)

[34] I2C kommunikáció: https://electro.blog.hu/2020/04/26/i2c_kommunikacio (2023. 11. 26.)

[35] Medium: MQTT, https://medium.com/@ghostlulzhacks/message-queuing-telemetry-transport-mqtt-hacking-cfa932a025f (2023. 11. 20.)

[36] HiveMQ: MQTT, https://www.hivemq.com/mqtt/ (2023. 11. 20.)

[37] Androidalapú szoftverfejlesztés tárgy, 2022/23 tavaszi félév, 6. előadás diasor, Object Relation Mapping

[38] Androidalapú szoftverfejlesztés tárgy, 2022/23 tavaszi félév, 5. előadás diasor, MVVM Architektúra

[39] Android MVVM, https://medium.com/@sharma.dev2506/android-mvvm-must-haves-aa1ad9b7e223 (2023. 11. 26.)

[40] hannesa2 – Paho MQTT: https://github.com/hannesa2/paho.mqtt.android (2023. 11. 02.)

[41] Androidalapú szoftverfejlesztés tárgy, 2022/23 tavaszi félév, 4. előadás diasor, Navigáció Compose esetén

[42] Androidalapú szoftverfejlesztés tárgy, 2022/23 tavaszi félév, 4. előadás diasor, Jetpack Compose

Függelék

A megírt kódok tömörített fájlban feltöltésre kerültek mellékletként.

GreenHouse Android alkalmazás: NagyBorbala_melleklet\Android_app

Raspberry-ken futó python kódok: NagyBorbala_melleklet\Raspberry_kodok