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

Szentgyörgyi Ábel

ELEKTRONIKUS ORR FELHASZNÁLÁSA PRECÍZIÓS MEZŐGAZDASÁGBAN

KONZULENS

Dr. Szabó Sándor

BUDAPEST, 2023

Tartalomjegyzék

HALLGATÓI NYILATKOZAT

Alulírott **Szentgyörgyi Ábel**, 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, 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, 2023.12.07.

Szentgyörgyi Ábel

Összefoglaló

Az állattenyésztés egy nagy részét képzi a baromfitenyésztés. Az állatok jóléte kulcsfontosságú tényező az állatok szempontjából is és a gazda szempontjából is. Ha rossz körülmények között fejlődnek, kevesebb tojást tojnak, lassabb ütemben növekednek, de akár végzetes kimenetelekhez is vezethetnek. Mivel a baromfikat elsősorban zárt tartásban tartják, az egyik legfontosabb környezeti tényező a levegőminőség. Fontos, hogy az állatok állandóan jó minőségű légtérben legyenek, mivel több kutatás is kimutatta, hogy a levegő különböző paraméterei hatással lehetnek az állatokra. Egyik ilyen paraméter az ammóniakoncentráció, melyet folyamatosan monitorozni kell. A probléma az ammóniaszenzorok költsége. Sajnos egyelőre csak elég magas áron lehet beszerezni komplett ammónia érzékelőket, ami így az alacsonyabb költségvetésű gazdaságoknak nehézséget jelent.

Szerencsére több kutatás is arra mutat, hogy van lehetőség olcsóbb szenzorokból álló ammónia érzékelő rendszer építésére fém-oxid szenzorok bevonásával. Az ötlet izgalmas része az, hogy mindezt úgy lehet kivitelezni, hogy közben konkrét ammónia szenzor nincs a rendszerben. A célom az, hogy egy saját kutatás során én is közelebb kerülhessek ezen megoldáshoz, illetve beleláthassak az ehhez potenciálisan szükséges technológiák működésébe.

A dolgozat írása során sikerült összeállítanom egy olyan rendszert, amivel megvizsgálhattam a szenzorok tulajdonságait, mindezt egy olyan környezetben felépítve, ami akár a kész rendszer alapját is képezheti a továbbiakban.

Mint minden elektronikai rendszer esetén, itt is szembesültem néhány problémával, nehézséggel, amiket a tőlem telhető módon igyekeztem javítani, viszont előzetes konklúzióként elmondhatom, hogy bár az alapgondolatot az ammóniaszenzor kiváltására a kutatásom során én is működőnek véltem, precízebb műszerekre van szükség a teljes siker érdekében.

Abstract

Animal husbandry is a significant part of poultry farming. Animal welfare is a crucial factor for both the animals and the farmer. If animals develop in poor conditions, they lay fewer eggs, grow more slowly, and can even lead to fatal outcomes. Since poultry are primarily kept in enclosed spaces, one of the most important environmental factors is air quality. It's essential for the animals to be in a constantly high-quality airspace because several studies have shown that various parameters of the air affect the animals. One of these parameters is ammonia concentration, which needs to be continuously monitored. The problem lies in the cost of ammonia sensors. Unfortunately, complete ammonia sensors are currently only available at a relatively high price, which poses a challenge for low-budget farms.

Fortunately, several studies suggest that it's possible to build a cheaper ammonia sensor system using metal-oxide sensors. The exciting part of the idea is that this can be achieved without having a specific ammonia sensor in the system. My goal is to get closer to this solution through my own research and gain insight into the potentially required technologies.

During the writing of the thesis, I managed to assemble a system that allowed me to examine the properties of the sensors, all in an environment that could potentially serve as the basis for further development.

As with any electronic system, I encountered some problems and challenges in this project, which I tried to improve to the best of my ability. However, as a preliminary conclusion, I can say that while I considered the concept of replacing the ammonia sensor to be feasible during my research, more precise instruments are needed for complete success.

1 Bevezetés

A bevezetésen belül szeretném bemutatni a precíziós mezőgazdaság fejlődését, dolgozatom indokoltságát és motivációit, illetve azokat a háttérinformációkat, amik a téma megértéséhez szükségesek.

1.1 Mezőgazdaság fejlődése

Az ipari forradalom Ipar 4.0-ként [1] elnevezett 4. állomása a különböző elektronikai és informatikai technológiák gyors fejlődésével bontakozhatott ki igazán. Az új vívmányok megjelenésével lehetőség nyílt a korábban csak rengeteg ember és idő igénybevételével megoldható folyamatok automatizálásához, korszerűsítéséhez. A régi módszerek nagy problémája az volt, hogy az emberi tényezővel sok hiba keletkezett, egyszerűen nem tudott olyan precíz lenni az összes folyamat, mint a gépek segítségével. Humán munkaerővel pedig komplex, magas integráltságú rendszerek átlátása pedig egyenesen lehetetlen volt. A mezőgazdaságot tekintve egy másik probléma az, hogy manapság egyre kevesebb fiatal kezd bele a gazdálkodásba. Ez azzal jár, hogy a növénytermesztéshez és állattenyésztéshez szükséges munkaerő lecsökken, amit valahogy meg kell oldani. Ezt a csökkenést a KSH adatai alapján igazolni is lehet. [2]

A gyors fejlődés hatással volt a társadalomra, gazdaságra, de legfőképpen az iparra. A mezőgazdaság egy jó példa ezen fejlődés ütemének megfigyelésére, mivel még ma is sok újdonság érkezik és sok részén van lehetőség fejlesztésre. Külön érdemes megemlíteni, hogy a mezőgazdaság fejlődése nem teljesen homogén. Ezt úgy értem, hogy a nagyobb újításokat, automatizálást elsősorban a nagyobb gazdaságok veszik fel, a kisebb, vidéki gazdák gyakran nem lépnek túl a hagyományos módszereken és eszközökön, ami nyilvánvalóan financiális tényezőkön is múlik, ugyanakkor a legtöbb esetben akár kisebb költségű technológiai befektetésekkel is nagy nyereséget lehet elkönyvelni. A nagyobb gazdálkodások befektetései azonban lehetőséget nyújtanak az újdonságok fejlesztésére, így elérkezhettünk a precíziós mezőgazdaság megjelenéséhez is.

1.2 Precíziós mezőgazdaság

A PLF-en (Precision Livestock Farming – Precíziós mezőgazdaság) belül növénytermesztési és állattenyésztési precíziós megoldásokról beszélhetünk, amik azt jelentik, hogy valamilyen precíziós eszköz segítségével az állatok és növények tartásához kellő munkálatokat és környezeti tulajdonságokat tudjuk a lehető legoptimálisabban befolyásolni. Az ehhez kellő eszköz az informatika, mely segítségével adatokat tudunk gyűjteni és ezen adatok feldolgozásával és elemzésével optimalizálni tudjuk a rendszereinket.

Az optimalizáláshoz gyakran közvetlen beavatkozás tartozik, például amikor a hőmérsékletet egy adott intervallumon belül kell tartani és egy automatizált hőmérsékletszabályzó rendszer azonnal elvégzi a kellő beavatkozásokat, például a fűtés fentebb vagy lentebb állításával. Az adott feladat néha közvetett beavatkozással oldható csak meg, mely emberi tevékenységeket igényel. Például amikor egy kamerarendszer észleli, hogy egy etetőhöz nem járnak állatok, mivel az eldugult, a gazdának szükséges odamennie és javítani a hibán, mivel ezen feladat megoldása csak nehezen automatizálható.

A PLF egy folyamatosan újabbnál újabb megoldásokat igénylő ágazat, mely nagyszerű módon ötvözi a mezőgazdaság több évszázados tapasztalatát a mérnöki világ újdonságaival. Ez a munkában szoros együttműködést igényel a gazda és a mérnök között, így úgy gondolom, ez a terület egy nagyon izgalmas kitekintést nyújthat a mérnöknek, hogy egy igazán életközeli területen dolgozhasson, néha a számítógépes asztalt elhagyva. A PLF kulcsa az adatelemzés, melynek hála az állatok jóléte javítható, a gazdaság kiadásai csökkenthetők, a gazda kényelmesebben tud dolgozni és elkerülhetők a felesleges munkálatok, kiadások.

Egyetemi tanulmányaim alatt lehetőségem nyílt a témalabor és önálló labor tárgyak során több precíziós mezőgazdasághoz köthető kutatást, fejlesztést is megismerni, amik mind azt mutatták, hogy van jövője az ilyesfajta törekvéseknek. A mesterséges intelligencia bevetésével pedig számos olyan munkafolyamatot és problémát oldhatunk meg a gazdálkodásoknak, amikhez eddig csak emberi beavatkozásos módszerek voltak.

1.3 Elektronikos orr

Az elektronikus orr [3] egy már ma is széleskörűen alkalmazott eszköz, mely alapvetően az emberi szaglás helyettesítésére, szagok, gázok érzékeléséhez használatos. Rendkívül sok problémára megoldást tud nyújtani, például a gyenge szaglású embereket figyelmeztetheti földgáz jelenlétére, tüdőrák, vagy más egészségügyi állapotok kimutatására is alkalmas lehet, illetve minőség-ellenőrzés is végezhető vele a húsfeldolgozáskor.

Az eszköz 3 fő részből áll:

- mintaszállító rendszer
- érzékelő rendszer
- számítási rendszer

A minaszállító rendszer felelős azért, hogy az adott mintát az érzékelőkhöz juttassa. Ez abban az esetben fontos, ha a minta nem tud közvetlenül az érzékelőkhöz jutni, esetleg nagyobb koncentrációt kell előre készíteni és azt érzékelik a szenzorok. Ez a rész a tervezett rendszeremben nem szükséges, mivel a mintát a légtérből kapja a rendszer, így azt direkt módon monitorozza.

Az érzékelő rendszer több különböző érzékenységű szenzor együttese. A legelterjedtebb szenzortípusok: fém-oxid félvezető (MOSFET) eszközök, vezető polimerek - szerves polimerek, polimer kompozitok, kvarckristály-mikromérleg (QCM), felületi akusztikus hullám (a mikroelektromechanikus rendszerek (MEMS) egy osztálya), tömegspektrométerek. Én ezek közöl a fém-oxid félvezetőket alkalmazom.

A számítási rendszer az elektronikus orr agya, ő végzi a szenzoroktól érkező jelek mintavételezését, rögzítését, feldolgozását. A feldolgozás során a szenzorok jeleit együttesen elemzi, amiket összehasonlítva tud eredményeket adni. Én a rendszeremben egy ESP32 mikrokontrollert használok.

Az elektronikus orr tehát egy nagyszerű és jelentőségteljes eszköz, ami bevethető lehet akár a precíziós mezőgazdaság különböző területein is, viszont én most elsősorban a baromfitenyésztésre koncentrálok.

1.4 Dolgozat célja

A dolgozatomban a fő cél az, hogy kimutassam, hogy a lentebb bővebben is bemutatásra kerülő fém-oxid félvezető szenzorok (későbbiekben MQ (Metal-Oxid) szenzor) képesek-e az ammónia detektálására, illetve koncentrációjának mérésére. Ehhez összesen 9 különböző MQ szenzort használok, kiegészítve egy szintén MQ típusú ammónia szenzorral, illetve egy hőmérséklet és páratartalom mérésére alkalmas szenzorral.

További célnak tűztem ki, hogy az összeállított szenzorrendszer egy olyan platformon működjön és képes legyen egy olyan kommunikációra, ami a valós felhasználás során is működő lehet. Elsősorban vezeték nélküli kommunikációt szerettem volna megvalósítani, mivel egy nagy telepen sok nehézséget és extra költséget jelentene a vezetékes megoldás, így ez tűnt optimálisabbnak, továbbá a rendszer bővíthetősége is könnyebb lesz ezáltal.

Mindezeket beleértve a személyes célom az volt, hogy a rendszer felépítéséhez szükséges technológiákat mélyebben is megismerhessem, jobban belelássak a mikrokontrollerek és szenzorok világába és a kommunikációs protokollok működési elveit is elsajátíthassam.

Mivel a későbbiekben taglalt irodalomkutatás során talált eredmények megfelelően sikeresnek tűntek, bele mertem vágni egy hasonló kutatás elkészítésébe, azonban mivel egyik kutatás sem foglalkozik valós körülmények közötti teszteléssel, komplex adatfeldolgozással és nekem sincs lehetőségem valós körülmények közötti tesztelésre, a rendszer valós hatékonyságának vizsgálatát nem tudtam biztosan ellenőrizni, így a konklúzióként várt eredmények kifejezetten a szenzorok ammóniára való érzékenységének vizsgálatára irányulnak.

2 Irodalomkutatás

A továbbiakban a dolgozathoz szükséges kutatásokat, cikkeket fogom áttekinteni, mely után a feladat megoldásához kellő paraméterek könnyebben eldönthetők lesznek.

2.1 Állatjólét

Az állatjólét szerencsére napjainkban már nagyon fontos és komolyan is veszik mind a gazdák, mind pedig a hatóságok. Több ezzel kapcsolatos rendeletet is kiadtak világszerte, melyek arra hivatottak, hogy az állatok megfelelő környezetben növekedhessenek. A dolgozatomban kulcsfontosságú beállítani, hogy a szenzorok milyen határokon belül mérjenek és mennyire legyenek pontosak, ezért több ehhez kapcsolódó rendeletet is áttekintettem, amik az ammónia koncentrációt vizsgálják.

A szárnyasokat általában zárt térben tartják, így könnyen mérhetjük és szabályozhatjuk a környezeti viszonyokat. Kuney (1998) [4] felfedezte, hogy minimális hőmérsékleti ingadozások is szignifikánsan befolyásolják a baromfik takarmányfogyasztását, ami befolyásolja a termelést. A nem megfelelő környezeti tényezők ezen túl hozzájárulhatnak az állatok megbetegedéséhez, stresszt okozhatnak.

A baromfik jólléte érdekében az agrárminisztérium által kiadott „*11/2019. (IV. 1.) AM rendelet a baromfi ágazatban igénybe vehető állatjóléti támogatások feltételeiről*” című rendeletének 9. § (3) b) pontjának értelmében csak úgy lehet uniós támogatáshoz jutni, ha a gazda teljesíti többek között az ammóniára vonatkozó feltételt is: *(3) „Az igénylőnek az istálló gázkoncentrációjának javítása érdekében biztosítani kell az istállóban az ammónia 14 ppm-nél alacsonyabb mértékét.”* [5] **(**ppm = part per million**).** A továbbiakban ki is emelik, hogy ezen feltétel teljesülését szolgáltató állatorvos helyszíni ellenőrzése során igazolni is kell.

Egy 2013-mas, *Incorporating Smart Sensing Technologies into the Poultry Industry* [4] című tanulmányban sok fontos paramétert összegyűjtöttek a baromfik élettani jellemzőikről a környezetük függvényében. A kutatás a legnagyobb problémának a megfelelő szellőztetés hiányát említi. Az energiagazdálkodás miatt fontos odafigyelni a megfelelő szigetelésre, hogy a hőmérséklet a lehető legkevesebb energiával tudjon konstans szinten maradni, viszont szellőztetés nélkül a NH_3, CO_2, magas páratartalom és por nem, vagy csak nehezen tud alacsonyabb szintre kerülni.

Az ammónia elsősorban az állatok trágyájából keletkezik. Ha nő a páratartalom, nő a trágyában lévő mikroorganizmusok száma, ami így növekvő ammóniatartalomhoz vezet. Ehhez, ha magasabb hőmérséklet is társul, az tovább tudja gyorsítani a folyamatot. A tanulmányban leírt megfigyelés alapján 10 °C-ról 25 °C-ra való növekedés 10 ppm-ről 25-30 ppm-re való emelkedést mutatott. Czarick and Fairchild (2012) kutatása alapján a páratartalom 60%-on való tartása esetén az ammónia elfogadható tartományban maradt, 70% fölött viszont 50 ppm fölé került.

Ahogy a tanulmányban olvashatjuk, 25 ppm ammónia koncentráció esetén észlelhetően lassul a brojlercsirkék (húsfeldogozásért tenyészetett csirkék) növekedése, 50-75 ppm-es ammónia szintnél pedig már kimutathatóan nagyobb a csirkék halálozása, így több ország szabályozásában ezek a szintek jelennek meg. Írországban a szabályozás értelmében nem szabad túllépni az ammónia tartalomnak 8 órán keresztül 20 ppm szintet, 10 perces időtartamot tekintve pedig 35 ppm-et *(European Communities, 2007: Bord Bia, 2008).* A többi európai ország hasonló értékekkel állították be a határokat, Németország 20 ppm, Svédország és Egyesült Királyság 25 ppm 8 órás ciklusra. Svédországban ezen felül 50 ppm-es határt kell tartani 5 perces ablakban. Érdekességként a sertések esetét is tekintve egy sertéstartásra vonatkozó hasonló rendeletben 9,5 ppm szint a maximáls. [6]

A fenti számokat tekintve eldönthető, hogy a dolgozatomban tervezett elektronikus orrnak legalább az 50 ppm-es ammónia szintet ki kell tudnia jelezni, mivel ezen szint fölött tapasztalható jelentősebb élettani romlás a baromfiknál, viszont, ha jogszabályi limitek mérésére is alkalmasnak szeretnénk tekinteni, akkor legalább a 20 ppm-es gázkoncentrációt is eredményesen kell jeleznie, továbbá célszerű hőmérsékletet és páratartalmat is mérni.

2.2 Gazdasági tényezők

A mai árfolyamok és energiaárak mellett nagyon fontos odafigyelni, hogy mire és mennyit költünk. Ezek közül a mezőgazdaságban és azon is belül a baromfitenyésztésben számos olyan paraméter van, amin sokat spórolhatunk, vagy bukhatunk, ha nincs megfelelő kontroll alatt.

2.2.1 Mezőgazdaság

Ahogy fentebb említettem, természetesen más-más kasszából tudnak gazdálkodni a más méretű tenyészetek, így valószínűleg nagyobb gazdálkodások könnyebben vesznek új eszközöket, fejlesztik a rendszereiket precíziós mezőgazdasági eszközökkel. Egyik célom így az, hogy minden gazdának elérhető áron fejlesszek egy olyan terméket, ami akár rövid távon is kifizetődő hatással van az termelésre és az állatokra is. Mivel a legtöbb pályázatban az ammóniakoncentrációra vonatkozó feltételek is vannak, így a gazdák ennek segítségével akár könnyebben nyerhetnek el pályázati forrásokat.

Az ammóniaszint mérésére természetesen léteznek szenzorok, viszont ezek igen költséges rendszereket jelentenek. Például egy SMART-R-NH3 4-20MA típusú ammónia érzékelő 253 365 Ft [7], amihez, ha a telepítés költségeit is hozzávesszük, igen magas költségek gyűlhetnek fel. Az általam is használt MQ szenzort alapul véve egy kicsit jobb a helyzet, az MQ137 elnevezésű ammónia mérésére alkalmas szenzor 10-20 ezer forint között kapható, viszont összevetve a többi fém-oxid szenzorral, amik akár 600-700 forint között kaphatóak, így is jóval nagyobb az ára.

2.2.2 Energiagazdálkodás

Mindannyian tisztában vagyunk a mai fűtési árak magasságával, ami a baromfik zárt tartása miatt az állattenyésztésre is nagy hatással van. A már említett kutatás [4] egy 2011-es mérés eredményét közölte, amiben a teljes energiafelhasználás 84%-a fűtésre, teljes elektromos áram felhasználás 45%-a pedig a szellőztetésre fordítódik. Ebből könnyen levonhatjuk, hogy a fűtésre és szellőztetésre különös figyelmet kell szentelni. Mivel a levegőminőség szintén kritikus paraméter, aminek javítását szellőztetéssel érhetjük el, még nagyobb hangsúlyt kap a teljes energiagazdálkodás kontroll alatt tartása és állandó monitorozása.

2.3 Feladat indokoltsága

A fejezetben leírtakat figyelembevéve indokolt alkotni egy olyan szenzorrendszert, ami méri a levegő ammóniakoncentrációját, olcsóbb a kapható ammóniaszenzoroknál és megfelelően pontos is. Ha sikerülne a célnak megfelelő több szenzorból álló, de olcsóbb rendszert megalkotni, sok gazdának jelenthetne könnyebbséget. További jó indok a megoldásra, hogy az MQ szenzorok felhasználási körét tovább bővítsem.

2.4 Hasonló kutatások összevetése

A dolgozatom egy másik motivációja az, hogy léteznek az enyémhez hasonló kutatások, amikből ki tudtam indulni és amik bátorítottak, hogy működőképes lehet a készülő rendszer. A módszer eredményességének vizsgálatáról több kutatás is készült, melyek közül a leginkább releváns az „*Advance in electronic nose technology developed for the detection and discrimination of ethanol, ammonia, and hydrogen sulfide gases”* [8] című. A kutatás 3 gázt is megvizsgált az olcsóbb MQ szenzorok segítségével, mely gázok az ammónia, etanol, hidrogén-szulfid.

A szenzorokat egy erre a célra létrehozott vizsgáló csőbe helyezték, amelybe az adott gázból meghatározott koncentrációban engedtek, és figyelték a szenzorok válaszát.

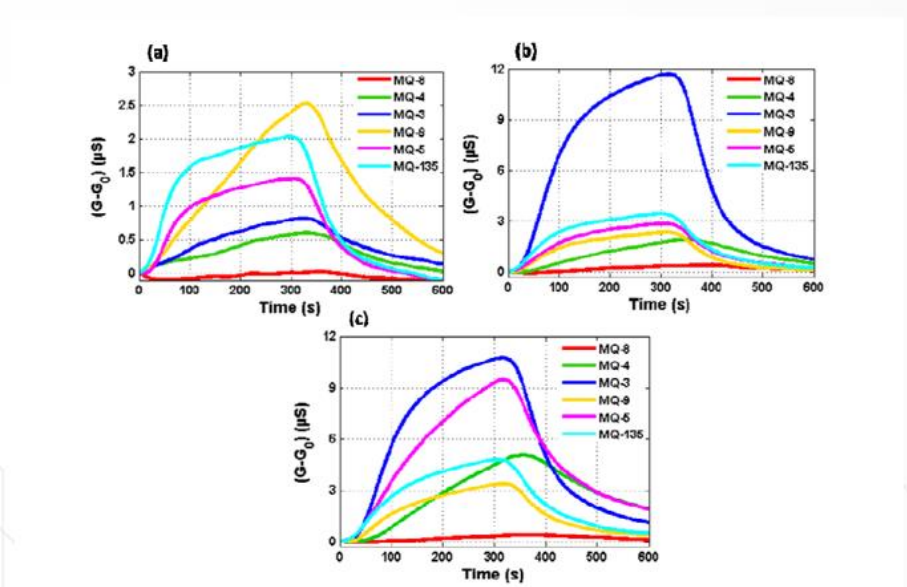


Fig. 2. Electrical conductance changes in presence of: (a) 7.3 ppm of NH_3, (b) 3 ppm of C_2H_5OH, and (c) 6 ppm of H_2S.

1. ábra - Különböző gázokra adott szenzorválaszok [8]

Az 1. ábrán jól látható, ahogy a különböző szenzorok milyen görbét mutatnak az adott gázzal való érintkezésükkor az idő függvényében. Érdemes megfigyelni, hogy az ammónia jelenléte esetén az MQ9, MQ135 és MQ5 szenzorok mutatnak nagyobb kilengést. Ez a tény teszi végső soron lehetővé az ammónia szenzor MQ szenzorokkal való helyettesítését.

Másik számomra releváns kutatást a ***Spirocco KFT***.-nek köszönhetem, mely vállalkozás egyik munkatársa, ***Molnár Marcell*** foglalkozott egy hasonló konstrukció megalkotásában és tapasztalatával segítette az én haladásomat is. A kutatásukban felhasznált szenzorok ugyanolyan típusúak voltak, mint a fentebb említett kutatásban szereplők, azonban ők kihagyták az MQ9-es szenzort, mely az előző kutatás eredményei alapján a legnagyobb érzékenységet mutatta az ammóniára. Ők nem tesztelték külön ammóniára a szenzorokat, viszont a 2. ábrán egy valós környezetben való mérésük eredménye látható, ahol hasonló kimozdulásokat mutatnak bizonyos szenzorok.

2. ábra - MQ szenzorok jelei valós környezetben [9]

A saját kutatásom során biztosra szeretnék menni, így az összes elérhető MQ szenzor hatékonyságát tesztelem, így nagyobb mérési adathalmazt kapva, hogy a lehető legjobban tudjam közelíteni az ammóniaszenzor eredményét.

3 Feladat megoldása

A továbbiakban taglalom a feladat megoldásához használt eszközöket, technológiákat és ismertetem azokat a megfontolásokat, amelyekkel ezeket kiválasztottam.

3.1 Pontos célkitűzés

Az ammóniaszenzor magas költsége és MQ szenzorok felhasználási lehetőségeinek bővítése céljából a fő célom az, hogy megállapítsam és feldolgozzam a kapható fém-oxid félvezető szenzorok ammóniára adott válaszát. Ezt egy több szenzorból álló rendszerrel teszem meg, ami tartalmaz 9 db olcsóbb (500-600 forintos) különböző gázokra érzékeny MQ szenzort, illetve egy ammóniára érzékeny MQ szenzort is. A mérési eredmények kiértékelése az ammóniaszenzor és a többi szenzor válaszának összehasonlítása során valósul meg. Az összehasonlítás alapját a szenzorok válaszának MQ137 - ammónia szenzorhoz mért korrelációs együtthatója alapján fogom megállapítani. A korrelációs együttható megmutatja, hogy mekkora és milyen irányú a mérés során keletkező két összehasonlított adathalmaz közötti lineáris kapcsolat, tehát, hogy mennyire mozognak együtt. Pontos korrelációkhoz nagyobb adathalmaz szükséges, így mérés során fontos elegendő időt hagyni a szenzorok mozgására. [10] A mérési eredmények után a szenzorokra felállítok egy karakterisztikát, mely megmutatja, hogy milyen válasz várható bizonyos ammóniakoncentrációknál.

A rendszer megalkotása során igyekszek figyelni arra is, hogy a valós felhasználásnak megfelelő technológiákat használjon, illetve az elemzésem kiterjed a módszer pontosságának vizsgálatára is.

3.2 Kezdeti lépések

A módszer átgondolásával és a rendszer összeállításával önálló labor keretein belül kezdtem el foglalkozni, az irodalomkutatás nagy részét és eszközök rendelését akkor sikerült elvégezni.

A rendelt hardverekből még akkor megépítettem egy olyan tesztelési rendszert, amivel a szakdolgozat alatt is dolgozni tudtam pár kisebb módosítás után, így több idő maradt a szoftver fejlesztésére.

3.3 Használt eszközök

Ahogy a bevezetőben szerepelt, az elektronikus orr 3 fő egységből áll, a mintaszállító, érzékelő és számítási rendszerből, amikből a feladathoz elsősorban az érzékelő és számítási rendszer szükséges, mivel a mintát közvetlenül a légtérből fogjuk olvasni.

3.3.1 Számítási rendszer

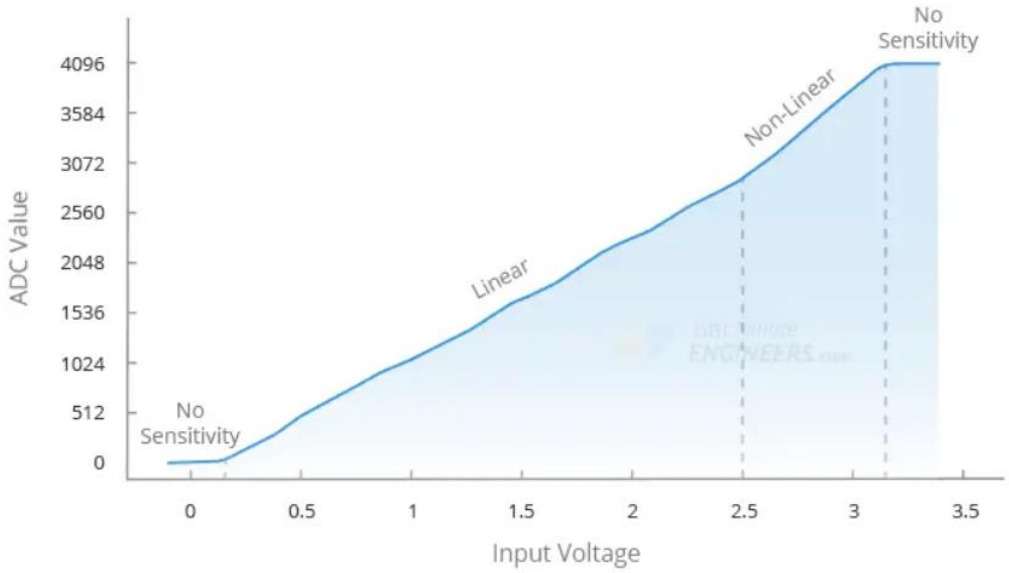
A számítási rendszeremet egy mikrokontroller fogja képviselni. A mikrokontrollerek [11] manapság rendkívül elterjedt elektronikai eszközök, melyeket igen széleskörben alkalmazzák. Népszerűségének legfőbb oka, hogy könnyen programozhatóak, kis méretűek és nagyon kedvező áron hozzájuthatunk. Ezek közül a legfontosabb a mérete és az ára, mivel szeretném, hogy a lehetőségekhez mérten elég kompakt rendszert kaphassak, illetve, mivel a rendszerrel a költséghatékonyság is cél, az ára is fontos tényező.

Maga a mikrokontroller gyakorlatilag egy miniatűr számítógép, mely vezérlési feladatok ellátásához készült. Hogy kedvezzenek a hobbielektronikát kedvelőknek gyakran egy lapkára integráltan árulják, amin könnyebben hozzáférhetőek a ki- és bemenetei, illetve sokszor különböző kiegészítő modulokat, áramköri egységeket szintén integrálnak hozzá a könnyebb használat érdekében.

Manapság elég sok mikrokontrollert kaphatunk a piacon, így ezek közül ki kellett választanom a legjobbat. Ez egészen könnyen ment, mivel az egyik félévben hallgatott tárgyam keretein belül megismerkedhettem az ESP32 mikrokontrollerrel és egy kis kutatómunka után úgy döntöttem ez megfelel minden elvárásnak.

A döntés fő okai a versenytársaihoz képest (pl. Arduino Uno, Nano, STM32), hogy 12 bites analóg-digitális átalakítója (ADC) miatt nagy pontosságú adatgyűjtést tesz lehetővé (Arduino legtöbb modellje csak 10 bites ADC-jű). Ez azt jelenti, hogy a teljes mérési tartományt 2^{12} db „szeletre osztva” tudja monitorozni. Az ESP32 ADC-jének felső mérési határa 3.3V (ez a FS, azaz Full Scale – mérési tartomány), 1 bit változásnak tehát 3.3V / 2^{12} = 0.806mV változás felel meg. (Ezt mérési felbontásnak is nevezzük, a jele Q, mértékegysége V/bit → ebben az esetben $Q = 0.806\ \mathrm{mV/bit}$)

Ennek az ADC-nek az egyik fontos jellemzője, hogy nem teljesen lineáris, ahogy a 3. ábrán is láthatjuk [12]:

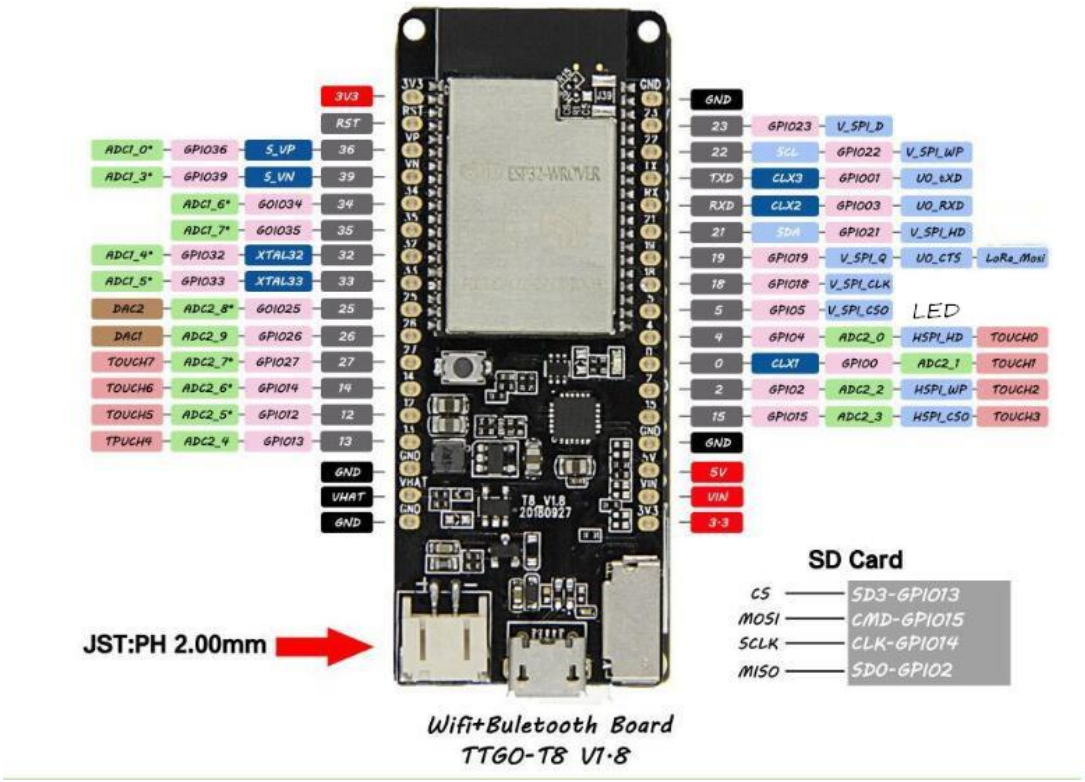


3. ábra - ESP32 ADC karakterisztikája [12]

Az ábrán jól látható, hogy alacsony feszültségeken (0-0.13V között) gyakorlatilag nem érzékel az ADC, 0 bit értékeket ad, 2.5V és 3.2V között nem lineáris, 3.2V-3.3V között pedig szintén nem érzékel, végig 4095 értékre áll, ami további mérési hibát eredményezhet. Ezeket összevetve tisztában kell lenni vele, hogy bizonyos mérési hiba biztosan keletkezni fog a tesztelés során, különösen akkor, ha 0.13V alatti feszültségeket mérnék.

Az általam választott TTGO T8 V1.8 típusú ESP32 előnye, hogy beépített wifi és bluetooth modullal is rendelkezik, illetve van rajta egy SD kártyafoglalat, amit az adatok lokális mentése során ki is fogok használni a tesztelés során. A használt SD kártya tárhelye 16GB, így biztosan nem fog gondot okozni a generálandó adatmennyiség nagysága. A mérések során majd másodpercenként veszek mintát, egy mérés eredménye pedig ~250Byte-ot foglal, így 16GB / 250B = 64 millió mérési pont, amit nagyjából 2 év folyamatos méréssel lehetne elhasználni.

A használt ESP32 lábkiosztása az alábbi 4. ábrán látható:



4. ábra - ESP32 TTGO T8 V1.8 lábkiosztása [13]

Ez az ábra azért is különösen fontos, mivel több megkötés is található rajta, amit figyelembe kell venni tervezés során. Az ESP wifi és SD kártya modulját is használom rajta. A wifi használata mellett a zöld keretben lévő ADC2 nevű bemenetek nem használhatóak, csak az ADC1-esek, az SD kártya pedig a GPIO-2, -13, -14, -15-ös lábakat foglalja, így ezek másra való használata sem lehetséges. Az I2C kommunikációhoz a 21 és 22-es lábak használhatóak, illetve a 34, 35, 36, 39-es lábak csak kimenetként funkcionálnak, így például a multiplexer csatornaválasztásához szükséges kimeneti jelet ezekkel nem lehet létrehozni. Érdemes még kiemelni, hogy ahhoz, hogy a mérések pontosak legyenek, fontos az ESP GND-jét is azonos földpontra helyezni, mint a szenzoroké, illetve az ESP a VIN lábára feszültséget kötve üzemkész állapotba kerül a mikrokontroller. **[13]**

Fontosnak tartom megemlíteni itt, hogy a mért adathalmazt összetettebb módszerekkel is feldolgozom a laptopom segítségével, illetve a mérések vezérlését is laptoppal végzem, így az is része a számítási rendszernek.

3.3.2 Érzékelő rendszer

Az elektronikus orrhoz érzékeléshez alapvetően szükséges érzékelő eszközök a már említett MQ szenzorok. Ezek fém-oxid félvezető (MOS) alapú érzékelők, ami azt jelenti, hogy a működésük során egy fém-oxid bevonat vezetőképességének változása fogja mutatni a gáz koncentrációját.

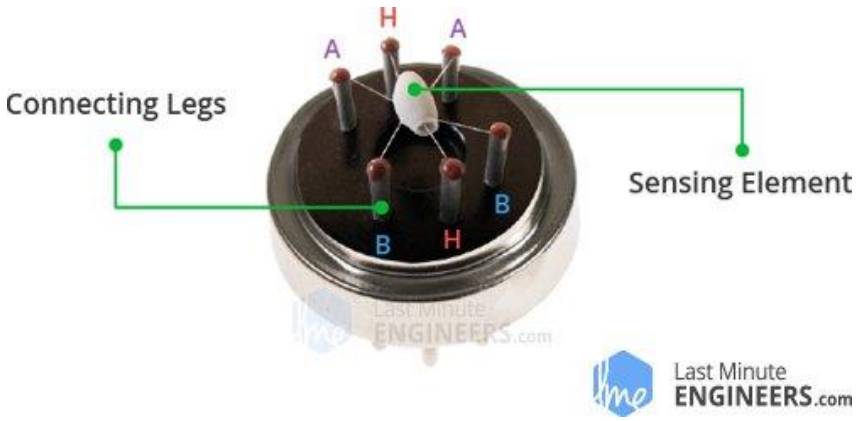
Az MQ szenzorokból egészen sok fellelhető, viszont én most kifejezetten az olcsóbbakat szeretném felhasználni, amik az MQ2, MQ3, MQ4, MQ5, MQ6, MQ7, MQ8, MQ9, MQ135 nevű szenzorok. Minden szenzor eltérő gázokra érzékenyebb, illetve több átfedésük is van. Például metánra az MQ2 és MQ4 is érzékeny, LPG-re az MQ5 és MQ6. Felépítésük többnyire azonos, egyedüli különbség az érzékelésért felelős rétegben van, illetve néhány szenzor burkolata nem fém, hanem műanyag.

A teszteléshez, hogy tudjam, pontosan milyen koncentrációjú ammóniát mérek, használtam egy MQ137 nevű, ammóniára érzékeny szenzort is. Érdemes itt is megemlíteni, hogy az MQ137 ára ~8000 Ft volt, a többi fent felsorolt szenzor pedig egyenként ~600 forintba kerültek (2023. március). Ezekből mindet a könnyebb használat érdekében úgy rendeltem, hogy egy-egy egyszerű áramkörre integrálva legyenek, amit később részletesen is bemutatok. Sajnos a szenzorok megbízhatósága és paraméteri kétségesek, mivel nem mellékeltek adatlapot a szenzoroknak, így amely áramköri paraméterekre szükségem volt, azt számolással vagy mérésekkel tudtam kihozni, ahol pedig adatlapi értékre volt szükség, ott megkerestem egy interneten fellelhető ugyanolyan szenzor adatlapját.

Az 5. ábrán látható maga a szenzor. A tetején lévő védőháló azért fontos, hogy a szenzornak biztosíthassuk, hogy például por ne, hanem csak a gázrészecskék jutnak el hozzá, illetve védelmi szerepe is van, hogy ne robbanjon be gyúlékony gáz vizsgálata esetén a szenzor.

5. ábra - MQ2 szenzor a védőhálóval [14]

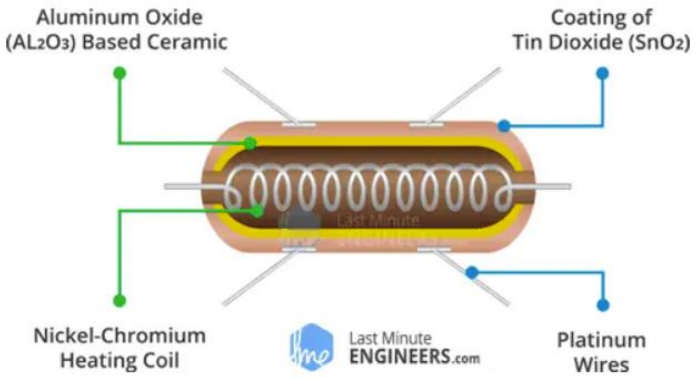
A 6. ábrán látható H pontok közötti vezeték fogja az üzemi hőmérsékletet (150-300°C [15]) biztosítani az fém-oxid bevonatú vezetőnek, az A és B pontok között pedig ezen vezetőn eső feszültséget mérhetjük, mely a vezető ellenállásának változásával együtt változik.



6. ábra - MQ szenzor a védőháló nélkül [14]

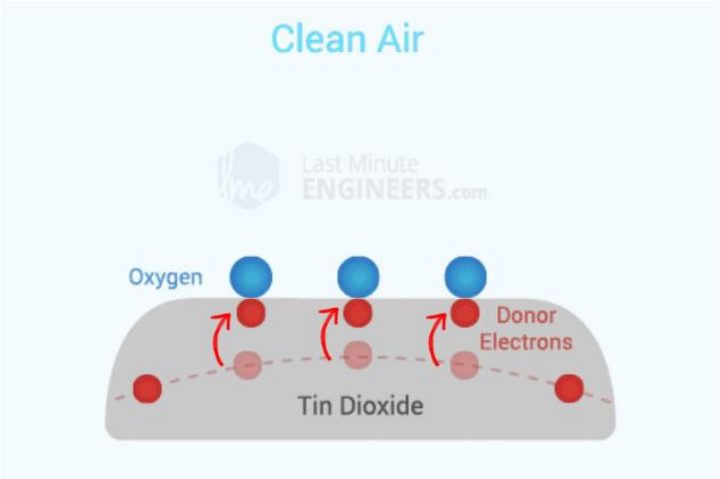
A magas üzemi hőmérsékletet egy 33 Ω-os nikkel-króm tekercs 5V feszültséggel való táplálásával érjük el, ami így azt jelenti, hogy egy szenzor működéséhez szükséges áram ~150mA, így 10 szenzorhoz nagyjából 1.5A áram kell. Ez mikrokontrolleres környezetben igen nagy, főleg, ha összevetjük az ESP32 ~35mA-es áramfelvételével.

Az érzékelő elem (MQ2 szenzor esetén) egy Ón-dioxid (SnO_2) réteg, ami egy alumínium-oxid (Al_2O_3) alapú kerámiahengeren helyezkedik el, ahogy a 7. ábrán is látható. Az alumínium-oxid alapú kerámia a fűtés hatékonysága és folyamatos üzemi hőmérséklet biztosítása miatt fontos elem. Ezeken felül az érzékelőhöz kapcsolódik egy platina vezeték, aminek segítségével a megfelelő 5V-os mérési tápfeszültséget visszük a szenzorra.



7. ábra - Érzékelő elem felépítése [14]

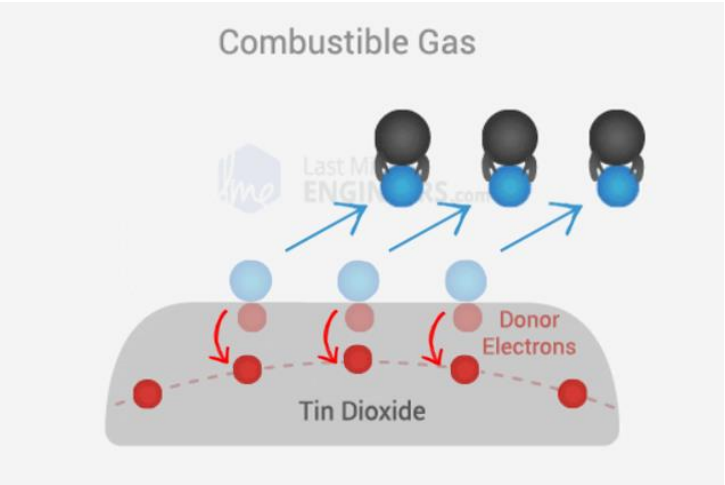
Maga az érzékelés úgy működik, hogy az ón-dioxid magas hőmérsékletre való fűtése után, ha tiszta a levegő, a felületére oxigén molekulák kötődnek, amik magukhoz vonzzák az ón-dioxid vezetési sávjában lévő elektronokat.



8. ábra - Érzékelő tiszta levegő esetén [14]

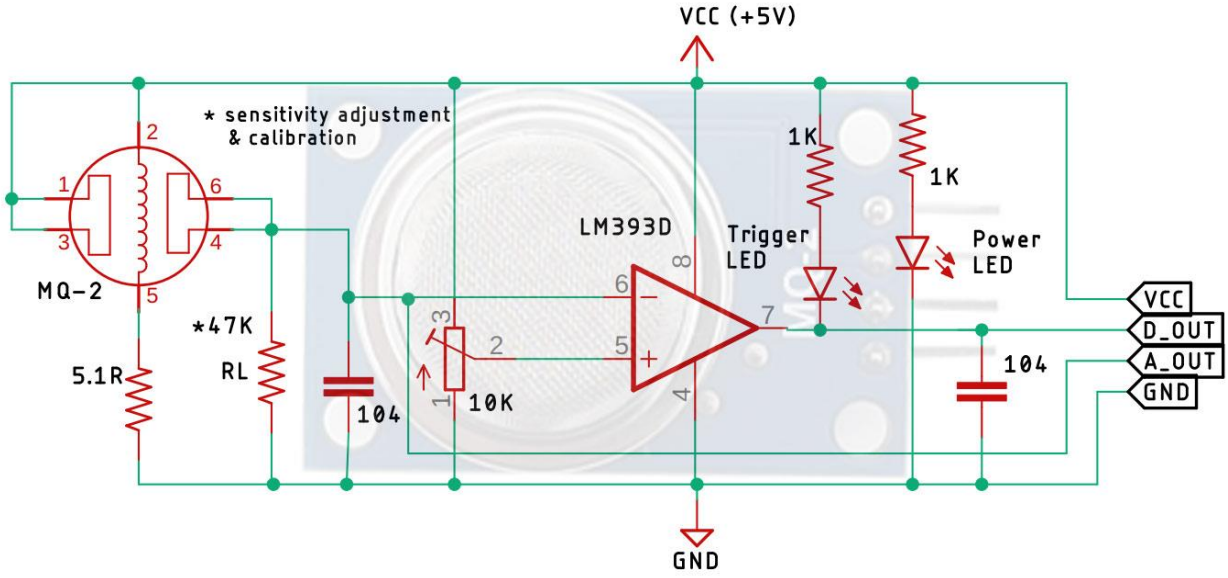
Az oxigénmolekuláknál lévő elektronok ezáltal egy kiürített tartományt hoznak létre, „lekötnek” elektronokat közvetlenül a felszín alatt. Ez azt jelenti, hogy csökken a szabad töltéshordozók, vezetésre képes szabad elektronok száma, így nő az érzékelő elem ellenállása.

Ha redukáló/gyúlékony gázzal (pl. hidrogén, propán-bután) érintkezik a szenzor, a félvezető felületén lévő oxigén molekulával kapcsolódik és elvonja azt, amivel így az eredetileg tiszta levegőn az oxigénhez kapcsolódott elektron visszakerül a vezetési sávba, nő a szabad elektronok száma, így csökken az érzékelő ellenállása. [14]



9. ábra - Érzékelő gáz jelenlétében [14]

Az MQ szenzorokat a prototípushoz áramkörre építve szereztem be, hogy könnyebb legyen tesztelni őket. Az áramkörök felépítése megegyezik, de néhány áramköri elem (pl. RL) értéke viszont eltér a különböző szenzorok áramkörén. A kapcsolási rajz a 10. ábrán látható:



10. ábra - MQ szenzor áramköri rajta [16]

Az ábra bal oldalán található maga a szenzor a 6 lábával. Látható, hogy a 2, 5-ös számmal jelzett lábakhoz csatlakozik a fűtésért felelős króm-nikkel tekercs, 1, 3 számmal jelzett lábakon pedig a méréshez szükséges VCC feszültséget kapja a szenzor, ami 5V.

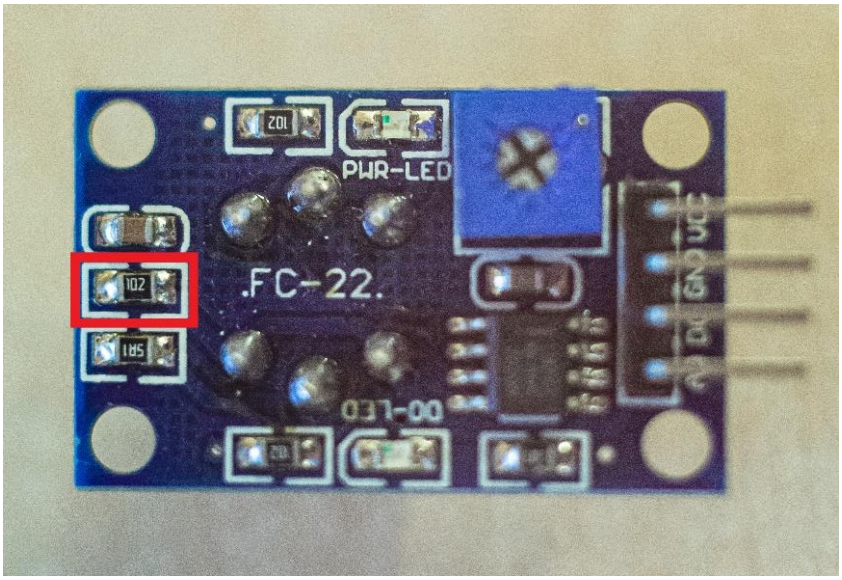
A szenzor használatakor lehetőség van az áramkör analóg kimenetét vizsgálni. Ez úgy történik, hogy az RL ellenálláson eső feszültséget mérhetjük az AOUT és GND kimenetek között. Az érzékelő ellenállása (továbbiakban RS) és az RL egy feszültségosztót képez, így, ha ismerjük az RL ellenállás értékét, a bemeneti és RL-en eső feszültségeket, ebből kiszámíthatjuk RS-t is, amiből a megfelelő karakterisztika alapján kiszámíthatjuk a gáz koncentrációját.

Az áramkör többi része egy digitális kimeneti jelet is biztosít számunkra egy komparátor és egy állítható potenciométer segítségével. A potenciométer lehetőséget nyújt a digitális kimenet döntési küszöbét állítani. A digitális kimenet magas állásban egy LED-et is meghajt, így közvetlenül is értesülhetünk az érzékelőnk állapotáról, illetve a tápfeszültség adásakor is nyit egy LED dióda, így láthatjuk is, ha tápfeszültség alá került a szenzor és üzemel.

Előzetes számítások

A szenzor karakterisztikájában az ammónia koncentráció ppm értékeihez vannak rendelve RS/R0 mért értékei, így ebből az arányszámból leolvashatjuk az ammónia ppm értékét. Itt RS a szenzor gáz jelenlétében mért ellenállása, R0 pedig a szenzor friss levegőn mért ellenállása, így RS/R0 a szenzor gáz jelenlétében és friss levegőn mért ellenállásának aránya. Ebből R0-t előzetesen meg kell határozni.

Mivel nincsenek pontos adatlapi értékeim, az áramkörön található RL értékét is meg kell határozni, amihez két módszert találtam. Mivel az áramkör úgy lett kialakítva, hogy RL-en eső feszültséget AOUT és GND kimenetek között mérni tudjuk, egy multiméterrel erre rámérve RL ellenállás értékét is megkaphatjuk. Másik módszer RL ellenállás megtalálására, hogy az áramkörről leolvassuk az SMD (Surface Mounted Device) ellenállás kódját, ami az áramkörön az alábbi pirossal keretezett helyen található:



11. ábra - MQ137 szenzor áramköre

Erről leolvasható, hogy a kódja 102. A kódban az első két digit a szignifikáns, harmadik pedig a 10 hatványán áll, a következő számítással pedig megkaphatjuk annak értékét: **10***10^2 = 1000 Ω, ami a multiméteres mérésnek is megfelel.

A fentebb írtaknak megfelelően az 5V-os Vcc tápfeszültséget a szenzor RS értéke és egy RL ellenállás osztja, az RL-en eső V_{RL} -ből pedig kiszámíthatjuk az aktuális RS értéket a következő egyenlet átrendezése után:

$$V\_RL = Vcc * RL/(RS + RL)$$

$$Vcc/V\_RL = (RS + RL)/RL$$

$$RS = RL * (Vcc/V\_RL - 1)$$

R0 értékének meghatározásához az RS-hez használt képlet alkalmazható, mivel a R0 ugyanúgy a szenzor ellenállása, csak szabad levegőn. Mivel minden szenzoron el kell végezni ezt a mérést, hogy megkaphassuk végül az adott szenzor RS/R0-NH3 ppm karakterisztikáját, a mérés gyorsítása érdekében írtam egy Arduino kódot az ESP32-re, ami minden szenzorról leolvassa a V_{RL} értékét, majd ebből kiszámítja R0-at. Az eredmények a 12. ábrán láthatóak $k\Omega$ mértékegységben:

```
MQ2 R0 = 25.17
MQ3 R0 = 26.08
MQ4 R0 = 44038.45
MQ5 R0 = 2936895.50
MQ6 R0 = 4628.21
MQ7 R0 = 518.21
MQ8 R0 = 10.66
MQ9 R0 = 3365.51
MQ135 R0 = 3366511.75
MQ137 R0 = 12.95
```

12. ábra - ESP32 által mért értékek

Mivel nagyon kis feszültségeken (kb. 0.01-0.05V között) az ESP mérései nem lettek teljesen pontosak, kézzel is lemértem a szenzorok V_{RL} értékét és kijavítottam az itt kiugróan magas értékeket, így a korrigálás után:

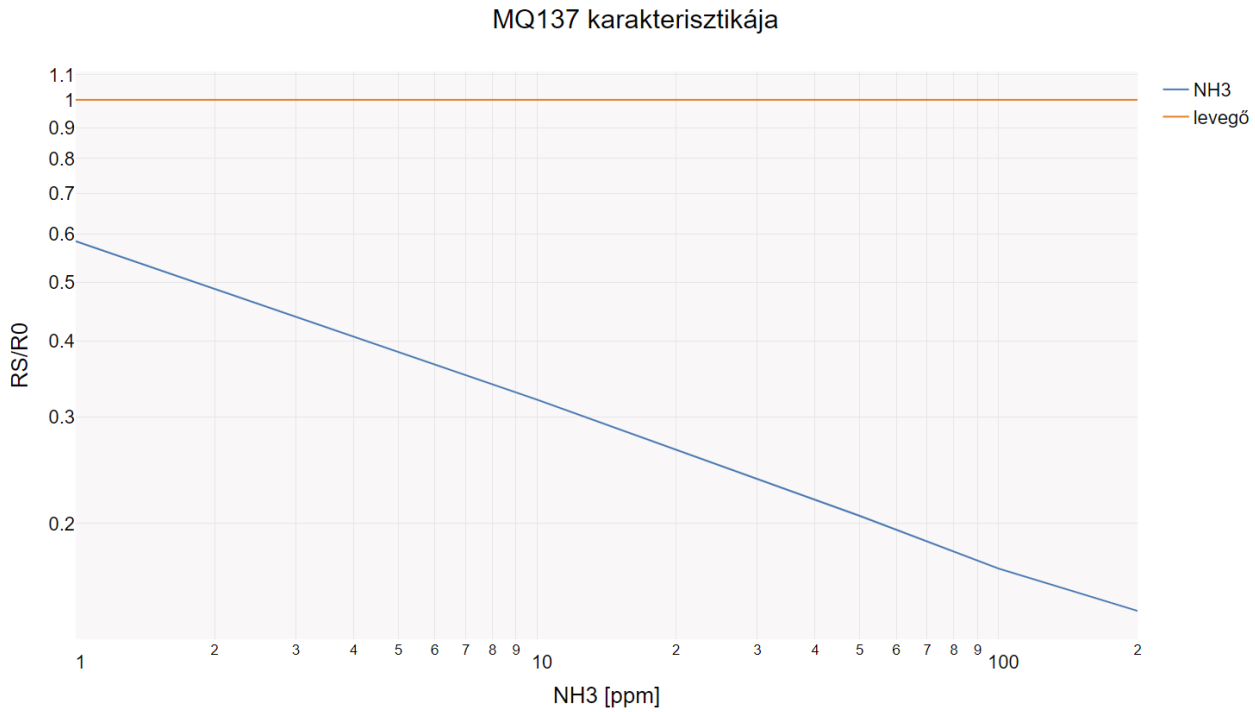
MQ4 = 1485 $k\Omega$

MQ5 = 61.5 $k\Omega$

MQ6 = 568.24 $k\Omega$

MQ135 = 124 $k\Omega$

Ahhoz, hogy tudjam a tesztelés során a levegő ammónia koncentrációját, alapul vettem az MQ137-es szenzor karakterisztikáját (Winsen gyártotta szenzor adatlapját [17] alapul véve, https://apps.automeris.io/ és https://plot.ly/ weboldalakkal digitalizálva a nagyobb pontosságért):



13. ábra - MQ137 karakterisztikája

Ebből, ha tudom R0, RL és V_{RL} értékét, meghatározható az NH3 ppm értéke. Ehhez szükség van a karakterisztika egyenletére. Ezt a karakterisztika 3 pontjából meghatározhatjuk [18]:

A logaritmikus skálán lévő egyenes egyenlete: log(RS/R0) = m*log(PPM) + b
m: meredekség,
b: egyenes metszéspontja NH3 = 1 ppm-mel (itt pl. $10^b = 10^{-0.2276} = 0.5921$),
y1..y3: RS/R0 értéke(növekvő sorrendben), **x1…x3:** PPM értéke(növekvő sorrendben)

$$m = [\log(y3) - \log(y1)] / [\log(x3) - \log(x1)]$$

$$\mathbf{m} = \log(0.16846/\ 0.58427) / \log(100/1) = \mathbf{-0.2701}$$

$$b = \log(y2) - m*\log(x2)$$

$$\mathbf{b} = \log(0.20587) - (-0.2701)*\log(50) = \mathbf{-0,2276}$$

Ezekből pedig meghatározható a PPM értéke az alábbi képlettel:

$$\mathbf{PPM} = \mathbf{10}^{\frac{\log(RS/R0)-b}{m}} = \mathbf{10}^{\frac{\log(RS/R0)-(-0,2276)}{-0.2701}}$$

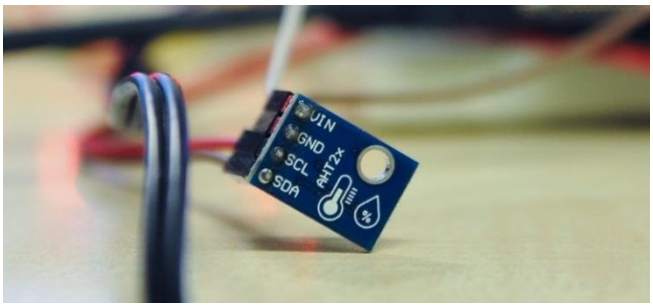
3.3.3 Egyéb eszközök

A tesztrendszer megépítéséhez szükségem volt néhány további eszközre, szenzorra is, amiket itt felsorolok.

Hőmérséklet és páratartalom mérő szenzor

Ahogy az irodalomkutatásom alatt megállapítottam, a baromfikra és az ammónia koncentráció mértékére is nagy hatással van a hőmérséklet és páratartalom, így fontosnak gondoltam ennek a mérését is.

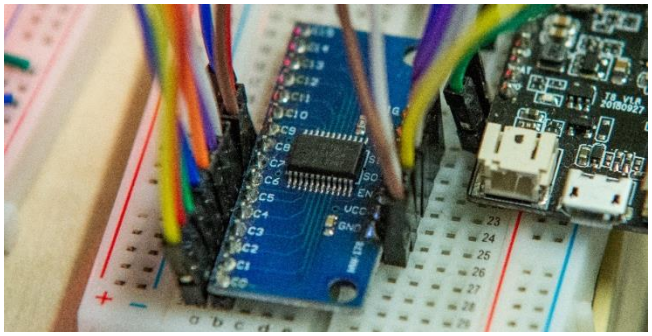
Ezek méréséhez egy AHT21 típusú szenzort használtam, mely hőmérsékletet és páratartalmat is képes mérni, ezen túl méretéből adódóan igen kompakt eszköz. Ez a szenzor I2C-n keresztül kommunikál, illetve az Adafruit termékcsalád könyvtára könnyedén használható a programozás során. Az I2C miatt az SCL-t az ESP 22-es, SDA-t a 21-es bemenetére kötöttem.



14. ábra - AHT21 hőmérséklet és páratartalom méréséhez

Multiplexer

A könnyebben átlátható rendszer érdekében és a sok szenzor miatt egy 16 csatornás multiplexert is használtam a rendszer megépítésénél. A multiplexerhez csatlakoztatva a szenzorok analóg kimenetét, megfelelő vezérléssel sorban kiolvashatjuk a rákötött jeleket. Ez jelentősen megkönnyítette a szenzorok adatainkat mintavételezését,mivel elég volt az ADC1-6-os lábra (GPIO34) kötni a multiplexer kimenetét.



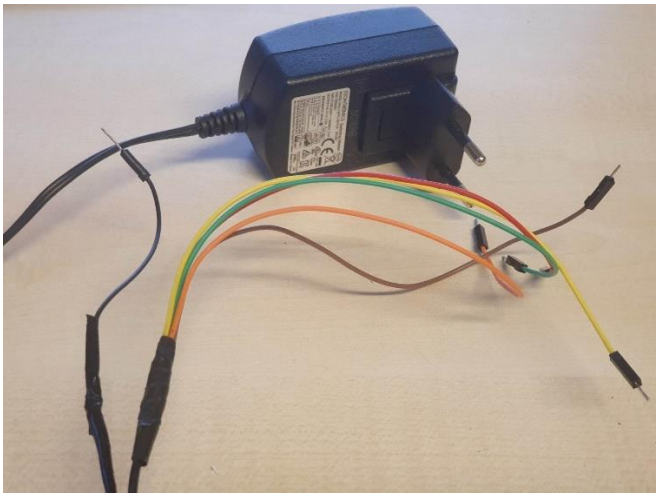
15. ábra - CD74HC4067 típusú multiplexer

Tápegység

A tesztrendszer összerakása során a legnagyobb kihívást az okozta, hogy a 10 db MQ szenzor egyenként ~150mA áramot igényel, így összességében végül min. ~1,5 A áramot kellett biztosítani a rendszer számára. Ez ilyen környezetben viszonylag soknak számít, így olyan probléma is felmerült, ami egy egyszerűbb rendszer esetén nembiztos, hogy előfordul. Ez a tápegység kábele miatti feszültségesésben mutatkozott meg.

Elsőnek egy 5V, 3A-es tápegységről próbáltam egy-egy kábellel a földet és 5V-ot tovább vinni a breadboardra, azonban így a teljes ~1.5A áram egy vezetéken folyt. A használt ~10 cm-es kábel ellenállása 200 $m\Omega$ volt, ami így $R*I^2 = 0.2\ \Omega * (1.5A)^2 = 0.45V$ feszültségesést eredményezett, amit a gyakorlatban le is tudtam mérni, a szenzor bemenetére ~4.6V került, amely rontott volna a mérési eredményeken. Ezt úgy tudtam kiküszöbölni és helyesen 5V-ot adni a szenzorok bemenetére, hogy a pozitív feszültségű kábelt 5 db kábellel 5 csatlakozási ponton csillagkapcsolásban adtam rá a breadboard „+” feszültségű sínjére. Ebben a konstrukcióban a kábelek összesen $R = \frac{(0.2\ \Omega)^5}{5*([\![0.2\ \Omega)]\!]^4} = 0.04\ \Omega \rightarrow \Delta V = 0.04\ \Omega * (1.5A)2 = 0.09V$ feszültségesést fognak okozni, ami már jóval kedvezőbb. A mérések során megfigyeltem, hogy különböző gyártású, de azonos hosszú kábelek is egészen különböző feszültségeséseket produkáltak, amit a vezetékek eltérő anyagának és keresztmetszetének tudok be.

A használt tápegység először egy 5V, 3A-es táp volt, melyet később lecseréltem egy megbízhatóbb 5.1V, 2.5A-es tápra, melynél magasabb feszültsége miatt nem okozott gondot az új 0.09V-os feszültségesés sem, 5V került a szenzorok bemenetére.



16. ábra 5.1V, 2.5A-es táp a szétosztott vezetékével

Feszültségosztó

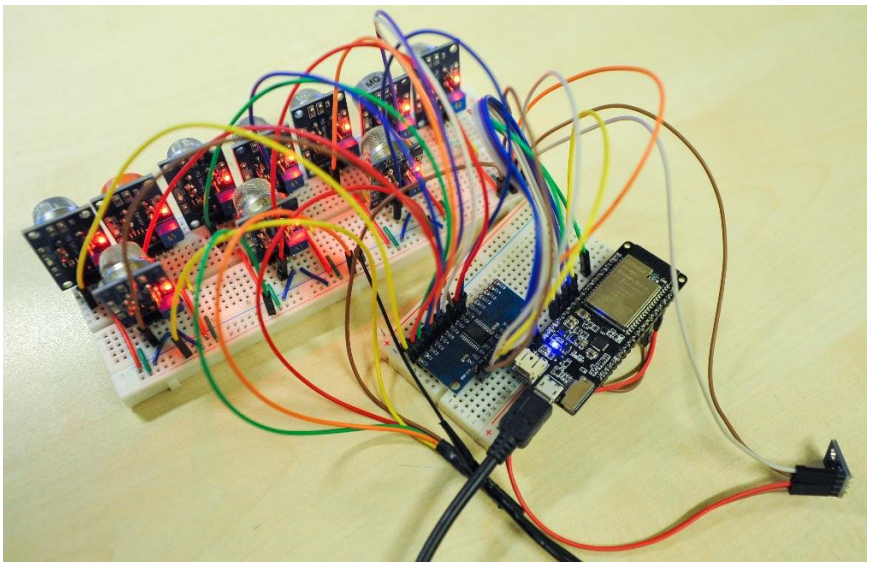
Az ESP32 ADC 3.3V-os mérési tartománya és az MQ szenzorok 5V-os elvi maximális kimeneti feszültségének különbsége miatt egy 3 ellenállásból álló feszültségosztót is gondoltam beiktatni a breadboardra, hogy az ADC bemenetére eső feszültséget arányosan csökkenthessem. Ehhez egy 20 $k\Omega$ -os és két párhuzamosan kapcsolt 20 $k\Omega$ -os (RxR = R/2 → így 10 $k\Omega$) ellenállást használtam. A szenzor kimenetét sorban kapcsoltam a 20 $k\Omega$, eredő 10 $k\Omega$ ellenállásokkal, illetve a GND-vel. A 10 $k\Omega$-on mérhető feszültség így maximálisan 3.33V, mely már optimális az ESP ADC-jéhez. Ez a megoldás elméletben jó, gyakorlatban viszont az ADC alacsony feszültségen való gyenge érzékenysége és több szenzor alacsony kimeneti feszültsége miatt végül kihagytam a tesztrendszerből. Ez a felépítés magasabb feszültségeket mért, amikből így pontosabb eredményeket hozhattam ki.

Breadboard, kábelek

A rendszerhez használtam még két breadboardot, illetve kábeleket is. Mivel sok szenzorom van, amiknek szükséges feszültséget, földet vinni és onnan az analóg kimeneti jelet a multiplexerhez, így nagyon sok kábel kellett. Ezek egy részét merev vezetékkel oldottam meg, amik a tápot és földet vitték a breadboard + és – jelű szekcióiról a szenzoroknak, a mért feszültségértékeket pedig 10cm-es hajlékony vezetékekkel vittem egy másik breadboardon lévő multiplexernek és ESP-nek, így egy kicsit átláthatóbb lett a rendszer.

Kész rendszer

A teljes, kész tesztrendszer látható összeállítva a 17. ábrán:



17. ábra - Tesztrendszer

3.4 Használt technológiák, szoftverek

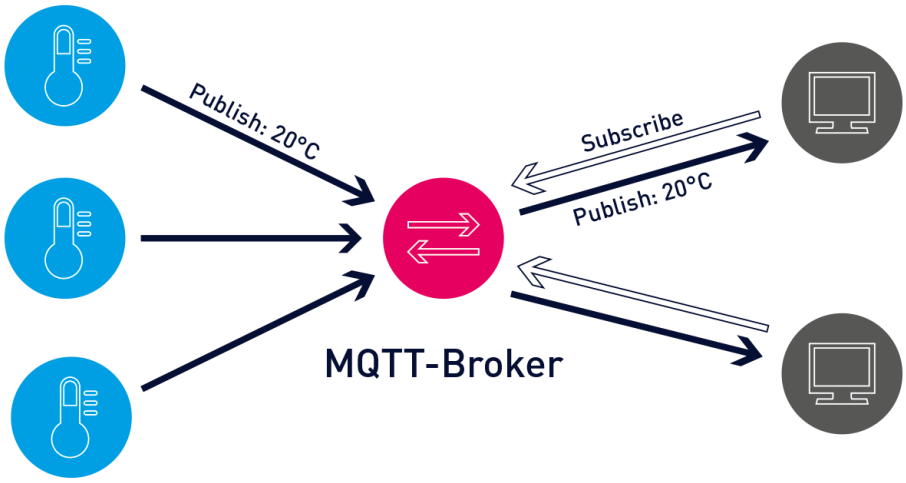
Az ESP32 mikrokontroller programozásához, mérési adatok feldolgozásához és kiértékeléséhez, illetve a kommunikációs protokoll megvalósításához szükség van néhány szoftverre, programozási környezetre, amiket ebben a fejezetben fogok bemutatni.

3.4.1 MQTT

Mivel egy baromfitelepen akár nagyobb távolságokat is át kell hidalni a rendszerrel, így logikusnak gondoltam valamilyen vezeték nélküli megoldás használatát. Ezzel sokkal könnyebben bővíthető a rendszer és például így az adattovábbításért felelős vezeték meghibásodása, elszakadása sem okozhat gondot.

Az MQTT (Message Queue Telemetry Transport) [19] egy „pehelysúlyú”, publish-subscribe modellt használó, TCP/IP transzport rétegre épülő hálózati protokoll. A pehelysúly itt arra utal, hogy rendkívül kicsi a sávszélesség igénye, így nagyon gyors és kevésbé stabil hálózatoknál is jól alkalmazható. Egyre növekvő népszerűségének legfőbb oka a használatának egyszerűsége és kis memóriaigénye, ami IoT (Internet of Things) eszközöknél nagyon fontos.

A protokoll működését az 18. ábra jól szemlélteti:



18. ábra - MQTT működése [20]

A működés 3 hálózati szereplő között zajlik, az ábra bal oldalán kék színnel látható publisher, a jobb oldali szürke subscriber és a középső MQTT-broker. A publisherek és subscriberek másnéven kliensek, amik egyaránt tudnak publisherként vagy subscriberként is működni. Ilyen például az ESP32 vagy a Node-Red, amit később mutatok be.

A publisher feladata az adatok küldése a brokernek különböző topic-okon. Egy topic-ra küldhet több publisher is adatot. Egy üzenetben a fejléc tartalmaz több, az adott üzenetre jellemző beállítást. Ezek közül a legfontosabb, amit én is használni fogok, a QoS, azaz Quality of Service (szolgáltatás minősége). A QoS 3 szintje azt határozza meg, hogy egy üzenet küldése milyen biztosítással történjen.

A QoS 0 az alapvető beállítás, „küld és felejt” jellegű, tehát a brokertől nem kap megerősítést az üzenet megérkezéséről, csak egyszer elküldi és onnantól nem foglalkozik az üzenet sorsával.

A QoS 1 már egy biztosabb küldést tesz lehetővé, legalább egyszer biztosan megérkezik a csomag. Ebben az esetben első próbálkozáskor az üzenet kap egy message ID-t, a publisher a fejléc DUP flag-jét 0-be állítja, ami a duplikálást jelzi 1 állásban. Ha az üzenetet a broker megkapta, visszaküld egy PUBACK típusú üzenetet a message ID-val együtt, amivel jelzi, hogy megérkezett az üzenet. Ha egy meghatározott ideig nem érkezik, akkor a küldő újra megpróbálkozik, de ebben az esetben egy olyan üzenettel, amiben a DUP-ot 1-be állította. Ez addig megy tovább, amíg vissza nem érkezik a PUBACK üzenet, így, ha valami miatt a broker nem válaszol, többször is nála lehet a csomag.

QoS 2 esetén pontosan egyszer kell megérkezni az üzenetnek. Az ehhez használt logika eleje megegyezik a QoS 1-es logikájával, azonban itt a broker nem PUBACK üzenettel válaszol, hanem egy PUBREC-cel, amivel jelzi, elmentette az üzenetet. Ekkor a publisher válaszol a brokernek egy PUBREL-lel, majd a broker kiüríti a mentéshez használt buffert és az üzenetküldés egy publishernek válaszolt PUBCOMP üzenettel zárul. [21]

Ezek közül én a QoS 2-t fogom használni, mivel a mérési adatok közléséhez ez tűnt legoptimálisabbnak, hogy ne maradjon ki adat, illetve nem is legyen duplikálás.

A subscriber a brokernél adott topic-okra feliratkozva fogadja az üzeneteket. A topic-ok az üzenetek osztályozása miatt fontos. A topic-ok fa elrendezésben vannak, hierarchikusan szervezettek, ami egy olyan struktúrát jelent, mint például a Windows-os mapparendszer. Van egy fő topic, ami az én esetemben az „esp32/”, amihez tartoznak altopic-ok, például „esp32/MQ137”, „esp32/temperature”. A szenzorjaim ezekre a topic-okra gyártanak adatokat, amit az ESP elküld a brokernek.

A broker a kommunikáció középpontja, hozzá érkeznek be a publisherek által küldött üzenetek és nála lehet feliratkozni az adott topic-okra.

3.4.2 ThingSpeak

A ThingSpeak egy kifejezetten IoT alkalmazásokhoz készült weboldal, ami a MATLAB-os integrációval végezhető adatelemzést tesz lehetővé. A számomra kedvező tulajdonsága, hogy MQTT broker-ként is működik, így használható az én projectemhez is. A negatívuma, hogy csak interneteléréssel működik, így a kezdeti lendület után rá kellett jönnöm, hogy nem ez lesz a megfelelő broker a projecthez, mivel egy baromfitelepen nem feltétlen biztosított a kellő internetelérés.

A rendszer első változatát azonban még önálló labor keretein belül ennek segítségével fejlesztettem le, amikor még nem tértem ki mindenre, így az első teszteredmények a rendszerrel ezen platform segítségével születtek. Nagy előnye az volt, hogy valós időben láthattam interaktív felületen grafikonokon az adatokat, illetve a későbbiekben implementálásra került adatfeldolgozás itt valós időben történhet. Íme egy kezdeti tesztelés eredménye a 19. ábrán:

19. ábra - ThingSpeak-es teszt (kézfertőtlenítős fújás a szenzorok mellett)

3.4.3 Mosquitto

A Mosquitto egy széles körben ismert és alkalmazott nyílt forráskódú MQTT broker. Számomra a legnagyobb előnye, hogy a ThingSpeak-kel ellentétben internethozzáférés nélkül is működőképes, így akár saját helyi hálózaton is használhatjuk.

Ez a broker az én esetemben a laptopomon működött, amin Windows 10 operációs rendszer fut, így a később bemutatott parancsokat ennek megfelelően írtam. A telepítése egyszerű, a telepítő végig vezet a szükséges beállításokon és már fent is van a rendszerünkön [22]. A brokert lehet manuálisan is indítani, de a szolgáltatás, ha úgy telepítettük fel, a rendszer indításakor automatikusan is elindul.

A manuális indításhoz az *„sc start mosquitto”* parancsot használhatjuk a parancssorba írva. A sikeres indulást a *„sc query mosquitto”* paranccsal ellenőrizhetjük.

Alapértelmezetten a broker a 1883-mas porton figyeli a forgalmat. Az első indításkor az én laptopomon még nem volt ez a port megnyitva, így a Windows beállításokon belül létre kellett hozni ezt. Ezután a *„netstat -an | findstr 1883”* parancs beírásával megnézhetjük, a hálózati kapcsolatokat a 1883-mas porton. Látható, hogy a 192.168.137.1:1883-on kapcsolódik a laptophoz az ESP32:

```
C:\Users\szent>netstat -an | findstr 1883
  TCP    0.0.0.0:1883           0.0.0.0:0              LISTENING
  TCP    127.0.0.1:1883         127.0.0.1:56208        ESTABLISHED
  TCP    127.0.0.1:56208        127.0.0.1:1883         ESTABLISHED
  TCP    192.168.137.1:1883     192.168.137.201:58888  ESTABLISHED
  TCP    [::]:1883              [::]:0                 LISTENING
  TCP    [::1]:1883             [::1]:57374            TIME_WAIT
```

20. ábra – A 1883-mas port

Lehetőség van Mosquitto-n belül publisher-t vagy subscriber-t létrehozni. Például a *„mosquitto_sub -i mosq_sub1 -t „esp32/#” -d -q2”* paranccsal feliratkozhatunk az esp32/ alatt lévő összes topicra, és láthatjuk a hálózati forgalmat is:

```
C:\Users\szent>mosquitto_sub -i mosq_sub1 -t "esp32/#" -d -q 2
Client mosq_sub1 sending CONNECT
Client mosq_sub1 received CONNACK (0)
Client mosq_sub1 sending SUBSCRIBE (Mid: 1, Topic: esp32/#, QoS: 2, Options: 0x00)
Client mosq_sub1 received SUBACK
Subscribed (mid: 1): 2
Client mosq_sub1 received PUBLISH (d0, q0, r0, m0, 'esp32/temperature', ... (5 bytes))
28.26
Client mosq_sub1 received PUBLISH (d0, q0, r0, m0, 'esp32/humidity', ... (5 bytes))
41.07
Client mosq_sub1 received PUBLISH (d0, q0, r0, m0, 'esp32/MQ2', ... (6 bytes))
827.00
Client mosq_sub1 received PUBLISH (d0, q0, r0, m0, 'esp32/MQ3', ... (7 bytes))
3043.00
Client mosq_sub1 received PUBLISH (d0, q0, r0, m0, 'esp32/MQ4', ... (6 bytes))
835.00
Client mosq_sub1 received PUBLISH (d0, q0, r0, m0, 'esp32/MQ5', ... (6 bytes))
186.00
Client mosq_sub1 received PUBLISH (d0, q0, r0, m0, 'esp32/MQ6', ... (6 bytes))
230.00
Client mosq_sub1 received PUBLISH (d0, q0, r0, m0, 'esp32/MQ7', ... (5 bytes))
64.00
Client mosq_sub1 received PUBLISH (d0, q0, r0, m0, 'esp32/MQ8', ... (6 bytes))
246.00
Client mosq_sub1 received PUBLISH (d0, q0, r0, m0, 'esp32/MQ9', ... (5 bytes))
65.00
Client mosq_sub1 received PUBLISH (d0, q0, r0, m0, 'esp32/MQ135', ... (4 bytes))
0.00
Client mosq_sub1 received PUBLISH (d0, q0, r0, m0, 'esp32/MQ137', ... (6 bytes))
919.00
```

21. ábra - Subscriber létrehozása Mosquitto-n belül és a megérkezet adatok

A laptopomon lévő brokerhez az ESP úgy csatlakozott, hogy bekapcsoltam a laptop „Mobil elérési pont”-ját, amihez az ESP-vel wifin keresztül kapcsolódtam.

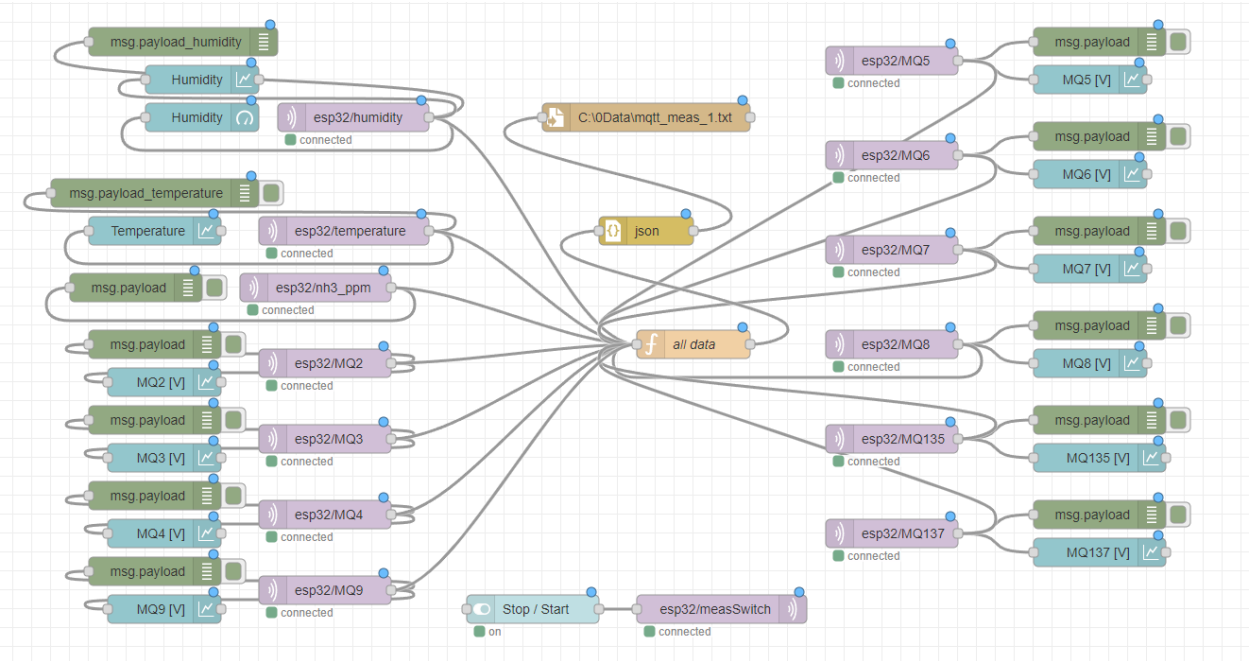
3.4.4 Node red

Az ESP által küldött adatok valós idejű monitorozásához is szükségesnek véltem megoldást találni, mivel fontos, hogy azonnal tudjam, ha esetleg valami hiba történt mérés során, hogy ne vesszen kárba a 10-20 perces mérés.

Ehhez egy jó megoldás a Node-RED, ami különböző hardverekhez, IoT alkalmazásokhoz használható egyszerű, folyam alapú programozási eszköz. [23] Jó tulajdonsága, hogy képes MQTT üzenetek olvasására is, így kliensként is használhatjuk.

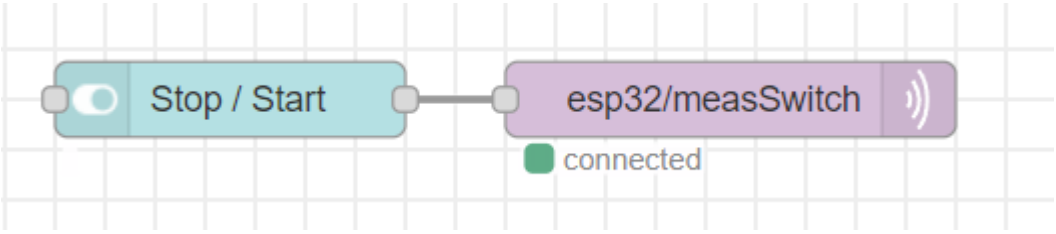
A Node-RED telepítése szintén nem nagy nehézség, npm csomagkezelővel könnyedén telepíthető Windows-ra a *„npm install -g --unsafe-perm node-red”* paranccsal. Indítani szintén parancssorból lehet, ha beírjuk a nevét: „*node-red*”. Indítás után elérhetővé válik a *http://localhost:1880/* címen pedig elérjük a Node-RED fejlesztői felületét, ahol a különböző modulokból létrehozhatjuk a szükséges logikai működést a kliensünknek.

A 22. ábrán látható a teljes működés. A lila modulok feliratkoznak különböző topic-okra. A kék színűek a grafikonokért felelősek, amiket futás közben a felhasználói oldalon láthatunk. Ennek beállításánál meghatározhatjuk, hogy pontosan milyen limitek között látszódjon a grafikon, mi legyen a felirata és a színe. A zöldek az adatok kiolvasását végzik az üzenetből. A középső *„all data”, „json”, „C:\0Data\mqtt_meas_1.txt”* együttesen a megadott helyre egy txt-be mentik a mérési adatokat.



22. ábra - Node-RED folyamatábra

A 23. ábrán látható a Stop/Start blokk, ami egy kapcsolót valósít meg. Ha a kapcsolót állítjuk, az az *„esp32/measSwitch”* topic-ra küld egy üzenetet, ami a mérés elindítását, vagy leállítását oldja meg. Ez azért kellett, mert szükséges volt valamilyen manuális módszer arra, hogy a mérést vezérelni tudjuk, fizikai kapcsoló pedig nem állt rendelkezésre, így kihasználva a Node-RED funkcióit, ez is belekerült. Ezzel a kliens publisherként is működik.



23. ábra - Mérés elindításáért felelős kapcsoló moduljai

A Dashboard vagy felhasználói felület a *http://localhost:1880/ui* címen érhetjük el. Ezen a felületen láthatjuk a fejlesztői oldalon lévő modulokból összerakott grafikonokat és bal oldalon fent a kapcsolót. A 24. ábrán sajnos az MQ3-mas szenzoron éppen látszik is, hogy hibás eredményt mutat (végig 4096-ot mutat, ami 3.3V-ot jelentene, de ez nem egyezik a valósággal), így mérés közben tudtam orvosolni a hibát és nem csak több mérés után láttam meg, hogy valami gond van.

24. ábra - Valós idejű grafikonok a Node-RED felületén

3.4.5 Arduino IDE

Az ESP32 programozásához kell egy programozói környezet, ahol megírhatom a kódot és lefordítja azt úgy, hogy az ESP is értse és feldolgozhassa. Ehhez a nyílt forráskódú Arduino integrált fejlesztői környezetet (IDE) használtam.

Ezen fejlesztői környezet előnye, hogy rengeteg kész könyvtár létezik hozzá, amikben előre megírva szerepelnek a különböző szenzorok, hardverek vezérlési logikái és beépített soros kommunikációja is van, amivel könnyen csatlakoztathatjuk a mikrokontrollert hozzá, majd flashelhetjük. Az Arduino saját szintaktikát használ, ami a népszerű C++ programozási nyelven alapszik [24] és sok egyszerűsítés is van benne, így mindenki egyszerűen beletanulhat.

Az Arduino kódok .ino kiterjesztésű fájlokban vannak és 2 fő részből épülnek fel. Ezek két nagy függvényt jelentenek, egyik a `setup()`, másik pedig a `loop()`. A `setup()` függvény a mikrokontroller indításakor fut le egy alkalommal, így ebben a futáshoz szükséges beállításokat, paraméterezéseket kell elvégezni. Ide kerül például az inputok és outputok beállítása, wifi setup-olása, MQTT brokerhez való kapcsolódás, SD kártya inicializálása. A `loop()` függvényben a vezérlési logika, ehhez szükséges függvényhívások kerülnek. Ennek a függénynek az a tulajdonsága, hogy folyamatosan újra és újra lefut, így például ide kerül az ESP32 ADC-je által minden lefutott periódusban mért feszültség, MQTT publish függvénye, amivel az üzeneteket küldi az ESP a brokernek és az SD kártyára való írás is.

R0 számítása

A mérésekhez először is szükség volt egy olyan logikára, ami a szenzorok R0 értékét kiszámolja. Ez ugye a szenzorok szabad levegőn mért ellenállása, ami egy fontos paraméter a karakterisztikájuk számolásakor. Az R0 értéket a szenzorok V_{RL} feszültségéből, tehát az AOUT kimeneten mérhető feszültségből tudjuk kiszámítani, illetve a szenzorok áramkörén lévő RL ellenállásból. RL és R0 ellenállások meghatározását korábban a 3.3.2-es fejezet *„Előzetes számítások”* részénél bővebben is kifejtettem, most pár mondatban bemutatom a hozzá szükséges kódot is. (A használt kódok a Függelékben található GitHub linken elérhető)

A számításhoz szükséges feszültségértéket a `mux_16()` float-ot visszaadó és egy int paramétert váró függvény adja át. Ez a függvény valósítja meg a multiplexer logikáját úgy, hogy átadjuk neki a kiolvasandó csatorna száma (ez 0 és 9 között van, mivel összesen 10 csatornáját használjuk a 10 szenzorhoz), majd ő a `digitalWrite()` függvénnyel aktiválja a mux megfelelő csatornáját, majd az `analogRead()` függvénnyel az ADC-n kiolvassa a feszültségértéket, amit a függvény vissza is ad.

Ez a kiolvasást egy for cikluson belül 500-szor végbemegy, egy változóhoz hozzáadja a kapott értékeket, majd elosztja 500-zal, így az értékek átlagát kapjuk. Ebből az értékből a fentebb már bemutatott R0 = RL*(Vcc/V_{RL} -1) képlettel megkaphatjuk R0 értékét, amit a soros kimeneten keresztül kiír a laptopon futó Arduino IDE-ben.

Elektronikus orr kódja

Maga a tesztrendszer kódja ennél jóval összetettebb. Az átláthatóság érdekében a különböző funkciókért felelős függvényeket külön fájlokba szeparáltam, amiket az Arduino képes úgy felolvasni, mintha beimportáltam volna őket. A project nevét viselő fájlt olvassa először, majd betűrendben a többit, amik ugyanabban a mappában vannak. A főbb funkciók, függvények tehát 5 külön fájlban vannak implementálva.

A multiplexer működése a *mux.ino*-ban van implementálva a `mux_16()` nevű függvényként, működése megtalálható az előző *R0 számítása* résznél. Egy külön fájlba került még az ammónia koncentráció számítása is, ez a *calculate_ppm.ino* nevű fájlban van. Az ebben lévő függvény megkapja az MQ137-es ammónia szenzor feszültségértékét, majd ebből a 3.3.2-es fejezetben leírtaknak megfelelően a kiszámítja a karakterisztika alapján a ppm értéket.

Az SD kártyára való írás függvényei az *sd_card.ino* -ban található. Itt az `appendFile()`, `SDWriteStart()` és az `SDWriteEnd()` függvények vannak. Az `appendFile()` minden ciklus végén a kiolvasások és számolások után az aktuális fájl elérési útvonala alapján megnyitja azt, és beleírja a kapott karaktertömböt. Írás után bezárja a fájlt. Ez az utolsó lépés nagyon fontos, mivel, ha nincs bezárva a fájl és esetleg úgy vesszük ki, akkor meghibásodhat a memóriakártya. Az `SDWriteStart()` függvény a mérés indításakor, tehát az *„on”* üzenet *„esp32/measSwitch”* topicra érkezésekor hívódik meg. Megkeresi, hogy milyen elnevezésű mérési fájlok készültek eddig (meas_X.txt, ahol X egy egész szám), majd ha mindet megtalálta, létrehozza a következő mérési fájlt, amibe egyből bele is ír egy „[”-t az `appendFile()` függvényt meghívva. Ez a nyitott szögletes zárójel a memóriakártyára írt json formátum miatt kell, amit majd pythonban könnyen beolvashatok. Az *„off”* üzenet *„esp32/measSwitch”* topicra érkezésekor az `SDWriteEnd()` egy plusz „]”, záró szögletes zárójelet ír a mérés végén az aktuális méréshez, amivel az elején nyitott zárójelet bezárja.

Az *mqtt.ino* fájlban az összes MQTT protokollhoz és kommunikációhoz szükséges függvény található. Ahhoz, hogy MQTT-vel tudjunk üzeneteket küldeni, először fel kell építeni egy wifis, vezeték nélküli kapcsolatot a laptop és ESP között. Ezt a `setup_wifi()` függvény végzi. Ez a fő fájlban a `setup()` függvényen belül van meghívva, illetve minden ciklus elején is

megvizsgáljuk, hogy van-e kapcsolat, ha megszakadt, újra meghívódik. A `callback()` függvény akkor hívódik meg, amikor üzenet érkezik az *„esp32/measSwitch”* topicon, a hívást a beépített *PubSubClient* osztály loop függvénye végzi, ami szintén minden ciklus elején lefut. Itt kiolvashatjuk a Node-Red által küldött *„on”* vagy *„off”* üzenetet, ami a mérés elindításának és leállításának triggere, így itt hívódik meg a `SDWriteStart()` és `SDWriteEnd()`. Ha *„on”* üzenetet kap, akkor egy változóban elmenti az aktuális időt (ESP futásának kezdetétől eltelt ms idő), amiből később minden kártyára való íráskor kiszámoljuk az adott mérés elejétől számolt időt ms-ben, hogy a mérési fájlba így mindig 0-tól kezdődjön az idő. Az MQTT brokerhez a `reconnect()` függvénnyel tudunk csatlakozni. Ennek a kapcsolatnak a meglétét szintén minden ciklus elején check-oljuk, így, ha valami gond lenne, újra tudunk csatlakozni.

A legfontosabb `setup()` és `loop()` függvények az *elec_nose.ino* fájlban kaptak helyet. Itt a függvényeken kívül még található több globális változó deklarálása, inicializálása. A `setup()` függvényben, ahogy fentebb írtam a kezdeti beállításokat kell elvégezni, amiknek az ESP bekapcsolásakor, újraindításakor le kell futniuk. Itt találhatók például a `setup_wifi()`, MQTT-ért felelős PubSubClient osztály `setServer()`, `setCallback()`, `connect()`, `subscribe()` függvényeinek hívásai, SD kártya és AHT szenzor inicializálása. A `loop()` függvényben 1000ms-unként futnak le a mérést és számítást végző függvények. A futás további feltételei, hogy legyen wifi kapcsolat, az MQTT brokerrel kapcsolódva legyen az ESP, illetve a Node-Red-től megkapja az *„on”* üzenetet és ezután létre legyen hozva a megfelelő mérési fájl a memóriakártyán. Ha bármely feltétel nem teljesül, a mérés leáll egy hibaüzenettel, vagy az ESP megpróbál újracsatlakozni.

A 25. ábrán látható az ESP bekapcsolása, inicializálása és egy mérési ciklusa után soros porton kiírt logja:

```
Connecting to ssid
.
WiFi connected
IP address:
192.168.137.251
AHT10 or AHT20 found
SD CARD INITIALIZED.

The client esp32 connects to the MQTT broker
Public MQTT broker connected

================================================
Message arrived on topic: esp32/measSwitch
Message: on
================================================
Temperature: 28.34 degrees C
Humidity: 40.29% rH
Value of MQ2 is:        891.00
Value of MQ3 is:        3213.00
Value of MQ4 is:        819.00
Value of MQ5 is:        313.00
Value of MQ6 is:        259.00
Value of MQ7 is:        65.00
Value of MQ8 is:        627.00
Value of MQ9 is:        59.00
Value of MQ135 is:      0.00
Value of MQ137 is:      1162.00
Value of NH3 in ppm : 9.272998
================================================
Message arrived on topic: esp32/measSwitch
Message: off
================================================
```

25. ábra - ESP logolása mérés közben

3.4.6 Python

Python [25] manapság az egyik legnépszerűbb programozási nyelv. Nagy népszerűségének okai, hogy nagyon egyszerű szintaktikát használ, így könnyű beletanulni, folyamatosan növekvő és fejlődő könyvtárak és modulok érhetőek el hozzá, amik tovább növelik a felhasználóbarát működést.

A szakdolgozatomhoz az adatok feldolgozásához, grafikonok és logok gyártásához Python-t használtam, mivel szerettem volna jobban elmélyedni és beletanulni nyelv tulajdonságaiba, illetve kitűnő könyvárak/modulok léteznek minden szükséges részfeladat megoldásához.

A project átláthatósága érdekében itt is több fájlba szeparáltam a különböző logikai egységeket, így lettek a következő fájljaim:

- importModules.py
- init.py
- measEval.py
- measPrototype.py

importModules.py

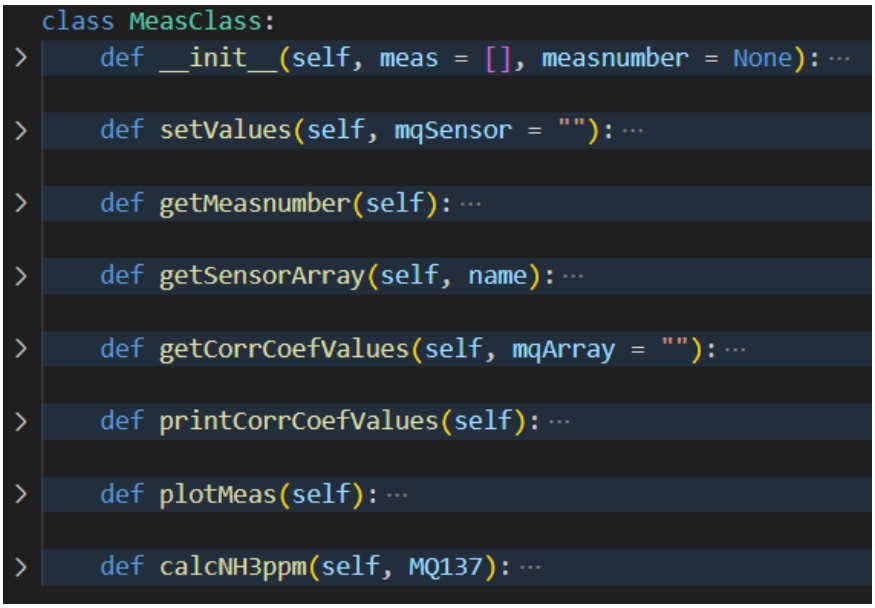
Ez a fájl a különböző külső modulok importálását teszi meg. Ezeket, ha még nincsenek telepítve lokálisan, először a parancssorban telepíteni kell a pip csomagkezelő [26] segítségével. Az itt importált könyvtárak:

- `os`: Az operációs rendszert elérő függvényeket tartalmaz, az `os.listdir()` függvényt használom belőle, amivel el tudom érni az adott útvonalon található mappa összes elemének útvonalát.
- `ast`: A mérések során generált .txt fájlokat beolvasva string-eket kapunk, amiket Python számára érthető változó típusokká kell konvertálni (list, dict), hogy műveleteket végezhessünk az adatokkal. Ehhez az `ast.literal_eval()` függvényt használtam, ami minden felolvasott stringet Pythonnak értelmezhető típussá konvertál.
- `numpy`: A numpy egy széles körben használt könyvtár, amiben matematikai számításokat végző függvények (pl. `numpy.log10`(), `numpy.average`()) vannak implementálva, illetve a `numpy.array()` függvénnyel létrehozhatunk speciális tömböket, amit például ha elosztunk vagy szorzunk egy számmal, akkor az a tömb minden elemén elvégződik.

- `matplotlib.pyplot`: Grafikonok generálásához, azok formázásához található függvényeket ebből a könyvtárból nyertem ki, amikkel szép, precíz grafikonokat tudtam kreálni.

init.py

Ebben a fájlban hozom létre a globális változókat, deklarálom a listákat, dict-eket, itt olvasom be a mérési fájlokból az értékeket, illetve kiválaszthatjuk a mérési adatok forrását (MQTT vagy SD). A beolvasott méréseket az itt implementált `MeasClass` osztályban feldolgozom és az osztály self-es, saját változóiba mentem listaként az adott szenzorokhoz. Vannak benne még az objektum változóinak eléréséért, kiírásért, grafikon készítésért felelős függvények is. Az osztályban található függvények a 26. ábrán láthatók:



26. ábra - MeasClass felépítése

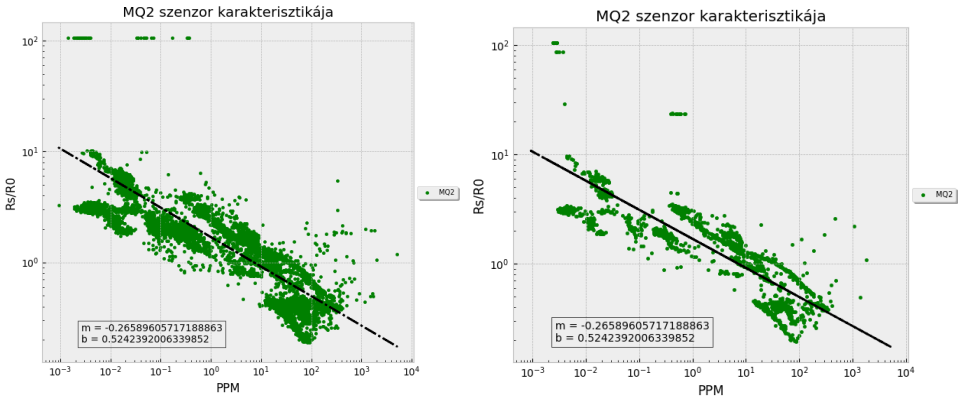
Ezek közül a get-tel kezdődő függvények arra hivatottak, hogy az osztály változóit kívülről is elérjük, a `get/printCorrCoefValues`() függvény kiszámítja/kiírja az adott objektumban lévő szenzorok MQ137- ammónia szenzorhoz mért korrelációs együttható (r) értékeket, a `plotMeas`() az összes szenzor értékeit kiplotolja (grafikusan ábrázolja) egymás után, a `calcNH3ppm`() pedig kiszámítja az MQ137 szenzor értékeiből az ammónia koncentrációt. Megjegyezném, hogy mivel a mérési fájlokban az ADC által kiadott 0-4095 közötti értékek vannak, itt történik a `setValues`() függvényen belül ezek feszültségértékké történő konvertálása is.

A mérési fájlokból itt még létrehozunk `MeasClass` típusú objektumokat, amiket a `measObjects` nevű listába mentünk.

measEval.py

Ebben a fájlban implementáltam a lényegesebb, minden adattal számoló függvényeket, amiket kicsit bővebben is bemutatok:

- `averageOfPPM(x, y, num_points):` A kapott x és y tömbökből num_points méretű ablakokban kiszámít egy-egy átlagot, majd ezekből egy új np.array-t ad vissza. Ez azért volt szükséges, hogy a nagy szórású méréseket kicsit szebbé tudjam tenni. A 27. ábrán látható, hogy ezzel valamivel szebb ábrát kaphatunk, ha num_points = 5-öt állítunk be.



27. ábra - MQ2 szenzor karakterisztikája átlagolás előtt (bal kép) és után (jobb kép)

- `curvefit(x, y):` Ezen a függvényen belül létrehozunk x és y, vagyis nálam a ppm és RS/R0 karakterisztikák adataira illesztett egyenest, ahogy a 27. ábrán fekete színnel látható. Ehhez a függvényen belül kiszámoljuk a `numpy.polyfit`() függvény segítségével az adatokra illesztett egyesen meredekségét és metszéspontját, amiből log(y) = meredekség*log(x)+metszéspont az egyenes egyenlete.
- `getallNH3ppm():` Megadja az összes mérésből az összes NH3 ppm értéket, amit egy numpy tömbként visszaad. Ez a karakterisztikák számításánál és plotolásánál kell.
- `getallRs():` Ugyanúgy, ahogy a ppm értékeknél, az RS értékeket is minden objektumból kiolvasás után kiszámítja a szenzorokra és visszaad egy dict-et, amiben a key-ek a szenzorok nevei, a hozzájuk tartozó érték pedig egy numpy tömb az RS értékekkel.

- **`getAllCCVal()`:** A korrelációs értékek kiírásakor, plotolásakor használom, kiszámít és visszaad egy dict-et, amiben a szenzorok a kulcsok, értékei pedig egy lista, amiben sorrendben a mérések korrelációs együttható értékei szerepelnek.
- **`printAVGCCVal()`:** Kiszámítja az összes mérés alapján az átlagos korrelációs együttható értékét a szenzoroknak.
- **`plotAllCCVal()`:** Kiplotolja az összes méréshez tartozó összes szenzor korrelációs együttható értékeit. Horizontális tengely a mérés száma, vertikális tengelyen 0-1 között láthatóak az r értékek. Ahhoz, hogy az egymás utáni mérések során az r alakulását is jól nyomon követhessük, egy folytonos vonallal össze is vannak kötve a mért értékek.
- **`plotAllMeas()`:** Minden méréshez kiplotolja egy nagy ábrán külön grafikonokban a mérési értékeket az idő függvényében.
- **`plotMQChar ()`:** Az összes mérési eredményt felhasználva ábrázolja egymás után a szenzorok számolt és illesztett karakterisztikáját. Ez gyakorlatilag a legfontosabb kimenete a méréseknek, ez alapján tudunk következtetni a szenzor ammóniára való érzékenységére. Egy kis ablakban bal alul kiírja az illesztett karakterisztika meredekségét és metszéspontját.

A függvények futtatása úgy történik, hogy a fájlokat tartalmazó mappában megnyitunk egy parancssort, ahol elindítjuk a Python-t a „py” vagy „python” beírásával. Ezután importáljuk a measEval-t, importálás után megkérdez a program, hogy mely forrásból szeretnénk, hogy vegye az adatokat, MQTT vagy SD, majd ezek után futtathatjuk a tetszőleges függvényeket. A 28. ábrán látható egy példa a futtatásra:

```
C:\0Data\Egyetem\7._felev\szakdoga__repo\szakdolgozat>python
Python 3.11.3 (tags/v3.11.3:f3909b8, Apr  4 2023, 23:49:59) [MSC v.1934 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license" for more information.
>>> import measEval as m
Adja meg a kívánt forrást! (MQTT / SD):SD
>>> m.printAVGCorrCoefValues()
MQ2 szenzor átlagos r értéke: 0.8516

MQ3 szenzor átlagos r értéke: 0.1468

MQ4 szenzor átlagos r értéke: 0.2118

MQ5 szenzor átlagos r értéke: 0.0738

MQ6 szenzor átlagos r értéke: 0.6273

MQ7 szenzor átlagos r értéke: 0.4890

MQ8 szenzor átlagos r értéke: 0.8274

MQ9 szenzor átlagos r értéke: 0.3010

MQ135 szenzor átlagos r értéke: 0.0000

MQ137 szenzor átlagos r értéke: 1.0000
```

28. ábra - Program futtatása parancssorból

measPrototype.py

A mérések kiértékelése után, ha megfelelőnek ítélek egy szenzort, a karakterisztikájának paramétereit (m – meredekség és b - metszéspont) felviszem a measPrototype.py fájl megfelelő helyére egy dict-ben mentve. Ebből a már meglévő mérések alapján képes leszek megmutatni, hogy az adott szenzorok milyen ammónia koncentráció értékeket mutattak volna valós tesztelés során, tehát egy újabb ellenőrzéssel megbizonyosodhatunk a karakterisztikák és módszer helyességéről vagy hibáiról.

Az ehhez szükséges függvények:

- `calcNH3ppm()`: A manuálisan megadott karakterisztikák alapján kiszámolja a ppm értékeket és visszaadja. Paraméterként vár egy szenzor nevet, illetve egy méréshez tartozó objektumot. Itt még belevittem egy kis korrekciós lehetőséget is, amivel tapasztalati úton, „magic” számokkal (nem számolt, csak tapasztalati úton megalapozott) tudjuk osztásokkal/szorzásokkal korrigálni a számolt ppm eredményeket. A függvény végén még egy filterezést is elvégez, amivel a 100ppm feletti értékeket lenullázza.
- `plotPPM()`: A kiválasztott szenzorok és MQ137-es ammónia szenzor kimenetét ábrázolja egy grafikonon. Ahhoz, hogy a hőmérséklet és páratartalom hatását is láthassuk, ezeket is hozzáadja az ábrához.
- `plotDiff()`: A szenzorok által számolt és ammónia szenzor általi ppm értékek közötti különbségét ábrázolja a mérési pontok során. E függvény által generált grafikon alapján eldönthetjük, hogy mennyire pontosan követik a szenzorok a valós koncentráció értékeit.

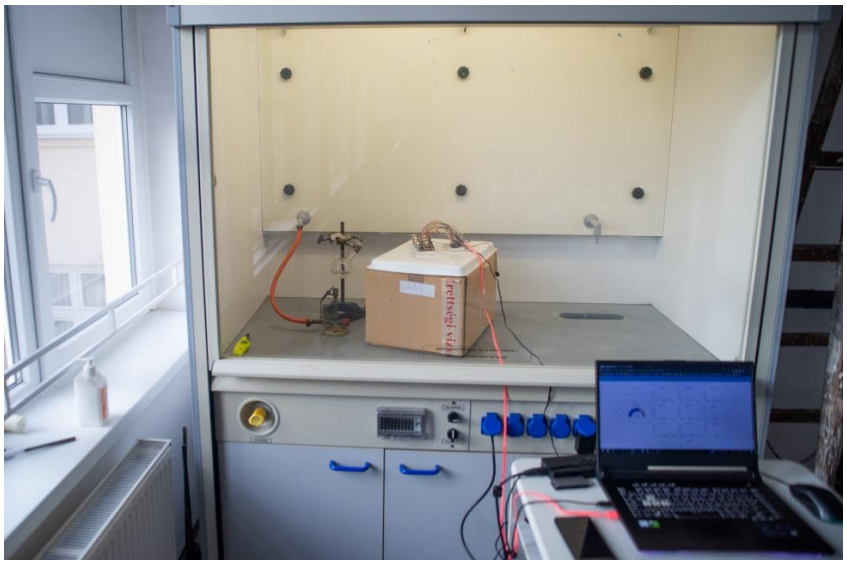
4 Tesztelés

A rendszer megismerése után rátérek a tesztelési metódusok, környezetek bemutatására, illetve a teszteredmények kiértékelésére. A tesztelésre két opció jutott eszembe. Egyik, hogy tesztelhetem a rendszert valós körülmények között, egy baromfiólba helyezve és mivel van ammónia szenzorom, látható lesz, hogy milyen ammónia koncentrációknál hogyan viselkednek a szenzorok. Ez a valós körülmény miatt jónak tűnhet, viszont sok időt igénybe vett volna, ami sajnos nem állt rendelkezésre, illetve jóval több hiba bezavarhatott volna (pl. más gázok, por, nem vízálló konstrukció), amikre nem volt még felkészítve a rendszer. Ezt a tesztelést választotta például 2.4-es Előzetes kutatások fejezetben említett *Spirocco KFT*.

A másik, végül kiválasztott tesztelési módszer az, hogy kifejezetten ammóniára teszteljük a rendszert, aminek előnye, hogy a szenzorok csak ammónia gázzal kerülnek egy légtérbe, így más gáz nem zavar be, jóval szélesebb koncentráció tartományban vizsgálhatóak a szenzorok, amivel pontosabb karakterisztikát tudunk felírni, illetve 10-20 perc alatt is elvégezhető egy mérés. Hátránya, hogy nekem kell előállítani az ammóniát, ami nagy mennyiségben mérgező, illetve igen kellemetlen, szúrós szagú gáz.

4.1 Laboratóriumi környezetben

A volt középiskolám lehetőséget biztosított arra, hogy a kémiateremben lévő, 29. ábrán látható tesztfülkében mérhessek, így ezt kihasználva itt nagy koncentrációban igyekeztem mérni az ammóniát. A mérés úgy állt zajlott, hogy egy Bunsen-égővel melegítettem egy kémcsőben lévő szalmiákszeszt (ammóniás víz), ami itt 30%-os oldatot jelentett, a szenzorrendszert pedig a kémcső szája mellé helyeztem. A szenzorrendszert a mérések során több pozícióba is elhelyeztem (30. ábra), mivel kíváncsi voltam, hogy viselkednek a szenzorok nagyon nagy koncentrációban, illetve kisebben. Mivel az ammónia gáz könnyebb a levegőnél, felfelé száll, így figyelnem kellett, hogy ha magas koncentrációt szeretnék elérni, magasabbra helyezzem a szenzorokat.



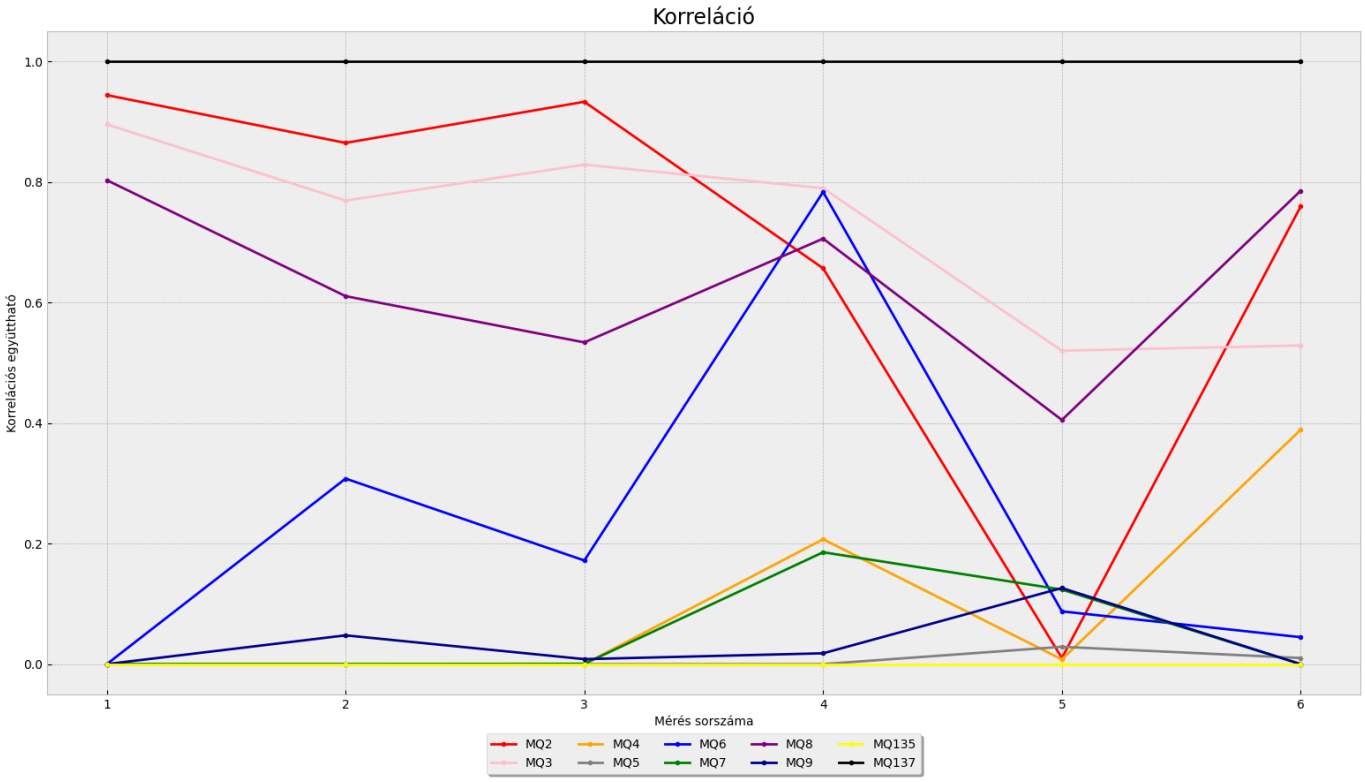
29. ábra - Tesztelési fülke a kémiateremben

A méréseket nagyjából 10-15 percesekre terveztem, az elején tiszta levegőn indítva, majd beindítva az égőt. Ekkor ammónia szabadult fel, nőtt a koncentráció, majd egy idő után leoltottam az égőt és bekapcsoltam az elszívót. Mivel elég nagy térben volt a mérés, így viszonylag több idő kellett ahhoz, hogy a szenzorok megfelelő változásokat mutassanak, így ennek lerövidítése érdekében közelebb helyeztem a szenzorrendszert a kémcső szájához. Ez sajnos azzal is járt, hogy egy óvatlan pillanatban a melegítés hatására kifröccsenő szalmiákszesz eltalált pár szenzort, ami elrontotta az egyik mérést, így ezt a mérést nem használtam fel. A mérések során feltűnt, hogy sajnos elég zajosak a mérési eredmények, amit elsősorban az ESP pontatlan, nem teljesen lineáris ADC-jének tudok be, illetve a használt multiplexer is okozhatott ehhez köthető mérési hibákat. Mérés során folyamatosan figyeltem a szenzorok kimenetét a Node-Red-es kliens segítségével, ami MQTT-n kapta az adatokat, így, ha bármi közbejött azonnal le tudtam állítani a mérést. A mérésekhez a 30. ábrán látható elrendezéseket használtam:

30. ábra - Szenzorrendszer különböző helyeken
(balról: dobozon, dobozon+tálcákkal körülvéve, asztalon)

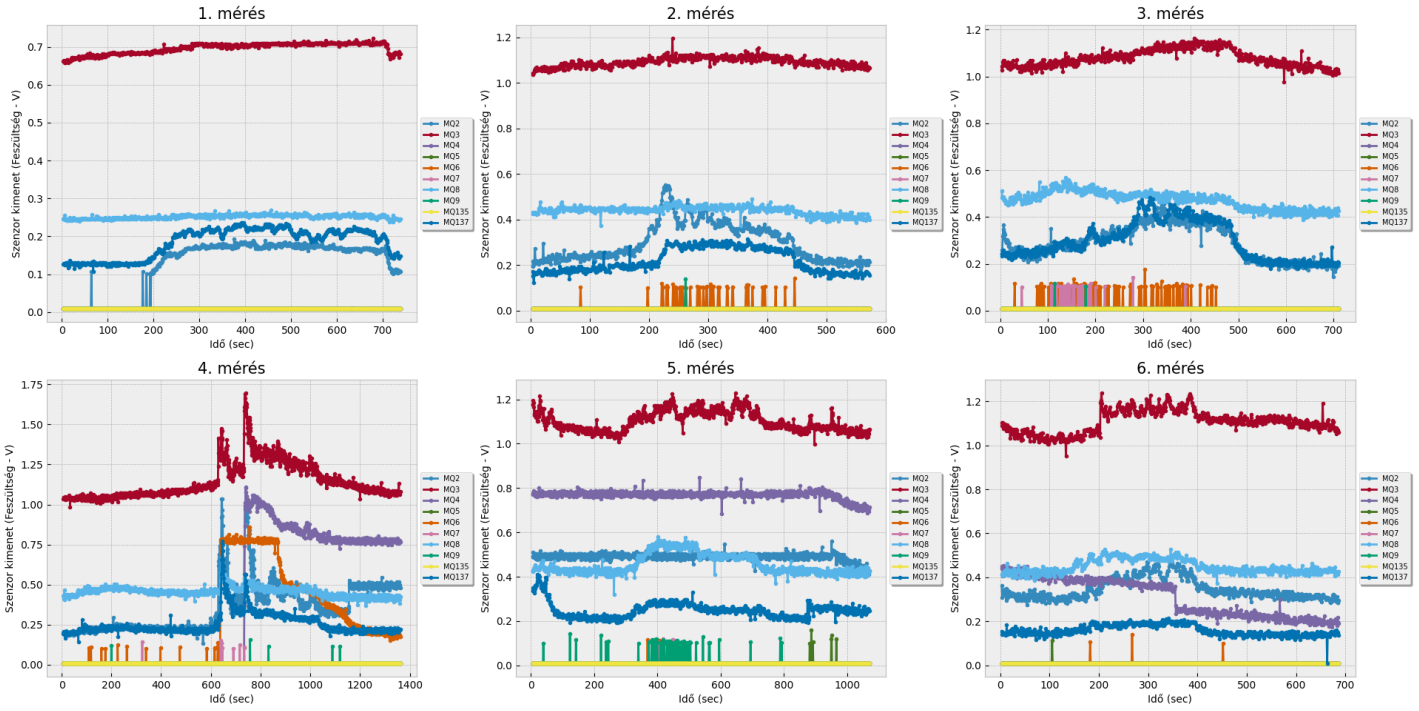
Ebből a mérésből származó eredmények sajnos nem lettek tökéletesek, mivel a nagy koncentrációk miatt nagy kiugrások is keletkeztek, illetve még nem kezeltem olyan magabiztosan a rendszert, így az itt mért összesen 6 mérés is több időt vett igénybe és sajnos nem fért bele több.

Ezen mérések korrelációs együtthatói a 31. ábrán láthatók. Ezen láthatjuk, hogy sajnos elég random módon aktiválódtak a szenzorok és nem lehet igazán megállapítani, hogy melyik mennyire követi az ammóniaszenzor jelét. Ezek miatt újabb mérésekre volt szükség.



31. ábra - Korrelációs együtthatók alakulása laboratóriumi mérés után

A 32. ábrán láthatjuk továbbá a 6 mérés idő- szenzor kimeneti feszültség grafikonjait. Látható, hogy a mérések során elég zajosak a jelek, viszont a sötétkék színű MQ137 ammónia szenzor jelét egészen látványosan követi pár szenzor a kevésbé meggyőző korrelációs együtthatók alakulása ellenére is.

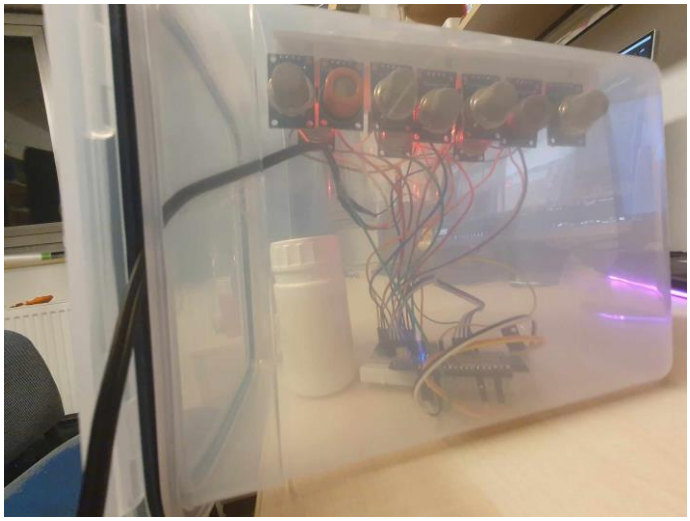


32. ábra - Laboratóriumi mérés eredményei

4.2 Otthoni környezetben

Mivel a laboros mérésből jövő adathalmazok nem voltak elegendően a pontosak, a méréseket meg kellett ismételnem. Ehhez saját, otthoni tesztkörnyezetet állítottam össze, amivel biztonságosan tudtam hasonló méréseket végezni.

A tesztelési összeállítás a 33. ábrán látható:

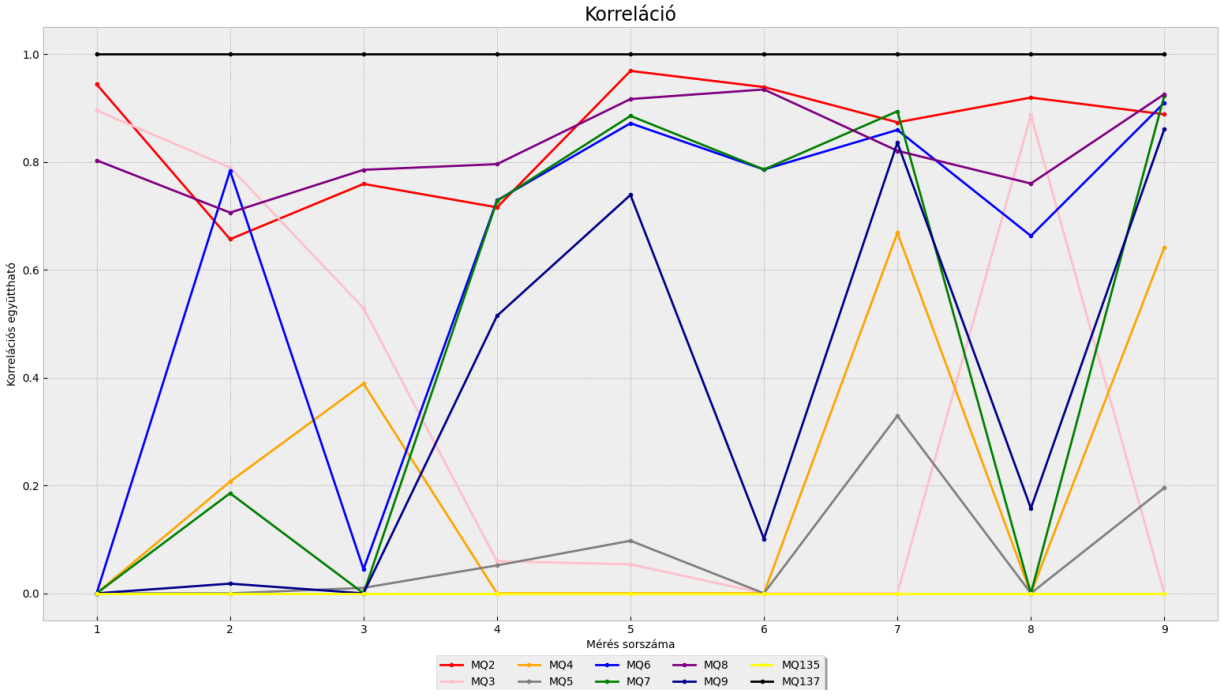


33. ábra - Otthoni tesztelési összeállítás

Az összeállításban egy nagyobb doboz tetejére ragasztottam a szenzorokat, illetve egy tubusba helyeztem alájuk egy kis szalmiákszeszt. A mérések során nem melegítettem az ammóniás vizet, csak párolgott, viszont a nagyon kis térfogat miatt így is látványos eredményeim keletkeztek, amikkel már biztosabb konklúziót tudok levonni. Mivel lezártam a dobozt, így az ammónia koncentráció szépen tudott növekedni, viszont fontos volt, hogy jól szellőző helyen teszteljem, mivel kis mértékben így is szivárgott kifelé a gáz.

Ennek a környezetnek a hátránya, hogy mivel a szenzoroknak nagy a hőtermelése a kis térfogatot nagyon gyorsan felfűti és a páratartalom emiatt igen gyorsan megnő. Ezt egy nagyobb dobozzal tudtam volna orvosolni, de sajnos ez állt csak rendelkezésre. A magas páratartalom és hőmérséklet azonban nagyon befolyásolták a szenzorokat, így csak rövidebb méréseket tudtam végezni, illetve figyelembe kellett venni az eredmények értékelésénél a szenzorok hőmérséklet és páratartalomtól való függését. Ezt a függést sajnos nem tudtam implementálni, mivel az ehhez tartozó karakterisztikákhoz különböző konstans ppm értékeknél kellett volna folyamatosan változtatni a hőmérsékletet és páratartalmat, viszont ilyen konstrukciót nem tudtam összeállítani, illetve közelítő eredményekhez az ezektől való eltekintés is megfelelőnek bizonyult.

A szenzorok korrelációinak alakulása ezen mérések és a mérési fájlok szelektálása után (a nagyon eltérő, zajos eredmények kiszűrése végett) a 34. ábrán látható:



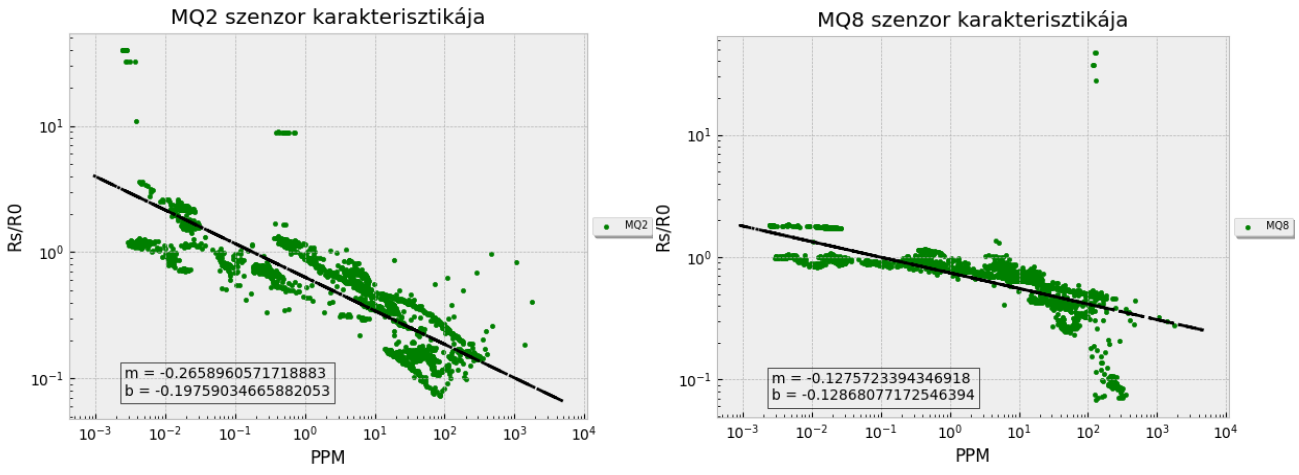
34. ábra - Korrelációs együtthatók alakulása az új mérések után

Itt már látható, hogy az MQ2 és MQ8-as szenzorok 0.8-as r érték felett maradnak nagyjából végig, így azokkal érdemes tovább foglalkozni és karakterisztikát készíteni hozzájuk. Az együtthatók összes mérésre vett átlagát is megkaphatjuk a `printAVGCCVal()` függvénnyel:

```
MQ2 szenzor átlagos r értéke: 0.8516
MQ3 szenzor átlagos r értéke: 0.1468
MQ4 szenzor átlagos r értéke: 0.2118
MQ5 szenzor átlagos r értéke: 0.0738
MQ6 szenzor átlagos r értéke: 0.6273
MQ7 szenzor átlagos r értéke: 0.4890
MQ8 szenzor átlagos r értéke: 0.8274
MQ9 szenzor átlagos r értéke: 0.3010
MQ135 szenzor átlagos r értéke: 0.0000
MQ137 szenzor átlagos r értéke: 1.0000
```

35. ábra - Átlagos r értékek

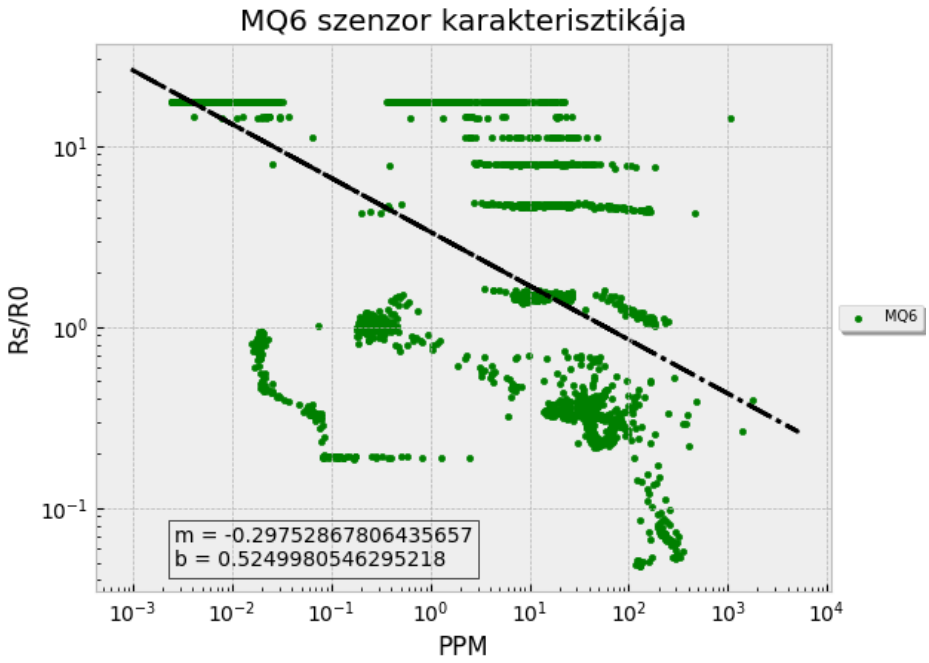
Ezen is látható, hogy az MQ2 és MQ8 szenzorok kerültek ki „győztesként” 0.8-as r érték feletti eredményekkel, így valószínűleg nekik közelítheti meg leginkább a karakterisztikája az MQ137 karakterisztikáját. Karakterisztikáikat a 36. ábrán láthatjuk:



36. ábra - MQ2 és MQ8 szenzorok nh3-Rs/R0 karakterisztikái

4.3 Eredmények kiértékelése

Ahogy az új karakterisztikákon is láthatjuk, a vártaknak megfelelnek az eredmények, szépen látszik, ahogy az illesztett karakterisztikák követik a szenzor ellenállásának változását. A többi szenzor karakterisztikái sajnos ennél jóval zajosabbak és véletlenszerűbbek, így azokra a program által illesztett karakterisztika nem lesz megfelelő. Példaként a 37.ábrán látható az MQ6-os karakterisztikája:

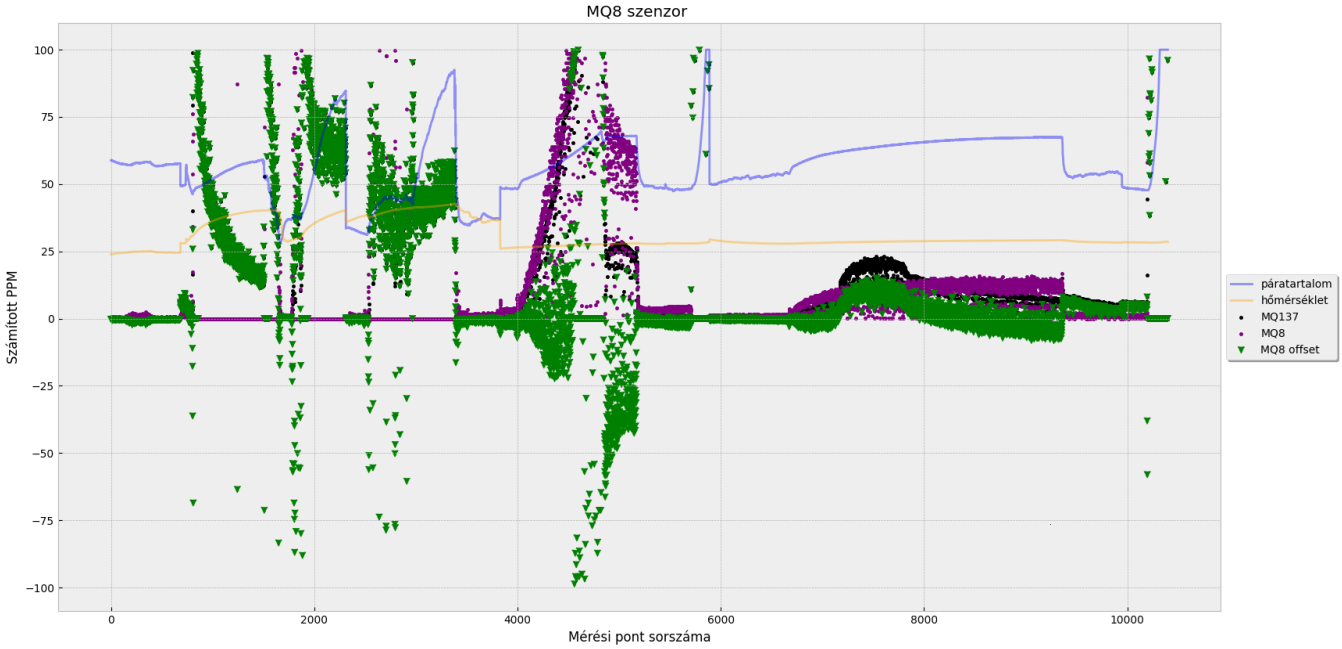


37. ábra - MQ6 szenzor nem megfelelő karakterisztikája

Látható, hogy talán még több, pontosabb méréssel jobban közelíthettünk volna egy karakterisztikához, mivel vannak egymáshoz közeli helyeken ponthalmazok, viszont az eredmények így is elég zajosak.

Az eredmények kiértékeléséhez a measPrototype.py kódját használtam, ami megmutatta, hogy az MQ2 és MQ8 karakterisztikái alapján mért ppm értékek mennyire térnek el az MQ137 szenzor által mérttől, ahogy a következő két ábra mutatja. Ahol a fekete pontok (MQ137 által számolt ppm) nem látszanak (3000.-6000. mérési pontok között) ott 100ppm fölött vannak az értékek, így azokat levágtam, hogy az ábrán jobban meg tudjuk figyelni a 0-100ppm-es tartományt.

38. ábra – MQ2 értékeinek eltérése (piros - MQ2, fekete – MQ137)



39. ábra - MQ8 értékeinek eltérése (lila – MQ8, fekete – MQ137)

Ahol nagyobb eltérések (kiugró zöld pontok) láthatóak, azokon a méréseken jellemzően nagyobb páratartalmat és hőmérsékletet mértem, ahol viszont ~25°C és ~60%-os páratartalom van, ott működőképes a szenzorok karakterisztikáinak számítása. Ezek alapján elmondható, hogy a tesztkörnyezet képes detektálni az ammóniát konkrét ammónia szenzor nélkül is, azonban a pontos ppm értékekhez további mérések és a szenzorok hőmérséklet és páratartalom függését is le kell mérni. Normál körülmények között azonban kisebb hibával ppm is számítható.

5 Összegzés és fejlesztési lehetőségek

A szakdolgozatomban a fő célom az volt, hogy megvizsgáljam a piacon fellelhető olcsóbb fém-oxid félvezető szenzorok ammóniára adott jelét, illetve ezek alapján következtessek arra, hogy alkalmasak-e ezen szenzorok ammónia detektálásra és ezáltal egyfajta elektronikus orrként precíziós mezőgazdaságban való felhasználásra.

A kutatásomat az előző fejezetben bemutatott eredmények alapján sikeresnek tekintem, mivel sikerült korrelációt találni az ammónia detektálásra alkalmas MQ137 szenzor és az olcsóbb MQ2 és MQ8 szenzorok válasza között, amely után bemutattam, hogy bizonyos környezeti megkötések mellett kisebb hibával tudunk ammónia koncentrációt is számolni az adott szenzorokkal.

Az irodalomkutatásokat újra áttekintve, ahol az MQ9, MQ135 és MQ5 szenzorok mutattak legnagyobb érzékenységet az ammóniára, a saját méréseimben ezek a szenzorok nem mutattak igazán hasonlóságot. Ezt részben annak tudom be, hogy ahogy már említettem, az ESP ADC-je sajnos nem tökéletes, ezen szenzorok kimenete pedig jellemzően pár tized volt feszültségnek adódott, amely tartományra az ADC érzéketlen. Ez alapján egy jó fejlesztési lehetőség az ADC kiszervezése és mondjuk egy 16 bites, dedikált ADC-vel elvégezni újra a méréseket, amivel pontosabb eredményeink lehetnek.

Mivel a szenzorok a 0-100ppm-es tartományban nagyjából azonos arányokkal változnak, együtt mozognak, úgy gondolom alkalmas lehet a rendszer például egy olyan rendszerben integrálva vezérlési feladatot ellátni, ahol 50ppm-es határnál riasztást kell adni, illetve ekkor bekapcsol egy szellőztetést. Ehhez már ebben a rendszerben is rendelkezésre áll a wifi és MQTT kommunikáció által egy vezeték nélküli jelzési, riasztási lehetőség.

Sajnos a beszerzett áramkörre szerelt szenzorok RL ellenállás értékei nem voltak minden esetben indokoltak (néhol túl kicsik voltak) így a jövőben célszerű lenne saját áramkört építeni, amire felkerülhetnek a szenzorok, így is növelve a mérések pontosságát, illetve a most használt ESP is felkerülhetne rá, amivel kompaktabb lenne a rendszer. A tesztelés során a mérések folyamatát is célszerű lenne strukturáltabban, azonos forgatókönyvet követve végezni, hogy a mérésekből jobban lehessen a szenzorok pontosságára is következtetni.

Összességében elégedett vagyok az elért eredményekkel, viszont az belátható, hogy a tökéletes eredményekhez több mérés és pontosabb eszközök szükségesek.

Irodalomjegyzék

[1] „Negyedik ipari forradalom,” [Online]. Available: https://hu.wikipedia.org/wiki/Negyedik_ipari_forradalom. [Hozzáférés dátuma: 20. 10. 2023.].

[2] „Csökken a mezőgazdasággal foglalkozók száma,” [Online]. Available: https://agraragazat.hu/hir/ksh-csokkent-a-mezogazdasaggal-foglalkozok-szama-az-elmult-harom-evben/. [Hozzáférés dátuma: 13. 11. 2023.].

[3] „Elektronikus orr,” [Online]. Available: https://hu.wikipedia.org/wiki/Elektronikus_orr. [Hozzáférés dátuma: 20. 10. 2023.].

[4] „Incorporating Smart Sensing Technologies into the Poultry Industry,” [Online]. Available: https://core.ac.uk/download/pdf/35316433.pdf. [Hozzáférés dátuma: 13. 03. 2023.].

[5] „11/2019. (IV. 1.) AM rendelet a baromfi ágazatban igénybe vehető állatjóléti támogatások feltételeiről,” [Online]. Available: https://net.jogtar.hu/jogszabaly?docid=a1900011.am. [Hozzáférés dátuma: 10. 03. 2023].

[6] „Klímavédelmi szempontrendszer integrálása a mezőgazdasági szaktanácsadásba III.,” [Online]. Available: https://www.nak.hu/kiadvanyok/kiadvanyok/3705-az-allattenyesztes-es-a-klimavaltozas/file. [Hozzáférés dátuma: 10. 03. 2023].

[7] „Gázérzékelő SMART-R-NH3,” [Online]. Available: https://www.vagyonvill.hu/Gazerzekelo-SMART-R-NH3-4-20mA-0-100ppm. [Hozzáférés dátuma: 02. 10. 2023.].

[8] „Advance in electronic nose technology developed for the detection and discrimination of ethanol, ammonia, and hydrogen sulfide gases,” [Online]. Available: https://ieeexplore.ieee.org/document/9789636. [Hozzáférés dátuma: 13. 03. 2023.].

[9] M. Marcell, *mérési eredmények - Kamerakép-alapú súlybecslés a precíziós baromfitartásban - Szabó Sándor előadásában,* 2023..

[10] „Korreláció,” [Online]. Available: https://hu.wikipedia.org/wiki/Korreláció. [Hozzáférés dátuma: 02. 10. 2023.].

[11] Wikipédia, „Mikrovezérlők,” [Online]. Available: https://hu.wikipedia.org/wiki/Mikrovezérlő. [Hozzáférés dátuma: 19. 11. 2023.].

[12] „ESP32 ADC karakterisztikája,” [Online]. Available: https://lastminuteengineers.com/esp32-basics-adc/. [Hozzáférés dátuma: 02. 10. 2023.].

[13] „ESP32 pinouts,” [Online]. Available: https://www.alibaba.com/product-detail/LILYGO-T8-V1-8-ESP32-WROVER_1600649715077. [Hozzáférés dátuma: 12. 10. 2023.].

[14] „How MQ2 Gas/Smoke Sensor Works?,” [Online]. Available: https://lastminuteengineers.com/mq2-gas-senser-arduino-tutorial/. [Hozzáférés dátuma: 19. 11. 2023.].

[15] „MQ szenzor üzemi hőmérséklete: An overview on room-temperature chemiresistor gas sensors based on 2D materials: Research status and challenge,” [Online]. Available: https://www.sciencedirect.com/science/article/pii/S135983682200751X. [Hozzáférés dátuma: 11. 10. 2023.].

[16] „How Does MQ-2 Flammable Gas and Smoke Sensor Work with Arduino? (MQ áramköri rajz),” [Online]. Available: https://circuitdigest.com/microcontroller-projects/interfacing-mq2-gas-sensor-with-arduino. [Hozzáférés dátuma: 20. 11. 2023.].

[17] „MQ137 datasheet,” [Online]. Available: https://cdn.sparkfun.com/assets/b/6/7/5/8/MQ137_datasheet.pdf. [Hozzáférés dátuma: 12. 11. 2023.].

[18] „Measuring PPM from MQ Gas Sensors using Arduino (MQ-137 Ammonia),” [Online]. Available: https://duino4projects.com/measuring-ppm-mq-gas-sensors-using-arduino-mq-137-ammonia/. [Hozzáférés dátuma: 02. 10. 2023.].

[19] „What Is MQTT and Why Is It the Best Protocol for IoT?,” [Online]. Available: https://www.emqx.com/en/blog/what-is-the-mqtt-protocol. [Hozzáférés dátuma: 12. 10. 2023.].

[20] „What is MQTT?,” [Online]. Available: Forrás: https://www.paessler.com/it-explained/mqtt. [Hozzáférés dátuma: 12. 10. 2023.].

[21] „Sensors, interfaces and bus systems (SENIN, BUSSY) MQTT,” [Online]. Available: https://www.weigu.lu/tutorials/sensors2bus/06_mqtt/index.html. [Hozzáférés dátuma: 12. 10. 2023.].

[22] „How to Install Mosquitto MQTT Broker on Windows,” [Online]. Available: https://cedalo.com/blog/how-to-install-mosquitto-mqtt-broker-on-windows/. [Hozzáférés dátuma: 11. 10. 2023.].

[23] „Node-RED,” [Online]. Available: https://nodered.org/about/. [Hozzáférés dátuma: 12. 10. 2023.].

[24] „What are the Key Pros and Cons of the Arduino Programming Language?,” [Online]. Available: https://emeritus.org/blog/coding-arduino-programming-language/. [Hozzáférés dátuma: 30. 11. 2023.].

[25] „What Is Python Used For?,” [Online]. Available: https://www.coursera.org/articles/what-is-python-used-for-a-beginners-guide-to-using-python. [Hozzáférés dátuma: 30. 11. 2023.].

[26] „PIP install,” [Online]. Available: https://pypi.org/project/pip/. [Hozzáférés dátuma: 30. 11. 2023.].

Függelék

Az alábbi linken található az összes felhasznált és bemutatott kód és mérési eredmény egy GitHub repo-ban:

https://github.com/szabel001/szakdolgozat