SZAKDOLGOZAT FELADAT

Berta Máté (HPD5LB)

Villamosmérnök hallgató részére

Talaj nedvességtartalom mérés IoT eszközzel

Manapság népszerű kutatási, fejlesztési terület az IoT. Ezen belül is nagy népszerűségnek örvendenek a különféle otthon automatizálási megoldások. A hallgató feladata is ebbe a témakörbe illeszkedik egy automatizált kerti öntözőrendszerhez tartozó mérőegység megtervezésével. Az egységgel szemben további elvárás, hogy illeszkedjen a hallgató önálló laboratórium feladatában tervezett rendszerbe, a méréseket adott időközönként és módon végezze, a gyűjtött adatokat a rendszer központi egysége felé továbbítsa.

A feladat első része a mérőegység rendszertervének elkészítése, figyelembe véve a már meglévő rendszerbe történő illesztés támasztotta követelményeket. Ennek része a megfelelő perifériák (például rádiós kommunikációhoz, beavatkozó szervhez) kiválasztása, a mérési adatok megfelelő formátumra hozása, a mérőegység villamosenergia igényének figyelembevétele (becslés akkumulátoros üzemidőre), költségszámítás és az elosztott szenzorhálózatba építés realitásának mérlegelése.

A hallgató feladatának – a fentieken felül – a következőkre kell kiterjednie:

- Szakirodalmi kutatás a talaj nedvességtartalom kapacitív elven történő méréséről
- Kapacitív szenzor kiválasztása
- A választott szenzorhoz különböző vizsgáló jeleket létrehozó áramkörök illesztése
- Kapcsolási rajz készítés
- NYÁK terv készítés
- Az elkészült áramkör bemérése, élesztése
- A mérőegység feldolgozó egységére beágyazott szoftver készítése
- Kalibrációs mérés a szakirodalomban is megtalálható módszerek segítségével
- A központi egység szoftverében módosítások végzése a mért adatok rendszerezett megtekinthetőségéne elérése érdekében

Tanszéki konzulens: Naszály Gábor, mestertanár

Budapest, 2020.12.08.

.....

Dr. Dabóczi Tamás
tanszékvezető
egyetemi tanár, DSc

Budapesti Műszaki és Gazdaságtudományi Egyetem
Villamosmérnöki és Informatikai Kar
Méréstechnika és Információs Rendszerek Tanszék

1117 Budapest, Magyar Tudósok krt. 2. I. ép. I.E.444.
Telefon: 463-2057, Fax: 463-4112
http://www.mit.bme.hu • e-mail: mitadm@mit.bme.hu

Budapesti Műszaki és Gazdaságtudományi Egyetem
Villamosmérnöki és Informatikai Kar
Méréstechnika és Információs Rendszerek Tanszék

Berta Máté

TALAJ NEDVESSÉGTARTALOM MÉRÉS IOT ESZKÖZZEL

KONZULENS

Naszály Gábor

BUDAPEST, 2020

Tartalomjegyzék

HALLGATÓI NYILATKOZAT

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

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

Kelt: Budapest, 2020. 12. 08.

...…….
Berta Máté

Összefoglaló

Szakdolgozatom egy talaj nedvességtartalom kapacitív mérésére alapuló mérő és beavatkozó egység elkészítésének folyamatát mutatja be. A mérőegység alacsonyfogyasztású és egy IoT rendszerbe illeszkedik, mely központi egységének felépítése és szoftvere is bemutatásra kerül.

Az első fejezetben bemutatom az alapötletet és a rendszertervet, ezt követően a második fejezetben a talaj nedvességtartalom definíciója és annak mérési módszereinek bemutatása után, egy kísérletet jegyzőkönyvezek a mérőjel spektrumának hatásáról a kapacitív szenzorok karakterisztikájára.

A harmadik fejezetben hardveres tervezés egyes lépései kerülnek részletezésre és az elkészült prototípus élesztése utáni tapasztalatokat összegzem.

A negyedik fejezetben a mérő- és központiegység szoftverét tárgyalom. Bemutatásra kerül, hogy ESP32 processzorra hogyan készíthetünk beágyazott operációs rendszeren (FreeRTOS) futó alacsony fogyasztású működést biztosító firmwaret. A központi egység szoftverének leírása betekintést nyújt IoT alkalmazásokban gyors fejlesztést biztosító célszoftverek használatába, részletesen bemutatja Node-RED szoftver alapvető használatát.

Összegzésben a prototípussal szerzett tapasztalatok alapján, összevetem a mérőegység előnyeit és hátrányait, majd a továbbfejlesztés lehetőségeit tárgyalom és irányát is kitűzöm.

Abstract

My thesis presents the process of making a measuring and actuator unit based on the capacitive measurement of soil moisture content. The measuring unit is low-power and fits into an IoT system, the structure and software of which are also presented.

In the first chapter I present the basic idea and the system design, then in the second chapter after the definition of soil moisture content and its measurement methods, I record an experiment on the effect of the measurement signal spectrum on the characteristics of capacitive sensors.

In the third chapter, the individual steps of hardware design are detailed and I summarize the experience after the completion of the completed prototype.

In the fourth chapter, I discuss the software of the measuring and central unit. It shows how to create low-power firmware running on an embedded operating system (FreeRTOS) for an ESP32 processor. The description of the central unit's software provides insight into the use of applications for rapid development in IoT projects, and details the basic use of Node-RED software.

In the last chapter, based on the experience gained with the prototype, I compare the advantages and disadvantages of the measuring unit, then I discuss the possibilities of further development and set its development's direction.

1 Bevezetés

1.1 Otthon automatizálás kertkapcsolatos házakban

Az otthonainkban egyre jobban betörő IoT eszközök megváltoztatják elvárásainkat háztartási eszközeinkkel szemben. Az „smart” jelzővel ellátott használati tárgyaink száma egyre nő, melyek gyakran, egyes kényelmi funkciókon túl, távoli elérést tesznek lehetővé. Olyan információkat gyűjthetünk otthonunkról, mint például napelem celláink teljesítménye, házunk áramfelvétele, háztartási eszközeink fogyasztása vagy akár bejárati ajtónkról élő kameraképet nézhetünk távolról.

Az otthon automatizálási feladatok ellátásával szemben állított jellemző műszaki követelmények közé tartozik az alacsony fogyasztás, integrálhatóság valamilyen IoT rendszerbe, könnyű kezelőfelület és relatív alacsony ár.

Hazánkban lakóövezet szerint a házak 62%-a családi ház, zártkerti vagy külterületi ingatlan. [1] Az ilyen típusú ingatlanok gyakran rendelkeznek kertkapcsolattal, ahol konyhakert vagy gyümölcsös található vagy telepíthető. Az ilyen ingatlanok esetén az otthon automatizálást kiterjeszthető a kerti feladatok automatizálására, melyek közül az egyik legkevesebb infrastrukturális változtatást megkövetelő az öntözőrendszer „okosítása”.

Sok kert rendelkezik kiépített öntözőrendszerrel, amely egy vagy több csap megnyitásával lehetővé teszi az öntözést. Ezek a rendszerek gyakran időzítőkkel vannak ellátva, mely bizonyos szintű automatizált megoldást nyújt, de ezek a rendszerek nem tudják figyelembe venni az időjárási körülményeket, gyakran sok emberi beavatkozást is igényelnek, illetve állapotuk nem ellenőrizhető távolról.

A kiépített öntözőrendszerek viszont lehetőséget adnak a könnyű automatizálásra, hiszen egy mágnesszelepes csap segítségével villamos jelekkel irányítható az öntözés.

1.2 Öntözőrendszer bemutatása

Jelen szakdolgozat témája egy öntözőrendszerbe építhető mérő- és beavatkozó egység. A rendszer követelményeit és a mérőegység alapelvének működőképességét önálló laboratórium feladatom definiálja és támasztja alá, az erről készült dokumentum elérhető nyilvánosan. [2]

1-1. ábra SWAN logó

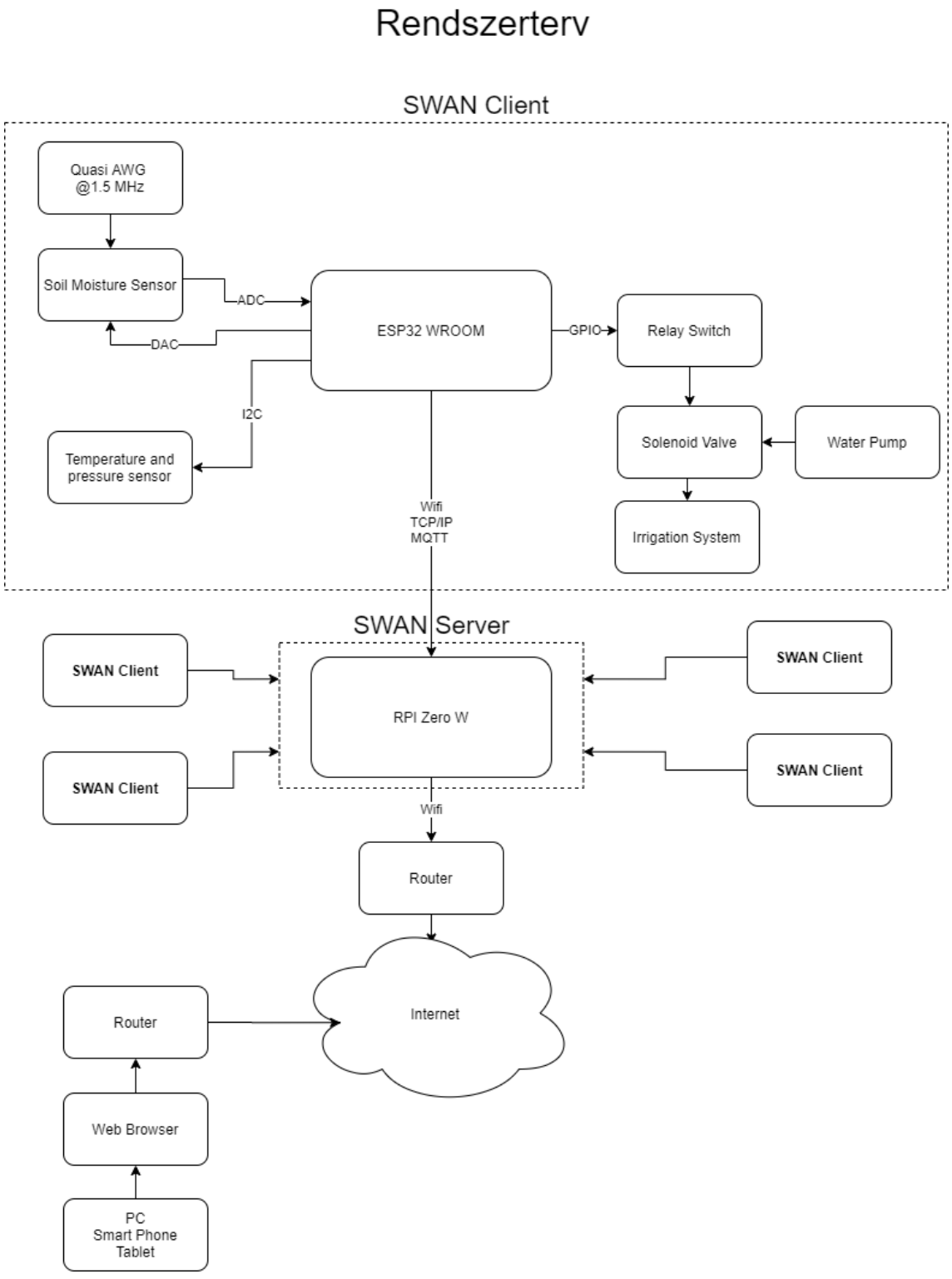
Az öntözőrendszer a Smart Watering Automation System (S.W.A.N.) nevet kapta. A mérő- és beavatkozó egységre SWAN kliensként, a rendszerhez tartozó központi egységre SWAN szerverként hivatkozok a továbbiakban.

Minden kliens a talaj nedvességtartalmát, a levegő hőmérsékletét és nyomását méri. A mért adatok felhasználásával avatkozik be minden egység, amely ellátott az ehhez szükséges hardver elemekkel. Minden mért adatot batchenként közlik a kliensek a szerverrel, mely utasítást is adhat a klienseknek, amikor azok bejelentkeznek hozzá.

A szerver és a kliensek feltételezik, hogy saját rádiós perifériáiknak hatótávolságán belül vezeték nélküli kapcsolaton (WIFI) keresztül hozzáférésük van a helyi hálózathoz, melyen kommunikálnak egymással.

A kliensek akkumulátoros üzemre optimalizáltak, viszont hálózatról (3,5V-6V DC) is meghajthatóak. Az akkumulátor töltése DC tápon keresztül történik, a szükséges töltések közötti idő minél hosszabbra növelése szempontja mind a hardveres, mind a szoftveres tervezésnek.

A SWAN kliens és szerver az alábbi rendszerterv szerint épül fel és csatlakozik egymáshoz.



1-2. ábra S.W.A.N rendszerterv

2 Kapacitív talaj nedvességtartalom mérés

2.1 Fizikai jelenségek

2.1.1 Definíció és validációs módszer

A talaj nedvességtartalma általában százalékban kifejezett mennyiség, amely megmutatja, hogy 1 gramm talaj hány gramm vizet tartalmaz. Mértéke tehát $\left[\frac{g}{g} \cdot 100\right] = [\%]$.

Egyszerű mérési módja, ha ismert tömegű, adott nedvességtartalmú talajt kiszárítunk és tömeg különbségét az elpárolgott víz tömegének vesszük. Ekkor a következőképpen számíthatunk:

$$S_{[\%]} = \frac{m_{H_2O}}{m_{talaj}} = \frac{m_{H_2O}}{m_{minta} - m_{H_2O}} = \frac{m_{minta} - m_{minta}'}{m_{minta}'}$$

Másik mérési módszer lehet, hogy ismert tömegű száraz, zérusnak tekinthető vizet tartalmazó talajhoz ismert tömegű vizet adunk. Ekkor különbség képzés nélkül megkaphatjuk ugyan azt az arányszámot:

$$S_{[\%]} = \frac{m_{H_2O}}{m_{talaj}}$$

Mindkét módszer tömegmérésre visszavezetett, amely jó pontosságú a jelenleg ismert technológiával. Viszont ezek a mérési módszerek nem alkalmazhatóak gyakorlati helyzetekben, hiszen a vizsgálandó talaj általában nem távolítható el és nem vonható ki belőle minden nedvesség. Emellett a talajba jutó vízmennyiség tömege gyakran nem ismert.

Az előzőekben ismertetett módszerek viszont alkalmasak más mérési módszerek ellenőrzésére, azoknak a validációjára. Hiszen más fizikai jelenséget kihasználó mérési módszerrel mért értékekhez, ezekkel a módszerekkel lehetőség nyílik konkrét számértékeket rendelni.

Megjegyzendő, hogy a talaj nincs definiálva jelen dokumentumban, annak ellenére, hogy a talaj nedvességtartalom definícióját nem, de a továbbiakban bemutatott mérési módszert befolyásolja a talaj milyensége. Ennek vizsgálatára nem került sor, az elvégzett mérések kvarchomokban történtek, amely megfelelő tisztaságú és mennyiségű könnyen beszerezhető, illetve jól kiszárítható.

2.1.2 Kapacitív szenzor mérési elve

A talaj nedvességtartalmához valamilyen fizikai mennyiséget kell rendelnünk, annak érdekében, hogy mérni tudjuk a 2.1.1 fejezetben bemutatott módszerek alkalmazása nélkül.

Cél, hogy a mért fizikai mennyiséget feszültségmérésre vezessük vissza, mivel feszültséget kellő pontossággal és mikroprocesszorral tudunk mérni. Figyelembe vehetjük, hogy a talaj elektromos térben való viselkedése jellemezhető a komplex relatív permittivitásával (ε_r^*). [3]

A következő egyenlet szerint változik ε_r^*:

$$\varepsilon_r^* = \varepsilon_r' - j \cdot \varepsilon_r'' = \varepsilon_r' - j\left(\varepsilon_{relax}'' + \frac{\sigma_{dc}}{2\pi f \varepsilon_0}\right)$$

Ahol σ_{dc} a talaj vezetőképessége, ε_{relax}'' molekulák nyugalmi hozzájárulása, ε_r' egy külső tér által az anyagban tárolt energiáját fejezi ki és j az imaginárius egység. A komplex permittivitás imaginárius része jellemzi, hogy az anyag mennyire „veszteséges az elektromos térre nézve. A veszteség mértéke függ a frekvenciától, a nedvességtartalomtól, illetve az anyag só és ion tartalmától. Tehát a közeg nedvesség tartalmával összefüggésben lévő fizikai mennyiség a komplex relatív permittivitás.

Amennyiben G_0 ismert geometriájú szenzor segítségével villamos teret juttatunk a talajba a következő összefüggés szerint definiálhatjuk a kapacitást:

$$C = \varepsilon_r^* \varepsilon_0 G_0$$

Az így bevezetett kapacitás lineárisan függ a komplex relatív permittivitástól, ráadásul villamosan mérhető mennyiség. Kapacitás mérésre több ismert módszer is létezik, melyek többsége feszültség mérésre vezeti vissza a problémát.

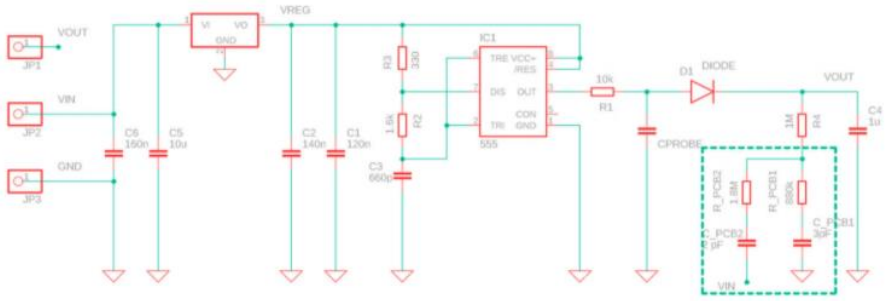
2.2 Szenzor jellemzése és kalibrációs mérés

2.2.1 Választott szenzor és mérési módszere

A választott szenzor egy márkajelzés nélküli különböző internetes áruházakban olcsón kapható és hobbi IoT alkalmazásokban gyakran használt mérőegység. A szenzor gyártója DFROBOT és SKU:SEN0193 néven érhető el. Specifikációja csak a tápfeszültség határait adja meg, működését és kapcsolási rajzát a hivatkozott cikk alapján ismertem meg. [4]

A szenzor egy vizsgálójelet terhel a talaj által reprezentált kapacitással, mely hatására a vizsgálójel csúcsértéke változik. A csúcsérték pedig (csúcsérték detektoron keresztül) követlenül ADC-vel mérhető egyenfeszültség.

A szenzor vizsgáló jelként 1.5 MHz frekvenciájú, egyenkomponenssel rendelkező négyszögjelet alkalmaz, amely előállítására bistabil multivibrátort alkalmaz 555 időzítő IC és passzív komponensek felhasználásával.



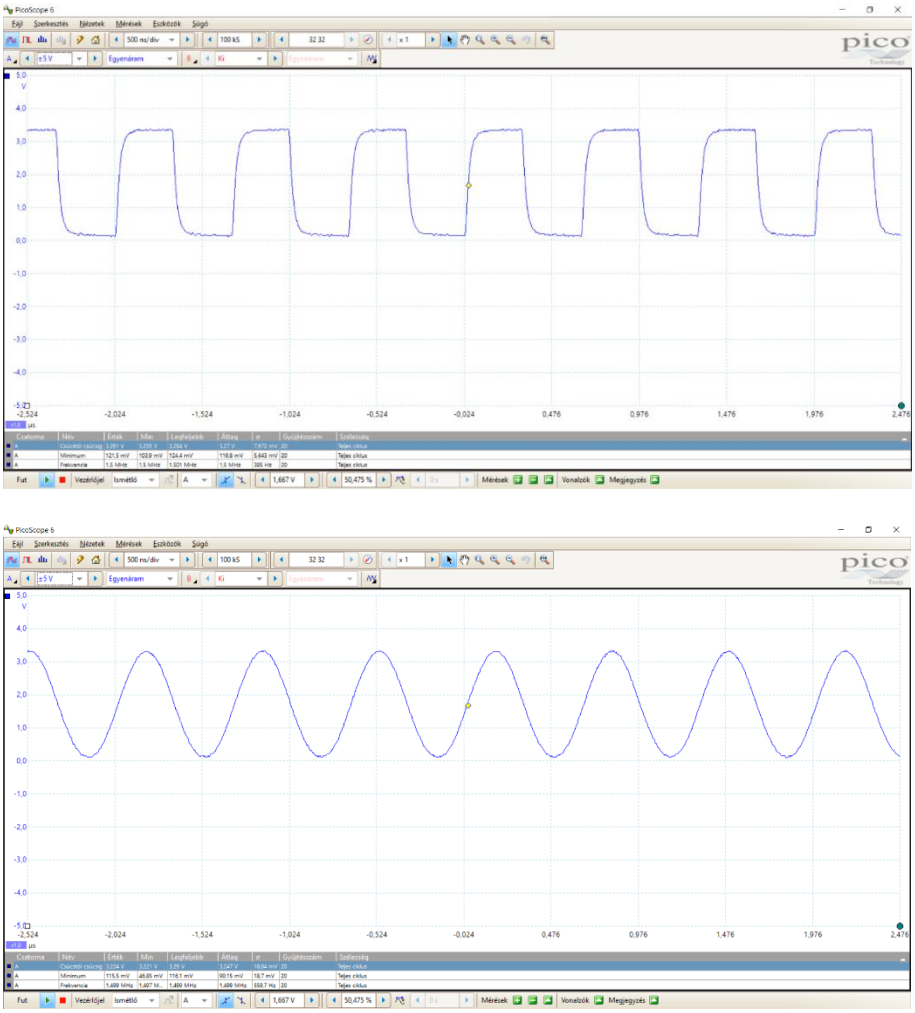
2-1. ábra Capacitive Soil Moisture Sensor v1.2 kapcsolási rajza

2.2.2 Mérőjel alakjának befolyásoló hatásának vizsgálata

A kiválasztott szenzor vizsgáló jele négyszögjel, melyről Fourier-sorfejtés segítségével megállapítható, hogy felharmonikusokat tartalmaz. A kapacitás és a komplex relatív permittivitás (közvetlenül és közvetetten) függ a frekvenciától. Vizsgálatom célja, hogy megfigyeljem milyen hatása van a mérőjel alakjának, ezáltal spektrumának, a mért feszültségre.

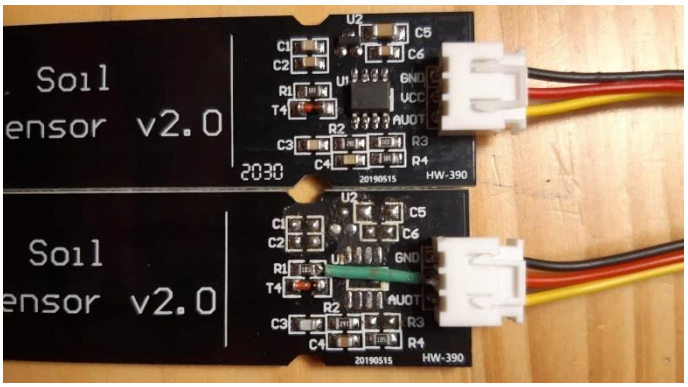
A vizsgálat továbbá lehetőséget nyújt, hogy a mért feszültségek és a talaj nedvességtartalma közötti kapcsolatot karakterizáljam, kalibrációs mérésként felhasználjam, erre a 2.1.1 fejezetben említett második módszert alkalmaztam.

A vizsgálat a szenzor által előállított négyszögjelnél meredekebb élekkel rendelkező négyszögjellel és szinusz jellel történt. Minden jel 0 és 3.3 V közötti, egyenfeszültségű komponenst is tartalmazó és 1.5 MHz frekvenciájú. A használt mérőjelek a 2-2. ábraán láthatóak.



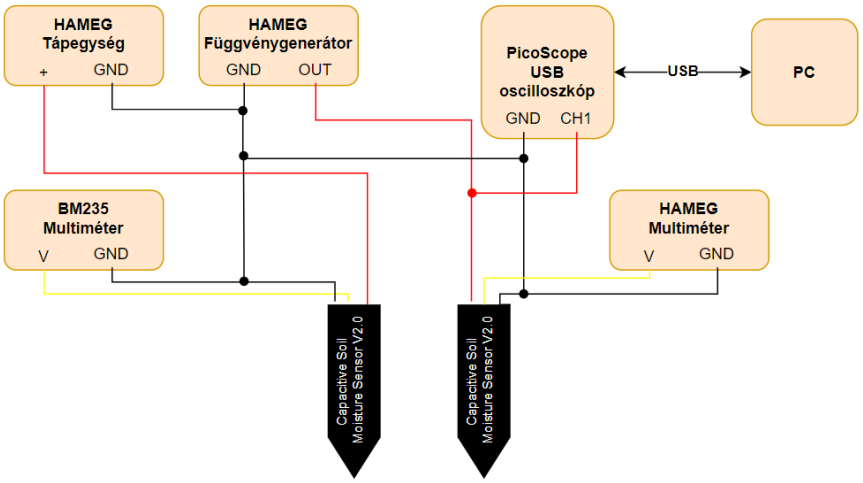
2-2. ábra Függvénygenerátorral előállított mérőjelek

A mérésem két egy napon gyártott szenzor felhasználásával történt, melyek közül az egyiket módosítottam úgy, hogy külső mérőjel csatlakozhasson hozzá. A módosítás során az 555-ös IC és a hozzá tartozó áramköri elemeket leforrasztottam és az eredeti kapcsolás R1 ellenállását közvetlenül a V_{in} bemenettel kapcsoltam össze. A két szenzor egymás mellett a 2-3. ábraán látható.



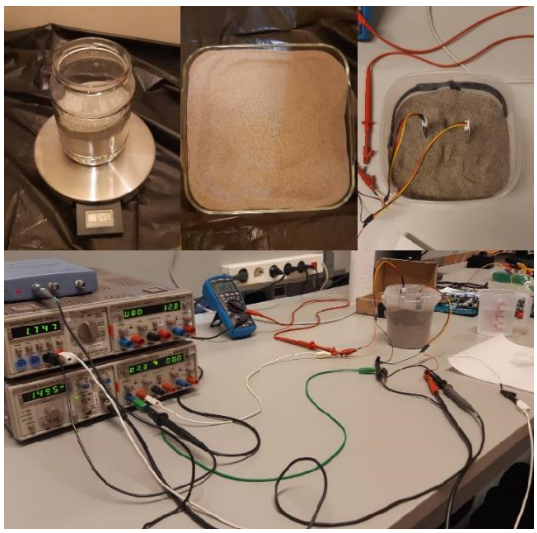
2-3. ábra Eredeti (felső) és módosított (alsó) szenzor

A kalibrációs mérések során a nem módosított és a módosított szenzor feszültségértékeit jegyzem fel és hasonlítom össze táblázatos és grafikus formában. A mérési elrendezés sematikus vázlata a 2-4. ábraán látható.



2-4. ábra Kalibrációs mérés

A mérés előkészületeként 5000 g szárított kvarchomokot egy üvegedényben 1 órán keresztül 110 ºC hőmérsékleten tovább szárítottam. Az így kapott homokot tekintem zérus nedvességtartalmú talajnak, számításaim során 0 g víztartalmat tulajdonítok neki. A 5000 g homokot 1:2:2 arányban felosztottam, egy teszt mérés és két éles mérés számára. A tesztmérés során megfigyeltem, hogy a homok milyen módon veszi fel a vizet, mennyire számít a szenzorok mozgatása és talajban lévő mélysége a mérés során. A 2-5. ábra a mérés előkészítése és a mérés elvégzése közben készített képeket tartalmazza.

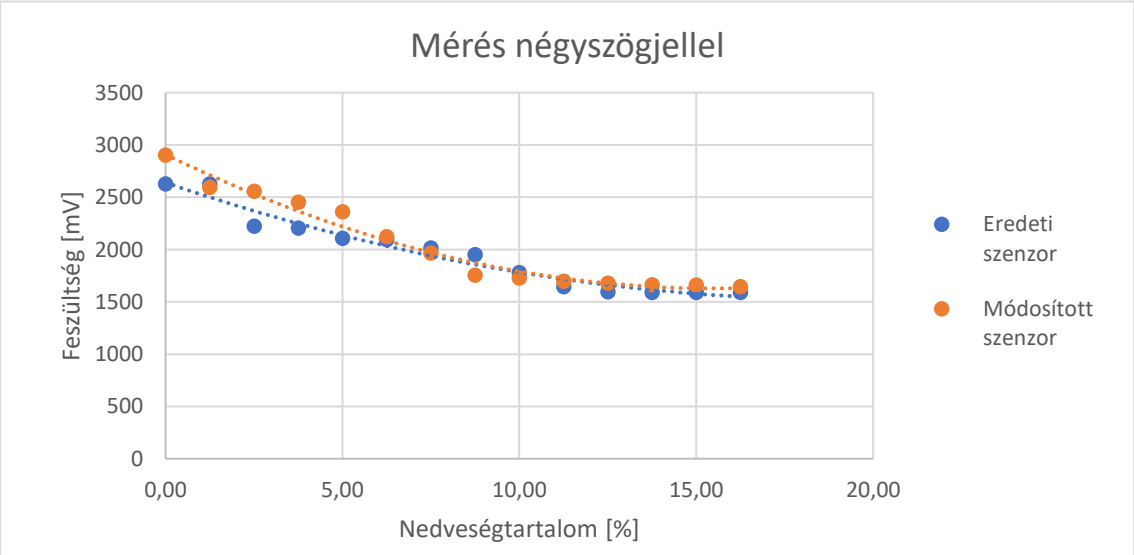


2-5. ábra Kalibrációs mérés fázisai

Az éles mérések során ügyeltem, hogy a szenzorok szimmetrikusan helyezkedjenek el a mérőedényben, teljes felületüket talaj fedje és a hozzáadott víz azonos mértékben jusson el hozzájuk. Víz hozzáadása előtt feljegyeztem a szenzorok által mutatott feszültséget, majd 25ml-es lépésenként adtam hozzá szobahőmérsékletű csapvizet. A víz hozzáadása után, megvártam, amíg a multiméterek legalább két tizedes jegyig állandó értéket mutatnak és feljegyeztem a feszültséget egy táblázatba. A víz hozzáadását addig folytattam, amíg legalább öt, azonosnak tekinthető feszültségértéket nem olvastam le. A két éles mérés között a szenzorokat nedvszívó törlőkendővel tisztítottam meg, annak érdekében, hogy a felületen ne maradjon nedves homok.

Az első mérés négyszögjellel történt és a következő táblázatban rögzített adatokat kaptam, melyeket grafikonon is ábrázoltam.

Nedvességtartalom [%]	Feszültség ("kontroll" szenzor) [mV]	Feszültség (kísérleti szenzor) [mV]
0,00	2214	2901
1,25	1842	2592
2,50	1801	2556
3,75	1645	2451
5,00	1474	2360
6,25	1151	2121
7,50	1064	1965
8,75	979	1755
10,00	961	1728
11,25	945	1697
12,50	946	1678
13,75	947	1662
15,00	945	1659
16,25	939	1644



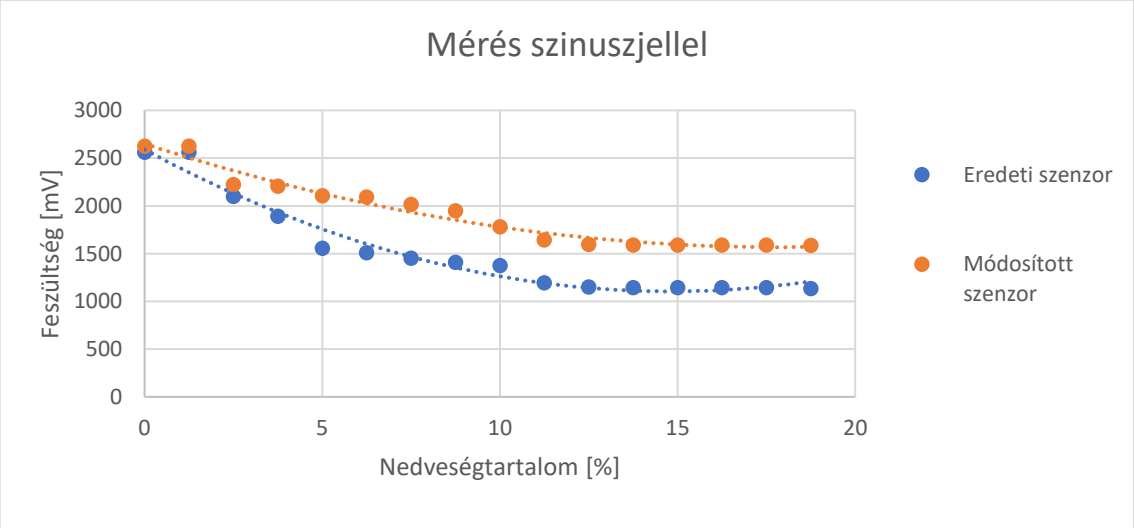
2-6. ábra Kalibrációs mérés (négyszögjel)

A 2-6. ábraán látható trendvonalak (szaggatott) másodfokú polinomiális görbék. Jól látható, hogy 5% felett a két görbe jól illeszkedik egymásra.

Az eredeti szenzor adatsorában található 1,25%-hoz tartozó kiugró érték oka, hogy a kézi adagolás miatt nem egyenletesen oszlott el a folyadék a mérés ezen lépésénél.

Megállapítható, hogy a meredekebb éleket tartalmazó vizsgálójel szélesebb feszültségtartományt rendel a 0-16 % nedvességtartalomhoz, viszont a görbe jellegét tekintve nincs szignifikáns változás.

Nedvességtartalom [%]	Feszültség ("kontroll" szenzor) [mV]	Feszültség (kísérleti szenzor) [mV]
0	2563	2627
1,25	2562	2626
2,5	2100	2223
3,75	1893	2205
5	1555	2105
6,25	1510	2093
7,5	1452	2015
8,75	1410	1950
10	1375	1780
11,25	1195	1643
12,5	1152	1595
13,75	1144	1590
15	1145	1590
16,25	1144	1590
17,5	1143	1589
18,75	1134	1587



2-7. ábra Kalibrációs mérés (szinuszjel)

A 2-7. ábrán látható trendvonalak (szaggatott) másodfokú polinomiális görbék. A módosított szenzorhoz tartozó karakterisztika keskenyebb feszültség tartományt rendel a nedvességtartalomhoz és az első 0-10%-hoz tartozó „lineáris” szakaszának meredeksége is kisebb az eredeti szenzorhoz képest.

$$s_e = \frac{V_{e0} - V_{e9}}{S_{e0} - S_{e9}} = \frac{2563 - 1375}{0 - 10} = -118{,}8 \left[{}^{V}/_{\%} \right]$$

$$s_m = \frac{V_{m0} - V_{m9}}{S_{m0} - S_{m9}} = \frac{2627 - 1780}{0 - 10} = -84{,}7 \left[{}^{V}/_{\%} \right]$$

2.2.3 Összegzés

Az előzőekben ismertetett mérésim arra engednek következtetni, hogy a mérőjel jelalakja, és ezáltal spektruma, ugyan befolyásolja a mérés eredményét, de nem szignifikánsan. A vizsgált négyszög- és szinuszjel közül a négyszögjel adott olyan karakterisztikát, amely jobban kihasználja a rendelkezésre álló feszültség tartományt, ezért használatra ezt javaslom.

A módosított szenzor pontossága hasonló a gyári szenzorhoz viszonyítva és előnye, hogy a mérőjelet előállító egység hozza működésbe, nem szükséges DC feszültséggel táplálni. Emiatt a módosított szenzor használatát javaslom, négyszögjel meghajtással, minden alacsony fogyasztást igénylő alkalmazás esetén.

További tapasztalat volt a mérések során, hogy a szenzor talajjal érintkező felületének nagysága sokkal jobban befolyásolta a mérés eredményét, ezért egy mérőegység elhelyezésénél erre fokozott figyelmet kell fordítani.

Felmerülő kérdés, hogy a kvarchomok, mellyel a karakterizálás történt mennyire reprezentálja jól az átlagos termőföldet, mely más fizikai, kémiai és biológiai tulajdonságokkal bír.

Belátható, hogy a talaj nedvességtartalmának kapacitív elven történő mérése viszonylag olcsó és könnyen megérhető technológiával lehetséges. Ennek ellenére objektíven meghatározott pontossághoz további fejlesztésre és információgyűjtésre van szükség.

3 Hardver tervezés

3.1 Elvi kapcsolási rajz

3.1.1 Mágnesszelep működése elve

A kliensek kimenete egy az öntözőrendszerben meglévő beavatkozó szervhez kapcsolódik, mely működési elvének megértése szükséges a megfelelő vezérlőjel előállításához.

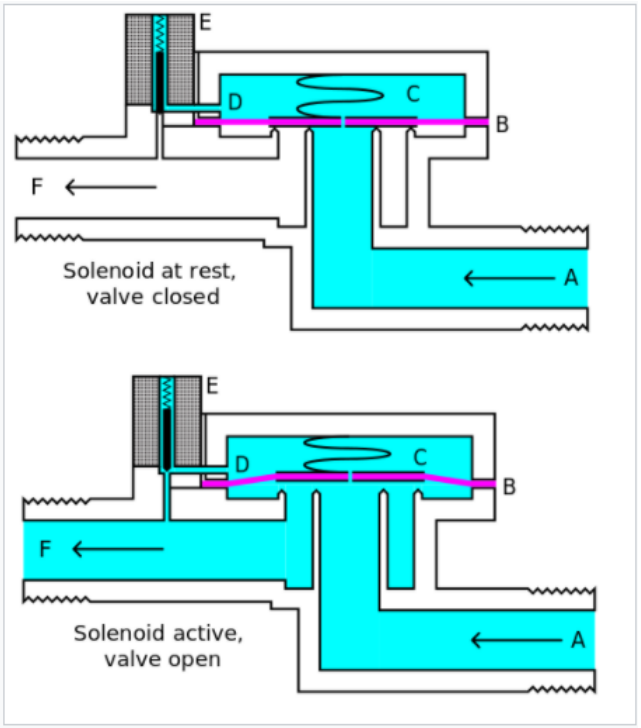
Ez a beavatkozószerv egy mágnesszelep, amely elektromechanikusan működtetett szelep. A mágnesszelepek a folyadékokat szállító rendszerekben leggyakrabban használt vezérlőelemek. Számos alkalmazási területen megtalálhatók. Feladatuk a folyadékok útjának elzárása, felszabadítása, a folyadék adagolása, elosztása vagy keverése. A mágnesszelepek gyorsak és biztonságosak, nagy megbízhatóságúak, hosszú élettartammal rendelkeznek.

A 3-1. ábra egy általános szelep kialakítását mutatja, amely a víz áramlását szabályozza ebben a példában. A legfelső ábrán a szelep zárt állapotban van. A nyomás alatt levő víz A pontba kerül. B egy rugalmas membrán, fölötte pedig egy gyenge rugó található, amely lenyomja a membránt. A membrán középpontjában lyuk található, amely lehetővé teszi, hogy nagyon kis mennyiségű víz áramoljon rajta keresztül. Ez a víz úgy tölti ki a membrán másik oldalán lévő C üreget, hogy a nyomás a membrán mindkét oldalán egyenlő legyen, azonban az összenyomott rugó összességében lefelé irányuló erőt szolgáltat. A rugó gyenge, és csak azért képes bezárni a beömlőnyílást, mert a membrán mindkét oldalán kiegyenlítődik a víznyomás.

Amint a membrán bezárja a szelepet, az alja kimeneti oldalán lévő nyomás csökken, és a fenti nagyobb nyomás még szilárdabban zárja. Így a rugónak nincs jelentősége a szelep zárva tartása szempontjából.

Mindez azért működik, mert a kis D lefolyójáratot egy csap rögzíti, amely az E mágnesszelep armatúrája, amelyet egy rugó nyom le. Ha az áram áthalad a mágnesszelepen, a csapot mágneses erővel visszahúzza, ekkor a C kamrában lévő víz gyorsabban üríti ki a D járatot, mint amennyit a lyuk képes feltölteni. A C kamrában a nyomás csökken, a beérkező nyomás megemeli a membránt, ezáltal kinyitva a fő szelepet. A víz a mágnesszelep inaktiválódásáig közvetlenül A-ból F-be áramlik.

Amikor a mágnesszelep inaktiválódik, a D járat ismét záródik, ekkor a rugónak nagyon kevés erőre van szüksége a membrán újbóli lenyomásához, a főszelep bezárul. A gyakorlatban gyakran nincs külön rugó; a membrán úgy van kialakítva, hogy rugóként is funkcionáljon. [5]



A- Input side
B- Diaphragm
C- Pressure chamber
D- Pressure relief passage
E- Electro Mechanical Solenoid
F- Output side

3-1. ábra Mágnesszelep működése

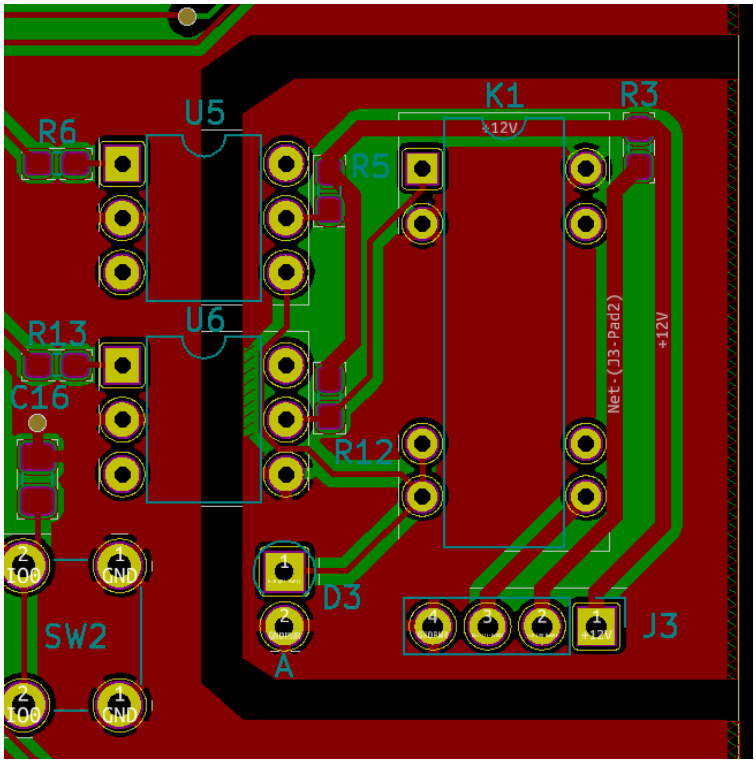
Ez a kialakítás normál állapotában zárt (NC), amely ebben az alkalmazásban előnyös, hiszen energiát, csak nyitott állapot közben fogyaszt. Egy öntözőrendszer egy nap során legtöbbet zárt állapotban van, mivel nem öntözünk folyamatosan.

3.1.2 Mágnesszelep vezérlése

A mágnesszelep működési elvéből kitalálhatjuk, hogy villamos vezérlés szempontjából egy tekercset (vagy gondolkodhatunk úgy is, hogy egy elektromágnest) kell energizált állapotba hozni, majd kikapcsolni. A rendszer olyan szelepeket tud kezelni, melyeket elegendő DC feszültséggel ellátni, így a tekercs energizálásához nem szükséges szinuszos vezérlő jelet előállítani. A méretezés ebben az esetben 12 V-ról működő és maximum 6 W-ot fogyasztó szelep alapján történt.

A kliensek alacsony fogyasztású mérőeszközök, nem rendelkeznek 12 V-os tápfeszültséggel, viszont 12 V-os táp csatlakoztatásával vezérelhető lesz a mágnesszelep.

A 12 V-os táp galvanikusan leválasztott a kliensek akkumulátorról előállított feszültségeitől. A leválasztást optikai csatoló áramkörrel és külön PWRGND rézréteg segítségével realizáltam, az IPC2221 szabvány által definiált minimumtávolság [6] többszörös betartásával. (3-2. ábra)



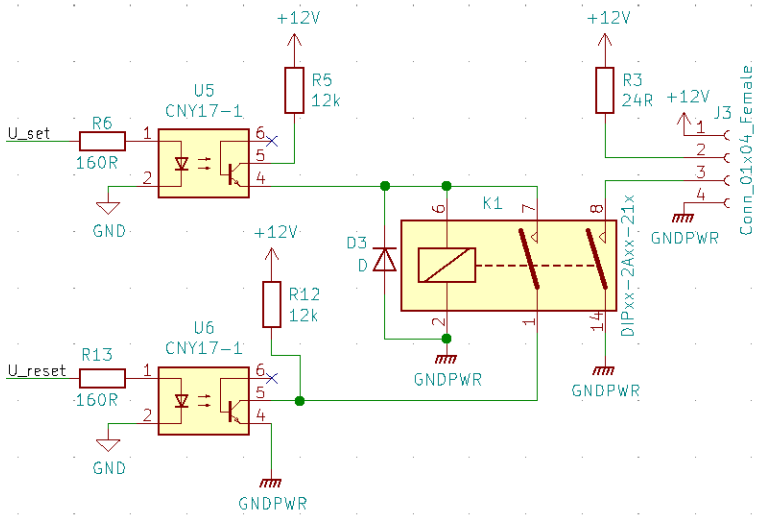
3-2. ábra Galvanikusan leválasztott öntartó relé kapcsolás realizációja

A 3-3. ábra tartalmazza az elvi kapcsolási rajzot. Az áramkörnek két bemenete van U_{set} és U_{reset}, melyek a be- és kikapcsolásért felelnek. A be- és kikapcsoló vezérlőjelek azonosak, legalább 6 µs széles 3.3 V feszültségű impulzusok. A R_6, R_{13} soros ellenállások áramkorlátozó szereppel rendelkeznek, 2 mA a maximális áram.

A bekapcsoló impulzus hatására U5 kapcsolón keresztül 1 mA-es áram folyik K1 relé tekercsén, mely hatására K1 NC kontaktusai bezárnak és J3 csatlakozón keresztül maximum 500 mA áram mellett a mágnesszelep tápot kap.

A bekapcsoló impulzus után az áramkör öntartásban marad K1 relé 1-7 kontaktusán keresztül, mindaddig amíg a relé tekercsének potenciálját a kikapcsoló impulzus hatására U6 kapcsoló 0 V-ra nem húzza.

A K1 relé tekercsének kikapcsolásakor figyelembe kell venni, hogy a tekercs mágneses terében tárolt energia kontrollált módon disszipálódjon el. D3 teljesítménydióda ezt a feladatot látja el, kikapcsolás esetén a tekercs a diódán keresztül disszipálja el energiáját.



3-3. ábra Galvanikusan leválasztott öntartó relé kapcsolás

A 12V-os külső táp J3 csatlakozón keresztül kerül a NYÁK-ra, így szükség esetén más DC feszültséget is kapcsoltathatunk a kliens eszközökkel, például 24 V-ot. Ekkor az ellenállás értékek változtatása szükséges lehet. A huzalok szélessége és egymástól való távolsága megengedi 3.3 V-tól 24 V-ig terjedő külső feszültség használatát.

3.1.3 Kliens tápellátása

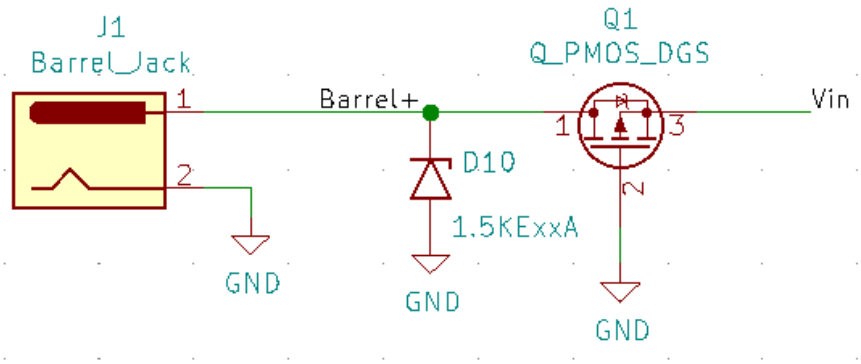
A kliensek mérési helyszíne nem teszi lehetővé a kényelmes csatlakozást a villamos hálózathoz. Mindenképpen telepes vagy akkumulátoros tápellátásra van szükség.

Alacsony fogyasztású alkalmazásoknál a telepes tápellátás nagy előnyökkel bír, hiszen technológiai előnye az akkumulátorokkal szemben, hogy elhanyagolható az önkisülési jelenség. Viszont egy tipikus CR2032 telep 100 mA-es csúcsnál nagyobb áram leadására nem képes, emiatt nagy méretű hidegítő kondenzátorokra (akár szuper kondenzátorokra) lenne szükség, hogy az ESP32 vezeték nélküli kommunikációja közben elegendő áramot tudjon szolgáltatni egy ilyen cella. Mindemellett 200-300 mAh-s tárolókapacitása, sem lenne elég a kliens ellátására.

Az viszonylag magas áramleadási szükséglet és nagy 2000-3000 mAh-s tárolókapacitás érdekében Li-ion akkumulátor alkalmazása mellett döntöttem.

A választás egy 18650-es és 3.7 V nominális feszültséggel rendelkező akkumulátorra esett, melynek a kapacitása 2600 mAh és maximális áramterhelhetősége 10A. A 18650 elnevezés az akkumulátor felépítésének dimenzióit adja meg, emiatt legalább 6,5 cm (praktikusan 8 cm) helyre van szükség a NYÁK-on az akkumulátornak. A helyigény mellett az akkumulátor beszerzési ára a legnagyobb hátránya ennek a megoldásnak, hiszen az körülbelül az alkatrészek árának ¼ része. (A hardveres költségek összegzését külön alfejezet tárgyalja.)

3.1.4 Bemeneti feszültség védelmei



3-4. ábra ESD és fordított polaritás védelem

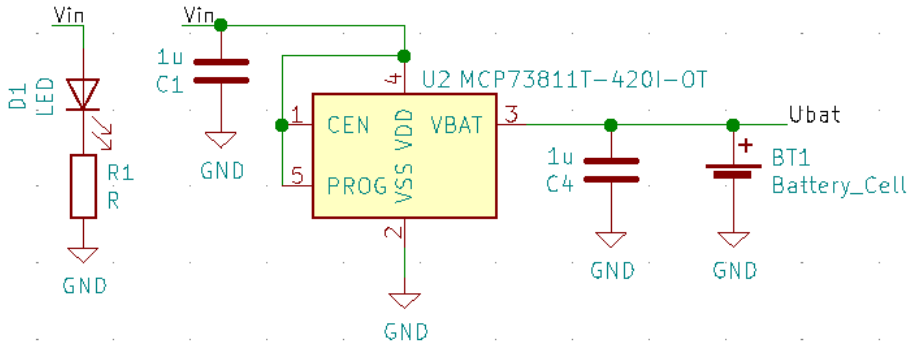
A 3-4. ábra a bemeneti feszültségre alkalmazott ESD és fordított polaritás védelem áramkörének kapcsolási rajzát tartalmazza. Az ESD védelmet TVS dióda látja

el. Amennyiben J1 csatlakozón fordított polaritású feszültség jelenik meg Q1 P csatornás MOS-FET szakadásként viselkedik, ezáltal védve az áramkört.

Ennek bizonyítására végezzünk gondolatkísérletet. Tegyük fel, hogy Q1 kapcsoló zárt állapotban van, ekkor V_{in} GND-hez képest negatív feszültség. Egy P csatornás FET csak negatív gate-source feszültség (U_{GS}) mellett van zárt állapotban. Viszont V_{in} GND-hez viszonyítva pontosan -$U_{GS,}$ tehát negatív V_{in} feszültség esetén U_{GS} pozitív, melyből következik, hogy a tranzisztor nyitott állapotban kell legyen.

Összegezve Q1 tranzisztor csak helyes bemeneti feszültség mellett lehet nyitott állapotban, U_{DS} maximális feszültségig védi az áramkört a fordított polaritás ellen.

3.1.5 Akkumulátor töltő áramkör



3-5. ábra Li-ion akkumulátor töltő áramkör

Az akkumulátor töltését egy single-cell (egy cellás) Li-ion akkumulátor töltő integrált áramkör látja el. Az akkumulátort konstans áramú töltés mellett 4.2 V feszültségig tölti az áramkör. Az IC PROG lábának logikai magas feszültségre kötésével 500 mA-es töltőáram lett beállítva. Így az akkumulátor $\frac{2600\ mAh}{500\ mA} = 5.2h \sim 5h$ alatt tölthető fel. Azáltal, hogy töltésre ritkán van szükség és az akkumulátor kevesebb, mint egy éjszaka alatt feltölthető, így elfogadható a töltési idő.

C_1 és C_4 kondenzátorok a be- és kimeneti feszültségek hullámosságának csökkentésére és impulzusszerű áramfelvételek töltés szükségletének ellátására szolgálnak. Más szempontból nézve passzív RC szűrőként funkcionálnak a tápvonalakon.

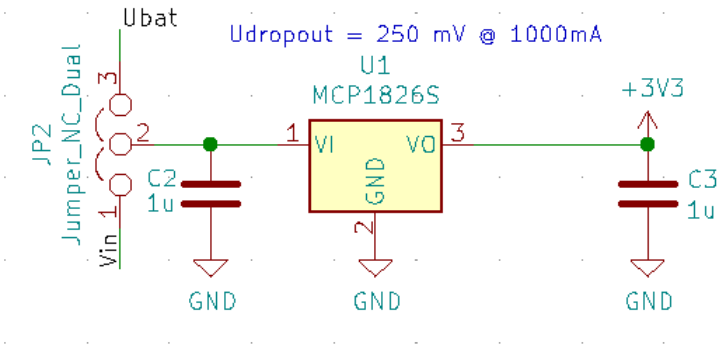
3.1.6 Szabályozott tápfeszültség előállítása

Az ESP32 és egyéb a kliens részét képző IC-k számára a tápfeszültséget egy 1000 mA-t leadni képes Low Dropout Voltage lineáris ún. áteresztő tranzisztoros szabályozó látja el. LDO-ra szükség az akkumulátor feszültség nominális értékének és 3.3 V-os előállítandó feszültségnek közelsége és a felesleges hőfejlesztés elkerülése miatt volt szükség.

C_2 és C_3 kondenzátorok szerepe megegyezik a 3.1.5 pontban szereplő C_1 és C_4 kondenzátorokkal.

A feszültségszabályzó bemenetére jumper segítségével külső tápfeszültséget is lehet csatlakoztatni, de ennek szerepe szigorúan fejlesztési célú.

A lineáris feszültségszabályozókat nagyfrekvenciás zajok ellen védeni kell, ennek gyakori megoldása egy a szabályzóval sorba kötött L-C szűrő. A szűrésre, azért van szükség, mivel a szabályozóban lévő tranzisztor nem képes végtelen sávszélességgel kapcsolni és a többletfeszültséget eldisszipálni. Azonban a Li-ion akkumulátorok feszültsége kevés zajt tartalmaznak, eleve jól szűrt tápforrásnak tekinthetők. Emellett még passzív LC szűrő alkalmazásához is szükség van egy viszonylag drága áramköri elemre, a tekercsre. A felsorolt érvek miatt (kis zajtartalmú forrás, költségminimalizálás) nem került szűrő az LDO bemenetére.



3-6. ábra LDO fix 3.3 V kimeneti feszültséggel

A kimeneten található 1µF-os kerámia kondenzátor mellett minden IC táplábaihoz közel 100 nF-os kondenzátor van párhuzamosan kötve C_3 kapacitással. Ennek segítségével növelve a kondenzátorok effektív sávszélességét.

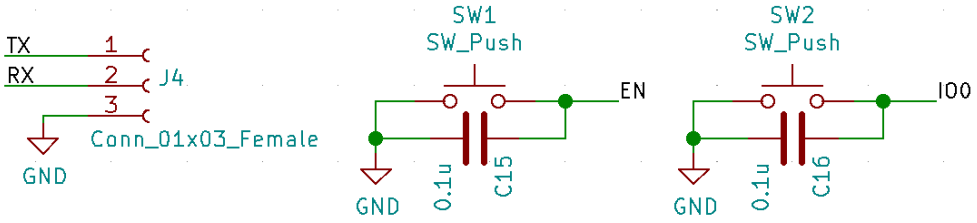
3.1.7 Programozó interface

Az ESP32 programozása UART periférián keresztül történik. Ez az UART periféria nem csak a programozhatóságot teszi lehetővé, hanem debug portként is használható fejlesztés közben, vagy javítások elvégzésénél. (Hibaüzenetek küldhetők rajta.)

Az ESP32 modul programozható állapotba hozásához szükséges a processzor IO0 lábát alacsony logikai szinten tartani, erre nyomógomb segítségével van lehetőség. Programozás után hardveres reset szükséges, melyet EN jel logikai alacsony szintre húzásával érhetünk el, nyomógomb segítségével.

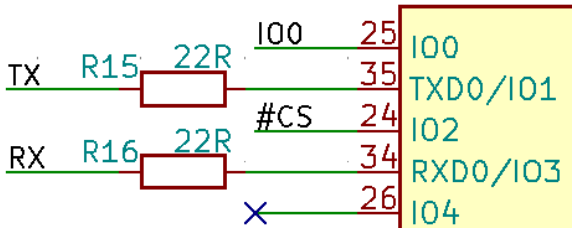
A nyomógombok pergésmentesítésére az ESP32 Dev Kit C adatlapja által javasolt megoldást alkalmaztam. [7]

A programozó interface és a pergésmentesített nyomógombok kapcsolási rajzát a 3-7. ábra tartalmazza.



3-7. ábra Programozó interface

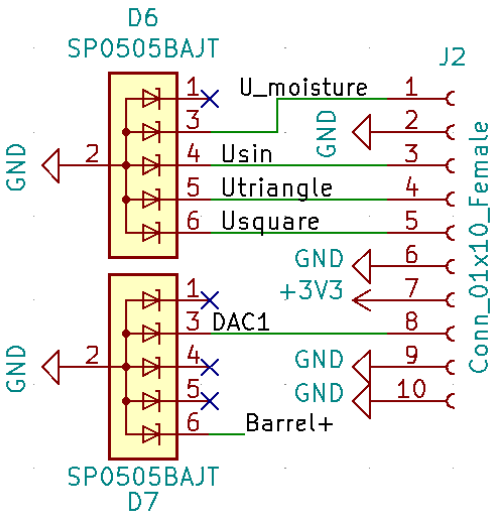
Kihasználva a NYÁK-on megjelenő parazitakapacitásokat, kis értékű 22 Ω értékű ellenállások felhasználásával TX és RX jelek meredek éleit szűröm, ezzel csökkentve az általuk a NYÁK-on keltett zajt.



3-8. ábra Digitális jel szűrőáramköre

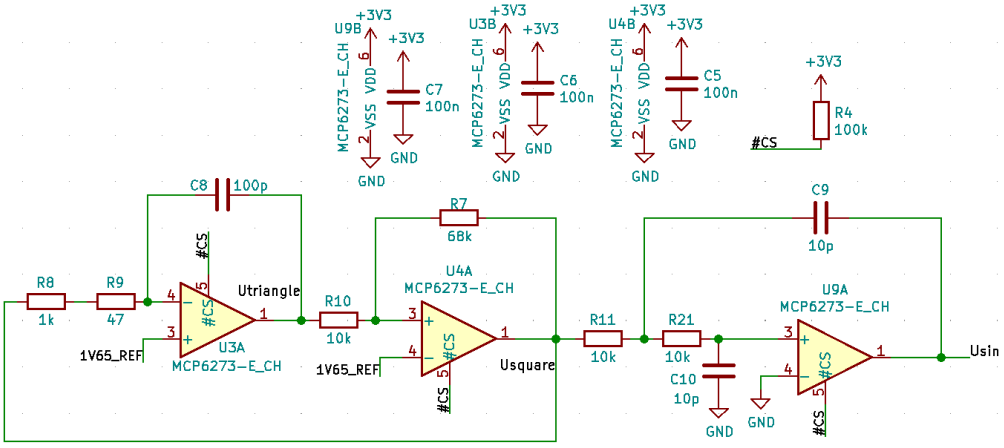
3.1.8 Méréshez szükséges csatlakozó

A méréshez szükséges gerjesztőjelek és a mért jel, 3.3 V és GND jelekkel kiegészítve tüskesor csatlakozón találhatók a NYÁK-on. A mérővezetékek hossza, illetve a külvilággal való rendeltetésszerű kapcsolata miatt ESD védelemmel láttam el a tüskesor csatlakozót, TVS dióda tömb alkalmazásával.



3-9. ábra Tüskesor csatlakozó ESD védelme

3.1.9 Mérőjel előállító áramkör



3-10. ábra Mérőjel előállító áramkör

A 3-10. ábra tartalmazza a méréshez szükséges 1.5 MHz-es jeleket előállító áramkört. A méréshez DC komponenst is tartalmazó, négyszögjelet és szinuszjelet állítunk elő.

Az áramkör három alapkapcsolásból, egy invertáló integrátorból, egy Schmitt-triggerből és egy Sallen-Key szűrőből áll, melyek együttesen hozzák létre a kívánt jeleket.

A 3-10. ábra által ábrázolt kapcsolásban a létrehozott négyszögjel kerül szűrésre, egy Sallen-Key kapcsolás másodfokú aluláteresztő kapcsolás alkalmazásával.

Mindhárom alapkapcsoláshoz lineáris műveleti erősítőre van szükség, mellyel szemben a következő feltételeket határoztam meg:

- legalább 1.5 MHz működési sávszélesség
- aszimmetrikus táp lehetősége
- rail-to-rail működés
- alacsony fogyasztás

A választás a MCP6273 gyártói jelű integrált áramkörre esett, mely egy 170 µA fogyasztási móddal és aszimmetrikus táppal rendelkező 2 MHz sávszélességű rail-to-rail műveleti erősítőt tartalmaz. [8]

A méretezés során figyelembe vettem, hogy 0603 és 0508 méretekkel rendelkező SMD alkatrészekkel történik a megvalósítás, illetve az E24-es szabványsor szerinti értékek érhetők el.

Az erősítő engedélyezéséhez egy aktív alacsony jelet kell a mikrovezérlőnek előállítani. A $\overline{CS}$ lába az IC-nek 100 kΩ ellenálláson keresztül 3.3 V feszültségre van kötve. A mérőjel frekvenciáját meghatározó RC-tag a 3-10. ábra C8, R8 és R9 elemeiből tevődik össze.

A frekvencia a következő egyenlet szerint számítható:

$$f = \frac{1}{2\pi} \cdot \frac{1}{(R_8 + R_9) \cdot C_8} \cong 1520104.52\ \text{Hz} \approx 1.5\ \text{MHz}$$

A komparátor (Schmitt-trigger) billenési feszültségét a következő módon származtathatjuk: [9]

$$IN_+ = U_{ref}$$

$$IN_+ = U_{trigangle} \cdot \frac{R_7}{R_{10} + R_7} + U_{square} \cdot \frac{R_{10}}{R_{10} + R_7}$$

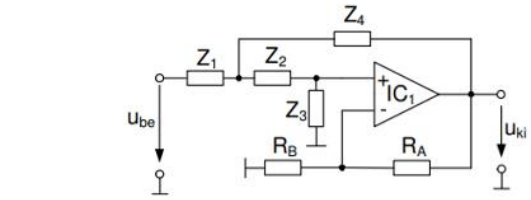
$$U_{billenési} = \left(1 + \frac{R_{10}}{R_7}\right) \cdot U_{ref} - \frac{R_{10}}{R_7} \cdot U_{square}$$

,ahol $U_{ref} = 1.65V$, $R_7 = 68\ k\Omega$ *és* $R_{10} = 10\ k\Omega$

Ezek alapján a billenési feszültségek: $U_1 \cong 1.41V$ $U_2 \cong 1.89\ V$

A bevezetett hiszterézis ugyan a háromszögjel szimmetriáján ront, viszont biztosítja, hogy nem történhet oszcillálás a komparátor kimenetén.

A szinuszos mérőjel ellőállításához Sallen-Key alaptagot alkalmaztam, mellyel másodfokú aluláteresztő szűrőt valósítok meg. [10]

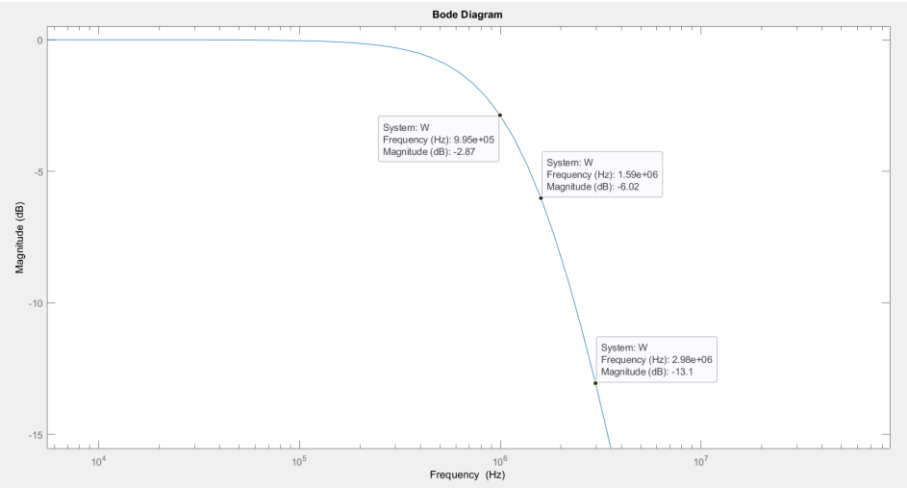


Z1	Z2	Z3	Z4	MEGVALÓSÍTOTT SZŰRŐ
R1	R2	C3	C4	Másodfokú aluláteresztő szűrő

3-11. ábra Sallen-Key alaptag

A 3-10. ábra szerinti kapcsolásban R_1 szerepét R_{11}, R_2-t R_{21}, C_3-t C_{10} és C_4-t C_9 veszi fel. A kapcsolás átvitele a következő:

$$\frac{U_{ki}\,(s)}{U_{be}\,(s)} = \frac{\frac{1}{R_{11}C_{10}R_{21}C_9}}{s^2 + s\left(\frac{1}{R_{21}C_9} + \frac{1}{R_{10}C_9}\right) + \frac{1}{R_{11}C_{10}R_{21}C_9}}$$



3-12. ábra Sallen-Key szűrő Bode-diagramja

Az előzőekben ismertetett tervezési folyamat a műveleti erősítő aszimmetrikus táplására miatt hibás. Erre a hibára a NYÁK bemérése során derült fény, melynek okát a 3.2.5 pontban tárgyalom.

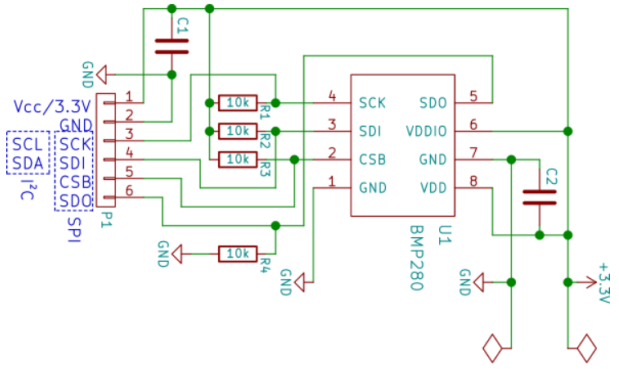
3.1.10 Hőmérséklet- és nyomásmérő áramkör

Az öntözőrendszer számára fontos, hogy a környezeti hőmérsékletet (elsősorban talajközeli levegőréteg hőmérsékletét) legalább 1 ºC pontossággal mérje, annak érdekében, hogy fagy vagy fagyközeli levegőhőmérséklet esetén, ne kapcsolja be az öntözést, még a talaj nedvességtartalmának csökkenése esetén sem.

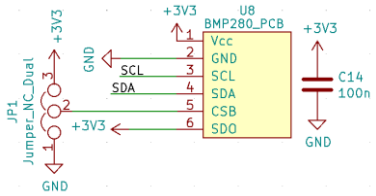
A levegő nyomásának mérésével következtethetünk az időjárás változására például a nyári záporok előtt nyomásváltozás történik. [11]

Erre a feladatra a BMP280 elnevezésű szenzort használtam. Amellett, hogy megfelelő pontossággal mér hőmérsékletet és nyomást, kapható ún. breakout boardon is, így külön kis NYÁK-ra került a szenzor és a nagyáramot is szállító NYÁK hőmérséklete kevésbé befolyásolja a hőmérséklet mérést.

A szenzort I2C (opcionálisan SPI) buszon lehet elérni 0x76 és 0x77 címen, melyen hardveresen lehet állítani, egy jumper segítségével. A kapcsolás belső felépítését a 3-13. ábra, a kliens NYÁK-hoz csatlakozását a 3-14. ábra mutatja. [12]

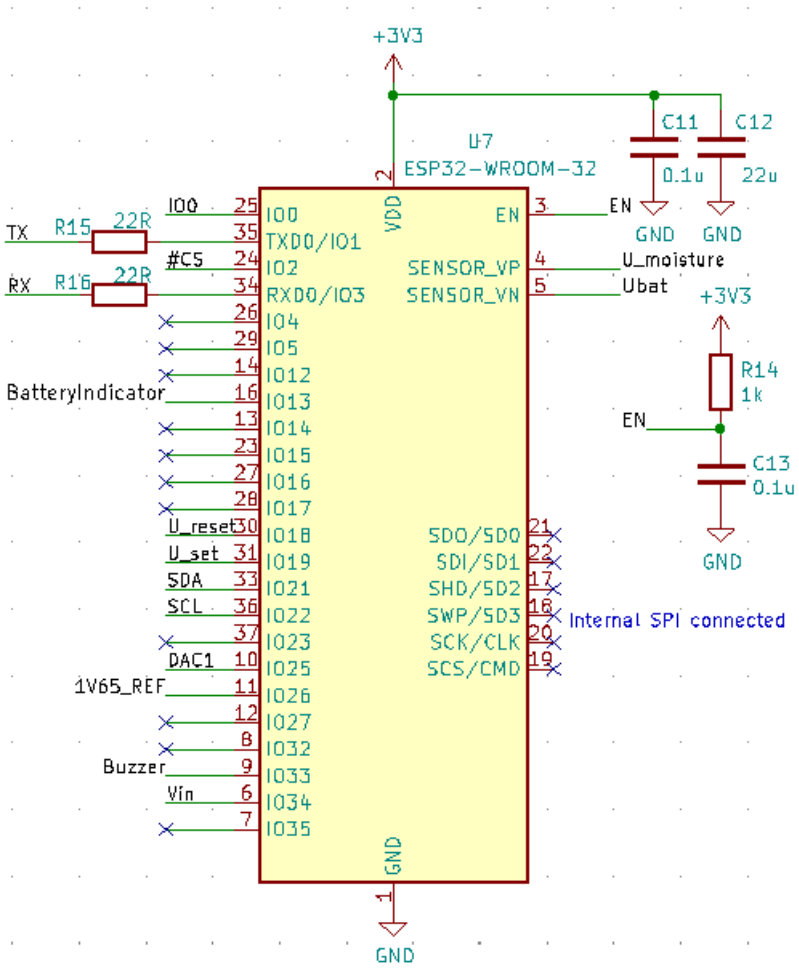


3-13. ábra BMP280 PCB belső felépítése



3-14. ábra BMP280 PCB külső áramköre

3.1.11 ESP32 WROOM-32 periféria áramkörei



3-15. ábra ESP32 WROOM-32

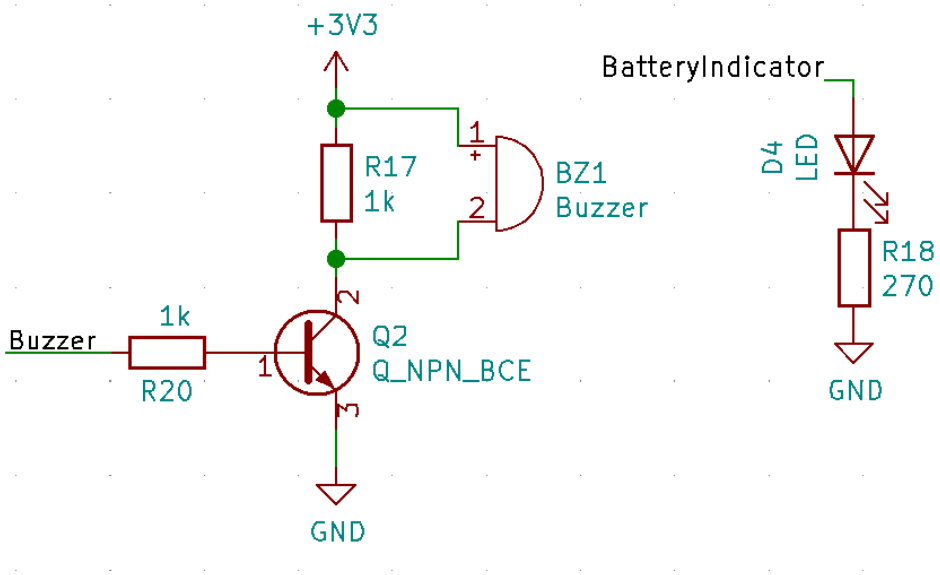
Az ESP32 modul ezen alkalmazáshoz szükséges periféria áramkörök nagy részét tartalmazza, így nem volt szükség sok külső alkatrész csatlakoztatásához. [13]

A 3-15. ábra C_{11} és C_{12} hidegítő kondenzátorok a gyártó által javasolt méretekkel rendelkeznek.

R_{14} és C_{13} az engedélyező jel (EN) szűrését látják el.

Az általam használt WROOM-32 modul tartalmaz 4 MB flash memóriát, amely SPI buszon keresztül összeköttetésben áll a processzorral. Emiatt a modul 17-22 lábait nem szabad semmilyen külső egységhez kötni.

3.1.12 Egyéb funkciókat ellátó áramkörök



3-16. ábra Buzzer és LED meghajtó áramkörök

A 3-16. ábra tartalmazza a keresési funkcióhoz szükséges buzzer meghajtó áramkört, illetve az akkumulátor indikátor áramkört. A LED 5 mA-es árammal és 2 V nyitófeszültséggel lett méretezve. A buzzer áramkört 4 kHz közötti négyszögjellel kell meghajtani.

3.2 Nyomtatott huzalozású áramköri lemez terv

3.2.1 Csatlakozó választás

A klienst alkotó áramkört nem prototipizáló panelen, hanem nyomtatott huzalozású lemezen készítettem el, annak érdekében, hogy a feszültség mérés pontossága minél nagyobb és a méréshez szükséges „nagy” frekvenciás jelek zajterhelése minél kisebb legyen.

A tervezés közben prototípusként kategorizáltam az elkészítendő NYÁK-ot, ezért kivétel nélkül minden csatlakozó tüske- és hüvelysor, melyek az olcsó és gyors fejlesztést könnyítették meg. Az elkészült NYÁK továbbfejlesztett verziója elkészítésekor már mindenképpen figyelembe kell venni a külvilág fel való csatlakozás gyakorlati kényszereit is.

3.2.2 Technológiai paraméterek

A NYÁK terv az UniPCB kft. legolcsóbb gyártási technológiájának megfelelő méreteket tartalmazza [14], melyek betartásához két huzal osztály rendeltem, egy alapértelmezett (default) és teljesítmény (power) osztályt.

Net Classes

Name	Clearance	Track Width	Via Size	Via Drill	µVia Size	µVia Drill	dPair Width	dPair Gap
Default	0.25 mm	0.25 mm	0.9 mm	0.7 mm	0.3 mm	0.1 mm	0.25 mm	0.25 mm
Power	0.25 mm	0.3048 mm	1.5 mm	1 mm	0.3 mm	0.1 mm	0.25 mm	0.25 mm

3-17. ábra Huzal osztályok

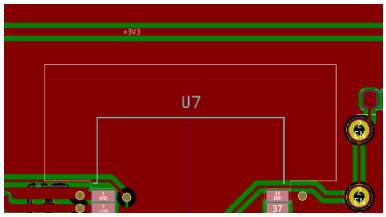
A NYÁK két oldalas és forrasztásgátló maszkkal bevont, szitanyomással feliratozás is került rá (alkatrész jelzés, felirat és logó). A rézrétegek között FR4 szigetelő anyag található.

A NYÁK mérete 10x10 cm, amelynek legfőbb oka, hogy 18650-es akkumulátor is a mechanikailag rögzített hozzá. A panel magasságát is az akkumulátor mérete határozza meg, 23 mm a legmagasabb pontja a BOT réteghez képest.

3.2.3 Alkatrész választás és topológia

Felületszerelt és furatszerelt alkatrészek egyaránt felhasználásra kerültek. Elsősorban nyomógombok, csatlakozók és a galvanikus leválasztást megvalósító alkatrészek furatszereltek, minden többi alkatrész esetén előnyben részesített a felületszerelt technológia. Annak érdekében, hogy figyelembe lehessen venni, hogy milyen alkatrészek állnak rendelkezésre már a tervezés folyamata közben, így az előzőekben említett általános elv alól kivételeket képez néhány alkatrész (erre jó példa Q2 tranzisztor).

Az ESP32 antennájának pár milliméter szabad helyet kell biztosítani, emiatt csak alacsony frekvenciás jelek huzaljai közelítik meg, illetve GND kitöltés veszi körül TOP és BOT rétegeken egyaránt. Emellett alkatrész sem található a NYÁK ezen részén egyik szerelési oldalon sem.

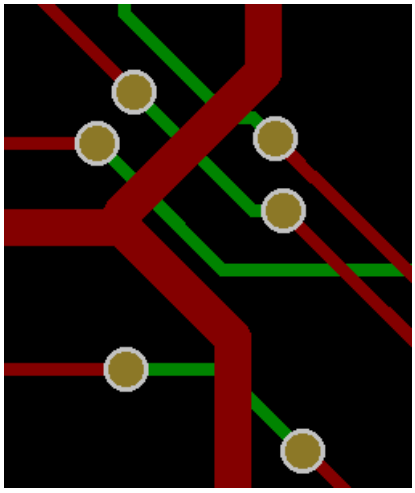


3-18. ábra ESP32 antenna környezete

Az alkatrészek elhelyezése további kötöttségektől mentes volt és hely a panelen bőven rendelkezésre állt az akkumulátor által megszabott méret miatt.

3.2.4 Tápfeszültséget szállító hálózat

A két réteges kialakítás miatt tápréteg kialakítására csak GND számára volt lehetséges, illetve a síkbarajzolhatóság miatt GND táprétegre is kerültek rövid jelszállító huzalok. A 3,3 V-os huzal elágazásai 45 fokos szöggel lettek implementálva (3-19. ábra). Emellett ügyeltem, hogy ne hozzak létre hurkot, ezzel minimalizálva az induktív zajok (pl. a NYÁK-on található relék kapcsolásakor keletkező zaj) hatását a tápfeszültség hullámosságára.



3-19. ábra 3,3 V huzal elágazás

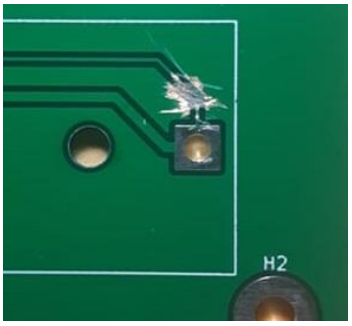
A tápfeszültséget 100nF-os kondenzátorok hidegítik, minden aktív alkatrész fizikai közelében. A processzor mellett 22 nF és 100 nF kondenzátorok párhuzamosan elhelyezve szélesebb frekvenciasávban képesek szűrni a processzor tápvonalát. A tápfeszültséget előállító lineáris feszültség szabályozó fizikai közelében 1 µF értékű kondenzátorok elhelyezésével csökkentem tüske szerű nagy áramok felvételekor a feszültségesést a tápvonalon.

A 12 V-os tápfeszültséggel rendelkező beavatkozó hálózat megvalósítását a 3.1.2 pontban tárgyaltam.

3.2.5 Beültetés és bemérés

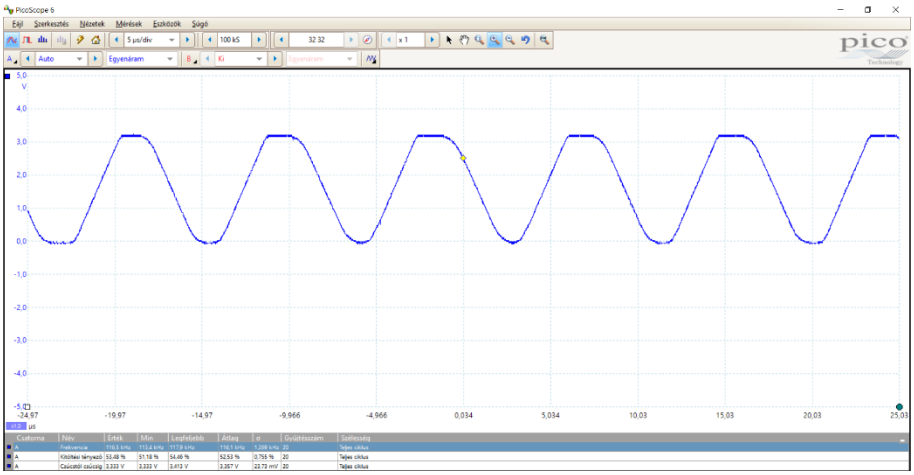
A legyártott NYÁK-re kerülő alkatrészeket a tanszéki önálló labor eszközeinek segítségével kézi forrasztással ültettem be. A beültetés megkezdése előtt két ponton egy-egy huzalt utólagosan megszakítottam, mivel azok tervezési hibából adódóan a processzort károsították volna. Ubat jel mérésére feltétlenül szükség van, ezért utólag furatszerelt ellenállásokkal és szigetelt kábel segítségével $\frac{U_{bat}}{2}$-t kötöttem a processzor 5. padjéhez, így biztosítva, hogy az ne károsodjon az akkumulátor feszültségétől.

3-20. ábra Vin javítása (BOT)

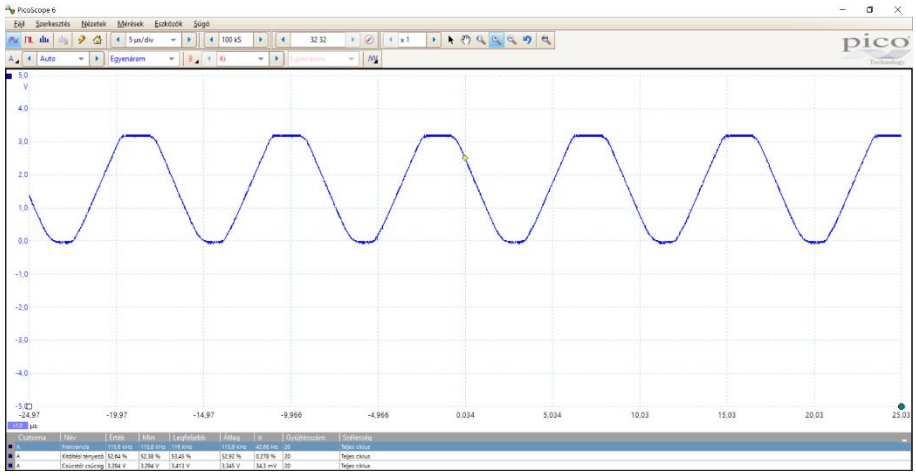


3-21. ábra Ubat javítása (TOP)

A 3.1.9 pontban ismertetett kapcsolás nem működőképes negatív tápfeszültség nélkül, mivel az integrátor kimenetén kialakuló váltakozó előjelű meredekséggel rendelkező háromszögjel nem tud kialakulni csak nemnegatív tápfeszültség mellett. Ez egy tervezési hiba, amely utólag nem került javításra. Az áramkör bemérésekor mindössze 150 kHz körüli „szinuszos” oszcillációt mértem az áramkör kimenetein (3-22. ábra és 3-23. ábra).

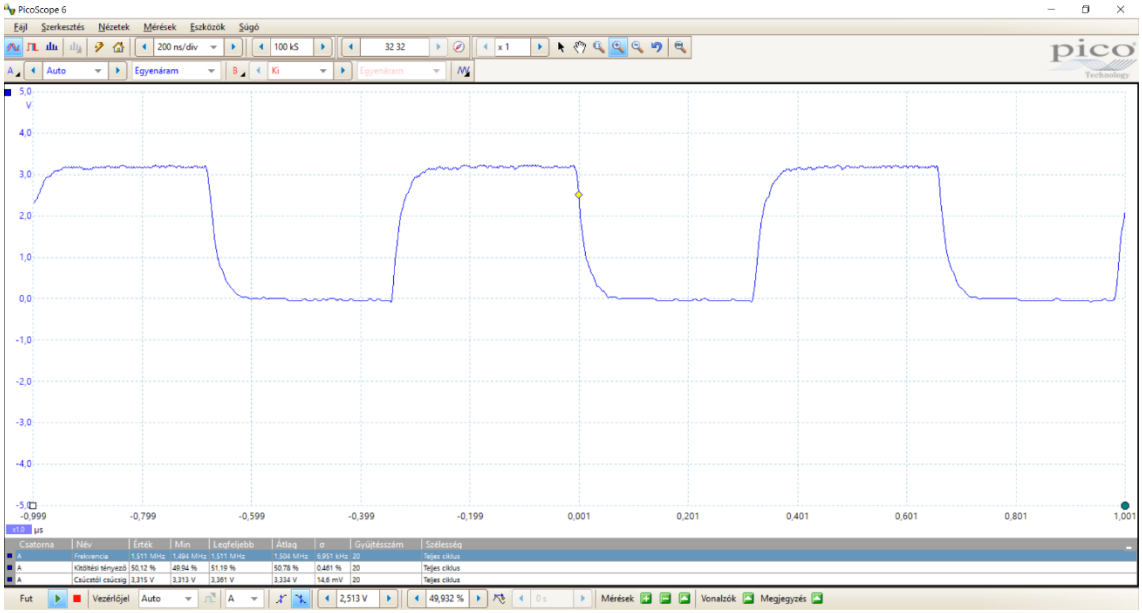


3-22. ábra Utrig jel



3-23. ábra Usqr jel

A mérőjel előállítására az ESP32 szoftveresen konfigurálható PWM generátorának egyik csatornáját használtam fel, két bites felbontással és 50%-os kitöltési tényezővel (3-24. ábra).



3-24. ábra Mérőjel

Az így feleslegessé vált műveleti erősítőket a panelról eltávolítottam, hogy fogyasztásuk ne adódjon hozzá az összefogyasztáshoz.

A NYÁK-on megvalósított többi hardveres funkció az elvi kapcsolási rajzon ismertetett működésnek megfelelően működik.

3.2.6 Kliens egység árkalkulációja

Egy kliens elkészítéséhez szükséges alkatrészek árát a hivatkozott táblázat tartalmazza. [Táblázat]

Összesen tehát kb. 14 ezer forint egy kliens ára. Ennek az árnak 22%-a az akkumulátor, 13%-a pedig a beavatkozó szervhez szükséges relé. Mivel nem szükséges minden kliensnek beavatkozó szervként is működni, vagy épp a beavatkozó szervként működő akár hálózatról is táplálható, az akkumulátor vagy relé nélküli variációk jelentősen olcsóbban előállíthatók. Ezek a költségcsökkentések jelentősek, viszont nem ellensúlyozzák az eleve magas árat, illetve ebben a kalkulációban nem szereplő, de a terepi működéshez szükséges csatlakozók beszerelése további költségeket jelent.

Összesítve, kis számú mérőegység telepítése még elképzelhető egy szenzorhálózatba, de komoly költségcsökkentő változtatásokra szorul a kliens egység műszaki terve.

3.2.7 Kliens egység fogyasztás

A fogyasztás jellemzően két állapottal rendelkezik. A mérés és a kommunikáció alatt történő nagy áramfelvétel és a mélyalvás során történő kicsi. A nagy áramfelvétel 1 percig tart, ezt követi 15 perc mélyalvás. Méréssel meghatároztam mindkét értéket:

$$I_m = 44\ \text{mA}\,, I_{ds} = 1.4\ \text{mA}$$

Fentiek alapján az átlag áramfelvétel:

$$\bar{I} = \frac{I_m \cdot 60\ [s] + I_{ds} \cdot 900\ [s]}{960\ [s]} = 4{,}0625\ \text{mA}$$

Az akkumulátor tároló kapacitása:

$$C = 2600\ mAh = 2600 \cdot 3600 = 9\,360\,000\ mAs$$

A mérőegység névleges üzemideje tehát,

$$T_{\text{üzem}} = \frac{C}{\bar{I}} = 2304000\ \text{s} = 640\text{h} \cong 26\ \text{nap}$$

4 Szoftver tervezés

4.1 Kliens beágyazott szoftvere

4.1.1 Platform választás

A tavaszi önálló laboratórium feladat során megismertem az Espressif IoT Development Framework nevű könyvtárkészletet és annak használatát. [2] Ez a gyártó által kiadott szoftvercsomag az ESP32 minden funkciójának kihasználására lehetőséget nyújt, viszont nincs hivatalosan támogatva semmilyen fejlesztőkörnyezet által. Az önálló laboratórium során fejlesztő környezetet bonyolultan és lassan sikerült felállítani a könyvtár használatához és fél év alatt verzió frissítés miatt nem működőképes szakdolgozatom írásakor.

Az előzőekben említett nehézség miatt, illetve mivel a szükséges szoftveres funkciók elérhetőek más platformon is, az Arduino IDE fejlesztőkörnyezetet használtam az ESP32 modul programozására. A fejlesztőkörnyezet gyors prototipizálást tesz lehetővé, míg az általam használt funkciók teljes egészét elérhetővé teszi a fejlesztés során. [15]

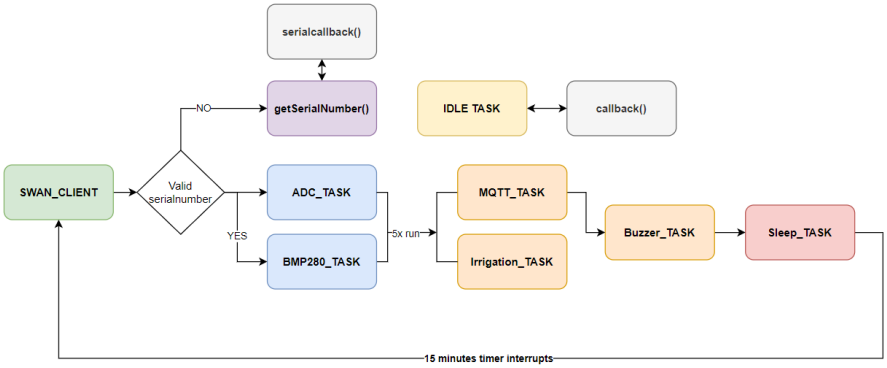
4.1.2 FreeRTOS ESP32 mikrovezérlőn

A FreeRTOS egy processzor maggal rendelkező mikroprocesszorokra tervezett, ezért az ESP32-n való futáshoz módosításokat végzett a gyártó az operációsrendszer forrásfájljaiban is.[16]

Az arduino fejlesztőkörnyezet, ezt a módosított FreeRTOS operációs rendszert fordítja az alkalmazásunk kódjával együtt futtatható fájlba. Az arduino platformon megszokott belépési pont a setup() függvény, míg a loop() függvény a FreeRTOSban használt idle taszk szerepét kapja meg. A setup függvényben vagy tetszőleges más helyen FreeRTOS taszkokat hozhatunk létre. Az operációs rendszer konfigurálására sajnos nincs lehetőség, ez ennek a fejlesztési platformnak a legnagyobb hátránya (pl. tickless idle nem engedélyezhető).

4.1.3 Implementált taszkok és függvények

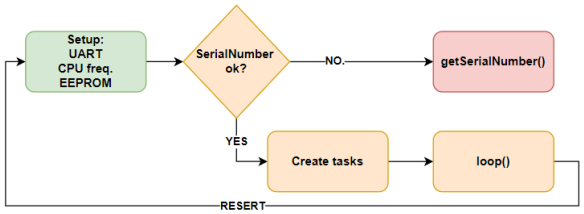
A 4-1. ábra az összes taszkot és függvényt, illetve azok egymáshoz viszonyulását mutatja be. A bootloader futása utáni belépési pont a zöld színnel jelölt taszk. A pirossal jelölt a mélyalvást indító taszk, ezért ezt „kilépési” pontnak tekinthetjük. A lila színnel jelölt függvényen kívül minden taszk akár egyszerre is futhat.



4-1. ábra Taszkok ütemezése

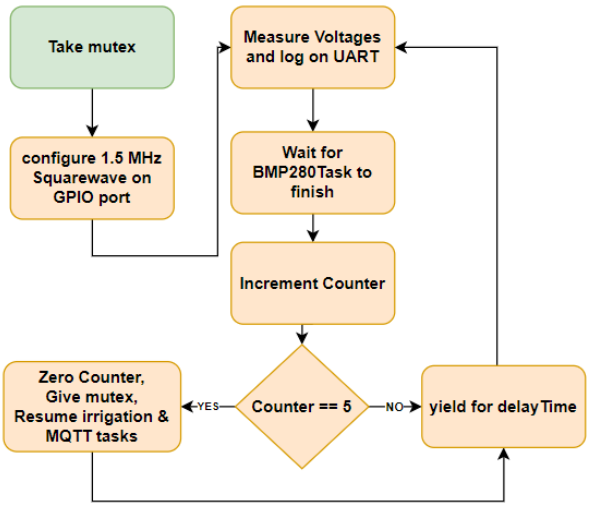
A következőkben egyesével ismertetem a kliensen futó taszkokat és feladataikat.

SWAN_CLIENT:



Bootloader futása után a zölddel jelölt ponton lépünk be. A futás kétféle lehet, vagy sorozatszámot szerzünk a szervertől, vagy létrehozzuk a taszkokat és átadjuk az irányítást az ütemezőnek.

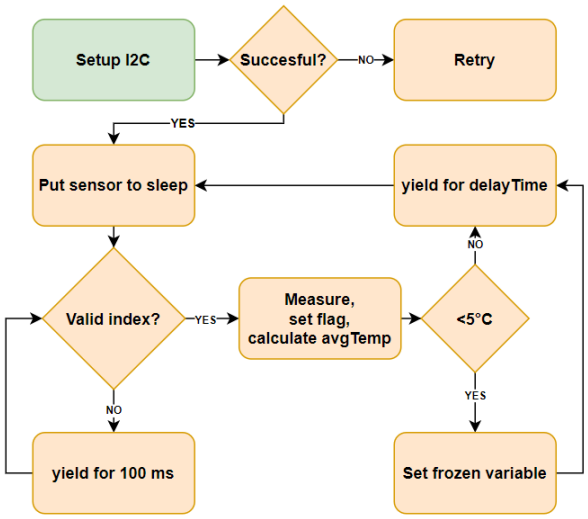
ADC_TASK:



A taszk feladata, hogy az akkumulátor és a szenzor feszültségét mérje. Mivel ciklikus bufferben tárolja a méréseket, ezért az indexváltozót is növelnie kell. Az index változót használja egy másik taszk is, így az közös erőforrás, de egy egyszerű „zászló” változó segítségével szinkronizáció történik, így nincs szükség az operációs rendszer által biztosított erőforrásigényesebb megoldás alkalmazására.

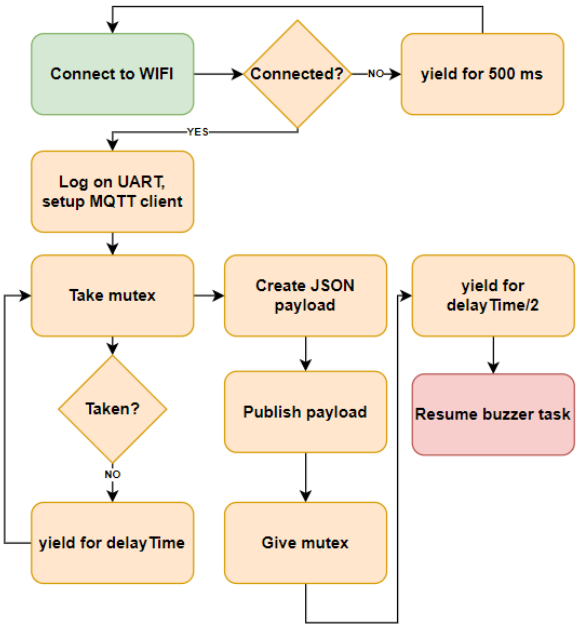
BMP280_TASK:

A taszk feladata a hőmérséklet és a nyomásmérés. 5 ºC alatti átlaghőmérséklet esetén szükséges az öntözés tiltása egy változó segítségével. A mérés I2C buszon lévő szenzor segítségével történik és amennyiben nincs jelen szenzor a buszon UART üzenetként jelez a kliens és újra próbálkozik 100 ms elteltével.



MQTT_TASK:

Az MQTT taszk tartja a kapcsolatot a szerverrel, amennyiben WIFI kapcsolatot tud létrehozni, akkor inicializál egy MQTT klienst, amely callback függvény segítségével fogad üzenetet. Amennyiben sikeresen elvette a mutexet (történt 5 mérés), az adatokat JSON objektum szerinti formátumban karakterláncként teszi közzé a SWAN/pub témában.



callback:

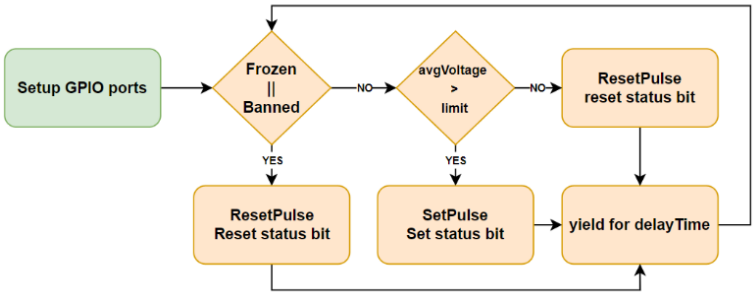
Amennyiben üzenetet tesznek közzé, bármilyen témában a kliens fogadja azt és ezt a callback függvényt hívja meg. A függvény karakterláncba menti mind témát, mind az üzenetet. Amennyiben speciális üzenet érkezik SWAN/recv témában az annak megfelelő változók beállítása is itt történik.

IDLE_TASK:

Az arduino fejlesztőkörnyezet a loop() függvényben található kódot fordítja az idle taszk kódjaként. Mivel tickless idle nem valósítható meg, ezért ezt a taszkot az MQTT kliens működéséhez szükséges client.loop() függvény hívására használom, mivel ezt a függvényt gyakran kell hívni a szoftverdokumentáció szerint. [17]

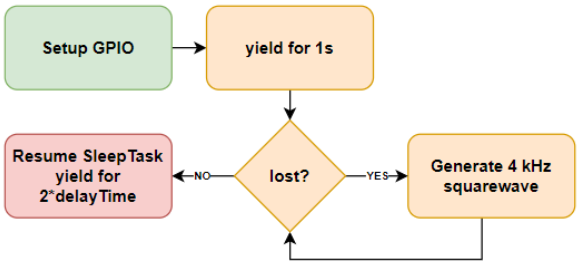
Irrigation_TASK:

A taszk feladata az öntözés ki- és bekapcsolásához szükséges pulzust előállítani és a státuszt leíró változót ennek megfelelően beállítani.



BUZZER_TASK:

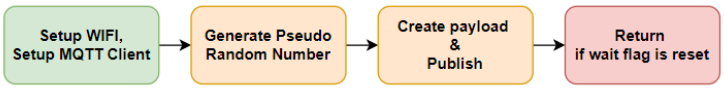
A taszk feladata, hogy 4 kHz frekvenciájú négyszögjelet generáljon, amíg a szervertől azt az információt kapja, hogy „elveszett”. Amennyiben nincs ilyen üzenet a mélyalvást okozó taszk elengedésével többször nem fog ez a taszk sem futni.



Sleep_TASK:

A taszk feladata, hogy inicializálja a mélyalvásból felkeltő időzítőhöz kötött megszakítást. Ezek után mélyalvás következik.

getSerialNumber:



A kliensek egyedi azonosítóval rendelkeznek, melyet újraindítások között is megtartanak, ezért flash memóriában tárolom őket. Amennyiben a flash memóriában tárolt érték 255 az azt jelenti, hogy az ESP még a gyári értéket tartalmazza a 0 címen. Ekkor ennek a függvénynek a feladata, hogy WIFIhez csatlakozzon és MQTT üzenetekkel azonosítót szerezzen. A függvény hívása után a kliens szoftveres újraindítást kezdeményez.

serialcallback:

Az azonosító sorszám megszerzése során használt callback függvény, amely az MQTT brókertől érkező karakterláncot feldolgozza és az új sorszámot elmenti a flash memóriába.

4.2 Központi egység szoftvere

4.2.1 SWAN szerver szoftveres környezete

A SWAN szerver egy Raspberry Pie Zero W single board [18] számítógépen fut. A RPI hivatalos operációs rendszerét DietPi [19] szoftverre cseréltem, annak érdekében, hogy minél kevesebb erőforrást használjon el az operációs rendszer és több maradjon az MQTT broker futtatására és a webes felület kiszolgálására.

A teljes rendszerhez való hozzáférés SSH-n keresztül vagy monitor és USB billentyűzet csatlakoztatásával lehetséges. Ezen felül az eszköznek kiosztott IP címen keresztül különböző telepített szoftverek kezelőfelületei is elérhetőek lokális hálózatról.

A teljes szoftveres felépítéshez egy publikusan elérhető SD kártya image-t használtam fel, amely több használt alkalmazás szoftverét előre telepített és „élesztett” állapotban tartalmazta.

https://www.sensorsiot.org/diet-pi-supporting-material-videos-126-and-128/

4.2.2 MQTT broker

Az MQTT protokoll ideális IoT rendszerek adatainak közvetítésére, mivel kis erőforrás és kód igényű, megbízható és kétirányú kommunikációt valósít meg.

Az MQTT Publish/Subscribe architektúrát leíró protokoll, amelyben alap esetben egy broker és több kliens közötti kommunikációt tesz lehetővé.

Minden üzenet küldés akár egy kliens, akár a bróker küldi egy adott témába (topic) kerül publikálásra. A kliensek feliratkoznak egyes témákra, amelyet a bróker jegyez és abban az esetben, ha arra üzenet érkezik minden feliratkozott kliensnek közvetíti. A brókeren keresztül megy minden üzenet, nincs lehetőség kliens-kliens közvetlen kapcsolatra.

A választásom a brókert megvalósító szoftverre, a mosquitto mqtt broker, amely egy nyílt forráskódú lightweight megoldás. [20]

4.2.3 SQLite

A kliensek felől érkezett adatokat tárolni kell a szerveren, hogy azokat feldolgozhassuk, illetve megjeleníthessük a felhasználó számára. Nagy mennyiségű adat rendszerezett tárolására az egyik legelterjedtebb módszer a relációs adatbázisok

alkalmazása. Az adatbázisokhoz való hozzáférés szabványosan Structured Query Language (SQL) nyelv segítségével tehetjük meg. [21]

A SWAN szerver SQLite adatbázis motort alkalmazza [22], mely nyílt forráskódú lightweight megoldást nyújt. Lekérdezések mellett böngészőben megnyitott felületen is van lehetőségünk elérnünk az adatbázis kezelő motort. Ilyen módon került létrehozásra az a két tábla (SWAN,serialnumbers), amelybe adatokat szúr be a szerver működése során.

CREATE TABLE 'SERIALNUMBERS' ('key' INTEGER PRIMARY KEY NOT NULL, 'timestamp' DATETIME DEFAULT CURRENT_TIMESTAMP)

CREATE TABLE 'SWAN' ('primkey' INTEGER PRIMARY KEY NOT NULL, 'id' INTEGER, 'moisture' REAL, 'temperature' REAL, 'pressure' REAL, 'status' INTEGER, 'battery' REAL, 'timestamp' DATETIME DEFAULT CURRENT_TIMESTAMP)

4.2.4 Node-RED

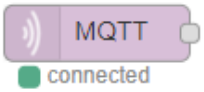
A SWAN szerver három fő feladatot lát el, felhasználói felületet biztosít a mért adatok megtekintéséhez és a kliensekhez való hozzáférésnek, MQTT brókert futtat és adatbázisban tárolja a mért adatokat.

A három fő funkció összefésülésére egy IoT rendszerekben gyakran használt programozási eszközt használtam, a Node-RED alkalmazást. [23]

Az alkalmazás lehetővé teszi, hogy úgynevezett node-k összekötésével egyszerű és átlátható módon készítsünk backend-et és frontend-et összekötő szoftvert. Lehetővé teszi, hogy adatáramlást figyelve programozzunk, magas absztrakciós „nyelven”. Node.js alapon működik, amely egy esemény vezérelt modellű lightweight szoftverrendszer. [24]

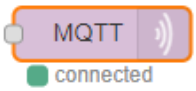
Az alkalmazást, úgynevezett flow-t, böngészőben elérhető grafikus felületen érhetjük el. A következőben ismertetem a SWAN szerver elkészült szoftverében szereplő node-okat általánosan, majd külön alfejezetben a konkrét programot.

MQTT input node:



Feladata a MQTT brókeren keresztül érkező üzenet fogadása egy megadott témában. Tekinthetjük úgy, hogy a node-red flow így iratkozik fel egy témára. Legfontosabb kimenete, msg.payload amely az üzenetet tartalmazó karakterlánc.

MQTT output node:



Kimenetként az MQTT bróker felé továbbít egy üzenetet egy megadott témával. Tekinthetjük úgy, hogy a node-red flow így publikál egy adott témában. Legfontosabb kimenetei; msg.payload és msg.topic. A node beállításakor a Quality of Service és Retain flag beállítására is ügyelnünk kell.

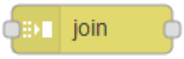
JSON node:

JSON objektumot karakterláncá vagy karakterláncot JSON objektummá alakít. msg.payload attribútumon kívül mást nem módosít.

Function node:

Bármilyen javascript programot tartalmazhat ez a node. Kimenetét return kulcsszóval vagy node.send() függvénnyel küldheti tovább a következő node-nak. A kimeneti üzenet a node-red konvenciónak megfelel, amennyiben a függvény blokk nem állítja a kapott msg attribútumait azokat tovább adja a következő node számára.

Join node:

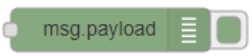


Több node által küldött üzeneteket egyesíthetünk ezzel a node-dal. Beállítása szerint az üzenetek száma, időkorlát vagy msg.complete attribútum bebillentése okozza az addigi üzenetek összekapcsolását. Szintén beállítható az egyesítés módja (tömb, kulcs-érték objektum vagy objektum összefonás).

SQLite node:

Az adatbázissal kapcsolatot teremtő node. A bemenetére érkezett üzenet témáját (msg.topic) kérdezi le az adatbázistól. Az adatbázis válaszát a kimenetre üzenetként továbbítja (msg.payload).

Debug node:

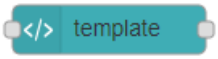


Fejlesztéshez elengedhetetlen, hogy a debug fülön nyomon követhető legyen az adatáramlás. Ez a node a bemenetét a debug fülre továbbítja, illetve időbélyeggel is ellátja azt. A jobb oldalán található zöld gombbal ki/be kapcsolható hatása, a könnyebb fejlesztés érdekében.

Inject node:

Beállításai szerint induláskor vagy periodikusan vagy a fejlesztő felületen megnyomásakor az aktuális időbélyeget küldi ki üzenetként. Debugolásra vagy timer megvalósításra alkalmas node.

UI template node:



A felhasználói felületet UI node-k reprezentálják. A template node bármilyen HTML állomány fogadására képes. A HTML állományban leírt objektum a node-hoz rendelt dashboard elemen jelenik meg.

Chart node:

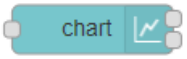
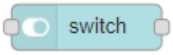


Chart node segítségével egyszerűen hozhatunk létre dinamikusan formálódó kétdimenziós ábrát. A bemenetén kapott Y értékeket automatikusan ábrázolja az aktuális időbélyeg szerint egy kétdimenziós ábrán. A bejövő üzenet témája (msg.topic) külön ábrázolja az adatokat. Állapotát nem őrzi meg, minden újraindítás után üresen indul az ábra. Lehetőség van a Chart.js könyvtár dokumentációja szerinti paraméterezésre, amellyel egyszerre több X-Y pár megadása is lehetséges (pl. régebbi adatok betöltése érdekében). Ezt a lehetőséget a SWAN szerveren futó node-red verzióval viszont nem sikerült kihasználni. [25]

Dashboard Switch node:

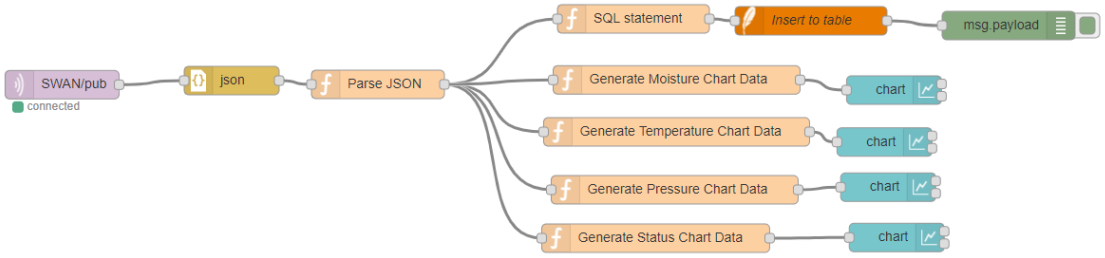


A felhasználói felületen egy kapcsoló jelenik a beállított dashboard elemen. A kapcsolóra való kattintás során a node előre beállított üzenetet továbbít.

4.2.5 SWAN SERVER flow

A szervert megvalósító node-red flow négy különálló részre bontható. Az adatokat fogadó és ábrázoló részre, új kliensek számára sorszámot szolgáltató részre, a felhasználói felületet kezelő részre és az adatbázist karbantartó/lekérdező részre.

A 4-2. ábra az adatokat fogadó és ábrázoló részt megvalósító node-k közötti kapcsolatok tartalmazza.



4-2. ábra Mérési adatok mentése és ábrázolása

MQTT

Feladat:	SWAN/pub témájú üzenetek fogadása.
Beállítás:	server: localhost:1883, QoS2 annak érdekében, hogy legfeljebb egyszer fogadjunk minden üzenetet.

Parse JSON

Feladat:	A hálózaton érkező üzenet JSON objektumot karakterlánc formátumra hozta az előző JSON node. A Parse JSON függvény feladata külön változókba való adatmentés, illetve a bizonyos adatok átlagolása és két számjegyre kerekítése.
Kód:	

```
1  var avgtemp;
2  var avgmoist;
3  var avgpres;
4  var status= msg.payload.status;
5  var id=msg.payload.Client;
6  var battery =
7  msg.payload.Battery_Voltage;
8
   avgtemp = (msg.payload.temp[0]+
   msg.payload.temp[1]+
   msg.payload.temp[2]+
   msg.payload.temp[3]+
9  msg.payload.temp[4])/5;
   avgpres = (msg.payload.pressure[0]+
   msg.payload.pressure[1]+
   msg.payload.pressure[2]+
10 msg.payload.pressure[3]+
   msg.payload.pressure[4])/5;
   avgmoist = (msg.payload.U_moisture[0]+
```

```
            msg.payload.U_moisture[1]+
            msg.payload.U_moisture[2]+
            msg.payload.U_moisture[3]+
            msg.payload.U_moisture[4])/5;
11
12
13  temp = avgtemp.toFixed(2);
14  pres = avgpres.toFixed(2);
15  moist = avgmoist.toFixed(2);
16
17  var data = {payload:
              [id,moist,temp,pres,status,battery]};
18
19  return data;
```

SQL statement (insert payload)

Feladat:	A beérkező adatokból SQL állítás összeállítása.

Kód:

```
1  var topic = " INSERT INTO \"SWAN\"
   (\"ID\"\,\"moisture\"\,\"temperature
   \"\,\"pressure\"\,\"status\"\,\"battery\")
   VALUES ('"+msg.payload[0]+"\'\,\'"+msg.payload[1]
   +"\'\,\'"+msg.payload[2]+"\'\,\'"+msg.payload[3]+
   "\'\,\'"+msg.payload[4]+"\'\,\'"+msg.payload[5]+"\')";
2  var newmsg={}
3  newmsg.topic = topic;
4  return newmsg;
```

Generate Moisture Chart Data

Feladat:	Feszültség és százalék átszámítása a 2.2.2 fejezetben készített karakterisztika alapján. Illetve a Chart node számára szükséges formátum előállítása.

Kód:

```
 1 var moistPercent;
 2
 3 if ((msg.payload[1]-1.7)<0.01){moistPercent = 12.5}
 4 else {//Inverse of y = -0.1173*x +2.901 => 8.52515
   (2.901 - x)
 5     moistPercent = 8.52515*(2.901 - msg.payload[1]);
 6 }
 7
 8
 9 var msg1 = { payload:moistPercent,
   topic:msg.payload[0] };
10
11 return msg1;
```

Generate Temperature Chart Data

Feladat:	Chart node számára szükséges formátum előállítása.

Kód:	`var msg1 = { payload:msg.payload[2], topic:msg.payload[0] };`
	
	`return msg1;`

Generate Pressure Chart Data

Feladat:	Chart node számára szükséges formátum előállítása.

Kód:	`var msg1 = { payload:msg.payload[3], topic:msg.payload[0] };`
	
	`return msg1;`

Generate Status Chart Data

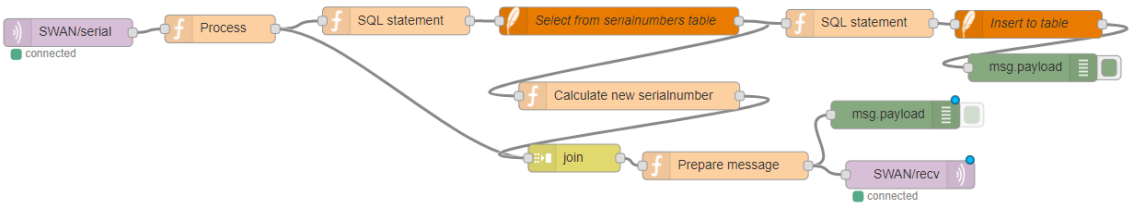
Feladat:	Chart node számára szükséges formátum előállítása.

Kód:	`var msg1 = { payload:msg.payload[4], topic:msg.payload[0] };`
	
	`return msg1;`

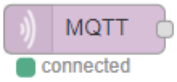
Chart

Feladat:	A felhasználói felületen kétdimenziós ábra megjelenítése.
Beállítás:	X-tengely HH:mm:ss formátummal definiált, az elmúlt egy napban érkezett adatokat, de legfeljebb 1000 pontot ábrázolunk.

A 4-3. ábra a kliensek sorszámát generáló részt megvalósító node-k közötti kapcsolatok tartalmazza.



4-3. ábra Kliensek sorszám generálása



MQTT

Feladat:	SWAN/serial témájú üzenetek fogadása.
Beállítás:	server: localhost:1883, QoS2 annak érdekében, hogy legfeljebb egyszer fogadjunk minden üzenetet.

Process

Feladat:	A beérkezett karakterlánc egy egész számot tartalmaz. Ez a függvény a karakterlánc és egész szám közötti konverziót végzi.

Kód:	`msg.payload = parseInt(msg.payload);`
	
	`return msg;`

SQL statement (Select maxkey)

Feladat:	A függvény feladata, hogy SQL node számára előállítsa a SQL lekérést. newmsg.topic tartalmazza az érvényes SQL állítást. A serialnumbers táblából SELECT utasítással a key oszlop értékeit kérjük le. MAX() függvénnyel módosítjuk a lekérést, ezáltal csak az oszlop legnagyobb értékét kapjuk vissza. A létrejövő MAX(key) javascript objektumot hibás szintaxisú, ezért AS kulcsszóval új nevet adunk neki, maxkey, amely már javascript kompatibilis.

Kód:	1 var topic = "SELECT MAX(key) AS maxkey FROM serialnumbers"; 2 3 var newmsg={} 4 newmsg.topic = topic; 5 return newmsg;

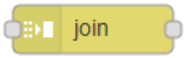
SQL statement (inject newkey)

Feladat:	A beérkező adatokból SQL állítás összeállítása.
Kód:	1 var tempnum = msg.payload[0].maxkey+1; 2 serialnumber = tempnum.toString(); 3 4 5 var topic = " INSERT INTO \"SERIALNUMBERS\" (\"key\") VALUES ('"+serialnumber+"\')"; 6 var newmsg={} 7 newmsg.topic = topic; 8 return newmsg;

Calculate new serialnumber

Feladat:	A lekérdezés a jelenlegi legnagyobb sorszámot adja vissza. Az új sorszám eggyel nagyobb lesz.

Kód:	1 if (msg.payload[0].maxkey === null){
	2 msg.payload = 1;}
	3 else{
	4 msg.payload = msg.payload[0].maxkey + 1
	5 };
	6 return msg;



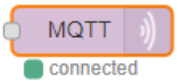
Join

Feladat:	Process és Calculate new serialnumber függvények üzeneteit tömbbe tömöríti.
Beállítás:	Két üzenet beérkezése után üzenet küldése. Üzenet formátuma tömb.

Prepare message

Feladat:	A beérkező tömb két elemét , karakterrel elválasztva karakterláncként tovább küldeni .

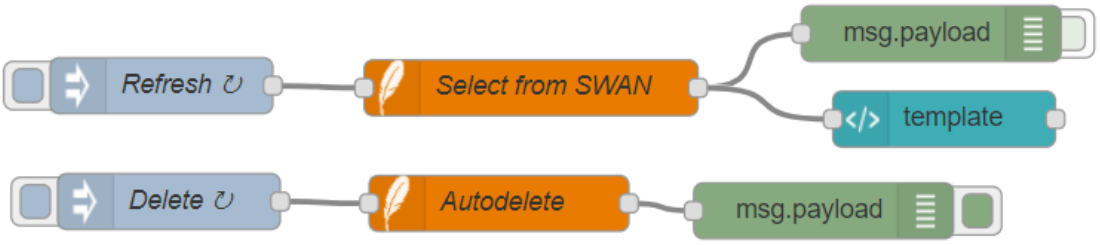
Kód:	`msg.payload = msg.payload[0].toString() + "\," + msg.payload[1].toString();`
	
	`return msg;`



MQTT

Feladat:	SWAN/recv témájó üzenet publikálása. A SWAN SERVER flow-ban több helyen is.
Kód:	server: localhost:1883, QoS1 annak érdekében, hogy legalább egyszer elküldjünk minden üzenetet. Retain:false, mivel fontos elkerülni, hogy egy kliens az energiacsökkentett állapotából visszatérve téves üzenetet kapjon, mert a bróker új feliratkozóként regisztrálja.

A 4-4. ábra az adatbázist karbantartó részt megvalósító node-k közötti kapcsolatok tartalmazza.



4-4. ábra Adatbázis karbantartása

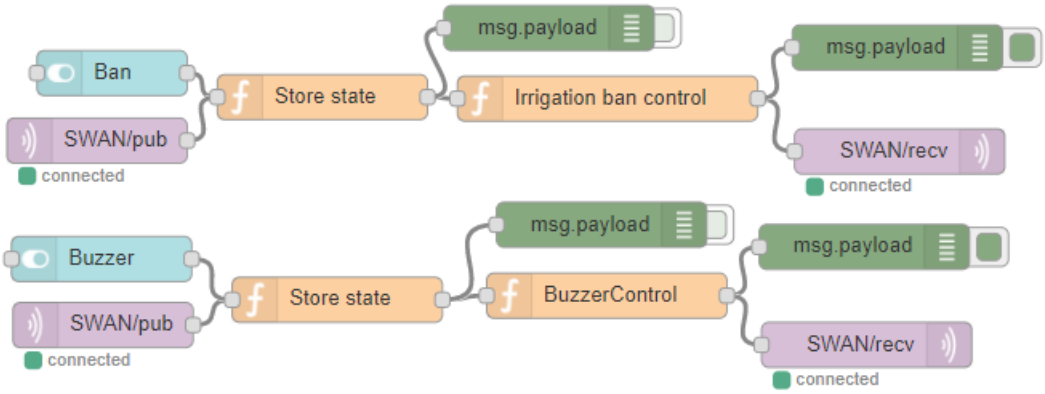
Refresh

Feladat:	Friss adatok lekérdezése percenként.
Beállítás:	Inject at start once? és perces ciklusidő beállítva. Topic: *SELECT * FROM SWAN ORDER BY primkey DESC*

Delete

Feladat:	Óránként a 30 napnál régebbi adatok törlése az adatbázisból.
Kód:	Egy órás ciklusidő beállítva. Topic: *DELETE FROM SWAN WHERE TIMESTAMP <= strftime('%s','now', '-30 days')*1000*

A 4-5. ábra az felhasználó felületet kezelő részt megvalósító node-k közötti kapcsolatok tartalmazza. Az azonos nevű node-k azonos funkciót látnak el, ezért a már korábban bemutatott node-okat nem kerülnek újra bemutatásra.



4-5. ábra Felhasználói felület kezelése

Store state

Feladat:	A switch node állapota nem lekérdezhető, ezért valamilyen javascript nyelvi eszközzel minden állapot változást követni kell. Erre a kontextus a megfelelő eszköz. Ez a függvény elmenti a switch állapotát, majd minden futása során a kimeneten továbbítja azt.

Kód:	`var state = context.get('state') || false;`
	
	`if (msg.topic == 'switch') {`
	`  state = msg.payload;`
	`  context.set('state', state);`
	`}`
	
	`return {payload: state};`

Irrigation ban control

Feladat:	A felhasználói felületen található kapcsoló állapotáról az éppen a hálózatra jelentkező kliensnek tudnia kell. Ez a függvény a kapcsoló állapota szerinti karakterláncot továbbít az MQTT node-nak.

Kód:	`if (msg.payload === true){`
	`  msg.payload = "banned";`
	`}`
	`else {`
	`  msg.payload = "enabled";`
	`}`
	
	`return msg;`

Buzzer Control

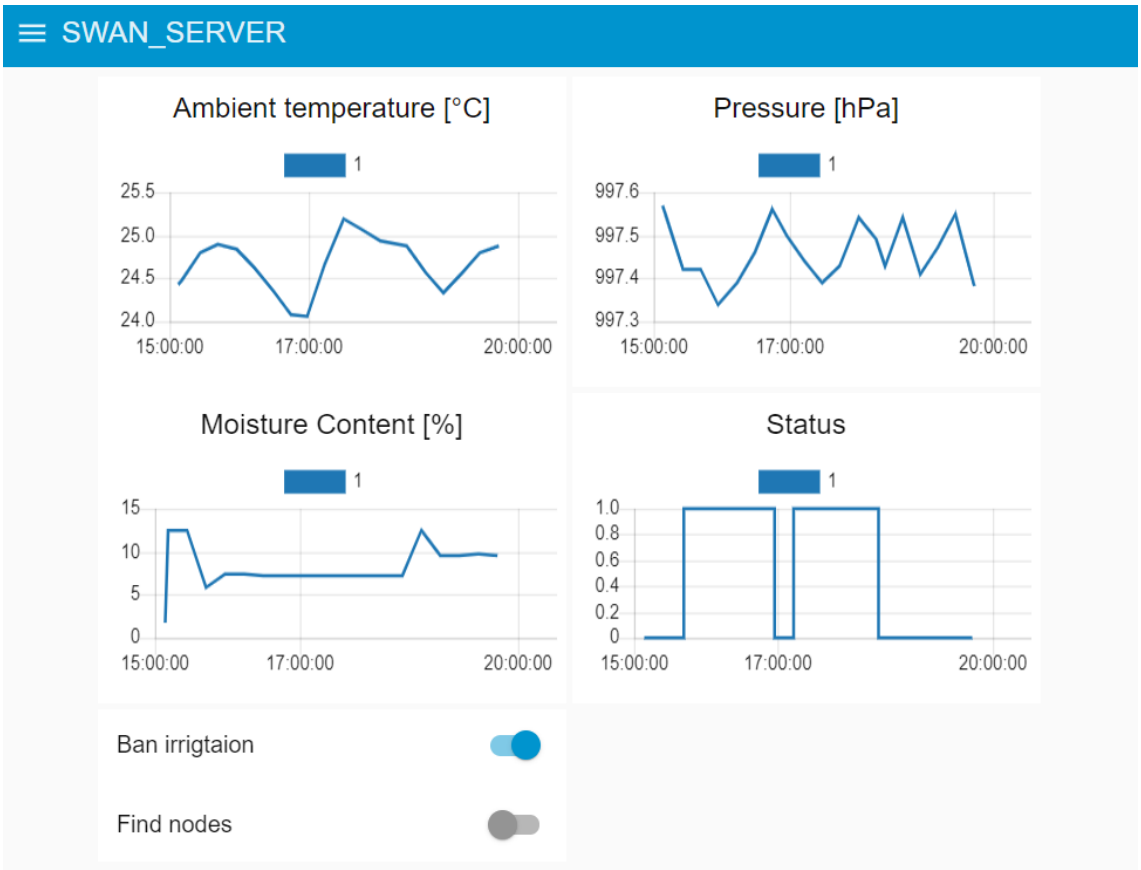
Feladat:	A felhasználói felületen található kapcsoló állapotáról az éppen a hálózatra jelentkező kliensnek tudnia kell. Ez a függvény a kapcsoló állapota szerinti karakterláncot továbbít az MQTT node-nak.

Kód:	`if (msg.payload === true){`
	`  msg.payload = "lost";`
	`}`
	`else {`
	`  msg.payload = "found";`
	`}`
	`return msg;`

4.2.6 Felhasználói felület

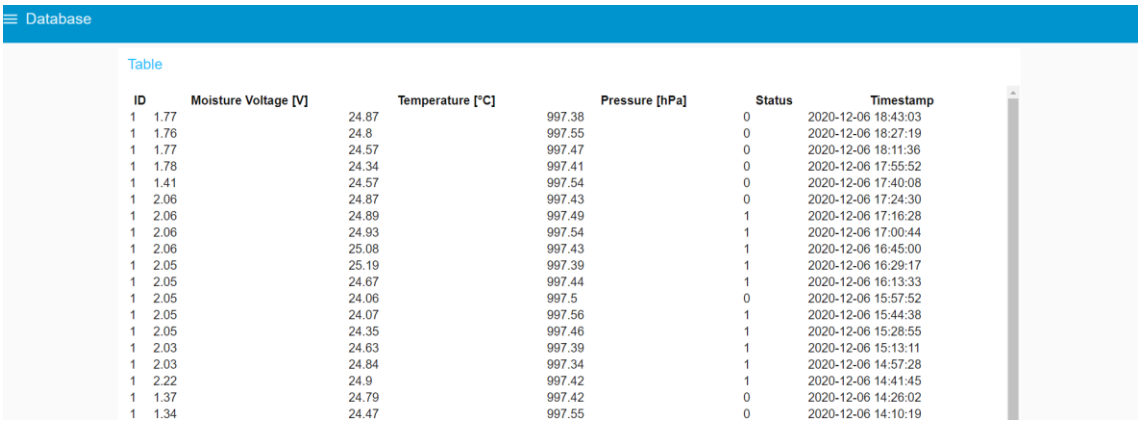
A felhasználói felületet a helyi hálózatról bármilyen böngészőről el elehet érni, felhasználónév és jelszó segítségével (http://192.168.0.202:1880/ui/#/0).

A felhasználót a 4-6. ábraán látható weboldal fogadja, melyről a bal felső sarokban található gomb segítségével tovább navigálhat a 4-7. ábraán látható oldalra.



4-6. ábra Felhasználói felület (1. lap)

A felhasználói felület grafikonjai a szerver indítása óta érkezett adatokat vizualizálja. Viszont legfeljebb az elmúlt 1 nap vagy 1000 adatot jeleníthet meg egy grafikon. Az oldalon található kapcsolókkal tilthatjuk az öntözést vagy kapcsolhatjuk be kliensek keresését megkönnyítő hangjelzést.



4-7. ábra Felhasználói felület (2.lap)

A második lapon egy olyan táblázat található, amely az adatbázisban tárolt összes információt dinamikusan listázza. Időbélyeg alapján csökkenő (legújabb elől) sorrendben listázza az információkat a táblázat.

5 Konklúzió

5.1 Eredmények

A rendszer működése az előírtaknak megfelelő és szoftveres tesztelése során sem jelentkezett hibás működésre utaló jel. A kapacitív talaj nedvességtartalom mérés megvalósult és az alacsony fogyasztásúvá tett szenzor megbízható adatokat szolgáltatott. A létrehozott kliens egység kis számban szenzorhálózatba építhető.

A rendszer alkalmas automatizált módon az öntözést biztosítani, akár egy mérési pont segítségével is. Mind a beágyazott, mind a szerveren futó szoftver alkalmas rugalmas bővítésre, mellyel növelhető a felhasználó kényelme.

A kapacitív talaj nedvességtartalom méréssel kapcsolatos kísérlet működő karakterizáló módszer, ennek segítségével más talajtípusok is vizsgálhatók.

5.2 Tovább fejleszthetőség

Mindenképpen lehetőség van további hardveres változtatások mentén a kliens fejlesztésére. Az akkumulátor költsége és helyigénye nagy, ezért érdemes lenne ezen a területen fejleszteni. A megfelelő csatlakozók az előző fejezetekben tárgyaltak miatt még nincsenek beépítve, ennek a problémának megoldása is további fejlesztési pont.

Szoftveres fejlesztési lehetőségek közé tartozik, további fogyasztáscsökkentő megoldások használata (pl. feszesebb időzítések, debug üzenetek elhagyása) vagy új funkciók hozzáadása. Kifejezetten fontos további szoftveres funkció lenne, ha a felhasználó a szerveren keresztül állíthatná a kliensek működését és egyedi beállításokkal hangolhatná a teljes rendszert saját kedvére. További szoftveres fejlesztési pont, hogy a szerver több felhasználóval és nem csak a helyi hálózatról biztonságosan elérhető legyen.

A kapacitív talaj nedvességtartalom mérés területén is érdemes még vizsgálódni. Elsősorban különböző talajtípusok hatásának karakterizálása és a szenzor geometriai tulajdonságainak módosítása segítené a működést.

A továbbfejlesztés nyomon követhető GitHub oldalon keresztül, ahol a projekttel kapcsolatos összes releváns információ rendszerezett formában tárolva van. [26]

6 Irodalomjegyzék

[1] KSH, A lakott lakások lakóövezeti jelleg szerint (2001)

http://www.ksh.hu/nepszamlalas/docs/tablak/lakasviszonyok/12_02_28.xls (2020.12.06)

[2] Önálló laboratórium projekt

https://github.com/Zaion-BM/SWAN

[3] Kelleners, T.J.; Soppe, R.W.O.; Robinson, D.A.; Schaap, M.G.; Ayars, J.E.; Skaggs, T.H. : Calibration of Capacitance Probe Sensors using Electric Circuit Theory
https://acsess.onlinelibrary.wiley.com/doi/epdf/10.2136/sssaj2004.4300 (2020.12.06)

[4] Pisana Placidi , Laura Gasperini, Alessandro Grassi, Manuela Cecconi & Andrea Scorzoni : Characterization of Low-Cost Capacitive Soil Moisture Sensors for IoT Networks
https://www.researchgate.net/publication/342450119_Characterization_of_Low-Cost_Capacitive_Soil_Moisture_Sensors_for_IoT_Networks (2020.12.06)

[5] Mágnesszelep működése
https://en.wikipedia.org/wiki/Solenoid_valve (2020.10.15)

[6] IPC2221-B 6-1. táblázat
https://www.smpspowersupply.com/ipc2221pcbclearance.html (2020.12.06)

[7] ESP32 Dev Kit C Schematic
https://dl.espressif.com/dl/schematics/esp32_devkitc_v4-sch.pdf (2020.12.06)

[8] MCP6273T-E/CH adatlap
https://hu.mouser.com/datasheet/2/268/21810e-76316.pdf (2020.12.06)

[9] Témalabor dokumentáció
https://github.com/Zaion-BM/Temalabor

[10] Varjasi István, Balogh Attila, Futó András, Gájász Zoltán, Hermann Imre, Kárpáti Attila : ELEKTRONIKA 2

[11] Nyomás és csapadék kapcsolata
https://physics.stackexchange.com/questions/65199/high-and-low-pressure-area-and-raining (2020.12.06)

[12] BMP280 belső felépítése
https://startingelectronics.org/pinout/GY-BMP280-pressure-sensor-module/ (2020.10.15)

[13] ESP32 WROOM-32 adatlap
https://www.espressif.com/sites/default/files/documentation/esp32-wroom-32_datasheet_en.pdf (2020.10.10)

[14] UniPCB árlista
http://www.unipcb.hu/prices.html (2020.12.06)

[15] ESP32 core az arduino fejlesztőkörnyezethez
https://github.com/espressif/arduino-esp32 (2020.12.07)

[16] FreeRTOS az ESP32-n
https://docs.espressif.com/projects/esp-idf/en/latest/esp32/api-reference/system/freertos.html (2020.12.07)

[17] MQTT kliens könyvtár arduino platformra
https://pubsubclient.knolleary.net/api (2020.12.07)

[18] Raspberry Pi Zero W
https://www.raspberrypi.org/products/raspberry-pi-zero-w/?resellerType=home (2020.12.07)

[19] DietPi szoftver dokumentáció
https://dietpi.com/dietpi-software.html (2020.12.07)

[20] Mosquitto MQTT bróker dokumentáció
https://mosquitto.org/ (2020.12.07)

[21] Relációs adatbázisok
https://vik.wiki/images/f/f0/Info2_Adatb%C3%A1zisok_jegyzet_%28r%C3%A9gi%29.pdf (2020.12.07)

[22] SQLite dokumentáció
https://www.sqlite.org/docs.html (2020.12.06)

[23] Node-RED dokumentáció
https://nodered.org/docs/ (2020.12.07)

[24] Node JS API referencia
https://nodejs.org/en/docs/ (2020.12.07)

[25] Scatter chart API referencia
https://www.chartjs.org/docs/latest/charts/scatter.html (2020.12.07)

[26] Szakdolgozat GitHub oldala
https://github.com/Zaion-BM/SWAN_BSc_Version (2020.12.07)

7 Függelék

Költség összegzés táblázat

Alkatrész	Ár [HUF]	Darabszám [db]	Végösszeg [HUF]
SMD capacitor 0805	9,99	16	159,84
SMD resistor 0603	9,99	21	209,79
SMD LED 0603	51	1	51
ACCU_ICR18650_26JM	3098	1	3098
BHC-18650-1P	749,3	1	749,3
MCP73811T-420I/OT IC	200	1	200
MCP1826S-3302E/DB IC	266,9	1	266,9
MC3572-1220-123 REED RELAY	1964,4	1	1964,4
SP0505BAJTG DIODE	463,8	2	927,6
PIEZOELECTRONIC BUZZER,	301	1	301
CNY17-1 OPTOCOUPLER	281,9	2	563,8
Analog Capacitive Soil Moisture Sensor V1.2 Corrosion Resistant With Cable	300,7	1	300,7
ESP-WROOM-32	1368,7	1	1368,7
New BMP280 Digital Sensor Temperature Board	375,3	1	375,3
MCP6273-E_CH	172,3	3	516,9
FDN302P	160	1	160
Push Button	9,2	2	18,4
Male Pin Header 1x40	242,4	1	242,4
2N2222 NPN transistor	101,6	1	101,6
PCB	1000	1	1000
400C5 150x110x75mm IP55	1460	1	1460

SallenKeyFilter.m

```
1 %% Sallen-Key 2nd order Low Pass Filter transfer function
2
3     %Parameters
4     R10 = 1.E3;
5     R21 = 1.E4;
6     C10 = 1.E-11;
7     C9 = 1.E-11;
8
9     %Transfer function
10    W = tf((1/(R10*C10*R21*C9)),[1, 1/(R21*C9)+1/(R10*C9), ...
11        1/(R10*C10*R21*C9) ]);
12
13    %Bode diagram solely with amplitude
14    h = bodeplot(W);
15    setoptions(h,'FreqUnits','Hz','PhaseVisible','off');
```