Széchenyi István Egyetem
Gépészmérnöki, Informatikai és
Villamosmérnöki Kar

DIPLOMAMUNKA

Pataki Péter
Villamosmérnöki MSc

2022

SZÉCHENYI EGYETEM
UNIVERSITY OF GYŐR

SZÉCHENYI ISTVÁN EGYETEM · GYŐR
TÁVKÖZLÉSI TANSZÉK

DIPLOMAMUNKA

Országos spektrum monitoring hálózat központi menedzselése

Pataki Péter

Villamosmérnöki MSc szak
Infokommunikáció szakirány

2022.

1/b. melléklet: Feladat-kiíró lap diplomamunkához

Feladat-kiíró lap diplomamunkához

Hallgató adatai
Név: Pataki Péter *Neptun-kód: WV964G*
Szak: Villamosmérnök MSc
Specializáció: Infokommunikáció *Tagozat:* levelező

A diplomamunka adatai
Kezdő tanév és félév:
Nyelv:
Típus: nyilvános

Országos spektrum monitoring hálózat központi menedzselése

Feladatok részletezése:

1. Vizsgálja meg a spektrummonitoring rendszer felépítését és annak menedzselhetőségét. Térjen ki az alkalmazható és használható protokollokra a vizsgálata során.

2. Készítsen szoftveres megoldást, mely az országos spektrummonitoring rendszer kiegészítő mérőállomásait integrálhatóvá teszi az országos rendszerbe.

3. Mutassa be az elkészített szoftver működését, használhatóságának peremfeltételeit.

Belső konzulens adatai
Név: Dr. Vári Péter
Tanszék: Távközlési Tanszék
Beosztás: egyetemi docens

Külső konzulens adatai
Név: Bozi Milán Pál
Munkahely: Nemzeti Média- és Hírközlési Hatóság
Beosztás: Spektrum monitoring mérnök

Győr, 2022. 05. 08.

belső konzulens

külső konzulens

Távközlési Tanszék
Dr. Borbély Gábor]

Nyilatkozat

Alulírott, Pataki Péter (WV964G), villamosmérnök MSc szakos hallgató kijelentem, hogy az Országos spektrum monitoring hálózat központi menedzselése című diplomamunka feladat kidolgozása a saját munkám, abban csak a megjelölt forrásokat a megjelölt mértékben használtam fel, az idézés szabályainak megfelelően, a hivatkozások pontos megjelölésével.

Eredményeim saját munkán, számításokon, kutatáson, valós méréseken alapulnak, és a legjobb tudásom szerint hitelesek.

Győr, 2022. május 11. _____

hallgató

Kivonat

Diplomamunkámban bemutatom azt a rendszert, amit a Nemzeti Média- és Hírközlési Hatóság országos mérőhálózatának kiegészítő mérőállomásaihoz fejlesztettem ki. Programom grafikus módon összehangoltan képes több mérőállomás mérési feladatainak kiadására, összehangolására és azok üzemi paramétereinek lekérésére.

A diplomamunka első részében bemutatom mindazon protokollokat és alapvető ismereteket, amik szükségesek a későbbi témák megértéséhez. Ezután bemutatom részletesen a probléma felvetését, hogy miért volt szükséges és miért hasznos a munkámnak eredménye. Kitérek program felépítésére, részegységeire és arra, hogy miért esett választásom az adott megoldásra.

A dolgozat végén rövid összefoglaló keretében egy valós mérés során bemutatom a rendszer sikeres működését és kiértékelem annak eredményét.

Abstract

In this thesis I present my system which can control the supplementary measuring stations of the Hungarian measuring network operated by the Nemzeti Média- és Hírközlési Hatóság. This program is able to make measurement tasks of several measuring stations and it can download their operating parameters.

In the first part of my thesis I present all of the protocols and basic knowledges what needed to understand the later topics. I present the details of the problem and I explain why this system is useful. I will cover the structure of the program, its components and why I chose the given solution.

At the end of the thesis I will summarize the experiences and the results. I will present the successful operation of the system during a measurement and evaluate its results.

Tartalomjegyzék

1. Bevezetés

A Nemzeti Média- és Hírközlési Hatóság mérésügyi feladatai közé tartozik Magyarország teljes rádiófrekvenciás megfigyelése és ellenőrzése. Ennek a szolgálatnak segítésére az országban összesen 69 fixen telepített mérőállomás szolgál: 15 darab teljes kiépítettségű, 16 darab zavarkeresést támogató és 38 darab kisebb kiépítettségű, úgynevezett kiegészítő mérőállomás. Minden mérőállomás egy komplex rendszer része, de önállóan, úgymond standalone módon is képesek üzemelni. A dolgozat témáját az utóbb említett 38 darab kiegészítő állomás adja, aminek megoldandó a központi konfigurációja eseti mérések elvégzése érdekébe.

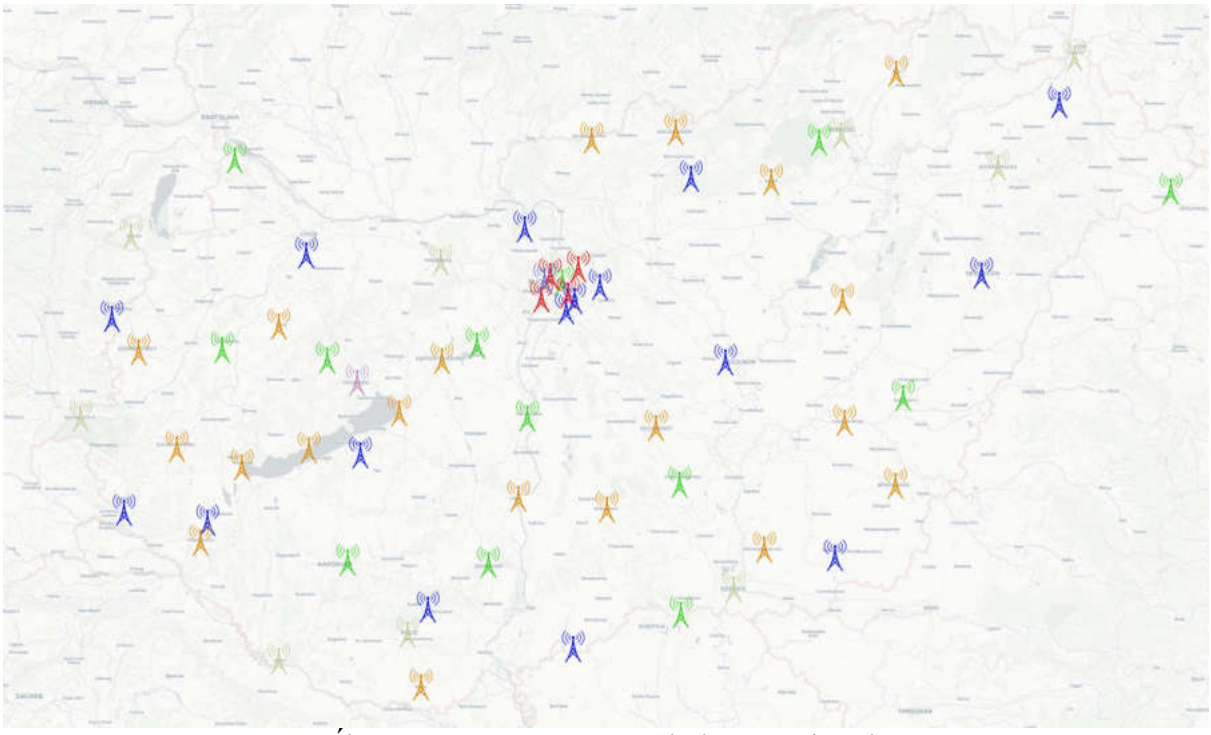
A mérések összeállítása kézileg történik JSON fájlok megfelelő mezőinek módosításával. Felmerült a probléma, hogy több paraméter módosítása számítás és munka igényes, így érdemes lenne automatizálni a folyamatot. Dolgozatom egy általam fejlesztett automatizált kezelőfelületnek bemutatásáról szól, valamint arról, hogy milyen módon juttathatjuk el a létrehozott konfigurációt a mérőállomásra és tekinthetjük meg annak sikeres eredményeit. Külön kitérek a használt protokollokra, illetve a modulok használatára a pontos megértés érdekébe. A modulárisan felépített program bemutatása után a Dolgozat végén egy teszt méréssel mutatom be ennek működőképességét és hatékonyságát.

Ezen a ponton szeretnék köszönetet mondani konzulensemnek és mentoromnak, Dr. Vári Péternek a témajavaslatért és a rengeteg segítségért amivel támogatta a kutatási munkám. Köszönettel tartozom továbbá a Nemzeti Média- és Hírközlési Hatóság Rádiómonitoring Osztályéról Egyed Péternek, Bodzsár Krisztiánnak és Bozi Milánnak, akik lehetővé tették számomra a mérőhálózaton való tesztelést és segítettek a programomat minél hatékonyabbá tenni.

2. Mérőhálózat bemutatása

A Nemzeti Média- és Hírközlési Hatóság kezelésében álló mérőhálózat jelenleg 2022 februárjában 69 darab fixen telepített mérőállomással rendelkezik. Ezek telepítése különböző szempontok szerint valósult meg: a teljes kiépítettségű állomások általában olyan helyeken állnak, ahol a rádiófrekvenciás vétel és iránymérés nem akadályozott tereptárgyak által és a rádiófrekvenciás terjedés mesterségek zavaroktól mentes. A kiegészítő mérőállomások lakott területekre lettek installálva a helyi rádiófrekvenciás mérések elvégzésére. Ezek gyakran magas épületek tetején helyezkednek el a teljes kiépítettségű állomásokhoz képest szerényebb antennarendszerrel és mérési képességgel.

Az 1. ábra Magyarország mérőhálózatának térképét mutatja be külön színekkel jelölve a különböző típusú mérőállomásokat. Megfigyelhető, hogy a kék színnel jelölt teljes kiépítettségű állomások többnyire az országhatár közelében helyezkednek el, ahol lehet magaslaton vagy hegycsúcson. A zavarkeresést segítő, úgynevezett VÉDA állomások zöld színnel, míg a kiegészítő mérőállomások – típustól függően – piros, narancssárga vagy keki zöld színben jelennek meg.

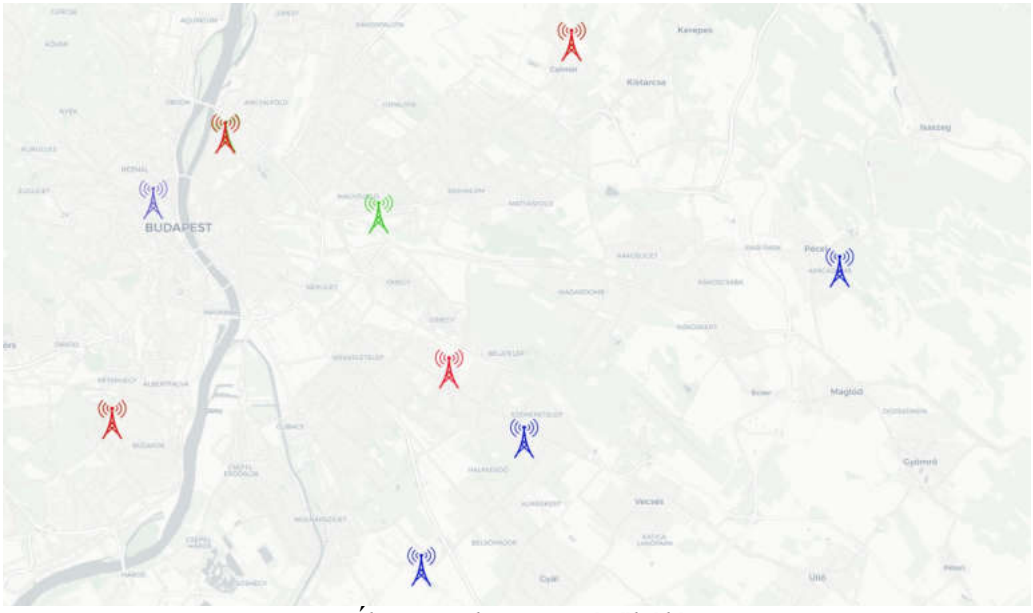


1. Ábra: Magyarország mérőhálózatának térképe

A térkép jelölései csoportonként:

2. Ábra: Térkép jelölések

Mérési mennyiség tekintetében Budapest kiemelt helyen szerepel. Mivel a leggyakrabban a fővárosban és annak környékén szembesül a hatóság zavarokkal esetleg kalózrádiókkal így ennek területe és környezete sűrűbben van ellátva mérővevőkkel, mint ahogy az a 3. ábrán is látszik.



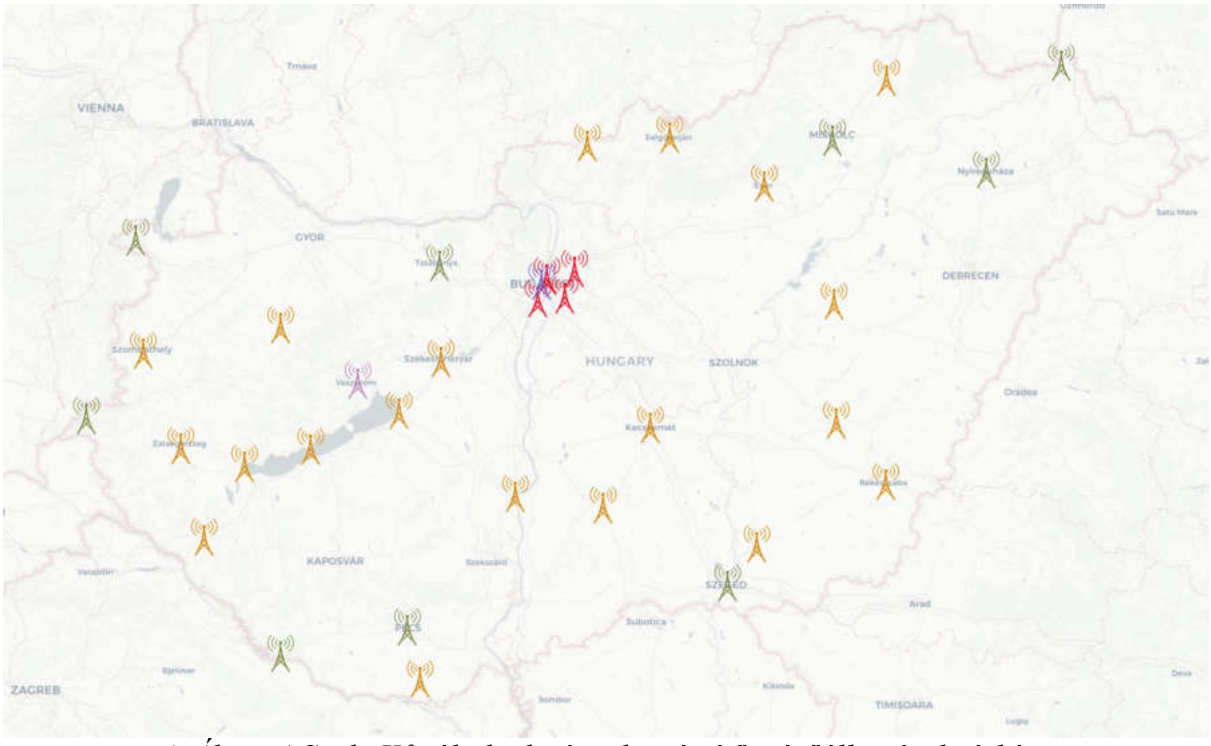
3. Ábra: Budapesti mérőhálózat

A térkép jelöléseinek magyarázata a 2. ábrán láthatóak.

A mérőhálózat eszközeinek sokszínűsége szükségessé tesz egy olyan vezérlési eljárás kidolgozását, aminek köszönhetően egységesen egy felületről lehet több mérőállomásnak

kiadni mérési utasításokat anélkül, hogy bonyolult fájlszerkezeteket kézzel kelljen módosítani, majd manuálisan feltölteni az állomás számítógépén futó logger alkalmazás számára.

Mivel a teljes kiépítettségű mérőállomások kezelése az NMHH által fejlesztett SIMI mérésvezérlő szoftveren keresztül történik, így a Svale Kft. által forgalmazott kiegészítő mérővevők konfigurálásának megkönnyítése volt a rendszerem célja, a továbbiakban ezzel foglalkozunk részletesebben. A térképre szerint a következő helyszíneken állnak rendelkezésre ezek a típusú mérőeszközök:



4. Ábra: A Svale Kft. által telepített kiegészítő mérőállomások térképe

3. Mérőrendszer felépítése

A kiegészítő mérőállomások felépítése telephelyenként változhat. Több állomáson csak körsugárzó antennák segítségével végezhető a mérés, de néhány helyen előfordulnak forgatható, irányított antennákkal felszerelt mérőpontok is, mint például Miskolcon (6. ábra). Az azimuth és elevációs irányba mozgatható antennák ellenére a mérési képességek nem változnak, csupán az egyes irányokból érkező jelek koncentrált mérése és irányban szelektálása valósítható meg ezáltal a konstrukció által.

5. Ábra: Kiegészítő mérőállomás körsugárzó antennákkal Nagykanizsa belvárosában

A mérőrendszer lelkét képező monitoring vevőket két csoportba soroljuk: a Marvell és a Nexus típusúakra. A kettő típust a mérési képességeik különböztetik meg, illetve üzembeállításuk ideje. A Marvell típusúak korábban lettek beszerezve mint a Nexus-ok, így azok szerényebb, bár hasonlóan hatékony képességekkel rendelkeznek, mint az újabb generációs társaik.

6. Ábra: Kiegészítő mérőállomás forgatható irányított antennával Miskolcon

A Marvell típusú mérővevőkben Marvell 88F6281 típusú, 1 GHz órajelű processzor dolgozik míg a Nexus-okban Intel E3845 4 magos, 1,91 GHz-es. Mindkettőről közösen elmondható, hogy a gigabites hálózati adapterükön keresztül PoE táplálásra alkalmasak, ezáltal a rendszerbe integrálhatóságuk és helyigényük jelentősen csökkenthető adapteres társaikhoz képest. A távoli elérés és mérési adatok mozgatása is megoldott ugyan ezen a vezetéken keresztül.

A mérővevők komplett számítógépként is képesek funkcionálni, mindazonáltal, hogy egy Debian Linux alapú operációs rendszer fut rajtuk a mérések elvégzéséhez optimalizált csomagkészlettel. Az adatok belső tárolása SSD tárolókra van megoldva.

A rádiófrekvenciás mérések Marvell állomás esetén 10 MHz és 6 GHz között biztosítottak 20 MHz folyamatos sávszélesség mellett 1 Hz-es hangolási felbontással. A Nexus mérővevők 9 MHz és 8 GHz között képesek méréseket végezni 40 MHz folytonos sávszélességgel és 1 MHz hangolási felbontással.

A kiegészítő mérőállomásokon használt antennák a meghatározott mérési frekvenciatartományok kiszolgálására lettek optimalizálva: minden állomáson található alsósávi és felsősávi mérőantenna. Ezek pontos típusa állomásonként változhat a felrögzítési konzol és a mérőállomás elhelyezkedése függvényében.

Bizonyos kiegészítő állomások, mint a 6. ábrán szereplő Miskolci mérőállomás rendelkeznek forgatható logperiodikus antennákkal. Ezek célja a különböző irányokból érkező jelek detektálása és az irányuk megközelítőleges meghatározása.

4. Kommunikációs és hálózati protokollok

A mérőállomásokkal való kapcsolattartás érdekében többféle protokollt használunk a kommunikációra. Ezek nagy része valamilyen transfer protokoll adatok mozgatására vagy buszrendszer parancsok kiadására. A kommunikációs protokollok közül kimondottan azokat mutatom be, amik szükségesek a konfiguráló felület működéséhez, illetve annak használatához. Mivel a különböző kiépítettségű állomásokon más-más kommunikációs eljárás van jelen, így összegyűjtöm azokat a minél teljesebb és pontosabb megértés érdekébe. Az egyes protokollokra a későbbi fejezetekben hivatkozom különkülön is.

4.1. TCP

A TCP – angolul Transmission Control Protocol – egy byte-stream alapú kapcsolatorientált full-duplex protokoll. Megbízhatósága a biztonságosan kiépített kapcsolatban rejlik, mely a szerver és a kliens közt mindaddig jelen van, ameddig a kapcsolat fennáll és használatban van.

A kommunikáció során a két fél között úgynevezett csomagok (más néven datagramok) segítségével történik az információ átadás, amik egy vagy több byte-folyam szétdarabolt, 64 kB méretet meg nem haladó egységei. Ezeket a csomagokat ültetjük Ethernet keretekbe. Mikor egy TCP fejléccel ellátott datagram érkezik a feldolgozó szoftver részére (ami gyakran a kernelben foglal helyet), akkor az onnan a TCP-entitáshoz kerül és ennek feladata visszaállítani az eredeti adatfolyamot a bájt töredékekből. [1.] A TCP fejlécben jelölt portszám alapján az a program kapja meg a csomagot, ami a jelölt számú porton „hallgat”. A forrás és célportok ismerete lehetővé teszi egy időben több TCP kapcsolat kiépítését is más-más portokon. A TCP fejléc felépítése egy meghatározott rendszer alapján történik, amit az 1. táblázat mutat be.

Bit	1. bájt		2. bájt	3.bájt	4. bájt
0	Forrásport			Célport	
32	Datagram (szekvencia) sorszáma				
64	Elfogadási nyugta szám				
96	Adat offset	Fenntartott bitek	Állapot bitek	Ablak méret	
128	Ellenőrző összeg			Fontossági mutató	
160	Opciók (ritkán használt)				Kitöltés
192	Adat kezdete				

1. Táblázat: A TCP fejléc szerkezete [1.]

A full-duplex működés biztosítása érdekében a kapcsolat feleinek szinkronizálni kell az átviteli sebességet minél pontosabban, hogy ne történhessen meg adatvesztés. Megfelelő szinkronizálás esetén előfordulhat az a jelenség, aminek során a nagyobb sebességre képes eszköz a kisebb sebességű számára olyan gyorsan szolgáltat adatot, amit az nem képes feldolgozni és így az az úgynevezett elárasztási állapotba kerülhet. Ennek megfelelően a TCP kapcsolat a felépülés előtt és alatt képes figyelni az effektív sávszélességet[1] és ez alapján optimalizálni a megfelelő átviteli sebességet.[27.]

4.1.1. Minta egy TCP csomagra:

```
44 55 4d 4d 59 2d 54 1a f9 91 ba 0d 08 00 45 00
00 37 aa b0 40 00 80 06 8e 53 ac 1e 00 ca 0a c8
0a 0d 6f 04 27 11 48 22 a3 98 1a 79 85 c9 50 18
ff 7c e6 dc 00 00 4d 4b 20 31 20 53 54 41 54 55
53 20 52 0d 0a
```

2. Táblázat: Egy TCP datagram felépítése

1 Az effektív sávszélesség az a sávszélesség, ami szükséges egy kapcsolatban ahhoz, hogy a minőségi paramétereket minél inkább a garantáltan elfogadható tartományon belül tartsuk. Ennek az értéke az adott forgalom csúcs és átlag sávszélessége közötti tartományban helyezkedik el. Ezt a minőséget nagyban befolyásolja a generálódott forgalom paraméterei, a másik linken lévő forgalmak tulajdonságai, a link jellemzői és a felállított minőségi paraméterek [12.]

A hexadecimális értékek értelmezését kezdjük az alapoktól: a bájtsorozat egy 69 bájtos, vagyis 552 bites Ethernet keret.

- A piros háttérszínnel jelölt bájtok magát az Ethernet fejlécet jelölik. Ebben adjuk meg a cél -és forrás eszköz MAC címeit, illetve azt, hogy a kapcsolat milyen típusú lesz.
- Zöld szín mutatja az IPv4 fejlécet. Itt találhatók meg többek közt olyan fontos információk, mint a feladó és cél eszköz IP címe.
- Kék háttérrel a TCP csomagok fejléce látható, ami az 1. táblázatban foglalt adatokat tartalmazza. Kicsit ponotsabban kielemezve:
  - 6f 04: Forrásport (28420)
  - 27 11: Célport (10001)
  - 48 22 a3 98: Datagram sorszáma (1210229656)
  - 1a 79 85 c9: Nyugta száma (444171721)
  - 50: Fejléc hossza (20 bájt)
  - 18 Állapotbitek (részletezése: [28.])
  - ff 7c: Ablak méret (65404)
  - e6 dc: Ellenőrző összeg
  - 00 00: Fontossági mutató (0)
- Sárga szín jelöli a TCP csomagba küldött adatot

Az adatok mezőben bármilyen szabadon választott adat lekódolható. A mérőállomások esetében gyakran vezérlési parancsok kiadása történik ezen a protokollon keresztül, de például a valós idejű mérést feldolgozó és vezérlő szoftverek is ilyen módon kommunikálnak a mérővevővel.

4.2. UDP

Az UDP – User Datagram Protocolt – egy kapcsolat nélküli datagram szállítási protokoll, mely az IP-re épülve – a TCP-hez hasonlóan – biztosítja a portok használatát a programok számára. A TCP-vel ellentétben az UDP nem épít ki biztonságos kapcsolatot a szerver és kliens között, így ennek használata során nem garantált a csomagok célba jutása.

Az UDP kapcsolatok nagy előnye a TCP-hez képesti gyengeségben rejlik: mivel nem épít ki kapcsolatot a két fél között, így jelentős időt megspórol a kommunikáció során a TCP-nél használt újraküldési mechanizmus nélkül. Az így létrejövő kapcsolat kvázi valós idejű, egyetlen kérésre egyetlen választ adó alkalmazások esetén használatos, ahol fontos a gyors átvitel, de nem szempont, hogy minden csomag pontosan célba érjen.

Az UDP 8 bájtos fejléccel rendelkező szegmenseket használ:

Bit	**1. bájt**	**2. bájt**	**3.bájt**	**4. bájt**
0	Forrásport		Célport	
32	Szegmens hossz		Ellenőrző összeg	
64	Adat kezdete			

3. Táblázat: A TCP fejléc szerkezete [1.]

- 1-2 bájt a forrásport meghatározása
- 3-4 bájt a célport meghatározása
- 5-6 bájt a szegmens hossza
- 7-8 bájt ellenőrző összeget képvisel.

Ezt követik a felhasználói adatok.

Az UDP datagramok feldolgozása során a szállítási réteg a port szám alapján dönti el, hogy pontosan melyik folyamat számára kell továbbítani az adatokat. A forrásport első sorban abban az esetben lényeges, ha választ is várunk a csomagunkra.

A legkisebb szegmenshossz 8 bájt lehet, leghosszabb 65515 bájt. A legkisebb esetén csak maga a fejléc továbbítása történik meg. A legnagyobb szegmenshossz meghatározásánál határt szab az IP által maximálisan továbbítható adatmennyiség, ami az IPv4 szabvány esetén

65535 bájt lehet maximálisan fejlécekkel együtt, aminek kiszámítása a 16 bites rendszerben 2^{16}-1 = 65535 bájt [16.].

4.3. SSH

Az SSH-t (Secure SHell) számítógépek közötti biztonságos adatkapcsolat létrehozására alkotta meg Tatu Ylönen finn informatikus 1995-ben [23.]. A protokoll kliens-szerver architektúrán működik. Egy szerverhez – a beállításoktól függően – akár több kliens is kapcsolódhat és adhat ki parancsokat annak parancsértelmezőjére. A protokoll további nagy előnye, hogy képes adatokat tömörítetten átvinni, ezáltal csökkenteni a használt sávszélességet vagy épp kapcsolatokat forwardolni. Utóbbi segítségével megoldható, hogy VPN kapcsolat megléte nélkül továbbítsunk adott portokat a kiszolgáló és távoli eszköz között biztonságos módon.

A szerver alapértelmezett kommunikációs portja a 22-s TCP port. A kapcsolat abban az esetben jön létre, ha a gépek rendelkeznek megfelelő titkos (private) és nyilvános (public) kulcsokkal. Kulcs alapú autentikáció esetén nélkülözhetetlen a felhasználónkénti titkos és nyilvános kulcspár.

Az SSH kapcsolatok kiépítése a következő sorrendben történik [15.]:

1. Titkosított csatorna létrehozása a kiszolgáló és kliens között.
2. A kliens azonosítása titkosított módon.
3. Kiépül a titkosított kommunikáció.

Az első pontban jelzett titkos kapcsolat kiépítése csak abban az esetben valósulhat meg, ha a kliens ismeri a kiszolgáló szerver nyilvános kulcsát. Enélkül az előfeltétel nélkül a kapcsolat nem épülhet ki biztonságosan, sebezhetővé válik a kommunikáció. Amint – az ismert kulcsokkal – kiépült a csatorna, a szerver generál egy úgynevezett kapcsolatkulcsot, ami addig él, ameddig a kapcsolatot le nem zárja az egyik oldal. Ez a kapcsolatkulcs általában valamilyen szimmetrikus kulcsú algoritmussal generálódik, mint például a CAST128, 3DES, SHA, blowfish vagy DSA.

A rendszer biztonsága sebezhetővé válik abban az esetben, ha a kliens gépén nem található meg a szerver nyilvános kulcsa. Kapcsolódáskor ebben az esetben a program felajánlja, hogy egy csatornán keresztül átmásolja a szerver a saját nyilvános kulcsát. Ha

azonban egy hálózati csalás miatt ezt a kulcsot nem a valós szervertől kapja meg a kliens, hanem egy támadó szervertől, akkor innentől az SSH kapcsolat a támadó fél gépe felé fog kiépülni.

A kliens azonosítására általában háromféle mód áll rendelkezésre. Ezek biztonság tekintetében egyre gyengülve [15.]:

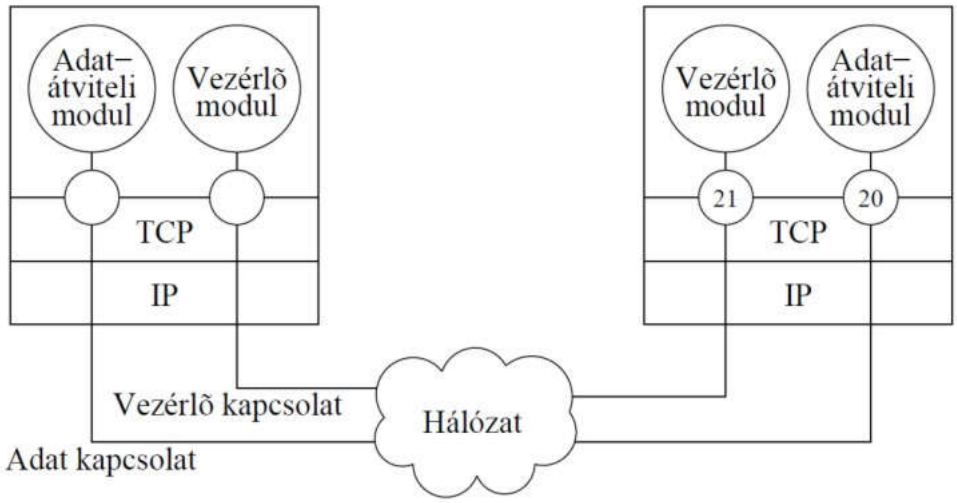
1. Erős azonosítás nyilvános kulcsú módszerrel
2. Jelszavas azonosítás
3. Berkeley r* szerű megoldás használata (bizalom alapú, felhasználónév/jelszó nélküli bejelentkezés)

Az előző lépések hibátlan végrehajtása esetén a kapcsolat készen áll, a teljes kommunikáció titkosítva lesz a korábban felsorolt titkosítási algoritmusok egyikének segítségével.

4.4. SFTP

Az SFTP (SSH File Transfer Protocol) egy fájlmozgatási protokoll, ami ötvözi az SSH és az FTP előnyeit. Mivel manapság a klasszikus FTP protokoll több szempontból is hiányosnak bizonyul, így érthető, miért volt szükség ennek a protokollnak létrehozására. Az FTP-t még az internet hajnalán fejlesztették ki, amikor még nem volt akkora hangsúlya a titkosított kapcsolatoknak, mint manapság. Ennek köszönhetően az FTP összeköttetéshez használt jelszavak és felhasználói adatok is titkosítatlanul közlekednek az interneten, amiket ezáltal meg is szerezhet egy illetéktelen személy. A másik jelentős problémát a tűzfalak okozhatják. Egy jól beállított számítógép vagy szerver esetén mindig csak azok a portok vannak engedélyezve a tűzfalon, ami feltétlenül szükséges a szolgáltatások ellátása érdekébe. Az FTP ez esetben nem feltétlenül működne megfelelően, mivel egy porttartományból választ ki véletlenszerűen egyet, és azon létesít kapcsolatot. A hálózat felé egy egész tartománnyi nyitott port komoly kockázatot jelenthet.

Kezdetnek, hogy pontosabban látszódjon mi is az SFTP lényege, bevezetésképpen a 7. ábra szemlélteti az FTP működését és felépítését. Az SSH bemutatása elérhető az előző alfejezetben.



7. Ábra: Az FTP kapcsolatok működése [15.]

Az FTP (angolul File Transfer Protocol) egy kliens-szerver alapú transfer, vagyis fájlok mozgatására alkalmas protokoll, ami az IP fölött a TCP-t használja az átvitelre. A szervert a kliens program megszólítja szabványosan annak 21-es TCP portján, majd ezen keresztül felépíti a három utas kézfogással a TCP kapcsolatot. Ezt hívjuk vezérlő kapcsolatnak. A parancsok kiadása a vezérlő kapcsolaton történnek, aminek köszönhetően a megfelelő parancsok hatására kiépül még egy TCP kapcsolat, szabványosan 20-as porton. Ez kétféle lehet irányától függően:

- Aktív mód: ez esetben a szerver építi ki a kapcsolatot a kliens fele, aki elküldi a szerver számára a szükséges IP cím és port párost, amin keresztül a kliens várj a kapcsolat kiépítését. A mint a szerver megkapja a megfelelő kapcsolati adatokat, akkor a 20-as TCP portjáról megkezdi a kapcsolat kiépítését a kliens által megadott IP cím és TCP port felé. A kért fájlok mozgatása után ez a kapcsolat bomlik. [14.]
- Passzív mód: ez esetben a szerver küldi el az IP címet és a port számot a kliens felé, a kliens ez alapján építi fel a kapcsolatot a szerverrel. [14.]

A két módszer közül a passzív mód az általánosan használt manapság. Ennek oka, hogy a tűzfalak szűrésén gyakran fennakadnak az aktív módú kapcsolatok.

4.5. HTTP

A HTTP (HyperText Transfer Protocol) egy, a szerverek és kliensek közti információcserére használt protokoll. Működése kérdés-válasz elven működik. Meghatározott

parancsokkal a kliens TCP kapcsolaton keresztül elküldi üzenetét a szervernek, ami arra válaszol. Ezek az egyszerű kérdések és válaszok széleskörűen elterjedtté és kedveltté tették a módszert világszerte. A HTTP alkalmazható alkalmazásrétegbeli protokollként, de nem ritka, hogy a transfer rétegben is jelen van különböző online programok esetén (médialejátszó alkalmazások, víruskeresők stb.).

A TCP kapcsolat felépítése esetén általában a kliens böngészője vagy alkalmazása veszi fel a kapcsolatot a szerverrel. Ez a szabványokat követve a 80-as vagy 443-as TCP porton történik. A TCP használatának hála a stabilitás biztosítható az összeköttetéshez, illetve megfelelő titkosítási eljárásokkal a biztonság is növelhető.

A HTTP hajnalán a kapcsolatok úgy néztek ki, hogy a kliens és szerver között kiépült egy TCP kapcsolat, ami a kérdés-válasz lezajlása után bontódott. Gyorsan felmerült a probléma, hogy modernebb webalkalmazások esetén ez nem működhet megfelelően a rengeteg egymás utáni TCP kapcsolatépítés és bontás miatt (pl valós időben futó alkalmazások, streamerek esetén), amik rengeteg időt és energiát emésztenek fel. Kialakították az újabb szabványt, ami már megengedte a „kapcsolatok újrahasznosítását” (connection reuse) [1.]. A kapcsolat terhelése így nem csak kisebb lesz, de a kéréseket úgynevezett pipe-ba is lehet fűzni, aminek hála az egyik kérdés megválaszolásának visszaérkezte előtt lehet már küldeni az újabb kérdést. A kapcsolatok lezárása nem történik egyből a válasz megérkezése után. Általában meg van határozva egy időkorlát, amekkora hosszúságú tétlenségig a kapcsolat kiépítve marad (pl. 60 másodperc).

4.6. ICMP

Az ICMP (Internet Control Message Protocol) az IP forgalom vezérlési és hibaüzeneteit szállítja. Az ICMP integráns részét képezi az IP-nek, gyakorlatilag egy külön IP csomag típus, amit 1-es azonosítóval jelölnek az IP fejlécben [27.]. Az IP kommunikáció során az ICMP csomagok feldolgozásáért mindkét oldalon az erre vonatkozó dedikált segédprogram a felelős, tehát az IP alatt futó TCP és UDP csomagok kommunikációjában ezek nem vesznek részt. Ilyen üzenetek gyakran olyan esetben jönnek létre, ha például egy hálózati hiba, torlódás vagy csomagvesztés miatt a cél gép nem kapja meg, vagy hibásan kapja meg a csomagot. Ekkor ICMP üzenet segítségével tudatja a másik félnek, hogy valami hiba csúszott az átvitelben és bizonyos esetben újra kéri a küldést.

Csomagjainak típusai lehetnek:

- ICMP destination-unreachable: Nem érhető el a cél
- ICMP echo-request: Válasz kérés
- ICMP Redirect: a router a kliense felé ajánl egy új, lehetőleg hatékonyabb útvonalat
- ICMP Time-exceeded: a router jelzi a kliens felé, hogy a vett csomag élettartama lejárt

Az ICMP jelentősége nem csak az IP vezérlésében van dolgozatom során, hanem az úgynevezett pingelés során is ilyen protokollnak megfelelő csomagokat használunk. A parancs hatására egy vagy tübb ICMP echo-request kérést küldünk a célként meghatározott eszköz felé, ami válaszol szintén ICMP üzenettel. A program nem csak azt képes meghatározni, hogy elérhető-e a célgép, hanem az ahhoz vezető kapcsolat adatait is képes kiszámítani. Ilyen fontos információ például a TTL, ami a két eszköz távolságát határozza meg az útválasztók száma alapján. Az RTT szintén nagyon jól jellemzi a kapcsolatot, az időben meghatározva mutatja meg a kérdés elküldése és a válasz megérkezése közti időt.

Nagyon fontos megjegyezni, hogy az ICMP ilyen jellegű használata gyakran biztonsági okokból kiindulva a tűzfalon keresztül tiltva van, így a gépünket vagy hálózati eszközünket el lehet rejteni a ping alapján vadászó és felderítő alkalmazások elől. Az ICMP echo-request csomagok tiltása Linux alapú rendszereknél például a következő IPTables szabállyal lehetséges:

```
iptables -A INPUT -p icmp --icmp-type echo-request -j REJECT
```

Más tűzfalak esetén is kínálnak erre megoldást gyártóspecifikus módszerekkel. Ennek a tiltó szabálynak a használata ajánlott a „világ” felé néző eszközök esetében, de privát helyi hálózatokon hasznosabb lehet az echo-request csomagok engedélyezése például az elérhető eszközök felderítésére.

4.7. RS232, UART

Az RS232 (Recommended Standard 232) aszinkron soros átvitel szinte a legősibb soros kommunikációs rendszer, aminek életútja közel egyidős a digitális technikával. A folyamatos fejlődés során többféle szabványt és ajánlást dolgoztak ki az RS232-re alapozva a megnövekedett sebességre való igény és stabilitás okán.

A szabvány által definiált legfontosabb szempontok: [21.]

- Karakterisztikai jellemzők: magába foglalja például a jel feszültségszintjét, időzítési adatokat, átviteli sebességet, terhelhetőséget
- Interfészre vonatkozó tulajdonságokat: csatlakozók kialakítása és lábkiosztása
- Az interfészekben elhelyezett egyes áramkörök funkcióit
- Az általános alkalmazás alcsoportjait
- Az átvitel jellegét, ami UART

A kommunikáció alap koncepciója, hogy a legtöbb módszerrel ellentétbe itt nem használnak szinkron órajelet, ami összehangolja a feleket. Ebből adódóan nincs meghatározott vezérlőjel, ami meghatározhatja mikor történhet adás a vonalra és mikor kell hallgatni. A pontos vétel feltétele, hogy a vevő és az adó azonos frekvenciával dolgozzanak. Amennyiben az adó nagyobb frekvencián küldi az adatokat, mint amivel a vevő oldal mintavételezi, akkor hibás lesz a kommunikáció. Ellenkező esetben is igaz ez a példa.

A kapcsolat alapja hasonló az OSI modellből ismert szerkezetre, ahol a különböző rétegek mind egymásra épülve teszik lehetővé a kapcsolat használatát. Ezek a rétegek a következőek lentről felfelé haladva a táblázatban a legalapvetőbbtől kiindulva

	Réteg neve	**Réteg rövid jellemzése**
5	Alkalmazási réteg	Az adatcsomagok szervezését definiálja
4	Adat réteg	A kapcsolat sebességét, bitek számát, jelzőbitek tulajdonságait definiálja
3	Logikai réteg	A logikai 0 és 1 feszültségszintjének definiálása
2	Elektromos réteg	A kapcsolat feszültségszintjeit definiálja
1	Fizikai réteg	A csatlakozót és vezetékeket foglalja magába

4. Táblázat: Az RS232 rétegei[2]

A kétirányú TTL UART alapú kommunikáció esetében 3 vezeték használata a megszokott. Egy szolgál az adásra (TX), egy a vételre (RX) és egy a közös föld pontenciál kialakítására. Abban az esetben ha csak adni vagy venni szeretnénk ez leegyszerűsíthető egy adásmód és a föld kábelre.

A nem TTL alapú RS232 kommunikációnál az az általánosan bevezetett eljárás, hogy a -3 é -25 Volt közé eső feszültség a logikai egy és a +3 és +15 Volt közötti feszültségszint a

2 A táblázat a [[7.]] forrás adatai alapján készült

logikai nullát jelöli. Ezek megnevezésére külön kifejezést használnak: a logikai 1 a MARK a logikai 0 a SPACE névre hallgat.

A klasszikus RS232 a ±25 Volt feszültségszintet támogatja, de a mai modern eszközöknél az 5 voltos TTL logika alkalmazása az elterjedt. Ez esetben a logikai egy az 5 Volt a logikai nulla a 0 Volt feszültségszint. Számítógépek esetében a soros portként nevezett kommunikációs interfészen általában a ±12 Voltos jelszintet használják, de ez a mai napra szinte eltűnt a hétköznapi számítógépektről.

A feszültségszint mértéke befolyásolja a kapcsolattal áthidalható távolságot is. A távolság növekedésével drasztikusan csökken az átviteli sebesség. Az ajánlott maximális kábelhosszok a különböző adatsebességekhez társítva:

Adatátviteli sebesség (bps)	Távolság (m)
2400	60
4800	30
9600	15
19200	7,6
38400	3,7
56000	2,6

5. Táblázat: Az RS232 átviteli sebességeihez ajánlott maximális kábelhossz[3]

Az adatátvitel során a továbbítandó bájtokat bitekké kell alakítani. A kapcsolat specifikálása során fontos megadni, hogy a kapcsolatban milyen mennyiségű és milyen szereppel rendelkező biteket fogunk használni. Ezeknek az adó és vevő oldalon is meg kell egyezni. Ilyen különleges szereppel rendelkező bitek a start bit, a stop bit és a paritás bit. A start bit jelzi a vevő készülék felé, hogy adás fog következni. Ekkor a vevő oldal felkészül az adatok fogadására. A meghatározott számú adatbit után egy stopbittel zárul a kommunikáció, ami utal a vevő számára az üzenet végére. A paritásbitek a hibajavítások során alkalmazottak.

A legalapvetőbb esetben is az átvitelhez az adatbitek száma + 2 bittel kell számolni beleértve a start és stopbitet is. Tehát egy klasszikus ASCII karaktert, amit 8 biten kódolunk el 10 biten lehet minimálisan továbbítani RS232 esetén. Az átviteli sebességből és a 10 bites hosszból kiindulva könnyen ki tudjuk számolni, hogy a 9600 bps adatátviteli sebesség esetén

3 A táblázat a [10.] forrás alapján készült

$\frac{9600}{10} = 960$ bájtnyi adatot tudunk másodpercenként továbbítani. Egy bit elküldéséhez $\frac{1}{9600} = 0.000104$ másodperc (104 μs) szükséges.

Röviden bemutatva a TTL adás folyamatát kezdjük az üzenet nélküli vonallal. Adásszünetben a vezetéken 5 Volt van, ami a logikai 1-et jelenti. Az adás indulásakor a start bit logikai 0 szintre húzza a feszültséget jelezve az üzenet kezdetét. Egy ilyen szimbólum hossza a bemutatott számítással határozható meg. A 9600-as sebesség esetén a start bit hossza (azonosan a többi bittel együtt) 104 μs. A start bit után folyamatosan adásra kerül az összes adatbit. A adatok után használhatunk paritás bitet is, de a gyakorlatban nagyon ritkán fordul ez elő. Miután minden bit el lett küldve egy stopbit jelzi az átvitel végét. A stop bit mindig egy logikai 1-es, tehát magas állapot. Ezután a kommunikáció nélküli vezetéken az adásszüneti 5 Voltos feszültség jelenik meg. Ha több bájtot továbbítunk, akkor a stop bit után egyből jön a következő bájt start bitje, nincs kihagyott szünet.

4.8. RS485

Az automatizálási technológiákban az egyik leggyakrabban használt kommunikációs protokol az RS485. Az RS232-vel szemben nem egy pont-pont kapcsolatot létesít, hanem buszrendszerként több eszköz csatlakozhat a párhuzamos vonalra.

A buszrendszernek köszönhetően a protolkoll képes akár 2 Mbit/s-os átvitelre a maximálisan meghatározott 32 állomás között, akár több száz méteren keresztül is. Hátránya, hogy a fullduplex üzemeltetés nem biztosított, ahhoz már négy vezeték kéne így a kommunikáció a halfduplex módra korlátozódik. A maximális egységek száma szegmensenként a meghatározott 12 kΩ-mal megegyező ellenállásúnak kell lennie. Ennek a meghajtó terhelésnek csökkentése esetén lehet növelni az egységek számát. A meghajtó terhelés az ellenállás értékkel arányos, tehát ha az egységek számát négyszeresére szeretnénk növelni, akkor a meghajtó terhelést is négyszeresére kell növelni, tehát 48 kΩ esetén a maximális rákapcsolható eszközök száma 128. Az egységek meghatározásába beletartozik az adóegység is, tehát 32 egység esetén 32 – 1 = 31 vevő egység használható.

Az RS485 nem határozza meg a protokoll alkalmazási rétegét, csak a fizikai szinten foglal helyet, így meglehetősen sok egyéb protokoll alapja épülnek erre, mint például a Profibus, Modbus vagy a DMX512.

5. Konfigurációs fájlok

5.1. A JSON struktúra

A JSON a JavaScript Object Notation azaz JavaScript objektumjelölés szó kezdőbetűiből származik. A formátum létrehozásánál célnak tűzték ki, hogy egy olyan struktúrát hozzanak létre, ami az emberek számára első ránézésre is jól olvasható és feldolgozható legyen. Kezdetben adatcserére használták, de napjainkban már számtalan helyen előfordul, így például a mérőrendszer programjainak konfigurációjánál is.

A JSON felépítése a JavaScript programozási nyelven alapszik, bár több klasszikus, a C nyelvből eredő egyéb programnyelv előnyös konvencióihoz is igazodik. Ennek köszönhetően a különböző programozási nyelveken dolgozó fejlesztők is könnyen megérthetik a struktúra logikáját.

Két típusú JSON felépítést különböztetünk meg [13.]:

- **Név-érték párok**: elsősorban a klasszikus értelemben vett objektumok és rekordok felépítéséhez hasonlítható, de asszociatív tömbök is lehetnek ilyenek más nyelvekben.
- **Értékek kezdeti listája**: ez leginkább a tömbökhöz és listákhoz hasonló felépítés

A későbbiekben az előbbit, azaz a név-érték párokat fogjuk használni. Ennek felépítése a következőképp néz ki [13.]:

- Az egyes objektumok mindig kapcsos zárójellel kezdődnek ({) és zárulnak (})
- Minden nevet kettőspont követ, amikor értéket adunk neki
- Az egyes név-érték párok vesszővel vannak elválasztva. Az utolsó név-érték pár után nincs vessző
- A tömbök mindig szögletes zárójellel kezdődnek ([ ) és zárulnak ( ])
- Az egyes tömböket vesszővel választjuk el egymástól
- A karakterláncból álló értékeket idézőjelek közé írjuk hasonlóan például a C nyelvben megszokott karakterláncokhoz vagy a Java-ban használt string típusokhoz.

- Karakter pontosan egy egy hosszúságú karakterlánc (mint ahogy a C nyelvben is ismert char típus)
- Numerikus értékeket idézőjel nélkül írunk, a tizedeseket ponttal választjuk el.
- Használatosan a logikai true és false értékek is, illetve a null.
- Számok esetében nem használható hexadecimális vagy oktális formátum, csak a decimális számok alkalmazása megengedett.
- A white-space karakterek bárhol használhatóak (kivéve néhány kódolási részletben, amik a teljes egészében a nyelv leírására használatos)

5.2. Konfigurációs fájl struktúrája

A belső fejlesztésű mérési adatgyűjtő (logger) programok között kétféle konfigurációs struktúrát használunk. Ezeknek a különbsége a mérések típusától függ: külön beállítási lehetőség van SWEEP méréshez és IQ rögzítéshez.

A SWEEP mérés esetén a műszer egy frekvencialépéses módszerrel végigpásztázza a kívánt frekvenciatartományt a megadott lépésközzel, majd ennek eredményeit rögzíti frekvencia és kapocsfeszültség párosban. Természetesen ehhez rögzítve vannak még a feldolgozást segítő adatok is, mint például geolokációs koordináták és időbélyegek.

A konfiguráció során minden mező kitöltése kötelező. Ha valamelyik funkcióra nincs szükségünk, akkor azt az null értékkel jelezhetjük. Minden egyes konfigurációnak saját egyedi azonosítója van, amik az egymástól való megkülönböztetést segítik a logger program számára.

A teljes konfiguráció három fő beállító fájlból épül fel:

- **Measurement-config.json:** minden egyes mérési feladatot tartalmaz. A megszabott struktúrába akár több mérési feladat is rögzíthető és ezek paraméterei egymástól függetlenül módosíthatóak.
- **Node-config.json:** az adott mérővevő beállításait, valamint a jelutakhoz rendelt vevőantenna paramétereit és K-faktorait tartalmazza.

- **Program-config.json**: a mérési adatgyűjtő program beállításait tartalmazza, többek közt a logok és mérési eredmények elérési útvonalát, mentési ciklusokat.

A beállítások mindaddig nem lépnek érvénybe, ameddig a futó konfigurációt (vagy a teljes mérővevőt) le nem állítjuk és újra nem indul a mérési adatgyűjtő program. Ennek leggyakoribb módja a teljes mérőrendszer újraindítása, ugyanis ilyenkor nem csak az adatgyűjtő alkalmazás de a Linux alapú rendszer is újra indul és újraolvas minden beállító fájlt.

5.2.1. Sweep mérés konfigurációs struktúrája

A sweep mérésekhez használt beállítási lehetőségek a következő módon épülnek fel:

```
{
   "version": 6,                                          Konfiguráció verziója
   "id": 1549032600,                                      Konfiguráció létrehozásának ideje

   "sweep_list":                                          SWEEP konfiguráció kezdete
   [
      {
         "id": 1,                                         Elvégzendő mérés azonosítója
         "start_freq_mhz": 87,                            Alsó frekvenciahatár egész értéke
         "stop_freq_mhz": 108,                            Felső frekvenciahatár egész értéke
         "start_freq_millihz": 000000000,                 Alsó frekvenciahatár ezredes értéke
         "stop_freq_millihz": 000000000,                  Felső frekvenciahatár ezredes értéke
         "quality": "low noise",                          Feldolgozás típusa
         "speed": "high",                                 A pásztázás sebessége
         "res_bandwidth_hz": 1000,                        Felbontási sávszélesség
         "input": 1,                                      Jelút száma
         "agc": true,                                     Automatikus erősítés szabályozás
         "manual_atten": 0,                               Csillapítás beiktatása
         "repeat_time_sec": 0,                            Mérés periódusideje másodpercben
         "repeat_time_nanosec": 0,                        Mérés periódusideje  nanoszekundum pontossággal
         "sample_aggregation": null                       Aggregáció módja
      }
   ],
   "iq_list":                                             Az IQ lista üres, mivel SWEEP mérést konfigurálunk
   [
   ],
   "measurement_task_list":
   [
      {
         "id": 0,
         "measurement_list":                             A mérési feladat során végrehajtandó elemi mérések listája
         [
            {
               "type": "sweep",                           Elvégzendő mérés típusa
               "id": 1,                                   Elvégzendő mérés azonosítója
               "start_time_sec": 0,                       Mérés kezdési ideje
               "start_time_nanosec": 0,                   Mérés kezdési idejének nanoszekundum töredéke
               "duration_sec": null,                      Mérés időtartama másodpercben
```

```
                "duration_nanosec": null,      Mérés időtartamának nanoszekundum töredéke
                "repeat_time_sec": null,              Mérés periódusideje másodpercben
                "repeat_time_nanosec": null,    Mérés periódusidejének nanoszek. töredéke
                "priority": 0                                 Mérés prioritása (0-100)
            }
        ]
    }
],

"band_power_template_list":     További mérési opciók, amik nem jelentősek a Sweep mérésnél
[
],

"band_power_list":
[
],

"spectral_mask_template_list":
[
],

"spectral_mask_list":
[
],

"action_list":
[
]
}
```

5.2.2. IQ mérés konfigurációs struktúrája

```
{
   "version": 8,                                                  Konfiguráció verziója
   "id": 1549032600,                                    Konfiguráció létrehozásának ideje
   "sweep_list":                                  A SWEEP lista üres, mivel IQ mérést konfigurálunk
   [
    ],
   "iq_list":                                                    IQ mérési lista kezdete
   [
      {
         "id": 0,                                              Elvégzendő mérés azonosítója
         "center_freq_mhz": 88,                          Középfrekvencia egész értéke
         "center_freq_hz": 900000,                     Középfrekvencia maradék értéke
         "bandwidth_hz": 500000,                             Sávszélesség Hz-ben
         "duration_ms": 9000,                          Mérés időtartama másodpercben
         "input": 1,                                                  Jelút száma
         "custom":                                              Egyedi beállítások
         {
            "agc": true,                                Automatikus erősítés szabályzás
            "manual_atten": 0,                             Csillapítás beiktatása
```

```
            "dds_scale": 2,                                  DDS skálázási faktor
            "quality": "low noise",                          Feldolgozás típusa
            "no_retune": false                               Rádió újrahangolásának mellőzése
         }
      }
   ],
   "measurement_task_list":
   [
      {
         "id": 0,
         "measurement_list":
         [
            {
               "type": "iq",                                 Elvégzendő mérés típusa
               "id": 0,                                      Elvégzendő mérés azonosítója
               "start_time_sec": 1637577000,                 Mérés kezdési ideje
               "start_time_nanosec": 0,     Mérés kezdési idejének nanoszekundum töredéke
               "duration_sec": 30,                           Mérés időtartama másodpercben
               "duration_nanosec": 0,        Mérés időtartamának nanoszekundum töredéke
               "repeat_time_sec": null,                      Mérés periódusideje másodpercben
               "repeat_time_nanosec": null,  Mérés periódusidejének nanoszek. töredéke
               "priority": 0                                 Mérés prioritása (0-100)
            }
         ]
      }
   ],
   "band_power_template_list":    További mérési opciók, amik nem jelentősek az IQ mérés esetén
   ],

   "band_power_list":
   [
   ],

   "spectral_mask_template_list":
   [
   ],

   "spectral_mask_list":
   [
   ],

   "fft_list":
   [
   ],

   "filter_list":
   [
   ],

   "action_list":
   [
   ]
}
```

5.2.3. Node konfigurációs struktúra

Egy példa konfiguráció és az ahhoz kapcsolódó legfontosabb paraméterek a következő sorokban olvashatóak.

```
{
  "version": 1,                                          A konfiguráció verziója
  "id": 1539368023,                                      A konfiguráció egyedi azonosítója
  "general":                                             A Node általános paraméterei
  {
    "id": 1,
    "name": "RFEye-teszt",                               Mérővevő megnevezése azonosítás véget
    "description": "Teszt node"                          Leírás a mérővevőről
  },
  "binlog":                                              Nyers mérési adatok beállításai
  {
    "rotate_secs": 3600,                                 Fájl lezárás gyakorisága
    "delete_after_cda_ack": true,
    "storage_thresh": 0.9,
    "storage_thresh_action": "delete oldest acked"
  },
  "alarm": null,                                         SMS-ben küldött riasztások beállításai
  "signal_path":                                         Rádiós jelutak konfigurációja
  {
  "antenna":                                             Antenna adatok konfigurációjának kezdete
  [
     {
       "id": 0,                                          Antenna azonosítója
       "name": "Antenna-Type-1",                         Antenna neve
       "description": "XPO2v-0.8-6.0-GF/1441",           Antenna leírása
       "min_freq_mhz": 600,                              Antenna alsó határfrekvenciája
       "max_freq_mhz": 6000,                             Antenna felső határfrekvenciája
       "antenna_factors": [                              Antennafaktorok
         [600, 32.76], [650, 31.05], [700, 30.61], [750, 29.84],
         [800, 26.09], [850, 26.95], [900, 26.83], [950, 27.27],
         [1000, 28.5], [1200, 29.76], [1400, 31.01], [1600, 32.69],
         [1700, 33.02], [1800, 33.41], [1900, 34.65], [2000, 35.42],
         [2100, 35.7], [2200, 36.23], [2500, 37.88], [3000, 37.95],
         [3500, 38.38], [4000, 39.13], [4500, 40.15], [5000, 40.91],
         [5500, 42.04], [6000, 45.41] ]
     }
   ],
  "input":                                               Bemenetek konfigurációjának kezdete
  [
    {
      "id": 2,                                           Bemenet azonosítója
      "antenna_id": 1,                                   Bemenethez rendelt antenna azonosítója
      "input_factors": null                              Jelútfaktor
    },
    {
      "id": 3,                                           Bemenet azonosítója
      "antenna_id": 0,                                   Bemenethez rendelt antenna azonosítója
      "input_factors": null                              Jelútfaktor
    }
  ]
  },
```

```
  "vendor_specific": null                                   Gyártó specifikus Node beállítások
}
```

5.2.4. Program konfigurációs struktúra

```
{
  "version": 3,
  "id": 1539368023,
  "general":
   {
      "work_path": "/mnt/internal/pp-iq-teszt",             Logfájlok mentési helye
      "input": "ncp",                                       Adatok forrása
      "loglevel": 2,                                        Logolási szintek (prioritás szerint)
      "logfile": "debug.log"                                Hibakereséshez kapcsolódó logok célmappája
   },
   "cda":
   {
      "host": "",                                           CDA hosztcíme
      "port": 4444,                                         Hálózati port
      "timeout": 1                                          Időtúllépési korlát
   },
   "binlog":
   {
      "path": "log",                                        A bináris sweep logok mentési útvonala
      "extension": ".bin",                                  A bináris logok fájlkiterjesztése
      "write_iq": false                                     IQ adatok mentésének engedélyezése
   },
   "iqlog":
   {
      "path": "log-iq",                                     Az IQ logok mentési útvonala
      "write_iq": true                                      IQ adatok mentésének engedélyezése
   },
   "ncp_input":                                             Az NCP démon elérése
   {
      "host": "127.0.0.1",                                  A kiszolgáló IP címe
      "port": 9999,                                         Kiszolgáló portja
      "timeout": 30,                                        Időtúllépési korlát
      "lock_radio": false                                   Vevő lezárása (más folyamat ne érje el)
   },
   "file_input":
   {
      "path": "log-archive",                                Log archívum eléri útja
      "file_list": [ "*" ]                                  Fájlok szelektálásának lehetősége
   }
}
```

5.3. Számítandó értékek meghatározása

A konfiguráció során bizonyos értékek meghatározása nem közvetlenül történik, hanem a felületen megadott adatokból valamilyen matematikai módszerek segítségével állítható elő. A következő esetekben van szükség matematikai számítások végzésére, hogy a mérőműszer által várt értékeknek megfelelő bemeneti értékeket szolgáltathassunk.

5.3.1. IQ mintavétel center frekvenciája

$$f_c = (1000000 \cdot \text{center\_freq\_mhz}) + \text{dds\_offset\_hz}$$

Ahol *center_freq_mhz* a kívánt középfrekvencia MHz-ben kifejezve és *dds_offset_hz* a beállított középfrekvencia eltolása Hz-ben

5.3.2. IQ mintavételi sávszélesség

$$Bandwidth[Hz] = \frac{40000000}{decimation}$$

Ahol a *decimation* a decimálási faktort jelképezi, azaz azt a frekvencia tartományt, amekkorát mérni szeretnénk a centerfrekvencia körül

5.3.3. Az IQ mintavételi időtartama:

$$t[s] = \frac{\text{num\_samples}}{BW}$$

Ahol *num_samples* a vételezett IQ mintapárok száma, *BW* a vizsgált sávszélesség nagysága az 5.2.2. pontban meghatározottak szerint.

6. Kezelőfelület bemutatása

A kezelőfelület létrehozásánál szempont volt, hogy minden egyes mérési feladatot külön lehessen kezelni attól függetlenül, hogy milyen mérési tartománnyal és ehhez kapcsolódó rádiófrekvenciás, aggregációs vagy időzítési feladattal rendelkezik. Ezek az elemek SWEEP mérés esetén a vevőkészülék frekvenciatartományán belül több módon rögzíthetőek különféle feltételek szerint, míg az IQ mérés esetén mindig csak egy frekvenciasáv rögzíthető egy időben.

A kényelmes konfigurációs lehetőségek biztosítása érdekébe a program egy központi oldallal rendelkezik, aminek felülete dinamikusan változik a feldolgozandó adatok és a beállítandó célnak szánt állomástípustól függően. Minden egyes részfunkció külön modulként is használható, így a nagy program szétdarabolása esetén akár több kisebb eszköz is rendelkezésre áll a kiegészítő mérőállomások vezérléséhez.

A Python nyelv adta lehetőségeket kihasználva a program megtervezésénél igyekeztem minél rugalmasabbra és újra felhasználhatóbbra írni a forráskódot. A Tkinter grafikus megjelenítő csomag adottságai sok esetben korlátot szabtak az elképzeléseimnek, de ezek megkerülésével minden eltervezett grafikus elem megvalósítása sikeres lett.

A következő bekezdések rövid magyarázattal betekintés nyújtanak a felület kinézetébe és megmutatják annak funkcióit.

6.1. Konfiguráció szerkesztő

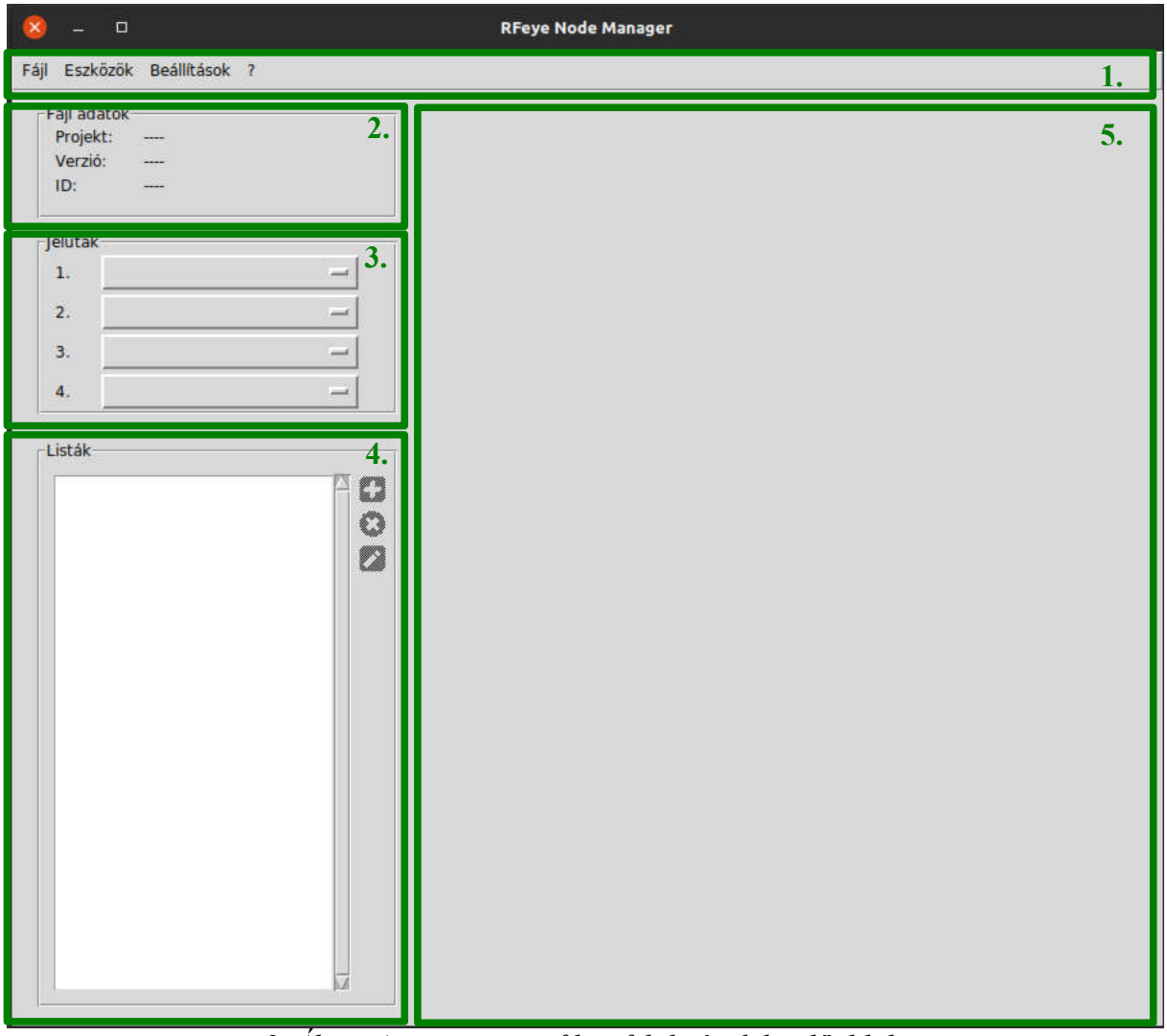
Az 5. fejezetben bemutatottak szerint felépülő konfigurációs fájlok szerkesztésére egy olyan ablakot készítettem, ami a kétféle mérésmód (SWEEP vagy IQ mérés) és a jelenleg használt mérőműszer típusok (Marvell és Nexus) függvényében mindig a megfelelő információt biztosítja a könnyed átláthatóság érdekébe.

Az ablak felépítése öt fő elemből épül fel:

1. **Menüsor**: A menüsor tartalmazza az elsődleges irányítási, fájlműveleti funkciókat, konfigurációs lehetőségeket, illetve innen érhető el a súgó, ami segítséget nyújt a felhasználónak a konfigurációs folyamat végrehajtásához.

2. **Fájl adatok mező**: Ez a mező tartalmaz minden olyan információt, ami szükséges a mérőműszer és konfigurációs fájl közötti kompatibilitás ellenőrzésére. Ezek az adatok nem a mérést befolyásolják, csupán az annak üzemszerű működésének biztosítását segítik elő.
3. **Jelutak mező:** A mérővevők négy jelúttal rendelkeznek, amikre különböző paraméterek mellett köthetőek különböző antennák. Minden mérés beállítható tetszőleges jelútra, amiket itt konfigurálhatunk. A legördülő listából az antenna szerkesztő modulon keresztül beállított antennákat kapcsolhatjuk jelutakhoz, ezzel azt elérve, hogy minden jelutat a megfelelő antenna tulajdonságokkal megegyezően konfigurálhassunk.
4. **Mérési listákat tartalmazó mező**: a különböző méréseket listákba rendezzük, amiket felparaméterezhetünk az adott felhasználásnak megfelelően. A SWEEP mérések esetén a mérőműszer sávszélességén belül több mérési tartományt hozhatunk létre, míg az IQ méréseknél egyet.
5. **Konfigurációs mező**: igaz a program indulásakor üres a konfigurációs mező, de új projekt létrehozása vagy meglévő megnyitása esetén az adatok ezen belül tekinthetőek meg vagy módosíthatóak.

A 8. ábra szemléltei a program indulásakor látható felületet és megkülönböztethetőek rajta a felsorolt részegységek.

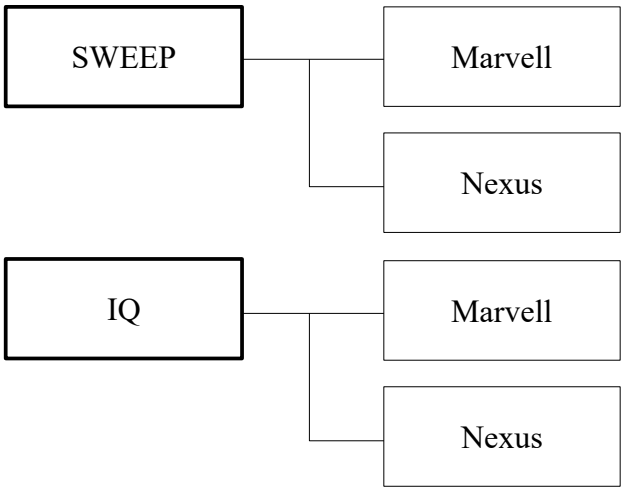


8. Ábra: A program grafikus felületének kezdőoldala

6.2. Új projekt létrehozása

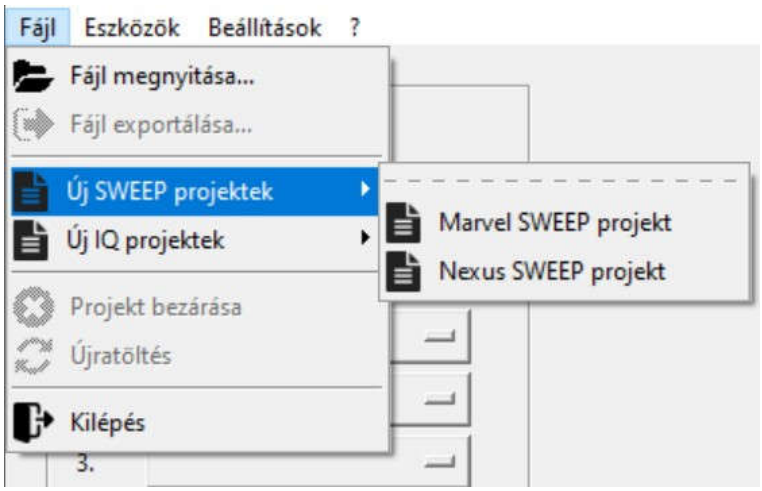
Különböző projektek segítik a munkát a teljes konfiguráció alatt. Mint ahogy az 5. fejezetben bemutattam minden mérési konfiguráció három fájl meglétét biztosítja: a mérési, a program és a node konfigurációját tartalmazót. Ha projektben dolgozunk ezek legenerálása automatikusan megtörténik a mentés során és nincs szükség kézzel más szerkesztő igénybevételére.

A kétféle mérési típushoz különböző projektek tartoznak, így ezek kiválasztásával hozhatóak létre a kívánt mérési feladatok. A következő opciók közül lehet választani projekt létrehozásánál a jelenleg használt két műszertípusnak megfelelően:



9. Ábra: Projekttípusok

Mint az a 9. ábrán látszik a SWEEP és IQ mérés paraméterei függenek a műszer típusától. Ennek oka az, hogy a dokumentációkban is olvasható módon a típusok között eltérés van a sávszélesség és frekvenciatartomány tekintetében, amik által más eredmények szükségesek a beállítások során. A minél pontosabb mérés érdekében ezek a paraméterek nem elhanyagolhatóak, így fontos kellő figyelmet fektetni arra, hogy a megfelelő műszertípushoz készüljön a konfiguráció. Ennek programbeli megvalósítását a 10. árba mutatja.



10. Ábra: Projekt létrehozása a kezelőfelületen keresztül

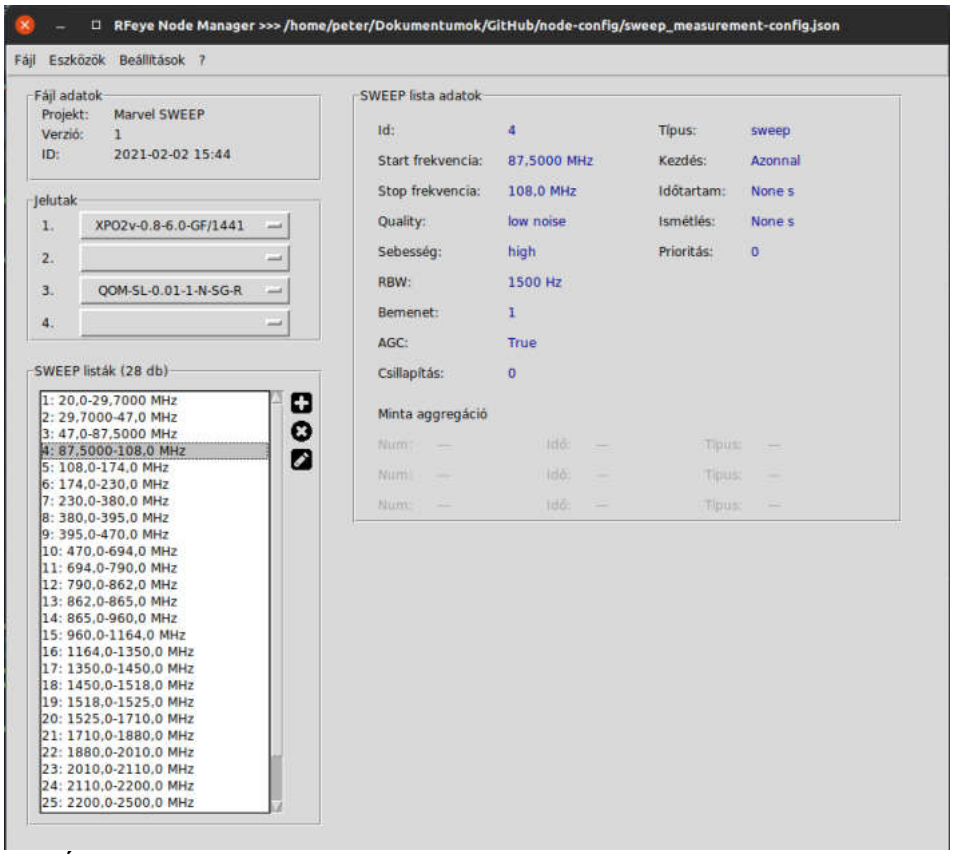
Kiválasztva az új projektet a fájl adatok mezőben megjelenik a projekt típusa. A verzió és ID a mentés, illetve feltöltés után generálódik le. Ezek a konfigurációs fájlok azonosítására szolgálnak, így az aktuális időbélyeget felhasználva generálja azt a program a minél precízebb megkülönböztethetőség kedvéért.

A jelutak beállítása a legördülő menüből könnyedén lehetséges, ahol egyesével kiválaszthatjuk a megfelelő antennákat az antennaszerkesztő listában rögzített megnevezések és paraméterek alapján.

Új projekt létrehozása esetén egy teljesen üres mérési listát kapunk, amire fel kell venni a megfelelő mérési feladatokat. Ezeket a lista melletti gomb segítségével tehetjük meg. Törlésre az gomb szolgált míg módosításra a .

6.3. Létező projekt megnyitása és módosítása

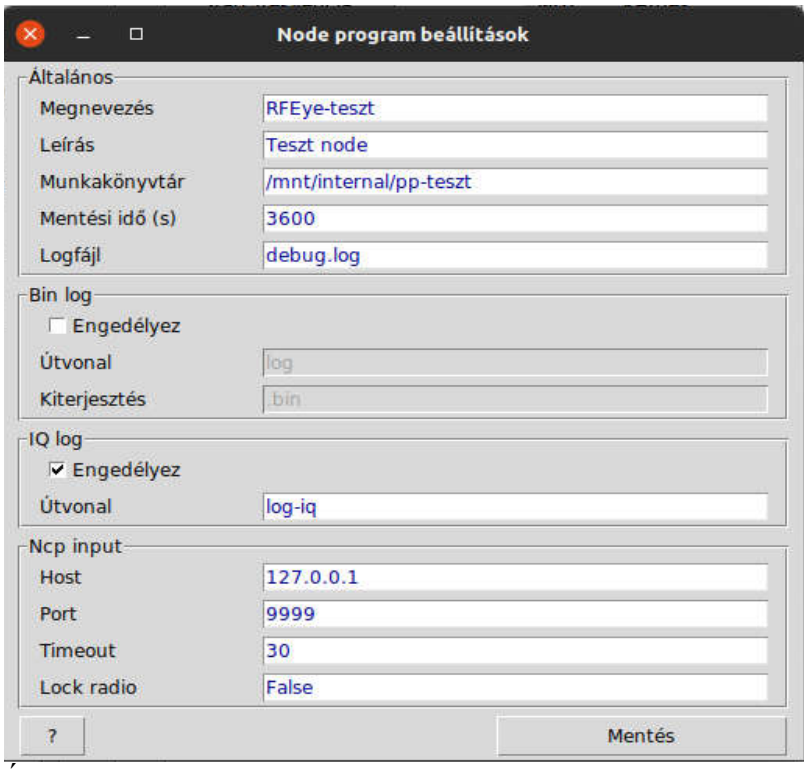
A menü „Fájl” – „Fájl megnyitása...” gombbal létező projektet tölthetünk be. Ennek egyik legfontosabb feltétele, hogy a három konfigurációs fájl egyazon útvonalon legyen. Amennyiben ez nem így van, a program nem képes beolvasni minden adatot és egy felugró hibajelzéssel megszakítja a megnyitást.



11. Ábra: Megnyitott létező projekt több mérési utasítással a listában

Amennyiben minden fájl a megfelelő helyen volt a megnyitás során, akkor betöltődnek a fájlokban tárolt adatok. Első ránézésre csak a „measurement-config.json” fájl tartalmának bizonyos elemei jelennek meg a felületen. Ennek oka, hogy a mérések legnagyobb részén ezek módosítása szükséges és így a legkézenfekvőbb ezeket megjeleníteni először.

A többi fájl tartalma is beolvasódik ilyenkor és eltárolódnak. Ezek legfőbb paramétereinek megjelenítésére az „Eszközök” – „Program config...” menügomb szolgál, ami felfedi mindazon paramétereket, amik módosítása szükséges lehet. Ennek ablakát szemlélteti a 12. ábra. A konfigurációs fájlokat elemezve feltűnhet, hogy nem minden adat jelenik meg ezen a felületen a fájlokból. Ennek oka, hogy vannak olyan konstansnak vett paraméterek és útvonalak, amik módosítása a konfigurációk szinte összes esetén érintetlenek és nem befolyásolnak semmilyen mérési eredményt.

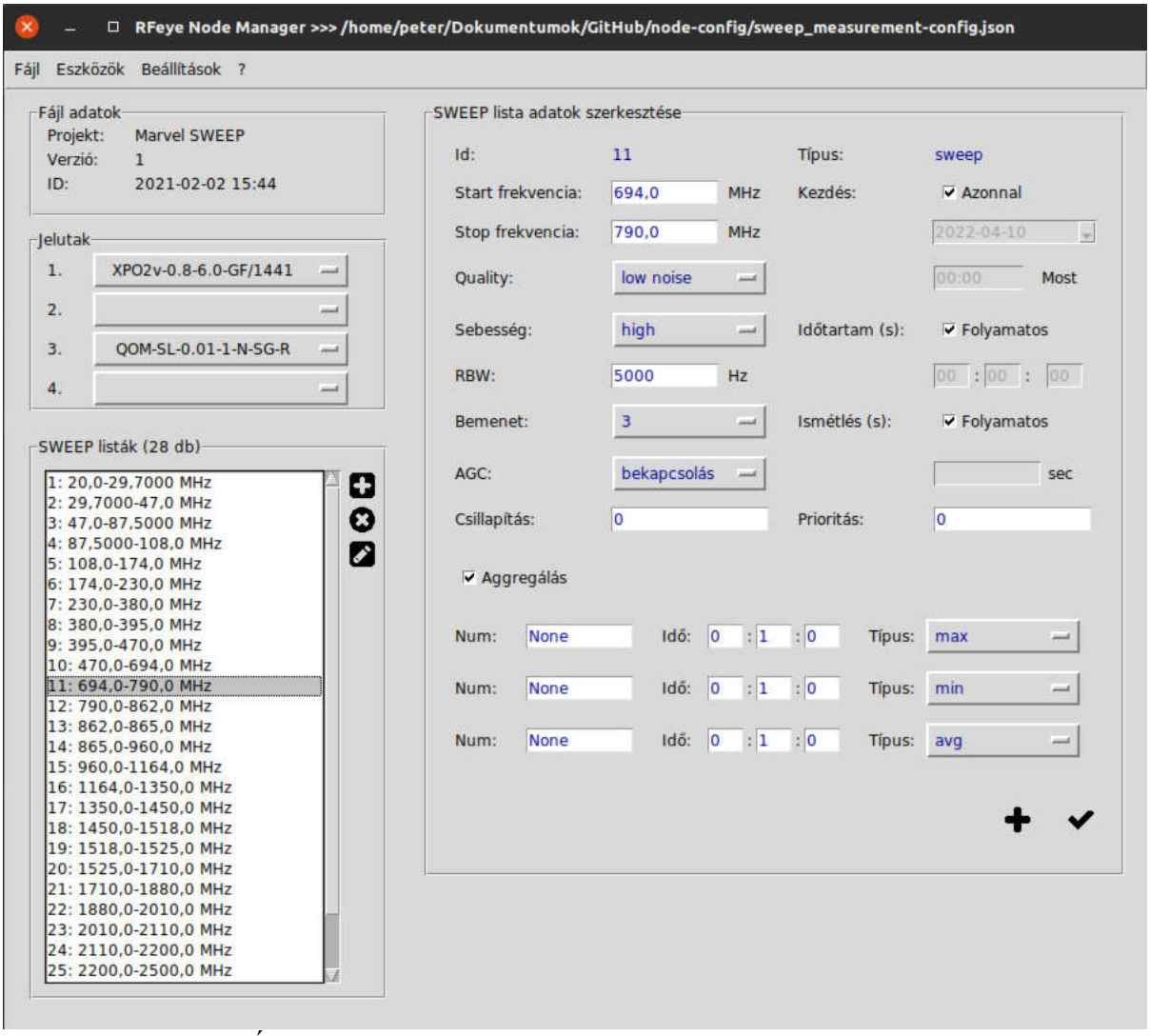


12. Ábra: A program beállítások megjelenítése és szerkesztése menüből

Az ablak bal alsó sarkában található „?” feliratú gomb a gyorsabb kitöltést szolgálja. Erre kattintva olyan mintaadatokkal töltődnek fel a mezők, amik önmagukban már nem igényelnek külön módosítást a felhasználó részéről, legfeljebb némi személyre szabást az adott feltételeknek megfelelően. Eddig tapasztalatok szerint ennek használata az új

projektekkel való munkát nagyban felgyorsítja, ugyanis a látható lehetőségek közül leggyakrabban a mérés megnevezése, leírása, munkakönyvtár útvonala és a mentési periódusidő módosítása történik meg, mint legfontosabb paraméterek.

A már listában lévő elemek megjelenítése mellett módosításukra is van lehetőség. A lista elemeire dupla kattintással megjelennek az értékek egyszerű Label (ls. 7.2. bekezdés) típusú widgetként, de a ceruza ikonnal jelzett módosító gombra a mezők értéke szerkeszthetővé válik, mint ahogy a 13. ábra is mutatja.



13. Ábra: A listában szereplő elemek szerkesztése, módosítása

A pipa ikonú gomb elmenti a módosításokat, a „+” -ra kattintva pedig a mező már módosított változatát menthetjük el külön listaelembe, ami abban az esetben hasznos, ha a konfiguráció paramétereinek nagy részét át szeretnénk másolni új konfigurációba időspórolásból.

6.4. Antenna szerkesztő

Mivel a program támogatja azt a funkciót, hogy mind a négy jelúthoz más-más antennát kapcsoljuk, így biztosítani kell azt a konfiguráció generálása során, hogy ezeknek a jelutaknak a beállítása megfelelő antenna paramétereket hordozzon. Minden antenna más és más frekvenciatartományban rendelkezik a legnagyobb nyereséggel, így többféle antenna használata szükséges a frekvenciaspektrum minél hatékonyabb lefedéséhez. Ez a különbözés az, ami szükségessé teszi a számításokhoz, hogy ne ugyan azzal a konstans paramétersorozattal számoljunk, hanem minden egyes antenna esetén az annak legmegfelelőbbel.

Annak érdekében, hogy biztosítani lehessen a mérések pontosságát, és ez a lehető legfelhasználóbarátabb módon történhessen létrehoztam egy antenna szerkesztő modult, aminek adatai a rögzítés után bekerülnek az egyes jelutakhoz tartozó listákba. Mivel a jelutak egymástól függetlenek, így nem okoz az se problémát, ha mindegyikhez más paraméterű antennát állítunk be, illetve az sem, ha ugyan azokat a paramétereket rendeljük több jelúthoz nagyobb számú azonos antenna esetén.

Az antennák egyértelmű azonosítása miatt mindegyiket el kell látni egy egyedi névvel, ami megjelenik a főablakban a jelutak választásánál. Ezeken túl lehetőség van megjegyzés megadására is, ami nem kötelező, csupán annak megértését szolgálja, hogy például több azonos antennatípus esetén is meg tudjuk egymástól különböztetni őket vagy elmentsünk egyéb tulajdonságokat. Az antenna üzemi frekvenciái nélkülözhetetlen információk. A logger program ez alapján határozza meg, hogy pontosan milyen tartományban lehet mérést végezni a kiválasztott jelúton. Az antenna ajánlott frekvencia tartománya meghaladhatja a mérővevő által biztosított maximális mérési frekvenciát is, de természetesen ez esetben csakis a mérőegység által meghatározott maximális frekvencia lesz a mérvadó a mérés során. Amennyiben viszont fordítva igaz ez, tehát az antenna frekvenciája kisebb, mint a mérővevő maximális mérőfrekvenciája, akkor a mérés maximális frekvenciáját az antenna határozza meg. Mivel mint említettem minden antenna más-más tulajdonságokkal bír, így ezek az antennák egyesével be vannak mérve, így meghatározva azt, hogy milyen frekvencián milyen nyereséggel bírnak. Ezek az értékek az antenna faktorok és a mérésekből adódó számításokhoz (például térerősség számítás, ERP számítás) használhatóak fel. Ezek a gyártó vagy a mérőlabor által adott eredményeket tartalmazó fájlból olvashatóak be.

A program működése egy adatbázison alapul, aminek struktúrája egy saját logika alapján meghatározott rendszer. Az adatbázis alapját a JSON által használt struktúra alapozza, amiben könyvtáras módszerrel helyezkednek el az egyes antennák és azoknak tulajdonságai. A fájl hossza nincs meghatározva, így az antennák száma is tetszőlegesen bővíthető.

A program biztosítja antennák felvételét, törlését és az adatok módosítását. Minden módosítás vagy új rekord létrehozása után a mentés automatikusan megtörténik a kényelem biztosítása érdekébe.

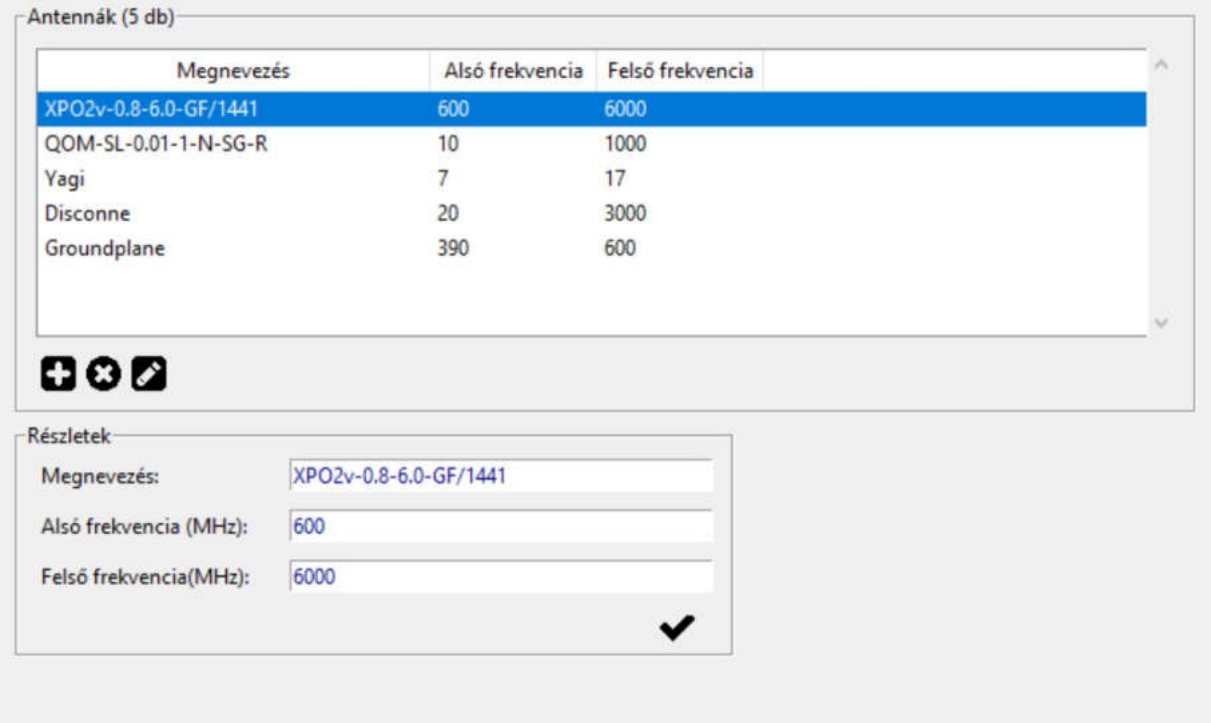
Egy példa az adatbázisból jobban szemlélteti a felépített struktúrát:

```
{
    "antennak": [
        {
            "name": "XPO2v-0.8-6.0-GF/1441",
            "description": "XPO2v-0.8-6.0-GF/1441",
            "min_freq_mhz": 600,
            "max_freq_mhz": 6000,
            "antenna_factors": [
                [
                    600,
                    32.76
                ],
                [
                    650,
                    31.05
                ],

                ...

                [
                    6000,
                    45.41
                ]
            ]
        }
    ]
}
```

A teljes tartalom nagy helyigénye miatt a mintát lerövidítve mutatom be, a három pont helyére az antenna faktorok további értékei kerülnek.

A mezők megnevezései, illetve tartalma megegyezik az 5.2.3. fejezetben bemutatott struktúrával. Különbség a tárolás módjában rejlik. Mivel ezek az adatok nem korlátozott számban vannak jelen az adatbázisban, így biztosítani kellett a megfelelő értelmezési lehetőséget azok számára is, akik a forráskódot vagy az adatbázis szeretnék böngészni nagy mennyiségű adat esetén is. Ezt az egyértelmű változó nevekkel és címekkel biztosítottam.

A 14. ábra egy példa az antennák szerkesztési módjára. Mint látható, az adatbázisból kiragadott antenna példájára a program felületén tekinthetjük át a korábban felsorolt antenna paramétereket. Ez a lista dinamikusan változtatható tetszés szerinti adatokkal. Az ebből generált adatbázis fájl a főprogram minden indulásakor beolvasódik, de manuálisan frissíthetjük a jelutakhoz való hozzárendelését az „Eszközök” – „Antenna lista újratöltése” gombra kattintva, ezzel biztosítva a friss adatok elérését.



14. Ábra: Példa egy antenna adatainak szerkesztésére

6.5. Adatok feltöltése

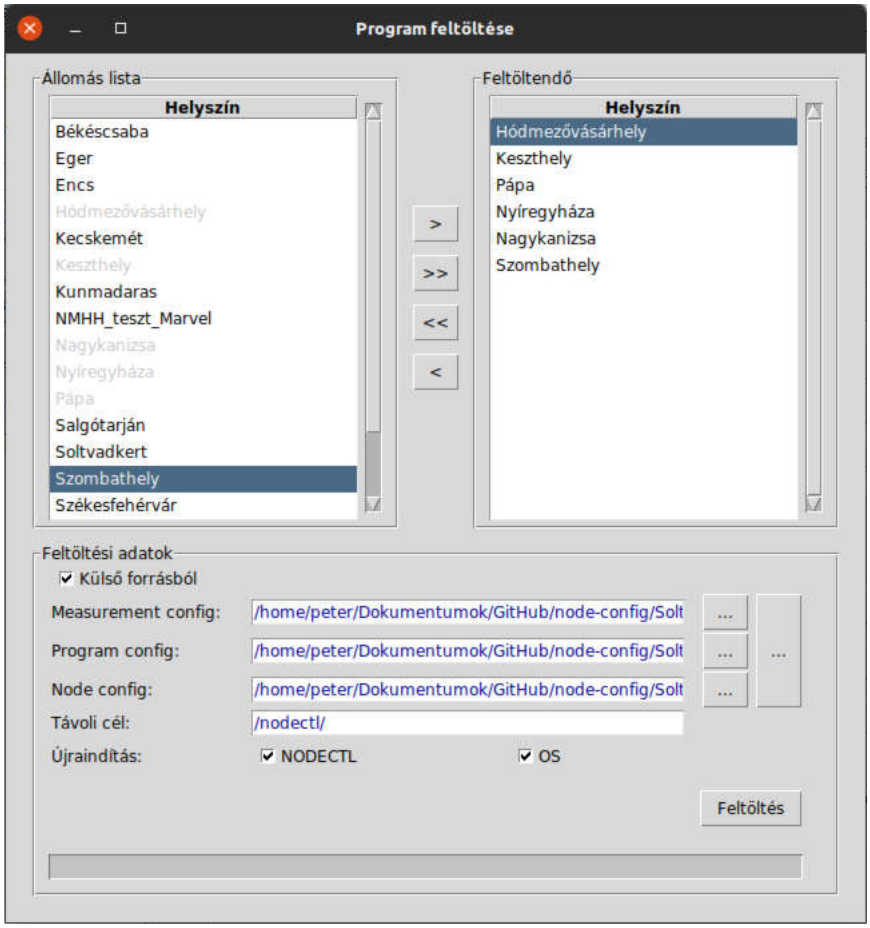
6.5.1. Program feltöltő ablak

Az adatok mozgatásához nélkülözhetetlen egy olyan biztonságos, ám mégis gyors protokoll használata, ami képes akár publikus hálózaton keresztül is titkosítottan továbbítani adatokat. Erre a célra az SSH kapcsolatot választottam, ami a 4.3. fejezetben bemutatottak alapján két számítógép között hoz létre titkosított szerver-kliens kapcsolatot. A protokoll alkalmas fájlok mozgatására is az SSH-ra épülő SFTP protokoll segítségével, ami egyesíti az

SSH és a régi, de mégis a mai napig népszerű FTP protokoll előnyeit. Erről a 4.4. fejezetben írtam részletesebben.

A programom elkészítése előtt a kézzel létrehozott konfigurációs fájlokat valamilyen SFTP kapcsolatot kiépítő kliensprogrammal (pl FileZilla, Bitvise, WinSCP, DoubleCommander) mozgatták át egyik gépről a másikra.

Nem egyszer fordul elő az az eset, hogy azonos beállításokat kell eljuttatni több állomásra is. Ennek módja a korábbi módszert követve nem más, mint hogy egyesével be kell lépni minden állomás rendszerébe és egyesével átmásolni mind a három konfigurációs fájlt. Mondani se kell, hogy ez például húsz mérőállomás beállítása esetén milyen idő és munkaigényes feladat. Ennek a monoton folyamatnak kiküszöbölése érdekébe a programba beépítettem egy olyan modult, ami az SFTP protokollt használva képes a létrehozott konfigurációt a mérőállomásokra eljuttatni. Lehetőség van egyetlen állomásra, vagy egynél többre is átmásolni azonos beállítást. Ennek a felületnek a bemutatása látható a 15. ábrán.



15. Ábra: Konfigurációs program feltöltési felülete

Az ablak felépítése egy könnyen átlátható, kéthasábos listára épül. Az első lista a mérőállomásokat tartalmazza, a második, vagyis jobb oldali feltöltési lista a célállomások neveit gyűjti össze. A bal oldali lista elemeire dupla kattintással vagy a > gombbal van lehetőség az állomás hozzáadására a jobb oldali listához. Ekkor a bal listában lévő elem neve szürkére vált ezzel is jelezve, hogy a jobb oldali listába fel lett véve. A program a háttérben rendelkezik egy olyan védelemmel is, ami megakadályozza, hogy ugyan azt az állomást többször is hozzáadjuk a feltöltési listához. Több állomás hozzáadásánál értelem szerűen több állomást kell kattintással vagy gombnyomással hozzáadni. Az összes állomás hozzáadására a >> gomb megnyomásával van lehetőség.

A jobb oldali listából való eltávolítást szintén dupla kattintással vagy a < gomb megnyomásával lehet véghez vinni. A << gomb az összes elemet eltávolítja a feltöltési listából.

Alap esetben a program az aktuális projektet tölti fel a kijelölt állomásokra. Ez felgyorsítja a munkát, mivel elég egyszer megnyitni vagy szerkeszteni egy projektet és egyből feltölthető az alkalmazási helyére. A program a feltöltés előtt ilyenkor legenerálja a megfelelő három konfigurációs JSON fájlt és ezeket másolja a célcímekre. Miután a fájlmozgatás folyamata lezajlott az ideiglenesen létrehozott fájlokat törli.

Sok esetben jön elő az az igény, hogy előre elkészített vagy korábban létrehozott konfigurációt szeretne valaki több állomásra feltölteni gyorsan és kényelmesen. Ehhez is biztosít a programom megoldást. A feltöltési adatok mezőben ha a "Külső forrásból" jelölő négyzet ki van pipálva akkor nem szükséges a projekt megnyitása a szerkesztő ablakba, hanem bárhonnan beimportálhatók a fájlok a számítógépről. Ennek is két módja van. A mezőkbe a „...” feliratú gombokkal könnyedén, grafikusan kiválaszthatjuk a feltöltendő fájlt egyesével. Ennek jelentősége akkor mutatkozik meg, amikor nem egy mappában helyezkedik el a három fájlt. Ilyenkor külön meg lehet adni mind a három elérhetőségét egymástól függetlenül. Másik opció, ha a jobb oldali, hosszú „...” feliratú gombra kattintunk. Ilyenkor a programot egy olyan mappába kell navigálni a párbeszédablakban, ahol egy mappán belül helyezkedik el a három JSON fájl. Az egyikre kattintva automatikusan kitöltődik a három mező és beillesztődik a három útvonal. Ezeknek a funkciónak kitalálásánál is a minél sokoldalúbb és minél rugalmasabb alkalmazási lehetőségek kidolgozása volt a célom.

A mérővevőkön futó logger program minden újraindításnál betölti a legfrissebb mérési beállítást és ez alapján futtatja a műveleteket. Logikus, hogy ez az újraindítás is automatizálható legyen, ne kelljen a felhasználónak egyesével újraindítási parancsot kiküldeni az állomásoknak. Az újraindítási lehetőségek között kétfélét ajánl fel a felület:

- **NODECTL újraindítása**: ez esetben a Node-on futó logger service újraindítása történik meg.
- **OS újraindítása**: a teljes Linux alapú operációs rendszer újraindítása. Ez a funkció a Hard Reset-ként is ismert funkciója a mérővevőnek, mivel itt az operációs rendszerrel együtt minden egyéb szolgáltatás újra indul.

A két lehetőség közül egy általános mérési feladat kiadása után elégséges a NODECTL szolgáltatás újraindítása.

A feltöltés gomb megnyomása a program megjelenít egy előugró abrakot, amin nyomon lehet követni a feltöltés folyamatát. Az SFTP művelet végrehajtásához az állomásokat tartalmazó adatbázisból kiválasztja egy algoritmus a kiválasztott állomásokhoz tartozó IP címet, felhasználói nevet és jelszót, majd a megfelelő könyvtárba juttatja a konfigurációs fájlokat. Ha az újraindítás jelölőnégyzet segítségével aktiválva van, akkor az is végrehajtódik.

Több állomásra való feltöltés esetén a művelet egymást követve sorban fut le. Az állomásokra felmásolás és újraindítás után a program megvárja, ameddig a rendszer újraindul és ennek tényéről jelet ad a feltöltő egység számára ICMP (4.6. fejezet) csomagok segítségével. Ha a jel nem érkezik meg egy meghatározott várakozási időn belül, akkor a hiba jegyzőkönyvezése után a következő állomás feltöltéséhez kezd a program. Miután minden lefutott, a választ nem küldő állomások újraellenőrzése fut le.

A feltöltés sebességét befolyásolja a hálózati kapcsolat és a feltölteni kívánt állomások száma. A program fejlesztésekor kísérleteztem olyan megoldással, hogy több állomás feltöltésekor egyidőben kezdődjön meg a feltöltés minden állomásra több szálon futó program segítségével. Ezt a megoldást a kezdeti tesztek után el is vetettem, mivel ha valaki gyengébb számítógépről végzi a munkát vagy az internet kapcsolat sebessége korlátozott, akkor több tíz párhuzamos SFTP szál kiépítésénél a számítógép és a hálózat leterhelődik a nagy mennyiségű kapcsolattól. Ennek elkerülése érdekébe választottam a lassabb, de biztonságosabb módszert.

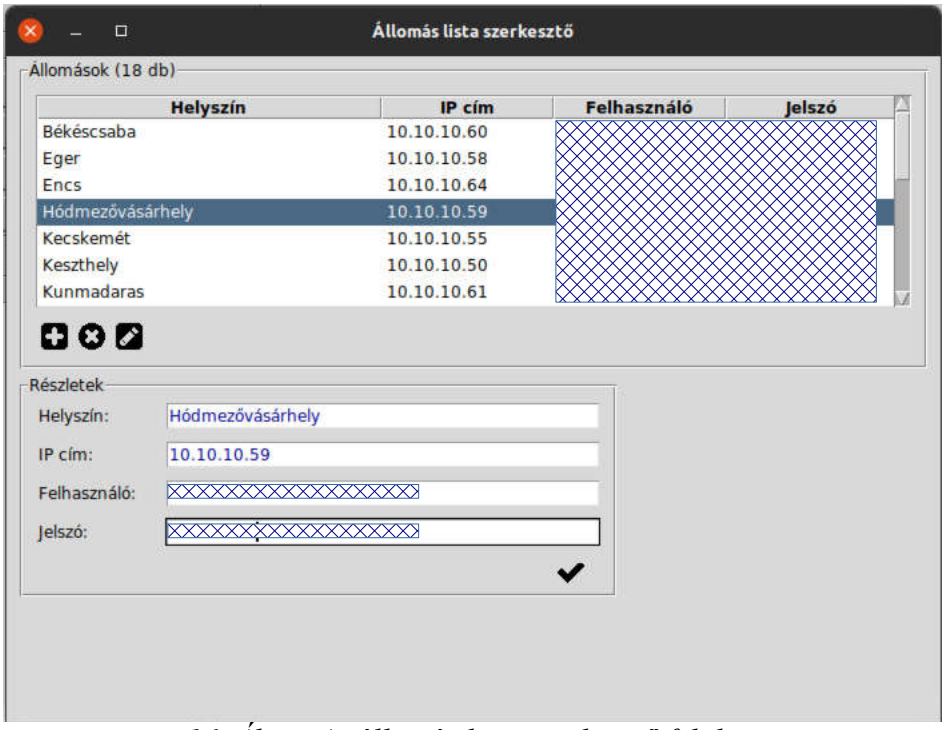
6.5.2. Állomásszerkesztő ablak

A 15. ábrán bemutatott ablakon megfigyelhető, hogy automatikusan képes kilistázni a magyarországi spektrummonitoring hálózat kiegészítő állomásait. Ennek az adatbázisnak a hátterét – az antenna adatokat tartalmazó adatbázishoz hasonlóan – szintén egy JSON alapú strukturált adatfájl adja.

```
{
    "allomasok": [
        {
            "helyszin": "Békéscsaba",
            "ip": "10.10.10.60",
            "user": "***",
            "jelszo": "***"
        },
        {
            "helyszin": "Eger",
            "ip": "10.10.10.58",
            "user": "***",
            "jelszo": "***"
        }
}
```

Minden állomás a szótár szerkezetű fájlban az "allomasok" objektum külön-külön mezői, amik az SSH kapcsolódáshoz szükséges adatokat tartalmazzák. Minden állomást azonosítunk egy névvel, ami a mérőpont helyszínére utal. Ennek egyedinek kell lennie, mivel ez az elsődleges azonosítási mező az adatbázisban. Általában egy városban csak egy kiegészítő mérőállomás van, így ez a féle azonosítás nem okoz gondot. Kivétel ez alól Budapest, ahol több ilyen mérőpont is telepítve van. Ezek nevei általában a városon belüli közigazgatási terület alapján lettek kiosztva (például Vadász hegy, Dolgozó utca). Az IP cím a hálózaton belüli elérést biztosítja, így ennek a pontos megadása az alapköve a kapcsolat kiépítésének. Az SSH kapcsolat létrejöttéhez szükséges még a felhasználó azonosítása, így a felhasználó és a jelszó adatainak rögzítésére is van lehetőség (amennyiben szükséges). Mivel ezek a mezők érzékeny adatok tartalmaznak, így az adatvédelem érdekében ki lettek takarva a kódrészletben és a 16. ábrán.

A 16. ábra a leírt felületet szemlélteti.



16. Ábra: Az állomás lista szerkesztő felülete

6.6. Árbócvezérlés

A kvázi fix mérőállomás rendelkezik egy olyan antennatartó árbóccal, aminek magassága állítható. A kvázi fix név onnan ered, hogy a mérőállomás egy gépjármű rakterében lett kialakítva, felszereltsége megegyezik a fixen telepített mérőállomásokéval. Legnagyobb előnye, hogy a járműnek köszönhetően az országban bárhol használható mérések végzésére.

A járműbe épített, legfeljebb 10 méter magasságba kiereszthető árbóc vezérlése szintén a program részét képezi, ugyanis ezen is használatban van RFeye Node mérővevő.

Az árbóc egy katonai felhasználásra készített, nagyon masszív és megbízható vezérlőegységgel szerelt konstrukció, aminek a vezérlése RS232 protokollon keresztül végezhető. Az RS232 soros adatátvitelre képes protokoll bemutatása a 4.7. fejezetben lett levezetve.

Az árbóc vezérlő áramköre RS232-USB konverter segítségével kapcsolódik a mérő számítógéphez. Ezen a kapcsolaton kétirányú forgalmat bonyolít: a parancsok kiadása és a vezérlő állapotának, pozíciójának lekérdezése is ezen keresztül zajlik. A Python nyelv egy

kiegészítő könyvtára, a pySerial a 7.6. fejezetben van bemutatva részletesebben. Ennek segítségével képes a program kapcsolódni a vezérlő áramkörhöz és a parancsok kiadásának segítségével elérni különböző mozgatási vagy lekérdezési feladatokat.

17. Ábra: Árbóc vezérlő kezelőfelülete

A felület a legszükségesebb elemeket tartalmazza, ami a hatékony munkához nélkülözhetetlen. A 17. ábra alapján ilyenek:

- **Árbóc magassága**: a H-val jelölt érték a földtől mérhető magasság beleértve a jármű magasságát is, az árbóc magassága pedig csak az árbóc magassága a jármű méreteitől függetlenül.
- **Automatikus irányítás**: a fel és le funkcióval végállástól végállásig lehet ki és behúzni az árbócot egy gombnyomással. A középen lévő "Állj" gomb azonnal megállítja a mozgást.
- **Manuális irányítás:** a szövegbeviteli mezőbe megadott értékre állítja az árbóc magasságát a "Beállít" gomb megnyomása után. Az itt jelenlévő "Állj" gomb megállítja a mozgatást

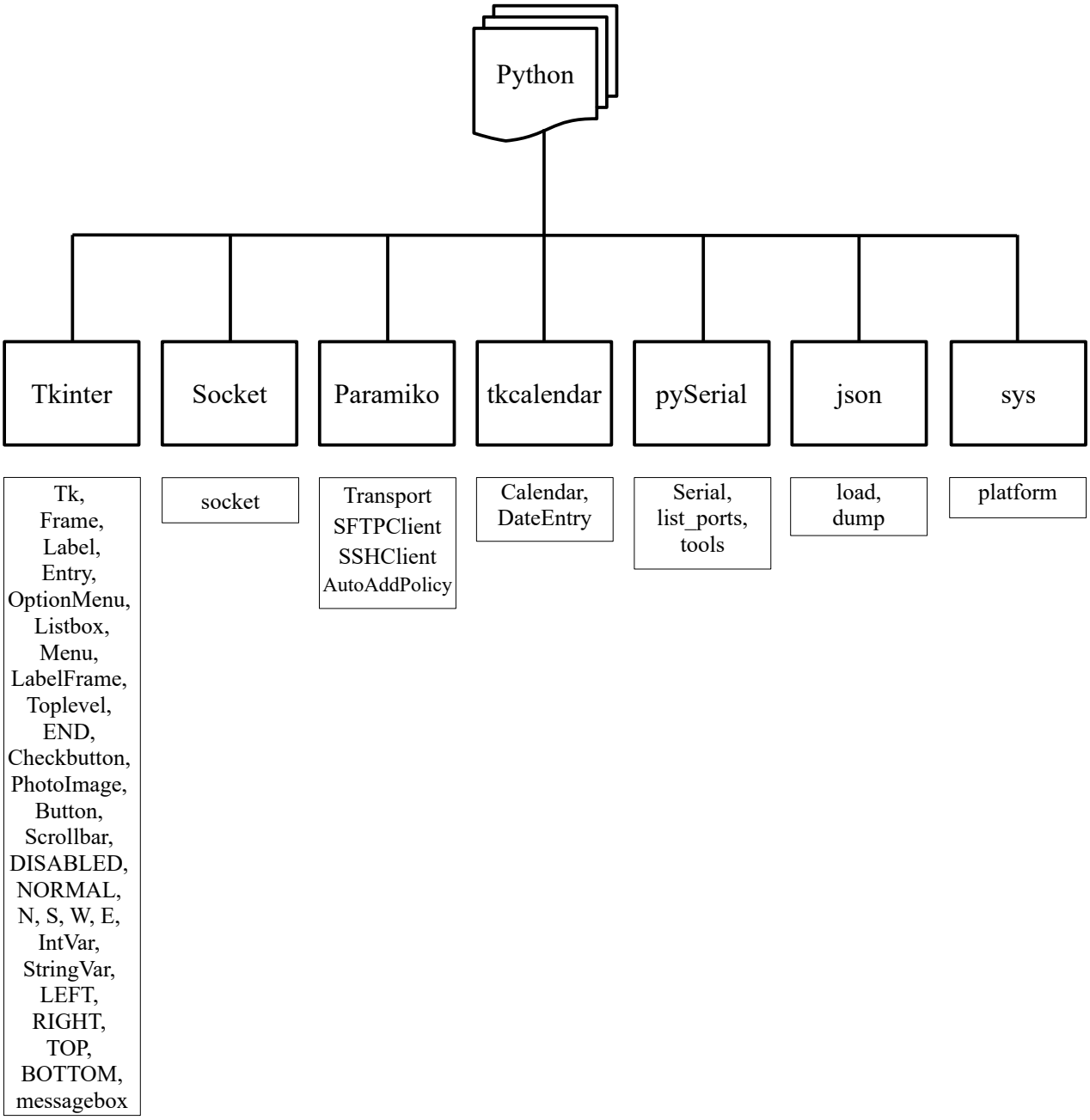
- **Soros port beállítások**: a számítógép soros portjának címe, amihez az árbóc vezérlő elektronikája csatlakozik.
- **Nézet mód**: a felületi elemek méretét befolyásolja. Érintő képernyős módban megváltoznak a gombok és beviteli mezők méretei.

Az árbóc vezérlő modul elsősorban egyedi felhasználásra készült a mérőjárműbe. Mivel ez egy érintő képernyőn folyamatosan futó program ezért úgy kellett megtervezem, hogy annak kezelése és használata kényelmes legyen az érintőkijelzős képernyőn és mindenek mellett a program stabilitása és működése ne veszélyeztesse az árbóc működését. Színvilágának illeszkedni kell a sötétben történő munkavégzéshez, ne legyen túl hivalkodó a kijelzőn.

A program a soros kapcsolat beállítása után több szálon kezd el futni. Külön programszál vezérli az elektronikát és külön szál kéri le az információkat az árbóc pozíciójáról. Emellett készült még egy szál, ami a vészleállítás miatt fontos. Bármilyen műveletet hajtson végre a vezérlő, az árbóc megállításának minden esetben magasabb prioritást kell élveznie. Ennek biztosítása érdekében a megállítás szála megszüntet minden egyéb szálat, ami a biztonsági leállítást megakadályozná. A felületen két "Állj" feliratú gomb lett elhelyezve, amik egyazon funkciót hajtják végre. A két helyre való elhelyezése azért volt indokolt, mert az ember ha egy gomb megnyomásával elindít egy folyamatot, akkor annak megállítását is az indító gomb környékén keresi. Egy esetleges vészhelyzetben nincs idő a megfelelő gomb keresgélésére, így – némi pszichológiai oknál fogva – került két megállító gomb a felület különböző pontjaira.

7. Kezelőfelület szoftveres felépítése

A program elkészítését segítette az alap könyvtárakon kívül több Python könyvtár, amiknek használatát a következő ábra szemlélteti.



Az ábrán szereplő könyvtárakon kívül még az Pythonhoz tartozó alap datetime könyvtár lett felhasználva az időformátumok konvertálásához.

A felsorolt kiegészítők és osztályok segítségével a fejlesztés rugalmasabbá tehető volt, valamint több olyan funkciót sikerült beépíteni a végső programba, ami hatékonyabbá tette a felhasználó munkáját. A fejlesztés során rengeteg idő ment el a dokumentációk olvasásával, aminek köszönhetően jobban megismertem a könyvtárak által ajánlott lehetőségeket, így sokkal hatékonyabban ki tudtam használni a funkciók által nyújtott előnyöket.

Az összehangolt funkciók megértéséhez nagyon nagy segítséget nyújtottak a hivatalos dokumentációk és wiki oldalak. Ezekből a legfontosabbakat kiemelve mutatom be, milyen részegységekből is épülnek fel a komplett programmegoldások.

7.1. A Python nyelv

A Python nyelvről könyveket lehetne írni, sokan mások meg is tették ezt korábban így csak egy nagyon rövid bemutatást írok róla a teljesség igénye nélkül.

A Python nyelv választása több szempont miatt történt. Egy olyan programnyelvre volt szükség, aminek segítségével a programom fejlesztését gyorsítják előre megírt megfelelő könyvtárak és osztályok, a program futtatása többféle architektúrán és operációs rendszeren azonosan futtatható, illetve legyen képes az a modern kor igényeinek megfelelő kapcsolatok kiépítésére a hálózati kommunikációk során. Plusz érv emellett a nyelv mellett, hogy remek dokumentációkkal rendelkezik és az online közösségi felhasználás során is gyorsan megoldást lehet találni a problémákra, kérdésekre.

Általánosságban elmondható a Python nyelvről, hogy "egy könnyen tanulható, de hatékony programozási nyelv. Magas szintű adatstruktúrái, az objektum-orientáltság egyszerű megközelítése, elegáns szintaxisa, dinamikus típusossága és interpreteres mivolta ideális script-nyelvvé teszi. Kiválóan alkalmas gyors fejlesztői munkákra, nagyobb projektek összefogására. " [8.]

A fejlesztéseim során a legfrissebb programverziót használtam, ami a Python 3 sorozatba sorolható. Az ezekhez köthető kiegészítő könyvtárakról a következő bekezdésekben írok, bemutatva a tervezés és felépítés során milyen funkcióit mire használtam belőlük.

7.2. Tkinter

A Tkinter egy grafikus keretrendszer Python alatt fejlesztett programokhoz. Maga a keretrendszer Python nyelven íródott így az egyik legnépszerűbb és leghatékonyabb megoldás

lehet, ha grafikus elemeket építünk programunkba. A hivatalos dokumentáció szerint legfőbb erényei között szerepel, hogy platformfüggetlen, így az elemek egyformán jelennek meg Windows, Linux és MacOS rendszerek alatt is. Ez tapasztalataim szerint részben igaz csak, vannak bizonyos esetekben eltérések az egyes elemek megjelenései között, amik akár szét is csúsztathatnak összetettebb szerkezeteket, de valóban egyszerűbb ablakok esetén azonos lehet a megjelenés. Az ablakok elemei alkalmazkodnak a futtató operációs rendszer beállításaihoz, annak elemi stílusait használja fel így nem ütnek el a widgetek más ablakok stílusától.

A Tkinter első sorban asztali alkalmazásokhoz alkalmas, felülete nem, vagy csak korlátozottan teszi lehetővé például érintőképernyős alkalmazások gesztusokkal vezérlését, ezért elsősorban számítógépes, egérrel történő kattintásokhoz vagy egyszerűbb érintőképernyős appokhoz ajánlott használata.

A Tkinter a Tk GUI eszközkészleten alapul, ami a Python nyelv egyik beépített csomagja. Képes objektum orientált grafikus felületeket létrehozni és ezt implementálni az akár már meglévő vagy újonnan fejlesztet applikációba.

A használata egyszerű, ezt a következő lépések is bemutatják:

1. Tkinter modul importálása
2. GUI alkalmazás főablakának létrehozása
3. Felületi elemek, widgetek hozzáadása a főablakhoz
4. Fő vezérlő hurok létrehozása az események figyeléséhez

Ezeket alapján a lépések alapján létre is lehet hozni egy egyszerű ablakot, amit egy példán keresztül be is mutatok:

```
import tkinter
foablak = tkinter.Tk()
foablak.mainloop()
```

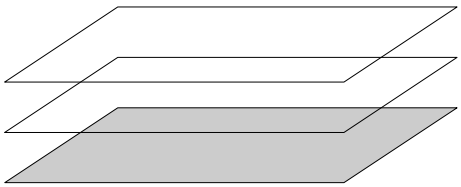
Ezzel a három sorral a futtatás után meg is jelenik egy egyszerű ablak.

A widgetek a grafikus felületre két lépésben kerülhetnek ki.

1. lépés A widgetek deklarálása, létrehozása

2. lépés A widgetek elhelyezése a képernyőn egyszerűbb esetben rögzítéssel (pack) vagy összetettebb, több elemet tartalmazó felület esetén rácsra (gridre) illesztéssel.

Amikor létrehozunk egy widgetet, meg kell adnunk a szülőjét. Ez az a widget, amelybe az új kerül. Tehát esetemben a Tk által definiált ablakra rákerülnek a Fram-ek, majd azokra az egyéb widgetek. Ennek felépítését úgy kell elképzelni, mint egymásra rakott fóliákat, amik mindig hozzá tesznek valami új elemet az alattuk levő réteghez. Mivel egy fólián több elem is szerepelhet, így a fóliák cseréjével válthatóak a felületi elemek. Ezt a logikát a 18. ábra szemlélteti.



18. Ábra: Tkinter ablak felépítésének szemléltetése réteges elven

A 18. ábrán látható felépítési elven jól átlátható, hogy is épülnek fel a programom ablakai. A legalsó szürke réteg maga a Tk() osztály által létrehozott fő ablak. Ez az alapja mindennek, amikre el vannak helyezve frame-eken keresztül a többi widgetek, elemek.

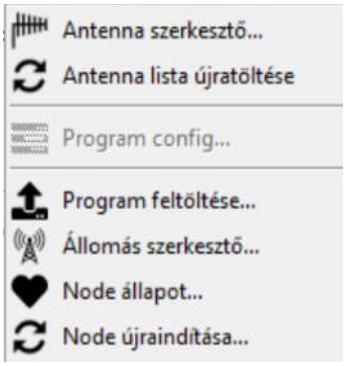
A Tkinter számos widgetet és elemeket nyújt a programozók rendelkezésére. Programomban a következőket használtam fel és a következő módokon:

- **Tk**: A fő ablak létrehozását teszi lehetővé. Ezen lesznek elhelyezve az egyes frame-k. Ennek tulajdonságai az ablak méterét és alap funkcióit tartalmazzák.
- **Frame**: A frame egyfajta réteg, amin el lehet helyezni a különböző elemeket. Több frame esetén rugalmasan lehet kezelni, hogy a fő ablak felületén milyen elemek legyenek láthatóak, vagy milyen sorrendben, pozícióban helyezkedjenek azok el. A programom minden elemét frame-ekre helyeztem, hogy a felület könnyebben átlátható és módosítható legyen későbbi fejlesztések során. Ezek a fő rácsra vannak illesztve előre kigondolt logika szerint.
- **Label**: Egyszerű szöveges mező. Szöveg kiírására használható. Beállítási paraméterei közt módosítható a karakterek mérete, betűtípusa, margója, stb. Általában az olyan

megjelenítésekhez használom, ahol valaminek a magyarázata szerepel, például, hogy egy szövegbeviteli mezőbe milyen értéket kell beilleszteni.

- **Entry**: Szövegbeviteli mező. Ennek értékének kiolvasása után mindig valamilyen string-et kapunk. Számok esetén ez módosítható a szükséges adattípussá.
- **OptionMenu**: Az opcióválasztót gyakran lenyíló menünek is szokták nevezni. Ennek a fő jelentőségét akkor használom, amikor az előre meghatározott lehetőségből egyet kell kiválasztani. Ennek elemei egy előre meghatározott lista alapján kerülnek be az opciók közé. A választott elem kinyerésére valamilyen StringVar-t kérünk le, aminek szülője az OptionMenu-t tartalmazó ablak vagy frame kell hogy legyen.
- **Listbox**: Listák létrehozására szolgál. A programomban az egyes mérési listák létrehozására és szerkesztésére használom, mivel átláthatóan lehet benne a különböző mérési konfigurációkat létrehozni. A lista elemei a program futása közben bővíthetőek, törölhetőek vagy tetszés szerint módosíthatóak.
- **Menu**: Az ablak tetején megjelenő menüsort ezzel a widgettel lehet létrehozni. Ez tartalmazza strukturált módon a program vezérléséhez szükséges lehetőségeket és innen indíthatóak a fájlműveletek is. A Tkinter által támogatott módon többféle elem tartozhat az egyes menüpontokba: elválasztók, gombok, rádió gombok vagy jelölőnégyzetek. Ezekhez ikonokat és gyorsbillentyűket rendelhetünk, melyek a menürendszer átláthatóságát és könnyedebb használatát segítik elő.
- **LabelFrame**: Olyan frame, ami magyarázó szöveggel látható el. Ennek használata során átláthatóbbá lehet tenni a felületet a sima framekhez képest.
- **Toplevel**: A toplevel előugró ablakok esetén hasznos. A szülő ablak kezeli ezeket, tehát például a főablak bezárása esetén ezek is bezáródnak. Az alkalmazások tetszőleges számú toplevelt használhatnak. Érdemes megjegyezni, hogy a név ellenére nem feltétlenül a főablak előtt kell lenniük, az ablaksorrend szabadon választható. Ezek felületeire épp úgy helyezhetőek frame-ek vagy más widgetek mint a fő Tk ablakra, függése csupán a kezelésben van a fő ablaktól.
- **END**: A Tkinterben olyan elemek esetén használatos, ahol valamilyen lista vagy többsoros beviteli mező áll a felhasználó rendelkezésére. Mindig a lista legutolsó elemére utal.

- **Checkbutton**: Jelölőnégyzetes ablakszerkezeteknél használható. Két előre definiált állapot között válthatunk vele, akár egy kapcsolóval, aminek értéke IntVar().
- **IntVar, StringVar**: Az IntVar() függvénnyel definiált változó egész számadatokat tartalmaz, amiknek értéke mindig egész szám lehet. Értékének lekérése getter és módosítása setter metódusokkal lehetséges. Ezeket a változókat különféle widget paramétereknek lehet átadni, például a RadioButton és a CheckBox változó paraméterére vagy Label textvariable paraméterére, stb. Amint ezek a változók csatlakoznak a widgetekhez, a kapcsolat mindkét irányban működik: ha az IntVar() változó megváltozik, a widget értéke is automatikusan frissül az új értékkel, ami a program interaktív működését segíti elő.
- **PhotoImage**: A Tkinterben néhány widget képes megjeleníteni képeket ikonok gyanánt. Ilyennel rendelkező Button a Menu legtöbb eleme is. Ezek a widgetek egy képargumentumot vesznek fel, amely lehetővé teszi számukra egy kép megjelenítését az adott widgetben. Így lehet elérni, hogy a menükben vagy a gombokon képből származó ábra tegye egyedibbé és átláthatóbbá a megjelenést.



19. Ábra: Példa képek megjelenítésére menüben

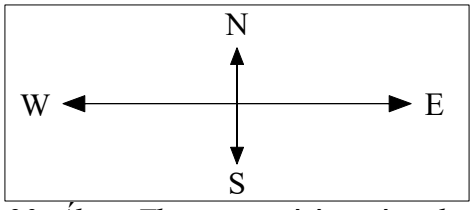
- **Button**: Gombok létrehozására alkalmas. A gombok olyan widgetek, amelyek kattintás után egy parancsot hajtanak végre. Ez a parancs értelmezhető úgy is, hogy egy előre definiált függvényt hívnak meg.
- **Scrollbar**: Az widgeten belül el nem férő elemek mozgatását Scrollbar-ral lehet elérni. Ezek a csúszkák mozgathatják például egy lista elemeit egy listboxon belül, mint ahogy azt a mérési lista megjelenítésére szolgáló widgetben teszik. A csúszka

pozíciója lehet vízszintes vagy függőleges. Ezeknek definiálására szolgál az „yscrollcommand” és az „xscrollcommand”, amik a widgeteken belüli görgetés pozíciójára adnak utasítást. Például ha egy Listboxot szeretnénk mozgatni, a következő paranccsal érhetjük ezt el:

```
listbox1 = Listbox(frame_lista1, width=27, height=25,
yscrollcommand=scrollbar_lista1.set)
```

Ez a példa azt mutatja be, hogy a frame_lista1-re elhelyezett csúszkával a scrollbar_lista1 pozícióját állítjuk be.

- **DISABLED, NORMAL**: A widgetek állapota lehet normál vagy tiltott a programomban. Ezek között válthatunk, hogy a "state" tulajdonságnak milyen értéket adjuk meg. Ezen a kettőn kívül még felvehetik bizonyos widgetek a HIDDEN vagy READ-ONLI értéket is, de ezeket nem alkalmaztam programomban.
- **N, S, W, E**: A Tkinter az égtájaknak megfelelően azonosítja a rácson belüli illesztéshez az irányokat. Ha előttünk van az ablak, akkor a 20. ábra szerint az elrendezés értelmében az ablak felső része az N (North), bal oldal a W (West), jobb oldal az E (Easth), alsó az S (South)



20. Ábra: Tkinter igazítási irányok

- **LEFT, RIGHT, TOP, BOTTOM**: A felületre illesztési módok közül a "pack" esetén meghatározza, hogy a szülő widget melyik oldalához illeszkedjen az elhelyezni kívánd widget.
- **messagebox**: A felhasználó számára előugró ablakok formájában jelenítenek meg különböző üzeneteket. Ezek lehetnek hibaüzenetek, információk vagy egyszerű eldöntendő kérdések. A következő típusokat különböztetjük meg ezekből a widgetekből:
  - *message* - általános üzenet megjelenítése
  - *showinfo* - információ megjelenítése

- *showwarning* - értesítés megjelenítése
- *showerror* - hiba megjelenítése
- *askquestion* - egyszerű, eldöntendő kérdések feltétele
- *askokcancel* - elfogadásra vagy megszakításra vonatkozó döntés
- *askretrycancel* - az askokcancel-hez hasonló, újrapróbálás vagy megszakítás opciókkal
- *askyesno* - egyszerű igen-nem eldöntendő kérdés
- *askyesnocancel* - igen, nem, megszakítás opciókat tartalmazó ablak

7.3. Socket

A modul használatával hozzáférést kaphatunk a BSD alacsony szintű socket interfészhez, amin keresztül többféle protokollnak megfelelő hálózati forgalmat vagyunk képesek generálni vagy fogadni. A modul használata során érdemes odafigyelni arra a tényre, hogy a különböző operációs rendszerek socket interfészei másképp viselkednek. Ezek az eltérések problémát okozhatnak a program futásakor így mindenképp érdemes alaposan tesztelni, szükség esetén operációs rendszereknek megfelelő megoldásokkal kiegészíteni a kapcsolat felépítését. A rendszerek megkülönböztetésére én a sys modult (7.8.) használtam.

A socket modul segítségével létrehozhatóak szerver kliens applikációk. Az egyes socketek kétirányúan kommunikálni képesek egymással egy folyamaton belül, ami lehet akár TCP vagy UDP kapcsolat is. Kettő különbségéről a 4.1. és a 4.2. fejezetben olvasható.

A socketek azonosítására és működtetésére a következők ismerete szükséges:

- **Domain**: Meghatározza az átvitelhez használt protokollcsaládot. Ez lehet például PF_UNIX, PF_INET, PF_X25,AF_INET, stb.
- **Type**: Kétféle módon határozhatjuk meg a végpontok közötti kapcsolatok típusát: lehetnek kapcsolatorientáltak vagy kapcsolat nélküliek. Kapcsolatorientált lehet például a TCP (4.1.), aminek alkalmazásánál a SOCK_STREAM értékkel kell ellátni a type opciót, míg kapcsolat nélkülinél, például az UDP (4.2.) esetén a SOCK_DGRAM-mal.
- **Protocol**: normál esetben nincs használatban ez az opció, tehát értéke nulla. Típuson belüli protokollok beállítására alkalmas.

- **Hostname**: A hálózati interfész azonosítására használatos. Egy olyan karakterláncot vár, ami tartalmazza egy számítógép host nevét, IPv4 vagy IPv6 címét. Ha nincs konkrét célunk, akkor broadcas üzeneteket is továbbíthatunk vele. Mivel esetemben csakis célzott üzenetek lettek továbbítva, így utóbbi funkciót nem volt szükséges alkalmaznom.
- **Port**: Az üzeneteknél meg kell határozni, hogy a célzott eszköz melyik hálózati porton várja az üzenetet. Ennek azonosítására meg kell adni a port számát.

Programom során a TCP és UDP protokollnak megfelelő csomagokat használtam, de az egyes hálózati eszközök felderítésére az ICMP-t (4.6.) is igénybe vettem a csomagból.

7.4. Paramiko

A Paramiko modul segítségével SSHv2 protokollnak (4.3.) megfelelő kapcsolatot lehet kiépíteni Python programokon belül, ami kliens és szerver funkciókat is képes biztosítani. A kliens API használatához létre kell hozni egy SSHClient objektumot, aminek segítségével a shell parancsok küldésén túl biztonságos fájltovábbítást is lehet végezni az SFTP protokoll (4.4.) segítségével.

Ennek a könyvtárnak a használatával valósítottam meg az állomásokkal való biztonságos összeköttetést. Egyszerű használata és stabil működése miatt a konfigurációs beállítások továbbítására is kiválóan megfelelt, aminek segítségével sikerült elkészítenem a 6.5.1. fejezetben bemutatott funkcionalitással rendelkező programmodult.

SSH kapcsolat esetén fontos a biztonsági szintek összehangolása és a két kliens kölcsönös kulcsának kezelése. Erre használhatunk előre meghatározott kulcsokat, amiket a kliens a szerverrel elcserél az automatikus azonosításhoz, de a Paramiko képes a házirendeknek megfelelően automatikusan megoldani a kulcs nélküli kapcsolatot.

```
ssh = paramiko.SSHClient()
ssh.set_missing_host_key_policy(paramiko.AutoAddPolicy())
ssh.connect(ip, username=user, password=jelszo)
```

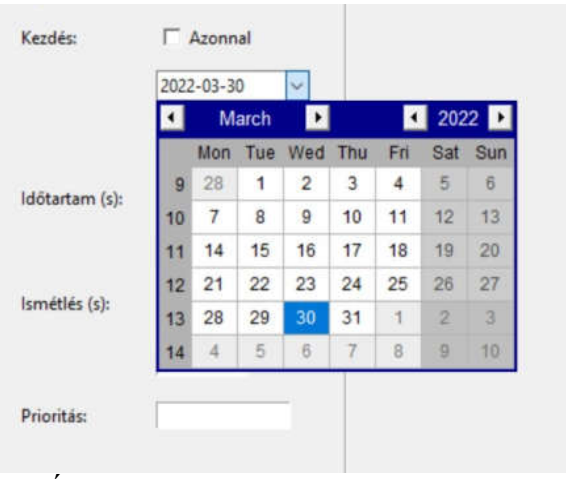
A minta programban látható, hogy egy házirend osztálynak vagy annak példányának egy alosztályát használjuk. Ez az AutoAddPolicy, ami automatikusan elvégzi a kulcsok azonosítását majd gyorsító tárazva eltárolja azokat. Ezt a házirendet csak akkor használjuk, amikor egy korábban nem ismert kiszolgálóhoz szeretnénk csatlakozni. Később már a kulcsok azonosítása nem szükséges.

Például küldjünk ki egy újraindítás parancsot a mérőműszernek az előbb létrehozott ssh objektumnak:

```
stdin, stdout, stderr = ssh.exec_command('reboot')
ssh.close()
```

7.5. tkcalendar

A tkcalendar egy olyan Python modul, ami a Tkinter alapú grafikus felületek számára biztosít naptári adatok bevitelét és megjelenítést. A widget segítségével a DateEntry típusú Tkinter widgethez hasonló formátumban, de grafikailag előnyösebb kivitelezésben lehet dátum és idő adatokat bekérni. A mérési konfigurációk során ennek használatával adhatja meg a felhasználó, hogy a mérési időzítések mikor lépjenek érvénybe.



21. Ábra: Példa a tkcalendar felhasználására

7.6. pySerial

A pySerial modul segítségével Python nyelven írt programjaink képesek lesznek elérni a soros portokat és azon kommunikációt véghez vinni a legnépszerűbb operációs rendszereken. Az osztály alapú modul segítségével hozzáférhetünk a port beállításokhoz, ahol – a soros portot kezelő programokhoz hasonlóan – beállíthatjuk a kapcsolat legalapvetőbb paramétereit, amiket a 4.7. fejezet is taglal. A modul alap esetben bináris adatokkal dolgozik, de konverzió segítségével a legtöbb adattípus átadható neki.

A 6.6. fejezetben bemutatott árbócot vezérlő programmodul kommunikációs alapját a pySerial adja. Ennek segítségével kommunikál a program az árbócot vezérlő elektronikával soros – USB konverter segítségével.

Az elérhető soros portok kilistázására a modul biztosít beépített függvényt, aminek értékeit egy OptionMenu segítségével jelenítem meg. Ezzel az opcióval lehetővé válik, hogy több elérhető soros port esetén is a nekünk megfelelő legyen kiválasztva grafikus úton. Ennek egyszerű használatát egy példán keresztül bemutatva:

```
portok = serial.tools.list_ports.comports()
```

A visszatérési érték, amit a "portok"-ba elmentünk egy lista lesz, minek elemei az elérhető kommunikációs soros portokat tartalmazza.

7.7. json

A Python egyik beépített modulja, ami a JSON fájlokkal való munkát teszi lehetővé. A JSON fájlok részletezését az 5.1. fejezet tartalmazza. Mivel a mérési konfigurációk és a programom adatbázisainak nagy részét is JSON struktúrájú adatokat használ, így ez a modul az egyik alappillére a konfiguráló program helyes működéséhez. A modul támogat többféle karakterkódolási eljárást is, aminek köszönhetően alkalmazhatóak a fájlokban a magyar ABC karakterei is.

A modul alkalmas JSON fájlok írására és olvasására egyaránt, valamint még egy hasznos funkciója, amit előszeretettel használok az a formázott adatkezelés. Ez teszi lehetővé, hogy program nélkül is átlátható módon olvashassuk a JSON fájlokat például egy szövegszerkesztőben.

Példa fájl megnyitására és kiírására a modul használatával:

```
fajlMegnyit = open("megnyitando.json", encoding="utf-8")
beolvasottAdat = json.load(fajlMegnyit)



fajlKiir = open("kiirando.json", "w")
json.dump(kiirandoDictionary, fajlKiir, sort_keys=False, indent=4)
```

A kiírással kapcsolatos példánál a "sort_keys=False, indent=4" teszi lehetővé, hogy a kiírt fájl szerkezete ne egy sorban legyen tárolva, hanem a könyvtárszerkezetnek megfelelően több sorba széttördelve és behúzásokkal ellátva. Az "indent" értéke határozza meg a sorok behúzásának mértékét.

7.8. sys

A Python sys modulja különféle funkciókat és változókat biztosít, amelyek a Python futási környezet különböző részeinek kezeléséhez használhatók fel. Hozzáférést biztosít az interpreter számára olyan változókhoz, amik az operációs rendszerektől függően változhatnak.

Ilyen változó például az operációs rendszer típusa. Ennek felhasználásával valósítottam meg azt, hogy a különböző operációs rendszereken is megfelelően fusson programom. Hogy a Linux alapú rendszerek mellett Windows-on is használható legyen, így a következő megoldással sikerült elkerülni a disztribúciók és rendszertípusok közötti különbségeket. A program indulásakor megvizsgáljuk milyen rendszert futtat a számítógép a sys modul segítségével és ennek a feltételnek megfelelően változik a program. A következő két programkóddal határozom meg, hogy Linux vagy Windows fut-e a gépen:

```
if (sys.platform == "linux"):
      ...
elif (sys.platform == "win32"):
      ...
```

7.9. datetime

A Python datetime modulja olyan osztályokat tesz elérhetővé, amikkel dátumokkal kapcsolatos manipulációs műveleteket végezhetünk el. A mérőállomások – Linux operációs rendszerükből kifolyólag – UNIX timestramp alapján konfigurálhatóak, tehát például a mérések kezdő időpontját ilyen formátumban kell megadni.

A Unix időt többféle képen is nevezik, UNIX Epoch vagy POSIX idő néven is előfordul. Lényege, hogy a Unix korszak kezdete óta eltelt másodperceket számolja, tehát azt, hogy hány másodperc telt el 1970. január 1. 00:00:00 UTC óta. [9.] Ez a szám mindig egy egész, kivéve ha a tized és ezred másodperceket is figyelembe szeretnénk venni.

A műszer beállítása tehát Unix timestam alapján történik. Az tkcalendar (7.5.) által beállított idő erre konvertálására szolgál a detetime modul a következő képen:

Dátumból időbélyeg:

```
idobelyeg = int(datetime.timestamp(datetime .strptime(datumValtozo, '%Y-%m-
%d %H:%M')))
```

A datumValtozo string típusú dátum, ami az utána megadott formátumban tárolja el a dátumot, tehát év-hónap-nap óra:perc formátumban. A kapott érték a timestamp változóba íródik, aminek típusa integer.

Időbélyegből dátum:

```
datum = datetime.fromtimestamp(idobelyeg).strftime('%Y-%m-%d %H:%M')
```

A datum string típusú időt tároló változó, melynek értékét az időbélyegből az év-hónap-nap óra:perc formátumban adja vissza az összetett függvény.

Mint látható, a két típusú időmegjelenítés között a datetime modul kényelmes és egyszerű konverziót biztosít.

8. Tesztmérés végzése

A program helyes működésének bemutatásához vegyük alapul a következő mérési feladatot.

8.1. Mérési eljárás leírása

Mérjük meg a 87,5 – 108 MHz, 230 – 380 MHz, 380 – 395 MHz és 694 – 790 MHz közötti frekvenciasávok foglaltságát Soltvadkert, Kecskemét és Kunmadaras mérőállomásokról. A mérés során a felbontási sávszélesség legyen 1,5 kHz és használjunk AGC-t a jelek megfelelő vételéhez.

A frekvencia foglaltság vizsgálata esetén SWEEP mérést kell végezni, mivel a mérendő frekvencia tartomány nagyobb, mint a műszer által maximálisan mérni képes sávszélesség. A több, jelen esetben négy darab vizsgálandó tartomány darabszáma is a SWEEP mérést teszi szükségessé, mivel IQ mérés esetén egyszerre csak egy frekvencia mérhető megadott sávszélesség mellett.

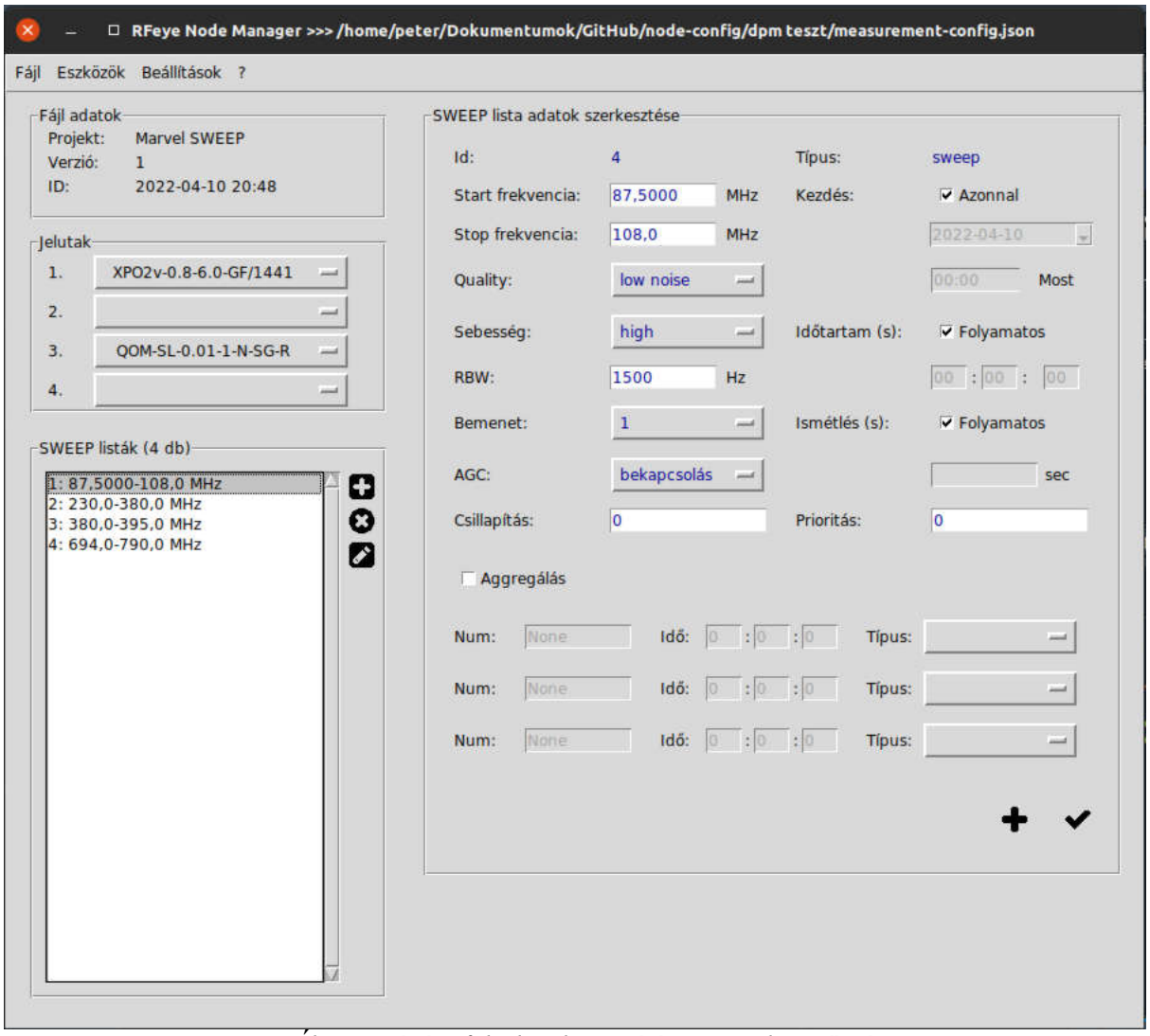
A mérés megkezdése a program feltöltése után automatikusan induljon el és időtartama mindaddig tartson, ameddig egyéb utasítást nem adunk neki. Az eredmények aggregációjára nincs szükség.

8.2. Mérés konfigurációja

Első lépésként hozzuk létre a programban egy projektet. Mivel tudjuk, hogy a mérési helyszíneken Marvell típusú RFeye állomások üzemelnek, így a 10. ábrának megfelelően Marvell típusú SWEEP projektet generálunk. Innentől a mérési paraméterek ennek az állomástípusnak megfelelően fognak létrejönni.

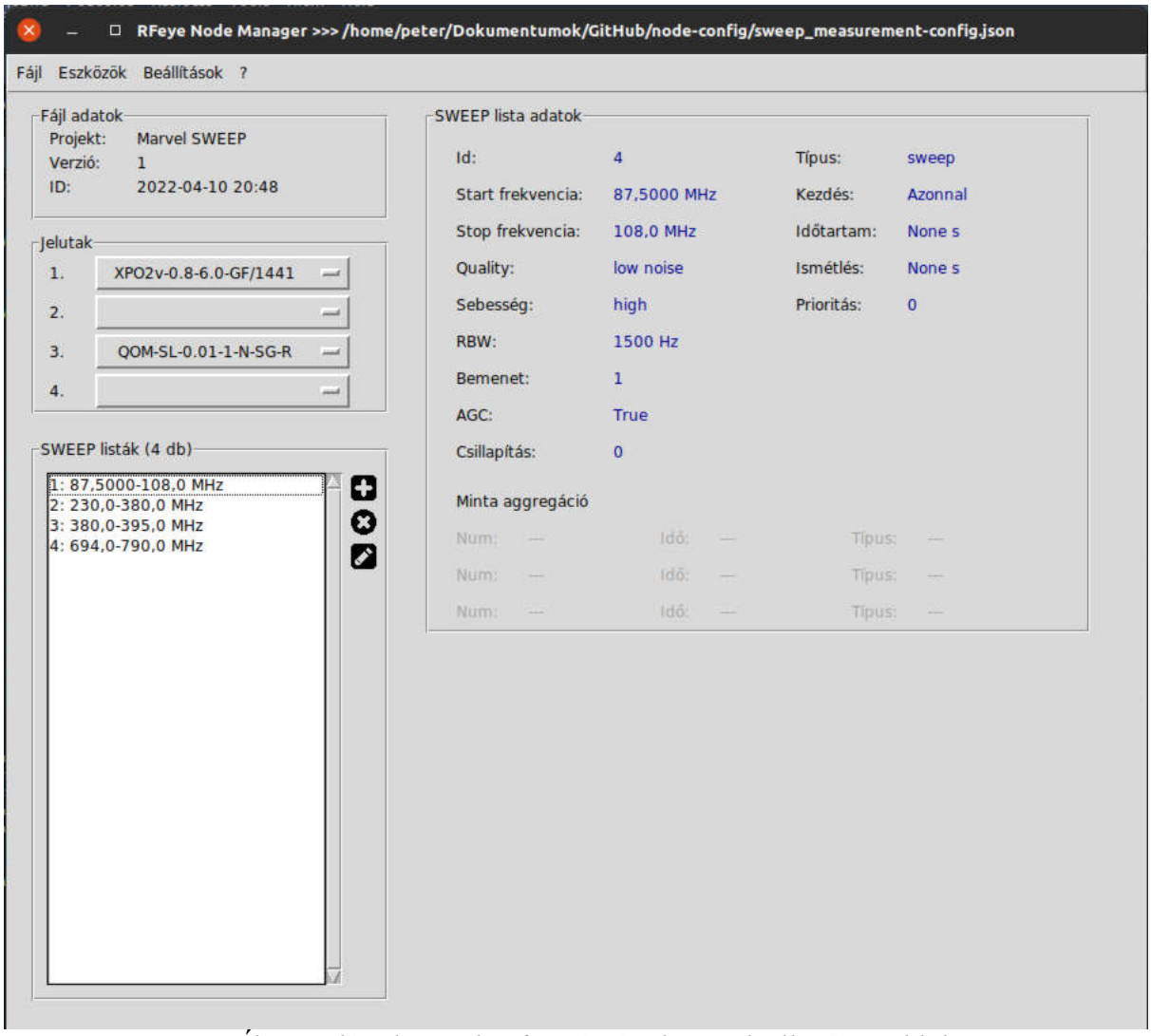
Második lépésben a használni kívánt jelutakat állítjuk be. Ezek a beállítások minden állomás esetén egyformák, mivel azok telepítése során ugyan azon jellegű antennák lettek felszerelve. A különbség a jelutak hosszából adódhat, ami a méréseket befolyásolja. Hosszabb jelút esetén nagyobb lesz a beiktatási csillapítás, így a műszer bemenetén mérhető kapocsfeszültség ennek megfelelően csökkenhet. Ennek kompenzációja a mérővevő belső szoftverében rögzítve van, mint konstans érték és nem szükséges minden alkalommal a megadása.

Harmadik lépésben létrehozzuk a mérendő frekvenciatartományok listáját a szerkesztő nézetben. Ezekben a mérési feladatokban a mérési eljárásnak megfelelően leírtakat rögzítjük, akár sávonként különböző paraméterekkel. Esetünkben négy sáv figyeléséről szól a kiírás, így ennyi listaelemet is kell létrehozni, ahogy azt a 22. ábra is mutatja.



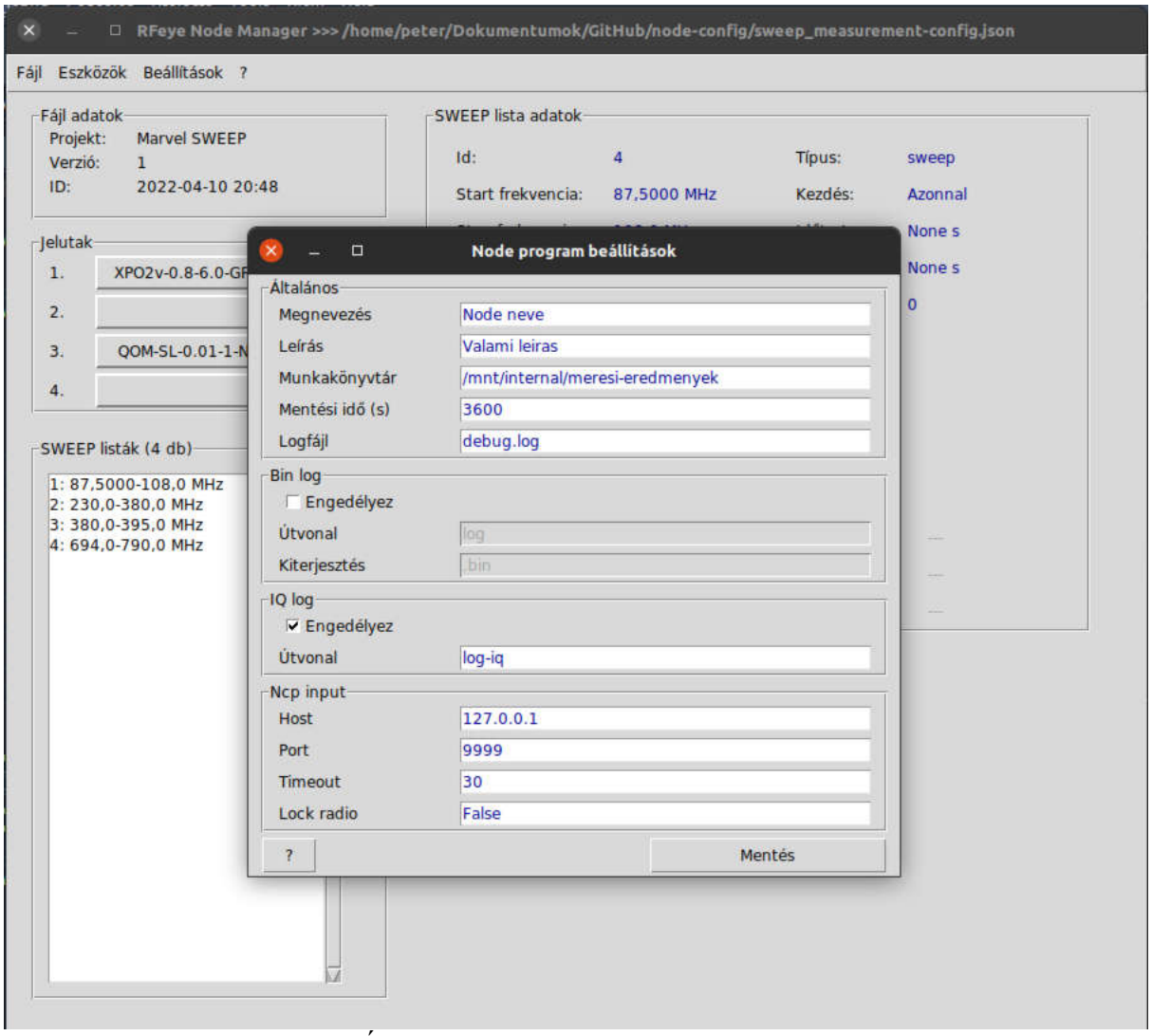
22. Ábra: Mérési feladatok rögzítése - szerkesztő nézet

A mentés gomb segítségével rögzülnek az eredmények és visszatér a program a 23. ábrán is látható adatok megjelenítéséhez. Itt már a konfigurációs fájlnak megfelelő értékek szerepelnek a mezőkben, hogy egyértelműbb legyen a műszer paramétereinek az értelmezése.



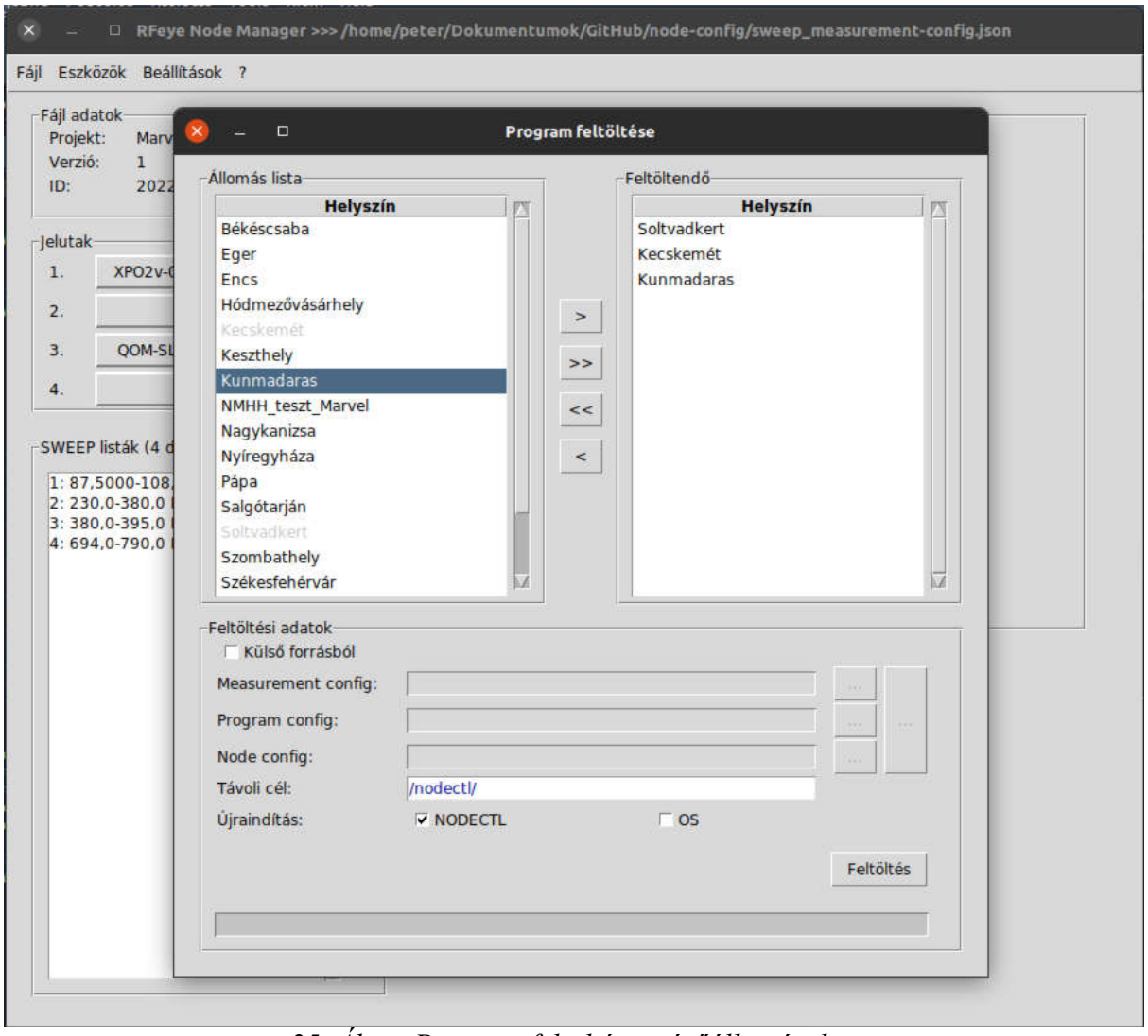
23. Ábra: A létrehozott konfiguráció adatainak ellenőrző ablaka

Az elkészített konfigurációs lista ellenőrzése után a logger program konfigurációját is ellenőrizzük le. Itt a legfontosabb paraméterek közül a logok mentésének helyét és a mentési időt állítsuk be. Az adatvesztés elkerülése érdekében érdemesebb több napos és hosszabb idejű mérések esetén is kisebbre venni a mentési időt, így elkerülve azt, hogy több órás esetleg napos eredmények vesszenek kárba egy rendszerhiba miatt. A másodpercben meghatározott értéknél az egy órás mentési idő elégséges lesz.



24. Ábra: A Logger program konfigurációja

Az utolsó lépés a mérési konfiguráció állomásra feltöltése. A "Program feltöltése" menü megnyitásakor 25. ábra szerint a bal oldali listából a három állomást – Soltvadkert, Kecskemét, Kunmadaras – kiválasztva adjuk hozzá őket a jobb oldali listához. A konfiguráció a programba be van töltve, mivel ott hoztuk létre így nem kell külső forrásból importálni azokat. A másolás célja minden állomás esetén a „/nodectl” mappa, ebben fogja újraindítás után keresni a beállításokat a logger program. A feltöltés gomb megnyomása után megvárjuk, ameddig a felugró ablakban lévő folyamatjelző jelzi a feltöltés végét és a sikeres újraindítást. Amint minden sikeresen lezajlott és a program értesít bezárhatjuk az ablakot, elkészült a mérés beállítása.



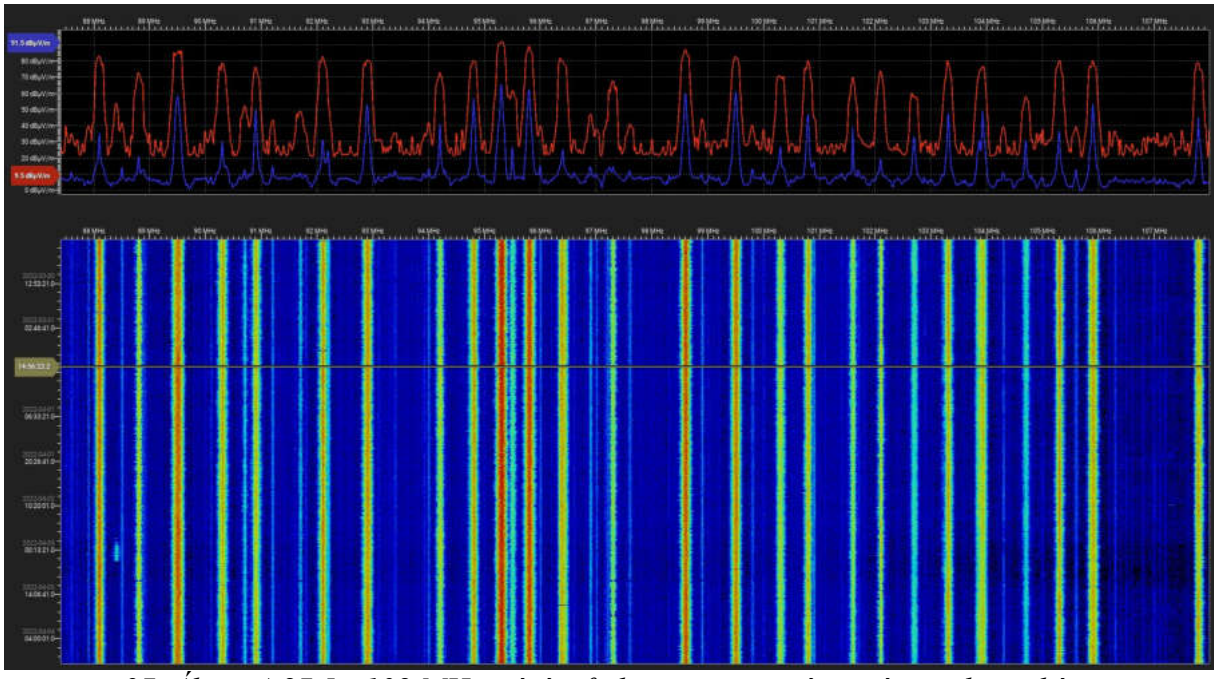
25. Ábra: Program feltöltése mérőállomásokra

A feltöltés sikerességét egy fájlkezelőn keresztül is tudjuk ellenőrizni, ha megnyitjuk a mérővevő belső memóriáját és megkeressük a mérési konfigurációkat tartalmazó mappát. Ha mind a három 5. fejezetben részletezett fájl elérhető ott a 26. ábra szerint, akkor a feltöltés sikeresen lezajlott.

26. Ábra: A feltöltött mérési konfigurációk

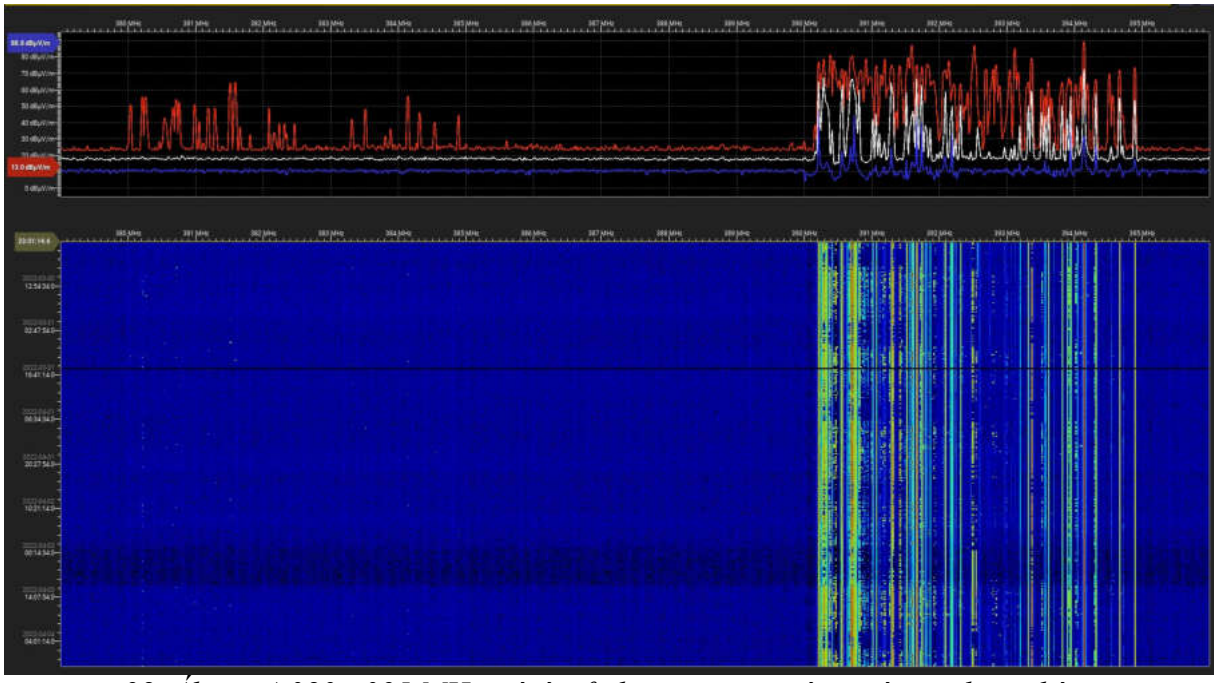
Mivel a feltöltés sikeres volt és a program nem jelzett hibát, így a mérés is sikeresen elindult. A 24. ábrán szereplő beállítás szerint a mérési eredmények az adott mappába kerülnek olyan módon, hogy a fájl folyamatos feltöltése óránként zárul le, tehát egy-egy mérési ciklus minden eltelt órában véglegesedik. Ez után a ciklusidő után (vagy a mérés közben is) a fájl megfelelő programmal való megnyitásával ellenőrizhető a mérési eredmény. A megjelenítéshez a SVALE SpecView programot használom, ami az NMHH kimondottan bináris SWEEP és IQ mérések megjelenítésére szolgáló belső fejlesztésű programja.

Szemléltetésképpen a mérési eredményekből létrehozott spektrumképekből a 87,5 – 108 MHz (27. ábra) és a 380 – 395 MHz-es (28. ábra) frekvenciasávot kiválasztva mutatom be a mérés sikerességét.

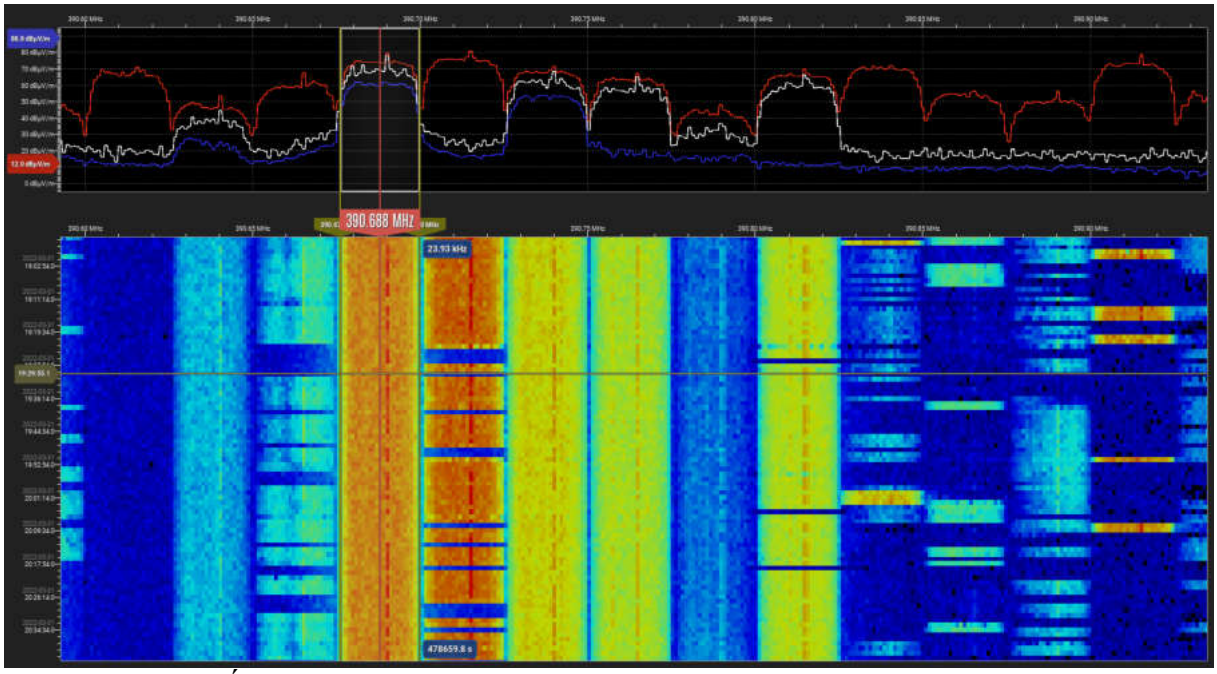


27. Ábra: A 87,5 - 108 MHz mérési frekvenciatartomány mért spektrumképe

Mit a spektrumképekről leolvasható, a mérővevő pontosan a konfigurációnak megfelelő mérési tartományban dolgozott és a beállítások során létrehozott konfigurációs értékeknek megfelelő a mérési eredmény. A mérés felbontása a 29. ábrán is látható módon az elvártaknak megfelelő mértékű, így a spektrumkép nagyítása során is jól kivehetőek a rádiófrekvenciás jelek tulajdonságai, formai jellemzőik.



28. Ábra: A 380 - 395 MHz mérési frekvenciatartomány mért spektrumképe



29. Ábra: A 380 - 395 MHz-es EDR felhasználású sáv közelített képe

9. Összefoglalás

A munkám során sikerült kifejleszteni egy olyan szoftvert, aminek köszönhetően a Nemzeti Média- és Hírközlési Hatóság Rádiómonitoring Osztályának munkája összefogható és ezáltal hatékonyabbá tehető. A fejlesztés során kikísérletezett és felhasznált módszerek, eljárások a későbbi fejlesztési irányoknak is alapot szolgálhatnak, ahol esetlegesen lehetővé vállhat további mérési funkciók kidolgozása a mérési optimalizációhoz és centralizált vezérléshez.

A tesztmérés során végbement mérési eljárás a programnak a megfelelőségét bizonyította, tehát alkalmas az országos kiegészítő mérőállomások konfigurálására. A mérési kiírásnak megfelelően a frekvenciatartományok mérése hiánytalanul lefutott, ezáltal a teszt SWEEP mérés során minden frekvenciasáv külön vizsgálhatóvá vált.

A teszt során kiderült, hogy a megfelelő mérési utasítás esetén a programom használatával a mérőállomások konfigurációja jelentős időmegtakarítással jár a konfigurálni kívánt mérőállomások számától függően. A mérési lista összeállítása és az adatok feltöltése a példában látható módon időben nem haladja meg a 3 perc 40 másodpercet, ami a korábbi manuális módhoz képest jelentősen felgyorsítja a Hatóság alkalmazottainak munkáját.

10. Ábrajegyzék

11. Irodalomjegyzék

[1.] **Andrew S. Tanenbaum:** Számítógép-hálózatok, Panem Könyvek, Taramix Kft, 2013, Budapest, 5. kiadás

[2.] **CRFS Rfeye Node 20-6 Intelligent Spectrum Monitor gyártói dokumentáció**, CRFS, 2015. szeptember

[3.] **CRFS Rfeye Node 40-8 Intelligent Spectrum Monitor gyártói dokumentáció**, CRFS, 2021. november

[4.] **datetime - Basic date and time types**: https://docs.python.org/3/library/datetime.html, letöltve 2022. 04. 02.

[5.] **Dr. Madarász László**: Soros adatátvitel, A GAMF Közleményei, Kecskemét, XIII. évfolyam (1996–97)

[6.] **ECMA-404 The JSON Data Interchange Standard**. Letöltve 2022. 03. 15.

https://www.json.org/json-en.html

[7.] **Egyszerű soros kommunikáció AVR-rel (UART)**, letöltve 2022.03.02

https://www.hobbielektronika.hu/cikkek/egyszeru_soros_kommunikacio_avr-rel_uart.html?pg=2

[8.] **ELTE IK, Programozási Nyelvek és Fordítóprogramok Tanszék**: A Python programozási nyelv, Utolsó módosítás: 2010. 05. 26 , letöltve 2022. 04. 02.

http://nyelvek.inf.elte.hu/leirasok/Python/

[9.] **Epoch & Unix Timestamp Conversion Tools**: https://unixtime.org/, letöltve 2022. 04. 02.

[10.] **Glen Turner**: Remote Serial Console HOWTO, letöltve 2022. 03. 02.

https://tldp.org/HOWTO/Remote-Serial-Console-HOWTO/

[11.] **Informatikai hálózatról magyarul** - ICMP, letöltve 2022. 02. 27.

https://net.fandom.com/hu/wiki/ICMP

[12.] **Integrált hálózati technológiák laboratórium -br/ IP telefon mérés,**

https://vik.wiki/Integrált_hálózati_technológiák_laboratórium_-br/_IP_telefon_mérés (letöltve: 2022. 02. 22)

[13.] **Introducing JSON**, https://www.json.org/json-en.html, letöltve 2022.03.02.

[14.] **Lencse Gábor: Hálózati alkalmazások**, Széchenyi István Egyetem Távkötlési Tanszék, 2008, 1. kiadás

[15.] **Lencse Gábor: Hálózati alkalmazások,** Széchenyi István Egyetem Távkötlési Tanszék, 2017, 2. kiadás

[16.] **Lencse Gábor**: IPV4 Médiakommunikációs hálózatok fóliavázlat, 2018. február 27., Budapest
https://www.tilb.sze.hu/tilb/targyak/NGB_TA007_1/05-IPv4-2018.pdf

[17.] **Paramiko**: https://docs.paramiko.org/en/stable/, letöltve 2022. 03. 28.

[18.] **Pataki Péter:** Spektrummonitoring állomás telepítése a Széchenyi István Egyetemre, 2020

[19.] **pySerial**: https://pyserial.readthedocs.io/en/latest/pyserial.html#overview letöltve 2022.04.04.

[20.] Python - GUI Programming (Tkinter), letöltve 2022. 04. 04. https://www.tutorialspoint.com/python/python_gui_programming.htm

[21.] **RS-232,** letöltve: 2022. 03. 02.

https://www.ob121.com/doku.php?id=hu:comm:bus_rs232

[22.] **RS-485**, letöltve 2022. 03. 07.

https://www.ob121.com/doku.php?id=hu:comm:bus_rs485

[23.] **Secure Shell**: https://en.wikipedia.org/wiki/Secure_Shell ,letöltve: 2022. 02. 14

[24.] **socket - Low-level networking interface**: https://docs.python.org/3/library/socket.html, letöltve 2022. 03. 29.

[25.] **SVALE Technology Kft.:** SV-BIM1 beépülő modul -felhasználói leírás V1.0

[26.] **TkDocs**: http://tkdocs.com/tutorial/

[27.] **Turányi Zoltán Richárd:** HÁLÓZATI TRENDEK, Budapesti Műszaki Egyetem Elméleti Villamosságtan Tanszék, 1996. Budapest

[28.] **Understanding TCP Flags SYN ACK RST FIN URG PSH**, letöltve: 2022. 02. 22.

https://www.howtouselinux.com/post/tcp-flags