



MISKOLCI EGYETEM
GÉPÉSZMÉRNÖKI ÉS INFORMATIKAI KAR



TUDOMÁNYOS DIÁKKÖRI DOLGOZAT

Távolról vezérelhető villamos hálózat diagnosztikai rendszer fejlesztése

Bencs Róbert

II. éves mérnök informatikus MSc hallgató

Konzulens:

Váradiné Dr. Szarka Angéla

egyetemi docens

Elektrotechnikai-Elektronikai Tanszék

Miskolc, 2013.

Tartalomjegyzék

1. Bevezetés.....	3
2. Elméleti áttekintés	4
2.1. A villamos energia minősége	4
2.1.1. Hálózati frekvencia	4
2.1.2. A tápfeszültség nagysága	5
2.1.3. A tápfeszültség változásai.....	5
2.1.4. Gyors feszültségváltozások	5
2.1.5. Tápfeszültség-letörések	6
2.1.6. A tápfeszültség kimaradásai.....	6
2.1.7. Tápfeszültség-aszimmetria	7
2.1.8. Felharmonikus feszültségek	7
2.2. Bevezetés az analóg jelek digitalizálásába	8
2.3. Számítógépes mérőrendszerek	9
3. Előzmények.....	12
3.1. A korábban használt mérőrendszer.....	12
3.2. A jelenleg használt mérőrendszer bemutatása	13
3.2.1. A mérőberendezés	13
3.2.2. A korábbi mérőprogram	17
3.2.3. A feldolgozószoftver	20
4. A távoli vezérlés tervezése	23
4.1. Motiváció	23
4.2. Követelményelemzés	23
4.2.1. Funkcionális követelmények	23
4.2.2. Nem funkcionális követelmények	23
4.3. A korábban használt megoldások bemutatása, értékelése	24
4.3.1. GPRS alapú rendszer felügyelet	24
4.3.2. HTML alapú felügyeletet biztosító függvénykönyvtár.....	24
4.3.3. Korábbi eredmények értékelése	25
4.4. A kidolgozott koncepció bemutatása	25
5. A megvalósítás részletei	27
5.1. Kommunikáció	27
5.1.1. TCP/IP	27
5.1.2. WebSocket	28
5.1.3. Nginx.....	28
5.1.4. Részletes architektúra	28
5.1.5. A rendszer elemzése adatátviteli szempontból.....	29

5.1.6. Vezérlési, információs adatok átvitele.....	30
5.1.7. Mérési adatok átvitele.....	31
5.2. Kliens oldal.....	32
5.2.1. Knockout	32
5.2.2. LESS	33
5.2.3. Toastr.....	34
6. Az új mérőprogram bemutatása	35
6.1. A rendszer legfontosabb funkciói röviden	35
6.2. Daemon	36
6.2.1. Konfiguráció	36
6.3. Monitor	36
6.3.1. Kezdőképernyő (Home)	37
6.3.2. Konfigurációs képernyő (Config).....	37
6.3.3. Aktuális mérési adatok képernyő (Live)	39
7. Irodalomjegyzék	41
8. Mellékletek.....	42
8.1. Részlet a feldolgozóprogram által generált jegyzőkönyvből	42
8.2. A kidolgozott kommunikációs protokoll üzenetei.....	43
8.2.1. Kliens → szerver irány.....	43
8.2.2. Szerver → kliens irány	43
8.3. A paraméterfájl felépítése.....	44
8.4. Az adatfájlok bináris formátuma	45

1. BEVEZETÉS

A villamos energia napjainkra életünk meghatározó részévé vált. Jelentősége mindenki számára akkor válik világossá, ha ez a – minden nap magától értetődő módon – rendelkezésre álló energia valamilyen oknál fogva rövidebb, hosszabb időre megszűnik. A fosszilis energiaforrások felhasználásának csökkentése, illetve a megújuló energiaforrások hasznosításának növelése fontos elemei a különböző Európai Unió célkitűzéseinek is, ezáltal a megújuló energiaforrások villamos hálózatokra való visszahatásának vizsgálata is nagyon aktuális és meghatározó téma. Tehát fontossága miatt elengedhetetlen, hogy a megfelelő mérések segítségével vizsgálni tudjuk a villamos energia minőségét, annak különböző paramétereit, valamint a felhasználás hatékonyságát.

A Miskolci Egyetem Elektrotechnikai-Elektronikai Tanszékén az elmúlt években, évtizedekben is folytak a villamos energia minőségével kapcsolatos kutatások, mérések, és a felhalmozódott tapasztalatok alapján nemrégiben kifejlesztésre került egy új villamos hálózat diagnosztikai rendszer. A rendszer egyik szoftverkomponensének fejlesztésében én is részt vettem korábbi TDK munkám során, ekkor az adatok utólagos kiértékelést biztosító feldolgozószoftvert fejlesztettem ki.

A rendszerrel történő ipari méréseket általában az egyetemen kívül, onnan akár több száz kilométerre végezzük, ezért szükség lenne a rendszer távoli monitorozására. Az eddigi próbálkozások során nem sikerült megfelelő megoldást találni a problémára. Céлом tehát egy olyan rendszer kifejlesztése, amely biztosítja a méréseink távoli felügyeletét, ellenőrzését.

A jelenlegi mérőrendszert és a távoli vezérlés témakörében készült próbálkozásokat értékelve arra jutottam, hogy mindenképpen szükséges már a mérőprogram elkészítése során is figyelembe venni a távoli vezérlés igényét. Ennek érdekében a jelenlegi mérőszoftver helyett egy újat terveztem, amely kialakítása során kiemelt figyelmet szenteltem a távoli elérés megvalósítására.

Az új mérőszoftvert a kliens-szerver modell alapján terveztem meg. A szerver oldali alkalmazás C# nyelven készül, a kliens oldali komponens pedig HTML5 és JavaScript nyelveken, a kommunikációt pedig a korszerű WebSocket protokoll biztosítja.

A rendszer így bármilyen eszközről elérhető, amely rendelkezik internetkapcsolattal és egy böngészővel: a hagyományos számítógépek mellett így az okostelefonok ill. tabletek is támogatottak.

2. ELMÉLETI ÁTTEKINTÉS

2.1. A villamos energia minősége [1]

Ebben a fejezetben a villamos energia minőségének vizsgálatába nyerhetünk egy kis elméleti betekintést.

A villamos energia napjainkban talán a legfontosabb alapanyag. Ez az energia egy különös árucikk, mert nem tárolható gazdaságosan nagy mennyiségben, tehát minden pillanatban annyi villamos energiát kell termelni, amennyit a fogyasztók igényelnek. Természetesen a valóságban a villamos energia lényegesen különbözik bármilyen egyéb terméktől: a felhasználás helyétől távol állítják elő, sok hasonló generátorral együtt táplálják a hálózatba, és több transzformátoron, sok kilométernyi szabadvezetéken keresztül érkezik meg a felhasználás helyére. A felhasználási helyen szolgáltatott energia minőségének biztosítása nem könnyű feladat, nincs lehetőség a szabványnak nem megfelelő villamos energia kivonására a táphálózatból, vagy arra, hogy a fogyasztó a felhasználását megtagadja.

A fogyasztók szemszögéből a probléma még sokkal nehezebb. Rendelkezésre áll néhány korlátozott mértékű statisztika a szolgáltatott energia minőségéről, de az energiaszolgáltató (és az ipari szakmai felügyelet) által elfogadhatónak értékelt minőségi szint nagyon különbözhet a fogyasztó elvárásaitól.

A legnyilvánvalóbb energiaszolgáltatási hibák a teljes feszültség kimaradások (amelyek néhány másodperctől kezdve akár több órán át tarthatnak), illetve a feszültségletörések, amikor egy rövid időre lecsökken a feszültség. A hosszú feszültség kimaradások nyilvánvalóan gondot jelentenek valamennyi fogyasztó számára, de sok technológiai művelet nagyon érzékeny még az igen rövid kimaradásokra is.

Feltehetjük a kérdést: mit értünk tehát a villamos energia minőségén? A tökéletes energiaellátás olyan lenne, amely folyamatosan rendelkezésre áll, a feszültség és frekvencia mindig a tűrési sávon belül van, a hullámalak tökéletesen szinuszos. A felhasználói alkalmazás jellegétől, a berendezésektől és a felhasználó igényétől függ, hogy a tökéleteshez képest mekkora eltérés fogadható el.

A közcélú kisfeszültségű hálózatok feszültségjellemzőivel az MSZ EN 50160 szabvány foglalkozik. A továbbiakban áttekintem a szabvány által előírt feltételeket.

2.1.1. Hálózati frekvencia

A tápfeszültség névleges frekvenciája 50 Hz legyen. Normál üzemi körülmények között az alapharmonikus frekvencia átlagértéke 10 másodpercen keresztül mérve a következő

tartományban legyen: (az együttműködő hálózathoz szinkron csatlakoztatású hálózat esetében):

$50 \text{ Hz} \pm 1\%$	(azaz $49,5 - 50,5 \text{ Hz}$)	az év 99,5%-ában,
$50 \text{ Hz} + 4\%/-6\%$	(azaz $47 - 52 \text{ Hz}$)	az idő 100%-ában.

2.1.2. A tápfeszültség nagysága

A közcélú kisfeszültség, U_n szabványos névleges feszültsége a következő legyen négyvezetékű háromfázisú hálózatokban:

$U_n = 230 \text{ V}$ a fázis és a nullvezető között mérve.

2.1.3. A tápfeszültség változásai

Normál üzemi körülmények között, a feszültségkimaradásokat figyelmen kívül hagyva

– a tápfeszültség 10 perces átlagos effektív értékei 95%-ának bármely egyhetes időszakban az $U_n \pm 10\%$ tartományban kell lennie.

– a tápfeszültség minden 10 perces átlagos effektív értékének $U_n +10\%/-15\%$ tartományban kell lennie.

2.1.4. Gyors feszültségváltozások

A gyors feszültségváltozások nagysága

A tápfeszültség gyors feszültségváltozásait elsősorban a fogyasztói villamos berendezések terhelésváltozásai vagy a hálózaton történő kapcsolások okozzák.

Normál üzemi körülmények között egy gyors feszültségváltozás általában nem haladja meg U_n 5%-át, de bizonyos körülmények között a változás, naponta néhányszor, rövid időre U_n 10%-át is elérheti.

Villogásmérték

Villogás vagy flicker definíciója: időben ingadozó fényességű vagy színképi eloszlású fényinger által létrehozott látásérzet-ingadozás hatása.

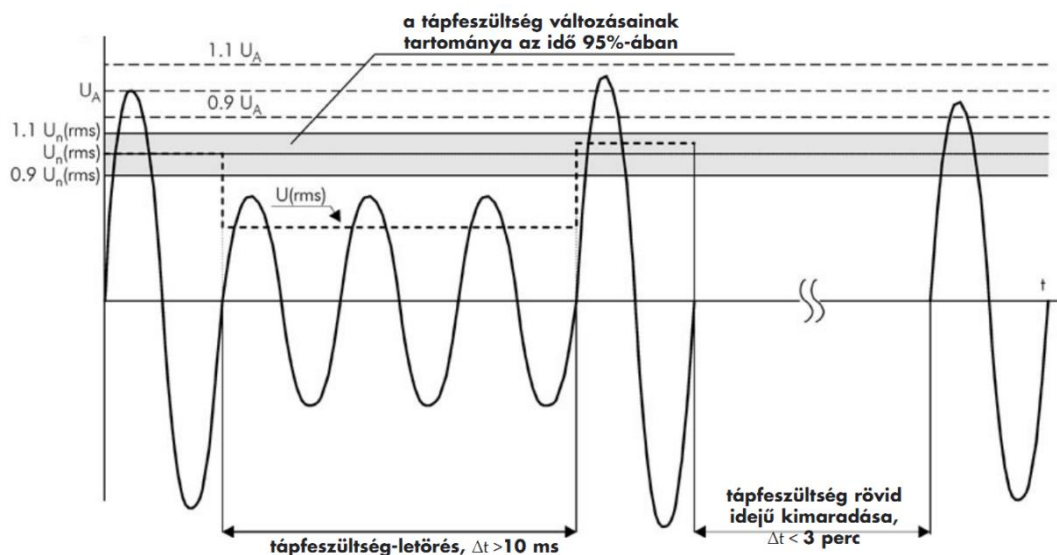
Normál üzemi körülmények között a feszültség-ingadozás által okozott hosszú idejű villogásmértéknek, bármely egyhetes időszakban, az idő 95%-ában $P_{it} \leq 1$ értékűnek kell lennie.

MEGJEGYZÉS: A villogással szembeni érzékenység szubjektív és változhat a villogás észlelt okától és a fennállásának időtartamától.

2.1.5. Tápfeszültség-letörések

A tápfeszültség hirtelen csökkenése az U_n megegyezéssel feszültség 90%-a és 1%-a közötti értékre, amit egy rövid időtartam után a feszültség visszaállása követ. A feszültségletörés szokásos ideje 10 ms és 1 perc között van. A feszültségletörés nagysága a megegyezéssel feszültség és a feszültségletörés idején előforduló legkisebb feszültség effektív értékeinek különbsége. Azok a feszültségváltozások, amelyek a tápfeszültség értékét nem csökkentik az U_n megegyezéssel feszültség 90%-a alá, nem tekinthetők letöréseknek.

A feszültségletöréseket általában a fogyasztói villamos berendezésekben vagy a közcélú elosztóhálózatokban előforduló hibák okozzák. Ezek előre nem látható, véletlenszerű események. Az éves gyakoriság változik, nagymértékben függ a táphálózat típusától és a megfigyelési ponttól. Ezen kívül az éves eloszlás is nagyon szabálytalanul változhat.



1. ábra: Feszültség-letörés és a feszültség rövid idejű kimaradásának szemléltetése

U_n – a táphálózat névleges feszültsége (effektív érték),

U_A – a tápfeszültség amplitúdója,

U_{eff} – a tápfeszültség tényleges effektív értéke

2.1.6. A tápfeszültség kimaradásai

Az az eset, amikor a csatlakozási pontokon a feszültség kisebb, mint az U_n megegyezéssel feszültség 1%-a. A táplálás kimaradása a következőképpen osztható:

- előre tervezett, az elosztóhálózaton beütemezett munkák végrehajtása céljából, amelyről a fogyasztókat előzetesen tájékoztatták, vagy
- véletlenszerű, amelyet külső eseményekkel, villamos szerkezetek hibájával vagy zavarásával kapcsolatos tartós, vagy tranzienst okoznak, amely lehet:

- hosszú idejű kimaradás (három percnél hosszabb), amelyet tartós hiba okoz;
- rövid idejű kimaradás (három percen belül), amelyet tranzienst okoz.

2.1.7. Tápfeszültség-aszimmetria

Feszültségaszimmetria definíciója: háromfázisú hálózat olyan állapota, amelynél a fázisfeszültségek effektív értékei vagy az egymást követő fázisok közötti szögek nem egyenlők egymással.

Normál üzemi körülmények között a tápfeszültség negatív fázissorrendű összetevője 10 perces átlag effektív értékeinek 95%-a, bármely egyhetes időszakban a pozitív fázissorrendű összetevő 0-tól 2%-os tartományában legyen. Egyes területeken, ahol egyes fogyasztói villamos berendezésekhez egy vagy két fázis csatlakozik, a háromfázisú csatlakozópontoknál 3%-os aszimmetria is előfordulhat.

2.1.8. Felharmonikus feszültségek

A tápfeszültség alapharmonikus frekvenciájának egész számú többszörösével megegyező frekvenciájú szinuszos feszültség. A felharmonikus feszültségeket a következőképpen lehet számítani:

- egyenként, az U_1 alapharmonikus feszültséghez viszonyítva az U_h relatív amplitúdójukat, ahol h a harmonikus rendszám (2. ábra);
- átlagolva, például a THD (Total Harmonic Distortion) teljes harmonikus torzítási tényező segítségével a következő kifejezéssel számolva:

$$THD_U = \frac{\sqrt{\sum_{h=2}^{40} (U_h)^2}}{U_1}$$

Normál üzemi körülmények között minden egyes felharmonikus feszültség tízperces átlag effektív értékeinek 95%-a, bármely egyhetes időszakban nem lehet nagyobb az alábbi táblázatban megadott értékeknél. Az egyes felharmonikusok esetében a rezonancia nagyobb feszültségeket is okozhat.

Páratlan felharmonikusok				Páros felharmonikusok	
3-nak nem többszöröse		3-nak többszöröse			
Rendszám (h)	Relatív feszültség	Rendszám (h)	Relatív feszültség	Rendszám (h)	Relatív feszültség
5	6,0 %	3	5,0 %	2	2,0 %
7	5,0 %	9	1,5 %	4	1,0 %
11	3,5 %	15	0,5 %	6 – 24	0,5 %
13	3,0 %	21	0,5 %		
17	2,0 %				
19	1,5 %				
23	1,5 %				
25	1,5 %				

2. ábra: Harmonikus feszültségek határai

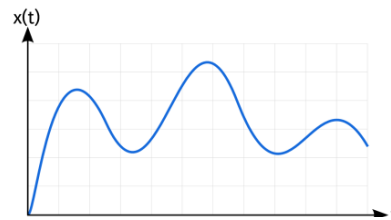
Az előzőeken túlmenően a tápfeszültség THD-értéke (beleértve az összes felharmonikust a 40-es rendszámig) nem lehet 8%-nál nagyobb.

2.2. Bevezetés az analóg jelek digitalizálásába

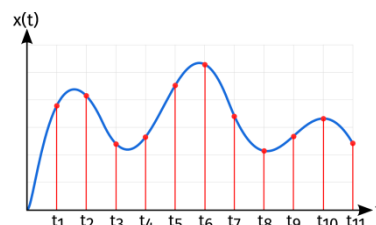
A mérendő analóg jeleket a számítógép számára digitális jelekké kell alakítani. Digitalizálás során az analóg jelet először mintavételezni, majd kvantálni kell. Mintavételezésnek nevezzük, amikor egy folytonos analóg jelből meghatározott időközönként mintát veszünk (tehát ezáltal egy időben diszkrét jelet kapunk). A kvantálás során pedig a jel értékkészletét diszkrétizáljuk.

Egy jelből olyan gyakorisággal kell mintát venni, hogy az eredeti jel reprodukálható legyen. Ezt Shannon mintavételi tételének betartásával biztosíthatjuk: a tétel kimondja, hogy a mintavételi frekvenciát (f_{mv}) úgy kell megválasztani, hogy az nagyobb legyen, mint a mintavételezett analóg jel legnagyobb frekvenciájú összetevőjének a kétszerese ($f_{jel} \max$).

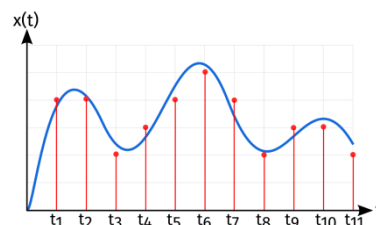
$$f_{mv} > 2 \cdot (f_{jel})_{\max}$$



3. ábra: Analóg jel

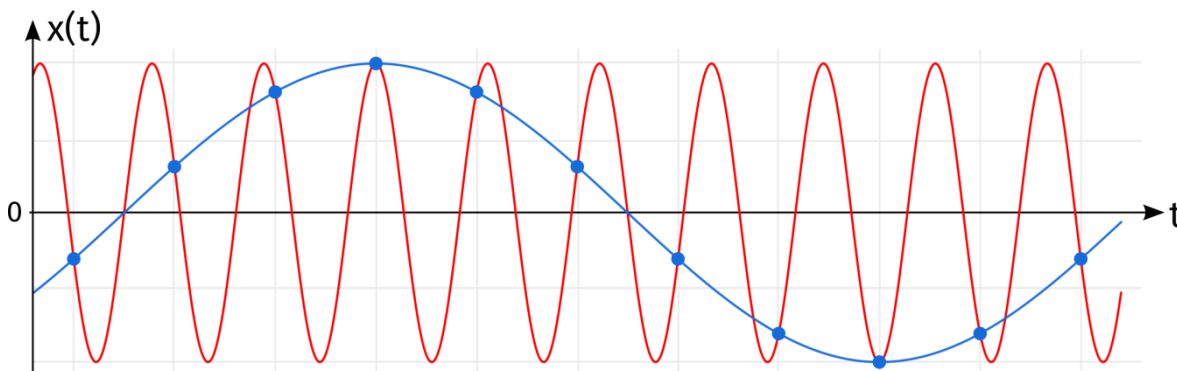


4. ábra: Mintavételezett jel



5. ábra: Digitalizált jel

Abban az esetben, ha nem tartjuk be a Shannon-törvényt, akkor a mintavételezett jelben nem létező összetevők, ún. alias jelek jelennek meg. Ezt a jelenséget nevezzük aliasing jelenségnek (6. ábra). Az aliasing ellen anti-aliasing szűrővel lehet védekezni, ami egy nagy meredekségű aluláteresztő szűrő, aminek vágási határfrekvenciája a mintavételezési frekvencia fele. A szűrő feladata értelemszerűen azon frekvenciájú jelek leszűrése, amelyekből alias jelek keletkezhetnek.



6. ábra: Az aliasing jelenség

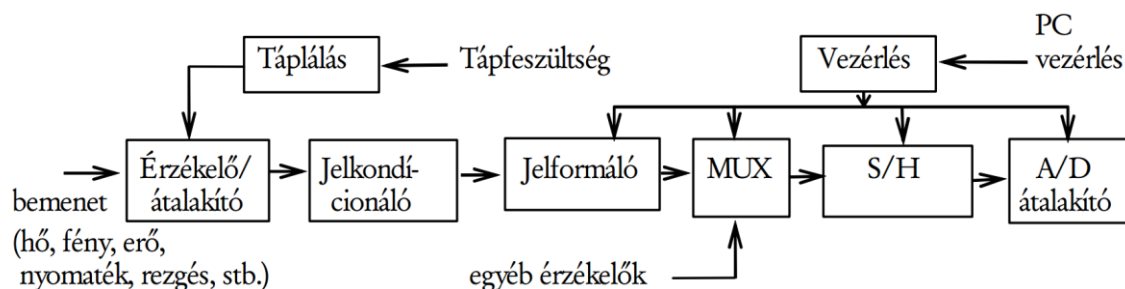
2.3. Számítógépes mérőrendszerek

[2]

A számítógép a méréstechnikában csakúgy, mint a legtöbb számítástechnikai területen a rendszer olyan eszköze, amelynek a feladata az információ összegyűjtése, feldolgozása, összegzése és rendszerezése. A számítógép alkalmazása a méréstechnika területén is sokszorosára növelte a hatékonyságot. Feladata a mérőrendszerben röviden az alábbiakban foglalható össze:

- adatkezelés: gyűjtés, tömörítés, egyszerűsítés, kiértékelés, tárolás,
- műszerek, egyéb perifériák, folyamatok vezérlése,
- mérési folyamat fejlesztés,
- dokumentálás.

A számítógéppel vezérelt mérőrendszer alapfunkciója, hogy egy fizikai/kémiai mennyiséget fogadjon, azt a számítógép által használható formára (bináris) alakítsa, szükség szerint archiválja, esetleg továbbítsa. Az alábbi ábra egy mérőrendszer funkcionális felépítését mutatja be. Minden egységnek jól meghatározható feladata van a rendszerben, jóllehet a különböző alkalmazásoknál egyik vagy másik egység hiányozhat.



7. ábra: Számítógéppel vezérelt mérőrendszer funkcionális felépítése

Érzékelő/átalakító:

Fizikai/kémiai mennyiséget érzékel és azt valamilyen ismert törvényszerűség szerint átalakítja az adatgyűjtés számára alkalmas, általában feszültségjel formára. Egyes érzékelőknek külső energiaforrásra is szükségük lehet a működésükhöz. Bizonyos érzékelők az érzékelésen túl a mérőkör leválasztását is megoldják. Ez különösen a villamos mennyiségek mérésénél lehet lényeges, mert ha például egy esetleges meghibásodás miatt túl nagy mérendő mennyiség jelenne meg az érzékelő bemenetén, akkor védelem nélkül a számítógép bemenetére olyan szintű feszültség juthat, amely a berendezés meghibásodásához vezethet. Ezért a mérőkört mindig megfelelő módon célszerű leválasztani a mérendő körről.

Jelkondicionáló:

Az átalakító kimeneti jele a számítógép analóg bemenetére kapcsolható olyan feszültség jel, melyet sok esetben a megfelelő pontosságú, minőségű átalakítás elérése érdekében „kezelni”, kondicionálni kell. A jelkondicionálás legtipikusabb példája a szűrés és erősítés. A mért jelek általában zajosak és mindig tartalmaznak olyan egyéb, nem hasznos összetevőt, amit a feldolgozás előtt szükséges kiszűrni. Gyakran fordul elő, hogy a jel nagysága nem elegendő a digitalizáláshoz. Ha például a jel a maximális bemeneti tartományhoz képest túl kicsi, akkor digitalizálásnál az A/D átalakító csak az alsó néhány bitet használja fel, ami az átalakítás pontosságát nagymértékben csökkenti. A leggyakrabban alkalmazott analóg jelkondicionálók: erősítő, osztó, zajsűrő, antialiasing sűrő.

Jelformáló:

Vannak olyan esetek, amikor a mért jelen a szükséges átalakításokat érdemes még a „számítógépen kívül” elvégezni. Ez főleg akkor hasznos, ha processzoridőt kívánunk megtakarítani, amivel a rendszer sebessége jelentősen növelhető. Processzoridőt takaríthatunk meg például egy integráló áramkör beiktatásával, vagy jelformáló lehet egy egychipes számítógép, amely segítségével a processzor terhelése jelentősen csökkenthető.

Mintavevő/tartó (Sample and hold - S/H):

A rendszernek ez az egysége végzi az adatgyűjtő berendezés analóg bemeneti csatornájára kapcsolt jel mintavételezését. A mintavételezett analóg feszültségértéket gyakran „várákoztatni” kell amíg az feldolgozásra kerül, ezért szükség van egy olyan áramkörre, amely a jelet a szükséges ideig „tartja”.

A mintavevő-tartó áramkörök a funkcióiknak megfelelően két kiemelten fontos paraméterrel rendelkeznek: az egyik a mintavételezési idő, a másik a tartási drift. A mintavételezési idő alatt azt az időt értjük, ami egy minta beolvasás vezérlőjelének kiadásától az áramkör jelkövető állapotának beállásáig eltelik. A tartási drift azt mutatja meg, hogy egységnyi idő alatt mennyit esik a tartó áramkör kimenetén a feszültség.

MUX (Multiplexer):

Ha több bemenő jelet akarunk egyidőben a számítógépbe vinni, akkor vagy több A/D átalakítót kell alkalmazni, vagy a jeleket valamilyen sorrendben, csatornánként kell behívni és átalakítani. Mindkét megoldást alkalmazzák, többségében azonban az A/D átalakítók magas ára miatt olyan sorrend kapcsolókat, ún. multiplexereket alkalmaznak, amelyek több csatornát képesek fogadni, a jeleket sorba rendezni, vezérelni és továbbítani az A/D átalakító bemenetére.

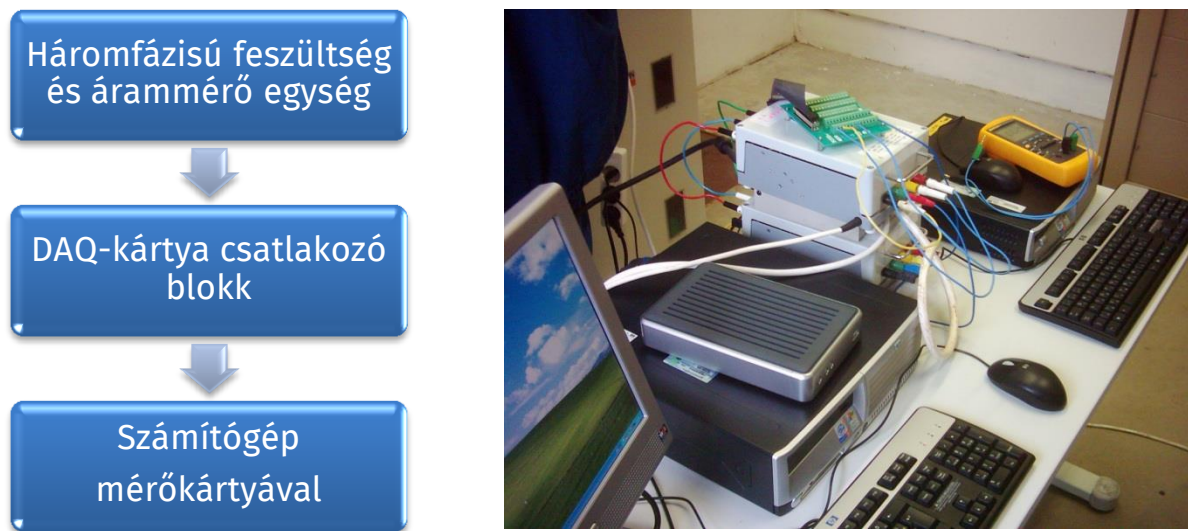
A/D átalakító:

A mintavételezett analóg feszültségértéket digitális (bináris) formára alakítja át. Működési elvük, felbontásuk, sebességük, stb. szerint nagyon sokféle A/D átalakító van forgalomban, ezek között a legelterjedtebb az ún. fokozatos közelítéses (successive approximation) elven működő átalakítók, melyek mind sebességüket, pontosságukat, mind az árukat figyelembe véve gazdaságos megoldást kínálnak a legtöbb mérési feladathoz.

3. ELŐZMÉNYEK

3.1. A korábban használt mérőrendszer

Az elmúlt időszakban a tanszék saját fejlesztésű számítógépes mérőrendszerei egy számítógépbe helyezhető, 16 bites mérőkártyát, és a hozzá kapcsolt, egyenként három fázis mérésére alkalmas feszültség- és árammérő egységeket integráltak. A rendszer blokkvázlata, illetve egy összeállított mérés látható a 8. ábrán.



8. ábra: Hagyományos mérés összeállításának blokkvázlata és felszerelése

Ezeknél a méréseknél lényeges volt a mérési körülmények és paraméterek pontos dokumentálása, mivel a rendszer nem volt képes a berendezések automatikus felismerésére, a méréshatárok és kalibrációs adatok betöltésére. A dokumentációnak tartalmaznia kellett, hogy melyik mérődobozt alkalmazzuk, milyen méréshatárban használjuk, melyik csatornára milyen jelet és milyen sorrendben kapcsolunk, illetve azt, hogy melyik dobozra milyen primer oldali feszültséget vagy áramot csatlakoztatunk. Ez igen bonyolulttá tette a manuális adminisztrációt, és jelentősen növelte a hibázás lehetőségét.

A mérések során a rendszer csak adatrögzítést végzett, a mért adatok feldolgozása, a vizsgált paraméterek kiszámítása a mérés után történt egy külön alkalmazás segítségével. Ez a program gyakran több napig futott (a mérés hosszúságától, illetve a használt csatornák számától függően), és a mérési adatok kiértékelése csak ezután kezdődhetett el.

3.2. A jelenleg használt mérőrendszer bemutatása

A Miskolci Egyetem Elektrotechnikai-Elektronikai Tanszéke az elmúlt évek, évtizedek során nagy tapasztalatra tett szert villamos hálózatok analizálása terén. A technika fejlődésével párhuzamosan mindig újabb és fejlettebb mérőrendszerek kerültek kialakításra.

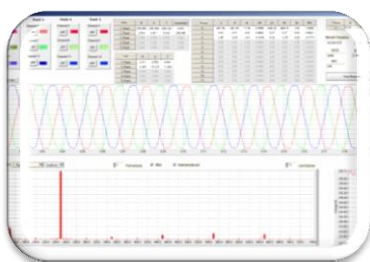
Nemrégiben is kifejlesztésre került egy teljesen új, korszerű mérőrendszer, melynek segítségével minden eddiginél pontosabban tudjuk vizsgálni a villamos energiát. Egy olyan komplex mérőrendszer került kialakításra, melynek képességei, rugalmassága messze túlmutat a konkurensekén. A munka során a *National Instruments* különböző hardver és szoftvertermékeit használtuk.

Az új mérőrendszert három részre bonthatjuk (9. ábra):

- mérőberendezés (hardver)
- mérőprogram
- feldolgozószoftver



Mérőberendezés



Mérőprogram



Feldolgozószoftver

9. ábra: Az új mérőrendszer részei

Korábbi TDK dolgozatom és diplomamunkám kapcsán találkoztam először a rendszerrel, ekkor a feldolgozószoftver elkészítése volt a feladatom.

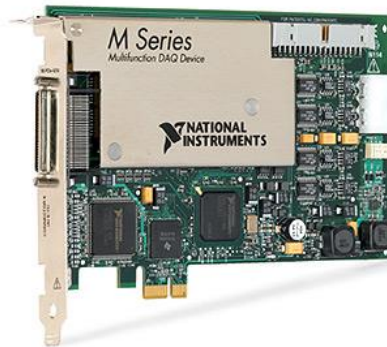
3.2.1. A mérőberendezés

[3]

A tanszéken három műszer került kialakításra, ezek különböző méretűek, illetve a legnagyobb dobozba (11. ábra) beépítésre került egy NI PCIe-6251 típusú multifunkciós mérésadatgyűjtő kártya (10. ábra) és egy komplett számítógép is, míg a kisebbek használatához szükség van egy külön PC-re is.

A rugalmasságot szem előtt tartva a műszerek moduláris felépítésűek: fiókok kerültek kialakításra (12. ábra), melyek a tényleges mérési feladatok igényei szerint cserélhetők. A műszer első felében található a feszültség- illetve árammérő fiókok, melyeket tetszőleges

sorrendben és összeállításban helyezhetünk a mérőműszerbe. Ezen modulok végzik a mért jelek átalakítását mérhető tartományú feszültségjellé.



10. ábra: NI PCIe-6251 típusú mérési adatgyűjtő kártya

Az 5 cserélhető modul különböző méréshatárokból készült, melyek a következők:

- Feszültségmérő: 5 V, 200 V, 500 V méréshatár
- Árammérő: 1,5 A és 5 A méréshatár

A mérőrendszer egy szabványos méretű műszerházba került beépítésre. Ez biztosítja a blokszerűen elhelyezkedő fiókrendszer rögzítését és lehetőséget nyújt a táp-, biztonsági-, fő- és segédberendezések elhelyezésére.



11. ábra: Mérőműszer integrált számítógéppel

Mivel a DAQ-kártya maximálisan ± 5 V feszültség fogadására alkalmas (analóg bemeneti csatornák), ezért áram- és feszültségváltók alkalmazására volt szükség, melyek a cserélhető fiókokba lettek beépítve.

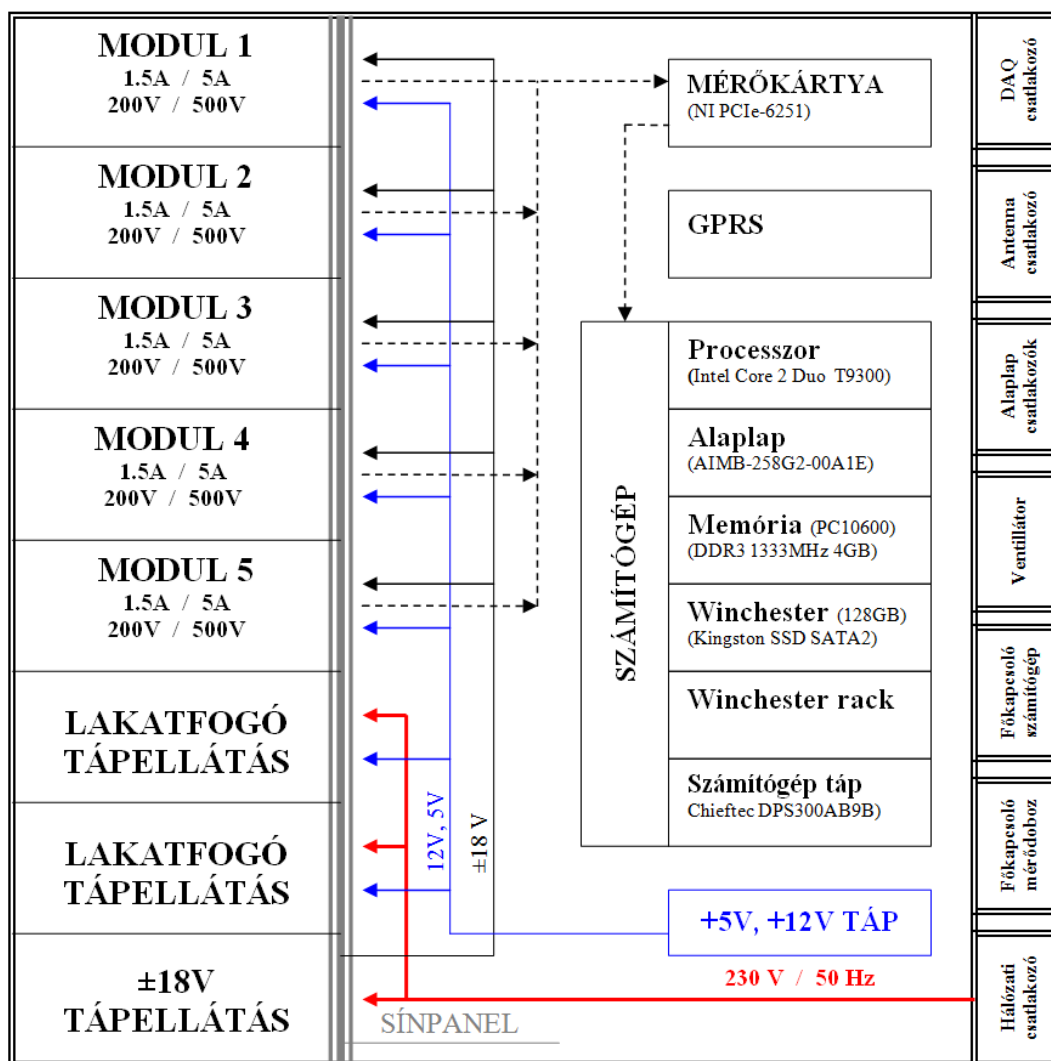
A fiókokban a LEM cég *Hall-effektuson* alapuló érzékelői találhatóak. A Hall-generátoros mérő-átalakítók a Hall effektuson alapulnak. Az Edwin Hall által 1879-ben felfedezett jelenség, mely szerint, ha egy vezetőben vagy félvezetőben áram folyik, és azt mágneses térbe helyezünk, akkor az áramot hordozó részecskékre fémeknél elektron Lorentz-erő hat, ami azzal jár, hogy a vezető két oldalán feszültségkülönbség lesz, ezt a feszültséget Hall-feszültségnek nevezik. A Hall-generátoros mérő-átalakítók elsősorban különböző villamos mennyiségek mérése területén nyertek teret, de más mechanikai mennyiségek mérésére is alkalmazzák őket, pl. szögelfordulás, nyomás mérésére. Az ilyen típusú érzékelők nagy előnye, hogy a mérendő körről galvanikus leválasztást is biztosítanak.

A rendelkezésre álló mérőkártyák maximum 16 csatorna mérésére alkalmasak, és mivel mindig háromfázisú rendszereket vizsgálunk, így a műszerek segítségével maximum $5 \cdot 3$ fázis áram és/vagy feszültség mérhető.



12. ábra: Feszültségmérő fiók

A kifejlesztésre került berendezés lényegesen szélesebb körben alkalmazható, mint a kereskedelemben kapható komplett hálózatanalizáló berendezések, mindemellett lényegesen kisebb költséggel készült, mint a konkurens berendezések. A kereskedelemben kapható analízátorok többsége nagyon drága és többségük még az érvényes szabványok előírásait sem elégítik ki, vagy ha kielégítik, akkor igen korlátozott számú jel mérhető velük (általában 1 háromfázisú rendszer, azaz összesen 6 jel). Ezzel szemben a saját fejlesztésű műszer nagyobb mintavételi sebességgel, pontosabb szinkronizálási eljárásokkal, szélesebb körű jelfeldolgozást tesz lehetővé összesen 15 jel mérésével. A műszer részletes felépítés megtekinthető az alábbi blokkvázlaton (13. ábra).

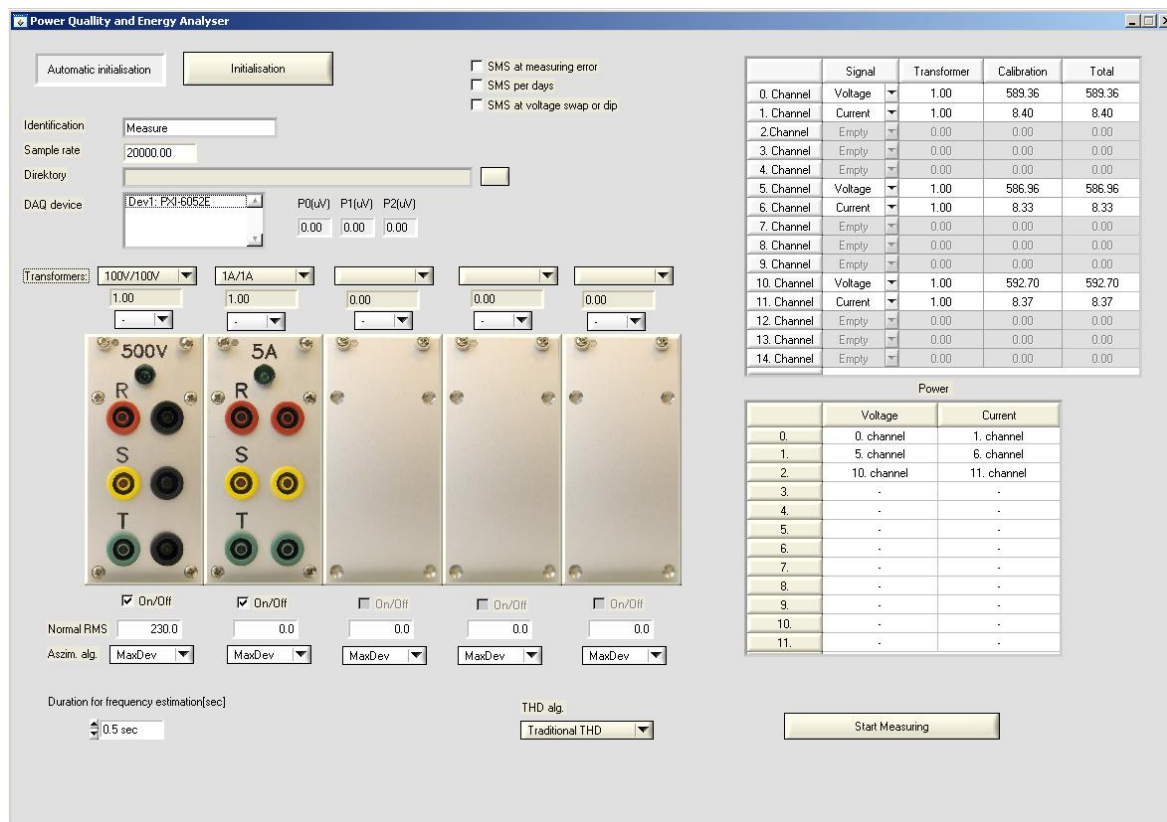


13. ábra: A műszer blokkvázlata

3.2.2. A korábbi mérőprogram

[4]

A korábbi mérőprogram a *National Instruments LabWindows/CVI* fejlesztőkörnyezetében készült. Az alábbiakban a program legfontosabb tulajdonságaival ismerkedhetünk meg. A program kezelőfelülete látható a mérés elindítása előtt a 14. ábrán, mérés közben pedig a 16. ábrán.

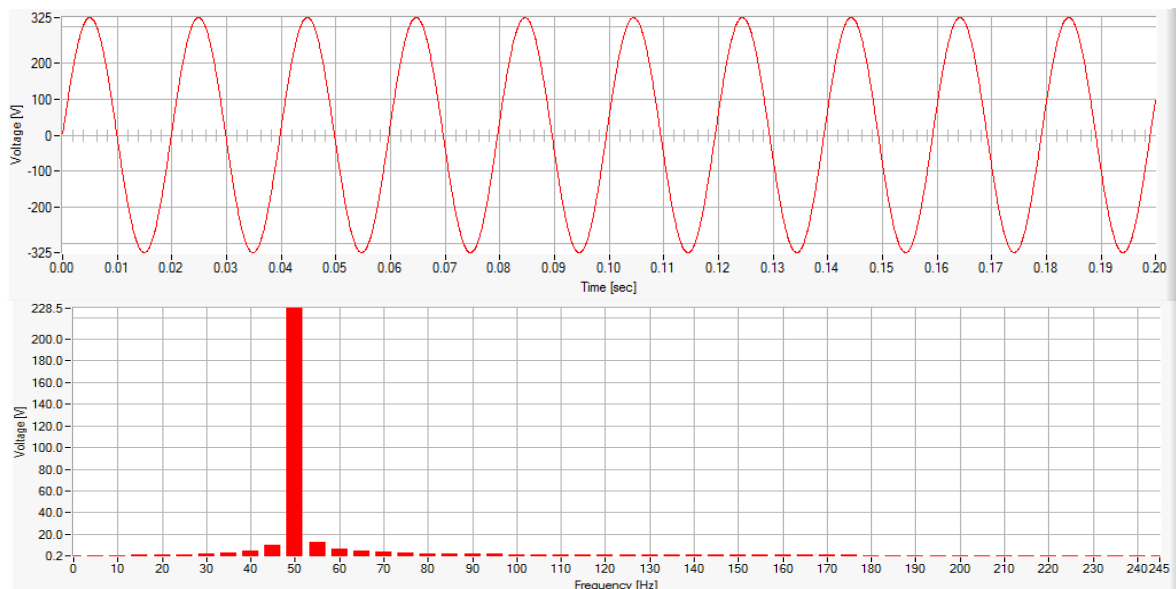


14. ábra: A korábbi mérőprogram a mérés elindítása előtt - inicializálás

Mintaszámmal történő szinkronizálás

A pontos és szabvány szerint történő villamos hálózati vizsgálat megköveteli, hogy az egyes villamos hálózat minőségét leíró paraméterek minden egész 10 hálózati periódusra legyenek meghatározva. A hálózati frekvencia ingadozása miatt azonban a 10 periódusok ideje nem állandó, tehát valamilyen módszerrel garantálnunk kell azt, hogy a mintavételezés során mindig 10 periódus legyen analizálva, különben az amplitúdó-frekvencia spektrumon ún. sátras spektrumkép alakul ki, ami hibás THD számítást eredményez (15. ábra).

Megállapítható tehát, hogy valamilyen módon szinkronizálni kell a mérést a hálózati frekvenciához. Általában ezt egy PLL áramkör beépítésével oldják meg, ami a kimenetén egy olyan órajelet állít elő, amelynek frekvenciája egész számú többszöröse a hálózati frekvenciának. A mintavételezés sebességét ez az órajel határozza meg, ami így biztosítja, hogy a 10 periódusokból mindig egy meghatározott számú minta legyen véve annak a rovására, hogy a minták között eltelt idő nem azonos.



15. ábra: Sűrűsített spektrumkép az amplitúdó-frekvencia spektrumon

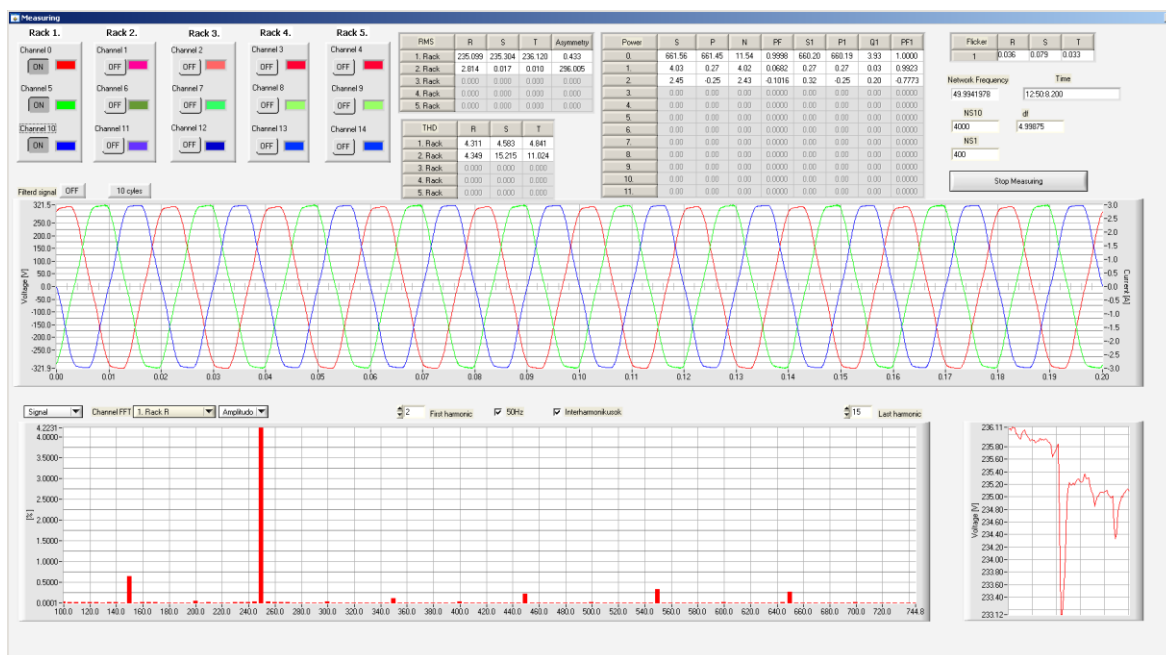
Egy másik, új módszer a *mintaszámmal történő szinkronizálás*. Ebben az esetben nem a minták közötti idő változtatásával állunk rá a 10 periódusra, hanem a minták számának a változtatásával. Kellően magas mintavételezési frekvencia lehetőséget nyújt arra, hogy a minták számának mérés közben való növelésével vagy csökkentésével mindig 10 periódus legyen analizálva a szabványok által előírt pontosságággal.

Ha ismerjük a hálózati frekvencia (f_h) pontos értékét, akkor a mintavételezési frekvencia segítségével (f_{mv}) meghatározható, hogy hány mintát ($N_{10\text{periódus}}$) kell venni ahhoz, hogy biztosan 10 periódus legyen analizálva.

$$N_{10\text{periódus}} = \left\lfloor \frac{10f_{mv}}{f_h} + 0.5 \right\rfloor$$

Paraméterek számítása mérés közben

A program másik nagy újdonsága az, hogy minden vizsgált paraméter (pl. RMS, THD) kiszámítása már a mérés alatt, valós időben megtörténik. Ez eddig úgy zajlott, hogy a mérés után, egy külön program dolgozta fel a mért adatokat. Ez a folyamat gyakran napokig is eltartott. Az új mérőprogram segítségével tehát jelentős időmegtakarítás érhető el a paraméterek online kiszámításával, viszont az így keletkezett adatok mentését is meg kell oldani.



16. ábra: A korábbi mérőprogram mérés közben

Az adatok mentése

Minden egyes mérés megkezdésekor le kell tárolni bizonyos pluszinformációkat, hogy a későbbiek során értelmezni tudjuk a lementett adatokat. Ilyen kísérő információ például, hogy milyen fiókokkal, milyen mintavételezési frekvenciával mértünk, és még sok más egyéb adat is lementésre kerül. Ezt a konfigurációs fájlt paraméterfájlnak nevezzük (kiterjesztése *.par), pontos felépítése megtalálható a mellékletek között.

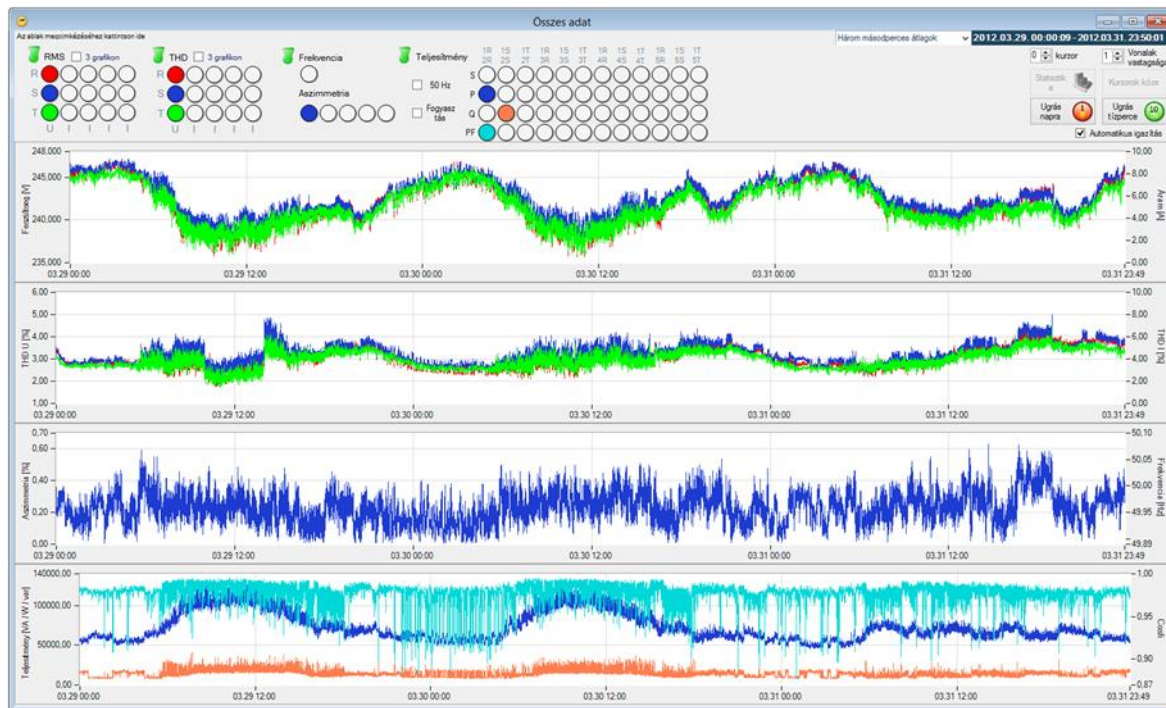
Az adatok mentése a mérés közben történik, mindig tízpercenként. Az egyes tízperces időszakok adatai mindig egy-egy új fájlba kerülnek, az ún. adatfájlokba (kiterjesztésük *.dat). Az összes csatorna jelalakjainak mentése folyamatosan történik, az összes kiszámított paraméter mentése pedig a tízperces periódusok végén. Az adatfájlok felépítése is megtekinthető a mellékletekben.

Előfordulhat az is, hogy a mérés valamilyen okból kifolyólag leáll, például a mérő számítógép tápellátása megszűnik. A rendszer ilyen helyzetek kezelésére is fel van készítve. Amikor újraindul a számítógép, a mérőprogram a paraméterfájlból elmentett információk alapján folytatja a mérést, ilyenkor új adatfájlt kezd. A feldolgozónak tehát arra is fel kell készülnie, ha hiányzik néhány adatfájl.

3.2.3. A feldolgozószoftver

[5]

A feldolgozószoftver felelős a mérőprogram által lementett adatok utólagos megjelenítéséért, kiértékeléséért. Ennek az alkalmazásnak az elkészítését én végeztem korábban TDK és diplomamunka keretében.

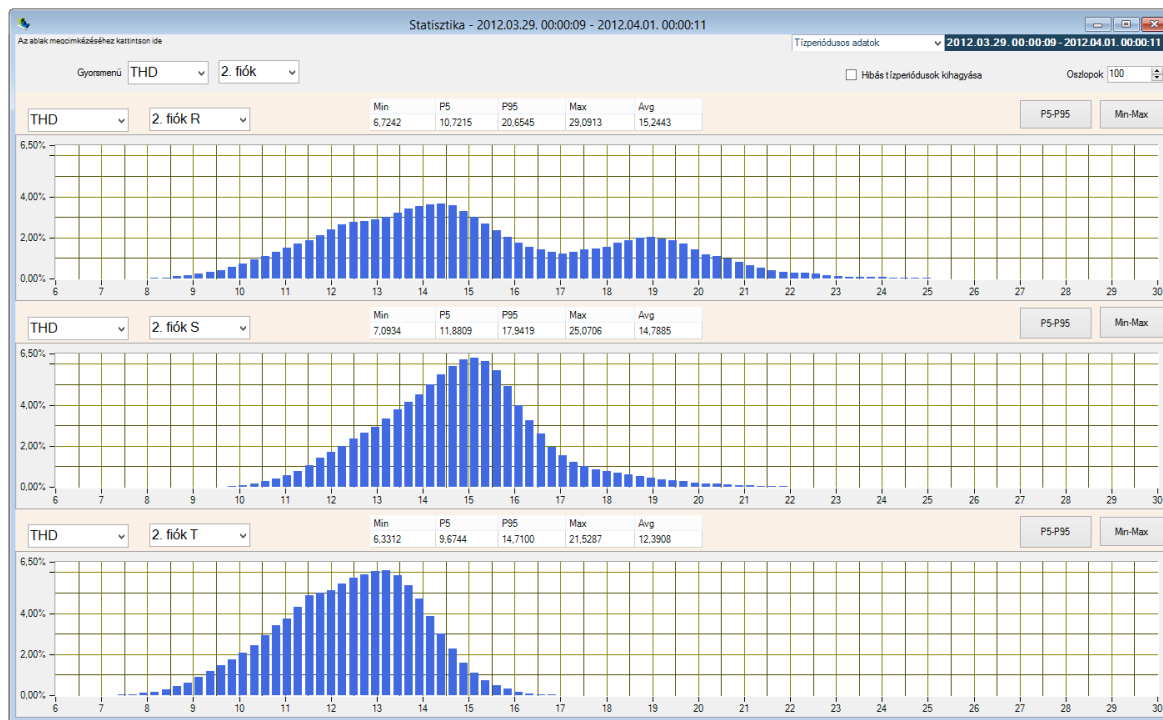


17. ábra: A feldolgozószoftver „Összes adat” képernyője

Az alábbi rövid felsorolás segítségével az elkészített alkalmazás legfőbb funkcióiba nyerhetünk betekintést.

- Grafikon ábrázolhatjuk az RMS, THD, aszimmetria, hálózati frekvencia, $\cos \varphi$ és különböző teljesítményértékeket bármely időtartományban (a teljes mérési intervallum adatai láthatóak az Összes adat képernyőn – lásd 17. ábra).
- Bármely időpillanatra megjeleníthetők a feszültség, az áram és a teljesítmény jelalakok, illetve ezek amplitúdó-frekvencia spektruma, amely segítségével leolvashatjuk az egyes felharmonikusokhoz tartozó relatív amplitúdó értékeket.
- A program tud villamosenergia-fogyasztást is számolni a kijelölt időintervallumra.
- A feszültség- és áramharmonikusok által okozott torzítási teljesítmények is elemezhetők.
- Megjeleníthetők a harmonikusok hatásos teljesítményei, egészen 2000 Hz-ig, így láthatjuk ezek időben történő változását is.
- Statisztikai analízist végezhetünk mindegyik vizsgált paraméterre (18. ábra).

- A mérés során történt különböző tranziens jelenségek (például feszültség letörések, növekedések) ki vannak gyűjtve egy külön táblázatba, illetve megtekinthetők az egyes eseményekhez tartozó RMS értékek, valamint jelalakok.
- Minden egyes feszültség-csatornához tartozó flicker értékek grafikonon ábrázolhatók.



18. ábra: A feldolgozószoftver statisztikai elemző nézete

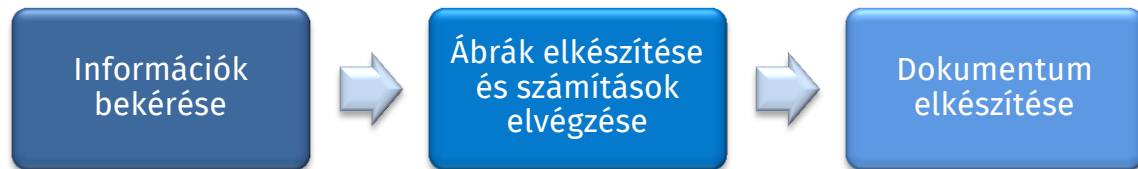
Jegyzőkönyvkészítés

Az egyik legfontosabb szolgáltatás a jegyzőkönyvkészítés. Segítségével a mérési eredményeket összefoglaló jelentés néhány gombnyomás után, teljesen automatikusan generálódik, nagyban megkönnyítve ezzel a felhasználó munkáját. (A jegyzőkönyvek elkészítése eddig manuálisan történt, a *PrintScreen* billentyű és valamilyen képszerkesztő segítségével, ez akár több napi munkát is jelenthetett.)

A program a jelentéskészítést több logikai lépésben végzi el (19. ábra). Az első lépésben egy varázsló segítségével bekérjük a felhasználóktól a szükséges információkat, mint például a melléklet címe, a csatornák megnevezései, mely napok adatait tartalmazza a dokumentum. A korábban megadott adatokból a program automatikusan ajánlatot tesz az egyes megnevezésekre, így általános esetben például a fiókok neveinek megadása után a csatornák és teljesítmények megnevezéseinek nem is kell változtatni.

Miután minden szükséges információt megadtunk, hátra is dőlhetünk, hiszen a jegyzőkönyv elkészítése ezután automatikusan zajlik.

Második lépésben a program automatikusan beolvassa a szükséges adatokat, ábrázolja ezeket grafikonon, ill. kiszámolja a szükséges statisztikai értékeket. Az összes grafikont lementi képfájlként a háttértárolóra *.png* formátumban (ha esetleg szükség lenne külön a grafikonokra is). A képek felbontása állítható, a kívánt minőség elérése érdekében. Túl nagy felbontást sem érdemes viszont beállítanunk, mert ilyenkor akár több száz MB is lehet az elkészített állomány mérete, melynek kezelése nehézkes lehet.



19. ábra: A feldolgozószoftver jegyzőkönyvkészítés funkciójának logikai lépései

Harmadik lépésben a konkrét dokumentum elkészítése történik, az előző lépésekben lementett adatok és ábrák alapján. A jegyzőkönyv *.docx* formátumban készül el. Egy elkészített jegyzőkönyvből [6] láthatunk egy rövid mintát a 8 mellékletben.

A dokumentum elkészítése gombnyomásra, de nem pillanatok alatt történik. A jelentéskészítő rutin lefutásához szükséges idő azonban nagymértékben függ attól, hogy hány csatorna adatait kell feldolgozni, valamint attól, hogy milyen hosszú mérési időszakokat jelöltünk ki. Természetesen fontos a programot futtató számítógép teljesítménye is. Például egy 15 csatornás mérésből készített 7 nap adatait tartalmazó dokumentum körülbelül 5 perc alatt készült el egy viszonylag gyorsnak mondható 4 magos számítógépen.

4. A TÁVOLI VEZÉRLÉS TERVEZÉSE

[7]

4.1. Motiváció

A villamos hálózat diagnosztikai mérések sokszor az ország távoli részében, akár több száz kilométerre az egyetemtől történnek, ezért szükség lenne a mérőrendszer távoli felügyeletének biztosítására.

4.2. Követelményelemzés

Első lépésként összegyűjtöttem és elemeztem, hogy a távoli vezérlés témakörében pontosan milyen követelmények megvalósítása szükséges. Ehhez nagy hasznosnak bizonyult az, hogy korábban már személyesen is részt vettem méréseken, de természetesen konzultáltam a leendő felhasználókkal is.

4.2.1. Funkcionális követelmények

- *Riasztások küldése:* az első és legfontosabb igény, hogy a felhasználót valamilyen módon értesítse a rendszer bizonyos események bekövetkezésekor, a rendszer küldjön értesítést a kritikus mérési ill. hálózati hibákról.
- *Aktuális mérési adatok lekérdezése:* a futó mérés során a távoli felhasználó lássa az aktuális mérési adatokat, paramétereket.
- *Beavatkozás:* előfordulhat olyan eset is, amikor elkerülhetetlen az egyes mérési paraméterek módosítása, a mérés leállítása ill. újraindítása. Ezért biztosítani kell a rendszer teljes körű távolról vezérelhetőségét is.

4.2.2. Nem funkcionális követelmények

- *Biztonságos működés:* fontos követelmény a biztonságos kommunikáció ill. a rendszer védelme az illetéktelen felhasználókkal szemben. Ez alatt elsősorban azt értem, hogy jogosulatlan egyének ne befolyásolhassák a rendszer működését, hiszen ezzel anyagi károkat is okozhatnak.
- *Egyszerű használhatóság:* lehetőség szerint ne kelljen külön kliens programot telepíteni a felhasználó számítógépére, hanem egy egyszerű böngészőn keresztül is el lehessen érni a rendszert. Továbbá legyen egyszerű és letisztult a felhasználói felület.

4.3. A korábban használt megoldások bemutatása, értékelése

A rendszer távoli felügyeletének igénye már korábban is felmerült a tanszék dolgozóinak körében. A megvalósításra is történtek már különböző próbálkozások, ebben a részben röviden áttekintem, majd értékelem ezeket.

4.3.1. GPRS alapú rendszer felügyelet

A mérőberendezésbe beépítésre került a SIMCom cég SIM900 típusú GPRS modulja [8]. Ennek segítségével jelenleg kétféle felügyeleti lehetőség biztosított: SMS és FTP alapú [3].

A program képes SMS-t küldeni az alábbi események bekövetkezésekor:

- a mérés valamilyen hiba következtében leállt,
- tranziens jelenség észlelésekor (például feszültségletörés, növekedés),
- bizonyos, előre konfigurált időintervallumonként (így biztosítva az üzemeltetőt arról, hogy a rendszer működik).



20. ábra: A mérődobozba épített GSM/GPRS modul

Másik lehetőség, hogy egy tárhelyre bizonyos információkat küldjön a program FTP kapcsolat segítségével, meghatározott időközönként.

A módszer legnagyobb hátránya az, hogy semmilyen beavatkozási lehetőséget nem biztosít a rendszer üzemeltetőjének, csak a rendszer állapotáról biztosít (korlátozott körű) információkat.

4.3.2. HTML alapú felügyeletet biztosító függvénykönyvtár

A jelenlegi mérőprogramhoz csatolható függvénykönyvtár készítése volt az egyik diplomatervező hallgató feladata a korábbi években [9]. A mérőprogram mellé integrált egy webszervert, amely kiszolgálja a távoli HTML klienseket, a kommunikáció pedig a HTTP protokoll segítségével történik.

A módszer hátrányai:

- ez sem biztosít teljes körű elérést,
- egy-egy új szolgáltatás beépítésekor mindkét felületet (külön a mérőszoftver és külön a távoli kliens felületét) módosítani kell, így többletmunkát jelent,
- a két felület bár hasonlít egymásra, a különbségek zavarhatják a felhasználót,
- a munka nem készült el teljesen, nincs letesztelve, kipróbálva.

4.3.3. Korábbi eredmények értékelése

A bemutatott módszereket tanulmányozva arra jutottam, hogy egyik sem felel meg teljes mértékben az azonosított követelményeknek, a módszerek legnagyobb hibája, hogy nem teljes körűek, ill. a távoli felügyelet lehetőségét csak egy plusz, korlátozott lehetőségként integrálták a rendszerbe, a helyi felügyelet mellett.

A teljes körű felügyelet biztosításának érdekében már a mérőszoftver tervezésekor is szükséges figyelembe venni a távoli elérés igényét, hogy megfelelően biztosítani tudjuk az azonosított követelmények megfelelő megvalósítását.

4.4. A kidolgozott koncepció bemutatása

A mérőprogramot a távoli adminisztráció hatékony megvalósítása érdekében a kliens-szerver architektúra alapján felbontottam funkciók alapján két komponensre (lásd 21. ábra):

- *Monitor*: a mérés konfigurálásáért, ill. a mérési adatok valósídejű megjelenítéséért felel, gyakorlatilag a kliens szerepét tölti be.
- *Daemon*: minden mást, a tényleges működést végzi, és szerverként kiszolgálja a monitort adatokkal.



21. ábra: A rendszer logikai felépítése

A két komponens pedig hálózati kapcsolaton keresztül kommunikál egymással, ezáltal biztosított mind a távoli, mind a helyi felügyelet lehetősége is egyben. A megoldás másik nagy előnye, hogy az erőforrásigényes megjelenítési feladatok nem terhelik a szerveret, mivel ezek elvégzése a kliens feladata.

Az előző mérőrendszerrel megoldással szemben az a különbség, hogy itt nem egy plusz lehetőségként, úgymond extraként építettem a rendszerbe a távoli vezérlést, hanem ennek ismeretében terveztem meg a teljes architektúrát, ezért sokkal hatékonyabb ennek működése.

A *Monitor* alkalmazás tervezésekor fontos szempont volt, hogy minél egyszerűbb elérhető legyen a rendszer, lehetőség szerint ne kelljen külön kliens programot telepíteni, ezért általánosan elterjedt, modern webes technológiákat (HTML, JavaScript, CSS) alkalmaztam. Ezáltal a *Monitor* alkalmazás platformfüggetlen, használatához mindössze egy modern böngészőre van szükség, elérhető tehát akár asztali számítógépről, tabletről vagy okostelefonról is.



22. ábra: A *Monitor* alkalmazás elkészítéséhez használt modern webes technológiák

5. A MEGVALÓSÍTÁS RÉSZLETEI

Ebben a fejezetben betekintést nyerhetünk a rendszer megvalósításának részleteibe, a felhasznált technológiákba, eszközökbe.

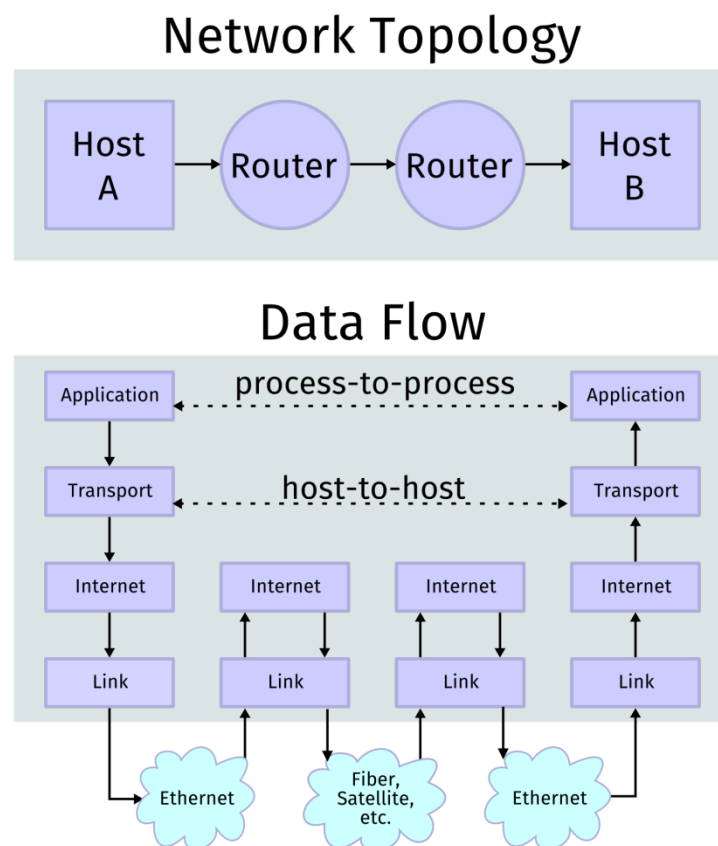
5.1. Kommunikáció

5.1.1. TCP/IP

[10]

A TCP/IP alatt hálózati protokollok gyűjteményét értjük, nevét két legfontosabb protokolljáról, a Transmission Control Protocol-ról (TCP) és Internet Protocol-ról (IP) kapta.

A TCP/IP rétegekből áll. Minden réteg egy jól definiált feladatot végez el, és a rétegek egymás között szolgálatelérési pontokon keresztül kommunikálnak. Minden réteg csak a szomszédos rétegekkel képes kommunikálni. A TCP/IP-ben négy, ill. öt réteget különböztetnek meg: adatkapcsolati, hálózati, szállítási és alkalmazási réteg (az adatkapcsolati rétegen belül egyes források megkülönböztetnek fizikai réteget is). A kommunikáció menetét mutatja be a 23. ábra.



23. ábra: A TCP/IP rétegei és a kommunikáció [10]

5.1.2. WebSocket

A WebSocket [11] egy viszonylag új internetes technológia, ami kétirányú, duplex kommunikációs csatornák kiépítését teszi lehetővé egyetlen TCP socketen keresztül. Kifejlesztésének fő motivációja volt, hogy a webböngészőben futó alkalmazások képesek legyenek a szerverrel való kétirányú kommunikációra különböző kerülőmegoldások nélkül, a HTTP szabványból eredő korlátozások kiküszöbölésével. [12]

Lehetőség van a kapcsolat teljes körű titkosítására SSL/TLS segítségével, így biztonságos kommunikációs csatornát nyitva a komponensek között.

5.1.3. Nginx

Az Nginx [13] egy ingyenes, nyílt forráskódú HTTP ill. reverse proxy szerver. A NetCraft 2013. októberi felmérése szerint a világon a harmadik legnépszerűbb webservert, a világ összes honlapjának több mint 16%-át Nginx szerverrel szolgálják ki [14].

Az Nginx egyik legnagyobb előnyeként a gyorsaságát szokták megemlíteni. Ennek az egyik oka az, hogy a konkurensével ellentétben (legfőképpen Apache) nem hoz létre minden új kapcsolathoz új szálat vagy processzt, hanem egy aszinkron eseményvezérelt megközelítést alkalmaz, ez többek között sokkal kisebb és kiszámíthatóbb memóriahasználatot eredményez.

Nem elhanyagolható előnye még, hogy konfigurációs fájljai egyszerűek, jól átláthatóak, így nagyon egyszerűen konfigurálható egy komplett webservert.

Az Nginx Unix rendszereken fut a legoptimálisabban, leggyorsabban, azonban létezik Windows-os változata is, dokumentációja szerint néhány funkciót nem támogat, nem skálázódik is olyan jól ill. kisebb teljesítményű is, leginkább csak fejlesztési célra ajánlják Windowson.

Mivel azonban nekünk nincs szükség túlzottan nagy teljesítményre és skálázhatóságra (az egyidejűleg csatlakozott kliensek száma 1-3 lesz maximum), ill. a fejlesztés és tesztelés közben borzasztóan stabilnak bizonyult, valamint nem találtam hasonló kvalitásokkal bíró alternatívát, ezért az Nginx használata mellett döntöttem Windows rendszeren is.

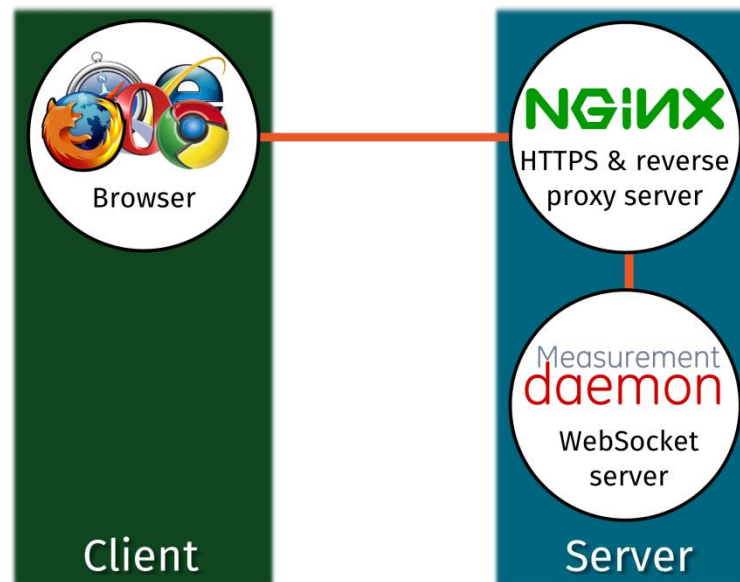
5.1.4. Részletes architektúra

A kliensek a szerveren közvetlenül csak az Nginx-et érik el, biztonságos, HTTPS csatornán keresztül.

Az Nginx két URL-re „válaszol”:

/Measurement: a webalkalmazás innen érhető el, azaz a klienseknek ezt a címet kell a böngészőjük címsorába írni. Például ha a szerver IP címe 192.168.1.48, akkor a megfelelő URL: <https://192.168.1.48/Measurement>

/mrdaemon_service: az erre az URL-re érkező kéréseket (a kliens számára transzparens módon) továbbítja a Daemon alkalmazásnak (azaz ebben az esetben az Nginx reverse proxyként funkcionál).



A megoldás előnye, hogy így elég a szerveren csak egy portot megnyitni (az webszerver által használt portot, célszerűen a HTTPS-hez tartozó 443-as portot), valamint így a kapcsolat titkosítását teljes egészében az Nginx végzi, nincs szükség a Daemon alkalmazást külön erre konfigurálni.

5.1.5. A rendszer elemzése adatátviteli szempontból

Alapvetően kétféle kommunikáció szükséges a komponensek (*Daemon* és *Monitor*) között:

- a) Vezérlés, információcsere:
 - mindkét irányban
 - jellemzően ritkán
 - rövid, szöveges üzenetek
- b) Mérési adatok átvitele:
 - csak szerver → kliens irányban
 - nagy mennyiségű adatátvitel



24. ábra: A kommunikáció típusai és irányai a rendszer komponensei között

A következőkben részletesen bemutatom a kétféle kommunikáció megvalósítását, működését.

5.1.6. Vezérlési, információs adatok átvitele

Mivel a kliens oldalon a JavaScript nyelvet használom, melyben nagyon kényelmes a JSON objektumok kezelése, és szerver oldalon is rendelkezésre állnak megfelelő eszközök használatához, ezért célszerű volt ezt a nyelvet választani a kommunikációra.

A JSON (JavaScript Object Notation) egy kis méretű, szöveg alapú szabvány ember által is olvasható adatcserére. A JavaScript nyelvből alakult ki egyszerű adatstruktúrák és asszociatív tömbök reprezentálására. A JavaScripttel való kapcsolata ellenére nyelvfüggetlen, több nyelvhez is van értelmezője. Általánosságban strukturált adatok tárolására, továbbítására szolgál. A JSON-t legtöbbször egy szerver és egy kliens számítógép közti adatátvitelre használják, az XML egyik alternatívájaként. A JSON előnyeként az egyszerűségét, és kisebb méretét szokták emlegetni.

Az üzenetek tehát JSON formátumban utaznak a kliens és a szerver között. Minden üzenet az alábbi séma szerint épül fel:

```

{
  "Type": "MESSAGE_TYPE",
  "Value": {}
}
  
```

Tehát minden egyes üzenet tartalmazza a saját típusát, illetve magát az üzenet tartalmát (értékét). A Value érték mező szintén egy JSON objektumot tárol, ennek felépítése azonban az adott üzenettípustól függ. A bejövő üzenetek feldolgozásának első lépése mind szerver, mind kliens oldalon tehát az, hogy megvizsgáljuk, milyen típusú üzenetet kaptunk, hiszen minden üzenet típus más és más feldolgozást kíván.

A lehetséges üzenet típusokat csoportosítottam irányuk szerint (*Kliens* $\rightarrow$ *szerver*, vagy *Szerver* $\rightarrow$ *kliens*), ezek táblázatos formában a 438.2.-es mellékletben tekinthetők meg.

Egy konkrét üzenetre példát az alábbiakban láthatunk. Ez egy *Szerver* $\rightarrow$ *kliens* irányú üzenet, amely tartalmazza a mérő számítógépen elérhető mérési adatgyűjtő kártyákat:

```
{
  "Type": "DAQ_CARDS",
  "Value": [
    {
      "Code": "Dev1",
      "Name": "PCI-6251"
    },
    {
      "Code": "Dev2",
      "Name": "PCIE-6341"
    }
  ]
}
```

5.1.7. Mérési adatok átvitele

A mérési adatok valós idejű átvitelére értelemszerűen akkor van szükség, amikor a mérés fut, és a felhasználó látni szeretné a mérés aktuális állapotát. A mérési paramétereket tíz hálózati periódusonként átlagoljuk, tíz periódusra számítjuk ki a minőségi paramétereket, ezért minden egyes tíz periódushoz tartozó adatok egy egységet képeznek.

Mivel a villamos hálózatok alapfrekvenciája 50 Hz nálunk, ezért egy másodpercben 5 db ilyen tízperiódusos egységet dolgozunk fel, ezek átvitelére van szükség.

Az egy tízperiódusos egységhez tartozó adatokat típusuk szerint két csoportra bonthatjuk:

a) Számított paraméterek

- Például RMS, THD és a különböző teljesítményparaméterek.
- Ezek jellemzően kis mennyiségben keletkeznek, számosságuk a konfigurált csatornák számától függ (ez maximum 15 lehet a műszer felépítéséből adódóan).
- Valós számként számítjuk és tároljuk őket a szerveren (**double** típusként pontosan).
- A kliens számára azonban nem szükséges a **double** adattípus nyújtotta kétszeres pontosság, ezért a számított paraméterek átvitele **float** típusban, egyszeres pontossággal történik, ezáltal feleannyi adatot kell csak átvinnünk (a **float** típus 32 bitet, míg a **double** típus 64 bitet használ fel a számok tárolására).
- Egy tízperiódusos egység számított paraméterei kb. 300 bájtot tesznek ki.

b) Jelalakok

- A mérőkártya által közvetlenül mért értékek. Számosságuk szintén függ a csatornaszámtól, ill. a választott mintavételezési frekvenciától is. Például legyen a csatornaszám 15, a mintavételi frekvencia pedig 20 kHz. Ekkor egy tízperiódusban $15 * 20000 / 5 = 60000$ értéket kapunk.

- 16 bites mérőkártyát használunk, ami azt jelenti, hogy a mérőkártyáról leolvasott értékek 16 bitesek, ezért ezek tárolása célszerűen **short** típusban történik.
- Ahhoz azonban, hogy megkapjuk a valós, mért értékeket, a fent említett short értékekre előbb alkalmaznunk kell a mérőkártyára (pontosabban annak egyes csatornáira) jellemző polinomot, illetve meg kell szoroznunk az így kapott értékeket a mérés konfigurálásakor beállított szorzókkal, így végül ezekből is valós értékeket kapunk.
- A hatékonyság jegyében a hálózati átvitelnél a **short** értékek átvitele mellett döntöttem, hiszen így negyedannyi adatot kell átvinnünk a kliensekre (a **short** típus 16 bitet, míg a **double** típus 64 bitet használ fel a számok tárolására). A valós értékek megjelenítéshez azonban így az előző pontban említett műveleteket a kliens oldalon is el kell végezni, azonban ez nem jár túlságosan nagy erőforrásigénnyel, ezért ez bőven megéri.
- Mivel a jelalakok nagy adatmennyiséget képviselnek, ezért ezek átvitele alap esetben nem történik meg. A felhasználó kiválaszthatja, hogy mely csatornák jelalakjait szeretné látni, és csak ekkor történik meg a jelalakok átvitele, természetesen csak a kiválasztott csatornák adatait vesszük át.

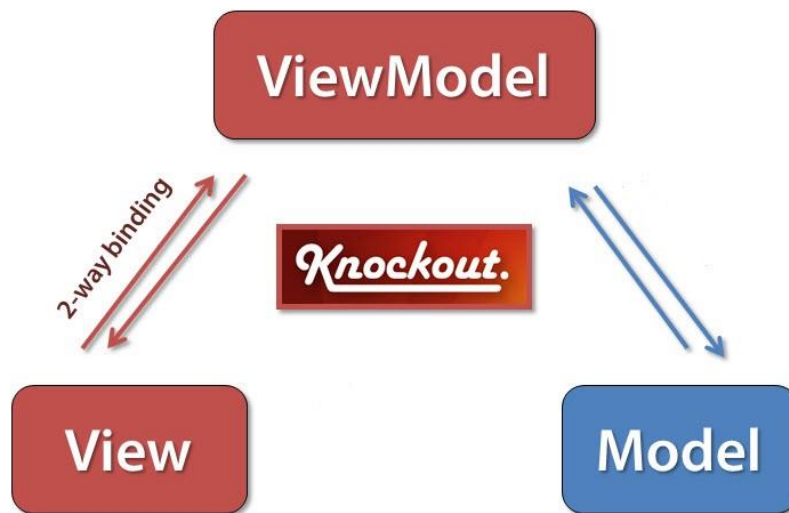
A mérési adatok átvitele bináris formában történik, mivel ezek eleve ilyen formában állnak rendelkezésre, ezért így a leghatékonyabb. Mivel a WebSocket technológia támogatja bináris adatok átvitelét is, ezért ez nagyon egyszerűen megvalósítható.

A felhasználó beállíthatja továbbá, hogy a szerver milyen gyakorisággal küldje a mérési adatokat. Ezért ha nincs szükség a folyamatos megfigyelésre, kiválaszthatjuk, hogy a rendszer ne küldje el minden egyes tízperiódus adatait, hanem csak minden 3., 5., 10., vagy 15. tízperiódusét.

5.2. Kliens oldal

5.2.1. Knockout

A Knockout [15] egy Model-View-ViewModel mintára épülő JavaScript keretrendszer. Általánosságban az MVVM rendszerek alap gondolata, hogy világosan külön kell választani az alkalmazásban az üzleti logikát és a megjelenítést.



25. ábra: A Knockout MVVM architektúrája
(A kép forrása: <http://adampurkiss.com/knockout-remoting/>)

A Model feladata az üzleti logikával való interakció fenntartása. Itt kérhetünk ki adatokat az adatelérési rétegből, kommunikálhatunk szolgáltatásokkal. A View kizárólag az adott felhasználói felületet tartalmazza, mögöttes kódjában nem találhatók objektumok az üzleti logikából. A sorból már csak a ViewModel maradt ki, melynek feladata a kettő összekapcsolása. Az MVVM minta nagy erőssége a binding, magyarul az adatkötés. Ennek segítségével nagyon egyszerű a ViewModel és a különböző megjelenítő komponensek összekapcsolása.

5.2.2. LESS

A LESS CSS [16] egy ingyenesen használható dinamikus stíluslap nyelv, gyakorlatilag a szabványos CSS kibővítése, jelentős tovább fejlesztése. A LESS lényege (azontúl amire neve is utal), hogy a CSS kódunkat dinamikusabbá tegyük, bizonyos részeket újra feltudjunk használni, ezáltal kisebb és dinamikusabb legyen a kód. Ezért tehát nem szabványos CSS kódot kell írunk, hanem a LESS által kiegészített szintaktikát kell használni. Ennek eredménye általában egy **.less** kiterjesztésű fájl, amit le kell fordítanunk egy LESS fordítóval hagyományos CSS kóddá. A fordítás történhet a szerver oldalon, így a kliens már csak a létrehozott **.css** fájlt kapja meg, de történhet teljesen kliens oldalon is, ekkor a kliens a **.less** fájlt kapja meg, és abból generálja JavaScript segítségével a CSS kódot. Mindkét megoldásnak vannak előnyei és hátrányai, kinek melyik megoldás a szimpatikusabb, illetve használhatóbb.



A LESS-ben rejlő lehetőségekről röviden:

- Változók használata (variables)

- Osztályok tulajdonságainak egymásba ágyazása, kevert használata (mixins)
- Szabályok egymásba ágyazása (nested rules)
- Függvények és operátorok használata (functions & operations)

Mindezek segítségével a CSS fejlesztés sokkal egyszerűbbé és rugalmasabbá válik a fejlesztő, designer számára.

5.2.3. Toastr

A különböző információs üzenetek megjelenítésére a toastr [17] nevű kis JavaScript függvénykönyvtárat használtam. Ez egy nagyon egyszerű, de nagyszerű kis modul, segítségével előugró értesítéseket, információs üzeneteket jeleníthetünk meg a felhasználó számára, látványos és informatív módon.

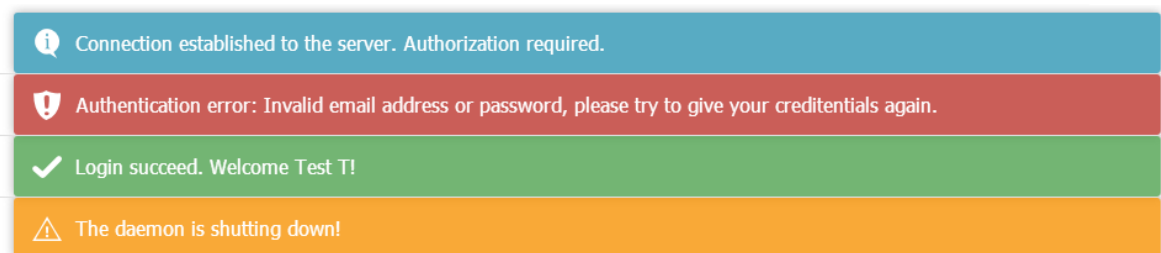
Használatához mindössze le kell töltenünk egy `.css` és `.js` fájlt, hivatkoznunk kell ezekre a HTML fájlban, és csak az alábbi négy parancsot kell megjegyeznünk.

```
// Információs üzenet megjelenítése
toastr.info("Connection established to the server. Authorization required.")

// Hibaüzenet megjelenítése
toastr.error("Invalid email address or password, please try to give your
credentials again.")

// „Siker” üzenet megjelenítése
toastr.success("Login succeed. Welcome Test T!")

// Figyelmeztető üzenet megjelenítése
toastr.warning("The daemon is shutting down!")
```



26. ábra: Toastr segítségével megjelenített látványos üzenetek

6. AZ ÚJ MÉRŐPROGRAM BEMUTATÁSA

6.1. A rendszer legfontosabb funkciói röviden

Az új mérőprogram megvalósításakor biztosítani kellett mindazon szolgáltatásokat, melyeket a 3.2.2 fejezetben bemutatott korábbi mérőprogram, ezért a legfontosabb szolgáltatásokat csak az alábbi rövid felsorolásban említem meg, az új mérőprogram újdonságait részletesebben a fejezet további részében mutatom be.

A legfontosabb funkciók:

- Mérési paraméterek beállítása
 - Saját műszer esetén fiókok felismerése
 - Csatornák beállítása
 - Mintavételezési frekvencia beállítása
 - Szorzók beállítása
 - Teljesítmény párosítások beállítása
- Mérés vezérlése
 - A hálózati frekvencia meghatározása
 - Szoftveres szinkronizáció a hálózati frekvenciához
- Villamos hálózat minőségi paramétereinek online kiszámítása a mért jelekből:
RMS, THD, hálózati frekvencia, aszimmetria, teljesítmények (S , P , Q), teljesítménytényező (PF)
- A mért és számított adatok lementése
 - A mérési konfigurációt (paraméterfájl – felépítés a 8.3 mellékletben)
 - A mért és számított adatokat (adatfájlok – felépítése a 8.4 mellékletben)
- A mért és számított értékek online kirajzolása
 - Jelalakok
 - FFT ábra
 - A fent említett paraméterek
- Értesítések küldése bizonyos események bekövetkezésekor (email formában)
 - Hálózati tranziensek esetén
 - Program hiba esetén
 - Előre megadott időközönként
- Robosztus működés
 - Kritikus hibák esetén a lehető legrövidebb idő alatt a normál működés helyreállítása

6.2. Daemon

A *daemon* egy konzolos alkalmazás (lásd 27. ábra), C# nyelven, a Visual Studio fejlesztőkörnyezetben készül. Ez a program fut a mérőszámítógépen, feladata a mérés vezérlése, adatgyűjtés, és szerverként kommunikációs felület biztosítása a klienseknek. A program grafikus felülettel egyáltalán nem rendelkezik, a fejlesztést megkönnyítendő jelenleg egy konzolablakba ír üzeneteket, de a későbbiekben elképzelhető, hogy éles környezetben csak háttérben, szolgáltatásként (service-ként) fog futni. A program működése során naplóállományt készít, melybe minden fontos eseményt feljegyez.

```

file:///C:/Documents and Settings/Administrator/My Documents/measurement/src/daemon/Daemon/bin/Debug/Measure...
2013-09-08 20:42-09 INFO Program MrDaemon program started.
2013-09-08 20:42-10 INFO Program The configuration file was successfully processed.
2013-09-08 20:42-10 DEBUG ControlServer ControlServer Thread ID: 2704
2013-09-08 20:42-10 INFO Fleck Server started at ws://127.0.0.1
2013-09-08 20:42-10 INFO JobSettings JobSettings file loaded.
2013-09-08 20:42-10 INFO Controller We found a previously running job, trying to get the parameter
file.
2013-09-08 20:42-10 INFO Controller Looking into the directory: G:\temp\MrDM_temp\Test_measurement
2013-09-08 20:42-10 INFO Controller ParameterFile deserialization successfull, starting the measur
ement.
2013-09-08 20:42-10 INFO JobContext New job is starting.
2013-09-08 20:42-10 DEBUG DataFileSaver DataFileSaver Thread ID: 2552
2013-09-08 20:42-10 DEBUG Acquisitor Acquisitors Thread ID: 3104
2013-09-08 20:42-10 DEBUG DataProcessor DataProcessor Thread ID: 2944
2013-09-08 20:42-10 DEBUG Acquisitor DAQ input buffer size: 100000
2013-09-08 20:42-10 INFO DataProcessor New datafile started: 2013-09-08_20-42-10
2013-09-08 20:42-10 INFO Acquisitor Data acquisition started.
2013-09-08 20:42-10 DEBUG DataGenerator Data simulator Thread ID: 3504

```

27. ábra: A daemon alkalmazás képernyője

6.2.1. Konfiguráció

A program alapvető beállításait egy konfigurációs fájl tartalmazza, ezt a program mappájában találhatjuk meg, neve: *daemon.ini*. Többek között az alábbi beállításokat tehetjük meg itt:

- hálózati beállítások (port, max. egyidejűleg csatlakozott kliensek száma),
- email beállítások: email címek és az SMTP szerver elérésének konfigurálása,
- riasztási beállítások testreszabása,
- alapvető mérési beállítások,
- mentési hely beállítása.

6.3. Monitor

Ebben a fejezetben röviden bemutatom a Monitor alkalmazás működését, a jobb szemléltetés érdekében több képernyőképpel is. Az alkalmazást a korszerű HTML5 és JavaScript technológiák segítségével készítettem el.

6.3.1. Kezdőképernyő (Home)

Az alkalmazás megnyitásakor a kezdőképernyő tárul elénk (28. ábra). A rendszer eléréshez először is be kell jelentkezünk az alkalmazásba, ezt itt, a kezdőképernyőn tehetjük meg, ehhez a rendszerben tárolt email címünket és jelszavunkat kell megadnunk.



28. ábra: Kezdőképernyő (bejelentkezés előtt)

Az ablak jobb felső sarkában kialakításra került egy ún. információs panel, amely mindig mutatja, hogy csatlakozva vagyunk-e a mérő számítógéphez, illetve hogy fut-e a mérés (29. ábra).



29. ábra: Kezdőképernyő (bejelentkezés után)

6.3.2. Konfigurációs képernyő (Config)

Miután bejelentkeztünk a szerverre, áttérhetünk a program konfigurációs felületére. Itt állíthatjuk be (illetve olvashatjuk le futó mérés esetén) a mérési paramétereket. Először is meg kell adnunk a legfontosabb mérési beállításokat:

- a mérés nevét (azonosítóját),
- a használni kívánt mérési adatgyűjtő kártyát,
- a mintavételezési frekvenciát,
- a frekvenciaszámítás időegységét.

A program kétféle beállítási módszert kínál: manuálisat és automatikus. A manuális inicializálás lehetősége a kompatibilitás megőrzése miatt volt fontos, hiszen előfordulhat, hogy a [3.2.](#) fejezetben bemutatott régebbi hardverek használatára kényszerülünk.

Az új mérődobozok használatakor csak ki kell választanunk az automatikus inicializálás opciót, és a szoftver automatikusan beolvassa az eszközök EEPROM memóriájában tárolt információkat: a fiók azosítóját, típusát, kalibrációs adatait (a felismert fiókokról készült képek meg is jelennek a felületen - 30. ábra). Ezáltal a mérés beállítása sokkal egyszerűbbé és gyorsabbá válik, ill. csökken a lehetőség a felhasználók figyelmetlenségéből adódó helytelen beállítások megadására.

The screenshot shows the 'CONFIG' tab of the 'MeasurementMonitor' interface. It includes settings for measurement name, initialization type (Automatic), DAQ card (PCI-6251), sample rate (20000 Hz), and frequency calculation period (0.4 sec). Below these are sections for 'Racks' (showing five rack units with terminals R, S, T) and 'Channels' (a table of 14 channels). To the right is a 'Powers' section with dropdowns for voltage and current channels. At the bottom are 'Start measurement' and 'Stop measurement' buttons.

	Signal	Name	Rack m.	Channel m.	Full m.
0.	U	Rack 0 R	10	0	0
1.	I	Rack 1 R	1	0	0
2.	I	Rack 2 R	100	0	0
3.	I	Rack 3 R	1	0	0
4.	I	Rack 4 R	1	0	0
5.	U	Rack 0 S	10	0	0
6.	I	Rack 1 S	1	0	0
7.	I	Rack 2 S	100	0	0
8.	I	Rack 3 S	1	0	0
9.	I	Rack 4 S	1	0	0
10.	U	Rack 0 T	10	0	0
11.	I	Rack 1 T	1	0	0
12.	I	Rack 2 T	100	0	0
13.	I	Rack 3 T	1	0	0
14.	I	Rack 4 T	1	0	0

	Voltage CH	Current CH
0.	Rack 0 R	Rack 2 R
1.	Rack 0 S	Rack 2 S
2.	Rack 0 T	Rack 2 T
3.	Rack 0 R	Rack 3 R
4.	Rack 0 S	Rack 3 S
5.	Rack 0 T	Rack 3 T
6.	-	-
7.	-	-
8.	-	-
9.	-	-
10.	-	-
11.	-	-

30. ábra: Mérési paraméterek konfigurálása – automatikus inicializálás

Manuális inicializálás esetén kézzel kiválasztanunk a használni kívánt mérőeszközök típusát, méréshatárát (31. ábra).

Lehetőségünk van egyedileg elnevezni az egyes fiókokat, ill. azok csatornáit (egy fiók segítségével egy háromfázisú jelet mérhetünk, ezért egy fiókhoz három csatorna tartozik). Például: 0. fiók: „Betáp feszültség”, 1. fiók: „C leágazás árama”, 3. fiók: „E leágazás árama”. Ezen elnevezések eltávolítása szintén csökkenti a mérés beállításakor szükséges kézi adminisztrációt.

Minden fiókhoz konfigurálhatunk egy közös szorzót (ez a fiókhoz tartozó összes csatornára vonatkozik), valamint meg kell adnunk azt is, hogy mely fiókok legyenek aktívak, hiszen gyakran előfordul, hogy az adott mérési feladathoz nincs szükség mind az öt fiókra, hanem csak kevesebbre.

Az egyes feszültség és áram fiókok csatornáit tetszőlegesen párosíthatjuk, és az így konfigurált „párookra” számol teljesítményértékeket a program. Az egyszerű használhatóság érdekében a szoftver megpróbálja kitalálni a megfelelő párosításokat, azonban van lehetőség ennek felülbírálására és a teljesen manuális megadásra is.

A képernyő alján található továbbá két fontos gomb: a mérés elindítására ill. leállítására szolgáló gombok. A mérés elindítása előtt természetesen ellenőrzi, validálja a program a megadott beállításokat, például nem indítható el a mérés, ha nem állítottunk be egy fiókot sem.

MeasurementMonitor

HOME CONFIG LIVE

Measurement name: Test measurement

Initialization type: Manual

DAQ card: Dev1: PCI-6251

Sample rate: 20000 Hz

Freq. calc. period: 0.4 sec

Connected ☒ Measurement not running ☒

Racks

Rack type:	Rack type:	Rack type:	Rack type:	Rack type:
HK-03V	Choose...	HK-07A	HK-06A	Choose...
Range: 500V			Range: 16A	
Name: Rack 1	Rack 1	Rack 2	Rack 3	Rack 4
Multiplier: 10	1	100	1	1
On/Off: ☒	☐	☒	☒	☐

Channels

Signal	Name	Rack m.	Channel m.	Full m.
0.	U Rack 0 R	10	497.5	4975
1.	I Rack 1 R			
2.	I Rack 2 R	100	2.0746888	207.46888
3.	I Rack 3 R	1	16	16
4.	I Rack 4 R			
5.	U Rack 0 S	10	497.5	4975
6.	I Rack 1 S			
7.	I Rack 2 S	100	2.0746888	207.46888
8.	I Rack 3 S	1	16	16
9.	I Rack 4 S			
10.	U Rack 0 T	10	497.2	4972
11.	I Rack 1 T			
12.	I Rack 2 T	100	2.0746888	207.46888
13.	I Rack 3 T	1	16	16
14.	I Rack 4 T			

Powers

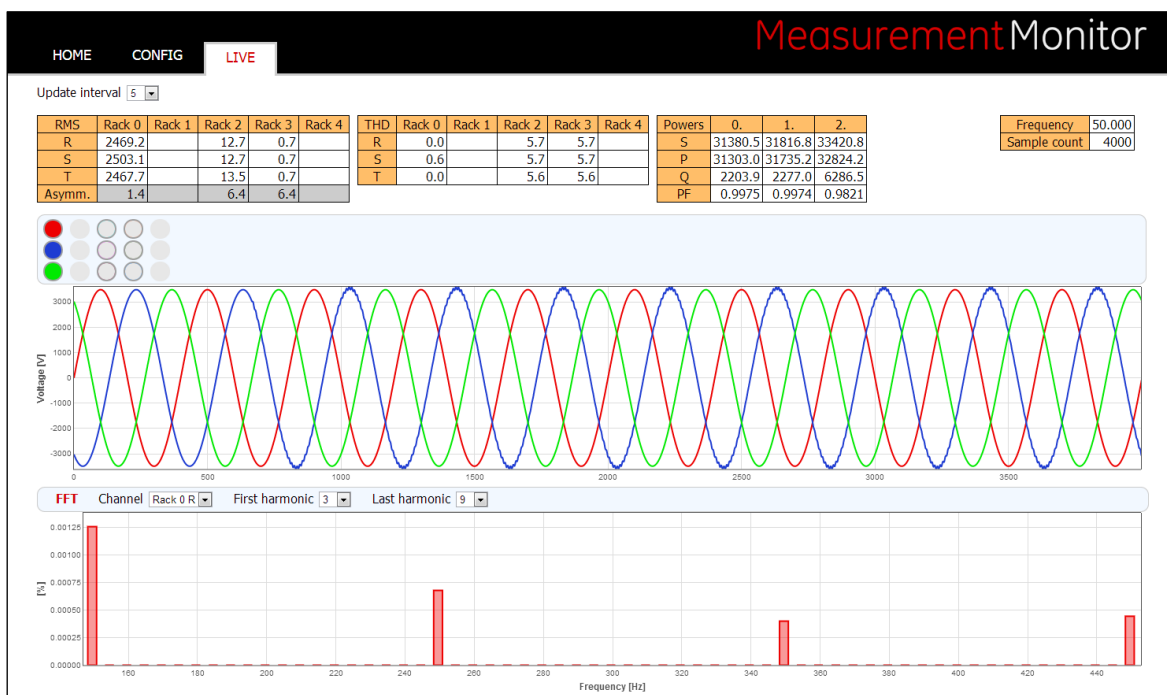
	Voltage CH	Current CH
0.	Rack 0 R	Rack 2 R
1.	Rack 0 S	Rack 2 S
2.	Rack 0 T	Rack 2 T
3.	Rack 0 R	Rack 3 R
4.	Rack 0 S	Rack 3 S
5.	Rack 0 T	Rack 3 T
6.	-	-
7.	-	-
8.	-	-
9.	-	-
10.	-	-
11.	-	-

Start measurement Stop measurement

31. ábra: Mérési paraméterek konfigurálása – manuális inicializálás

6.3.3. Aktuális mérési adatok képernyő (Live)

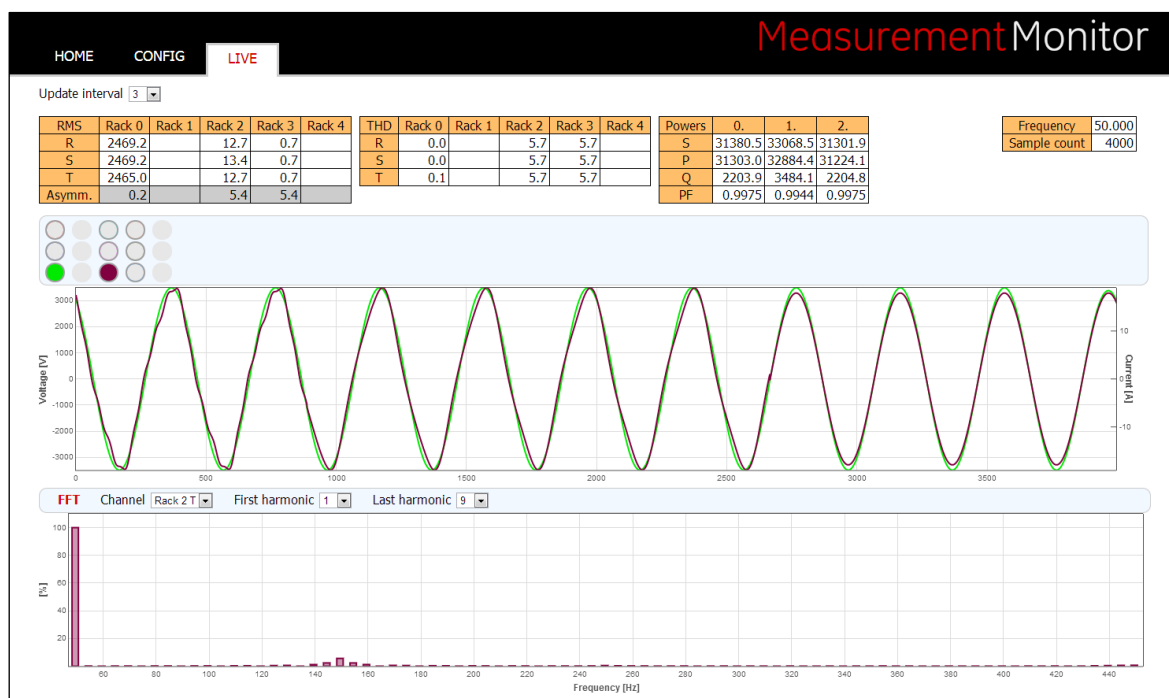
Ezen a képernyőn az aktuális mérési adatokat tekinthetjük meg. Az „Update interval” opcióval kiválaszthatjuk, hogy milyen gyakran frissítse a szoftver az értékeket.



32. ábra: Az aktuális mérési adatok megjelenítése (három fázis feszültsége)

Megtekinthetjük az előzőekben már említett villamos energia minőségi paramétereket, mutatókat táblázatos formában. Lehetőségünk van továbbá a mért jelalakok vizsgálatára is, itt a kis kör alakú vezérlőkkel (a fiókok mintájára) kapcsolhatjuk be és ki az egyes csatornák megjelenítését.

Ezen felül ábrázolhatjuk az egy kiválasztott csatornához tartozó FFT analízis eredményét is, itt megadható az is, hogy hányadik harmonikustól hányadikig szeretnénk megjeleníteni az értékeket.



33. ábra: Mérési adatok megjelenítése (egy fázishoz tartozó feszültség és áramjel)

7. IRODALOMJEGYZÉK

- [1] MSZ EN 50160 szabvány: A közcélú elosztóhálózatokon szolgáltatott villamos energia feszültségjellemzői
- [2] Bátorfi Richárd, Hegedűs János, Unhauzer Attila, and Váradiné dr. Szarka Angéla, Méréstechnika, Miskolci Egyetem, Miskolc, 2007.
- [3] Nagy Balázs, Sarvajcz Kornél, and Unhauzer Edina, Intelligens villamos hálózat analizáló műszer tervezése, TDK dolgozat, Miskolc, 2010.
- [4] Bátorfi Richárd, A szolgáltatott villamos energia minőségének és a felhasználás hatékonyságának on-line vizsgálata, PhD értekezés, Miskolc, 2013.
- [5] Bencs Róbert, Mérési adatok automatizált feldolgozó- és dokumentum generáló szoftverrendszere villamos hálózat diagnosztikai alkalmazáshoz, TDK dolgozat, Miskolc, 2012.
- [6] A DEOEC fázisjavító berendezés telepítését megelőző villamos hálózat energiahatékonysági és minőségi vizsgálata, Szakértői jelentés, Uni-Energy Egyetemi Kutató és Fejlesztő Közhasznú Nonprofit Kft., 2012.
- [7] Bencs Róbert, Villamos hálózat diagnosztikai rendszer távoli adminisztrációjának kidolgozása, Műszaki Tudomány az Észak-kelet Magyarországi Régióban 2013 konferencia, Debrecen, 2013.
- [8] [http://www.simcom.us/act_admin/supportfile/SIM900_HD_V1.01\(091226\).pdf](http://www.simcom.us/act_admin/supportfile/SIM900_HD_V1.01(091226).pdf)
- [9] Zsják Soma, Hálózat analizáló rendszer távoli felügyelete, Szakdolgozat, Miskolc, 2011.
- [10] http://en.wikipedia.org/wiki/Internet_protocol_suite
- [11] <http://tools.ietf.org/html/rfc6455>
- [12] <http://en.wikipedia.org/wiki/WebSocket>
- [13] <http://nginx.org>
- [14] <http://news.netcraft.com/archives/2013/10/02/october-2013-web-server-survey.html>
- [15] <http://knockoutjs.com>
- [16] <http://lesscss.org>
- [17] <https://github.com/CodeSeven/toastr>

8. MELLÉKLETEK

8.1. Részlet a feldolgozóprogram által generált jegyzőkönyvből

Uni-Energy Közhasznú Nonprofit Kft.

I. melléklet - A transzformátor

I. melléklet - A transzformátor

I.1. 2012.07.17. 03:10:10 - 2012.07.17. 03:40:10

I.1.1. Feszültség effektív értékének elemzése

	Min	P5	P95	Max	Átlag
A transzformátor szekunder oldali feszültség R fázis [V]	240.63	241.41	241.99	242.16	241.73
A transzformátor szekunder oldali feszültség S fázis [V]	241.24	242.27	242.96	243.19	242.67
A transzformátor szekunder oldali feszültség T fázis [V]	239.76	240.48	241.24	241.44	240.92



I.1. ábra: A transzformátor szekunder oldali feszültség R fázis effektív értékének idődiagramja



I.2. ábra: A transzformátor szekunder oldali feszültség R fázis effektív értékének statisztikai eloszlása



I.3. ábra: A transzformátor szekunder oldali feszültség S fázis effektív értékének idődiagramja



I.4. ábra: A transzformátor szekunder oldali feszültség S fázis effektív értékének statisztikai eloszlása



I.5. ábra: A transzformátor szekunder oldali feszültség T fázis effektív értékének idődiagramja

8.2. A kidolgozott kommunikációs protokoll üzenetei

8.2.1. Kliens → szerver irány

Üzenet típusa	Leírás
AUTHORIZE_USER	Bejelentkezéskor küldendő, tartalmazza az alkalmazásba belépni kívánó felhasználó email címét és jelszavát.
GET_CURRENT_JOB	Lekérdezi, hogy fut-e éppen mérési feladat, és ha fut, annak részleteit.
START_JOB	Elindítja a paraméterben mérési feladatot.
STOP_CURRENT_JOB	Leállítja az éppen futó mérési feladatot.
GET_DAQ_CARDS	Lekérdezi az elérhető mérőkártyákat.
GET_RACK_INFO	Lekérdezi a műszerbe helyezett mérődobozokat automatikus inicializálás esetén.
START_LIVE_TRANSFER	Elindítja a mérési adatok valós idejű küldését.
STOP_LIVE_TRANSFER	Leállítja a mérési adatok küldését.
SET_LIVE_UPDATE_FREQUENCY	Beállítja, hogy milyen gyakorisággal küldje a szerver az aktuális mérési adatokat.
SET_LIVE_RAW_CHANNELS	Beállítja, hogy mely csatornákhöz tartozó jelalakokat küldje a szerver.
SET_LIVE_FFT_OPTIONS	Beállítja a valós idejű FFT adatok küldésének paramétereit (csatorna, hányadik harmonikustól hányadikig).

8.2.2. Szerver → kliens irány

Üzenet típusa	Leírás
AUTH_SUCCESS	A felhasználó sikeresen bejelentkezett az alkalmazásba.
AUTH_ERROR	Nem sikerült a bejelentkezés (érvénytelen üzenet, vagy hibás email-cím/jelszó).
CURRENT_JOB	Státusz információ a mérés állapotáról. Fut-e mérés, ha fut, milyen paraméterekkel.
DAEMON_EXIT	Információs üzenet, akkor küldjük amikor a daemon leáll.
DAQ_CARDS	Az elérhető mérőkártyák.
RACK_INFO	Automatikus inicializáláshoz a felismert fiókok adatait tartalmazza.
INFO	Információs üzenet a szervertől.
ERROR	Hibaüzenet a szervertől.

8.3. A paraméterfájl felépítése

```
{
  "MeasurementName": "Test measurement",
  "SampleRate": 20000,
  "FrequencyCalculationPeriod": 1,
  "Racks": [
    {
      "ID": 0,
      "PhysicalID": 0,
      "Type": "HK-05V",
      "Name": "Rack 0",
      "Description": null,
      "IsActive": true,
      "IsVoltage": true,
      "Multiplier": 1,
      "Channels": [
        {
          "ID": 0,
          "PhysicalID": 0,
          "Name": "Rack 0 R",
          "Description": null,
          "Multiplier": 200.2,
          "ChannelPolynom": [0, 0.00015259254737998596]
        },
        {
          "ID": 2,
          "PhysicalID": 5,
          "Name": "Rack 0 S",
          "Description": null,
          "Multiplier": 199.95,
          "ChannelPolynom": [0, 0.00015259254737998596]
        },
        {
          "ID": 4,
          "PhysicalID": 10,
          "Name": "Rack 0 T",
          "Description": null,
          "Multiplier": 200.32,
          "ChannelPolynom": [0, 0.00015259254737998596]
        }
      ]
    },
    {
      "ID": 1,
      "PhysicalID": 1,
      "Type": "HK-06A",
      "Name": "Rack 1",
      "Description": null,
      "IsActive": true,
      "IsVoltage": false,
      "Multiplier": 1,
      "Channels": [
        {
          "ID": 1,
          "PhysicalID": 1,
          "Name": "Rack 1 R",
          "Description": null,
          "Multiplier": 16,
          "ChannelPolynom": [0, 0.00015259254737998596]
        },
        {
          "ID": 3,
          "PhysicalID": 6,
          "Name": "Rack 1 S",
          "Description": null,
          "Multiplier": 16,

```

```

        "ChannelPolynom": [0, 0.00015259254737998596]
    },
    {
        "ID": 5,
        "PhysicalID": 11,
        "Name": "Rack 1 T",
        "Description": null,
        "Multiplier": 16,
        "ChannelPolynom": [0, 0.00015259254737998596]
    }
]
},
"Powers": [
    {
        "ID": 0,
        "Name": "",
        "VoltageChannelID": 0,
        "CurrentChannelID": 1
    },
    {
        "ID": 1,
        "Name": "",
        "VoltageChannelID": 2,
        "CurrentChannelID": 3
    },
    {
        "ID": 2,
        "Name": "",
        "VoltageChannelID": 4,
        "CurrentChannelID": 5
    }
]
}

```

8.4. Az adatfájlok bináris formátuma

8 byte (long)	• Időpont 10 perces kezdet
4 byte (int)	• 10 periódusok száma a 10 perc alatt (tiz_periodusok_szama)
4 byte (int)	• Összes mintaszám a 10 perc alatt csatornánként (osszmintaszam)
CS * MSZ * 2 byte (short)	• Minták (csatornaszám * osszmintaszam)
PSZ * 4 byte (int)	• Minták száma a 10 periódusokban
PSZ * 8 byte (double)	• Hálózati Frekvencia értékek(tiz_periodusok_szama)
CS * PSZ * 1 byte (char)	• Flag hiba jelzés (csatornaszám * tiz_periodusok_szama)
CS * PSZ * 8 byte (double)	• RMS értékek (csatornaszám * tiz_periodusok_szama)
CS * PSZ * 8 byte (double)	• THD értékek (csatornaszám * tiz_periodusok_szama)
5 * PSZ * 8 byte (double)	• Aszimmetria értékek (FIOKOK_SZAMA * tiz_periodusok_szama)
4 * TSZ * PSZ * 8 byte (double)	• Teljesítményértékek(S, P, Q, cosfi) (4 * teljes_szama * tiz_periodusok_szama)
10 * TSZ * PSZ * 8 byte (double)	• Teljesítményértékek (S1,P1,Q1,PF1,SN,PH,DU,DI,SH,DH) (10 * teljes_szama * tiz_per_szama)
40 * TSZ * PSZ * 8 byte (double)	• Harmonikus teljesítmények (40 * teljes_szama * tiz_periodusok_szama)
FESZ_CS * 60 * 8 byte (double)	• Flicker értékek (Pst)

Köszönetnyilvánítás

A kutatás az Európai Unió és Magyarország támogatásával a TÁMOP 4.2.4.A/2-11-1-2012-0001 azonosító számú „Nemzeti Kiválóság Program – Hazai hallgatói, illetve kutatói személyi támogatást biztosító rendszer kidolgozása és működtetése konvergencia program” című kiemelt projekt keretei között valósult meg.

