ÓBUDAI EGYETEM
ÓBUDA UNIVERSITY

KANDÓ KÁLMÁN VILLAMOSMÉRNÖKI KAR
ELEKTRONIKAI ÉS KOMMUNIKÁCIÓS RENDSZEREK INTÉZET
MŰSZERTECHNIKAI ÉS AUTOMATIZÁLÁSI TANSZÉK

SZAKDOLGOZAT

OE-KVK
2022.

Balla Domán
T008604/FI12904/K

ÓBUDAI EGYETEM
ÓBUDA UNIVERSITY

Óbudai Egyetem
Kandó Kálmán Villamosmérnöki Kar
Műszertechnikai és Automatizálási Intézet

SZAKDOLGOZAT FELADATLAP

Hallgató neve: Balla Domán
Szakdolgozat száma: SZD22030109188201
Törzskönyvi száma: T008604/FI12904/K **Neptun kódja:** [redacted]

Szak: villamosmérnöki (magyar nyelven)
Specializáció: Műszer-automatika specializáció

A dolgozat címe: Mérésadatgyűjtő rendszerek felügyelete
A dolgozat címe angolul: Surveillance of measurement data collection systems
A feladat részletezése: Mérésadatgyűjtő rendszerek felhasználásnakt, monitorozási lehetőségeinek és a mérési eredmények megjelenítésének bemutatása.

Intézményi konzulens neve: Sándor Tamás

A kiadott téma elévülési határideje: 2025. március 1.
Beadási határidő: 2022. 05. 15.

A szakdolgozat: Nem titkos.

Kiadva: Budapest, 2022. 03. 21.

P.H.

Óbudai Egyetem Kandó Kálmán Villamosmérnöki Kar Elektronikai és Kommunikációs Rendszerek Intézet

.....
Intézetigazgató

A dolgozatot beadásra alkalmasnak találom:

.....
belső konzulens

.....
külső konzulens

ÓBUDAI EGYETEM
ÓBUDA UNIVERSITY

KANDÓ KÁLMÁN VILLAMOSMÉRNÖKI KAR
ELEKTRONIKAI ÉS KOMMUNIKÁCIÓS RENDSZEREK INTÉZET
MŰSZERTECHNIKAI ÉS AUTOMATIZÁLÁSI TANSZÉK

HALLGATÓI NYILATKOZAT

Alulírott hallgató kijelentem, hogy a szakdolgozat saját munkám eredménye, a felhasznált szakirodalmat és eszközöket azonosíthatóan közöltem. Az elkészült szakdolgozatban található eredményeket az egyetem és a feladatot kiíró intézmény saját céljára térítés nélkül felhasználhatja, a titkosításra vonatkozó esetleges megkötések mellett.

Budapest, 2022. május 15.

.....
hallgató aláírása

Tartalom

ÖSSZEFOGLALÓ

A szakdolgozat célja egy mérésadatgyűjtő rendszer monitorozási lehetőségeinek feltárása. Ennek keretében a dolgozat egy áttekintés ad a megcélzott technológiákról és az alkalmazott informatikai eszköztárról beleértve a mérési adatok gyűjtéséért felelős játszó adatgyűjtő modult, az információ továbbításban és tárolásában szerepet játszó modulokat és programnyelveket. Egy mérésadatgyűjtő rendszer több, egymástól független mérőállomásból áll, melyek egy központi számítógépre továbbítják a kiértékeléshez szükséges adathalmazt. Dolgozatomban bemutatott programozás során kialakításra kerül egy korszerű, JSON szintaktikát alkalmazó megoldás, amely az információ továbbítását látja el. A leprogramozott megoldás által szállított mérési adatok szolgáltatják az alapot a mérési adatgyűjtés során kezelt a kezelt információk HTML alapú prezentálásához. Ez az adathalmaz magába foglalja a mérőállomás pontos azonosítóját, a mért adatokat és a mérést jellemző egyéb azonosítókat. A cél ennek az adathalmaznak feldolgozási lehetőségeinek vizsgálata, egy feldolgozásra alkalmas software kidolgozása, mely magába foglalja egy webes böngészőben történő grafikus megjelenítést is. Ennek célja az adatok átláthatóságának növelése a könnyebb kiértékelhetőség végett. Mindezekkel rá szeretnék mutatni arra, hogy a napjaink energetikai kihívásai, illetve az együttműködő energiarendszerek monitorozása tekintetében milyen egyszerű és legfőképp gazdaságos megoldásokkal is elősegíthetőek a részletes információ szolgáltatások. A digitális mérés, az ezen alapuló jeltovábbítás korlátlan lehetőségeiből egy vékony szeletet szándékoztam felvillantani.

SUMMARY

The aim of this thesis is to explore the monitoring capabilities of a measurement data collection system. In this context, the thesis gives an overview of the targeted technologies and the IT tools used, including the data acquisition module responsible for the collection of measurement data, the modules and programming languages involved in the transmission and storage of information. A measurement data acquisition system consists of a number of independent measuring stations which transmit to a central computer the data set required for evaluation. The programming presented in my thesis will be used to develop a state-of-the-art solution using JSON syntax to transfer the information. The measurement data delivered by the programmed solution will provide the basis for the HTML-based presentation of the processed information that is processed during the measurement data collection. This data set includes the exact identifier of the measurement station, the measured data and other identifiers that characterise the measurement. The aim is to investigate the processing possibilities of this data set and to develop a software for processing, including a graphical presentation in a web browser. The aim is to increase the transparency of the data for easier evaluation. In this way, I would like to show how simple and, above all, economical solutions can be used to facilitate detailed information services for today's energy challenges and monitoring of interoperable energy systems. I wanted to show a thin slice of the unlimited possibilities of digital metering and the signal transmission based on it.

1 BEVEZETÉS

A XXI. század teljesítényorientált világában szinte lehetetlen olyan tevékenységet vagy területet találni, ahol nem lehetséges valamely meghatározó tényező mérése, a mérés által a tevékenység kontrollja és irányítása. Az elektronikus megoldások teremtik meg a lehetőséget számtalan mérésre és ezen mérések eredményeinek gyors és pontos értelmezésére. A legtöbb esetben az elvárt mérési eredményeket több mérőműszer együttes használatával, a mérési eredményeiknek egyszerű, vagy bonyolult algoritmusokon keresztül végzett transzformációjával kaphatjuk meg.

A szakdolgozatom keretében a villamos fogyasztásmérő rendszerek bemutatásával, majd a mérőállomások által közölt jel feldolgozási lehetőségeinek tárgyalásával, végezetül pedig a feldolgozott jelek webes felületen történő grafikus megjelenítésével foglalkozok.

A fő feladat egy olyan program létrehozása Python nyelven, mely képes a fogyasztásmérő műszerek által közölt JSON formátumú adatcsomagot feldolgozni és azt HTML környezetben megjeleníteni, ezzel bemutatva egy, a rendszert felügyelő központi számítógép szerepét és fontosságát a mérésadatgyűjtő rendszerek területén.

A műszerek által közölt feldolgozandó jel minden lényeges információt tartalmaz. Kiolvasható a mérési eredmények kiértékeléséhez szükséges összes adat, így például a mérőállomás egyértelmű azonosítója, a mért eredmény, a mérés pontos ideje, a mérés sorszáma és minden mérési eredmény egyedi azonosítója. Ezek az adatok azonban a begyűjtött formában nehezen értelmezhetőek, kiértékelésük pedig rendkívül időigényes már egy mérőpont esetén is.

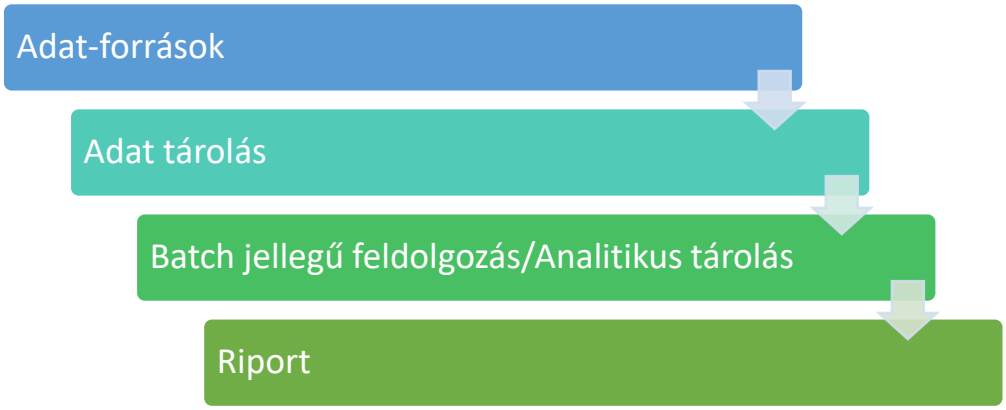
A célom ezeknek az adatoknak a beolvasása, feldolgozása és egy könnyen értelmezhető, átláthatósága miatt a kiértékelést nagy mértékben segítő webes felületen történő megjelenítése.

2 A MEGOLDANDÓ PROBLÉMA MEGFOGALMAZÁSA

Az energetikai fejlődés és korszerűsítés velejárója az energetikai adatok mérésével kapcsolatos követelmények emelkedése, illetve a mérési adatok mennyiségének exponenciális növekedése.

Ezen változás egyúttal lehetőséget teremt az energetikai viselkedésformák, szokások mélyebb elemzéséhez, ugyanakkor ez csakis akkor tud megvalósulni, ha a mérési adatok továbbítása, tárolása, feldolgozása terén is hathatós változások történnek mind minőségben, mind pedig mennyiségi mutatókban. Napjainkban ehhez minden szükséges technikai feltétel adott. Az IoT alapú mérési adat gyűjtő szenzorok, végponti eszközök rendkívüli minőségű és mennyiségű adatszolgáltatásra képesek, melyhez kapcsolódóan a szélessávú Internet, a 4G és az 5G-s mobilhálózatok megfelelő kommunikációs közeget biztosítanak.

Korszerű adattárolási megoldásokkal rendelkezünk, illetve nagy teljesítményű számítási kapacitások állnak rendelkezésre az ilyen módon előálló hatalmas mennyiségű információ strukturált kezelésére értékelésére és elemzésére.



1. ábra – Mérési adatfolyamat adattárolással

2. ábra – Mérési adatfolyamat real-time feldolgozással

A keletkező információk folyamatos vagy szekvenciális feldolgozása révén kerülhetnek előállításra az elemzések, riportok.

Dolgozatomban ezen információ áramlási folyamatra, ezen belül annak bizonyos szakaszaira szeretnék fókuszálni.

Megítélésem szerint egy jelentős probléma (vagy másképpen megoldandó feladatot jelentő dolog) az, hogy mérni ott nyílik lehetőségünk, ahol a mérendő dolog található, létezik vagy előfordul. Az adattárolás, feldolgozás elemzés pedig ettől távol található, így az információt – egy posti csomaghoz hasonlóan – meg kell címezni, fel kell adni, utaztatni szükséges, valamint át kell adni a címzettnek. Az adatfolyamat egészén belül az alábbi területeket érintem dolgozatomban:

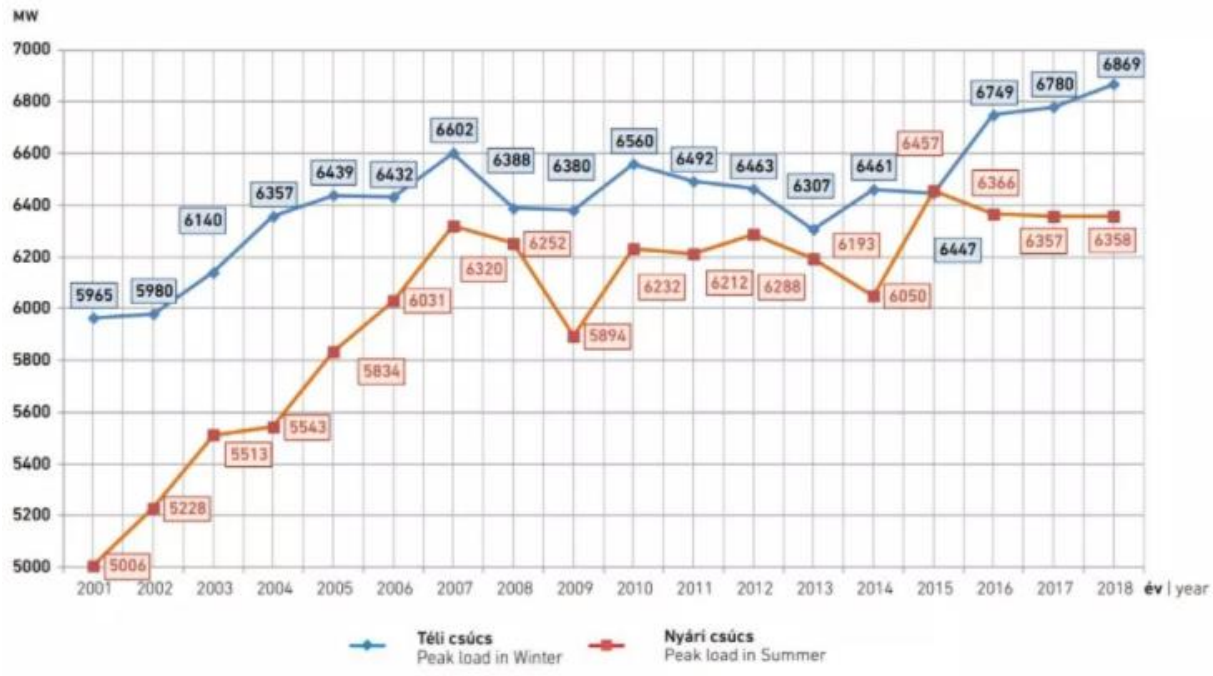
A. A végponti berendezésektől, a mérőegységektől érkező, a mérési adatokat tartalmazó információkból álló üzenetek kinyerése és struktúrába helyezése az egyik terület.
B. A másik részfolyamattal pedig az összegyűjtött információ, mint analitikus adatforrásból a megjelenítő közeg felé történő információ továbbítást érintem.

A jelenkor problémáját abban látom, hogy nagyon sok párhuzamos, egymással versengő megoldás működik a világban, nagy vendorok (pl.: Oracle) megoldásai uralják piac meghatározó szegmensét. Ezek mellett sok területen megjelentek a nyílt forráskódú megoldások, egyszerű és a hagyományokkal szakító adatkezelés, információ továbbítás megoldások. Dolgozatomban a JSON szintaxis, egy text alapú szabvány segítségével szeretném megoldani a nagy tömegű mérési adatgyűjtés említett lépéseit.

3 TECHNOLÓGIAI KOMPONENSEK ISMERTETÉSE

3.1 Villamos fogyasztásmérők

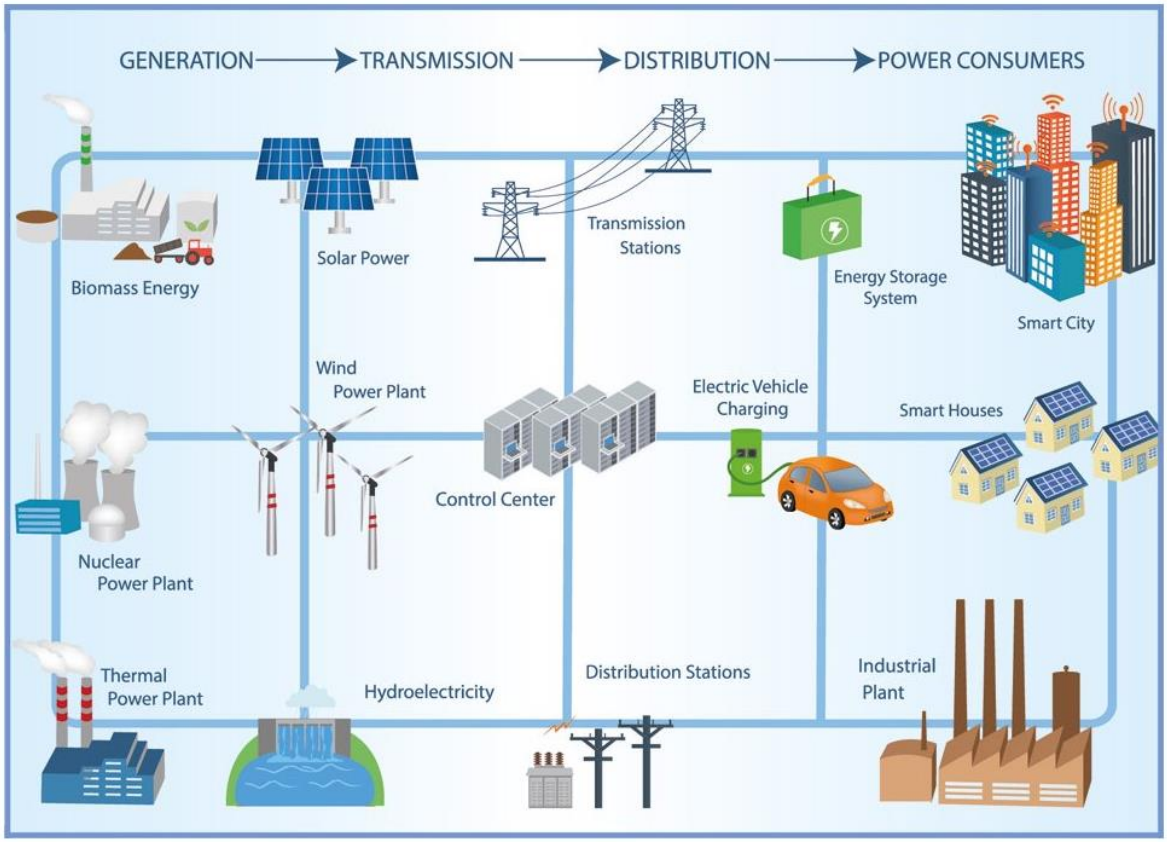
Az energiaköltségeket a termelőiparban hagyományosan anyagköltségnek tekintik. Az elmúlt évtizedekben a gyártás, termelés energiafelhasználása, az elektromos energia költségek drámaian megnőttek és többé már nem tekinthetők rezsiköltségnek, hanem inkább értékes erőforrásnak, amellyel stratégiailag kell gazdálkodni.



3. ábra – Magyarország elektromos energiafogyasztásának éves csúcsértékei az elmúlt években (1)

Ez azonban csak a villamosenergia-fogyasztási adatok pontos és folyamatos mérése, nyomon követése, gyűjtése és elemzése révén érhető el. Az ipar előnyben részesíti az elektromos energiát, mivel az könnyen átalakítható számos alacsonyabb energiaformává, például hővé, fénnyé, sűrített levegővé, mechanikus nyomatékká és számos más dologgá. Más energiaforrásokkal összehasonlítva az elektromos energia fogyasztása viszonylag könnyen és pontosan mérhető. Ennek eredményeképpen más energiaformákat jellemzően érzékelők és átalakítók alakítanak át elektromos jellé, amelyek aztán az elektromos jelek mérési technikáinak szabványos eljárásaival detektálhatók. A villamossági mérőműszerek a mérési alkalmazások széles körére szolgálnak. Egyeseket, például az

oszcilloszkópokat nagy pontosságú és valós idejű megfigyelésre fejlesztették ki, míg másokat, például a hálózatminőség-elemzésre szolgáló multimétereket úgy tervezték, hogy különféle feladatokra, például a hálózati forgalom elemzésére alkalmasak legyenek. Az egyes műszerek műszaki specifikációi eltérőek, és a felhasználónak tisztában kell lennie ezekkel a különbségekkel, mielőtt döntést hozna arról, hogy melyik műszert vásárolja meg. Az egyes elvégzendő feladatokra vonatkozó egyedi műszaki specifikációkat a mérési stratégia megfogalmazásával és az információs fok konkrét specifikációs tartományainak megadásával kell tisztázni. (2) A számítástechnika és távközlés integrálása az energiahálózatok berendezéseibe a hálózat rugalmasabb, hatékonyabb, „okosabb" működtetését teszi lehetővé. A helyi energiatermelés, az energiatárolás, a villamos járművek, az okos háztartási készülékek elterjedése, az igény a részletes információra a piac valamennyi szereplője számára megteremtette a lehetőséget a sokfunkciós, adatcserére képes mérő- és vezérlőeszközök kifejlesztésére, amelyek az okos hálózatok alapinformációit nyújtják.



4. ábra – Okos hálózatok aktív szereplői (3)

(4) A tárgyak internete (IoT) olyan technológia, amely egyre népszerűbbé válik a mai világban. Gyakran használják a csatlakoztatott eszközök vagy "dolgok" gyorsan bővülő hálózatára, amelyek képesek alacsony sávszélességű hálózaton keresztül adatot cserélni, és egyre elterjedtebbé válnak. Különböző iparágak, köztük az autóipar, a logisztika, az egészségügy, az intelligens hálózat és az intelligens városok használják a dolgok internetét. Ennek fényében érthető, hogy elengedhetetlen, egy megbízható és egységes adatcsere formátum. (5)

A hagyományos elektromos fogyasztásmérők jellemzően analóg mérőműszerek, ahol az alapműködésből fakadóan nem áll elő olyan jel, amely hatékony megoldást adhat nagy távolságokra történő jeltovábbítás megvalósításához.

Okos hálózatok elterjedéséhez, az okos mérés térhódításához elengedhetetlen a korszerű digitális mérés és jeltovábbítás, amelynek megoldást kell biztosítania a jelen kihívásaira. Ennek megfelelően egy villamosenergia vonatkozású okos megoldásnak biztosítania szükséges a kétirányú villamosenergia-forgalom mérését, figyelnie kell a feszültség és frekvencia minőségét, kezelnie szükséges a különböző tarifákat, lehetőséget kell biztosítania különböző távoli beavatkozásokra (pl.: korlátozásra vagy vezérlésre) és természetesen mindezeket lehetővé tevő adatcsere lehetőségét is biztosítania kell. E így önmagában is kimerítő felsorolás volt, ugyanakkor nem is említettünk olyan további lehetőségeket, mint például az otthon vezérléséhez, automatizáláshoz (home automation interface) helyi kapcsolat biztosítása, vagy például több szolgáltatás, vagy közmű kiszolgálása (gáz, víz).

3.2 IoT alapú mérés, jeltovábbítás

A hagyományos elektromos árammérők nem képesek információt szolgáltatni a mérési adatokról, így ezek nem illeszthetőek be egy okos hálózati környezetbe.

A mérési adatokról történő jeltovábbítás problémájára potenciális megoldást jelenthet a thaiföldi Nakhon Phanom Egyetem mérnöki kara által kidolgozott IoT villamos energiafelügyelő rendszer. Ebben a rendszerben q feszültség, áram, aktív teljesítmény és

az összesített energiafogyasztás mérése érdekében a rendszer energiafigyelő csomópontokból áll, amelyek a Peacefair PZEM-004T-t, egy olcsó energiamérőt használnak, amely egy nem invazív CT (áramváltó) érzékelőt, valamint az SD3004 energiamérő chipet és mikrokontrollert használja.

5. ábra – Peacefair PZEM-004T - Multifunkciós feszültség, áram, teljesítmény mérő modul

A mért adatok JSON (JavaScript Object Notation) formátumban, MQTT (Message Queuing Transport Protocol) protokollon keresztül kerülnek a szerverre. A Raspberry Pi 3 B modellje került kiválasztásra a projekt helyi kiszolgálói feladatainak ellátásához szükséges számítógépnek. Ennek eredményeképpen a felhasználók lokálisan, vagy távolról, az interneten keresztül elérhető webes alkalmazáson keresztül férhetnek hozzá az energiafogyasztásukra vonatkozó információkhoz. (5)

A fentiekben bemutatott eszköz előnye, hogy a mára már széleskörben elterjedt RS-232 porton keresztül képes kommunikációt folytatni. Ez alapján rendkívül könnyen illeszthető mobil, GSM alapú kommunikációs egységhez, vagy Wi-fi adapterhez, melyeken keresztül az adatcsere szinte korlátlanul megvalósítható.

3.3 A Python programozási nyelv

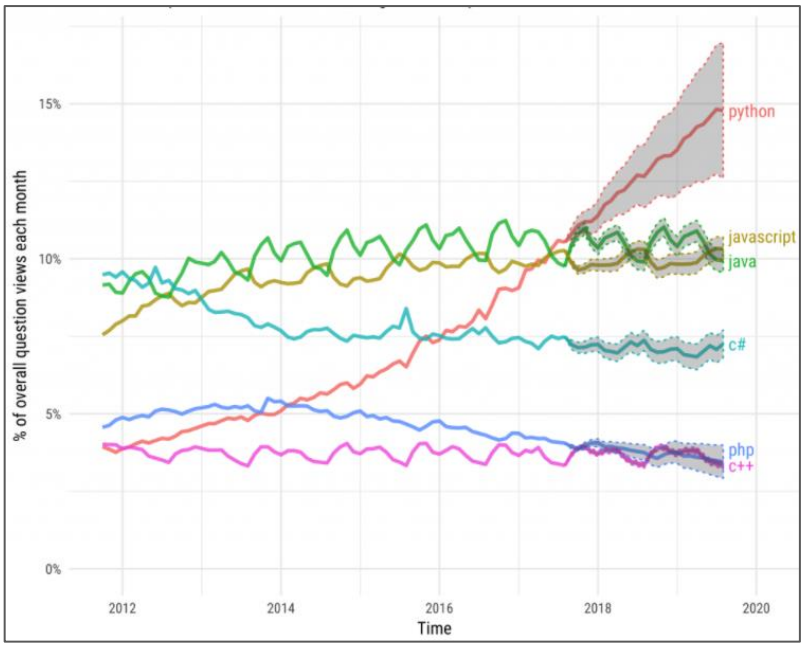
3.3.1 A Python bemutatása

A Python egy magas szintű, általános célú, értelmezett programozási nyelv. Az első verziója 1991-ben került kiadásra, melynek verziószáma a Python 0.9.0. volt.

Tervezési filozófiájában a jelentős behúzás alkalmazása a kód olvashatóságát hangsúlyozza. Nyelvi elemei és objektumorientált megközelítése arra irányul, hogy segítse a programozókat a világos, logikus kód megírásában, mind a kisebb, mind a nagyobb léptékű projektek esetében. (6)

Más programozási nyelvekkel ellentétben a Pythont úgy tervezték, hogy nagymértékben modulokkal bővíthető legyen, nem pedig úgy, hogy minden képességét a magjába építse. Kompakt és moduláris felépítése miatt különösen alkalmas arra, hogy meglévő rendszerekhez programozható interfészeket adjunk hozzá. (7)

Előnyei miatt rövid idő alatt nagy térhódításon ment keresztül, kedveltsége alapján a 4. legnépszerűbb programozási nyelv.

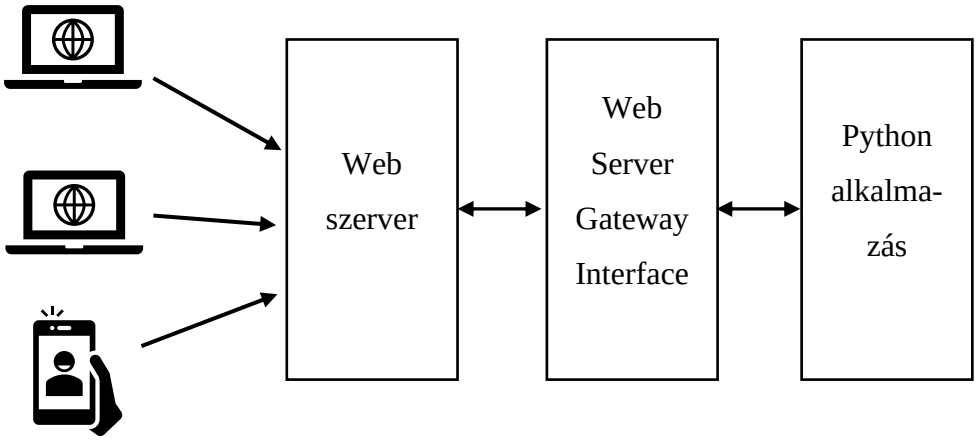


6. ábra – A főbb programozási nyelvek várható térhódítása 2018. (8)

A Pythont úgy tervezték, hogy felhasználóbarát programozási nyelv legyen. Formázása tiszta, és gyakran használ angol kulcsszavakat, ahol más nyelvek írásjeleket használnak. Sok más nyelvvel ellentétben nem használ kapcsos zárójeleket a blokkok elválasztására, és a pontosvesszők az utasítások után ugyan engedélyezettek, de ritkán, vagy egyáltalán nem használatosak. Kevesebb szintaktikai kivétel és speciális eset van, mint a C-ben vagy a Pascalban. (9) A Python nagyméretű szabványkönyvtára - amelyet széles körben a nyelv egyik legnagyobb értékének tartanak - olyan eszközöket tartalmaz, amelyek alkalmasak a legkülönbözőbb feladatok ellátására, kapcsolhatóak számos szabványos formátummal és protokollal működő megoldások, így például a MIME és a HTTP támogatott és az internettel együttműködő alkalmazások számára. Tartalmaz modulokat grafikus felhasználói felületek fejlesztéséhez, relációs adatbázisokhoz való csatlakozáshoz, pszeudorandom számok generálásához, tetszőleges pontosságú tizedesjegyekkel történő számítások elvégzéséhez, továbbá nagy mértékben támogatja a hibakeresést és a tesztelés, a funkció- és folyamattesztek elvégzését. (10) (11) A specifikációk a szabványos könyvtár egyes részeire vonatkoznak. Például a WSGI (Web Server Gateway Interface) wsgiref implementációja betartja a PEP 333 szabványt, amely egy a web szerverek és a Python web applikációk közötti interfészt definiálja.

A szabványos könyvtár része a platformokon átívelő, úgynevezett cross-platform Python-kód. Továbbá a Python Package Index (PyPI), a harmadik féltől származó Python szoftverek hivatalos tárháza, 2021 szeptemberében körülbelül 329 000 csomagot tartalmaz, a legkülönbözőbb képességekkel, amely többek között magába foglal adatelemzéshez, adatbáziskezeléshez és webes keretrendszerek kezelésére alkalmas modulokat. Ezeket a modulokat a kódjuk, a belső dokumentációjuk és a tesztcsomagjaik határozzák meg. Viszonylag könnyű a modulok módosításával, egyes kódrészek újraírásával új variációk, megoldások megvalósítása (12). A nyelv sokoldalúsága és széleskörű alkalmazási lehetőségei miatt ideális választás a diplomamunkámban kitűzött feladat elvégzésére.

A fentiekben vázolt alkalmazás komponensek felhasználásával a diplomamunkámban kialakítandó architektúra a 7. ábrában kerül bemutatásra.



7. ábra – A tervezett architektúra ábrája

3.3.2 Python JSON csomagja

A JSON az adatok tárolására és cseréjére alkalmas Java alapú szintaxis, melyre a dolgozatomban, a későbbiekben részletesen kitérek. A Python programozási nyelv tartalmaz egy beépített json nevű csomagot, amely az ilyen formátumú adatok feldolgozására kitűnően alkalmas. (13) Ennek a modulnak köszönhetően a mérésadatgyűjtőktől beérkező JSON adatcsomagot könnyedén fel lehet dolgozni. A Python és JSON formátumok közötti konverziót tovább egyszerűsíti az, hogy a legfontosabb objektumoknak a saját megfelelői mindkettőben megtalálhatóak. (14)

Python	JSON Equivalent
`dict`	object
`list`, `tuple`	array
`str`	string
`int`, `float`, `int`	number
`True`	true
`False`	false
`None`	null

8. ábra - Python-objektumok és azok JSON-ba történő átalakítása. (14)

A 9. ábrán jól látható, a Python és JSON formátumok közötti konverzió egyszerűsége, amely a programozói gyakorlatban a végrehajtást, az egyes transzformációk kidolgozását nagymértékben segíti.

```
import json

# some JSON:
x =  '{ "name":"John", "age":30, "city":"New York"}'

# parse x:
y = json.loads(x)

# the result is a Python dictionary:
print(y["age"])
```

9. ábra - JSON formátumról Python formátumra való konvertálás (13)

A 9. ábrán egy kódrészlet látható, amely reprezentálja a két formátum közötti konverzió egyszerűségét a programozói gyakorlatban is.

3.4 A JSON adatstruktúra

A JSON (JavaScript Object Notation) adatformátum a JavaScript programozási nyelv adattípusain alapul és kiválóan alkalmas különböző objektumok adatbázisban történő reprezentálására. A JSON az elmúlt években óriási népszerűségre tett szert a webfejlesztők körében, és az interneten történő információcsere elsődleges formátumává vált. (15) (16) (17)

A JSON egy egyszerűen használható adatcsere-formátum, melynek olvasása és írása, szintaktikai adottságai miatt, egyszerű feladat az emberek számára. Ezen felül nagyon könnyen lehet programozottan elemezni, illetve generálni. (18) Ezen túlmenően a JSON-adatok könnyen továbbíthatóan számítógépek között, és bármilyen programozási nyelvvel felhasználhatók, mivel kizárólag szöveges formátumúak. (19) Az, hogy egy programozási nyelv támogatja-e az objektumokat, és ha igen, milyen tulajdonságokkal és korlátozásokkal rendelkeznek, nagyon eltérő. Az objektumorientált-modellek rendkívül változatosak lehetnek, és folyamatosan változnak, de a JSON struktúra által biztosított

lehetőségek ezzel szemben egyszerű módot biztosítanak a név/érték párok kezelésére. (17) A JSON két strukturált típust és négy primitív típust tud reprezentálni. A négy primitív típus a karakterláncok, számok, logikai típusú változók és a null, illetve a struktúráltak az objektumok és a tömbök.

Alapvetően a JSON adatformátum két struktúrára épül: (18)

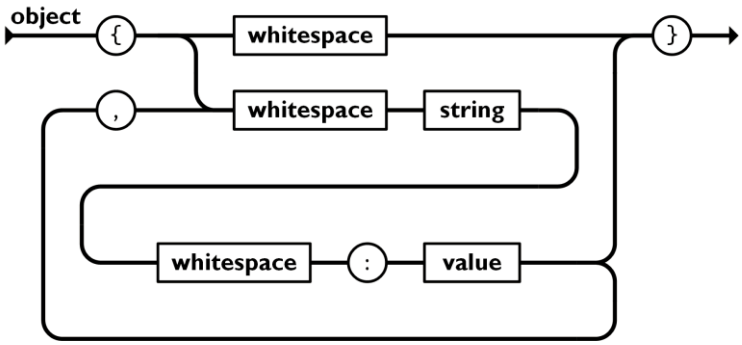
- Név/érték párok gyűjteménye. A különböző nyelveken ez objektum, rekord, struktúra, szótár, hash-tábla, kulcsos lista vagy asszociatív tömb formájában valósul meg.
- Értékek rendezett listája. A legtöbb nyelven ez tömb, vektor, lista vagy szekvencia formájában valósul meg.

Ezek olyan adatszerkezetek, amelyek általánosan alkalmazhatóak. Gyakorlatilag minden modern programozási nyelv támogatja őket valamilyen formában. Ennek fényében ésszerű elvárni, hogy egy olyan adatformátum, amely a programozási nyelvekkel felcserélhető, ezekre a struktúrákra épüljön. (18)

3.4.1 JSON struktúra elemek

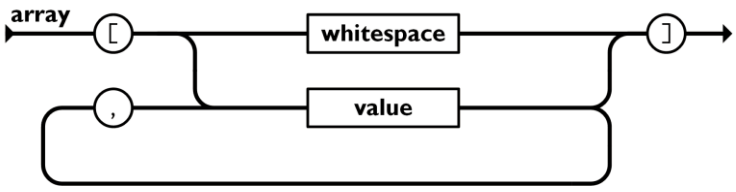
A JSON adatformátumban a struktúrák az alábbi formákat vehetik fel:

- Objektum: Egy pár szögletes zárójel veszi körül a nulla vagy több név/érték párost, hogy egy objektumstruktúrát ábrázoljon. A név egy karakterlánc. Minden nevet egyetlen kettőspont jel követ, amely elválasztja a nevet az értéktől. Az értéket a következő névtől egyetlen vesszővel választjuk el. A JSON szintaxis nem korlátozza a névként használt karakterláncokat, nem követeli meg, hogy a névsorok egyediek legyenek, és nem ad súlyt a név/érték párok bemutatásának sorrendjére. Ezek mind olyan szemantikai megfontolások, amelyeket a JSON-adatcsere konkrét felhasználását meghatározó specifikációk határozhatnak meg. (17) (18) (15)



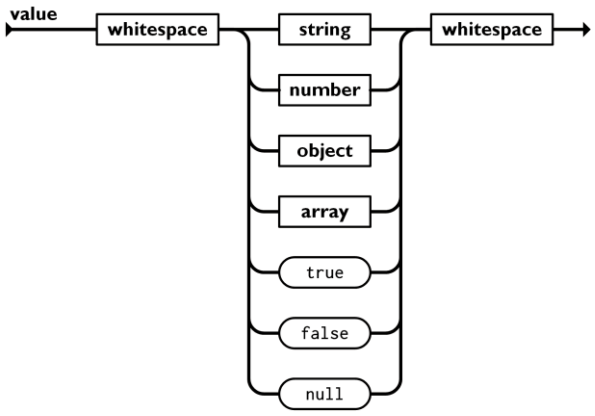
10. ábra - JSON objektum felépítése (18)

- Tömb: Egy tömbszerkezetben nulla vagy több értéket vesz körül egy pár szögletes zárójel. Az értékek elválasztására vesszőket használunk. Az értékek sorrendjének nincs különösebb jelentősége a JSON szintaxisban, tömbszerkezetet viszont gyakran használják olyan helyzetekben, amikor a sorrendnek van valamilyen szemantikája. (15) (18) (17)



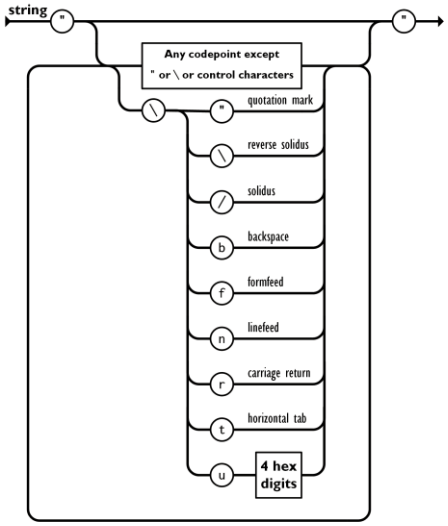
11. ábra - JSON tömb felépítése (18)

- Értékek: Az érték lehet egy karakterlánc idézőjelben, egy szám, igaz, hamis, null, objektum vagy tömb. Ezek a struktúrák egymásba ágyazhatók. (18)



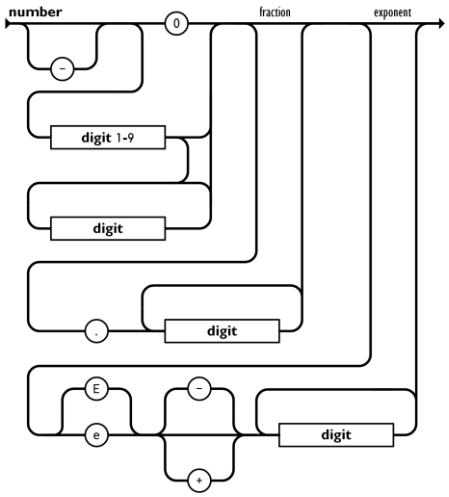
12. ábra - JSON érték felépítése (18)

- Karakterláncok: A karakterláncok ábrázolása a C programozási nyelvcsaládban használt konvenciókhoz hasonlóan történik. Az idézőjelek a karakterláncok kezdetére és végére kerülnek. Néhány karakter kivételével minden Unicode karakter idézőjelek közé helyezhető. (18) (17) (15). Ezek a karakterek az idézőjel (U+0022), a fordított perjel (U+005C) és az U+0000 és U+001F között elhelyezkedő kontroll karakterek, mint például \n új sor karakter (U+000A). Ez a szabály megkerülhető bizonyos esetekben úgy, hogy a nem engedélyezett karakterekből kettőt használunk. Ennek megfelelően egy karakterlánc mely egyetlen fordított perjelet tartalmaz “ \\ “ formában reprezentálható. (15) (17)



13. ábra - JSON karakterlánc felépítése (18)

- Számok: A szám hasonló a C vagy Java programozási nyelvekben használt számokhoz, azzal a kivétellel, hogy a JSON nem használja az oktális vagy hexadecimális formátumot. (18)



14. ábra - JSON szám felépítése (18)

- Szóköz: A struktúrákon belüli elemek elválasztásán felül jelentéktelen szóköz használata megengedett bármelyik szerkezeti karakter előtt vagy után. (17) A szóköz struktúra magában foglalhat tabulátor és új sor karaktereket is. (18)

Mint láthatjuk a JSON-adatformátum nagy mértékben építkezik az Unicode karakterekre (15) (17) (18), azonban fontos megjegyezni, hogy a JSON nyelvtan olyan kódpontokat is megenged, amelyekhez a Unicode jelenleg nem biztosít karakter hozzárendeléseket. (17)

3.4.2 A JSON lehetséges felhasználásai

A JSON struktúra elemekről szólófejezetben bemutatott elvek hozzásegítik a felhasználót, programozót, hogy olyan egyszerű, szövegszerű elektronikus üzeneteket hozhassanak létre, melyek továbbítása nem támaszt különösebb követelményt az átviteli közeggel szemben, illetve a struktúra segítségével akár rendkívül összetett kommunikációt is megvalósíthatnak.

Ez alapján a JSON rendkívüli adottságai számos funkció programozására, ezáltal a legkülönbözőbb megoldások kiszolgálására teszik alkalmassá. A JSON egy szintaxis, amely alkalmas információ tároláshoz, információ cseréhez rendszerek között, akár Interneten keresztül is, de az alkalmazások és a web-szerver, vagy a web-szerver és a felhasználók között. A teljesség igénye nélkül a lehetséges felhasználások:

- alkalmazás integrációban,
- biztonsági megoldásokban,

- adattárolásban.

Dolgozatomban ez utóbbi két felhasználási módra részletesebben is kitérek.

JSON alapú authentikáció

A JSON Web Encryption (JWE) egy olyan módszer a titkosított tartalom webes megjelenítésére, amely JSON-alapú adatstruktúrákat használ. E specifikáción kívül a különálló JSON Web Algorithms (JWA) specifikáció és az abban meghatározott IANA-nyilvántartások tartalmazzák a kriptográfiai algoritmusok és egyedi azonosítók leírását a specifikációval való használatra. (20) A JSON Web Signature (JWS) specifikáció a JSON Web Signature (JWS) specifikációhoz kapcsolódó digitális aláírások és message authentication codes (MAC) képességeit írja le. A JWS a digitálisan aláírt vagy MAC-kódolt tartalmat JSON adatstruktúrákkal és base64url kódolással reprezentálja, amelyeket a Java Web Service egyaránt támogat. Az RFC 7159 (15) 2. szakaszának megfelelően megengedett, hogy ezek a JSON adatstruktúrák a JSON-értékek vagy szerkezeti karakterek előtt és után szóközöket és/vagy sortöréseket tartalmazzanak. (21)

A JWS az alábbi logikai értékeket tartalmazza:

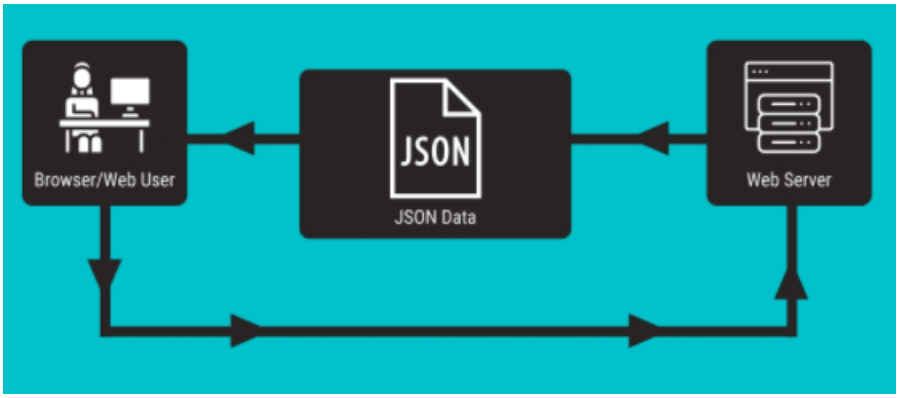
- JOSE Header
- JWS Payload
- JWS Signature

A JWS JOSE fejlécet reprezentáló JSON objektum(ok) tagjai leírják a JWS Protected Headerre és a JWS Payloadra alkalmazott digitális aláírást vagy MAC-et, valamint opcionálisan más JWS-tulajdonságokat. A JWS-elemzőknek vagy el kell utasítaniuk a duplikált fejparaméterneveket tartalmazó JWS-eket, vagy olyan JSON-elemzőt kell használniuk, amely csak a lexikailag utolsó duplikált tag nevét adja vissza. (21)

Ezen túlmenően az autentikációs folyamatokhoz kialakítható egy un. JSON Web Token megoldás is, melyekkel a biztonsági lehetőségek bővíthetőek.

JSON alapú adatbázisok

A közelmúltban az alkalmazásfejlesztés új korszaka jelent meg, amely az un. Big Data, avagy a nagy adatmennyiségű technológián és az adatokhoz, erőforrásokhoz való könnyű hozzáférésen, például a mobileszközökön keresztüli hozzáférésen alapul.



15. ábra – Példa beágyazott JSON alapú adatcserére

Mindezek a kérdések hatékonyabban kezelhetők a JSON (és a JavaScript) technológiával. Szinte minden relációs adatbázis-rendszer támogatja a JSON-t, részben az ANSI SQL szabványnak megfelelően, részben pedig más specifikációknak megfelelően. (22) A relációs adatbázis-kezelő rendszerek (RDBMS) az idők során rendkívül hatékonynak bizonyultak a jól meghatározott sémákkal rendelkező strukturált adatok kezelésében. Bár ez így van, a relációs rendszerek általában nem az első választás az adatkezelésre olyan helyzetekben, ahol a sémák nincsenek előre definiálva, vagy ahol az adatkezelésnek rugalmasnak kell lennie a változások és variációk tekintetében. (23)

Az SQL-adatbázisok helyett gyakran a JSON-t támogató NoSQL adatbázis-rendszereket használják az ilyen alkalmazások perzisztenciájának biztosítására. Az SQL/XML (24) (25) (26) (27) lehetővé teszi a félig strukturált adatkezelést az RDBMS-en belül. Az XML adatformátum viszont a dokumentumfeldolgozás világából származik, és számos szabályt tartalmaz a dokumentumorientált félig strukturált adatok ábrázolására. Bonyolultsága miatt az XML rossz választás az adatorientált félstruktúrált adatok ábrázolására. A JSON (18) népszerű formátum lett az adatorientált felhasználási esetek számára, mivel egyszerű, kompakt és viszonylag könnyen feldolgozható. Népszerűségét elősegítette, hogy olyan nyelvek, mint a JavaScript, natívan támogatják. A NoSQL-közösség (23) egyre inkább a JSON-ra összpontosít, az olyan termékek, mint a MongoDB (28) natív JSON-tárolókat biztosítanak. A JSON-adatok mennyisége növekszik, mivel a JavaScriptet egyre gyakrabban használják a mobil- és webes alkalmazások fejlesztésében (klienseken és szervereken egyaránt), és felgyorsul a NoSQL rendszerek használatának trendje az adatok tárolására. (23) A JSON objektumok tárolását támogató relációs

adatbázis a képességek bővítése mellett lehetővé teszi a strukturált és félig strukturált adatok egyetlen egységbe történő integrálását. A külön tárolt és különböző programok által használt JSON-objektumok esetében minden program felelős a saját adatainak kezeléséért. Ha a JSON-támogatást egy relációs adatbázis-rendszer biztosítja, akkor a rendszer felelős az összes adatadminisztrációért. Hasonló érvelés hozható fel a biztonság és a tranzakciók kezelése esetében is, mivel a rendszer átveszi a felelősséget a biztonság és a tranzakciók kezelésének adminisztrációjáért, így a felhasználók programjainak nem kell megvalósítaniuk ezeket a funkciókat. Végül a JSON integrálása egy relációs adatbázis-kezelő rendszerbe (RDBMS) növeli a termelékenységet, mivel a rendszer számos olyan feladatot átvesz, amelyet egyébként a programozók programjaikban végeznének el. (22)

3.5 HTML

A Szakdolgozatomhoz kapcsolódó programozási feladat, illetve kidolgozandó megoldás részeként adatok webes elérése, azaz adatok webes felületen történő hozzáférhetőségének biztosítása is feladat. A modellben begyűjtött mérési adatok lekérdezhetőségének biztosítása, az adatok publikálása a megcélzott funkcionalitás része, melyre különböző protokollok biztosítanak megoldást.

A HTML kifejezés a Hypertext Markup Language (hiperszöveges jelölőnyelv) rövidítése. A hiperszöveg megnevezés több évtizedes múltra, gyakorlatilag az Internet elterjedéséig nyúlik vissza, sőt talán ennek alkalmazása segített az Internet elterjedésében. Tim Berners-Lee nevéhez fűződik a programnyelv kifejlesztése. A HTML kifejezetten az Internetre készült, az ezen a nyelven létrehozott, egymással kapcsolatban álló weboldalak hálózatát nevezzük World Wide Web-nek.

Funkciója az, hogy meghatározzon linkeket, melyek egy állományban találhatóan, és ezáltal lehetővé tegye, hogy a dokumentum, vagy más dokumentum különböző pontjai között navigáljon szabadon a felhasználó. A HTML alapvetően a webes felületek létrehozásának, programozásának sztenderd nyelvezete.

3.5.1 A HTML felépítése

Egy HTML dokumentumban három alapvető rész különböztetünk meg:

- dokumentum típus definíció

- fejléc
- törzs

Napjainkban már nagyon ritka esetben programozunk html nyelven, köszönhetően annak, hogy számos fejlesztést segítő, akár grafikus megoldás is létezik.

4 PROBLÉMA ELEMZÉSE, A SPECIFIKÁCIÓ KIDOLGOZÁSA

A szakdolgozat során első lépésként a már korábban ismertetett elméleti hátteret felhasználva szeretnék megoldást találni a mérésadatgyűjtő rendszertől beérkezett jel feldolgozására. Ezt egy tesztfájlon keresztül fogom megtenni. Itt egyből felmerül az adattömbök felépítésének kérdése. Jelen esetben ez nem okoz problémát, hiszen a forrásfájl ismert, azonban egy olyan rendszerben, ahol több egymástól független, esetleg több különböző gyártó által fejlesztett eszközöket szeretnénk monitorozni, kiemelt figyelmet kell fordítani a beérkezett adattömb uniformitására. Amennyiben ez nem teljesül, azt a specifikációban jelölni kell, és ezeket a kivételeket szoftveresen kezelni kell, annak érdekében, hogy a rendszer ne fusson már az első lépésnél hibára, vagy ne közöljön hamis adatokat a formátumbéli eltérések végett. A beérkezett adatok feldolgozása alatt a megfelelő értékek pontos kiolvasása és tárolása értendő. Ugyan jelen példában az összes beérkező jel már megtalálható a számítógépen, azonban ez egy valós idejű IoT rendszerre nem igaz. Fontos, hogy az összes beérkezett mérési eredmény tárolva legyen. Ennek okai többek között a visszakövethetőség, illetve a nem valós idejű kiértékelés.

A következő lépés a tárolt adatok konverziója és grafikus megjelenítése. Mint azt már említettem a HTML formátum mellett döntöttem a megjelenítés során. Ennek oka az, hogy a konverzió a JSON, Python és HTML formátumok között gyors és egyszerű. Ez azért fontos mert egy nagyobb rendszerben valós körülmények között százezres, vagy akár milliós nagyságrendű is lehet a beérkezett adatok száma adott időintervallum alatt. A HTML így erre a célra tökéletes példamegoldásként szolgálhat. Végül pedig szeretnék javaslatot tenni a feladat megoldása során feltárt potenciális problémák megoldására, ezzel kiindulási alapként szolgálva egy komplex mérésadatgyűjtő rendszer monitorozásához.

5 A BEÉRKEZŐ ADAT FELDOLGOZÁSA

5.1 Felhasznált eszközök, fájlok

5.1.1 PyCharm 2021.1.1, Python 3.8

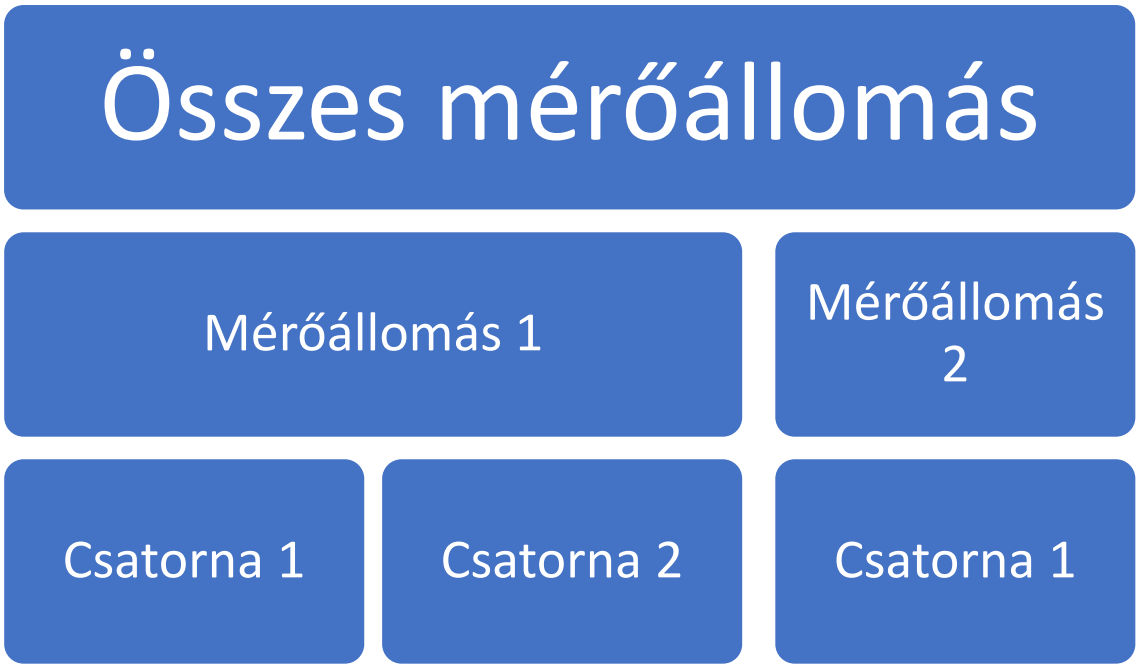
A PyCharm egy integrált fejlesztőkörnyezet (Integrated Development Environment, gyakran használt rövidítése: IDE) melynek segítségével különböző fejlesztői eszközöket egy közös grafikus felület segítségével lehet használni. Támogatja mind a Pythont és a HTML formátumot, így egy fejlesztői környezet használatával a probléma teljes egésze megoldható. Ezen felül rendelkezik beépített debuggerrel, illetve számos olyan csomaggal, mint például a Python JSON, mely a feladat elvégzéséhez elengedhetetlen (29). Továbbá a PyCharm számos funkciója között megtalálható az intelligens kódkiegészítés, a kódvizsgálat, a hibák azonnali kiemelése és a gyors javítás, valamint az automatikus kódjavítás és a kiterjedt navigációs lehetőségek (30). Ezek az eszközök nagy mértékben megegyszerűsítik egy a szakdolgozatomban tárgyalthoz hasonló rendszer nulláról való kidolgozását.

5.1.2 Tesztfájl bemutatása

A tesztelés alapjaként használt JSON formátumú fájl egy részlete:

{"time": "2021.05.28. 00:57:34", "dev_id": 2, "input_count": 16, "sample": [{"ch_id": 0, "bar_code": "7527674436568", "value": 847.955, "unit": "kWh"}, {"ch_id": 1, "bar_code": "1414104700084", "value": 2419.1431, "unit": "kWh"}, {"ch_id": 2, "bar_code": "4916432499885", "value": 1509.863, "unit": "kWh"}, {"ch_id": 3, "bar_code": "5692408084864", "value": 2267.1841, "unit": "kWh"}, {"ch_id": 4, "bar_code": "1357370018951", "value": 86.007, "unit": "kWh"}, {"ch_id": 5, "bar_code": "5672421455384", "value": 2366.9341, "unit": "kWh"}, {"ch_id": 6, "bar_code": "6012484431267", "value": 23.686, "unit": "kWh"}, {"ch_id": 7, "bar_code": "6005784273149", "value": 7788.4229, "unit": "kWh"}, {"ch_id": 8, "bar_code": "2892986536028", "value": 301.32, "unit": "kWh"}, {"ch_id": 9, "bar_code": "9219781756407", "value": 93.747, "unit": "kWh"}, {"ch_id": 10, "bar_code": "7411702275277", "value": 2550.1321, "unit": "kWh"}, {"ch_id": 11, "bar_code": "3845440149301", "value": 2018.0129, "unit": "kWh"}, {"ch_id": 12, "bar_code": "", "value": 0.0, "unit": ""}, {"ch_id": 13, "bar_code": "", "value": 0.0, "unit": ""}, {"ch_id": 14, "bar_code": "", "value": 0.0, "unit": ""}, {"ch_id": 15, "bar_code": "", "value": 0.0, "unit": ""}]}

Ez a mérésadatgyűjtő rendszer egy mérőállomása által gyűjtött adat egy adott időpontban. Alapvetően itt is a korábbi fejezetekben bemutatott ismereteket kell alkalmazni. Láthatjuk, hogy a mérési eredmények több alap formátum kombinációjaként kapjuk meg. Ezeknek a megfelelő sorrendben történő visszafejtése a kulcs a fájl feldolgozásához. A legfelső és egyben legnagyobb egységet a fenti példán nem látható tömb adja, amely magába foglalja az összes mérési eredményt. Ezen belül a következő alegység a fenti példán bemutatott objektum. Itt egy mérőállomás egy adott időpont béli mért eredményeit láthatjuk. Ez az objektum számos alegységet tartalmaz. Ezek az alegységek egy kivétellel mind a JSON fejezetben megismert név-érték párok, mely jelen esetben leírják a mérés minden mérési eredményére érvényes paramétereket. Az egy kivétel a mérőműszer sajátosságaiból adódik. Jelen műszer több csatornán mér. A csatornákon mért eredményeket az objektumon belüli tömb formájában találhatjuk meg. Ez a tömb további objektumokat tartalmaz, melyben a név-érték párok már az adott csatorna karakterisztikáit, így például a konkrét mért eredményt, mértékegységet és a csatorna azonosítóit, tartalmazzák. Az itt leírtak ismeretével könnyedén, szisztematikusan ki lehet fejteni a szükséges adatokat a JSON objektumból.



16. ábra - A tesztfájl vázlatos felépítése

5.2 A fájl beolvasása és felolgozása

Ahhoz, hogy a tesztfájlt Pythonban megfelelően kezelni tudjam létrehoztam egy osztályt, amely a JSON fájl különböző változóit képes tárolni.

```
class Sample:

    def _init_(self, time, dev_id, ch_id, bar_code, value, unit):

        self.time = time #mérés időpontja
        self.dev_id = dev_id #mérőeszköz
        self.ch_id = ch_id  #csatorna azonosítói
        self.bar_code = bar_code #csatorna azonosítói
        self.value = value #mért eredmény
        self.unit = unit #mértékegység
```

Ennek segítségével az objektum kezelése sokkal könnyebben megoldható, továbbá lehetővé teszi, hogy minden csatorna minden mért eredményét külön kezeljük. Ez segíti a pontosabb monitorozást, és javítja az átláthatóságot.

A következő fontos lépés a releváns fájlok megnyitása és változók létrehozása után a teljes JSON objektumon való iteráció, mely során kiolvassuk a fontos adatokat. Ezt az alábbi kód segítségével értem el:

```
for i in range(0, len(parsed_input)):

  if "dev_id" in parsed_input[i]:
     measurements.time = parsed_input[i]['time']
     measurements.dev_id = parsed_input[i]['dev_id']
     output_data.write(json.dumps(measurements.dev_id))
     output_data.write(json.dumps(measurements.time))
     for s in range(parsed_input[i]['input_count']):
      measurements.ch_id = parsed_input[i]['sample'][s]['ch_id']
      measurements.bar_code = parsed_input[i]['sample'][s]['bar_code']
      measurements.value = parsed_input[i]['sample'][s]['value']
      measurements.unit = parsed_input[i]['sample'][s]['unit']
      measurements_array.append(measurements)
      output_data.write(json.dumps(measurements.ch_id))
      output_data.write(json.dumps(measurements.bar_code))
      output_data.write(json.dumps(measurements.unit))
      output_data.write(json.dumps(measurements.value))
```

A fontosabb elemek:

- A külső for ciklus segítségével a program végigfut az összes mérőállomáson. Futási paraméternek a parsed_input változót adtam meg. Ez a változó tartalmazza az összes beolvasott adatot, ezt szeretném feldarabolni feldolgozható méretekre.
- Mivel nem garantált, hogy az összes mérőműszer sorszáma egymást követő, és nincs köztük kimaradt szám, így fontos vizsgálni, hogy értelmezhető-e a beolvasni kívánt dev_id. Amennyiben ezt a lépést kihagyjuk, a program hibára futhat.

- Amennyiben a dev_id értelmezhető azt kiolvassuk az idővel együtt és beírjuk egy fájlba. Ezt érdemes még itt, a következő for ciklus előtt megtenni, hiszen több mérési ereménynek azonosak a dev_id és time paraméterei.
- Mivel itt a cél a különálló csatornák mérési eredményeinek kiolvasása, a JSON fájlban megtalálható input_count változót nem tároljuk. Ezt arra használjuk, hogy a mérőállomás csatornáin végig iteráljunk. Ezt egy újabb for ciklus segítségével lehet megtenni, mely paraméternek megkapja az input_count változót.
- Végül kiolvassuk a többi releváns változót, és azokat kiírjuk egy fájlba

Megjegyzések: A kódban nem csak a fájlba történő kiírás látható, hanem megtalálható egy változó is, melynek feladata kizárólag a tárolás. Továbbá az is látható, hogy az összes adatot egy output_data nevű fájlba írom ki. Ezeknek okai az egyszerűbb tesztelés és könnyebb prezentáció. Jelen kód sablonként szolgálhat egy későbbi, tovább fejlesztett verzióhoz, amely dinamikus változókat, vagy akár online tárolási lehetőségeket is lehetővé tesz.

5.3 A beolvasott fájlok HTML konverziója

A HTML konverziónál a kiindulási alap az előző fejezetrészben feldolgozott fájlrész.

{

"time":"2021.05.28. 00:57:34",

"dev_id":2,

"ch_id":0,

"bar_code":"7527674436568",

"value":847.955,

"unit":"kWh"

}

Jól látható, hogy itt már csak egy mérőállomás egy adott csatornájának a mérési eredménye látható.

5.3.1 Adatok megjelenítése böngészőben, táblázat formájában

A PyCharm számos moduljának köszönhetően a feldarabolt, és külön lementett részfájlokat könnyű HTML formátumra konvertálni. A felhasznált modul:

```
import json2table
```

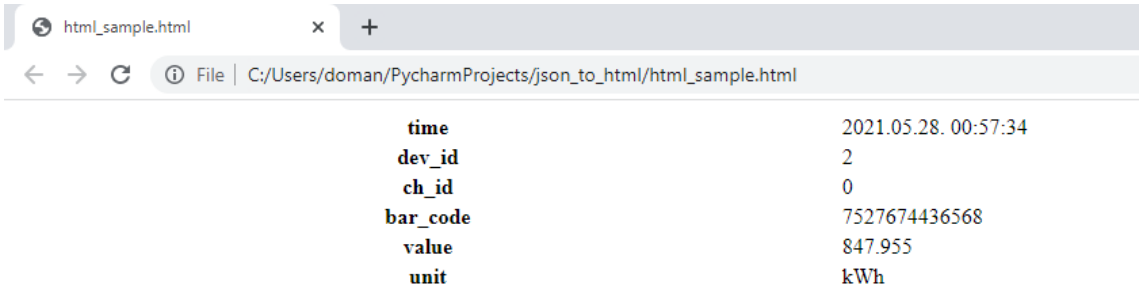
A specifikációk ismeretével a konverzió könnyen elvégezhető:

```
sample_json = open('data_processed.json', 'r')
html_sample = open('html_sample.html', 'w')
sample_dict = json.load(sample_json)
direction = "LEFT_TO_RIGHT"
attributes = {"style": "width:100%"}
html_sample.write(json2table.convert(sample_dict,
build_direction=direction,table_attributes=attributes))
```

A kapott HTML fájl:

```
<table style="width: 100%;">
    <tr>
        <th>time</th>
        <td>2021.05.28. 00:57:34</td>
    </tr>
    <tr>
        <th>dev_id</th>
        <td>2</td>
    </tr>
    <tr>
        <th>ch_id</th>
        <td>0</td>
    </tr>
    <tr>
        <th>bar_code</th>
        <td>7527674436568</td>
    </tr>
    <tr>
        <th>value</th>
        <td>847.955</td>
    </tr>
    <tr>
        <th>unit</th>
        <td>kWh</td>
    </tr>
</table>
```

A táblázat tetszőleges böngészőben megjelenítve:



17. ábra - HTML táblázat böngészőben

Jelen esetben a Google Chrome böngészőt használtam, de ennek nincs különösebb jelentősége.

Ezen felül lehetőség van közvetlenül a kódból megnyitni a böngészőt. Ehhez szükséges importálni a webbrowser modult.

```
import webbrowser
webbrowser.open_new_tab(html_sample)
```

5.3.2 Adatok megjelenítése böngészőben, grafikus módon

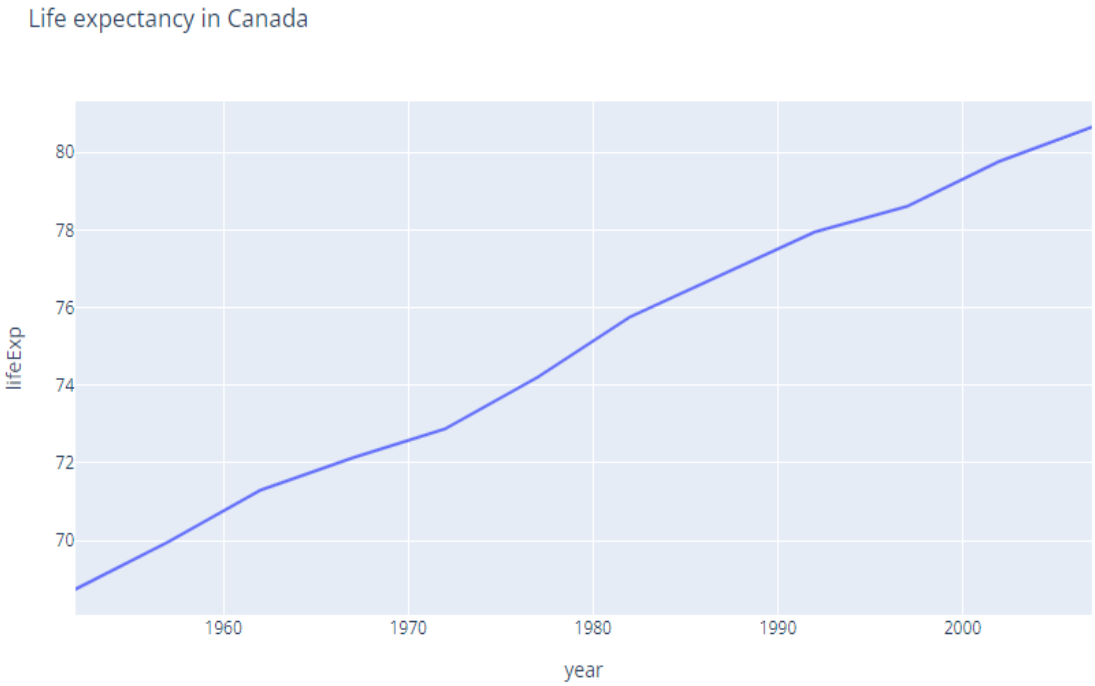
Ahhoz, hogy grafikusan Pythonból meg tudjunk ábrákat jeleníteni, érdemes először a grafikont elkészíteni, majd ezt HTML formátumba konvertálni.

Az ilyen grafikonok elkészítésére tökéletesen alkalmas modul például a Plotly vagy a Plotly Express. A Plotly Express egy magas szintű felület a Plotlyhoz, amely egyszerűen használható, és könnyen stilizálható ábrákat készít. A Plotly számos adattípussal dolgozik, és egyszerűen stilizálható ábrákat készít. A Px.line minden egyes adatpontot egy kétdimenziós (2D) térben lévő polivonalas jelölés csúcspontjaként ábrázol (amelynek helyét az x és y oszlopok értékei határozzák meg) (31).

```
import plotly.express as px

df = px.data.gapminder().query("country=='Canada'")
fig = px.line(df, x="year", y="lifeExp", title='Life expectancy in
Canada')
fig.show()
```

Példa a plotly express használatára egy egyszerű grafikon létrehozásakor (31)



18. ábra - PlotLy Express grafikon (31)

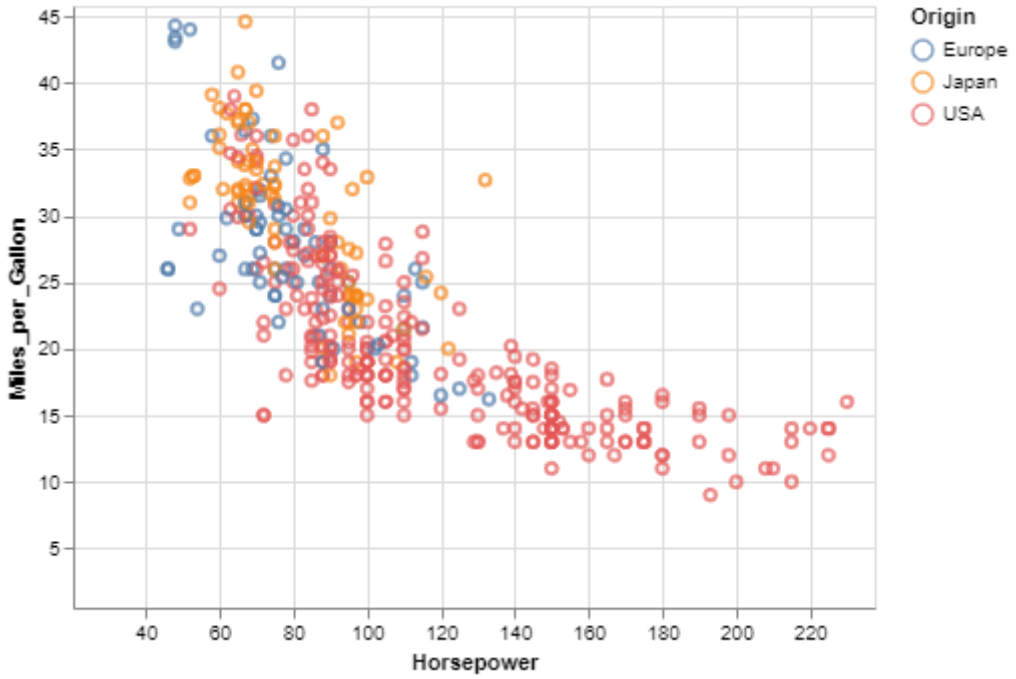
A másik lehetőség a grafikon generálására az Altair modul. Az Altair egy Python könyvtár deklaratív statisztikai vizualizációhoz, amely a Vegán és a Vega-Lite-on alapul. Az Altair egy hatékony és tömör vizualizációs nyelvezetet biztosít, amely lehetővé teszi, hogy rövid idő alatt statisztikai vizualizációk széles skáláját hozhassuk létre. (32)

Példa az Altair használatára egy egyszerű grafikon létrehozása esetén (32):

```
import altair as alt

# load a simple dataset as a pandas DataFrame
from vega_datasets import data
cars = data.cars()

alt.Chart(cars).mark_point().encode(
    x='Horsepower',
    y='Miles_per_Gallon',
    color='Origin',
).interactive()
```



19. ábra - Altair grafikon (32)

.

A felmerült probléma: A beérkezett JSON fájl, és az ebből készített szintén JSON fájlrészek nem megfelelő formátumban tartalmazzák az adatot. Ugyan a változók fel vannak dolgozva, és a különböző mérőállomások mérési eredményei elérhetőek, azonban a begyűjtött adat “time” név/érték párja az időt nem egész szám (integer) formájában, hanem karakterlánc (string) formájában tárolja. A PlotLy és hozzá hasonló modulok nem képesek ennek a formátumnak grafikonná való átalakításává. Ez okból kifolyólag hiába

tudjuk a mért adatokat táblázatosan, akár az összeset egyszerre megtekinteni, jelen formátumban agrafikus megjelenítés nem lehetséges.

time	dev_id	input_count	sample/0/ch_id	sample/0/bar_code	sample/0/value	sample/0/unit	sample/1/ch_id	sample/1/bar_code	sample/1/value	sample/1/unit	sample/2/ch_id	sample/2/bar_code	sample/2/value	sample/2/unit
2021.05.28. 00:57:34	2	16	0	7527674436568	847.955	kWh	1	1414104700084	2419.1431	kWh	2	4916432499885	1509.863	kWh
2021.05.28. 00:57:34	2	16	0	7527674436568	847.955	kWh	1	1414104700084	2419.1431	kWh	2	4916432499885	1509.863	kWh
2021.05.28. 00:29:14	3	16	0	0000000000000	0	kWh	1	5326807498938	7662.8198	kWh	2	3092874884609	20.386	kWh
2021.05.28. 00:29:14	3	16	0	0000000000000	0	kWh	1	5326807498938	7662.8198	kWh	2	3092874884609	20.386	kWh
2021.05.28. 01:02:10	8	14	0	5425269603727	501.801	kWh	1	4246724247932	2760.8501	kWh	2	9947865605357	74.4	kWh
2021.05.28. 01:02:10	8	14	0	5425269603727	501.801	kWh	1	4246724247932	2760.8501	kWh	2	9947865605357	74.4	kWh
2021.05.28. 01:04:39	55	12	0	9599006175993	22.3	kWh	1	2548944354059	2.942	kWh	2	5250949263577	0.742	kWh
2021.05.28. 01:04:39	55	12	0	9599006175993	22.3	kWh	1	2548944354059	2.942	kWh	2	5250949263577	0.742	kWh
2021.05.28. 01:01:23	6	16	0	1802251935006	961.854	kWh	1	3088952898972	2754.0569	kWh	2	2783471941949	9061111808	kWh
2021.05.28. 01:01:23	6	16	0	1802251935006	961.854	kWh	1	3088952898972	2754.0569	kWh	2	2783471941949	9061111808	kWh
2021.05.28. 00:59:24	7	16	0	7147442102435	0	kWh	1	4362080693241	12588.1787	kWh	2	6878930926325	8349.3779	kWh
2021.05.28. 00:59:24	7	16	0	7147442102435	0	kWh	1	4362080693241	12588.1787	kWh	2	6878930926325	8349.3779	kWh
2021.05.28. 00:35:19	4	16	0	8867716193196	78.682	kWh	1	3358668088913	2974.5391	kWh	2	5739980936058	1293.255	kWh
2021.05.28. 00:35:19	4	16	0	8867716193196	78.682	kWh	1	3358668088913	2974.5391	kWh	2	5739980936058	1293.255	kWh
2021.05.28. 00:56:22	11	11	0	5203319787975	17177.9082	kWh	1	7414209246634	295.557	kWh	2	1020733118055	1488.826	kWh
2021.05.28. 00:56:22	11	11	0	5203319787975	17177.9082	kWh	1	7414209246634	295.557	kWh	2	1020733118055	1488.826	kWh
2021.05.28. 01:02:23	12	6	0	4398617744445	2178.262	kWh	1	3374406099317	108.585	kWh	2	9414861798289	2291.7729	kWh
2021.05.28. 01:02:23	12	6	0	4398617744445	2178.262	kWh	1	3374406099317	108.585	kWh	2	9414861798289	2291.7729	kWh
2021.05.28. 00:48:09	15	10	0	4398617744445	9510.8643	kWh	1	3374406099317	30237.3926	kWh	2	9478236436848	6933.8032	kWh
2021.05.28. 00:48:09	15	10	0	4398617744445	9510.8643	kWh	1	3374406099317	30237.3926	kWh	2	9478236436848	6933.8032	kWh
2021.05.28. 01:12:34	2	16	0	7527674436568	847.956	kWh	1	1414104700084	2419.1431	kWh	2	4916432499885	1509.9189	kWh
2021.05.28. 01:12:34	2	16	0	7527674436568	847.956	kWh	1	1414104700084	2419.1431	kWh	2	4916432499885	1509.9189	kWh

20. ábra - A teljes JSON fájl táblázatos megjelnítésének részlete

6 Lehetőségek a további fejlesztése

6.1 MySQL adatbázis használata

A használt tesztfájl feldolgozása során egy helyi, állandó fájlba gyűjtöttem a feldolgozott részfájlokat. A fejezetrészben említettem, hogy erre a problémára megoldás lehet, egy online adatbázis.

A MySQL egy relációs adatbázis-kezelő rendszer. A számítógépes adatbázisban tárolt adatok hozzáadásához, eléréséhez és feldolgozásához egy adatbázis-kezelő rendszerre, például a MySQL Szerverre van szükség. Az adatbázis-kezelő rendszerek önálló segédprogramként vagy más alkalmazások összetevőiként központi szerepet játszanak a számítástechnikában, és az itt bemutatott probléma megoldására is alkalmas lehet. Ezt a feltételezést tovább alátámasztja az, hogy a MySQL szoftver nyílt forráskódú, ami azt jelenti, hogy bárki használhatja és módosíthatja. A MySQL-t bárki ingyenesen használhatja, ha letölti az internetről. A felhasználók tanulmányozhatják a forráskódot és módosíthatják azt az igényeiknek megfelelően. A MySQL a GPL (GNU General Public

License) alapján határozza meg, hogy a különböző helyzetekben mit lehet és mit nem lehet a szoftverrel csinálni.

A MySQL Server-t azért hozták létre, hogy a meglévő megoldásoknál sokkal gyorsabban kezelje a nagyméretű adatbázisokat, és ebből kifolyólag már évek óta használják nagy volumenű termelési környezetekben. A MySQL Server annak ellenére, hogy folyamatosan fejlesztik, mára már nagy és hasznos funkciókkal rendelkezik. A MySQL Server a csatlakoztathatósága, sebessége és biztonsága miatt kiválóan alkalmas adatbázisok interneten keresztüli elérésére. (33)

Ezen felül lehetséges MySQL adatbázist Pythonon kereszül felépíteni, amely gördülékenyebbé teszi a folyamatot, hiszen az eddigi munka is Pythonban folyt.

Példa, a Pythonban történő MySQL csatlakozásra (34):

```
import mysql.connector

mydb = mysql.connector.connect(
  host="localhost",
  user="yourusername",
  password="yourpassword"
)
```

6.2 Beérkező adatformátum megfelelő formátumválasztása

Jelen munkában a legnagyobb problémát több ízben is a beérkezett adat formátuma okozta. Ennek megoldására két fő lehetőséget látok:

- A beérkező adatok formátuma kompatibilis többek között a grafikus megjelenítésre alkalmas programokkal.
- Új grafikus megjelenítési módok feltérképezése és használata

7 Összefoglalás

A szakdolgozatom során feltártam a mérésadatgyűjtő rendszerek működési alapelveit, a működésének megértéséhez szükséges elméleti tudást és egy tesztfájlon keresztül kidolgoztam egy sablont, mely alkalmas arra, hogy ezt alapul véve a későbbiekben egy nagyobb volumenű, akár több ezer mérőműszerből álló rendszert monitorozhassunk.

Ezen felül tárgyaltam a begyűjtött adatok megjelenítési lehetőségeit, mind táblázatosan, mind pedig grafikus formában. Itt kitértem a lehetséges hibákra, melybe a fejlesztés során ütközhetünk. Ilyen például az adatok formátuma miatti inkompatibilitás.

Végezetül pedig javaslatot tettem a továbbfejlesztés lehetséges irányzataira, melyek segítségével egy kompetens mérésadatgyűjtő rendszer felügyelete kidolgozható, és mért adatainak kiértékelése hatékony.

8 Köszönetnyilvánítás

Hálával tartozom Sándor Tamásnak, amiért mindig rendelkezésre állt kérdéseim során, szakmai tudásával segítséget nyújtott a kutatásomban. Köszönettel tartozom konzulensemnek, az Óbudai Egyetem tanári karának, hogy megtanítottak a laborban történő precíz munkavégzésre, hasznos tanácsokkal láttottak el, és készségesen segítettek a tanulmányaimban, a méréseimben és a munka során felmerülő kérdéseimben. Továbbá szeretném megköszönni az Egyetem többi munkatársaknak, hogy hozzájuk is fordulhattam, amikor segítségre volt szükségem. Végül szeretném megköszönni a családomnak, és barátaimnak, hogy mindig számíthattam rájuk, és hogy kedves szavaikkal támogattak tanulmányaim során.

9 IRODALOMJEGYZÉK

1. **(MEKH), Magyar Energetikai és Közmű szabályozási Hivatal.** A MAGYAR VILLAMOSENERGIARENDSZER 2020. ÉVI ADATAI. *A MAGYAR VILLAMOSENERGIARENDSZER 2020. ÉVI ADATAI.* 2020. old.: 22.

2. *Electricity Metering and Monitoring in Manufacturing Systems.* **S. Kara, G. Bogdanski, W. Li.** 2011.

3. **na.** *Precise Timing for Power Industries and the Smart Grid.* [web] na : MEINBERG Funkuhren GmbH & Co. KG.

4. **Villamosenergia-mérés és -szabályozás. [Online] Magyar Szabványügyi Testület, 2021. [Hivatkozva: 2021. 12 12.] https://prod.mszt.hu/hu-hu/mszt-rol/iec-tc-13-titkarsag?fbclid=IwAR2OAEpnohdXyUe5oQKLpt9xCuypEqho31Sun27M_WoIc-xD4WBNdGtCc_k.**

5. *Design of an IoT Energy Monitoring System.* Komkrit Chooruang, Kraison Meekul. hely nélk. : Sixteenth International Conference on ICT and Knowledge Engineering, 2018.

6. Kuhlman, Dave. *A Python Book: Beginning Python, Advanced Python, and Python Exercises.* 2009.

7. *The Making of Python.* Venners, Bill. hely nélk. : Aritma, 2003.

8. Robinson, David. *The Incredible Growth of Python.* Internet : ismeretlen szerző, 2017.

9. *Is Python a good language for beginning programmers?* Foundation, Python Software. hely nélk. : Python FAQ, 2012.

10. Piotrowski, Przemyslaw. *Build a Rapid Web Development Environment for Python Server Pages and Oracle.* hely nélk. : Oracle Technology Network, 2006.

11. Batista, Facundo. *PEP 327 – Decimal Data Type.* hely nélk. : Python Software Foundation, 2003.

12. Eby, Phillip J. *Python Web Server Gateway Interface v1.0.* 2003.

13. Python JSON. ***W3Schools.*** **[Online] Refsnes Data. [Hivatkozva: 2021. december 7.] https://www.w3schools.com/python/python_json.asp.**

14. Programiz. ***Programiz.*** **[Online] Parewa Labs Pvt. Ltd. [Hivatkozva: 2021. december 7.] https://www.programiz.com/python-programming/json.**

15. ***The JavaScript Object Notation (JSON) Data Interchange Format.*** **T. Bray, Ed. https://datatracker.ietf.org/doc/html/rfc7159 : (IETF), Internet Engineering Task Force, 2014 Március.**

16. ***The JavaScript Object Notation (JSON).*** **Bray, T. 2014.**

17. ***The JSON data interchange syntax.*** **International, ECMA. https://www.ecma-international.org/publications-and-standards/standards/ecma-404/ : ismeretlen szerző, 2017 December.**

18. JSON. json.org. [Online] [Hivatkozva: 2021. december 8.] https://www.json.org/json-en.html.

19. JSON - Introduction. ***W3Schools.*** **[Online] Refsnes Data. [Hivatkozva: 2021. december 8.] https://www.w3schools.com/js/js_json_intro.asp.**

20. ***JSON Web Encryption (JWE).*** **M. Jones, J. Hildebrand. https://www.hjp.at/doc/rfc/rfc7516.html#sec_3 : Internet Engineering Task Force (IETF) , 2015.**

21. ***JSON Web Signature (JWS).*** **M. Jones, J. Bradley, N. Sakimura. https://www.hjp.at/doc/rfc/rfc7515.html#sec_3 : Internet Engineering Task Force (IETF), 2015.**

22. ***JSON Integration in Relational Database Systems.*** **Petković, Dušan. Rosenheim, Germany : International Journal of Computer Applications, 2017.**

23. Zhen Hua Liu, Beda Hammerschmidt, Doug McMahon. ***JSON Data Management – Supporting Schema-less Development in RDBMS.*** **Redwood Shores, CA 94065, USA : Oracle Corporation.**

24. (ISO), I.O. for Standardization. Information TechnologyDatabase Language SQL-Part 14: XML-Related Specifications (SQL/XML) .

25. *One Part Relational, One Part XML.* K. Beyer, R. Cochrane, V. Josifovski, J. Kleewein, G.Lapis, G.M.Lohman, R.Lyle, F. Özcan, H. Pirahesh, N. Seemann, T. C. Truong, B.V. Linden, B. Vickery, C. Zhang. hely nélk. : SIGMOD Conference, 2005.

26. *Indexing XML Data Stored in a Relational Database.* Pal, S, és mtsai. hely nélk. : VLDB, 2004.

27. *Towards an enterprise XML architecture.* R. Murthy, Z. H. Liu, M. Krishnaprasad, S. Chandrasekar, A. Tran, E. Sedlar, D. Florescu, S. Kotsovolos, N. Agarwal, V. Arora, V. Krishnamurthy. hely nélk. : SIGMOD Conference, 2005.

28. MongoDB. [Online] MongoDB. [Hivatkozva: 2021. 12 12.] https://www.mongodb.com/.

29. PyCharm. *JetBrains.* [Online] JetBrains s.r.o. [Hivatkozva: 2021. december 15.] https://www.jetbrains.com/pycharm/features/.

30. PyCharm. *JetBrains.* [Online] JetBrains s.r.o. [Hivatkozva: 2021. december 15.] https://www.jetbrains.com/pycharm/features/coding_assistance.html.

31. Plotly Graphing Libraries. *Plotly.* [Online] Plotly. [Hivatkozva: 2021. december 15.] https://plotly.com/python/line-charts/.

32. Altair. *Altair.* [Online] Altair Developers. [Hivatkozva: 2021. december 15.] https://altair-viz.github.io/getting_started/overview.html.

33. MySQL. *MySQL - What is MySQL?* [Online] Oracle Corporation. [Hivatkozva: 2021. december 15.] https://dev.mysql.com/doc/refman/8.0/en/what-is-mysql.html.

34. W3Schools. *W3Schools - Python MySQL.* [Online] Refsnes Data. [Hivatkozva: 2021 . december 15.] https://www.w3schools.com/python/python_mysql_getstarted.asp.

10 ÁBRAJEGYZÉK