ScadaSys Ipari Automatizálási kft.
1118 Budapest Holdvilág utca 6
tel:(361)212-7051,fax:(361)375-6643
www.scadasys.hu

PMSS

Prototípus fejlesztés Megvalósulási dokumentáció

Tartalom

Bevezetés

A PMSS egy összetett moduláris rendszer, mely energiaméréshez szükséges valamennyi hardver és szoftver elemet tartalmazza. A prototípus fejlesztés során az volt a célunk, hogy egy robosztus, megbízható, könnyen kezelhető adatgyűjtő és feldolgozó rendszert hozzunk létre. Villamos- víz- gáz- és hőenergia-fogyasztásmérőkkel való együttműködésre is ajánljuk. Szabványos adatbázisban tárolt adatokat képes feldolgozni, megjeleníteni és más rendszerek számára elérhetővé tenni.

A PMSS egy felhő alapú megoldás, amely a mérők adatait távolról, on-line módon tudja megmutatni. Képes a számlázáshoz szükséges dokumentumok előállítására, illetve más személyre szabható, egyedi riportok előkészítésére.

Alapvető funkciói:

- fogyasztási adatok gyűjtése
- adattovábbítás felhőn keresztül a felhasználóhoz
- adatfeldolgozási és megjelenítő funkciók
- rendszerintegráció
- speciális igény esetén, egyedi szoftverfejlesztés

Villamos fogyasztásmérés esetén:

- hatásos- és meddő villamos energia fogyasztás mérése
- feszültség-, áram- és teljesítménymérés
- teljesítménytényező (cos fi) mérés
- villamos minőségi jellemzők (felharmonikus tartalom) mérése
- online monitorozás
- mért és számított adatok tárolása, archiválása
- események generálása, naplózása (pl. feszültség kimaradások, határérték túllépések)
- diagramok, kimutatások készítése a mért értékek alapján (napi, havi, éves kimutatások)
- SMS riasztások küldése, megadott telefonszámokra

Energiafogyasztás csökkentés

- a pazarlások felismerése (hálózati és fogyasztói)
- döntés előkészítés, valós fogyasztási adatok alapján
- azonnali ellenőrzési és beavatkozási lehetőségek támogatása

Beruházási döntéstámogatás

- beruházási döntések támogatása a valós fogyasztási adatok alapján
- beruházások elő- és utómonitorozása, megtérülési adatok számítása
- a valós fogyasztási profil/szerkezet elkészítése

Üzemeltetési támogatás

- lekötött kapacitás kihasználtságának feltárása
- határérték-átlépések elkerülése
- alhálózati/egyedi fogyasztói felhasználások mérése
- költségek megosztása a bérlők illetve szervezeti egységek között
- szabályos számlakép generálása

Kommunikáció

- másodperces felbontású időbélyeg hozzárendelése az adatokhoz
- Ethernetes vezetékes vagy WIFI, esetleg GPRS-s kapcsolat
- titkosított, hitesített kommunikációs protokoll

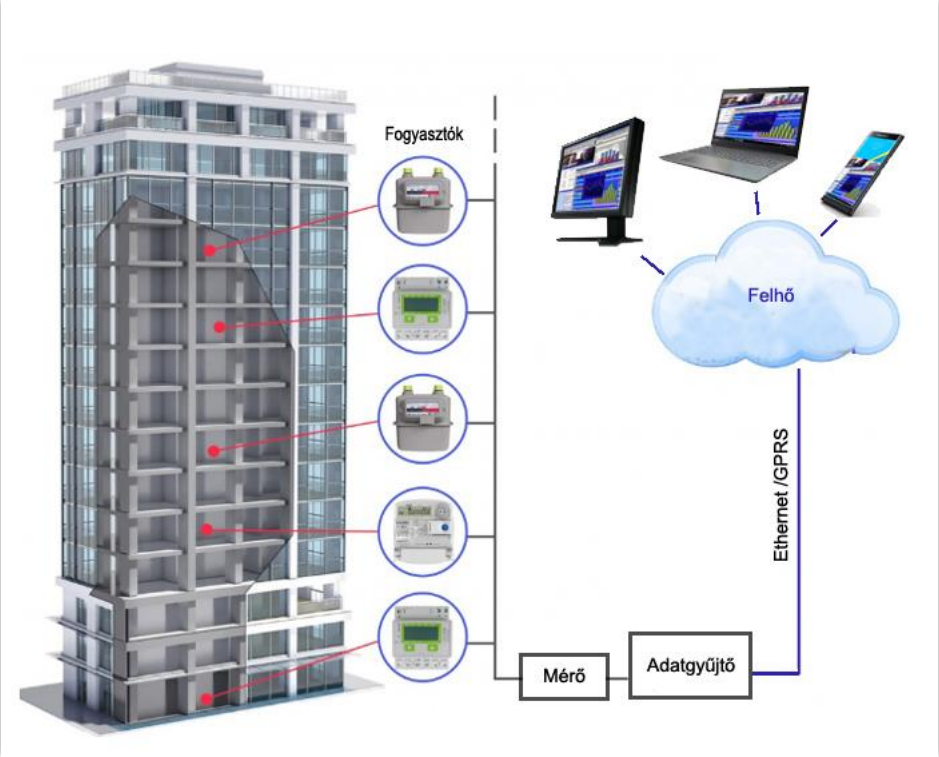
Struktúra

A prototípusfejlesztést úgy tervezzük, hogy az eredményt, a lehető legáltalánosabban lehessen alkalmazni. Nem teszünk megkötést sem az alkalmazási területre (ipar, mezőgazdaság, szolgáltatás, épületfelügyelet ...), sem a mérendő mennyiségek fajtájára (folyadék, gáz, villamos stb.). Fejlesztés során a vonatkozó szabványok betartásával a legszélesebb felhasználói kört tudjuk elérni. A korszerű IT technológiákat felhasználva egy rugalmasan, ugyanakkor biztonságosan működő rendszert alakítunk ki. Az alaprendszer a következő elemekből áll:

- Fogyasztásmérők (teljesítménymérők). Ezek robosztus kialakítású nagy megbízhatóságú eszközök.
- Ipari lokális hálózat, amely felfűzi a mérőket és illeszti az adatgyűjtőhöz. (pl. MODBUS)
- Adatgyűjtő (ipari PC). Ez kommunikál a mérőkkel, gyűjti, és tárolja a mért és számított adatokat. Ebben az eszközben futnak a rendszer algoritmusai. Ez a gép továbbítja az adatokat a felhasználók felé. A felsőszintű kommunikáció lehet Ethernet alapú (kábel, Wi-Fi), vagy lehet GPRS.
- A felhasználók saját számítógépükön tekinthetik meg az adatgyűjtőben tárolt on-line és archív adatokat. (Asztali PC, Notebook, okos telefon)

Felhőalapú

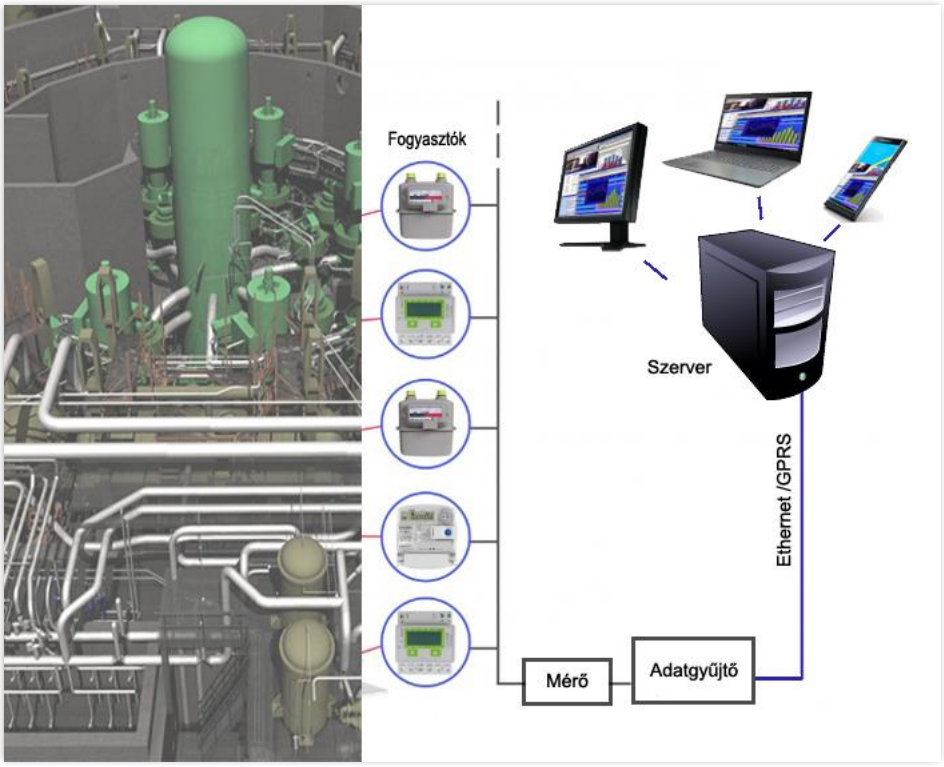
Azokban az alkalmazásokban, ahol a fogyasztásmérés lokális, de az adathozzáférés globális, a felhőalapú megoldás látszik megfelelőnek (Ilyenek lehetnek például irodaházak, lakóparkok, közüzemi szolgáltatók, mezőgazdasági üzemek). A mérőeszköz lokális hálózaton keresztül juttatja az adatokat a felhőbe. A kommunikáció ezen a szinten már nem speciális protokollon keresztül, hanem *adatbázis-adatbázis* kapcsolaton keresztül történik. Az adatokat kódolva kerülnek a kommunikációs csatornába, és a felhőben is a kódoltan tárolódnak. Ennek megfelelően a felhasználók is kódolt adatokat tudnak elérni, amit csak a végfelhasználó gépe dekódol. Ezzel a módszerrel garantálható a magas fokú adatbiztonság. (Ha bárhol is illetéktelen kezekbe kerül az adat, az használhatatlan lesz.)



1. ábra Felhő alapú struktúra

Szerver alapú

Ezt a struktúrát azokban az alkalmazásokban lehet használni, ahol a fogyasztásmérés és a felhasználás is lokális. (Ilyen lehetnek erőművek, gyárak vagy termelő üzemek.) A mérőeszköz lokális hálózaton keresztül juttatja a szerver gépbe. A kommunikáció ezen a szinten már nem speciális protokollon keresztül, hanem *adatbázis-adatbázis* kapcsolaton keresztül történik. Ha az átviteli hálózat nyitott a külvilág (internet) felé akkor az adatokat kódoljuk, ha nem akkor akár kódolatlanul kerülnek a kommunikációs csatornába és a szerver adatbázisába. A lokális felhasználók a szerverből tudják lekérdezni és megjeleníteni a szükséges információkat.



2. ábra Szerver alapú struktúra

Eszközök

Folyadék és gázmennyiség mérők

A legtöbb ilyen készülék illeszthető a PMSS-hez, hiszen rendelkeznek belső adattároló kapacitással, és szabványos kommunikációs felülettel. Például alkalmasnak tartjuk a Uniflow-200-as készüléket. A Uniflow-200 egyidejűleg 8 áramló mennyiség mérésére alkalmas, vagyis egyidejűleg akár 8 különböző csőben áramló anyag nyomás és hőmérséklet kompenzált mennyiségének meghatározását végzi. Legtöbb estben a mennyiségjelen kívül, a mérőkörhöz kompenzáló jelek is tartozhatnak (pl. nyomásjel, hőmérsékletjel, sűrűségjel, v. gázkromatográf jelek). Nagy pontossággal méri az áramló anyagok mennyiségét:

- referencia körülmények között:+/- 0.05 %.
- 0 - 50 °C környezeti hőmérséklet tartományban: +/- 0.1 %.

Működési hőmérséklet tartománya -10 és 50 °C között van.

Az áramlásmérő lehet: mérőperem, mérőtorok, Venturi-cső, Annubar, V-cone, mérőturbina, Vortex-mérő, forgódugattyús mérő, ultrahangos mérő, Coriolis mérő.

Ezzel a készülékkel a leggyakrabban előforduló folyékony és gáznemű anyag mennyiségét meg tudjuk mérni. (földgáz, kamragáz, kohógáz, kőolaj, kőolajtermékek, kenőolaj, gázkondenzátum, levegő, nitrogén, oxigén, hidrogén, argon, szén-dioxid, szén-monoxid, etilén, ammónia, propán, víz, vízgőz, víz-glikol keverék)

Rendelkezik soros RS485 felülettel, és MODBUS (RTU, ACII) kommunikációs protokollal, így gond nélkül illeszthető a rendszerünkhöz.

Teljesítménymérők és hálózat analizátorok

Fontos feladat az elektromos hálózatok paramétereinek mérése a hálózatok karbantartásánál, a hibakeresésnél, és különböző készülékek, kapcsolók, védelmi rendszerek határérték szerinti működtetésénél. A teljesítménymérés és hálózati-/teljesítményanalízis adatai felvilágosítással szolgálnak a terhelések állapotáról, a pillanatnyi teljesítményéről, valamint a hálózati jelalak terhelések okozta torzításiról.

Manapság egyre szaporodnak a nemlineáris fogyasztók. Ilyenek a korszerű világítástechnikai berendezések, a fényerő szabályozós lámpák, a nagyteljesítményű gépekhez- légtechnikai berendezésekhez tartozó villamos hajtások, légkondicionálók, valamint a kapcsolóüzemű tápegységeket. Ezek a berendezések többnyire nem szinuszos áramot vesznek fel a hálózatból, melynek nagy a felharmonikus tartalma.

A felharmonikusok káros hatásai: □□

- Védelmi berendezések véletlenszerű működése (pl. megszakító téves kioldása)
- Többletveszteségek a vezetékeken és a transzformátoron □□
- Berendezések fokozott öregedése □□
- Felharmonikusok ra érzékeny berendezések meghibásodása □□
- Fogyasztásmérők hibás működése

Csak mérések alapján derül ki hogy az egy rendszere mennyire korszerű és tényleg ellátja-e a pénzmegtakarító funkciót amire szánták vagy sem.

Előzetes hálózati analízis során állapítható meg, hogy milyen meddő kompenzálást vagy felharmonikus szűrűket kell bekötni a hálózatba. Az energiaminőség javítása és az üzembiztos villamos energia ellátás elengedhetetlen minden beruházásnál, hiszen egy váratlan feszültség kimaradás, vagy egyéb villamos hálózat okozta probléma komoly károkat tud okozni a munkában.

A nagy harmonikus tartalomhoz tipikusan társított káros hatások a világítástechnikai berendezések és tápegységek átlag fölötti meghibásodási rátájához és idő előtti öregedéséhez, valamint a villamos infrastruktúra elemeinek, különösen a nullavezetőnek a jelentős melegedését okozták.

A probléma kezelésének legjobb egy pontosan tervezett aktív elhelyezett aktív felharmonikus szűrő felszerelése, mely képes:

• harmonikus kompenzációra

• meddőteljesítmény kompenzációra

• terhelés szimmetrizálásra

Több gyártó is kínál a villamos hálózatok teljesítményanalízisére, hálózati paraméterek mérésére alkalmas műszereket.

A készülék feszültségbemenete közvetlenül vagy feszültségváltón keresztül csatlakozhat a mérendő hálózatra. Lehet közvetlen, vagy áramváltós árambemenettel is csatlakoztatni a mérendő hálózathoz. Ha az árammméréshez áramváltót használunk, az áramváltó áttétele a készülékeken beállítható.

A készülékek általában a következő értékeket mérik: feszültség, áram, hatásos, meddő és látszólagos teljesítmény, hatásos és meddő energia, teljesítménytényezők, frekvencia, felharmonikus tartalom (THD). A harmonikustartalom összetevőnként és a fázisfeszültségekre/áramokra külön-külön mérhető.

A legtöbb készülék rendelkezik RS-485 (MODBUS) interfésszel is, amely keresztül felfűzhetőek és lehetővé teszik a mért értékek további adatfeldolgozását.

A készülékek alkalmasak egyfázisú kétvezetékes, és háromfázisú három- és négyvezetékes, szimmetrikusan és aszimmetrikusan terhelt hálózatok paramétereinek mérésére, beleértve az áram és feszültség harmonikusokat is az 51. összetevőig.

A készülékekkel a feszültség, az áram, a hatásos/meddő- és látszólagos teljesítmény, a fázisszög mellett az energiafogyasztás is mérhető. A mért paraméterek maximum- és minimumértékei is tárolásra kerülnek.

Könyvtár szerkezet

Kliens gép (adatgyűjtő)

A főkönyvtár neve tetszőleges lehet, célszerű a projekt nevének valamilyen egyszerű formáját választani**.** Az adatgyűjtő gépen kötelező létrehozni egy **#Kliens** nevű alkönyvtárat, ezen belül pedig egy **Kliens_Data** nevű könyvtárat.

Szerver gép (megjelenítő gép)

A főkönyvtár neve itt is tetszőleges lehet. A megjelenítő gépen kötelező létrehozni egy **#Server** nevű alkönyvtárat, ezen belül pedig egy **Server_Data**, és egy **Pict** nevű könyvtárat is. A **Server_Data** könyvtáron belül létre kell hozni egy **Pillanat** nevű alkönyvtárat, ebbe kerülnek a mért és számított pillanatértékek.

Megjegyzés: Ki lehet alakítani egy gépes rendszert is, ebben az esetben ugyanazon a gépen fut az adatgyűjtés és a megjelenítés. (Ezt leginkább tesztelési célból szokás megtenni.) Ekkor értelem szerűen egy főkönyvtárban nyitjuk meg a #Kliens és a #Server könyvtárat is)

Kommunikáció

MODBUS

Az internet ipari, távmérési alkalmazásainak térhódításával jelent meg az az igény, hogy a mért paraméterek az interneten keresztül is hozzáférhetők legyenek. Ez különösen távfelügyeleti rendszerek esetében nagy előny, mivel alacsony költség mellett teszi lehetővé a mért értékek távoli kiolvasását, szükség esetén a készülék átprogramozását, az adatgyűjtést és archiválást.

A PMSS rendszer az adatgyűjtők illesztésére a MODBUS kommunikációs protokollt használja. Ez egy soros protokoll, melyet alapvetően programozható logikai vezérlők számára fejlesztettek ki. Jellemzői:

- Egyszerű és robusztus.
- Nyilvános, jogdíjmentes
- Könnyű telepíteni, karbantartani
- Master/Slave adatlekérdezés (Master olvassa és írja a Slave adatait)
- Egy rendszerben egy Master, de sok Slave lehet
- RS-232: két pont kapcsolat, maximum 15 méteres távolságon belül.
- RS-485: Multi pont kapcsolat, maximum 247 eszköz, 1200 méteren belül.

A PMSS értelem szerűen az RS-485-ös felületet használja a kommunikációs célra.

Lokális hálózat

Az adatgyűjtő Ethernet lokális hálózatot használ a mért adatok továbbítására. Ma ez a legelterjedtebb és legjobban támogatott protokoll. Jellemzője az ütközések detektálása és menedzselése (CSMA/CD), valamint a réteges felépítés (OSI). A pontos protokoll leírást az IEEE 802.3 szabvány rögzíti.

A PMSS prototípus adatgyűjtője rendelkezik Ethernet port-al, így server alapú topológia esetén közvetlenül összeköttetést lehet létrehozni, egy szabványos Ethernet kábellel.

Wifi

Ez egy vezeték nélküli adatátviteli protokoll, mely az OSI modell két alsó rétegét, a fizikai adatkapcsolati réteget definiálja. A pontos protokoll leírást az IEEE 802.11 szabvány rögzíti.

A PMSS prototípusba a GEMBIRD nagy teljesítményű USB WiFi adaptert építettük be. Az eszköz átviteli sebessége 300 megabit/sec, ami elegendő a megkívánt adatforgalomhoz.

SMS/GPRS

A prototípushoz tartozik egy SMS küldő berendezés is, mely képes küldeni és fogadni rövid szöveges üzeneteket, eseményeket. A készülék típusa:

Szimulátor

Az adatgyűjtőkkel való kommunikáció működésének ellenőrzéséhez szimulátor programokat készítettünk. Ezek helyettesítik a technológiát, biztosítják a teljes rendszer tesztelését abban az esetben, ha nem áll rendelkezésre a technológia, vagy a technológia-illesztés. A szimulációs modulnak nincs saját konfigurációs fájlja, bemenetként az *adatbázis leírót* használja. Az ott specifikált jeleket állítja elő és beteszi a *Kliens_Data* adatbázisba. A program indítása után ciklikusan generálja a *db_leiro.csv*-ben definiált jeleket, a *méréstartomány* figyelembevételével és beírja a kliens *real-time* és *archív* adatbázisába. A ciklusidő default értéke 20 másodperc. A *Symulator.py* modul csak azokat a jeleket szimulálja, amelyeknek az adatbázis leíróban a *típus* és *Cim* oszlopában szerepel valami. Ezzel azt tudjuk elérni, hogy a szimuláció során az adatgyűjtéssel azonos jelek kapjanak értéket. Így tesztelni tudjuk a feldolgozó modulok működését. Értelem szerűen, ne futtassuk egy időben szimulátor és adatgyűjtő programot, mivel mindkét programnak ugyanaz a kimenete, ezért összekeverednének a mért és szimulált jelek.

Ha azt szeretnénk, hogy az adatbázis leíróban szereplő valamennyi jelet generálja a rendszer, akkor a *SymulatorAll.py* programot kell elindítani. Ezt akkor használjuk, ha nem akarjuk a feldolgozó modulokat tesztelni, viszont pl. a megjelenítőben látni akarjuk valamennyi adtabázis jelet. A kliens könyvtár szerkezete megegyezik az előzőekben látottakkal, csak több jelet tartalmaz.

Adatbázis

Az adatok mérése és tárolása ciklikusan történik. Az adatgyűjtő a mért adatokat skálázza és a megadott formátum szerint elhelyezi az adatbázisban. Az adatok leírása egy konfigurációs táblában adható meg. Ebben az esetben is arra törekedtünk, hogy a lehető legegyszerűbb és legáltalánosabb módon lehessen megfogalmazni, ezért a konfigurációs állomány egy 'csv' (comma-separated values) fájl, amit Excel-el vagy tetszőleges szövegszerkesztővel elő lehet állítani. Az adatbázis leíró fájl neve *DB_leiro.csv*.

A leíróban szereplő adatok az oszlopok sorrendjében a következők:

- tag_name (az adatbázis jel azonosítója)
  Hossza tetszőleges, de célszerűségi okokból korlátozni kell, viszont egy adott rendszerben egyedinek kell lennie. Az adatfeldolgozás egyszerűbbé tétele érdekében ajánlott olyan jelneveket használni, melyekre könnyű ciklust szervezni.(Lásd számítások fejezet.)

- DF (adat formátum)

  A mért és a megjelenített adat formátumának leírását tartalmazza. A „#" karakterek a számjegyeket reprezentálják, számuk meghatározza az egész és tizedes jegyeket. (A '.' előtti karakterek az egészeket, az utániak a tizedes jegyeket jelölik. Ha például egy adatot 3 egész és 2 tizedes pontossággal akarunk ábrázolni, akkor az adat formátuma: ###.##. Az esetlegesen előforduló negatív előjel nem számít bele az egészek számába)
  Az adatgyűjtő ipari PC és a megjelenítő állomások között adatbázis-adatbázis kapcsolat van (nincs protokoll, csak jel titkosítás) ezért az adatok az itt definált formátumban közlekednek a felhőbe és vissza.

- Merestart. (Méréstartomány)

  A mért, vagy számított jelek méréstartományát kell ebben az oszlopban megadni. A méréstartomány is fizikai jeltartományban értendő. Az alsó és felső méréshatárt '-' jellel kell elválasztani. (pl. egy 0,4 kV-os feszültség mérés méréstartománya: 0-400)

- Unit (Mértékegység)

  Az adott jel mértékegységét lehet itt megadni. A mértékegység tetszőleges szöveg lehet és csak a megjelenítéskor van funkciója. (Ha olyan jelet használunk, amelynek nincs mértékegysége pl. cosfi, akkor a csv fájlba '-' karaktert kell tenni)

- Típus

  Az adatgyűjtő eszközben tárolt adat formátuma. Ennek megadása a mért adatok skálázásához szükséges. Definiált adat típusok:

  - byte (8 bit)
  - unit (8 bit)
  - word (16 bit)
  - float (4 byte lebegőpontos)

- Cim

  Az adat címe két részből áll melyeket '/' jel választ el. A perjel előtti szám a felfűzött adatgyűjtő eszköz azonosítására szolgál. Az adott adatgyűjtőn belüli adat címzése található hexadecimális formátumban a '/' jel után. (pl. a 2-es számú készülékből szeretnénk az R fázis feszültséget felkérni, akkor a cím: 2/100)

Példaként lássunk egy 6 jelből álló konfiguráció leíró részletet:

tag_name	DF	Merestart.	unit	tipus	Cim
FFF_M001_R	###	0-250	V	float	1/100
FFF_M001_S	###	0-250	V	float	1/102
FFF_M001_T	###	0-250	V	float	1/104
PPP_M001	#.##	0-5	kW	float	1/144
PPM_M001	#.##	0-5	kVA	float	1/14C
CCC_M001	####.#	0-9999	kWh	float	1/15F

Adatgyűjtés

Az adatgyűjtést a *mero.py* mérőmodul végzi. Ciklikusan lekérdezi a felfűzött mérőeszközök jelei közül azokat, amelyeket az adatbázis leíróban (DB_leiro.csv) definiáltunk. Az alap ciklusidő 20 másodperc. A mért pillanatértékeket eltárolja a kliens gép adatbázisában. A pillanatértékeket a *Put_data.py* program továbbítja a felhőbe (vagy a szerverbe). A mérőmodul a napi archívumokat is folyamatosan tölti.

Kliens

Az adatok a *Kliens_Data* könyvtárban találhatóak. A pillanatértékek az *Aktual.csv* fájlban találhatóak. Ez a fájl minden mérési ciklus után frissül a pillanatértékekkel, és addig található ebben a könyvtárban, amíg a kommunikációs modul nem továbbítja azt. (A fájl jelenléte szemaforként is szolgál.)

A pillanatértékeket tartalmazó fájl szerkezete:

(A fájl első sora különbözik a többitől, itt az első oszlopban a dátum, a második oszlopban az időbélyeg található)

- Első oszlop: tag_name (az adatbázis jel azonosítója)
- Második oszlop: aktuális érték (lebegőpontos szám, string formátumban, tizedespont)

2018.02.09	15:33:20
FFF_M001_R	214.0
FFF_M001_S	185.0
FFF_M001_T	228.0
PPP_M001	3.79
PPM_M001	4.72
CCC_M001	1824998
FFF_M002_R	243.0
FFF_M002_S	201.0
FFF_M002_T	250.0
PPP_M002	43161
PPM_M002	2.86
CCC_M002	2954174

Az archív adatok napi bontásban kerülnek tárolásra. Minden nap új könyvtár keletkezik, melynek nevében megtalálható a dátum (pl. Napi_2018.02.09). A könyvtáron belül, annyi fájl szerepel ahány mérést az adatbázis leíróban definiáltunk. A fájlok a jel neveit tartalmazzák (pl. FFF_M001.csv). Az archív fájlok szerkezet rendkívül egyszerű:

- Első oszlop: időbélyeg (óra:perc:másodperc)
- Második oszlop: az időbélyeghez tartozó érték (lebegőpontos szám, string formátumban)

13:05:00	210.0
13:05:20	194.0
13:05:40	183.0
13:06:00	232.0
13:06:20	223.0
13:08:20	223.0
13:08:40	213.0
13:33:00	219.0
13:33:20	246.0
13:33:40	219.0
14:59:20	224.0
14:59:40	226.0
15:00:00	183.0
15:00:20	210.0
15:00:40	186.0

A napi archívumokat egy évig tárolja a kliens. Ha a szerverből hozzá akarunk jutni egy korábbi adathoz, akkor használjuk az Archiválás fejezetben ismertetett módszert.

Szerver

Az adatok a *Server_Data* könyvtárban találhatóak. Ezen belül a *Pillanat* alkönyvtárba kerülnek a pillanatértékek (real-time adatok), és a *Napi_####.##.##* alkönyvtárakba az archívumok.

A **pillanatértékek** annyi *#####.csv* fájlban tárolódnak ahány jelet az adatbázis leíróban definiáltunk. A fájlok nevei megegyeznek az ott szereplő egyedi jelazonosítókkal (*tag_name*). Minden egyes fájlban az időbélyeggel ellátott utolsó mért érték található.

Az **archívumok** könyvtárban annyi *#####.csv* fájlban tárolódnak ahány jelet az adatbázis leíróban definiáltunk. A fájlok nevei megegyeznek az ott szereplő egyedi jelazonosítókkal (*tag_name*). Minden fájlban az időbélyeggel ellátott mért érték találhatóak, soronként egy-egy adat. A fájlok maximum egy napi adatot tartalmaznak, éjfél után automatikusan új archív könyvtár és ebben új adatfájlok keletkeznek.

Az archív fájlok szerkezet megegyezik a kliens archívumoknál bemutatottal:

- Első oszlop: időbélyeg (óra:perc:másodperc)
- Második oszlop: az időbélyeghez tartozó érték (lebegőpontos szám, string formátumban)

Adatvédelem

Az egyes modulok közötti adatkapcsolat megbízhatósága alapvetően befolyásolja a rendszerbiztonságot. Nem fordulhat elő, hogy egy modul hibás működése, vagy megállása, megzavarja a többi modul működését.

A leggyakoribb félreértés a belső ipari hálózatokat illetően az, hogy azok a külső hálózatoktól függetlenek (elszigeteltek). Valóságban – a McAffe report szerint- az ipari rendszerek több mint kétharmada kapcsolódik az Internethez, vagy valamilyen hasonló IP hálózathoz, és bevallottan több mint felénél megoldatlan az adatbiztonság biztosítása. 2001- óta, a regisztrált esetek 70%-ban az ipari hálózatokon kívülről származtak a támadások.

Míg az üzleti hálózatok jól ellátottak adatbiztonság kezelése szempontjából, a külső támadások kivédésére, felhasználva a legkorszerűbb biztonságtechnikai eszközöket, addig a legtöbb ipari rendszer szinte védtelen a leggyakoribb támadásokkal szemben.

A közismert módszereken túl (tűzfal, jelszavas védelem, stb.) mindenképen meg kell akadályozni, a technológiába történő beavatkozás lehetőségét, ezért semmilyen körülmények között nem megengedett, a Web felületen történő parancsadás, vagy paraméterállítás.

Fenyegetettség (veszély)	A támadás hatása
A rendszer vagy felhasználó adatainak módosítása	Adatok integritásának elvesztése, másodlagos hatás lehet az adatok elérhetőségének elvesztése.
Tárolt adatok megváltoztatása/törlése. Üzenet tartalmának módosítása. Kontrol jelek megváltoztatása Paraméterek, alapjelek megváltoztatása	Adatvesztés
Kijelző értékeinek megváltoztatása. A kezelőnek mutatott hibás információból származó mellékhatások.	Adatok/üzenetek integritásának elvesztése
Vezérlő üzenetek „elszipkázása”	Bizalomvesztés

Kommunikáció elterelése, blokkolása	Hozzáférés elvesztése
Eszközök leállítása	Hozzáférés elvesztése
Rosszindulatú kód (vírus)	A romboló szándékától függő bármilyen szétesést okozhat

Az információ biztonság fokozásának fő területei:

- *A hálózat védelme, biztonságosabbá tétele*
- *Riasztás illetéktelen behatolás esetén*
- *Az illetéktelen behatolás megakadályozása*
- *Kritikus adatok titkosítása a hálózati kommunikáció során*

A kommunikációs hálózat biztonság növelésének lehetséges eszközei:

- *Tűzfalak, UTM (Unified Threat Management) eszközök*
- *Vírusszűrő, e-mail biztonsági rendszerek*
- *Webes alkalmazások védelme*
- *Hálózati forgalom titkosítása, hálózati végpont védelem*
- *Wi-Fi hálózatok biztonsági megoldásai*

Az információ biztonság fenntartásának eszközei:

- *Az információbiztonsági rendszer folyamatos fenntartása, fejlesztése*
- *Az irányítási rendszerben feltárt kockázatok folyamatos kezelése, a védelmi rendszer fejlesztése és frissítése*
- *Az információbiztonsági tudatosság folyamatos növelése*
- *Információbiztonsági szabályozás*
- *Folyamatmenedzsment*
- *Felhasználó- és jogosultságkezelés folyamatos megújítása*

Adat titkosítás

Az adatok tikosítása úgy valósul meg, hogy a közvetlenül a mérőeszköz kódolja az adatokat, és kódolva küldi át az adatátviteli csatornákon, és így is kerül tárolásra az adatbázisban. Lekérdezéskor, szintén titkosított adatok mozognak, és csak az adatfeldolgozás és megjelenítés során dekódolja azt a rendszer.

Így biztosítható, hogy ha az adatút bármely elemén illetéktelenek kezébe kerülnek az információk, akkor sem tudják azt felhasználni.

A titkosítást AES-módszerrel végezzük. (Advanced Encryption Standard). Az AES egy szimmetrikus titkosítási kulcsot használó kódolás olyan változata, ahol a blokkméret szigorúan 128 bites, 4x4-es mátrixot használ, és a titkosítási ciklusok

száma 10. Minden ciklus több lépést foglal magába, ezek között van az a lépés is, ami kulcs alapján módosítja a mátrixot. A visszaalakítás során ugyanennyi ellentétes ciklust hajtanak végre a kulcs segítségével.

Az AES eljárás

1. A tényleges kulcsok előállítása a nyers kulcsból a Rijndael-féle módszerrel.
2. Előkészítés
    1. ***AddRoundKey*** mátrix minden bájtját bitenkénti XOR művelettel módosítják a tényleges kulcs segítségével.
3. Ciklusonként ismétlődő lépések.
    1. ***SubBytes*** - egy nemlineáris helyettesítési kódolás a Rijndael S-box szerint.
    2. ***ShiftRows*** - egy keverési lépés ahol a sorokat körkörösen (tehát a sor végi elemek megjelennek a sor elején) eltolják egy meghatározott mértékkel.
    3. ***MixColumns*** - egy lépés ahol az oszlopok mind a négy bájtját kombinálják.
    4. AddRoundKey
4. Utolsó ciklus (nincs MixColumns lépés)
    1. SubBytes
    2. ShiftRows
    3. AddRoundKey

Sub Bytes lépés

A SubBytes szakaszban, minden bájtot lecserélnek egy helyettesítési kódolással, S; $b_{ij} = S(a_{ij})$.

Ebben a lépésben a mátrixban minden bájtot lecserélnek egy 8 bites helyettesítési tábla, a Rijndael S-box segítségével. Ez a művelet biztosítja, hogy az adat nemlineáris lesz a kódolási szakaszban. Hogy az egyszerű algebrai tulajdonságokra támaszkodó támadásokat kiküszöböljék, az S-boxot egy Galois test(2^8) multiplikatív inverzének és egy affin transzformáció segítségével készítik.

Shift Rows lépés

A ShiftRows a sorok elforgatását jelenti. Ennek során a mátrix sorait balra forgatjuk a sorok számával arányos lépéssel.

Minden sort egy meghatározott, de soronként különböző mértékben eltolják. Az AES-ben az első sor változatlan marad ebben a lépésben. A második sor minden bájtját egyszeresen toljuk balra. Hasonlóan, a harmadik és a negyedik sort 2 illetve 3 bájttal tolódik balra. 128 és 192 bites blokkokban az eltolás megegyezik. Az n. sort n-1 bájttal toljuk el. Így minden oszlop a ShiftRows lépés után csak azokból a bájtokból áll, amiket a transzformáció előtt is tartalmazott az adott sor. 256 bites blokkméret esetében az első sor változatlan marad, míg a 2., 3., 4. esetében az eltolás 1 bájt, 3 bájt és 4 bájt.

Mix Columns lépés

A MixColumns az oszlopok összekeverését jelenti. Ennek során a mátrix oszlopait külön-külön helyettesítjük, úgy hogy a helyettesített bájt a keverendő oszlop minden elemének függvénye. A művelet leírható a következő mátrix szorzással:

$$\begin{bmatrix} 02 & 03 & 01 & 01 \\ 01 & 02 & 03 & 01 \\ 01 & 01 & 02 & 03 \\ 03 & 01 & 01 & 02 \end{bmatrix} \begin{bmatrix} s_{0,0} & s_{0,1} & s_{0,2} & s_{0,3} \\ s_{1,0} & s_{1,1} & s_{1,2} & s_{1,3} \\ s_{2,0} & s_{2,1} & s_{2,2} & s_{2,3} \\ s_{3,0} & s_{3,1} & s_{3,2} & s_{3,3} \end{bmatrix} = \begin{bmatrix} s'_{0,0} & s'_{0,1} & s'_{0,2} & s'_{0,3} \\ s'_{1,0} & s'_{1,1} & s'_{1,2} & s'_{1,3} \\ s'_{2,0} & s'_{2,1} & s'_{2,2} & s'_{2,3} \\ s'_{3,0} & s'_{3,1} & s'_{3,2} & s'_{3,3} \end{bmatrix}$$

Az 1-gyel való szorzás nem változtat, a 2-vel való szorzást úgy értelmezzük, mint eltolást balra, a 3-mal való szorzás eltolás balra, majd kizáró vagy művelet az eredeti el nem tolt értékkel. Az eltolás után, ha a keletkezett érték 0xFF-nél nagyobb, akkor még egy XOR-t kell elvégezni az eredménnyel a 0x11B értékkel.

Általánosabban: minden oszlopot úgy kezelünk mint egy polinomot GF(28) felett és aztán ezt szorozzuk meg egy meghatározott polinommal c(x) = 0x03 · x3 + x2 + x + 0x02 modulus x4+1-gyel. Az együtthatókat hexadecimális formában használjuk. A MixColumns lépést úgy is tekinthetjük, mint szorzást egy megfelelő MDS mátrix-al egy véges testen.

Add Round Key lépés

Az AddRoundKey a körkulcs hozzá adását jelenti. Ennek során a mártix elemeit kizáró vagy kapcsolatba hozza az előre generált körkulcs elemeivel. Mivel az eddig tárgyalt műveleteket több ciklusban hajtja végre egymás után a program, ezért 10 ciklus esetén 40 szóból áll a körkulcs. A kulcs előállításának kritériumai: a transzformáció invertálható legyen, ne legyen szimmetrikus, és lineáris. A kulcs egyes részeinek ismerete ne segítse a többi meghatározását.

Az AES kód megfejtése (dekódolása)

A fenti négy lépésnek van inverz lépése, ezeket fordított sorrendben kell alkalmazni a kódon.

Példa a kódolásra

Létrehoztunk egy szövegfájlt melynek a neve **Próba.dat**. Titkosítsuk a fájl levét a fent ismertetett módszerrel.

A kapott eredmény: 'ef5f760d05292753eccd80a6bfc0bd13'

Adatáramlás

A kliens gépen összegyűjtött adatokat, el kell juttatni a valamelyik szerver gépe, ahol további feldolgozást, illetve megjelenítést tudunk elvégezni.

Amint a **Struktúra** fejezetben láttuk két esetet kell megkülönböztetni. A **felhőalapú** megoldásnál a Put.py program figyeli az adatváltozást és ciklikusan eljuttatja a „felhőbe". Az adatokat és eseményeket kódolja, és a kódolt fájlokat FTP protokollal küldi a felhőben lévő FTP szerver adatbázisába. Az FTP szerveren mindig a legutolsó mérések és események találhatóak. Ha fut az adatgyűjtő program és a Put.py program a kliensen, akkor az FTP szerver adatbázisában mindig a legfrissebb adatok és események találhatóak, ha nem akkor az utoljára felküldött és el nem olvasottak vannak ott.

A Put.py program alapfunkciói: adatváltozás figyelés, eseményváltozás figyelés, kódolás, FTP fájlküldés, kérésre archívum felküldés, kérésre esemény felküldés, időszinkron.

Ha bárhol a világban hozzá akarunk jutni a mérési eredményekhez, és rendelkezésünkre áll a PMSS program, akkor csak el kell indítani a **Get_data_from_FTP.py** programot, és az lekérdezi az adatokat a felhőből és beírja a lekérdező gép adatbázisába. (Természetesen a lekérdező gépnek rendelkeznie kell internet kapcsolattal.)

A Get program alapfunkciói: adatváltozás figyelés, eseményváltozás figyelés, FTP fájlolvasás, dekódolás, adatbázisba írás, időszinkron küldés a kliens gépnek.

A Put és a Get programokat paraméterezni nem kell, és ciklikusan kérdezik az adatokat, a default ciklusidő 20 másodperc. A lekérdezés megtörténtét a konzolon egy-egy sor kiírásával jelzi.

Szerver alapú, közvetlen adatlekérdezés

Ha adattak a feltételek, akkor megoldható a közvetlen adatkinyerés is a kliensből. Ezt a megoldást csak ott ajánljuk, ahol internet független, helyi hálózaton keresztül el tudjuk érni az adatgyűjtő gépet. Ebben az esetben nincs szükség adat titkosításra, sőt adatküldő programra sem. A kliens gépen ki kell ajánlani a

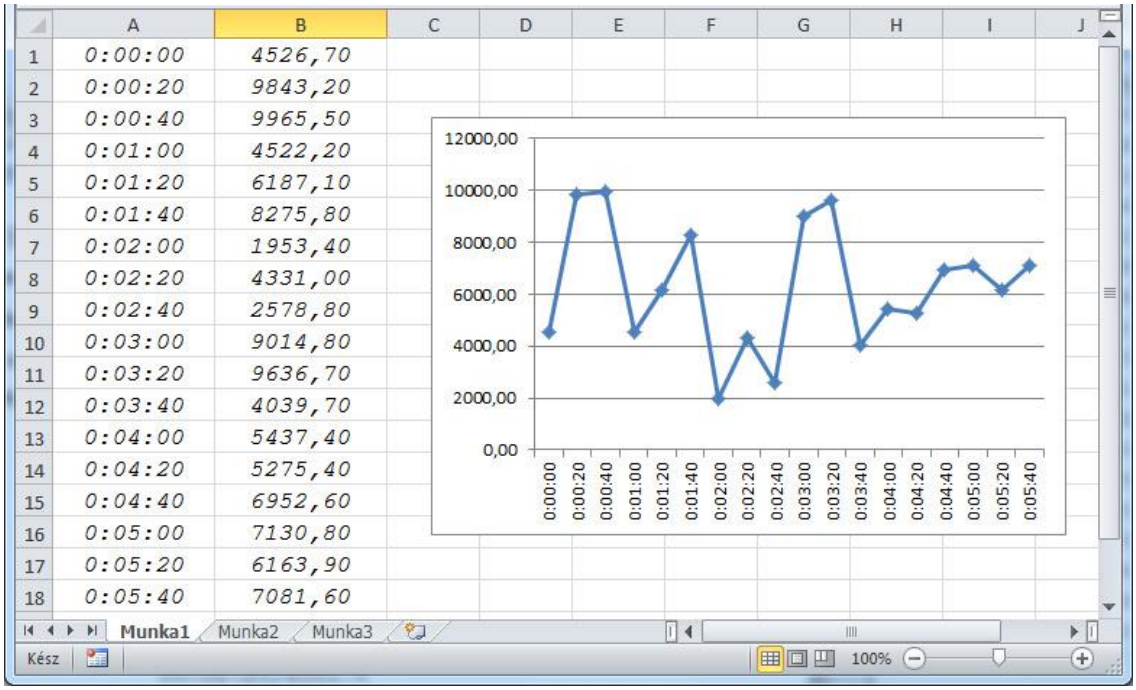
„Kliens_Data” és „Kliens_Event” könyvtárakat, szerver gépről be kell jelentkezni a kliens gépbe, így elérhetővé válik a kliens adatbázisa. A szerveren futtatott **Get_data_from_kliens.py** program ciklikusan átmásolja a megváltozott adatokat a szerver adatbázisába. Az eredmény megegyezik a felhő alapú megoldás során kapottal.

Archiválás

Az adatgyűjtő (kliens) és a megjelenítő (szerver) gép is végez archiválást. Az adatgyűjtő minden 20. másodpercben eltárolja a mért pillanatértékeket. Az archívumok jelenként napi bontásban tárolódnak. A napi archív fájlok, *'csv'* formátumban soronként egy időbélyeget és egy hozzá tartozó mért értéket tartalmaznak. Például a nap első öt percében gyűjtött adatokat tartalmazó fájl szerkezete:

```
00:00:00;4526.7
00:00:20;9843.2
00:00:40;9965.5
00:01:00;4522.2
00:01:20;6187.1
00:01:40;8275.8
00:02:00;1953.4
00:02:20;4331.0
00:02:40;2578.8
00:03:00;9014.8
00:03:20;9636.7
00:03:40;4039.7
00:04:00;5437.4
00:04:20;5275.4
00:04:40;6952.6
00:05:00;7130.8
00:05:20;6163.9
00:05:40;7081.6
```

A kapott *'csv'* fájl már közvetlenül felhasználható, például egy Excel segítségével akár grafikon formájában is vizsgálhatjuk:



A kliens gép bekapcsolt állapotában folyamatosan gyűjti és tárolja az adatokat. A pillanatértékeket folyamatosan, az archívumokat csak külön kérésre küldi fel a szerver gépbe.

A szerver is végez archiválást a kapott pillanatértékekből. A szerverben tárolt archívumok felépítése azonos a kliens gépben tárolttal. Ha mindkét gép folyamatosam működik, akkor a két gép archívumai megegyeznek. Ha olyan archív jeleket szeretnénk megjeleníteni, vagy feldolgozni, amelyek nincsenek a szerver adatbázisában, akkor azok egy célprogram segítségével feltölthetőek. A program a #Server könyvtárban található, neve ***Arc_Request.py***. A programbak nincs argumentuma, indítása után egy ablakot nyit, melyben kiválaszthatók azok a napok, melyekre szükségünk van.

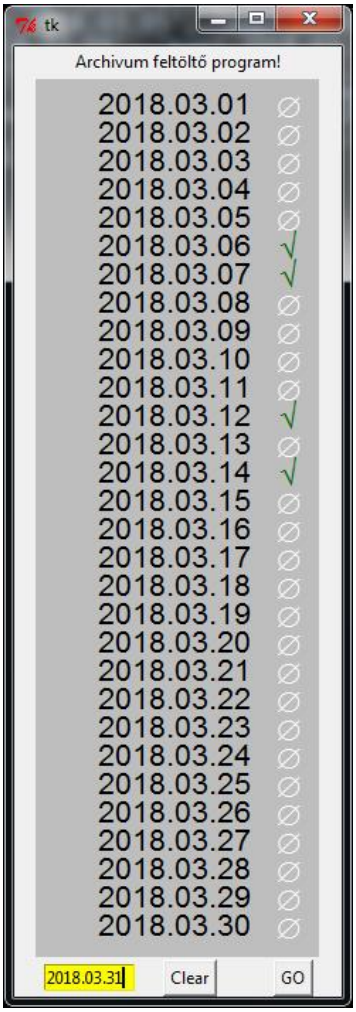
Az ablak bal alsó sarkában, sárga háttérben az aktuális dátum jelenik meg, ezt kell módosítani a kívántra. Enter lenyomása után beíródik a felső lista végére. A program ellenőrzi a beírt számokat, és csak akkor adja a listához az új értéket, ha az formailag és tartalmilag is helyes. Ellenkező esetben a program konzoljára kiírja, hogy *„Hibás dátum!”*

Ezzel a módszerrel maximum 30 nap választható ki. Ha elrontottuk a kiválasztást, akkor a ***Clear*** gomb lenyomásával törölhetjük a listát. Ha elkészült a listánk, akkor a ***Go*** gombbal lehet a lekérdezést kezdeményezni. (A konzolon megjelenik az *„Archivum lekérdezés”* üzenet.) Ha a kliens gépen fut a ***Put_data.py*** program és működik az adatátvitel a gépek között, akkor megjelennek a kért archívumok a szerver adatbázisában. (Természetesen csak azok, amelyek a kliens adatbázisában szerepelnek.)

Az adatátvitel eredményéről az *Archívum feltöltő program* ablakában kapunk visszajelzést. Ha sikerült egy adatot felhozni, akkor egy zöld pipa jelenik meg a dátum után, ha olyan jelet alkönyvtárat akarunk felkérni, amelyik nem szerepel a kliens archív adatbázisában, akkor egy fehér áthúzott kör jelenik meg. (Ha a sor végén egyik fenti jel sem jelenik meg, akkor az a napi lekérdezés, az adatátvitel során elveszett vagy sérült. Erről a konzolon az *„adatvesztés”* üzenet figyelmeztet)

Ha ugyanazt a listát szeretnénk lekérdezni még egyszer, akkor elég csak a ***Go*** gombot megnyomni. (Túl sok értelme nincs.)

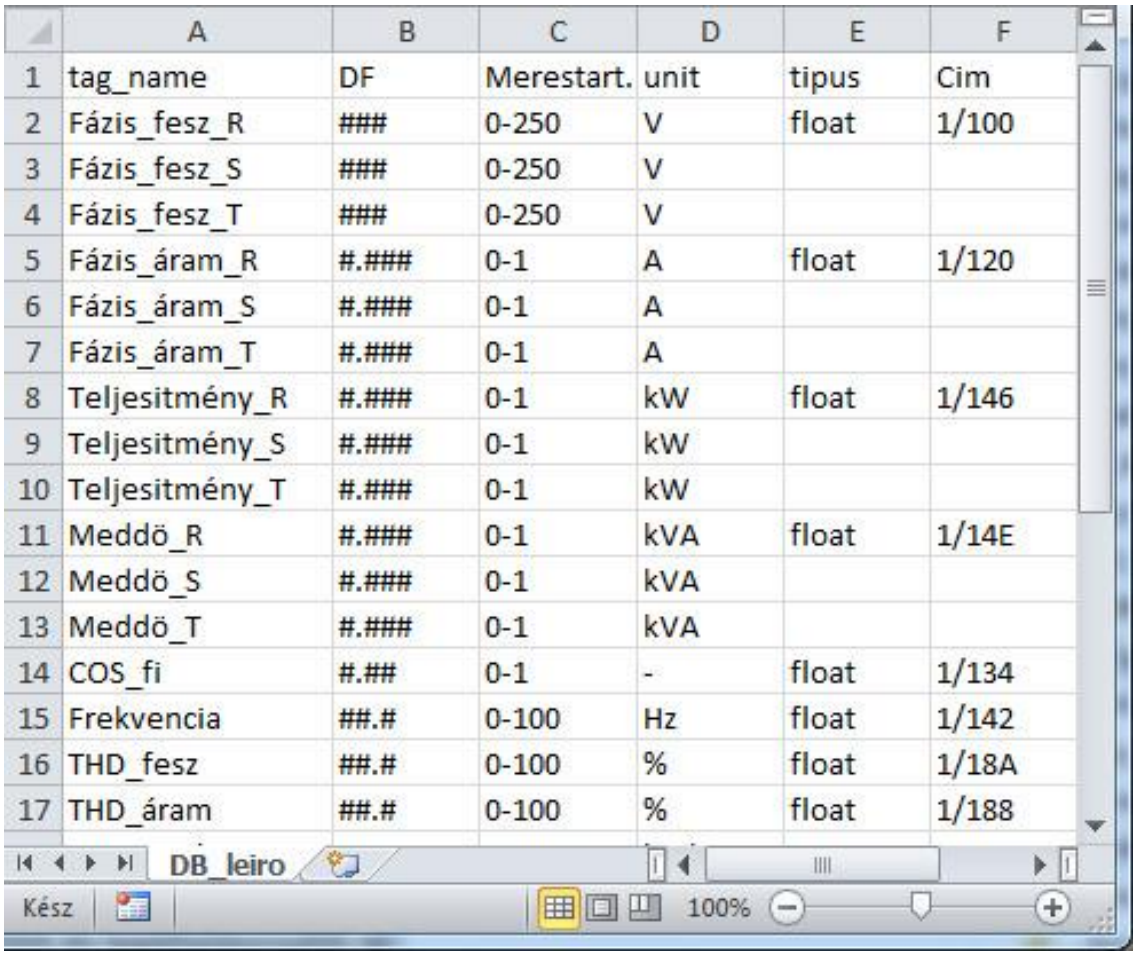
Ha újabb listát kívánunk készíteni, akkor a ***Clear*** gomb lenyomásával, tiszta lappal indulhatunk.



Megjegyzés: A *#Server* könyvtárban két segéd fájl keletkezik a tranzakció folyamán. Az egyik az *„Arc_request.csv”* mely a lekérdezendő alkönyvtárak neveit tartalmazza, pontosvesszővel elválasztva. A másik fájl (*„err.txt”*), az adatátvitel eredményét tartalmazza. Ez mindig két sorból áll, az elsőben a sikeresen felhozott alkönyvtárak vannak felsorolva, a másodikban azok, amelyeket nem tudott felhozni.

Megjelenítés

A mért jelek grafikus megjelenítésére szolgál a megjelenítő modul. A modul indítási helyén lévő **PIC** alkönyvtárban kell lennie a leíróknak. Ebben az esetben is arra törekedtünk, hogy a lehető legegyszerűbb és legáltalánosabb módon lehessen kialakítani a modul környezetét, ezért a konfigurációs állomány egy 'csv' (comma-separated values) fájl, amit Excel-el vagy tetszőleges szövegszerkesztővel elő lehet állítani. A **PIC** könyvtárban annyi leíró fájlt kell létrehozni ahány sémaképet meg akarunk jeleníteni. A leírok kiterjesztés nélküli nevei egyben a képnevek. (pl: KÉPNÉV.csv). A modul induláskor feltérképezi a PIC könyvtárat és a menü sorban megjeleníti a megtalált képek listáját. Ha egy képet el szeretnénk távolítani, akkor elég kitörölni vagy átnevezni (pl: KÉPNÉV.old).



	A	B	C	D	E	F
1	tag_name	DF	Merestart.	unit	tipus	Cim
2	Fázis_fesz_R	###	0-250	V	float	1/100
3	Fázis_fesz_S	###	0-250	V		
4	Fázis_fesz_T	###	0-250	V		
5	Fázis_áram_R	#.###	0-1	A	float	1/120
6	Fázis_áram_S	#.###	0-1	A		
7	Fázis_áram_T	#.###	0-1	A		
8	Teljesitmény_R	#.###	0-1	kW	float	1/146
9	Teljesitmény_S	#.###	0-1	kW		
10	Teljesitmény_T	#.###	0-1	kW		
11	Meddö_R	#.###	0-1	kVA	float	1/14E
12	Meddö_S	#.###	0-1	kVA		
13	Meddö_T	#.###	0-1	kVA		
14	COS_fi	#.##	0-1	-	float	1/134
15	Frekvencia	##.#	0-100	Hz	float	1/142
16	THD_fesz	##.#	0-100	%	float	1/18A
17	THD_áram	##.#	0-100	%	float	1/188

DB_leiro

Kész 100%

A leíróban minden egyes sor egy grafikus objektumot ír le, melynek 6 paramétere van, ezek az oszlopok sorrendjében a következők:

- A: **Típus** (a jel megjelenésének formája)

  Egy mért vagy számított jelnek meg lehet jeleníteni a nevét az értékét, mértékegységét. Az értékét is lehet számmal, vagy bar diagrammal megjeleníteni. A használható típusok felsorolása:

  - ***tagnév***: a jel adatbázisbeli nevét teszi ki a képernyőre
  - **szám_érték**: numerikusan jeleníti meg a jel értékét. (A szám formátumát az adatbázis leíró második (B) oszlopa tartalmazza)
  - **mértékegység**: a jel adatbázisban deklarált mértékegységét jeleníti meg
  - ***v_bar_érték***: vízszintes oszlopdiagram formájában jeleníti meg a jel értékét. (A jel adatbázisbeli méréstartománya szerint határozza meg az oszlop kezdő- és végértékét)
  - ***f_bar_érték***: függőleges oszlopdiagram formájában jeleníti meg a jel értékét. (A jel adatbázisbeli méréstartománya szerint határozza meg az oszlop alsó- és felső értékét)
  - ***graf_obj***: a jel értékétől függően más-más grafikus objektumot jelenít meg ugyanazon a pozíción. A grafikus objektum fázisait tetszőleges szerkesztővel megrajzolhatjuk. Az objektumok neve tetszőleges, a fázisokat a fájlnév végén '_' utáni sorszám azonosítja, a kiterjesztése 'gif'. Például, egy két fázisú kapcsoló:

    kapcs_0.gif kapcs_1.gif

- B: **Tag_name** (az adatbázis jel azonosítója)

  A megjelenítendő jel adatbázisbeli nevét (A oszlop) kell megadni. Ez a jelnév rendeli össze a kép leírót az adatbázissal. (Csak létező nevet lehet megadni, ellenkező esetben a megjelenítő program hibajelzés

után leáll. Ebből következik, hogy először célszerű az adatbázis leírót elkészíteni, és csak utána a megjelenítő leíróját)

- C: **Pos_x** (az objektum x koordinátája)

  A grafikus objektum vízszintes elhelyezkedését határozza meg a megjelenítő ablakban. A megjelenítő ablak az egyszerűség kedvéért fix méretű (800*600 képpont). A megjelenítő ablak és a grafikus objektumok kezdőpontja a bal felső sarok. A szöveges objektumok (tagnév, szám_érték, mértékegység) a tartományok közepére igazodnak.

- D: **Pos_y** (az objektum y koordinátája)

  A grafikus objektum függőleges elhelyezkedését határozza meg a megjelenítő ablakban. (A függőleges bar diagram értelem szerűen alulról felfelé növekszik, annak ellenére, hogy a bal felső sarokra mutat az objektum pozíciója)

- E: **Szin**(az objektum színe)

  A grafikus objektum az itt megadott színnel fog megjelenni. (Oszlopdiagramok esetén a háttérszín, és az itt megadott szín valamelyike látszik.) A következő táblázat tartalmazza a színkódok standard neveit: https://www.webucator.com/blog/2015/03/python-color-constants-module/

  A grafikus objektumnál a **Szin** információ mást jelent. Ebben az oszlop a gif kiterjesztésű objektum nevét tartalmazza, fázis jelölése nélkül. (pl. „kapcs”)

- F: **Méret** (az objektum mérete)

  A grafikus objektum méretét lehet definiálni. Itt is az egyszerűségre törekedtünk. Minden objektumnál 3 féle méret adható meg: *kicsi, közepes, nagy*. Az egyes típusoknál a következő jelentéssel bír.

  Tagnév, szám_érték, mértékegység esetén a karakter méretét jelenti: kicsi=10, közepes=16, nagy =24 pixel.

Bár diagramok esetén az objektum szélessége pixelben: kicsi=8, közepes=40, nagy =80.

(Az diagramok hossza fix. minden esetben 100 pixel.)

A grafikus objektumnál a **Méret** információ is mást jelent. Az objektum fázisainak számát jelöli. A fázisokat 0-tól számozzuk folyamatosan. A fázisok száma tetszőleges lehet. (pl. a kétállapotú kapcsoló esetén: 0-1)

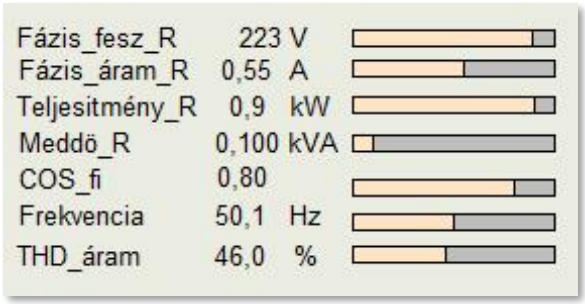
Ha nincs egyéb információ, akkor a megjelenítő modul a könyvtárban található leírók alapján, szürke háttéren jeleníti meg a leíróban definiált objektumokat. Ha háttér információkat is szeretnénk megjeleníteni, akkor elegendő létrehozni a PIC könyvtárba egy *'gif'* formátumú statikus háttérképeket. Az összerendelést a fájlok neve jelenti. Ha a leíró neve KÉPNÉV.csv, akkor a hozzá tartozó háttérkép neve KÉPNÉV.gif.

A háttérképek tetszőleges tartalmúak lehetnek, Készülhetnek bármilyen képszerkesztő programmal, vagy importálhatók CAD rendszerből, vagy lehetnek fotók is.

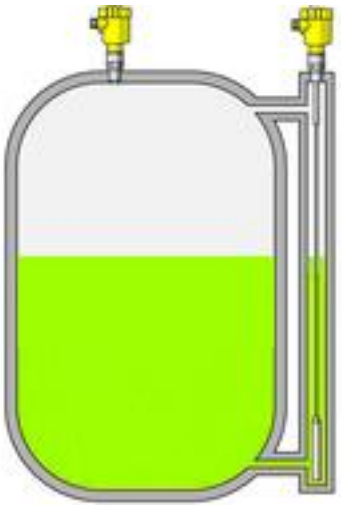
Épületfelügyelet esetén az ***alaprajzokat*** célszerű CAD rendszerből átvenni, és erre kerülhetnek fel a dinamikus adatok.

Diagramok

A bar diagramok sokrétűen használhatóak. Hagyományos felhasználási lehetőség, az analóg jelek pillanatértékeinek grafikus megjelenítése oszlopdiagramok formájában. Ezek az alkalmazástól függően lehetnek függőlegesek vagy vízszintesek.

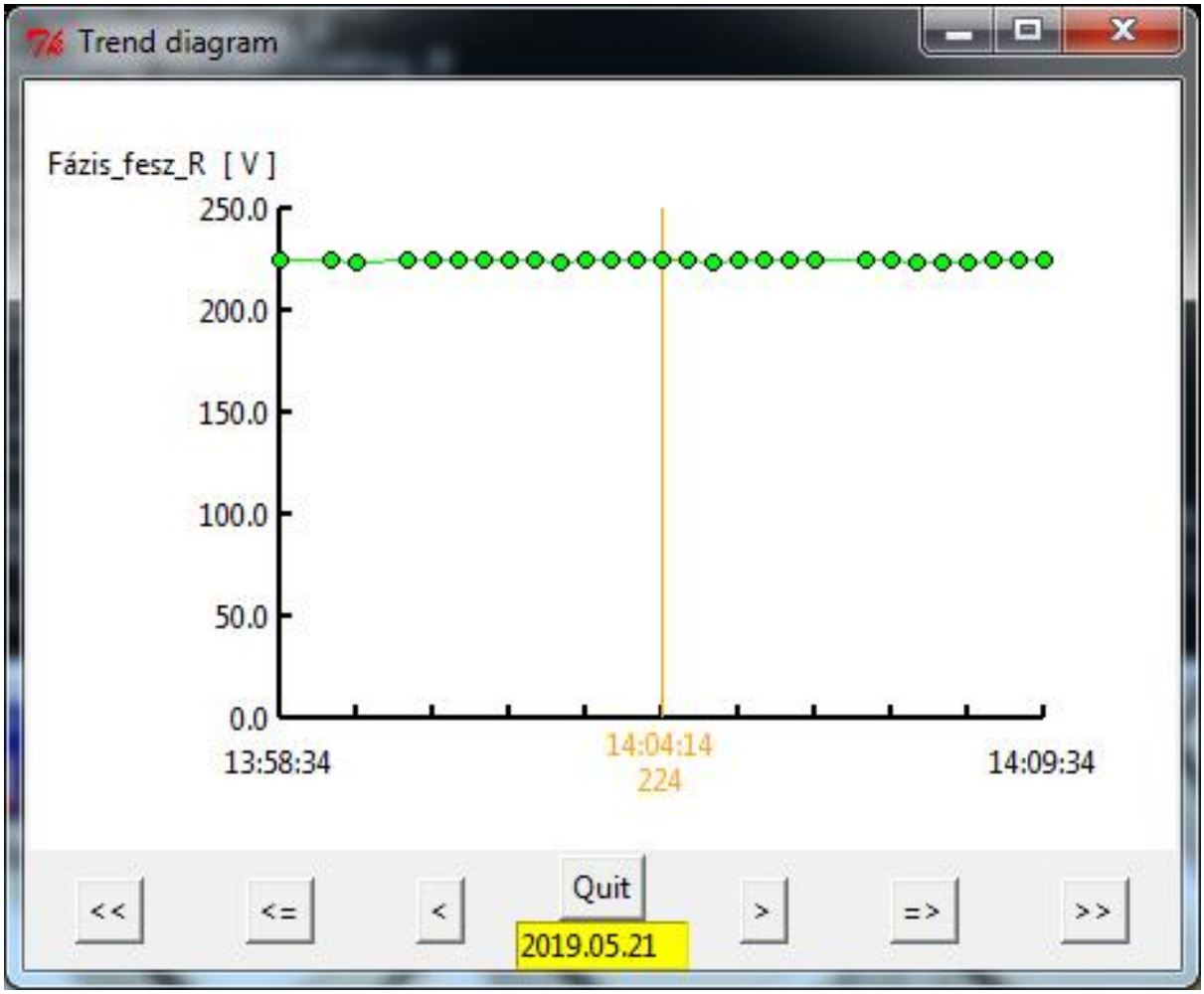


Jól használhatók a bárgráf-ként vagy tartályszintek online kijelzésére.



Trendek

Azok a jelek melyek a sémaképre oszlopdiagram formájában is szerepelnek, idődiagram (trend) formájában is kijelezhetők. Ehhez elegendő az oszlopdiagramra kattintani az egérrel. Ekkor a kiválasztott jel utolsó 10 percének idő diagramja egy új ablak jelenik meg:



Vízszintes tengelyen az idő (óra:perc:másodperc), függőleges tengelyen a vizsgált jel méréstartományának megfelelő intervallum látható. Az ablak bal felső sarkában található a jel neve és mértékegysége. A mért értékeket egy-egy kis zöld karika reprezentálja. (Ha nem volt folytonos az archiválás, akkor a hiányzó időhöz tartozó pont is hiányzik.) Egy időben csak egy analóg jel idődiagramját lehet megmutatni. Az ablak nem elmozdítható, de nem átméretezhető.

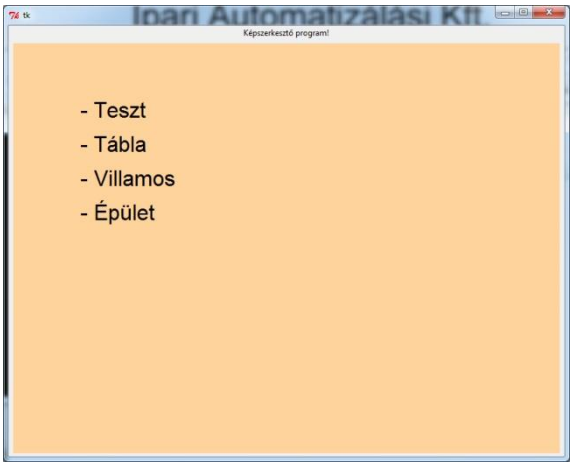
A megjelenített időintervallum az ablak alsó részén található gombok segítségével lapozható mindkét irányban. A << gombbal a nap legelső archivált jeleihez tudunk lapozni, a >> gombbal a nap legutoljára mért jeleit láthatjuk. (On-line módban ez természetesen mindig változik, hiszen folyamatosan archiválja a rendszer a mért értékeket.) A => gombbal előre, a <= gombbal vissza lehet lapozni tíz perces intervallumokban.

Az ablak alján középen láthatjuk a dátumot. Trend lehívásakor ez mindenkor az aktuális dátumot jelenti. Ha nem az aznapi archívumra vagyunk kíváncsiak, akkor ez a dátum, a szokott módon átírható. Ha létezik a kiválasztott jelnek archívuma a kért napon, akkor megjelenik az időgörbe, és a navigáló gombok, ha nem akkor üresen marad a terület, de új dátum megadható. Ezt az üzemmódot off-line trendnek nevezzük. Ennek segítségével tetszőleges korábbi időpontban készült mérések trend görbéit vizsgálhatjuk.

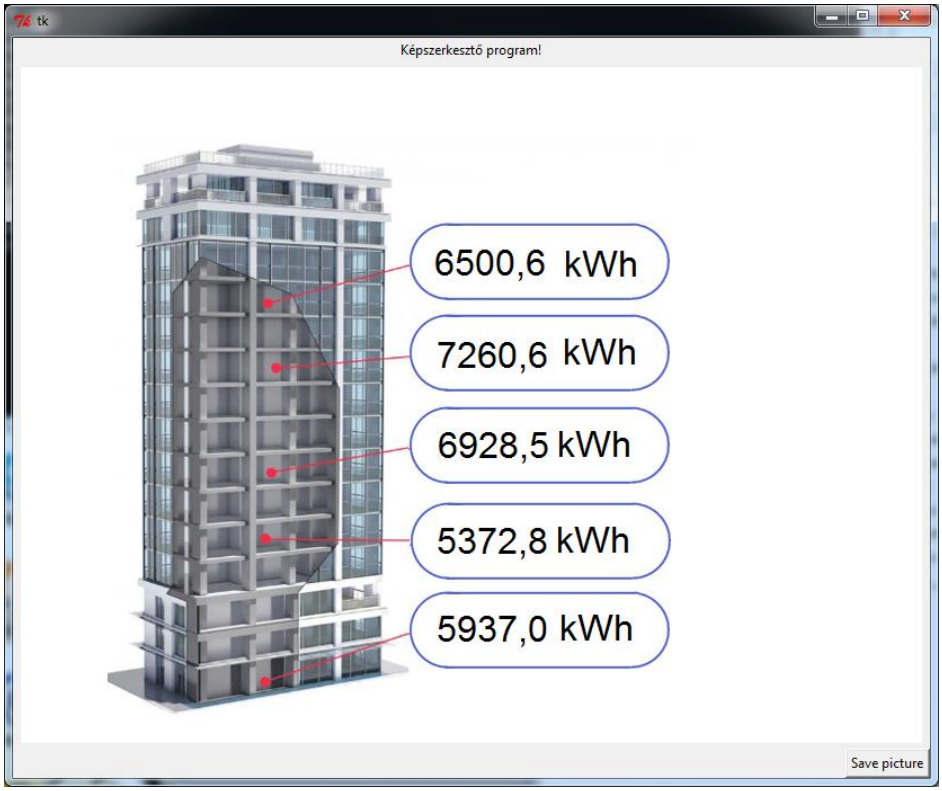
A trend diagramon van egy narancssárga függőleges szálkereszt, mely mindig az adott időintervallumhoz tartozó aktuális (mért) értéket mutatja. A szálkereszt mozgatására szolgál a < és > gomb. A szálkereszt csak olyan időpontokra ugrik, ahol történt mérés és a szerverben rendelkezésre áll az archívumban.

Képszerkesztés

A programcsomag tartalmaz egy képszerkesztő programot is (Editor.py), melynek segítségével a legalapvetőbb szerkesztési műveletek (mozgatás, másolás, törlés) elvégezhetőek. A képszerkesztő program indulása után egy menü jelenik meg a képernyőn, mely a **Pict** könyvtárban található képleírók alapján kínálja fel szerkesztésre a képeket.

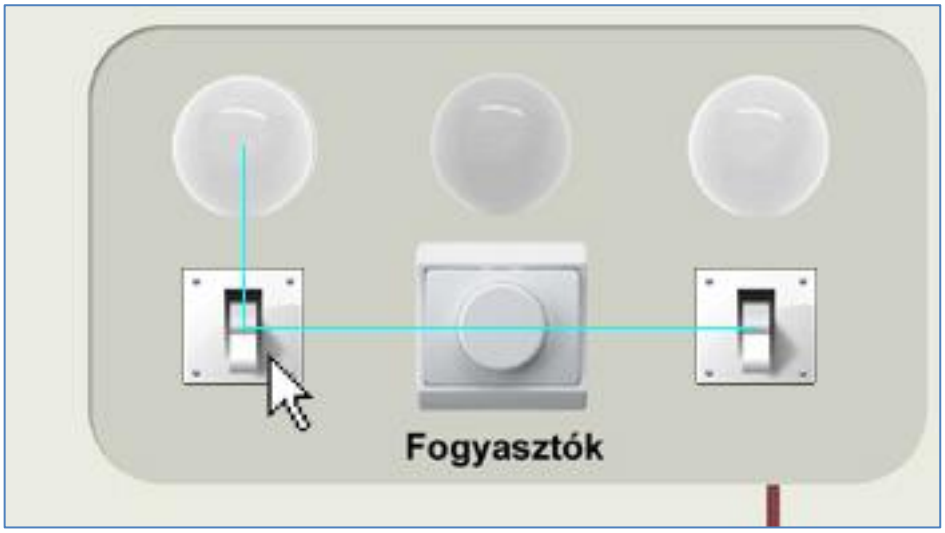


Egér segítségével történik a kiválasztás. A menü ezután eltűnik és helyette a kiválasztott képet láthatjuk a képernyőn. Ha befejeztük az adott kép szerkesztését, akkor a ***„SAVE picture”*** gomb lenyomásával elmenthetjük azt. Ha nem akarjuk a mentést, akkor egyszerűen zárjuk be a szerkesztett képet. A képet mindig a betöltési nevén menti el a program, viszont elmentéskor átnevezve megtartja a régi fájlt is, melynek neve: Képnév$.csv. Ezeket a „dolláros” fájlokat elővehetjük, ha hibás képet mentenénk el.



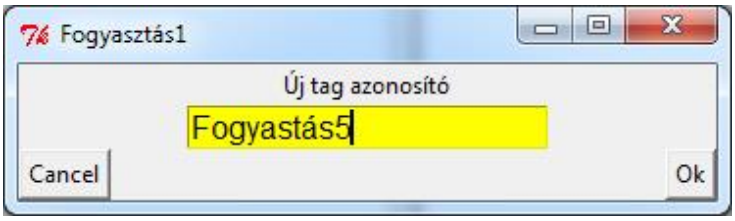
- **Mozgatás** (objektumok pozícionálása)

Ha egy objektumot mozgatni szeretnénk, akkor az egér bal gombjának lenyomva tartásával tetszőleges helyre pozícionálhatjuk az ablakon belül. Ezt a műveletet támogatja a program egy-egy vízszintes és függőleges (türkiz színű) vonal megjelenésével. A vonalak akkor, és csak akkor jelennek meg, ha a mozgatott objektum akár függőlegesen, akár vízszintesen egy vonalba kerül valamelyik másik objektummal.



- **Másolás (objektumok duplikálása)**

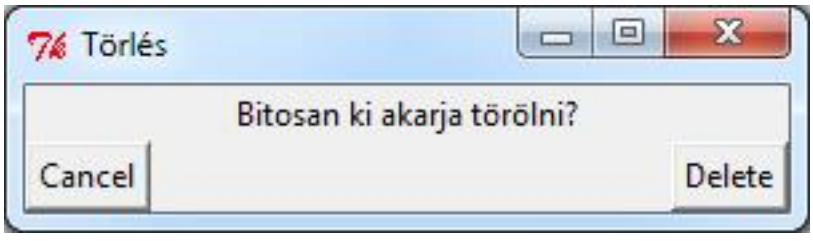
Új objektum létrehozásának a legegyszerűbb módja, ha egy azonos típusú meglévő objektumot lemásolunk és kicseréljük a jelhivatkozást a kívántra. A másolás úgy történik, hogy az egér bal gombjával kétszer ráklikkelünk a másolni kívánt objektumra. Ennek hatására a következő ablak jelenik meg a képernyőn:



A keret bak felső szélén olvasható az eredeti objektumhoz rendelt adatbázis jel neve („Fogyasztás1”). A sárga mezőbe bele kell írni azt a jelnevet, amit az új objektumhoz akarunk rendelni. (Az ablak mindaddig nem csukódik be, amíg létező adatbázis nevet nem sikerül beírnunk.) Ha a megadott név szerepel az adatbázisban, akkor az **Ok** lenyomása után ez az ablak becsukódik, és a szerkesztett képen megjelenik a másolat is. Ezt az új objektumot már csak a helyére kell mozgatni.

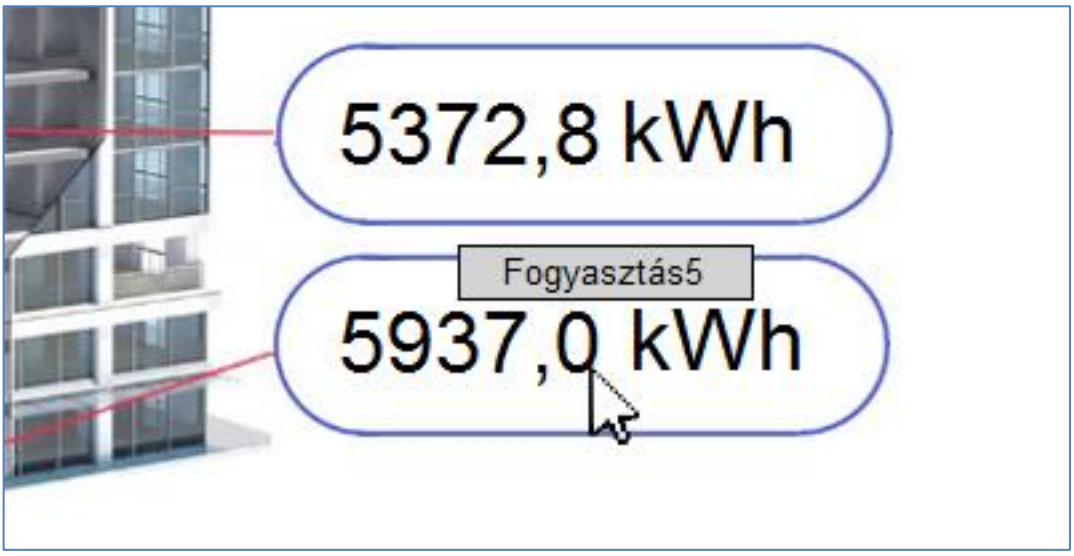
- **Törlés (objektum eltávolítása)**

Ha egy objektum feleslegessé vált, azt le tudjuk törölni. A kiválasztott objektum felett nyomjuk meg az egér középső gombját, melynek hatására kinyílik a törlés ablak, melyben meg kell erősítenünk a törlési szándékunkat.



- **Tagname (objektum adatbázis hivatkozásának kiíratása)**

Ha szerkesztés közben kíváncsiak vagyunk, hogy egy objektum viselkedése milyen adatbázis jeltől függ, akkor nem kell feltétlenül a képleíróban utánanézni, egyetlen gombnyomással kiírathatjuk azt.



Ehhez a művelethez mozgassuk az egeret a kérdéses objektum fölé és nyomjuk meg a jobb egérgombot. Ennek hatására megjelenik egy kis ablakban a jel neve, mely addig látszik, míg el nem engedjük a gombot.

Eseménykezelés

A mérési eredmények több dimenzióban: pillanatértékek, archívumok, és események formájában is elérhetőek. Az **események** mindig valamilyen változásról, valamilyen új állapot bekövetkezéséről adnak (általában szöveges) információt a kezelőknek. Az eseményeknek is nélkülözhetetlen tartozéka az időbélyeg, mely rögzíti az esemény bekövetkezésének időpontját. A rendszerünkben főleg (kivéve néhány rendszer üzenetet) a kliens gépen keletkeznek az események, és a szerver gépen láthatja azt a kezelő. A kliensen fut az ***Event_Handler.py*** program, mely a konfigurációs fájljában leírt feltételek bekövetkezésekor eseményt generál. Eseményt válthat ki egy kétállapotú jel megváltozása, egy analóg jel határérték átlépése, vagy egy hibajelzés fellépése. A technológiai jelekből származó eseményeket az Event_leiro.csv fájlban kell konfigurálni. Ezeken kívül léteznek még rendszerüzenetek is, amelyek szintén belekerülnek az eseménytárba. Ezeket külön nem kell konfigurálni, mivel a program modulok maguk állítják elő. Ilyen lehet egy modulindulás, egy hibajelzés, kommunikáció megszakadása, vagy helyreállása. A program másodperces ciklusidővel fut, ennek megfelelően az időfelbontása 1 sec. Ha ugyanabban a másodperc intervallumban több esemény is bekövetkezik, akkor a feldolgozás sorrendjében kerülnek be az eseménynaplóba azonos időbélyeggel.

Események definiálása

Az eseményeket a Kliens könyvtárban található **Event_leiro.csv** fájban kell definiálni.

	A	B	C	D	E	F	G
1	Output	trigger	unit	tipus	Jelnév	Esemény szöveg	
2	alarm	0-1		byte	Kapcsoló1	#NÉV?	
3	alarm	1-0		byte	Kapcsoló1	#NÉV?	
4	36309922644	0-1		byte	Kapcsoló3	#NÉV?	
5	alarm	1-0		byte	Kapcsoló3	#NÉV?	
6	warning	>0,3	A	float	Fázis_áram_R	értéke magas	
7	warning	<0,1	A	float	Fázis_áram_R	értéke alacsony	
8	warning	<0,1	kW	float	Teljesitmény_R	értéke alacsony	
9	warning	>0,11	kW	float	Teljesitmény_R	értéke magas	
10	limit	<0,7	-	float	COS_fi	elmászott	
11	limit	>0,8	-	float	COS_fi	már jó	
12	limit	<50	Hz	float	Frekvencia	alacsony	
13	limit	>50	Hz	float	Frekvencia	magas	
14	limit	>10	%	float	THD_áram	kritikus	
15	limit	<10	%	float	THD_áram	megfelelö	
16	limit	>100	%	float	Átlag_meddö_R	értéke magas	
17	limit	>80	%	float	Átlag_meddö_S	értéke magas	
18	limit	>120	%	float	Átlag_meddö_T	értéke magas	

A felépítése a korábban tárgyalt konfigurációs fájlokhoz hasonló. Az első sor fejléc, a további sorokban egy-egy esemény leírása történik. Az oszlopok jelentése a következő:

- A: **Output** (az esemény megnevezése, típusa)
  Lehet warning, alarm, limit, SMS küldés.

  - A ***warning*** figyelmeztető üzenetet jelent. Ezt akkor használjuk, ha valamilyen mért, vagy származtatott érték eltér a megszokottól, de nem igényel közvetlen kezelői beavatkozást.

  A **limit** olyan határérték túllépést jelent, mely nem igényel közvetlen kezelői beavatkozást.

  - Az ***alarm*** vészjelzést jelent. Ezzel az eseménnyel figyelmeztetjük a kezelőt a beavatkozásra. Ezt kiválthatja egy határérték túllépés, vagy egy kétállapotú hibajelzés fellépése is.

Ha a kezelő valamilyen okból nem tartózkodik folyamatosan a megjelenítő állomás közelében akkor célszerű az ***SMS küldés***-t használni. Ez tetszőleges esemény bekövetkezésekor, haladéktalanul üzenetet küld a megadott telefonszámra. Ezzel egy időben egy szöveges eseményt is küld a szerverre, melyben rögzíti az SMS küldés tényét. (Ez a dokumentálás érdekében fontos. Ne lehessen később arra hivatkozni, hogy a kezelő nem is kapott figyelmeztetést).

Az első oszlopban tehát a fenti címszavak szerepelnek, kivéve az SMS küldést, mert ennél az a telefonszám szerepel, amelyre az eseményt küldeni kell. (Mindig + jellel kell kezdődnie a telefonszámnak, ellenkező esetben hibajelzést ad az Event_Handler program.)

Tetszőleges, új **Output** típust is lehet használni. Nem kell mást tenni, mint a konfigurációs fájl első oszlopába beírni a kívánt nevet. Ez ugyanúgy végigmegy a rendszeren, csupán az event_wiew.py modulban az esemény szűrőt kell módosítani (sajat.colt tömb).

```
# itt lehet módosítani a szűrő feltételeket.
sajat.colt = {}         # háttérszínek összerendelése
sajat.colt["System"]="beige"
sajat.colt["alarm"]="indianred1"
sajat.colt["warning"]="orange"
sajat.colt["limit"]="olivedrab1"
sajat.colt["SMS->"]="orchid1"
sajat.colt["->SMS"]="orchid3"
```

- B: **trigger** (az esemény létrehozás feltételét rögzíti)

  Kétállapotú jeleknél külön lehet hivatkozni a fellépésre (0-ból 1-re vált a jel), vagy a megszűnésre (1-ből 0-ra vált a jel). Értelemszerűen az előbbit a ’0-1’ karakter sorozattal, az utóbbit a ’1-0’ stringgel jelöljük.

  Analóg jelek megváltozásának definiálására a kisebb-nagyobb jeleket használjuk.

  (Például a ’<50’ –el definiált esemény akkor következik be, ha a jel értéke kisebb lesz 50-nél.)

- C: **unit** (az eseményt kiváltó jel mértékegysége)

Erre azért van szükség, hogy az eseménynaplóban szerepeltetni lehessen. (Határérték túllépéskor zárójelek között belekerül az eseménynaplóba a pillanatérték, és a mértékegység is.)

- D: **tipus**

  Ez a mező a jel típusát deklarálja. Lehet byte, word, vagy float.

- E: **Jelnév** (az eseményt kiváltó jel egyedi azonosítója)

  Ebbe az oszlopba kerül be annak a jelnek az adatbázisbeli egyedi azonosítója, amelynek változása okozza az eseményt. A ***B*** oszlopban szereplő trigger feltétel erre a jelre vonatkozik. Természetesen ugyanahhoz a jelhez több eseménysort is definiálhatunk, ha azokban különbözik a trigger feltétel. Ha több jel valamilyen kombinációja együttesen vált ki egy eseményt, akkor a matematika modult kell bevetni.

- F: **Esemény szöveg** (az esemény szöveges része)

  Az utolsó az oszlopba írjuk azt a szöveget, amit látni szeretnénk az eseménynaplóban, ha bekövetkezik az esemény.

Output	trigger	unit	tipus	Jelnév	Esemény szöveg
alarm	0-1		byte	Kapcsoló1	-et elkapcsolták
alarm	1-0		byte	Kapcsoló1	-et lekapcsolták
+36309922644	0-1		byte	Kapcsoló3	-at elkapcsolták
alarm	1-0		byte	Kapcsoló3	-at lekapcsolták
warning	>0,3	A	float	Fázis_áram_R	értéke magas
warning	<0,1	A	float	Fázis_áram_R	értéke alacsony
warning	<0,1	kW	float	Teljesitmény_R	értéke alacsony
warning	>0,11	kW	float	Teljesitmény_R	értéke magas
limit	<0,7	-	float	COS_fi	elmászott
limit	>0,8	-	float	COS_fi	már jó
limit	<50	Hz	float	Frekvencia	alacsony
limit	>50	Hz	float	Frekvencia	magas
limit	>10	%	float	THD_áram	kritikus
limit	<10	%	float	THD_áram	megfelelö

Ha a kliens gépen folyik az adatgyűjtés és fut az Event_Hendler.py program, akkor a egy trigger feltétel teljesülésekor, eseményt fog generálni. A Kliens_Event alkönyvtárban megjelenik az ***event.csv*** fájl mely mindig az utolsó eseményt (eseményeket) tartalmazza. Ez a fájl szemaforként is szerepel a rendszerben, azaz

ha fut az adatküldő (Put.py) program, akkor az továbbítja az eseményeket a felhőbe, és ha sikerült, akkor letörli az event.csv fájlt.

Ezzel egy időben események napi bontásban kerülnek archiválásra, ugyanebben az alkönyvtárba. A 'csv' kiterjesztésű fájlok nevükben hordozzák a keletkezés dátumát. Tartalmuk a következő:

Az első oszlopban található az időbélyeg (év.hó.nap óra:perc:másodperc). Az időbélyeg mindig a keletkezés idejét jelöli.

A második oszlopban az esemény típusa szerepel, még a harmadikban az összetett esemény szöveg. Az esemény szövegben majdnem mindig szerepel a jelnév, kivéve a rendszerüzeneteket, ezen túl az esemény szöveges leírása. Ez utóbbi az esemény típustól függően a pillanatérték is tartalmazhatja

pl.

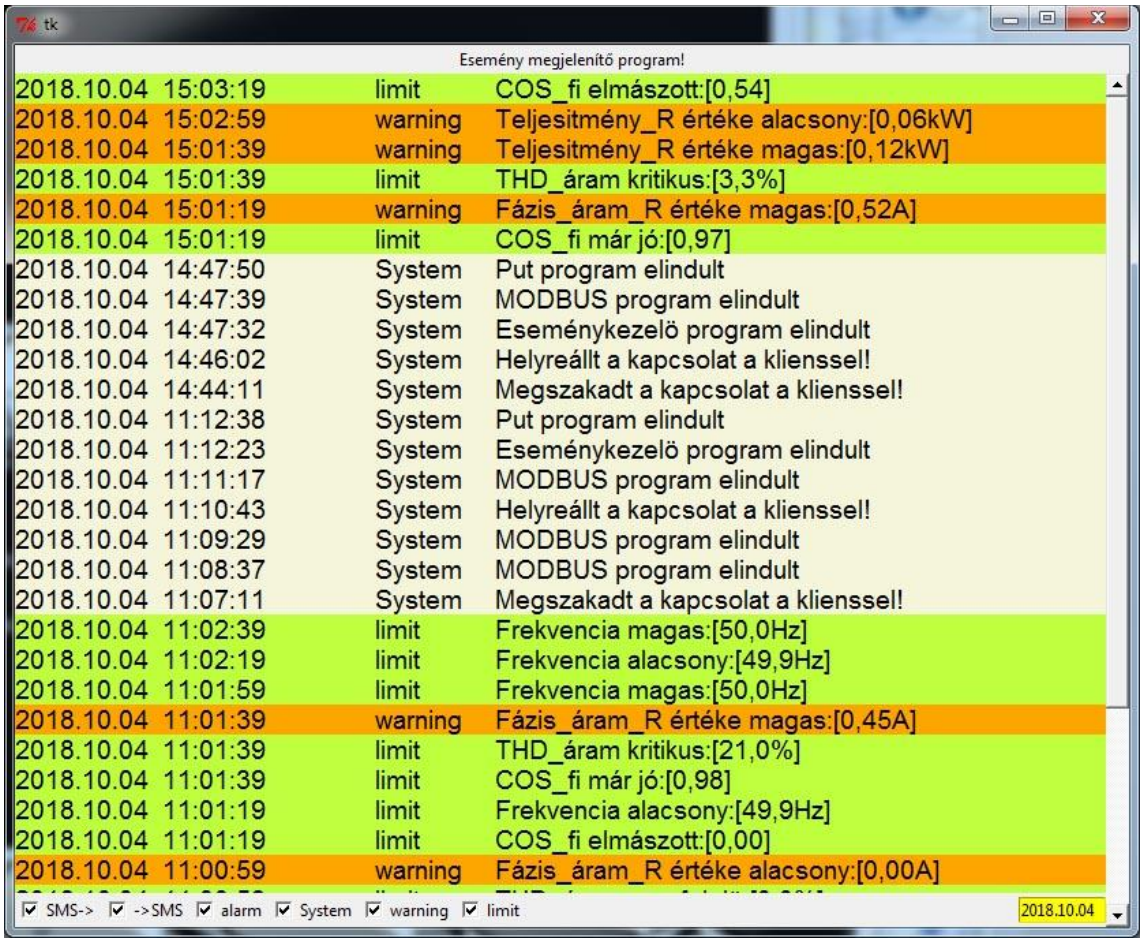
2018.08.09	14:30:00	System	MODBUS program elindult
2018.08.09	14:32:19	alarm	Kapcsoló3-at lekapcsolták
2018.08.09	14:33:59	limit	THD_áram kritikus:(7,7)
2018.08.09	14:34:59	limit	COS_fi már jó:(0,99)
2018.08.09	14:52:19	warning	Teljesitmény_R értéke alacsony:(0,01kW)
2018.08.09	14:52:19	limit	Frekvencia magas:(50,0Hz)

Ha a szerver gépen fut az adatolvasó (Get.py) program, akkor az a kapott eseményeket újból archívumba teszi. Az archívumok a Server_Event könyvtárban vannak és szerkezetük megegyezik a fent leírtakkal. Az esemény archívumokat tetszőleges szövegszerkesztővel, vagy táblázatkezelő programmal meg tudjuk nézni, de készült egy grafikus esemény megjelenítő is, melyet a következő fejezetben ismertetünk .

Események megjelenítése

A szerver gép megjelenítőjén el kell indítani az ***Event_view.py*** programot, melynek segítségével folyamatosan, on-line követhetjük a rendszer eseményeket. A program indulásakor mindig az aktuális nap eseményeit tölti be. Ha aznap még nem született esemény, akkor a képernyő legfelső sorába a „Nincs erre a napra vonatkozó esemény archívum!" figyelmeztetés jelenik meg. Dátumot váltani a képernyő jobb alsó sarkában található dátum módosításával lehet. A dátum átírása után azonnal megjelennek az aznapra vonatkozó események az időbélyegek szerint rendezve úgy, hogy a legutolsó esemény kerül legfelülre. Ha több esemény

van, mint amennyi ráfér a képernyőre, akkor a jobb oldali csúszka segítségével tudunk lapozgatni az események között.

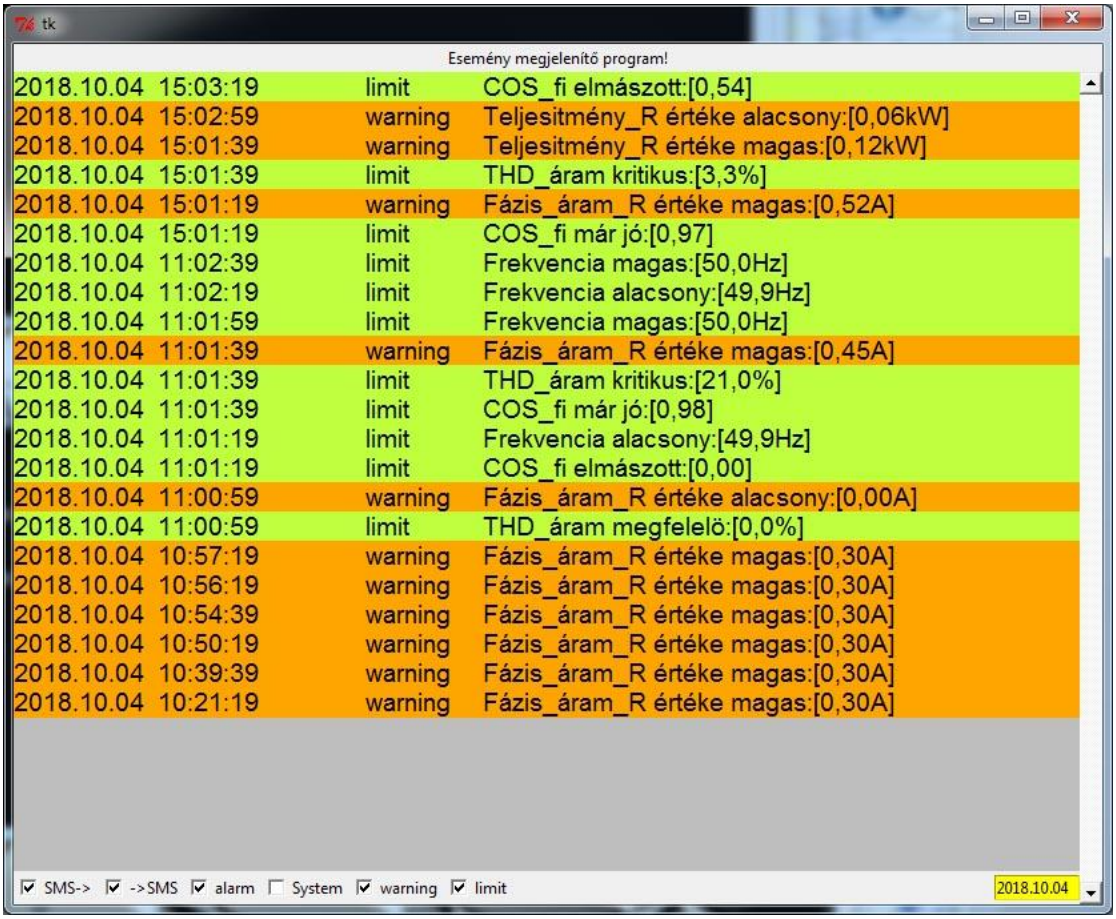


A könnyebb azonosítás érdekében, az egyes események típusát az eseménysorok háttérszíne is megkülönbözteti. A színek jelentése a következő:

["System"]="beige"
["alarm"]="indianred1"
["warning"]="orange"
["limit"]="olivedrab1"
["SMS->"]="orchid1"
["->SMS"]="orchid3"

Ha valamelyik eseménytípust nem akarjuk látni, akkor a képernyő legalsó sorában lévő szűrő mezőben a „pipát” el kell távolítani a kurzorral történő klikkeléssel.

Visszatenni értelem szerűen egy újabb klikkeléssel lehet. A szűrés eredménye azonnal láthatóvá válik (pl. a System jelek kiszűrése):



Archív események

Az adatgyűjtő (kliens) és a megjelenítő (szerver) gép is napi bontásban tárolja a rendszerben keletkezett eseményeket.

Ha az adatgyűjtő (kliens) és a megjelenítő (szerver) gép is folyamatosam működik és él közöttük az adatátvitel, akkor a két gép esemény archívumai megegyeznek. Ha viszont olyan archív eseményekre vagyunk kíváncsiak, amelyek nincsenek a szerver adatbázisában, akkor azok egy célprogram segítségével feltölthetőek a kliensből. A program a #Server könyvtárban található, neve ***Event_Request.py***. A programbak nincs argumentuma, indítása után egy ablakot nyit, melyben kiválaszthatók azok a napok, melyekre szükségünk van.

Az ablak bal alsó sarkában, sárga háttérben az aktuális dátum jelenik meg, ezt kell módosítani a kívántra. Enter lenyomása után beíródik a felső lista végére. A program ellenőrzi a beírt számokat, és csak akkor adja a listához az új értéket, ha az formailag és tartalmilag is helyes. Ellenkező esetben a program konzoljára kiírja, hogy *„Hibás dátum!”*

Ezzel a módszerrel maximum 30 nap választható ki. Ha elrontottuk a kiválasztást, akkor a ***Clear*** gomb lenyomásával törölhetjük a listát. Ha elkészült a listánk, akkor a ***Go*** gombbal lehet a lekérdezést kezdeményezni. (A konzolon megjelenik az *„Archivum lekérdezés”* üzenet.) Ha a kliens gépen fut a ***Put_data.py*** program és működik az adatátvitel a gépek között, akkor megjelennek a kért archívumok a szerver adatbázisában. (Természetesen csak azok, amelyek a kliens adatbázisában szerepelnek.)

Az adatátvitel eredményéről az *Event feltöltő*

program ablakában kapunk visszajelzést. Ha sikerült egy napi adatot felhozni, akkor egy zöld pipa jelenik meg a dátum után, ha olyan jel alkönyvtárát akarunk felkérni, amelyik nem szerepel a kliens archív adatbázisában, akkor egy fehér áthúzott kör jelenik meg. (Ha a sor végén egyik fenti jel sem jelenik meg, akkor az a napi lekérdezés, az adatátvitel során elveszett vagy sérült. Erről a konzolon az *„adatvesztés”* üzenet figyelmeztet)

Ha ugyanazt a listát szeretnénk lekérdezni még egyszer, akkor elég csak a ***Go*** gombot megnyomni. (Túl sok értelme nincs.)

Ha újabb listát kívánunk készíteni, akkor a ***Clear*** gomb lenyomásával, tiszta lappal indulhatunk.

Megjegyzés: A *#Server* könyvtárban két segéd fájl keletkezik a tranzakció folyamán. Az egyik az *„Event_request.csv”* mely a lekérdezendő alkönyvtárak neveit tartalmazza, pontosvesszővel elválasztva. A másik fájl (*„err.txt”*), az adatátvitel eredményét tartalmazza. Ez mindig két sorból áll, az elsőben a sikeresen felhozott alkönyvtárak vannak felsorolva, a másodikban azok, amelyeket nem tudott felhozni.

Számítások (matematika modul)

Az adatbázisban tárolt mért adatokon további számításokat lehet végezni. Ezek lehetnek egyszerű *alapműveletek*, vagy nagyon bonyolult *algoritmusok is.* Az adatfeldolgozás támogatására kifejlesztettünk egy modult mellyel ezek a feladatok hatékonyan elvégezhetőek. Ez a modul (Matek.py) a Python programnyelvben definiált valamennyi aritmetikai, logikai és relációs műveletet realizálja. Ezen túl lehet benne ciklust szervezni, feltételes utasításokat használni. Ezek szintaktikája a Python szintaktikájával azonos.

Aritmetikai operátorok:

- \+ összeadás
- \- kivonás
- * szorzás
- / osztás
- // egészosztás (nincs maradék)
- % maradékosztás (csak a maradék, modulo)
- ** hatványozás

Logikai operátorok:

- & és
- | vagy
- ^ kizáró vagy
- ~ komplemens
- << shift balra
- >> shift jobbra

Relációs operátorok:

- == egyenlő
- != nem egyenlő
- > nagyobb
- < kisebb
- >= nagyobb egyenlő
- <= kisebb egyenlő

Az értékadás operátor értelemszerűen a „=” jel.

A műveletek közötti precedenciák a matematikában jól ismert módon értelmezhetők.

A real-time adatbázissal való kapcsolatot a „db.py” program tartalmazza. Ezt a programot kell *importálni* a matematika modulba, és mindössze három függvényhívást kell használni:

- db.Inic Betölti a az adatbázis leírót
- db.GET beolvas egy adatot az adatbázisból
- db.PUT kiír egy adatot az adatbázisba

Az egységes kezelés adatkezelés érdekében minden műveletet lebegőpontos adatokon végzünk. Ennek megfelelően a beolvasott és kimentett adatok is *float* típusúak. Ha ciklikusan szeretnénk végrehajtani a számítási algoritmusokat, akkor erre létrehozhatunk egy egyszerű *while* ciklust. Például, ha azt szeretnénk, hogy 5 másodpercenként számoljuk ki a rendelkezésre álló áram és feszültség értékből a teljesítményt akkor azt pl. a következő módon tehetjük meg:

while(True):

```
Ur = db.GET("Fesz_mérés_001_R")
Ir = db.GET("Áram_mérés_001_R")
Pr = Ur*Ir
db.PUT("Telj_001_R" , Pr)
sleep(5)
```

A ***db.GET*** argumentuma a beolvasni kívánt jel adatbázisbeli tag neve. A db.PUT függvénynek két argumentuma van, az első a kiírásra használt jel adatbázisbeli tag neve, a második a kiírandó érték változójának lokális neve. (Ahogy ez a Python programozásban megszokott, nem kell előre deklarálni a változókat.) A tag nevek string típusúak, ezért ezeket aposztrófok közé kell tenni, természetesen lehetnek string változók is. A várakozásra a sleep függvényt célszerű használni, mivel ez nem terheli a processzort.

Alkalmasan megválasztott tag nevekkel racionalizálni tudjuk a számítások leírását és futtatását. Példaként számoljuk ki 9 db mérés teljesítmény tényezőjét, ha mérjük az áramot a feszültséget és a hatásos teljesítményt. Képlet szerint:

```
cosφ = Pw / (U *I)
```

Megvalósítása a matematika modullal, ha a jelnevek rendre ####_mérés_01, ***####_mérés_02***, ***####_mérés_03*** stb. Az így definiált jelekre könnyen ciklust tudunk szervezni:

```
for i in range(9):

     Pw = db.GET("Telj_mérés_0")"+str(i+1))
     U =   db.GET("Fesz_mérés_0")"+str(i+1))
     I  =   db.GET("Áram_mérés_0")"+str(i+1))
     db.PUT("Cosfi_0"+str(i+1) , Pw/U/I))
```

A modulok reentráns módon lettek kialakítva ezért egyszerre, párhuzamosan több is futtatható belőlük. Mivel a matematika modult fogjuk használjuk a különböző algoritmusok kiszámítására is, ezért azokat célszerű külön-külön modulban megfogalmazni és párhuzamosan futtatni.

Szimuláció

A fejlesztés során szükség van olyan megoldásokra melyek segítségével tesztelni lehet a rendszer működését anélkül, hogy rendelkezésre állna a valós technológia, vagy a technológiaillesztés. Legegyszerűbb lehetőség az adatbázis jeleinek manipulálása. Mivel az adatbázis jelek „csv" formátumban tárolódnak, ezért egy tetszőleges editor segítségével meg tudjuk változtatni a jelek pillanatértékét. Ehhez nem kell mást tenni, mint a **szerver** adatbázisában átírni a megfelelő fájlban a jel értékét. (pl. az R fázis áramának szimulálásához meg kell változtatni a #Server/Server_Data/Pillanat/Fázis_áram_R.csv fájlban a jel értékét.)

Ennél bonyolultabb szimuláció elvégzésére külön szimulációs modult fejlesztettünk ki.

A **szimulációs modulnak** sincs saját konfigurációs fájlja, bemenetként az adatbázis leírót használja. Az ott specifikált jeleket állítja elő és beteszi a *Kliens_Data* adatbázisba. A program indítása után ciklikusan generálja a *db_leiro.csv*-ben definiált jeleket, a *méréstartomány* figyelembevételével és beírja a kliens *real-time* és *archív* adatbázisába. A ciklusidő default értéke 20 másodperc. A ***Symulator.py*** modul csak azokat a jeleket szimulálja amelyeknek az adatbázis leíróban a *típus* és *Cim* oszlopában szerepel valami. Ezzel azt tudjuk elérni, hogy a szimuláció során az adatgyűjtéssel azonos jelek kapjanak értéket. Így tesztelni tudjuk a feldolgozó modulok működését anélkül, hogy működne az adatgyűjtés. Értelem szerűen, ne futtassuk egy időben a *Symulator.py* és a *Modbus_Client.py* programot! Mivel mindkét programnak ugyanaz a kimenete, ezért összekeverednének a mért és szimulált jelek.

Algoritmusok

Szabályos számlakép generáló program

A program (***Szamla.py***) a fogyasztási adatok alapján havonta számlákat generál. A program bemenetei a következők:

- Fogyasztók adatai (Szamla_leiro.csv)
- Minta számlakép (szamla/Etalon_szamla.xls)
- A számla kibocsájtó logója (szamla/image1.png)
- Fogyasztási adatok Server_Data/Havi_Fogyasztás/)

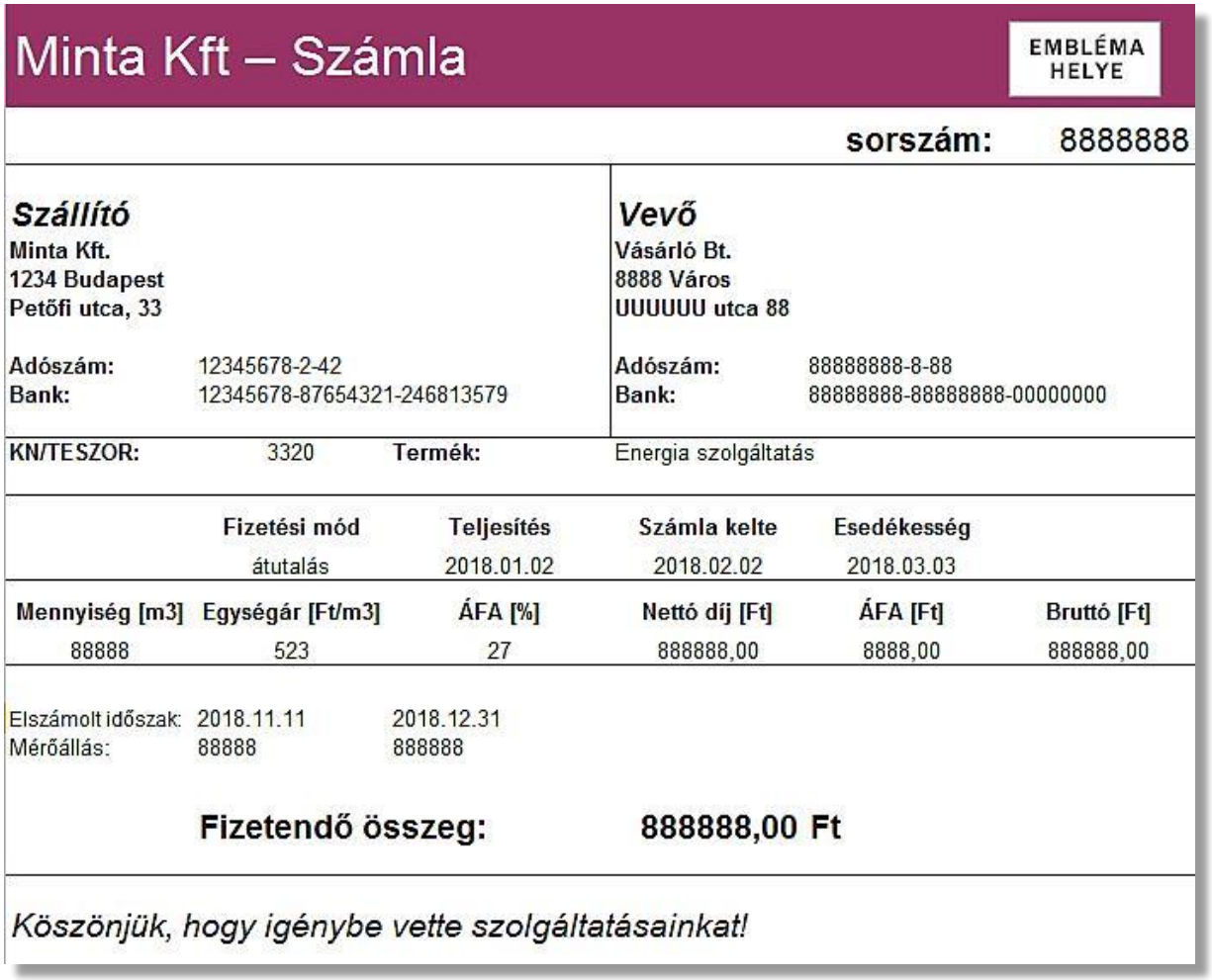
A **számla leíró** tartalmazza mindazokat a fogyasztói egyedi adatokat, melyek a számla kitöltéséhez szükségesek. A „CSV” fájl minden sora egy-egy fogyasztó adatait határozza meg.

tag_name	Név	Város	utca/házszám	adószám	Számlaszám	Határidö
Fogyasztás1	Elsö fogyasztó Kft.	1234 Budapest	Elsö utca 11	12345678-2-11	12345678-12345678-00000000	30
Fogyasztás2	Második Bt.	9281 Piripócs	Akármi út 4	87654321-2-42	87654321-70716253	15

A fájl első sora komment. Az első oszlop a mérés adatbázisbeli nevét tartalmazza. Ez az azonosító egyértelmű összerendelést biztosít az on-line adatgyűjtés és a számlázott mennyiség között. A második oszlopban a fogyasztó nevét tartalmazza, a harmadik és negyedik a pontos számlázási címet adja meg. Az ötödik a fogyasztó adószámának a helye, a hatodik a számlaszámé. A hetedik oszlopban a fizetési határidő hosszát rögzíti.

Ezeket az adatokat egyszer kell kikölteni, és csak akkor kell változtatni, ha bővíteni akarjuk, vagy valamelyik adat időközben megváltozott. A fájlt Excel-el, vagy tetszőleges szövegszerkesztővel módosíthatjuk. A módosított fájlt ugyanolyan néven kell elmenteni.

A **minta számlaképe** egy „XLSX” fájl mely az elkészítendő számla formáját, adatainak helyét, esetlegesen értékét is tartalmazza.



Minta Kft – Számla

EMBLÉMA HELYE

sorszám: 8888888

Szállító
Minta Kft.
1234 Budapest
Petőfi utca, 33

Adószám: 12345678-2-42
Bank: 12345678-87654321-246813579

Vevő
Vásárló Bt.
8888 Város
UUUUUU utca 88

Adószám: 88888888-8-88
Bank: 88888888-88888888-00000000

KN/TESZOR: 3320 Termék: Energia szolgáltatás

Fizetési mód	Teljesítés	Számla kelte	Esedékesség
átutalás	2018.01.02	2018.02.02	2018.03.03

Mennyiség [m3]	Egységár [Ft/m3]	ÁFA [%]	Nettó díj [Ft]	ÁFA [Ft]	Bruttó [Ft]
88888	523	27	888888,00	8888,00	888888,00

Elszámolt időszak: 2018.11.11 2018.12.31
Mérőállás: 88888 888888

Fizetendő összeg: 888888,00 Ft

Köszönjük, hogy igénybe vette szolgáltatásainkat!

A mintával sok teendőnk nincs, a benne szereplő „dummy” adatok nagy részét a *számlázó program* cseréli le valóságos adatokkal, a *számla leíró* és az adatgyűjtés során adatbázisba került fogyasztási adatok alapján. A *szállító* adatain kívül csak két változót kell itt megadni, az egyik az egységár, a második az ÁFA aktuális értéke %-ban. Ezek ritkán módosuló tételek, csak változás esetén szükség aktualizálni őket. (Ha nem a mintában szereplő fogyasztástípusról akarunk számlát készíteni, akkor a mennyiség és az egységár mértékegységét is át kell írni.)

A szállító saját logójának is van helye a számlán, ennek aktualizálásához az *image1.png* képfájl kell megváltoztatni. (Ez tetszőleges képszerkesztővel megtehető, pl. *Paint*.)

A számlázó programot minden hónapban egyszer le kell futtatni. Mindegy, hogy melyik napot választjuk, a program mindig az előző hónap mért adatai alapján állítja ki a számlát. Ha szándékosan, vagy véletlenül többször indítjuk el a programot, akkor csak az esetlegesen újonnan felvett fogyasztókra állít ki számlát, a már meglévőket nem

írja felül. (Ilyen esetben a következő figyelmeztetést adja pl.: *„Van már ebben a hónapban ilyen számla! Fogyasztás1_2018_11.xlsx”*)

Minta Kft – Számla

sorszám: 12345690

Szállító
Minta Kft.
1234 Budapest
Petőfi utca, 33

Adószám: 12345678-2-42
Bank: 12345678-87654321-246813579

Vevő
Elsö fogyasztó Kft.
1234 Budapest
Elsö utca 11

Adószám: 12345678-2-11
Bank: 12345678-12345678-00000000

KN/TESZOR: 3320 Termék: Energia szolgáltatás

Fizetési mód	Teljesítés	Számla kelte	Esedékesség
átutalás	2018.11.20	2018.11.20	2018.12.20

Mennyiség [m3]	Egységár [Ft/m3]	ÁFA [%]	Nettó díj [Ft]	ÁFA [Ft]	Bruttó [Ft]
45	523	27	23535,00	6354,45	29889,45

Elszámolt időszak: 2018.10.01 2018.10.31
Mérőállás: 2465 2510

Fizetendő összeg: 29889,45 Ft

Köszönjük, hogy igénybe vette szolgáltatásainkat!

A számlák fájljai archívumba (#Server/szamla/Elkeszult_szamlak/) (kerülnek és bármikor - akár évek múlva is- ellenőrizhetőek.

Kontingensfigyelés, menetrend

Az algoritmus célja: A fogyasztó és a szolgáltató között fennálló energiafogyasztási keretszerződésben rögzített, időszakosan meghatározott energiafogyasztási limit betartása. Az előző év időszakos fogyasztási trendjeiből képzett, a következő évre vonatkozó fogyasztási tervezet előkészítése.

A létesítmény energia korlátozását a hálózati topológia és a fogyasztásmérő jeleinek, valamint az operátor beavatkozásainak függvényében végzi a rendszer.

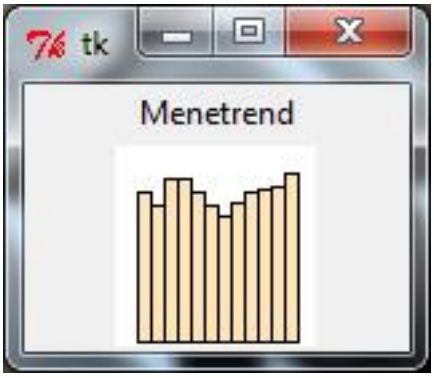
Túlfogyasztás közelében a rendszer javasolja a villamos fogyasztás csökkentését.

A központi teljesítményfelvételt korlátozó algoritmus tehát nem végez kapcsolásokat, csak tanácsolja fogyasztók kikapcsolását, ami komfort fokozat csökkenéssel jár együtt.

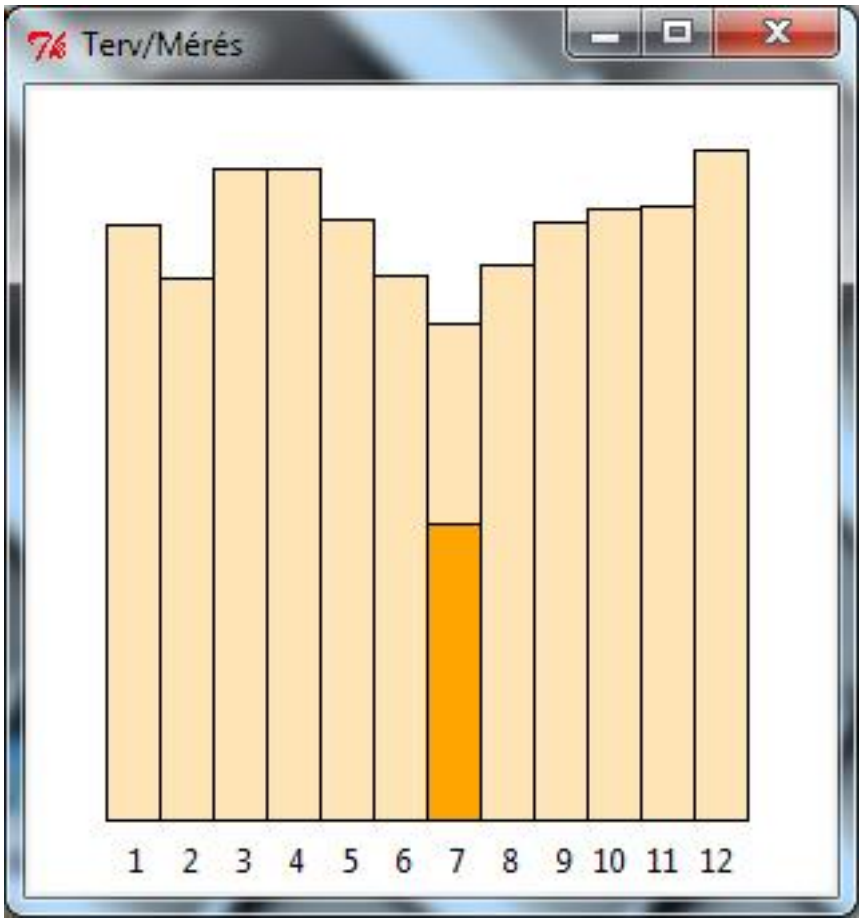
Korábbi évek adatainak elemzésén alapuló, várható fogyasztás előrejelző algoritmus épült be a rendszerbe. Energiagazdálkodási, terhelésvezérlési modult fejlesztettünk, mely eredményeként a program képes éves trendek készítésére, költségszámításra, költségelosztásra, tarifák kezelésére, menetrendfigyelésre szerkesztésre, lekötött kontingens elérésére való figyelmeztetésre. A kontingensfigyelő modul működése független a többi modultól, csak az adatbázis jeleiből dolgozik. Feladata a korábbi évek fogyasztási adatai alapján készült lekötési kontingens folyamatos ellenőrzése, és a kezelők figyelmeztetése ha megközelít vagy átlépi a fogyasztás a kritikus értéket. A kontingenseket egy csv fájlban kell definiálni, mely havi bontásban tartalmazza fogyasztás értékeket. A fájl neve „Fogyasztas_terv.csv" és a „#Server/Menetrend/" könyvtárban található:

	A	B
1	2018. január	11132
2	2018. február	10161
3	2018. március	12203
4	2018. április	12211
5	2018. május	11248
6	2018. június	10194
7	2018. július	9322
8	2018. augusztus	10383
9	2018. szeptember	11210
10	2018. október	11465
11	2018. november	11510
12	2018. december	12534

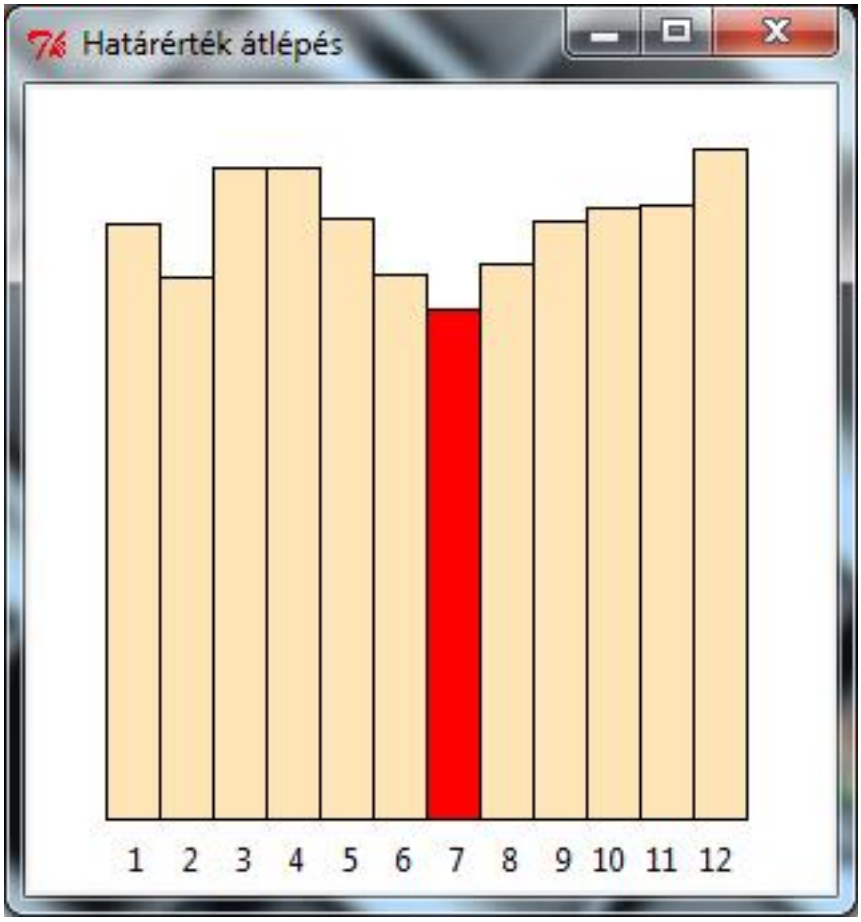
A modul (Menetrend.py) indítása után egy kis ablak jelenik meg a képernyőn:



Ha erre klikkelünk az egérrel, akkor egy másik, nagyobb ablakban láthatóvá válik az éves kontingens és az aktuális havi fogyasztás:



A világos oszlopdiagramok mutatják a terv értékeit, a sötét oszlop, az aktuális hónap fogyasztási állapotát jelzi. Ha az aktuális fogyasztás eléri vagy meghaladja a tervben szereplő kritikus értéket, akkor riasztást küld kezelők számára. Ennek formája a nagyobb ablak megnyitása a fejlécben a figyelmeztetéssel: „Határérték átlépés”. (Ha a nagyobb ablak nyitott, akkor is újra nyitja, így nem vész el a figyelmeztetés.)


Határérték átlépés
1 2 3 4 5 6 7 8 9 10 11 12

Tutorial

Egy egyszerű mintapéldán keresztül bemutatjuk a rendszer fejlesztés legalapvetőbb lépéseit. A mintapélda egy teljesítménymérő illesztésén, adatainak megjelenítésén keresztül mutatja be a konfigurálás lépéseit.

Könyvtár szerkezet létrehozása

Adatgyűjtő gép (kilens)

A főkönyvtár neve tetszőleges lehet, célszerű a projekt nevének valamilyen egyszerű formáját választani. Jelen példánkban legyen a főkönyvtár neve **Tutorial.** Az adatgyűjtő gépen kötelező létrehozni egy **#Kliens** nevű alkönyvtárat, ezen belül pedig egy **Kliens_Data** nevűt. A #Kliens alkönyvtáron belül a következő programokat találjuk:

- db.py
- encrypt.py, (S_doboz.csv)
- Modbus_Client.py
- Put_data.py

Opcionálisan:

- Symulator.py
- Copy_data_to_server.py

Megjelenítő gép (server)

A főkönyvtár neve itt is tetszőleges lehet. Példánkban legyen a neve ennek is **Tutorial.** Az megjelenítő gépen kötelező létrehozni egy **#Server** nevű alkönyvtárat, ezen belül pedig egy **Server_Data**, és egy **Pict** nevűt is. A **#Server** alkönyvtáron belül a következő programokat találjuk:

- db.py
- encrypt.py, (S_doboz..csv)
- Get_data.py
- Ablakok.py

- Matek.py

Opcionálisan:

- Editor.py
- Szamlazo.py

A **Server_Data** könyvtáron belül létre kell hozni egy **Pillanat** nevű alkönyvtárat. Ebbe kerülnek a mért és számított pillanatértékek.

Adatbázis létrehozása

Az adatbázis leíró létrehozása az első lépés. A leíró neve db_leiro.csv A leírót legegyszerűbb Excel programmal létrehozni, de bármilyen szövegszerkesztővel editálható a táblázat. A lényeg, hogy a sorok elemei között ';' legyen az elválasztójel. A leíró első sora komment, mely a következő tételeket tartalmazza:

- **tag_name**: jel azonosító
- **DF**: adat formátum
- **Merestart**: méréstartomány
- **Unit**: mértékegység
- **tipus**: jel típus
- **Cím**: PLC azonosítója / a jel címe

Az első sor megléte kötelező, tartalma tetszőlegesen átírható. Ezek után soronként definiáljuk az egyes adatbázis jeleket. Példaként definiáljuk a teljesítménymérő egyik fázis feszültségét, melyet az 1-es című eszközből kapjuk MODBUS-kommunikáción keresztül.

tag_name	*DF*	*Merestart.*	*unit*	*tipus*	*Cim*
Fázis_fesz_R	*###*	*0-250*	*V*	*float*	*1/100*

A fenti táblázatban látható, hogy a háromfázisú hálózatból az R fázist mérjük, ezért a jel neve Fázis_fesz_R.

A jelet három helyi értéken tizedesek nélkül akarjuk ábrázolni, ezért a második oszlopban '###' található.

A fázisfeszültség névlegesen 230V, így méréstartománynak 0-250-et adunk meg.

A feszültség mértékegysége értelem szerűen Volt.

A teljesítménymérő lebegőpontos adatként szolgáltatja a fázisfeszültséget, így a jel típusának 'float'-nak kell definiálni. (Ezt az információt a kommunikációs modul használja, ez alapján konvertálja a táviratban kapott 4 adatbájtot.)

Annyi kommunikációs modult indítunk, ahány eszközt illesztünk az adatgyűjtő hálózatra. Minden eszköznek egyedi azonosítója van, melyet a kommunikációs modul fejlécében kell megadni. Jelen esetben a Modbus_Client.py program 18. sorában :

Eszkoz_unit =1 # az 1-es azonosítóju eszközt kérdezzük

Ennek megfelelően a cím első része 1, a perjel után a jel **hexadecimális** címét kell megadni, ami jelen esetben 100.

Az adatbázis leíró további sorait hasonló elvek alapján kell kitölteni. Példánkban felvesszük az adatbázisba a többi fázisfeszültséget, a három fázisáramot, hatásos és meddő teljesítményt. Az áramokat amperben mérjük. Mivel a teszt eszközben kis terhelések vannak, ezért a méréstartomány nullától ötig terjed csak. Hasonló megfontolásból a teljesítmény mérések méréstartománya csak nullától egyig terjed, mivel a mérő kW-, illetve kVA-ben adja fel az adatokat.

Az adatbázis leíróban minden jelet fel kell sorolni, amelyekre később hivatkozni akarunk akár az adatfeldolgozás, akár a megjelenítés során. Így kerültek a táblázatba az S és T fázis jelei vagy a meddő teljesítmények. Ezeket ugyanúgy egy-egy sorban definiáljuk a leíróban avval a különbséggel, hogy ezekhez nem tartozik *típus* és *Cim* adat. Ne írjunk ezekbe az oszlopokba semmit, csak az elválasztó pontosvessző szerepeljen. (Az Excel automatikusan így menti el CSV formátum esetén). A kommunikációs modulok és a szimuláció ezen információ alapján szűri az adatbázis elemeket.

Példánkban felvettünk még négy mért jelet. A teljesítmény tényezőt (cos_fi), a frekvenciát, és a feszültség illetve az áram felharmonikus tartalmát mutató (THD_fesz, THD_áram) jeleket:

tag_name	*DF*	*Merestart.*	*unit*	*tipus*	*Cim*
Fázis_fesz_R	*###*	*0-250*	*V*	*float*	*1/100*
Fázis_fesz_S	*###*	*0-250*	*V*		
Fázis_fesz_T	*###*	*0-250*	*V*		
Fázis_áram_R	*#.##*	*0-5*	*A*	*float*	*1/120*
Fázis_áram_S	*#.##*	*0-5*	*A*		

Fázis_áram_T	*#.##*	*0-5*	*A*		
Teljesitmény_R	*#.##*	*0-1*	*kW*	*float*	*1/144*
Teljesitmény_S	*#.##*	*0-1*	*kW*		
Teljesitmény_T	*#.##*	*0-1*	*kW*		
Meddö_R	*#.##*	*0-1*	*kVA*		
Meddö_S	*#.##*	*0-1*	*kVA*		
Meddö_T	*#.##*	*0-1*	*kVA*		
COS_fi	*#.##*	*0-1*	*-*	*float*	*1/134*
Frekvencia	*##.#*	*0-100*	*Hz*	*float*	*1/142*
THD_fesz	*##.#*	*0-100*	*%*	*float*	*1/18A*
THD_áram	*##.#*	*0-100*	*%*	*float*	*1/188*

A prototípus kipróbálásához telepíteni kell a szükséges futtató környezetet, a ***Melléklet/Rendszer*** telepítés fejezet szerint.

Adatkapcsolatok a technológiával (Kommunikáció)

A soros aszinkron vonalon keresztül MODBUS protokoll szerinti adatgyűjtés. Ehhez semmi mást nem kell tenni, mint elindítani a *Modbus_Client.py* programot. A programnak nincs saját konfigurációs fájlja, bemenetként az előző pontban ismertetett adatbázis leírót használja. Az ott specifikált jeleket kérdezi le és tárolja a *Kliens_Data* adatbázisba. Mivel a python program egy szöveges fájl, ezért nem láttuk szükségesnek külön szöveges konfigurációs fájlt alkalmazni, csupán a program 13. sorától kell megadni a soros vonali paramétereket a következő módon:

- P_unit = 1 # az 1-es azonosítóju eszközzel kommunikálunk
- P_method = 'rtu' # MODBUS rtu protokoll szerinti kommunikáció
- P_port = 'com4' # A gép 4. soros csatornáján keresztül kommunikál
- P_timeout = 1 # 1 másodpercig vár a válaszra
- P_parity = 'N' # 8 bites jelek mennek paritás bit nélkül
- P_baudrate = 9600 # Az átvitel sebessége 9600 bit / sec

Ezután el kell indítani a programot, mely egy inicializálási procedúra után ciklikusan lekérdezi a db_leiro.csv-ben definiált jeleket, konvertálja azokat és beírja a kliens *real-time* és *archív* adatbázisába. A ciklusidő default értéke 20 másodperc. A program minden ciklusban kiírja a konzolra a lekérdezés idejét, és az esetleges hibaüzeneteket. Leggyakoribb hibaüzenet:

- *nincs kapcsolat az adatgyűjtővel*

aminek a következő okai lehetnek: nincs összeköttetés az adatgyűjtővel, rossz a kábel, nincs bekapcsolva az adatgyűjtő, nincs szinkronban a beállított paraméter lista a valósággal.

Szimuláció

Ez a modul biztosítja a teljes rendszer tesztelését abban az esetben, ha nem áll rendelkezésre a technológia, vagy a technológia illesztés. A szimulációs modulnak sincs saját konfigurációs fájlja, bemenetként az ez is az adatbázis leírót használja. Az ott specifikált jeleket állítja elő és beteszi a *Kliens_Data* adatbázisba. A program indítása után ciklikusan generálja a *db_leiro.csv*-ben definiált jeleket, a *méréstartomány* figyelembevételével és beírja a kliens *real-time* és *archív* adatbázisába. A ciklusidő default értéke 20 másodperc. A *Symulator.py* modul csak azt a hét jelet szimulálja, amelyeknek az adatbázis leíróban a *típus* és *Cim* oszlopában szerepel valami. Ezzel azt tudjuk elérni, hogy a szimuláció során az adatgyűjtéssel azonos jelek kapjanak értéket. Így tesztelni tudjuk a feldolgozó modulok működését. Értelem szerűen, ne futtassuk egy időben a *Symulator.py* és a *Modbus_Client.py* programot! Mivel mindkét programnak ugyanaz a kimenete, ezért összekeverednének a mért és szimulált jelek.

Jelen esetben a Kliens_Data könyvtárban megjelenik az *Aktual.csv* adatfájl, melynek tartalma az alábbihoz hasonló:

2018.04.16 10:36:35
Fázis_fesz_R 248
Fázis_áram_R 0.83
Teljesítmény_R 0.85
COS_fi 0.87
Frekvencia 10.1
THD_fesz 21.7
THD_áram 76.5

Ugyanekkor létrejön az archívumok könyvtára is. Jelen esetben: *Napi_2018.04.16* néven.
Ez tartalmazza az egyes szimulált jelek archívumait, példánkban:

COS_fi.csv
Fázis_áram_R.csv
Fázis_fesz_R.csv
Frekvencia.csv

Teljesítmény_R.csv
THD_fesz.csv
THD_áram.csv

Az archív fájlok tartalma például így néz ki (*Fázis_fesz_R.csv*):

10:33:15	185
10:33:35	195
10:33:55	183
10:34:15	187
10:34:35	203
10:34:55	182
10:35:15	223
10:35:35	191
10:35:55	231
10:36:15	215
10:36:35	246

Ha mégis azt szeretnénk, hogy az adatbázis leíróban szereplő valamennyi jelet generálja a rendszer, akkor a *SymulatorAll.py* programot kell elindítani. Ezt akkor használjuk, ha nem akarjuk a feldolgozó modulokat tesztelni, viszont pl. a megjelenítőben látni akarjuk valamennyi adtabázis jelet. A kliens könyvtár szerkezete megegyezik az előzőekben látottakkal, csak több jelet tartalmaz.

Adat küldés a felhőbe

A *Put_data.py* modul feladata a kliens adatbázisában keletkező jeleket kódolt formában eljuttatni a felhő adatbázisába. A modul inicializálás során FTP kapcsolatot alakít ki a (program login sorában) specifikált internet címmel. Sikeres kapcsolatfelvétel után 20 másodpercenként átmásolja a rel-time adatokat a felhőbe. Az megfelelő adatbiztonság elérése érdekében a jeleket kódolt formában küldi a felhő adatbázisba és ott kódolt formában is tárolja. Ha mozgatás során ideg kézbe kerül is az adat, a kulcs ismerete nélkül azzal nem tud semmit kezdeni. Esetünkben egy online adatcsomag például a következő formában közlekedik a NET-en:

neve:

b022d1c0396c6bc2df538d140605ffa025bb9e103c01f8f8c553410abf549f6c.dat

tartalma:

d27006ecb27edce4dba12ed15e23d2e525bb9e2d3c01dcf8c584410a01549f6c
4ccd12672748b38147397b1e1aa018d2
4cb7f2672748b381d4397b1e1aa018c7

4cdb12672748b3813e397b1e1aa01871
4ccd4f672748b381db3973861addff7025bb9e103c01f8f8c58b410a5e549f6c
4cb74f672748b381863973861addff7025bb9e103c01f8f8c58b410acb549f6c
4cdb06672748b381a73973861addff7025bb9e103c01f8f8c58b410a27549f6c
b0a8aab14e422b7f4754f3e9da44386e25bb9e103c015cf8c5f6410afd549f6c
b0a88ab14e422b7f4754f3e9da44386e25bb9e103c01f5f8c5d8410afd549f6c
b0a876b14e422b7f4754f3e9da44386e25bb9e103c018bf8c58c410afd549f6c
4c889e5602013900c55434adeffb5f6c
4c559e56a1013900c55434adefd35f6c
4c669e568a013900c55434adef8d5f6c
9ebb9e8fdc015e9ec5dbb3727830426c
ec297941f88b3e0bc529f8771aa33563
ec729e8fe5dfe19ec551bef1da99786c
f7729e8f8ab6e131c5513e51daa1786c
b029e83ef8e0b82d01fc59441a75181225bb9e103c01f8f8c58b410a14549f6c
b06a4f3ef88bb82d01fc596c1a75187125bb9e103c01f8f8c58b410a14549f6c
b029813ef88bb82d01fc59b21a75184b25bb9e103c01f8f8c58b410a6c549f6c
b06a4f3ef8f4b82d01fc59dd1a75186a25bb9e103c01f8f8c58b410aa4549f6c
b066323ef8dfb82d01fc59f11a75181125bb9e103c01f8f8c58b410a01549f6c
b029123ef8dfb82d01fc59d11a75181125bb9e103c01f8f8c58b410a5e549f6c
2dbb9ea01f017081c5d7afff06ffd26c
2dbb9ea085017081c5d7afff06ff7c6c

Adat lekérdezés a felhőből

Az eddig tárgyalt modulok a kliensen futottak a *Get_data.py* modul a szerveren (megjelenítő gépen) fut, feladata a felhőben ciklikusan frissülő adatok kiolvasása, dekódolása és a szerver adatbázisába helyezése. A program az indítást követő inicializálás (FTP kapcsolatteremtés) után, folyamatoson figyeli a real-time adatok frissülését. Ha új adatok érkeznek, azokat azonnal feldolgozza.

A *Server_Data* könyvtárban található *Pillanat* alkönyvtárba kerülnek a real-time adatok jelenként. Ezzel párhuzamosan frissülnek az archív jelek is *Napi_2018.04.16* könyvtárban. A modul másodpercenként fut le, és ezen belül veszi észre a jelváltozást. A húsz másodperces ciklusidejű mérés (vagy szimuláció) esettén ez egy maximum 1 sec-os holtidőt okozhat a jelfeldolgozásban.

Időszinkronizálás

Mivel a kliens gép autonóm módon működik, ezért a mérési eredmények és események időbélyegjeinek pontosságát a gép belső órája határozza meg. A használt ipari PC-k ben viszonylag pontos órák található, ennek ellenére, hosszabb időtávon ezek is késhetnek vagy siethetnek pár másodpercet. Ezeknek a

csúszásoknak a kiküszöbölésének érdekében kifejlesztettünk egy időszinkronizáló algoritmust. A szerver pontos időt küld a kliensek felé

Közvetlen adatmásolás

Ha az adatgyűjtő (kliens) gépen is szeretnénk megjeleníteni a mért adatokat akkor nem kell feltétlenül „megjáratni" azokat a felhőn keresztül, közvetlenül is hozzá tudunk férni a *Copy_data_to_server.py* modul segítségével. Ebben az esetben az adatgyűjtő gépen is létre kell hozni a *#Server* könyvtárat a fejezet elején leírt módon. Ha itt is rendelkezésre áll a *Server_Data* könyvtár, akkor a kliens könyvtárban el kell indítani a fenti modult (*Copy_data_to_server.py)*, mely egyszerűen átmásolja az adatokat a szerver könyvtárba, oly módon és ugyanolyan ciklusidővel, mintha a bonyolult kódolt adatátvitelen keresztül érkeztek volna. Ettől kedve a következő pontokban ismertetett módon tudjuk megjeleníteni azokat.

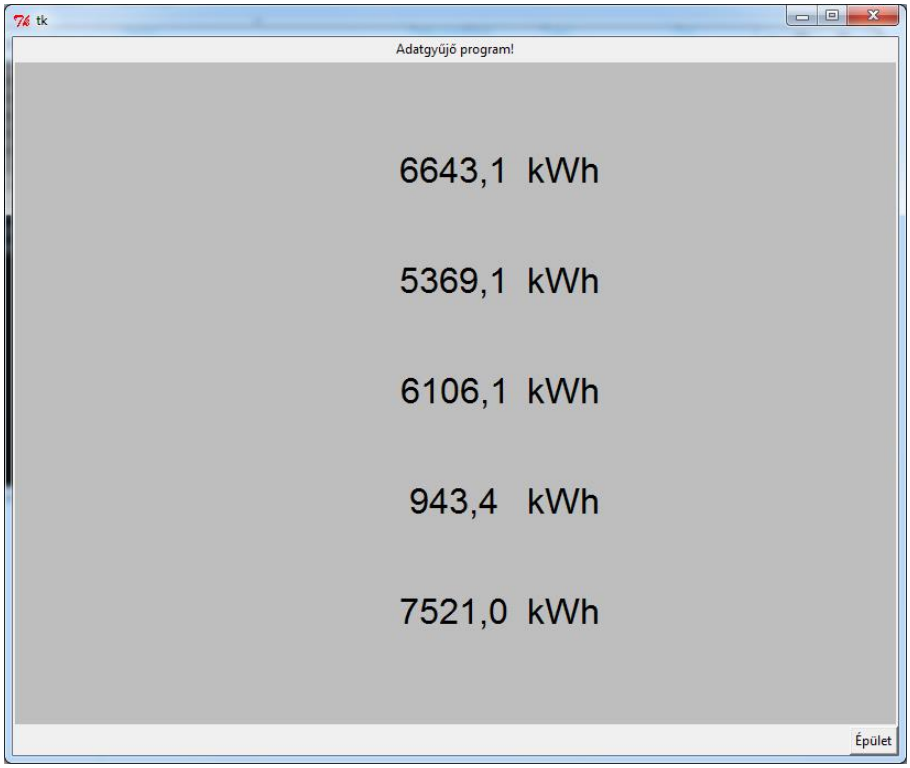
Képek szerkesztése (megjelenítés)

Ahhoz hogy a jeleket megjelenítsük, képeket kell készítenünk. A képek helye a *#Server* könyvtár *Pict* alkönyvtára. A képeket Megjelenítés fejezetben tárgyalt módon kell elkészíteni. A tutorial 4 képet tartalmaz, ennek megfelelően négy csv kiterjesztésű képleíró fájl szerepel a Pict könyvtárban.

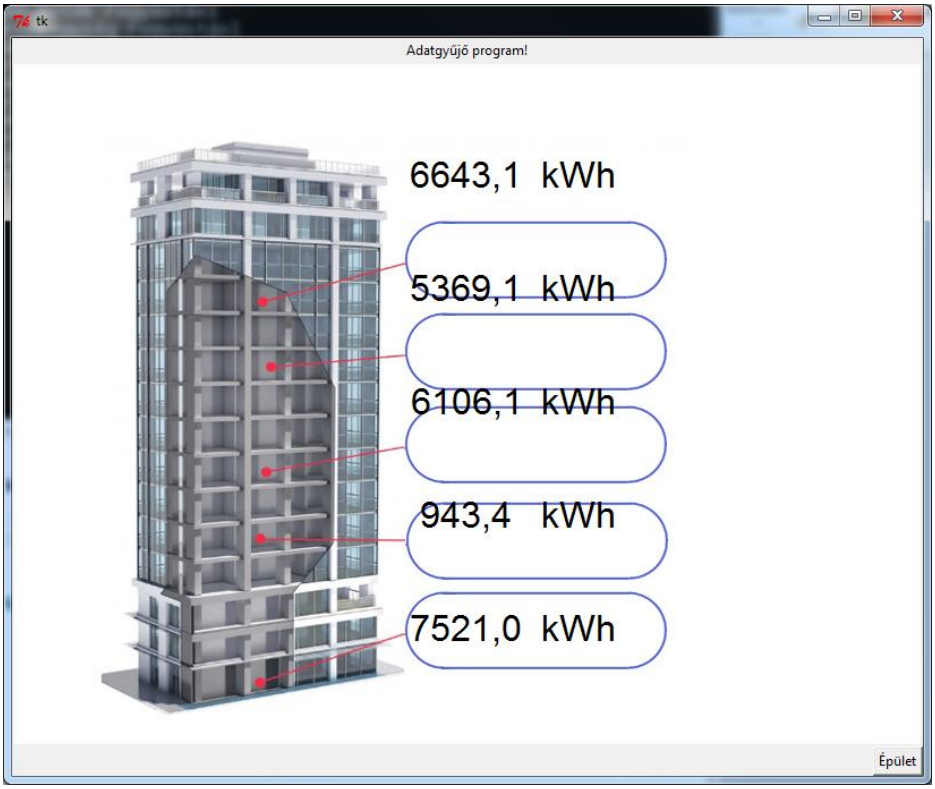
Az első kép legyen egy egyszerű épületfelügyelet, mely mindössze 5 fogyasztásmérést tartalmaz. A jeleket egymás alatt nagy számokkal, a mértékegységeikkel együtt szeretnénk ábrázolni. Töltsük ki az *Épület.csv* fájlt a következő módon:

Tipus	tag_name	Pos_x	Pos_y	szin	tulajdonság
szám_érték	Fogyasztás1	400	100	black	nagy
mértékegység	Fogyasztás1	500	100	black	nagy
szám_érték	Fogyasztás2	400	200	black	nagy
mértékegység	Fogyasztás2	500	200	black	nagy
szám_érték	Fogyasztás3	400	300	black	nagy
mértékegység	Fogyasztás3	500	300	black	nagy
szám_érték	Fogyasztás4	400	400	black	nagy
mértékegység	Fogyasztás4	500	400	black	nagy
szám_érték	Fogyasztás5	400	500	black	nagy
mértékegység	Fogyasztás5	500	500	black	nagy

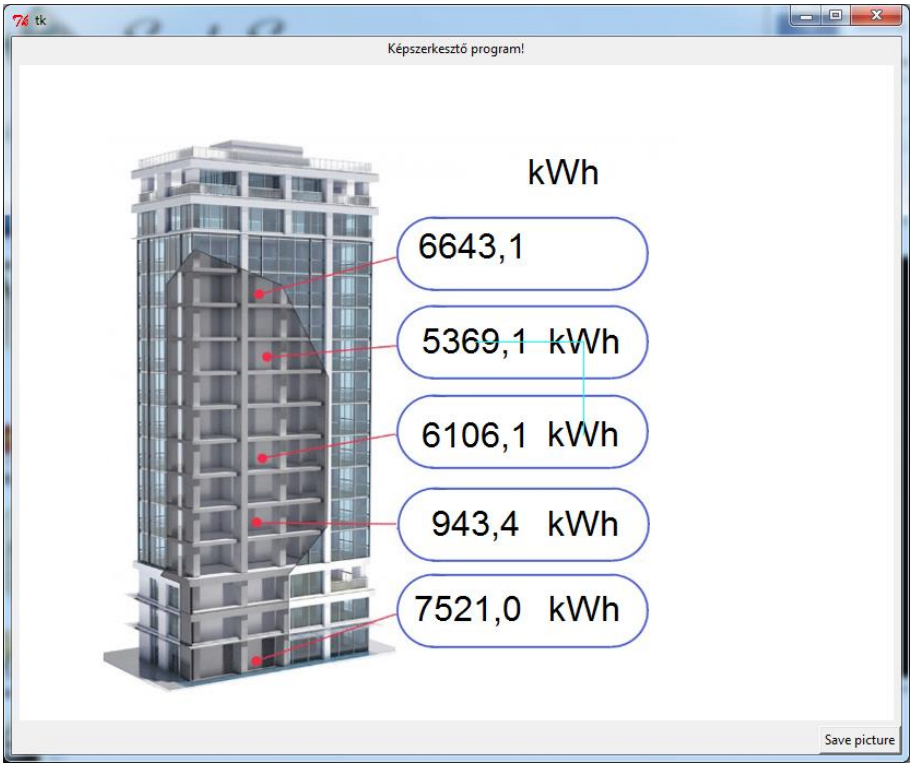
Az koordinátákat tetszőlegesen vegyük fel vigyázva, hogy a képméretnek megfelelően a látható tartományba (jelen esetben 800x600) essen. A kép leíró elmentése után indíthatjuk a megjelenítő programot. A következő ábrán látható eredményt kapjuk. (Azért nem inicializálási nullákat látunk, mert korábban futott a szimuláció.)



Tetszőleges képszerkesztővel rajzoljuk meg az *Épület.gif* fájlt, és másoljuk a *Pict* könyvtárba. Most elindítva a megjelenítőt a következő képet fogjuk látni:

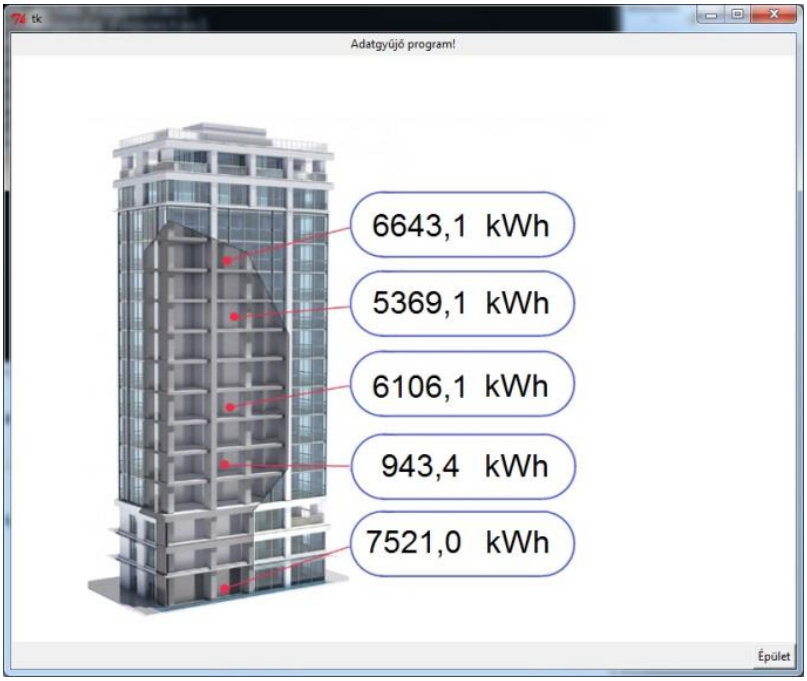


Most már csak annyi a teendő, hogy a képszerkesztő programmal (*Editor.py*) a helyükre mozgassuk a grafikus objektumokat, majd a *Save picture* gombbal elmentsük a képet:



Az objektumok pozicionálását segíti a függőleges és vízszintes türkiz vonalak, amelyek akkor jelennek meg a képernyőn, ha a mozgatott objektum egy vonalba kerül valamelyik másik objektummal.

Az eredmény grafikusan:



és táblázatosan:

Tipus	tag_name	Pos_x	Pos_y	szin	tulajdonság
szám_érték	Fogyasztás1	419	176	black	nagy
mértékegység	Fogyasztás1	519	176	black	nagy
szám_érték	Fogyasztás2	419	255	black	nagy
mértékegység	Fogyasztás2	519	255	black	nagy
szám_érték	Fogyasztás3	419	340	black	nagy
mértékegység	Fogyasztás3	518	340	black	nagy
szám_érték	Fogyasztás4	419	422	black	nagy
mértékegység	Fogyasztás4	518	422	black	nagy
szám_érték	Fogyasztás5	412	500	black	nagy
mértékegység	Fogyasztás5	519	500	black	nagy

Grafikus objektumok

Készítsünk egy újabb képet (*Teszt.csv*), melyen egy kapcsoló és egy szelep található. Az első viselkedését a *Kapcsoló1,* a másodikat a *Kapcsoló2* jel állapota határozza meg. A két jel értékét számmal a másodikat oszlopdiagrammal is jelenítsük meg!

A *teszt.csv* képleíró a következő módon nézhet ki:

Tipus	tag_name	Pos_x	Pos_y	szin	tulajdonság
graf_obj	Kapcsoló1	147	78	kapcs	0-1
szám_érték	Kapcsoló1	91	78	black	közepes
graf_obj	Kapcsoló2	146	169	kepecske	0-2
szám_érték	Kapcsoló2	91	169	black	közepes
v_bar_érték	Kapcsoló2	197	165	white	kicsi

A *graf_obj* sorokban szereplő „*kapcs*" és „*kepecske*" objektumokat el kell készítenünk:


kapcs_0.gif

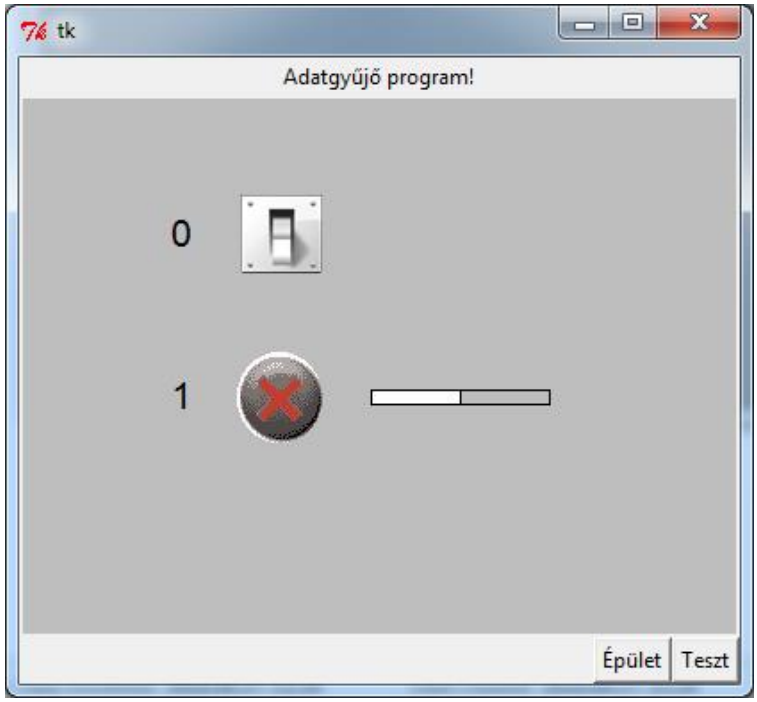

kapcs_1.gif


kepecske_0.gif

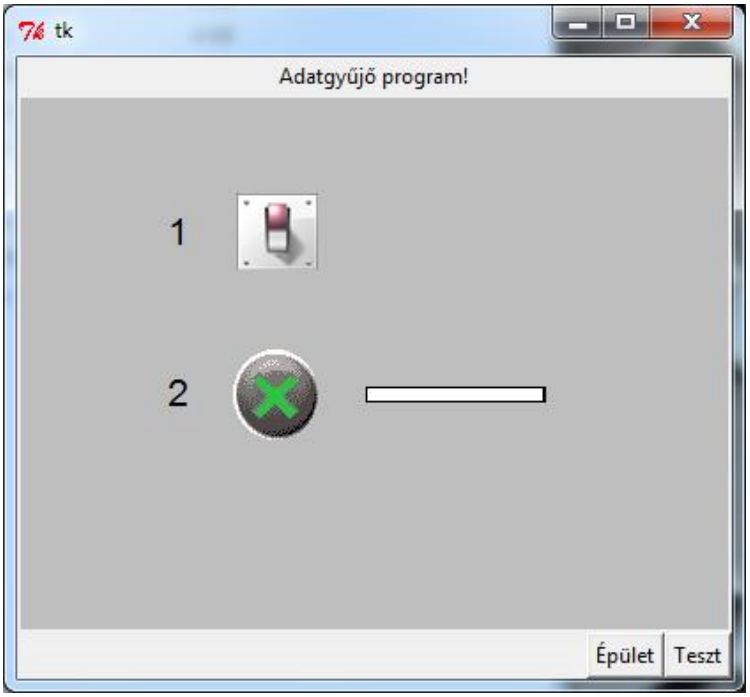

kepecske_1.gif


kepecske_2.gif

Most elindítva a megjelenítőt a következő képet fogjuk látni:



Ha a két jel változik, akkor a grafikus objektumok képe is megváltozik:



Adatfeldolgozás (matematika modul)

A fenti példában a két- illetve három-állapotú jelek előállítására a matematika modult használjuk:

```
trig =  db.GET("Kapcsoló1")
if trig == 0: trig = 1
else: trig = 0
db.PUT("Kapcsoló1",trig)

trig =  db.GET("Kapcsoló2")
if trig < 2: trig = trig +1
else: trig = 0
db.PUT("Kapcsoló2",trig)
```

Az első algoritmus a *Kapcsoló1* jelet felváltva 0-ba és 1-be állítja a második a *Kapcsoló2* jelnek 0, 1, 2 értéket ad. Mivel a *Matek.py* program öt másodperces ciklusidővel újraszámítja a képleteket, ezért a fenti képletek eredményeként a két jel öt másodpercenként új értéket vesz fel.

A következő egyszerű számtanpéldán bemutatjuk, hogyan kell a matematika programban kiszámítani a meddő teljesítményt, ha mérjük a feszültséget, az áramot és teljesítménytényezőt. Először kiolvassuk az adatbázisból a három jel pillanatértékét:

```
Ur = db.GET("Fázis_fesz_R")
Ir = db.GET("Fázis_áram_R")
cos = db.GET("COS_fi")
```

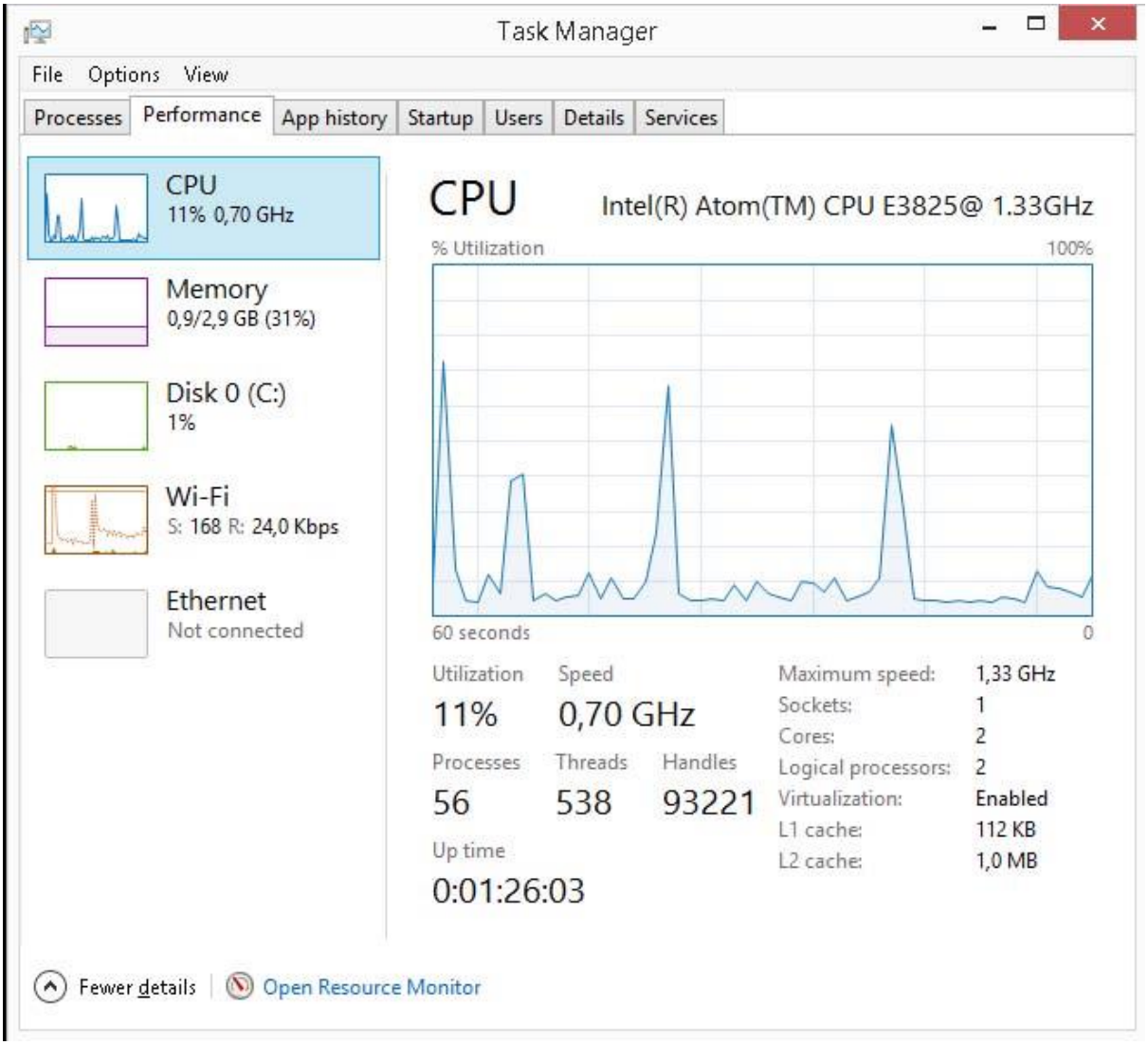
Majd elvégezzük a műveletet, és az eredményt beírjuk az adatbázisba:

```
db.PUT("Meddö_R",Ur*Ir*cos/1000)
```

Tesztek

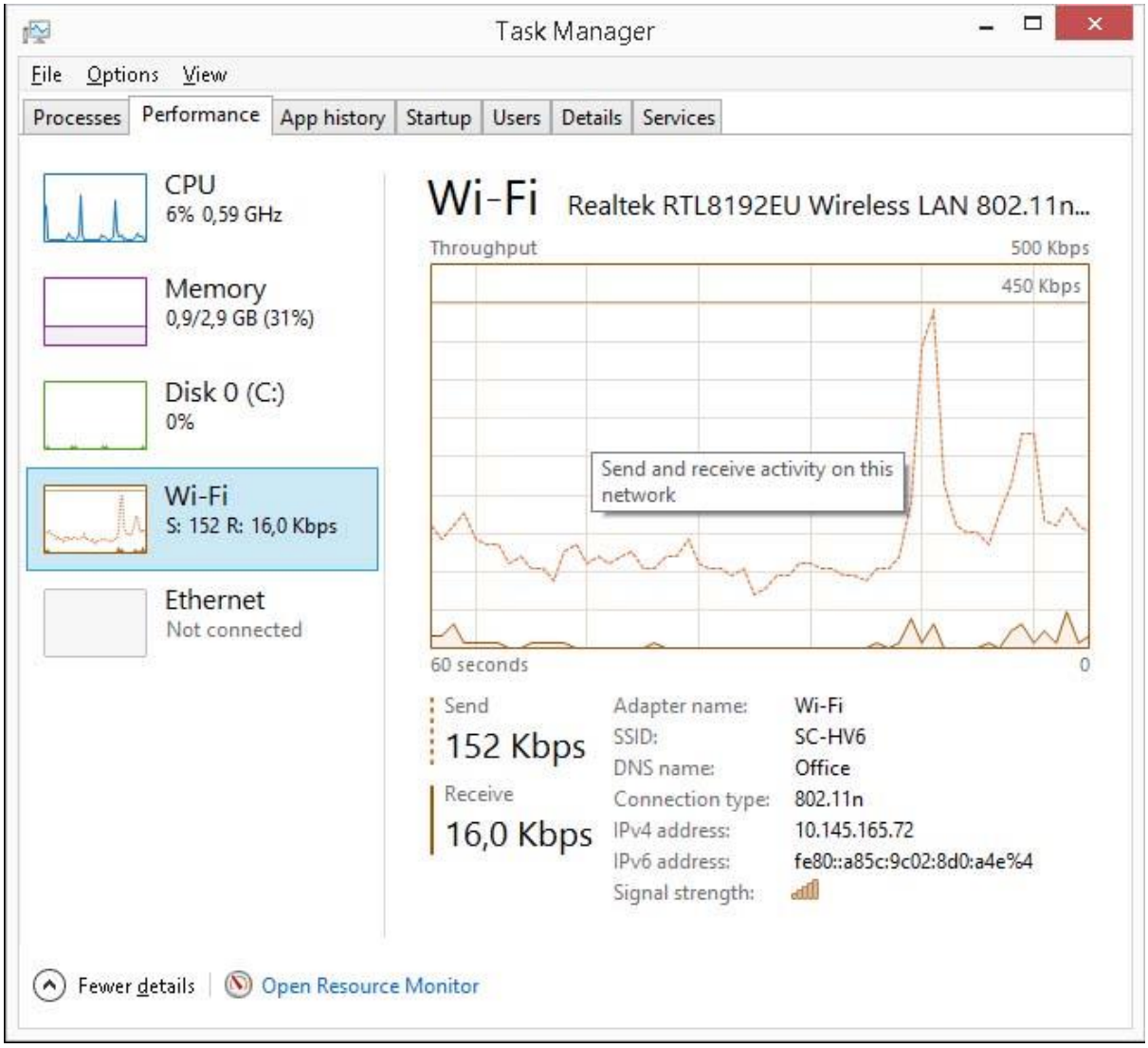
Sebesség tesztek

Megvizsgáltuk az adatgyűjtő ipari PC terheltségi állapotát adatgyűjtés közben, és vizsgáltuk az adatátvitel sebességét. Tesztelésre a Task Managert használtuk.

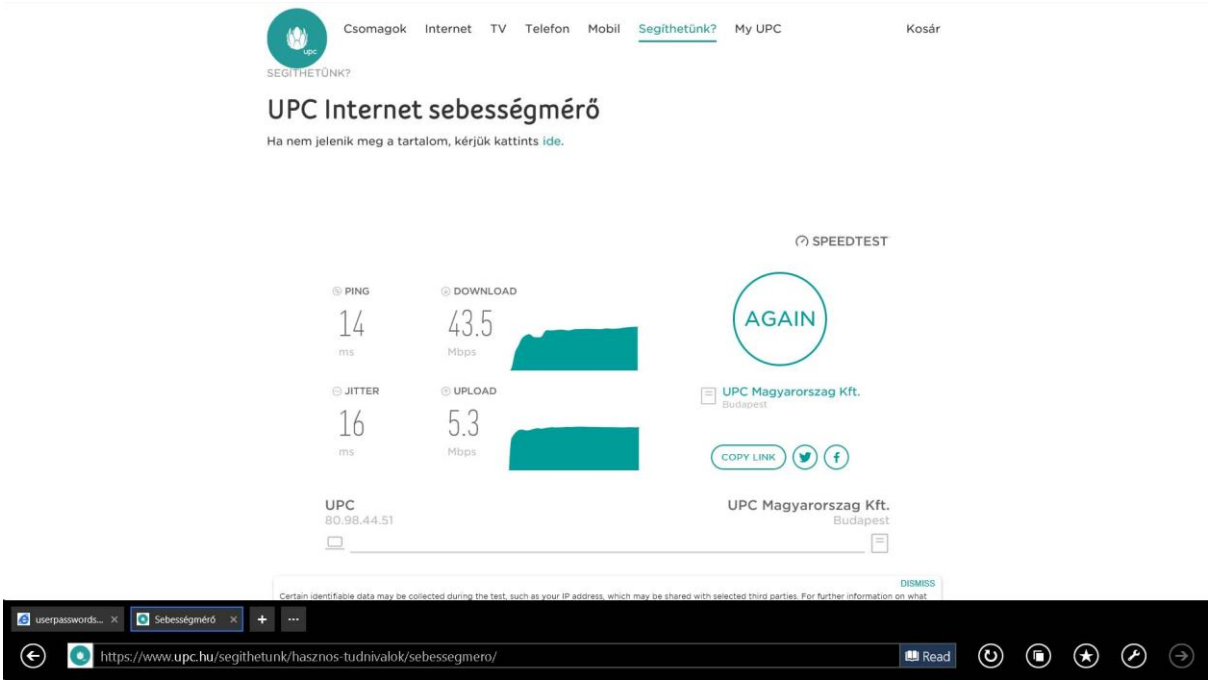


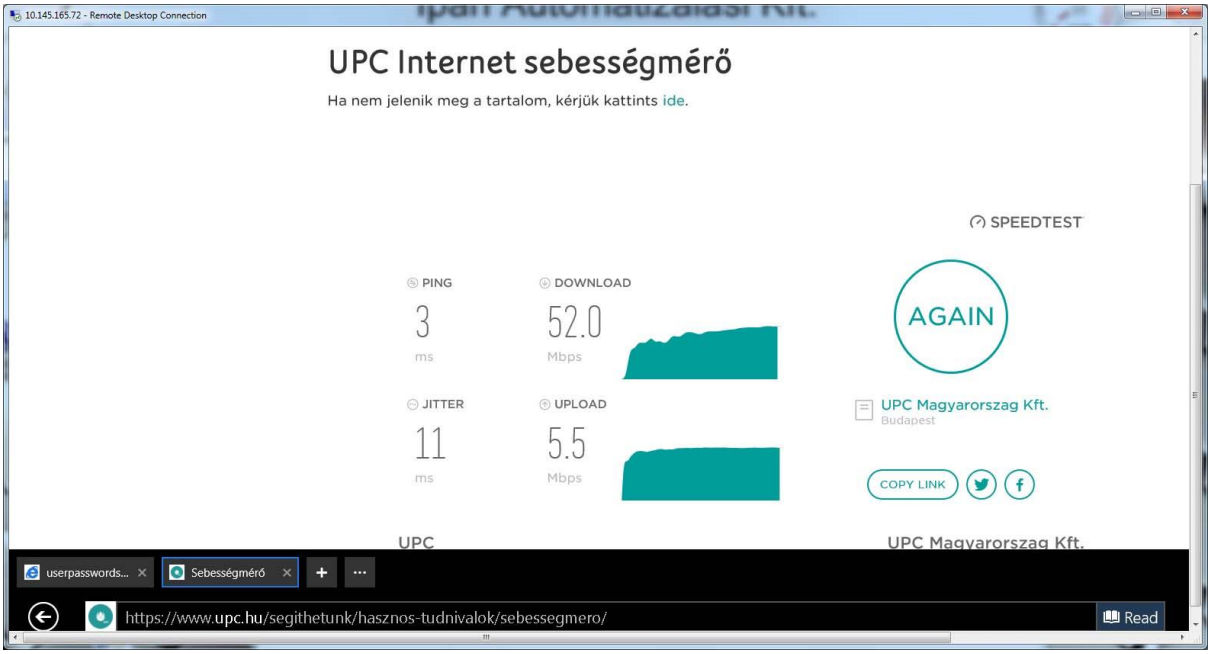
A processzor átlagosan 5-10%-os kihasználtsággal üzemel. Tesztelés alatt az adatgyűjtés 20 másodperces ciklusidővel zajlott. Ezek a csúcsok jól láthatóak a trend diagramon. A mérések igazolják, hogy a kiválasztott eszköz alkalmas a feladat ellátására, elegendő erőforrás tartalékkal rendelkezik nagyobb jelszámú technológia mérésére és feldolgozására is.

Az adatgyűjtő és a felhő között az adatátvitel Wi-Fi hálózaton keresztül történt. Méréseink szerint az átviteli sebesség adásra 160 kbps, vételre 20 kbps körül mozgott. Az általunk használt Gembird USB WiFi adpater 300 Mbps sebességre képes, ezért feltétlenül alkalmas a feladat ellátására. Az estlegesen jelentkező csúcsok is nagyságrenddel alatta maradtak az elvi átviteli sebesség határnak.



Az adatátvitel sebességének mérését megismételtük a UPC Internet sebességmérő programjával is, és ezzel is nagyon hasonló értékeket kaptunk:





Adatforgalom tesztek

Az elkészült prototípuson elvégeztük az alapvető on-line teszteket. A teljes adatútvonalat teszteltük a méréstől a megjelenítésig. A mérő a rákötött fogyasztókat illesztette az adatgyűjtőhöz. A fogyasztási adatokat az adatgyűjtő ciklikusan mérte (20 sec/ciklus), és tárolta a kliens adatbázisban. Elsőként a kliens adatbázisba került adatok helyességét ellenőriztük. Az „Aktual.csv” fájlban

szereplő adatokat vizsgáltuk érték és formátum szempontjából. Minden adat helyesnek bizonyult:

2019.02.25 ;13:40:20

Fázis_fesz_R;225

Fázis_áram_R;0.108

Teljesitmény_R;0.014

Meddö_R;0.021

COS_fi;0.57

Frekvencia;50.1

THD_fesz;13.9

THD_áram;15.8

Átlag_meddö_R;0,471

Fogyasztás1;1036,2

Az adatgyűjtő Wi-Fi hálózaton keresztül küldte a mért jeleket a felhőbe. Az ott lévő adatokat egy asztali PC-n keresztül kérdeztük le, és jelenítettük meg.

Egyfázisú teljesítménymérést végeztünk különböző terhelések mellett. Ellenőriztük az adatok helyességét értékre, formátumra.

1

tag_name	Méréstart.	unit	típus	Cim	Időpont	Mért érték
Fázis_fesz_R	0-250	V	float	1/100	2019.02.25 13:40:20'	**225**
Fázis_áram_R	0-1	A	float	1/120	2019.02.25 13:40:20'	**0**
Teljesitmény_R	0-1	kW	float	1/146	2019.02.25 13:40:20'	**0**
Meddö_R	0-1	kVA	float	1/14E	2019.02.25 13:40:20'	**0**
COS_fi	0-1	-	float	1/134	2019.02.25 13:40:20'	**0**
Frekvencia	0-100	Hz	float	1/142	2019.02.25 13:40:20'	**50**
THD_áram	0-100	%	float	1/188	2019.02.25 13:40:20'	**0**

2

tag_name	Méréstart.	unit	típus	Cim	Időpont	Mért érték
Fázis_fesz_R	0-250	V	float	1/100	2019.02.25 13:40:20'	**225**
Fázis_áram_R	0-1	A	float	1/120	2019.02.25 13:40:20'	**0,108**

Teljesitmény_R	0-1	kW	float	1/146	2019.02.25 13:40:20'	**0,014**
Meddö_R	0-1	kVA	float	1/14E	2019.02.25 13:40:20'	**0,02**
COS_fi	0-1	-	float	1/134	2019.02.25 13:40:20'	**0,57**
Frekvencia	0-100	Hz	float	1/142	2019.02.25 13:40:20'	**50,1**
THD_áram	0-100	%	float	1/188	2019.02.25 13:40:20'	**40,6**

3

tag_name	Méréstart.	unit	típus	Cim	Időpont	Mért érték
Fázis_fesz_R	0-250	V	float	1/100	2019.02.25 13:55:44'	**226**
Fázis_áram_R	0-1	A	float	1/120	2019.02.25 13:55:44'	**0,42**
Teljesitmény_R	0-1	kW	float	1/146	2019.02.25 13:55:44'	**0,078**
Meddö_R	0-1	kVA	float	1/14E	2019.02.25 13:55:44'	**0,052**
COS_fi	0-1	-	float	1/134	2019.02.25 13:55:44'	**0,83**
Frekvencia	0-100	Hz	float	1/142	2019.02.25 13:55:44'	**50**
THD_áram	0-100	%	float	1/188	2019.02.25 13:55:44'	**15,8**

4

tag_name	Méréstart.	unit	típus	Cim	Időpont	Mért érték
Fázis_fesz_R	0-250	V	float	1/100	2019.02.25 14:00:06'	**224**
Fázis_áram_R	0-1	A	float	1/120	2019.02.25 14:00:06'	**0,846**
Teljesitmény_R	0-1	kW	float	1/146	2019.02.25 14:00:06'	**0,182**
Meddö_R	0-1	kVA	float	1/14E	2019.02.25 14:00:06'	**0,051**
COS_fi	0-1	-	float	1/134	2019.02.25 14:00:06'	**0,96**
Frekvencia	0-100	Hz	float	1/142	2019.02.25 13:55:44'	**50,1**
THD_áram	0-100	%	float	1/188	2019.02.25 14:00:06'	**7,4**

5

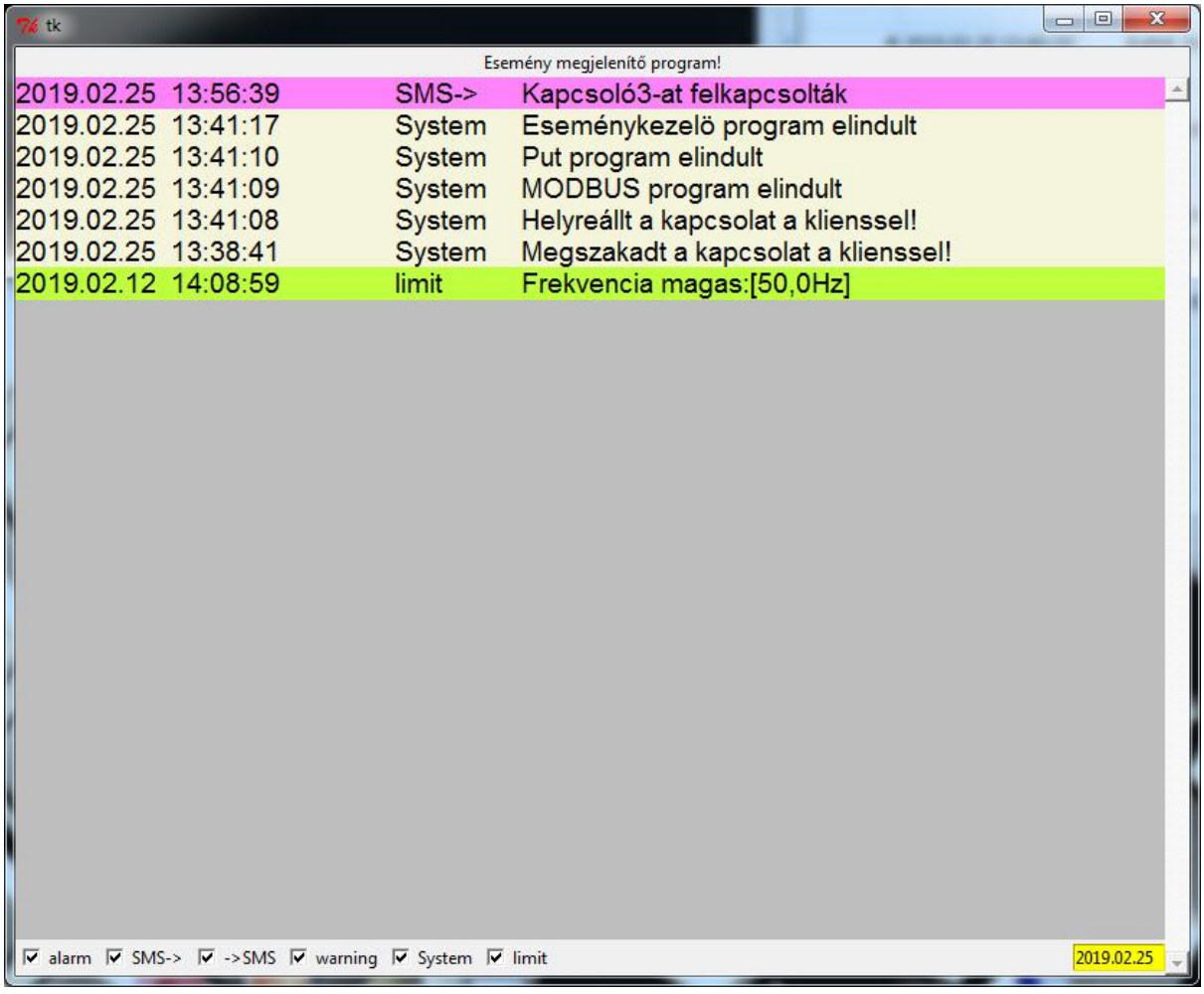
tag_name	Méréstart.	unit	típus	Cim	Időpont	Mért érték
Fázis_fesz_R	0-250	V	float	1/100	2019.02.25 14:02:34'	**224**
Fázis_áram_R	0-1	A	float	1/120	2019.02.25 14:02:34'	**0,976**
Teljesitmény_R	0-1	kW	float	1/146	2019.02.25 14:02:34'	**0,216**
Meddö_R	0-1	kVA	float	1/14E	2019.02.25 14:02:34'	**0,032**
COS_fi	0-1	-	float	1/134	2019.02.25 14:02:34'	**0,99**
Frekvencia	0-100	Hz	float	1/142	2019.02.25 14:02:34'	**50**
THD_áram	0-100	%	float	1/188	2019.02.25 14:02:34'	**4,4**

1. Az első esetben nem volt fogyasztó bekapcsolva. Ekkor feszültségen és frekvencián kívül minden érték nulla volt.
2. Második esetben a szabályozható kapcsolót kb. fél állásba forgatva mértük a fogyasztási értékeket. Mint a táblázatból látható, megjelent az áram és teljesítmény érték is, sőt meddő teljesítményt és THD áramot is tudtunk mérni. Ennek az a magyarázata, hogy a kapcsoló üzemű feszültség szabályozás szaggatja a szinuszos jeleket. Az is jól kiolvasható, hogy a cos fi is nagyon rossz ebben az üzemmódban.

3. Harmadik esetben bekapcsoltunk egy újabb fogyasztót, egy LED-es izzót. Ez tovább növelte az áramot és a teljesítményt. A cos fi jelentősen javult és a THD áram is kisebb lett.
4. Negyedik esetben újabb fogyasztót nem kapcsoltunk be, hanem a szabályozót tekertük fel a végállásig. (Teljes terhelés esete.) Tovább nőtt az eredő áram, továbbá a hatásos teljesítmény. Értelem szerűen a meddő teljesítmény nem változott.
5. Ötödik esetben újabb fogyasztót (hagyományos izzót) kapcsoltunk be, mely tovább növelte az eredő áramot, a hatásos teljesítményt. Ebben az esetben legjobb a cos fi, és a THD áram is tizede a második esetben mértnek.

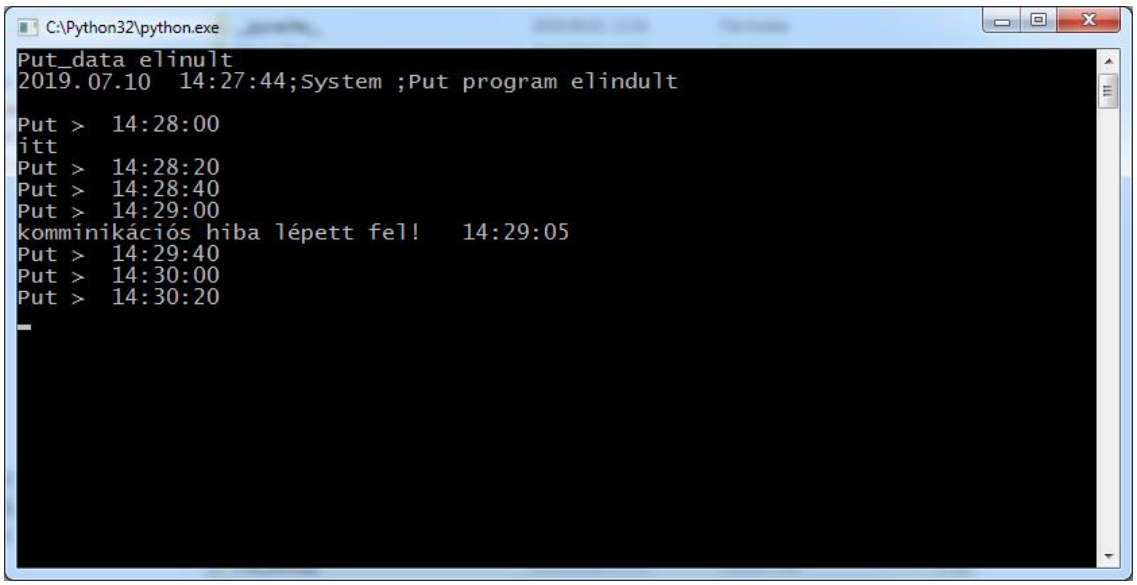
Összefoglalva a mérési eredményeket, igazoltuk, hogy a prototípus helyesen méri a fogyasztási adatokat. Az adatátviteli csatornán átküldött és megjelenített adatok hibátlanok.

A következő ábrán a mérések közben készült eseménynapló látható:



Adatátvitel tesztelése

Meg kell vizsgálni, hogy mi történik abban az esetben, ha az adatátvitel megszakad a kliens és a „felhő” között. Ez azt jelenti, hogy valamilyen ok miatt nem működik az FTP kapcsolat. A vizsgálat célja ez, hogy ellenőrizzük, hogy egy hiba fellépése után, a hiba elhárulásával, automatikusan helyre áll a az adatkapcsolat. Ennek érdekében egy kábel kihúzásával szándékosan megszakítjuk a kommunikációt, megvárjuk, míg a rendszer észleli a hibát, majd a kábel visszahelyezése után megnézzük az eredményt.

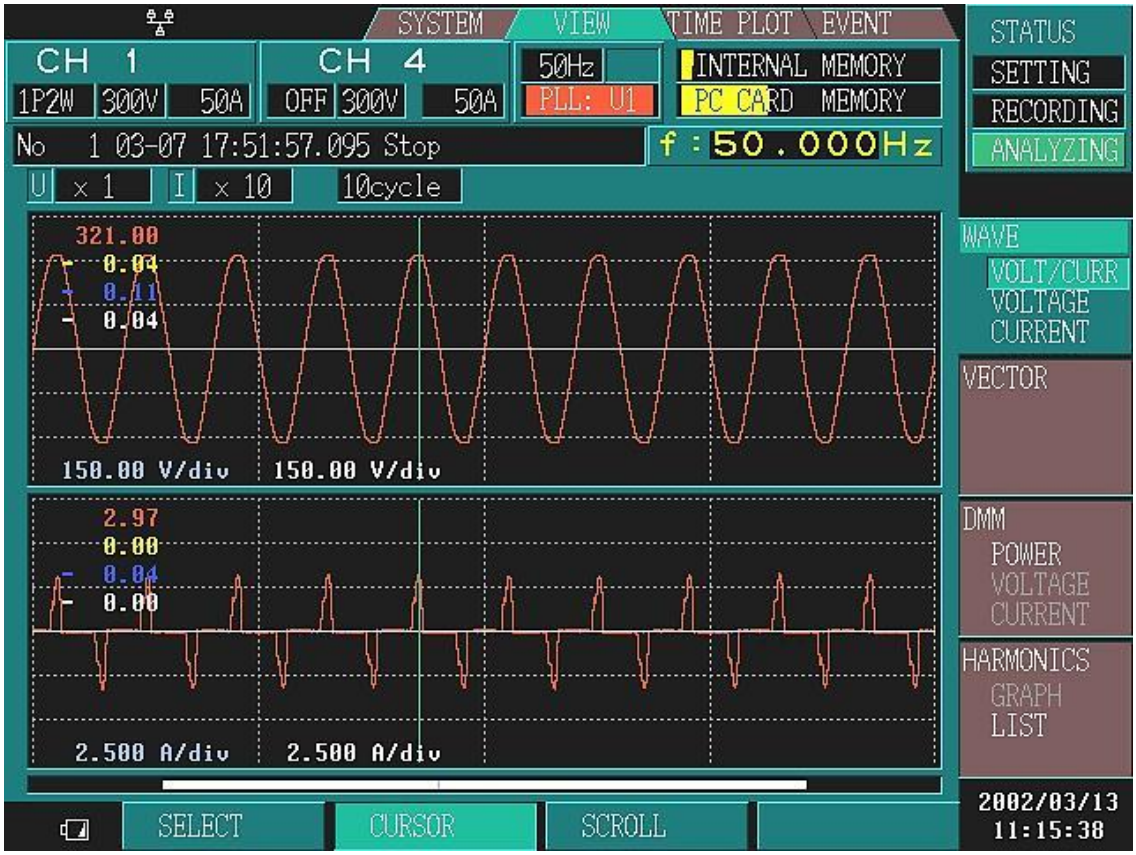


A **put.py** programot 14:27:44-kor indítottuk el, megvártunk három 20 másodperces ciklust, majd 14:29:00-kor elvágtuk az adatátviteli csatornát. A képernyő ábrából látszik, hogy a program 5 másodperc múlva vette észre a hibát. Ezt követően helyreállítottuk a kommunikációt, amit követően, 14:29:40-kor ismét elkezdte küldeni az adatokat a PUT program. Semmilyen külső beavatkozásra nem volt szükség.

A felvételből az is látszik, hogy mindössze egy ciklus maradt ki az adatküldésből. Ez egy ilyen jellegű technológiánál semmilyen problémát nem okoz.

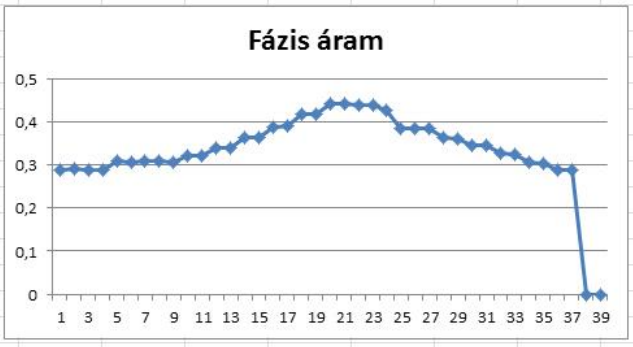
Nemlineáris fogyasztók tesztelése

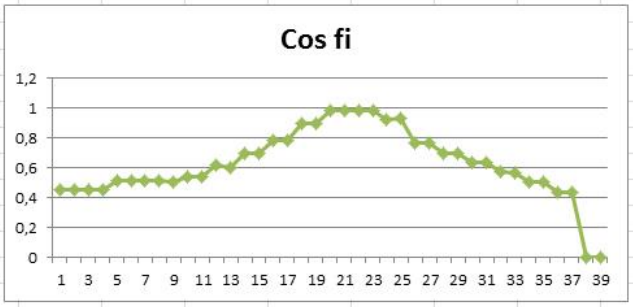
E tesztelés során azt vizsgáljuk, hogy milyen hatással van a villamos hálózatokra az egyre szaporodó nemlineáris eszközök használata. A prototípus segítségével teszteltük, hogy a kereskedelemben kapható fényerő szabályozó kapcsoló milyen módon torzítja a hálózat jellemzőit. Mérjük a fázis feszültséget, fázis áramot, cosfi-t és az áram felharmonikus tartalmát (Thd), miközben a fényerő szabályozót a minimumról a maximumig és vissza, mozgatjuk. Az alábbi ábrából jól látszik, hogy a szinuszos feszültség mellett az áram az alap összetevőn kívül milyen jelentős felharmonikussal rendelkezik:

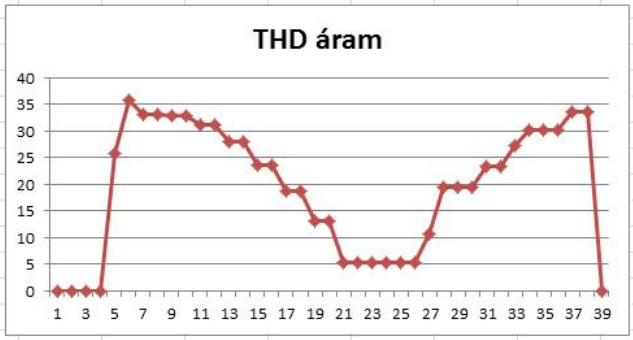


A szerver adatbázisában gyűjtöttük a mérési eredményeket. A mérés mindössze 13 percig tartott, a kapott CSV fájlokat Excelben dolgoztuk fel. A táblázatokon látszanak a mérési eredmények és a trend diagramok:

Idő	Fázis áram	Cos fi	THD áram
12:41:54	0,289	0,45	0
12:42:14	0,29	0,45	0
12:42:34	0,288	0,45	0
12:42:54	0,287	0,45	0
12:43:14	0,309	0,51	25,8
12:43:34	0,307	0,51	35,8
12:43:54	0,309	0,51	33,1
12:44:14	0,309	0,51	33,1
12:44:34	0,307	0,5	32,9
12:44:54	0,321	0,54	32,9
12:45:14	0,32	0,54	31,1
12:45:34	0,339	0,61	31,1
12:45:54	0,338	0,6	27,9
12:46:14	0,364	0,69	27,9
12:46:34	0,362	0,69	23,6
12:46:54	0,388	0,78	23,6
12:47:14	0,389	0,78	18,6
12:47:34	0,416	0,89	18,6
12:47:54	0,418	0,89	13
12:48:14	0,442	0,98	13
12:48:34	0,441	0,98	5,3
12:48:54	0,44	0,98	5,3
12:49:14	0,44	0,98	5,3
12:49:34	0,426	0,92	5,3
12:49:54	0,384	0,93	5,3
12:50:14	0,384	0,76	5,3
12:50:34	0,383	0,76	10,6
12:50:54	0,364	0,69	19,4
12:51:14	0,361	0,69	19,4
12:51:34	0,345	0,63	19,4
12:51:54	0,345	0,63	23,3
12:52:14	0,326	0,57	23,3
12:52:34	0,323	0,56	27,1
12:52:54	0,306	0,5	30,1
12:53:14	0,304	0,5	30,1
12:53:34	0,289	0,43	30,1
12:53:54	0,287	0,43	33,5
12:54:14	0	0	33,5
12:54:54	0	0	0







A fázis feszültség effektív értéke 230 V, jelalakja szabályos szinusz. Az áram effektív értéke 0,29A –ről 0,44A ig növekedett, majd visszacsökkent kezdeti értékre (végül kikapcsoltuk az áramot). A közel háromtized amperes áramnál az izzó elkezdett világítani, és 0,44 ampernél teljes fényerővel izzott. Ha nem vizsgálnánk a fázisszög és a felharmonikus tartalom változását, akkor akár lineárisnak is tekinthetnénk az áramkört. Ha valóban lineáris lenne a kapcsolás, akkor fázisszög (cosfi) konstans egy lenne. Az ábrából jól látszik, hogy a mérés során ez az érték csak a maximális áramnál éri el a kívánt értéket, és szélső esetben eléri a 0,43 értéket is. Másként fogalmazva a hatásos és látszólagos teljesítmény (Pl) aránya követi az áram effektív értékének változását. Kis áramoknál a meddő teljesítmény (Pm) megnövekszik, szélsőséges esetben meg is haladja a hatásos teljesítmény (Ph) értékét.

$$Ph = Pl * cosfi, \text{ vagy } Ph = (Ph+Pm) * cosfi$$

A fentiekből következik, hogy az ilyen jellegű feszültség szabályozás jelentős veszteségeket okoz, ami többletköltségeket okoz a felhasználóknak.

Az áram felharmonikus tartalma is változik a feszültség szabályozás alatt. Ez a görbe is követi az áram változását, maximumát a kis áramoknál éri el.

(THD áram a 2-31 felharmonikusok mértani középértéke és az alapharmonikus abszolút értékének hányadosa.) Az ábrából leolvasható, hogy csak a feszültség szabályozó kikapcsolt állapotában szűnnek meg a felharmonikusok, a THD áram 5,3% és 33,5% között változik. A kis áramoknál érzékelt jelentős felharmonikus jelenlét visszahat a teljes villamos hálózatra, transzformátor túlmelegedést, készülék-zavarást, meghibásodást okozhat. A fenti jelenségek kiküszöbölésére aktív és passzív felharmonikus szűrőket kell alkalmazni.

Mellékletek

Rendszer telepítés

A prototípus használatához telepíteni kell bizonyos programokat a futtató számítógépekre. A kliens és szerver gépeken is Windows 7, Windows 8, vagy Windows 10 operációs rendszer valamelyikének kell lennie. A programok futtatásához telepíteni kell a Python programcsomagot. A **Utils** könyvtárban található a **python-3.6.4.exe** program, melyet futtatva feltelepíti a gépre a python könyvtárat.

A kliens gépen a MODBUS kommunikáció számára telepíteni kell pymodbus programcsomagot és a soros vonal kezeléséhez a pyserial-3-4 csomagot. Mindegyik megtalálható a **Utils** könyvtárban.

A soros vonal kezelő installálása:

- Bemásolni a pyserial-3.4 könyvtárat a kliens gép HD-re
- parancssorból belépni a főkönyvtárba
- elindítani az installálást: *python setup.py install*

(ha a python-t nem találja, akkor be kell írni a teljes elérési útvonalat)

MODBUS installálása:

- Belmásolni a pymodbus-master könyvtárat a kliens gép HD-re
- parancssorból belépni a főkönyvtárba
- elindítani az installálást: *python setup.py install*

(ha a python-t nem találja akkor be kell írni a teljes elérési útvonalat)

A programcsomag hibás ezért installálás után javítani kell:

- a /pymodbus/utilities.py programban ki kell venni a six-re való hivatkozást (#from six import string_types)
- újra kell installálni az előbbiek szerint

SMS küldő, és fogadó modul inicializálása

Az SMS küldő eszközt soros porton keresztül, az úgynevezett 'AT' parancsok segítségével lehet konfigurálni.

```
def Inic():
#     Output:    False = soros portot nem sikerült megnyitni
#                ser = a soros port azonosítója

# soros port inicializálás
    try:
       ser = serial.Serial('COM2', 9600, timeout=0)
    except:
        print("Soros vonal nem nyitható meg")
        return(False)
    if ident(ser) == False:
        print("Hibas készülék, vagy vonalhiba!")
        return(False)
    else:
        print("ok")
        ser.write(b'ATE\r\n') # Text mód beállítás
        print("Text mód beállítás")
        sz = várakozas(ser)
        if sz != 'OK': print("Hibás válasz:", sz)

        ser.write(b'AT+CMGF=1\r') # Üzenet formátum beállítás
        ser.write(b'chr(10)') # \n
        print("Üzenet formátum beállítás")
        sz = várakozas(ser)
        if sz != 'OK': print("Hibás válasz:", sz)

        ser.write(b'AT+CNMI=2,1\r\n') # SMS jelzésmód beállítás
        print("SMS jelzésmód beállítás")
        sz = várakozas(ser)
        if sz != 'OK': print("Hibás válasz:", sz)

        ser.write(b'AT+CMGD=1\r\n') # Clearbox küldés
        print("Clearbox küldés")
        sz = varakozas(ser)
        if sz != 'OK': print("Hibás válasz:", sz)

        ser.write(b'AT+CSCS = "UCS2"\r\n') # UTF váltás
        print("UTF váltás")
        sz = varakozas(ser)
        if sz != 'OK': print("Hibás válasz:", sz)
        return(ser)
```

Program listák (kliens)

Atlag.py

 # Átlagoló és Archiváló modul

```
import os
from time import sleep
import db

db.Inic() # Feltöltjük az adatformátumot

# itt kezdődnek az algoritmusok
print("Atlagolo, archíváló   program elindult")

# globális változók
ciklus = 5                    # 5 másodpercenkén fut le
negyed_szamlalo = 60/ciklus * 15  # negyed óra
oras_szamlalo = 3600/ciklus       # óra

puffer_meddo_R = 0
puffer_meddo_S = 0
puffer_meddo_T = 0
puffer_telj_R = 0

negyed_trigger = negyed_szamlalo
oras_trigger = oras_szamlalo

while(True):
    db.Fill_data()
    #   Átlag számítás
    puffer_meddo_R = puffer_meddo_R + db.GET("Meddö_R")
    puffer_meddo_S = puffer_meddo_S + db.GET("Meddö_S")
    puffer_meddo_T = puffer_meddo_T + db.GET("Meddö_T")
    negyed_trigger = negyed_trigger - 1
    if negyed_trigger == 0:
        print("negyed",puffer_meddo_R,negyed_szamlalo)
        db.PUT("Átlag_meddö_R",puffer_meddo_R/negyed_szamlalo)
        db.PUT("Átlag_meddö_S",puffer_meddo_S/negyed_szamlalo)
        db.PUT("Átlag_meddö_T",puffer_meddo_T/negyed_szamlalo)
```

```
    puffer_meddo_R = puffer_meddo_S = puffer_meddo_T = 0
    negyed_trigger = negyed_szamlalo
#   Archiválás
puffer_telj_R = puffer_telj_R + db.GET("Teljesitmény_R")
oras_trigger = oras_trigger - 1
if oras_trigger == 0:
    elozo = db.GET("Fogyasztás1")
    print("ora",elozo,puffer_telj_R)
    db.PUT("Fogyasztás1",elozo + puffer_telj_R/oras_szamlalo)
    puffer_telj_R = 0
    oras_trigger = oras_szamlalo
sleep(ciklus) #ciklus másodperces várakozás
```

Copy_data_to_server.py

```
# Ez a modul a mért értékeket kódolatlanul átmásolja a server adatbázisába
import  sys, os, time
from time import sleep
from db import Cserel

aktual= "#Kliens\\Kliens_Data\\Aktual.csv"
event= "#Kliens\\Kliens_Event\\Event.csv"
Serv_dirname = "#Server\\Server_Data\\Pillanat"
Serv_dirnapi = "#Server\\Server_Data\\Napi_"

def Kirak(tomb):
    # fájlba menti a friss adatokat

    t = time.localtime(time.time())
    datum = '%4d%1s%02d%1s%02d' % (t[0],'.',t[1],'.',t[2])
    tt= '%02d%1s%02d%1s%02d' % (t[3],':',t[4],':',t[5])

    nyers_idop = tomb[:20]
    idop = Cserel(nyers_idop,';',")
    data = tomb[21:]
    ii = 0
    while(True):

        i_dat = data[ii:].find('\n')
        if i_dat == -1: break            # tömbnek vége

        i_nev = data[ii:].find(';')
        if i_nev == -1:
            print("adat hiba",data[ii:])
            break
        tagname = data[ii:ii+i_nev]
        nyers_adat =  data[ii+i_nev+1:ii+i_dat]
        ertek = Cserel(nyers_adat,'.',',')
        ii = ii + i_dat + 1

        try:                        # pillanatértékek
            fp = open(Serv_dirname+'\\'+tagname+'.csv','w')
```

```
            fp.write(idop+';'+ertek)
            fp.close()
        except:
            print("Nem sikerült megnyitni: ",Serv_dirname+'\\'+tagname+'.csv')

        try:                        # napi adatok
            if os.path.exists(Serv_dirnapi+datum) == False:      # nincs ilyen direktori : megnyitja
                os.mkdir(Serv_dirnapi+datum)
            fn = open(Serv_dirnapi+datum+'\\'+tagname+'.csv','a')
            fn.write(idop[11:]+';'+ertek+'\n')
            fn.close()
        except:
            print("Nem sikerült megnyitni: ",Serv_dirnapi,datum,tagname+'.csv')
    print(tt)

def Kirak_event(tomb):
    # fájlba menti a friss eseményeket

    t = time.localtime(time.time())
    datum = '%4d%1s%02d%1s%02d' % (t[0],'.',t[1],'.',t[2])
    tt= '%02d%1s%02d%1s%02d' % (t[3],':',t[4],':',t[5])

    nyers_idop = tomb[:20]
    idop = Cserel(nyers_idop,';','')
    data = tomb[21:]
    ii = 0
    while(True):

        i_dat = data[ii:].find('\n')
        if i_dat == -1: break             # tömbnek vége

        i_nev = data[ii:].find(';')
        if i_nev == -1:
            print("adat hiba",data[ii:])
            break
        tagname = data[ii:ii+i_nev]
        nyers_adat =  data[ii+i_nev+1:ii+i_dat]
        ertek = Cserel(nyers_adat,'.',',')
        ii = ii + i_dat + 1
```

```
        try:                    # pillanatértékek
            fp = open(Serv_dirname+'\\'+tagname+'.csv','w')
            fp.write(idop+';'+ertek)
            fp.close()
        except:
            print("Nem sikerült megnyitni: ",Serv_dirname+'\\'+tagname+'.csv')

        try:                    # napi adatok
            if os.path.exists(Serv_dirnapi+datum) == False:     # nincs ilyen direktori: megnyitja
                os.mkdir(Serv_dirnapi+datum)
            fn = open(Serv_dirnapi+datum+'\\'+tagname+'.csv','a')
            fn.write(idop[11:]+';'+ertek+'\n')
            fn.close()
        except:
            print("Nem sikerült megnyitni: ",Serv_dirnapi,datum,tagname+'.csv')
    print(tt)

print("Copy Data & Event to server")
akt = -1
os.chdir("..")                  # fellépünk a főkönyvtárba
while(True):
# minden 20 másodpercben elküldi aszerverbe az aktuális adatokat

    t = time.localtime(time.time())
    td = '%4d%1s%02d%1s%02d ' % (t[0],'.',t[1],'.',t[2])
    tt= '%02d%1s%02d%1s%02d' % (t[3],':',t[4],':',t[5])
    if t[5]==0 or t[5]==20 or t[5]==40:
        if akt != t[5]:
            try:
                aktual_file = open(aktual,'r') # megnézi, hogy jött-e új adat
                tombadat = aktual_file.read()
                aktual_file.close()
                Kirak(tombadat)
                os.remove(aktual)
            except:
                print('Adatra várunk')
            try:
                event_file = open(event,'r') # megnézi, hogy jött-e új adat
```

```
            tombadat = event_file.read()
            event_file.close()
            Kirak_event(tombadat)
            os.remove(event)
        except:
            print('Eseményre várunk')
        akt = t[5]
    else:
        akt = t[5]
sleep(1)                    # wait for 1 secund
```

db.py

```
# Adatbázis olvasás és írás

import os, time

bemenet = "DB_leiro.csv"
data_path = 'Kliens_Data\\'
Kliens_dirnapi = "Kliens_Data\\Napi_"
formatum = {}                  # tagnev , tizedesek száma
mertek = {}                    # tagnev ,true/False
database ={}                         # reltime adatbázis


def Inic():
# feldolgozza a konfig (bemenet) fájlt
    global formatum

    try:
        beo = open(bemenet,'r')
    except:
        print('open error:',bemenet)

    sor = beo.readline() # első sort eldobjuk
    while True:
        sor = beo.readline() #
        sor = sor.strip()
        if sor == '': break  # ha vége a leírónak

        if sor[0] != '#' and sor[0] != ';':
            hosz = sor.find(';')          # tag név
            if hosz < 1:                  # hibás sor
                print("Hibás tagnév : ",sor)
                break
            else:
                tnev = sor[0:hosz]
            tol = hosz + 1
            hosz = sor[tol:].find(';')        # adat formátum
            if hosz < 1:       # hibás sor
```

```
            print("Hibás adat formátum : ",sor)
            break
        else:
            dataform = sor[tol:tol+hosz]
            at = dataform.find('.')
            if at != -1:            # ha van benne pont
                tizedes = len(dataform)-at-1
            else:
                tizedes =  -1
        formatum[tnev] = tizedes
        mertek[tnev] = False
    beo.close()

def Formaz(tagname,value):
# formázza az adatot a leírónak megfelelően
    global formatum

    svalue = str(value)
    pont = svalue.find('.')
    if pont > -1:          # Ha van tizedes pont
        hosz = len(svalue)
        tizedes = hosz-pont-1
        fadat = svalue[:hosz-tizedes+formatum[tagname]]
    else:
        fadat = svalue
    fstring = Cserel(fadat,'.',',')
    return(fstring)

def Cserel(string,mit,mire):
# lecseréli a stringben az egyik karaktert a másikra
    while(True):
        hol = string.find(mit)
        if hol == -1: break
        string = string[:hol]+mire+string[hol+1:]
    return(string)

def GET(azon):
# beolvassa az adott jel pillanatértékét
# input: tag név
```

```
# output: adat float

    global database

    ertek = database[azon]
    fertek = Cserel(ertek,',','.')
    return(float(fertek))

def PUT(azon,adat):
# kiirja a származtatott jel pillanatértékét, és hozzáadja a napi gyűjtéshez
# input: tag név, adat
    global database

    t = time.localtime(time.time())
    td = '%4d%1s%02d%1s%02d ' % (t[0],'.',t[1],'.',t[2])
    tt= '%02d%1s%02d%1s%02d' % (t[3],':',t[4],':',t[5])
    try:
        fp = open(data_path+'Aktual.csv','a')
    except:
        print("Hibás fájl hivatkozás:",data_path,'Aktual.csv')
        exit()
    ertek_f = Formaz(azon,adat)
    try:
        fp.write(azon+';'+ertek_f+'\n')
    except:
        print("Adat írás hiba",azon,adat)
    fp.close()
    tdd = td.strip()
    if os.path.exists(Kliens_dirnapi+tdd) == False:      # nincs ilyen direktori akkor megnyitja
        os.mkdir(Kliens_dirnapi+tdd)
    try:                            # napi adatok
        fn = open(Kliens_dirnapi+tdd+'\\'+azon+'.csv','a')
        fn.write(tt+';'+ertek_f+'\n')
        fn.close()
    except:
        print("Nem sikerült megnyitni: ",Kliens_dirnapi+tdd+'\\'+azon+'.csv')


def Fill_data():
```

```
# beolvassa az összes pillanatértékét
# output: ertekek
    global database

    try:
        fg = open(data_path+'Aktual.csv','r')
    except:
        print("Nincs adatbázis fájl",'Aktual.csv')
        return(-1)
    maszk = mertek
    sor = fg.readline()     # első sort eldobjuk
    while sor!= '':
        sor = fg.readline()     # adatok
        hosz = sor.find(';')    # tag név
        tn = sor[:hosz]
        value = sor[hosz+1:].strip()
        ertek = Cserel(value,'.',',')
        database[tn]=ertek            # feltölti az adatbázis tömböt
        maszk[tn] = True
    fg.close()
    for tn in maszk:
        if maszk[tn] == False:
            database[tn]='0'          # feltölti a nem mért adatokat
```

encrypt.py

```
# titkosító program
import  sys, os, errno, time

S_tomb = []
Kulcs = {}
matrix = {}
ciklus = 8

def Kulcs_generalas(W0):
# Létrehozza a kulcs matrixot [4:10]
    global Kulcs



    vv = W0
    for j in range(160):
        vi = hex(int(vv[2*j:2*j+2],16)^ int(vv[2*j+6:2*j+8],16))[2:]
        if int(vi,16) < 16: vi = '0'+ vi # Ha egyjegyü
        vv = vv +vi
    for i in range(40):
        Kulcs[i] = vv[8*i:8*i+8]

def AddRoundKey(cikl):
# kulcs XOR matrixot [4:4]
    global Kulcs, matrix

    for j in range(4):
        bsor = ''
        for i in range(4):
            vi =  hex(int(matrix[j][2*i:2*i+2],16) ^ int(Kulcs[j+4*cikl][2*i:2*i+2],16))[2:]
            if int(vi,16) < 16: vi = '0'+ vi # Ha egyjegyü
            bsor = bsor + vi
        matrix[j] = bsor
        if len(bsor) < 8: print('itt a hiba: ', bsor)

def SubBytes():
# behelyettesítés az S_tomb-bol [4:4]
    global S_tomb, matrix
```

```
    for x in range(4):
        bsor = ''
        for y in range(4):
            bsor = bsor + S_tomb[int(matrix[x][2*y],16)*16+int(matrix[x][2*y+1],16)]
        matrix[x] = bsor

def Inverse_SubBytes():
# vissza kódol az S_tomb alapján [4:4]
    global S_tomb, matrix

    for x in range(4):
        bsor = ''
        for y in range(4):
            data = matrix[x][2*y]+matrix[x][2*y+1]
            for i in range(256):
                if S_tomb[i] == data:
                    vi = hex(int(i / 16)*16+int(i % 16))[2:]
                    if int(vi,16) < 16: vi = '0'+ vi # Ha egyjegyü
                    bsor = bsor + vi
        matrix[x] = bsor

def ShiftRows():
# eltolja a sorokat balra első marad, második 1-et, harmadik 2-öt ...
    global matrix

    for i in range(3):
        i = i +1
        matrix[i] = matrix[i][2*i:8]+matrix[i][0:2*i]

def Inverse_ShiftRows():
# eltolja a sorokat jobbra első marad, második 1-et, harmadik 2-öt ...
    global matrix

    for i in range(3):
        i = i +1
        matrix[i] = matrix[i][8-2*i:8]+matrix[i][0:8-2*i]

def Mix():
```

```
# elforgatja a mátrixot a mellékátló körül
    global matrix

    uj = {}
    for i in range(4):
        bsor = ''
        for j in range(4):
            bsor = bsor + matrix[3-j][2*(3-i):2*(3-i)+2]
        uj[i] = bsor
    matrix = uj


def Feltolt(filenev):
# Feltölti az S_tömböt
    global S_tomb

    try:
        hsf = open(filenev,'r')

        for y in range(16):          # sorok feltöltése
            sor = hsf.readline()
            tol = 0
            for x in range(16):
                if x < 15: hosz = sor[tol:].find(';')
                else: hosz = sor[tol:].find('\n')
                if hosz == -1:
                    print("hibas matrix!")
                    break
                ig = tol + hosz
                szam = sor[tol:ig]
                S_tomb.append(szam)
                tol = ig + 1
        hsf.close()
        # ellenőriz
        for a in range(256):
            minta = str(hex(a))[2:]
            if a <16: minta = '0'+minta
            if not minta in S_tomb:
                print(a)
```

```
    except:
        print ("\nFeltolt: fájl nyitás hiba: ",filenev)

def Karakterizal():
# karakterekké alakítja matrixban lévő hexa számokat
    global matrix

    su = ''
    sor = matrix[0]+matrix[1]+matrix[2]+matrix[3]
    lim = sor.find("ff")
    if  lim > -1: # ha van benne dummy
        sor = sor[0:lim]
    for j in range(int(len(sor)/2)):
        ah = sor[2*j:2*j+2]
        su = su + chr(int(ah,16))
    return(su)

def Koder(string):
# kodolja a megadott sort
    global matrix, ciklus

    dummy ="ffffffffffffffffffffffffffffffff"
    vektor = ''
    for c in string:
        vektor = vektor + hex(ord(c))[2:]
    darab = len(vektor)
    if darab < 32:
        vektor = vektor + dummy[0:32-darab]
    for i in range(4):                  # feltöltjük a matrixot
        matrix[i] = vektor[8*i:8*i+8]
    AddRoundKey(0)
    # itt kezdődik a ciklus
    for cc in range(ciklus):
        SubBytes()
        ShiftRows()
        Mix()
        AddRoundKey(cc+1)
    return(matrix[0]+matrix[1]+matrix[2]+matrix[3])
```

```
def DeKoder(string):
#dekodolja az eredményt
    global matrix, ciklus

    matrix[0] = string[0:8]
    matrix[1] = string[8:16]
    matrix[2] = string[16:24]
    matrix[3] = string[24:32]

    AddRoundKey(ciklus)
    for cc in range(ciklus):
        Mix()
        Inverse_ShiftRows()
        Inverse_SubBytes()
        AddRoundKey(ciklus-cc-1)
    string = Karakterizal()
    return(string)

def Sor_Dekoder(sor):
# dekódolja a megadott sort
# visszatér a dekodolt sorral

    sor = sor.strip()
    K_sor = ''
    while len(sor) > 32:
        ss = sor[0:32]
        K_sor = K_sor + DeKoder(ss)
        sor = sor[32:]
    else:
        K_sor = K_sor + DeKoder(sor)
    return(K_sor)


def Beo(K_filename):
# beolvassa a majd dekódolja a megadott fájlt
# a fájl neve a kodolt nev
# visszatér a dekodolt adatsorokkal

    K_adat = ''
```

```
    inp = open(K_filename,'r')
    while(True):
        sor = inp.readline()
        if sor == '' or sor == '\n': break
        sor = sor.strip()
        while len(sor) > 32:
            ss = sor[0:32]
            K_adat = K_adat + DeKoder(ss)
            sor = sor[32:]
        else:
            K_adat = K_adat + DeKoder(sor)
        K_adat = K_adat + '\n'
    inp.close()
    return(K_adat)


def Beo_Kio(filenev):
# beolvassa a majd kódolja a megadott fájlt
# kiteszi a diszkre
# visszatér a fájl kódolt nevével

    try:
        K_filename = ''
        sor = filenev[:-4]
        while len(sor) > 16:
            ss = sor[0:16]
            K_filename = K_filename + Koder(ss)
            sor = sor[16:]
        else:
            K_filename = K_filename + Koder(sor)

        out = open(K_filename+'.dat','w')
        inp = open(filenev,'r')
        print("itt")
        while(True):
            K_adat = ''
            sor = inp.readline()
            if sor == '' or sor == '\n': break
            sor = sor.strip()
```

```
        while len(sor) > 16:
            ss = sor[0:16]
            K_adat = K_adat + Koder(ss)
            sor = sor[16:]
        else:
            K_adat = K_adat + Koder(sor)

        for kc in K_adat:
            out.write(kc)
        out.write('\n')

    out.close()
    inp.close()
    return(K_filename)
except:
    print ("\nBeo: fájl nyitás hiba: ",filenev)
```

Event.py

 # Esemény logolo program

```
import  sys, os
import SMS

szam = '+36309922644'
def szuro(szov):

    itt = {}
    itt['ő']='ö'
    itt['ű']='ü'
    itt['Ő']='Ö'
    itt['Ű']='Ü'
    ssk = ''
    if 'ő' in szov or 'Ő' in szov or 'ű' in szov or'Ű' in szov:
        for c in szov:
            if c in itt: ssk=ssk+itt[c]
            else: ssk=ssk+c
        return(ssk)
    else: return(szov)

def SMSLog(ser,eventsz):
    #import esemény szöveg

    SMS.send(ser,szam,eventsz)



def Log(eventsz,mode):
    #import esemény szöveg
    #import write or append

    eventsz = szuro(eventsz) # kiszűrjük a hosszú ekezetes karaktereket
    aktevent= "Kliens_Event\\Event.csv"
    evnapi= "Kliens_Event\\Event_"
    print(eventsz)
    try:
        aktual_file = open(aktevent,mode)
```

```
    aktual_file.write(eventsz)
    aktual_file.close()
except:
    print("Nem sikerült megnyitni: ",aktevent)
    input()
fn = evnapi+eventsz[:10]+'.csv'
try:
    napi_file = open(fn,'a')
    napi_file.write(eventsz)     # a napi adtokhoz hozzáadja az aktuálist
    napi_file.close()
except:
    print("Nem sikerült megnyitni napi fájlt")
    input()
```

Event_handler.py

```
# Esemény kezelő modul
import  sys, os, time
from time import sleep
import Event
from db import Cserel
import SMS

bemenet = "Event_leiro.csv"
data_path = 'Kliens_Data\\'
aldir= "Kliens_Event"
alnapi= "Kliens_Event\\Event_"
tomb = []   #event,event_type,trigger,unit,tipus,tnev

t = time.localtime(time.time())
td = '%4d%1s%02d%1s%02d ' % (t[0],'.',t[1],'.',t[2])
tt= '%02d%1s%02d%1s%02d' % (t[3],':',t[4],':',t[5])

def Inicializal():
# feldolgozza a konfig (bemenet) fájlt
    global tomb, bemenet, td, tt

    try:
        beo = open(bemenet,'r')
    except:
        print('open error:',bemenet)
        hiba = td+' '+ tt+';' + "System ;Hiányzik a " + bemenet+ " fájl\n"
        Event.Log(hiba,'a')
        exit(None)

    sor = beo.readline() # első sort eldtdobjuk
    while True:
        sor = beo.readline() #
        if sor == '': break  # ha vége a leírónak
        l = len(sor)
        if sor[0] != '#' and sor[0] != ';':
            hosz = sor.find(';')          # event_type (output)
            if hosz < 1:                  # hibás sor
```

```
    print("Hibás output : ",sor)
    break
else:
    event_type =sor[0:hosz]
tol = hosz + 1

hosz = sor[tol:].find(';')        # trigger
if hosz < 1:      # hibás sor
    print("Hibás trigger : ",sor)
    break
else:
    trigger = sor[tol:tol+hosz]
    at = trigger.find('-')
    if at == -1 and trigger[0]!='<' and trigger[0]!='>' :
        print("Trigger format error!: ",trigger)  # ha van benne kötőjel
        break
tol = tol + hosz + 1
hosz = sor[tol:].find(';')        # mértékegység
merte = sor[tol:tol+hosz]

tol = tol + hosz + 1
hosz = sor[tol:].find(';')        # tipus
if hosz < 1:      # hibás sor
    print("Hibás tipus adat: ",sor)
    break
else:
    tipus = sor[tol:tol+hosz]
tol = tol + hosz + 1              # jelnév
hosz = sor[tol:].find(';')
if hosz < 1:      # hibás sor
    print("Hibás jelnév: ",sor)
    break
else:
    tnev = sor[tol:tol+hosz]

tol = tol + hosz + 1
event = sor[tol:]
event = event.strip('\n')
```

```
    vekt = event,event_type,trigger,tipus,tnev,merte
    tomb.append(vekt)

beo.close()
```

def GET_data():

```
    global tomb, td, tt
# beolvassa az összes pillanatértékét
# output: ertekek

    db ={}
    try:
        fg = open(data_path+'Aktual.csv','r')
    except:
        print("Nincs adatbázis fájl",'Aktual.csv')
        return(-1)
    sor = fg.readline()     # első sort eldobjuk
    while sor!= '':
        sor = fg.readline()     # adatok
        hosz = sor.find(';')    # tag név
        tn = sor[:hosz]
        value = sor[hosz+1:].strip()
        ertek = Cserel(value,'.',',')
        db[tn]=ertek            # feltölti az adatbázis tömböt

    fg.close()

    ertekek = []                # Az adatbázis előző értékek vektora
    ujtomb = []                 # Minden alkalommal újragyártjuk a tömböt a létező adatbázis alapján
    for ss in tomb:
        try:
            ertekek.append(db[ss[4]])
            ujtomb.append(ss)
        except:
            print('Nem létezö adatbázis jelre hivatkozás',ss[4])
    tomb = ujtomb
    return(ertekek)


def gener(ser,ertekek,td,tt):
# eseményt generál
# input ser: soros vonal azonosító (False : nincs SMS küldés)
# input ertekek: Get_data által beolvasott összes korábbi érték, ha nincs adat -1
```

```
# input td, tt: aktuális dátum és idő
# output ertekek: Get_data által beolvasott összes érték, ha nincs adat változás -1

    global aldir, alnapi, tomb, szam

    akt = GET_data()
    if akt != -1:       # Ha jó a mérés a mérés
        if ertekek == -1: # Ha eddig rosz volt de megjavult
            print("Megjavult az adatbázis kapcsolat")
            return(akt)
        else:           # Ha edig is jó volt
            elozo = ertekek
            ertekek = akt
    else:               # Ha elromlott
        print("Elromlottaz adatbázis kapcsolat")
        return(-1)
    i = 0
    eventsz =''
    for sor in tomb: # 0:event,1:event_type,2:trigger,3:tipus,4:tnev,5:mértékegység
        if sor[2] == '0-1' and elozo[i]=='0' and ertekek[i]=='1': # Ha történt váltás
            if sor[1][0] == '+':
                if ser != False:
                    szoveg = td+' '+tt+';'+sor[4]+' '+sor[0]+'\n'
                    Event.SMSLog(ser,szoveg)
                    eventsz = eventsz + td+' '+ tt+';' + "SMS-> " + ';' + sor[4] + sor[0] + '\n'
            else:
                eventsz = eventsz +td+' '+ tt+';' + sor[1] + ';' + sor[4] + sor[0] + '\n' # tároljat

        if sor[2] == '1-0' and elozo[i]=='1' and ertekek[i]=='0': # Ha történt váltás
            if sor[1][0] == '+':
                if ser != False:
                    szoveg = td+' '+tt+';'+sor[4]+' '+sor[0]+'\n'
                    Event.SMSLog(ser,szoveg)
                    eventsz = eventsz + td+' '+ tt+';' + "SMS-> " + ';' + sor[4] + sor[0] + '\n'
            else:
                eventsz = eventsz +td+' '+ tt+';' + sor[1] + ';' + sor[4] + sor[0] + '\n' # tárolja

        if sor[2][0] == '<' and (elozo[i] >= sor[2][1:]) and (ertekek[i] < sor[2][1:]) : # Ha kisebb
            if sor[1][0] == '+':
```

```
            if ser != False:
                szoveg = td+' '+tt+';'+sor[4]+' '+sor[0]+'\n'
                Event.SMSLog(ser,szoveg)
                eventsz = eventsz + td+' '+ tt+';' + "SMS-> " + ';' + sor[4] + sor[0] + '\n'
        else:
            if sor[5] =='-': me = ''        # Ha nincs mértékegység
            else: me = sor[5]
            eventsz = eventsz+td+' '+tt+';'+sor[1]+';'+sor[4]+sor[0]+':['+ertekek[i]+me+']\n' # tárolja

    if sor[2][0] == '>' and (elozo[i] <= sor[2][1:]) and (ertekek[i] > sor[2][1:]) : # Ha nagyobb
        if sor[1][0] == '+':
            if ser != False:
                szoveg = td+' '+tt+';'+sor[4]+' '+sor[0]+'\n'
                Event.SMSLog(ser,szoveg)
                eventsz = eventsz + td+' '+ tt+';' + "SMS-> " + ';' + sor[4] + sor[0] + '\n'
        else:
            if sor[5] =='-': me = ''        # Ha nincs mértékegység
            else: me = sor[5]
            eventsz = eventsz+td+' '+tt+';'+sor[1]+';'+sor[4]+sor[0]+':['+ertekek[i]+me+']\n' # tárolja
    i = i+1

  if eventsz != '':         # Ha keletkezett esemény
    Event.Log(eventsz,'w')

  return(ertekek)

# itt kezdődik a program
print("Esemény kezelö ilindult")
Inicializal()               # feldolgozza a konfig (bemenet) fájlt
ser = SMS.Inic()

if ser == False:
  print("Nem tud küldeni és fogadni SMS-t!")
  hiba = td+' '+ tt+';' + "System ;Nem tud küldeni és fogadni SMS-t!\n"
  Event.Log(hiba,'a')
vvv = GET_data()
akt = -1

while(True):
```

```
    t = time.localtime(time.time())
    td = '%4d%1s%02d%1s%02d ' % (t[0],'.',t[1],'.',t[2])
    tt= '%02d%1s%02d%1s%02d' % (t[3],':',t[4],':',t[5])
    ts = td.strip()
    if akt == -1:               # Ha most indul az eseménykezelés
        event = td+' '+ tt+';' + "System ;Eseménykezelő program elindult\n"
        Event.Log(event,'a')
        akt = -2

    if ser != False:
        w=ser.read(256)
        if w.strip() == b'+CMTI: "MT",1': #Ha jött SMS
            ack = SMS.recive(ser)
            event = td+' '+ tt+';' + "->SMS" + ';' + ack + '\n'
            Event.Log(event,'w')

    if os.path.exists(aldir) == False:          # nincs ilyen direktori akkor megnyitja
        os.mkdir(aldir)

    if t[5]==59 or t[5]==19 or t[5]==39:
        if akt != t[5]:
            vvv = gener(ser,vvv,td,tt)
            akt = t[5]
        else:
            akt = t[5]
    filok = os.listdir("Kliens_Event")
    if ('SMS_event' in filok) and (ser != False):         # Ha egy másik program SMS-t akar küldeni
        be = open("Kliens_Event\\SMS_event")
        eve =be.readline()
        Event.SMSLog(ser,eve)
        be.close()
        os.remove("Kliens_Event\\SMS_event")
    sleep(1)                        # wait for 1 secund
beo.close()
```

MatekK.py

```
# Kliens oldali matematika modul

import os, time
from time import sleep
import db

db.Inic() # Feltöltjük az adatformátumot

# itt kezdődnek a képletek
print("Matematika program elindult")

while(True):

    t = time.localtime(time.time())
    if t[5]==56 or t[5]==16 or t[5]==36:
        db.Fill_data()                  # minden ciklusban feltölti a realtime adatbázist
        Ir = db.GET("Fázis_áram_R")
        if Ir > 0.1:  db.PUT("Kapcsoló1",1)    # Ha folyik áram, akkor felkapcsolták
        if Ir == 0:   db.PUT("Kapcsoló1",0)
        if Ir > 0.3:  db.PUT("Kapcsoló3",1)    # Ha nagyobb az áram, akkor ezt is
        if Ir <0.25:  db.PUT("Kapcsoló3",0)

    MR= db.GET("Meddö_R")
    db.PUT("Meddö_S",Mr*0.8)
    db.PUT("Meddö_T",Mr*1.2)

    sleep(1) #1 másoperces várakozás
```

Modbus_Client.py

```
# Soros aszinkron MODBUS kommunikációs modul (rtu)
import  sys, os, time
from time import sleep
from pymodbus.client.sync import ModbusSerialClient as ModbusClient
import logging
from struct import *
import Event

bemenet = "DB_leiro.csv"
aldir= "Kliens_Data"
alnapi= "Kliens_Data\\Napi_"
tomb = []   #tnev,sform,also_hatar,felso_hatar,tipus,unit,cim

P_unit =1                   # az 1-es azonosítójú eszköz paraméterei
P_method='rtu'
P_port='com1'
P_timeout=1
P_parity='N'
P_baudrate=9600
t = time.localtime(time.time())
td = '%4d%1s%02d%1s%02d ' % (t[0],'.',t[1],'.',t[2])
tt= '%02d%1s%02d%1s%02d' % (t[3],':',t[4],':',t[5])

def Inicializal():
# feldolgozza a konfig (bemenet) fájlt
    global tomb, td, tt

    try:
        beo = open(bemenet,'r')
    except:
        print('open error:',bemenet)
        hiba = td+' '+ tt+';' + "System ;Hiányzik a " + bemenet+ " fájl\n"
        Event.Log(hiba,'a')
        exit(None)

    sor = beo.readline() # első sort eldobjuk
    while True:
```

```
sor = beo.readline() #
sor = sor.strip()
if sor == '': break  # ha vége a leírónak
l = len(sor)
if sor[l-1] ==";" and sor[l-2] ==";":
    print("Nem mért: ",sor)
    continue  # ha nincs címe,nem kell mérni
if sor[0] != '#' and sor[0] != ';':
    hosz = sor.find(';')          # tag név
    if hosz < 1:                  # hibás sor
        print("Hibás tagnév : ",sor)
        break
    else:
        tnev =sor[0:hosz]
    tol = hosz + 1
    hosz = sor[tol:].find(';')        # adat formátum
    if hosz < 1:      # hibás sor
        print("Hibás adat formátum : ",sor)
        break
    else:
        dataform = sor[tol:tol+hosz]
        at = dataform.find('.')
        if at != -1:              # ha van benne pont
            tizedes = len(dataform)-at-1
            sform = '%.'+str(tizedes)+'f'
        else:
            sform = '%.0f'
    tol = tol + hosz + 1
    hosz = sor[tol:].find(';')        # méréshatár
    if hosz < 1:      # hibás sor
        print("Hibás méreshatár : ",sor)
        break
    else:
        mereshat = sor[tol:tol+hosz]
        at = mereshat.find('-')
        if at != -1:              # ha van benne kötőjel
            also_hatar = mereshat[:at]
            felso_hatar = mereshat[at+1:]
        else:
```

```
                print("Méréshatár format error!: ",mereshatár)
            tol = tol + hosz + 1
            hosz = sor[tol:].find(';')        # mértékegység nem használjuk
            tol = tol + hosz + 1
            hosz = sor[tol:].find(';')        # tipus
            if hosz < 1:      # hibás sor
               print("Hibás tipus adat: ",sor)
               break
            else:
               tipus = sor[tol:tol+hosz]

            tol = tol + hosz + 1
            cim = sor[tol:]
            at = cim.find('/')
            if at != -1:              # ha van benne per
               unit = cim[:at]
               rcim = cim[at+1:]
            else:
               print("Hibás cim adat: ",sor)
               break

         vekt = tnev,sform,also_hatar,felso_hatar,tipus,unit,rcim
         tomb.append(vekt)

      beo.close()
      print('init vege')

def lebegopont(b1,b2):
# 4 bátos lebegőpontos adatao float-á konvertálja
# input: also word, felső word
# output: float(b1,b2)

      szaz = 0x100
      ezer = 0x10000
      c1 = szaz*(b1%szaz)+b1//szaz # felcseréli a bytokat
      c2 = szaz*(b2%szaz)+b2//szaz

      csere = ezer*c1+c2          # 4 bytos szám csere után
      hexbytes = (csere).to_bytes(4, byteorder='big') # 4 byte-ot csinál
```

```
    f1 = unpack("<f",hexbytes)[0]
    return(f1)


def lekérdezo(datum,ido):
# lekérdezi az összes definiált adatpontot
# beírja a pillanatértékeket az adatbázisba
# a mért értékekkel aktualizálja az archívumokat

    global aldir, alnapi, tomb

    try:
        aktual_file = open(aldir+'\\Aktual.csv','w')
    except:
        print("Nem sikerült megnyitni: ",aldir,'\Aktual.csv')

    client = ModbusClient(method=P_method, port=P_port, timeout=P_timeout, parity=P_parity
,baudrate=P_baudrate,unit=P_unit)
    ee = client.connect()
    aktual_file.write(td+';'+tt+'\n')          # az első sor a dátum és idő

    for sor in tomb:
        if int(sor[5]) == P_unit:

            try:
                if sor[4] == "float":
                    rr = client.read_holding_registers(int(sor[6],16), 2, unit=int(sor[5]))
                    ertek =lebegopont(int(rr.registers[0]),int(rr.registers[1]))
                    fertek = sor[1] % ertek
                elif sor[4] == "byte"or sor[4] == "word":
                    rr = client.read_holding_registers(int(sor[6],16), 1, unit=int(sor[5]))
                    fertek =hex(int(rr.registers[0]))
                else:
                    print("nem lebegőpontos adat")
                    break
            except:
                print("nincs kapcsolat az adatgyűjtővel")
                break

            aktual_file.write(sor[0]+';'+fertek+'\n') # kírja a pillanaértékeket
```

```
            try:
                fn = alnapi+datum.strip()+'\\'+ sor[0].strip()+'.csv'
                napi_file = open(fn,'a')
                napi_file.write(ido+';'+fertek+'\n')     # a napi adtokhoz hozzáadja az aktuálist
                napi_file.close()
            except:
                print("Nem sikerült megnyitni napi: ",fn)
        else:
            print("nem definiált UNIT-ra való hivatkozás:",sor[5])
    client.close()
    aktual_file.close()

# itt kezdődik
print("MODBUS >adatgyűjtés")
Inicializal()                # feldolgozza a konfig (bemenet) fájlt
akt = -1
event = td+' '+ tt+';' + "System ;MODBUS program elindult\n"
Event.Log(event,'a')

while(True):
    t = time.localtime(time.time())
    td = '%4d%1s%02d%1s%02d ' % (t[0],'.',t[1],'.',t[2])
    tt= '%02d%1s%02d%1s%02d' % (t[3],':',t[4],':',t[5])
    ts = td.strip()
    if os.path.exists(aldir) == False:          # nincs ilyen direktori akkor megnyitja
        os.mkdir(aldir)
    if os.path.exists(alnapi+ts) == False:      # nincs ilyen aldirektori akkor megnyitja
        os.mkdir(alnapi+ts)

    if t[5]==54 or t[5]==14 or t[5]==34:
        if akt != t[5]:
            lekérdezo(td,tt)
            akt = t[5]
        else:
            akt = t[5]
    sleep(1)                        # wait for 1 secund
print("------------------ elkészült!------------")
beo.close()
```

Put_data.py

```
# Ez a modul a mért értékeket kódolja és elküldi a felhőbe
# Kérésre felküldi az archivumokat és eseményeket
# idöszinkronizál

import  sys, os, time
from time import sleep
import  encrypt
from ftplib import FTP
import tarfile
import Event



def aktual_fel(ftpy,idop):
    # ciklikusan hívják, és felküldi az akzuális adatokat
    global ftpdirnapi

    aktfile= "Kliens_Data\\Aktual.csv"
    aktevent= "Kliens_Event\\Event.csv"
    print("Put > ",idop)
    try:
        aktual_file = open(aktfile,'r') # megnézi, hogy jött-e új adat
        aktual_file.close()
        ujadat = True
    except:
        ujadat = False
    try:
        aktual_file = open(aktevent,'r') # megnézi, hogy keletkezett-e új esemény
        aktual_file.close()
        ujevent = True
    except:
        ujevent = False
    if ujadat:
        K_filename = encrypt.Beo_Kio(aktfile) # elkészíti a kódolt a fájlt
        ftpy.storbinary('STOR '+ K_filename+'.dat',open(K_filename+'.dat','rb'))
        os.remove(aktfile)
        os.remove(K_filename+'.dat')
    if ujevent:
```

```
        K_filename = encrypt.Beo_Kio(aktevent) # elkészíti a kódolt a fájlt
        ftpy.storbinary('STOR '+ K_filename+'.dat',open(K_filename+'.dat','rb'))
        os.remove(aktevent)
        os.remove(K_filename+'.dat')

def Rendez(lin):
# feldarabolja a ;-vel elválasztott stringet
    lin =lin.strip()
    su = []
    tol = 0
    while True:
        ig = lin[tol:].find(';')
        if ig == -1:
            return(su)
        su.append(lin[tol:tol+ig])
        tol = tol + ig + 1



#Inicializal()                  # feldolgozza a konfig (bemenet) fájlt

print("Put_data elinult")
encrypt.Feltolt('S_doboz.csv')  # Inicialitálja a titkosítás modult
#encrypt.Kulcs_generalas('412163d9') # legenerálja a kulcsot
encrypt.Kulcs_generalas('415263d9') # legenerálja a kulcsot
try:
    ftpx = FTP('cp1.ezit.hu')
    ftpx.login('kovari@scadasys.hu','kb1118BP')
    #FTP_Request = encrypt.Koder(encrypt.Koder("Arc_request")+".csv" # az adatfájl kódolt
neve
except:
    print("Nincs internet kapcsolat! (Dugd be az antennát, vagy a kábelt!)")
    try:
        ft=open("Kliens_Event\\SMS_event",'w')
        ft.write("Nincs internet kapcsolat! (antenna vagy kábel hiba)")
        ft.close()
    except:
        print("Fájl nyitási hiba: SMS_event")
    exit(None)
```

```
akt = -1
while(True):
# minden 20 másodpercben elküldi a felhőbe az aktuális adatokat

    t = time.localtime(time.time())
    td = '%4d%1s%02d%1s%02d ' % (t[0],'.',t[1],'.',t[2])
    tt= '%02d%1s%02d%1s%02d' % (t[3],':',t[4],':',t[5])
    if akt == -1:               # Ha most indul, esemény generálás
        event = td+' '+ tt+';' + "System ;Put program elindult\n"
        Event.Log(event,'a')
        akt = -2

    if t[5]==0 or t[5]==20 or t[5]==40:
        if akt != t[5]:
            aktual_fel(ftpx,tt)
            akt = t[5]
        else:
            akt = t[5]

# itt kezdődnek a kérésre felküldések
    dirlist =FTP.nlst(ftpx)
     #   Achívum felküldés kérésre
    FTP_Request = "Arc_request.csv"
    if FTP_Request in dirlist: # Ha van adat kérés(FTP_Request!!)
        jok = []
        hibasok = []
        filok = os.listdir("kliens_data")
        LocTmp = open(FTP_Request, "wb")
        ftpx.retrbinary('RETR '+ FTP_Request,LocTmp.write)
        LocTmp.close()
        ssf = open(FTP_Request,"r")
        sor = ssf.readline()
        sss = Rendez(sor)
        FTP.delete(ftpx,FTP_Request)
        ssf.close()
        os.remove("Arc_request.csv")
        print("Adatkérés",sss)
        tar = tarfile.open("arvhive.tar", "w")
        os.chdir('Kliens_Data')     # bemegyünk az adatbázisba
```

```
        for dn in sss:
            if dn in filok:     # ha benne van az archivumban
                jok.append(dn)
                tar.add(dn)
            else:
                hibasok.append(dn)
        tar.close()
        os.chdir('..')              # kijövünk az adatbázisból
        err =open("err.txt",'w')
        ftpx.storbinary('STOR '+ 'arvhive1.tar',open("arvhive.tar",'rb'))
        err.write("sikerült:")
        os.remove("arvhive.tar")
        for jj in jok:
            err.write(jj+';')
        err.write("\nnincs:")
        for hh in hibasok:
            err.write(hh+';')
        err.close()
        ftpx.storbinary('STOR '+ 'error.txt',open("err.txt",'rb'))
        os.remove("err.txt")

# Eseménytár felküldés kérésre
    FTP_Request = "Event_request.csv"
    if FTP_Request in dirlist: # Ha van esemény kérés(FTP_Request!!)
        jok = []
        hibasok = []
        filok = os.listdir("kliens_event")
        LocTmp = open(FTP_Request, "wb")
        ftpx.retrbinary('RETR '+ FTP_Request,LocTmp.write)
        LocTmp.close()
        ssf = open(FTP_Request,"r")
        sor = ssf.readline()
        sss = Rendez(sor)
        FTP.delete(ftpx,FTP_Request)
        ssf.close()
        os.remove("Event_request.csv")
        print("Esemény kérés",sss)
        tar = tarfile.open("arvhive.tar", "w")
        os.chdir('Kliens_Event')     # bemegyünk az adatbázisba
```

```
        for dn in sss:
            if dn+'.csv' in filok:     # ha benne van az esemény archivumban
                jok.append(dn+'.csv')
                tar.add(dn+'.csv')
            else:
                hibasok.append(dn+'.csv')
        tar.close()
        os.chdir('..')              # kijövünk az eseménykönyvtárból
        err =open("err.txt",'w')
        ftpx.storbinary('STOR '+ 'arvhive1.tar',open("arvhive.tar",'rb'))
        err.write("sikerült:")
        os.remove("arvhive.tar")
        for jj in jok:
            err.write(jj+';')
        err.write("\nnincs:")
        for hh in hibasok:
            err.write(hh+';')
        err.close()
        ftpx.storbinary('STOR '+ 'error.txt',open("err.txt",'rb'))
        os.remove("err.txt")

# Időszinkonizálás
    FTP_Request = "Time_synchron.csv"
    if FTP_Request in dirlist: # Ha van az időállítás kérés(FTP_Request!!)
        LocTmp = open(FTP_Request, "wb")
        ftpx.retrbinary('RETR '+ FTP_Request,LocTmp.write)
        LocTmp.close()
        ssf = open(FTP_Request,"r")
        sor = ssf.readline()
        FTP.delete(ftpx,FTP_Request)
        ssf.close()
        os.remove("Time_synchron.csv")
        print("idö állítás:",sor)
        os.system(sor)              # beállítja az aktuális időt

    sleep(1)                        # wait for 1 secund
```

SMS.py

```
#     SMS küldés Siemens telefonom keresztül soros vonal használatával
import os
import serial
from time import sleep

def Inic():

    try:
        ser = serial.Serial('COM2', 9600, timeout=0) # soros port inicializálás
    except:
        print("Soros vonal nem nyithato meg")
        return(False)
    if ident(ser) == False:
        print("Hibas készülék, vagy vonalhiba!")
        return(False)
    else:
        print("ok")
        ser.write(b'ATE\r\n') # Text mód beállítás
        print("Text mód beállítás")
        sz = varakozas(ser)
        if sz != 'OK': print("Hibás válasz:", sz)

        ser.write(b'AT+CMGF=1\r') # Üzenet formátum beállítás
        ser.write(b'chr(10)') # \n
        print("Üzenet formátum beállítás")
        sz = varakozas(ser)
        if sz != 'OK': print("Hibás válasz:", sz)

        ser.write(b'AT+CNMI=2,1\r\n') # SMS jelzésmód beállítás
        print("SMS jelzésmód beállítás")
        sz = varakozas(ser)
        if sz != 'OK': print("Hibás válasz:", sz)

        ser.write(b'AT+CMGD=1\r\n') # Clearbox küldes
        print("Clearbox küldes")
        sz = varakozas(ser)
        if sz != 'OK': print("Hibás válasz:", sz)
```

```
        ser.write(b'AT+CSCS = "UCS2"\r\n') # UTF váltás
        print("UTF váltás")
        sz = varakozas(ser)
        if sz != 'OK': print("Hibás válasz:", sz)
        return(ser)


def decod_UTC_2(szoveg):
    # input :   '002B0041003200E90151'
    # output:   '+A2éő'

    db = len(szoveg)
    i = 0
    string =''
    while True:
        if i>= db/4:
            return(string)
        else:
            string = string + chr(int(szoveg[4*i:+4*i+4],16))
        i = i + 1

def encod_UTC_2(szoveg):
    # input :   +A2éő'
    # output:  '002B0041003200E90151'

    itt = {}
    itt[0xE9]='00E9' #'é'
    itt[0xE1]='00E0' #'á'
    itt[0xF3]='00F2' #'ó'
    itt[0xF6]='00F6' #'ö'
    itt[0xFA]='00F9' #'ú'
    itt[0xFC]='00FC' #'ü'
    itt[0xED]='00EC' #'í'
    itt[0xC9]='00C9' #'É'
    itt[0xC1]='00C4' #'Á'
    itt[0xD3]='00F2' #'Ó'
    itt[0xD6]='00D6' #'Ö'
    itt[0xDA]='00F9' #'Ú'
```

```
    itt[0xDC]='00DC' #'Ü'
    itt[0xCD]='00EC' #'Í'

    itt[0x151]='00F6' #'ő'
    itt[0x171]='00FC' #'ű'
    itt[0x150]='00D6' #'Ő'
    itt[0x170]='00DC' #'Ű'
    charsor ="0123456789ABCDEF"

    db = len(szoveg)
    i = 0
    string =''
    while True:
        if i >= db:
            return(string)
        else:
            a =ord(szoveg[i])
            if a < 0xC0:            # Ha ékezetmentes karakter
                string = string +'00'+charsor[((a&0xf0)>>4)]+charsor[a&0x0f]
            elif a in itt:          # Ha ékezetes karakter
                string = string +itt[a]
            else:
                string = string + '?'
                print("nem ismert karakter:",i,chr(a))
            i = i+1

def recive(ser):

    ser.write(b'AT+CMGR=1\r\n') # SMS olvasás parancs
    sz = varakozas(ser)
    sz1 = varakozas(ser)
    rsz =decod_UTC_2(sz[94:])
    if rsz != '':
        ser.write(b'AT+CMGD=1\r\n') # Clearbox küldes
        return(rsz)
    else:
        print("Üres üzenet!!!")
        return("Üres üzenet jött!!!")
```

```
def ident(ser):

    print("keszulek azonositas:")
    ser.write(b'AT+CGMI\r\n') # gyártó azonosítás
    sleep(1)
    w=ser.read(100)
    if w.strip() != b'SIEMENS\r\n\r\nOK': return(False)
    ser.write(b'AT+CGSN\r\n') # sorozatszám
    sleep(1)
    w=ser.read(100)
    if w.strip() != b'350301412210039\r\n\r\nOK': return(False)
    else: return(True)

def varakozas(ser):

    while(True):
        w=ser.read(256)
        if w!= b'':
            szurt = ''
            for c in w:
                if c!= 10 and c!= 13:   # cr lf szűrés
                    szurt = szurt+chr(c)
            return(szurt)
        else: sleep(1)

def send(ser,szam,szoveg):
# input: Soros vonal azonosító
# input: Telefonszám (pl. "+36309922644")
# input: Üzenet (pl. "szöveg minta")
# output: készülék válasza

    e_szam= encod_UTC_2(szam)
    e_szoveg= encod_UTC_2(szoveg)
    xxx = b'AT+CMGS=' + e_szam.encode() + b'\r'
    ser.write(xxx) # SMS küldés a telefonra
    sz = varakozas(ser)


    if sz != '> ':
```

```
    print("Hibás válasz:", sz)
else:
    ser.write(e_szoveg.encode()) # szöveg küldés
    ser.write(b'\x1A') # lezárás
    sz = varakozas(ser)
    return(sz)
```

Symulator.py

```
# Azokat a jeleket szimulálja amelyeknek címe van, vagyis mérni kellene
import  sys, os, errno, math, random, time
from time import sleep

bemenet = "DB_leiro.csv"
aldir= "Kliens_Data"
alnapi= "Kliens_Data\\Napi_"
tnev = []
also_hatar = []     # a mérések alsó határértékének tömbje
felso_hatar = []    # a mérések felső határértékének tömbje
tizedes  = []       # a mérések formátumának tömbje

def Inicializal():
# feldolgozza a konfig (bemenet) fájlt
    global tnev, also_hatar, felso_hatar, tizedes

    try:
        beo = open(bemenet,'r')
    except:
        print('open error:',bemenet)

    sor = beo.readline() # első sort eldobjuk
    while True:
        sor = beo.readline() #
        sor = sor.strip()
        if sor == '': break  # ha vége a leírónak
        l = len(sor)
        if sor[l-1] ==";" and sor[l-2] ==";":
            print("nem szimulált jel:",sor)
            continue  # ha nincs címe,nem kell szimulálni
        if sor[0] != '#' and sor[0] != ';':
            hosz = sor.find(';')          # tag név
            if hosz < 1:                  # hibás sor
                print("Hibás tagnév : ",sor)
                break
            else:
                tnev.append(sor[0:hosz])
```

```
            tol = hosz + 1
            hosz = sor[tol:].find(';')        # adat formátum
            if hosz < 1:      # hibás sor
               print("Hibás adat formátum : ",sor)
               break
            else:
               dataform = sor[tol:tol+hosz]
               at = dataform.find('.')
               if at != -1:              # ha van benne pont
                  tizedes.append(len(dataform)-at-1)
               else:
                  tizedes.append(0)
            tol = tol + hosz + 1
            hosz = sor[tol:].find(';')        # méréshatár
            if hosz < 1:      # hibás sor
               print("Hibás méreshatár : ",sor)
               break
            else:
               mereshat = sor[tol:tol+hosz]
               at = mereshat.find('-')
               if at != -1:              # ha van benne kötőjel
                  also_hatar.append(mereshat[:at])
                  felso_hatar.append(mereshat[at+1:])
               else:
                  print("Méréshatár format error!: ",mereshatár)

      beo.close()
      print('init vege')

def szimulal(datum,ido):
      global aldir, alnapi, tnev, tizedes, also_hatar, felso_hatar

      print(ido)
      try:
         aktual_file = open(aldir+'\\Aktual.csv','w')
      except:
         print("Nem sikerült megnyitni: ",aldir,'\Aktual.csv')
      aktual_file.write(td+';'+tt+'\n')          # az első sor a dátum és idő
```

```
    for i in range(len(tizedes)):
        eltolas = int(math.pow(10,tizedes[i]))
        ah, fh = int(also_hatar[i]), int(felso_hatar[i])
        ah = int(ah + 0.7*(fh-ah))
        if tizedes[i] == 0:
            ertek = str(random.randint(ah,fh))
        else:
            ertek = str(random.randint(ah,fh*eltolas)/eltolas)
        aktual_file.write(tnev[i]+';'+ertek+'\n') # kírja a pillanaértékeket

        try:
            fn = alnapi+datum.strip()+'\\'+ tnev[i].strip()+'.csv'
            napi_file = open(fn,'a')
            napi_file.write(ido+';'+ertek+'\n')     # a napi adtokhoz hozzáadja az aktuálist
            napi_file.close()
        except:
            print("Nem sikerült megnyitni: ",fn)
    aktual_file.close()

# itt kezdődik
print("szimulal")
Inicializal()                   # feldolgozza a konfig (bemenet) fájlt
akt = -1
while(True):
    t = time.localtime(time.time())
    td = '%4d%1s%02d%1s%02d ' % (t[0],'.',t[1],'.',t[2])
    tt= '%02d%1s%02d%1s%02d' % (t[3],':',t[4],':',t[5])
    ts = td.strip()
    if os.path.exists(aldir) == False:          # nincs ilyen direktori akkor megnyitja
        os.mkdir(aldir)
    if os.path.exists(alnapi+ts) == False:      # nincs ilyen aldirektori akkor megnyitja
        os.mkdir(alnapi+ts)

    if t[5]==55 or t[5]==15 or t[5]==35:
        if akt != t[5]:
            szimulal(td,tt)
            akt = t[5]
        else:
            akt = t[5]
```

```
    sleep(1)                        # wait for 1 secund
print("------------------ elkészült!------------")
beo.close()
```

Symulator_all.py

```
import  sys, os, errno, math, random, time
from time import sleep

bemenet = "DB_leiro.csv"
aldir= "Kliens_Data"
alnapi= "Kliens_Data\\Napi_"
tnev = []
also_hatar = []     # a mérések alsó határértékének tömbje
felso_hatar = []    # a mérések felső határértékének tömbje
tizedes  = []       # a mérések formátumának tömbje

def Inicializal():
# feldolgozza a konfig (bemenet) fájlt
    global tnev, also_hatar, felso_hatar, tizedes

    try:
        beo = open(bemenet,'r')
    except:
        print('open error:',bemenet)

    sor = beo.readline() # első sort eldobjuk
    while True:
        sor = beo.readline() #
        sor = sor.strip()
        if sor == '': break  # ha vége a leírónak
        if sor[0] != '#' and sor[0] != ';':
            hosz = sor.find(';')          # tag név
            if hosz < 1:                  # hibás sor
                print("Hibás tagnév : ",sor)
                break
            else:
                tnev.append(sor[0:hosz])
            tol = hosz + 1
            hosz = sor[tol:].find(';')        # adat formátum
            if hosz < 1:       # hibás sor
                print("Hibás adat formátum : ",sor)
                break
```

```
            else:
                dataform = sor[tol:tol+hosz]
                at = dataform.find('.')
                if at != -1:              # ha van benne pont
                    tizedes.append(len(dataform)-at-1)
                else:
                    tizedes.append(0)
            tol = tol + hosz + 1
            hosz = sor[tol:].find(';')        # méréshatár
            if hosz < 1:       # hibás sor
                print("Hibás méreshatár : ",sor)
                break
            else:
                mereshat = sor[tol:tol+hosz]
                at = mereshat.find('-')
                if at != -1:              # ha van benne kötőjel
                    also_hatar.append(mereshat[:at])
                    felso_hatar.append(mereshat[at+1:])
                else:
                    print("Méréshatár format error!: ",mereshatár)

    beo.close()
    print('init vege')

def szimulal(datum,ido):
    global aldir, alnapi, tnev, tizedes, also_hatar, felso_hatar

    print(ido)
    try:
        aktual_file = open(aldir+'\\Aktual.csv','w')
    except:
        print("Nem sikerült megnyitni: ",aldir,'\Aktual.csv')
    aktual_file.write(td+';'+tt+'\n')          # az első sor a dátum és idő

    for i in range(len(tizedes)):
        eltolas = int(math.pow(10,tizedes[i]))
        ah, fh = int(also_hatar[i]), int(felso_hatar[i])
        ah = int(ah + 0.7*(fh-ah))
        if tizedes[i] == 0:
```

```
            ertek = str(random.randint(ah,fh))
        else:
            ertek = str(random.randint(ah,fh*eltolas)/eltolas)
        aktual_file.write(tnev[i]+';'+ertek+'\n') # kírja a pillanaértékeket

        try:
            fn = alnapi+datum.strip()+'\\'+ tnev[i].strip()+'.csv'
            napi_file = open(fn,'a')
            napi_file.write(ido+';'+ertek+'\n')     # a napi adtokhoz hozzáadja az aktuálist
            napi_file.close()
        except:
            print("Nem sikerült megnyitni: ",fn)
    aktual_file.close()

# itt kezdődik
print("szimulal")
Inicializal()                   # feldolgozza a konfig (bemenet) fájlt
akt = -1
while(True):
    t = time.localtime(time.time())
    td = '%4d%1s%02d%1s%02d ' % (t[0],'.',t[1],'.',t[2])
    tt= '%02d%1s%02d%1s%02d' % (t[3],':',t[4],':',t[5])
    ts = td.strip()
    if os.path.exists(aldir) == False:          # nincs ilyen direktori akkor megnyitja
        os.mkdir(aldir)
    if os.path.exists(alnapi+ts) == False:      # nincs ilyen aldirektori akkor megnyitja
        os.mkdir(alnapi+ts)

    if t[5]==55 or t[5]==15 or t[5]==35:
        if akt != t[5]:
            szimulal(td,tt)
            akt = t[5]
        else:
            akt = t[5]
    sleep(1)                        # wait for 1 secund
print("------------------ elkészült!------------")
beo.close()
```

Program listák (server)

Ablakok.py

```
#      Megjelenítő program
import os, time
from tkinter import *
from db import Cserel

com_nélkül = True
akt = 0         # aktualis x lépés (idő)
maxim = -1       # maximális x lépés (adatok vége) ha nincs adat -1

pontsor = []    # fájlból beolvasott értékek: oo:pp:ss, sec, érték
adatsor = []    # fájlból beolvasott értékek: oo:pp:ss, sec, érték
database = []   # Adatbázis leírót tartalmzza
pic = []        # Aktuális kép paramétereit tartalmazza
(típus,tagnév,posx,posy,color,tizedes,bar1,bar2)
t = time.localtime(time.time())
datum = '%4d%1s%02d%1s%02d' % (t[0],'.',t[1],'.',t[2])

P_width = 800     # a képernyő mérete
P_height = 600
meret_tomb = (10,16,24),(8,40,80) # objektumok méretei, szoveg, bar szél ; kicsi, közepes,
nagy
meret_index ='kicsi','közepes','nagy'

def kepek_keresese():

    keplista = []
    kk = os.listdir("Pict")
    for ki in kk:
        dollar = ki.find("$.csv")
        if dollar > 0: continue
        holapont = ki.find(".csv")
        if holapont > 0:
            keplista.append(ki[:holapont])
    return(keplista)



def trend_beolvas(tnn):
```

```
    # bealvassa az aktuális trend adatait
    global adatsor, pontsor, akt, maxim, datum, database

    adatsor = []
    pontsor = []
    Serv_dirnapi = "Server_Data\\Napi_"+datum
    try:
        for tn in range(len(database)):
            if tnn == database[tn][0]: break
        print("trend_beolvaso",tnn,tn)
        os.chdir(Serv_dirnapi)
        fp = open(database[tn][0]+'.csv','r')
        while(True):
            sor = fp.readline()
            if sor == '': break
            time_sec = 3600*int(sor[0:2])+60*int(sor[3:5])+int(sor[6:8])
            sor = Cserel(sor,',','.')
            ertek = float(sor[9:].strip())
            a = sor[:8], sor[9:].strip()
            adatsor.append(a)
            a = sor[:8],100+(time_sec)/2, 250-int(200*ertek/(database[tn][3]-database[tn][2]))
            pontsor.append(a)
        os.chdir('..')
        os.chdir('..')
        fp.close()
        maxim = akt = len(pontsor)-31
    except:
        print("Nincs a dátumhoz tartozó pontsor: ",Serv_dirnapi)
        maxim = -1

def adatbázis_feltöltés():
    # database feltöltése a leíróból
    global database

    fp = open('DB_leiro.csv','r')
    sor = fp.readline()     # első sor komment
    sor = fp.readline()     # 1 leíró sor
    i = 0
    while(sor != '' and sor[0] !=';'):     # Adatbázis feltöltés
```

```
        f1 = sor.find(';')
        a0 = sor[:f1]                    # Tagname
        f2 = sor[f1+1:].find(';')
        df = sor[f1+1:f1+f2+1]           # Formatum
        at = df.find('.')
        if at != -1:             # ha van benne pont
            a1=(len(df)-at-1)
        else:
            a1=0
        f3 = sor[f1+f2+2:].find(';')
        ahv = sor[f1+f2+2:f1+f2+f3+2:]
        ahfh = Cserel(ahv,',','.')
        a2 = float(ahfh[:ahfh.find('-')])    # Also határ
        a3 = float(ahfh[ahfh.find('-')+1:])  # Felső határ
        f4 = sor[f1+f2+f3+3:].find(';')
        a4 = sor[f1+f2+f3+3:f1+f2+f3+f4+3] # Mértékegység
        a = a0,a1,a2,a3,a4
        database.append(a)
        sor = fp.readline()                      # kovetkezö sor
        i = i+1
    fp.close()

def adat_feltöltés(azon):
# beolvassa az adott jel pillanatértékét
# output: adat string
    try:
        fp = open('Server_Data\\Pillanat\\'+azon+'.csv','r')
        sor = fp.readline()     # adatok kiolvasása
    except:
        print("Hibás adatbázis hivatkozás:",azon)
        return('0')

    fp.close()
    f1 = sor.find(';')
    return(sor[f1+1:].strip())           # Adat

def kep_feltöltés(kep):
    # aktuális kép paraméterinek feltöltése a leíróból
    global pic
```

```
    pic = []
    fp = open("Pict\\"+kep+'.csv','r')
    sor = fp.readline()     # első sor komment
    sor = fp.readline()     # 1 leíró sor
    i = 0
    while(sor != '' and sor[0] !=';'):     # képleíró feltöltés
        if sor[0] != "#":                  # komment kihagyása
            f1 = sor.find(';')
            a0 = sor[:f1]                      # Típus
            f2 = sor[f1+1:].find(';')
            a1 = sor[f1+1:f1+f2+1]             # Tagname
            f3 = sor[f1+f2+2:].find(';')
            a2 = sor[f1+f2+2:f1+f2+f3+2:]      # PosX
            f4 = sor[f1+f2+f3+3:].find(';')
            a3 = sor[f1+f2+f3+3:f1+f2+f3+f4+3] # PosY
            f5 = sor[f1+f2+f3+f4+4:].find(';')
            a4 = sor[f1+f2+f3+f4+4:f1+f2+f3+f4+f5+4] # Szin
            f6 = sor[f1+f2+f3+f4+f5+5:].find(';')
            if f6 == -1: # Ha nincs a végén ;
                f6 = sor[f1+f2+f3+f4+f5+5:].find('\n')
            a5 = sor[f1+f2+f3+f4+f5+5:f1+f2+f3+f4+f5+f6+5] # Tulajdonság
            a = a0,a1,a2,a3,a4,a5
            pic.append(a)
        sor = fp.readline()                        # kovetkezö sor
        i = i+1
    fp.close()

def at_pakol(sajat,ak):
    # az pontsorban lévő napi adatokból átpakolja a látható pontok adatait
    global pontsor

    sajat.sc = []
    sajat.hely = []
    lyukak = 0
    for i in range(31):
        if i == 0:
            t0 = pontsor[ak][1]
        tu = pontsor[ak+i][1]
```

```
        a = tu-t0+100,pontsor[ak+i][2]
        sajat.sc.append(a)

    dif = int((tu-t0)/10)-len(sajat.sc)
    if dif >=0:                # Ha van ugrás
        sajat.sc = sajat.sc[:-(dif+1)]
    print("kutritva:", tu-t0,len(sajat.sc),sajat.sc)



def tengelyek(sajat,mode):
    global database, datum



    for tn in range(len(database)):
        if sajat.tnn[sajat.selObject[0]] == database[tn][0]: break
    if mode == 1:           # fisítés akkor töröl
        sajat.canvas.create_rectangle(0,0,450,300, fill='white', outline='white') # töröl
    sajat.canvas.create_line(100,250,400,250, width=2) # X tengely
    sajat.canvas.create_line(100,250,100,50, width=2)  # Y tengely

    for i in range(11):     # X tengely
        x = 100 + (i * 30)
        sajat.canvas.create_line(x,250,x,245, width=2)
    for i in range(6):      # Y tengely
        y = 250 - (i * 40)
        sajat.canvas.create_line(100,y,105,y, width=2)
        sajat.canvas.create_text(96,y, text='%5.1f'% (database[tn][2] + i*(database[tn][3]-
database[tn][2])/5), anchor=E)
        sajat.szal_posx = 15
        sajat.szal_index = 15
        xs = 100+ 10* sajat.szal_posx
    sajat.szal = sajat.canvas.create_line(xs,50,xs,250, width=1, fill='orange') # szálkereszt



def rajzol(sajat):
    # görbe rajzóló
    global pontsor, adatsor, akt, database

    for tn in range(len(database)):
```

```
        if sajat.tnn[sajat.selObject[0]] == database[tn][0]: break
    at_pakol(sajat,akt)
    scaled = sajat.sc
    sajat.canvas.create_text(100,260, text='%s'% (pontsor[akt][0]), anchor=N) # idő kezdete
    sajat.canvas.create_text(400,260, text='%s'% (pontsor[akt+30][0]), anchor=N) # idő vége
    sajat.canvas.szalido  =sajat.canvas.create_text(250,260, text='%s'% (adatsor[akt+15][0]), fill
= 'orange') # aktuális idő a szálnál
    sajat.canvas.szaladat =sajat.canvas.create_text(250,274, text='%s'% (adatsor[akt+15][1]), fill
= 'orange') # aktuális érték a szálnál
    if sajat.selObject[0] in sajat.color:
        color = sajat.color[sajat.selObject[0]]
    else:
        color = 'SkyBlue2'
    sajat.canvas.create_line(scaled, fill=color,smooth=1)
    for x,y in scaled:
        sajat.canvas.create_oval(x-3,y-3,x+3,y+3, width=1,outline='black', fill=color)
    sajat.canvas.create_text(40,25, text='%s'% (database[tn][0]), anchor=N) # Tagnév
    hossz=len(database[tn][0])
    sajat.canvas.create_text(40+4*hossz,25, text='%s'% ('[ '+database[tn][4]+' ]'), anchor=N) #
Mértékegység



class Draw(Frame):
    #"A program ablakát definiáló osztály"
    def __init__(sajat):
        Frame.__init__(sajat)
        # A vászon létrehozása:
        Label(sajat, text ='Adatgyűjő program!').pack(side=TOP, padx=40)
        sajat.hatterszin = 'grey'
        sajat.vaszon = Canvas(sajat, width =P_width, height =P_height, bg = sajat.hatterszin)
        sajat.vaszon.pack(padx =0, pady =0)
        adatbázis_feltöltés()
        sajat.kepek = kepek_keresese()
        print("kepek",sajat.kepek)
        sajat.kep = sajat.kepek[0]
        kep_feltöltés(sajat.kep)
        # Az <egéreseményekek> hozzákapcsolása a (vászon) widget-hez :
        sajat.vaszon.bind("<Button-1>", sajat.mouseDown)
        sajat.vaszon.bind("<Button-3>", sajat.mouseRightDown)
        sajat.vaszon.bind("<Button3-ButtonRelease>", sajat.mouseRUP)
```

```
    sajat.pack()
    sajat.nyitott = ''
    sajat.jobb_sz = -1
    lista = os.listdir("pict\\")
    if sajat.kep+'.gif' in lista:       # Ha van grafikus háttér
        sajat.img = PhotoImage(file ="pict\\"+sajat.kep+'.gif')
        sajat.vaszon.create_image(402, 302, image =sajat.img)

    sajat.frissit(True)
    i = 1
    for kk in sajat.kepek:
        Button(sajat, text=kk, command=lambda arg=i:sajat.képváltás(arg)).pack(side=RIGHT)
        i = i+1

def mindent_torol(sajat):

    for go in sajat.obj:
        sajat.vaszon.delete(go)

def képváltás(sajat,ii):

    sajat.kep = sajat.kepek[ii-1]
    sajat.mindent_torol()
    print("képváltás:",sajat.kep)
    lista = os.listdir("pict\\")
    if sajat.kep+'.gif' in lista:       # Ha van grafikus háttér
        sajat.img = PhotoImage(file ="pict\\"+sajat.kep+'.gif')
        sajat.vaszon.create_image(402, 302, image =sajat.img)
    else:
        sajat.vaszon.create_rectangle(0,0,800,600, fill = sajat.hatterszin)

def frissit(sajat,init):
    global pic, database, ssl, meret_tomb, meret_index

    try:
        if com_nélkül: a= 1/0                   # ha kommunikáció, akkor az utolso adat
        fc = open("Server_Data\\Pillanat\#OLVASTAM.txt",'r') # megnézi, hogy jött-e új adat
        fc.close()
        print("nincs aktuális adat!!")
```

```
        except:                                 # ha van új adat
            if sajat.kep != '':
                kep_feltöltés(sajat.kep)
                sajat.kep = ''
                init = True


            if init:
                sajat.picture =  []
                sajat.obj = []              # az összes objektum tömje
                sajat.color = {}             # diagramok szine
                sajat.tnn = {}              # a trend nevek szótára
                sajat.tag = []              # az összes objektum tag neve
                sajat.kepnevek = {}         # grafikus objektumokat tartalmazo lista


                for j in range(len(pic)):
                    tipus = pic[j][0]
                    tag_nev = pic[j][1]
                    print(j,tipus,tag_nev)
                    posX = int(pic[j][2])
                    posY = int(pic[j][3])
                    color = pic[j][4]
                    if pic[j][5] in meret_index:
                        meret = meret_index.index(pic[j][5])
                    elif tipus != "graf_obj":
                        meret = 1
                        print("merethiba",pic[j][5])
                    i = 0
                    for db in database:
                        if tag_nev in db:
                            break
                        elif i< len(database)-1:
                            i = i + 1
                        else: print("adatbázis hivatkozás hiba:",tag_nev)
                    if tipus == 'tagnev':
                        t = sajat.vaszon.create_text(posX,posY, text='%s'% tag_nev, fill = color,
font=('Arial', meret_tomb[0][meret])) # jelnév
                        sajat.tnn[t] = tag_nev
                        st = tag_nev,posX,posY
                        sajat.tag.append(st)
```

```
                sajat.obj.append(t)
            if tipus == 'mértékegység':
                o = sajat.vaszon.create_text(posX,posY, text='%s'% (database[i][4]), fill = color,
font=('Arial', meret_tomb[0][meret])) # Mértékegység
                sajat.obj.append(o)
                st = tag_nev,posX,posY
                sajat.tag.append(st)
            if tipus == 'szám_érték':
                adat = adat_feltöltés(database[i][0])
                tp =adat.find(',')
                if tp == -1:            # nincs a számban tizedes vessző
                    adat = adat
                    tiz = 0
                else:
                    tiz = database[i][1]
                    if tiz == 0: adat = adat[:tp]
                    else: adat = adat[:tiz+tp+1]
                v = sajat.vaszon.create_text(posX,posY, text='%s'% (adat), fill = color,
font=('Arial', meret_tomb[0][meret])) # akt. érték
                ujpic = pic[j][0],pic[j][1],posX,posY,pic[j][4],meret,v,tiz
                sajat.picture.append(ujpic)
                sajat.obj.append(v)
                st = tag_nev,posX,posY
                sajat.tag.append(st)
            if tipus == 'v_bar_érték':
                adat = adat_feltöltés(database[i][0])
                ah, fh = float(database[i][2]), float(database[i][3])
                delta = fh-ah
                ertek = float(Cserel(adat,',','.'))
                x1 = 100*(ertek-ah)/delta
                v1=sajat.vaszon.create_rectangle(posX,posY,
posX+x1,posY+meret_tomb[1][meret], fill = color) # bar diagram eleje
                v2=sajat.vaszon.create_rectangle(posX+x1,posY,
posX+100,posY+meret_tomb[1][meret], fill = sajat.hatterszin ) # bar diagram vége
                ujpic = pic[j][0],pic[j][1],posX,posY,pic[j][4],meret,v1,v2,ah,fh
                sajat.picture.append(ujpic)
                sajat.tnn[v1] = tag_nev
                sajat.tnn[v2] = tag_nev
                sajat.color[v1] = color
                sajat.obj.append(v1)
                sajat.obj.append(v2)
```

```
                    st = tag_nev,posX,posY
                    sajat.tag.append(st)
                    sajat.tag.append(st)
                if tipus == 'f_bar_érték':
                    adat = adat_feltöltés(database[i][0])
                    ah, fh = float(database[i][2]), float(database[i][3])
                    delta = fh-ah
                    ertek = float(Cserel(adat,',','.'))
                    y1 = 100*(ertek-ah)/delta
                    v1=sajat.vaszon.create_rectangle(posX,posY+100,
posX+meret_tomb[1][meret],posY+100-y1, fill = color) # bar diagram eleje
                    v2=sajat.vaszon.create_rectangle(posX,posY+100-y1,
posX+meret_tomb[1][meret],posY, fill = sajat.hatterszin ) # bar diagram vége
                    ujpic = pic[j][0],pic[j][1],posX,posY,pic[j][4],meret,v1,v2,ah,fh
                    sajat.tnn[v1] = tag_nev
                    sajat.tnn[v2] = tag_nev
                    sajat.color[v1] = color
                    sajat.picture.append(ujpic)
                    sajat.obj.append(v1)
                    sajat.obj.append(v2)
                    st = tag_nev,posX,posY
                    sajat.tag.append(st)
                    sajat.tag.append(st)
                if tipus == 'graf_obj':
                    kepszam = pic[j][5]
                    ksz =kepszam.find("-")
                    if ksz == -1:
                        print("Nincs kötőjel a képszám mezőben:",pic[j])
                    else:
                        sr = kepszam[:ksz]
                        sv = kepszam[ksz+1:]
                        imm = PhotoImage(file ="pict\\"+pic[j][4]+'_'+kepszam[:ksz]+'.gif')
                        sajat.kepnevek[str(j)+pic[j][4]+'_'+kepszam[:ksz]] =imm
                        v=sajat.vaszon.create_image(posX,posY,image=
sajat.kepnevek[str(j)+pic[j][4]+'_'+kepszam[:ksz]])
                        ujpic = pic[j][0],pic[j][1],posX,posY,pic[j][4],kepszam,v
                        sajat.picture.append(ujpic)
                        sajat.obj.append(v)
                        st = tag_nev,posX,posY
                        sajat.tag.append(st)
```

```
        else:                           # ha dinamikus
            index = 0
            for gobj in sajat.picture:
                index = index+1
                if gobj[0] == 'szám_érték':
                    adat = adat_feltöltés(gobj[1])
                    tiz = int(gobj[7])
                    tp =adat.find(',')
                    if tp != -1 and tiz != 0:  # ha nem egész érték (van tizedes)
                        adat = adat[:tiz+tp+1]
                    v = sajat.vaszon.itemconfig(gobj[6], text='%s'% (adat)) # akt. érték
                if gobj[0] == 'v_bar_érték':
                    adat = adat_feltöltés(gobj[1])
                    ertek = float(Cserel(adat,',','.'))
                    ah, fh = gobj[8], gobj[9]
                    delta = fh-ah
                    x1 = 100*(ertek-ah)/delta
                    sajat.vaszon.coords(gobj[6],gobj[2],gobj[3],
gobj[2]+x1,gobj[3]+meret_tomb[1][gobj[5]]) # bar diagram eleje
                    sajat.vaszon.coords(gobj[7],gobj[2]+x1,gobj[3],
gobj[2]+100,gobj[3]+meret_tomb[1][gobj[5]]) # bar diagram vége
                if gobj[0] == 'f_bar_érték':
                    adat = adat_feltöltés(gobj[1])
                    ertek = float(Cserel(adat,',','.'))
                    ah, fh = gobj[8], gobj[9]
                    delta = fh-ah
                    y1 = 100*(ertek-ah)/delta
                    sajat.vaszon.coords(gobj[6],gobj[2],gobj[3]+100,
gobj[2]+meret_tomb[1][gobj[5]],gobj[3]+100-y1) # bar diagram eleje
                    sajat.vaszon.coords(gobj[7],gobj[2],gobj[3]+100-y1,
gobj[2]+meret_tomb[1][gobj[5]],gobj[3]) # bar diagram vége
                if gobj[0] == 'graf_obj':
                    adat = adat_feltöltés(gobj[1])
                    tp =adat.find(',')
                    if tp == -1:            # nincs a számban tizedes vessző
                        adat = adat
                    else:
                        print('hibás adat, tizedes',adat)
```

```
                kepszam = gobj[5]
                ksz =kepszam.find("-")
                if ksz == -1:
                   print("Nincs kötőjel a képszám mezőben:",gobj)
                else:
                   sr = kepszam[:ksz]
                   sv = kepszam[ksz+1:]
                   if adat>= sr and adat<= sv:
                      imm = PhotoImage(file ="pict\\"+gobj[4]+'_'+str(adat)+'.gif')
                      sajat.kepnevek[str(gobj[6])+gobj[4]+'_'+str(adat)] =imm
                      v=sajat.vaszon.itemconfig(gobj[6],image=
sajat.kepnevek[str(gobj[6])+gobj[4]+'_'+str(adat)]) # akt. érték

         ff = open("Server_data\\Pillanat\#OLVASTAM.txt",'w')
         ff.close()
      sajat.after(2000,sajat.frissit,False) # várunk 2 másodpercet

   def kepkirako(sajat,kepnev,x,y):

      return(vv)



   def jobbra(sajat):
      global akt, maxim

      if akt < maxim-30: akt = akt+30
      tengelyek(sajat,1)
      rajzol(sajat)

   def vege(sajat):
      global akt, maxim

      akt = maxim
      tengelyek(sajat,1)
      rajzol(sajat)

   def balra(sajat):
      global akt
      if akt > 30: akt = akt-30
      tengelyek(sajat,1)
```

```
        rajzol(sajat)

    def eleje(sajat):
        global akt
        akt = 0
        tengelyek(sajat,1)
        rajzol(sajat)

    def szal_mozgato(sajat):
        global adatsor, akt

        xs = int(sajat.szal_posx)
        xa = int(xs+(250-xs)*0.2)
        sajat.canvas.coords(sajat.szal,xs,50,xs,250)
        sajat.canvas.delete(sajat.canvas.szalido)
        sajat.canvas.delete(sajat.canvas.szaladat)
        sajat.canvas.szalido  =sajat.canvas.create_text(xa,260, text='%s'%
(adatsor[akt+sajat.szal_index][0]), fill = 'orange') # aktuális idő a szálnál
        sajat.canvas.szaladat =sajat.canvas.create_text(xa,274, text='%s'%
(adatsor[akt+sajat.szal_index][1]), fill = 'orange') # aktuális érték a szálnál

    def jszal(sajat):

        if sajat.szal_index < len(sajat.sc)-1:
            sajat.szal_index = sajat.szal_index + 1
            sajat.szal_posx = sajat.sc[sajat.szal_index][0]
            sajat.szal_mozgato()

    def bszal(sajat):

        if sajat.szal_index > 0:
            sajat.szal_index = sajat.szal_index - 1
            sajat.szal_posx = sajat.sc[sajat.szal_index][0]
            sajat.szal_mozgato()

    def fetch(sajat,dat):
        # dátum beolvasó ablak
        global datum, akt
        if int(dat[0:4])>2016 and int(dat[0:4])<2036\
                and int(dat[5:7])> 0 and int(dat[5:7])<13\
```

```
            and int(dat[8:10])> 0 and int(dat[8:10])<32\
            and dat[4] =='.' and dat[7] =='.'\
            and len(dat) == 10:
           datum = dat # átírjuk a dátumot
    print("akt:",akt)
    akt = 0
    sajat.trendablak()

def trendablak(sajat):
    global maxim, datum

    if sajat.nyitott != '':
        sajat.nyitott.destroy()
        sajat.nyitott = ''
    sajat.trend = Toplevel(sajat)
    sajat.trend.title('Trend diagram')
    sajat.canvas = Canvas(sajat.trend, width=450, height=300, bg = 'white')
    sajat.nyitott = sajat.trend
    sajat.canvas.pack(side=TOP)
    trend_beolvas(sajat.tnn[sajat.selObject[0]])
    if maxim > 0:
        Button(sajat.trend, text='<<', command=sajat.eleje).pack(side=LEFT, padx=20)
        Button(sajat.trend, text='<=', command=sajat.balra).pack(side=LEFT, padx=20)
        Button(sajat.trend, text='<', command=sajat.bszal).pack(side=LEFT, padx=20)
        Button(sajat.trend, text='>>', command=sajat.vege).pack(side=RIGHT, padx=20)
        Button(sajat.trend, text='=>', command=sajat.jobbra).pack(side=RIGHT, padx=20)
        Button(sajat.trend, text='>', command=sajat.jszal).pack(side=RIGHT, padx=20)
        Button(sajat.trend, text='Quit', command=sajat.trend.destroy).pack()
    ent = Entry(sajat.trend, width=10, bg="yellow", relief=SUNKEN)
    ent.insert(END, datum)
    ent.pack(side=TOP, fill=X) # grow horiz
    ent.bind('<Return>', (lambda event: sajat.fetch(ent.get()))) # on enter key

    tengelyek(sajat,0)
    if maxim > 0:  rajzol(sajat)

def mouseDown(sajat, event):
    global pic
```

```
        #"Balegérgomb lenyomására végrehajtandó művelet"
        # event.x és event.y tartalmazzák a kattintás koordinátáit :
        sajat.x1, sajat.y1 = event.x, event.y
        # <find_closest> a legközelebbi rajz referenciáját adja meg :
        sajat.selObject = sajat.vaszon.find_closest(sajat.x1, sajat.y1)
        print(sajat.selObject,sajat.tnn)
        if sajat.selObject[0] in sajat.tnn:
            sajat.trendablak()

    def mouseRightDown(sajat, event):
        global pic
        #"Jobb egérgomb lenyomására kiírja a jelnevet"
        sajat.selObject = sajat.vaszon.find_closest(event.x, event.y)
        if sajat.selObject[0] not in sajat.obj:
            return()
        oo = sajat.obj.index(sajat.selObject[0])
        x = sajat.tag[oo][1]
        y = sajat.tag[oo][2]
        tag_nev = sajat.tag[oo][0]
        hossz = len(tag_nev)
        for sor in pic:
            try:
                sor.index(tag_nev)
                break
            except:
                folyt = True

        tipus =  sor[0]
        if tipus == "v_bar_érték":
            dx = event.x-x
            dy = event.y-y
            if dx > 0 and dx < 100 and dy > 0 and dy < 40:
                sajat.jobb_ha = sajat.vaszon.create_rectangle(event.x-hossz*4,event.y-40,
event.x+hossz*6,event.y-20, fill = "lightgray") # háttér bar
                sajat.jobb_sz = sajat.vaszon.create_text(event.x+10,event.y-30, text='%s'% tag_nev,
fill = "black", font=('Arial',10)) # jelnév
        elif tipus == "f_bar_érték":
            dx = event.x-x
            dy = event.y-y
            if dx > 0 and dx < 40 and dy > 0 and dy < 100:
```

```
            sajat.jobb_ha = sajat.vaszon.create_rectangle(event.x-hossz*4,event.y-40,
event.x+hossz*6,event.y-20, fill = "lightgray") # háttér bar
            sajat.jobb_sz = sajat.vaszon.create_text(event.x+10,event.y-30, text='%s'% tag_nev,
fill = "black", font=('Arial',10)) # jelnév
        else:
            dx = abs(event.x-x)
            dy = abs(event.y-y)
            if dx < 8*hossz and dy < 13:
                sajat.jobb_ha = sajat.vaszon.create_rectangle(event.x-hossz*4,event.y-40,
event.x+hossz*6,event.y-20, fill = "lightgray") # háttér bar
                sajat.jobb_sz = sajat.vaszon.create_text(event.x+10,event.y-30, text='%s'% tag_nev,
fill = "black", font=('Arial',10)) # jelnév


    def mouseRUP(sajat, event):
        #"Jobb egérgomb felengedésekor letörli a jelnevet"
        sajat.selObject = sajat.vaszon.find_closest(event.x, event.y)
        if sajat.jobb_sz != -1:
            sajat.vaszon.delete(sajat.jobb_sz)
            sajat.vaszon.delete(sajat.jobb_ha)
            sajat.jobb_sz = -1


if __name__ == '__main__':
    Draw().mainloop()
```

Editor.py

```
#      Képszerkesztő program
import os
from tkinter import *
from db import Cserel


com_nélkül = True
database = []   # Adatbázis leírót tartalmazza
pic = []        # Aktuális kép paramétereit tartalmazza
(típus,tagnév,posx,posy,color,tizedes,bar1,bar2)
meret_tomb = (10,16,24),(8,40,80) # objektumok méretei, szoveg, bar szél ; kicsi, közepes,
nagy
meret_index ='kicsi','közepes','nagy'
van_hatter = 0
kepek = []      # grafikus objektumokat tartalmazo lista
cindex = -1      # grafikus objektumokat mutatója


def kepek_keresese():

    keplista = []
    kk = os.listdir("Pict")
    for ki in kk:
        dollar = ki.find("$.csv")
        if dollar > 0: continue
        holapont = ki.find(".csv")
        if holapont > 0:
            keplista.append(ki[:holapont])
    return(keplista)



def Betesz(index,irany,ertek):
    global pic
# beteszi az új koordinátát a képbe
# input: index objektun sorszáma a pic -ben
# input: irány 2: x koordináta
# input: irány 3: y koordináta

    i = 0
    ujsor = []
```

```
    for ite in pic[index]:
        if i ==irany:
            ujsor.append(str(ertek))
        else:
            ujsor.append(ite)
        i = i + 1
    pic[index] = ujsor

def adatbázis_feltöltés():
    # database feltöltése a leíróból
    global database

    try:
        fp = open('DB_leiro.csv','r')
    except:
        print("Nincs adatbázis leíró: DB_leiro.csv")
        input()
        exit()
    sor = fp.readline()     # első sor komment
    sor = fp.readline()     # 1 leíró sor
    if sor == '':
        print("Hiányos az adatbázis leíró: DB_leiro.csv")
        input()
        exit()

    i = 0
    while(sor != '' and sor[0] !=';'):     # Adatbázis feltöltés
        f1 = sor.find(';')
        a0 = sor[:f1]                    # Tagname
        f2 = sor[f1+1:].find(';')
        df = sor[f1+1:f1+f2+1]           # Formatum
        at = df.find('.')
        if at != -1:              # ha van benne pont
            a1=(len(df)-at-1)
        else:
            a1=0
        f3 = sor[f1+f2+2:].find(';')
        ahv = sor[f1+f2+2:f1+f2+f3+2:]
        ahfh = Cserel(ahv,',','.')
```

```
        a2 = float(ahfh[:ahfh.find('-')])    # Also határ
        a3 = float(ahfh[ahfh.find('-')+1:])  # Felső határ
        f4 = sor[f1+f2+f3+3:].find(';')
        a4 = sor[f1+f2+f3+3:f1+f2+f3+f4+3] # Mértékegység
        a = a0,a1,a2,a3,a4
        database.append(a)
        sor = fp.readline()                          # kovetkezö sor
        i = i+1
    fp.close()

def adat_feltöltés(azon):
# beolvassa az adott jel pillanatértékét
# output: adat string
    fp = open('Server_Data\\Pillanat\\'+azon+'.csv','r')
    sor = fp.readline()     # adatok kiolvasása
    fp.close()
    f1 = sor.find(';')
    return(sor[f1+1:].strip())            # Adat

def kep_feltöltés(kep):
    # aktuális kép paraméterinek feltöltése a leíróból
    global pic

    pic = []
    print("feltolt",kep,"Pict\\"+kep+'.csv')
    fp = open("Pict\\"+kep+'.csv','r')
    sor = fp.readline()     # első sor komment
    sor = fp.readline()     # 1 leíró sor
    i = 0
    while(sor != '' and sor[0] !=';'):     # képleíró feltöltés
        f1 = sor.find(';')
        a0 = sor[:f1]                    # Típus
        f2 = sor[f1+1:].find(';')
        a1 = sor[f1+1:f1+f2+1]            # Tagname
        f3 = sor[f1+f2+2:].find(';')
        a2 = sor[f1+f2+2:f1+f2+f3+2:]      # PosX
        f4 = sor[f1+f2+f3+3:].find(';')
        a3 = sor[f1+f2+f3+3:f1+f2+f3+f4+3] # PosY
        f5 = sor[f1+f2+f3+f4+4:].find(';')
```

```
        a4 = sor[f1+f2+f3+f4+4:f1+f2+f3+f4+f5+4] # Szin
        f6 = sor[f1+f2+f3+f4+f5+5:].find(';')
        if f6 == -1: # Ha nincsa végén ;
            f6 = sor[f1+f2+f3+f4+f5+5:].find('\n')
        a5 = sor[f1+f2+f3+f4+f5+5:f1+f2+f3+f4+f5+f6+5] # Tulajdonság
        a = a0,a1,a2,a3,a4,a5
        pic.append(a)
        sor = fp.readline()                    # kovetkezö sor
        i = i+1
    fp.close()


class Draw(Frame):
    #"A program ablakát definiáló osztály"
    def __init__(sajat):
        Frame.__init__(sajat)
        # A vászon létrehozása:
        Label(sajat, text ='Képszerkesztő program!').pack(side=TOP, padx=40)
        sajat.hatterszin = 'burlywood1'
        sajat.vaszon = Canvas(sajat, width =800, height =600, bg = sajat.hatterszin)
        sajat.vaszon.pack(padx =5, pady =3)
        adatbázis_feltöltés()
        sajat.kepek = kepek_keresese()
        # Az <egéresemények> hozzákapcsolása a (vászon) widget-hez :
        sajat.vaszon.bind("<Button3-ButtonRelease>,<Up>", sajat.kijel)
        sajat.vaszon.bind("<Button-1>", sajat.mouseDown)
        sajat.vaszon.bind("<Double-1>", sajat.mouseDouble)
        sajat.vaszon.bind("<Button-2>", sajat.mouseDel)
        sajat.vaszon.bind("<Button1-ButtonRelease>", sajat.mouseUp)
        sajat.vaszon.bind("<Button-3>", sajat.mouseRightDown)
        sajat.vaszon.bind("<Button3-ButtonRelease>", sajat.mouseRUP)
        sajat.pack()
        sajat.init = True
        sajat.kepszam = -1
        sajat.elagazo()
        sajat.vonx = -1
        sajat.vony = -1
        sajat.talalt = 0
        sajat.obj = []          # az összes objektum tömje
```

```
    def save(sajat):
        global pic
        print(sajat.kep," mentése")
        if os.path.exists("pict\\"+sajat.kep+"$.csv"):
            os.remove("pict\\"+sajat.kep+"$.csv")
        os.rename("pict\\"+sajat.kep+".csv","pict\\"+sajat.kep+"$.csv")
        try:
            fp=open("pict\\"+sajat.kep+".csv","w")
            fp.write("Tipus;tag_name;Pos_x;Pos_y;szin;tulajdonság;\n")
            for obj in pic:
                sor =""
                for resz in obj:
                    sor = sor + resz + ';'
                sor = sor + '\n'
                fp.write(sor)
            fp.close()
        except:
            print("Nemsikerült lementeni a","pict\\"+sajat.kep+".csv","képet" )


    def elagazo(sajat):
        i= 0
        sajat.pt =[]
        for pp in sajat.kepek:
            print(pp)
            t = sajat.vaszon.create_text(100,100+50*i, text='%s'% '- '+pp, anchor=W, fill = "black",
font=('Arial', 24 ))
            sajat.pt.append(t)
            i = i +1


    def mindent_torol(sajat):


        print("lengo",len(sajat.obj))
        for go in sajat.obj:
            sajat.vaszon.delete(go)


    def Illeszt(sajat):
        global pic
    # megkeresi a hasonló koordinátájú objektumokat
    # input: sajat.ii = index objektun sorszáma a pic -ben
```

```
    # input: sajat.xakt = x koordináta
    # input: sajat.yakt = y koordináta

        i = 0
        for pp in pic:
            if i != sajat.ii:      # Ha nem a saját objektum
                if int(pp[2]) == sajat.xakt:
                    if sajat.vonx == -1:            # Ha még nincs ilyen vonal
                        sajat.vonx = sajat.vaszon.create_line(int(pp[2]),int(pp[3]),int(pp[2]),sajat.yakt, fill
= "cyan")
                        sajat.talalt = i
                elif i == sajat.talalt and sajat.vonx != -1:
                    sajat.vaszon.delete(sajat.vonx)
                    sajat.vonx = -1
                if int(pp[3]) == sajat.yakt:
                    if sajat.vony == -1:            # Ha még nincs ilyen vonal
                        sajat.vony = sajat.vaszon.create_line(int(pp[2]),int(pp[3]),sajat.xakt,int(pp[3]), fill
= "cyan")
                        sajat.talalt = i
                elif i == sajat.talalt and sajat.vony != -1:
                    sajat.vaszon.delete(sajat.vony)
                    sajat.vony = -1
            i = i +1


    def rajzol(sajat,j):
        global pic, database, ssl, meret_tomb, meret_index, kepek, cindex
        print(pic[j])
        tag_nev = pic[j][1]
        posX = int(pic[j][2])
        posY = int(pic[j][3])
        color = pic[j][4]

        if pic[j][5] in meret_index:
            meret = meret_index.index(pic[j][5])
        elif pic[j][0] != 'graf_obj':
            meret = 1
            print("mérethiba",pic[j])
        i = 0
        for db in database:
            if tag_nev in db:
```

```
                break
            elif i< len(database)-1:
                i = i + 1
            else: print("adatbázis hivatkozás hiba:",tag_nev)
        if pic[j][0] == 'tagnev':
            t = sajat.vaszon.create_text(posX,posY, text='%s'% tag_nev, fill = color, font=('Arial',
meret_tomb[0][meret])) # jelnév
            sajat.obj.append(t)
        elif pic[j][0] == 'mértékegység':
            o = sajat.vaszon.create_text(posX,posY, text='%s'% (database[i][4]), fill = color,
font=('Arial', meret_tomb[0][meret])) # Mértékegység
            sajat.obj.append(o)
        elif pic[j][0] == 'szám_érték':
            adat = adat_feltöltés(database[i][0])
            tp =adat.find(',')
            if tp == -1:              # nincs a számban tizedes vessző
                adat = adat
            else:
                tiz = database[i][1]
                if tiz == 0: adat = adat[:tp]
                else: adat = adat[:tiz+tp+1]
            v = sajat.vaszon.create_text(posX,posY, text='%s'% (adat), fill = color, font=('Arial',
meret_tomb[0][meret])) # akt. érték
            sajat.obj.append(v)
        elif pic[j][0] == 'v_bar_érték':
            adat = adat_feltöltés(database[i][0])
            v=sajat.vaszon.create_rectangle(posX,posY, posX+100,posY+meret_tomb[1][meret], fill
= color) # bar diagram eleje
            sajat.obj.append(v)
        elif pic[j][0] == 'f_bar_érték':
            adat = adat_feltöltés(database[i][0])
            v=sajat.vaszon.create_rectangle(posX,posY+100, posX+meret_tomb[1][meret],posY, fill
= color) # bar diagram eleje
            sajat.obj.append(v)
        elif pic[j][0] == 'graf_obj':
            kepszam = pic[j][5]
            ksz =kepszam.find("-")
            if ksz == -1:
                print("tulajdonság hiba",pic[j])
            else:
                try:
```

```
                imm = PhotoImage(file ="pict\\"+pic[j][4]+'_'+kepszam[:ksz]+'.gif')
                kepek.append(imm)
                cindex = cindex+1
                v =sajat.vaszon.create_image(posX,posY, image = kepek[cindex])
                sajat.obj.append(v)
            except:
                print("Nincs ilyen grafikus elem:",pic[j][4]+'_'+kepszam[:ksz]+'.gif')
                input("kilépés")
                exit(None)
    else:
        print("Hibás objektum azonosító:",pic[j][0])
        input(">")
        exit(None)

def frissit(sajat):
    global pic

    sajat.obj = []              # az összes objektum tömje
    for jp in range(len(pic)):
        sajat.rajzol(jp)

def torol(sajat):
    global database, pic
    obj = sajat.obj.index(sajat.selObject[0])
    print("Most aztán tényleg kitörlöm!!!",sajat.obj,obj,pic[obj])
    ujpic = []
    oo = []
    for i in range(len(sajat.obj)):
        sor = pic[i]
        if i == obj:            # ezt kell kitörölni
            sajat.vaszon.delete(sajat.obj[i])
        else:
            ujpic.append(sor)
            oo.append(sajat.obj[i])
    pic = ujpic
    sajat.obj = oo
    print(oo)
    sajat.beo.destroy()
```

```
    def delete(sajat):
        sajat.beo = Toplevel(sajat)
        sajat.beo.title('Törlés')
        Label(sajat.beo, text="Bitosan ki akarja törölni?", width= 40).pack(side=TOP)
        btn = Button(sajat.beo, text="Cancel", command=sajat.beo.destroy).pack(side=LEFT)
        bti = Button(sajat.beo, text="Delete", command=sajat.torol).pack(side=RIGHT)

    def csinal(sajat,name):
        global database, pic

        talalt = False
        for db in database:
            if name == db[0]:
                talalt = True
                break
        if talalt:
            print(name)
            a0 = pic[sajat.obj.index(sajat.selObject[0])][0]
            a1 = name
            a2 = str(int(pic[sajat.obj.index(sajat.selObject[0])][2]) + 10)
            a3 = str(int(pic[sajat.obj.index(sajat.selObject[0])][3]) - 10)
            a4 = pic[sajat.obj.index(sajat.selObject[0])][4]
            a5 = pic[sajat.obj.index(sajat.selObject[0])][5]
            a = a0,a1,a2,a3,a4,a5
            pic.append(a)
            print("új:",sajat.obj)
            sajat.rajzol(len(pic)-1)
            sajat.beo.destroy()


    def dupla(sajat,neve):
        sajat.beo = Toplevel(sajat)
        sajat.beo.title(neve)
        Label(sajat.beo, text="Új tag azonosító", width = 50).pack(side=TOP)
        ent = Entry(sajat.beo, bg="yellow", relief=SUNKEN, font =16 )
        ent.pack(side=TOP)
        btn = Button(sajat.beo, text="Cancel", command=sajat.beo.destroy).pack(side=LEFT)
        bti = Button(sajat.beo, text="Ok", command=(lambda:
sajat.csinal(ent.get()))).pack(side=RIGHT)
```

```
def mouseDouble(sajat, event):
    global pic
    #dupla Jobb egérgomb lenyomására duplikálja az objektumot

    hsz = {}
    hsz['kicsi']=8
    hsz['közepes']=40
    hsz['nagy']=80
    sajat.selObject = sajat.vaszon.find_closest(event.x, event.y)
    x = int(pic[sajat.obj.index(sajat.selObject[0])][2])
    y = int(pic[sajat.obj.index(sajat.selObject[0])][3])
    dx = abs(event.x-x)
    dy = abs(event.y-y)
    tip = pic[sajat.obj.index(sajat.selObject[0])][0]
    nev = pic[sajat.obj.index(sajat.selObject[0])][1]
    mer = pic[sajat.obj.index(sajat.selObject[0])][5]
    if tip == 'v_bar_érték':  # Ha vízszintes bar
        if  dx < 100 and dy < hsz[mer]:
            sajat.dupla(nev)
    elif tip == 'f_bar_érték':  # Ha függőleges bar
        if  dx <  hsz[mer] and dy < 100:
            sajat.dupla(nev)
    else: sajat.dupla(nev)

def mouseDel(sajat,event):
    #középső egérgomb lenyomására törli az objektumot

    hsz = {}
    hsz['kicsi']=8
    hsz['közepes']=40
    hsz['nagy']=80

    sajat.selObject = sajat.vaszon.find_closest(event.x, event.y)
    x = int(pic[sajat.obj.index(sajat.selObject[0])][2])
    y = int(pic[sajat.obj.index(sajat.selObject[0])][3])
    dx = abs(event.x-x)
    dy = abs(event.y-y)
    tip = pic[sajat.obj.index(sajat.selObject[0])][0]
    nev = pic[sajat.obj.index(sajat.selObject[0])][1]
```

```
        mer = pic[sajat.obj.index(sajat.selObject[0])][5]
        if tip == 'v_bar_érték':  # Ha vízszintes bar
            if  dx < 100 and dy < hsz[mer]:
                sajat.dupla(nev)
        elif tip == 'f_bar_érték':  # Ha függőleges bar
            if  dx <  hsz[mer] and dy < 100:
                sajat.delete()
        else: sajat.delete()


    def mouseRightDown(sajat, event):
        global pic
        #"Jobb egérgomb lenyomására kiírja a jelnevet"
        sajat.selObject = sajat.vaszon.find_closest(event.x, event.y)
        if sajat.selObject[0] not in sajat.obj: return()
        tipus = pic[sajat.obj.index(sajat.selObject[0])][0]
        tag_nev = pic[sajat.obj.index(sajat.selObject[0])][1]
        hossz = len(tag_nev)
        x = int(pic[sajat.obj.index(sajat.selObject[0])][2])
        y = int(pic[sajat.obj.index(sajat.selObject[0])][3])
        if tipus == "v_bar_érték":
            meret = meret_index.index(pic[sajat.obj.index(sajat.selObject[0])][5])
            my = meret_tomb[1][meret]
            dx = event.x-x
            dy = event.y-y
            if dx > 0 and dx < 100 and dy > 0 and dy < my:
                sajat.jobb_ha = sajat.vaszon.create_rectangle(event.x-hossz*4,event.y-40,
event.x+hossz*6,event.y-20, fill = "lightgray") # háttér bar
                sajat.jobb_sz = sajat.vaszon.create_text(event.x+10,event.y-30, text='%s'% tag_nev,
fill = "black", font=('Arial',10)) # jelnév
        elif tipus == "f_bar_érték":
            meret = meret_index.index(pic[sajat.obj.index(sajat.selObject[0])][5])
            mx = meret_tomb[1][meret]
            dx = event.x-x
            dy = event.y-y
            if dx > 0 and dx < mx and dy > 0 and dy < 100:
                sajat.jobb_ha = sajat.vaszon.create_rectangle(event.x-hossz*4,event.y-40,
event.x+hossz*6,event.y-20, fill = "lightgray") # háttér bar
                sajat.jobb_sz = sajat.vaszon.create_text(event.x+10,event.y-30, text='%s'% tag_nev,
fill = "black", font=('Arial',10)) # jelnév
        else:
            dx = abs(event.x-x)
```

```
            dy = abs(event.y-y)
            if dx < 8*hossz and dy < 13:
                sajat.jobb_ha = sajat.vaszon.create_rectangle(event.x-hossz*4,event.y-40,
event.x+hossz*6,event.y-20, fill = "lightgray") # háttér bar
                sajat.jobb_sz = sajat.vaszon.create_text(event.x+10,event.y-30, text='%s'% tag_nev,
fill = "black", font=('Arial',10)) # jelnév


    def kijel(sajat, event):


        print("siker!!")


    def mouseRUP(sajat, event):
        #"Jobb egérgomb felengedésekor letörli a jelnevet"
        sajat.selObject = sajat.vaszon.find_closest(event.x, event.y)
        if sajat.selObject[0] not in sajat.obj: return()
        if sajat.jobb_sz != -1:
            sajat.vaszon.delete(sajat.jobb_sz)
            sajat.vaszon.delete(sajat.jobb_ha)
            sajat.jobb_sz = -1


    def mouseDown(sajat, event):
        global pic
        #"Bal egérgomb lenyomására végrehajtandó művelet"
        sajat.x1, sajat.y1 = event.x, event.y
        # <find_closest> a legközelebbi rajz referenciáját adja meg :
        sajat.selObject = sajat.vaszon.find_closest(sajat.x1, sajat.y1)
        if sajat.selObject[0] not in sajat.obj: return() # Ha nem létező objektumra klikkel
        sajat.px, sajat.py = event.x, event.y
        if sajat.kepszam != -1:
            sajat.xakt = int(pic[sajat.obj.index(sajat.selObject[0])][2])
            sajat.yakt = int(pic[sajat.obj.index(sajat.selObject[0])][3])
            sajat.vaszon.bind("<Button1-Motion>", sajat.mouseMove)
        if sajat.vonx != -1:        # ha van illesztő vonal, kitöröljük
            sajat.vaszon.delete(sajat.vonx)
            sajat.vonx = -1
        if sajat.vony != -1:        # ha van illesztő vonal, kitöröljük
            sajat.vaszon.delete(sajat.vony)
            sajat.vony = -1


    def mouseMove(sajat, event):
```

```
    #"Lenyomott balgombbal mozgó egérrel végrehajtandó művelet"
    x2, y2 = event.x, event.y
    dx, dy = x2 -sajat.x1, y2 -sajat.y1
    if sajat.selObject:
        sajat.vaszon.move(sajat.selObject, dx, dy)
        sajat.x1, sajat.y1 = x2, y2
        sajat.xakt, sajat.yakt = sajat.xakt+dx, sajat.yakt+dy

        sajat.ii = sajat.obj.index(sajat.selObject[0])
        sajat.Illeszt()


def mouseUp(sajat, event):
    global van_hatter, pic

    #"Balegérgomb lefelengedésére végrehajtandó művelet"
    if sajat.init:
        sajat.kep = sajat.kepek[sajat.selObject[0]-1]
        sajat.kepszam = 0
        for cl in sajat.pt:
            sajat.vaszon.delete(cl)
            sajat.kepszam = sajat.kepszam +1
        kep_feltöltés(sajat.kep)
        lista = os.listdir("pict\\")
        if sajat.kep+'.gif' in lista:           # Ha van grafikus háttér
            sajat.img = PhotoImage(file ="pict\\"+sajat.kep+'.gif')
            sajat.vaszon.create_image(402, 302, image =sajat.img)
            van_hatter = 1

        sajat.vaszon.bind("<Button1-Motion>", sajat.mouseMove)
        sajat.frissit()
        Button(sajat, text="Save picture", command=sajat.save).pack(side=RIGHT)
        sajat.init = False
    else:
        print("obj:",sajat.selObject)
        if sajat.selObject[0] not in sajat.obj: return() # Ha nem létező objektumra klikkel
        ujx =int(pic[sajat.obj.index(sajat.selObject[0])][2])+event.x-sajat.px
        Betesz(sajat.obj.index(sajat.selObject[0]),2,ujx)
        ujy =int(pic[sajat.obj.index(sajat.selObject[0])][3])+event.y-sajat.py
```

```
            Betesz(sajat.obj.index(sajat.selObject[0]),3,ujy)

if __name__ == '__main__':
    Draw().mainloop()
```

Event_Request.py

```
#      Archiv esemény lekérdező prgram
#https://www.webucator.com/blog/2015/03/python-color-constants-module/

import os, time
from time import sleep
from tkinter import *
from ftplib import FTP
import tarfile

t = time.localtime(time.time())
datum = '%4d%1s%02d%1s%02d' % (t[0],'.',t[1],'.',t[2])
posY =20
lista = ""
logfile = "err.txt"

def Rendez(lin):

    su = []
    tol = 0
    while True:
        ig = lin[tol:].find(';')
        if ig == -1:
            return(su)
        su.append(lin[tol:tol+ig])
        tol = tol + ig + 1

def felhoz(ftpy):
    # folyamatosan kérdezi a felhőből az aktuális adatokat
    global logfile

    timou = 0
    while(True):
        if timou >= 10: return(False)
        dirlist =FTP.nlst(ftpy)
        if "error.txt" in dirlist:              # Ha jött adat
            errTmp = open(logfile, "wb")
            ftpy.retrbinary('RETR '+ "error.txt",errTmp.write)
```

```
            errTmp.close()
            FTP.delete(ftpy,"error.txt")    # letörlöm a hibafájlt
            return(True)
        else: timou = timou +1
        sleep(1)


class Draw(Frame):
    #"A program ablakát definiáló osztály"
    def __init__(sajat):
        global datum

        Frame.__init__(sajat)
        # A vászon létrehozása:
        Label(sajat, text ='Esemény feltöltő program!').pack(side=TOP, padx=40)
        sajat.vaszon = Canvas(sajat, width =200, height =620, bg ='gold2')
        sajat.vaszon.pack(padx =0, pady =0)
        sajat.pack()
        # Az <egéresemények> hozzákapcsolása a (vászon) widget-hez :
        Button(sajat, text='GO', command=sajat.letolt).pack(side=RIGHT, padx=20)
        Button(sajat, text='Clear', command=sajat.torol).pack(side=RIGHT, padx=20)
        ent = Entry(sajat, width=10, bg="yellow", relief=SUNKEN)
        ent.insert(END, datum)
        ent.pack(side=RIGHT, fill=X) # grow horiz
        ent.bind('<Return>', (lambda event: sajat.fetch(ent.get()))) # on enter key
        sajat.obj = []

    def fetch(sajat,dat):
        # dátum beolvasó ablak
        global datum, posY, lista

        if posY > 600: return
        if len(dat) == 10\
           and int(dat[0:4])>2016 and int(dat[0:4])<2036\
            and int(dat[5:7])> 0 and int(dat[5:7])<13\
            and int(dat[8:10])> 0 and int(dat[8:10])<32\
            and dat[4] =='.' and dat[7] =='.'\
            and dat not in lista:
                lista = lista + "Event_"+dat+';'
```

```
            datum = dat # átírjuk a dátumot
            o = sajat.vaszon.create_text(100,posY, text='%s'% datum, fill = "black", font=('Arial',
16))
            sajat.obj.append(o)
            posY = posY +20
        else: print("Hibás dátum!")

    def torol(sajat):
        global lista,posY

        for gobj in sajat.obj:
            sajat.vaszon.delete(gobj)
        sajat.obj = []
        lista = ""
        posY =20
        print("torol",posY)

    def letolt(sajat):
    # frissíti az archívumokat
        global lista, logfile, posY

        if lista == "": return
        acs = open("Event_request.csv", "w")
        acs.write(lista)
        acs.close()
        ftpx = FTP('cp1.ezit.hu')
        ftpx.login('kovari@scadasys.hu','kb1118BP')
        print("Esemény lekérdezés")
        dirlist =FTP.nlst(ftpx)
        if "error.txt" in dirlist:              # Ha van log fájl letöröljük
            FTP.delete(ftpx,"error.txt")    # letörlöm a hibafájlt
        if "arvhive1.tar" in dirlist:               # Ha van log fájl letöröljük
            FTP.delete(ftpx,"arvhive1.tar")    # letörlöm a régi archivumot

        ftpx.storbinary('STOR '+ 'Event_request.csv',open('Event_request.csv','rb')) # elküldjük a
kérést
        ok = felhoz(ftpx)
        if ok:              # kiértékel
            os.remove('Event_request.csv')
            ssf = open(logfile,"r")
```

```
            jok = ssf.readline()        # első sor, sikeresek
            hib = ssf.readline()        # második sor, sikertelenek
            ssf.close()
            os.remove(logfile)
            posY = 20
            ls = Rendez(lista)
            for tn in ls:
                if tn in jok:
                    o = sajat.vaszon.create_text(180,posY, text=chr(214), fill = "darkgreen",
font=('Symbol', 16))
                    sajat.obj.append(o)
                elif tn in hib:
                    o =sajat.vaszon.create_text(180,posY, text=chr(198), fill = "white", font=('Symbol',
16))
                    sajat.obj.append(o)
                else:  print("adatvesztés: ",tn)
                posY = posY +20

            if "Event_" in jok:             # Ha van jó adat is
                os.chdir('Server_Event')     # bemegyünk az adatbázisba
                arcTmp = open("tmp.tar", "wb")
                ftpx.retrbinary('RETR '+ "arvhive1.tar",arcTmp.write)
                arcTmp.close()
                FTP.delete(ftpx,"arvhive1.tar")    # letörlöm az adatfájlt
                tar = tarfile.open("tmp.tar")
                tar.extractall()
                tar.close()
                os.remove("tmp.tar")
                os.chdir('..')              # kijövünk az adatbázisból
                print("jött adat!")
        else:
            print("Nincs kapcsolat a Kliens programmal!!")
if __name__ == '__main__':
    Draw().mainloop()
```

Event_view.py

```
#      Esemény megjelenítő prgram
#https://www.webucator.com/blog/2015/03/python-color-constants-module/

import os, time
from time import sleep
from tkinter import *
from db import Cserel

t = time.localtime(time.time())
datum = '%4d%1s%02d%1s%02d' % (t[0],'.',t[1],'.',t[2])
mai = datum



class Draw(Frame):
   #"A program ablakát definiáló osztály"
   def __init__(sajat):

      Frame.__init__(sajat)
      # A vászon létrehozása:
      Label(sajat, text ='Esemény megjelenítő program!').pack(side=TOP, padx=40)
      sajat.vaszon = Canvas(sajat, width =800, height =600, bg ='gray')
      sajat.vaszon.config(highlightthickness=0) # no pixels to border
      sbar = Scrollbar(sajat)
      sbar.config(command=sajat.vaszon.yview) # xlink sbar and canv
      sajat.vaszon.config(yscrollcommand=sbar.set) # move one moves other
      sbar.pack(side=RIGHT, fill=Y) # pack first=clip last
      sajat.vaszon.pack(expand=YES, fill=BOTH) # canv clipped first
      sajat.pack()

      # itt lehet módosítani a szűrő feltételeket.
      sajat.colt = {}          # háttérszínek összerendelése
      sajat.colt["System"]="beige"
      sajat.colt["alarm"]="indianred1"
      sajat.colt["warning"]="orange"
      sajat.colt["limit"]="olivedrab1"
      sajat.colt["SMS->"]="orchid1"
      sajat.colt["->SMS"]="orchid3"
```

```
    sajat.obj = []
    sajat.states = []

    # Szűrő gombok :
    for kk in sajat.colt:
        var = IntVar()
        chk = Checkbutton(sajat, text=kk, variable=var).pack(side=LEFT)
        sajat.states.append(var) # szűrő feltételek
        var.set(1)
    ent = Entry(sajat, width=10, bg="yellow", relief=SUNKEN)
    ent.insert(END, datum)
    ent.pack(side=RIGHT, fill=X) # grow horiz
    ent.bind('<Return>', (lambda event: sajat.fetch(ent.get()))) # on enter key
    sajat.mask = [var.get() for var in sajat.states]
    sajat.tomb = sajat.Fajl_beo(datum)
    if sajat.tomb != False:
        sajat.tomb.sort(reverse=True)
        sajat.kitesz()
    sajat.frissit()

def fetch(sajat,dat):
    # változásokat kiteszi
    global datum

    sajat.mask = [var.get() for var in sajat.states]
    if len(dat) == 10\
       and int(dat[0:4])>2016 and int(dat[0:4])<2036\
        and int(dat[5:7])> 0 and int(dat[5:7])<13\
        and int(dat[8:10])> 0 and int(dat[8:10])<32\
        and dat[4] =='.' and dat[7] =='.':
        datum = dat # átírjuk a dátumot
        sajat.torol()
        sajat.tomb = sajat.Fajl_beo(datum)
        if sajat.tomb != False:
            sajat.tomb.sort(reverse=True)
            sajat.kitesz()

    else: print("Hibás dátum!")
```

```
    def kitesz(sajat):
    # Kiirja a tömb elemeit a képernyőre
    # input : sajat.tomb (események)
    # input : sajat.colt (típusok listája)
    # input : sajat.mask (maszk)

        tilto = []
        j = 0
        for tip in sajat.colt:
            if sajat.mask[j] == 1:
                tilto.append(tip)
            j = j + 1

        posY = 0
        sj = 0
        color ="white"
        for sss in sajat.tomb:
            tipus = (sss[21:21+sss[21:].find(';')]).strip()
            if tipus in tilto:
                color = sajat.colt[tipus.strip()]
                sst = sss.strip()
                sst = Cserel(sst,';','\t')
                ur = 108-len(sst)
                for i in range(ur): sst = sst + ' '
                t = sajat.vaszon.create_text(0,posY, anchor=NW, text='%s'% sst, fill = "black",
font=('Arial', 14))
                r = sajat.vaszon.create_rectangle(sajat.vaszon.bbox(t),fill=color, outline=color)
                sajat.vaszon.tag_lower(r,t)
                sajat.obj.append(t)
                sajat.obj.append(r)
                posY = posY +22
                sj = sj + 1
        sajat.vaszon.config(scrollregion=(0, 0, 800, 22*sj)) # canvas size corners

    def Fajl_beo(sajat,datum):
    # feltolti a tombot az eseményfájlból

        os.chdir('Server_Event')     # bemegyünk az adatbázisba
        filename = "Event"+datum+".csv"
        try:
```

```
            ssf = open(filename,"r")
            print("open",filename)
        except:
            os.chdir('..')
            sajat.torol()
            t = sajat.vaszon.create_text(0,0, anchor=NW, text='Nincs erre a napra vontkozó
esemény archivum!', fill = "black", font=('Arial', 14))
            sajat.obj.append(t)
            return(False)
        tomb = []
        while True:
            sor = ssf.readline()
            if sor ==":
                ssf.close()
                os.chdir('..')
                if tomb == []:              # Ha üres a napi archivum
                    sajat.torol()
                    t = sajat.vaszon.create_text(0,0, anchor=NW, text='Ezen a napon nem történt
esemény!', fill = "black", font=('Arial', 14))
                    sajat.obj.append(t)
                return(tomb)
            tomb.append(sor)


    def torol(sajat):
        for gobj in sajat.obj:
            sajat.vaszon.delete(gobj)
        sajat.obj = []


    def frissit(sajat):
        global datum, mai


        if sajat.mask !=[var.get() for var in sajat.states]: #Ha változott a szűrőfeltétel
            sajat.mask =[var.get() for var in sajat.states]
            sajat.torol()
            sajat.kitesz()


        if datum == mai:
            try:
                fc = open("Server_Event\\Ev_flag.txt",'r') # megnézi, hogy jött-e új adat
                fc.close()
```

```
            print("jött új esemény!!")
            os.remove("Server_Event\\Ev_flag.txt")
            sajat.fetch(datum)
        except:
            fc = False
    sajat.after(1000,sajat.frissit) # várunk 1 másodpercet

if __name__ == '__main__':
    Draw().mainloop()
```

Get_data_from_FTP.py

```
# Get data from FTP server
# titkosítja a fájlok nevét
# beolvassa a pillanatérték fájlokat a felhőből
# os.system('time 10:20:50')

import os, time
from time import sleep
from ftplib import FTP
import  encrypt
from db import Cserel
import EventS

def Kirak(tomb):
    # fájlba menti a friss adatokat

    Serv_dirname = "Server_Data\\Pillanat"
    Serv_dirnapi = "Server_Data\\Napi_"
    t = time.localtime(time.time())
    datum = '%4d%1s%02d%1s%02d' % (t[0],'.',t[1],'.',t[2])
    tt= '%02d%1s%02d%1s%02d' % (t[3],':',t[4],':',t[5])

    nyers_idop = tomb[:20]
    idop = Cserel(nyers_idop,';',")
    data = tomb[21:]

    ii = 0
    while(True):

        i_dat = data[ii:].find('\n')
        if i_dat == -1: break            # tömbnek vége

        i_nev = data[ii:].find(';')
        if i_nev == -1:
            print("adat hiba",data[ii:])
            breakFTP_aktual
        tagname = data[ii:ii+i_nev]
        nyers_adat =  data[ii+i_nev+1:ii+i_dat]
```

```
        ertek = Cserel(nyers_adat,'.',',')
        ii = ii + i_dat + 1

        try:                        # pillanatértékek
            fp = open(Serv_dirname+'\\'+tagname+'.csv','w')
            fp.write(idop+';'+ertek)
            fp.close()
        except:
            print("Nem sikerült megnyitni: ",Serv_dirname,tagname+'.csv')

        try:                        # napi adatok
            if os.path.exists(Serv_dirnapi+datum) == False:      # nincs ilyen direktori akkor
megnyitja
                os.mkdir(Serv_dirnapi+datum)
            fn = open(Serv_dirnapi+datum+'\\'+tagname+'.csv','a')
            fn.write(idop[11:]+';'+ertek+'\n')
            fn.close()
        except:
            print("Nem sikerült megnyitni: ",Serv_dirnapi,datum,tagname+'.csv')
    print('>Get ',tt)

def Ev_Ki(tomb):
    # fájlba menti a friss eseményeket

    t = time.localtime(time.time())
    datum = '%4d%1s%02d%1s%02d' % (t[0],'.',t[1],'.',t[2])

    Serv_event = "Server_Event"

    try:                        # napi adatok
        if os.path.exists(Serv_event) == False:      # nincs ilyen direktori akkor megnyitja
            os.mkdir(Serv_event)
        fn = open(Serv_event+'\\Event'+datum+'.csv','a')
        fn.write(tomb)
        fn.close()
        print("Esemény jött",tomb)
        fn = open(Serv_event+'\\Ev_flag.txt','w')
        fn.close()
    except:
         print("Nem sikerült megnyitni: ",Serv_event+'\\Event'+datum+'.csv')
```

```
# Itt kezdődik a program


print("Get_data")
encrypt.Feltolt('S_doboz.csv')
#encrypt.Kulcs_generalas('412163d9')
encrypt.Kulcs_generalas('415263d9') # legenerálja a kulcsot

ftpx = FTP('cp1.ezit.hu')
ftpx.login('kovari@scadasys.hu','kb1118BP')
sor ="Kliens_Data\\Aktual"
FTP_aktual = encrypt.Koder(sor[0:16])+encrypt.Koder(sor[16:])+".dat" # az adatfájl kódolt neve
sor ="Kliens_Event\\Event"
FTP_aktual_ev = encrypt.Koder(sor[0:16])+encrypt.Koder(sor[16:])+".dat" # az adatfájl kódolt
neve



var = 0
while(True):
# folyamatosan kérdezi a felhőből az aktuális adatokat
# beleteszi a pillnatértékeket a helyükre
# frissíti az archívumokat
# naponta egyszer szikronizál 12 óra 12 perc 12 másodperc



    t = time.localtime(time.time())
    td = '%4d%1s%02d%1s%02d' % (t[0],'.',t[1],'.',t[2])
    tt= '%02d%1s%02d%1s%02d' % (t[3],':',t[4],':',t[5])
    if t[3]==00 and t[4]==45 and t[5]==00: # Ha itt az idő
        ftpx.storbinary('STOR '+ 'Time_synchron.csv',open('Time_synchron.csv','rb')) # elküldjük a
kérést
        print("Idoszinkron Server")
    if var ==60:
        print("Megszakadt a kapcsolat a klienssel!!")
        hiba = td+'  '+ tt+';' + "System ;Megszakadt a kapcsolat a klienssel!\n"
        EventS.Log(hiba)
    dirlist =FTP.nlst(ftpx)
    if FTP_aktual in dirlist: # Ha van friss adat
        LocFile = open(FTP_aktual, "wb")
```

```
    ftpx.retrbinary('RETR '+ FTP_aktual,LocFile.write)
    LocFile.close()
    eredmeny = encrypt.Beo(FTP_aktual)
    Kirak(eredmeny)
    FTP.delete(ftpx,FTP_aktual)
    if var > 60:
        print("Helyreállt a kapcsolat a klienssel!!")
        t = time.localtime(time.time())
        td = '%4d%1s%02d%1s%02d' % (t[0],'.',t[1],'.',t[2])
        tt= '%02d%1s%02d%1s%02d' % (t[3],':',t[4],':',t[5])
        hiba = td+'  '+ tt+';' + "System ;Helyreállt a kapcsolat a klienssel!\n"
        EventS.Log(hiba)
    var = 0
else: var = var +1
if FTP_aktual_ev in dirlist: # Ha van friss esemény
    LocFile = open(FTP_aktual_ev, "wb")
    ftpx.retrbinary('RETR '+ FTP_aktual_ev,LocFile.write)
    LocFile.close()
    eredmeny = encrypt.Beo(FTP_aktual_ev)
    Ev_Ki(eredmeny)
    FTP.delete(ftpx,FTP_aktual_ev)
sleep(1)
```

Get_data_from_kliens.py

```
# Get data from Kliens
# beolvassa a pillanatérték fájlokat a klienstől
#net use \\10.145.165.251\Kliens_Data admin
#net use \\10.145.165.251\Kliens_Data admin /user:admin

import os, time, shutil
from time import sleep
from db import Cserel
import EventS

def Kirak(csvfile):
    # fájlba menti a friss adatokat

    tomb = ''
    tmp = open(csvfile,'r')
    while True:
        sor = tmp.readline()
        if sor == '': break
        tomb= tomb + sor
    Serv_dirname = "Server_Data\\Pillanat"
    Serv_dirnapi = "Server_Data\\Napi_"
    t = time.localtime(time.time())
    datum = '%4d%1s%02d%1s%02d' % (t[0],'.',t[1],'.',t[2])
    tt= '%02d%1s%02d%1s%02d' % (t[3],':',t[4],':',t[5])

    nyers_idop = tomb[:20]
    idop = Cserel(nyers_idop,';',"")
    data = tomb[21:]

    ii = 0
    while(True):

        i_dat = data[ii:].find('\n')
        if i_dat == -1: break            # tömbnek vége

        i_nev = data[ii:].find(';')
        if i_nev == -1:
```

```
            print("adat hiba",data[ii:])
            break
        tagname = data[ii:ii+i_nev]
        nyers_adat =  data[ii+i_nev+1:ii+i_dat]
        ertek = Cserel(nyers_adat,'.',',')
#         print(tagname,nyers_adat,ertek)
        ii = ii + i_dat + 1

        try:                        # pillanatértékek
            fp = open(Serv_dirname+'\\'+tagname+'.csv','w')
            fp.write(idop+';'+ertek)
            fp.close()
        except:
            print("Nem sikerült megnyitni: ",Serv_dirname,tagname+'.csv')

        try:                        # napi adatok
            if os.path.exists(Serv_dirnapi+datum) == False:      # nincs ilyen direktori akkor
megnyitja
                os.mkdir(Serv_dirnapi+datum)
            fn = open(Serv_dirnapi+datum+'\\'+tagname+'.csv','a')
            fn.write(idop[11:]+';'+ertek+'\n')
            fn.close()
        except:
            print("Nem sikerült megnyitni: ",Serv_dirnapi,datum,tagname+'.csv')
    print('>Get ',tt)

def Ev_Ki(csvfile):
    # fájlba menti a friss eseményeket

    tomb = ''
    tmp = open(csvfile,'r')
    while True:
        sor = tmp.readline()
        if sor == '': break
        tomb= tomb + sor
    t = time.localtime(time.time())
    datum = '%4d%1s%02d%1s%02d' % (t[0],'.',t[1],'.',t[2])

    Serv_event = "Server_Event"
```

```
    try:                        # napi adatok
        if os.path.exists(Serv_event) == False:      # nincs ilyen direktori akkor megnyitja
            os.mkdir(Serv_event)
        fn = open(Serv_event+'\\Event'+datum+'.csv','a')
        fn.write(tomb)
        fn.close()
        print("Esemény jött",tomb)
        fn = open(Serv_event+'\\Ev_flag.txt','w')
        fn.close()
    except:
         print("Nem sikerült megnyitni: ",Serv_event+'\\Event'+datum+'.csv')



# Itt kezdődik a program

print("Get_data_from_Kliens")
#os.system('net use \\10.145.165.251\Kliens_data admin /user:admin')
#print(os.system('net use \\10.145.165.251\Kliens_Event admin /user:admin'))

ppp = '\\\\10.145.165.251\Kliens_data'
pev = '\\\\10.145.165.251\Kliens_Event'

jo_volt = True
while(True):
# folyamatosan kérdezi a klienstől az aktuális adatokat
# beleteszi a pillnatértékeket a helyükre
# frissíti az archívumokat



    t = time.localtime(time.time())
    td = '%4d%1s%02d%1s%02d' % (t[0],'.',t[1],'.',t[2])
    tt= '%02d%1s%02d%1s%02d' % (t[3],':',t[4],':',t[5])
    try:
        dirlist =os.listdir(ppp)
        if not jo_volt:
            jo_volt = True
            print(tt," Helyreállt a kapcsolat a klienssel!!")
            hiba = td+'  '+ tt+';' + "System ;Helyreállt a kapcsolat a klienssel!\n"
            EventS.Log(hiba)
```

```
    if 'Aktual.csv' in dirlist: # Ha van friss adat
        shutil.copy(ppp+'\Aktual.csv','tmp.csv')
        Kirak('tmp.csv')
        os.remove(ppp+'\Aktual.csv')
except:
    if jo_volt:
        jo_volt = False
        print(tt," Megszakadt a kapcsolat a klienssel!!")
        hiba = td+'  '+ tt+';' + "System ;Megszakadt a kapcsolat a klienssel!\n"
        EventS.Log(hiba)
try:
    dirlist =os.listdir(pev)
    if 'Event.csv' in dirlist: # Ha van friss esemény
        shutil.copy(pev+'\Event.csv','teve.csv')
        Ev_Ki('teve.csv')
        os.remove(pev+'\Event.csv')
except:
    if jo_volt:  jo_volt = False
sleep(1)
```

Matek.py

```
# Matematika modul

import os
from time import sleep
import db

db.Inic() # Feltöltjük az adatformátumot

# itt kezdődnek a képletek
print("Matematika program elindult")

while(True):


    Ur = db.GET("Fázis_fesz_R")
    db.PUT("Fázis_fesz_S",Ur*0.9)
    db.PUT("Fázis_fesz_T",Ur*1.1)

    Ir = db.GET("Fázis_áram_R")
    db.PUT("Fázis_áram_S",Ir*0.9)
    db.PUT("Fázis_áram_T",Ir*1.1)

    Pr = db.GET("Teljesitmény_R")
    db.PUT("Teljesitmény_S",Pr*0.8)
    db.PUT("Teljesitmény_T",Pr*1.2)


    cos = db.GET("COS_fi")

    db.PUT("Meddö_R",Ur*Ir*cos/1000)
    db.PUT("Meddö_S",Ur*Ir*0.8*cos/1000)
    db.PUT("Meddö_T",Ur*Ir*1.2*cos/1000)




    for i in range(5):      # 2-6 fogyasztást szimulálja
```

```
    Fogy = db.GET("Fogyasztás"+str(i+2))
    db.PUT("Fogyasztás"+str(i+2),Fogy+1.3)



sleep(5) #5 másodperces várakozás
```

Szamlazo.py

```
# Számla készítő program

import os, zipfile, shutil, time
import db
import zipfile

adatok = [],[]      # a konfig fájlban szereplő paraméterek listája
szamxml = ''        # sheet1.xml fájl második sorának tartalma
stringxml = ''      # sharedStrings.xml fájl második sorának tartalma
afa = 0             # az érvényes ÁFA 0.27 alakban
szamlasorszam = 0   # az érvényes számla sorszám. Mindig nő eggyel.
offset = 0          # az eltolás kulcsa kul

def Inic(bemenet):
# feldolgozza a konfig (bemenet) fájlt és bepakolja az adatok-ba
    global adatok

    try:
        beo = open(bemenet,'r')
    except:
        print('open error:',bemenet)

    sor = beo.readline() # első sort eldobjuk
    fi = 0                  # fogyasztó sorszám
    while True:
        sor = beo.readline() #
        if sor == '': break  # ha vége a leírónka
        sor = sor.strip()
        sor = sor + ';'
        if sor[0] != '#' and sor[0] != ';':
            tol = 0
            for i in range(7):
                hosz = sor[tol:].find(';')
                if hosz < 1:       # hibás sor
                    print("Hibás mező: ",sor,i)
                    break
                else:
```

```
                adatok[fi].append(sor[tol:tol+hosz])
            tol = tol + hosz + 1
        fi = fi +1 # jön a következő fogyasztó
    beo.close()

def szamkio(pat):
    global szamxml
# kiolvassa a szamxml bináris stringből a pat utáni első számot
# output a szám egész formában
    r = szamxml.find(pat)
    ve = szamxml[r:].find(b'<v>')
    vv = szamxml[r:].find(b'</v>')
    num = szamxml[r+ve+3:r+vv]
    return(int(num))

def szamcsere(pat, uj):
    global szamxml
# kicseréli a szamxml bináris stringben a pat utáni első számot uj-ra

    r = szamxml.find(pat)
    v = szamxml[r:].find(b'<v>')
    num = szamxml[r+v:].find(b'</v>')-3
    us =szamxml[:r+v+3] + str(uj).encode() + szamxml[r+v+3+num:]
    szamxml = us

def stringcsere(pat, uj):
    global szamxml, stringxml
# kicseréli a stringxml bináris stringben az nn-edik stringet uj-ra

    r = szamxml.find(pat)
    v = szamxml[r:].find(b'<v>')
    num = szamxml[r+v:].find(b'</v>')-3
    nn = int(szamxml[r+v+3:r+v+3+num])+1
    r = 0
    for i in range(nn):
        rk = stringxml[r:].find(b'<si><t>')
        r = r+rk+1

    v = stringxml[r:].find(b'</t>')
```

```
    us =stringxml[:r+6] + str(uj).encode() + stringxml[r+v:]
    stringxml = us

def Havi_fogy(azon,akt_honap):
    # kiszámítja az előző havi fogyasztást
    # input azon: adatbázis azonosító
    # input akt_honap: az aktuális dátumból az év és hónap azonosító
    # output: a fogyaszás egész számmal

    os.chdir('..')
    os.chdir('..')

    eredm = []    # előző havi érték, mostani érték, különbség (fogyasztás)
    try:
        beo = open("Server_Data/Havi_fogyasztas/"+azon+'.csv','r')
    except:
        print('open error: Server_Data/Havi_fogyasztas/'+azon+'.csv')

    while(True):
        sor = beo.readline()
        if sor == '':
            print("nincs az adott hónaphoz fogyasztás",azon,akt_honap)
            break
        hosz = sor.find(';')
        if sor[:hosz] == akt_honap:     # ha megtaláltuk az adott hónapot
            akt = int(sor[hosz+1:].strip())
            eredm.append(elozo)
            eredm.append(akt)
            eredm.append(akt-elozo)
            beo.close()
            os.chdir('szamla/')
            os.chdir('munka/')
            return(eredm)
        else:
            elozo = int(sor[hosz+1:].strip())



def Datum_szam(tt):
    # átszámítja a kapott dátumot 1900-tól kezdődő napi száláló értékre
```

```
    # input: 2017.11.14
    # output: 43418
    hon = 0,31,59,90,120,151,181,212,243,273,304,334
    hos = 0,31,60,91,121,152,182,213,244,274,305,335
    #      2018  2019  2020  2021
    evek = 43100,43465,43830,44196

    szam = evek[int(tt[0])-2018]
    if int(tt[0])%4 !=0: # Ha nem szökőév
        szam = szam + hon[int(tt[1])-1]
    else:
        szam = szam + hos[int(tt[1])-1]
    return(szam + int(tt[2]))

def Idoszak(tt):

    idosz = []    # időpont vektor
    eredm = []    # az elszámolási időszak kezdete, vége napoknan 1900-tól
    ho = 31,28,31,30,31,30,31,31,30,31,30,31
    if tt[1] != 1:  # ha nem január van
        idosz.append(tt[0])
        idosz.append(tt[1]-1)
        idosz.append(1)
        kezd = Datum_szam(idosz)
        idosz[2] = ho[tt[1]-2]
        veg = Datum_szam(idosz)
    else:
        idosz.append(tt[0]-1)
        idosz.append(12)
        idosz.append(1)
        kezd = Datum_szam(idosz)
        idosz[2] = 31
        veg = Datum_szam(idosz)
    eredm.append(kezd)
    eredm.append(veg)
    return(eredm)

print("Számlázó program elindult")
Inic("Szamla_leiro.csv")
```

```
t = time.localtime(time.time())
mai_datum = Datum_szam(t)


# átmásoljuk a munka könyvtárba
os.chdir('szamla/')

#szamlasorszam
kulcs = open("num.key",'rb')
of = int(kulcs.read())      # Sorszám
kulcs.close()
szamlasorszam = 123456689 + (of^0x55)

ddd = []
for root, dirs, files in os.walk('munka'): # ha létezik le kell törölni a fájlokat
    for file in files:
        if root not in ddd:
            ddd.append(root)
        os.remove(os.path.join(root, file))
ddd.reverse()
for dd in(ddd):                         # kitörli a könyvtárakat
    try:
        os.removedirs(dd)
    except:
        pass
os.mkdir('munka')                               # létrehozza a kitörölt munka könyvtárat
shutil.copy('Etalon_szamla.xlsx','munka/')          # átmásolja az etalon excel fájlt
os.chdir('munka/')
os.rename('Etalon_szamla.xlsx','Etalon_szamla.zip')     # átnevezi Zip fájlnak
zz = zipfile.ZipFile('Etalon_szamla.zip','r')
zipfile.ZipFile.extractall(zz)                  # kibontja
zz.close()
os.remove('Etalon_szamla.zip')                      # letörli a zip-et

minta = open("xl/worksheets/sheet1.xml",'rb')       # kicseréljük a számokat
sor1 = minta.readline()
szamxml = minta.readline()
minta.close()
```

```
egysegar = szamkio(b'r="B17"')
afa = szamkio(b'r="C17"')/100

minta = open("xl/sharedStrings.xml",'rb')            # kicseréljük a szövegeket
sor2 = minta.readline()
stringxml = minta.readline()
minta.close()
shutil.copy('../image1.png','xl/media/')             # bemásoljuk az emblémát

irtak = False
for adda in adatok:
   szamcsere(b'r="F2"',szamlasorszam)            # számla sorszám
   szamlasorszam = szamlasorszam +1
   fogy = Havi_fogy(adda[0],str(t[0])+'.'+str(t[1]))

   szamcsere(b'r="A17"',fogy[2])                 # Fogyasztás
   szamcsere(b'r="D17"',egysegar*fogy[2])        # Nettó díj
   szamcsere(b'r="E17"',egysegar*fogy[2]*afa)    # ÁFA Ft
   szamcsere(b'r="F17"',egysegar*fogy[2]*(1+afa))  # Bruttó
   szamcsere(b'r="D21"',egysegar*fogy[2]*(1+afa))  # Fizetendő

   szamcsere(b'r="C15"',mai_datum)
   szamcsere(b'r="D15"',mai_datum)
   szamcsere(b'r="E15"',mai_datum+int(adda[6]))

   isz = Idoszak(t)
   szamcsere(b'r="B19"',isz[0])
   szamcsere(b'r="C19"',isz[1])

   szamcsere(b'r="B20"',fogy[0])
   szamcsere(b'r="C20"',fogy[1])

   stringcsere(b'r="D5"',adda[1])
   stringcsere(b'r="D6"',adda[2])
   stringcsere(b'r="D7"',adda[3])
   stringcsere(b'r="E9"',adda[4])
   stringcsere(b'r="E10"',adda[5])

   kesz = open("xl/worksheets/sheet1.xml",'wb')
```

```
    kesz.write(sor1)
    kesz.write(szamxml)
    kesz.close()

    kesz = open("xl/sharedStrings.xml",'wb')
    kesz.write(sor2)
    kesz.write(stringxml)
    kesz.close()


    # összeZIP-eljük
    os.chdir('..')
    os.chdir('Elkeszult_szamlak')
    zipf = zipfile.ZipFile('Zipped_file.zip', 'w', zipfile.ZIP_DEFLATED)
    os.chdir('..')
    os.chdir('munka/')
    path1 ='.'
    for root, dirs, files in os.walk(path1):
        for file in files:
            zipf.write(os.path.join(root, file))
    zipf.close()
    os.chdir('..')
    os.chdir('Elkeszult_szamlak')
    filename = adda[0]+'_'+str(t[0])+'_'+str(t[1])+'.xlsx'
    try:
        os.rename('Zipped_file.zip',filename)
        print("Elkészült: ",filename)
        irtak = True
    except:
        print("Van már ebben a hónapban ilyen számla! ",filename)
        os.remove('Zipped_file.zip')

if irtak:
    os.chdir('..')
    kulcs = open("num.key",'wb')
    of1 = (szamlasorszam-123456689) ^ 0x55
    kulcs.write(str(of1).encode())
    kulcs.close()
```

Menetrend.py

```
#  Menetrend modul
import os, time
from tkinter import *
import db

adatsor = []
colors = ["seagreen3","palegreen","orange1","orangered1","red","deeppink3"]

def trend_beolvas():
    # bealvassa az terv adatait
    global adatsor

    dir_terv = "Menetrend"
    try:
        print("terv_beolvaso",dir_terv)
        os.chdir(dir_terv)
        fp = open('Fogyasztas_terv.csv','r')
        for tn in range(12):
            sor = fp.readline()
            a = float(sor[8:].strip())
            adatsor.append(a)
        os.chdir('..')
        fp.close()
    except:
        print("Hiányzik a terv fájl: ",dir_terv)

class Draw(Frame):
    #"A program ablakát definiáló osztály"
    def __init__(sajat):
        Frame.__init__(sajat)
        # A vászon létrehozása:
        Label(sajat, text ='Menetrend').pack(side=TOP, padx=40)
        sajat.hatterszin = 'white'
        sajat.vaszon = Canvas(sajat, width =75, height =75, bg = sajat.hatterszin)
        sajat.vaszon.pack(padx =0, pady =0)
        # Az <egéresemények> hozzákapcsolása a (vászon) widget-hez :
        sajat.vaszon.bind("<Button-1>", sajat.mouseDown)
```

```
    sajat.pack()
    sajat.nyitott = ''
    sajat.atlepve = False
    trend_beolvas()
    sajat.rajzol()
    sajat.frissit()


def rajzol(sajat):  # felrajzolja kicsiben a terv oszlopait
    global adatsor
    posX = 10
    posY = 75
    for adat in adatsor:
        sajat.vaszon.create_rectangle(posX,posY, posX+5,posY-adat/200, fill = "moccasin")
         # bar diagram eleje
        posX = posX +5

def trendablak(sajat,cim):
    global adatsor

    if sajat.nyitott != '':
        sajat.nyitott.destroy()
    sajat.trend = Toplevel(sajat)
    sajat.canvas = Canvas(sajat.trend, width=300, height=300, bg = 'white')
    sajat.nyitott = sajat.trend
    sajat.trend.title(cim)
    sajat.canvas.pack(side=TOP)
    posX = 30
    posY = 275
    i = 1
    for adat in adatsor:
        sajat.canvas.create_rectangle(posX,posY, posX+20,posY-adat/50, fill = "moccasin")
         # bar diagram eleje
        if i == sajat.honap:
            sav = adat//5
            colindex = int(sajat.ertek//sav)
            if colindex > 4: colindex =4
            color = colors[colindex]
            sajat.canvas.create_rectangle(posX,posY, posX+20,posY-int(sajat.ertek/50), fill =
```

```
            color)                          # aktual hónap
            posX = posX +20
            sajat.canvas.create_text(posX-5,290, text='%2i'% i, anchor=E)

            i = i+1

    def frissit(sajat):

        t = time.localtime(time.time())
        sajat.honap = t[1]
        sajat.ertek = db.GET('Fogyasztás1')
        if sajat.ertek > adatsor[sajat.honap-1]:
            if sajat.atlepve == False:          # Ha most lépte át a határt
                print(t[0],':',t[1],':',t[2]," Határérték túllépés a",t[1]," hónapban: ",sajat.ertek)
                sajat.trendablak('Határérték átlépés')
                sajat.atlepve = True
        else:
            sajat.atlepve = False

        sajat.after(1000,sajat.frissit) # várunk 1 másodpercet



    def mouseDown(sajat, event):
        global pic

        #"Balegérgomb lenyomására végrehajtandó művelet"
        # event.x és event.y tartalmazzák a kattintás koordinátáit :
        sajat.trendablak('Terv/Mérés')

if __name__ == '__main__':
    Draw().mainloop()
```