ProCem adapteriohjelmat

ProCem

This work has been sponsored by Business Finland and the following companies or organizations: ABB Oy, Empower IM Oy, Headpower Oy, KONE Oyj, Landis+Gyr Oy, MX Electrix Oy, Wapice Oy, Fingrid Oyj, Elenia Oy, Jyväskylän Energia Oy, Lempäälän Energia Oy, Loiste Sähköverkko Oy, Satapirkan Sähkö Oy, Tampereen Sähköverkko Oy, Suomen Yliopistokiinteistöt Oy, and Sitra.

Main author of this document: Ville Heikkilä

Sisällysluettelo

1 Johdanto

Tässä dokumentissa kerrotaan ProCem-projektissa käytetyistä datalähteistä ja niistä dataa lukevista adapteriohjelmista.

Jokaiselle datalähteelle on oma adapteriohjelmansa, joka lukee halutun datan, muuntaa sen yhteiseen IoT-Ticket-yhteensopivaan muotoon ja lähettää edelleen Procem RTL -ohjelmalle, joka hoitaa datan tallennuksen ja lähetyksen IoT-Ticket-tietokantaan (ohjelman kuvaus erillisessä dokumentissa). Adapteriohjelmien kommunikaatio Procem RTL -ohjelman kanssa on toteutettu UDP-protokollalla ja kaikki ohjelmat on kirjoitettu Python-ohjelmointikielellä.

Eri ohjelmia käynnistäessä pidempää ajoa varten kannattaa käyttää esim. "screen"-ohjelmaa, koska ohjelmat eivät käynnisty taustalle, vaan valtaavat komentotulkin. Kaikki datan keräykseen tarvittavat ohjelmat voi käynnistää kerralla "start_procem.sh"-tiedostossa olevan skriptin avulla (olettaen siis, että mikään niistä ei ole valmiiksi käynnissä) komennolla `./start_procem.sh` (skripti käynnistää kaksi screen sessiota: *procem_rtl*, josta löytyy ProCem RTL –komponentti, datan pakkausohjelma sekä prosessien uudelleenkäynnistäjä, ja *adapters*, josta löytyy kaikki käynnistetyt adapteriohjelmat). Vaihtoehtoisesti voi käyttää skriptiä ”simple_start.sh” (komento: `./simple_start.sh`), jolla käynnistetään prosessien uudelleenkäynnistäjä, joka hoitaa muiden komponenttien käynnistämisen.

2 Laatuvahti 3

Sähkön laadun mittauksia Kampusareenalta.

2.1 Konfiguraatiotiedostot

- `mxelectrix_main_config.json`

Pääkonfiguraatiotiedosto, joka annetaan adapterille käynnistettäessä komentoriviparametrina. Tiedostossa määritellään kahden muun parametritiedoston nimet sekä ip-osoite ja porttinumero Laatuvahti-mittarille.

- `mxelectrix_model_config.json`

Tässä tiedostossa määritellään csv-tiedostosta luettavien sarakkeiden nimet. Tätä tiedostoa ei todennäköisesti tarvitse muuttaa myöhemmin.

- `Laatuvahti_measurement_IDs.csv`

Tässä tiedostossa määritellään eri mittaussuureiden id-numerot ja käytetyt nimet ja polut IoT-Ticket-tallennusta varten kukin mittaussuure omalla rivillään. Sarakkeen *serial_number* avulla mittaus yhdistetään oikeaan Laatuvahti-mittariin ja sarakkeen *laatuvahti_name* avulla oikeaan mittaussuureeseen. Käytetyt mittayksiköt luetaan Laatuvahti-mittarin kautta, joten niitä ei aseteta tässä tiedostossa. Sarakkeeseen *iot_ticket* on merkitty "`x`" niiden mittauksien kohdalle, jotka lähetetään myös IoT-Ticketin tietokantaan. Repositoriossa on valmiina pohja, jossa on esimerkki mittaussuurerivistä.

2.2 Adapteriohjelma

Adapteriohjelma on kirjoitettu Python-ohjelmointikielellä. Mittauksien muuntamiseen yhdenmukaiseen muotoon ja lähettämiseen käytetään kaikille adapteriohjelmille yhteisiä ohjelmakomponentteja. Adapteriohjelman käyttämät tiedostot:

- `adapters/mxelectrix_adapter.py`
- `adapters/mxelectrix_model.py`
- `adapters/mxelectrix_parser.py`
- `adapters/common_utils.py`
- `adapters/mxelectrix_main_config.json`
- `adapters/mxelectrix_model_config.json`
- `adapters/Laatuvahti_measurement_IDs.csv`

Ohjelma käynnistetään komennolla:

```
python3 mxelectrix_adapter.py mxelectrix_main_config.json
```

2.3 Mittaukset

Mittausarvot ovat erilaisia sähkönlaatua kuvaavia suureita. Laatuvahteja ProCem:n datan keräyksessä on yhteensä 9. Jokaisesta mittarista tulee tällä hetkellä 179 eri mittausta, kukin mittaus sekunnin välein. Koska mittausarvoja tulee kohtuullisen paljon, vain osa mittausdatasta lähetetään edelleen IoT-Ticketin tietokantaan. Kaikki mittausarvot kuitenkin pidetään mukana omassa tallennuksessa.

2.4 Datan lukeminen

Adapteriohjelma luo käynnistyessään TCP-serverin, johon Laatuvahdin mittarit ottavat yhteyden XPORT-moduulin kautta. Yhteys käynnistyy automaattisesti olettaen, että mittarit ovat päällä ja konfiguraatiot on asetettu oikein. Kun yhteys mittariin on muodostettu, adapteriohjelma käynnistää mittausarvojen lähetyksen (datavirran). Lähetys käynnistetään varmistamalla ensin, että mittarin on ns. vapaassa tilassa eli esim. aikaisempi datavirta ei ole enää päällä ja lähettämällä sitten mittarille datavirran avauskäskyt. Tarvittaessa edellinen datavirta keskeytetään lähettämällä sulkemisviesti ja käynnistämällä yhteys uudestaan. Uuden datavirran alussa mittari lähettää viestin, josta selviää mitä suureita mittari lähettää sekä suureiden yksiköt. Datavirran käynnistyksen ja suureiden sekä yksiköiden lukemisen jälkeen mittari alkaa lähettää noin sekunnin välein mittausarvoja ilmoittamilleen suureille. Adapteriohjelma yhdistää saadut mittausarvot yksiköihin ja konfiguraatiotiedostossa määriteltyihin suureiden nimiin ja lähettää muodostetun datan Procem RTL -ohjelmalle.

Mittausarvojen aikaleima otetaan ohjelmaa ajavan koneen kellon mukaan heti mittauksien saapumisen jälkeen.

3 Sähköenergiatekniikan sääasema

Laadukas aurinkosähkön tutkimusvoimala. Jussi Aholan diplomityö liittyen tutkimusvoimalan mittaus- ja tallennusjärjestelmään: Photovoltaic research power plant measuring and data storing system

3.1 Konfiguraatiotiedostot

- `postgres_weatherstation_config.json`

Tässä tiedostossa määritellään sääaseman tietojen lukemiseen tarvittavan tietokantayhteyden parametrit sekä käytettyjen tietokantataulujen nimet ja taulujen lukuaikavälit. Lisäksi määritellään sääaseman dataloggerin ja ohjelmaa ajavan koneen kellojen aikaeron määritykseen liittyvät parametrit. Alla joidenkin parametrien selityksiä:

- `address:` tietokannan osoite
- `port:` tietokannan portti
- `name:` tietokannan nimi
- `username:` käyttäjän nimi tietokantaan
- `password:` käyttäjän salasana
- `time_offset_s:` oletusarvo tietokannan ja ohjelmaa ajavan koneen aikaleimojen erolle
- `time_offset_check:` suoritetaanko aikaleimojen eron automaattinen määritys
- `time_offset_parameters:` aikaleimojen eron määritykseen liittyvät parametrit
- `tables:` luettavat tietokantataulut
    - `interval_ms:` vähimmäisaikaväli millisekunteina kahden peräkkäisen talteenotetun mittauksen välillä
    - `delay_ms:` kahden perättäisen tietokantahaun välinen aika millisekunteina
    - `time_field:` sarake, josta löytyvät mittauksien aikaleimat

Lisäksi tässä tiedostossa määritellään csv-tiedostosta luettavien sarakkeiden nimet.

- `Wheather_Station_measurement_IDs.csv`

Tässä tiedostossa määritellään eri mittaussuureiden id-numerot ja käytetyt nimet, polut ja yksiköt IoT-Ticket-tallennusta varten. Sarakkeen *table* avulla mittaus yhdistetään oikeaan tietokantatauluun ja sarakkeen *field* avulla oikeaan mittaussuureeseen. Koska nyt on käytössä vain yksi tietokanta, josta lukuja tehdään, saraketta *database_id* ei ole laitettu mukaan (sarakkeen puuttuessa käytetään oletusarvoa 1), mutta tarvittaessa sitä voi käyttää erottelemaan eri tietokannoista luettavia arvoja. Repositoriossa on valmiina pohja, jossa on esimerkki mittaussuurerivistä.

3.2 Adapteriohjelma

Adapteriohjelma on kirjoitettu Python-ohjelmointikielellä ja tarvittavaan PostgreSQL-tietokantayhteyden muodostamiseen on käytetty apuna psycopg2-kirjastoa (julkaistu LGPL-lisenssin alla). Tarvittaessa kirjaston voi asentaa esimerkiksi komennolla:

```
pip install psycopg2 --user
```

Mittauksien muuntamiseen yhdenmukaiseen muotoon ja lähettämiseen käytetään kaikille adapteriohjelmille yhteisiä ohjelmakomponentteja. Adapteriohjelman käyttämät tiedostot:

- `adapters/postgres_generic_adapter.py`
- `adapters/postgres_utils.py`
- `adapters/postgres_weatherstation_model.py`
- `adapters/common_utils.py`
- `adapters/postgres_weatherstation_config.json`
- `adapters/Wheather_Station_measurement_IDs.csv`

Ohjelma käynnistetään komennolla:

```
  python3 postgres_generic_adapter.py
postgres_weatherstation_config.json
Wheather_Station_measurement_IDs.csv
```

3.3 Mittaukset

Sääasemalta kerättäviä mittauksia ovat ilman lämpötila, ilmankosteus, tuulen nopeus ja suunta sekä Auringon säteilyvoimakkuus. Mittausarvoja on saatavilla 100 millisekunnin välein, mutta arvoja haettiin projektissa vain yksi mittaus kutakin sekuntia kohti.

3.4 Datan lukeminen

Mittausarvoja on saatavilla PostgreSQL-tietokannasta, johon sääaseman dataloggeri lähettää mittaukset. Koska tietokantaa päivitetään aina 10 sekunnin mittausdatalla kerrallaan, käy tehty adapteriohjelma lukemassa tietokannasta uudet mittausarvot aina 10 sekunnin välein. PostgreSQL-tietokanta on asetettu hyväksymään yhteydenoton ohjelmaa ajavan koneen IP-osoitteesta. Käytössä on tunnukset, joilla on lukuoikeudet tarvittaviin tauluihin.

3.5 Mittauksien aikaleimat

Tietokannassa olevien mittauksien aikaleima tulee sääaseman dataloggerilta, jonka kellossa ei käytetä kesäaikaa ja joka on edistänyt sääaseman käynnistyksestä 2011 tähän hetkeen noin 12 minuuttia. Dataloggerin kellon edistäminen on ollut lineaarista eli kellon edistää on noin sekunnin neljässä vuorokaudessa. ProCem:n datan tallennuksessa mittauksien aikaleimoina käytetään kuluneita millisekunteja ajanhetkestä 1.1.1970 (UTC) (UNIX-aika millisekunnin tarkkuudella) ja referenssikellona käytetään Linux-konetta, jossa kaikki ohjelmat pyörivät. Sääaseman dataloggerin kellon edistämisen vuoksi adapteriohjelma tutkii automaattisesti aikaeron Linux-koneen ja dataloggerin kellon välillä. Datan tallennusta varten laskettu aikaero lisätään tietokannasta luettuihin aikaleimoihin. Aikaeron selvittäminen tehdään aina adapteriohjelman käynnistyksen yhteydessä sekä vuorokauden vaihtuessa ja sen tekeminen kestää noin minuutin. Koska sääaseman dataloggerin kellon edistämisen on todettu olevan lineaarista, lisätään vuorokauden vaihtuessa laskettuun aikaeroon 0,125 sekuntia, että aikaleimat olisivat mahdollisimman tarkasti kohdallaan keskipäivän aikaan.

Kellojen aikaeron selvittäminen tehdään tekemällä useita nopeita tietokantakyselyjä, joiden avulla selvitetään aikaero juuri sillä hetkellä, jolloin tietokantaa on päivitetty uusilla mittausarvoilla. Mahdollisen viiveen tietokantakyselyissä (tiedon tallentamisessa ja sen hakemisessa) on oletettu olevan niin pieni, että sitä ei ole otettu huomioon.

3.6 Huomioita

Projektissa dataa luettiin vain yhdestä tietokannasta ja yhdestä taulusta, mutta samalla ohjelmalla olisi mahdollista lukea dataa myös useammasta taulusta tai useammasta tietokannasta vain konfiguraatiotiedostoja muokkaamalla.

Aikaleimojen korjaukseen tarvittavan kellojen aikaeron laskeminen voitaisiin tehdä myös useammin kuin kerran päivässä, mutta koska virheen on arvioitu olevan suurimmillaan 100 millisekunnin suuruusluokkaa, ei siihen ole nähty tarvetta.

4 Aurinkovoimalan invertterit

Kampusareenan aurinkovoimalan sähköntuotto luetaan voimalan inverttereistä.

4.1 Konfiguraatiotiedostot

- `modbus_solarplant_config.json`

Tässä tiedostossa määritellään aurinkovoimalan invertterien lukemiseen tarvittavat osoiteparametrit sekä arvojen lukuaikavälit. Alla joidenkin parametrien selityksiä:

- `model_name:` käytetyn Python-tietomallitiedoston nimi (ilman .py-päätettä)
- `devices:` laitteet, joista dataa luetaan
  - `<id-numero>:` laitteen tunnuksena käytetty id-numero
  - `ip:` laitteen ip-osoite
  - `unit_id:` laitteen väylänumero
  - `interval_ms:` aikaväli, jolla arvoja luetaan, millisekunteina
  - `delay_ms:` vähimmäisaikaväli, kahden lukupyynnön välillä
  - `max_groups:` muodostettavien rekisteriryhmien maksimimäärä
  - `max_group_size:` rekisteriryhmien maksimikoko (peräkkäisinä rekistereinä)

Lisäksi tässä tiedostossa määritellään csv-tiedostosta luettavien sarakkeiden nimet.

- `Solar_Plant_measurement_IDs.csv`

Tässä tiedostossa määritellään eri mittaussuureiden id-numerot ja käytetyt nimet, polut ja yksiköt IoT-Ticket-tallennusta varten. Sarakkeen *inverter* avulla mittaus yhdistetään oikeaan Modbus-laitteeseen ja sarakkeiden *registers* ja *count* avulla oikeaan mittaussuureeseen laitteella. Jokaiselle mittausarvolle on asetettu rekisterien sana- ja tavujärjestyksen sekä mittausarvon etumerkin olemassaolon kertovat asetusarvot sarakkeissa *wordorder*, *byteorder* ja *signed*. Jos rekistereistä luettu mittausarvo vastaa sarakkeessa *NaN_hex* olevaa asetusarvoa, asetetaan luetuksi mittausarvoksi nolla. Repositoriossa on valmiina pohja, jossa on esimerkki mittaussuurerivistä.

4.2 Adapteriohjelma

Adapteriohjelma on kirjoitettu Python-ohjelmointikielellä ja tarvittavaan Modbus-laitteen rekisterien luentaan on käytetty apuna pymodbus-kirjastoa (julkaistu BSD-lisenssin alla). Tarvittaessa kirjaston voi asentaa esimerkiksi komennolla:

```
pip install pymodbus --user
```

Mittauksien muuntamiseen yhdenmukaiseen muotoon ja lähettämiseen käytetään kaikille adapteriohjelmille yhteisiä ohjelmakomponentteja. Adapteriohjelman käyttämät tiedostot:

- `adapters/modbus_generic_adapter.py`
- `adapters/modbus_generic_model.py`
- `adapters/modbus_utils.py`
- `adapters/common_utils.py`
- `adapters/modbus_solarplant_model.py`
- `adapters/modbus_solarplant_config.json`
- `adapters/Solar_Plant_measurement_IDs.csv`

Ohjelma käynnistetään komennolla:

```
python3 modbus_generic_adapter.py modbus_solarplant_config.json
Solar_Plant_measurement_IDs.csv
```

4.3 Mittaukset

Aurinkovoimalalla on neljä invertteriä, joista kustakin kerätään taajuuteen, virtaan, jännitteeseen, tehoihin ja energioihin liittyviä mittaussuureita. Lisäksi luetaan myös invertterien tilatietoja. Tilatiedot luetaan kerran päivässä ja muut suureet kerran sekunnissa.

4.4 Datan lukeminen

Mittausarvot luetaan inverttereiltä Modbus/TCP-protokollan avulla. Invertterit päivittävät mittausarvoja laitteiden rekistereihin sekunnin välein, joten kaikki halutut arvot pyritään myös lukemaan sekunnin välein. Haluttujen mittausarvojen määritystä varten on csv-konfiguraatiotiedostoon listattu kullekin mittausarvolle aloitusrekisteri sekä kyseistä arvoa koskevien rekisterien määrä. Koska rekistereiden lukeminen on melko hidasta (keskimääräinen lukuaika on 100-200 millisekunnin välillä), ryhmitellään halutut mittausarvot rekisteriryhmiin siten, että toisiaan lähellä olevat rekisterinumerot pyritään laittamaan samaan rekisteriryhmään. Yhden rekisteriryhmän arvot luetaan sitten yhdellä kyselyllä ja halutut yksittäiset mittausarvot erotellaan saadusta vastauksesta. Modbus/TCP-protokollan kautta luettuna peräkkäisten rekisterien muodostaman ryhmän (ryhmän maksimikoko on 125 rekisteriä) lukeminen on käytännössä lähes yhtä nopeaa kuin yksittäisen rekisterin lukeminen. Tämän takia rekisteriryhmiin jaon avulla kaikki halutut arvot saadaan luettua kerran sekunnissa. Lisäksi lukuajan pienentämiseksi inverttereiden tilatiedot luetaan vain adapterin käynnistyessä sekä aina kerran uudelleen vuorokauden vaihtuessa.

Mittausarvojen aikaleima otetaan ohjelmaa ajavan koneen kellon mukaan heti rekisteriluvun jälkeen.

4.5 Huomioita

Aurinkovoimalan inverttereiden lukeminen on sujunut adapteriohjelman ajon aikana lähes häiriöttömästi. Tämän takia ohjelmaan tehtyjä häiriöstä toipumisen ominaisuuksia ei juuri käytetä ja niistä kerrotaan tarkemmin ISS:n mittarointi -kohdassa.

5 Kampusareenan sähköenergiajärjestelmän mittarointi (ISS)

Kampusareenan sähkön kulutukseen liittyviä tietoja luetaan ISS:n asentamalta mittauskeskittimeltä.

5.1 Konfiguraatiotiedostot

- `modbus_ISS_config.json`

Tässä tiedostossa määritellään ISS:n sähköenergiajärjestelmän mittaroinnin lukemiseen tarvittavat osoiteparametrit sekä arvojen lukuaikavälit. Alla joidenkin parametrien:

- `model_name:` käytetyn Python-tietomallitiedoston nimi (ilman .py-päätettä)
- `devices:` laitteet, joista dataa luetaan
  - `<id-numero>:` laitteen tunnuksena käytetty id-numero
  - `ip:` laitteen ip-osoite
  - `unit_id:` laitteen väylänumero
  - `source_ip:` yhteyden ottavan laitteen ip-osoite
  - `source_port:` yhteyden ottavan laitteen porttinumero
  - `interval_ms:` aikaväli, jolla arvoja luetaan, millisekunteina
  - `delay_ms:` vähimmäisaikaväli, kahden lukupyynnön välillä
  - `max_groups:` muodostettavien rekisteriryhmien maksimimäärä
  - `max_group_size:` rekisteriryhmien maksimikoko (peräkkäisinä rekistereinä)

Lisäksi tässä tiedostossa määritellään csv-tiedostosta luettavien sarakkeiden nimet.

- `ISS_measurement_IDs.csv`

Tässä tiedostossa määritellään eri mittaussuureiden id-numerot ja käytetyt nimet, polut ja yksiköt IoT-Ticket-tallennusta varten. Sarakkeen *inverter* avulla mittaus yhdistetään oikeaan Modbus-laitteeseen ja sarakkeiden *registers* ja *count* avulla oikeaan mittaussuureeseen laitteella. Jokaiselle mittausarvolle on asetettu rekisterien sana- ja tavujärjestyksen sekä mittausarvon etumerkin olemassaolon kertovat asetusarvot sarakkeissa *wordorder*, *byteorder* ja *signed*. Jos rekistereistä luettu mittausarvo vastaa sarakkeessa *NaN_hex* olevaa asetusarvoa, asetetaan luetuksi mittausarvoksi nolla. Repositoriossa on valmiina pohja, jossa on esimerkki mittaussuurerivistä.

5.2 Adapteriohjelma

Adapteriohjelmana käytetään parametritiedostoja lukuun ottamatta samaa ohjelmaa kuin aurinkovoimalan invertterien lukemisessa. Adapteriohjelman käyttämät tiedostot:

- `adapters/modbus_generic_adapter.py`
- `adapters/modbus_generic_model.py`
- `adapters/modbus_utils.py`
- `adapters/common_utils.py`
- `adapters/modbus_ISS_model.py`
- `adapters/modbus_ISS_config.json`
- `adapters/ISS_measurement_IDs.csv`

Ohjelma käynnistetään komennolla:

```
  python3 modbus_generic_adapter.py modbus_ISS_config.json
ISS_measurement_IDs.csv
```

5.3 Mittaukset

Kampusareenan energiajärjestelmästä luetaan muuntajan ja sähkön pääsyöttöjen mittauskeskittimistä saatavia taajuus, virta, jännite, teho ja energia-arvoja. Kukin mittausarvo luetaan 30 sekunnin välein.

5.4 Datan lukeminen

Mittausarvot luetaan valituilta mittauskeskittimiltä Modbus/TCP-protokollan avulla vastaavasti kuin aurinkovoimalan inverttereiltä. Ilmeisesti käytettyjen mittarien ohjelmistojen rajoituksista johtuen pitkien rekisteriryhmien lukeminen yhdellä kyselyllä epäonnistuu lähes aina. Lyhyempien, 4-6 rekisterin mittaisten, ryhmien lukeminen onnistuu melko suurella varmuudella ja yksittäisten rekistereiden lukeminen epäonnistuu erittäin harvoin. Tästä rajoituksesta johtuen on halutut mittausarvot jouduttu jakamaan selvästi suurempaan määrään ryhmiä kuin aurinkovoimalan invertterien tapauksessa. Tästä myös johtuu se, että kutakin mittausarvoa luetaan vain kerran 30 sekunnissa, vaikka arvoja mittalaitteissa päivitetäänkin sekunnin välein. Konfiguraatiotiedostossa on oletusarvoisesti määritetty rekisteriryhmien maksimikooksi 8 rekisteriä. Adapteriohjelma on tehty vikasietoiseksi siten, että jos rekisteriryhmän lukeminen epäonnistuu ensimmäisellä kerralla, jaetaan ryhmä kahteen osaan ja luetaan molemmat osat erikseen. Edelleen jos luku vielä epäonnistuu, puolitetaan välin pituus edelleen. Lukuvälejä puolitettaessa otetaan huomioon se, että yhtä mittausarvoa koskevat rekisterit luetaan aina samalla lukuyrityksellä (että samaa mittausta koskevat eri rekisterien arvot eivät ehdi päivittyä lukuyritysten välillä).

Mittausarvojen aikaleima otetaan ohjelmaa ajavan koneen kellon mukaan heti rekisteriluvun jälkeen.

5.5 Huomioita

Käytössä olevia mittauskeskittimiä luetaan myös muiden toimesta ainakin kerran tunnissa energiankulutuksen laskutusta varten. Tämän takia mittauskeskittimiä ei saa pommittaa liian suurella lukukyselytiheydellä. Projektissa käytössä on ollut 100 millisekunnin vähimmäisodotusaika, joka vähintään odotetaan ennen seuraavan rekisteriryhmän lukuyritystä. Nykyisellä järjestelyllä kaikista valituista mittauksista on saatu mittausarvot lähes poikkeuksetta 30 sekunnin välein, mutta tuota lukuväliä ei kuitenkaan voi kovin paljoa pienentää. Esim. 20 sekunnin lukuväli olisi ehkä mahdollinen, mutta paljon sitä pienempi ei enää onnistu ainakaan kovin suurella varmuudella.

Mittasuurelistausta tehdessä ja testiarvoja luettaessa havaittiin, että ISS:n mittarikeskittimien rekisterinumeroiden ja vastaavien laitteiden dokumentaatioista saatujen rekisterinumerojen välillä oli mittasuureesta riippuen joko yhden tai kahden rekisterinumeron ero. Yhden suuruinen ero havaittiin kaikissa luetuissa kokonaislukuarvoissa ja kahden suuruinen ero IEEE-standardin mukaisessa muodossa olleista liukuluvuista. Kokonaislukujen kohdalla piti dokumentaatiosta löytyviä rekisterinumeroita vähentää yhdellä ja liukulukujen kohdalla vastaavasti vähentää kahdella, että saatiin luettuja oikeat mittaussuureet.

6 Kiinteistöautomaatio (Siemens)

Kampusareenan Siemensin Desigo-kiinteistöautomaatiojärjestelmän mittauksia luetaan BACnet-protokollan kautta. Tietoturvasyistä BACnet-adapterin koodia ei ole laitettu mukaan julkisesti julkaistuun koodirepositorioon.

6.1 Konfiguraatiotiedosto

- `bacnet_config.json`

Tässä tiedostossa määritellään kiinteistöautomaatiojärjestelmän lukemiseen tarvittavat parametrit. Alla parametrien selitykset:

- `my_ip:` oma ip-osoite, johon BACnet lähettää viestit
- `my_port:` oma porttinumero, johon BACnet lähettää viestit
- `bacnet_port:` porttinumero BACnet:iin
- `subscriber_id:` tilaustunnus BACnet:n tilauksiin (0-255)
- `subscriber_lifetime:` tilauksien pituus sekunteina
- `csv_file:` mittausarvojen tiedot sisältävän tiedoston nimi

- `BACnet_config.csv`

Tässä tiedostossa määritellään kiinteistöautomaation mittausarvojen lukemiseen tarvittavat osoiteparametrit sekä eri mittaussuureiden id-numerot ja käytetyt nimet, polut ja yksiköt IoT-Ticket-tallennusta varten. Luottamuksellinen mittausdata on merkitty *confidential*-sarakkeessa "`x`"-merkillä. Repositoriossa on valmiina pohja, jossa on esimerkki mittaussuurerivistä.

6.2 Adapteriohjelma

Adapteriohjelma on kirjoitettu Python-ohjelmointikielellä. Mittauksien muuntamiseen yhdenmukaiseen muotoon ja lähettämiseen käytetään kaikille adapteriohjelmille yhteisiä ohjelmakomponentteja. Adapteriohjelman käyttämät tiedostot:

- `adapters/bacnet_adapter.py`
- `adapters/bacnetDataModel.py`
- `adapters/common_utils.py`
- `adapters/bacnet_config.json`
- `adapters/BACnet_config.csv`

Ohjelma käynnistetään komennolla:

```
python3 bacnet_adapter.py bacnet_config.json
```

6.3 Mittaukset

Siemensin kiinteistöautomaatiosta saadaan laajasti erilaisia eri Kampusareenan tiloihin liittyviä muuttujia. Useimmista tiloista on saatavilla mm. sisäilman lämpötilaa, ilmanlaatua ja läsnäoloa kuvaavia suureita. Lisäksi luetaan mm. Kampusareenan ilmanvaihtoon, jäähdytykseen ja lämmitykseen liittyviä suureita. Yhteensä eri suureita on 2883 kappaletta. Suureista saadaan tieto, kun niiden arvo on muuttunut. Tämän takia jostakin suureista voidaan saada esim. 100 000 arvoa vuorokaudessa (useammin kuin kerran sekunnissa) ja toisesta suuresta vain 1 tai 2 arvoa vuorokaudessa. Osa luetuista mittauksista tulevat Kampusareenan vuokralaisten tiloista, joten niitä ei laiteta julkiseen jakeluun (kuten IoT-Ticketin tietokantaan), vaan ne pidetään pelkästään omassa tallennuksessa.

6.4 Datan lukeminen

Mittausarvot luetaan kiinteistöautomaatiojärjestelmältä BACnet-protokollan kautta. Kommunikaatio kiinteistöautomaatiojärjestelmän ja adapteriohjelman välillä tapahtuu UDP-viesteinä. Adapteriohjelman tilaa valitut mittausarvot kiinteistöautomaatiojärjestelmältä. Tilauksen jälkeen järjestelmä lähettää mittaussuureen arvon adapteriohjelmalle ja aina kun suureen arvo muuttuu, adapteriohjelmalle lähetetään uusi muuttunut arvo. Tilauksen jälkeen lähetettäviä arvoja ei välttämättä lähetetä välittömästi, vaan kaikkien mittausarvojen saamisen saattaa mennä useita (kymmeniä) minuutteja. Koska arvojen tilauksen maksimiaika on rajoitettu, tehdään tilaus aina uudelleen kahdesti vuorokaudessa.

7 Ilmatieteenlaitoksen sääennuste (HIRLAM)

Ilmatieteenlaitoksen avoin sääennustedata, joka on laskettu HIRLAM (High Resolution Limited Area Model) säämallilla.

http://ilmatieteenlaitos.fi/avoin-data-saaennustedata-hirlam

7.1 Konfiguraatiotiedostot

- `hirlam_configuration.json`

Tässä tiedostossa määritellään HIRLAM-sääennusteen lukemiseen tarvittavat sijaintitiedot, valitut ennustearvot sekä aikaan liittyvät parameterit. Lisäksi määritellään IoT-Ticket:ssä käytettävät nimet ja polut. Alla joidenkin parametrien selityksiä:

- `configuration:` konfiguraatioasetuksien tunniste luettaessa tiedostosta rest_api_configuration.json
- `verbose:` kuinka monen onnistuneen luvun jälkeen tulostetaan viesti?
- `latitude:` sääennusteen sijainnin leveysaste
- `longitude:` sääennusteen sijainnin pituusaste
- `start_hour:` mistä tunnista lähtien ennusteesta otetaan talteen arvoja (0 = simulaation alku)
- `end_hour:` kuinka pitkälle tunteina ennustearvoja haetaan
- `time_step_minute:` kahden ennustearvon aikaväli minuutteina
- `fields:` ennusteeseen mukaan otettavat suureet
- `time_interval_s:` normaali aikaväli sekunteina kahden ennustekyselyn välillä
- `time_interval_min_s:` ennustekyselyjen välinen aikaväli vähintään sekunteina
- `rtl_id_base:` pienin tallennus id-numero, jota käytetään ennustearvoille
- `iot_ticket_path_base:` IoT-Ticket:ssä käytetty polku ennusteille
- `iot_ticket_name_base:` IoT-Ticket:ssä käytetty nimi ennusteille
- `description:` ennustearvojen kuvaus tallennettavaa ennustelistatiedostoa varten

Kentissä iot_ticket_path_base, iot_ticket_name_base ja description voi käyttää merkintöjä `{base:}`, `{name:}` ja `{desc:}`, joiden arvot luetaan käytetystä REST API:sta ohjelman käynnistyessä.

- `rest_api_configuration.json`

Eri REST API -datalähteille yhteinen konfiguraatiotiedosto, jossa on määritelty kyselyille tarvittavat osoitteet ja parametrit sekä missä muodossa parametrit annetaan. Lisäksi on määritelty kyselyihin tarvittavat tunnuksien avaimet. HIRLAM-ennusteelle parametrit määritellään `"HIRLAM":` -kohdassa. Kyselyiden osoite annetaan kohdassa `"host":`, kyselyjen parametrien muoto kohdassa `"query_params":` ja luodun API-tunnuksen kyselyihiin tarvittava API-avain kohdassa `"authentication": { "api_key":`. Ellei API:n rajapinnassa tapahdu muutoksia tai jouduta ottamaan käyttöön toista API-tunnusta, tähän tiedostoon ei pitäisi joutua tekemään muutoksia.

7.2 Adapteriohjelma

Adapteriohjelma on kirjoitettu Python-ohjelmointikielellä. Ohjelman runkona käytetään kaikille REST-rajapinnan kautta haettaville datoille yhteistä geneeristä REST-adapter-ohjelmaa, johon on lisätty HIRLAM-kutsujen konfigurointi sekä vastauksena saatavan datan käsittely. Mittauksien muuntamiseen yhdenmukaiseen muotoon ja lähettämiseen käytetään kaikille adapteriohjelmille yhteisiä ohjelmakomponentteja. Adapteriohjelman käyttämät tiedostot:

- `adapters/rest_generic_adapter.py`
- `adapters/rest_utils.py`
- `adapters/common_utils.py`
- `adapters/hirlam_api_adapter.py`

- `adapters/rest_api_configuration.json`
- `adapters/hirlam_configuration.json`

Ohjelma käynnistetään komennolla:

```
python3 rest_generic_adapter.py hirlam_configuration.json
```

7.3 Mittaukset

Haettu sääennuste on piste-ennuste Kampusareenan alueelle. Ennustettavia suureita ovat mm. ilman lämpötilan, ilmanpaine, ilmankosteus, tuulen nopeus ja suunta, sademäärä, pilvisyys sekä säteilyvoimakkuus. Sääennusteet lasketaan neljä kertaa vuorokaudessa (UTC-ajassa ennusteiden laskeminen aloitetaan kellon aikoina 00:00, 06:00, 12:00 ja 18:00). Ennuste ulottuu 54 tunnin päähän alkuhetkestä ja tällä hetkellä ennusteesta poimitaan ennustearvoja 10 minuutin aikavälillä. Ennusteen laskemisen aloitushetki tallennetaan myös erikseen omaksi muuttujakseen.

7.4 Datan lukeminen

Sääennuste haetaan Internetin yli HTTP-yhteyden kautta ilmatieteenlaitoksen tarjoamasta REST API:sta. API:n kautta on aina saatavilla uusin loppuun asti laskettu ennuste. REST API:n käyttöä varten on luotu API-tunnukset, joihin liittyvä API-avain liitetään jokaiseen hakuun. Adapteriohjelma on asetettu siten, että se tutkii API-haun jälkeen, onko haettu ennuste uudempi kuin edellinen. Jos kyseessä on uusi ennuste, lähetetään se Procem RTL -ohjelmalle tallennettavaksi ja odotetaan kunnes seuraava ennuste voi olla saatavilla. Muussa tapauksessa tehdään uusi haku lyhyen ajan (15 minuutin) päästä.

7.5 Huomioita

Sääennusteen laskemiseen menee ilmatieteenlaitokselta kohtuullisesti aikaa. Uusi sääennuste on yleensä saatavilla noin 4 tuntia laskennan aloittamista.

API:n muutoksen vuoksi välillä 28.8. klo 12:00 - 3.9. klo 09:00 on ennusteen laskemisen aloitushetken sijaan haettu laskemisen valmistushetki. Kyseisen aikavälin jälkeen on adapteria muutettu hakemaan jälleen laskemisen aloitushetki. Varsinaisiin ennustearvoihin tai niiden hakemiseen API:n muutoksella ei ole ollut vaikutusta.

8 Nord Poolin data

Nord Poolin internetsivun kautta haetut sähkön tuntihinnat (Day-ahead prices).

https://www.nordpoolgroup.com/Market-data1/Dayahead/Area-Prices/ALL1/Hourly/?view=table

8.1 Konfiguraatiotiedostot

- `nordpool_configuration.json`

Tässä tiedostossa määritellään sähkön tuntihintojen lukemiseen tarvittavat parametrit Nord Pool:n palvelusta. Lisäksi määritellään IoT-Ticket:ssä käytettävät nimet ja polut. Alla joidenkin parametrien selityksiä:

- `configuration:` konfiguraatioasetuksien tunniste luettaessa tiedostosta rest_api_configuration.json
- `verbose:` kuinka monen onnistuneen luvun jälkeen tulostetaan viesti?
- `currency:` käytetty rahayksikkö
- `areas:` lista haetuista alueista
- `time_interval_s:` normaali aikaväli sekunteina kahden ennustekyselyn välillä
- `time_interval_min_s:` ennustekyselyjen välinen aikaväli vähintään sekunteina
- `rtl_id_base:` pienin tallennus id-numero, jota käytetään hinta-arvoille
- `iot_ticket_path_base:` IoT-Ticket:ssä käytetty polku hinnoille
- `iot_ticket_name_base:` IoT-Ticket:ssä käytetty nimi hinnoille
- `description:` hinta-arvojen kuvaus tallennettavaa hintalistatiedostoa varten

Kentissä iot_ticket_path_base, iot_ticket_name_base ja description voi käyttää merkintää `{area:}`, joka luetaan käytetystä REST API:sta sekä merkintää `{area_long:}`, joka luetaan parametritiedostosta rest_api_configuration.json.

- `rest_api_configuration.json`

Eri REST API -datalähteille yhteinen konfiguraatiotiedosto, jossa on määritelty kyselyille tarvittavat osoitteet ja parametrit sekä missä muodossa parametrit annetaan. Lisäksi on määritelty kyselyihin tarvittavat tunnuksien avaimet. Nord Pool:n kautta haettaville sähkön hintatietokyselyille parametrit määritellään `"NordPool":` -kohdassa. Kyselyiden osoite annetaan kohdassa `"host":` ja kyselyjen parametrien muoto kohdassa `"query_params":`. Ellei API:n rajapinnassa tapahdu muutoksia, tähän tiedostoon ei pitäisi joutua tekemään muutoksia.

8.2 Adapteriohjelma

Adapteriohjelma on kirjoitettu Python-ohjelmointikielellä. Ohjelman runkona käytetään kaikille REST-rajapinnan kautta haettaville datoille yhteistä geneeristä REST-adapter-ohjelmaa, johon on lisätty Nord Pool -kutsujen konfigurointi sekä vastauksena saatavan datan käsittely. Mittauksien muuntamiseen yhdenmukaiseen muotoon ja lähettämiseen käytetään kaikille adapteriohjelmille yhteisiä ohjelmakomponentteja. Adapteriohjelman käyttämät tiedostot:

- `adapters/rest_generic_adapter.py`
- `adapters/rest_utils.py`
- `adapters/common_utils.py`
- `adapters/nordpool_api_adapter.py`
- `adapters/rest_api_configuration.json`
- `adapters/nordpool_configuration.json`

Ohjelma käynnistetään komennolla:

```
python3 rest_generic_adapter.py nordpool_configuration.json
```

8.3 Mittaukset

Haettu data sisältää sähkön tuntikohtaiset spot-hinnat Suomessa. Seuraavan päivän hinnat ovat normaalisti saatavilla kello 13:00 mennessä (Suomen kesäaikaa).

8.4 Datan lukeminen

Adapteriohjelma hakee spot-hinnat Internetin yli HTTPS-yhteyden kautta samalla kyselyllä kuin Nord Poolin tarjoama internetsivu. Kyselyn tekemiseen ei tarvita erillisiä tunnuksia. Adapteriohjelma on asetettu siten, että käynnistyessään se hakee aloituspäivän hinnat sekä seuraavan päivän hinnat, jos ne ovat jo saatavilla. Onnistuneen hintahaun jälkeen ohjelma odottaa, kunnes on kulunut vuorokausi edellisen hintojen päivityksen ajankohdasta, jonka jälkeen tehdään uusi hintahaku. Jos uudet hinnat eivät vielä hakuhetkellä ole saatavilla odotetaan 10 minuuttia ja yritetään uudestaan.

9 Fingridin data

Fingridin tarjoama avoin data sähkömarkkinoista ja voimajärjestelmästä:

https://data.fingrid.fi/dataset

9.1 Konfiguraatiotiedostot

- `fingrid_configuration.json`

Tässä tiedostossa määritellään kantaverkon tilan lukemiseen tarvittavat parametrit Fingridin palvelusta. Haluttujen arvojen lista tarvittavine lisätietoineen löytyy csv-muotoisesta parametritiedostosta. Alla joidenkin parametrien selityksiä:

- `configuration:` konfiguraatioasetuksien tunniste luettaessa tiedostosta rest_api_configuration.json
- `verbose:` kuinka monen onnistuneen luvun jälkeen tulostetaan viesti?
- `csv_filename:` käytetty halutut arvot listaavan konfiguraatiotiedoston nimi
- `min_waiting_time:` vähimmäisodotusaika arvokyselykierroksien välillä (sekunteina)

iot_ticket_path_base, iot_ticket_name_base ja description voi käyttää merkintää `{area:}`, joka luetaan käytetystä REST API:sta sekä merkintää `{area_long:}`, joka luetaan parametritiedostosta rest_api_configuration.json.

- `Fingrid_IDs.csv`

Tässä tiedostossa määritellään eri suureiden id-numerot ja käytetyt nimet, polut ja yksiköt IoT-Ticket-tallennusta varten. Sarakkeen *variable_id* avulla suure yhdistetään Fingridin API-palvelun tarjoamaan suureeseen. Fingridin palvelussa kutakin suuretta vastaavat *variable_id*:t löytyvät kyseisen suureen oman sivun API-ohjelmointirajapinnan "Lisää tietoa"-sivulta. Suurekohtaisen kahden peräkkäisen kyselyn aikavälin voi asettaa sekunteina "*query_interval (s)*"-sarakkeessa. Ennustesuureille tulee merkitä "`x`" *is_prediction*-sarakkeeseen ja ennusteen pituus sekunteina "*prediction_length (s)*"-sarakkeeseen. "*store_interval (s)*"-sarakkeeseen voi merkitä sekunteina aikavälin, joka vähintään vaaditaan ennen kuin seuraava arvolle luetaan. Esim. arvo 3600 sarakkeessa "*store_interval (s)*" tarkoittaa, että suureesta luetaan arvoja tunnin välein. Jos kyseinen sarake on tyhjä, luetaan kaikki saatavat arvot.

- `rest_api_configuration.json`

Eri REST API -datalähteille yhteinen konfiguraatiotiedosto, jossa on määritelty kyselyille tarvittavat osoitteet ja parametrit sekä missä muodossa parametrit annetaan. Lisäksi on määritelty kyselyihin tarvittavat tunnuksien avaimet. Fingridin kautta haettaville tietokyselyille parametrit määritellään `"Fingrid":` -kohdassa. Kyselyiden osoite annetaan kohdassa `"host":`, kyselyjen parametrien muoto kohdassa `"query_params":` ja luodun API-tunnuksen kyselyihiin tarvittava API-avain kohdassa `"authentication": { "params": { "value":`. Ellei API:n rajapinnassa tapahdu muutoksia tai jouduta ottamaan käyttöön toista API-tunnusta, tähän tiedostoon ei pitäisi joutua tekemään muutoksia.

9.2 Adapteriohjelma

Adapteriohjelma on kirjoitettu Python-ohjelmointikielellä. Ohjelman runkona käytetään kaikille REST-rajapinnan kautta haettaville datoille yhteistä geneeristä REST-adapter-ohjelmaa, johon on lisätty Fingrid-kutsujen konfigurointi sekä vastauksena saatavan datan käsittely. Mittauksien muuntamiseen yhdenmukaiseen muotoon ja lähettämiseen käytetään kaikille adapteriohjelmille yhteisiä ohjelmakomponentteja. Adapteriohjelman käyttämät tiedostot:

- `adapters/rest_generic_adapter.py`
- `adapters/rest_utils.py`
- `adapters/common_utils.py`

- `adapters/fingrid_api_adapter.py`
- `adapters/rest_api_configuration.json`
- `adapters/fingrid_configuration.json`
- `adapters/Fingrid_IDs.csv`

Ohjelma käynnistetään komennolla:

```
python3 rest_generic_adapter.py fingrid_configuration.json
```

9.3 Mittaukset

Fingridin tarjoamasta avoimesta datasta haetaan reaaliaikatietoja (3 minuutin välein päivittyvät arvot) Suomen kantaverkon taajuudesta, eri tuotantomuotojen (ydinvoima, tuulivoima, jne.) tuotannosta sekä sähkön kokonaiskulutuksesta ja -tuotannosta. Tarjolla olevaa reaaliaikatietoa sähkönsiirtotiedoista Suomen ja eri naapurimaiden välillä tai tehoreservin tuotannosta ei nyt ole laitettu hakuun. Lisäksi avoimesta datasta haetaan tuuli- ja aurinkovoiman tuotantoennusteet sekä kokonaiskulutusennuste. Ennusteet päivittyvät tunnin välein ja tuotantoennusteet ulottuvat 36 tunnin päähän ennustehetkestä ja kokonaiskulutusennuste seuraavan vuorokauden loppuun asti.

9.4 Datan lukeminen

Data haetaan Internetin yli HTTPS-yhteyden kautta Fingridin tarjoamasta REST API:sta. REST API:n käyttöä varten on luotu API-tunnukset, joihin liittyvä API-avain liitetään jokaiseen hakuun. Jokaiselle haettavalle suureelle on määritelty aikaväli, kuinka usein uusia tietoja haetaan ja adapteriohjelma pitää kirjaa kullekin suureella viimeisestä ajanhetkestä, jolloin on onnistuneesti haettu uusia arvoja. Onnistuneen haun jälkeen kyseisen suureen kohdalla odotetaan aina vähintään suureelle määritellyn aikavälin verran ennen seuraavaa hakua. Epäonnistuneen haun (ei saatu yhtään uutta arvoa) jälkeen hakujen välinen aikaväli voi olla pienempi (kuitenkin vähintään konfiguraatiotiedostossa määritellyn suuruinen). Ennustesuureilla haetaan arvoja aina alkaen hakuhetkestä (tai tarkemmin edellisestä alkaneesta tunnista alkaen). Muista suureista haetaan aina vain uusia arvoja.

9.5 Huomioita

Käytetyllä ilmaisella API-tunnuksella voidaan tehdä korkeintaan 10 000 hakua vuorokaudessa. Nykyisillä asetuksilla tämä raja ei tule vastaan, mutta kannattaa ottaa huomioon, jos luettavien suureiden määrää lisätään. Yhdellä haulla voi hakea vain yhtä suuretta kerrallaan, mutta aikavälin kuhunkin hakuun voi valita vapaasti.

10 Adapteriohjelmien ja ProCem RTL -ohjelman välinen kommunikointi

Datanvälitys adapteriohjelmilta ja datan tallennuksesta huolehtivalle ProCem RTL -ohjelmalle on toteutettu kaikissa adapteriohjelmissa UDP-protokollan mukaisten viestin avulla. Kaikki adapteriohjelmat käyttävät nyt yhteistä `common_utils.py` tiedostossa olevaa datanlähetysfunktiota. Kaikki adapteriohjelmat myös muuttavat samassa tiedostossa olevien yhteisen funktion avulla lähetettävät mittausarvot ennen lähetystä IoT-Ticket-yhteensopivaan muotoon. Mukana on siis aina mittausarvon ja aikaleiman lisäksi mittauksen nimi, polku, yksikkö, tietotyyppi, id-numero sekä tieto luottamuksellisuudesta.

10.1 Kommunikoinnin puskurointi ja toteutus

Jokainen adapteriohjelma luo oman jonotietorakenteen (käytetään Pythonin säieturvallista jonoa `queue`), johon kunkin adapterin keräämä data laitetaan. Jonojen lukemista ja datan edelleen lähettämistä varten jokaiselle jonolle käynnistetään omaan ohjelmasäikeeseen `procemSendWorker`-käsittelijä. Käsittelijä puskuroi jonoon laitetun datan paketeiksi, joiden maksimikoko on rajoitettu 8000 tavuun (normaaliasetuksilla UDP-pakettien maksimipituus on 8192 tavua, mutta käytännössä kannattaa käyttää hieman pienempää maksimirajaa). Kun datapaketti on muodostettu, lähetetään se ProCem RTL -ohjelmalle UDP-viestinä.

Käsittelijään on asetettu minimiaikaväli (nyt käytössä on 10 ms), joka aina vähintään odotetaan ennen kuin sama käsittelijä lähettää seuraavan paketin. Kun käytössä oli pienempi minimiaikaväli, havaittiin että huomattavasti suurempi määrä viestien lähetyksiä epäonnistui. Tämä kannattaa ottaa huomioon, etenkin jos datamäärää lisätään huomattavasti. Tällöin adapteriohjelmien ja ProCem RTL -ohjelman välistä kommunikointia on optimoitava jollain tavalla.

Varsinainen UDP-viestin lähetys tehdään avaamalla uusi lähetyssäie, jossa datapaketti lähetetään ProCem RTL -komponentin UDP-palvelimelle (teknisellä tasolla lähetyksessä käytetään Pythonin socket-luokkaa). Datapaketin lähetyksen jälkeen lähetyssäie suljetaan. Oma lähetyssäie kunkin datapaketin lähetykselle on tehty sen takia, että mahdollisten häiriötilanteiden aikaan (esim. UDP-palvelin ei väliaikaisesti pysty vastaanottamaan viestejä) käytetty datajono ei pääse kasvamaan liian suureksi ohjelman odottaessa edellisen viestin läpimenoa.

10.2 Kommunikoinnin onnistumisen varmistaminen

Koska kommunikoinnissa käytetty UDP-protokolla ei mitenkään takaa, että vastaanottaja todella saa lähetetyn viestin, on adapterien ja ProCem RTL -komponentin välisen kommunikointiin lisätty varmistusviestin lähettäminen merkiksi vastaanotetusta viestistä. ProCem RTL -komponentti lähettää aina datapaketin saatuaan datapaketin lähettäjälle varmistusviestin (nyt käytössä on pelkkä teksti "OK"). Datapaketin lähetyssäie tarkistaa aina paketin lähetyksen jälkeen, että se saa vastauksena mainitun varmistusviestin. Jos varmistusviestiä ei saada (odotusajaksi on nyt asetettu 0,5 sekuntia), yritetään datapaketin lähetystä uudelleen.

Jos datapaketin lähetykselle ei ole 5:n yrityskerran jälkeenkään saatu varmistusviestiä vastauksena, tallennetaan datapaketin sisältö tiedostoon ja suljetaan lähetyssäie. Käynnissä on erillinen ohjelma joka tasaisin väliajoin (tällä hetkellä 10 minuutin välein) käy lukemassa kyseisen tiedoston ja yrittää lähettää löytämänsä datan ProCem RTL -komponentille. Tässä lähetyksessä käytetään samaa varmistusviestijärjestelmää ja tiedostovarmistusta kuin adapterien käynnistämässä lähetyksessä.

Varmistusviestien ja tiedostovarmistuksen ansiosta dataa ei pääse häviämään enää sen jälkeen, kun ne on adapterien toimesta luettu, vaan käytännössä kaikki luettu data saadaan välitettyä edelleen ProCem RTL -komponentille ja siten myös mukaan lopulliseen datan tallennukseen. Tällä hetkellä luettava datamäärä ei ole niin suuri, että pelkästään sen lähetyksestä seuraisi normaalitilanteessa kovinkaan paljon häiriöitä datan välitykseen. Tiedostovarmistusta on nykytilanteessa käytetty suurimmaksi osaksi vain tilanteissa, joissa ProCem RTL -komponentti ei ole ollut päällä (esim. uudelleenkäynnistämisen vuoksi) tai joissa ohjelmia ajavalla Linux-koneella

on ollut jotain muuta raskasta ajoa samaan aikaan. Erityisesti silloin kun edellisen päivän datatiedostoa pakataan pidempiaikaista säilöntää varten (aloitetaan kello 01:00 ja kestää 2-4 tuntia) ja samaan aikaan lähetään reaaliaikaista dataa IoT-Ticket:iin, tiedostovarmistusta joudutaan jonkin verran käyttämään (kuitenkin vain hyvin pieni osa datasta päätyy väliaikaisesti tiedostovarmistukseen). Aikaisemmin kun datan lähetys IoT-Ticket:iin ei ollut päällä, datatiedoston pakkaus ei yksinään aiheuttanut riittävästi kuormaa, minkä takia tiedostovarmistus saattoi olla täysin käyttämättä usean vuorokauden ajan.

Kannattaa huomata, että varmistusviestit varmistavat vain sen, että jotain on vastaanotettu. Ne eivät kuitenkaan kerro onko vastaanottaja saanut juuri täsmälleen saman viestin mitä on lähetetty. Tämän ei pitäisi olla ongelma ProCem:n datan lähetyksessä. Varmistusviestien käytöllä on kuitenkin se seuraus, että joskus varmistusviesti itse saattaa hukkua matkalla, jolloin sama data lähetetään uudelleen ja päätyy siten ProCem RTL -komponentille kahteen kertaan. Käytännössä tämä näkyy siten, että päivittäisestä ProCem:n oman tallennuksen datatiedostosta saattaa löytyy identtisiä rivejä. IoT-Ticket:ssä nämä kahteen kertaan lähetetyt arvot ei näy, koska jos IoT-Ticket:iin lähettää samalle suureelle samalla aikaleimalla kaksi arvoa viimeisenä lähetty arvo kirjoitetaan aikaisemman päälle.

Kaaviokuva datan kulkemisesta adapterilta ProCem RTL -komponentille: