ProCem RTL –ohjelma

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 Toimintaperiaate

ProCem RTL -ohjelma toimii keskitettynä datankeräysohjelmana. Se vastaanottaa mittausdatan adapteriohjelmilta ja ohjaa sen edelleen sekä projektin sisäiseen datan tallennukseen että IoT-Ticket:n tietokantaan. Lisäksi ohjelmalta voi kysyä eri mittaussuureiden viimeisimpiä arvoja ja ohjelman voi myös asettaa välittämään tietyt mittaussuureet eteenpäin ulkopuoliselle ohjelmalle. Ohjelman ajoon tarvittavat asetustiedot luetaan ohjelman käynnistyksen yhteydessä, mutta joidenkin toimintojen vaihtaminen (kuten IoT-Ticket-lähetyksien päälle tai pois päältä asettaminen) on mahdollista ilman uudelleenkäynnistystä.

Ohjelma käynnistää useita eri ohjelmasäikeitä:

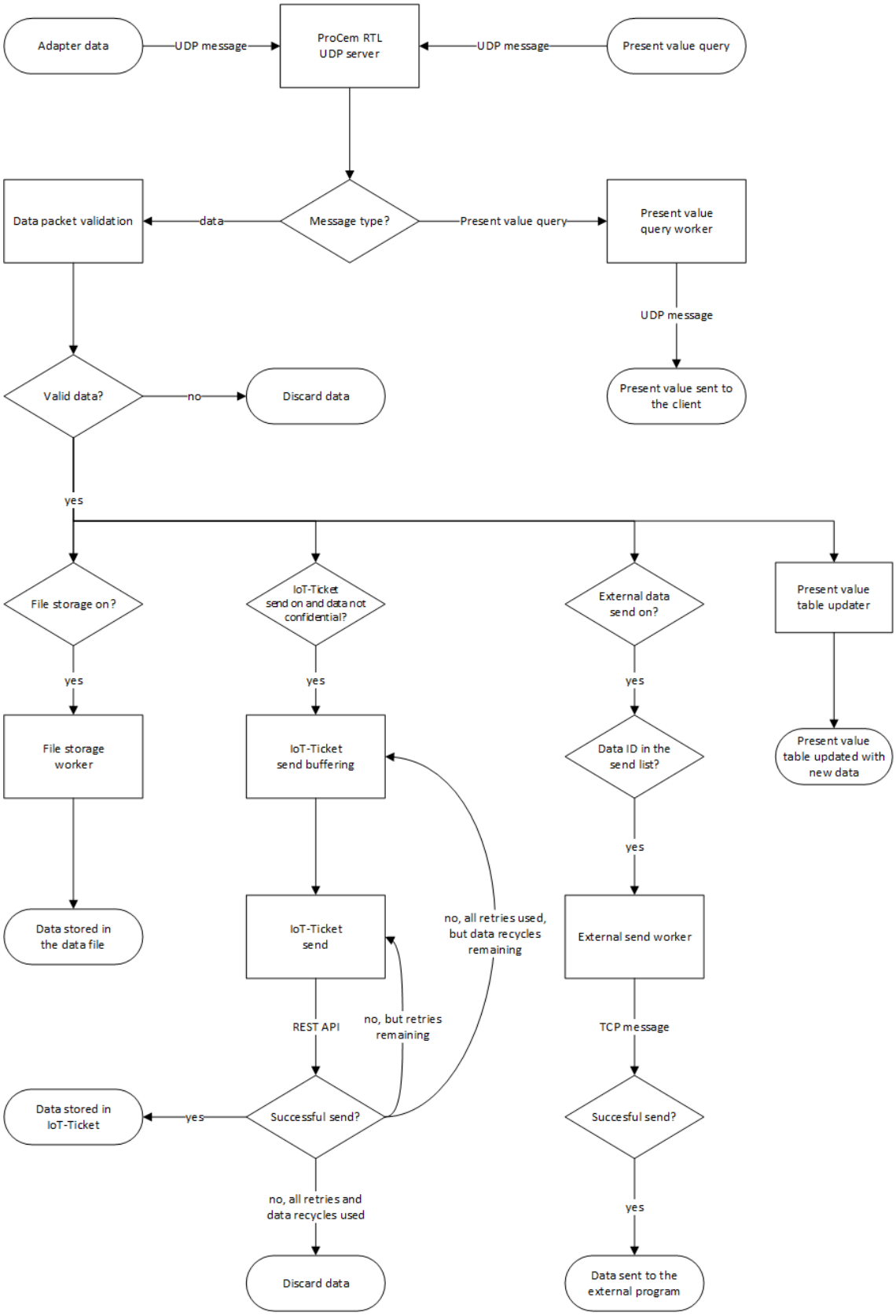
- Ohjelman käynnistyessä luotu pääsäie jää odottamaan interaktiivisia ohjauskomentoja
- Vastaanotetun datan validointia ja edelleen ohjausta varten käynnistetään oma säie.
- Omaa tiedostotallennusta varten käynnistetään oma säie.
- IoT-Ticket-lähetyksien puskurointia varteen käynnistetään oma säie.
- Jokainen IoT-Ticket-lähetys tehdään omassa säikeessä. Näiden yhtäaikaisten maksimimäärän voi asettaa parametritiedostossa
- Viimeisimpien arvojen kyselyiden käsittelemistä varten käynnistetään oma säie.
- Mahdollisen datanvälityksen ulkopuoliselle ohjelmalle hoitaa erillinen säie.
- Jokaisen adapteriohjelmalta vastaanotetun datapaketin vastaanotto tapahtuu omassa säikeessään.

Eri säikeiden välinen kommunikointi on toteutettu säieturvallisten jonojen avulla.

Kuvassa 1 on esitetty kaavioina datan kulku ohjelmassa.

Tässä dokumentissa on ProCem RTL -ohjelman toiminnan lisäksi dokumentoitu seuraavat erilliset datankeräysohjelmaan läheisesti liittyvät ohjelmakomponentit:

- datan pakkaus- ja varmuuskopiointiohjelma (luku 4)
- ohjelma tiedostoon tallennetun historidatan lähettämiseksi IoT-Ticket:iin (luku 9)



Kuva 1. Kaaviokuva datan kulkemisesta ohjelmassa.

2 Ohjelmakoodi ja ohjelman käyttö

Ohjelma on kirjoitettu Python-ohjelmointikielellä. Lista ohjelman käyttämistä tiedostoista:

- `procem_rtl.py`
- `iotticket_utils.py`
- `tcp_utils.py`
- `adapters/common_utils.py`
- `utils/datastorage.py`
- `rtl_configuration.json`

Ohjelma käynnistetään komennolla:

```
python3 procem_rtl.py rtl_configuration.json
```

Tiedostossa `rtl_configuration.json` asetetaan ohjelman tarvitsemat parametrit. Parametrien listaus selityksineen:

- `username:` IoT-Ticket:n REST API-tunnuksen käyttäjänimi
- `password:` IoT-Ticket:n REST API-tunnuksen salasana
- `baseurl:` IoT-Ticket:n REST API:n osoite
- `deviceid` IoT-Ticket-laitteen, johon kaikki data lähetetään, id-tunnus
- `iotticket-buffer-size:` IoT-Ticket-lähetyspuskurin minimiarvo
- `iotticket-max-packet-size:` mittauksien maksimimäärä yhdessä IoT-Ticket-lähetyksessä
- `iotticket-minimum-delay-s:` minimiaika kahden peräkkäisen IoT-Ticket-lähetyksen välillä
- `iotticket-maximum-retries:` datalähetyksien uudelleenyrityksien maksimimäärä
- `iotticket-max-data-cycles:` kuinka monta kertaa epäonnistuneiden lähetyksien dataa kierrätetään maksimissaan takaisin lähetysjonoon
- `procem-iotticket-workers:` yhtäaikaisten IoT-Ticket-lähetyksien maksimimäärä
- `db-queue-size:` datatiedostotallennusjonon maksimikoko
- `iotticket-queue-size:` IoT-Ticket-lähetyksiin käytettävän jonon maksimikoko
- `db_type:` datatiedostotallennuksen tyyppi (csv/json)
- `ids_for_battery:` id-lista ulkopuoliselle ohjelmalle välitettävistä mittauksista
- `db_storage_on:` datatiedostotallennus päällä (true/false)
- `iotticket_send_on:` IoT-Ticket-lähetys päällä (true/false)
- `battery_demo_on:` datanvälitys ulkopuoliselle ohjelmalle päällä (true/false)
- `present_value_count:` viimeisimpien arvojen lukumäärä mittausta kohti

Vain 4 ensimmäistä parametria (käyttäjänimi, salasana sekä IoT-Ticket:n osoite ja laite) ovat pakollisia. Muiden parametrien kohdalla parametriarvon puuttuessa käytetään ohjelmatiedostossa määriteltyjä oletusarvoja.

3 Datan vastaanottaminen adapteriohjelmilta

Ohjelman käynnistyksen yhteydessä käynnistetään myös UDP-palvelin, johon adapteriohjelmat voivat lähettää keräämänsä datan. Projektin toteutuksessa sekä adapteriohjelmia että Procem RTL –ohjelmaa ajetaan samalla palvelimella, mutta tämän muuttaminen ei vaadi suurta työtä. Palvelimen ip-osoite (127.0.0.1, joka tarkoittaa ohjelmaa ajavaa konetta) ja porttinumero (6666) määritellään tiedostossa `common_utils.py`. Porttinumeron voi vaihtaa kooditiedostoa muokkaamalla, mutta kannattaa huomata, että muutos näkyy vasta uudelleenkäynnistyksen yhteydessä eikä vaikuta ajossa oleviin ohjelmiin. Adapteriohjelmien ja ProCem RTL -ohjelman välisestä kommunikoinnista on kerrottu tarkemmin erillisessä dokumentissa (Procem adapteriohjelmat, luku 10).

Saadut datapaketit lisätään datajonoon, josta erillisessä validointisäikeessä toimiva funktio käy lukemassa datan. Luettu data validoidaan tarkistamalla, että siinä on mukana kaikki IoT-Ticket:n tarvitsemat tiedot ja että ne ovat IoT-Ticket:n REST API:n hyväksymässä muodossa. Validoinnissa ei tarkisteta mittausarvojen järkevyyttä, vaan pelkästään oikeamuotoisuutta.

Validoinnin jälkeen data ohjataan oman tiedostotallennuksen hoitavalle ohjelmasäikeelle, IoT-Ticket:iin lähetyksen hoitavalle säikeelle sekä datanlähetyksen ulkopuoliselle ohjelmalle hoitavalle säikeelle. Datanvälityksen voi estää kullekin säikeelle erikseen käyttämällä parametritiedostoissa parametreille `db_storage_on`, `iotticket_send_on` ja `battery_demo_on` arvoa `false`. Lisäksi ulkopuoliselle ohjelmalle välitetään vain data, jonka id-numero on `ids_for_battery` listassa.

Lisäksi jokaisen validoidun data-arvon vastaanottamisen yhteydessä päivitetään viimeisimmän arvon taulukkoa. Taulukossa pidetään yllä kunkin mittaussuureen aikaleimaltaan viimeisimpien mittauksien arvoja. Oletusarvoisesti muistissa pidetään vain viimeisimmän mittauksen arvoa (`present_value_count` on 1).

4 Datan tallennus omaa säilytystä varten

Adapteriohjelmilta saatu data tallennetaan myöhempää käyttöä varten tekstitiedostoihin, kunkin vuorokauden data omiin tiedostoihinsa. Tallennusmuodon voi valita parametrilla `db_type`. Projektissa aktiivikäytössä on kompaktimpi csv-muoto, jossa kustakin mittauksesta tallennetaan id-numero, mittausarvo ja mittauksen aikaleima. Datan tallennuksesta ja sen käytöstä on kerrottu tarkemmin sivulla erillisessä dokumentissa (Kampusareenan datan lukeminen). Json-muotoisessa tallennuksessa jokaisesta mittauksesta tallennetaan tiedostoon myös muut IoT-Ticket-lähetyksiin tarvittavat tiedot kuten mittauksien nimet sekä polut, joten json-muoto on tarkoitettu lähinnä ohjelmiston toiminnan testaukseen. Datan tiedostotallennus on mahdollista estää kokonaan käyttämällä parametrin `db_store` arvoa `false`.

Kunkin vuorokauden datan sisältävät tekstitiedostot pakataan pienempään tilaan ja kopioidaan projektin ryhmähakemistoon säilytystä varten aina aamuyön aikana. Datan pakkaus ja kopiointi tehdään erillisellä backup-ohjelmalla. Backup-ohjelman käyttämät tiedostot:

- `backup_procem_data.py`
- `adapters/common_utils.py`
- `backup_procem_data.json`

Ohjelma pitää kopion edellisen päivän datasta ajohakemistossa ja kopioit edellisen viikon datasta erillisessä hakemistossa. Viikkoa vanhemmat datat löytyvät vain projektin ryhmähakemistosta.

Backup-ohjelma käynnistetään komennolla:

```
python3 backup_procem_data.py backup_procem_data.json
```

Erillisessä parametritiedostossa `backup_procem_data.json` annettavien parametrien kuvaukset:

- `date_format`: päivämäärien esitysmuoto tiedostonimissä
- `data_filename_end`: datatiedostojen nimien loppuosa
- `counter_filename_end`: datalaskuritiedostojen nimien loppuosa
- `counter_delimiter`: käytetty erotinmerkki id:n ja määrien välillä datalaskuritiedostoissa
- `local_file_permissions`: tiedosto-oikeudet ajokoneella pidettävässä datassa
- `remote_file_permissions`: tiedosto-oikeudet projektin ryhmähakemistossa
- `backup_hour`: pakkauksen aloitustunti (1 = pakkaus aloitetaan 01:00)
- `backup_days_cwd`: ajohakemistossa pidettävän datan määrä päivissä
- `backup_days_backup_dir`: ajokoneella erillisessä hakemistossa pidettävän datan määrä päivissä
- `compression_command`: käytettävä pakkauskomento
- `compressed_file_extension`: pakattujen tiedostojen tiedostopääte
- `compression_success_message`: onnistuneen pakkauksen jälkeen saatava viesti pakkausohjelmalta
- `local_data_directory`: hakemisto ajokoneella pidettävälle datalle
- `local_counter_directory`: hakemisto ajokoneella pidettäville datalaskuritiedostoille
- `remote_backup_server`: ryhmähakemiston tunnus ja palvelin
- `remote_data_directory`: ryhmähakemiston hakemisto datalle
- `remote_counter_directory`: ryhmähakemiston hakemisto datalaskuritiedostoille

5 Datan lähetys IoT-Ticket:iin

Adapteriohjelmilta saatu mittausdata lähetetään IoT-Ticket:n tietokantaan käyttämällä IoT-Ticket:n REST API:a. Vain dataa, jota ei ole merkitty luottamukselliseksi, lähetetään IoT-Ticket:iin. Koska yksittäisten mittauksien lähettäminen REST API:lla on liian hidasta, puskuroidaan dataa suuremmiksi paketeiksi. Datapakettien koko määräytyy dynaamisesti lähettävän datamäärän ja käytettyjen parametrien perusteella. Parametrilla `procem-iotticket-workers` määritetään kuinka monta yhtäaikaista REST API -lähetystä voi korkeintaan olla käynnissä yhtä aikaa. Lisäksi parametrilla `iotticket-minimum-delay-s` määritetään kuinka monta sekuntia vähintään pitää kulua aikaa aloitetun lähetyksen jälkeen ennen kuin seuraava lähetys voidaan aloittaa. Datan lähetys IoT-Ticket:iin on mahdollista estää kokonaan käyttämällä parametrin `iotticket_send_on` arvoa `false`.

Teknisellä tasolla jokainen IoT-Ticket-lähetys ajetaan erillisinä ohjelman osina omissa lähetyssäikeissään. 22.8.2018 käytetyllä IoT-Ticket:iin lähetettävällä datamäärällä (ja parametreilla `procem-iotticket-workers` = 5 ja `iotticket-minimum-delay-s` = 10) datapakettien koko asettuu normaalitilanteessa noin reiluun 20 000:een. Datapakettien lähetyksiä on käynnissä 4 tai 5 yhtä aikaa ja yhden tällaisen datapaketin lähettämiseen menee nykyisellä systeemillä noin 60-70 sekuntia. Datan lähetystä on testattu kokeilemalla parametrille `procem-iotticket-workers` eri arvoja ja testien perusteella on todettu, että optimiarvo nykyisellä systeemillä on 4 tai 5. Pienemmällä arvolla yksittäiset lähetykset toimivat hieman tehokkaammin, mutta datapakettien kokoa joudutaan kasvattamaan suuremmaksi, jolloin koko datapaketin lähetykseen menee enemmän aikaa. Käytettäessä suurempaa määrää yhtäaikaisia lähetyksiä taas lähetykset alkavat häiritä toisiaan sen verran paljon, että yksittäisten lähetyksien tehokkuus laskee selvästi. Parametrin `iotticket-minimum-delay-s` ensisijainen tarkoitus on estää monen yhtäaikaisen lähetyksen käynnistyminen samanaikaisesti ProCem RTL -ohjelman käynnistyksen yhteydessä. Jos IoT-Ticket:iin lähetettävän datan määrää pienennetään kannattaa pienentää samalla myös viiveparametrin arvoa.

Koska datan lähetys REST API:n kautta IoT-Ticket:iin on melko hidasta, on lähetyksen nopeuttamiseksi kokeiltu yhtäaikaisten lähetysten määrän ja datapakettien kokojen säätämisen lisäksi kokeiltu myös muita mieleen tulleita mahdollisia optimointikeinoja. Yksi toimiva keino myös löydettiin ja on käytössä ohjelmassa. Järjestämällä lähetettävän suuren (n. 20 000 mittausta) datapaketin mittauksien nimien perusteella ja jakamalla järjestetyn datapaketin pienempiin osiin (500 mittausta per osa) ja lähettämällä kunkin osan yksi kerrallaan havaittiin, että koko suuren datapaketin lähettämiseen järjestettyinä osina menee vain noin puolet siitä ajasta mikä kuluu, jos koko datapaketti lähetetään kerralla.

Alla taulukko yhden testidatapaketin (koko n. 10800 mittausta) lähettämiseen kuluneesta ajasta sekä järjestämisen ja osiin jakamisen (osien koko 500 mittausta) vaikutuksesta. Pelkällä järjestämisellä ei ole mitään vaikutusta kokonaislähetysaikaan, vaan nopeutus saadaan vain yhdistämällä järjestäminen osiin jakoon. Osiin jakaminen ilman järjestämistä taas hidastaa datan lähettämistä huomattavasti.

datapaketin lähetysaika	lähetys yhdessä paketissa	lähetys useassa paketissa
data järjestetty ennen lähetystä	85 s	45 s
dataa ei järjestetty ennen lähetystä	85 s	495 s

Ongelmatilanteista toipumista varten IoT-Ticket-lähetyksiin on tehty uudelleenlähetysominaisuus, joka yrittää niiden datapakettien lähettämistä uudelleen, joiden lähettäminen epäonnistui ensimmäisellä kerralla. Uudelleenlähetykset tehdään osakohtaisesti eli jos viiteen osaan jaetun suuren datapaketin ensimmäisen osan lähettäminen epäonnistuu ja muiden osien lähettäminen onnistuu, yritetään toisella lähetyskierroksella lähettää uudelleen vain ensimmäistä osaa. Lähetyskierroksien maksimimäärän voi asettaa parametrissa `iotticket-maximum-retries`.

Jos kaikkien lähetysyrityskertojen jälkeenkään ei kaikkea dataa olla saatu lähetettyä, kierrätetään lähetyksissä epäonnistunut osa mittauksista takaisin IoT-Ticket:n lähetysjonoon odottamaan uusia lähetysyrityksiä. Dataa, jota ei siis saatu lähetettyä useammallakaan yrityksellä, päätyy

myöhemmin takaisin lähetykseen ja sitä yritetään lähettää uudelleen. Jokaisen mittauksen yhteydessä ylläpidetään laskuria siitä, kuinka monta kertaa dataa on kierrätetty takaisin lähetysjonoon. Maksimikierrätysmäärän voi asettaa parametrissa `iotticket-max-data-cycles`. Laskurin avulla data, jota ei pysty ollenkaan lähettämään IoT-Ticket:iin, lopulta poistuu kierrosta. Aikaisemmin tehtävän datan validoinnin takia tilanteet, joissa yritetään lähettää dataa muodossa, jota IoT-Ticket ei hyväksy, eivät pitäisi olla mahdollisia, mutta poikkeustilanteet ovat kuitenkin mahdollisia. Alun perin datan kierrätysominaisuus on tehty vaihtoehtoiseksi suoran uudelleenlähetyksen kanssa, mutta molempia on mahdollista käyttää yhtä aikaa. Liian suuria arvoja uudelleenlähetyksien määrään ei kannata laittaa, koska tällöin IoT-Ticket-lähetyksien toimiessa hitaasti saattaa lähetyspuskuri odottaessaan vapautuvaa lähetyssäiettä kasvaa liian suureksi (mahdollisesti varaten kaiken käytettävissä olevan muistin ja kaataen ohjelman).

Varmistukset voi halutessaan ottaa kokonaan pois käytöstä käyttämällä parametrien arvoja `iotticket-maximum-retries` = 1 ja `iotticket-max-data-cycles` = 0.

5.1 Huomio datan lähetyksestä IoT-Ticket:n päivityksen jälkeen

Kun projektissa käytetty IoT-Ticket päivitettiin marraskuun 2018 alussa, havaittiin myös, että REST API:n varsinkin datan lähetykseen suhteen toimii huomattavasti nopeammin kuin aikaisemmin. Aikaisempaan verrattuna datan lähetysnopeus REST API:n kautta on karkeasti arvioituna noin kymmenen kertaa nopeampaa kuin aikaisemmin.

6 Viimeisimmän arvon kysely

Ohjelma pitää yllä taulukkoa kunkin mittauksen viimeisimmästä arvoista. Viimeisin arvo tarkoittaa tässä yhteydessä mittausarvoa, jonka aikaleima on suurin. Oletuksena pidetään yllä vain yhtä viimeisintä arvoa, mutta parametrilla `present_value_count` voi säätää kuinka monta arvoa kustakin mittauksesta pidetään muistissa.

Viimeisimpiä arvoja voi pyytää ohjelmalta UDP-viestillä. Ohjelma lähettää viimeisimmän arvon paluuviestinä pyynnön lähettäjälle. Tällä hetkellä ohjelman ylläpitämä UDP-palvelin toimii ohjelmaa ajavan koneen portissa 6666. Jos porttinumeroa pitää vaihtaa, pitää editoida tiedostoa `adapters/common_utils.py`, jossa porttinumero on asetettu vakion arvona, ja käynnistää kaikki ohjelmat uudelleen. Viimeisimmän arvon pyyntöviesti pitää olla UTF-8-muodossa ja sen sisältö pitää olla muotoa:

```
get_value:<id_numero>
```

Esim. jos halutun mittauksen id-numero on 5001, niin kyseisen mittauksen viimeisimmän arvon voi kysyä viestillä "`get_value:5001`".

Vastausviesti on myös UTF-8-muodossa ja sen sisältö on

```
<id_numero>;<mittausarvo>;<aikaleima>
```

missä aikaleima on UNIX-aika millisekunteina (aikaväli millisekunteina hetkestä 1.1.1970 00:00 UTC).

Jos käytössä on useamman kuin yhden viimeisimmän arvon pitäminen muistissa, paluuviestissä on mukana kaikki muistissa pidetyt arvot. Esimerkiksi parametrin `present_value_count` arvolla 3 paluuviestin muoto on

```
<id_numero>;<mittausarvo1>;<aikaleima1>;<mittausarvo2>;<aikaleima2>;
<mittausarvo3>;<aikaleima3>
```

Kaikkein viimeisin arvo on listassa viimeisenä (edellisessä esimerkissä `<mittausarvo3>`).

Ennustesuureiden kohdalla kannattaa huomata, että ennusteiden viimeisimmän arvon aikaleima on tulevaisuudessa ja kysymällä viimeisintä arvoa ei siis saada suureen tämänhetkistä arvoa vaan esim. ennustearvo vuorokauden päähän tulevaisuudessa.

7 Datan välitys ulkopuoliselle ohjelmalle

Ohjelman voi myös asettaa välittämään tiettyjen mittauksien arvot eteenpäin toiselle ulkopuoliselle ohjelmalle. Ominaisuus on tehty erityisesti projektiin liittyvän akkudemon aikana tehtävän ohjaukseen tarvittavan tiedon välittämiseen. Tiedon välitys tapahtuu TCP-yhteydellä siten, että ulkopuolinen ohjelma avaa TCP-palvelimen, jolle ProCem RTL -ohjelma lähettää valitut mittausarvot TCP-viesteinä heti kun mittausarvot ovat saapuneet RTL-ohjelmalle. RTL-ohjelma lukee TCP-palvelimen osoitteen ja porttinumeron kooditiedostosta `tcp_utils.py` (nyt käytössä on "127.0.0.1" eli ohjelmaa ajava kone itse ja porttinumero 7777). Tiedonvälityksen TCP-yhteyden kautta voi asettaa päälle tai pois `battery_demo_on` parametrilla ja parametri `ids_for_battery` määrittää niiden mittauksien id-numerolistan, jotka tiedonvälityksen päällä ollessa välitetään ulkopuoliselle ohjelmalle.

Jos tiedonvälitysparametri on päällä, mutta RTL-ohjelma ei saa yhteyttä toiseen ohjelmaan, käännetään tiedonvälitys väliaikaisesti pois päältä. Tällöin 10 minuutin päästä tiedonvälitys käännetään automaattisesti takaisin päälle. Tämä automaattinen keskeytys virhetilanteessa on tehty sen takia, että RTL-ohjelma ei turhaan yritä välittää tietoa silloin kun vastaanottava ohjelma ei ole toiminnassa.

8 Ohjelman suorituksen interaktiivinen ohjaaminen

Ohjelmaan on toteutettu muutamia ohjauskomentoja, joilla osaa ohjelman toimintaa ohjaavista parametreista voi muuttaa. Ohjelman käynnistyessä ohjelma lukee ensin parametritiedoston ja tulostaa sitten tekstin "Give command or press enter key to end:". Tämän jälkeen voi ohjelmalle antaa ohjauskomentoja.

Toteutetut ohjauskomennot ovat (suluissa vastaavien parametrien arvot komennon jälkeen):

- `db-store on`

  Oma datatiedostotallennus päälle (`db_storage_on = true`)

- `db-store off`

  Oma datatiedostotallennus pois päältä (`db_storage_on = false`).

- `iot-ticket on`

  Datan lähetys IoT-Ticket:iin päälle (`iotticket_send_on = true`).

- `iot-ticket off`

  Datan lähetys IoT-Ticket:iin pois päältä (`iotticket_send_on = false`).

- `battery-demo on`

  Datan välitys akkudemo-ohjelmalle päälle (`battery_demo_on = true`).

- `battery-demo off`

  Datan välitys akkudemo-ohjelmalle pois päältä (`battery_demo_on = false`).

- `battery-demo add [id-numbers]`

  Lisää annetut id-numerot akkudemolle välitettäviin mittauksiin (`ids_for_battery += id-numbers`). id-numbers voi olla yksittäinen numero tai useampi numero välilyönneillä erotettuina.

- `battery-demo remove [id-numbers]`

  Poistaa annetut id-numerot akkudemolle välitettävistä mittauksista (`ids_for_battery -= id-numbers`). id-numbers voi olla yksittäinen numero tai useampi numero välilyönneillä erotettuina.

- `list`

  Tulostaa parametrien `db_storage_on`, `iotticket_send_on`, `battery_demo_on` ja `ids_for_battery` arvot.

- `exit`

  Sulkee ohjelma. Myös tyhjä rivi sulkee ohjelman.

9 Historiadatan lähettäminen IoT-Ticket:iin

Datatiedostoihin tallennettua historiallista dataa on mahdollista lähettää IoT-Ticket:n tietokantaan sitä varten kirjoitetulla apuoohjelmalla `procem_data`. Apuohjelman käyttöön tarvitaan IoT-Ticket:n REST API:n tunnukset.

9.1 Apuohjelman käyttö

Ohjelma on kirjoitettu Python-ohjelmointikielellä. Ohjelman käyttämät tiedostot:

- `procem_data.py`
- `iotticket_utils.py`
- `adapters/common_utils.py`
- `procem_data_conf.json`
- `Procem_IDs.csv`

Ohjelman voi käynnistää komennolla:

```
python3 procem_data.py procem_data_conf.json alku_pvm loppu_pvm
```

missä `alku_pvm` on ensimmäinen päivä, josta dataa lähetetään ja `loppu_pvm` on viimeinen päivä. Jos viimeistä päivämäärää ei ole annettu, lähetetään vain yhden päivän data. Päivämäärät annetaan muodossa vuosi-kuukausi-päivä ja luvuissa sallitaan etunolla. Esim. heinäkuun 15. päivä vuonna 2018 voidaan antaa joko muodossa 2018-7-15 tai 2018-07-15.

Elokuun ensimmäisten viiden vuorokauden datan voisi lähettää komennolla:

```
python3 procem_data.py procem_data_conf.json 2018-8-1 2018-8-5
```

9.2 Parametritiedoston kuvaus

Parametritiedoston `procem_data_conf.json` parametrien selitykset:

- `id-configuration-file:` mittauslistatiedoston nimi (csv-tiedosto)
- `id-delimiter:` csv-tiedostossa käytetty sarakkeita erottava merkki
- `data-folder:` hakemisto, josta datatiedostoja luetaan
- `remote-server:` ulkopuolisen palvelimen tunnus ja osoite
- `remote-folder:` hakemisto ulkopuolisella palvelimella
- `data-filename-format:` tiedostonimissä käytetty päivämäärien muoto
- `data-filename-suffix:` purettujen datatiedostojen nimen tunniste
- `data-compressed-suffix:` pakattujen datatiedostojen nimen tunniste
- `data-delimiter:` datatiedostoissa käytetty sarakkeita erottava merkki
- `username:` IoT-Ticket:n REST API:n käyttäjätunnuksen nimi
- `password:` IoT-Ticket:n REST API:n käyttäjätunnuksen salasana
- `base-url:` IoT-Ticket:n REST API:n osoite
- `deviceid:` IoT-Ticket:n laitteen tunnus
- `iotticket-max-packet-size:` yksittäisen lähetyksen maksimikoko lähetettäessä dataa IoT-Ticket:iin
- `iotticket-minimum-delay-s:` vähimmäisaika kahden IoT-Ticket-lähetyksen aloittamisen välillä
- `iotticket-maximum-retries:` IoT-Ticket-lähetysyrityksien maksimäärä
- `iotticket-queue-size:` puskurin maksimikoko luettaessa mittauksia datatiedostoista
- `iotticket-buffer-size:` kerättyjen datapakettien maksimikoko IoT-Ticket:lle lähetettäessä

Mittauslistatiedostossa `Procem_IDs.csv` luetellaan jokaisen käsiteltävän mittaussuureen id-numero sekä IoT-Ticket:n tarvitsemat tiedot: nimi, polku, yksikkö ja datatyyppi. Lisäksi jokaiselle mittaussuurelle on merkitty tieto mahdollisesta luottamuksellisuudesta ja erikseen merkintä siitä

lähetetäänkö kyseisen mittaussuureen arvot IoT-Ticket:iin. Repositoriossa on pohja mittauslistatiedostolle, jossa on mukana esimerkki mittaussuurerivistä.

9.3 Apuohjelman toiminnan kuvaus

Apuohjelman toiminnan vaiheet:

- Komentorivillä annetaan parametritiedosto ja alku- ja loppupäivämäärät (lähetään vain 1 päivä jos annetaan vain alkupäivämäärä).
- Luetaan tarvittavat tiedot parametritiedosta.
- Haluttujen vuorokausien data käsitellään aikajärjestyksessä vanhimmasta uusimpaan.
- Yksi vuorokausi kerrallaan luetaan datatiedoston dataa sisään.
    - Tiedostonimien muodot ja polut annetaan parametritiedostossa
    - Jos valmiiksi purettua datatiedostoa ei löydy, puretaan pakattu tiedosto väliaikaisesti lukua varten.
    - Jos pakattua tiedostoakaan ei löydy, yritetään kopioida pakattu datatiedosto ulkoisesta tietovarastosta ja puretaan se kopioinnin jälkeen.
- Luettu data kootaan datapaketeiksi, joiden kokoa voi säätää parametrilla `iotticket-buffer-size`.
- Vain mittaukset, jotka on merkitty lähetettäväksi IoT-Ticket:iin ja jotka eivät ole luottamuksellisia, otetaan mukaan datapaketteihin (valittuun dataan voi vaikuttaa muokkaamalla ID-listaustiedostoa).
- Kootut datapaketit lähetetään IoT-Ticket:iin yksi kerrallaan.
    - Käytössä on sama optimointisysteemi (järjestäminen ja pienempiin osiin jako) kuin Procem RTL -komponentissa. Erona vain yksi lähetys tehdään kerrallaan ja epäonnistuneiden lähetyksien kierrätys takaisin lukujonoon ei ole käytössä.
- Kun koko vuorokauden data on käsitelty, poistetaan ohjelman aikana luodut tiedostot ja siirrytään seuraavaan vuorokauden datan käsittelyyn, kunnes kaikki halutut vuorokaudet on käsitelty.

9.4 Huomioita

- Jos halutaan siirtää vain tiettyjen tuntien data, pitää ohjelmaa joko muokata ottamaan lähetykseen mukaan vain halutut tunnit tai vaihtoehtoisesti suodattaa datatiedostot erikseen sisältämään vain halutun ajan dataa.