TARTU ÜLIKOOL
Arvutiteaduse instituut
Infotehnoloogia mitteinformaatikutele õppekava

Margit Aun

Instrumentide monitoorimissüsteem Tartu observatooriumi atmosfääriseire töörühmale

Magistritöö (15 EAP)

Juhendaja(d): Ilmar Ansko, MSc
Tõnis Eenmäe, MSc

Tartu 2020

Instrumentide monitoorimissüsteem Tartu observatooriumi atmosfääriseire töörühmale

Lühikokkuvõte:

Magistritöö raames loodi Tartu Ülikooli Tartu observatooriumi atmosfääriseire töörühmale instrumentide monitoorimissüsteem. Süsteemi peamine eesmärk on jälgida mõõtmiste toimumist ja uute andmete puudumise või veateadete korral saata vastutavale isikule e-kirja teel teavitus. Süsteemi töö tulemusel jõuab teave instrumentide töö katkemisest kiiremini vastutavate isikuteni ja seetõttu vähenevad mõõdetavate andmete aegridade katkestuste kestused. Süsteem on ülesehitatud JavaScript Node.js käitussüsteemi põhjal ja hõlmab kahte instrumenti. Esiteks ultraviolettkiirguse mõõtmiseks kasutatav topeltmonokromaator Bentham DMc150F-U. Spektromeetrit kontrollitakse päikesetõusust päikeseloojanguni iga 15 min järel vastavalt mõõtmiste sagedusele otse MySQL andmebaasist. Grafana töölaualt on võimalik jälgida mõõtmiste tulemust ning teavitus on seadistatud instrumendi töötemperatuuri vahemiku jaoks. Teiseks instrumendiks on NASA AERONET võrgustikku kuuluv päikesefotomeeter CIMEL CE318, millega mõõdetakse aerosoolide sisaldust atmosfääris. Seda instrumenti kontrollitakse üks kord päevas kell 08:00 UTC aja järgi. Kontrollitakse mõõtmisfailide tekkimist instrumendi juhtarvutisse ja andmete edastamist ning veateateid AERONETi kodulehe kaudu. Süsteemi ülesehitus võimaldab tulevikus lisada uusi instrumente.

Võtmesõnad:

Monitoorimine, keskkonnamõõtmised, Node.js, Grafana, spektromeeter, CIMEL, AERONET

CERCS:

P170 Arvutiteadus, arvutusmeetodid, süsteemid, juhtimine

Instrument monitoring system for the atmospheric monitoring working group of Tartu Observatory

Abstract:

A monitoring system for instruments used by the group working on remote sensing of atmosphere in University of Tartu Tartu Observatory was created. The main aim of the system is to monitor ongoing measurements and send a notification by e-mail in the case of instrument misbehavior. As a result, the information about malfunction will reach to the operator faster and therefore the caps in measured timeseries will shorten. The system has been built by using the Node.js implementation of JavaScript and is currently monitoring two instruments. The first instrument is Bentham DMc150F-U spectroradiometric system, a double monochromator used for spectral measurements of ultraviolet radiation. There is also a dashboard set up for Bentham using freely available Grafana software. The instrument is being checked directly from MySQL database from sunrise to sunset every 15 min. Grafana enables to check the measurements results and alerting has been set for the temperature of the instrument. The second instrument is CIMEL CE318 Sun photometer belonging to NASA AERONET that measures aerosols in atmosphere. This instrument is being checked once a day, at 08:00 UTC. Presence of the measurement file is checked in the controlling computer while the data delivery and error messages are extracted from the AERONET webpage. The architecture of the system enables to add more instruments in the future.

Keywords:

Monitoring, environmental measurements, Node.js, Grafana, spectrometer, CIMEL, AERONET

CERCS:

P170 Computer science, numerical analysis, systems, control

Sisukord

1. Sissejuhatus

Looduskeskkonna kirjeldamiseks ja seal toimuvatest protsessidest aru saamiseks on vajalik koguda ja töödelda suurel hulgal andmeid. Need andmed saavad pärineda kohalikest ehk *in situ* mõõtmistest, satelliitidelt või mudelarvutustest. Neist olulisimad ja täpseimad on just esimesed – kindlas ruumipunktis tehtavad mõõtmised, mis võimaldavad kirjelda antud geograafilise asukoha hetke tingimusi, jälgida ajalisi muutusi ja valideerida satelliidiandmeid ning mudelarvutuste tulemusi (Lamy et al., 2018; Loew et al., 2017; Verhoelst et al., 2015). *In situ* mõõtmisi on võimalik teha lühiajaliste kampaaniate raames, kuid tihti on olulised pikaajalised rutiinmõõtmised. Paljudele looduslikele parameetritele on iseloomulik suur varieeruvus ja pikaajaliste muutuste hindamiseks on oluline mõõtmisandmete pikkade aegridade olemasolu (Staehelin et al., 2009). Mõõtmiste läbiviimisel mängivad olulist rolli erinevad instrumendid ja sensorid, mis salvestavad mõõtetulemused elektrooniliselt ja on ühendatud arvutivõrku (Devadithya et al., 2005).

Võrgustiku kasutamise ideele teaduslikel eesmärkidel tuli interneti pioneer J. C. R. Licklider 1960ndate alguses (Foster ja Kesselman, 2004). Erinevate arvutite ja nende ressursside ühendamise ideed arendasid edasi 1990ndatel Carl Kesselman, Ian Foster ja Steve Tuecke ja seda nimetati võrkandmetöötluseks (ingl *grid computing*)[1]. Reeglina on eesmärgiks töötada ühise ülesandega, mis on jaotatud mitmeks väiksemaks. Devadithya et al (2005) arendas välja CIMA (Common Instrument Middleware Architecture), mis võimaldab võrku loogiliselt ühendada väga erinevaid instrumente ja neid võrgu kaudu juhtida. Samuti on sellise süsteemi oluline osa andmete kogumine.

Taoline süsteem aga ei sobi kõikide instrumentide jaoks. Tihti on vaja jälgida individuaalseid instrumente, tagamaks mõõtmisandmete usaldusväärsust ja järjepidevust. Sageli ei piisa töö katkemisel juhtprogrammi kaudu taaskäivitamisest vaid vajalik on teadlase või inseneri sekkumine, kuna katkestuse võis põhjustada riistvaraline viga.

Loodud on mitmeid erinevaid andmevoogude ja süsteemide jälgimissüsteeme, mis võimaldavad kogutud andmeid visualiseerida, töödelda ning saata veateateid. Tasuliste programmide näiteks on Redash[2], Tableau[3] ja Geckoboard[4]. Tasuta ja tihti avatud lähtekoodiga lahendusteks on muu hulgas Grafana[5], Kibana[6], Netdata[7] ja Graphite[8]. Kõik need lahendused erinevad üksteisest, kuid peamine eesmärk on siiski andmete visualiseerimine ja tõlgendamine, kusjuures osadel juhtudel (nt. Tableau) on peamiseks eesmärgiks teadusandmed, samas kui teistel juhtudel keskendutakse süsteemi enda monitoorimisandmetele (nt. Netdata). Mitmel nimetatud vahendil (nt. Tableau, Kibana) puudub lihtne sisseehitatud teavitussüsteem[9], mis on antud töö peaeesmärgiks.

Tartu observatooriumi (TO) atmosfääriseire töörühmal on kasutusel 2 instrumenti, mis teostavad igapäevaselt automatiseeritud atmosfääriparameetrite mõõtmisi. Näiteks mõõdetakse ultraviolettkiirgust (UV-kiirgus), mis on elektromagnetkiirgus lainepikkuste vahemikus 100 – 400 nm. Kiirgus lainepikkusega alla 280 nm maapinnani ei jõua (UNEP, 1998). Kiiritustihedus maapinnal sõltub peamiselt Päikese seniitnurgast, pilvkattest, atmosfääri koosisest

[1] https://www.techopedia.com/definition/87/grid-computing
[2] https://redash.io/
[3] https://www.tableau.com/
[4] https://www.geckoboard.com/
[5] https://grafana.com/
[6] https://www.elastic.co/kibana
[7] https://www.netdata.cloud/
[8] https://graphiteapp.org/
[9] https://logz.io/blog/grafana-vs-kibana/

(aerosoolide kogus ja tüüp) ning aluspinna albeedost (Kerr, 2005). UV-kiirguse mõõtmine on oluline tema peamiselt kahjuliku mõju tõttu elusorganismidele (sh inimestele) ja materjalidele (Kerr, 2005; Medhaug et al., 2009), samas mängib UV-kiirgus olulist rolli vitamiin D sünteesil (Olds, 2010). Alates 2009. aastast mõõdab Tõraveres UV-kiirguse spektraalset kiiritustihedust radiomeetriasüsteem topeltmonokromaatoriga Bentham DMc150F-U spektri vahemikus 280–400 nm sammuga 1 nm. Mõõtmised toimuvad iga 15 minuti järel kui Päikese seniitnurk on alla 90 kraadi. Teine kasutusel olev instrument on NASA AERONET võrgustikku kuuluv päikesefotomeeter CIMEL CE318. See instrument mõõdab spektraalset Päikese kiiritustihedust ja taeva kirkust päeva jooksul sõltuvalt Päikese kõrgusest u. 15-minutilise intervalliga ning peamine tulem on aerosooli optiline paksus (AOD), mis iseloomustab kiirguse neeldumist atmosfääris aerosoolide tõttu (Holben et al., 1998).

Mõlema nimetatud instrumendi abil tehakse pikaajalisi rutiinmõõtmisi ja vajavad seega pidevalt töötavat jälgimissüsteemi. Instrumentide töö on aeg-ajalt häiritud või katkeb keskkonnast tingitud või tehnilistel põhjustel. Tartu observatooriumis on loodud rakendus, mis võimaldab jälgida radiomeetriasüsteemi tööd ning kuvada mõõtmistulemusi. Veebirakendus on olemas ka AERONET-i võrgustikul, kust on võimalik näha mõõtmistulemusi peaaegu reaalajas. Mõõtmiste toimumise kohta informatsiooni saamiseks peaks teadustöötaja või operaator neid veebilehti pidevalt külastama. See on organisatsiooniliselt ja tehniliselt tülikas ning aeg-ajalt võib ette tulla pikemaid perioode, mil mõõteriistad on valveta või hooldatud vaid riistvaraliselt. Sellistel juhtudel võib tekkinud tehniline tõrge jääda märkamata ja mõõtmiste aegridadesse võib tulla pikem katkestus. Kogutavad andmed ei ole aga vajalikud ainult seadet hooldava töörühma liikmetele vaid leiavad kasutust ka globaalselt (nt. Volkova et al., 2018), seega on vajalik viia andmekadu miinimumini.

Töö eesmärgiks on luua TO atmosfääri seire töörühma instrumentide monitoorimissüsteem, mis annab vastutavatele isikutele e-kirja teel märku tekkinud probleemidest. Esialgu kaasatakse Bentham DMc150F-U ja CIMEL CE318, kuid süsteem peab olema piisavalt paindlik uute instrumentide lisamiseks. Esiteks valitakse välja vabavaraline tarkvararakendus, mis seadistatakse eelpoolnimetatud instrumentide monitoorimiseks. Teiseks luuakse Javascript Node.js[10] abil originaalne lahendus sama eesmärgi täitmiseks. Kolmandaks antakse valminud lahendustele hinnang.

[10] https://nodejs.org/en/

2. Mõisted ja terminid

Alljärgnevalt on töö paremaks mõistmiseks lahti seletatud mõned keskkonnaseire valdkonna mõisted ning toodud on ka infotehnoloogia alased mõisted, et kasutatud tõlked oleksid üheselt arusaadavad.

2.1 Keskkonnaseire valdkonna mõisted ja terminid

Kui ei ole märgitud teisiti, siis on alljärgnevalt kasutatud "Eesti kiirguskliima teatmikus" (Russak ja Kallis, 2003) kasutatud mõistete seletusi.

AERONET-i inversiooni produkt kirjeldab aerosooli optilisi omadusi terves atmosfäärisambas, mis on saadud otse- ja hajuskiirguse mõõtmistest[11].

Aerosooli optiline paksus (AOD, ingl *aerosol optical depth*) on kvantitatiivne hinnang aerosoolide hulgale atmosfääris. AOD mõõdab valguskiirguse nõrgenemist atmosfääris aerosoolide tõttu. Mida suurem on AOD, seda enam kiirgus väheneb[12].

Albeedo on pinnalt hajunud või peegeldunud ja pinnale langenud elektromagnetilise kiirguse suhe; sõltub kiirguse langemisnurgast, spektraalsest koostisest, pinna ainest, struktuurist ja niiskuse sisaldusest

Atmosfäär on Maad ümbritsev gaasikiht[13].

Atmosfääri aerosool on atmosfääris lenduvad väikesed (mõõtmed ulatuvad mõnest nanomeetrist kuni mõnekümne mikromeetrini) tahked või vedelad osakesed, mis võivad olla nii looduslikud kui inimtekkelised.

Hajuskiirgus on osa päikesekiirgusest, mis pärast hajumist õhu molekulidel, aerosoolil, veeaurul ja pilvedes langeb maapinnale.

Kollimaator on optikasüsteem, mida kasutatakse paralleelsete kiirte kimbu saamiseks. Koosneb objektiivist ja selle fokaaltasandis asetsevast valgustatud pilust või muust märgist[14].

Otsekiirgus ehk **Päikese otsene kiirgus** on see osa kiirgusest, mis jõuab päikeseketta suunast maapinnale praktiliselt paralleelsete kiirte kimbuna. Seda mõõdetakse kiirtega risti pinnal.

Päikese seniitnurk (SZA, ingl *solar zenith angle*) on Päikese suuna ja seniidi vaheline nurk. Kui Päike on otse pea kohal on SZA 0° ja horisondil 90°.

Sadestatav vesi ehk **sadestatav veeaur** on suurus, mis iseloomustab atmosfääris oleva veeauru hulka (Abbasy et al., 2017).

Ultraviolettkiirgus (**UV-kiirgus**) on elektromagnetkiirgus lainepikkuste vahemikus 100-400 nm[15].

[11] https://aeronet.gsfc.nasa.gov/new_web/Documents/Inversion_products_for_V3.pdf
[12] http://cimss.ssec.wisc.edu/goes/OCLOFactSheetPDFs/ABIQuickGuide_BaselineAerosolOpticalDepth.pdf
[13] http://entsyklopeedia.ee/artikkel/atmosf%C3%A4%C3%A4r3
[14] http://entsyklopeedia.ee/artikkel/kollimaator1
[15] https://www.ilmateenistus.ee/wp-content/uploads/2013/01/eesti_ilma_riskid_2012_ultraviolettkiirgus.pdf

2.2 Infotehnoloogia valdkonna mõisted ja terminid

Kui ei ole märgitud teisiti, siis on alljärgnevate mõistete seletamiseks kasutatud veebilehte www.vallaste.ee.

Võrkandmetöötlus (ingl *grid computing*) tähendab paljude võrku ühendatud arvutite üheaegset kasutamist ühe ülesande lahendamiseks.

Hoidla (ingl *repository*) on rakendustarkvara juurde kuuluva info andmebaas, mis sisaldab autori nime, andmeelemente, sisendeid, protsesse, väljundeid ja nendevahelisi suhteid. Hoidlat kasutatakse CASE või rakenduste arendussüsteemis objektide ja ärireeglite identifitseerimiseks nende korduvkasutuseks. See võib olla projekteeritud ka kolmandate osapoolte CASE-toodete integreerimiseks.

Avatud lähtekood (ingl *open source*) on mistahes programm, mille lähtekood on tehtud programmeerijatele ja kasutajatele kättesaadavaks nii kasutamiseks kui muutmiseks. Patenttarkvara (omandiõigusega kaitstud tarkvara) tootjad üldiselt ei avalda lähtekoode. Avatud lähtekoodiga tarkvara töötatakse välja ja arendatakse koostöös avalikkusega ning see on saadaval tasuta.

Sündmusjuhitav programm (ingl *event-drive*) on programm, mis ootab sündmuste toimumist ja reageerib neile, selle asemel et teha läbi ette kindlaksmääratud sammud. Sündmus võib olla näit. hiirega klikkimine mingis kohas ekraanil või klaviatuurilt sisestatud käsk.

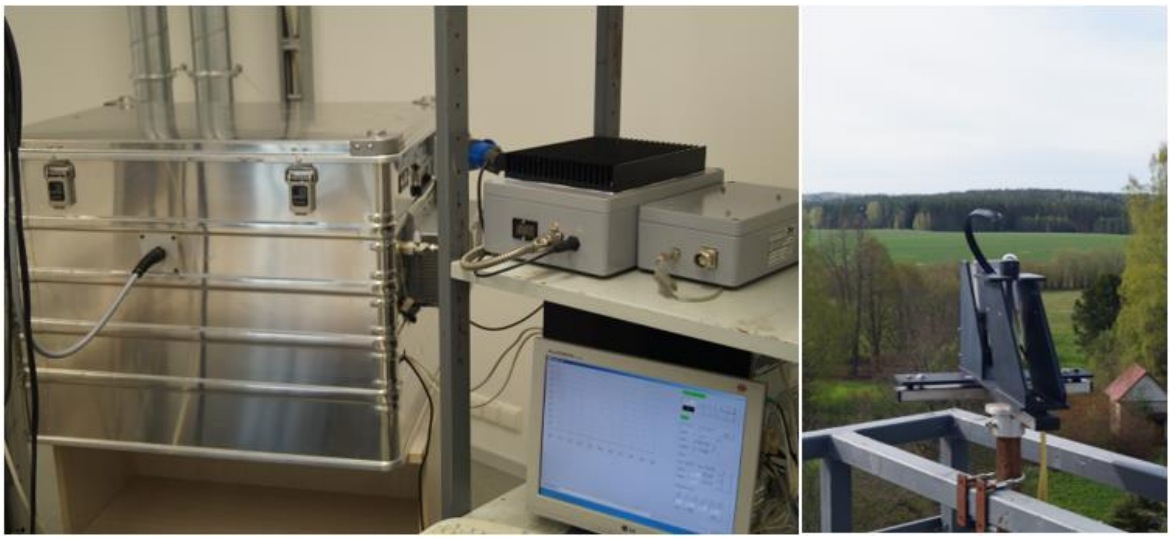
Käituskeskkond ehk **käitusfaasikeskkond** (ingl *runtime environment*) on keskkond, milles programmi täidetakse. Põhimõtteliselt on see arvuti "platvorm", kuhu kuuluvad keskprotsessor, opsüsteem ja muud süsteemiprogrammid (andmebaasihaldur, võrgustamistarkvara jne).

Virtuaalne privaatvõrk (**VPN**) (ingl *virtual private network*) on privaatvõrk, mis kasutab avalikku telekommunikatsiooni infrastruktuuri, säilitades samal ajal privaatsuse ja turvalisuse. Turvalisuse tagamiseks kasutatakse tunneldamist ja vastavaid turvaprotseduure.

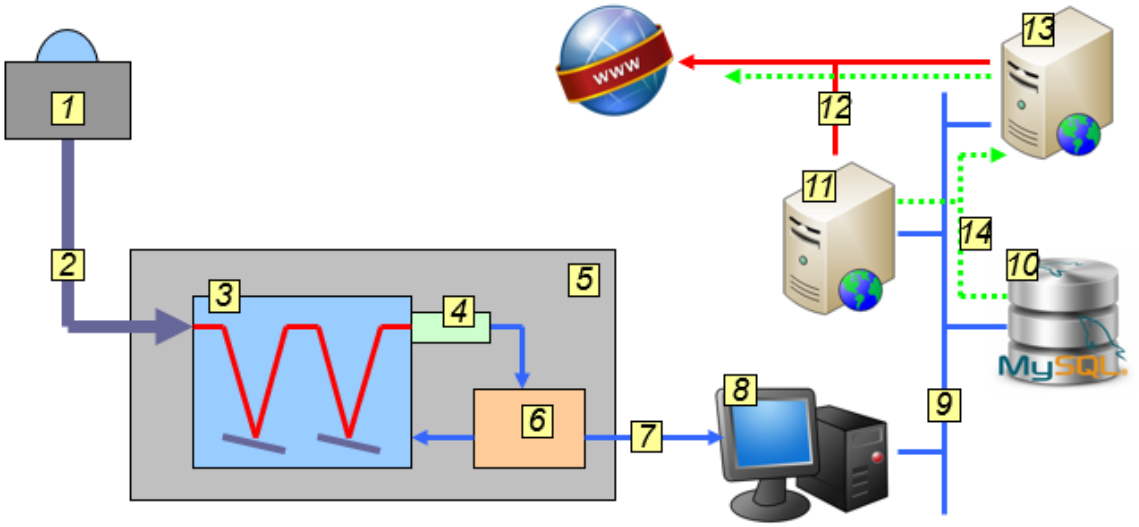
3. Instrumendid ja nende senine monitoorimine

3.1 Bentham DMc150F-U

Alates 2009. aastast on Tartu observatooriumis UV-kiirguse spektraalse kiiritustiheduse mõõtmiseks kasutusel Bentham Instruments Ltd. (Suurbritannia) poolt toodetud topeltmonokromaator Bentham DMc150F-U (Joonis 1) (Aun, 2017). Süsteemi kuulub fotokordisti ja aparatuur asub termostaatkastis Envirobox. Kasti sisetemperatuur hoitakse 21 °C juures, et tagada mõõtmiste stabiilsus. Temperatuuri kõikumisi tuleb eriti ette suvel, kui kuumade ilmade korral jääb jahutusvõimsust napiks ja temperatuur termokastis võib tõusta üle soovitud väärtuse. Kiirguse vastuvõtjaks on koosinus hajutaja D6_env, mis on ühendatud spektromeetriga 6 m pikkuse kvartsfiiber valgusjuhi abil. Hajutaja asub Tartu observatooriumi peahoone katusel ning termokast instrumendiga selle all asuvas ruumis (Joonis 2). Mõõtesüsteemi on täpsemalt kirjeldatud Veismanni ja Eerme poolt (2011).



Joonis 1. Päikese spektroradiomeetriasüsteem monokromaatoriga Bentham DMc150F-U. Siseruumides hoitav termostaatkamber (vasakul) ja katusel asuv koosinushajutaja.



Joonis 2. Bentham DMc150F-U tööskeem Tartu observatooriumis. 1 – koosinushajutaja; 2 – valgusjuht; 3 – topeltmonokromaator; 4 – fotokordisti; 5 – Envirobox; 6 – kontrolleriplokk; 7 – USB-liides; 8 – juhtarvuti; 9 – observatooriumi laboritevõrk; 10 – MySQL andmebaas; 11 – veebiserver; 12 – laivõrk; 13 – monitoorimissüsteemi arvuti; 14 – andmete liikumine monitoorimissüsteemis.

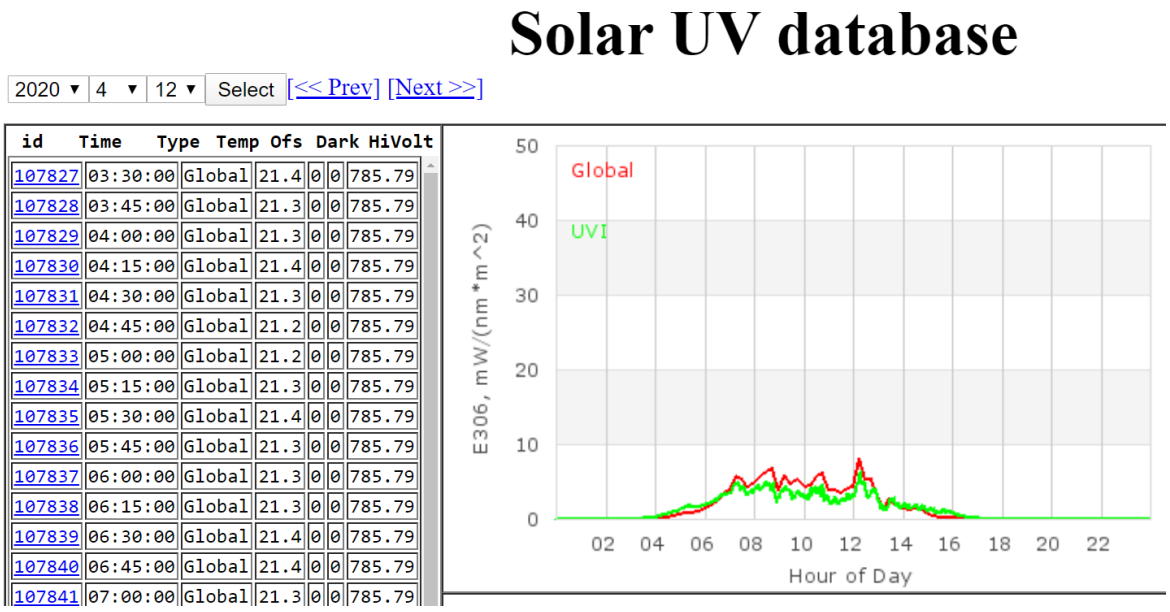
Radiomeetriline kalibreerimine välitingimustes toimub kaks korda aastas kasutades kiirgusallikana Bentham CL6 välikalibraatorit, mida omakorda kalibreeritakse FEL-lambiga TO laboris. TO kiiritustiheduse mõõtmised on sertifitseeritud ning jälgitavad rahvusvahelise mõõtühikute süsteemini SI. Spektraalse kiiritustiheduse laiendmääramatus (k=2) on kuni 9 % UVB piirkonnas (280–315 nm) ja 6% UVA piirkonnas (315–400 nm).

Mõõtmised algavad hommikul, kui päikeseseniitnurk (SZA) on alla 90 kraadi ja kestavad 15 min intervalliga kuni SZA tõuseb üle 90 kraadi.

Monokromaator on USB-liidese kaudu ühendatud juhtarvutiga. Juhtprogramm on töötatud välja TO-s, kompilaatoriks vabavaraline Free Pascal[16], arvuti operatsioonisüsteemiks on MS Windows XP. Juhtprogramm salvestab skaneeritud spektrid ASCII-tüüpi andmefailidesse ning andmed salvestatakse ka MySQL andmebaasi, mis asub observatooriumi varundatud serverivõrgus.

Spektrikirjed on paigutatud ühte andmebaasitabelisse. Kirjes on kokku 139 välja: lisaks spektriandmetele mõõtmise ajatempel, termokasti temperatuur, fotoelektronkordisti anoodpinge jm tehnilised parameetrid. Iga mõõtmine on identifitseeritav ainulaadse identifikaatoriga (MySQL PRIMARY KEY). Praeguseks on andmebaasis üle 93500 mõõtmise (12.04.2020). Mõõtmised on jälgitavad ja andmed kättesaadavad eraldi loodud veebilehel (Joonis 3).

[16] www.freepascal.org



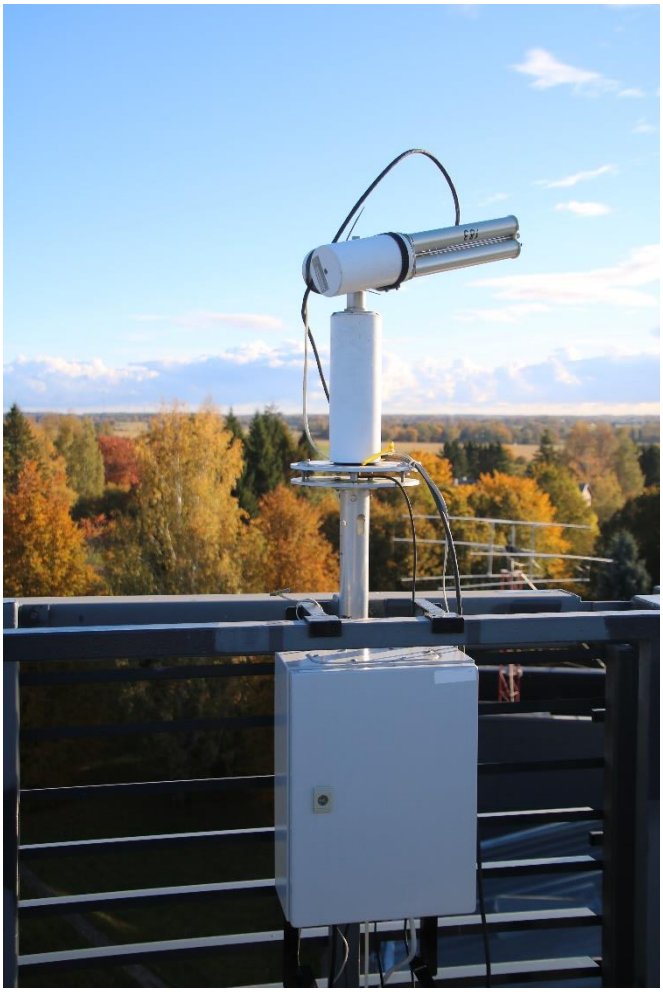
Joonis 3. Bentham DMc150F-U andmetele ligipääsu võimaldava kodulehe kuvatõmmis (12.04.2020).

Kuigi eelpool toodud veebileht on kõikidele vabalt ligipääsetav, on see mõeldud töögrupi liikmetele mõõtmiste jälgimiseks ja seetõttu ei ole siinkohal toodud ka veebilehe aadressi. Instrumendi töö jälgimiseks oli siiani kaks võimalust: 1) instrumendi füüsilise hoolduse käigus tehtav kontroll ja 2) veebilehe monitoorimine töörühma liikmete poolt. Viimase kaudu on võimalik kontrollida, kas spektriandmete salvestamine toimub. Probleemide korral teavitatakse vastutavat inseneri, kes kontrollib instrumenti, kõrvaldab vea ja taaskäivitab mõõtmised. Veebilehel olevalt graafikult on võimalik näha ka andmete suuri anomaaliaid, nagu null-väärtused, kuid tegelik hinnang andmete kvaliteedile antakse kalibreerimise ja hilisema andmetöötluse käigus (nt. mudelarvutuste vm võrdlusmõõtmiste käigus). Kuna UV-kiirguse väärtused on looduslikult väga varieeruvad ei ole näiteks 10 % muutus aparatuuri tundlikukuse triivi tõttu ilma täpsemate arvutusteta tuvastatav. Seetõttu on monitoorimise esmane eesmärk jälgida just mõõtmiste plaanipärast toimumist.

3.2 CIMEL CE318

Ameerika Ühendriikide Riiklikule Lennundus-ja Kosmoseagentuurile (NASA, National Aeronautics and Space Administration) kuuluva Aerosoolide Robootilise Võrgustikuga[17] (Holben et al., 1998) (AERONET, AErosol RObotic NETwork) liitus TO 2002. aastal, mil alustati mõõtmisi päikesefotomeetriga CIMEL (Cimel Electronique, Prantsusmaa[18]). Hetkel on kasutusel CIMEL CE318, nr 128, mis asub Tartu observatooriumi peahoone katusel (Joonis 4). Selle abil mõõdetakse aerosooli optilist paksust (AOD), inversiooni produkte ja sadestatava vee hulka.

[17] https://aeronet.gsfc.nasa.gov/
[18] https://www.cimel.fr/

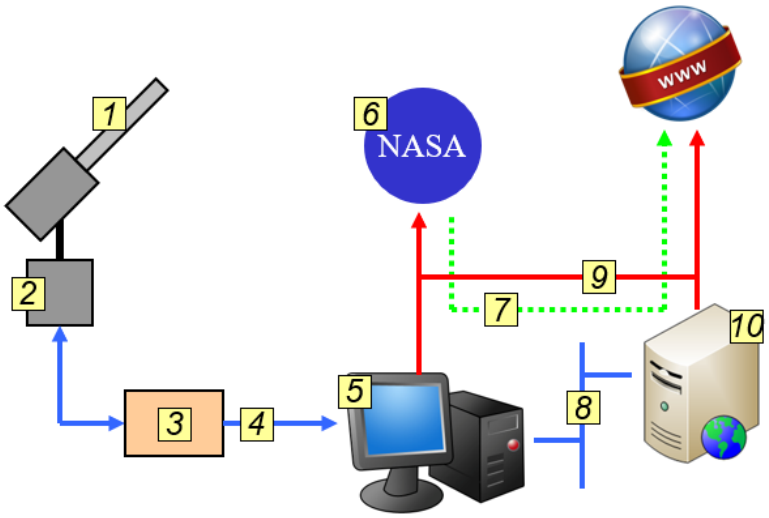


Joonis 4. CIMEL päikesefotomeeter Tartu Observatooriumi katusel.

Süsteem koosneb mõõtepeast kahe kollimaatoriga, mis on kinnitatud motoriseeritud aluse (roboti) külge, mille abil pööratakse mõõtepea Päikese suunas (või taevasse). CIMEL-i juhtkontroller asub ilmastikukindlas kastis mõõtepea lähedal. Juhtkontroller on ühendatud katusealuses seadmeruumis asuva arvutiga RS-232 kaabli abil. CIMEL teostab mõõtmisi autonoomselt kontrollerisse salvestatud programmi järgi. Arvuti loeb perioodiliselt mõõtmistulemused kontrolleri ringpuhvrist, saadab AERONET-i serverisse ning salvestab ka kettafaili (laiendiga .k7). Juhtprogramm on kompileeritud AERONET-ilt saadud lähtekoodist ja töötab operatsioonisüsteemis Debian Linux. Kettafail varundatakse täiendavalt observatooriumi serverisse (Joonis 5).

Kasutusel on kaks mõõterežiimi – Päikese otsekiirgus ja taeva kiirgus. Päikesekiirgust mõõdetakse 8 erineval lainepikkusel: 340, 380, 440, 500, 670, 870, 940 ja 1020 nm. Mõõtmised toimuvad suurte SZA korral 0,25 õhumassi intervalliga, väikestel SZA korral toimuvad mõõtmised iga 15 minuti järel. Mõõtmiste vaheajal on sensori pea suunatud nadiiri, et vältida sisendoptika saastumist sademete või tolmu tõttu. Mõõtmised toimuvad vaid juhul, kui Päikese ketas on pilvevaba. Mõõtesüsteem ja mõõtmised on detailsemalt kirjeldatud AERONET-i kodulehel[19].

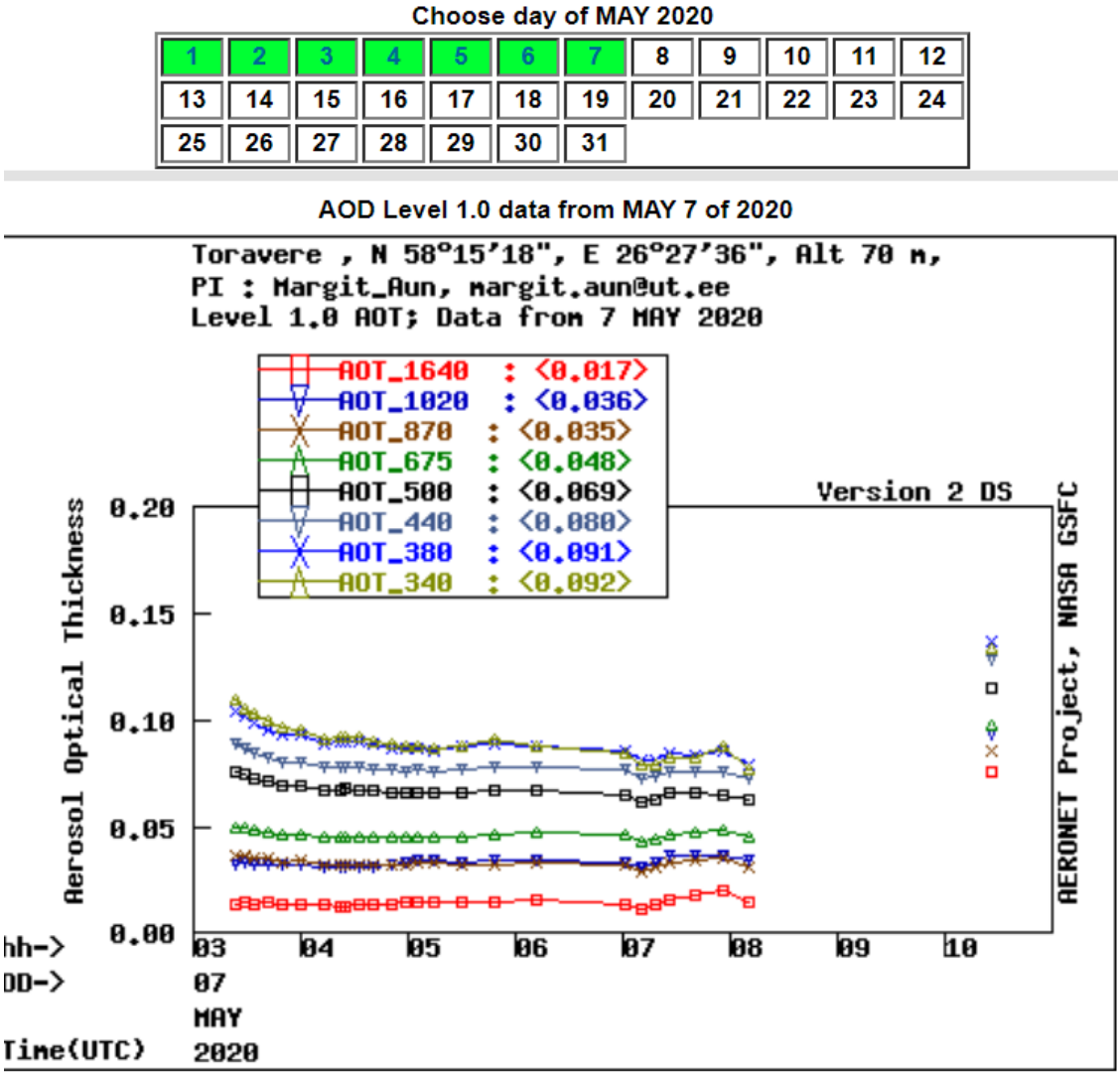
[19] https://aeronet.gsfc.nasa.gov/new_web/system_descriptions.html



Joonis 5. CIMEL CE318 tööskeem Tartu observatooriumis. 1 – kollimaatoriga mõõtepea; 2 – robot; 3 – juhtkontroller; 4 - RS-232 liides; 5 – logiv arvuti; 6 – NASA AERONET'i andmebaas; 7 – andmete liikumine monitoorimissüsteemis; 8 – observatooriumi laborite-võrk; 9 – laivõrk; 10 – monitoorimissüsteemi arvuti.

Andmed on kätte saadavad läbi AERONET-i kodulehe[20] (Joonis 6). Andmed esitatakse kolmel erineval töötlustasemeel. Tase „1" on töötlemata toorandmed, tase „1.5" sisaldab automaatset pilvisuse filtreerimist ja tase „2" tähendab lõplikke kontrollitud kvaliteediga kalibreeritud tulemusi. Kalibreerimiseks saadetakse instrument AERONET-i laborisse. Instrumendi vahetus toimub reeglina üheaastase perioodiga. Kalibratsioon rakendatakse mõõtmistulemustele tagasiulatuvalt ja see tähendab, et tasemega "2" andmete saamiseks võib kuluda kuni 2 aastat.

[20] https://aeronet.gsfc.nasa.gov/cgi-bin/type_one_station_opera_v2_new?site=Toravere&nachal=2&level=1&place_code=10



Joonis 6. Väljavõte CIMEL CE318 andmetele ligipääsu võimaldava kodulehelt[21] (07.05.2020).

Koos mõõtmistulemustega esitatakse AERONET-i veebilehel ka metainfo, mis mh sisaldab instrumentide veateateid.[22].

CIMEL CE318 jälgimine on siiani toimunud seadme igapäevase visuaalse vaatluse ja AERONET-i veebilehe perioodilise kontrollimise teel. Lisaks saadetakse e-kirja teel informatsiooni probleemide tekkimisel. Need aga ei ole automatiseeritud kirjad, vaid saadetakse AERONET-i vastutava töötaja poolt vastavalt vigade tuvastamisele. Kuna võrgustikus on väga palju jaamu (üle 800), ei saa sellist teavitussüsteemi pidada otstarbekaks. Nendel põhjustel otsustati CIMEL CE318 lülitada käesolevas töös kirjeldatud monitoorimissüsteemi.

[21] https://aeronet.gsfc.nasa.gov/cgi-bin/type_one_station_opera_v2_new?site=Toravere&nachal=2&level=1&place_code=10

[22] https://aeronet.gsfc.nasa.gov/new_web/Data_Transfer_Logs/K7/last_1440_minutes.html

4. TO atmosfääriseire töörühmale loodud monitoorimissüsteem

Instrumentide töö kontrollimiseks loodi kaks tarkvaralist lahendust. Esimene põhines andmeanalüüsipaketil Grafana ja teine Javascript Node js-il.

Grafana valiti välja erinevate vabavaraliste lahenduste hulgas. Valik tugines programmide veebilehtedele, artiklitele, blogidele ja konsultatsioonidele kolleegide ning IT valdkonna spetsialistidega. Hinnati erinevate tarkvaralahenduste eesmärke, võimalusi ja puudusi. Grafana eelisteks oli lihtne liidestamine MySQL andmebaasiga, samuti sisseehitatud teavitussüteem, mis võimaldab täita töö peaeesmärki.

Grafana on vabavaraline ja avatud lähtekoodiga programm, mis on mõeldud andmete visualiseerimiseks ja analüüsimiseks. Selle abil on võimalik aegridade andmeid pärida ja visualiseerida ning vigade tekkimisel saata veateateid. Esimene versioon ilmus 2014. aastal[23]. Grafanas on võimalik luua ülesandele sobiv töölaud, mis kuvab vajalikud andmed soovitud kujul.

Originaallahenduse loomiseks kasutati JavaScripti asünkroonset sündmusjuhitavat käituskeskkonda Node.js-i. Node.js on mõeldud skaleeritavate võrgurakenduste loomiseks[24]. Tegu on avatud lähtekoodi ja platvormiülese keskkonnaga, mis on loodud 2009. aastal[25].

Kirjade saatmiseks loodi eraldi seisev Google'i konto aadressiga to.healthcheckservice@gmail.com. Kontot ja selle parooli kontrollib töö autor.

4.1 Süsteemi üldiseloomustus

Atmosfääriseire töörühma instrumentide monitoorimissüsteemi jaoks loodi observatooriumi serveris virtuaalarvuti atmos.to.ee, millele juurdepääs on võimaldatud Tartu Ülikooli arvutivõrgust (või väljastpoolt virtuaalse privaatvõrgu (VPN) vahendusel) läbi SSH. Juurdepääs on kasutajanimede ja paroolidega. Virtuaalarvutile on eraldatud 2-tuumaline 64-bitine protsessor, operatsioonisüsteemiks on GNU/Linux (Ubuntu 20.04 LTS[26]). Muutmälu on 4 GB ja kettamaht 31 GB.

Kontroll süsteem jookseb atmos.to.ee-s taustaprotsessina ööpäevaringselt, kuid instrumentide tööd kontrollitakse vaid etteantud aegadel. Mõlema instrumendi puhul on kontrollimissagedus kirjeldatud allpool (peatükid 4.2 ja 4.3).

4.2 Monitoorimissüsteem Grafana baasil

Töö teostamisel kasutati Grafana versiooni v6.7.2, mis on installeeritud monitoorimissüsteemi jaoks loodud virtuaalarvutis. Grafana keskkonda on võimalik siseneda Tartu Ülikooli arvutivõrgus või väljastpoolt kasutades Tartu Ülikooli VPN ühendust. Programm on kaitstud kasutajanime ja parooliga.

Grafanaga seoti spektroradiomeetriasüsteemi MySQL andmebaas ja loodi instrumendi jälgimiseks töölaud. Töölaud kuvab tabelit viimaste mõõdetud spektrite identifikaatoreid (id numbreid) koos mõõtmise ajatempliga ja graafikuid instrumendi termokasti sisetemperatuuri ja kiiritustiheduse kohta lainepikkusel 306 nm (Joonis 7). Need väärtused võimaldavad aparatuuri töökorda kõige operatiivsemalt hinnata. Temperatuur on oluline mõõtmistulemuste kvaliteedi hindamisel. Kiiritustihedus lainepikkusel 306 nm on oluline seetõttu, et samal lainepikkusel mõõdab kiiritustihedust ka Ilmateenistus Tõravere meteoroloogiajaam

[23] https://github.com/grafana/grafana/releases/tag/v1.0
[24] https://nodejs.org/en/about/
[25] https://nodejs.dev/
[26] https://ubuntu.com/

UV-radiomeetriga CUVB1 (Kipp & Zonen, Holland[27]). Selle instrumendi kogutavad mõõtmistulemused on jooksvalt kasutusel TO radiomeetriasüsteemi stabiilsuse hindamiseks. Kõiki loetletud suurusi kuvatakse viimase 7 päeva kohta (identifikaatornumbri tabelis ei ole kõik väärtused korraga nähtavad).

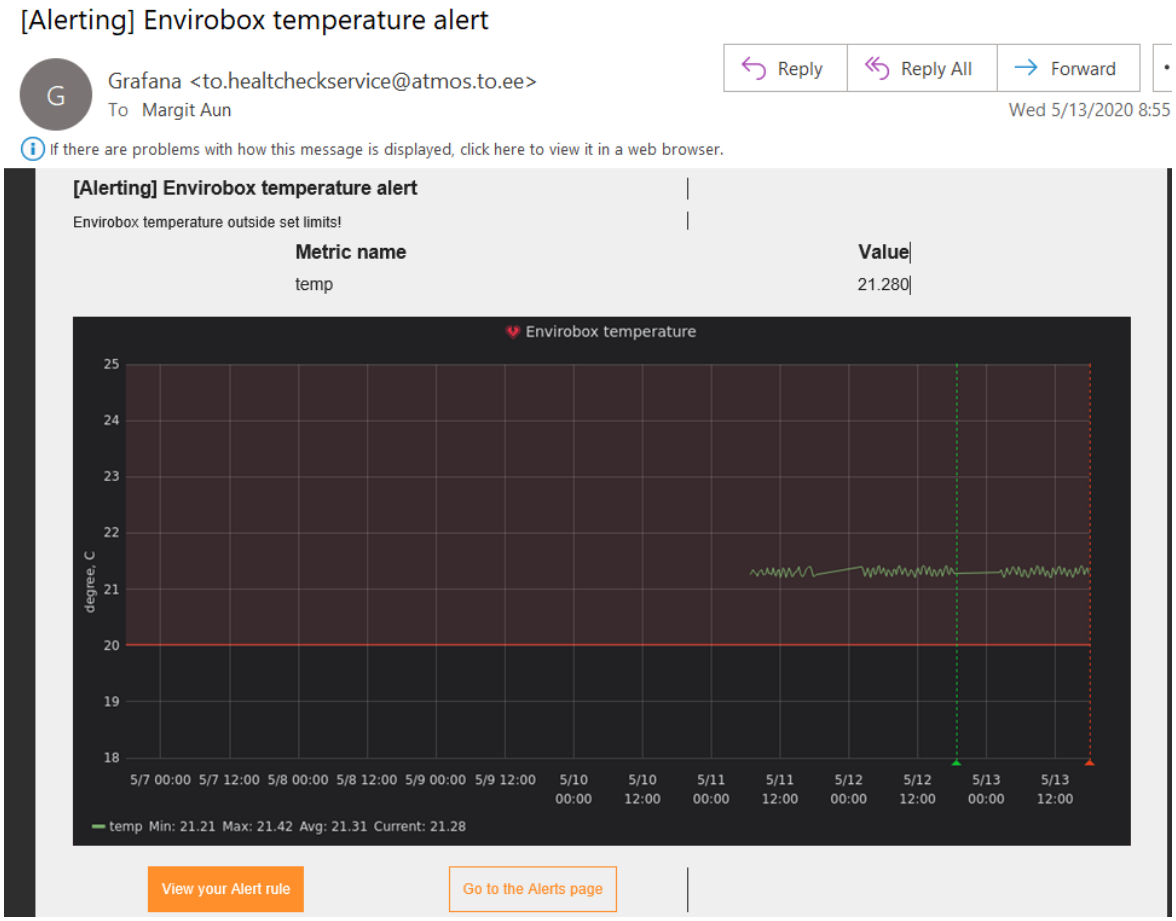
Joonis 7. Grafana töölaud Benthami jaoks. Vasakul üleval tabel viimaste mõõtmiste identifikaatornumbrite ja kuupäeva ning kellaajaga, üleval paremal kiiritustiheduse väärtused lainepikkusel 306 nm ja all termokasti Envirobox sisetemperatuuri graafik.

Grafana rakenduses seadistati ka teavitussüsteem. Teavituste saatjaks on Grafana töökeksond to.healtcheckservice@atmos.to.ee.

Monitoorimissüsteem registreerib termokasti sisetemperatuuri 15-minutilise intervalliga ja arvutab viimase 5 minuti keskmise temperatuuri, mis peab olema vahemikus 18 – 24 °C. Teavitus saadetakse, kui süsteemi staatus muutub olekust „OK“ olekusse „Alerting“ või vastupidi. Süsteem saab staatuse „Alerting“ juhul kui 5 min keskmine temperatuur on väljaspool eelnimetatud piirkonda. Andmete puudumisel (nt. öösel, kui mõõtmisi ei toimu) säilitab süsteem viimase oleku ja teavitust ei saadeta. Kuna Grafanas ei ole sisseehitatud kellajast sõltuvat teavitussüsteemi ei sobi see observatooriumis käivate radiomeetriliste mõõtmiste kontrollimiseks.

Teavituse testimiseks seati temperatuuri ülapiiriks 20 °C, saadud teavitus sisaldab tekkinud vea kirjeldust ja graafikut (Joonis 8).

[27] https://www.kippzonen.com/



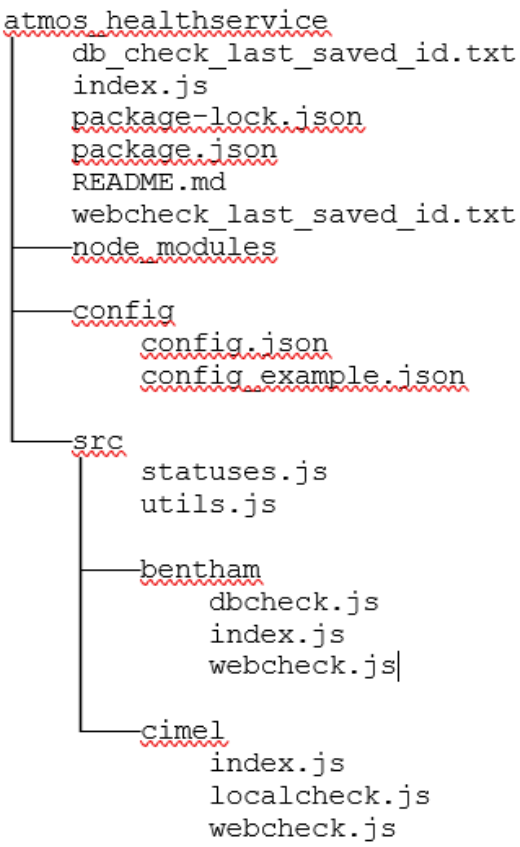
Joonis 8. Grafana teavituse näidis.

4.3 Monitoorimissüsteem Javascript Node.js baasil

Monitoorimissüsteemi arendustöö toimus läbi selleks loodud GitHub-i[28] privaatse hoidla ja töö avaldamise hetkel on Node.js versiooniks v12.16.2. Töös kasutatud Node.js paketid on kirjeldatud failis „package.json“ (Lisa II). Kontrollsüsteemi paigaldamiseks on vaja kloonida GitHub-i hoidla ja installeerida projektis kasutatavad moodulid käsu „npm install“ abil. Paketid salvestatakse kausta „node_modules“, mis asub projekti peakaustas.

Projekti peakaust „atmos_healthservice“ on jaotatud osadeks, nagu näidatud Joonis 9. Monitoorimisprogrammi käivitamine toimub failist „index.js“ (Lisa III), samuti on peakaustast failid „packages.json“, „package-lock.json“, ja „README.md“ (Lisa IV). Spektroradiomeetriliste mõõtmiste kontrollimiseks on abifailid, millesse salvestatakse viimase kontrollimise käigus tuvastatud viimase salvestatud spektri identifikaator. Andmebaasist ja veebilehelt tehtud kontrolli käigus leitud identifikaatornumbrid asuvad eraldi failides.

[28] https://github.com/



Joonis 9. Node.js abil loodud kontrollsüsteemi failipuu.

Kaustas „config“ asub fail „config.json“, milles hoitakse andmebaasi, CIMEL-i arvuti ja e-kirjade kasutamiseks vajalikku konfigureerimisinformatsiooni ja teavituste adressaatide nimekirja. GitHub-i projekti on turvalisuse huvides kaastatud vaid selle faili näidis „config_example.json“ ilma reaalsete andmeteta (Lisa V). See näidisfaili on oluline vorminduse defineerimiseks.

Kontrollsüsteemi tööfailid asuvad kausta „src“ (Joonis 9) alamkaustades, mis on nimetatud vastavalt instrumentidele „bentham“ ja „cimel“. Lisaks asuvad seal instrumentide ülesed failid: „statuses.js“, mis kirjeldab kasutuses olevaid staatuse väärtusi („ok“, „alert“) ja „utils.js“, mis sisaldab abifunktsioone (Lisa VI).

Kontrollsüsteemi poolt välja saadetud teavitused on inglisekeelsed, samuti on inglise keeles lähtekoodi kommentaarid. Inglise keele kasutamine on möödapääsmatu teadusasutuse töökorralduse tõttu.

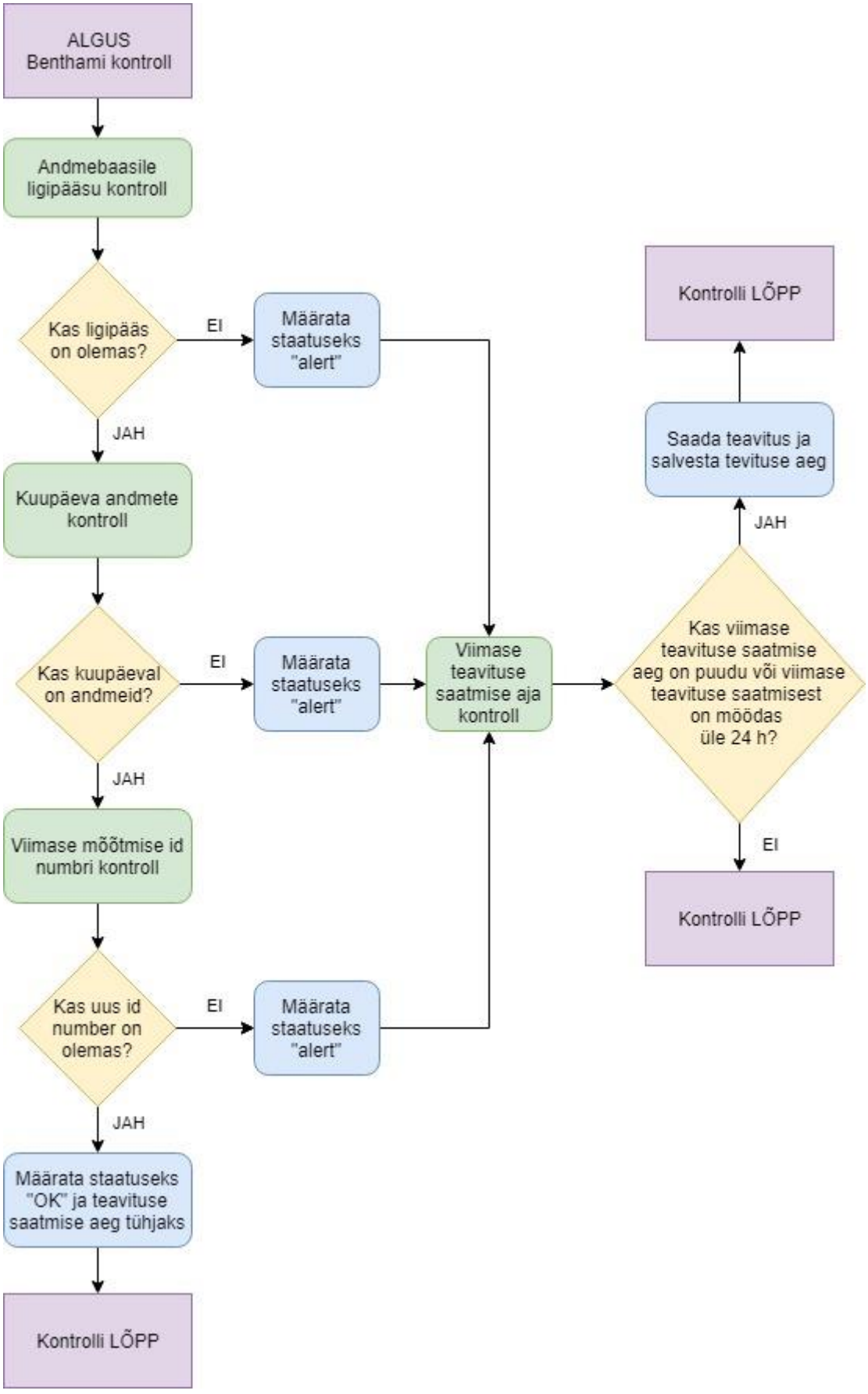
Spektroradiomeetriasüsteemi Bentham kontroll

Peamised funktsioonid, mille abil kontrollitakse Bentham DMc150F-U tööd, asuvad kausta „bentham“ failides (Joonis 9): „dbcheck.js“ andmebaasi kontrollimiseks (Lisa VII), „webcheck.js“ veebilehe kaudu andmete kontrollimiseks (Lisa VIII) ja „index.js“, mis määrab kontrolli algoritmi (Lisa IX). Kontrollimine toimub iga tunni 5., 20., 35. ja 50. minutil, kuid vaid juhul kui päikesetõusust on möödunud vähemalt 30 minutit peale ja päikeseloojanguni on enam kui 30 minutit.

Benthami töövõime kontrollimine toimub kahest allikast. Neist olulisem on andmebaas, mis asub TO serveris. Kontrollitakse kahte aspekti: ligipääsu andmebaasile ja andmete olemasolu. Viimane kontroll on omakorda jaotatud kaheks: tuvastatakse, kas kontrolli teostamise kuupäeval andmeid on ja kas viimasest kontrollimisest alates on uusi andmeid tekkinud. Selleks salvestatakse viimati registreeritud identifikaatornumber faili „db_check_last_saved_id.txt“.

Seadme normaalse toimimise korral on süsteemi staatuseks „ok“ ja teavituse saatmise kontrollajaks määratakse tühi väärtus (ingl *null*). Kui ligipääs andmebaasile puudub, antud kuupäeval andmeid ei ole või eelmisest kontrollist alates ei ole uut mõõtmist toimunud, määratakse süsteemi staatuseks „alert“. Veateate saamisel kontrollitakse, kas eelnev staatus oli „ok“ (teavituse saatmise kontrollaeg on tühi) või kas viimasest teavitusest on möödas rohkem kui 24 tundi. Kummagi kriteeriumi täitmisel saadetakse määratud e-kirja adressaatidele teavitus. 24 tunni kriteerium on kasutusel selleks, et mitte saata teavitust iga 15 minuti järel, kuna probleemide kõrvaldamine võib võtta aega näiteks juhul kui tegu on riistvaralise veaga ja/või vastutav insener viibib eemal. Kontrollalgoritmi plokkskeem on toodud Joonis 10.

Teavitus saadetakse välja ka siis, kui eelnevalt „alert“-staatuses olnud süsteem saab staatuseks „ok“. Sellisel juhul antakse teada, et probleem on lahendatud. Selle abil saab informatsiooni kiiresti edastada ka isikutele, kes vajavad süsteemi olekust teavitamist, aga ei ole seotud probleemide kõrvaldamise kohustusega.



Joonis 10. Benthami andmebaasi kontrollsüsteemi tööpõhimõte.

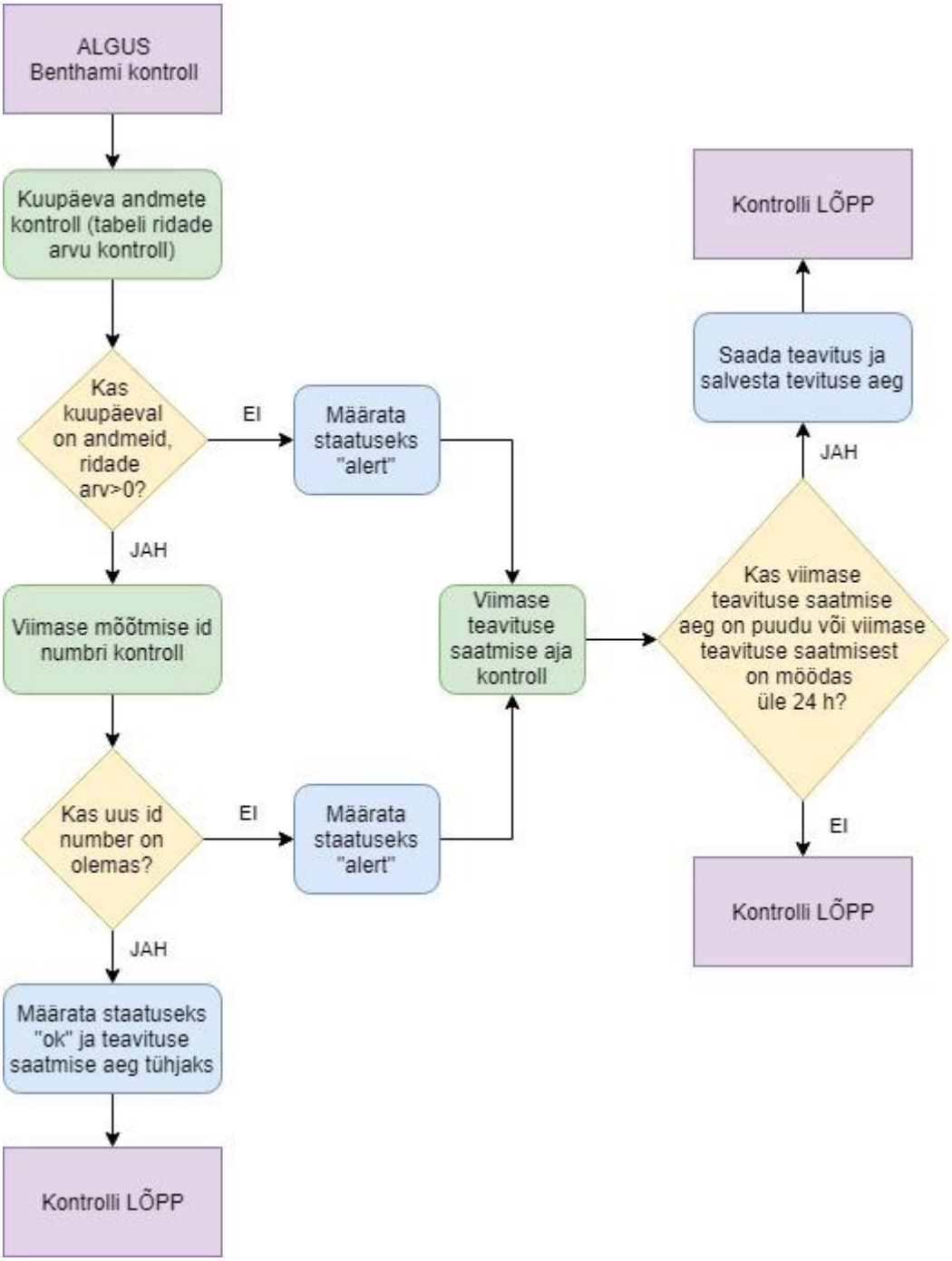
Saadetud teavituse teema ja sisu sõltuvad teavituse saatmise põhjusest (Tabel 1).

Tabel 1. Benthami kontrollsüsteemi poolt saadetud teavitused.

Põhjus	Teema	Sisu
Juurdepääs andmebaasile puudub	Bentham database error message	No database access!
Antud kuupäeval andmed puuduvad	Bentham database error message	No data today: [kuupäev]. Last data date: [viimane kuupäev]
Uus mõõtmine puudu	Bentham database error message	No new data available! Last measurement id [id]
Häire pole 24 h jooksul lahendatud	Bentham database error message - issue not resolved	Alert not solved in 24h! Resending alert.
Probleem lahendatud	Bentham database error message - FIXED	Data available again, error solved!

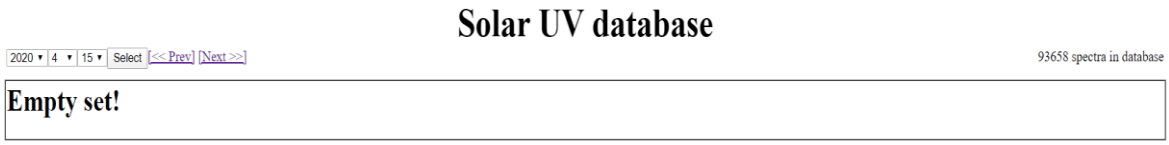
Benthami töövõime kontrollimise teiseks allikaks on radiomeetriasüsteemi veebileht (Joonis 3). Kontroll veebilehelt on vajalik, kui kaob ligipääs spektrite andmebaasile. See kontrollmeetod on võimalik ka siis, kui kasutajal puuduvad andmebaasi ligipääsuõigused või soovitakse kontrolli teostada väljaspool Tartu Ülikooli arvutivõrku. Sellisel juhul on võimalik programm seadista mis tahes laua- või sülearvutis.

Spektriandmete kontrollimine veebilehelt toimub lehe vasakul servas oleva tabeli abil (Joonis 3), milles on muuhulgas toodud mõõtmise identifikaator (id). Kontrollimise põhimõte on sarnane andmebaasi meetodile (v.a. ligipääsu kontroll): tuvastatakse, kas antud kuupäeval on mõõtmisi toimunud ja kas alates viimasest kontrollimisest on uus mõõtmine lisandunud (Joonis 11).



Joonis 11. Benthami veebilehe kontrollsüsteemi põhimõte.

Esmalt kontrollitakse tabeli ridade arvu. Juhul kui kontrollimise päeval ei ole veel ühtegi mõõtmist toimunud, antud tabel puudub ja ridade arv on 0 (Joonis 12).



Joonis 12. TO spektroradiomeetrilise mõõtesüsteemi kodulehe kuvatõmmis päevast (15.04.2020), kus mõõtmised puuduvad.

Kui jooksva kuupäeva kohta on veebilehel andmed olemas, loetakse eelpool nimetatud tabelist viimase rea identifikaatornumber ja võrreldakse seda eelmise kontrollimise käigus salvestatud numbriga failis „webcheck_last_saved_id.txt“. Kui faili salvestatud ja veebilehelt loetud viimane number erinevad, on uued andmed olemas ja viimaselt tabeli realt loetud identifikaatornumber salvestatakse faili, asendades eelmise salvestatud numbri. Kui faili salvestatud ja veebilehelt loetud viimane identifikaatornumber kattuvad, pole uusi mõõtmistulemusi lisandunud.

Teavituse saatmise põhimõte on sama, mis andmebaasi kontrollimise puhul: kui süsteemi staatus on „alert“, kontrollitakse teavituse saatmise aega, kui see puudub (st. eelnevalt oli staatus „ok“) või viimasest teavitusest on möödas rohkem kui 24 tundi, saadetakse adressaatidele teavitus. Teavitus saadetakse ka siis, kui eelnevalt „alert“-staatuses olnud süsteem saab staatuseks „ok“. Sellisel juhul antakse teada, et probleem on lahendatud.

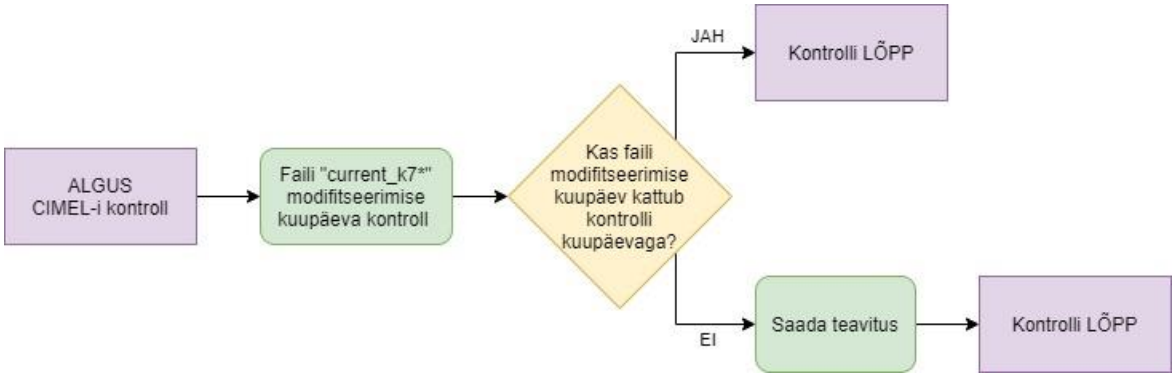
Veebilehe kontrollimine ei ole aktiivses kasutuses, kuid selle sisselülitamine on võimalik kutsudes failis „../bentham/index.js“ välja funktsiooni webData failist „webcheck.js“.

Päikesefotomeetri CIMEL kontrollimine

Instrumendi CIMEL CE318 tööd kontrollitakse kaustas „cimel“ asuvate failide abil: „localcheck“ CIMEL’i arvutist andmete kontrollimiseks (Lisa X), „webcheck.js“ veebilehe kaudu andmete kontrollimiseks (Lisa XI) ja „index.js“, mis määrab kontrollalgoritmi (Lisa XII).

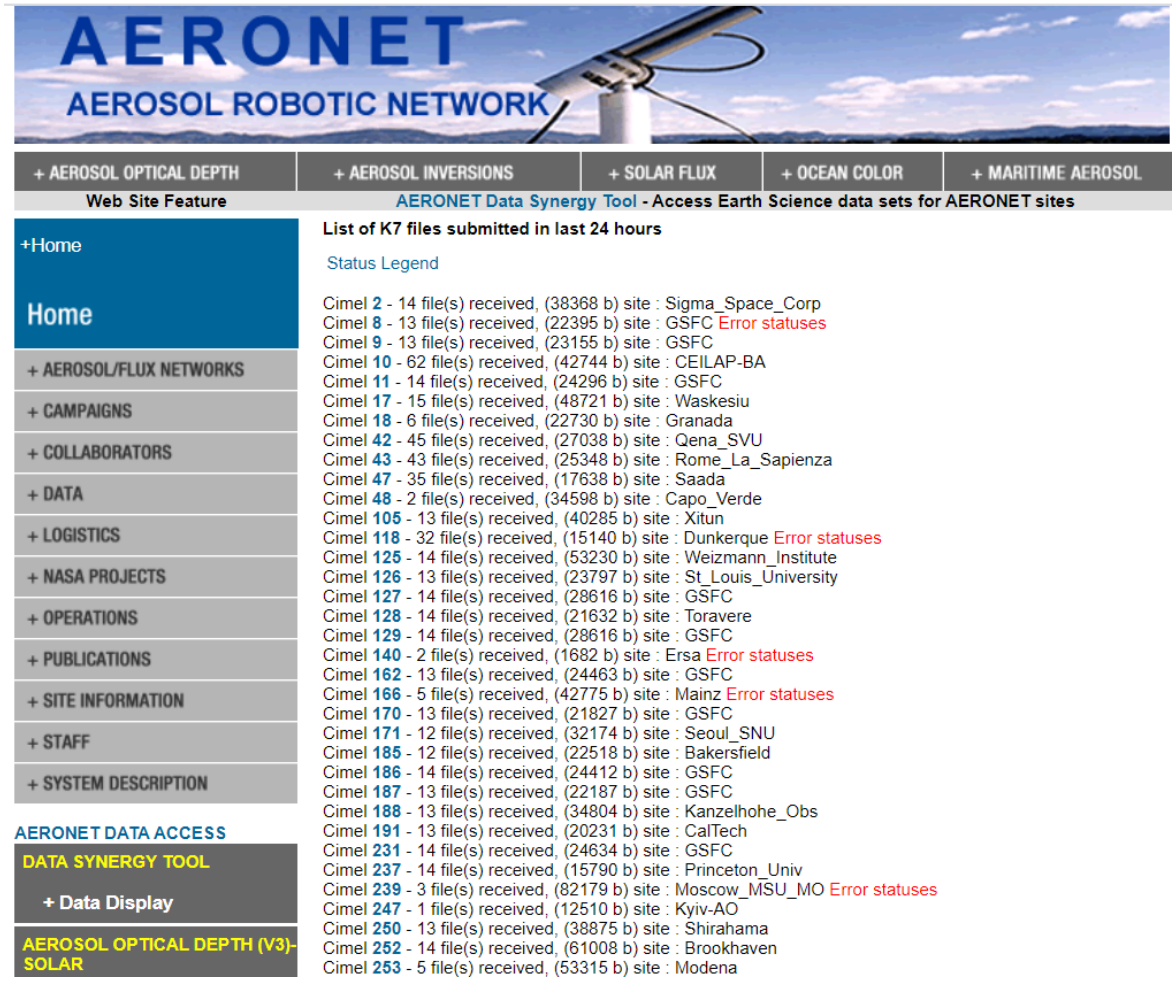
CIMEL-i kontrollimiseks on kasutusel kaks allikat – logiv arvuti ning AERONET-i koduleht.

Esmalt toimub instrumendi töö kontrollimine otse fotomeetriga ühenduses oleva arvuti kaudu. Ligipääsuks arvutile kasutatakse SSH-d. Kontrollsüsteem logib arvutisse ja otsib sealt faili „current_k7*“, millesse salvestatakse jooksvalt mõõtmistulemused. Seejärel võrreldakse faili modifitseerimis kuupäeva kontrollimise kuupäevaga. Kui kuupäev kattub, tähendab see, et sellel päeval on mõõtmisi toimunud ja süsteem töötab. Kui kuupäev erineb, saadetakse adressaatidele teavitus teemakirjega "Cimel PC error message" ja sisuga "Current_k7 fail not updating!" (Joonis 13).



Joonis 13. CIMEL-i andmete kontrolli põhimõte juhtarvutist.

Et olla kindel mõõtmisandmete edastamises, toimub teisene kontrollimine AERONET-i veebilehelt[29]. Veebileht näitab viimase 24 tunni jooksul edastatud andmeid kõikidest võrgustikku kuuluvatest mõõtejaamadest (Joonis 14).



Joonis 14. Veebilehe[29], millelt toimub CIMEL andmete kontroll, kuvatõmmis. (19.04.2020).

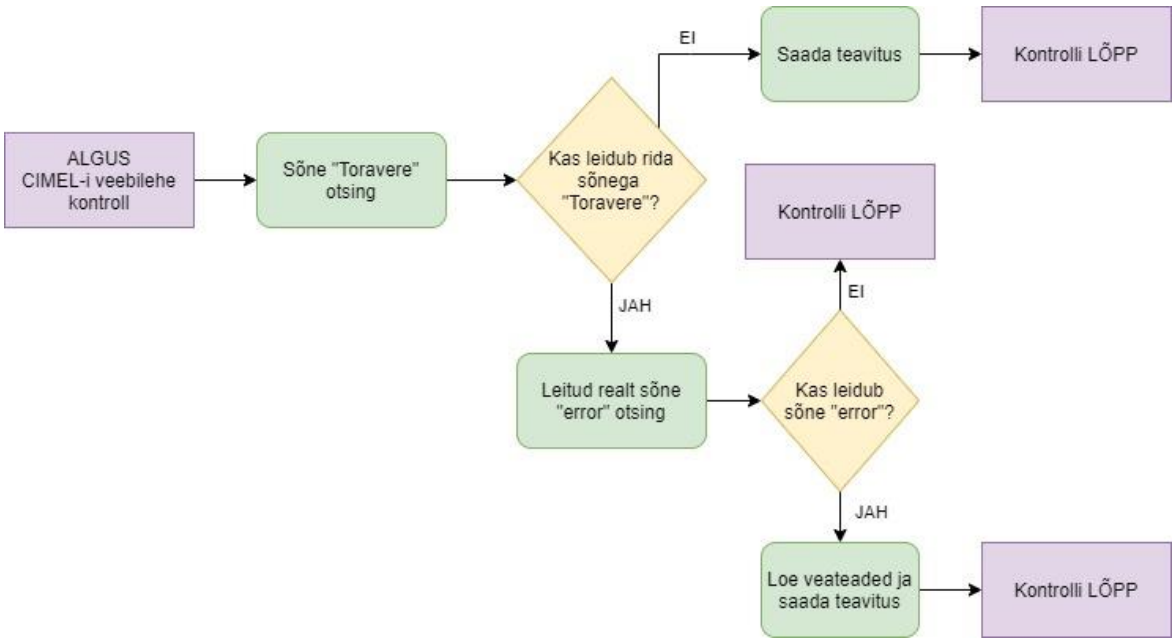
[29] https://aeronet.gsfc.nasa.gov/new_web/Data_Transfer_Logs/K7/last_1440_minutes.html

Kontrollsüsteem otsib parempoolsest nimekirjast rea, kus on sõne „Toravere“. Otsida saaks ka instrumendi numbri järgi, aga see muutub iga-aastaselt radiomeetrite kalibreerimise ja rotatsiooni tõttu. Töö kirjutamise hetkel mõõdab Tõraveres CIMEL CE318 nr 128. Kui rida sõnega „Toravere“ puudub, ei ole andmeid edastatud ja süsteemi staatuseks määratakse „alert“, saadetava sõnumi sisuks on "Data not commited!". Kui vastav rida leitakse, otsitakse sealt sõna „error“, mis tähendab, et edastatud failide seas oli veateatega mõõtmisi. AERONET-is on kasutusel on viis erinevat veateadet, millest neli on kriitilised (Tabel 2). Mittekriitiline veateade puudutab õhuniiskuse sensorit. Vähemalt ühe kriitilise veateate leidmisel määratakse süsteemi staatuseks „alert“ ja sõnumisse kirjutakse mõõtmise aeg koos veateatega.

Tabel 2. CIMEL-i veateated ja nende tähendused.

Veateade	**Vea sisu**
stab	Roboti viga
star	Lähtestamise viga
stap	Sensori pea ei reageeri kontrollile
stas	Filtri ketta viga
*stah	Niiskus sensor aktiviseeritud

Kui kontrollimistsükli lõpuks on süsteemi staatus „alert“, saadetakse adressaatidele teavitus teemakirjega „Cimel web error message“ ja veateate sisuga. Veebilehe kontrollimise põhimõte on kirjeldatud Joonis 15.



Joonis 15. CIMEL-i monitoorimissüsteemi tööpõhimõte.

Mõlema meetodi puhul käivitatakse kontrollimine üks kord päevas kell 08.00 UTC aja järgi. See annab operaatorile tööpäeva jooksul aega tekkinud probleemidega tegeleda.

4.4 Kontrollmeetodite võrdlus

Vabavaraline lahendus Grafana osutus seatud eesmärgi saavutamiseks ebasobivaks. Kuigi eelneva uurimistöö põhjal võis eeldada, et Grafana suudab etteantud ülesannet täita, tekkis mitmeid probleeme.

Suurimaks probleemiks osutus spektroradiomeetriliste mõõtmiste ebaregulaarsus ööpäeva lõikes. Testimisel selgus, et Grafana teavitussüsteemi ei ole sisseehitatud kontrolli kellajalist sõltuvust. Seda võimalust on kasutajate poolt küsitud ja tootja poolt ka plaani võetud, kuid seni on probleem lahendamata[30]. Oleks võimalus on viia ajakriteerium MySQLpäringusse[31]., kuid see on keeruline, eriti TO mõõtmiste puhul, kus mõõtmiste alguse ja lõpu kellaajad on pidevas muutumises. Sellise võimaluse rakendamine ei tundunud otstarbekas.

Teine probleem tekkis CIMEL-i andmete liitmine Grafanasse. Lihtsat meetodit CIMEL-i kontrollimiseks ei leitud. Kaudseks lahenduseks oleks olnud luua eraldi andmebaas CIMEL-i andmete jaoks, kuid seda ei peetud otstarbekaks.

Vaatamata loetletud puudustele on Grafanal positiivseid omadusi. Andmebaasi liides loob väga head ja hõlpsalt kasutatavad võimalused ülevaate saamiseks mõõtmisandmetest – võimalus on koostada erinevaid päringuid, muuta tulemuste esitlusviisi ja ajaskaalat, mis on andmete esmase hindamise jaoks kasulikud. Nagu temperatuuri jaoks, on teavitussüsteem võimalik üles seada ka erinevate kiiritustiheduste väärtuste jaoks. Kiiritustiheduste suure muutlikkuse tõttu on see aga aeganõudev ülesanne. Võimalik oleks leida ekstreemseid väärtusi, kuid parimate tulemuste saamiseks nõuab see suurt eeltööd, et vastavad vahemikud sõltuvalt aastaajast paika panna. Samuti on Grafanasse hiljem võimalik lisada instrumente, mille mõõtmised toimuvad ööpäevaringselt ühtlase intervalliga.

Eeltöö ja tulemus näitavad, et keskkonnamõõtmised ja seda teostavad instrumendid võivad olla väga erinevad ja teistele töörühmadele sobivat rakendust ei ole alati võimalik kohaldada enda vajadustele.

Node.js-i abil loodud lahendus osutus püstitatud ülesande täitmiseks optimaalseks. Erinevalt valmislahendusest oli võimalik luua just selline programm, nagu hetkeülesanded nõuavad. Andmete allikaiks saavad olla nii andmebaasid, arvutid kui ka veebilehed, on võimalik seadistada mõõtmistele vastav kontrolltsükkel ja testida instrumenti iseloomustavaid parameetreid ja struktuure (ligipääs andmebaasidele, andmefailide olemasolu, uute mõõtmiste toimumine, veateadete sisu). Sellesse süsteemi on lihtne lisada ka uusi seadmeid ilma, et muutuks midagi teiste seadmete monitoorimises. Programmi ülesehitus teeb selle kergesti jälgitavaks ning muudetavaks.

Algse ülesande täitis Node.js abil loodud lahendus, kuid oma lisaväärtusse andis ka Grafana. Mõlemad süsteemid on jooksvalt töös.

[30] https://grafana.com/docs/grafana/latest/alerting/rules/
[31] https://github.com/grafana/grafana/issues/6592

5. Kokkuvõte ja järeldused

Magistritöö raames loodi Tartu Ülikooli Tartu observatooriumi atmosfääriseire töörühmale erinevate keskkonnategurite pikaajalisi automatiseerituid mõõtmisi teostavate instrumentide monitoorimissüsteem. Süsteemi on kaasatud kaks instrumenti – UV-kiirgust mõõtev spektromeeter Bentham DMc150F-U ja atmosfääri aerosooli sisaldust mõõtev päikese fotomeeter CIMEL CE318, mis kuulub NASA AERONET võrgustikku.

Süsteemi asub Tartu observatooriumi arvutivõrgus selleks loodud virtuaalses Linuxil põhinevas arvutis. Ligipääs sellele arvutile on kasutaja põhine ja kaitstud parooliga.

Monitoorimissüsteemi esmane eesmärk on mõõtmiste katkemisel saata adressaatidele e-kirja teel teavitus. Eesmärk on tekkinud aastate pikkusest kogemusest erinevate pikaajaliste mõõtmistega ja nende soovimatu katkemisega. Seda eesmärki täidab nüüd töö käigus valminud Node.js programm. Programmi tööpõhimõte on kokkuvõttes lihtne: kontrollida, kas uusi mõõtmisandmeid on salvestatud ja probleemide korral teavituste saatmine.

Eesmärki prooviti saavutada kahte erinevat meetodit kasutades. Esiteks katsetati vabavaralist Grafana tarkvara, mis osutus töörühma mõõtmiste monitoorimiseks siiski ebasobivaks. Peamiseks probleemiks oli mõõtmiste ebaühtlane jaotus ööpäeva jooksul. Keerukaks osutus ka päikesefotomeetri kontrollimine Grafana abil. Grafanas on olemas aga väga head võimalused andmete graafiliseks esitamiseks ning kiire visuaalse ülevaate saamiseks. Seda näitas spektromeetri andmebaasi sidumine Grafanaga ja testimiseks loodud töölaud.

Teiseks kirjutati originaalprogramm Javascript Node.js abil. Loodud programm vastab püstitatud eesmärgile ja selle positiivseteks külgedeks on suur paindlikus toimimisloogika ning sisendite ja väljundite seadistamisel. On võimalik lisada uusi tingimusi, millele andmed peavad vastama. Monitoorimissüsteemi on samuti lihtne lisada uusi instrumente. Kontrollalgoritmi sisendeiks sobivad nii MySQL andmebaasid, failid kui ka veebilehed. Keskkonnategurite mõõtmisel kasutatavad instrumendid on väga erinevad ja tihti on nende andmete salvestus ning edastamine määratud instrumendi tootja või operaatori poolt, nagu näiteks CIMEL-i puhul.

Kuna kõik võimalike mõõteriistu on palju, on olemas ka väga palju erinevaid vabalt kättesaadavaid valmis lahendusi süsteemide ja andmete monitoorimiseks. Seetõttu oli käesoleva töö esmaseks faasiks nendega tutvumine. Sobivaima lahenduse välja valimine on siiski keeruline protsess. Kõikide erinevate programmide katsetamine on aeganõudev ja ebamõistlik. Valikut on võimalik kitsendada, tuginedes erinevatele arvamusartiklitele, blogidele, programmide kodulehtedele ning kolleegide ja IT-spetsialistide kogemustele. Nii nagu igal arvutiprogrammil on oma, teistest veidi erinev eesmärk, on ka igal kasutajal oma spetsiifilised vajadused. Kuigi eelnevale tuginedes valiti välja Grafana tarkvara, ei välista autor, et saadaval on püstitatud eesmärki paremini täitev programm. Käesoleval hetkel pole uue programmi otsimine aktuaalne, kuna püstitatud eesmärgid on saavutatud.

Töö tulemusena monitoorib TO atmosfääriseire töörühma instrumentide tööd Javascript Node.js abil loodud programm. Saavutatud on ka esimesed positiivsed tulemused, kus spektroradiomeetria süsteemi töö katkemine on avastatud operatiivselt just tänu sellele programmile. Kasutusse jäi ka Grafana platvormil põhinev lahendus, kuigi tema eesmärk ei vasta algsele ülesandepüstitusele.

Edaspidi on plaanis paremini ära kasutada Grafana poolt pakutavaid võimalusi. See aga nõuab eelnevat põhjaliku tööd ja testimist. Samuti võiks järgmiste sammudena täiendada Node.js süsteemi tööd. Võimalus on monitoorida andmebaasi kirjete sisu sarnaselt Grafanas teostatud termokasti temperatuuri kontrollimisega. Samuti on tulevikus võimalik lisada

CIMEL'i andmete põhjalikum kontroll koos andmefailide sisu tõlgendamisega. Selle vastu on huvi üles näidanud Tartu observatooriumis töötavad astronoomid, kes saaksid neid andmeid kasutada vaatluste planeerimisel. Sellisel juhul aga ei piisa kord päevas toimuvast kontrollist.

6. Viidatud kirjandus

Abbasy, S., Abbasi, M., Asgari, J., ja Ghods, A. (2017). Precipitable water vapour estimation using the permanent single GPS station in Zanjan, Iran. *Meteorological Applications*, *24*(3), 415–422. https://doi.org/10.1002/met.1639

Aun, M. (2017). *Dependence of UV radiation on climate factors. Reconstruction of UV doses in Estonia for past years*. University of Tartu.

Devadithya, T., Chiu, K., Huffman, K., ja McMullen, D. F. (2005). The common instrument middleware architecture: Overview of goals and implementation. *First International Conference on E-Science and Grid Computing (e-Science'05)*, 8 pp. – 585. https://doi.org/10.1109/E-SCIENCE.2005.77

Foster, I., ja Kesselman, C. (Eds.). (2004). Editors. In *The Grid 2 (Second Edition)* (Second Edition, pp. vii–viii). Morgan Kaufmann. https://doi.org/10.1016/B978-155860933-4/50000-6

Holben, B. N., Eck, T. F., Slutsker, I., Tanré, D., Buis, J. P., Setzer, A., Vermote, E., Reagan, J. A., Kaufman, Y. J., Nakajima, T., Lavenu, F., Jankowiak, I., ja Smirnov, A. (1998). AERONET—A Federated Instrument Network and Data Archive for Aerosol Characterization. *Remote Sensing of Environment*, *66*(1), 1–16. https://doi.org/10.1016/S0034-4257(98)00031-5

Kerr, J. B. (2005). Understanding the factors that affect surface ultraviolet radiation. *Optical Engineering*, *44*, 44-44–49. https://doi.org/10.1117/1.1886817

Lamy, K., Portafaix, T., Brogniez, C., Godin-Beekmann, S., Bencherif, H., Morel, B., Pazmino, A., Metzger, J. M., Auriol, F., Deroo, C., Duflot, V., Goloub, P., ja Long, C. N. (2018). Ultraviolet radiation modelling from ground-based and satellite measurements on Reunion Island, southern tropics. *Atmospheric Chemistry and Physics*, *18*(1), 227–246. https://doi.org/10.5194/acp-18-227-2018

Loew, A., Bell, W., Brocca, L., Bulgin, C. E., Burdanowitz, J., Calbet, X., Donner, R. V., Ghent, D., Gruber, A., Kaminski, T., Kinzel, J., Klepp, C., Lambert, J.-C., Schaepman-Strub, G., Schröder, M., ja Verhoelst, T. (2017). Validation practices for satellite-based Earth observation data across communities. *Reviews of Geophysics*, *55*(3), 779–817. https://doi.org/10.1002/2017RG000562

Medhaug, I., Olseth, J. A., ja Reuder, J. (2009). UV radiation and skin cancer in Norway. *Journal of Photochemistry and Photobiology B: Biology*, *96*(3), 232–241.

Olds, W. J. (2010). *Elucidating the links between UV radiation and vitamin D synthesis: Using an in vitro model* [PhD Thesis, Queensland University of Technology]. https://eprints.qut.edu.au/32073/

Russak, V., ja Kallis, A. (2003). *Eesti kiirguskliima teatmik*. Eesti Meteoroloogia ja Hüdroloogia Instituut. file:///C:/Users/kasutaja/Downloads/eesti_kiirguskliima_teatmik.pdf

Staehelin, J., Vogler, C., ja Brönnimann, S. (2009). The Long History of Ozone Measurements: Climatological Information Derived from Long Ozone Records. In C. Zerefos, G. Contopoulos, ja G. Skalkeas (Eds.), *Twenty Years of Ozone Decline* (pp. 119–131). Springer Netherlands.

UNEP. (1998). *Environmental effects of ozone depletion: 1998 assessment* (p. 193).

Veismann, U., ja Eerme, K. (2011). *Päikese ultraviolettkiirgus ja atmosfääriosoon*. Ilmamaa.

Verhoelst, T., Granville, J., Hendrick, F., Köhler, U., Lerot, C., Pommereau, J.-P., Redondas, A., Van Roozendael, M., ja Lambert, J.-C. (2015). Metrology of ground-based satellite validation: Co-location mismatch and smoothing issues of total ozone comparisons. *Atmospheric Measurement Techniques*, *8*(12), 5039–5062. https://doi.org/10.5194/amt-8-5039-2015

Volkova, K. A., Poberovsky, A. V., Timofeev, Yu. M., Ionov, D. V., Holben, B. N., Smirnov, A., ja Slutsker, I. (2018). Aerosol Optical Characteristics Retrieved from CIMEL Sun Photometer Measurements (AERONET) near St. Petersburg. *Atmospheric and Oceanic Optics*, *31*(6), 635–641. https://doi.org/10.1134/S1024856018060180

Lisad

I. Lühendid

AERONET - Aerosoolide Robootiline Võrgustik (ingl *AErosol RObotic NETwork*)

AOD – aerosoolide optiline paksus (ingl *Aerosol Optical Depth*)

NASA - Ameerika Ühendriikide Riiklikule Lennundus-ja Kosmoseagentuur (ingl *National Aeronautics and Space Administration*)

UV-kiirgus – ultraviolettkiirgus

VPN – virtuaalne privaatvõrk (ingl *Virtual Private Network*)

II. package.json

```
{
  "name": "nodejs",
  "version": "1.0.0",
  "description": "",
  "main": "index.js",
  "dependencies": {
    "cheerio": "^1.0.0-rc.3",
    "csv-parser": "^2.3.2",
    "mysql": "^2.18.1",
    "node-cron": "^2.0.3",
    "nodemailer": "^6.4.2",
    "remote-exec": "0.0.3",
    "request": "^2.88.0",
    "request-promise": "^4.2.5",
    "simple-ssh": "^1.0.0",
    "ssh2": "^0.8.9",
    "sunrise-sunset-js": "^2.1.1",
    "system-sleep": "^1.3.6"
  },
  "devDependencies": {},
  "scripts": {
    "test": "echo \"Error: no test specified\" && exit 1"
  },
  "keywords": [],
  "author": "",
  "license": "ISC"
}
```

III. Käivitav index.js

```
/* Following code starts the monitoring system*/

const bentham = require('./src/bentham');
const cimel = require('./src/cimel');

function start() {
  bentham.check-
Bentham("5,20,35,50 * * * *") //process runs every hour at 5, 20, 35 and 50
 minutes
  cimel.checkCimel('00 8 * * *') //process runs every morning at 08.00 lo-
cal time
}

start();
```

IV. README.md

Atmos_healthservice

The program is designed to check the availability of data from the instruments of Tartu University Tartu Observatory group of remote sensing of the atmosphere. Two instruments are included, Bentham DMc150F-U and CIMEL CE-318. Bentham is checked in every 15 minutes from sunrise + 30 min to sunset - 30 min. CIMEL is checked once a day at 08:00 UTC.

Getting Started

These instructions will get you a copy of the project up and running.

Prerequisites

Install node.js version 12.

Installing

For installing the dependencies execute the following command in project root directory:

```
npm install
```

Configuration

For creating configuration file make a copy from **config/config_example.json** and save to **config/config.json**. Fill all the fields.

Check times can be changed in **index.js**.

Starting the program

Start program by executing the following command in project root directory:

```
node index.js
```

Author

Margit Aun

V. config_example.json

```
{
  "mail": {
    "user": "to.healthcheckservice@gmail.com",
    "pass": ""
  },
  "db": {
    "user": "margit",
    "password": "",
    "host": "***",
    "database": "bentham_uv"
  },
  "mail_to": "margit.aun@ut.ee",
  "aeronet": {
    "host": "",
    "port": 22,
    "username": "",
    "password": ""
  }
}
```

VI. utils.js

```
/*File consist helping functions for CIMEL and Bentham checks*/

const nodemailer = require("nodemailer");
const fs = require("fs").promises;
const path = require("path");
const sleep = require("system-sleep");
const config = require("./../config/config.json");
const statuses = require("./statuses");

function sendMail(subject, message) {
  // Send e-mail through gmail with given subject and message.

  var transporter = nodemailer.createTransport({
    service: "gmail",
    auth: config.mail,
  });

  var mailOptions = {
    from: config.mail.user,
    to: config.mail_to,
    subject: subject,
    text: message,
  };

  transporter.sendMail(mailOptions, function (error, info) {
    if (error) {
      console.log(error);
    } else {
      console.log("Email sent: " + info.response);
    }
  });
}

function emptyFolder(directory) {
  // Delete folder content.

  console.log("Starting cleanup: " + directory);
  fs.readdir(directory, (err, files) => {
    if (err) throw err;

    for (const file of files) {
      fs.unlink(path.join(directory, file), (err) => {
        if (err) throw err;
      });
    }
  });
```

```
}

function compareDates(fileDate, currentDate) {
  if ((currentDate.getTime() - fileDate.get-
Time()) / (1000 * 3600 * 24) >= 2) {
    sendMail("ARONTET error", "Data missing! Last file date: " + fileDate);
    sleep(5000);
    process.exit();
  }
}

function getLatestFile(dirpath, filter) {
  // Finds latest file with given extension. If l

  let latest;
  const files = fs.readdirSync(dirpath);
  const filteredFiles = files.filter(
    (file) => path.extname(file).toLowerCase() === filter
  );

  filteredFiles.forEach((filename) => {
    // Get the stat
    const stat = fs.lstatSync(path.join(dirpath, filename));
    // Pass if it is a directory
    if (stat.isDirectory()) return;

    // latest default to first file
    if (!latest) {
      latest = { filename, mtime: stat.mtime };
    }
    // update latest if mtime is greater than the current latest
    if (stat.mtime > latest.mtime) {
      latest.filename = filename;
      latest.mtime = stat.mtime;
    }
  });
  const fileDate = new Date(latest.mtime);
  const currentDate = new Date();
  sleep(5000);
  compareDates(fileDate, currentDate);
  console.log(latest.mtime);
  console.log("-- found: ", latest.filename);
  return latest.filename;
}

function getFilterFile(dirpath, filter) {
  // Check if dirpath exist or not right here
  const files = fs.readdirSync(dirpath);
```

```
  const filteredFiles = files.filter(
    (file) => path.extname(file).toLowerCase() === filter
  );

  filteredFiles.forEach((filename) => {
    filteredFile = filename;
  });
  return filteredFile;
}

function getCurrentDate() {
  let date_ob = new Date();
  let date = ("0" + date_ob.getDate()).slice(-2);
  let month = ("0" + (date_ob.getMonth() + 1)).slice(-2);
  let year = date_ob.getFullYear();
  return [year, month, date];
}

// Find last saved measurement id (priorId) from file
async function getPriorId(fileName) {
  try {
    const data = await fs.readFile(fileName, "utf8");
    return parseInt(data);
  } catch (error) {
    console.log("getPriorId: File does not exist!: " + fileName);
    return 0;
  }
}

async function storeId(lastID, lastIdFile) {
  await fs.writeFile(lastIdFile, lastID, function (err) {
    if (err) throw err;
    console.log("Saved!");
  });
}

function hoursSinceTime(lastAlert) {
  return (new Date().getTime() - lastAlert) / (1000 * 60 * 60);
}

/* Function checks the status, if it is not "alert" sends a message,
   if last "alert" was sent over 24 h ago then sends a new message*/
function alertRecurrence(message, subject, lastAlert) {
  if (lastAlert === null) {
    sendMail(subject, message);
    lastAlert = new Date().getTime();
  } else if (hoursSinceTime(lastAlert) > 24) {
```

```
    console.log("Alert not solved in 24h! Resending alert.");
    sendMail(subject + " - issue not resolved", message);
    lastAlert = new Date().getTime();
  }
  return lastAlert;
}

async function idComparison(priorId, lastId, lastIdFile) {
  let status = statuses.OK;
  let message = "OK";
  if (priorId === lastId) {
    console.log("New data not available!");

    message = "No new data available! Last measurement id " + lastId;
    status = statuses.ALERT;
  } else {
    console.log("New data available!");
    await storeId(lastId, lastIdFile);
  }
  return {
    status,
    message,
  };
}

// From https://codeforgeek.com/how-to-check-date-is-today-javascript/
function isToday(dateParameter) {
  var today = new Date();
  return (
    dateParameter.getDate() === today.getDate() &&
    dateParameter.getMonth() === today.getMonth() &&
    dateParameter.getFullYear() === today.getFullYear()
  );
}

module.exports = {
  sendMail,
  emptyFolder,
  getLatestFile,
  getFilterFile,
  getCurrentDate,
  getPriorId,
  alertRecurrence,
  storeId,
  idComparison,
  isToday,
};
```

VII. Benthami andmebaasi kontroll, dbcheck.js

```
/*
The following code enables the check of data availabi-
lity of Bentham DMc150F-U on the roof of Tartu Observatory.
Checks are done through databas. Configurations for database ac-
cess are in ../config/config.json
*/
const mysql = require("mysql");
const util = require("util");

const statuses = require("../statuses");
const utils = require("../utils");
const config = require("../../config/config.json");

const subject = "Bentham database error message";
let alertTime = null;
let lastIdFile = "db_check_last_saved_id.txt";

function dbInit(config) {
  const connection = mysql.createConnection(config);
  return {
    query(sql, args) {
      return util.promisify(connection.query).call(connection, sql, args);
    },
    close() {
      return util.promisify(connection.end).call(connection);
    },
  };
}

//Function that checks data availability from Bentham database
async function dbData() {
  let message;
  let status = statuses.OK;

  console.log("checkBentham: dbdata");
  try {
    const db = dbInit(config.db);
    let result = null;

    try {
      result = await db.query("SELECT id, dt FROM L0 ORDER BY ID Desc LI-
MIT 1");
      console.log(result);
    } catch (err) {
      console.log("DB err" + err);
    } finally {
```

```
      await db.close();
    }

    console.log("checkBentham: dbdata, connection created");

    let lastDate = new Date(result[0].dt);
    let nowDate = new Date();

    //Check if any data for the current date is available
    if (
      lastDate.getFullYear() !== nowDate.getFullYear() ||
      lastDate.getMonth() !== nowDate.getMonth() ||
      lastDate.getDate() !== nowDate.getDate()
    ) {
      console.log("checkBentham: dbdata, no data");
      message = "No data today: " + now-
Date + ". Last data date: " + lastDate;
      status = statuses.ALERT;
    } else {
      const lastId = result[0].id;
      const priorId = await utils.getPriorId(lastIdFile);

      console.log("Last measurement id:" + lastId);
      console.log("priorID:" + priorId);

      const idCompResult = await utils.idComparison(
        priorId,
        lastId,
        lastIdFile
      );
      status = idCompResult.status;
      message = idCompResult.message;
    }
  } catch (error) {
    console.log(error);
    status = statuses.ALERT;
    message = "No database access!";
  }

  if (status === statuses.ALERT) {
    // Send alert
    alertTime = utils.alertRecurrence(message, subject, alertTime);
  } else if (alertTime !== null && status === statuses.OK) {
    utils.sendMail(subject + " - FIXED", "Data available again, error sol-
ved!");
    status = statuses.OK;
    alertTime = null;
  }
```

```
  console.log("DB check end.");
}

module.exports = {
  dbData,
};
```

VIII. Benthami veebikontroll, webcheck.js

```
/* The following code enables the check of data availabi-
lity of Bentham DMc150F-U on the roof of Tartu Observatory.
Checks are done through website ***. */

const rp = require("request-promise");
const $ = require("cheerio");
const statuses = require("../statuses");
const utils = require("../utils");

const url = "***";
const subject = "Bentham web error message";
let alertTime = null;

let lastIdFile = "webcheck_last_saved_id.txt";

// Function that checks data availability from ***
async function webData() {
  let message;
  let status = statuses.OK;

  const table = await rp(url).then(function (html) {
    return $("table div table tr", html); // Find cor-
rect table from the html
  });

  console.log("Number of rows: " + table.length);

  // Check if any data for the current date is available
  if (table.length === 0) {
    message = "No data today: " + new Date(new Date().toUTCString());
    status = statuses.ALERT;
  } else {
    const row = $(table[table.length - 1]).find("td")[0];
    const lastId = $(row).text();
    const priorId = await utils.getPriorId(lastIdFile);

    console.log("Last measurement id:" + lastId);
    console.log("priorID:" + priorId);

    const idCompResult = utils.idComparison(priorId, lastId, lastIdFile);
    status = idCompResult.status;
    message = idCompResult.message;
  }

  if (status === statuses.ALERT) {
    // Send alert
```

```
    alertTime = utils.alertRecurrence(message, subject, alertTime);
  } else if (alertTime !== null && status === statuses.OK) {
    utils.sendMail(subject + " - FIXED", "Data available again, error sol-
ved!");
    status = statuses.OK;
    alertTime = null;
  }
}

module.exports = {
  webData,
};
```

IX. Bentham index.js

```
const cron = require("node-cron");
const { getSunrise, getSunset } = require("sunrise-sunset-js");

const dbcheck = require("./dbcheck");

// Set scheduled check for Bentham
function checkBentham(schedule) {
  console.log("checkBentham: Starting checkBentham, schedule: " + sche-
dule);
  let j = cron.schedule(schedule, function () {
    let sunset = getSunset(58.26585, 26.465926); //Calculates sun-
rise time for Tartu Observatory
    let sunrise = getSunrise(58.26585, 26.465926); //Calculates sun-
rset time for Tartu Observatory

    sunset.setMinutes(sunset.getMinutes() - 30);
    sunrise.setMinutes(sunrise.getMinutes() + 30);

    console.log("checkBentham: Sunrise: " + sunrise);
    console.log("checkBentham: Sunset: " + sunset);

    let currentTime = new Date(new Date().toUTCString());
    console.log("checkBentham: Current time: " + currentTime);

    //Run Bentham check only when current time is between sunrise and sun-
set
    if ((currentTime > sunrise) & (currentTime < sunset)) {
      dbcheck.dbData();
    } else {
      console.log("Skipping check: Sun below horizon.");
    }
  });
}

module.exports = {
  checkBentham,
};
```

X. CIMEL-i arvuti kontroll, localcheck.js

```
// Allows check of CIMEL from control computer

const Client = require("ssh2").Client;
const utils = require("../utils");
const config = require("../../config/config.json");

async function getLastUpdate() {
  // Finds the date of modification for file "current_k7*"
  let conn = new Client();
  let dataDate = new Promise(function (resolve, reject) {
    conn
      .on("ready", function () {
        console.log("Client :: ready");
        conn.exec(
          " ls -C --time-style=long-iso -1 /home/cimel/ | grep cur-
rent_k7 | awk '{print $6, $7}'",
          function (err, stream) {
            if (err) throw err;
            stream
              .on("close", function (code, signal) {
                conn.end();
              })
              .on("data", function (data) {
                dataDate = new Date(data);
                console.log(dataDate);
              })
              .stderr.on("data", function (data) {
                console.log("STDERR: " + data);
              });
          }
        );
      })
      .on("end", function () {
        resolve(dataDate);
      })
      .on("error", function () {
        console.log("getLastUpdate failed");
        reject();
      })
      .connect(config.aeronet);
  });
  return await dataDate;
}

async function checkDate() {
```

```
  // compares last modification date for file "cur-
rent_k7*" to check date and incase of mismatch sends an e-mail
  console.log("CIMEL localcheck:");
  let measurementDate = await getLastUpdate();
  let currentDate = new Date();
  const today = utils.isToday(measurementDate);
  console.log("Measurement: " + measurementDate);
  console.log("Current date: " + currentDate);
  console.log("Today: " + today);
  if (today === false) {
    utils.sendMail("Cimel PC error message", "Curren_k7 fail not up-
dating!");
  }
}

module.exports = {
  checkDate,
};
```

XI. CIMEL-i kodulehe kontroll, webcheck.js

```
// Allows check of CIMEL from an AERONET webpage ""https://aero-
net.gsfc.nasa.gov/new_web/Data_Transfer_Logs/K7/last_1440_minutes.html"

const utils = require("../utils");
const url =
  "https://aeronet.gsfc.nasa.gov/new_web/Data_Trans-
fer_Logs/K7/last_1440_minutes.html";
const statuses = require("../statuses");
const request = require("request");
const subject = "Cimel web error message";

// Check if webpage is receiving data and if there are any error messages
function webData() {
  let message;
  let status = statuses.OK;
  console.log("CIMEL webcheck:");
  request(
    {
      uri: url,
    },
    function (error, response, body) {
      const lines = body.split("\n");
      let match = lines.find((line) => line.includes("Toravere"));
      if (match) {
        console.log(match);
        let words = match.split(" ");
        let i = words.indexOf("file(s)") - 1;
        let value = parseInt(words[i]);
        console.log("Number of files commited: " + value);
        if (value > 0) {
          console.log("Data commited!");
        } else {
          status = statuses.ALERT;
          message = "Data not commited!";
        }
        if (words.includes("Error")) {
          const start =
            lines.findIndex((line) =>
              line.includes("Cimel No  128 System location : Toravere")
            ) + 1;
          const end = start + value;
          const toravereLines = lines.slice(start, end);
          for (index = 0; index < toravereLines.length; index++) {
            let dateTime = toravereLines[index].substring(3, 27);
            if (toravereLines[index].includes("stab")) {
              console.log(toravereLines[index]);
```

```
              message += dateTime + " ";
              message += "Status: b";
              message += ". Robot error!\n";
              console.log("Status: b. Robot error!");
              status = statuses.ALERT;
            }
            if (toravereLines[index].includes("star")) {
              message += dateTime + " ";
              message += "Status: r";
              message += ". Reset error!\n";
              console.log("Status: r.Reset error!");
              status = statuses.ALERT;
            }
            if (toravereLines[index].includes("stap")) {
              message += dateTime + " ";
              message += "Status: p";
              message +=
                ". Sensor head fails to respond to control box communica-
tion request.\n";
              console.log(
                "Status: p. Sensor head fails to respond to cont-
rol box communication request."
              );
              status = statuses.ALERT;
            }
            if (toravereLines[index].includes("stas")) {
              message += dateTime + " ";
              message += "Status: s";
              message += ". Filter wheel error!\n";
              console.log("Status: s. Filter wheel error!");
              status = statuses.ALERT;
            }
          }
        }
      } else {
        status = statuses.ALERT;
        message = "Data not delivered to AERONET";
      }
      if (status === statuses.ALERT) {
        utils.sendMail(subject, message);
      }
    }
  );
}

module.exports = {
  webData,
};
```

XII. CIMEL-i index.js

```
const cron = require("node-cron");

const webcheck = require("./webcheck");
const localcheck = require("./localcheck");

// Set scheduled check for CIMEL
function checkCimel(schedule) {
  console.log("Starting checkCimel, schedule: " + schedule);
  let j = cron.schedule(schedule, function () {
    let currentTime = new Date(new Date().toUTCString());
    console.log("Current time: " + currentTime);
    webcheck.webData();
    localcheck.checkDate();
  });
}

module.exports = {
  checkCimel,
};
```

XIII. Litsents

Lihtlitsents lõputöö reprodutseerimiseks ja üldsusele kättesaadavaks tegemiseks

Mina, Margit Aun,

(autori nimi)

1. annan Tartu Ülikoolile tasuta loa (lihtlitsentsi) minu loodud teose

„Instrumentide monitoorimissüsteem Tartu observatooriumi atmosfääriseire töörühmale“,

(lõputöö pealkiri)

mille juhendajad on Ilmar Ansko ja Tõnis Eenmäe,

(juhendaja nimi)

reprodutseerimiseks eesmärgiga seda säilitada, sealhulgas lisada digitaalarhiivi DSpace kuni autoriõiguse kehtivuse lõppemiseni.

2. Annan Tartu Ülikoolile loa teha punktis 1 nimetatud teos üldsusele kättesaadavaks Tartu Ülikooli veebikeskkonna, sealhulgas digitaalarhiivi DSpace kaudu Creative Commonsi litsentsiga CC BY NC ND 3.0, mis lubab autorile viidates teost reprodutseerida, levitada ja üldsusele suunata ning keelab luua tuletatud teost ja kasutada teost ärieesmärgil, kuni autoriõiguse kehtivuse lõppemiseni.

3. Olen teadlik, et punktides 1 ja 2 nimetatud õigused jäävad alles ka autorile.

4. Kinnitan, et lihtlitsentsi andmisega ei riku ma teiste isikute intellektuaalomandi ega isikuandmete kaitse õigusaktidest tulenevaid õigusi.

Margit Aun

13.05.2020