TALLINNA TEHNIKAÜLIKOOL
Infotehnoloogia teaduskond

Viljar Hera 030598IAPB

LÄMMASTIKU ANALÜSAATORI GAASIKULUREGULAATORITE KALIBREERIMISRAKENDUS

bakalaureusetöö

Juhendaja: Jaagup Irve

Tehnikateaduste magister

Tallinn 2018

Autorideklaratsioon

Kinnitan, et olen koostanud antud lõputöö iseseisvalt ning seda ei ole kellegi teise poolt varem kaitsmisele esitatud. Kõik töö koostamisel kasutatud teiste autorite tööd, olulised seisukohad, kirjandusallikatest ja mujalt pärinevad andmed on töös viidatud.

Autor: Viljar Hera

21.05.2018

Annotatsioon

Käesoleva bakalaureusetöö eesmärgiks on luua töölaua rakendus, mis võimaldaks lämmastiku analüsaatori Turbo N 4040 kuluregulaatorite kalibreerimist automatiseerida.

Töös käsitletakse kuluregulaatorite kalibreerimisega seotud seadmete tööprintsiipe ja kalibreerimise metoodikat. Esitatakse nõuded loodavale rakendusele, tuuakse lühidalt välja kuidas lahenduseni jõuti ja mida kasutati selle saavutamiseks.

Töö tulemusena valminud rakendust vaadeldakse arhitektuuriliselt, jaotades selle komponendideks ning käsitledes neid individuaalselt.

Lõputöö on kirjutatud eesti keeles ning sisaldab teksti 41 leheküljel, 5 peatükki, 19 joonist, 4 tabelit.

Abstract

Applicaton for calibrating the mass flow controllers of the nitrogen analyzer

The purpose of this bachelor's thesis is to create a desktop application to provide automatic calibration for the mass flow controllers of the Turbo N 4040 series nitrogen analyzer.

This thesis consists of operating principles of equipment and methodology for calibrating the mass flow controllers. The document presents requirements for the application and describes how the solution was reached and which tools were used to achieve it.

The main part of the thesis concentrates on the architectural overview of the created application dividing it into components. Every component is viewed separately.

The thesis is in estonian and contains 41 pages of text, 5 chapters, 19 figures, 4 tables.

Lühendite ja mõistete sõnastik

API	*Application programming interface*, rakendusliides
ASCII	*American Standard Code for Information Interchange*, Ameerika Informatsioonivahetuse Standardkood
COM	*Communication port*, järjestikport
FC1	*Flow Controller 1*, kuluregulaatori tüüp 1
FC2	*Flow Controller 2*, kuluregulaatori tüüp 2
HTML	*Hypertext Markup Language*, hüperteksti märgistuskeel
ML-500	*Met Lab 500 Series,* gaasikulu mõõteseade
N4040	Lämmastiku analüsaator Turbo N 4040
RS-232	Järjestikliidese standard
SQL	*Structured Query Language*, struktuurpäringukeel
UML	*Unified Modeling Language*, ühtne modelleerimiskeel

Sisukord

Jooniste loetelu

Tabelite loetelu

1 Sissejuhatus

Costech Microanalytical OÜ (edaspidi Costech) tegeleb mõõteseadmete tootmisega. Ettevõtte jaoks üks olulisemaid tooteid on analüsaator Turbo N 4040 (edaspidi N4040), mis võimaldab määrata lämmastikku või valgu sisaldust erinevates ainetes. N4040 kõrval on ettevõttel ka muid sama tehnoloogia alusel töötavaid tooteid, mis kõik on ehituselt sarnased. Nende ühisteks komponentideks on Costechi partnerettevõtte PTR Grupp OÜ poolt arendatud ja toodetud täppisgaasikuluregulaatorid, mida igal mõõteseadmel on kaks. Kuluregulaatorite paar peab tagama täpse gaasi voo läbi mõõteseadme, mis on omakorda ka täpse mõõtetulemuse eelduseks. Selle ülesande täitmiseks on tarvis kuluregulaatorid nõuetekohaselt kalibreerida.

Ettevõtte huvi on mõõteseadme tootmiseks ning seadistamiseks kuluvat aega vähendada. Siiani on seadmeid kalibreeritud operaatori juuresolekul ning juhtimisel, niiöelda käsitsi. Käsitsi kalibreerimine on võrdlemisi aeganõudev tegevus ning lisaks hõivab tööjõudu. Toodetavate mõõteseadmete arvu kasvades oleks ajaline võit märkimisväärne, seetõttu on olemas vajadus kalibreerimise automatiseerimiseks.

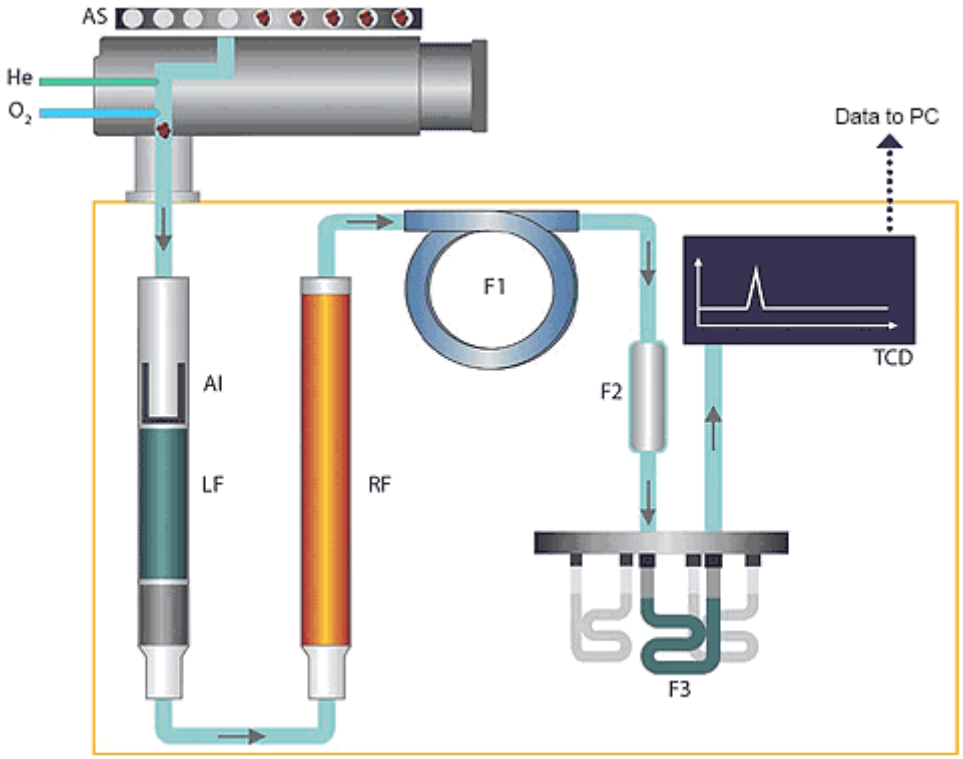
Käesoleva bakalaureusetöö **eesmärk** on luua tarkvara, mis võimaldab täppisgaasikuluregulaatoreid kalibreerida ning lisaks testida ja saadud tulemused arhiveerida. Tarkvara peaks seejuures töötama võimalikult vähese kasutajapoolsel osavõtul.

2 Taust

Käesolevas peatükis käsitletakse kuluregulaatorite kalibreerimisega seotud seadmete tööprintsiipe, kalibreerimise metoodikat ning suhtlust erinevate seadmetega.

2.1 Analüsaatori Turbo N 4040 tööpõhimõte

N4040 on mõeldud määrama kvantitatiivselt lämmastiku / valgu sisaldust toitudes, söötades, muldades, väetistes ja muudes orgaanilistes ühendites. Analüsaatori toimimise aluseks on Dumas meetod [1], mis näeb ette analüüsitava proovi põletamist ning seejärel lämmastiku sisalduse mõõtmist soojusjuhtivus detektoris (TCD). Järgnevalt kirjeldatakse proovi teekonda läbi mõõteseadme võttes aluseks joonise 1.



Joonis 1. Turbo N 4040 gaasiskeem [3, lk 10].

Analüsaator koosneb järjestik gaasikseemist, millest läbi tsirkuleerib kandegaasina heelium (He). Analüsaatori töö protsess algab autosamplerist (AS), millese asetatakse

tina kapslis järjestikku kaalutud proovid. Proov kukutatakse umbes 1000 °C temperatuuriga põlemisreaktorisse (LF), mis seejärel küllastuses hapniku (O_2) juuresolekul täielikult põleb. Orgaanilise aine põlemine toimub valemis (1) näidatud üldkujul [2, lk 7].

$$C_a H_b N_c + n\,O_2 \rightarrow a\,CO_2 + \frac{b}{2} H_2O + c\,NO_x \quad (1)$$

Põlemise käigus tekkinud süsihappegaas (CO_2), vesi (H_20), lämmastikoksiidi ühendid (NO_X) ning põlemisest järele jäänud hapnik (O_2) juhitakse kandegaasi abil redutseerimisreaktorisse (RF), mis on täidetud vase (Cu) graanulitega. Umbes 650 °C temperatuuri all lämmastik redutseerub ning kogu hapnik seotakse vasega valemi (2) näidatud viisil [2, lk 7].

$$O_2 + 2\,Cu \rightarrow 2\,CuO \quad (2)$$

Lämmastikoksiidi ühendid reageerivad samuti vase graanulitega ning tulemusena vask oksüdeerub ning vabaneb lämmastik vastavalt valemile (3) [2, lk 7].

$$NO_x + x\,Cu \rightarrow \frac{1}{2} N_2 + x\,CuO \quad (3)$$

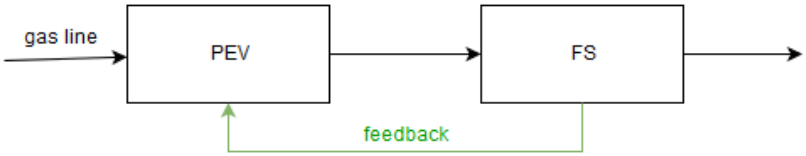
Mõlema reaktsiooni tulemuseks on vaskoksiid (CuO), mis on tahke aine ning seetõttu reaktorist edasi ei pääse. Redutseerimisreaktorist juhitakse gaasisegu, mis koosneb süsihappegaasist, lämmastikust, kandegaasist heelium ja aurustunud veest, veelõksudesse (F1; F2), kus gaasisegust eraldatakse kogu vesi. Seejärel gaasisegu läbib adsorberitorusid (F3), mis on täidetud poorsete molekulaarsõeladega. Molekulaarsõelte poorid on spetsiaalsete mõõtmetega, et süsihappegaasi molekulid, olles mõõtmetelt suuremad kui lämmastiku molekulid või heeliumi aatomid, jääksid pooridesse kinni. Järelejäänud heeliumi ja lämmastiku segu annab soojusjuhtivusdetektoris väljundsignaali, mis on proportsionaalne lämmastiku sisaldusega põletatud proovis [2, lk 7].

2.2 Täppisgaasikuluregulaatorite tööpõhimõte

Soojusjuhtivusdetektor reageerib lämmastiku ja heeliumi kontsentratsioonile aga samuti ka gaasisegu voolule ehk kulule. Viimase mõju neutraliseerimiseks on vajalik, et igal mõõtmisel oleks gaasi kulu ühesugune. Selle ülesande täidavad täppisgaasikuluregulaatorid (edaspidi kuluregulaatorid).

Lämmastikuanalüsaatoril on kaks erinevat tüüpi kuluregulaatorit: FC1 ja FC2. FC1 on paigutatud vahetult enne põletamisreaktorit ja selle ülesandeks on hapniku või heeliumi kulu hoidmine ettemääratud väärtusel, et tagada korrektne proovi põlemine ning ühtlane kandegaasi tsirkulatsioon läbi kogu analüsaatori gaasiskeemi.

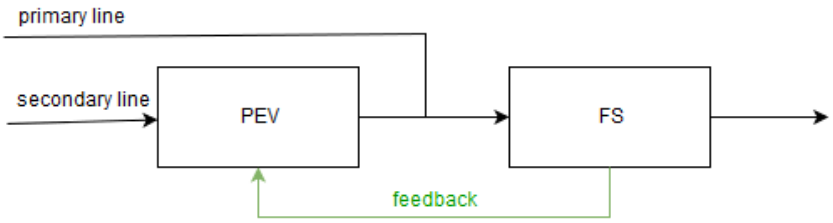
Kuluregulaatorisse on järjestikku ühendatud proportionaalventiil (PEV – *Proportional Electromagnetic Valve*) ja kuluandur (FS – *Flow Sensor*) joonisel 2 esitatud viisil. Proportsionaalventiil on tavaolekus täielikult avatud. Kui gaas läbib kuluandurit, reguleeritakse tagasisidega proportsionaalventiili vastavalt kuluanduris tekkinud signaalpingele, millele vastavalt ventiili läbilase kas suureneb või väheneb kuni saavutatakse soovitud kulu [3, Ptk 2.3].



Joonis 2. FC1 tööskeem [3, Joon 4].

FC2 on paigutatud gaasiskeemi vahetult enne soojusjuhtivusdetektorit eesmärgiga taastada gaasikulu, mis analüüsitava proovi põlemise järgselt väheneb. Kulu vähenemine on tingitud põlemisel tekkinud kõrvalsaaduste gaasisegust kinni püüdmise tõttu.

FC2 toimib samal põhimõttel kui eelnevalt kirjeldatud FC1. Oluliseks erinevuseks on täiendav gaasiliini sisend, mida on näha FC2 tööskeemil joonisel 3. Sekundaarliini läbilaset reguleeritakse vastavalt primaarliini kulu puudujäägile soovitud kulust.



Joonis 3. FC2 tööskeem [3, Joon 10].

Kulureguraatorites asetseb mikrokontroller, mis võimaldab tarkavaralist juhtimist ning seadistusi.

2.3 Etalonmõõteseadme ML-500 tööpõhimõte

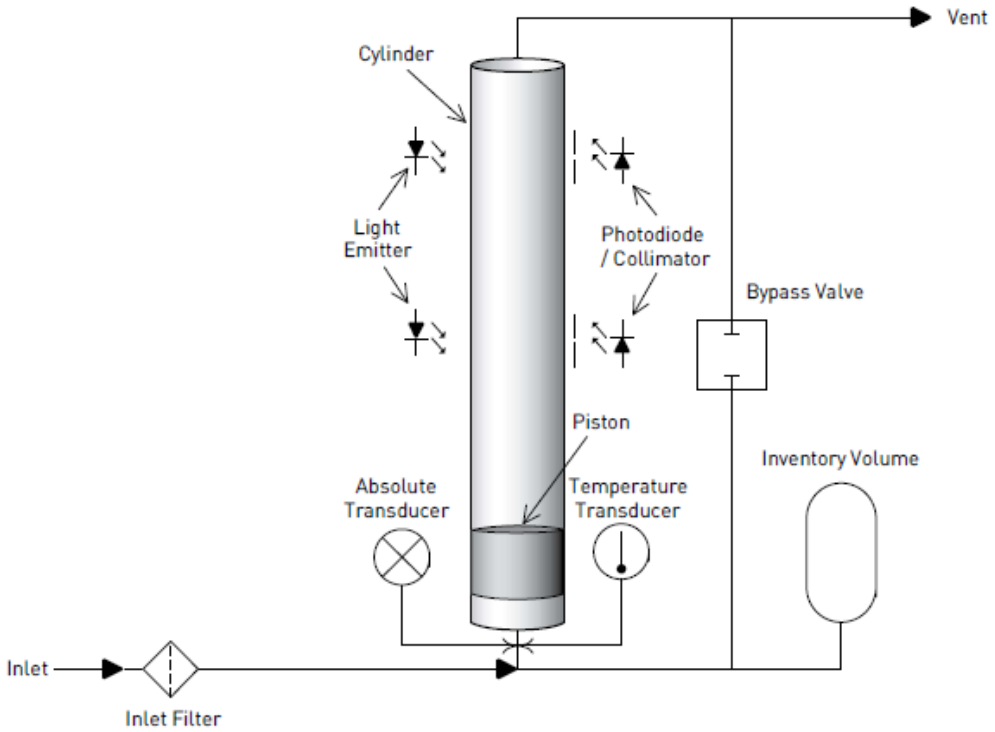
Kalibreerimise etaloniks on Bios DryCal tehnoloogial põhinev Met Lab seeriast mõõteseade ML-500, mille tootjaks on Mesa Labs. Mõõteseade koosneb kahest osast: alusest (*Base*) ja mõõtepuurist (*Flow Cell*). Alus sisaldab arvutit seadme juhtimiseks ning kristallkella ajavõtuks. Gaasi kulu mõõtmine toimub puuris, milles asetseb hermeetiliselt suletud silinder ning omakorda silindri sees liigub kolb [4, lk 2].

Kolbtõestaja (*Piston Prover*), mis koosneb massitust, lekkevabast, höördevabast, kujust-sõltumatust kolvist ning täiuslikust silindrist, on mahtkulu leitav vastavalt valemile (4) [4, lk 3]. Valemist järeldub, et mahtkulu on otseses seoses mõõtmise ajaga t kuna silindri füüsilisi mõõtmeid võib lugeda muutumatuteks.

$$q = \frac{V}{t} = \frac{\pi \cdot r^2 \cdot h}{t} \tag{4}$$

q – mahtkulu
V – mõõtmise ruumala
t – mõõtmise aeg
r – silindri raadius
h – mõõtmise teekonna pikkus

Aja mõõtmiseks on läbipaistva silindri (*Cylinder*) väliskülgedele kahele tasandile paigaldatud valgusdioodid (*Light Emitter*), mis kiirgavad valgust, ning vastaspoolele valgusanduritena fotodioodid (*Photodiode / Collimator*). Kui gaasi mõjul liikuv kolb (*Piston*) jõuab valgusdioodi ja fotodioodi vahele siis registreeritakse aeg. Tasandite vahele jääva ruumala ja kolbi ühelt tasandilt teiseni jõudmiseks kulunud aja suhe ongi mõõdetav mahtkulu [4, lk 3].



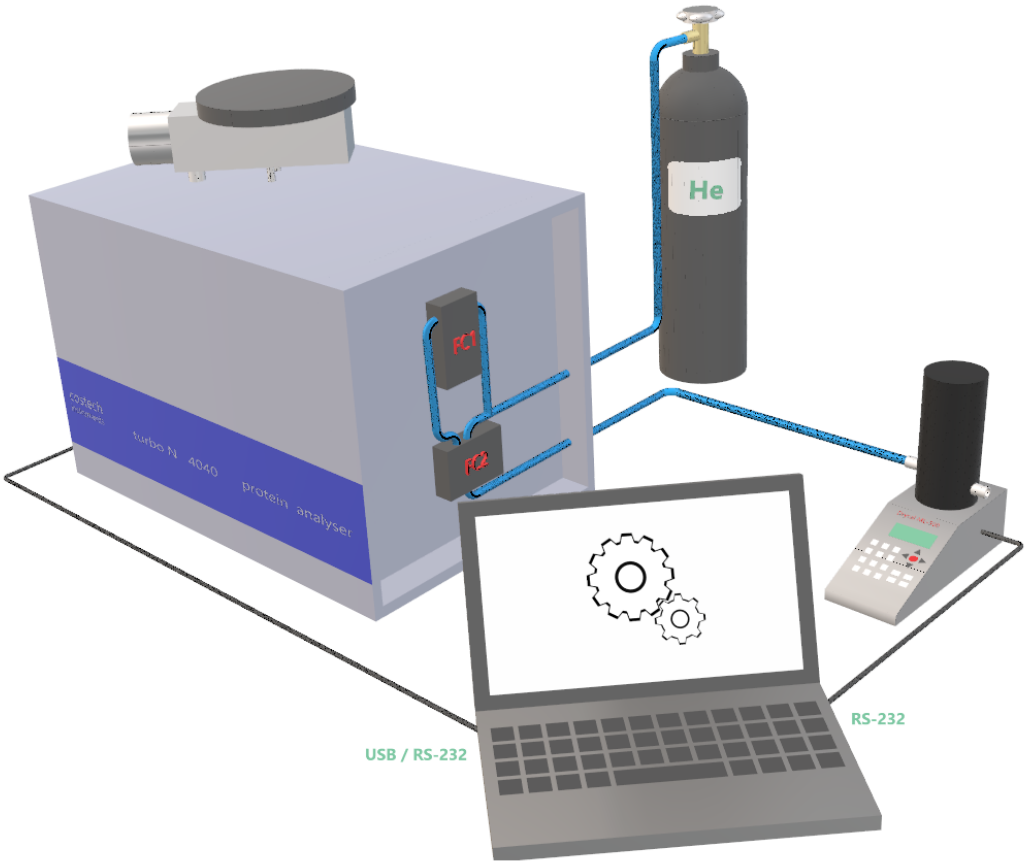
Joonis 4. ML-500 tööskeem [4, lk 3].

Lisaks sisaldab mõõtepuur rõhuandurit (*Absolute Transducer*) ning temperatuuriandurit (*Temperature Transducer*), mis võimaldab mõõtetulemusi standardiseerida [5]. Standardiseerimisel viiakse mõõtetulemus eelnevalt kokkulepitud standardtingimustele. See võimaldab erinevatel tingimustel mõõtmistel saadud tulemusi võrrelda. Käesoleva bakalaureuse töö raames on standardtingimusteks atmosfääri rõhk 760 mmHg ja ümbritsev temperatuur 0 °C. Edaspidi kui räägitakse kulust siis mõeldakse selle all standardiseeritud kulu.

ML-500 alusele on võimalik panna erineva mõõtepiirkonnaga mõõtepuure. Kasutusel olev puur, mudel 10, võimaldab mõõta mahtkulu (*volumetric flow*) täpsusega ±0,25% ja standardiseeritult (*standardized flow*) täpsusega ±0,40% piirkonnas 5-500 ml/min [4, lk 17].

2.4 Kommunikatsioon seadmetega

Kuluregulaatorite kalibreerimiseks on vajalik, et nii analüsaator Turbo N4040 kui ka etalonmõõteseade ML-500 oleksid ühenduses personaalarvutiga, kus kalibreerimisrakendus jookseb. N4040 saab ühendada RS-232 nullmodemkaabliga, kuid on võimalik kasutada ka USB (*Universal Serial Bus*) kaablit. USB liidese peale virtualiseeritakse operatsioonisüsteemi tasemel COM port (*Communication port*) ning seetõttu ei ole tarkvaraliselt mingisugust erinevust suhtlemisel üle kummagi liidese. ML-500 puhul kasutatakse RS-232 otsekaablit. Joonisel 5 on näha kuidas ühendatakse seadmed kalibreerimiseks tarkvaralisel juhtimisel.



Joonis 5. Seadmete omavahelised ühendused tarkvaraga kalibreerimisel

2.4.1 N4040-ga suhtlemine

Lämmastikuanalüsaatori elektroonilisi komponente ühendavas emaplaadis asetseb programmeeritud mikrokontroller, mis võimaldab API-t (*Application programming interface*) kogu seadme juhtimiseks. Kuigi kuluregulaatorid sisaldavad sarnaselt

emaplaadile samuti mikrokontrollereid, mis võimaldavad omakorda API-t nende juhtimiseks, on emaplaadi mikrokontroller vahendaja rollis. Seetõttu käsitletakse järgnevalt ainult personaalarvuti ja emaplaadi mikrokontrolleri vahelist suhtlust.

Personaalarvuti rakendus suhtleb emaplaadi mikrokontrolleriga kasutades ASCII (*American Standard Code for Information Interchange*) tähemärke. Joonisel 6 on esitatud 2 baidi pikkune päring, millega soovitakse teada FC1 kuluregulaatori polünoomi kordajaid.

```
\0x90\0x0A
```

Joonis 6. FC1 polünoomi kordajate päring. (\0x90 – päringu identifikaatorbait; \0x0A – andmejada lõppu tähistav bait)

Päringu vastus saadetakse joonisel 7 esitatud kujul, mis sisaldab polünoomi kordajaid kuueteistkümnendik süsteemis (esitatud paksus kirjas). Konverteerides saab neist nelja baidi pikkused ujuvkoma arvud.

```
\0x903E8678E2 4274926F 3FA397CC 3F62BB56 0\0x0A
```

Joonis 7. Vastus FC1 polünoomi kordajate päringule. (\0x90 – päringu identifikaatorbait; \0x0A – andmejada lõppu tähistav bait)

2.4.2 ML-500-ga suhtlemine

Sarnaselt N4040-le on võimalik personaalarvuti rakendusel ML-500-ga suhelda üle järjestikliidese kasutades ASCII tähemärke. Olulisim saadetav käsk on joonisel 8 esitatud mõõtmine.

```
$GET DQ DC\0x0D
```

Joonis 8. ML-500-ga mõõtmise alustamiseks saadetav käsk. (\0x0D – andmejada lõppu tähistav bait)

Käskluse kättesaamisele järgnev protseduur võtab olenevalt kulust pisut aega. Mõõtmise lõppedes saadab ML-500 tulemuse joonisel 11 esitatud kujul. Tulemuses on andmed komadega eraldatud ning kindlas, ML-500 dokumentatsioonis kirjeldatud, järjekorras [5].

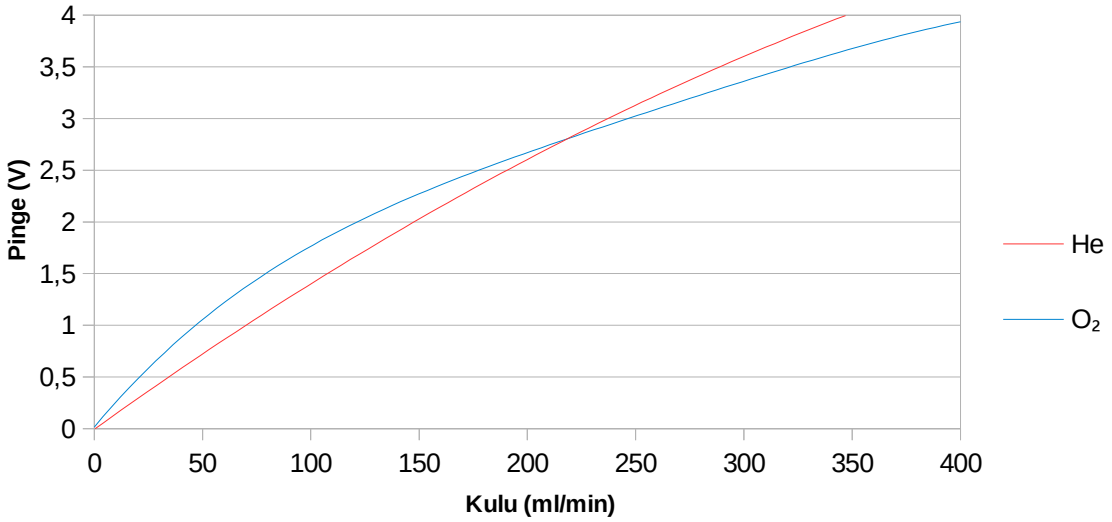
```
  46.120, 24.9,C, 756.4,mmHg ,756.5, 756.6,0.058,ML-500,Base,111244,
2.03,ML-500,Cell:10,111874, 2.03,,,,,,,,,\0x0D\0x0A
```

Joonis 9. Mõõtmise tulemus. (\0x0D\0x0A – andmejada lõppu tähistav kahest baidist koosnev baidikombinatsioon)

2.5 Kalibreerimise meetod

Kuluregulaatorite kalibreerimine tähendab sisuliselt sõltuvusfunktsiooni f(u) = q leidmist, kus u on kuluanduri signaalpinge ja q on gaasi kulu. Kusjuures funktsioon f(u) on polünoom.

Kuluregulaatoreid kalibreeritakse iga gaasiga eraldi, sest kuluanduri tundlikkus erinevate gaaside suhtes on erinev. Seda erinevust illustreerib kuluanduri tundlikkuse graafik joonisel 10. Graafikult on näha, et heeliumi (He) korral võrreldes hapnikuga (O_2) on kuluanduri tundlikkus kulu suhtes mõnevõrra lineaarsem.



Joonis 10: Kuluanduri tundlikkus heeliumi ja hapniku korral

Kuluanduri signaalpinge (u) ning näidu (q) vahelist seost arvutatakse valemis (5) esitatud 4-astmelise polünoomiga, mille kordajad (a_0-a_4) on leitavad huvipakkuvas piirkonnas tehtud mõõtmiste regressioonanalüüsiga. Antud juhul huvipakkuv piirkond on vahemikus 0-400 ml/min.

$$q = f(u) = a_4 \cdot u^4 + a_3 \cdot u^3 + a_2 \cdot u^2 + a_1 \cdot u + a_0 \quad (5)$$

Tehase seadetega kalibreerimata kuluregulaatoril on seadistatud kordajad väärtustega $a_4 = 0;\ a_3 = 0;\ a_2 = 0;\ a_1 = 100;\ a_0 = 0$. Selliste kordajatega polünoom võimaldab valemiga (6) lihtsat teisendust kuluregulaatori näidust kuluanduri signaalpingeks, mis on vajalik korrigeeritud polünoomi leidmiseks.

$$u = \frac{1}{100} \cdot q \tag{6}$$

Kuluregulaatorite kalibreerimise töökäik on järgnev. Kõigepealt seatakse kuluregulaatorisse kulu ette vastavalt valitud huvipakkuva piirkonna jaotusele. Kui gaasivool on stabiliseerunud registreeritakse kuluregulaatori näit ja mõõdetakse tegelik kulu etalonmõõteseadmega. Juhuslikku viga arvesse võttes, tehakse mõõtmisi igas etteseatud kulu punktis korduvalt ning leitakse nende tulemuste aritmeetiline keskmine. Valemiga (6) arvutatakse kuluregulaatori näidust kuluanduri signaalpinge. Seda tegevuskeemi korratakse igas huvipakkuva piirkonna etteseatud kulu punktis. Tulemuseks on tabelis 1 esitatud kuluanduri signaalpingete (märgitud sinisega) ja vastavate mõõdetud kulude (märgitud punasega) kogum, mille järgi arvutatakse regressioonanalüüsiga korrigeeritud polünoom.

Tabel 1. Kalibreerimistabel.

Etteseatud kulu (ml/min)	**Kuluregulaatori näit (ml/min)**	**Kuluanduri signaalpinge (V)**	**Mõõdetud kulu (ml/min)**
0	0.000	0	0.000
40	39.960	0.39960	16.025
80	80.018	0.80018	34.752
120	120.038	1.20038	58.427
160	159.992	1.59992	87.379
200	199.970	1.99970	122.786
240	240.010	2.40010	165.681
280	280.050	2.80050	216.400
320	320.030	3.20030	274.117
360	360.008	3.60008	339.567
400	400.000	4.00000	416.645

Leitud polünoomi kordajad viiakse sisse tarkvaraliselt kuluregulaatori mikrokontrollerisse, mille järgselt tehakse polünoomi verifitseerimiseks mõõtmised erinevates mõõtepiirkonna punktides. Tabelist 2 on näha väike kuluregulaatori näitude

ja mõõdetud kulude vaheline erinevus. Kui see erinevus jääb aktsepteeritavasse vahemikku igas mõõtepunktis siis võib lugeda kalibreerimise edukaks.

Tabel 2. Testimine peale polünoomi sisseviimist.

Etteseatud kulu (ml/min)	**Kuluregulaatori näit (ml/min)**	**Mõõdetud kulu (ml/min)**
200	200.013	200.569
300	300.160	299.952
400	400.100	399.683

3 Metoodika

Käesolevas peatükis esitatakse nõuded loodavale rakendusele, tuuakse lühidalt välja kuidas lahenduseni jõuti ja missuguseid vahendeid kasutati selle saavutamiseks

3.1 Nõuded

Rakendust hakatakse kasutama Windows keskkonnas laua- või sülearvutil, seetõttu on vajalik Windows XP/7/8/10 operatsioonisüsteemide tugi.

Rakendusel peaks olema graafiline kasutajaliides, kuid kuna seda hakatakse kasutama ainult mõne töötaja poolt ning asutuse siseselt, siis kasutajaliidese väljanägemine ei ole prioriteet. Siiski peaks rakendust olema mugav kasutada.

Rakendus peab võimaldama eelnevalt seadistada kalibreerimistingimusi:

- määrata gaasi tüüp
- valida kumbagi kuluregulaatorit üksikult või korraga
- valikuliselt sisestada atmosfääri rõhku
- sisestada kalibreerimispunkte
- sisestada mõõtepunkte kalibreerimisjärgseks testimiseks
- määrata mitu mõõtmist ühes punktis läbi viiakse

Kalibreerimine ei tohiks nõuda kasutajapoolset sekkumist välja arvatud vea olukordade puhul.

Rakendus peab võimaldama ka testida kalibreeritud kuluregulaatoreid, et saaks veenduda kalibreeringu õigsuses.

Kalibreeringu tulemused on tarvis arhiveerida, et need oleks kunagi hiljem vajaduse korral kergesti leitavad. Selleks peaks rakendus võimaldama otsida arhiivist kalibreeringuid ning kuvama nende kohta käivaid andmeid.

Arhiivi on tarvilik salvestada järgnevad andmed:

- kalibreeringu algus- ja lõppaeg
- kalibreerimise tingimused (gaas, atmosfääri rõhk, korduste arv)
- analüsaatori identifikaator
- kuluregulaatori(te) identifikaator(id)
- kuluregulaatori(te) termokompensatsiooni parameetrid
- mõõtmistulemused etalonmõõteseadmest
- mõõtepunktide mõõtetulemuste arvutatud keskmised
- mõõtetulemuste põhjal arvutatud polünoom ja korrelatsioonikordaja
- kuluregulaatori(te) temperatuur igal mõõtmisel

3.2 Tehnoloogia

Rakendust arendati Qt platvormi peale, kasutati versiooni 4. Platvorm valiti eelkõige peatükis 3.1 väljatoodud nõuetega sobivuse tõttu. Samuti sai määravaks arendaja eelnev kokkupuude antud platvormiga.

Qt on C++ keelel põhinev arendusraamistik mitmeplatvormsete rakenduste arendamiseks ja on mõeldud eelkõige graafiliste kasutajaliidestega rakendustarkvara loomiseks. Qt kasutab koodigeneraatoreid uic (*User Interface Compilator*) ja moc (*Meta Object Compilator*). XML (*Extensible Markup Language*) formaadis kasutajaliidese kirjeldusest genereerib C++ koodi uic, millega seatakse üles kasutajaliidese vorm. Moc genereerib kompilleerimise ajal juurde lisakoodi pakkudes lisavõimalusi võrreldes standardse C++iga [6]. Fundamentaalseim neist on signaal ja pesa mehhanism (*signal & slot mechanism*), mis on mõeldud objektide vaheliseks kommunikatsiooniks [7].

Andmebaasidega suhtlemiseks on Qt raamistikul olemas eraldi moodul QtSql, mis töötab abstraheerimise kihina ning toetab laia valikut SQL (*Structured Query Language*) andmebaase [8], [9]. Rakenduse nõuetest ning kasutamisviisist lähtuvalt valik osutus SQLite andmebaasi kasuks. SQLite on kergekaaluline andmebaasimootor ning täielik SQL implementatsioon. Toetatud on ka transaktsioonid. SQLite ei nõua

eraldi serveri protsessi päringute vastuvõtmiseks vaid kõik andmebaasi operatsioonid viiakse läbi rakenduse protsessis. Andmed on võimalik salvestada kohalikku faili [10].

Seadmetega suhtlemise eest üle järjestikliidese kannab hoolt QextSerialPort, mis on samuti Qt moodul. Erinevalt QtSql moodulist on QextSerialport kolmanda osapoole poolt arendatud ning ei tule Qt paketiga kaasa. QextSerialPort toetab Mac OS X-i, Windowsi, Linuxi ja FreeBSD operatsioonisüsteeme [11].

Kuluregulaatori polünoomi arvutamiseks kasutatakse GSL (*GNU Scientific Library*) teeki. GSL on C keele baasil kogum matemaatilise funktsioone, nende seas leiduvad ka statistilised funktsioonid regressioonanalüüsiks [12].

3.3 Arendusvahendid

Arenduse algfaasis leidis rakendust Eclipse IDE (*Integrated Development Environment*). Eclipse on ülesehitatud laiendatava moodulsüsteemina [13], mis võimaldab programeerimiskeskkonnal pakkuda tuge paljudele programmeerimiskeeltele ning raamistikele. Üks nendest on ka Qt. Läbi Qt integratiooni mooduli on võimaldatud Qt projektide arendus Eclipses.

Arenduse vältel sai selgeks, et Eclipse Qt integratsiooni moodul pakkus piiratud võimalusi ning see hakkas segama. Seetõttu toimus üleminek Qt Creatorile, mis on Qt arenduspaketiga vaikimisi kaasaskäiv IDE.

Andmebaasi haldusel oli abiks SQLite Manager, mis on Mozilla Firefoxi veebilehitseja lisa. SQLite Manager võimaldab viia läbi andmebaasi haldamisfunktsioone.

Abivahendina seadmetega suhtlemisel kasutati terminaliklienti PuTTY, mis võimaldab luua ühendusi üle järjestikliidese. Üle ühenduse saab antud programmiga saata ja vastu võtta teksti.

3.4 Arenduskäik

Rakenduse arendamine algas võimaluste uurimisega etalonmõõteseadme ML-500 juhtimiseks üle järjestikliidese. Kasutades tootja poolt dokumenteeritud juhiseid ning

terminaliklienti PuTTY viidi läbi katsetused, mille käigus selgitati välja kuidas toimib etalonmõõteseadme ML-500 protokoll. Sarnaselt katsetati ka analüsaatorit N4040, kuid dokumentatsiooni puudumise tõttu oli vaja protokoll eelnevalt kaardistada. Seda tehti koostöös väljatöötajaga.

Otsiti võimalusi Qt keskkonnas üle järjestikliidese andmeid saata ning vastu võtta. Õpiti tundma QextSerialPort moodulit, mille baasil arendati välja komponendid ML-500 ning N4040-ga suhtlemiseks.

Seadmetega suhtlemiseks loodud aluskiht võimaldas hakata arendama kalibreerimist läbiviivat loogikat. Töötati välja mõõtetsükkel, mille ümber lisati järjest uut funktsionaalsust. Selle hulka kuulub mõõtetulemuste standardiseerimine, polünoomi arvutus, andmete arhiveerimine.

Andmete arhiveerimiseks töötati välja andmete relatsiooniline mudel, mille alusel loodi SQLite andmebaas. SQL päringute arendamisel kasutati abistava vahendina SQLite Manager tööriista.

Kalibreerimist läbiviiva loogika kõrval arendati paralleelselt kasutajaliidest, et oleks võimalik valminud funktsionaalsust koheselt ka juhtida ning katsetada. Viimasena keskenduti arhiivi arendamisele, mis nägi ette kasutajaliidese täiendamist ning andmebaasi päringute lisamist.

4 Arhitektuur

Käesolevas peatükis käsitletakse valminud rakendust arhitektuurilises vaates, jaotades selle komponendideks ning käsitletakse neid individuaalselt.

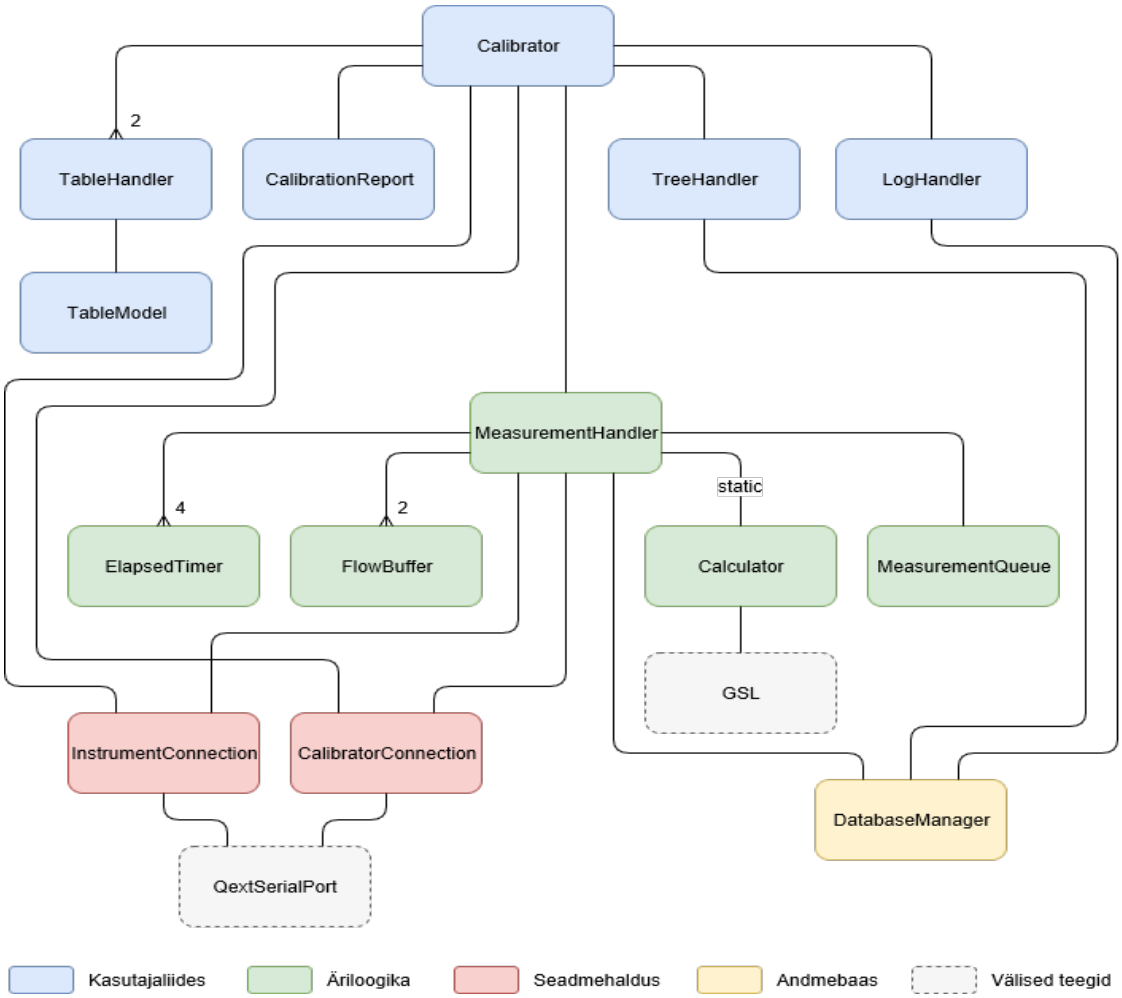
4.1 Rakenduse ülesehitus

Rakenduse lähtekood järgib objektorienteeritud programmeerimise põhimõtteid tulenevalt Qt arendusraamistiku olemusest. Qt arendusraamistikku on käsitletud peatükis 3.2.

Funktsionaalsuse alusel saab vaadelda rakenduse ülesehitust nelja kihina:

1. Kasutajaliides – Rakenduse ja kasutaja vaheline interaktsiooni loogika.
2. Äriloogika – Funktsionaalsus ning töökorraldus.
3. Seadmehaldus – Rakenduse ja juhitavate seadmete vaheline kommunikatsioon.
4. Andmebaas – Andmete pärimine ja salvestamine.

Joonisel 11 esitatakse UML (*Unified Modeling Language*) klassidigramm, milles on näidatakse arenduse käigus loodud klassid, nende omavahelised seosed ning grupiviisiline jaotus funktsionaalsuse alusel kihtideks.



Joonis 11. Rakenduse klassidiagramm.

Tabelis 3 esitatakse lühikirjeldus iga joonisel 11 esitatud klassidiagrammis oleva klassi kohta.

Tabel 3. Klasside semantika.

Klass / teek	**Semantika**
Calculator	Antud klass sisaldab staatilisi funktsioone gaasi kulu arvutamiseks ning polünoomi leidmiseks.
CalibrationReport	Kalibreerimise lõpetamisel näidatav aruande dialoog.
Calibrator	Rakenduse peaaken. Enamus kasutajaliidese loogikast paikneb antud klassis.
CalibratorConnection	Võimaldab etalonmõõteseadme ML-500 juhtimist.
DatabaseManager	Andmebaasi andmete salvestamine ja lugemine.
ElapsedTimer	Aega loendav taimer.
FlowBuffer	Selle klassi baasil luuakse kulureguaatori näite jälgivad puhvrid, mis võimaldavad leida aritmeetilist keskmist ning tuvastada näidu kõikuvust.
GSL	GNU Scientific Library – teek võimaldab polünoomi leidmist.
InstrumentConnection	Võimaldab lämmastiku analüsaatori N4040 jälgimist ning juhtimist.
LogHandler	Antud klassiga võimaldatakse logimine.
MeasurementHandler	Sisaldab kalibreerimise läbiviimise loogikat.
MeasurementQueue	lass, mille baasil luuakse mõõtejärjekord. Kalibreerimisel võetakse ükshaaval mõõtejärjekorrast tühjasid mõõtmiste objekte, mis peale mõõtmist mõõtetulemustega täiendatakse.
QextSerialPort	Qt moodul, mis võimaldab kommunikatsiooni üle järjestikliidese.
TableHandler	Kalibreerimise ja testimise tabelite, millesse kasutaja saab ette anda mõõtmiseks gaasi kulusid, käitumine on implementeeritud käesolevas klassis.
TableModel	Klass, mille baasil luuakse kalibreerimis- ja testimistabelite mudelid. Mudelid võimaldavad tabeli lahtritesse sisestada andmeid.
TreeHandler	Arhiivi navigatsiooni puu.

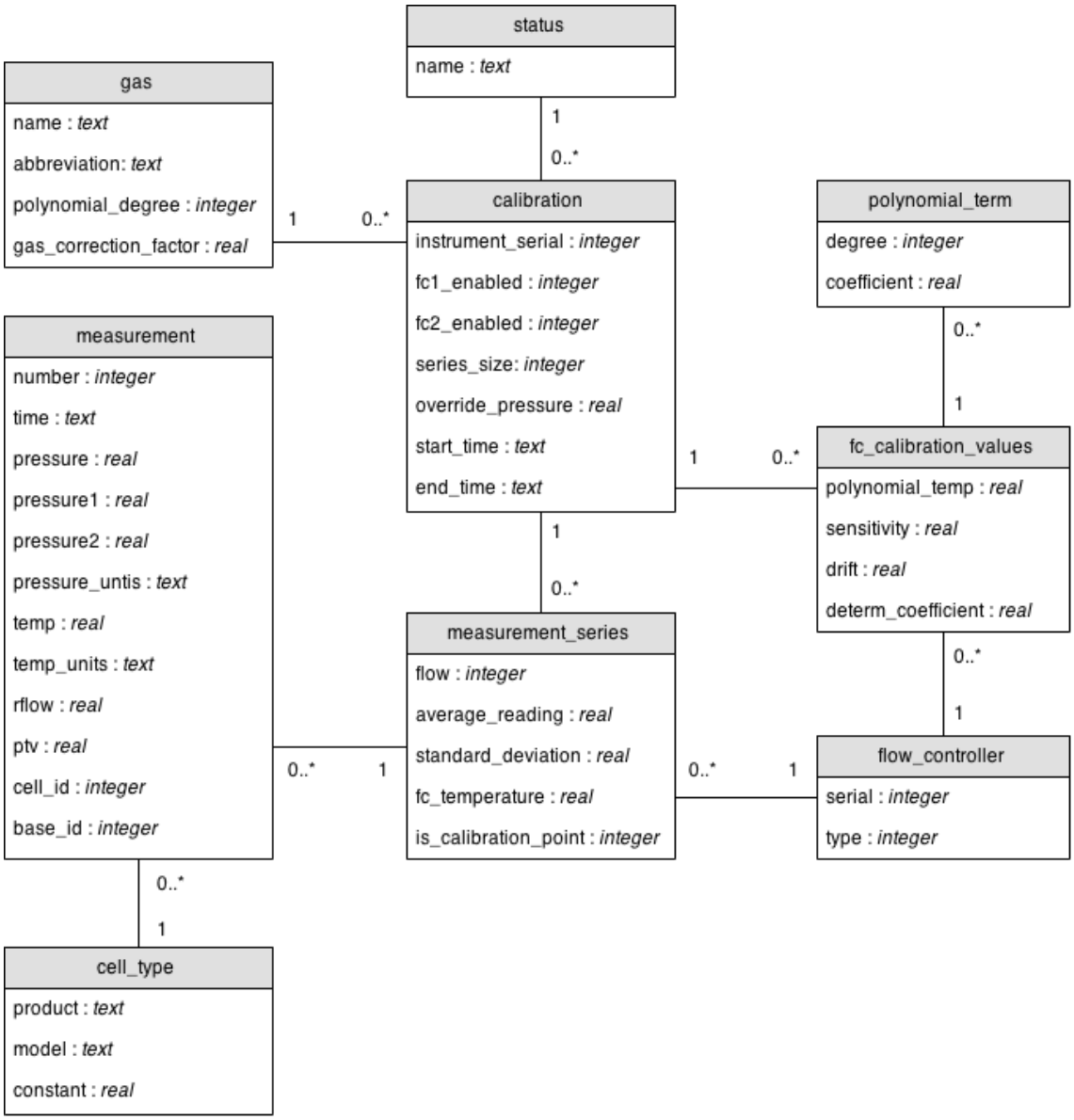
Rakenduse koodis kasutatakse võimalikult palju Qt võimalusi. Kõige iseloomulikum ja unikaalsem Qt võimalus on signaal ja pesa mehhanism. Mehhanism võimaldab sündmustepõhist programmi täitmist ning komponentide vahelist suhtlemist [7].

Rakenduse koodis on laialdaselt kasutusel muster, kus signaali sidumine pesaga toimub pesa omava objekti konstruktoris. Sellise meetodi korral pesa sisaldav objekt on teadlik signaali saatva objekti olemasolust, kuid mitte vastupidi. Sellise madala sidestusega (*loose coupling*) ehitus teeb rakenduse erinevad komponendid vajadusel lihtsamini asendatavateks.

4.2 Andmebaas

Rakendus salvestab kalibreerimise käigus tekkivad olulised andmed andmebaasi. Olulisteks andmeteks on kasutaja poolt valitud kalibreerimise tingimused, mõõtmiste tulemused ning nendega kaasnevad metaandmed, kuluregulaatori termokompensatsiooni parameetrid ning kalibreerimise tulemusena leitud polünoom. Loetletud andmeid kasutatakse kalibreerimise aruande genereerimiseks.

Järgnevalt esitatakse andmebaasi struktuur UML diagrammina joonisel 12.



Joonis 12. Andmebaasi struktuur.

Diagrammis on välja toodud andmebaasi tabelid ja nende vahelised seosed. Andmebaasi tabelite tähendused on kirjeldatud tabelis 4.

Tabel 4. Andmebaasi tabelite semantika.

Tabel	**Semantika**
calibration	Sisaldab andmeid kalibreeringu tingimuste kohta ning muid meta-andmeid nagu kalibreerimise algus- ja lõpuaeg.
cell_type	Etalonina on võimalik kasutada ka teisi Mesa Labs gaasikulu mõõteseadmeid ning ka erinevaid mõõtepuure. Tabel sisaldab mõõteseadmete ja mõõtepuuride kombinatsioonide eripäradest tingitud konstante, mida kasutatakse gaasi kulu arvutusel.
fc_calibration_values	Vahetabel, mis sisaldab konkreetse kuluregulaatori kalibreerimisega seotuid andmeid nagu termokompensatsiooni parameetrid.
flow_controller	Tabel sisaldab kulureegulaatoreid nende tüübi ja tootenumbriga.
gas	Gaaside eripäradest tingitud suurused ja konstandid, mida arvestatakse gaasikulu arvutuses.
measurement	Tabel sisaldab etalonmõõteseadmest saadud mõõtetulemust ning selle meta-andmeid.
measurement_series	Mõõtetulemuse juhuslikku viga silmas pidades tehakse samal kulul korduvaid mõõtmisi, millest arvutatakse aritmeetiline keskmine. Tabel komplekteerib samal kulul mõõtmised mõõteseeriaks.
polynomial_term	Kalibreerimise tulemusena saadud polünoom salvestakse liikmete kaupa antud tabelisse.
status	Tabel sisaldab võimalikke kalibreeringu olekuid.

Andmebaasi hoitakse kohaliku failina rakenduse juurkaustas. Kui andmebaasi faili juurkaustast ei leita siis luuakse see fail ja genereeritakse sinna andmebaasi struktuur ning sisestatakse algandmed vastava SQL päringuga.

Kõik andmebaasi päringud ning andmebaasi haldamine on implementeeritud *DatabaseManager* klassis. *DatabaseManager* käitub tarkvaralise liidesena võimaldades teistele rakenduse komponentidele lihtsustatult andmebaasiga suhtlemist.

Lihtsustamaks rakenduse sisest andmete transporti kasutatakse andmeklasse, mis on vastavuses samanimeliste andmebaasi tabelitega. Joonisel 13 esitatakse *DatabaseManager*-i funktsioon, milles tehakse andmebaasi päring ning tulemus laetakse ümber andmeklassi põhjal loodud objekti. Antud päringus küsitakse kuluregulaatori andmeid. Sarnaselt on koostatud kõik andmebaasi päringud.

```
FlowController * DatabaseManager::getFlowController(int id)
{
    FlowController *flowController = 0;
    QSqlQuery query;

    query.prepare(
        "SELECT id, serial, type "
        "FROM flow_controller "
        "WHERE id = :id"
    );
    query.bindValue(":id", id);

    if(query.exec()) {
        if(query.next()) {
            flowController = new FlowController();
            flowController->setId(query.value(0).toInt());
            flowController->setSerial(query.value(1).toInt());
            flowController->setType(query.value(2).toInt());
        }
    }

    return flowController;
}
```

Joonis 13. Andmebaasi päring.

4.3 Suhtlus üle järjestikliidese

Rakendus juhib lämmastiku analüsaatorit N4040 ja etalonmõõteseadet ML-500 ja saab nendelt seadmetelt vastu olekupõhiseid andmeid läbi RS-232 järjestikliidese. Seadmetega kommunikatsiooni rakenduse välisel tasemel käsitletakse peatükkides 2.4, 2.4.1 ja 2.4.2.

Kummagi seadmega kommunikatsiooni kirjeldab eraldi klass, mis võtab arvesse vastava seadme protokollilisi iseärasusi. Lämmastikuanalüsaatoriga suhtlemise loogika on implementeeritud *InstrumentConnection* klassis ning suhtlus etalonmõõteseadmega *CalibrationConnection* klassis. Antud klasse võib vaadelda kui teenuseid, mida osutatakse komponentidele rakenduse ülemistes kihtides.

Ehituselt on klassid väga sarnased kuna mõlemad seadmed suhtlevad üle järjestikliidese ja kasutavad selleks ASCII kodeeringus teksti. Klassid kasutavad ühenduse loomiseks Qt moodulit QextSerialPort. Komponentide omavahelist sarnasust silmas pidades käsitletakse nende ehitust järgnevates alapeatükkides koos *InstumentConnection* näitel.

4.3.1 Andmete vastuvõtmine

QextSerialPort-i kasutates luuakse sidekanal vastava seadmega suhtlemiseks. Sidekanal võimaldab signaal ja pesa mehhanismi abil sündmustepõhiselt andmeid vastuvõtta. Saabuvaid andmed loetakse sisse rea kaupa, mis seejärel koheselt töödeldakse vastavalt joonisele 14. Joonisel esitatakse näide N4040 seerianumbri vastuvõtmisest.

```
void InstrumentConnection::processData(QByteArray *incoming)
{
    unsigned char cmd = incoming->at(0);
    QByteArray data = incoming->right(incoming->length() - 1);
    bool ok;

    switch (cmd) {
        case SEND_INSTRUMENT_SERIAL_NB: {
            int serial = data.toInt(&ok, 16);
            if (ok)
                emit instrumentSerialReceived(serial);
            else
                emit invalidData(cmd, data);
            break;
        }

        ...

        default: {
           emit undefinedResponse(cmd);
           break;
        }
    }
}
```

Joonis 14. Vastuvõetud andmete töötlemine.

Andmete töötlemisel identifitseeritakse andmed ning seejärel konverteeritakse need vastavalt. Konverteeritud andmed saadetakse signaal ja pesa mehhanismi abil rakenduse ülemistesse kihtidesse.

4.3.2 Andmete saatmine

Andmete saatmiseks koostatakse saadetav käsk, mis koosneb käsu identifikaatorist, valikulistest andmetest ning käsu lõpu tähistusest. Käsust olenevalt konverteeritakse

vajadusel kaasa pandud andmed kujule, mida vastuvõtva seadme mikrokontroller soovib saada.

Rakendus on suuteline saatma andmeid kiiremini kui seadmete mikrokontrollerid suudavad neid vastu võtta. Selle probleemi lahendamiseks asetatakse saadetavad käsud kõigepealt järjekorda. Järjekorda asetamisel käivitatakse taimer, mis teatud intervalli tagant kutsub välja joonisel 15 esitatud funktsiooni *onReadyWrite()*.

```
void InstrumentConnection::onReadyWrite()
{
    if (!buffer->isEmpty()) {
        QString cmd = buffer->takeFirst();
        port->write(cmd.toAscii(), cmd.length());
    }
    else {
        timer->stop();
    }
}
```

Joonis 15. Käsu saatmine järjekorra alusel.

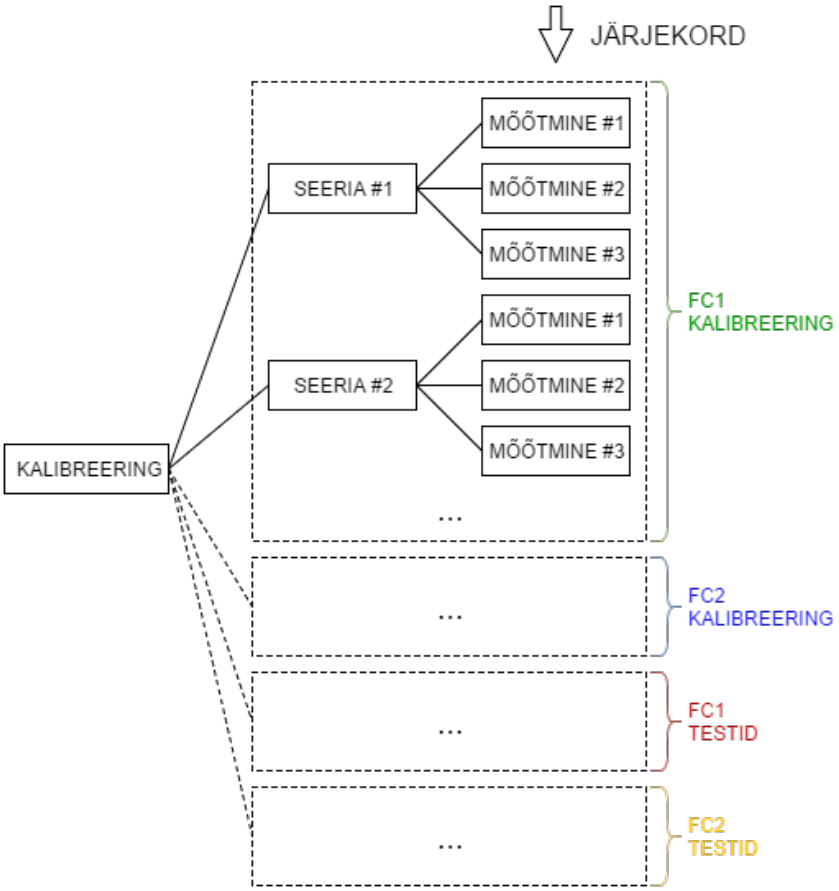
Selline lahendus annab vastuvõtvale seadmele taimeri intervalli kestvuse jagu aega eelmist käsku täita enne kui järgnev käsk peale tuleb.

4.4 Kalibreerimine tarkvaraliselt

Tarkvaralises vaates on kuluregulaatorite kalibreerimiseks tarvis koordineerida N4040 ja ML-500 tööd, registreerida seadmetest saadud andmed ning kogutud andmete kogumi põhjal arvutada kuluregulaatorite näitu korrigeerivad polünoomid. Kalibreerimise metoodikat käsitletatakse peatükis 2.5.

Kalibreerimise läbiviimise loogika on implementeeritud *MeasurementHandler*-i klassis. Teised äriloogika kihi klassid joonisel 11 on *MeasurementHandler* suhtes toetava iseloomuga. Järgnevalt kirjeldatakse kalibreerimise läbiviimist ning antud klasside osa selles tegevusskeemis.

Vastavalt kasutaja poolt määratud tingimustele genereeritakse objektide puu, mida illustreerib joonis 16. Antud objektide puu vastab andmebaasi tabelitele *calibration*, *measurement_series* ning *measurement* joonisel 12. Selline üks ühele vastavus andmebaasi tabelitega võimaldab andmeid vähese vaevaga andmebaasi sisestada. Puu käiakse läbi ning koostatakse selle järgi mõõtmiste järjekord, mille haldamisega tegeletakse *MeasurementQueue* klassis.



Joonis 16. Kalibreeringu puust koostatakse mõõtmiste järjekord.

Mõõtetulemusteta mõõtmisi järjekorrast tsükliliselt ükshaaval võttes otsustatakse konkreetse mõõtmise põhjal ettevalmistavad tegevused, mis eelnevad reaalsele mõõtmisprotseduurile kui edastatakse ML-500-le käsk mõõtmise alustamiseks.

Nendeks tegevusteks võivad olla:

- aktiivse kuluregulaatori vahetus
- uue gaasi kulu etteseadmine aktiivsele kuluregulaatorile
- gaasi kulu stabiliseerumise ootamine
- aktiivse kuluregulaatori näidu registreerimine
- polünoomi(de) arvutus ning sisestus kuluregulaatori(te)sse

Kuluregulaatori polünoomi arvutamiseks tarviliku kuluanduri signaalpinge leidmiseks jälgitakse kuluregulaatori näitu vahetult enne mõõtmise alustamist. Kogudes teatud hulga näite puhvrisse, on võimalik näha kui suur on näidu kõikumine. Samuti arvututatakse kogutud näitude aritmeetiline keskmine, mille järgi arvutatakse kuluanduri signaalpinge konkreetses mõõtepunktis. Eelkirjeldatud funktsionaalsust võimaldab *FlowBuffer* klass.

Selleks, et eelmine mõõtmine järgnevat ei mõjutaks, antakse gaasivool aega stabiliseerumiseks. Stabiliseerumine on eriti oluline kulu muutumisel, mis juhul antakse ekstra aega. Seda võimaldab teha *ElapsedTimer* klass. Samuti kasutatakse antud taimerit veaolukorra kontrolliks juhul kui ML-500-ni jõudev gaasi kulu on liiga väike või olematu. Sellisel juhul taimer jookseb nulli ning kalibreerimine katkestatakse.

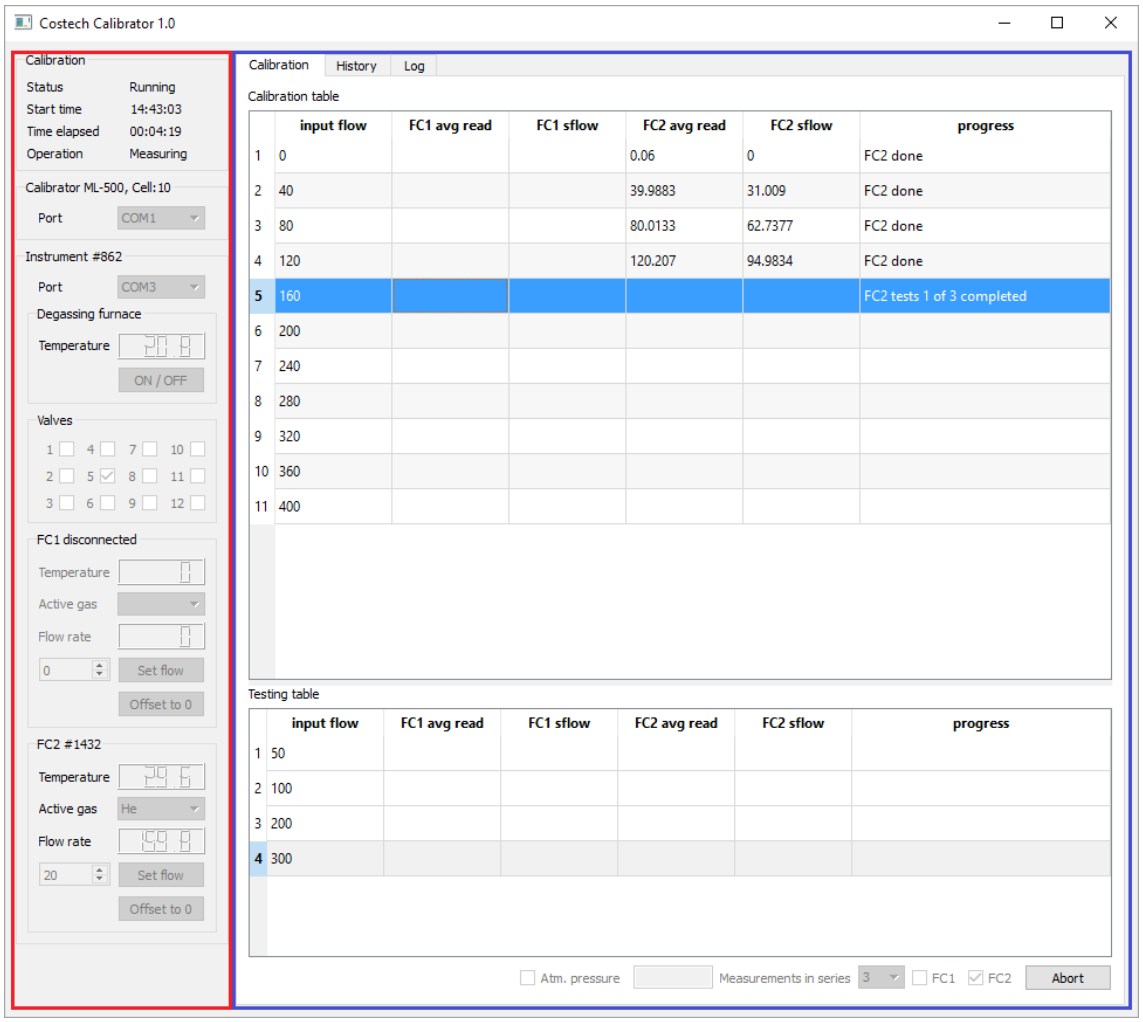
ML-500-st mõõtetulemuse saabudes salvestatakse see järjekorras aktiivsesse mõõtmise objekti ning koheselt ka andmebaasi. Kui mõõtepunktis on kõik mõõtmised tehtud arvutatakse selle seeria standardiseeritud aritmeetiline keskmine kulu kasutades arvutusfunktsioone *Calculator* klassis.

Kuluregulaatorite polünoomid arvutatakse kasutades mitmese lineaarse regressioonanalüüsiga funktsiooni sobitamist võimaldava GSL teegi funktsiooniga *gsl_multifit_linear()* [14]. Funktsiooni kasutamise viis esitatakse lähtekoodiga lisas 1.

Mõõtetulemustega täidetud puust genereeritakse kalibreerimise lõppedes HTML (*Hyper Text Markup Language*) baasil aruanne, mis saadetakse signaal ja pesa mehhanismi abil kasutajaliidese kihti kuvamiseks.

4.5 Kasutajaliides

Kasutajaliides on üles ehitatud kahe põhipaneeliga, kus ühel nendest on sisu alati nähtav ning teisel on sisu vahekaartide abil navigeeritav. See võimaldab rakenduse funktsionaalsust kasutusjuhu põhiselt jagada aga samal ajal kriitilist infot esitada olenemata millist kasutusjuhtu parajasti täidetakse. Antud paneelid on esiletõstetud joonisel 17 sinise ja punase kastiga.



Joonis 17. Kasutajaliidese põhipaneelid ja kalibreerimise vahekaart.

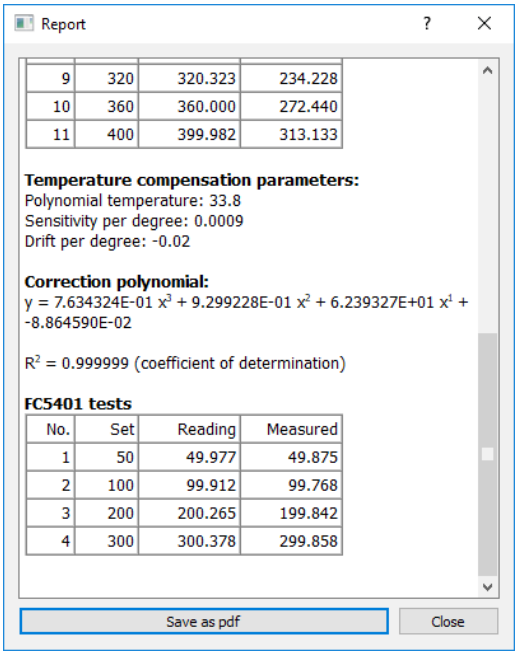
Kriitiline info, mis on vajalik alati nähtaval hoida, on paigutatud sektsioonide kaupa vasakule paneelile (joonisel 17 punasega esiletõstetud). Selles sisaldub kalibreerimise ning seadmete olekupõhine info ning seadmete juhthoovad.

Kalibreerimise vahekaart (Calibration), mis on joonisel 17 parajasti aktiveeritud (sinisega esiletõstetud), koosneb kahest tabelist. Nendest ülemine on mõeldud mõõtepunktidele, mille alusel leitakse kuluregulaatorite polünoomid. Alumine testimise

tabel on mõeldud mõõtepunktidele, millega verifitseeritakse kas kuluregulaatoritelt saadud näit vastab mõõtmiselt saadud tulemustele. Tabelitesse on võimalik kasutajal sisestada esimesse veergu (input flow) mõõtepunkte, teised veerud on mõeldud tulemuste ning mõõtmise staatuse kuvamiseks. Testimise tabeli all on väike riba, mis sisaldab kalibreerimise täiendavaid tingimusi. Riba paremas ääres on nupp, millega alustatakse või katkestatakse kalibreerimine.

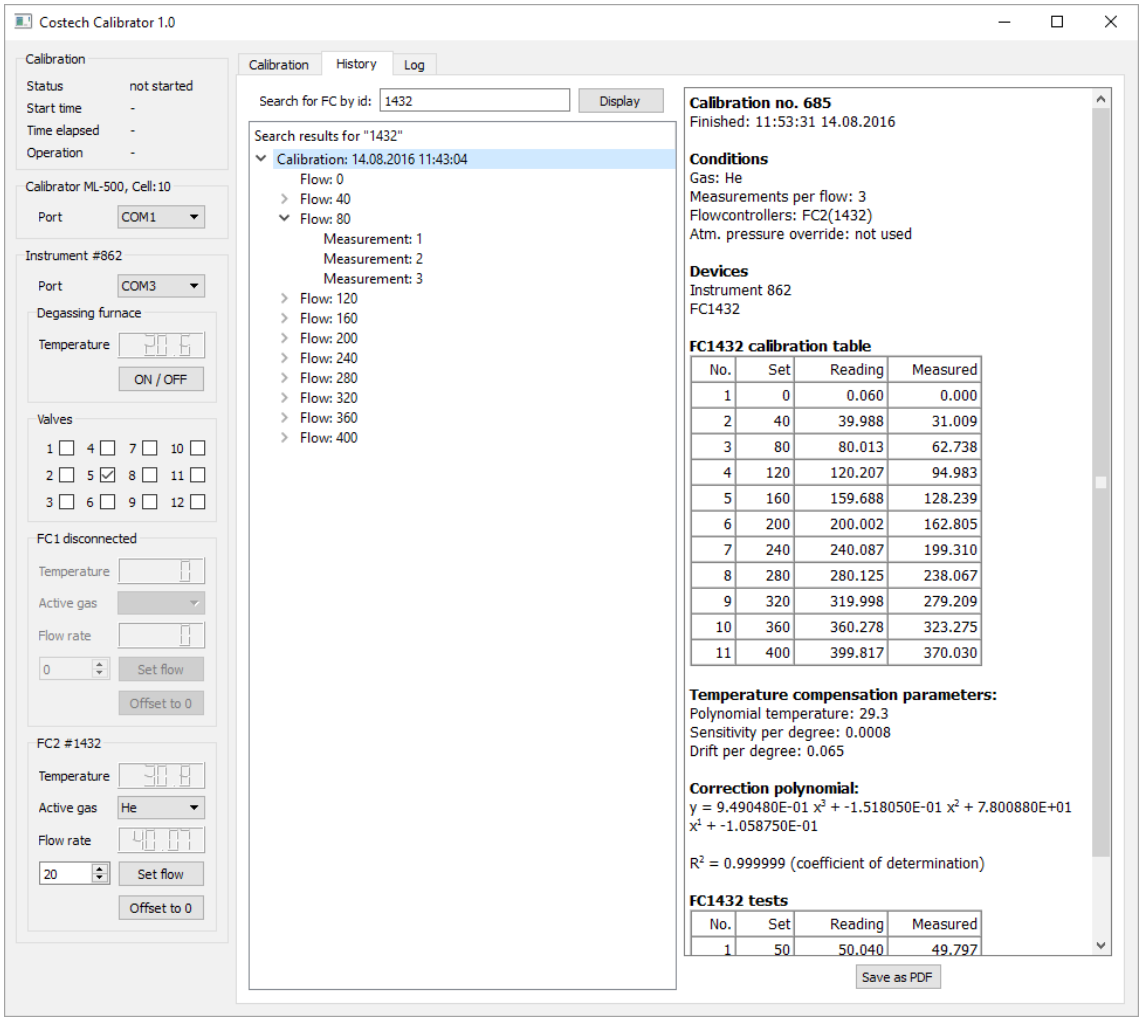
Joonisel 17 esitatud kuvatõmmis näitab kasutajaliidest kalibreerimise olekus, mille kestel vasaku põhipaneeli elemendid, kalibreermise vahekaardi alumise riba elemendid ning tabelid inaktiiveeriakse. See takistab kasutajal automaatsesse kalibreerimise protsessi sekkuda. Protsessi jooksul täiendatakse tabeleid vastavalt läbiviidavatele mõõtmistele andes kasutajale infot edenemisest.

Kalibreerimise edukal lõpetamisel kuvatakse eraldi aknas joonisel 18 aruande dialoog. Aruanne genereeritakse HTML-ina tekstikasti, kuid seda saab ka PDF (*Portable Document Format*) failina salvestada.



Joonis 18: Kalibreerimise lõpetamisel kuvatav aruanne.

Aruandeid saab hiljem kätte ka ajaloo vahekaardilt (History) joonisel 19. Vahekaart koosneb otsinguribast, navigatsioonipuust ning sisu tekstikastist. Otsinguga leitakse kuluregulaatori seerianumbri järgi ajalises järjestuses just selle kuluregulaatoriga läbiviidud kalibreeringud, mis kuvatakse navigeerimiseks mõeldud puuna. Puu elementidele klikkides kuvatakse paremas tekstikastis valitud elemendi kohta käivad andmed. See tagab kasutajale ligipääsu täpsematele andmetele, mis aitab probleemide korral kergemini välja selgitada nende põhjust. Aruanne kuvatakse kalibreeringu elemendi (esiletõstetud helesinisega) alt.



Joonis 19. Ajaloo vahekaart.

Viimasel vahekaardil (Log) kuvatakse logi kõigist tegevustest, mida kalibreerimise käigus rakendus ette võtab. See võimaldab samuti probleemide korral veatuvastust.

5 Kokkuvõte

Käesoleva bakalaureusetöö eesmärgiks oli luua rakendus, mille ülesandeks on lämmastiku analüsaatori Turbo N 4040 kuluregulaatoreid iseseisvalt kalibreerida ning saadud tulemusi arhiveerida. Töö toimus Costech Microanalytical OÜ tellimusel.

Rakendus loodi Qt arendusraamistiku baasil, millega arendati välja:

- komponendid kalibreerimisega seotud seadmetega suhtlemiseks
- funktsionaalsus kalibreerimise läbiviimiseks
- andmebaas tulemuste arhiveerimiseks
- kasutajaliides

Töö tulemusena jõuti eelnevalt kirjeldatud eesmärgini. Valminud rakendus on ettevõttes kasutusel, vähendades oluliselt kuluvat aega kalibreerimise läbiviimiseks.

Seoses lämmastiku analüsaatori elektroonika uuendamisega arendati rakendust käesoleva töö jätkuna edasi. Järgnevalt esitatakse nimekiri juba valminud edasiarendustest, mida käesolevas bakalaureusetöös pole käsitletud:

- Qt versioonilt 4 üleminek 5-le
- RS232 järjestikliidese asemel natiivne USB
- Paralleelkalibreerimine, millega mõõdetakse kuluregulaatorid samaaegselt
- Eeltäidetud tabelid

Kuigi töö kujunes ajamahukamaks kui esialgselt planeeritud, tuleb siiski tõdeda, et Qt platvorm sobib hästi sarnaste projektide elluviimiseks. Samuti on suur kasu laiapõhjalistest teadmistest, kui projektis kohtuvad mitmed erinevad valdkonnad.

Kasutatud kirjandus

[1] Dumas' method - Oxford Reference. [WWW]. http://www.oxfordreference.com/view/10.1093/oi/authority.20110803095734274. (21.05.2018).

[2] Nitrogen / Protein Analyser Model 4040 User Manual Version 2.10. Costech Analytica.

[3] Буц, А. управление газовыми линиями в азотном анализаторе : дипломная работа. Таллин, Таллинский Технический Университет. 2011.

[4] ML-500 User Manual. [WWW]. https://drycal.mesalabs.com/wp-content/uploads/sites/5/2013/12/ML-500-manual_RevJ.pdf. (21.05.2018).

[5] Mesa Met Lab Series Bi-Directional Communications Protocol. [WWW]. http://drycal.mesalabs.com/wp-content/uploads/sites/5/2015/02/MetLabCommunication-Protocol.24FEB2015.pdf. (21.05.2018).

[6] Using the Meta-Object Compiler (moc) | Qt 4.8. [WWW]. http://doc.qt.io/archives/qt-4.8/moc.html. (21.05.2018).

[7] Signals & Slots | Qt Documentation. [WWW]. http://doc.qt.io/qt-4.8/signalsandslots.html. (21.05.2018).

[8] SQL Database Drivers | Qt Documentation. [WWW]. http://doc.qt.io/qt-4.8/sql-driver.html. (21.05.2018).

[9] QtSql Module | Qt Documentation. [WWW]. http://doc.qt.io/qt-4.8/qtsql-module.html. (21.05.2018).

[10] SQLite. [WWW]. https://www.sqlite.org/about.html. (21.05.2018).

[11] QextSerialPort. [WWW]. https://qextserialport.github.io/. (21.05.2018).

[12] GNU Scientific Library. [WWW]. https://www.gnu.org/software/gsl/. (21.05.2018).

[13] Eclipse platform overview. [WWW]. http://help.eclipse.org/oxygen/topic/org.eclipse.platform.doc.user/gettingStarted/intro/overview.htm?cp=0_0. (21.05.2018).

[14] Linear Least-Squares Fitting | GSL 2.4 documentation". [WWW]. https://www.gnu.org/software/gsl/doc/html/lls.html?#c.gsl_multifit_linear. (21.05.2018).

Lisa 1 – Polünoomi arvutusfunktsioon

```
void Calculator::polynomialfit(int obs, int degree, double *dx, double *dy,
double *store, double *R2)
{
    gsl_multifit_linear_workspace *ws;
    gsl_matrix *cov;
    gsl_matrix *X;
    gsl_vector *y;
    gsl_vector *c;
    double chisq;
    double tss;

    int i, j;
    X = gsl_matrix_alloc(obs, degree);
    y = gsl_vector_alloc(obs);
    c = gsl_vector_alloc(degree);
    cov = gsl_matrix_alloc(degree, degree);

    for(i = 0; i < obs; i++) {
        gsl_matrix_set(X, i, 0, 1.0);
        for(j = 0; j < degree; j++) {
            gsl_matrix_set(X, i, j, pow(dx[i], j));
        }
        gsl_vector_set(y, i, dy[i]);
    }

    ws = gsl_multifit_linear_alloc(obs, degree);
    gsl_multifit_linear(X, y, c, cov, &chisq, ws);

    // Store result
    for(i = 0; i < degree; i++) {
        store[i] = gsl_vector_get(c, i);
    }

    // Coefficient of determination
    tss = gsl_stats_tss(dy, 1, obs);
    *R2 = 1 - chisq / tss;

    gsl_multifit_linear_free(ws);
    gsl_matrix_free(X);
    gsl_matrix_free(cov);
    gsl_vector_free(y);
    gsl_vector_free(c);
}
```