KAUNO TECHNOLOGIJOS UNIVERSITETAS
ELEKTROS IR ELEKTRONIKOS FAKULTETAS

Vilius Jakas

MIELIŲ BIOPROCESO EKSPERIMENTINIŲ PARAMETRŲ IDENTIFIKAVIMAS IR VALDYMO PLANAVIMAS

Baigiamasis magistro projektas

Vadovas
Doc. dr. Renaldas Urniežius

KAUNAS, 2018

KAUNO TECHNOLOGIJOS UNIVERSITETAS
ELEKTROS IR ELEKTRONIKOS FAKULTETAS
AUTOMATIKOS KATEDRA

MIELIŲ BIOPROCESO EKSPERIMENTINIŲ PARAMETRŲ IDENTIFIKAVIMAS IR VALDYMO PLANAVIMAS

Baigiamasis magistro projektas

Valdymo technologijos (621H66001)

Vadovas
Doc. dr. Renaldas Urniežius

Recenzentas
Prof. dr. Vytautas Galvanauskas

Projektą atliko
Vilius Jakas

KAUNAS, 2018

KAUNO TECHNOLOGIJOS UNIVERSITETAS

ELEKTROS IR ELEKTRONIKOS FAKULTETAS

(Fakultetas)

Vilius Jakas

(Studento vardas, pavardė)

Valdymo technologijos (621H66001)

(Studijų programos pavadinimas, kodas)

Baigiamojo projekto „Mielių bioproceso eksperimentinių parametrų identifikavimas ir valdymo planavimas“

AKADEMINIO SĄŽININGUMO DEKLARACIJA

20 18 m. Gegužės 18 d.

Kaunas

Patvirtinu, kad mano **Viliaus Jako** baigiamasis projektas tema **„Mielių bioproceso eksperimentinių parametrų identifikavimas ir valdymo planavimas“** yra parašytas visiškai savarankiškai, o visi pateikti duomenys ar tyrimų rezultatai yra teisingi ir gauti sąžiningai. Šiame darbe nei viena dalis nėra plagijuota nuo jokių spausdintinių ar internetinių šaltinių, visos kitų šaltinių tiesioginės ir netiesioginės citatos nurodytos literatūros nuorodose. Įstatymų nenumatytų piniginių sumų už šį darbą niekam nesu mokėjęs.

Aš suprantu, kad išaiškėjus nesąžiningumo faktui, man bus taikomos nuobaudos, remiantis Kauno technologijos universitete galiojančia tvarka.

(vardą ir pavardę įrašyti ranka) *(parašas)*

Jakas, Vilius. Mielių bioproceso eksperimentinių parametrų identifikavimas ir valdymo planavimas. *Valdymo sistemų magistro baigiamasis projektas* / vadovas doc. dr. Renaldas Urniežius; Kauno technologijos universitetas, Elektros ir elektronikos fakultetas, Automatikos katedra.

Mokslo kryptis ir sritis: Elektros ir elektronikos inžinerija, Technologiniai mokslai

Reikšminiai žodžiai: *biotechnologija, bioreaktorius, matematinis modelis, identifikavimas, etanolis.*

Kaunas, 2018.69 psl.

SANTRAUKA

Biotechnologinių procesų atlikimui ir galutinio produkto kokybei keliami aukšti reikalavimai. Vykdomų fermentacijų metu sukaupiami dideli duomenų kiekiai, tačiau jie nėra pakankamai išanalizuojami, kad būtų galima pagerinti būsimas fermentacijas.

Darbe aprašomas ir išplėtojamas parinktas mielių augimo matematinis modelis. Aprašomas laboratorijoje naudojamas aerobinis bioreaktorius ir išplečiamos jo valdymo įrankių funkcijos. Sukalibruojami matavimo prietaisai.

Sudaromas dalies matematinio modelio lygčių parametrų identifikavimo iš realių duomenų aprašymas ir jie identifikuojami. Palyginama matematinio modelio ir realios fermentacijos eiga.

Pasiūlomas substrato tiekimo profilis, su kuriuo, manomai, būtų buvusi didesnė biomasės išeiga fermentacijos pabaigoje.

Jakas, Vilius. Identification of yeast bioprocess experimental parameters and control planning. Final project of control systems master / supervisor doc. dr. Renaldas Urniežius; Kaunas University of Technology, Faculty of Electrical and Electronics Engineering, department of Automation.

Research area and field: Electrical and Electronics Engineering, Technological Sciences
Key words: *biotechnology, bioreactor, mathematical model, identification, ethanol.*
Kaunas, 2018. 69 p.

SUMMARY

Biotechnology process and quality of the final product have to meet strict quality requirements. During the fermentation process large amount of data is gathered, but it is not fully examined to extract information for the improvement of fermentation in the future.

This paper describes mathematical model of yeast cultivation and develops it further. Defining the aerobic bioreactor used in the laboratory, paper expands fermentation control tools by adding new functions. Conducted calibration of measurement devices

Developed partial identification of mathematical model parameters from real world data. Results of the simulated mathematical model and the behavior of real world bioreactor is compared.

Proposal of the new substrate feeding profile, which may have led to higher biomass density with the same amount of glucose.

Turinys

ĮVADAS

Skaitmeninio modeliavimo metodai leidžia sumažinti vykdomų fermentacijų kaštus. Šie skaitmeninio modeliavimo įrankiai gali būti sudaromi naudojant mielių augimą aprašančius matematinius modelius. Matematiniams modeliams būtina parinkti teisingus parametrus, nes kitaip galima patirti nuostolių, sulėtinti mokslinių tyrimų eigą.

Esant aukštiems fermentacijų kaštams stinga resursų vykdyti bandomąsias fermentacijas, geriausias rezultatas privalo būti pasiekiamas iš karto[1]. Sudaromi duomenų analizavimo ir modelių parametrų identifikavimo metodai, kurie leistų padidinti jau surinktų duomenų vertę taip optimizuojant tyrimų kaštus.

Naujų funkcijų ir matavimo metodų pritaikymas leidžia pagerinti operatorių darbo sąlygas gaunant aukščiausios kokybės galutinį produktą. Pažangūs valdymo įrankiai leidžia tirti ir sudaryti optimalius maistinių medžiagų (substrato) tiekimo profilius, kurių dėka gaunama aukščiausia biomasės išeiga.

Darbo tikslas. Identifikuoti mielių bioproceso parametrus iš eksperimentinių duomenų ir pagal juos sumodeliuoti optimalų gliukozės pamaitinimą siekiant didžiausios biomasės išeigos sunaudojant tą patį gliukozės kiekį.

Darbo uždaviniai:

1. Išplėtoti matematinį mielių augimo modelį. Aprašyti mielių augimo procesus ir jų auginimui naudojamą įrangą.
2. Realizuoti matematinį modelį kompiuterinėje modeliavimo programoje.
3. Išplėsti naudojamos aparatinės valdymo įrangos funkcijas, sukalibruoti matavimo prietaisus.
4. Identifikuoti matematinio modelio parametrus ir palyginti modelį su realiai veikiančia sistema.
5. Sudaryti optimalų pamaitinimo algoritmą sunaudojant tą patį gliukozės kiekį.

1. MIELĖS IR MIELIŲ AUGIMAS

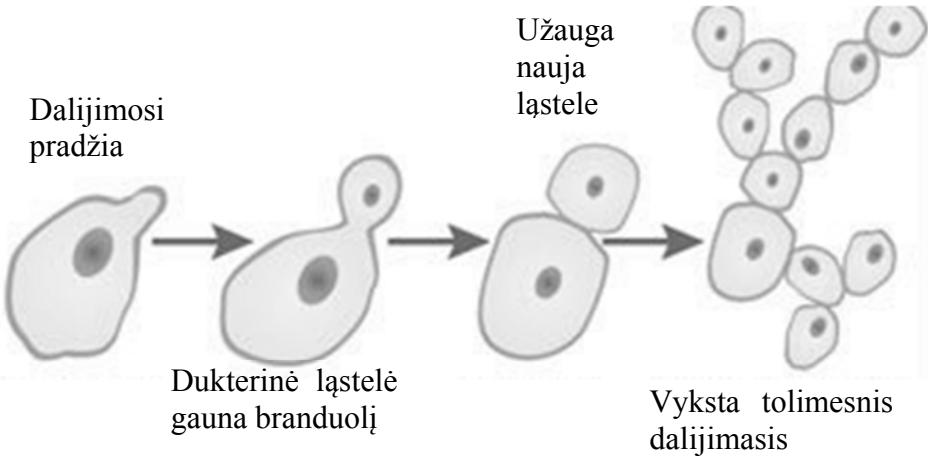
1.1 Mielių kultūra

Mielės – vienaląsčiai grybeliai, kurie naudojami alaus ir maisto pramonėje, atliekant mokslinius tyrimus. Mielėse randama daug B grupės vitaminų, naudingųjų fermentų. Jos naudojamos maisto papildų, citrinų rūgšties gamyboje[2].

Maisto ir alkoholio pramonėje dažniausiai naudojamos alaus mielės (*lot. Saccharomyces cerevisiae)*. Šis grybelių porūšis plačiai naudojamas moksliniuose tyrimuose dėl savo žemos kainos ir savybių panašių į žmogaus organizmą arba e.coli bakterijas. Šių grybelių fermentacija paprastesnė lyginant su e.coli.

Alaus mielės auga palyginus greitai, jų kiekio padvigubėjimo periodas – 1-2 valandos[3]. Biotechnologiniuose procesuose naudojant alaus mieles galima sąlyginai greitai užauginti didelį biomasės kiekį, iš kurio išskiriamos naudingosios medžiagos. Tai patogu vykdant mokslinius tyrumus ir kuriant automatines augimo palaikymo sistemas, kai valdymo sistemos trūkumai gali būti greitai identifikuoti ir pradedama nauja fermentacija.

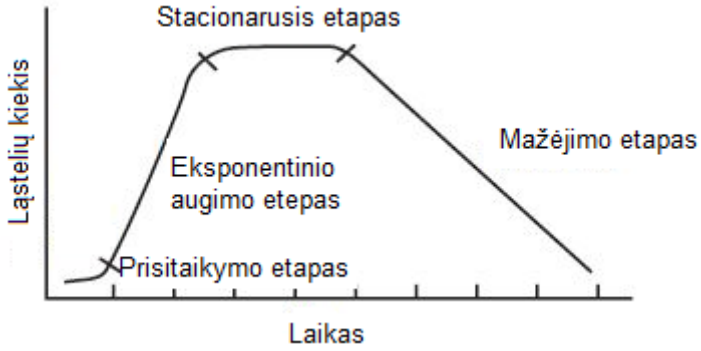
Alaus mielės dauginasi vegetatyviniu pumpuravimo būdu. Tai yra nelytinis dauginimosi būdas, kai ant motininės ląstelės susiformuoja naujos ląstelės. Pirmiausia ląstelėje esantis branduolys pasidalina į lygias dalis, kuriose išlieka pilni ir identiški chromosomų rinkiniai. Naujai susidaręs branduolys patenka į besiformuojantį iškyšulį ląstelės šone. Šis iškyšulys auga ir pasiekęs motininės ląstelės dydį tampa savarankiška ląstele. Naujai susiformavusios ląstelės būna genetiškai identiškos motininei ląstelei, nes jos susidaro pasidalinant motininės ląstelės branduoliui. Mielių ląstelių dalijimasis pumpuravimo būdu pavaizduota 1.1 paveikslėlyje.



1.1 pav. Mielių ląstelių dalijimasis[4].

1.2 Mielių augimo etapai

Mielių gyvybingųjų ląstelių augime išskiriami keturi etapai: prisitaikymo, eksponentinio augimo, stacionarusis ir mažėjimo. Kiekvieno etapo metu terpėje ir ląstelėse vyksta skirtingi procesai, skiriasi ląstelių augimo greitis. Grafiškai atvaizduojant ląstelių augimo greitį vertikalioje ašyje žymimas ląstelių kiekis logaritminėje ($\log_{10}$) skalėje, o horizontalioje – laikas nuo fermentacijos pradžios. Etapų trukmė priklauso nuo mielių ir terpės sąlygų. Šie etapai pavaizduoti 1.2 paveikslėlyje.



1.2 pav. Mielių augimo etapai.

Prisitaikymo etapo pradžioje ląstelės adaptuojasi prie aplinkos jas patalpinus į naują, dažniausiai didesnę terpę. Jos nesidalina, biomasės augimas nevyksta. Laikas, reikalingas ląstelėms prisitaikyti, skiriasi ir priklauso nuo konkrečios kultūros ir terpės sąlygų. Šio etapo pabaigoje prasideda ląstelių dalijimasis – tai eksponentinio augimo etapo pradžia. Ląstelių skaičius didėja eksponentiškai ir pastoviu greičiu, todėl šis etapas vadinamas eksponentinio augimo etapu. Augimas vyksta tol, kol terpėje pakanka deguonies, gliukozės ir kitų maistinių medžiagų bei yra palankios terpės sąlygos (rūgštingumas, temperatūra). Biomasės augimas sulėtėja pasiekus aukštą biomasės koncentraciją, tada terpėja atsiranda vis daugiau ląstelių išskiriamų atliekų. Stacionariajame etape ląstelių skaičius nebedidėja, naujai atsiradusių ląstelių skaičius lygus žuvusių skaičiui. Medicinos pramonėje šiame etape formuojasi kai kurie antibiotikai. Dėl maistinių medžiagų trukumo gali būti pastebimas ląstelių DNR kodo pasikeitimas. Tęsiant fermentaciją prasidėtų ląstelių mažėjimo etapas, kai gyvybingų ląstelių skaičius pradėtų tiesiškai mažėti[5][6].

Atliekant fermentacijas siekiama užauginti kuo didesnį biomasės kiekį, todėl stacionarusis ir mažėjimo etapai nepageidaujami. Parenkant maistingųjų medžiagų tiekimo ir terpės aplinkos parametrų palaikymo planus siekiama prailginto eksponentinio augimo etapą, kurio metų greičiausiai auga biomasė.

1.3 Fermentacijų vykdymo metodai

Vystantis farmacijos, fermentacijos, žemės ūkio, chemijos ir energetikos sritims biotechnologinių procesų optimizavimas ir automatizavimas - aktualus pramonės bei mokslinių tyrimų objektas[2]. Dažniausiai fermentacijas vykdo operatoriai, kurie pagal parengtą valdymo planą ir matuojamas proceso vertes valdo maistinių medžiagų tiekimą, terpės temperatūrą, ištirpusio deguonies kiekį ir kitus parametrus. Tai kruopštus ir daug patirties reikalaujantis darbas. Kuriamos tokios valdymo sistemos, kurios gebėtų palengvinti operatorių darbą ir sumažinti gamybos kaštus išlaikant tokią pat arba dar geresnę galutinio produkto kokybę. Pramonėje, dažniausiai, atliekamos trijų rūšių fermentacijos.

Pertraukiama fermentacija. Visas substratas su maistingomis medžiagomis patalpinamas į terpę prieš fermentaciją, o gautas produktas surenkamas po jos. Mažiausia riziką užkratui patekti į terpę yra vykdant šią fermentacija. Esant ribotam maistinių medžiagų kiekiui sutrumpėja eksponentinio augimo etapas ir galutinis biomasės kiekis. Šis būdas tinkamiausias vykdyti mažos apimties fermentacijas.

Pertraukiama fermentacija su pamaitinimu. Substratas tiekiamas į terpę pagal iš anksto sudarytą tiekimo algoritmą arba pagal grįžtamojo ryšio signalus. PID, *„Fuzzy“* ir kiti klasikiniai reguliatoriai netinkami tokios fermentacijos valdymui dėl proceso netiesiškumo ir atsirandančių augimą limituojančių faktorių. Pramonėje, dažniausiai, fermentaciją kontroliuoja operatoriai.

Nepertraukiama fermentacija. Substratas tiekiamas į terpę, o tuo pačiu metu fermentacijos produktas surenkamas iš terpės. Didžiausia riziką užkratui patekti į terpę yra vykdant šią fermentaciją. Iš terpės surenkant produktą kartu pašalinamos ir ląstelių išskiriamos atliekos. Augimą ribojančių veiksnių įtaka nepasireiškia, nes ląstelių kiekis terpėje išlieka pastovus. Nepertraukiamos fermentacijos vyksta palyginti sudėtingos konstrukcijos bioreaktoriuose, kurių savikaina ir eksploatacijos kaštai aukštesni nei pertraukiamos ir pertraukiamos su pamaitinimu fermentacijos metu.

Kiekvienas fermentacijos atlikimo metodas turi tam tikrų trūkumų ir privalumų, todėl būtina kruopščiai apgalvoti fermentacijai keliamus tikslus ir pagal juos parinkti tinkamiausią metodą. Šių metodų ypatybės pateiktos 1.1 lentelėje.

1.1 lentelė. Skirtingų fermentacijos metodų ypatybės[7]

	Pertraukiama fermentacija	**Pertraukiama fermentacija su pamaitinimu**	**Nepertraukiama fermentacija**
Bioreaktoriaus konstrukcija	Uždara	Dalinai uždara	Atvira
Substrato tiekimas	Nevykdomas	Vykdomas	Vykdomas
Terpės tūris	Pastovus	Didėja	Pastovus
Atliekų pašalinimas	Nevykdomas	Nevykdomas	Vykdomas
Užkrato rizika	Minimali	Vidutinė	Aukšta
Augimo etapai	Prisitaikymo Eksponentinio augimo Stacionarusis Mažėjimo	Prisitaikymo Eksponentinio augimo Stacionarusis Mažėjimo	Prisitaikymo Eksponentinio augimo
Ląstelių tankis	Kinta	Kinta	Nekinta
Produkto išeiga	Žema	Vidutinė	Aukšta

2. TEORINIS TYRIMAS

2.1 Pasirinktas matematinis modelis

Alaus mielių (*lot. Saccharomyces cerevisiae)* augimas, esant skirtingiems metabolizmo keliams, aprašomas 2.1-2.3 cheminių reakcijų lygtimis. Kiekvienas šių metabolizmo kelių turi skirtingus santykinius augimo greičius, kurie nulemia biomasės, gliukozės, etanolio, ištirpusio deguonies bei anglies dioksido koncentracijų kitimo greičius. Šis matematinis modelis[8][9] detaliai aprašomas, pertvarkomas ir pritaikomas laboratorijoje naudojamam bioreaktoriaus veikimui modeliuoti.

2.1.1 Cheminių reakcijų lygtys

1. Aerobinis biomasės augimas naudojant gliukozę:

$$C_6H_{12}O_6 + 6O_2 \rightarrow 6CO_2 + 6H_2O + (16\text{-}28)\ \text{ATP} \tag{2.1}$$

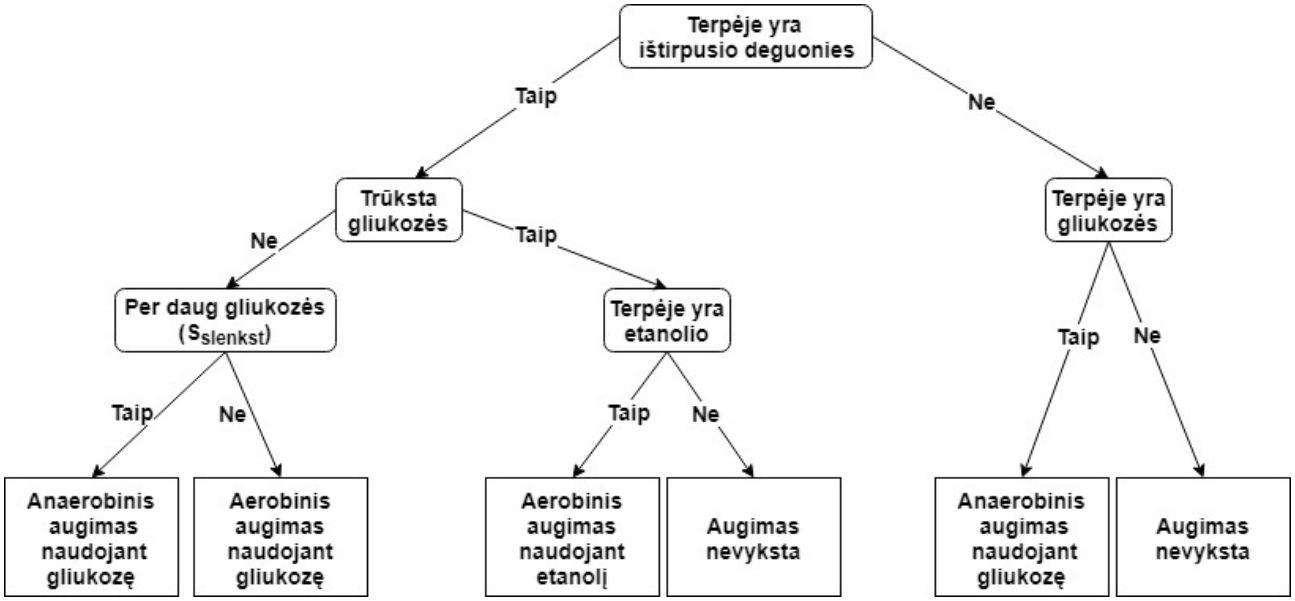
2. Anaerobinis biomasės augimas naudojant gliukozę (rūgimas):

$$C_6H_{12}O_6 \rightarrow 2C_2H_5OH + 2CO_2 + 2\ \text{ATP} \tag{2.2}$$

3. Aerobinis biomasės augimas naudojant etanolį:

$$C_2H_5OH + 3O_2 \rightarrow 2CO_2 + 3H_2O + (6\text{-}11)\ \text{ATP} \tag{2.3}$$

Bioreaktorius turi maišymo ir aeravimo sraigtą, todėl daroma prielaida, kad terpė yra homogeniška. Terpėje esant ištirpusio deguonies ir gliukozės koncentracijai neviršijant tam tikros slenkstinės gliukozės koncentracijos $S_{slenkst.}$ vyksta aerobinis biomasės augimas naudojant gliukozę. Šis augimo būdas yra labiausiai pageidaujamas, jo metu susidaro didžiausias ląstelių energijos šaltinio – Adenozino trifosfato (ATP) kiekis. Trūkstant terpėje ištirpusio deguonies arba esant gliukozės koncentracijai aukštesnei nei $S_{slenkst.}$ pradedamas fermentuoti etanolis. Biomasės augimas sulėtėja, nes ATP išskiriama gerokai mažiau, ląstelės pradeda fermentuoti etanolį, kuris terpėje kaupiasi, dalis jo pašalinama iš bioreaktoriaus kartu su išpučiamu oro srautu. Tai yra anaerobinis biomasės augimas naudojant gliukozę. Terpėje trūkstant gliukozės, tačiau esant ištirpusio deguonies ir etanolio pradeda vykti aerobinis biomasės augimas naudojant etanolį. Grafinis metabolizmo ir augimo greičio parinkimas pavaizduotas 2.1 paveiksle.



2.1 pav. Metabolizmo proceso parinkimas.

2.1.2 Masės balanso lygtys

Bioreaktoriaus matematinis modelis sudarytas masės (medžiagos) balanso lygtimis, kurios pateiktos diferencialinėje formoje. Medžiagos koncentracijos kitimo greitis lygus terpėje susidariusios ir sunaudotos medžiagos skirtumui per valandą. Pavyzdžiui, etanolio koncentracija didėja ląstelėms fermentuojant etanolį ir mažėja – vykstant aerobiniam biomasės augimui naudojant etanolį.

Biomasės koncentracijos dinamikos lygtis:

$$\frac{dX}{dt} = \left(\mu_s^o + \mu_s^f + \mu_e^o - D\right)X \tag{2.4}$$

Biomasės koncentracijos kitimo greitis priklauso nuo biomasės augimo greičių, vykstant skirtingiems metabolizmams sumos. Taip pat kitimo greitį sumažina terpės skiedimas, kuris žymimas *D*. Prieš fermentaciją terpė yra autoklavuojama, todėl mielių ląstelių (biomasės) pamaitinimo tirpale nėra. Pumpuojant pamaitinimo tirpalą didėja terpės tūris, o biomasės kiekis nekinta, todėl biomasės koncentracija terpėje mažėja.

μ_s^o – santykinis biomasės augimas aerobinėse sąlygose naudojant gliukozę;

μ_s^f – santykinis biomasės augimas anaerobinėse sąlygose naudojant gliukozę(rūgimas);

μ_e^o – santykinis biomasės augimas aerobinėse sąlygose naudojant etanolį;

Gliukozės koncentracijos dinamikos lygtis:

$$\frac{dS}{dt} = \left(-\frac{\mu_s^o}{Y_{\frac{X}{S}}^o} - \frac{\mu_s^f}{Y_{\frac{X}{S}}^f} \right) X + (S_f - S)D \tag{2.5}$$

Gliukozės koncentracija mažėja vykstant aerobiniam biomasės augimui naudojant gliukozę ir (arba) fermentuojant etanolį. Pumpuojant pamaitinimo tirpalą, kurio gliukozės koncentracija S_f, terpės tūris didėja skiedimo greičiu, kuris žymimas *D*. Laboratorijoje vykdant fermentacijas naudojamas pamaitinimo tirpalas, kuriame ištirpusios gliukozės koncentracija 0,55kg/l, tankis 1,18kg/l. Dėl šio tirpalo pumpavimo gliukozės koncentracija terpėje didėja.

$Y_{\frac{X}{S}}^o$ – Gliukozės kiekis tenkantis 1 gramui biomasės aerobinėse sąlygose g/g;

$Y_{\frac{X}{S}}^f$ – Gliukozės kiekis tenkantis 1 gramui biomasės anaerobinėse sąlygose g/g;

S_f – Gliukozės koncentracija pamaitinimo tirpale g/l;

Etanolio koncentracijos dinamikos lygtis:

$$\frac{dE}{dt} = \left(\frac{\mu_s^f}{Y_{\frac{X}{E}}^f} - \frac{\mu_e^o}{Y_{\frac{X}{E}}^{oe}} \right) X - DE \tag{2.6}$$

Etanolio koncentracijos kitimo greitis gali būti teigiamas arba neigiamas. Ląstelėms pradėjus fermentuoti etanolį, kai gliukozės tirpalo tiekiamo į terpę greitis didesnis už suvartojamos gliukozės greitį, etanolio koncentracijos kitimo greitis — teigiamas, etanolio koncentracija terpėje didėja. Sumažinus gliukozės tirpalo tiekimą terpėje gliukozės koncentracija pradeda mažėti. Biomasės palaikymui ir augimui pradedamas naudoti etanolis, etanolio koncentracijos kitimo greitis terpėje neigiamas, etanolio koncentracija terpėje mažėja. Darau šias prielaidas: terpė yra homogeniška, vienu metu vyksta arba etanolio gamyba arba suvartojimas. Taip pat etanolio koncentracijos kitimo greitį keičia pumpuojamas pamaitinimo tirpalas skiedimo greičiu *D*.

$Y_{\frac{X}{E}}^f$ – Etanolio kiekis tenkantis 1 gramui biomasės anaerobinėse sąlygose g/g;

$Y_{\frac{X}{E}}^{oe}$ – Etanolio kiekis tenkantis 1 gramui biomasės aerobinėse sąlygose g/g;

Ištirpusio deguonies koncentracijos dinamikos lygtis:

$$\frac{dO}{dt} = \left(-\frac{\mu_s^o}{Y_{\frac{X}{O}}^o} - \frac{\mu_e^o}{Y_{\frac{X}{O}}^{oe}} \right) X - DO + OTR \quad (2.7)$$

Ištirpusio deguonies koncentracijos kitimo greitį siekiama palaikyti teigiamą arba pastovų, o ištirpusio deguonies koncentraciją terpėje ne mažesnę kaip 10%. Deguonis – būtinas ląstelių gyvybiniams procesams palaikyti ir biomasei augti. Trūkstant deguonies ir esant gliukozės koncentracijai aukštesnei nei $S_{slenkst.}$ vyksta etanolio fermentacija nenaudojant deguonies – tai anaerobinis biomasės augimas. Vykstant biomasės augimui aerobinėse sąlygose iš gliukozės arba iš etanolio ištirpusio deguonies koncentracija terpėje mažėja. Laboratorijoje vykdant fermentacijas terpėje ištirpusio deguonies kiekis palaikomas 2.2 poskyryje aprašytais metodais. Ištirpusio deguonies koncentracijos kitimo greitį keičia pumpuojamas pamaitinimo tirpalas skiedimo greičiu, kuris žymimas *D*. Darau prielaidą, kad ištirpusio deguonies koncentracija mažesnė pamaitinimo tirpale nei terpėje, todėl pumpuojant pamatinimo tirpalą ištirpusio deguonies koncentracija terpėje mažėja.

$Y_{\frac{X}{O}}^{O}$ – Deguonies kiekis tenkantis 1 gramui biomasės aerobinėse sąlygose naudojant gliukozę, g/g;

$Y_{\frac{X}{O}}^{Oe}$ – Deguonies kiekis tenkantis 1 gramui biomasės aerobinėse sąlygose naudojant etanolį, g/g;

OTR – Deguonies tiekimo greitis;

Ištirpusio anglies dioksido koncentracijos dinamikos lygtis:

$$\frac{dC}{dt} = \left(\frac{\mu_s^o}{Y_{\frac{X}{C}}^o} + \frac{\mu_s^f}{Y_{\frac{X}{C}}^f} + \frac{\mu_e^o}{Y_{\frac{X}{C}}^{oe}} \right) X - DC - CT\mathrm{R} \quad (2.8)$$

Ištirpusio anglies dioksido koncentracijos kitimo greitis priklauso nuo biomasės augimo greičių, vykstant skirtingiems metabolizmams sumos. Pagal 2.1 – 2.3 lygtyse aprašytus biomasės augimo metabolizmus susidaro CO_2 dujos. CO_2 tirpimas terpėje priklauso nuo CTR vertės. Ištirpusio anglies dioksido koncentracijos kitimo greitį keičia pumpuojamas pamaitinimo tirpalas skiedimo greičiu *D*. Darau prielaidą, kad ištirpusio anglies dioksido koncentracija mažesnė pamaitinimo tirpale nei terpėje, todėl ištirpusio anglies dioksido koncentracija terpėje mažėja.

$Y_{\frac{X}{C}}^{o}$ – Anglies dioksido kiekis tenkantis 1 gramui biomasės aerobinėse sąlygose naudojant gliukozę, g/g;

$Y_{\frac{X}{C}}^{f}$ – Anglies dioksido kiekis tenkantis 1 gramui biomasės anaerobinėse sąlygose g/g;

$Y_{\frac{X}{C}}^{oe}$ – Anglies dioksido kiekis tenkantis 1 gramui biomasės aerobinėse sąlygose naudojant etanolį, g/g;

CTR – Anglies dioksido tiekimo greitis;

Terpės skiedimas su pamaitinimo tirpalu

Fermentacija vykdoma pertraukiamu su pamaitinimu metodu, todėl į terpę pumpuojamas pamaitinimo tirpalas greičiu $[F] = \frac{L_f}{val}$, kurio tirpinio – gliukozės konentracija siekia 0,55kg/l. Gliukozės koncentracija tirpale visada didesnė nei terpės gliukozės koncentracija, todėl 2.5 lygtyje gliukozės koncentracija didėja. Įtekantis gliukozės srautas didina terpės turį, vyksta skiedimas. Dėl didėjančio terpės tūrio biomasės, etanolio, ištirpusio deguonies ir anglies dioksido koncentracijos sumažėja proporcingai terpės skiedimo greičiui, kurį aprašo 2.9 lygtis. Šių medžiagų koncentracijos matuojamas gramais medžiagos viename litre

Terpės skiedimo greitis žymimas $[D] = \frac{1}{val}$ ir apskaičiuojamas 2.9 lygtimi, kurioje terpės tūris žymimas $[V] = L$:

$$D = \frac{F}{V} \tag{2.9}$$

Sukauptas terpės tūris arba terpės tūrio kitimo greitis priklauso nuo skiedimo greičio ir yra užrašomas diferencialine lygtimi 2.10

$$\frac{dV}{dt} = D \cdot V \tag{2.10}$$

Deguonies ir anglies dioksido tiekimo greitis

Deguonis ir anglies dioksidas iš dujų, esančių bioreaktoriuje, pereina į skystąją terpę difuzijos būdų. Tai yra molekulės skverbiasi iš aukštesnės koncentracijos vietų į žemesnės koncentracijos vietas. Deguonies ir anglies dioksido pernešimo iš dujų į terpę koeficientai atitinkamai žymimi $[k_L a_O] = \frac{1}{val}$ ir $[k_L a_C] = \frac{1}{val}$.

$$OTR = k_L a_O \cdot (O_{sat} - O) \tag{2.11}$$

$k_L a_O$– Deguonies pernešimo iš dujų į terpę koeficientas;

O_{sat}– Deguonies didžiausia galima koncentracija esamomis aplinkos sąlygomis;

$$CTR = k_L a_C \cdot (C - C_{sat}) \tag{2.12}$$

$k_L a_C$– Anglies dioksido pernešimo iš dujų į terpę koeficientas;

C_{sat}– Anglies dioksido didžiausia galima koncentracija esamomis aplinkos sąlygomis;

Mikroorganizmų augimo greičio matematinis modelis

Santykinis biomasės augimo greitis skirtingais gliukozės arba etanolio perdirbimo ciklais aprašomas pagal empirinį „Monod“ modelį[24]. Biomasės augimas priklauso nuo didžiausio galimo biomasės augimo greičio $\mu_{\max}$, augimui reikalingos medžiagos S ir „Monod“ koeficiento šiai medžiagai K_S. Koeficientai $\mu_{\max}$ ir K_S nustatomi empiriškai atskirai ląstelių kultūrai ir priklauso nuo aplinkos, kurioje jos auga, ypatybių. K_S lygus augimui reikalingos medžiagos vertei (koncentracijai), prie kurios biomasės augimo greitis lygus pusei didžiausio galimo biomasės augimo greičio. Šio dydžio matavimo vienetai tokie pat kaip augimui reikalingos medžiagos matavimo vienetai. „Monod“ lygties bendra išraiška pateikta 2.13 lygtyje.

$$\mu = \frac{\mu_{\max} S}{K_S + S} \tag{2.13}$$

„Monod“ lygtį galima išplėsti sudauginant papildomus augimo greitį ribojančius narius, jeigu augimui reikalingos kelios medžiagos. Lygtis 2.14 aprašo santykinį augimo greitį, kai augimui reikalinga *n* skirtingų medžiagų:

$$\mu = \mu_{\max} \cdot \frac{S_1}{K_{S1}+S_1} \cdot \frac{S_2}{K_{S2}+S_2} \cdot \ldots \cdot \frac{S_n}{K_{Sn}+S_n} \tag{2.14}$$

Augimą limituojantys veiksniai

Vykdant fermentacijas pastebėta, kad didėjant kai kurių medžiagų koncentracijai pradeda lėtėti biomasės augimas. Daroma prielaida, kad didėjant biomasės kiekiui sulėtėja jos augimas dėl terpėja susidarančių kenksmingų medžiagų, kurios yra ląstelių metabolizmo šalutinis produktas. Šis augimo sulėtėjimas įvertinamas biomasės augimo greičio lygtyse.

Biomasės augimo greičio išraiška aerobinėse sąlygose

$$\mu_S^o = \frac{\mu_{\max\_so} S}{K_S^o + S} \cdot \frac{K_{X\_so}}{K_{X\_so}+X} \tag{2.15}$$

$\mu_{\max\_so}$ – Didžiausia biomasės augimo greičio išraiška aerobinėse sąlygose naudojant gliukozę;

K_s^o – „Monod“ koeficientas aerobinėse sąlygose naudojant gliukozę;

K_{X_so} – Augimo limitavimo nuo biomasės koeficientas aerobinėse sąlygose naudojant gliukozę;

Biomasės augimo greičio išraiška anaerobinėse sąlygose

$$\mu_s^f = \frac{\mu_{\max\_sf} S}{K_s^f + S} \cdot \frac{K_{X\_sf}}{K_{X\_sf} + X} \frac{K_{E\_sf}}{K_{E\_sf} + E} \tag{2.16}$$

$\mu_{\max\_sf}$ – Didžiausia biomasės augimo greičio išraiška anaerobinėse sąlygose fermentuojant etanolį;

K_s^f – „Monod“ koeficientas anaerobinėse sąlygose fermentuojant etanolį;

K_{X_sf} – Augimo limitavimo nuo biomasės koeficientas aerobinėse sąlygose fermentuojant etanolį;

K_{E_sf} – Augimo limitavimo nuo etanolio koeficientas aerobinėse sąlygose fermentuojant etanolį;

Biomasės augimo greičio išraiška aerobinėse sąlygose naudojant etanolį

$$\mu_e^o = \frac{\mu_{\max\_eo} E}{K_E^{eo} + E} \cdot \frac{K_{X\_eo}}{K_{X\_eo} + X} \tag{2.17}$$

$\mu_{\max\_eo}$ – Didžiausia biomasės augimo greičio išraiška aerobinėse sąlygose naudojant etanolį;

K_E^{eo} – „Monod“ koeficientas aerobinėse sąlygose naudojant etanolį;

K_{X_eo} – Augimo limitavimo nuo biomasės koeficientas aerobinėse sąlygose naudojant etanolį;

2.2 Klasikinių fermentatoriaus parametrų stebėjimas ir reguliavimas

Ištirpusio deguonies koncentracija stebima siekiant palaikyti terpėje aerobinį biomasės augimą. Tai yra ląstelės oksiduoja gliukozę ir biomasė auga. Esant per mažai ištirpusio deguonies koncentracijai pradėtų trūkti deguonies gliukozės oksidavimui ir palaipsniui ląstelės pradėtų fermentuoti etanolį – vyktų rūgimo procesas. Tai yra anaerobinis biomasės augimas kuomet nenaudojamas deguonis, o gliukozė verčiama į etanolį. Anaerobinio proceso metu biomasės augimas greitis kelis kartus mažesnis nei aerobinio proceso metu naudojant gliukozę. Tai yra neefektyvus

biomasės augimas ir jis yra nepageidaujamas. Norint to išvengti stebimi bioreaktoriaus parametrai ir esant nuokrypiams stengiamasi juos pakoreguoti.

Ištirpusio deguonies koncentracija

Ištirpusio deguonies koncentracija gali būti matuojama keliais būdais[10]: titruojant mėginį, atliekant spalvinę analizę arba naudojant elektroninius matavimo prietaisus. Pastarasis būdas patogiausias dėl to, kad galima nepertraukiamai sekti terpėje ištirpusio deguonies koncentraciją ir pagal tai stengtis ją pakeisti. Esant 30°C temperatūrai ir atmosferos slėgiui 760mmHg, ištirpusio deguonies kiekis – 7,6 mg/l[11]. Šis kiekis kinta, bet yra proporcingas atmosferos slėgiui bei atvirkščiai proporcingas aplinkos temperatūrai. Naudojami įvairūs metodai, kurie padidina ištirpusio deguonies koncentraciją:

1. Dėl terpės paviršiaus įtempimo efekto deguonis pumpuojamas per panardintą burbuliavimo sistemą.
2. Oras pumpuojamas į bioreaktorių naudojant kompresorių ir sudaromas viršslėgis bioreaktoriaus talpoje.
3. Kartu su oru papildomai tiekiamas deguonis.
4. Terpė aeruojama naudojant *Rushtono* sraigtą, kurios sūkiai gali būti keičiami.

Atskiri metodai arba jų kombinacijos derinamos su elektroniniais valdymo prietaisais leidžia palaikyti užduotą ištirpusio deguonies koncentraciją ir sudaryti sąlygas optimaliam biomasės augimui pasiekti.

Terpės pH lygis

Netinkamas terpės pH lygis gali tapti inhibitoriumi ir sulėtinti biomasės augimą. Ląstelėms maistines medžiagas gamina fermentai, kurie mielių kultūrose geriausiai veikia, kai pH yra nuo 4,5 iki 5,5. Vykstant mielių augimui išsiskiria CO_2, kuris geriau nei deguonis tirpsta skystyje. Anglies dioksidui jungiantis su vandeniu terpėje susidaro anglies rūgštis H_2CO_3 pagal 2.18 lygtį. Dėl šios susidariusios rūgšties terpės rūgštingumas didėja, pH krenta.

$$CO_2 + H_2O \rightarrow H_2CO_3 \tag{2.18}$$

Terpės pH lygis gali būti matuojamas keliais būdais[12]: titruojant mėginį žinomos koncentracijos šarminiu tirpalu, atliekant spalvinę analizę su lakmuso arba kitos medžiagos indikatoriais, naudojat specialius elektrodus ir elektroninius matavimo prietaisus. Pastarasis būdas patogiausias dėl to, kad galima nepertraukiamai sekti terpės pH lygį, pagal tai tiekti šarminį tirpalą ir kompensuoti terpės rūgštėjimą.

Terpės temperatūra

Mielių aktyvumas tiesiogiai susijęs su molekulių kinetiniu judėjimu, kuris priklauso nuo terpės temperatūros. Remiantis empiriniais duomenimis ir eksperimentais geriausias augimo greitis pasiekiamas, kai terpės temperatūra 20-35°C[13]. Per aukštoje temperatūroje ląstelės žūva. Terpės temperatūra dažniausiai palaikoma ir reguliuojama naudojant elektronines matavimo ir kontroliavimo sistemas, kurios pagal išmatuotą realią terpės temperatūrą ją pašildo arba vėsina.

Išeinančių dujų analizavimas

Įvairių medžiagų koncentracija išeinančiose dujose leidžia stebėti kaip vyksta fermentacija. Deguonies ir anglies dioksido koncentracijų matavimas išeinančiose dujose leidžia stebėti augimo procesą. Etanolio koncentracijos jutiklis išeinančiose dujose fiksuoja kaip kinta etanolio koncentracija terpėje ir pagal tai galima nuspręsti ar vykstą etanolio gamyba ar sunaudojimas. Mirštant ląstelėms arba susidarant tarpiniams junginiams išeinančiose dujose fiksuojamas etanolio koncentracijos didėjimas, nors etanolio koncentracija terpėje nekinta. Deguonies koncentracija išeinančiose dujose palyginama su deguonies koncentracija į reaktorių tiekiamame oro mišinyje ir ištirpusio deguonies koncentracija. Suvartojamo deguonies kiekis proporcingas biomasės kiekiui, todėl pagal šiuos duomenis galima nustatyti biomasės augimo greitį

2.3 Teorinis matematinio modelio parametrų identifikavimas

Matematinio modelio, kuris aprašytas 2.1 poskyryje parametrai, kurie negali būti išmatuojami tiesiogiai, išskaičiuojami iš eksperimentų metu surenkamų duomenų. Naudojantis teorine literatūra bei atliktais eksperimentais aprašomas modelio matematinių lygčių verčių gavimas.

2.3.1 Mažiausių kvadratų metodas bioproceso parametrų identifikavimui

Matematinio modelio lygtyse identifikuojant nežinomus kintamuosius reikia rasti tokias jų reikšmes, su kuriomis matematinio modelio lygties rezultatas minimaliai skirtųsi nuo išmatuoto arba apskaičiuoto rezultato realiomis sąlygomis[14]. Bendra matematinio modelio lygties išraiška su keliais kintamaisiais aprašoma kaip šių kintamųjų funkcija $f(x_i, a, b, c, \dots, d)$, o išmatuotas arba apskaičiuotas rezultatas pažymimas y_i. Mažiausių kvadratų metodu siekiama rasti tokias, kintamųjų reikšmes, kad skirtumų kvadratų suma būtų kuo mažesnė. Skirtumo pakėlimas kvadratu leidžia išvengti neigiamų reikšmių. Lygtis 2.19 vaizduoja bendrą mažiausių kvadratų metodo išraišką, kuri bus naudojama identifikuojant lygčių kintamuosius.

$$S = \sum_{i=1}^{n} (y_i - f(x_i, a, b, c, \dots, d))^2 \rightarrow min \tag{2.19}$$

2.3.2 Gliukozės koncentracijos dinamikos lygties kintamieji

Vykdant fermentacijas fiksuojamos gliukozės bei biomasės koncentracijos vertės iš kurių galima gauti biomasės išeigos koeficientus. Dėl šios priežasties siekiama pertvarkyti gliukozės koncentracijos dinamikos lygtį šių koeficientų gavimui. Gliukozės koncentracija terpėja didėja į terpę tiekiant pamaitinimo tirpalą ir mažėja vykstant biomasės augimui. Pagal masių balanso lygtį tiekiamas substratas sunaudojamas vykstant aerobiniam biomasės augimui naudojant gliukozę arba – anaerobiniam biomasės augimui fermentuojant etanolį. Taip pat gliukozės koncentracijos kitimo greitis tiekiant gliukozę privalo būti lygus suvartojamos gliukozės kitimo greičiui. Daroma prielaida, kad 2.5 gliukozės koncentracijos lygtyje galima neįvertinti terpės skiedimo greičio. Ši lygtis pertvarkoma ir gaunama suvartojamos gliukozės lygtis 2.20.

$$\frac{dS_{vart.}}{dt} = \left(-\frac{\mu_S^o}{Y_{\frac{X}{S}}^o} - \frac{\mu_S^f}{Y_{\frac{X}{S}}^f} \right) X \tag{2.20}$$

Vykdant fermentaciją terpė pamaitinama didelės koncentracijos gliukozės tirpalu, kai gliukozės koncentracija joje sumažėja. Supumpuojant į terpę pamaitinimo tirpalą pažymimi šie parametrai : grynos gliukozės kiekis tirpale, laikas nuo fermentacijos pradžios, faktinis terpės tūris. Gliukozės koncentracijos kitimo greitis ją tiekiant $\frac{dS_{tiek.}}{dt}$ apskaičiuojamas pagal kumuliatyvų grynos gliukozės kiekį ir terpės tūrį. Apskaičiuojamas greitis – neigiamas, nes koncentracija mažėja. Tiekiamos gliukozės greitis aprašomas 2.21 lygtimi

$$\frac{dS_{tiek.}}{dt} = \frac{S_{(i+j)} - S_{(i-j)}}{\left(\frac{V_{vid} + V_i}{2}\right) \cdot (t_{(i+j)} - t_{(i-j)})} \tag{2.21}$$

S– Gliukozės koncentracija terpėje, g/l;

V_{vid}– Vidutinis terpės tūris fermentacijos pradžioje, l;

V_i– Terpės tūris skaičiuojamuoju momentu, l;

t– Laikas nuo fermentacijos pradžios, val;

i– Iteracijos indeksas;

j– Nuotuolis nuo dabartinio skaičiuojamojo momento vertės, pvz. 1,2,3;

Pagal masių balanso dėsnį ir fermentacijos proceso lygtis 2.1 ir 2.2 gliukozė sunaudojama biomasei augti, todėl tiekiamos gliukozės ir suvartojamos gliukozės greičiai yra lygus: $\frac{dS_{tiek.}}{dt} = \frac{dS_{vart.}}{dt}$ .Tuomet galima į 2.20 lygtį įstatyti tiekiamos gliukozės greitį:

$$\frac{dS_{tiek.}}{dt} = \left(-\frac{\mu_s^o}{Y_{\frac{X}{S}}^o} - \frac{\mu_s^f}{Y_{\frac{X}{S}}^f} \right) X \quad (2.22)$$

Išeigos koeficientus galima rasti mažiausių kvadratų metodu, todėl ši lygtis pertvarkoma išreiškiant skirtumą tarp tiekiamos ir suvartojamos gliukozės greičių. Šis skirtumas žymimas Δ_S ir keisis parenkant skirtingus ieškomus išeigos koeficientus. Bendras specifinis augimo greitis $\mu_{bendras}$ gali būti matuojamas naudojantis sausos biomasės kiekio arba optinio tankio skaičiavimo metodais. Remiantis literatūra ir empiriniais duomenimis biomasės augimo greitis μ_s^f prilyginimas konstantai. Biomasės augimo greitis μ_s^o išreiškiamas bendro biomasės augimo greičio ir konstantos skirtumu.

$$\Delta_S = -\frac{dS_{tiek.}}{dt} + \left(-\frac{\mu_{bendras} - \mu_s^f}{Y_{\frac{X}{S}}^o} - \frac{\mu_s^f}{Y_{\frac{X}{S}}^f} \right) X \quad (2.23$$

Dešiniąją lygties 2.23 pusę pakėlus kvadratu gaunamas kvadratinis nuokrypis. ΔS_{suma} pažymime kaip kvadratinių nuokrypių sumą ir išreiškiama kaip funkcija nuo parenkamų išeigos koeficientų.

$$\Delta_{S_{suma}}(Y_{\frac{X}{S}}^o, Y_{\frac{X}{S}}^f) = \sum_{i=1}^{n} \left(-\frac{dS_{tiek.}}{dt} + \left(-\frac{\mu_{bendras} - \mu_s^f}{Y_{\frac{X}{S}}^o} - \frac{\mu_s^f}{Y_{\frac{X}{S}}^f} \right) X \right)^2 \quad (2.24)$$

$Y_{\frac{X}{S}}^o$ – Ieškomas gliukozės kiekis tenkantis 1 gramui biomasės aerobinėse sąlygose, g/g;

$Y_{\frac{X}{S}}^f$ – Ieškomas gliukozės kiekis tenkantis 1 gramui biomasės anaerobinėse sąlygose, g/g;

$\frac{dS_{tiek.}}{dt}$ – Tiekiamos gliukozės koncentracijos kitimo greitis, g/(val·l);

$\mu_{bendras}$ – Eksperimento metu išmatuotas bendras biomasės augimo greitis;

μ_s^f – Biomasės augimo greitis anaerobinėmis sąlygomis, kuris lygus konstantai;

Atlikus fermentaciją ir sukaupus realaus proceso duomenis galima naudoti 2.24 funkciją gliukozės kiekio tenkančio vienam gramui biomasės koeficientų paieškai taikant mažiausių kvadratų metodą.

2.3.3 Ištirpusio deguonies koncentracijos dinamikos lygtis.

Vykdant fermentacijas fiksuojamos deguonies suvartojimo, biomasės augimo greičių bei etanolio koncentracijos vertės iš kurių galima gauti biomasės išeigos koeficientus. Dėl šios

priežasties siekiama pertvarkyti ištirpusio deguonies koncentracijos dinamikos lygtį šių koeficientų gavimui. Ištirpusio deguonies koncentracijos kitimo greitis priklauso nuo ląstelių suvartojamo deguonies kiekio OUR, kuris tiesiogiai priklauso nuo biomasės kiekio terpėje ir jos tūrio[15] , bei tiekiamo deguonies kiekio OTR ir aprašomas lygtimi 2.25.

$$\frac{dO}{dt} = -OUR + OTR \tag{2.25}$$

2.7 lygties dalis, kuri nusako kiek deguonies sunaudojama biomasei augti, yra lygi išmatuotai OUR vertei fermentacijos metu ir aprašyta 2.26 lygtimi .

$$OUR = \left(\frac{\mu_s^o}{Y_{\frac{X}{O}}^o} + \frac{\mu_e^o}{Y_{\frac{X}{O}}^{oe}} \right) X \tag{2.26}$$

Ryškiausi ir patikimiausiai identifikuojami aerobinio augimo etapai naudojant etanolį, kuriais nevyko aerobinis biomasės augimas naudojant gliukozę, $\mu_s^o = 0$, todėl lygtyje 2.26 biomasės augimo aerobinėse sąlygose naudojant gliukozę dalis lygi nuliui.

$$OUR = \frac{\mu_e^o}{Y_{\frac{X}{O}}^{oe}} X \tag{2.27}$$

2.4 lygtis aprašo kaip kito biomasė. Kadangi tiriamuose intervaluose vyko tik aerobinis biomasės augimas naudojant etanoli, tai augimo greičiai – $\mu_s^o = 0$ ir $\mu_s^f = 0$. 2.4 lygtis pertvarkoma į biomasės kitimo greičio lygtį tik etanolinio augimo metu. Terpės skiedimas *D* nevertinamas.

$$\frac{dX}{dt} = \mu_e^o X \tag{2.28}$$

Lygtyje 2.27 specifinio biomasės augimo greičio ir biomasės vertės sandaugą pakeičiama 2.28 lygtimi.

$$OUR = \frac{\frac{dX}{dt}}{Y_{\frac{X}{O}}^{oe}} \tag{2.29}$$

2.29 lygtis pertvarkoma į deguonies kiekio tenkančio vienam gramui biomasės koeficiento radimui vykstant biomasės augimui aerobinėse sąlygose naudojant etanolį.

$$Y_{\frac{X}{O}}^{oe} = \frac{\frac{dX}{dt}}{OUR} \tag{2.30}$$

Atlikus fermentaciją ir sukaupus realaus proceso duomenis galima naudoti 2.30 lygtį biomasės išeigos koeficiento $Y_{\frac{X}{O}}^{oe}$ gavimui.

2.3.4 Etanolio koncentracijos dinamikos lygtis.

Vykdant fermentacijas fiksuojamos biomasės augimo greičio bei etanolio koncentracijos vertės iš kurių galima gauti etanolio tenkančio vienam gramui biomasės koeficientus. Dėl šios priežasties siekiama pertvarkyti etanolio koncentracijos dinamikos lygtį šių koeficientų gavimui. Ryškiausi ir patikimiausiai identifikuojami aerobinio biomasės augimo etapai naudojant etanolį, kuriais nevyko anaerobinis biomasės augimas fermentuojant etanolį, $\mu_s^f = 0$, todėl lygtyje 2.6 biomasės augimo anaerobinėse sąlygose fermentuojant etanolį dalis lygi nuliui. Etanolio koncentracijos kitimą, kai vyksta tik biomasės augimas aerobinėse sąlygose naudojant etanolį, aprašo 2.31 lygtis.

$$\frac{dE}{dt} = \left(0 - \frac{\mu_e^o}{Y_{\frac{X}{E}}^{oe}}\right) X \tag{2.31}$$

2.4 lygtis aprašo kaip kito biomasė. Kadangi tiriamuose intervaluose vyko tik aerobinis biomasės augimas naudojant etanoli, tai augimo greičiai – $\mu_s^o = 0$ ir $\mu_s^f = 0$. Biomasės kitimo greitį šiomis sąlygomis aprašo 2.28 lygtis, kuri įstatoma į 2.31 lygtį ir gaunama 2.32 lygtis.

$$\frac{dE}{dt} = -\frac{\frac{dX}{dt}}{Y_{\frac{X}{E}}^{oe}} \tag{2.32}$$

Iš gautos lygties galima išreikšti ieškomą etanolio tenkančio vienam gramui biomasės išeigos koeficientą, kuris galioja vykstant biomasės augimui aerobinėse sąlygose naudojant etanolį.

$$Y_{\frac{X}{E}}^{oe} = -\frac{\frac{dX}{dt}}{\frac{dE}{dt}} \tag{2.33}$$

Atlikus fermentaciją ir sukaupus realaus proceso duomenis galima naudoti 2.33 lygtį biomasės išeigos koeficiento $Y_{\frac{X}{E}}^{oe}$ gavimui.

2.3.5 Biomasės augimo greičio aerobinėse sąlygose naudojant etanolį lygtis

Vykstant aerobiniam biomasės augimui naudojant etanolį fiksuojamos etanolio koncentracijos vertės kas valandą, o biomasės koncentracijos ir biomasės augimo greičio – kas minutę. Iš šių duomenų galima identifikuoti biomasės augimo greičio aerobinėse sąlygose naudojant etanolį maksimalų augimo greitį, „Monod“ ir limitavimo koeficientus. Kaupiamų duomenų intervalai

nesutampa, todėl lygtis pertvarkoma atskiriant minutinių ir valandinių duomenų narius. Lygtis 2.17 padalinama iš augimą limituojančio nario $\frac{K_X^{eo}}{K_X^{eo}+X}$.

$$\mu_e^o \cdot \frac{K_X^{eo}+X}{K_X^{eo}} = \frac{\mu_{\max\_eo} E}{K_E^{eo}+E} \tag{2.34}$$

Gauta lygtis 2.34, kurios kairiojoje pusėje – minutiniu intervalu matuoti duomenys, o dešiniojoje – valandiniu. Maksimalų augimo greitį ir koeficientus galima rasti mažiausio skirtumo metodu, todėl ši lygtis pertvarkoma išreiškiant skirtumų sumą tarp valandinių ir minutiniu intervalu išmatuotų duomenų. Ši skirtumų suma žymima $\Delta_{mju\_eo\_suma}$ ir keisis parenkant skirtingus ieškomus koeficientus. Tai bedimensinis dydis. $\Delta_{mju\_eo\_suma}$ išreiškiamas kaip funkcija nuo ieškomų ir keičiamų parametrų.

$$\Delta_{mju\_eo\_suma}\left(K_X^{eo}, \mu_{\max\_eo}, K_E^{eo}\right) = \sum_{i=1}^{n}\left(\mu_e^o \cdot \frac{K_X^{eo}+X}{K_X^{eo}} - \frac{\mu_{\max\_eo} E}{K_E^{eo}+E}\right) \tag{2.35}$$

$\mu_{\max\_eo}$ – Ieškoma didžiausia biomasės augimo greičio išraiška aerobinėse sąlygose naudojant etanolį;

K_E^{eo} – Ieškomas „Monod“ koeficientas aerobinėse sąlygose naudojant etanolį;

K_{X_eo} – Ieškomas augimo limitavimo nuo biomasės koeficientas aerobinėse sąlygose naudojant etanolį;

X – Biomasės koncentracija;

E – Etanolio koncentracija;

μ_e^o – Biomasės augimo greitis aerobinėmis sąlygose;

Atlikus fermentaciją, kurios metu vyktų tik aerobinis biomasės augimas naudojant etanolį, ir sukaupus realaus proceso duomenis galima naudoti 2.35 funkciją $K_X^{eo}, \mu_{\max\_eo}, K_E^{eo}$ koeficientų paieškai.

2.4 Matematinio modelio sudarymas simuliavimo aplinkoje

Skaitmeninio modeliavimo metodai leidžia sumažinti vykdomų fermentacijų kaštus. Stebint ir tiriant procesą įgautos žinios panaudojamos modelio sudarymui programinės įrangos *„Matlab“* grafiniu modeliavimo paketu *„Simulink“*. Pagal 2.1 poskyryje aprašyto matematinio modelio lygtis sudaroma modelio schema *„Simulink“* pakete.

Sudarytą klasikinę modelio schemą nenaudojant DEE diferencialinių lygčių modulio galima greitai ir lengvai papildyti reikalingais duomenų kaupimo ir atvaizdavimo įrankiais, kurie leidžia stebėti tarpines vertes. Pavyzdžiui, kiek fermentacijos metu buvo sukaupta ir sunaudota etanolio. Taip pat šis modelio sudarymo metodas yra universalus ir gali būti tiriamas su kitomis

matematinio modeliavimo programinėmis įrangomis. Integratoriaus blokas naudojamas pradinių proceso kintamųjų įvedimui. Tai reikalinga tam, kad kai kurios vertės fermentacijos pradžioje nėra lygios nuliui. Terpėje yra tam tikras ištirpusio deguonies bei etanolio kiekis, pH lygis, kurio svarba ląstelių augimui aprašyta 2.2 poskyryje.

Nagrinėjamame matematiniame modelyje nenumatytas terpės tūrio kitimas atliekant mėginių paėmimus iš terpės. Ištirpusio deguonies ir anglies dioksido koncentracija pamaitinimo tirpale taip pat nėra modeliuojama. Dėl šių priežasčių proceso verčių kitimas priklausomai nuo terpės skiedimo greičio nevertinamas.

Įtekančio pamaitinimo tirpalo srauto greitis modeliuojamas pagal 2.21 lygtį, kuri aprašo kaip kinta tiekiamos gliukozės koncentracija terpėje. Šia lygtimi apskaičiuota vertė per verčių lentelę (*angl. lookup table*) įvedama į gliukozės koncentracijos dinamikos lygties posistemę. Tiekiamo oro kiekis (*angl. Oxygen Trasfer Rate*) apskaičiuojamas pagal 2.36 lygtį[15]. Šios lygties schema sudaryta „OTR-CTR“ posistemėje ir pavaizduota 2.1 paveikslėlyje.

$$\mathrm{OTR} = \alpha \cdot \mathrm{N}^{\beta} \cdot \mathrm{Q}^{\gamma}\left(\mathrm{O}_{\mathrm{SAT}} - \mathrm{O(t)}\right) \tag{2.36}$$

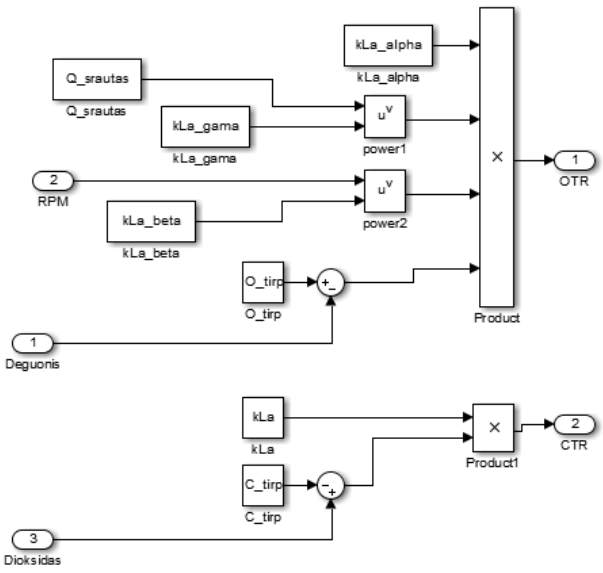
N – sraigto sukimosi greitis, RPM;

Q – į bioreaktorių tiekiamo oro greitis, l/s;

α,β,γ – empirinės konstantos, kurios parinktos pagal KTU biotechnologinių procesų modeliavimo ir valdymo laboratorijoje naudojamą bioreaktorių;

O – ištirpusios deguonies koncentracija terpėje, g/l;

O_{sat} – Deguonies didžiausia galima koncentracija esamomis aplinkos sąlygomis;



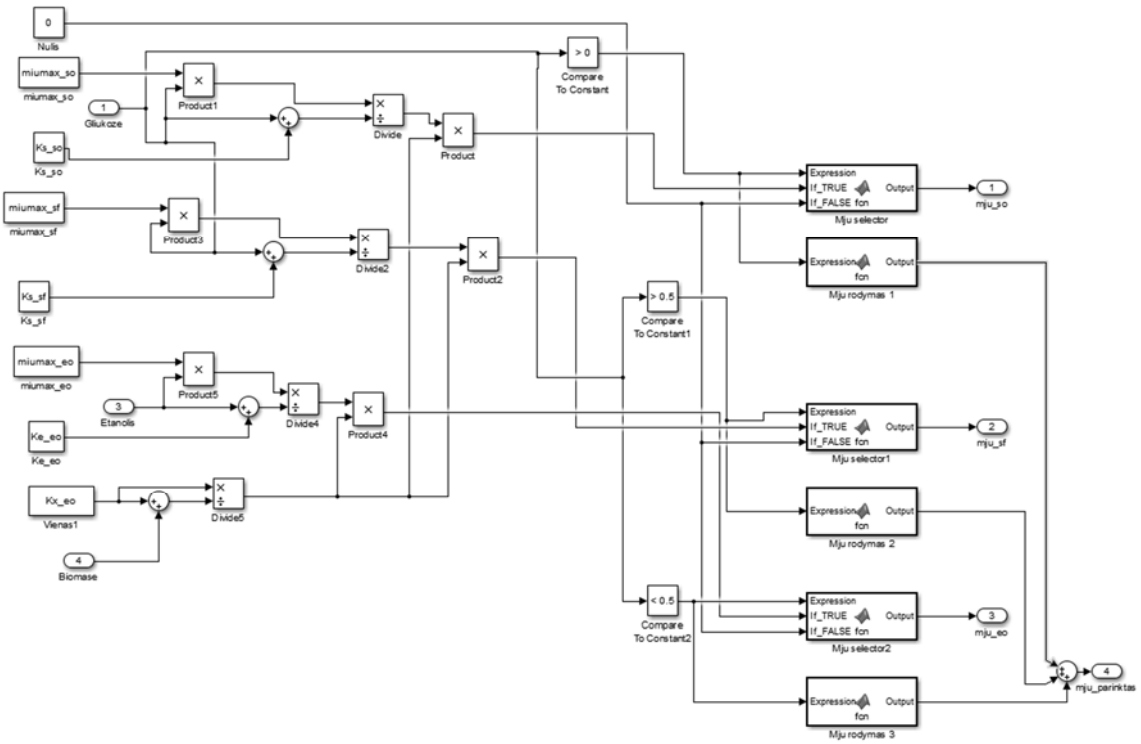
2.1 pav. *„OTR“* ir *„CTR“* verčių modeliavimo schema.

Tokiu pat metodu naudojant verčių lenteles į modelį galima įkelti biomasės augimo greičio, biomasės koncentracijos kitimo duomenis, kurie sukaupti vykdant realias fermentacijas.

Esant skirtingoms gliukozės koncentracijoms terpėje vyksta skirtingi biomasės augimo metabolizmai. Augimo greičiai, kurie aprašyti 2.15-2.17 lygtimis, parenkami pagal 2.1 lentelę[9]. Priimama, kad augimo limitavimo nuo biomasės įtaka vienoda visiems trims augimo metabolizmams. Augimo greičio parinkimo modelio schema pavaizduota 2.2 pav.

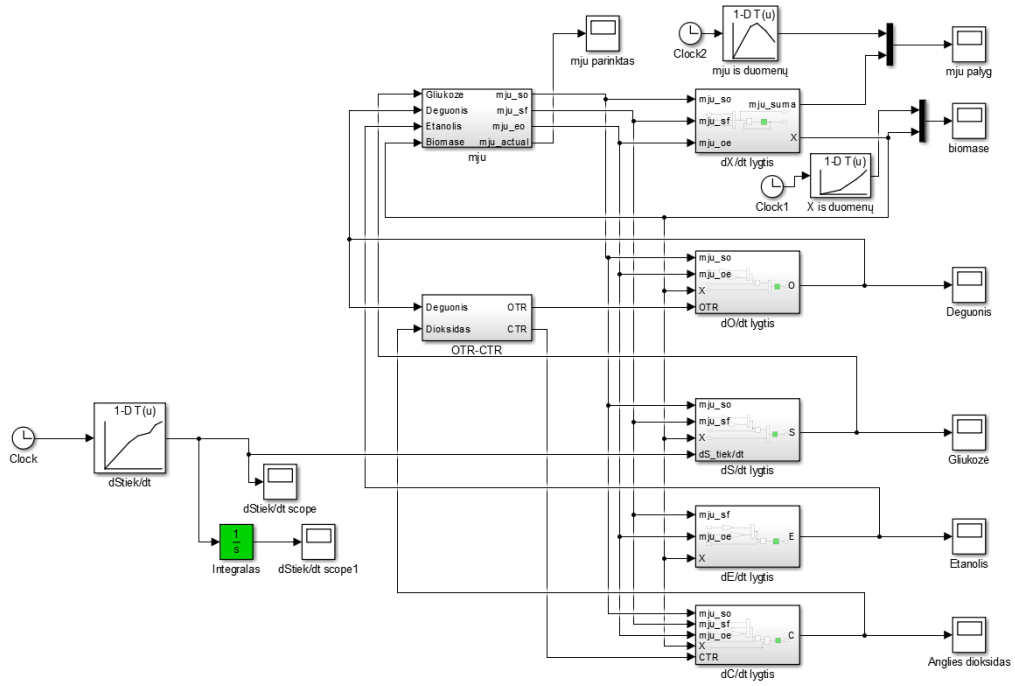
2.1 lentelė. Augimo metabolizmų parinkimas modelyje

Gliukozės koncentracija terpėje, g/g	Augimo metabolizmas
> 0	Aerobinėse sąlygose naudojant gliukozę
> 0,5	Anaerobinėse sąlygose naudojant gliukozę
< 0,5	Aerobinėse sąlygose naudojant etanolį



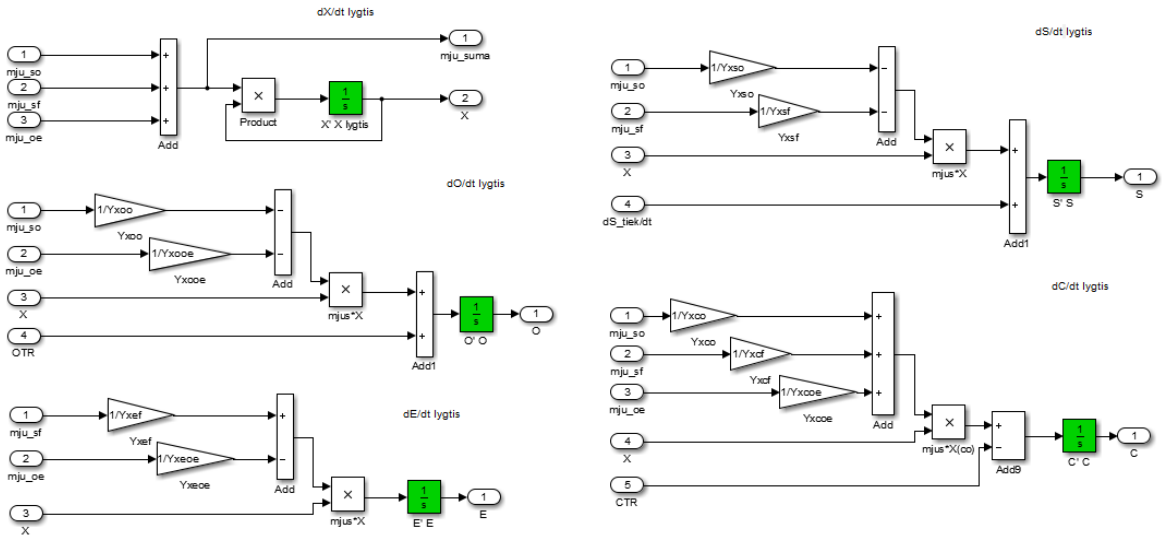
2.2 pav. Santykinio augimo greičio parinkimo schema

Pagal šias aprašytas modelio savybes ir atliktus pakeitimus sudaryta bendra modelio schema *„Simulink“* pakete pavaizduota 2.3 paveikslėlyje.



2.3 pav. Modelio schema.

Koncentracijos dinamikos lygtimis 2.4 - 2.8 aprašyti koncentracijos kitimai simuliavimo modelyje sudaromi iš atskirų komponentų. Šių lygčių modeliai patalpinami atskiruose posistemių blokuose (*angl. subsystem).* Šių posistemių vidiniai sujungimai pavaizduoti 2.4 paveikslėlyje.



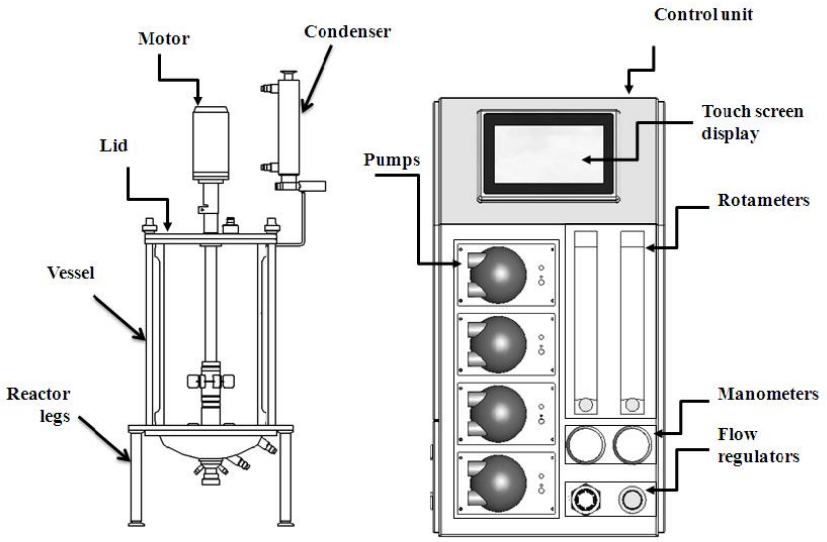
2.4 pav. Atskiros koncentracijos dinamikos lygtys posistemių viduje.

Šis sudarytas matematinis modelis bus palyginamas su duomenimis gautais iš atliktos fermentacijos KTU biotechnologinių procesų modeliavimo ir valdymo laboratorijoje.

3. EKSPERIMENTINIS TYRIMAS

3.1 Aparatinė įranga

Bioreaktorius tai įrenginys, kuriame vykdomi biotechnologiniai procesai su ląstelėmis arba biomase. Jame sukuriamos ir palaikomos pageidaujamos aplinkos savybės, kuriose geriausiai auga pasirinkta kultūra [16]. KTU biotechnologinių procesų modeliavimo ir valdymo laboratorijoje naudojamas aerobinio bioreaktoriaus bei automatines valdymo sistemos komplektas „EDF-5.4“, kurį pagamino „JSC “Biotehniskais centrs”



3.1 pav. EDF-5.4_1 bioreaktorius ir valdymo sistema[17].

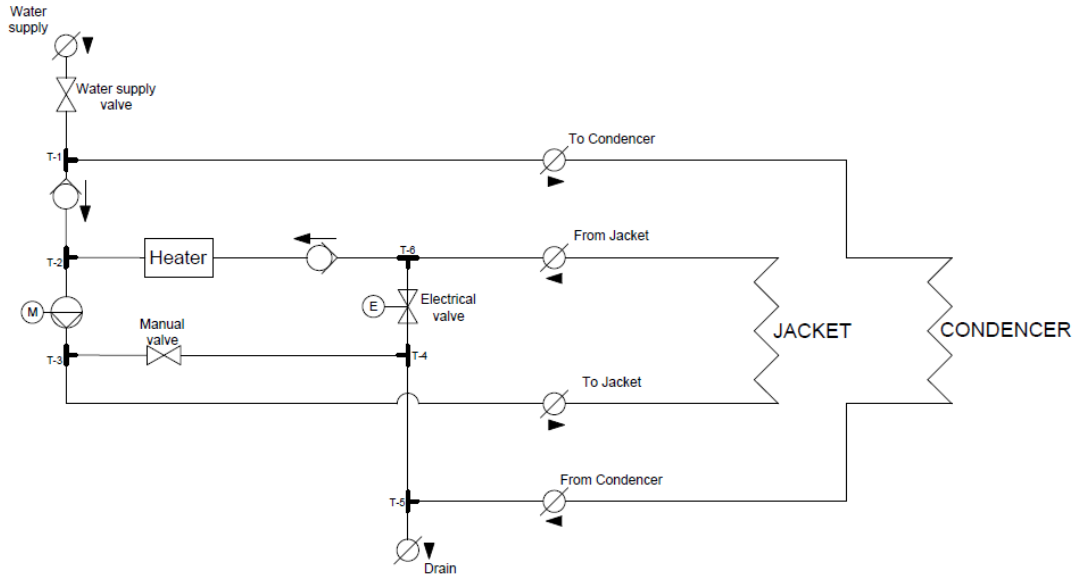
Šio bioreaktoriaus darbinis tūris gali kisti nuo 2 iki 4,5 L. Terpė maišoma sraigtu, sujungtu per magnetinę jungtį su bešepetėliniu nuolatinės srovės varikliu. Valdymo sistema su grįžtamuoju ryšių keičia variklio apsisukimų skaičių. Sistemos centrinis valdiklis – „Siemens IM151-8“ su įėjimų ir išėjimų praplėtimo moduliais. Valdiklis kontroliuoja šiuos terpės parametrus: temperatūra, pH lygis, pO_2 koncentracija bei maišyklės sukimosi greitis.

Temperatūros matavimas ir reguliavimas.

Terpės temperatūra matuojama varžiniu termometru su „PT100“ platininiu matavimo elementu. Termometras prijungtas prie valdiklio analoginių įėjimų modulio skirto įvairių termometrų signalui matuoti. Matavimo skiriamoji geba – 15 bitų.

Temperatūra palaikoma pumpuojant skirtingos temperatūros vandenį per „U“ formos šilumokaitį panardintą terpėje. Centrinis valdiklis su PID reguliatoriumi reguliuoja pumpuojamo vandens temperatūrą. Terpės temperatūra pakeliama pumpuojant vandenį uždaru ratu tarp šilumokaičio ir elektrinio šildytuvo, o sumažinama – įjungiant elektromagnetinį vožtuvą ir į

šilumokaitį pumpuojant vėsų vandentiekio vandenį. Hidraulinė šildymo ir vėsinimo sistemos schema pavaizduota 3.2 paveikslėlyje.



3.2 pav. Hidraulinė temperatūros palaikymo sistemos schema.

pH matavimas ir reguliavimas.

Terpės pH lygis matuojamas stikliniu elektrodu, kurio vidus užpildytas kalio chlorido elektrolitu. Šis matavimo prietaisas – gelinio tipo[18], jo nereikia papildyti elektrolitu todėl paprasčiau jį naudoti. Matuojamo pH lygio diapazonas – nuo 0 iki 14 pH. Jutiklio signalą apdoroja keitiklis, kuris milivoltų eilės įėjimo įtampą filtruoja ir konvertuoja ją į standartinių analoginių išėjimų 0-10V ir 4-20mA signalą, kurie atitinka 0-14 pH lygį. Tiesioginis įtampos matavimas be šio keitiklio negalimas, nes standartiniams matavimo prietaisams šis signalas yra per silpnas bei labai jautrus aplinkos trukdžiams. Šio siųstuvo 4-20mA išėjimo signalas sujungtas su centrinio valdiklio analoginių įėjimų modulių, kurio matavimo skiriamoji geba – 13 bitų.

Vykstant procesams aprašytiems 2.2 poskyryje, kai anglies dioksidas jungiasi su vandeniu ir susidaro anglies rūgštis, terpės rūgštingumas didėja, o pH lygis - mažėja. Centrinis valdiklis turi pH palaikymui skirtą PID reguliatorių, kuris valdo peristalinį siurblį pumpuojantį į terpę šarminį tirpalą.

pO_2 koncentracijos matavimas ir reguliavimas.

Terpėje ištirpusio deguonies kiekis matuojamas polarografiniu[19] pO_2 jutikliu, kurį sudaro auksinis arba platininis katodas ir sidabrinis anodas patalpinti elektrolite bei deguonies molekulėms laidi membrana. Jutiklis maitinamas nuolatine įtampa, o srovės stipris proporcingas terpėje esančio

deguonies koncentracijai. Signalą apdoroja siųstuvas, kuris nanoamperų eilės srovę filtruoja ir konvertuoja ją į standartinių analoginių išėjimų 0-10V arba 4-20mA signalą, kurie atitinka išmatuotą srovės stiprį. Tiesioginis srovės matavimas be šio siųstuvo negalimas, nes standartiniams matavimo prietaisams šis signalas yra per silpnas bei labai jautrus aplinkos trukdžiams. Šio siųstuvo 4-20mA išėjimo signalas sujungtas su centrinio valdiklio analoginių įėjimų modulių, kurio matavimų skiriamoji geba – 13 bitų.

Prieš kiekvieną fermentaciją būtina sukalibruoti jutiklį, kad būtų gaunami tikslūs matavimų duomenys. Jutiklio kalibravimo procedūra įjungiama naudojantis valdymo sistemos operatoriaus pultu ir atliekama dvejiems matavimo taškams. Nulinis (pirmasis) taškas sukalibruojamas patalpinant jutiklį į aplinką, kurioje nėra deguonies, tai gali būti Na_2SO_3 tirpalas. Antrojo taško kalibravimui jutiklis patalpinamas į tirpalą su žinoma ištirpusio deguonies koncentracija. Iš šių dviejų taškų duomenų valdiklio algoritmas apskaičiuoja santykį tarp išmatuoto elektrinio signalo ir ištirpusio deguonies kiekio terpėje.

Augant biomasei ir vykstant procesams aprašytiems 2.1 ir 2.3 lygtyse didėja deguonies sunaudojimas. Centrinis valdiklis turi pO_2 palaikymui skirtą PID reguliatorių, kuris automatiškai valdo maišyklės sūkių greitį. Kai terpės aeravimas maišykle – nepakankamas, tuomet panaudojami kiti metodai, aprašyti 2.2 poskyryje, skirti padidinti ištirpusio deguonies koncentraciją terpėje. Operatorius gali padidinti tiekiamo oro slėgį. Taip pat galima tiekti į terpę gryna >99% deguonį, tačiau KTU biotechnologinių procesų modeliavimo ir valdymo laboratorijoje tai nenaudojama.

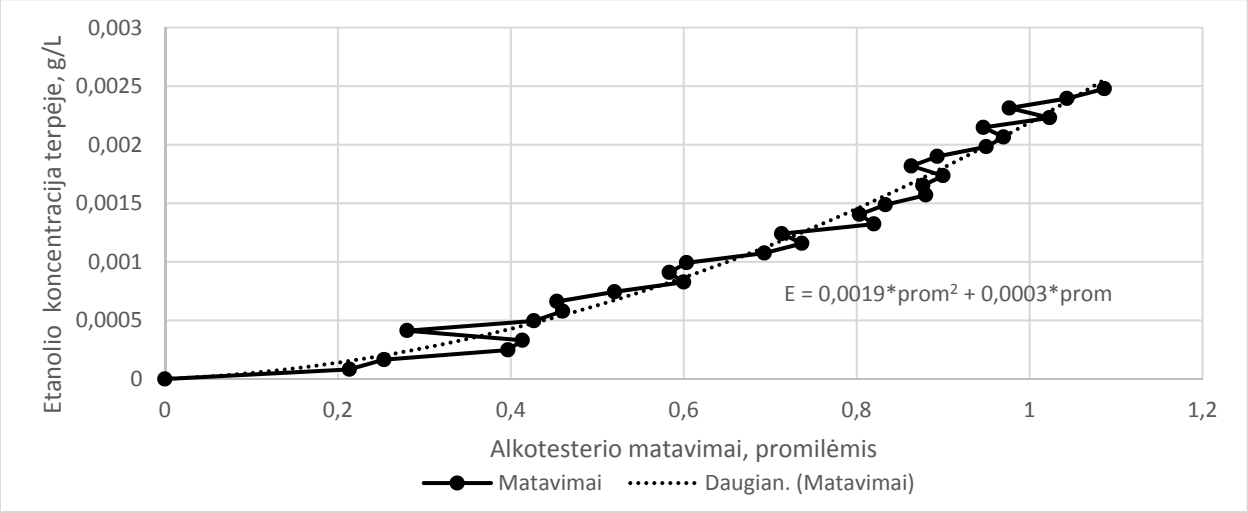
3.1.1 Alkotesterio kalibravimas.

Vykstant rūgimo procesui terpėje susidaro etanolis ir tarpiniai jo junginiai. Šių medžiagų kiekio tiesioginis matavimas su alkometru – sudėtingas ir nepatikimas būdas. Alkometras veikia skysčių tankio matavimo principų, o terpės tankis yra nepastovus. Dalis etanolio išpučiama lauk iš bioreaktoriaus. Panaudojus šią bioreaktoriaus savybę galima netiesiogiai matuoti bioreaktoriuje esančio etanolio koncentraciją analizuojant išeinančias dujas.

Etanolio koncentracija išeinančiose dujose buvo matuota alkotesteriu ir matavimų rezultatai užrašomi fermentacijos protokole. Matematiniame modelyje etanolio koncentracija terpėje matuojama g/l, tai yra gryno etilo alkoholio masė viename terpės litre. Alkotesteris etanolio koncentraciją iš bioreaktoriaus išeinančiame dujų mišinyje matuoja promilėmis.

Atlikus bandomąjį matavimą su alkotesteriu surinkti duomenys bus panaudoti perskaičiuojant išmatuotas promiles į alkoholio koncentraciją gramais litre. Siekiant atkartoti fermentacijos sąlygas į bioreaktorių buvo pripiltas 3,5 l vandens kiekis, kuris lygus terpės tūriui

fermentacijos metu, maišyklės sraigto greitis – 300 aps/min. Bandomojo matavimo metu į bioreaktorių buvo pilamas vienodas etilo alkoholio kiekis ir po kiekvieno papildymo suskaičiuojama etilo alkoholio koncentracija terpėje. Šio matavimo duomenys pateikti 1 priede. Matavimų kreivė pateikta 3.3 pav. Ištisinė linija su žymekliais – alkotesteriu išmatuotos vertės promilėmis, punktyrinė linija - tendencijos kreivės lygtis *(angl. trend line equation).*



3.3 pav. Etanolio koncentracijos ir išmatuotų promilių kreivė.

Etanolio kiekio išeinančiose dujose priklausomybė nuo terpėje esančio etanolio kiekio – netiesinė. Gauta tendencijos kreivės lygtis $E=0,0019*prom^2+0,0003*prom$ kur E – etanolio koncentracija terpėje ir *prom* – alkotesterių išmatuota vertė promilėmis. Su šia tendencijos kreivės lygtimi alkotesteriu išmatuota koncentracija promilėmis konvertuojama į etanolio koncentraciją terpėje gramais litre. Duomenys gauti atlikus konvertavimą bus naudojami identifikuojant matematinio modelio parametrus susijusius su etanolio koncentracija terpėje.

3.1.2 Dujų analizatoriaus kalibravimas.

Siekiant pagerinti etanolio koncentracijos matavimų tikslumą ir automatizuoti procesą panaudojamas elektroninis dujų analizatorius. Etanolio koncentracija matuojama „Grove - Multichannel Gas Sensor“ įvairių dujų koncentracijos matavimo prietaisu, kurio matavimo elementas – „SGX Sensortech MiCS-6814“ , o mikrovaldiklis – „Atmega 168PA“[20].

Matavimo elementas sudarytas iš trijų atskirų matavimo dalių, kurių kiekviena turi atskirus kaitinimo ir matavimo komponentus. Pirmasis matavimo elementas fiksuoja oksiduojančiąsias dujas: NO_2; antrasis – redukuojančiąsias dujas: CO, CH_4, H_2, C_2H_5OH, trečiasis – amoniakines dujas: NH_3, C_3H_8, C_4H_{10}. Matavimo komponentų jautrusis sluoksnis sudarytas iš metalo oksido, kuris yra kaitinamas su įterptu rezistoriumi. Dujoms patenkant ant jautriojo sluoksnio, kinta šio sluoksnio

elektrinė varža. Matuojamų dujų koncentracija ore apskaičiuojama pagal santykį R_s/R_0, kur R_s – dabartinė išmatuota jautriojo sluoksnio varža, R_0 – kontroliuojamomis sąlygomis išmatuota jautriojo sluoksnio varža patalpinus jutiklį į švaraus oro aplinką. Šis santykis konvertuojamas į dujų koncentraciją išreikštą milijoninėmis masės dalimis *(angl. ppm – parts per million).*

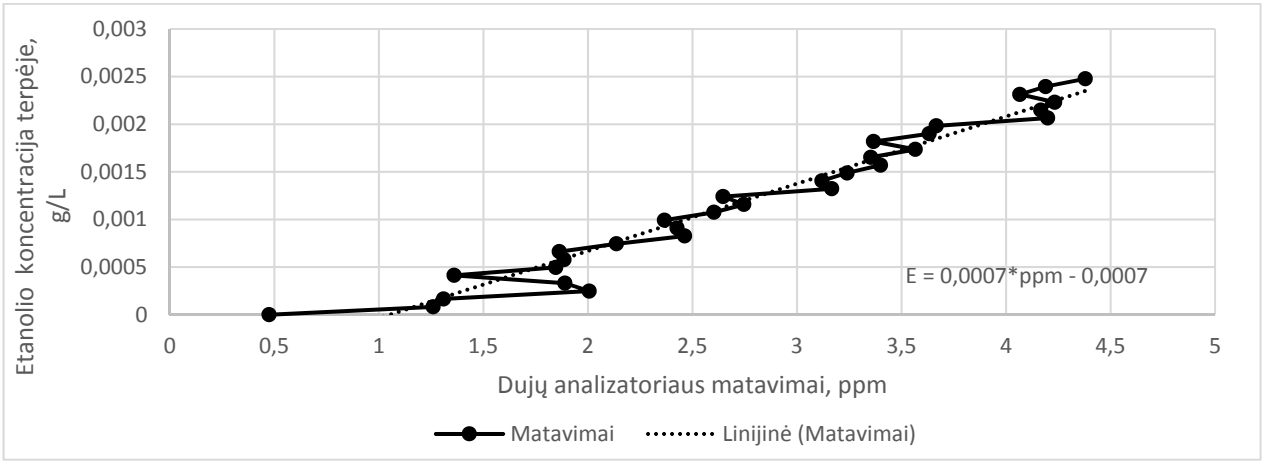
Matavimus ir duomenų perdavimą atlieka mikrovaldiklis „Atmega 168PA“. Jutiklio atskirų matavimo komponentų varža išskaičiuojama matuojant įtampos kritimą ant jų su atskirais analoginiais 10 skilčių (bitų) įėjimais. Atraminė matavimo įtampa – 5V. Vykdant duomenų nuskaitymą išorinis valdikliu siunčia užklausas „I^2C“ magistrale į matavimo prietaiso mikrovaldiklį, o šis siunčia atsakymus. Tai yra visų trijų matavimo komponentų varžų R_s ir R_0 skaitmenines R_{s_d} ir R_{0_d} reikšmes rėžiuose 0 – 1023. Varžų santykis apskaičiuojamas pagal šią lygtį[20]:

$$\frac{R_s}{R_0} = \frac{\mathrm{Rs\_d}}{\mathrm{R0\_d}} \cdot \frac{1023 - \mathrm{R0\_d}}{1023 - \mathrm{Rs\_d}}$$

Jutiklio gamintojas etanolio koncentracijai apskaičiuoti pateikią šią lygtį:

$$C_{Etanol.} = 1{,}622 \cdot \left(\frac{R_s}{R_0}\right)^{-1{,}552} \tag{3.1}$$

Siekiant perskaičiuoti jutikliu išmatuotas vertes į alkoholio koncentraciją terpėje buvo atlikti bandomieji matavimai. Bandymo sąlygos tokios pat kaip atliekant alkotesterio bandomąjį matavimą. Šio matavimo duomenys pateikti 2 priede. Matavimų kreivė pateikta 3.4 pav. Ištisinė linija su žymekliais – dujų analizatoriumi išmatuotos vertės milijoninėmis masės dalimis, punktyrinė linija - tendencijos kreivės lygtis *(angl. trend line equation).*



3.4 pav. Etanolio koncentracijos ir dujų analizatoriaus duomenų kreivė.

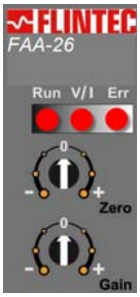
Etanolio kiekio išeinančiose dujose priklausomybė nuo terpėje esančio etanolio kiekio – tiesinė. Gauta tendencijos kreivės lygtis *E=0,0007*ppm-0,0007* kur E – etanolio koncentracija terpėje

ir *ppm* – „Grove - Multichannel Gas Sensor“ prietaisu išmatuota vertė milijoninėmis masės dalimis. Duomenys gauti atlikus šį bandomąjį matavimą bus naudojami tolimesnėse fermentacijose.

3.1.3 Terpės svorio matavimas.

Dažniausiai bioprocesuose būtina normalizuoti visas matuojamas charakteristikas vienam terpės litrui. Taip matuojamas ne absoliutus įvairių medžiagų kiekis, o koncentracija terpėje arba koncentracijos kitimas. Šiems matavimams atlikti būtina turėti duomenis apie terpės turį, tačiau tūrio matavimas yra nepatogus ir sudėtingai realizuojamas dėl šių priežasčių: santykinai brangūs ir nestabilus ultragarsiniai jutikliai, kuriems neigiama įtaką gali daryti maišoma terpė, būtinas sterilumas. Įvertinus šiuos veiksnius pasirinkta netiesiogiai matuoti terpės tūrį naudojant svėrimo platformą. Bioreaktorius su visa būtina įranga pastatomas ant svėrimo platformos, kurioje sumontuotas 50 kg svorio jutiklis, ir fiksuojant svorio pokyčius pagal terpės tankį suskaičiuojamas jos tūris. Nėra būtina matuoti absoliutinį terpės svorį, pakanka sekti terpės svorio kitimą nuo fermentacijos pradžios, todėl matavimo procesas supaprastėja.

Svorio jutiklio sužadinamas nuolatine įtampa ir jo išėjimo milivoltų eilės signalas proporcingas sužadinimo įtampai ir apkrovos svoriui. Tiesioginis jutiklio išėjimo įtampos matavimas negalimas, nes standartiniams matavimo prietaisams šis signalas yra per silpnas bei labai jautrus aplinkos trukdžiams. Išanalizavus bioreaktoriaus elektrines schemas nuspręsta naudoti svorio jutiklio keitiklį, kuris maitinamas 24V nuolatinę įtampa ir turi standartinį 4-20mA išėjimo signalą proporcingą apkrovos svoriui. Pasirinktas „Flintec FAA-26“ keitiklis, kuris pavaizduotas 3.5 paveikslėlyje. Šio keitiklio 4-20mA išėjimo signalas sujungtas su centrinio valdiklio analoginių įėjimų modulių, kurio matavimų skiriamoji geba – 13 bitų.



3.5 pav. Svorio keitiklio jungikliai ir indikatoriai.

Sumontavus svorio jutiklį arba jį pakeitus nauju būtina sureguliuoti keitiklio išėjimo signalą nustatant išėjimo signalo pradinę vertę ir stiprinimo koeficientą. Siekiant geriausio matavimo rezultato rekomenduojama bioreaktorių pastatyti svėrimo platformos centre. Keitiklio derinimas atliekamas šiais veiksmais:

1. „Zero“ ir „Gain“ jungikliai nustatotmi ties „0“ žyma, indikatorius „Run“ šviečia, o „Err“ užgesęs.
2. Ant matavimo platformos uždedamas tuščias bioreaktorius. Taip bus nustatoma apatinė matavimo riba.
3. „Zero“ jungiklis pasukamas ir laikomas „-“ arba „+“ pusėje tol, kol matuojamas signalas pasiekia apatinę matavimų riba – 4mA.
4. „Zero“ jungiklis gražinamas į „0“ padėtį.
5. Nepajudinant bioreaktoriaus ant platformos uždedamas didžiausias svoris, kuris gali būti pasiekiamas darbo metu. Tam naudojami svarsčiai, bendras apkrovos svoris įsimenamas.
6. „Gain“ jungiklis pasukamas ir laikomas „-“ arba „+“ pusėje tol, kol matuojamas signalas pasiekia viršutinę matavimų riba – 20mA.
7. „Gain“ jungiklis gražinamas į „0“ padėtį.
8. Derinimo procedūra baigta.

Svorio keitiklis išduoda analoginį 4-20mA signalą, todėl būtina jį konvertuoti į svorio vertes pagal 3.2 lygtį:

$$m = \frac{m_{max} - m_{min}}{I_{max} - I_{min}} * (I - I_{min}) + b \qquad (3.2)$$

m_{min} –Svoris ant svėrimo platformos vykdant derinimo punktą Nr. 2, kg;

m_{max} –Svoris ant svėrimo platformos vykdant derinimo punktą Nr. 5, kg;

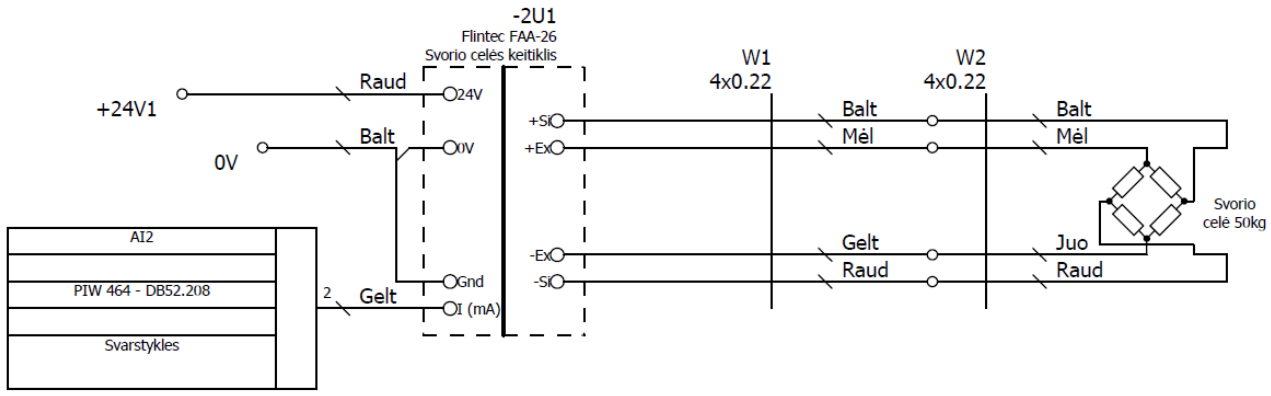
I_{min} – Apatinė 4-20mA analoginio signalo riba – 4mA;

I_{max} – Viršutinė 4-20mA analoginio signalo riba – 20mA;

I – išmatuojamas keitiklio išėjimo signalas, mA;

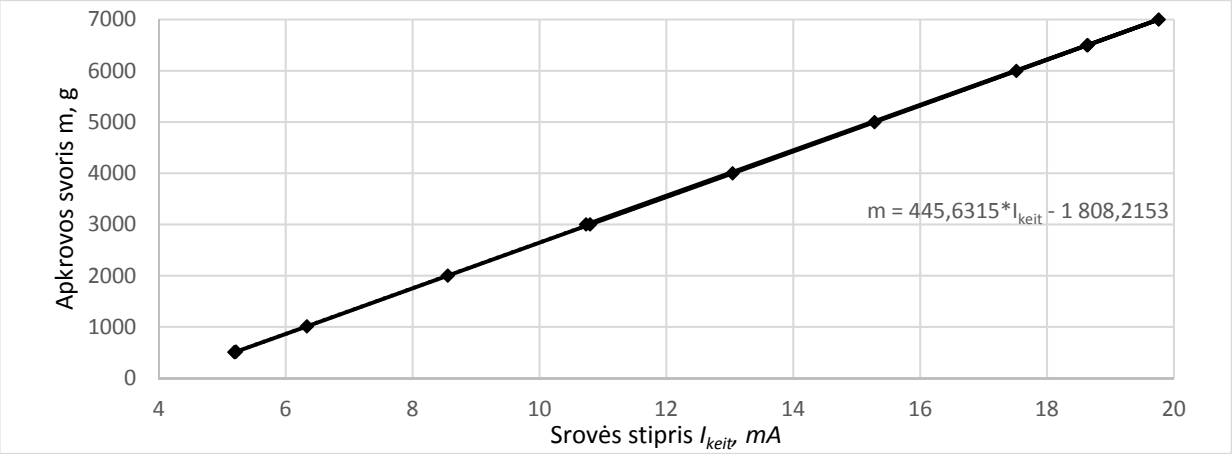
b – Tiesinės lygties narys, kuriu galima įvertint nulinio taško poslinkį, kg;

Svorio jutiklio keitiklis su bioreaktoriaus centrinio valdiklio analoginių įėjimų moduliu sujungtas pagal 3.6 paveikslėlyje pavaizduotą elektrinę schemą. Tuomet esama valdiklio programa modifikuota ir svėrimo platformos svoris gali būti operatoriaus nuskaitomas su kompiuteriu.



3.6 pav. Svėrimo platformos, keitiklio ir valdiklio sujungimo elektrinė schema

Svorio jutiklio keitiklis suderinamas pagal anksčiau aprašytą veiksmų seką. Tuščias bioreaktorius pastatomas platformos centre. Tai bus pradinės svėrimo skalės svoris, pasižymima m_{min}=0 kg. Įvykdžius derinimo instrukcijos punktus Nr. 1-4 pasižymima apatinė išmatuotos srovės reikšmė I_{min} =5,181 mA. Ant svėrimo platformos, po bioreaktoriumi, paeiliui dedami 0,5 kg arba 1 kg svarsčiai tol kol pasiekiamas numatomas darbinis bioreaktoriaus svoris su terpe ir papildoma galima įranga. Prieš uždedant, kiekvienas svarstis pasveriamas su laboratorijoje naudojamomis svarstyklėmis, kurių padalos vertė – 0,01g. Pasižymimas bendras apkrovos svoris m_{max}=6,9996 kg. Įvykdžius derinimo instrukcijos punktus Nr. 5-7 pasižymima viršutinė išmatuotos srovės reikšmė I_{max} =19,757 mA.



3.7 pav. Apkrovos svorio ir keitiklio išėjimo signalo kreivė

Keitiklio išėjimo signalas tiesiogiai ir tiesiškai proporcingas ant platformos uždėtam svoriui Gauta tendencijos kreivės lygtis *(angl. trend line equation)* $m=445,6315*I_{keit}-1808,2153$ kur m – terpės svoris ir I_{keit} – išmatuojamas keitiklio išėjimo srovės stipris. Sumontuota ir sukalibruota įranga bus naudojama tolimesnėse fermentacijose matuojant terpės tūrį.

3.1.4 Duomenų surinkimas ir perdavimas operatoriui

Centrinis valdiklis surenka ir perduoda duomenis į įrangos gamintojo tiekiamą vizualizacijos sistemą. Tai yra patogu standartinio proceso stebėjimui ir valdymui, tačiau riboja galimybes kurti įrankius išvestinių ir tiesiogiai nepamatuojamų parametrų įvertinimui. 2017 metais KTU automatikos katedroje buvo pradėtas vystyti projektas ir kompiuterinė programa „FeedingBox“, kuri pirmiausia automatizuoja pamaitinimo tirpalo tiekimą į bioreaktorių ir valdo jo srautą pagal sudarytą matematinį modelį. Tai yra modeliu paremtas *(angl. „Model based“)* valdymo algoritmas. Vystant projektą programos galimybės buvo praplėstos biomasės augimo greičio, sunaudojamo deguonies kiekio ir kitų verčių nustatymo įrankiais. Operatorius, duomenis reikalingus šiems įrankiams veikti, įvesdavo rankiniu būdų stebėdamas vizualizacijos sistemą. Taip pat buvo pastebėta, kad gamintojo tiekiama

vizualizacijos sistema veikia nepatikimai, atsiranda ryšio sutrikimų ir programos strigimų. Įvertintos įvairios galimybės automatiniam duomenų surinkimui iš bioreaktoriaus:

1. „OPC“ platforma – tai atviras standartas skirtas realaus laiko komunikacijoms tarp skirtingų gamintojų valdymo įrenginių. Sistemai su „OPC“ platforma reikalingi „OPC“ serveris, kuris iš centrinio valdiklio gautus duomenis persiunčia „OPC“ protokolu klientui, kuris nuskaito šiuos duomenis.
2. Iš vizualizacijos sistemos – bioreaktoriaus ir valdymo sistemos gamintojas aprašo metodus kaip vizualizacijos sistema tekstiniame dokumente periodiškai (kas 15 sekundžių) išsaugo iš anksto numatytus duomenis.
3. „Siemens S7“ – tai „Siemens“ įrenginių tarpusavio komunikavimo protokolas veikiantis „TCP“ duomenų pagrindu.

„OPC“ duomenų perdavimo variantas atmetamas dėl reikalingos papildomos programinės įrangos ir būtinybės turėti veikiančią gamintojo vizualizavimo sistemą. Duomenų nuskaitymas iš vizualizavimo sistemos išsaugojamų tekstinių dokumentų taip pat atmetamas dėl per mažos greitaveikos ir būtinybės turėti veikiančią gamintojo vizualizavimo sistemą. Pasirenkamas „Siemens S7“ protokolas, su kuriuo galima tiesiogiai nuskaityti duomenis iš valdiklio atmintyje esančių duomenų struktūrų, kurios šio gamintojo valdikliuose žymimos „DB## – DataBlock“ žymėjimu.

Pamaitinimo tirpalo tiekimo valdymo projektas „FeedingBox“ vystomas *C#* programavimo kalba. Dėl šios priežasties pasirenkama *„Snap7“* bibliotekos atmaina *„Sharp7“* perrašyta *C#* kalba. Ši biblioteka elgiasi kaip klientas, kuris vykdo komunikaciją su „Siemens“ valdikliu. Bibliotekoje išpildomos dauguma *„S7“* protokolo funkcijų: duomenų nuskaitymas ir įrašymas į įvairias atminties sritis, saugus prisijungimas naudojantis slaptažodžiais, valdiklio sustabdymas, paleidimas ir perkrovimas, užsakymo ir serijos numerio nuskaitymas iš valdiklio atminties jo identifikavimui.

Pagrindinės *„Sharp7“* bibliotekos funkcijos, kurios naudojamos prisijungimui ir duomenų nuskaitymui iš laboratorijoje fermentaciją kontroliuojančios valdymo sistemos valdiklio[21]:

public int ***ConnectTo****(String Address, int Rack, int Slot)* – Pabandoma prisijungti prie valdiklio, kurio IP adresas perduodamas į funkciją kintamuoju *„Address“*. Valdymo sistemos valdiklis - „Siemens IM151-8“. Tai yra „ET200S“ serijos valdiklis, kuris sukurtas „S7-314“ valdiklio pagrindu. „S7-300“ serijos valdikliuose gamintojo programinės įrangos konfigūravimo įrankyje *„Step7“* valdiklio vieta *(angl. slot)* ir eilė *(angl. rack)* visada būna atitinkamai pažymėtos *Rack 0* ir *Slot 2*. Jeigu pavyksta sėkmingai prisijungti prie valdiklio, funkcija gražina sveikojo skaičiaus reikšmę „0“. Tuomet galima naudoti kitas funkcijas duomenų nuskaitymui.

public void ***Disconnect()*** – Atsijungiama nuo valdiklio.

public int ***ReadArea****(int Area, int DBNumber, int Start, int Amount, int WordLen, byte[] Buffer)* – Tai pagrindinė duomenų nuskaitymo iš valdiklio funkcija. „S7“ protokolu galima nuskaityti duomenis iš skirtingų valdiklio atminties sričių, todėl iškviečiant šią funkciją būtina kintamuoju *„Area“* nurodyti iš kokios atminties srities nuskaitomi duomenis. Šios sritys ir jų pavadinimai aprašyti 3.1 lentelėje. Nuskaitant į duomenis iš duomenų struktūros su kintamuoju *„DBNumber“* nurodomas šios struktūros identifikavimo numeris. Kintamieji *„Amount“* ir *„Start“* atitinkamai nurodo kiek elementų ir nuo kurios atminties vietos pradėti skaityti duomenis. Loginių operacijų, sveikųjų skaičių, laikmačių ir skaitiklių, realiejų skaičių užrašymui reikalingas skirtingas atminties baitų kiekis. Kintamasis *„WordLen“* nurodo keliais baitais atmintyje išsaugomi nuskaitomi elementai. Nuskaityti duomenis grąžinami masyve *„Buffer“*.

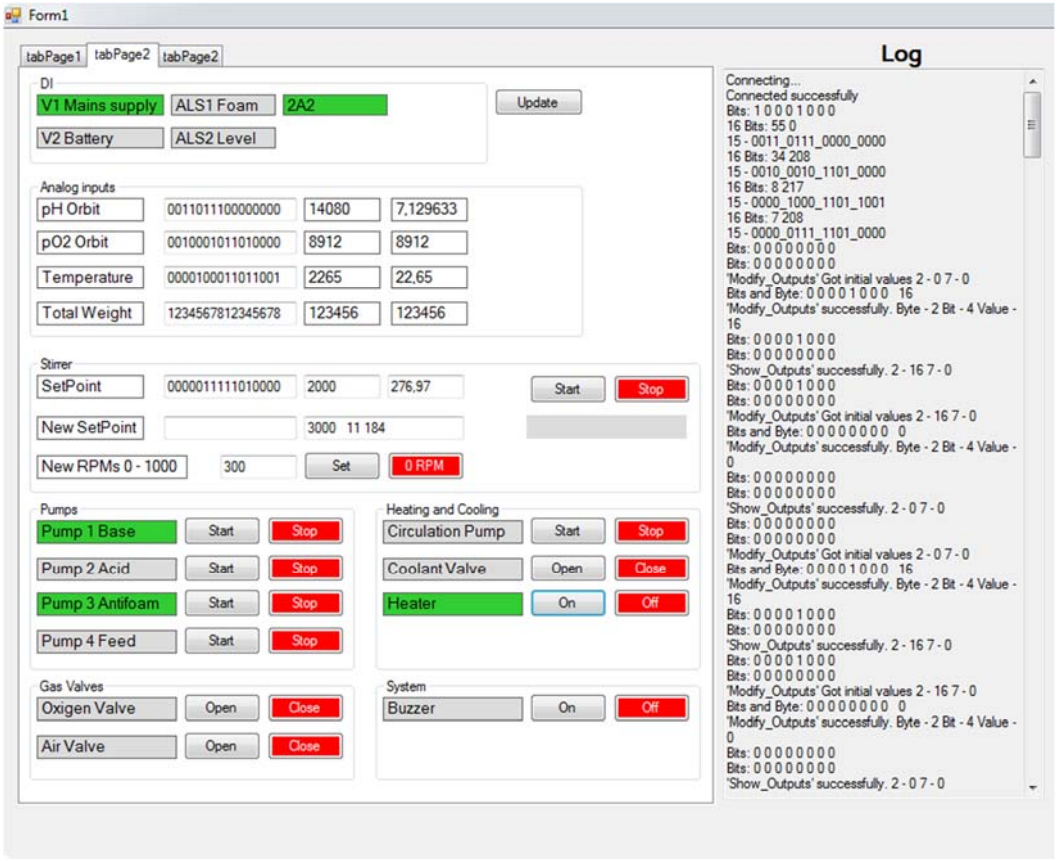
3.1 lentelė. Valdiklio atminties sritys

Atminties sritis	**Adresas**	**Paaiškinimas**
S7Consts.S7AreaPE	0x81	Valdiklio įėjimai
S7Consts.S7AreaPA	0x82	Valdiklio išėjimai
S7Consts.S7AreaMK	0x83	Atminties “markeriai”
S7Consts.S7AreaDB	0x84	Duomenų blokai“DB”
S7Consts.S7AreaCT	0x1C	Skaitikliai
S7Consts.S7AreaTM	0x1D	Laikmačiai

public int ***WriteArea****(int Area, int DBNumber, int Start, int Amount, int WordLen, byte[] Buffer)* – Tai pagrindinė duomenų įrašymo į valdiklį funkciją, kurios argumentai analogiški *„ReadArea“* funkcijai.

public int ***DBRead*** *(int DBNumber, int Start, int Amount, byte[] Buffer)* – Tai panaši funkcija į *„ReadArea“,* tačiau ši yra optimizuota duomenų nuskaitymui iš duomenų struktūra, ją iškviečiant nebereikia nurodyti norimos nuskaityti atminties srities ir nuskaitomo elemento dydžio.

Panaudojus šias aprašytas funkcijas sukurta bibliotekos patikrinimo programa *„C#“* kalba. Šios programos tikslas – prieš pradedant kurti įrankius automatiniam duomenų nuskaitymui iš valdiklio ir naudojimui „FeedingBox“ programoje patikrinti kaip veikia ši biblioteka, ar stabilus ryšis su valdikliu, ar galima stebėti ir kontroliuoti visus norimus bioreaktoriaus valdiklio parametrus. Paveikslėlyje 3.8 pavaizduoti visi bioreaktoriaus fizinių įėjimų ir išėjimų būsenos, sukurti mygtukai išėjimų valdymui ir užduoties reikšmių įvedimui. Šios programos kodas pateiktas 3 priede.



3.8 pav. Programa atvaizduojanti ir valdanti prie bioreaktoriaus valdiklio prijungtus įrenginius

Sėkmingai išbandžius biblioteką esama valdiklio programa paruošiama duomenų apsikeitimui naudojantis šio valdiklio programavimo įrankių *„Step7“* Taip pat suprogramuota funkcija, kuri nuskaito 3.1.3 punkte aprašytų svarstyklių signalą. Kintamieji, kurie reikalingi programos „FeedingBox“ naujoms funkcijoms, suprogramuojami į naują duomenų struktūrą, kuri visa bus nuskaitoma šios programos.

3.2 Parametrų identifiakvimas iš eksperimentinių duomenų

3.2.1 Sprendinių paieškos įrankis

Netiesinių lygčių sprendimas sudėtingas ir daugybės resursų reikalaujantis procesas. Matematinio modelio parametrų paieškai sudarytos lygtys aprašytos 2.3 poskyryje. Šie parametrai randami parenkant geriausius parametrų rinkinius, su kuriais paklaida arba klaidos suma būtų mažiausia.

Šių lygčių sprendimui galima naudoti „Microsoft Excel“ programinio paketo lygčių sprendinių paieškos priedą (*angl. „Solver“*). Aprašytos matematinės lygtis yra netiesinės, todėl sprendinio paieškai pasirenkamas sprendimo metodas *„GRG netiesinis“*. Šis metodas veikia pagal

„*Generalized Reduced Gradient*“ algoritmą[22]. Programos dialogo lange pasirenkamas tiriamos funkcijos rezultato sumos langelis, kurio reikšmę siekiama minimizuoti. Nurodomi laukeliai, kuriuose patalpinti tiriamos funkcijos parametrai, jie bus keičiami kad būtų pasiektas iškeltas tikslas. Sprendinių paieškos dialogo lange pravartu nurodyti taikomus apribojimus keičiamų kintamųjų vertėms, nes gali įvykti klaidos, kai bus bandoma dalinti iš nulio[23].

Sprendinių paieškos įrankiu bus apskaičiuojami matematinio modelio lygčių parametrai. Apskaičiuoti parametrai bus panaudoti išbandant matematinį modelį skaitmeninio modeliavimo programoje

3.2.2 Gliukozės lygties identifikavimas

2017-05-26 KTU biotechnologinių procesų modeliavimo ir valdymo laboratorijoje buvo atlikta fermentacija. Fermentacijos metų į protokolą buvo užrašomos ir kompiuterinėje valdymo programoje išsaugomos proceso vertės. Naudojant šiuos sukauptus duomenis ir 2.24 lygtį galima identifikuoti gliukozės kiekio tenkančio vienam gramui biomasės koeficientus, kurie yra naudojami gliukozės koncentracijos dinamikos lygtyje

3.2 lentelė. Duomenys naudoti išeigos koeficientų paieškai

Laikas nuo fermentacijos pradžios, val	**Gliukozės tiekimo greitis**	**Bendras augimo greitis**	**Biomasė**	**Kvadratinis skirtumas**
0	2,400	0,097	3,05	0,004
0,5	2,763	0,097	3,20	0,099
1	2,766	0,097	3,35	0,040
1,5	3,065	0,103	3,52	0,048
2	3,287	0,119	3,72	0,008
2,5	5,131	0,138	3,96	1,085
3,5	6,327	0,168	4,63	0,453
4	6,948	0,191	5,07	0,000
4,5	11,362	0,202	5,60	10,970
5	9,882	0,203	6,21	0,780
5,5	9,304	0,199	6,85	0,169
6	9,278	0,198	7,57	2,084
6,5	8,318	0,195	8,36	11,050
7	10,028	0,183	9,18	4,373
7,5	13,466	0,179	10,06	0,225
8	14,152	0,166	10,97	0,728
8,5	16,474	0,160	11,88	6,670
9	15,646	0,153	12,87	1,418
9,5	13,838	0,145	13,86	1,147
10	11,251	0,137	14,88	16,007
			Suma	**57,354**

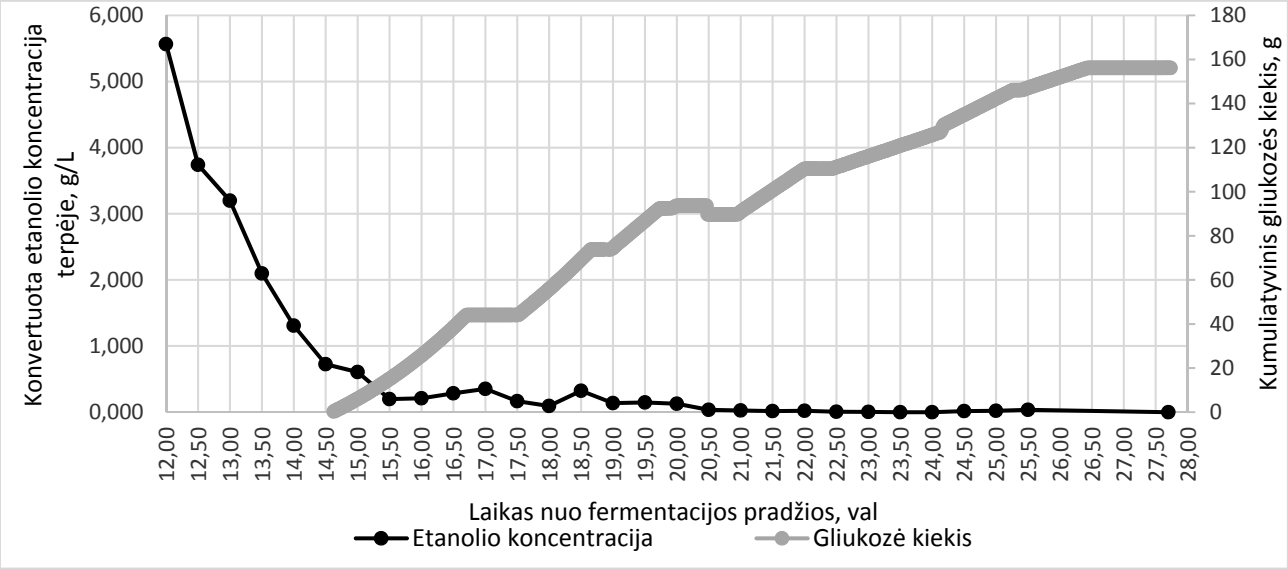
3.2 lentelėje pateikti duomenys, kurie buvo naudoti 2.24 lygtyje išeigos koeficientų paieškai mažiausių kvadratų metodu.

Panaudojus 3.2.1 punkte aprašytą verčių paieškos metodą gauta mažiausia kvadratinių skirtumų suma 57,354, kai $Y_{\frac{X}{S}}^{o}$=0,156 ir $Y_{\frac{X}{S}}^{f}$=0,107. Didžiausios kvadratinės paklaidos siekia 11-16 vienetų. Jos galėjo atsirasti dėl netikslių proceso metų išmatuotų reikšmių, vėlavimų bei kitų bioreaktoriuje vykstančių procesų, kurie nenumatyti tiriamame matematiniame modelyje. Šie gauti koeficientai gali būti panaudojami *„Simulink“* pakete sudarytame modelyje, kuris aprašytas 2.4 poskyryje.

3.2.3 Etanolinio augimo etapų išskyrimas aerobinėse sąlygose

Aiškiai išskyrus etanolinio augimo etapus galima identifikuoti išeigos koeficientus ir kitas matematinio modelio vertes. Šios vertės naudojamos etanolio bei deguonies dinamikos lygtyse, santykinio biomasės augimo aerobinėse sąlygose naudojant etanolį išraiškos lygtyje.

Esant gliukozės koncentracijai terpėje aukštesnei nei $S_{slenkst}$ (žr. 2.1 skyrių) vyksta etanolio fermentavimas. Etanolio koncentracijos ir kumuliatyvios gliukozės kiekio kitimas 2017-10-08 atliktos fermentacijos metu pavaizduotas 3.9 paveikslėlyje. Fermentacijos metu vykę procesai aprašyti 3.3 lentelėje pagal 3.9 paveikslėlį. Nustojus tiekti pamaitinimo tirpalą, mielės augdamos sunaudoja gliukozės likučius terpėje ir augimui pradeda naudoti terpėje esantį etanolį. Šios fermentacijos metu ryškiausiai išsiskiriami etanolinio augimo etapai. Etanolio koncentracijos matavimuose atsiranda iki 0,5 val. vėlavimas dėl žarnų bei dujų kondensavimosi inde, kuris yra tarp bioreaktoriaus ir alkotesterio.



3.9 pav. Etanolio koncentracijos ir kumuliatyvios gliukozės kiekio kitimas.

3.3 lentelė. Fermentacijos metu vykę procesai

Eil Nr	Laikas nuo fermentacijos pradžios		Biomasės augimas	Gliukozės tirpalas	Etanolio koncentracija
	Nuo	Iki			
1	12	14.626	Augimas aerobinėse sąlygose naudojant etanolį	Netiekiamas	Mažėja
2	14.626	17	Mišrus biomasės augimas naudojant gliukozę	Tiekiamas	Neapibrėžta
3	17	17.486	Augimas aerobinėse sąlygose naudojant etanolį	Netiekiamas	Mažėja
4	17.486	20.5	Mišrus biomasės augimas naudojant gliukozę	Tiekiamas	Neapibrėžta
5	20.5	21	Augimas aerobinėse sąlygose naudojant etanolį	Netiekiamas	Mažėja
6	21	22	Mišrus biomasės augimas naudojant gliukozę	Tiekiamas	Neapibrėžta
7	22	22.5	Augimas aerobinėse sąlygose naudojant etanolį	Netiekiamas	Mažėja
8	22.5	26.44	Mišrus biomasės augimas naudojant gliukozę	Tiekiamas	Neapibrėžta
9	26.44	27.736	Augimas aerobinėse sąlygose naudojant etanolį	Netiekiamas	Mažėja

Šie išskirti aerobinio biomasės augimo naudojant etanolį etapai bus panaudoti ieškant proceso verčių etanolio bei deguonies koncentracijos dinamikos lygtyse, santykinio biomasės augimo aerobinėse sąlygose naudojant etanolį išraiškoje.

3.2.4 Ištirpusio deguonies koncentracijos dinamikos lygtis

2.1 poskyryje aprašytame matematiniame modelyje biomasės suvartojamą deguonies kiekį aerobinio augimo metu naudojant etanolį nusako deguonies kiekio tenkančio vienam gramui biomasės išeigos koeficientas $Y_{\frac{X}{O}}^{oe}$. Išskyrus biomasės augimo naudojant etanolį etapus ir turint duomenis apie biomasės koncentracijos kitimo greičius, suvartojamo deguonies kiekį (*angl. sutrumpinimas OUR*) galima apskaičiuoti šiuos koeficientus naudojant 2.30 lygtį. 3.4 lentelėje pateiktos 2.30 lygtyje naudotos vertės ir apskaičiuotas rezultatas.

3.4 lentelė. $Y_{\frac{X}{O}}^{oe}$ koeficiento skaičiavimai

Eil. nr.	Pradžios laikas, val	Pabaigos laikas, val	X koncentracija pradžioje, g/L	X koncentracija pabaigoje, g/L	dX/dt, g/L/val	OUR Vidurkis	$Y_{\frac{X}{O}}^{oe}$
1	12.000	14.626	10.5500	13.8460	1.2551	1.9195	0.653
2	17.000	17.486	16.9700	17.6010	1.2984	1.9808	0.655
3	20.500	21.000	21.3330	21.7850	0.9040	1.6300	0.554
4	22.000	22.500	22.8030	23.2130	0.8200	1.3425	0.610
5	26.440	27.736	27.0610	28.1070	0.8071	1.3000	0.620
						Vidurkis	0.619
						Standartinis nuokrypis	0.037
						Pasikliautinumo intervalas	0.032

Išskiriamų etanolinio augimo intervalų pradžioje ir pabaigoje išmatuotos biomasės vertės pažymėtos atitinkamuose 3.4 lentelės stulpeliuose *X koncentracija pradžioje* ir *X koncentracija pabaigoje*. Biomasės koncentracijos kitimo greitis apskaičiuotas pagal biomasės pokytį etanolinio augimo intervale. *OUR* sunaudojamo deguonies kiekio vertės glodintos 3 valandų intervale ir šio intervalo vidurkis įrašytas į 3.4 lentelės *OUR vidurkis* stulpelį. $Y_{\frac{X}{O}}^{oe}$ stulpelyje apskaičiuojamas koeficientas pagal 2.30 lygtį. Vidutinė deguonies iš biomasės išeigos koeficiento $Y_{\frac{X}{O}}^{oe}$ vertė – 0.619 g/g.

3.2.5 Etanolio koncentracijos dinamikos lygtis

2.1 poskyryje aprašytame matematiniame modelyje etanolio kiekį, kurį suvartoja biomasė aerobinio augimo metu naudojant etanolį, nusako etanolio tenkančio vienam gramui biomasės išeigos koeficientai $Y_{\frac{X}{E}}^{oe}$. Išskyrus biomasės augimo naudojant etanolį etapus ir turint duomenis apie biomasės ir etanolio koncentracijos kitimo greičius galima apskaičiuoti šiuos koeficientus naudojant 2.33 lygtį. 3.5 lentelėje pateiktos 2.33 lygtyje naudotos vertės ir apskaičiuotas rezultatas. Etanolio kitimo greitis koreguotas atsižvelgiant į 2.2 poskyryje aprašyta matavimo prietaiso galimą jautrumą ne tik etanoliui, bet ir tarpiniams jo produktams.

3.5 lentelė. $Y_{\frac{X}{E}}^{oe}$ koeficiento skaičiavimai

Eil. nr.	Pradžios laikas, val	Pabaigos laikas, val	X koncentracija pradžioje, g/L	X koncentracija pabaigoje, g/L	dX/dt, g/hr/L	dE/dt, g/L/val	$Y_{\frac{X}{E}}^{oe}$
1	12,0	14,626	10,550	13,846	1,255	-4,136	0,303
2	17,0	17,486	16,970	17,601	1,298	-4,273	0,304
3	20,5	21,000	21,333	21,785	0,904	-3,003	0,301
4	22,0	22,500	22,803	23,213	0,820	-2,730	0,300
5	26,4	27,736	27,061	28,107	0,807	-2,652	0,304
						Vidurkis	0,303
						Standartinis nuokrypis	0,002
						Pasikliautinumo intervalas	0,001

Etanolinio augimo metu aerobinėmis sąlygomis etanolis buvo naudojamas biomasei augti, jo koncentracija mažėjo, todėl 3.5 lentelėje, $\frac{dE}{dt}$ stulpelyje, etanolio koncentracijos kitimo greitis neigiamas. $Y_{\frac{X}{E}}^{oe}$ stulpelyje apskaičiuojamas išeigos koeficientas pagal 2.33 lygtį. Vidutinė etanolio tenkančio vienam gramui biomasės koeficiento, vertė – 0,303 g/g.

3.2.6 Biomasės augimo greičio aerobinėse sąlygose naudojant etanolį lygtis.

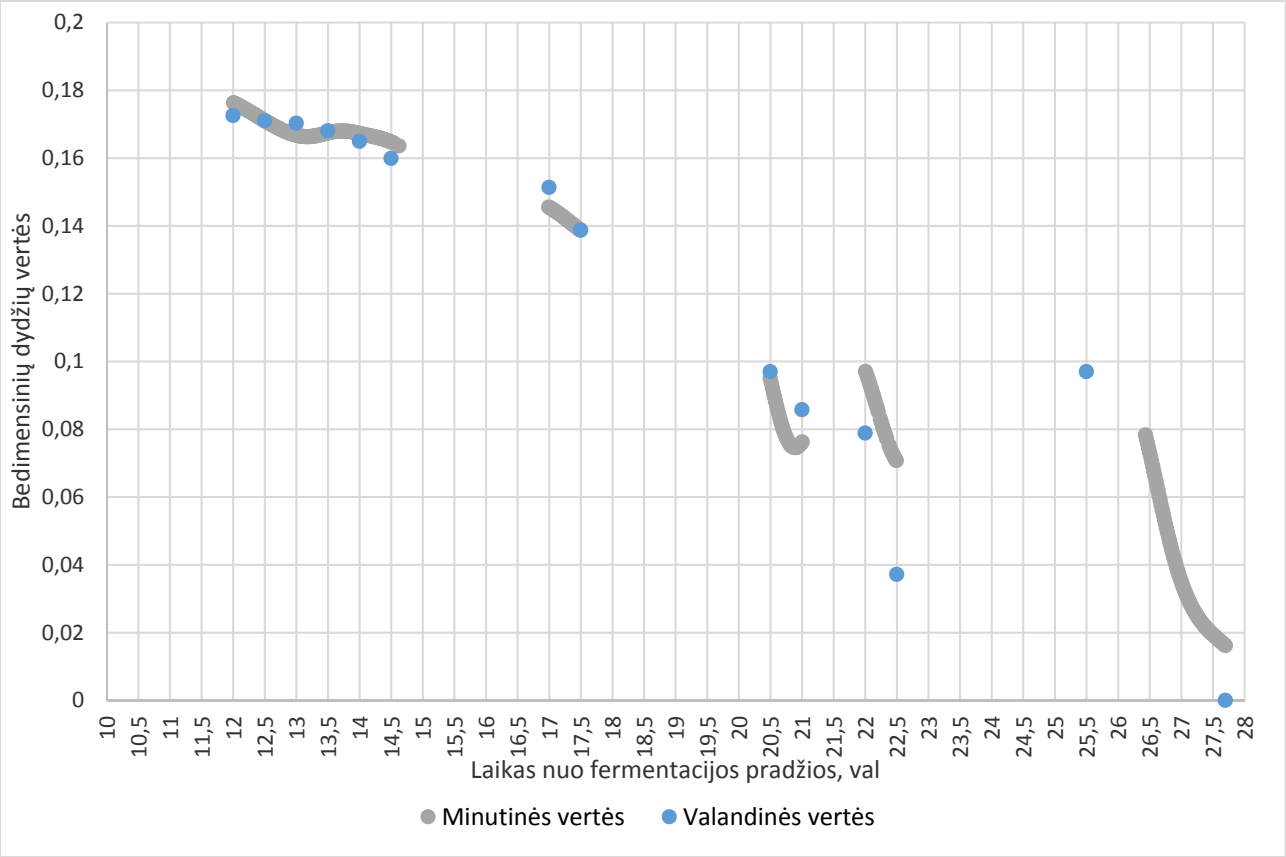
Išskyrus biomasės augimo naudojant etanolį etapus galima nustatyti santykinio augimo greičio lygties parametrus: maksimalų augimo greitį, „Monod“ ir limitavimo koeficientus. Šie parametrai bus reikalingi modeliuojant bioreaktoriaus veikimą, kai biomasė auga naudodama etanolį. Vykdant fermentaciją sukaupti duomenys aprašyti 3.2.3 punkte. Naudojant 2.35 funkciją galima ieškoti jos parametrų skaičiuojant skirtumą tarp valandiniu ir minutiniu intervalu kauptų duomenų.

Panaudojant 3.2.1. poskyryje aprašyta netiesinių lygčių sprendimo įrankį išbandomi įvairūs 2.35 funkcijos kintamieji. Mažiausia paklaidos suma gauta, kai $\mu_{\max\_eo} = 0{,}18$, $K_e^o = 0{,}077$, $K_X^{eo} = 20$. 3.6 lentelėje pateikti funkcijoje 2.35 naudoti duomenys ir apskaičiuotas skirtumas tarp minutinių ir valandiniu intervalais apskaičiuotų reikšmių pateiktas 3.6 lentelėje.

3.6 lentelė. Δ_{mju_eo} koeficiento skaičiavimai

Eil. nr.	**Laikas nuo fermentacijos pradžios, val**	**Biomasės koncentracija g/L**	**Biomasės augimo greitis val^-1**	**Etanolio koncentracija g/L**	Δ_{mju_eo}
1	12.00	10.5140	0.1155	1.86	0.0082
2	12.50	11.1252	0.1098	1.51	0.0036
3	13.00	11.7351	0.1050	1.39	0.0006
4	13.50	12.3562	0.1035	1.11	0.0003
5	14.00	13.0128	0.1013	0.86	0.0000
6	14.50	13.6804	0.0978	0.62	0.0001
7	17.00	16.9571	0.0787	0.41	0.0132
8	17.50	17.6199	0.0737	0.26	0.0100
9	20.50	21.4449	0.0458	0.09	0.0126
10	21.00	21.6389	0.0366	0.07	0.0158
11	22.00	22.8046	0.0453	0.06	0.0000
12	22.50	27.1426	0.0310	0.02	0.0129
13	27.70	27.6061	0.0068	0.00	0.0086
				Suma($\Delta_{mju\_eo\_suma}$)	**0.103**

Valandiniai ir minutiniai bedimensiniai dydžiai grafiškai atvaizduoti 3.10 pav.



3.10 pav. Bedimensinių valandinių ir minutinių verčių sulyginimas

Paveikslėlyje 3.10 matyti, kad minutinių ir valandinių duomenų tiesės yra sąlyginai panašios. Pilka minutinių valandų juosta gauta, dėl didelio matavimo duomenų kiekio, o mažas kiekis mėlynų taškų dėl retai atliekamų etanolio koncentracijos išeinančiose dujose matavimų. Skirtumai tarp šių kreivių galėjo atsirasti dėl matavimų paklaidų ir vėlinimų.

Mažiausia kvadratų suma Suma($\Delta_{mju\_eo\_suma}$)=0,103 gauta kai $\mu_{max\_eo} = 0{,}18$, $K_E^{eo} = 0{,}077$, $K_X^{eo} = 20$. Šie koeficientai bus panaudoti *„Simulink“* pakete sudarytame modelyje, kuris aprašytas 2.4 poskyryje ir gautos modelio vertės palygintos su realiai išmatuotomis vertėmis.

3.3 Matematinio modelio išbandymas ir pritaikymas

3.3.1 Matematinio modelio ir realios sistemos palyginimas

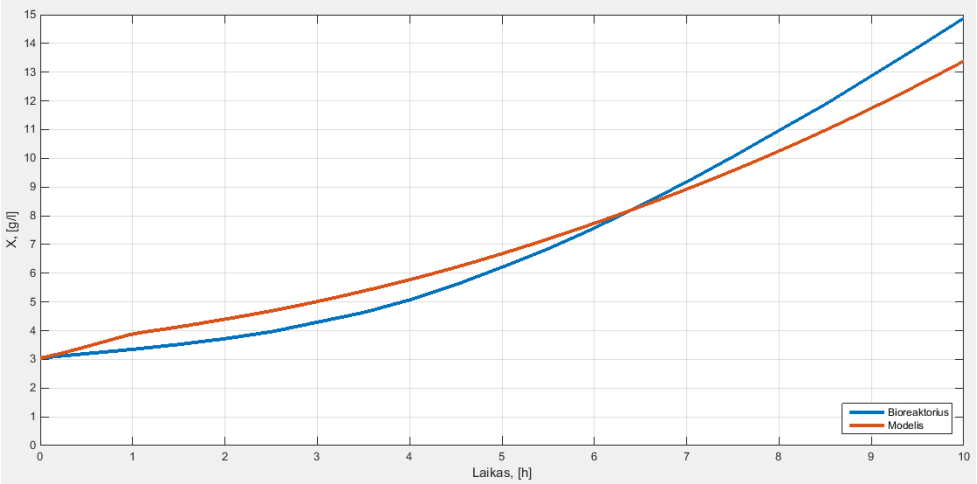
Sudaryto matematinio modelio dalis, kuri modeliuoja biomasės koncentracijos dinamiką palyginta su 2017-05-26 KTU biotechnologinių procesų modeliavimo ir valdymo laboratorijoje atliktos fermentacijos biomasės kitimo eiga. Fermentacija buvo vykdoma su 3.1 poskyryje aprašyta įranga.

Matematinio modelio, kurio sudarymas aprašytas 2.4 poskyryje, parametrai pateikti 3.7 lentelėje.

3.7 lentelė. Matematinio modelio parametrai.

„Simulink“ kintamasis	**Vertė**	**Matavimo vienetai**
miumax_so	0,25	g/l
miumax_sf	0,05	g/l
miumax_eo	0,18	g/l
Kx_eo	20	
kLa	400	
Yxso	0,16	g/g
Yxsf	0,11	g/g
Yxef	2	g/g
Yxeoe	0,25	g/g
Yxoo	1,2	g/g
Yxooe	0,619	g/g
Yxcf	1,2	g/g
Yxco	1	g/g
Yxcoe	1,5	g/g
kLa_alpha	0.00000184327	
kLa_beta	2.9693226	
kLa_gama	1	
Q_srautas	5,6	l/min
prV	3,4	l
prC	0	g/l
prX	3,4	g/l
prS	0	g/l
prE	1,9	g/l
prO	29	g/l

Biomasės kitimo eigos duomenys buvo įkelti į modelį naudojantis verčių lentele. Matematinio modelio ir realių duomenų palyginimas pavaizduotas 3.11 paveikslėlyje. Paveikslėlyje mėlyna linija žymi duomenis gautus iš realaus bioreaktoriaus, o oranžinė – matematinio modelio vertes.



3.11 pav. Matematinio modelio ir realios sistemos biomasės kitimo palyginimas

Pastebėti vizualūs panašumai palyginus matematinio modelio ir realios fermentacijos biomasės kitimo eiga. Vidutinė absoliutinė paklaida tarp šių kreivių duomenų apskaičiuojamas 3.3 lygtimi[25] ir yra lygi $\bar{\Delta}_{\%} = 9{,}7\%$.

$$\bar{\Delta}_{\%} = \frac{\sum_{i=1}^{n}\left(\frac{|X_{realus}(i) - X_{mod}(i)|}{X_{realus}(i)} * 100\%\right)}{n} \tag{3.3}$$

$\bar{\Delta}_{\%}$– Vidutinis skirtumas tarp realaus ir sumodeliuoto biomasės kitimo, %;

$X_{realus}(i)$– Realios biomasės vertė, g/l;

$X_{mod}(i)$– Sumodeliuotos biomasės vertė, g/l;

i– Iteracijos indeksas;

n– Duomenų kiekis;

3.3.2 Optimalaus substrato profilio paieška

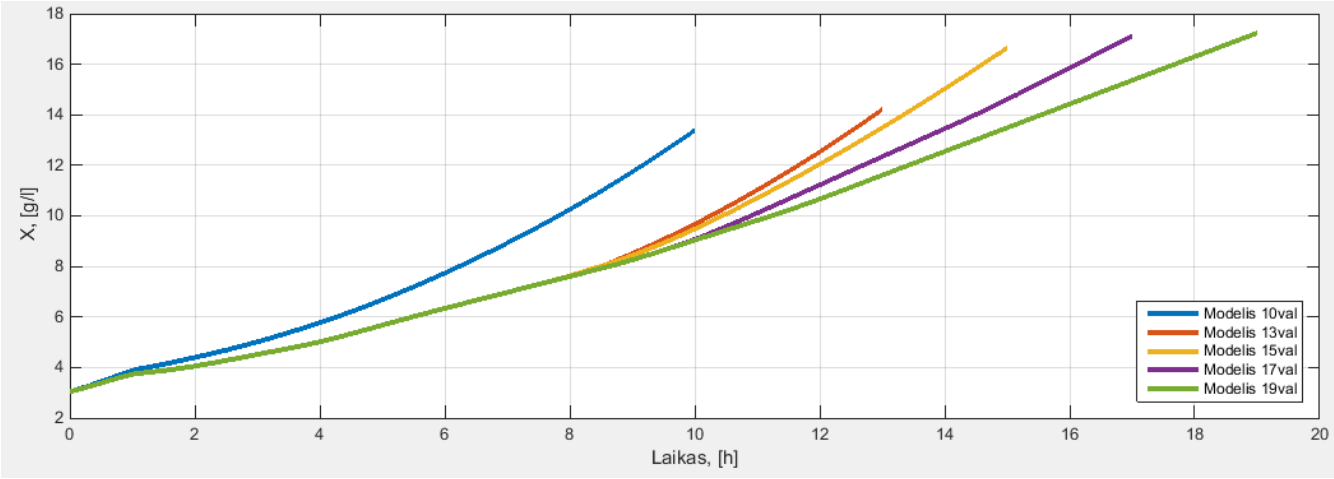
Biotechnologinių procesų pramonėje siekiama gauti kuo didesnį biomasės kiekį mažiausiais kaštais. Šiuos kaštus sudaro pasiruošimo darbai prieš fermentaciją, cheminių medžiagų ir substrato išeiga, operatorių darbas. Atsižvelgiant į ankstesniame poskyryje gautą sąlyginai mažą skirtumą tarp matematinio modelio ir 2017-05-26 KTU biotechnologinių procesų modeliavimo duomenų daroma prielaida, kad 2.4 poskyryje aprašytas modelis su ankstesniame poskyryje 3.7 lentelėje pateiktais parametrais yra adekvatus 2017-05-26 vykusiai fermentacijai.

Mielių santykinio augimo greičio išraiškos aprašytos 2.1 poskyryje. Augimo greitis priklauso nuo mielių kultūros, bioreaktoriaus ypatybių bei maistinių medžiagų tiekimo, kurios yra reikalingos augimui.

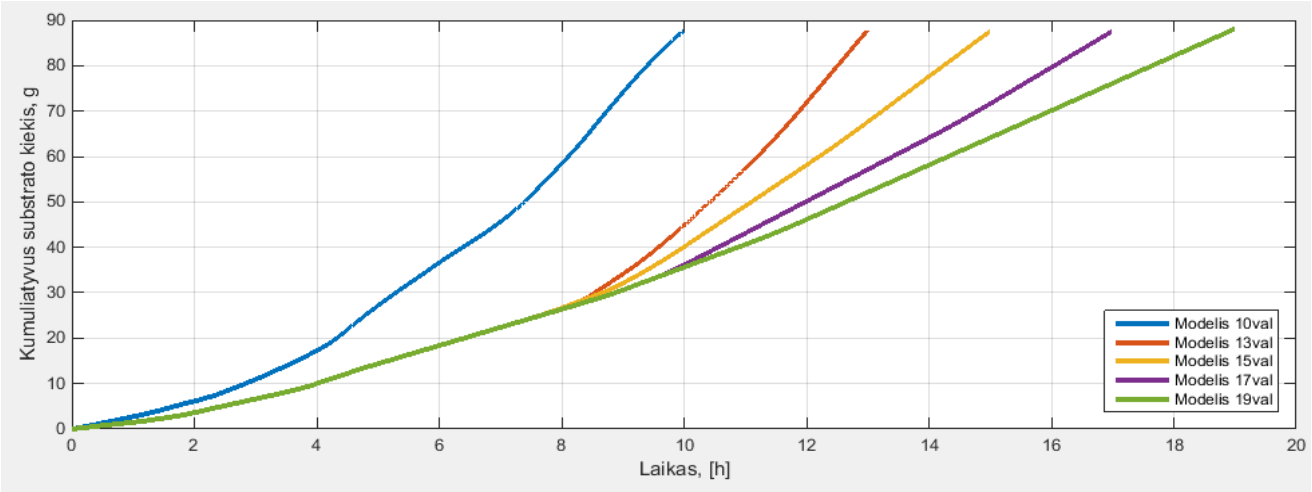
Sunaudojant tą patį substrato kiekį, bet jį lėčiau tiekiant skirtingais profiliais gaunama didesnė biomasės išeiga fermentacijos pabaigoje. Palaikant žemą gliukozės koncentraciją terpėje ir tiekiant orą galimai nevyksta rūgimas ir biomasė auga aerobinėse sąlygose naudojant gliukozę. Pasiruošimo darbai fermentacijai, mielių auginimas kolboje prieš inokuliuojant jas į bioreaktorių, bei darbai po fermentacijos gali užtrukti iki 36-48 valandų. Fermentacijos procesą, kuris vyksta bioreaktoriuje, pratęsiant nuo pradinių 10 iki 19 valandų biomasės kiekis fermentacijos pabaigoje būtų gaunamas vis didesnis, o bendra visos fermentacijos trukmė pailgėtų tik apytiksliai 20%. Šis fermentacijos prailginimas leistų gauti didesnę ekonominę naudą iš kiekvienos fermentacijos.

Fermentaciją pratesus nuo pradinių 10 valandų iki 13, 15, 17 ir 19 valandų atitinkamai gauta 6, 24, 28, 28 procentais didesnė biomasė. Fermentaciją vykdyti ilgiau kaip 17 valandų nenaudinga, nes augimas sulėtėja dėl didėjančios biomasės koncentracijos (žr. 1.2 poskyrį ir 2.1.2 punktą).

Modeliavimų biomasės kitimo kreivės pavaizduotos 3.12 pav., o supumpuoto kumuliatyvaus substrato kiekio profiliai – 3.13 pav.



3.12 pav. Biomasės kitimas esant skirtingai fermentacijos trukmei.



3.13 pav. Kumuliatyvaus substrato profiliai esant skirtingai fermentacijos trukmei.

Iš biomasės kitimo kreivių (3.12 pav.) manoma, kad lėčiau vykdžius 2017-05-26 vykusią fermentaciją buvo galima užauginti didesnį biomasės kiekį sunaudojant tą patį substrato kiekį. 2.1 poskyryje aprašyta ir 2.4 *„Simulink“* pakete sudaryta modelį galima naudoti substrato tiekimo profilio modeliavimui siekiant gauti didžiausią biomasės išeigą fermentacijos pabaigoje.

IŠVADOS

1. Apžvelgtas mielių augimo procesas, skirtingi augimo etapai ir auginimo metodai.
2. Detaliai aprašomos pasirinkto matematinio modelio atskirų medžiagų koncentracijos dinamikos lygtis.
3. Aprašytas matematinis modelis sudaromas *„Simulink“* modeliavimo programoje naudojant universalius blokus, kurie leidžia toliau plėtoti šį modelį ir kitose modeliavimo programose.
4. Sudaromi teoriniai gliukozės, ištirpusio deguonies, etanolio koncentracijos dinamikos lygčių bei biomasės augimo greičio išraiškos aerobinėse sąlygose naudojant etanolį metodai leidžiantis identifikuoti dalį šių lygčių parametrų. Šie kintamieji buvo identfikuoti panaudojus realių fermentacijų duomenis.
5. Aprašomas KTU biotechnologinių procesų modeliavimo ir valdymo laboratorijoje naudojamas bioreaktorius. Parenkama, sumontuojama, suprogramuojama ir sukalibruojama svėrimo platforma. Sumontuota platforma leidžia suskaičiuoti terpės turį ir panaudoti tai kuriant naujus programinius įrankius fermentacijos valdymui. Aprašyti ir sukalibruoti alkotesteris bei dujų analizatorius. Šie prietaisai yra tinkami netiesioginiam medžiagų matavimui terpėje.
6. Palyginama *„Simulink“* modelio ir realios fermentacijos biomasės kitimo eiga, vidutininė absoliutinė paklaida (*angl. MAPE*) – 9,7%. Sudarytas modelis galimai adekvatus palygintos fermentacijos eigai. Panaudojus šį modelį sudaryti alternatyvus lėtesni substrato tiekimo profiliai, kurie manomai, būtų leidę gauti 28% didesnę biomasės išeigą fermentacijos pabaigoje sunaudojant tą patį gliukozės kiekį.

LITERATŪROS ŠALTINIAI

1. FDA/CDER „SMALL BUSINESS CHRONICLES“ „Ne Drug Quality“, 2012 September 18th. [žiūrėta 2018 m. Gegužės 18 d.] Prieiga internetu : https://www.fda.gov/downloads/Drugs/DevelopmentApprovalProcess/SmallBusinessAssistance/UCM319879.pdf00
2. DIMŠIENĖ, Ieva. MIELĖS BIOLOGIŠKAI AKTYVIŲ - ORGANINIŲ JUNGINIŲ SINTEZĖJE, 2017 m.
3. BOEKHOUT, T.; ROBERT, V. Yeasts in Food: Beneficial and Detrimental aspects. Behr's Verlag. p. 322. ISBN 978-3-86022-961-3, 2003 m.
4. „Kingdom Fungi“ [žiūrėta 2018 m. sausio 5 d.] Prieiga internetu : https://www.askiitians.com/biology/biological-classification/kingdom-fungi.html
5. „Monitoring Growth of Beer Brewing Strains of Saccharomyces Cerevisiae - The Utility of Synergy H1 for Providing High Quality Kinetic Data for Yeast Growth Applications“ [žiūrėta 2018 m. sausio 8 d.] Prieiga internetu : https://www.biotek.com/resources/application-notes/monitoring-growth-of-beer-brewing-strains-of-saccharomyces-cerevisiae-the-utility-of-synergy-h1-for-providing-high-quality-kinetic-data-for-yeast-growth-applications/
6. „Bacterial growth curve“ [žiūrėta 2018 m. sausio 8 d.] Prieiga internetu : http://www.onlinebiologynotes.com/bacterial-growth-curve/
7. „Difference between batch, fed-batch and continuous culture technique“ [žiūrėta 2018 m. sausio 8 d.] Prieiga internetu : http://www.onlinebiologynotes.com/difference-batch-fed-batch-continuous-culture-technique/
8. MACHADO, Carlos, Pedro GOMES, Rui SOARES, Silvia PEREITA ir Filomena O. SOARES. Control of baker‘s yeast fermentation: PID and fuzzy algorithms. [žiūrėta 2017 m. lapkričio 23 d.] Prieiga internetu: http://repositorium.sdum.uminho.pt/bitstream/1822/6250/1/IECON_Filomena Soares%2528Final%2529.pdf
9. BARANSKAS, Deividas. „MODELIAVIMO APLINKOS, BIOTECHNOLOGINIŲ PROCESŲ ANALIZEI IR TYRIMAMS, SUKŪRIMAS“ 2017 m.
10. „How to measure the dissolved oxygen level of water“ [žiūrėta 2018 m. kovo 14 d.] Prieiga internetu: https://www.wikihow.com/Measure-the-Dissolved-Oxygen-Level-of-Water
11. „Dissolved Oxygen Fact Sheet“ “ [žiūrėta 2018 m. kovo 14 d.] Prieiga internetu: http://www.gsweventcenter.com/GSW_RTC_References/2004_0929_SWRCB_CleanWaterTeam.pdf

12. „LAQUA- Ways of Measuring pH“ [žiūrėta 2018 m. kovo 16 d.] Prieiga internetu: http://www.horiba.com/sg/application/material-property-characterization/water-analysis/water-quality-electrochemistry-instrumentation/ph-knowhow/the-story-of-ph/ways-of-measuring-ph/
13. „Dough Fermentation & Temperature“ [žiūrėta 2018 m. kovo 19 d.] Prieiga internetu: http://www.theartisan.net/dough_fermentation_and_temperature.htm
14. ALEKSA, Algiment. „Matematiak 1 – mažiausių kvadratų metodas“ [žiūrėta 2018 m. balandžio 10 d.] Prieiga internetu: https://www.personalas.ktu.lt/~algalek/files_mat1/maziausiu%20kvadr%20metodas.pdf
15. SURVYLA, Arnas. „Bioproceso deguonies sunaudojamo greičio estimatoriaus vystymas ir verifikavimas“ 2017 m.
16. Masinė žinduolinių lastelių gamyba [žiūrėta 2017 m. balandžio 15 d.] Prieiga internetu: http://www.imcdb.lt/courses/927/
17. JSC Biotehniskais centrs „User manual EDF-5.4 Laboratory bioreactor“
18. „Finesse – Sterilizable Sensors“ [žiūrėta 2018 m. kovo 16 d.] Prieiga internetu: http://www.finesse.com/media/181025/809-DST-001-Rev2_SterilizableSensors.pdf
19. „Dissolved Oxygen Sensors“ [žiūrėta 2018 m. kovo 16 d.] Prieiga internetu: https://wiki.metropolia.fi/display/sensor/Dissolved+Oxygen+Sensors
20. „Grove - Multichannel Gas Sensor“ [žiūrėta 2018 m. sausio 11 d.] Prieiga internetu: https://www.seeedstudio.com/Grove-Multichannel-Gas-Sensor-p-2502.html
21. „Step7 Open Source Ethernet Communication Suite „ [žiūrėta 2017 m. balandžio 20 d.] Prieiga internetu: http://snap7.sourceforge.net/
22. „Mathematical programming Glossary - Generalized reduced gradient method „[žiūrėta 2018 m. gegužės 3 d.] Prieiga internetu: https://glossary.informs.org/ver2/mpgwiki/index.php/Generalized_reduced_gradient_method
23. „Uždavinio apibrėžimas ir sprendimas, naudojant Sprendimo paiešką“ [žiūrėta 2018 m. gegužės 3 d.] Prieiga internetu: https://support.office.com/lt-lt/article/u%C5%BEdavinio-apibr%C4%97%C5%BEimas-ir-sprendimas-naudojant-sprendimo-paie%C5%A1k%C4%85-5d1a388f-079d-43ac-a7eb-f63e45925040?NS=EXCEL&Version=16&SysLcid=1063&UiLcid=1063&AppVer=ZXL160&HelpId=xladdin.chm1830&ui=lt-LT&rs=lt-LT&ad=LT

24. ”Monod equation“ [žiūrėta 2018 m. balandžio 4 d.] Prieiga internetu: https://en.wikipedia.org/wiki/Monod_equation
25. „Ekonometrika. Progrnozavimo uždavinio samrpata. MAPE rodiklis“ [žiūrėta 2018 m. gegužės 7 d.] Prieiga internetu: http://www.ilab.lt/stabingiene/sk8_5.html

PRIEDAI

Priedas Nr. 1 Alkotesterio kalibravimo duomenys

Eiles Nr	Etanolio koncentracija terpėje, g/l	Alkotesterio parodymai, promilėmis
0	0,00E+00	0,00
1	8,28E-05	0,21
2	1,66E-04	0,25
3	2,48E-04	0,40
4	3,31E-04	0,41
5	4,14E-04	0,28
6	4,97E-04	0,43
7	5,79E-04	0,46
8	6,62E-04	0,45
9	7,45E-04	0,52
10	8,27E-04	0,60
11	9,10E-04	0,58
12	9,93E-04	0,60
13	1,08E-03	0,69
14	1,16E-03	0,74
15	1,24E-03	0,71
16	1,32E-03	0,82
17	1,41E-03	0,80
18	1,49E-03	0,83
19	1,57E-03	0,88
20	1,65E-03	0,88
21	1,74E-03	0,90
22	1,82E-03	0,86
23	1,90E-03	0,89
24	1,98E-03	0,95
25	2,07E-03	0,97
26	2,15E-03	0,95
27	2,23E-03	1,02
28	2,31E-03	0,98
29	2,40E-03	1,04
30	2,48E-03	1,09

Priedas Nr. 2 Dujų analizatoriaus kalibravimo duomenys

Eiles Nr	Etanolio koncentracija terpėje, g/l	Dujų analizatoriaus parodymai, ppm
0	0,00E+00	0,0
1	8,28E-05	0,5
2	1,66E-04	1,0
3	2,48E-04	1,5
4	3,31E-04	2,0
5	4,14E-04	2,5
6	4,97E-04	3,0
7	5,79E-04	3,5
8	6,62E-04	4,0
9	7,45E-04	4,5
10	8,27E-04	5,0
11	9,10E-04	5,5
12	9,93E-04	6,0
13	1,08E-03	6,5
14	1,16E-03	7,0
15	1,24E-03	7,5
16	1,32E-03	8,0
17	1,41E-03	8,5
18	1,49E-03	9,0
19	1,57E-03	9,5
20	1,65E-03	10,0
21	1,74E-03	10,5
22	1,82E-03	11,0
23	1,90E-03	11,5
24	1,98E-03	12,0
25	2,07E-03	12,5
26	2,15E-03	13,0
27	2,23E-03	13,5
28	2,31E-03	14,0
29	2,40E-03	14,5
30	2,48E-03	15,0

Priedas Nr. 3 Bandomosios bioreaktoriaus valdymo programos kodas

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;
using Sharp7;
namespace WindowsFormsApplication2
{
    public partial class Form1  : Form
    // public partial class TTB_dec : Form
    {
        private S7Client client;
        // Outputs array for function
        Boolean[] Outputs2 = { false, false, false, false, false, false, false, false };
        Boolean[] Outputs7 = { false, false, false, false, false, false, false, false };
        Boolean initial = true;
        public Form1()
        {
            InitializeComponent();
            client = new S7Client();
        }
        //Connecto to PLC
        private void B_Connect_Click(object sender, EventArgs e)
        {
            int Rack = System.Convert.ToInt32(TB_RackID.Text);
            int Slot = System.Convert.ToInt32(TB_SlotID.Text);
            TB_Log.Text += "Connecting..." + Environment.NewLine;
            int connectionResult = client.ConnectTo(TB_IPaddress.Text, Rack, Slot);
            if (connectionResult == 0)
            {
                TB_Log.Text += "Connected successfully" + Environment.NewLine;
                TB_IPaddress.Enabled = false;
                TB_RackID.Enabled = false;
                TB_SlotID.Enabled = false;
                B_Connect.Enabled = false;
                B_Disconnect.Enabled = true;
                B_Get_status.Enabled = true;
                B_ReadInput.Enabled = true;
                B_Write.Enabled = true;
                B_Update_Dout.Enabled = true;
                B_StartPLC.Enabled = true;
                B_StopPLC.Enabled = true;
                Read_CPUinfo_OrderCode();
            }
            else
            {
                TB_Log.Text += "Connection failed" + Environment.NewLine;
            }
            //LA_Conn_status pakeitimas
            if (client.Connected)
            {
                LA_Conn_status.Text = "Online";
                LA_Conn_status.ForeColor = System.Drawing.Color.LimeGreen;
            }
            else
            {
                LA_Conn_status.Text = "Offline";
                LA_Conn_status.ForeColor = System.Drawing.Color.Black;
            }
        }   // private void B_Connect_Click
        //Disconnect from PLC
        private void B_Disconnect_Click(object sender, EventArgs e)
        {
            if (client.Connected)
            {
                client.Disconnect();
                if (client.Connected)
                {
                    LA_Conn_status.Text = "Online";
                    LA_Conn_status.ForeColor = System.Drawing.Color.Red;
                    TB_Log.Text += "Disconnection failed" + Environment.NewLine;
                }
                else
                {
                    LA_Conn_status.Text = "Offline";
                    LA_Conn_status.ForeColor = System.Drawing.Color.Black;
                    TB_Log.Text += "Disconnected successfully" + Environment.NewLine;
                    TB_IPaddress.Enabled = true;
                    TB_RackID.Enabled = true;
                    TB_SlotID.Enabled = true;
                    B_Connect.Enabled = true;
```

```
            B_StartPLC.Enabled = false;
            B_StopPLC.Enabled = false;
            B_Disconnect.Enabled = false;
            B_Get_status.Enabled = false;
            B_ReadInput.Enabled = false;
            B_Write.Enabled = false;
            B_Update_Dout.Enabled = false;
        }
    }
} //    private void B_Disconnect_Click
private void B_Get_status_Click(object sender, EventArgs e)
{
    GetPlcStatus();
} // B_Get_status_Click
private void B_ReadInput_Click(object sender, EventArgs e)
{
    int start = System.Convert.ToInt32(TB_byte_nr.Text);
    int amount = System.Convert.ToInt32(TB_DI_byte_q.Text);
    byte[] buf = new byte[10];
    Boolean[] bit_buf = { false, false, false, false, false, false, false, false };
    int Read_Res = client.EBRead(start, amount, buf);
    if (Read_Res == 0)
    {
        TB_Log.Text += "Byte value - ";
        for (int i = 0; i < amount; i++)
        {
            TB_Log.Text += buf[i] + " ";
        }
        TB_Log.Text += Environment.NewLine;
        ToBit(bit_buf, buf[0]);
        ToBit(bit_buf, buf[1]);
        ToBit(bit_buf, buf[0]);
        // bit 0
        if (bit_buf[0])
        {
            LS_I_0.BackColor = System.Drawing.Color.LimeGreen;
        }
        else
        {
            LS_I_0.BackColor = System.Drawing.Color.Gainsboro;
        }
        // bit 1
        if (bit_buf[1])
        {
            LS_I_1.BackColor = System.Drawing.Color.LimeGreen;
        }
        else
        {
            LS_I_1.BackColor = System.Drawing.Color.Gainsboro;
        }
        // bit 2
        if (bit_buf[2])
        {
            LS_I_2.BackColor = System.Drawing.Color.LimeGreen;
        }
        else
        {
            LS_I_2.BackColor = System.Drawing.Color.Gainsboro;
        }
        // bit 3
        if (bit_buf[3])
        {
            LS_I_3.BackColor = System.Drawing.Color.LimeGreen;
        }
        else
        {
            LS_I_3.BackColor = System.Drawing.Color.Gainsboro;
        }
        // bit 4
        if (bit_buf[4])
        {
            LS_I_4.BackColor = System.Drawing.Color.LimeGreen;
        }
        else
        {
            LS_I_4.BackColor = System.Drawing.Color.Gainsboro;
        }
        // bit 5
        if (bit_buf[5])
        {
            LS_I_5.BackColor = System.Drawing.Color.LimeGreen;
        }
        else
        {
            LS_I_5.BackColor = System.Drawing.Color.Gainsboro;
        }
        // bit 6
        if (bit_buf[6])
```

```
            {
                LS_I_6.BackColor = System.Drawing.Color.LimeGreen;
            }
            else
            {
                LS_I_6.BackColor = System.Drawing.Color.Gainsboro;
            }
            // bit 7
            if (bit_buf[7])
            {
                LS_I_7.BackColor = System.Drawing.Color.LimeGreen;
            }
            else
            {
                LS_I_7.BackColor = System.Drawing.Color.Gainsboro;
            }
        } //result check
    }   //B_ReadInput_Click
    private void B_Update_Dout_Click(object sender, EventArgs e)
    {
        int start = System.Convert.ToInt32(TB_DO_byte_nr.Text);
        int amount = System.Convert.ToInt32(TB_DO_byte_q.Text);
        byte[] buf = new byte[10];
        Boolean[] bit_buf = { false, false, false, false, false, false, false, false };
        int Read_Res = client.ABRead(start, amount, buf);
        if (Read_Res == 0)
        {
            TB_Log.Text += "Byte value - ";
            for (int i = 0; i < amount; i++)
            {
                TB_Log.Text += buf[i] + " ";
            }
            TB_Log.Text += Environment.NewLine;
            ToBit(bit_buf, buf[0]);
            ToBit(bit_buf, buf[1]);
            ToBit(bit_buf, buf[0]);
            // bit 0
            if (bit_buf[0])
            {
                LS_O_0.BackColor = System.Drawing.Color.LimeGreen;
            }
            else
            {
                LS_O_0.BackColor = System.Drawing.Color.Gainsboro;
            }
            // bit 1
            if (bit_buf[1])
            {
                LS_O_1.BackColor = System.Drawing.Color.LimeGreen;
            }
            else
            {
                LS_O_1.BackColor = System.Drawing.Color.Gainsboro;
            }
            // bit 2
            if (bit_buf[2])
            {
                LS_O_2.BackColor = System.Drawing.Color.LimeGreen;
            }
            else
            {
                LS_O_2.BackColor = System.Drawing.Color.Gainsboro;
            }
            // bit 3
            if (bit_buf[3])
            {
                LS_O_3.BackColor = System.Drawing.Color.LimeGreen;
            }
            else
            {
                LS_O_3.BackColor = System.Drawing.Color.Gainsboro;
            }
            // bit 4
            if (bit_buf[4])
            {
                LS_O_4.BackColor = System.Drawing.Color.LimeGreen;
            }
            else
            {
                LS_O_4.BackColor = System.Drawing.Color.Gainsboro;
            }
            // bit 5
            if (bit_buf[5])
            {
                LS_O_5.BackColor = System.Drawing.Color.LimeGreen;
            }
            else
            {
```

```
                LS_O_5.BackColor = System.Drawing.Color.Gainsboro;
            }
            // bit 6
            if (bit_buf[6])
            {
                LS_O_6.BackColor = System.Drawing.Color.LimeGreen;
            }
            else
            {
                LS_O_6.BackColor = System.Drawing.Color.Gainsboro;
            }
            // bit 7
            if (bit_buf[7])
            {
                LS_O_7.BackColor = System.Drawing.Color.LimeGreen;
            }
            else
            {
                LS_O_7.BackColor = System.Drawing.Color.Gainsboro;
            }
        } //result check
    }   // B_Update_Dout_Click
    private void B_Write_Click(object sender, EventArgs e)
    {
        int start = System.Convert.ToInt32(TB_DO_byte_nr.Text);
        byte[] buf = new byte[1];
        Boolean[] bit_buf = { false, false, false, false, false, false, false, false };
        if (CB_O_0.Checked)
        {
            bit_buf[0] = true;
        }
        if (CB_O_1.Checked)
        {
            bit_buf[1] = true;
        }
        if (CB_O_2.Checked)
        {
            bit_buf[2] = true;
        }
        if (CB_O_3.Checked)
        {
            bit_buf[3] = true;
        }
        if (CB_O_4.Checked)
        {
            bit_buf[4] = true;
        }
        if (CB_O_5.Checked)
        {
            bit_buf[5] = true;
        }
        if (CB_O_6.Checked)
        {
            bit_buf[6] = true;
        }
        if (CB_O_7.Checked)
        {
            bit_buf[7] = true;
        }
        ToByte(bit_buf, out buf[0]);
        int Write_Res = client.ABWrite(start, 1, buf);
        if (Write_Res == 0)
        {
            TB_Log.Text += "Output written successfully" + Environment.NewLine;
        }
        B_Update_Dout.PerformClick();
    }   //B_Write_Click
    //tab2 button update
    private void button1_Click(object sender, EventArgs e)
    {
        //int start = System.Convert.ToInt32(TB_byte_nr.Text);
        //int amount = System.Convert.ToInt32(TB_DI_byte_q.Text);
        byte[] buf_di = new byte[1];
        byte[] buf_db11 = new byte[330];
        //int Read_ResDI = client.EBRead(start, amount, buf);
        //if (Read_ResDI == 0)
        //queries to plc
        int Read_ResDI = client.EBRead(1, 1, buf_di);
        int Read_ResDB11 = client.DBRead(11, 218, 330, buf_db11); // (11, 218, 330, buf_db11)
        //Conversion byte to bit
        if (Read_ResDI == 0)
        {
            Boolean[] bit_buf_DI = { false, false, false, false, false, false, false, false };
            ToBit(bit_buf_DI, buf_di[0]);
            // bit 0
            if (bit_buf_DI[0])
            {
                LA_mainsOK.BackColor = System.Drawing.Color.LimeGreen;
```

```
                    }
                    else
                    {
                        LA_mainsOK.BackColor = System.Drawing.Color.Gainsboro;
                    }
                    // bit 1
                    if (bit_buf_DI[1])
                    {
                        LA_battery.BackColor = System.Drawing.Color.LimeGreen;
                    }
                    else
                    {
                        LA_battery.BackColor = System.Drawing.Color.Gainsboro;
                    }
                    // bit 2
                    if (bit_buf_DI[2])
                    {
                        LA_foam.BackColor = System.Drawing.Color.LimeGreen;
                    }
                    else
                    {
                        LA_foam.BackColor = System.Drawing.Color.Gainsboro;
                    }
                    // bit 3
                    if (bit_buf_DI[3])
                    {
                        LA_level.BackColor = System.Drawing.Color.LimeGreen;
                    }
                    else
                    {
                        LA_level.BackColor = System.Drawing.Color.Gainsboro;
                    }
                    // bit 4
                    if (bit_buf_DI[4])
                    {
                        LA_relay.BackColor = System.Drawing.Color.LimeGreen;
                    }
                    else
                    {
                        LA_relay.BackColor = System.Drawing.Color.Gainsboro;
                    }
                } //IF read_resDI
                if (Read_ResDB11 == 0)
                {
                    // pH
                    double scale_pH = 0.000506366;
                    double val_pH = 0;
                    Boolean[] bit_buf_pH = { false, false, false, false, false, false, false, false, false,
false, false, false, false, false, false, false };
                    string bits_pH = "";
                    int dec_pH = 0;
                    TB_phOrbit.Text = "";
                    ToBit16(bit_buf_pH, buf_db11[280], buf_db11[280 + 1], out dec_pH, ref bits_pH);
                    val_pH = scale_pH * dec_pH;
                    TB_phOrbit.Text = bits_pH;
                    LS_pH_dec.Text = dec_pH.ToString();
                    LS_pH_val.Text = val_pH.ToString();
                    // pO2
                    double scale_pO2 = 1;
                    double val_pO2 = 0;
                    Boolean[] bit_buf_pO2 = { false, false, false, false, false, false, false, false, false,
false, false, false, false, false, false, false };
                    string bits_pO2 = "";
                    int dec_pO2;
                    TB_pO2Orbit.Text = "";
                    ToBit16(bit_buf_pO2, buf_db11[320], buf_db11[320 + 1], out dec_pO2, ref bits_pO2);
                    val_pO2 = scale_pO2 * dec_pO2;
                    TB_pO2Orbit.Text = bits_pO2;
                    LS_pO2_dec.Text = dec_pO2.ToString();
                    LS_pO2_val.Text = val_pO2.ToString();
                    //Temperature
                    double scale_T = 0.01;
                    double val_T = 0;
                    Boolean[] bit_buf_T = { false, false, false, false, false, false, false, false, false,
false, false, false, false, false, false, false };
                    string bits_T = "";
                    int dec_T;
                    TB_Temperature.Text = "";
                    ToBit16(bit_buf_T, buf_db11[240], buf_db11[240 + 1], out dec_T, ref bits_T);
                    val_T = scale_T * dec_T;
                    TB_Temperature.Text = bits_T;
                    LS_Temperature_dec.Text = dec_T.ToString();
                    LS_Temperature_val.Text = val_T.ToString();
                }
                //Stirrer
                double scale_SP_a = 0.0629;    // val=a*DEC+b
                double scale_SP_b = 151.17;
                double val_SP = 0;
```

```
                    Boolean[] bit_buf_Stirrer = { false, false, false, false, false, false, false, false, false,
false, false, false, false, false, false, false };
                    string bits_Stirrer = "";
                    int dec_Stirrer;
                    int Stirrer_PQW_result;
                    byte[] buf_pqw_stirrer = new byte[2];
                    TB_SP.Text = "";
                    Stirrer_PQW_result = client.ABRead(432, 2, buf_pqw_stirrer);
                    if (Stirrer_PQW_result == 0)
                    {
                        ToBit16(bit_buf_Stirrer, buf_pqw_stirrer[0], buf_pqw_stirrer[0 + 1], out dec_Stirrer, ref
bits_Stirrer);
                        val_SP = scale_SP_a * dec_Stirrer + scale_SP_b;
                        TB_SP.Text = bits_Stirrer;
                        TB_SP_dec.Text = dec_Stirrer.ToString();
                        TB_SP_val.Text = val_SP.ToString();
                    }
                    else
                    {
                        TB_Log.Text += "Error while reading stirrer PQW - " + client.ErrorText(Stirrer_PQW_result)
+ Environment.NewLine;
                    }
                    TB_Log.ScrollToCaret();
                }   //update button
                    //-------------------
                void Read_CPUinfo_OrderCode()
                {
                    S7Client.S7OrderCode OC_Info = new S7Client.S7OrderCode();
                    S7Client.S7CpuInfo CI_Info = new S7Client.S7CpuInfo();
                    TB_CPUinfo.Text = "";
                    int OC_Result = client.GetOrderCode(ref OC_Info);
                    int CI_Result = client.GetCpuInfo(ref CI_Info);
                    if (OC_Result == 0 && CI_Result == 0)
                    {
                        TB_CPUinfo.Text += CI_Info.ModuleTypeName + "\r\n";  // CPU 314C...
                        TB_CPUinfo.Text += CI_Info.SerialNumber + Environment.NewLine;  // CPU serial nr.
                        TB_CPUinfo.Text += CI_Info.ASName + Environment.NewLine;  // CPU S7 300
                        TB_CPUinfo.Text += CI_Info.ModuleName + Environment.NewLine;  // From TIA
                        TB_CPUinfo.Text += OC_Info.Code;  // CPU order code 6ES...
                    }
                } //Read_CPUinfo_OrderCode
                void GetPlcStatus()
                {
                    int Status = 0;
                    int Result = client.PlcGetStatus(ref Status);
                    //ShowResult(Result);
                    if (Result == 0)
                    {
                        switch (Status)
                        {
                            case S7Consts.S7CpuStatusRun:
                                {
                                    LA_CPU_status.Text = "RUN";
                                    LA_CPU_status.ForeColor = System.Drawing.Color.LimeGreen;
                                    break;
                                }
                            case S7Consts.S7CpuStatusStop:
                                {
                                    LA_CPU_status.Text = "STOP";
                                    LA_CPU_status.ForeColor = System.Drawing.Color.Red;
                                    break;
                                }
                            default:
                                {
                                    LA_CPU_status.Text = "Unknown";
                                    LA_CPU_status.ForeColor = System.Drawing.Color.Black;
                                    break;
                                }
                        }
                    }
                    else
                    {
                        LA_CPU_status.Text = "???";
                        LA_CPU_status.ForeColor = System.Drawing.Color.Black;
                    }
                }   //GetPlcStatus
                void ToBit(Boolean[] array, byte value)
                {
                    TB_Log.Text += "Bits: ";
                    byte[] powers = { 1, 2, 4, 8, 16, 32, 64, 128 };
                    for (int i = 7; i >= 0; i--)
                    {
                        if (value / powers[i] > 0)
                        {
                            array[i] = true;
                            value -= powers[i];
                        }
                        else
```

```
                {
                    array[i] = false;
                }
            }
            for (int i = 0; i <= 7; i++)
            {
                if (array[i])
                {
                    TB_Log.Text += "1 ";
                }
                else
                {
                    TB_Log.Text += "0 ";
                }
            }
            TB_Log.Text += Environment.NewLine;
        } //ToBit
        void ToBit16(Boolean[] array, byte valueH, byte valueL, out int valueDec, ref string bits)
        {
            TB_Log.Text += "16 Bits: " + valueH + " " + valueL + Environment.NewLine;
            byte[] powers = { 128, 64, 32, 16, 8, 4, 2, 1 }; //{ 1, 2, 4, 8, 16, 32, 64, 128 };
            UInt16[] powers_to_dec = { 0, 16384, 8192, 4096, 2048, 1024, 512, 256, 128, 64, 32, 16, 8, 4,
2, 1 };
            int i;
            // Convert from two bytes to 16bit binary
            for (i = 0; i <= 7; i++)        // Low byte, right, second
            {
                if (valueH / powers[i] > 0)
                {
                    array[i] = true;
                    valueH -= powers[i];
                    bits += "1";
                }
                else
                {
                    array[i] = false;
                    bits += "0";
                }
            }
            for (i = 8; i <= 15; i++)    //  High byte, left, first
            {
                if (valueL / powers[i-8] > 0)
                {
                    array[i] = true;
                    valueL -= powers[i-8];
                    bits += "1";
                }
                else
                {
                    array[i] = false;
                    bits += "0";
                }
            }
            //Convert to signed decimal
            valueDec = 0;
            if (array[0])    //  value negative
            {
                for (i = 1; i <= 15; i++)   // skip first bit in array
                    if (!array[i])
                    {
                        valueDec -= powers_to_dec[i];
                    }
                valueDec -= 1;
            }
            else            // value positive
            {
                for (i = 1; i <= 15; i++)   // skip first bit in array
                {
                    if (array[i])
                    {
                        valueDec += powers_to_dec[i];
                    }
                }
            }
            // Dump output
            if (true)
            {
                TB_Log.Text += "15 - ";
                for (i = 0; i <= 15; i++)
                {
                    if (array[i])
                    {
                        TB_Log.Text += "1";
                    }
                    else
                    {
                        TB_Log.Text += "0";
                    }
```

```
                            if (i == 3 || i == 7 || i == 11)
                            {
                                TB_Log.Text += "_";
                            }
                        }
                        TB_Log.Text += Environment.NewLine;
                    }
                } //ToBit16_senas
                void To2Bytes(int valueDec, out byte valueH, out byte valueL, ref string bits)
                {
                    valueH = 0;
                    valueL = 0;
                    int valueDec_dump = valueDec;
                    UInt16[] powers_to_bin = { 0, 16384, 8192, 4096, 2048, 1024, 512, 256, 128, 64, 32, 16, 8, 4,
2, 1 };
                    byte[] powers = { 128, 64, 32, 16, 8, 4, 2, 1 };
                    Boolean[] array = { false, false, false, false,  false, false, false, false, false, false,
false, false, false, false, false, false };
                    Byte i;
                    // Convert decimal to 16 bit binary
                    if (valueDec < 0)    //  value negative
                    {
                        array[0] = true;
                        valueDec += 1;
                        for (i = 1; i <= 15; i++)   // skip first bit in array
                        {
                            if (valueDec / -powers_to_bin[i] > 0)
                            {
                                array[i] = false;
                                bits += "0";
                                valueDec += powers_to_bin[i];
                            }
                            else
                            {
                                array[i] = true;
                                bits += "1";
                            }
                        }
                    }
                    else            // value positive
                    {
                        array[0] = false;
                        for (i = 1; i <= 15; i++)   // skip first bit in array
                        {
                            if (valueDec / powers_to_bin[i] > 0)
                            {
                                array[i] = true;
                                bits += "1";
                                valueDec -= powers_to_bin[i];
                            }
                            else
                            {
                                array[i] = false;
                                bits += "0";
                            }
                        }
                    }
                    //Convert from 16 bit binary to 2 bytes
                    for (i = 0; i <= 7; i++)        // High byte, left, first
                    {
                        if (array[i])
                        {
                            valueH += powers[i];
                        }
                    }
                    for (i = 8; i <= 15; i++)        // Low byte, right, second
                    {
                        if (array[i])
                        {
                            valueL += powers[i-8];
                        }
                    }
                    // Dump output
                    if (true)
                    {
                        TB_Log.Text += "To2Byte decimal " + valueDec_dump + " valueH " + valueH + " valueL " +
valueL + Environment.NewLine;
                        for (i = 0; i <= 15; i++)
                        {
                            if (array[i])
                            {
                                TB_Log.Text += "1";
                            }
                            else
                            {
                                TB_Log.Text += "0";
                            }
                            if (i == 3 || i == 7 || i == 11)
```

```
                {
                    TB_Log.Text += "_";
                }
            }
            TB_Log.Text += Environment.NewLine;
        }
    }   //To2Bytes
    void ToBit16_blogas(Boolean[] array, byte valueH, byte valueL, out int valueDec, ref string bits)
    {
        valueDec = valueH * 256 + valueL;
        int valueDec_bit;
        if (valueDec > 32768)    // is value negative?
        {
            array[0] = true;
            bits += "1";
            valueDec -= 32768;
            valueDec_bit = valueDec;
            valueDec *= -1;
        }
        else
        {
            array[0] = false;
            bits += "0";
            valueDec_bit = valueDec;
        }
        TB_Log.Text += "16 Bits: " + valueH + " " + valueL + " " + valueDec + Environment.NewLine;
        UInt16[] powers = { 0, 16384, 8192, 4096, 2048, 1024, 512, 256, 128, 64, 32, 16, 8, 4, 2, 1
};
        byte i;
        //bits += "MSB - ";
        for (i = 1; i <= 15; i++)   // skip first bit in array
        {
            if (valueDec_bit / powers[i] > 0)
            {
                array[i] = true;
                bits += "1";
                valueDec_bit -= powers[i];
            }
            else
            {
                array[i] = false;
                bits += "0";
            }
        }
        //bits
        // to LOG dump
        if (true)
        {
            TB_Log.Text += "15 - ";
            for (i = 0; i <= 15; i++)
            {
                if (array[i])
                {
                    TB_Log.Text += "1";
                }
                else
                {
                    TB_Log.Text += "0";
                }
                if (i == 3 || i == 7 || i == 11)
                {
                    TB_Log.Text += "_";
                }
            }
            TB_Log.Text += Environment.NewLine;
        }
    } //ToBit16
    void ToBit16_backup(Boolean[] array, byte valueH, byte valueL, out int valueDec, ref string bits)
    {
        valueDec = valueH * 256 + valueL;
        int valueDec_bit = valueDec;
        TB_Log.Text += "16 Bits: " + valueH + " " + valueL + " " + valueDec + Environment.NewLine;
        UInt16[] powers = { 32768, 16384, 8192, 4096, 2048, 1024, 512, 256, 128, 64, 32, 16, 8, 4, 2,
1 };
        byte i;
        //bits += "MSB - ";
        TB_Log.Text += "..   ";
        for (i = 0; i <= 15; i++)
        {
            if (valueDec_bit / powers[i] > 0)
            {
                array[i] = true;
                bits += "1";
                valueDec_bit -= powers[i];
            }
            else
            {
                array[i] = false;
```

```
                        bits += "0";
                    }
                }
                TB_Log.Text += "..   ";
                //bits
                // to LOG dump
                if (true)
                {
                    TB_Log.Text += "15 - ";
                    for (i = 0; i <= 15; i++)
                    {
                        if (array[i])
                        {
                            TB_Log.Text += "1";
                        }
                        else
                        {
                            TB_Log.Text += "0";
                        }
                        if (i == 3 || i == 7 || i == 11)
                        {
                            TB_Log.Text += "_";
                        }
                    }
                    TB_Log.Text += Environment.NewLine;
                }
            } //ToBit16 laikinas
            void ToByte(Boolean[] array, out byte value)
            {
                TB_Log.Text += "Bits and Byte: ";
                byte[] powers = { 1, 2, 4, 8, 16, 32, 64, 128 };
                value = 0;
                for (int i = 0; i <= 7; i++)
                {
                    if (array[i])
                    {
                        value += powers[i];
                        TB_Log.Text += "1 ";
                    }
                    else
                    {
                        TB_Log.Text += "0 ";
                    }
                }
                TB_Log.Text += "  " + value + Environment.NewLine;
            }     // ToByte
            void Modify_Outputs ( byte byte_nr, byte bit, Boolean value, Boolean initial, ref Boolean[] byte_1,
ref Boolean[] byte_2)
            {
                int i;
                byte[] buf = new byte[10];
                byte[] buf_out = new byte[1];
                byte[] buf_off = { 0, 0, 0, 0, 0, 0, 0, 0, 0, 0 };
                if (initial && client.Connected)    //Get values from PLC on start
                {
                    int Read_Res = client.ABRead(0, 9, buf);
                    if (Read_Res == 0)
                    {
                        ToBit(byte_1, buf[2]);
                        ToBit(byte_2, buf[7]);
                        initial = false;
                        TB_Log.Text += "'Modify_Outputs' Got initial values 2 - " + buf[2] + " 7 - " + buf[7]
+ Environment.NewLine;
                    }
                }
                if (bit >= 0 && bit <= 7 && client.Connected)
                {
                    int Write_Res = 1;
                    switch (byte_nr)
                    {
                        case 0:
                            {
                                Write_Res = client.ABWrite(2, 6, buf_off);
                                if (Write_Res == 0)
                                {
                                    TB_Log.Text    +=    "'Modify_Outputs'    Stoped    successfully"    +
Environment.NewLine;
                                    // Call function to modify all labels
                                    Show_Outputs(ref byte_1, ref byte_2);
                                }
                                break;
                            }
                        case 2:
                            {
                                byte_1[bit] = value;
                                ToByte(byte_1, out buf_out[0]);
                                Write_Res = client.ABWrite(2, 1, buf_out);
                                if (Write_Res == 0)
```

```
                                {
                                    TB_Log.Text += "'Modify_Outputs' successfully. Byte - " + byte_nr + " Bit
- " + bit + " Value - " + buf_out[0] + Environment.NewLine;
                                    // Call function to modify all labels
                                    Show_Outputs(ref byte_1, ref byte_2);
                                }
                                break;
                            }
                        case 7:
                            {
                                byte_2[bit] = value;
                                ToByte(byte_2, out buf_out[0]);
                                Write_Res = client.ABWrite(7, 1, buf_out);
                                if (Write_Res == 0)
                                {
                                    TB_Log.Text += "'Modify_Outputs' successfully. Byte - " + byte_nr + " Bit
- " + bit + " Value - " + buf_out[0] + Environment.NewLine;
                                    // Call function to modify all labels
                                    Show_Outputs(ref byte_1, ref byte_2);
                                }
                                break;
                            }
                        // neveikia default
                    }
                }
            }
            void Show_Outputs(ref Boolean[] byte_1, ref Boolean[] byte_2)
            {
                if (client.Connected)
                {
                    byte[] buf = new byte[10];
                    Boolean[] bit_buf = { false, false, false, false, false, false, false, false };
                    int Read_Res = client.ABRead(0, 9 , buf);
                    if (Read_Res == 0)
                    {
                        ToBit(byte_1, buf[2]);
                        ToBit(byte_2, buf[7]);
                        TB_Log.Text += "'Show_Outputs' successfully. 2 - " + buf[2] + " 7 - " + buf[7] +
Environment.NewLine;
                        //                      PLC output Byte 2 (byte_1[])
                        if (byte_1[0])  // Pump1 - LA_Pump1
                        {
                           LA_Pump1.BackColor = System.Drawing.Color.LimeGreen;
                        }
                        else
                        {
                            LA_Pump1.BackColor = System.Drawing.Color.Gainsboro;
                        }
                        if (byte_1[1])  // Pump2 - LA_Pump2
                        {
                            LA_Pump2.BackColor = System.Drawing.Color.LimeGreen;
                        }
                        else
                        {
                            LA_Pump2.BackColor = System.Drawing.Color.Gainsboro;
                        }
                        if (byte_1[2])  // Pump3 - LA_Pump3
                        {
                            LA_Pump3.BackColor = System.Drawing.Color.LimeGreen;
                        }
                        else
                        {
                            LA_Pump3.BackColor = System.Drawing.Color.Gainsboro;
                        }
                        if (byte_1[3])  // Pump4 - LA_Pump4
                        {
                            LA_Pump4.BackColor = System.Drawing.Color.LimeGreen;
                        }
                        else
                        {
                            LA_Pump4.BackColor = System.Drawing.Color.Gainsboro;
                        }
                        if (byte_1[4])  // Stirrer start-stop - LA_Stirrer_Status
                        {
                            LA_Stirrer_Status.BackColor = System.Drawing.Color.LimeGreen;
                        }
                        else
                        {
                            LA_Stirrer_Status.BackColor = System.Drawing.Color.Gainsboro;
                        }
                        if (byte_1[5])  // Heating/Cooling Circulation pump - LA_CircPump
                        {
                            LA_CircPump.BackColor = System.Drawing.Color.LimeGreen;
                        }
                        else
                        {
                            LA_CircPump.BackColor = System.Drawing.Color.Gainsboro;
                        }
```

```
                        if (byte_1[6])  // Heating/Cooling Coolant Valve - LA_CoolValve
                        {
                            LA_CoolValve.BackColor = System.Drawing.Color.LimeGreen;
                        }
                        else
                        {
                            LA_CoolValve.BackColor = System.Drawing.Color.Gainsboro;
                        }
                        if (byte_1[7])  // Heating/Cooling Electrical heater - LA_Heater
                        {
                            LA_Heater.BackColor = System.Drawing.Color.LimeGreen;
                        }
                        else
                        {
                            LA_Heater.BackColor = System.Drawing.Color.Gainsboro;
                        }
                        //                      PLC output Byte 7 (byte_2[])
                        if (byte_2[0])  //  Oxigen Valve - LA_Oxigen
                        {
                            LA_Oxigen.BackColor = System.Drawing.Color.LimeGreen;
                        }
                        else
                        {
                            LA_Oxigen.BackColor = System.Drawing.Color.Gainsboro;
                        }
                        if (byte_2[1])  //  Air Valve - LA_Air
                        {
                            LA_Air.BackColor = System.Drawing.Color.LimeGreen;
                        }
                        else
                        {
                            LA_Air.BackColor = System.Drawing.Color.Gainsboro;
                        }
                        if (byte_2[2])  //  System Buzzer - LA_Buzzer
                        {
                            LA_Buzzer.BackColor = System.Drawing.Color.LimeGreen;
                        }
                        else
                        {
                            LA_Buzzer.BackColor = System.Drawing.Color.Gainsboro;
                        }
                    }
                }
            }
            private void tabPage1_Click(object sender, EventArgs e)
            {
            }
            private void tabPage2_Click(object sender, EventArgs e)
            {
            }
            //Start PLC
            private void B_StartPLC_Click(object sender, EventArgs e)
            {
                int start_res = client.PlcHotStart();
                string error = client.ErrorText(start_res);
                TB_Log.Text += "PLC start result - " + error + Environment.NewLine;
            }//Start PLC
            //Stop PLC
            private void B_StopPLC_Click(object sender, EventArgs e)
            {
                int stop_res = client.PlcStop();
                string error = client.ErrorText(stop_res);
                TB_Log.Text += "PLC stop result - " + error + Environment.NewLine;
            }//Stop PLC
            //test button
            private void test_button_Click(object sender, EventArgs e)
            {
               // byte[] buf_di = new byte[1];
                byte[] buf_db11 = new byte[10];
                //int Read_ResDI = client.EBRead(start, amount, buf);
                //if (Read_ResDI == 0)
                //queries to plc
              //  int Read_ResDI = client.EBRead(1, 1, buf_di);
                int Read_ResDB11 = client.DBRead(11, 32, 10, buf_db11); // (11, 218, 330, buf_db11)
                if (Read_ResDB11 == 0)
                {
                    // pH
                    double scale_pH = 0.000506366;
                    double val_pH = 0;
                    Boolean[] bit_buf_pH = { false, false, false, false, false, false, false, false, false,
false, false, false, false, false, false, false };
                    string bits_pH = "";
                    string bits_pH_n = "";
                    int dec_pH = 0;
                    int dec_raw;
                    TTB_bin.Text = "";
                    TTB_bytes_before.Text = "";
```

```
                    TTB_bytes_before.Text = "Before H " + buf_db11[0].ToString() + " L " + buf_db11[0 +
1].ToString();
                    ToBit16(bit_buf_pH, buf_db11[0], buf_db11[0 + 1], out dec_pH, ref bits_pH);
                //  val_pH = scale_pH * dec_pH;
                    TTB_bin.Text = bits_pH;
                    dec_raw = buf_db11[0] * 256 + buf_db11[0+1];
                    TTB_dec_raw.Text = dec_raw.ToString();
                    TTB_decimal.Text = dec_pH.ToString();
                    TTB_dec_bin.Text = "";
                    To2Bytes(dec_pH, out buf_db11[5], out buf_db11[5 + 1], ref bits_pH_n);
                    TTB_dec_bin.Text = bits_pH_n;
                    TTB_bytes_after.Text = "";
                    TTB_bytes_after.Text = "After H " + buf_db11[5].ToString() + " L " + buf_db11[5 +
1].ToString();
                }
                } //test button
            //                      Control buttons
        //--------                          PUMP
            private void B_Pump1_Start_Click(object sender, EventArgs e)
            {
                Modify_Outputs(2, 0, true, initial, ref Outputs2, ref Outputs7);
            }
            private void B_Pump1_Stop_Click(object sender, EventArgs e)
            {
                Modify_Outputs(2, 0, false, initial, ref Outputs2, ref Outputs7);
            }
            private void B_Pump2_Start_Click(object sender, EventArgs e)
            {
                Modify_Outputs(2, 1, true, initial, ref Outputs2, ref Outputs7);
            }
            private void B_Pump2_Stop_Click(object sender, EventArgs e)
            {
                Modify_Outputs(2, 1, false, initial, ref Outputs2, ref Outputs7);
            }
            private void B_Pump3_Start_Click(object sender, EventArgs e)
            {
                Modify_Outputs(2, 2, true, initial, ref Outputs2, ref Outputs7);
            }
            private void B_Pump3_Stop_Click(object sender, EventArgs e)
            {
                Modify_Outputs(2, 2, false, initial, ref Outputs2, ref Outputs7);
            }
            private void B_Pump4_Start_Click(object sender, EventArgs e)
            {
                Modify_Outputs(2, 3, true, initial, ref Outputs2, ref Outputs7);
            }
            private void B_Pump4_Stop_Click(object sender, EventArgs e)
            {
                Modify_Outputs(2, 3, false, initial, ref Outputs2, ref Outputs7);
            }
        //--------                              Stirrer control
            private void B_Stirrer_Start_Click(object sender, EventArgs e)
            {
                Modify_Outputs(2, 4, true, initial, ref Outputs2, ref Outputs7);
            }
            private void TB_Stirrer_Stop_Click(object sender, EventArgs e)
            {
                Modify_Outputs(2, 4, false, initial, ref Outputs2, ref Outputs7);
            }
            //                            Heating - Cooling
            private void B_CircPump_Star_Click(object sender, EventArgs e)
            {
                Modify_Outputs(2, 5, true, initial, ref Outputs2, ref Outputs7);
            }
            private void B_CircPump_Stop_Click(object sender, EventArgs e)
            {
                Modify_Outputs(2, 5, false, initial, ref Outputs2, ref Outputs7);
            }
            private void B_CoolValve_Open_Click(object sender, EventArgs e)
            {
                Modify_Outputs(2, 6, true, initial, ref Outputs2, ref Outputs7);
            }
            private void B_CoolValve_Close_Click(object sender, EventArgs e)
            {
                Modify_Outputs(2, 6, false, initial, ref Outputs2, ref Outputs7);
            }
            private void B_Heater_On_Click(object sender, EventArgs e)
            {
                Modify_Outputs(2, 7, true, initial, ref Outputs2, ref Outputs7);
            }
            private void B_Heater_Off_Click(object sender, EventArgs e)
            {
                Modify_Outputs(2, 7, false, initial, ref Outputs2, ref Outputs7);
            }
        //--------                              Oxigen and Air Valves
            private void B_Oxigen_Open_Click(object sender, EventArgs e)
            {
                Modify_Outputs(7, 0, true, initial, ref Outputs2, ref Outputs7);
```

```
                }
                private void B_Oxigen_Close_Click(object sender, EventArgs e)
                {
                    Modify_Outputs(7, 0, false, initial, ref Outputs2, ref Outputs7);
                }
                private void B_Air_Open_Click(object sender, EventArgs e)
                {
                    Modify_Outputs(7, 1, true, initial, ref Outputs2, ref Outputs7);
                }
                private void B_Air_Close_Click(object sender, EventArgs e)
                {
                    Modify_Outputs(7, 1, false, initial, ref Outputs2, ref Outputs7);
                }
            //--------                              System
                private void B_Buzzer_On_Click(object sender, EventArgs e)
                {
                    Modify_Outputs(7, 3, true, initial, ref Outputs2, ref Outputs7);
                }
                private void B_Buzzer_Off_Click(object sender, EventArgs e)
                {
                    Modify_Outputs(7, 3, false, initial, ref Outputs2, ref Outputs7);
                }
                private void TB_RPMS_TextChanged(object sender, EventArgs e)
                {
                    int rpms_read = System.Convert.ToInt32(TB_RPMS.Text);
                    double multiplier = 10;
                    double rpms_multiplied = rpms_read * multiplier;
                    int rpms_to_write = System.Convert.ToInt32(rpms_multiplied);
                    byte[] buf = new byte[2];
                    string bits = "";
                    if (rpms_read > 0 && rpms_read <= 1000)   //check limits. Good
                    {
                        TB_RPMS.BackColor = System.Drawing.Color.White;
                        B_Set.Enabled = true;
                        To2Bytes(rpms_to_write, out buf[0], out buf[0 + 1], ref bits);
                        TB_NSP_db.Text = rpms_to_write.ToString() + "    " + buf[0].ToString() + " " + buf[0 +
1].ToString();
                        TB_NSP.Text = bits;
                    }
                    else                            // Bad
                    {
                        TB_RPMS.BackColor =  System.Drawing.Color.Red;
                        B_Set.Enabled = false;
                    }
                }
                private void B_Set_Click(object sender, EventArgs e)
                {
                    int rpms_read = System.Convert.ToInt32(TB_RPMS.Text);
                    double multiplier = 10;
                    double rpms_multiplied = rpms_read * multiplier;
                    int rpms_to_write = System.Convert.ToInt32(rpms_multiplied);
                    byte[] buf = new byte[2];
                    string bits = "";
                    int Write_Res;
                    To2Bytes(rpms_to_write, out buf[0], out buf[0 + 1], ref bits);
                    Write_Res = client.ABWrite(432, 2, buf);
                    if (Write_Res == 0)
                    {
                        TB_Log.Text  +=  "Stirrer  speed  was  set  to  "  +  rpms_read.ToString()  +  "RPM"  +
Environment.NewLine;
                    }
                    else
                    {
                        TB_Log.Text += "Error while setting stirrer speed - " + client.ErrorText(Write_Res);
                    }
                }
            }   //public partial class Form1 : Form
        }
```