TALLINNA TEHNIKAÜLIKOOL
INSENERITEADUSKOND
Elektroenergeetika ja mehhatroonika instituut

AUTONOOMSE ROBOTLAEVA ENERGIATÕHUSUSE UURIMINE JA TÖÖKINDLUSE PARENDAMINE

ENERGY EFFICIENCY RESEARCH AND RELIABILITY IMPROVEMENT OF THE AUTONOMOUS ROBOTIC VESSEL

MAGISTRITÖÖ

Üliõpilane: Alexander Tsupsman

Üliõpilaskood 183134 AAAM

Juhendaja: vanemteadur Indrek Roasto

Tallinn 2020

(Tiitellehe pöördel)

AUTORIDEKLARATSIOON

Olen koostanud lõputöö iseseisvalt.
Lõputöö alusel ei ole varem kutse- või teaduskraadi või inseneridiplomit taotletud.
Kõik töö koostamisel kasutatud teiste autorite tööd, olulised seisukohad, kirjandusallikatest ja mujalt pärinevad andmed on viidatud.

"....." ..... 202…..

Autor: .....
/ allkiri /

Töö vastab bakalaureusetöö/magistritööle esitatud nõuetele

"....." ..... 202…..

Juhendaja: .....
/ allkiri /

Kaitsmisele lubatud

"....."..... 202… .

Kaitsmiskomisjoni esimees .....
/ nimi ja allkiri /

Lihtlitsents lõputöö reprodutseerimiseks ja lõputöö üldsusele kättesaadavaks tegemiseks[1]

Mina Alexander Tsupsman (a*utori nimi*)

1. Annan Tallinna Tehnikaülikoolile tasuta loa (lihtlitsentsi) enda loodud teose

Autonoomse robotlaeva energiatõhususe uurimine ja töökindluse parandamine,

(*lõputöö pealkiri*)

mille juhendaja on Indrek Roasto,

(*juhendaja nimi*)

1.1 reprodutseerimiseks lõputöö säilitamise ja elektroonse avaldamise eesmärgil, sh Tallinna Tehnikaülikooli raamatukogu digikogusse lisamise eesmärgil kuni autoriõiguse kehtivuse tähtaja lõppemiseni;

1.2 üldsusele kättesaadavaks tegemiseks Tallinna Tehnikaülikooli veebikeskkonna kaudu, sealhulgas Tallinna Tehnikaülikooli raamatukogu digikogu kaudu kuni autoriõiguse kehtivuse tähtaja lõppemiseni.

2. Olen teadlik, et käesoleva lihtlitsentsi punktis 1 nimetatud õigused jäävad alles ka autorile.

3. Kinnitan, et lihtlitsentsi andmisega ei rikuta teiste isikute intellektuaalomandi ega isikuandmete kaitse seadusest ning muudest õigusaktidest tulenevaid õigusi.

21.12.2020 (*kuupäev*)

---

[1] *Lihtlitsents ei kehti juurdepääsupiirangu kehtivuse ajal vastavalt üliõpilase taotlusele lõputööle juurdepääsupiirangu kehtestamiseks, mis on allkirjastatud teaduskonna dekaani poolt, välja arvatud ülikooli õigus lõputööd reprodutseerida üksnes säilitamise eesmärgil. Kui lõputöö on loonud kaks või enam isikut oma ühise loomingulise tegevusega ning lõputöö kaas- või ühisautor(id) ei ole andnud lõputööd kaitsvale üliõpilasele kindlaksmääratud tähtajaks nõusolekut lõputöö reprodutseerimiseks ja avalikustamiseks vastavalt lihtlitsentsi punktidele 1.1. ja 1.2, siis lihtlitsents nimetatud tähtaja jooksul ei kehti.*

LÕPUTÖÖ LÜHIKOKKUVÕTE

Autor: Alexander Tsupsman *Lõputöö liik:* Magistritöö

Töö pealkiri: Autonoomse robotlaeva energiatõhususe uurimine ja töökindluse parandamine

Kuupäev: 21.12.2020 61 *lk (lõputöö lehekülgede arv koos lisadega)*

Ülikool: Tallinna Tehnikaülikool

Teaduskond: Inseneriteaduskond

Instituut: Elektroenergeetika ja mehhatroonika instituut

Töö juhendaja(d): vanemteadur Indrek Roasto

Töö konsultant (konsultandid):

Sisu kirjeldus:

Käesoleva lõputöö eesmärk oli arendada diagnoostika ja loogimise tarkvarat mis aitab tagada NYMO robotlaeva töökindluse ja energiatõhususe. Töökäigus oli uuritud missugused autonoomsed süsteemid on olemas maailmas, missugused sarnased lahendused turul ja tööstuses kasutusel olemas. Analüüsiti ka NYMO robotlaeva ehitust ja tema juhtimissüsteemi andmete edastus. Loodi loogimis- ja diagnoostika tarkvarat. Ning tehti katsed reaalses keskkonnas. Püstitatud eesmärk oli täidetud.

Märksõnad: alusfail, juhend, lõpetamine, lõputöö vormistus, lühikokkuvõte, robotlaev, ASV, NYMO, autonoomne, võimsusetarbimine

ABSTRACT

Author: Alexander Tsupsman

Type of the work: Bachelor/Master Thesis

Title: Energy Efficiency Research and Reliability Improvement of the Autonomous Robotic Vessel

Date: 21.12.2021

61 *pages (the number of thesis pages including appendices)*

University: Tallinn University of Technology

School: School of Engineering

Department: Department of Electrical Power Engineering and Mechatronics

Supervisor(s) of the thesis: senior researcher Indrek Roasto

Consultant(s):

Abstract:

The purpose of this thesis was to develop diagnostic and logging software that will help ensure the reliability and energy efficiency of the NYMO robotic vessel. The course examined what kind of autonomous systems exist in the world, what similar solutions exist in the market and in industry. The construction of the NYMO robotic vessel and the data transmission of its control system were also analyzed. Logging and diagnostic software was created. And the experiments were performed in a real environment. The set goal was achieved.

Keywords: template, requirements, graduation, formatting of the thesis, abstract, robotic vessel, ASV, NYMO, autonomous, power consumption

LÕPUTÖÖ ÜLESANNE

Lõputöö teema:	**Autonoomse robotlaeva energiatõhususe uurimine ja töökindluse parandamine**
Lõputöö teema inglise keeles:	**Energy Efficiency Research and Reliability Improvement of the Autonomous Robotic Vessel**
Üliõpilane:	**Alexander Tsupsman 183134 AAAM**
Eriala:	**Energiamuundus- ja juhtimissüsteemid**
Lõputöö liik:	**magistritöö**
Lõputöö juhendaja:	**vanemteadur Indrek Roasto**
Lõputöö kaasjuhendaja: (ettevõte, amet ja kontakt)	
Lõputöö ülesande kehtivusaeg:	**14.12.2020**
Lõputöö esitamise tähtaeg:	**21.12.2020 kell 15:00**

_____	_____	_____
Üliõpilane (allkiri)	Juhendaja (allkiri)	Õppekava juht (allkiri)

_____
Kaasjuhendaja (allkiri)

1. Teema põhjendus

Autonoomse elektrilise robotlaeva kõige olulisemad kriteeriumid on energiatõhusus ja töökindlus. Töövõime ja -aeg on piiratud akus sisalduva energia hulkaga. Mida energiatõhususemalt suudab autonoomne robotlaev töötada seda kauemaks jagub energiat. Tööviljakus sõltub otseselt töökindlusest. Eesmärgiks siin on tagada täpne diagnoostika ja vigada õige-aegne tuvastamine.

2. Töö eesmärk

Töö eesmärgiks on uurida ja võrrelda erinevaid meetodeid autonoomse elektrilise robotlaeva töökindlususe ja energiatõhususe tagamiseks või parendamiseks.

3. Lahendamisele kuuluvate küsimuste loetelu:

Autonoomse elektrilise robotlaeva riist- ja tarkvaraga tutvumine, diagnoostika ja logimise süsteemi loomine, erinevate meetodite võrdlemine ja uurimine autonoomse elektrilise robotlaeva energiatõhusue parendamiseks.

4. Lähteandmed

Autonomse elektrilise robotlaeva elektrilised skeemid, mootori kontrolleri andmeleht, veoajamite tüüp.

5. Uurimismeetodid

Teoreetiline uurimine, algoritmi plokkskeemi koostamine, tarkvara kirjutamine, katseline kontroll ja tulemuste analüüs.

6. Graafiline osa

Autonoomse robotlaeva elektriskeem, tarkvara plokkskeem.

7. Töö struktuur

Titelleht; Sisukord; Lõputöö ülesanne ja sissejuhatus; diagnoostika ja loogimise süsteemi arendamine; Energiatõhususe mõõtmine reaalses katse-keskkonnas; Tulemuste analüüs; Kokkuvõte; Kasutatud kirjandus.

8. Kasutatud kirjanduse allikad

Andmelehed, raamatud, teadusartikleid.

9. Lõputöö konsultandid

vanemteadur Tanel Jalakas

10. Töö etapid ja ajakava

1. Teema püstitamine. (01.06.2020)
2. Autonoomse elektrilise robotlaeva rist- ja tarkvaraga tutvumine. (08.06.2020)
3. Algoritmi plokkskeemi koostamine ja tarkvara kirjutamine. (06.07.2020)
4. Energiatõhususe mõõtmine reaalses katse-keskkonnas. (07.09.2020)

5. Erinevate meetodite võrdlemine ja uurimine autonoomse elektrilise robotlaeva energiatõhusue tõstmiseks ja töökindluse tagamiseks. (02.11.2020)
6. Tehtud töö lõplik vormistamine, köitmine ja hindamisele esitamine. (21.12.2020) juhendajale läbilugemiseks saatmine, paranduste sisseviimine, juhendajale teiseks läbilugemiseks saatmine, töö lõplik versioon valmis.

SISUKORD

EESSÕNA

Käesoleva lõputöö teema anti autorile TalTech-i poolt. Eesmärgiks oli areneda tarkvarat mis aitab tagada robotlaeva energiatõhusust ja töökindlusust. Tarkvara koostamine ja arenemine toimus ülikooli poolt mulle eraldatud labori osas. Lõplik katsetamine toimus järvel. Teema püstitamisega ja algandmetega abistas Indrek Roasto, kes on lõputöö juhendaja ja ülikooli vanem teadur. Tarkvara koostamisega ja katsetamisega aitas Tanel Jalakas, ülikooli vanem teadur. Üldküsimuste osas pöördusin ka oma juhendaja poole.

SISSEJUHATUS

Süsteemid, mis võivad muuta oma käitumist operatsiooni ajal vastuseks ootamatule sündmustele nimetatakse "Autonoomseks". Selliste süsteemide võimekus ja nende rakendusalad viimastel aastatel märkimisväärselt laienenud. Enamasti tänu arengule masinõppe valdkonnas. Maismaal toimuvas tegevuses me õppisimie tundma autonoomsete autode edusamme. Sellega võrreldes autonoomsete laeva arendus kaugele ei jõudnud. Selle lõpuleviimine on hetkel aktuaalne ja nõuab erinevate väljakutsete ülesaamist, sealhulgas:

- autonoomsete meresüsteemide kasutamise hõlbustamine;
- süsteemide ostu- ja tegevuskulude vähendamine;
- autopiloodi GOLREGi reeglite arusaamist õpitamine;
- täpne navigeerimine iga ilmaga.

Isesõitvate laevade arendamisega tegelevad paljude suuremate mereriikide teadlasrühmad ja rida laevandusettevõtteid. Põhjus on lihtne – üha kasvava globaalse kaubanduse suurenevad veomahud ja vajadus kärpida vedude kulusid. Hinnanguliselt kulgeb kaupade tee sihtkohta vähemal või suuremal määral veeteed kasutades ca 80 protsendil juhtudest [1]. Autonoomsete laevade puhul saab tuntavalt vähendada personalikulusid ja ka kulutusi meeskondade väljaõppeks ning turvalisuse tagamiseks. Lisaks vabastaks isesõitvad laevad suure hulga inimesi kohati väga üksluisest, raskest ning ohtlikust tööst, mis liiatigi eeldab nende kauast ning sagedast eemalviibimist nii peredest kui igapäevastest elumugavustest. Ehkki Eestis tegeldakse edukalt nii autonoomsete pakirobotite kui isesõitvate autode arendamisega [2] - [3], ei ole meil varem isejuhtivat veesõidukit arendatud.

Paljud riigid intensiivselt isesõitvate aluste arendamisega. Norra on näiteks kavandanud, et robotlaevad, mis on ülimadalate heitkogustega, võiks välja tulla juba tuleval aastal [4]. Taani plaanib rakendada autonoomseid aluseid rohketel praamivedudel. Esialgu kavandatakse mehitatud autonoomseid laevu, kus meeskond saaks vajadusel juhtimise üle võtta. Ühtlasi täidaksid nad ka teenindavaid kohustusi. Madalmaades plaanitakse kasutada selliseid aluseid arvukatel kanalitel ja jõgedel. Soomes, Rootsis ja Norras on eraldatud spetsiaalsed merealad isesõitvate laevade katsetamiseks. Testitakse navigeerimist, automaatsildumist, laadimist ning lossimist [5] - [7].

Üks silmatorkavamaid näiteid on robotlaev SEA-KIT X, mis on kujutatud Joonis 0.1. SEA-KIT X on väga konfigureeritav ja loodud paljude mereoperatsioonide toetamiseks. Autonoomne robotlaev, mille pikkus on 12 m, võimaldab mitut missiooni, näiteks süvavee batümeetriat ja miinitõrjet, ilma inimressursse kahjustamata ja oluliselt väiksemate kuludega [8].

Joonis 0.1 SEA-KIT X robotlaev

NYMO on üks sellist laevast, mis arendab teadlaste töögrupp TalTechi elektroenergeetika ja mehhatroonika instituudist koostöös ettevõtetega Hyrles OÜ ja MEC Insenerilahendused OÜ (Joonis 0.2).

NYMO on multifunktsionaalne autonoomne robotlaev, mis suudab ilma välise sekkumiseta läbida GPS-punktidena etteantud teekonna ja sooritada selle jooksul eri ülesandeid, näiteks pakkide kohaletoimetamiseks väiksemate saarte vahel. Robotlaev võiks kaardistada ka merepõhja ja teha päästetöid nii järvel kui ka merel. Seda saab kasutada luuretegevuseks, droonide maandumisplatvormina ja reostuse detektorina. Ta on teadlik oma ümbrusest, tunneb ära merel olevad takistused ja oskab neist mööda sõita. Robotlaeva pikkus on 2,5 m ja laius on 1,1 m. Katamaraani eelis on väga hea stabiilsus, mis on nõnda väikese laeva puhul oluline, sest stabiilne alus ei lähe tormiga ümber ja hoiab hästi kurssi. Nymo liigub edasi puhtalt elektri jõul. Energiaallikaks on kaks liitiumakut.

Joonis 0.2 NYMO robotlaev

Septembris 2019 toimus esimene proovisõit Keri saarele, et kontrollida kas isesõitmise kontseptsioon ja juhtimissüsteemid ning jõusüsteem töötavad ka avamere nõudlikes tingimustes. Leppneeme sadamast Kerile (üks ots ligi 20 kilomeetrit, kokku 40 kilomeetrit) sõitis alus kiirusega neli sõlme, tagasi tuli kolmesõlmese kiirusega [9] - [11].

Teatavasti on akude energiamahtuvus suhteliselt piiratud, mistõttu on ka projekti üks eesmärk võimalikult suure energiatõhususe saavutamine. Selleks on mitmeid võimalusi, nt võimalikult kõrge akupinge valimine, energiamuundusastmete minimeerimine, optimaalse sõidukiiruse valimine, tuule ja lainetusega arvestavad liikumisalgoritmid jne. Selle saavutamiseks on vaja täpselt mõõta laeva energiakulu, milleks omakorda on vaja paindliku diagnoostika ja logimissüsteemi.

Diagnostika on selline valdkond, kus tehnoloogia sillutab teed murranguliste uute robotlaevade või teise autonoomsete sõidukite hoolduskontseptsioonide suunas, sealhulgas tehisintellekti ja süvaõppivate närvivõrkude kasutamine arenenud prognostiliste süsteemide väljatöötamiseks. Sõidukilt kogutud andmete tohutu kasv ja võime seda töödelda pakub hoolduse osas mitmesuguseid eeliseid [12].

Andmete pidev jälgimine reaalajas traadita võrgu kaudu avab laiema valiku võimalusi. Vead ja probleemid sai tuvastada reaalajas ning potentsiaalsed vead välja tuua enne, kui need põhjustavad suuremaid probleeme või kahjustusi. Operaatorile võiks probleemide eest teavitada ja suunata lähimasse teeninduskeskusesse, teabe ja diagnostika võiks aga eelnevalt saata kohalikku teeninduskeskusse, et nad oleksid remondiks valmis. Üks populaarsemaid lahendusi on digitaalne kaksik.

Digitaalse kaksiku kontseptsioon, mida mõnikord nimetatakse seadme varjuks, on olnud juba mõnda aega, kuid tehisintellekti ja asjade interneti tulekuga on kontseptsioon hakanud üha enam silma paistma.

Digitaalne kaksik on virtuaalne mudel, mis simuleerib füüsilist protsessi, teenust või süsteemi. Luues reaalmaailma vara digitaalse manifestatsiooni, saavad disainerid ja insenerid hõlpsamini tõhususe testimist ja probleemide tuvastamist, ilma et oleks vaja iga iteratsiooni füüsiliselt prototüüpida [13].

Digitaalne kaksik kasutab andureid, et jäädvustada füüsilise objekti hetkeseis, sealhulgas selle asukoht, tööseisund, kompositsioon ja palju muud. Seejärel edastatakse töötlemata andmed peaaegu reaalajas selle virtuaalsele vastaspoolele, võimaldades diagnostikat ja prognostilisi teste. Andmete parsimine, töötlemine ja modelleerimine võib toimuda lokaalses või pilvevõrgus, mis võimaldab testida osi ja üksusi, millele on füüsiliselt raske ligi pääseda.

Robotlaeva NYMO projekti käigul on ka plaan teha isejuhtivast veesõidukist digitaalne kaksik [14]. See aitab merel toimuvaid sündmusi teatud piires ette ennustada. Digitaalne kaksik teeb kindlaks, kas laeva akudes olevast energiast piisab, et jõuda marsruudi lõppunkti või mitte. Või kummalt poolt on tuule suunda ja laine kõrgust arvestades energiaefektiivsem mööda sõita. Samuti aitab kaksik merekaarti arvesse võttes marsruuti valida. Lisaks saab digitaalse kaksiku abil tulevikus laeva mehhaanikat, materjalikulu ja energiatarvet optimeerida ning rakendada erinevaid masinõppe algoritme. Sammuti on digitaalse kaksiku loomiseks vaja esmalt korraliku diagnostika ja logimisesüsteemi, mis ongi antud magistritöö üheks peamiseks eesmärgiks.

Tulenevalt eesmärgist töö on jaotatud kolmeks peatükiks.

Töö esimeses peatükis autor käsitleb robotlaeva rist- ja tarkvarat, kuna sisuliste probleemide lahendamiseks tuleb esmalt tutvuda robotlaeva struktuuriga ja toimimispõhimõtega.

Teises peatükis autor kirjeldab diagnoostika ja logimise süsteemi loomist. Selline süsteem kujutab endast tarkvarat, mis võimaldab reaalajas lugeda robotlaeva mootorite parameetrid ja salvestada neid serverisse.

Viimases peatükis autor analüüsib tehtud tööd ja võrreldab saadud tulemused.

1. ROBOTLAEVA EHITUS

Selles peatükis tuleb jutt NYMO tehnilistest ja kvaliteedinäitajatest.

1.1 Mehaaniline disain

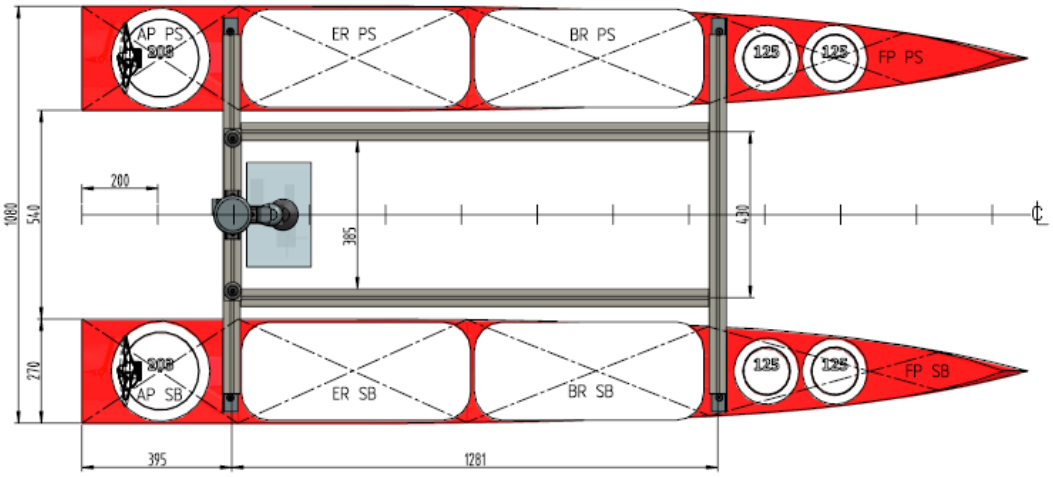
ASV (Autonomous surface vessel) on kahe 2,5 meetri pikkuse punaseks värvitud klaasplastist kerega katamaraan, millel laiust 1,1 meetrit, nagu näidatud Joonis 1.1. Niisugused mõõdud on valitud, et veerandtonnise veeväljasurvega laeva oleks mugav järelhaagisega vedada. Tekiehituseks on profiilalumiiniumist raam, millele on püstitatud antenne ja navigatsiooniseadmeid kandev lüheldane mast. Katamaraani kuju valiti tänu järgmistele eelistele:

- peaks olema võimalikult rohkem ruumi. Kandevõime võiks olla paigutatud ka kere vahele;
- suurem stabiilsus väiksema nihke korral. Selleks, et lainete mõju vähendada, stabiilsus on eriti oluline väikeste ASV jaoks;
- väga hea manööverdusvõime. Tänu kahele mootorile suudab katamaraan ümber oma telje pöörata.

Tabel 1.1 on peamised mõõtmed ja elektrilised parameetrid näidatud. Robotlaeva kasulik kandevõime on 100 kg.

Tabel 1.1 ASV tehnilised parameetrid

Nimetus	**Suurus**	**Ühik**
Pikkus	2560	mm
Laius	1100	mm
Sügavus	180...270	mm
Kandevõime	100	kg
Maksimaalne kiirus	6	kn
Takistus @6 kn	0,48	kN
Võimsus @6 kn	1,6	kW
Elektrilise ajami pinge	48	V
Juhtimissüsteemi pinge	12	V
Tõukejõu kiirus	1700	rpm
Võllide arv	2	



Joonis 1.1 ASV mõõtmeid [15]

1.2 Elektroonika

ASV koosneb sõltumatult juhitavast energiast allikatest, koormatest ja hoidlatest. Peamine eesmärk on säilitada pardal olev reguleeritud pinge ja energiatõhus töö.

Joonis 1. on näidatud ASV üldine elektriline struktuur. ASV võiks hõlmata mitu energiaallikat ja hoidlat nt. tuul, päike, vesinik või patareid. Umbes 90% energiast kulutab tõukejõusüsteem. Ülejäänud tarbib juhtimissüsteem. Elektrivarustussüsteem on jagatud kaheks pingegrupiks (Joonis 1.2): jõuahelad 48 VDC, juhtimissüsteem 12 VDC.

Mida suurem on toitepinge, seda väiksemad on kaod juhtmetes. Teiselt poolt maksavad madalpingeseadmed (12 V või 24 V) vähem ja pakutakse palju laiemat valikut. Seega pingetasemete valik on kompromiss efektiivsuse, paindlikkuse ja hinna vahel. ASV-d juhivad kaks 1500 W BLDC mootorid, mis töötavad 48 V pingega. Kõrgem pinge tähendab vähem kadusid juhtmetes. Seega on soovitatav varustada suure võimsusega koormusi suurema pingega, mille tulemuseks on väiksem koormusvool ja kaod. Sellepärast oli valitud 48 V mootorite ja akude jaoks.

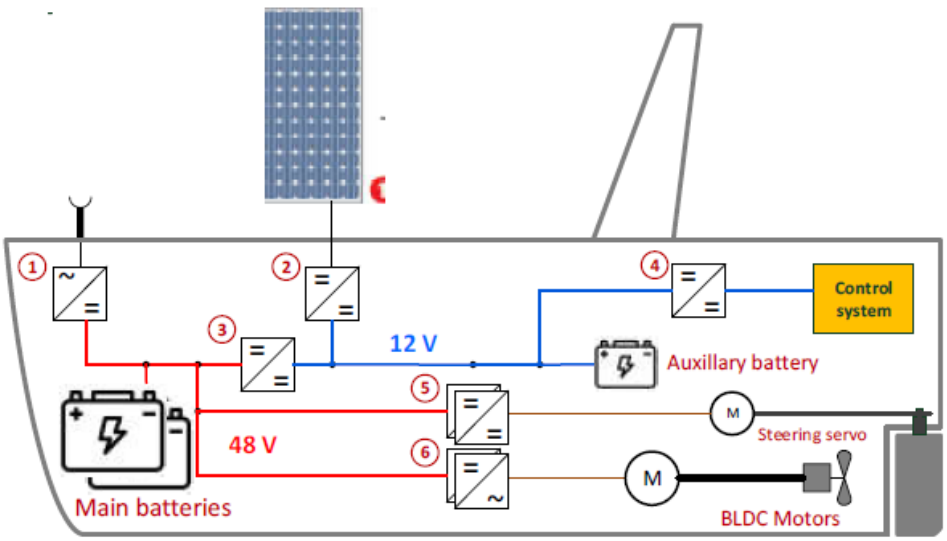
Juhtimisseadme ja abiseadme energiatarve on umbes vähem kui 10%. Pealegi juhtimissüsteem nõuab palju madalamaid pingeid. Pingetasemed on siin 3,3 V ja 5 V. Seega kavandati 12 V töötav juhtimissüsteem, millest saab madalamaid pingeid hõlpsasti teisendada.

Kõik koormused ja energiallikad ühendavad elektrilised muundurid, nagu näidatud Joonis 1.2. Neid saab jagada mitmeks gruppiks:

1. dokkimismuundur;
2. laadimiskontroller;
3. abimuundur erinevate alalisvoolusiinide vahel;
4. juhtimissüsteemi abimuundur;
5. servo kontrollerid;
6. tõukejõu kontroller.

Dokkimismuundur 1 on võimsuselektrooniline muundur mis ühendab ASV utiliidivõrguga. See muundur töötab ainult siis, kui ASV dokkib. Tema peamine ülesanne on laadida akusid. Laadimiskontroller 2 on valikuline dc-dc muundur, mis võimaldab akude laadimist päikeseenergia abil. Nad võivad töötada kogu aeg, kui päike või tuul on saadaval. Abimuundur 3 on dc-dc muundur, mis ühendab 48 V 12 V-alalisvoolusiiniga. Seega ilma tuule ja päikeseta võtab juhtimissüsteem toite põhiakudest. Teine oluline omadus on see, et abimuundur 3 saab vajadusel välja lülitada 48 V alalisvooluühendust. Vigade eraldamine vahemikus 48 V kuni 12 V alalisvoolusiinides suurendab ASV töökindlust. Juhtimissüsteem töötab ka tõsise rikke korral 48 V alalisvoolusiinil.

Abimuundur 4 on dc-dc muundur, mis tekitab juhtimissüsteemi jaoks stabiliseeritud 3,3 V ja 5 V. Servokontrollerid 5 tarnivad ASV rooliservosid. Tõukejõu kontroller tarnib peamasin ja juhib propellerite kiirust.



Joonis 1.2 ASV elektriline skeem [15]

Roboteq BLDC motor controller HBL2360A

Roboteq HBL2360A on suure võimsusega kahekanaliline kontroller hallisensoriga varustatud harjadeta alalisvoolumootorite jaoks. Ta juhib kaks 1,5 kW võimsusega ja 48 V tööpingega püsimagnetitega harjavaba kolmefaasilist asünkroonmootorit ning saadab info nende kohta Nano Jetsonile [16].

Joonis 1.7 Roboteq BLDC motor controller HBL2360A

Roboteq mootori kontroller jälgib mootorite peamisi parameetreid, nagu mootorite pinge, vool, kiirus, temperatuur jne. Peaarvutiga on võimalik kõik need andmed mootori kontrollerist vastu võtta, töödelda ja muuta reaalajas. Seda saab kasutada ka oma diagnoostika ja loogimise süsteemi loomiseks.

1.3 Toitesüsteem

ASV kujundati täiselektrisõidukina, seega kõik süsteemid töötavad elektriga. Jõuseadmete energiaallikaks on kaks mahukad LiFePo akud kogumahtuvusega 80 Ah ja pingega 48 V ning on näidatud Joonis 1.3. Liitiumraudfosfaadi aku pakub suurt keemilist stabiilsust, suurt tsüklite arvu ja kõrget energiatihedust.

Joonis 1.3 NYMO akud koos kontrolleritega

Lähtuvalt vajadustest oli valitud järgmine aku konfiguratsioon (Tabel 1.2):

Tabel 1.2 Aku konfiguratsioon

	Suurus
Pinge	48 V
Aku mahtuvus	40 Ah
Aku tüüp	LiFePo
Akude arv	2
Kogu mahtuvus	3840 Wh
Kaal	46 kg
Kruiisiaeg @ 6kn	2,4 h
Aku hind	1800 €
Energia hind	0,94 €/Wh
Kogu hind	3600 €

1.4 Modullaarne juhtimissüsteem

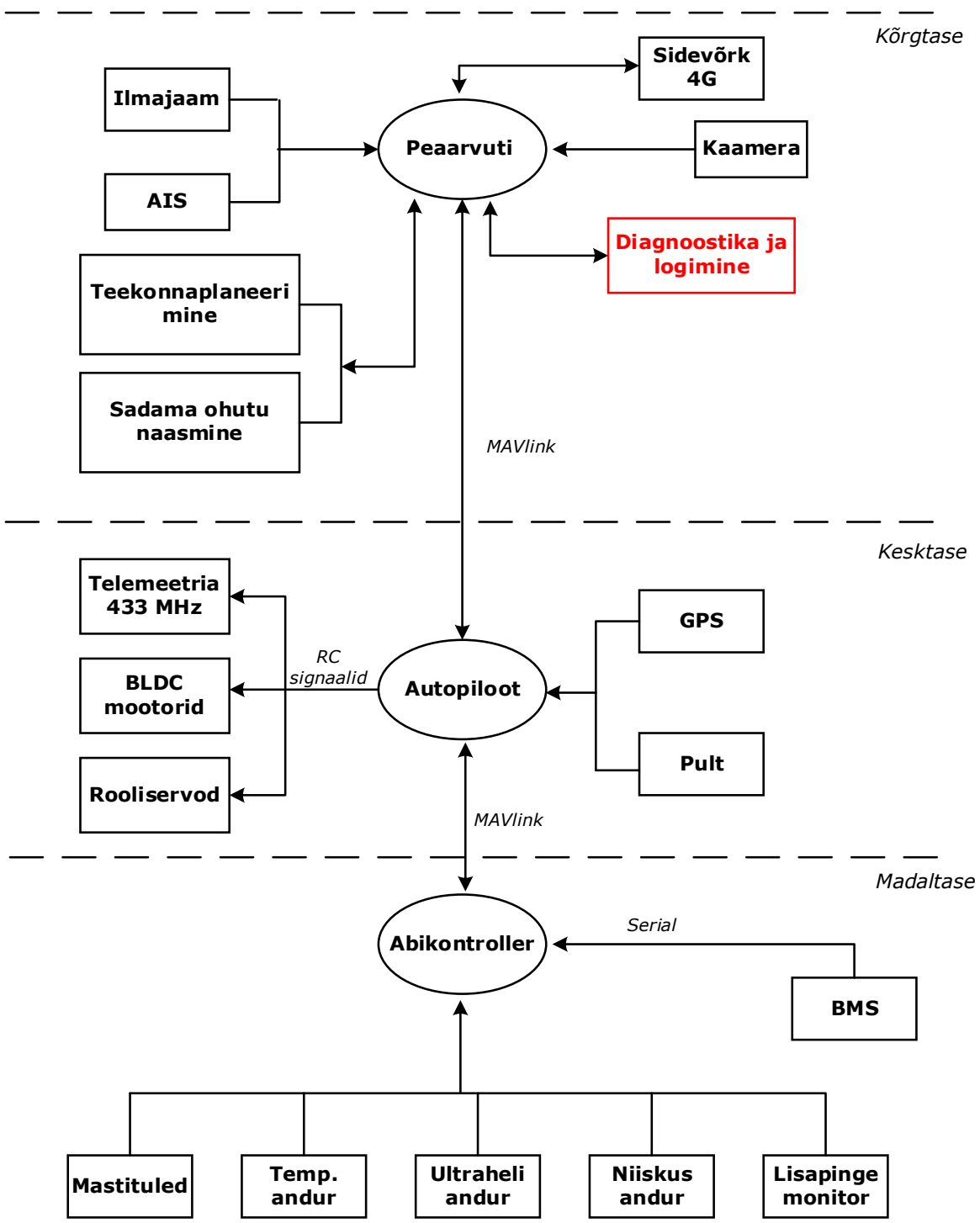
NYMO juhtimissüsteem on ehitatud veekindlasse plastikust kohvrisse. See on kogu isesõitva laeva arendamise projekti kõige olulisem osa – autonoomne juhtimissüsteem koos navigatsiooniseadmetega. Süsteem peab juhtima laeva sihtkohta ja samas vältima teel olevaid takistusi, nii liikuvaid kui ka seisvaid.

ASV juhtimissüsteem on sõltumatu moodul, mida saab hõlpsasti eemaldada ja paigutada teistele ASV-le. Seda varustab 12 V aku, mis on põhivarustussüsteemist

sõltumatu. Juhtimissüsteem asub veekindlas kohvris, mis on ühendatud veekindlate pistikutega, nagu on näidatud Joonis 1.4.

Joonis 1.4 ASV juhtimissüsteem

Juhtimissüsteem koosneb kolmest tasemest: madal, keskmine ja kõrge tase juhtimine (Joonis 1.5). Igal tasemel on oma kontroller. Madaltaseme ülesande eest vastutab abikontroller, autopiloot tegeleb keskmisetaseme juhtimisülesannetega ja peaarvuti hoolitseb kõrge taseme juhtimise eest. Kontrollülesanded rühmitatakse tasemetele olulisuse järgi. Madala taseme ülesanded ei mõjuta ASV üldist jõudlust, samal ajal kõrgetasemelise juhtimise vead võivad olla väga kriitilised. Juhtimistasandid määratlevad ka süsteemi hierarhia. Peaarvuti on põhikontroller ja temal on kõrgeim prioriteet. Autopiloot järgib juhiseid põhikontrollerist jne.



Joonis 1.5 Juhtimissüsteemi struktuuri plokkskeem

1.4.1 Madaltaseme juhtimisülesanded

Need on enamasti mitte-kriitilised ülesanded, mis ei mõjuta navigeerimist või ASV liikumist (Joonis 1.5). Abikontroller juhib mastitulesid, neid tuleks sisse lülitada uduse ajal ja öösel. See toimib ka andurina, st kogub temperatuuri- ja niiskusandurite väärtused, korrigeerib neid ja saadab autopiloodile. Samuti on ultraheli andurid ka ühendatud abikontrolleriga. ASV on varustatud viie veekindla lühikese kaugusega (<8 m) ultraheli anduriga, mida kasutatakse sadamas navigeerimiseks. Abikontroller

edastab juhtimissüsteemi aku ja laeva põhiaku laadimisolekut. Juhtimiskohvris asub varuaku, mille pinget mõõdetakse läbi pingejaguri. Teavet põhiakude kohta saab läbi Victron BMS-i (Battery management system). BMS mõõdab põhiakude pinget ja voolu süsteemi kogu voolu ning edastab selle edasi autopiloodile ja põhikontrollerile. Selle info põhjal töötabki antud magistritöö raames välja töötatud diagnostika ja logimissüsteem.

1.4.2 Keskmistaseme juhtimisülesanded

Keskmise taseme juhtimisülesannetega tegeleb autopiloodi kontroller (Joonis 1.5). Autopiloodil on olemas IMU (Inertial measurement unit) andurid ja baromeeter. Nad on ühendatud välise GPS-vastuvõtjaga, mida kasutatakse positsioneerimiseks ja navigeerimiseks. Autopiloot võtab vastu missioni teekonnapunkte ja käske abikontrollerilt. Kuid ta võib saada ka juhtsignaale kaugjuhtimispuldilt, millel on alati kõrgem prioriteet. See võimaldab operaatorile vajadusel kontrolli üle võtta. ASV liikumist juhivad kaks 800 W BLDC mootorid ja kaks rooliservot. Autopiloot saadab impulss-signaal otse mootoritele. Missiooni saab jälgida reaalajas telemeetria kaudu. Telemeetria diapasoon on umbes 1 km.

1.4.3 Kõrgetaseme juhtimisülesanded

Peaarvuti jälgib kõiki ASV protsessi toimuvust ja teeb autonoomseid otsuseid (Joonis 1.5). See hõlmab ASV kõige olulisemad funktsioonid nt. kokkupõrke vältimine, sadama ohutu naasmis funktsioon, videotöötlus. Põhikontroller jälgib põhiakude laadimisolekut, kogub ilma andmed ja hindab vajalike energiakoguse missiooni edukaks kandmiseks. Kui ta näeb, et missiooni ei ole võimalik lõpetada, siis automaatselt katkestab seda ja alustab uut sadama naasmis missiooni.

ASV on programmeeritud tegutsema vastavalt rahvusvahelisele eeskirjale kokkupõrgete vältimiseks merel (GOLREG reeglid). Robotlaev on varustatud automaatse identifitseerimissüsteemi (AIS) transiiveriga. See võimaldab AISiga varustatud laevadele automaatselt ja dünaamiliselt jagada oma positsiooni, kiirust, kurssi, laeva identiteeti ja muud teavet sarnaselt varustatud laevadele. AIS-transiiver tagab ASV-le täieliku nähtavust. AIS-i info põhjal suudab Nymo vältida kokkupõrkamist teiste laevadega nii udus, vihmas kui ka öösel.

Lisaks AIS-ile robotlaeval on olemas väike IP-kaamera lähiümbruse jälgimiseks. Teiseks sellist kaamerat kasutatakse objektide tuvastamiseks ja automaatseks navigeerimiseks. Kõiki protsesse saab jalgida ja kontrollida robotlaeva operaator. ASV-l on olemas 4G sideühendus mida saab kasutada kaamerapiltide, veakoodide, missiooni staatuse saatmiseks juhtimiskeskusele.

NVIDIA Jatson Nano

Jetson Nano on NVIDIA poolt välja töötatud väike, kuid väga võimas miniarvuti. See võimaldab jooksutada paralleelselt mitmeid erinevaid programme ja tegevusi. Jetson Nano on mõõtmetelt väike, nimelt 100 mm x 80 mm x 28 mm, ning töötab ainult 5 vatti pealt [17].

Joonis 1.6 NVIDIA Jetson Nano

See on juhtimissüsteemi kõige olulisem osa. Siin jooksevad kõik programmid, toimuvad erinevaid arvutused ja salvestakse anduritelt kogu tulev info.

2. DIAGNOOSTIKA JA LOOGIMISE SÜSTEEMI ARENDAMINE ENERGITARBIMISE MÕÕTMISEKS

2.1 Info edastamise kanalid

NYMO on täiselektreline robotlaev, mis sõltub täielikult akudest. Seega on väga tähtis tema energiakulu mõõta.

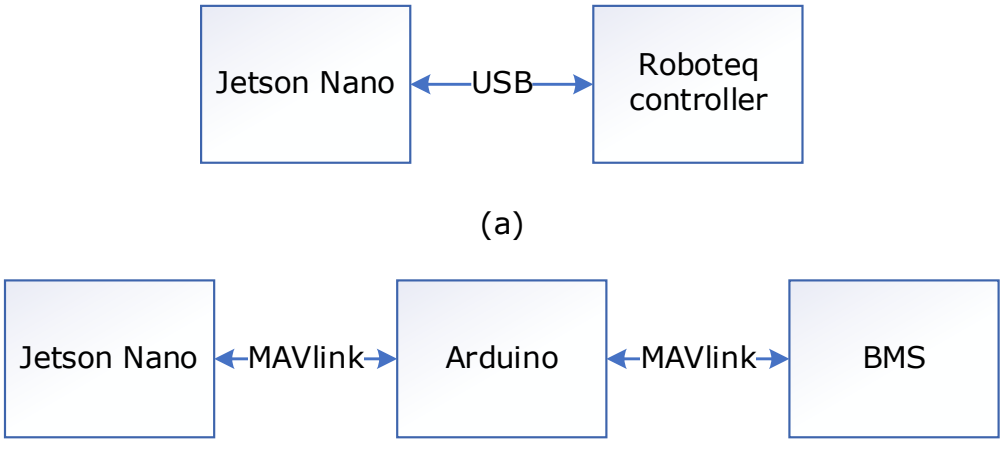
Robotlaeva terve süsteemi saab jagada kaheks osaks. Esimene on juhtimissüsteem ja teine on tõukejõusüsteem.

Juhtimissüsteem hõlmab põhimõtteliselt kõiki juhtimiskohvris olevaid seadmeid. Sealhulgas on miniarvuti Nvidia Jetson Nano, ruuter, Arduino mikrokontroller ja erinavad sensorid.

Tõukejõusüsteem omakorda koosneb tüürservodest ja mootoritest.

Juhtimissüsteemi energiatarbimise mõõtmiseks oli paigaldatud voolusensor ACS712 20A [18]. Aga testimise käigus selgus, et voolu mõõtmisek mA vahemikus ta on üsna kasutu. Sellepärast tuli voolu mõõta välise ampermeetriga vt. peatük 3.1.

Tõukejõusüsteemi energiatarbimist on võimalik automatiseerida. Selleks on olemas kaks võimalust. Kas lugeda mootorite andmed Roboteq kontrolleri abil ja saata neid Jetson Nano-le edasiseks analüüsiks (Joonis 2.1a) või võtta neid BMS-ist (Joonis 2.1b).



Joonis 2.1 NYMO andmevahetuskanalid. Andmeside juhtarvuti ja mootorikontrolleri vahel(a), andmeside BMS ja juhtarvuti vahel (b)

On väga oluline kasutada vähemalt kahte andmeedastusmeetodit, sest andmeteedastus USB kaudu pole piisavalt töökindel. Edastuse protsessis võib informatsioon kaduma minna väliste tegurite mõjul, näiteks elektromagnetilise ühilduvuse mõjul.

2.2. Algoritmi koostamine

Roboteq mootorikontroller jälgib ja mõõdab enamiku oma siseparameetreid.Selle hulgas on:

- Mootorite pinge
- Mootorite RMS vool
- Mootorite kiirus
- Mootoritele rakendunud võimsus (Duty cycle)
- Mootorite temperatuur
- Mootorite ja kontrolleri veakood. Sealhulgas on ühenduse kontroll ja lühise kontroll
- Peaaku laadimisolek
- Peaaku pinge

Kontrolleri sisendvoolu ja pinget ei saa mõõta. Kõige tähtsamad parameetrid, mida ma pean mõõta on mootorite vool ja pinge. Sest neid teades on võimalik jälgida, kui palju energiat robotlaev sõidu ajal tarbib. Samuti on oluline veakoodi jälgida, kuna seal sisaldatakse teavet ühenduse stabiilsuse kohta, mootorite ja kontrolleri oleku kohta.

2.2.1 Programmeerimiskeele valik

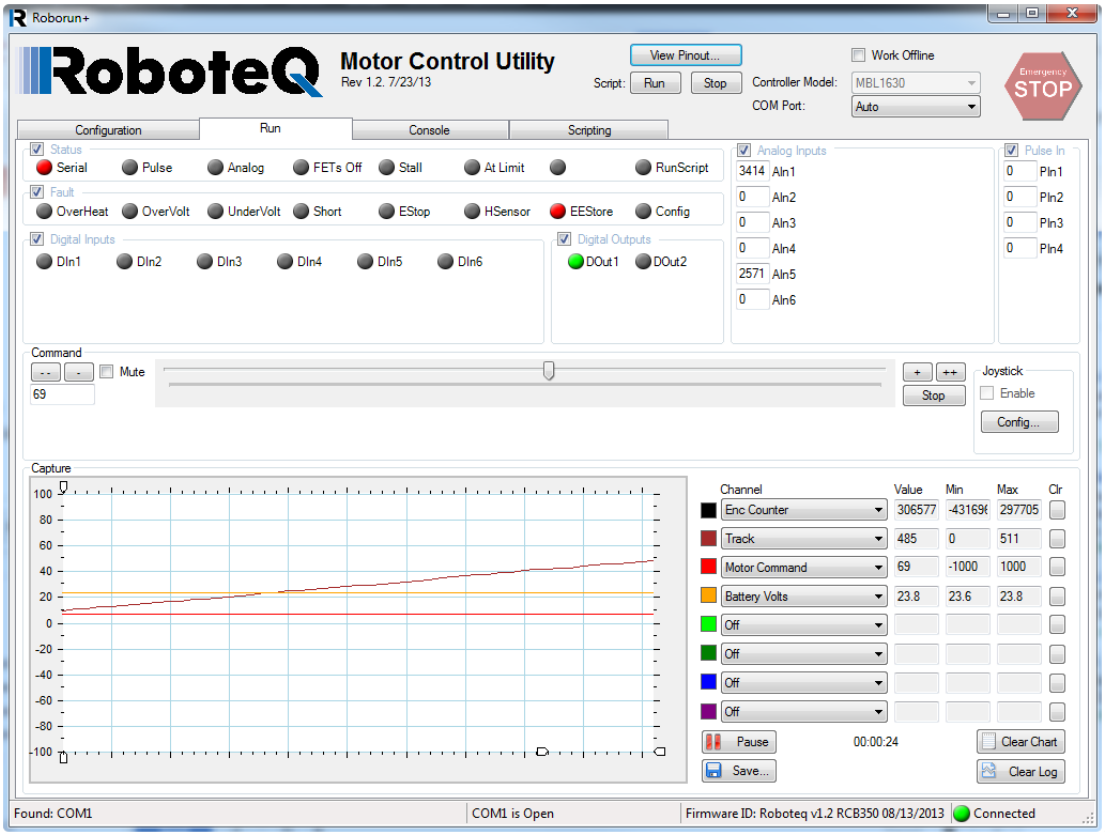
Nano Jetsonil on installitud Linuxi operatsioonisüsteem. Seega kõige ilmsem valik on Python. Ta on juba pikka aega üks populaarsemaid ja lihtsamaid objektorienteeritud programmeerimiskeeli kus on olemas palju erinevaid raamatukogusid.

2.2.2 Algoritmi koostamine

Enne programmi kirjutamise asumist, oli vaja aru saada, kuidas käib Roboteqi kontrolleriga suhtlus, mis andmeside ja mis käskude abil.

Jadaandmeside (RS232) võimaldab kontrollerit ühendada arvutiga, PLC-ga, mikrokontrolleriga või traadita modemiga. Seda ühendust saab kasutada nii käskude saatmiseks kui ka kontrollerilt reaalajas erinevate olekuteavete vastuvõtmiseks [20].

Roboteq kontrolleri ühendamisel arvutiga teeb jadaühendus mootorite diagnostika toimimist lihtsaks. Selleks aitab utiliit Roborun (Joonis 2.2).



Joonis 2.2 Roborun tarkvara

Roborun on arvuti programm, mis võimaldab kasutajatel konfigureerida oma Roboteqi mootorikontrollereid. Utiliidi abil on võimalik lisaks kontrolleri tööoleku monitoorile muuta ka kontrolleri konfiguratsiooniparameetreid. Kasutaja saab ise parameetreid muuta MicroBasic programmeerimiskeeles kirjutatud skriptide abil [21].

Roboteq MicroBasic on kõrgetasemeline programmeerimiskeel, mida kasutatakse programmide kirjutamiseks Roboteqi mootorikontrollerite jaoks. See kasutab süntaksit peaaegu nagu sama Basic süntaks koos mõningate muudatustega, et kiirendada kontrolleri programmi täitmist ja hõlbustada selle kasutamist.

Roboteq kontroller aktsepteerib ja tunnustab ainult nelja tüüpi käske [22]:

- Käivitusaja käsud. Nad algavad tähisega "!" kui kutsutakse serial-pordiga (RS232, RS485 või USB) või kasutades MicroBasic setcommand () funktsiooni. Tavaliselt need on mootori töö käsud, millel on kohene mõju (nt. mootori sisselülitamine, kiiruse seadmine või digitaalväljundi aktiveerimine).
- Käivitusaja päringud. Nad algavad tähisega "?" kui kutsutakse serial-pordiga (RS232, RS485 või USB) või kasutades MicroBasic getvalue () funktsiooni. Neid

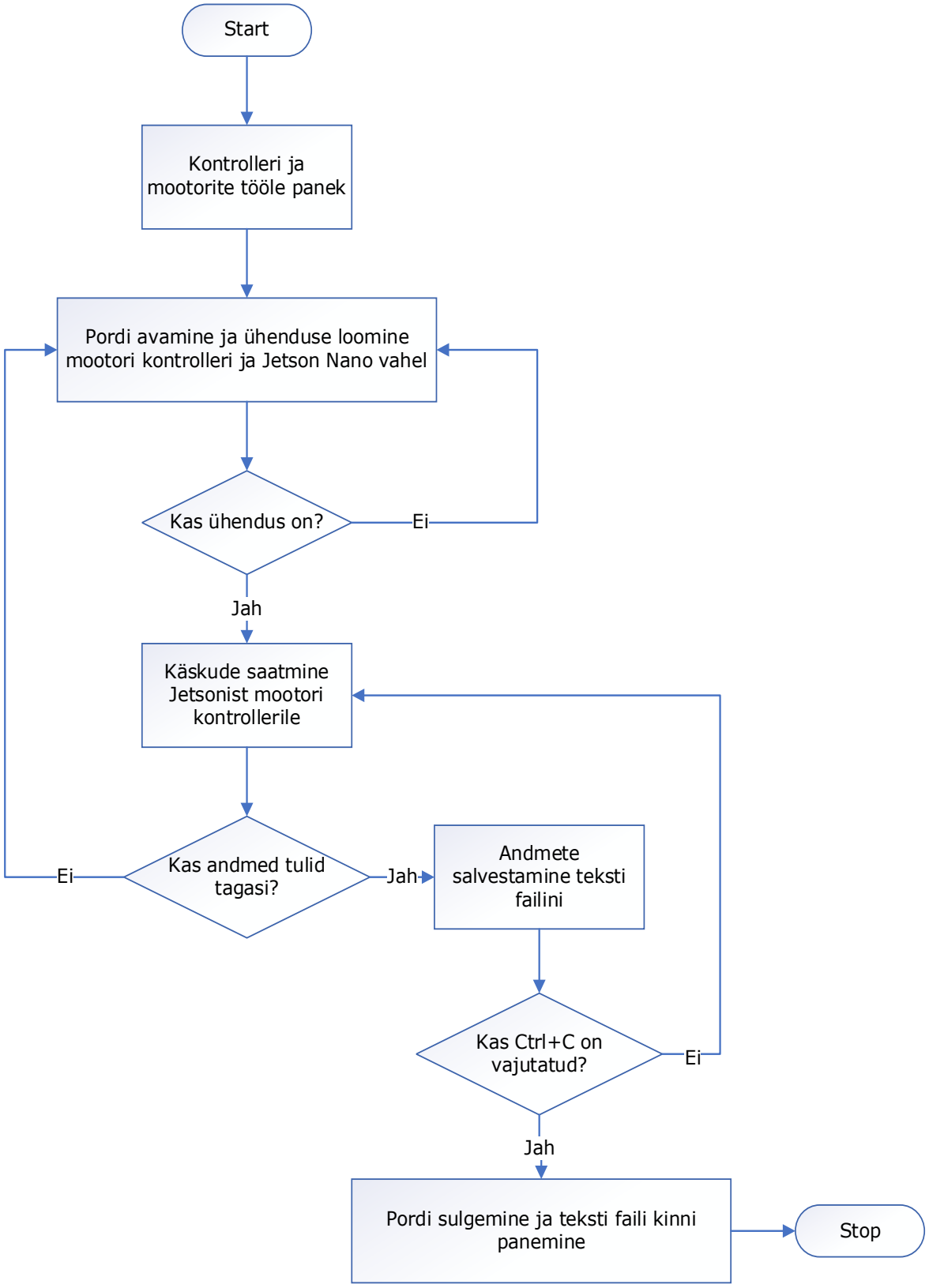
kasutatakse tööväärtuste lugemiseks reaalajas (nt. mootorite vool, pinge, võimsustaset, loenduri väärtused).

- Hoolduskäsud. Nad algavad tähisega "%" ja on saadaval ainult kui kutsutakse serial-pordiga (RS232, RS485 või USB). Neid kasutatakse kõigi hoolduskäskude jaoks (nt. kellaaja määramine, mootorite konfiguratsiooni salvestamine, taaskäivitamine jne).
- Konfigureerimiskäsud. Nad algavad tähisega "~" lugemiseks ja "^" kirjutamiseks kui kutsutakse serial-pordiga (RS232, RS485 või USB) või kasutades MicroBasic getconfig() ja setconfig() funktsioonide. Neid kasutatakse kontrolleri kõigi tööparameetrite lugemiseks või konfigureerimiseks (nt. seada või loe amprite piiri).

Kuna me ei saa kogu vajalikku infot Roboteqist, siis otsustati loobuda Roborun tarkvarast ja kasutada programmeerimisest tavalist Pythoni Integreeritud programmeerimiskeskkonda.

Pärast programmeerimiskeskkona valimist, koostasin algoritmi, mis tagab programmi tõrgete vaba töö (Joonis 2.3). Juhtimiskood on toodud Lisa 1 Juhtimiskood.

Tööpõhimõte on järgmine. Esiteks lülitame Roboteq kontroller sisse. Pärast paneme koodi käima. Kohe tuleb kontroll, kas ühendus mootorikontrolleri ja Jetson Nano vahel on või ei ole. Kui pordi avamine ja ühenduse loomine õnnestus, siis Jetson Nano saadab käsud otse mootori kontrollerile. Kui käsud jõuavad kontrollerini, siis saadab ta mootori andmed Jatson Nano-le tagasi. Kui andmed tulid edukalt tagasi siis salvestatakse neid tekstifailina. Mul juhul kontrollitakse kas port on avatud või kinni. Juhul kui on kinni, toimub restart ja pordi uuest avatakse. Ja seda kõike korratakse hetkeni, kui operaator ise vajutab Ctrl+C. Sellega pannakse tekstifaili ja pordi kinni. Sellega lõpeb energiatarbimise mõõtmine.



Joonis 2.3 Energiatarbimise mõõtmise algoritm

Vaatame koodi üksikasjalikumalt. Esiteks on importeeritud vajalikud moodulid – time, serial ja datetime. Moodul time pakub mitmeid kasulikke funktsioone ajaga seotud ülesannete käsitsemiseks. Moodul serial on vajalik pearvuti ja mootorikontrolleri vahel ühenduse loomiseks. Ja moodul datetime annab võimalust manipuleerida kuupäevadega ja kellaaegadega.

Juhtimiskoodi alguses on estitatud klass Log(). Tänu funktsioonidele open(), write() ja close(), luuakse tekstidokument, kuhu kirjutatakse ja salvestatakse kogu mootorikontrolleri ja mootorite andmed (Joonis 2.4).

```
#Create log file
class Log():

    def __init__(self):
        self.dateTimeObj_1 = datetime.now()
        self.timestampStr_1 = self.dateTimeObj_1.strftime("%d-%b-%H:%M")
        path = "/home/nymo/Nymo python scripts/Alx_Diagn/Logs/measurements.txt"
        new_path = '%s_%s' % (path, self.timestampStr_1)
        self.log = open(new_path, "w")
        self.log.write("Date"+"\t"+"M1 current"+"\t"+"M2 current"+"\t"+"M1 speed"+"\t"+"M2 speed"+"\t"+"M1 pwm"+"\t"+"M2 pwm"+"\t"+
                       "Input voltage"+"\t"+"Input current"+"\t"+"M1 temperature"+"\t"+"M2 temperature"+"\t"+"M1 consumption"+"\t"+
                       "M2 consumption"+"\t"+"Controller Error"+"\n")

    def write(self, s):
        self.log.write(s)

    def close(self):
        self.log.close()
```

Joonis 2.4 Klass Log()

Järgmine tuleb klass Buffer(). See on tavaline puhver, kus ajutiselt hoitakse andmed enne tekstidokumendisse kirjutamist. Kuna muutujate arv on määratud (minu juhul on 14), siis ta pidevalt uueneb (Joonis 2.5).

```
#Create buffer for received values and write them to file
class Buffer():

    def __init__(self, log):
        self.logFile = log
        self.lst = []
```

Joonis 2.5 Klass Buffer()

Kasutades meetodi init_serial() avatakse serial-pordi. Kui ühendus õnnestus, siis operaator näeb teavet "HBL2360A connected to:", muul juhul on teave "Can't open port" (Joonis 2.6).

```
#Connect to Roboteq motor controller USB/serial port
def init_serial():

    global serialnotavailable
    global serialnotcreated
    serialPort = "/dev/ttyACM1"
    baudRate = 9600

    try:
```

Joonis 2.6 Meetod init_serial()

Meetod serial.setup() võimaldab serial-ühenduse kaotuse korral seda kohe uuesti taastuda (Joonis 2.7).

```
def serial_setup():

    try:
        ser.open()
    except:
        return
```

Joonis 2.7 Meetod serial.setup()

ReadFromSerial() meetod on vaja andmete saatmiseks kontrollerile ja seejärel nende vastuvõtmiseks ja dekodeerimiseks. Peale selle seda kasutatakse ka serial-ühenduse pidevaks jälgimiseks (Joonis 2.8).

```
#Receive motor values from Roboteq controller
def ReadFromSerial(commandinput, motorChannel):

    global serialnotavailable
    global serialnotcreated

    if serialnotcreated == 1: #Try to create the serial connecti
        init_serial()

    serial_setup() #Set up serial connection

    command = commandinput + str(motorChannel) + ' \r'
    value = 0
```

Joonis 2.8 Meetod ReadFromSerial()

Meetod read_motor_values() on põhifunktsioon ja selle juhtimiskoodi sisenemispunkt. Ta kutsub kõiki muid juhtimiskoodis olevaid meetodeid ning klasse ja töötab ainult siis, kui mootorikontrolleri ja Jetson Nano vahel on olemas ühendus (Joonis 2.9).

```
def read_motor_values():

    global start_time
    global check_time
    start_time = time.time()

    try:
        while True:
            start_time2 = time.time()
```

Joonis 2.9 Meetod read_motor_values()

Karjääris katsetamise käigus leiti, et elektromagnetilise ühilduvuse (EMÜ) tõttu läheb osa usb kaudu edastatud andmetest kaduma.

Üks lahendus oli andmete korduv küsimine nii kaua kuni tuleb. Esimeses tarkvara variandis seda ei olnud, kuna laboris andmete edastamisega ei olnud mingit probleemi. Sellepärast oli vaja teha niimodi, et kui vastus tagasi ei tule, siis pordi avatakse pidevalt uuesti.

Teine lahendus oli teise andmeedastusmeetodi kasutamine (Joonis 2.1b). Peamine erinevus on selles, et serial-ühenduse asemel on vaja kasutada MAVlink-i (micro air vehicle link) (Lisa 2 Juhtimiskood 2.

Mõlemad lahendused said ellu viidud vaid ajapuuduse tõttu ei saanud teise reaalsetes tingimustes testida.

3. Katsetulemused ja analüüs

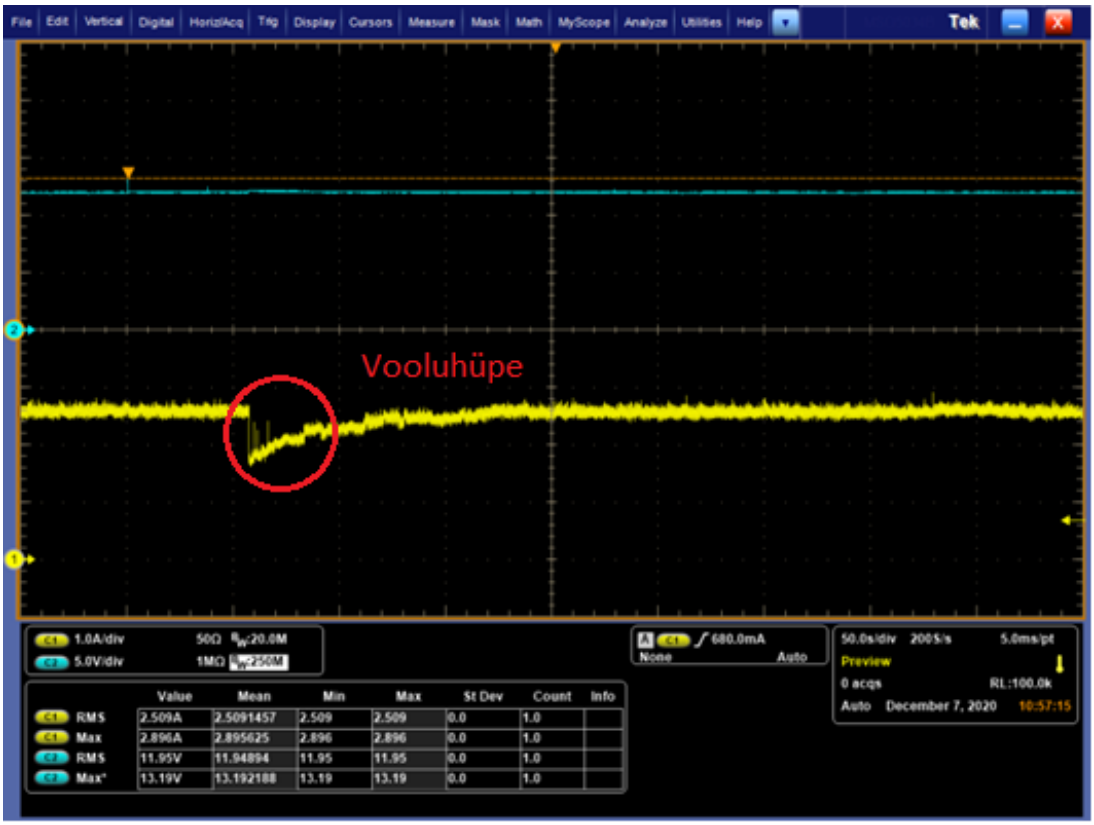
Et oleks täielik ülevaade sellest, kuidas robotlaeva energiatarbimist optimeerida, on vaja aru saada kui palju tarbivad energiat juhtimissüsteem ja tõukejõusüsteem. Üks töö eesmärke oli mõõta süsteemi erinevate komponentide energiatarbimist ja vajadusel uurida võimalusi energiakulu optimeerimiseks.

3.1 Juhtimissüsteemi võimsuse mõõtmine

Võimsuse mõõtmiseks ühendati ostsilloskoopi otse juhtimiskohvriga, kus asub NYMO juhtimissüsteem, nagu on näidatud Joonis 3.1.

Joonis 3.1 Juhtimisüsteemi voolu ja pinge mõõtmine

Kolme minuti jooksul mõõdeti kaks suurust - pinge ja vool. Tulemused on toodud Joonis 3.2.



Joonis 3.2 Mõõtmistulemused

Graafik näitab, et keskmine pinge on 12 V ja maksimaalne on 13,2 V ning kesmine vooluväärtus on 2,5 A ja maksimaalne 2,9 A. Graafiku alguses on märgatav vooluhüpe (Joonis 3.2). See on tingitud 4G ruuterile restardi tegemisest. Tema vool on 2 A. Kõik andmed on sisestatud Tabel 3.1.

Tabel 3.1 Mõõtmistulemused

	Pinge	**Vool**	**Võimsus**
Keskmine	12 V	2,5 A	30,1 W
Maksimaalne	13,2 V	2,9 A	38,3 W

Võimsuse arvutamiseks kasutan järgmist valemit:

$$P = U \cdot I \qquad (3.1.1)$$

kus, *P* – arvutuslik võimsus, W

U – juhtimissüsteemi pinge, V

I – juhtimissüsteemi vool, A

Vastavalt valemile (3.1.1) sain kätte juhtimissüsteemi keskmise ja maksimaalse võimsuse:

$P = 12 \cdot 2{,}5 = 30\ W$

$P_{max} = 13{,}2 \cdot 2{,}9 = 38{,}3\ W$

3.2 Tõukejõusüsteemi võimsuse mõõtmine

Tõukejõusüsteem koosneb kahest komponendist – tüürservod ja mootorid.

3.2.1 Tüürservod

Tüüre servo mootoriks on valitud D845WP servod (Joonis 3.3).

Joonis 3.3 D845WP servo

Servol tööpinge on 12 V, koormamata vool on 1,6 A ja vool maksimaalsel koormusel on 10 A [23]. Kuna koormusega ei ole võimalust mõõta ja servo ise töötab väga lühikest aega järjest, siis seetõttu võime väita, et töövool on umbes on 1,6 A. Seega keskmine ja maksimaalne servode võimsused on:

$P = 12 \cdot 1{,}6 = 19{,}2\ W$

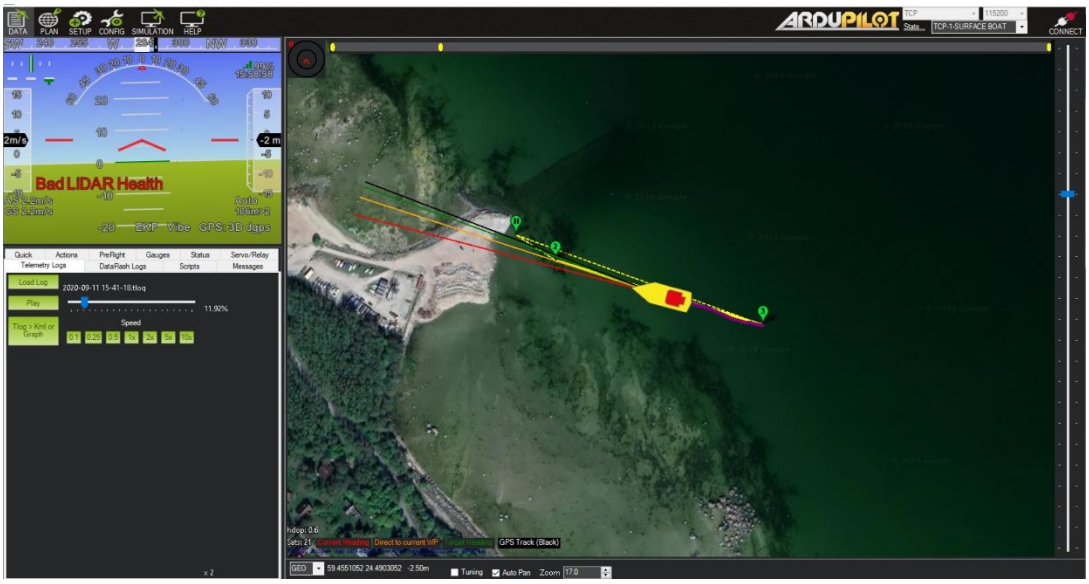
$P_{max} = 12 \cdot 10 = 120\ W$

3.2.2 Mootorid

Laeva veavad edasi kaks 1,5 kW võimsusega püsimagnetitega harjavaba kolmefaasilist asünkroonmootorit (Joonis 3.4). Nende tööpinge on 48 V [24].

Joonis 3.4 Robotlaeva mootor

Mootorite võimsusetarbimise mõõtmine toimus reaalses katse-keskkonnas Tilgu sadamas. Robotlaev läbis sama marsruudi erineva kiirusega – 1,2 m/s, 1,5 m/s ja 2,3 m/s. Samuti kasutati kahte režiimi - manuaalne ja automaat. (Joonis 3.5).



Joonis 3.5 Robotlaeva marsruut

Manuaalrežiimis operaator ise kontrollib ja juhib robotlaeva puldiga. Automaatrežiimi korral sõidab NYMO autonoomselt järgides etteantud kiiruseid ja teekonnapunkte.

Esimene katse - 1,2 m/s, manuaal

Esimesel katsel robotlaev sõitis kiirusega 1,2 m/s manuaal režiimis. Tulemused on näidatud Tabel 3.2.

Tabel 3.2 Mõõtmistulemused

	Pinge,V	**M1, A**	**M2, A**	**M1, W**	**M2, W**
	52,1	8,3	8,1	432,43	422,01
	52	8,4	7,5	436,8	390
	51,9	8,2	8,1	425,58	420,39
	51,6	8,7	8,3	448,92	428,28
	51,9	8,6	8,3	446,34	430,77
	51,9	8,6	8,6	446,34	446,34
	51,9	8,6	8,3	446,34	430,77
	51,8	8,5	7,9	440,3	409,22
	51,8	8,7	8,3	450,66	429,94
	51,9	8,7	8,1	451,53	420,39
	51,9	8,6	8,1	446,34	420,39
	51,6	8,6	8,3	443,76	428,28
	51,6	8,9	8,1	459,24	417,96
	51,8	8,9	8,2	461,02	424,76
	51,9	8,4	7,7	435,96	399,63
	51,8	6,5	8	336,7	414,4
	51,9	8,5	8,2	441,15	425,58
	51,8	8,8	8,2	455,84	424,76
	51,9	8,6	8,3	446,34	430,77
	51,9	8,7	8,4	451,53	435,96
Keskmine, W				**440,2**	**422,5**
Maksimaalne, W				**461,0**	**446,3**

Esimeses veerus on mootorite sisendpinge või akupinge. Teises ja kolmandas on mootorite töövoolud kiirusel 1,2 m/s. Neljandas veerus on vasaku mootori võimsustarbimine ja viimases on parema mootori võimsustarbimine. Eelvimases reas on keskmine võimsustarbimise, antud juhul vasaku mootori kohta oli see 440,2 W ja parema 422,5 W. Viimane rida on maksimaalne võimsus katse ajal.

Teine katse - 1,2 m/s, automaat

Teisel katsel robotlaev sõitis sama kiirusega 1,2 m/s vaid automaat režiimis. Tulemused on näidatud Tabel 3.3.

Tabel 3.3 Mõõtmistulemused

	Pinge,V	M1, A	M2, A	M1, W	M2, W
	51,5	11,1	6,8	571,65	350,2
	51,5	7,3	5,2	375,95	267,8
	51,4	10,3	14,2	529,42	729,88
	51,2	9,8	12	501,76	614,4
	51,6	3,5	4,4	180,6	227,04
	51,4	9	11,1	462,6	570,54
	51,6	4,5	8,7	232,2	448,92
	51,6	12,3	5,1	634,68	263,16
	51,6	7,4	9,1	381,84	469,56
	51,3	5,4	10,1	277,02	518,13
	51,5	9,2	7,5	473,8	386,25
	51,6	4,4	8,3	227,04	428,28
	51,5	8,9	8,1	458,35	417,15
	51,4	3,3	10,6	169,62	544,84
	51,6	6,8	11	350,88	567,6
	51,5	7,6	4,9	391,4	252,35
	51,5	9,5	3,7	489,25	190,55
	51,6	5,9	13,4	304,44	691,44
	51,7	6,6	12,6	341,22	651,42
	51,6	4,7	10,8	242,52	557,28
Keskmine, W				**379,8**	**457,3**
Maksimaalne, W				**634,7**	**729,9**

Teise katse jooksul mootorite keskmised võimsused on järgmised – 379,8 W ja 457,3 W, ning maksimaalsed – 634,7 W ja 729,9. Hoolimata sellele, et selle katse kiirus on täpselt sama, mis oli eelmises, tulemused on erinevad. See on tingitud asjaolust, et automaatrežiimis püüab robotlaev hoida täpselt antud kursis, mistõttu peab ta mõnikord pöörama ja vähendama oma kiirust.

Kolmas katse - 1,5 m/s, manuaal

Kolmas katsel robotlaev sõitis kiirusega 1,5 m/s manuaal režiimis. Tulemused on näidatud Tabel 3.4.

Tabel 3.4 Mõõtmistulemused

	Pinge,V	M1, A	M2, A	M1, W	M2, W
	51,3	12,7	12,7	651,51	651,51
	51,2	13	13,2	665,6	675,84
	51,2	12,5	13	640	665,6
	51,2	12,6	13,4	645,12	686,08
	51,2	12,7	13,2	650,24	675,84
	51,2	12,9	13,3	660,48	680,96
	51,2	12,7	13	650,24	665,6
	51,2	12,6	12,9	645,12	660,48
	51,2	12,6	13	645,12	665,6
	51,2	12,8	13,1	655,36	670,72
	51,2	13	13,2	665,6	675,84
	51,2	13	13,1	665,6	670,72
	51,2	12,9	13,1	660,48	670,72
	51,2	12,7	12,8	650,24	655,36
	50,8	12,9	13,2	655,32	670,56
	51,2	12,4	13,1	634,88	670,72
	51	12,9	12,7	657,9	647,7
	51,4	12,9	12,8	663,06	657,92
	51,2	12,6	13	645,12	665,6
	51,2	12,7	12,8	650,24	655,36
Keskmine, W				**652,9**	**666,9**
Maksimaalne, W				**665,6**	**686,0**

Kui võrrelda saadud tulemust manuaal režiimi eelmise katsega, siis on märgata, et kiiruse suurenemisega kasvas ka voolutugevus. Kuna robotlaev sõitis sirgelt ühe kiirusega siis ka võimsusetarbimine mõlemal mootoril põhimõtteliselt sama.

Neljas katse - 1,5 m/s, automaat

Neljas katsel robotlaev sõitis kiirusega 1,5 m/s automaat režiimis. Tulemused on näidatud Tabel 3.5.

Tabel 3.5 Mõõtmistulemused

	Pinge,V	M1, A	M2, A	M1, W	M2, W
	51,7	8,1	12,1	418,77	625,57
	51,9	6,9	12,6	358,11	653,94
	51,8	7,1	14,5	367,78	751,1
	51,5	6,9	13,6	355,35	700,4
	51,8	2,8	17,2	145,04	890,96
	51,8	7,8	14,5	404,04	751,1
	51,6	9,6	13,5	495,36	696,6
	51,6	7,3	8,8	376,68	454,08
	51,8	5,1	17,3	264,18	896,14
	51,6	7,7	11,2	397,32	577,92
	51,7	7,3	11,2	377,41	579,04
	51,7	4,4	13,2	227,48	682,44
	51,6	10,8	7,1	557,28	366,36
	52	7,4	9,4	384,8	488,8
	51,6	3,8	13,4	196,08	691,44
	51,5	6	13,8	309	710,7
	52,2	6,7	7,5	349,74	391,5
	51,7	5,7	8,8	294,69	454,96
	51,8	2	19,4	103,6	1004,92
	51,8	7,6	20	393,68	1036
Keskmine, W				**338,8**	**670,2**
Maksimaalne, W				**557,3**	**1036,0**

Sellel juhul on eelmise automaat katsega muster. Vasakul mootoril on võimsusetarbimine on kaks korda vähem kui paremal.

Viies katse – 2,3 m/s, manuaal

Viies katsel robotlaev sõitis maksimaal kiirusega 2,3 m/s manuaal režiimis. Tulemused on näidatud Tabel 3.6.

Tabel 3.6 Mõõtmistulemused

	Pinge,V	M1, A	M2, A	M1, W	M2, W
	49,6	35,7	36,6	1770,72	1815,36
	49,6	28,8	33,1	1428,48	1641,76
	49,6	29,1	33	1443,36	1636,8
	49,3	29	32,9	1429,7	1621,97
	49,3	28,5	32,8	1405,05	1617,04
	49,9	28,8	32,8	1437,12	1636,72
	49,2	28,5	32,9	1402,2	1618,68
	49,2	28,1	32,7	1382,52	1608,84
	49,3	29,6	33	1459,28	1626,9
	49,3	29,1	32,7	1434,63	1612,11
	49,3	29,1	32,6	1434,63	1607,18
	49,1	29	32,7	1423,9	1605,57
	49,1	28,8	32,4	1414,08	1590,84
	49,2	28,7	28,2	1412,04	1387,44
	49,5	29	32,5	1435,5	1608,75
	49,3	29,6	33,9	1459,28	1671,27
	49,8	29,4	32,3	1464,12	1608,54
	48,9	29,3	23,4	1432,77	1144,26
	50	28,7	33,2	1435	1660
	49	20,3	33,1	994,7	1621,9
Keskmine, W				**1425**	**1597,1**
Maksimaalne, W				**1770,7**	**1815,4**

Mootorite keskmised võimsused on 1425 W ja 1597,1 W ning maksimaal võimsused on 1770,7 W ja 1815,4 W.

Kuues katse – 2,3 m/s, automaat

Kuues katsel robotlaev sõitis maksimaal kiirusega 2,3 m/s automaat. Tulemused on näidatud Tabel 3.7.

Tabel 3.7 Mõõtmistulemused

	Pinge,V	M1, A	M2, A	M1, W	M2, W
	49,3	36,3	34,5	1789,59	1700,85
	50,5	36,1	36,1	1823,05	1823,05
	50,5	36,4	37,3	1838,2	1883,65
	50,1	31,3	26,5	1568,13	1327,65
	49,6	32,1	26,5	1592,16	1314,4
	49,8	29,7	33,4	1479,06	1663,32
	49,9	29,1	33,3	1452,09	1661,67
	50	26,4	34,2	1320	1710
	49,5	30,4	33,1	1504,8	1638,45
	49,3	30,1	21,7	1483,93	1069,81
	49,2	21,3	24,4	1047,96	1200,48
	49,2	28,7	32,9	1412,04	1618,68
	49,5	26,7	30,1	1321,65	1489,95
	49,8	20,2	33,7	1005,96	1678,26
	49,5	18,2	34	900,9	1683
	49,5	20,1	32,8	994,95	1623,6
	49,5	24,7	31,9	1222,65	1579,05
	49,3	29,5	32,7	1454,35	1612,11
	49,5	22,4	23,2	1108,8	1148,4
	49,8	29,1	32,6	1449,18	1623,48
Keskmine, W				**1388,5**	**1552,5**
Maksimaalne, W				**1838,2**	**1883,7**

Mootorite keskmised võimsused on 1388,5 W ja 1552,5 W ning maksimaal võimsused on 1838,2 W ja 1883,7 W.

Seitsmes katse – multispeed, automaat

See test tehti ainult automaat režiimis. Kiirus kogu marsruuti vältel varieerus 1,2 m/s kuni 2,3 m/s. Tulemused on näidatud Tabel 3.8.

Tabel 3.8 Mõõtmistulemused

	Pinge,V	M1, A	M2, A	M1, W	M2, W
	51,7	0,2	0,1	10,34	5,17
	51,7	0,3	2,2	15,51	113,74
	51,7	0,5	2,6	25,85	134,42
	51,8	1,4	4,6	72,52	238,28
	51,7	1,7	4,8	87,89	248,16
	51,5	2	5,3	103	272,95
	51,7	2	6,7	103,4	346,39
	51,9	2,5	7,1	129,75	368,49
	51,7	3	7,6	155,1	392,92
	51,8	3,5	9,2	181,3	476,56
	51,6	5,5	10	283,8	516
	51,7	6	10,3	310,2	532,51
	51,6	6,4	10,4	330,24	536,64
	51,8	6,4	10,6	331,52	549,08
	51,8	7	10,7	362,6	554,26
	51,3	7,2	11,2	369,36	574,56
	51,6	7,5	11,7	387	603,72
	51,2	7,6	11,7	389,12	599,04
	51,4	7,7	12,1	395,78	621,94
	51,6	8	12,4	412,8	639,84
	51,4	8,1	13	416,34	668,2
	51,2	9,6	19,1	491,52	977,92
	51,5	9,9	19,2	509,85	988,8
	50,5	10	21,7	505	1095,85
	50,6	10,1	26,7	511,06	1351,02
	50,8	10,1	27	513,08	1371,6
	50,3	10,2	27,6	513,06	1388,28
	50,6	10,8	28,6	546,48	1447,16
	50,8	11,7	29,8	594,36	1513,84
	50,5	12,1	31,9	611,05	1610,95
	49,8	12,7	34,5	632,46	1718,1
	49,5	12,9	34,7	638,55	1717,65
	50	13,3	34,9	665	1745
	49,7	13,9	34,9	690,83	1734,53
	50	14,3	35,2	715	1760
	49,8	14,4	35,4	717,12	1762,92
Keskmine, W				**391,9**	**892,3**
Maksimaalne, W				**717,2**	**1762,9**

Mootorite keskmised võimsused on 391,9 W ja 892,3 W ning maksimaal võimsused on 717,2 W ja 1762,9 W.

3.3 Tulemuste analüüs

Iga testimis ajaintervall oli 2 minutit. Eelmistes punktides kogutud andmete analüüsimiseks on vaja kostada üldist tabelit.

Tabel 3.9 Tulemuste koondtabel

	Keskmine, W	**Maksimaalne, W**	**Kokku, W**
Juhtimissüsteem	30	38,3	
Tüürservod	19,2	120	
M1, 1,2 m/s, manuaal	440,2	461	**931,1**
M2, 1,2 m/s, manuaal	422,5	446,3	
M1, 1,2 m/s, automaat	379,8	634,7	**905,5**
M2, 1,2 m/s, automaat	457,3	729,9	
M1, 1,5 m/s, manuaal	652,9	665,6	**1388,2**
M2, 1,5 m/s, manuaal	666,9	686,1	
M1, 1,5 m/s, automaat	338,8	557,3	**1077,4**
M2, 1,5 m/s, automaat	670,2	1036	
M1, 2,3 m/s, manuaal	1425	1770,7	**3090,5**
M2, 2,3 m/s, manuaal	1597,1	1815,4	
M1, 2,3 m/s, automaat	1388,5	1838,2	**3009,4**
M2, 2,3 m/s, automaat	1552,5	1883,7	
M1, multi, automaat	391,9	717,2	**1352,6**
M2, multi, automaat	892,3	1762,9	

Tabel 3.9 näitab, et kõige väiksem võimsusetarbimine on juhtimissüsteemis, keskmiselt ainult 29,98 W. Kõige suuremat keskmist tarbimist demonstreeris teine robotlaeva mootor 2,3 m/s manuaalrežiimis ja see võrdus 1597,1 W.

Kontrollime kui palju protsenti kogu võimsusetarbimist koostab juhtimissüsteem kõigel madalaimal koormusel. Selleks on vaja teada millises testis robotlaeva mootorid oli kõige vähem koormatud. See on esimene automaat režiimi katse.

Kogu võimsusetarbimine:

$$P_{kogu} = P_{juht} + P_{tõuk} = 29{,}98 + 30 \cdot 2 + 379{,}8 + 457{,}3 = 927{,}08\ W$$

$$\frac{P_{juht}}{P_{kogu}} \cdot 100\% = \frac{29{,}98}{927{,}08} \cdot 100 = 3\%$$

See tähendab, et isegi kõige väiksema koormuse korral juhtimissüsteem moodustab ainult 3% kogu energiatarbimisest. Seetõttu on võimalik julgelt öelda, et juhtimissüsteemi võimsusetarbimist pole mõtet optimeerida, sest suurte koormuste korral väheneb see protsent veelgi.

Tabelis 3.9 on näha, et automaat režiimi võimsusetarbimine on alati väiksem, kui võimsusetarbimine manuaal režiimis. Seda tulemust oodati, kuna automaat režiimis robotlaev ise sõidab lõpppunktini püüdes antud marsruuti võimalikult täpselt järgida.

Kontrollime, mitu protsentil oli automaat režiim säästlikum:

$$\frac{P_a}{P_m} = 100\% - \frac{905{,}5}{931} \cdot 100\% = 2{,}7\ \%$$
$$\frac{P_a}{P_m} = 100\% - \frac{1077{,}4}{1388{,}2} \cdot 100\% = 22{,}4\ \%$$
$$\frac{P_a}{P_m} = 100\% - \frac{3009{,}4}{3090{,}5} \cdot 100\% = 2{,}6\ \%$$

Arvutustest järeldub, et rohkem energiat säästakse, kiirusega 1,5 m/s sõitmisel. Sellest saab järelda, et liiga väikese või vastupidi liiga suure kiirusega sõitmisel, ei anna võõimalust palju säästada. Aga et selles kindell olla, on vaja rohkem katset teha.

Testimise ajal märgati veel ühte omadust. Automaat režiimi testimise ajal robotlaeva vasak mootor alati vähem koormatud. Mis selle põhjuseks on, on väga raske öelda. Sellel võib olla mitu põhjust. Näiteks tüürservo viga või see on ka seotud sellega, et robotlaev ise korrigeerib oma marsruudi. Võib olla ka põhjus EMC-s ja andmete edastamise protsessis osa neist lihtsalt kaob või tuleb veaga.

Andmete üksikasjalikuma analüüsi tegemiseks on vaja veel vähemalt paar testi teha.

KOKKUVÕTE

Käesoleva lõputöö eesmärk oli arendada diagnoostika ja loogimise tarkvarat mis aitab tagada NYMO robotlaeva töökindluse ja energiatõhususe.

Magistriöö esimene samm oli uurida missugused autonoomsed süsteemid on juba olemas maailmas. Missugused sarnased lahendused turul ja tööstusest kasutusel olemas. Samuti oli vaja mõista, mis on üldse autonoomsete süsteemide diagnoostika ja kuidas seda rakendatakse. Selle uurimuse järel tuli välja, et täna palju suured riigid, nagu Norra, Taani, Rootsi, juba alustavad autonoomseid lahenduseid rakendada merealas või intensiivselt seda arendavad. Põhjus on selle lihtne - üha kasvava globaalse kaubanduse suurenevad veomahud ja vajadus kärpida vedude kulusid.

Teise sammuna tuli ise tutvuda NYMO robotlaevaga, mille arendamisega tegelevad Tallinna Tehnikaülikooli elektroenergeetika ja mehhatroonika instituudi teadlased. Selle sammu käigus uurisin robotlaeva ehitust. Missugused mehaanilised lahendused on rakendunud ja millised on tema tugevused ja nõrkused. Kõige huvitam osa selles oli juhtimissüsteem. See on põhimõtteliselt kogu iselaeva kontseptsiooni aju, mis on võimalik lihtsalt ümber tõsta ja panna mõne teise aluse peale. Tutvusin, kuidas selline keeruline süsteem töötab ja kuidas erinevad väikesed osad selles süsteemis suhtlevad ja edastavad andmed omavahel.

Kolmas sammuga hakkasin läbi mõtlema, kuidas minu loogimissüsteem peaks töötama. Missuguseid andmeid on vaja koguda robotlaeva testimise ajal ja kuidas seda realiseerida. Kõigepealt oli vaja mõista, kuidas omavahel edastavad andmed Jetson Nano ja Roboteq mootorikontroller. Pärast seda koostasin algoritmi plokkskeemi ja kirjutasin koodi. Ainus hetk, mida ma laboris töötades arvesse ei võtnud on elektromagnetilised häired. Alles esimesel veekatsel märkasin, et koormuse all töötamise ajal osa andmetest kaob ning ühendus mootorikontrolleri ja Nano Jetsoni vahel pidevalt kaob. Selle tõttu pidin veetma lisaaega koodi parandamiseks ja testimiseks.

Lõputöö viimases osas on testimine ja katsetulemuste analüüs. Oli vaja mõõta juhtimissüsteemi ja tõukejõusüsteemi võimsusetarbimist. Selle tulemusena selgus, et juhtimissüsteem moodustab ainult 3% kogu energiatarbimisest ja automaat režiimi ajal võimsusetarbimine on vähem, kui manuaal režiimis. Samuti saadi muid huvitavaid tulemusi, kuid selle kontrollimiseks on vaja rohkem andmeid.

Tulevikus on vaja teha veel mõned katsed. Kuna tervikpildi saamiseks ei piisa ühest testist.

Hetkel iga tarkvara osa töötab eraldi ja ei suhtle kuidagi omavahel. Järgmine samm on teha niimoodi, et kõik andmed olid konverteeritud ühte formaati.

Vaatamata vähesele testide arvu, võiks püstitatud eesmärgid lugeda täidetuks.

SUMMARY

The aim of this thesis was to develop diagnostics and logging software that will help ensure the reliability and energy efficiency of the NYMO robotic vessel.

The first step of the master's thesis was to study what kind of autonomous systems already exist in the world. What similar solutions are available in the market and industry. It was also necessary to understand what is diagnostics of autonomous systems and how it is implemented in real projects. This study revealed that today many large countries, such as Norway, Denmark, Sweden, are already implementing or intensively developing autonomous solutions in the maritime field. The reason is simple - increasing transport volumes in growing global trade and the need to cut transport costs.

The second step was to get acquainted with the NYMO robotic vessel, which is being developed by researchers from the Department of Electrical Power Engineering and Mechatronics at Tallinn University of Technology. During this step, I researched the construction of a robotic vessel. What mechanical solutions have been implemented and what are its strengths and weaknesses. The most interesting part of this was the modular control system. It is basically the brain of the whole ASV concept and it was designed as an independent module that could be easily removed and placed onto other ASV. I got acquainted with how such a complex system works and how different small parts of this system communicate and transfer data to each other.

In the third step, I started thinking about how my logging system should work. What data needs to be collected during robotic vessel testing and how to implement it. First, it was necessary to understand how the data is communicated between the Nvidia Jetson Nano and the Roboteq motor controller. After that, I made the algorithm block diagram and wrote the code. The only moment I didn't take into account while working in the lab is electromagnetic interference. It was only during the first test on water that I noticed that while working under load, some of the data was lost and the connection between the Roboteq motor controller and the Nvidia Jetson Nano was constantly lost. Because of this, I had to spend extra time correcting and testing the code.

The last part of the thesis is testing and analysis of test results. It was necessary to measure the power consumption of the control system and the propulsion system. As a result, it was found that the control system accounts for only 3% of total power consumption, and in automatic mode, power consumption is less than in manual mode. Other interesting results were also obtained, but more data are needed to verify them.

Some more experiments are needed to be done in the future. Because one test is not enough to get a complete picture.

Currently, each piece of software works separately and does not communicate in any way with each other. The next step is to make it so that all the data was converted to one format.

Despite the small number of tests, the set goals could be considered fulfilled.

KASUTATUD KIRJANDUSE LOETELU

[1] Review of Maritime transport.
https://unctad.org/topic/transport-and-trade-logistics/review-of-maritime-transport

[2] Starship Technologies.
https://www.starship.xyz/

[3] Iseauto projekt.
https://iseauto.taltech.ee/

[4] "Norway to build fleet of robotics ships with ultra-low emission".
https://www.maritimebusinessworld.com/norway-to-build-fleet-of-robotic-ships-with-ultra-low-emission-1875h.htm (18.09.2020)

[5] D. Edwards, " Autonomous ships ahoy!".
https://roboticsandautomationnews.com/2020/01/18/autonomous-ships-ahoy/28706/ (18.01.2020)

[6] B. Marr, "The Incredible Autonomous Ships Of The Future: Run By Artificial Intelligence Rather Than A Crew".
https://www.forbes.com/sites/bernardmarr/2019/06/05/the-incredible-autonomous-ships-of-the-future-run-by-artificial-intelligence-rather-than-a-crew/?sh=452843256fbf (05.06.2019)

[7] F. Densford, "Un-man the Decks: How autonomous shipping vessels could reshape the waterways".
https://www.therobotreport.com/ghost-ships-irl-autonomous-cargo-boats-disrupt-massive-shipping-industry/ (10.10.2017)

[8] SEA-KIT X.
https://www.sea-kit.com/commercial (2020)

[9] S. Paulus, "NYMO shows the future of maritime industry".
https://researchinestonia.eu/2019/11/07/nymo-shows-the-future-of-maritime-industry/ (07.11.2019)

[10] A. Vill, " Iselaev käis mere taga".
https://director.ee/2019/10/02/iselaev-kais-mere-taga/ (02.10.2019)

[11] A. Alvela, "Robotlaev Nymo künnab laineid".
https://tehnikamaailm.ee/artikkel/robotlaev-nymo-kunnab-laineid (31.10.2019)

[12] C. Pawsey, "Advanced Automotive Diagnostics Systems - From Diagnostics to Prognostics".
https://www.automotive-iq.com/autonomous-drive/articles/advanced-automotive-diagnostics-systems-from-diagnostics-to-prognostics (13.11.2018)

[13] "Mis on digitaalne kaksik?".
https://est.4meahc.com/what-is-digital-twin-70962

[14] "DIGITAALNE KAKSIK HAKKAB ROBOTLAEVALE NYMO MEREL TEED NÄITAMA".
https://www.taltech.ee/uudised/digitaalne-kaksik-hakkab-robotlaevale-nymo-merel-teed-naitama (13.10.2020)

[15] I. Roasto, H. Mõlder, T. Jalakas, T. Möller, K. Tabri, M. Enok, "Design and Testing of an Universal Autonomous Surface Vehicle".

[16] Roboteq BLDC motor controller HBL2360A.
https://www.roboteq.com/products/products-brushless-dc-motor-controllers/hbl2360a-258-detail

[17] NVIDIA Jetson Nano.
https://developer.nvidia.com/embedded/jetson-nano-developer-kit

[18] ACS712 Current Sensor Module.
https://components101.com/sensors/acs712-current-sensor-module

[19] MAVLink Developer Guide.
https://mavlink.io/en/

[20] Advanced Brushed and Brushless Digital Motor Controllers. User Manual.
https://www.roboteq.com/docman-list/motor-controllers-documents-and-files/documentation/user-manual/272-roboteq-controllers-user-manual-v17/file

[21] RoboRun+ PC Utility. User Manual.
https://www.generationrobots.com/media/roboteq/roborun-+-utility-user-manual.pdf

[22] MicroBasic Reference.
https://www.roboteq.com/microbasic-reference

[23] D845WP Servo motor.
https://hitecrcd.com/products/servos/digital/d-series/d845wp-digital-mega-scale-monster-torque-waterproof-servo/product

[24] BLDC-108 motor.
https://www.miromax.lt/en/m-6/c-39/c-45-brushless_bldc_motor_with_permanent_magnet/product-509-bldc-108_motor_-_max_nominal_power_3kw

LISAD

Lisa 1 Juhtimiskood

```
import time
import serial
from datetime import datetime

#Create log file
class Log():

    def __init__(self):
        self.dateTimeObj_1 = datetime.now()
        self.timestampStr_1 = self.dateTimeObj_1.strftime("%d-%b-%H:%M")
        path = "/home/nymo/Nymo python scripts/Alx_Diagn/Logs/measurements.txt"
        new_path = '%s_%s' % (path, self.timestampStr_1)
        self.log = open(new_path, "w")
        self.log.write("Date"+"\t"+"M1 current"+"\t"+"M2 current"+"\t"+"M1
speed"+"\t"+"M2 speed"+"\t"+"M1 pwm"+"\t"+"M2 pwm"+"\t"+
                    "Input voltage"+"\t"+"Input current"+"\t"+"M1
temperature"+"\t"+"M2 temperature"+"\t"+"M1 consumption"+"\t"+
                    "M2 consumption"+"\t"+"Controller Error"+"\n")

    def write(self, s):
        self.log.write(s)

    def close(self):
        self.log.close()

#Create buffer for received values and write them to file
class Buffer():

    def __init__(self, log):
        self.logFile = log
        self.lst = []

    def push(self, value):
        self.dateTimeObj_2 = datetime.now()
        self.timestampStr_2 = self.dateTimeObj_2.strftime("%d-%b-%Y
(%H:%M:%S)")

        self.lst.append(value)


        if len(self.lst) == 13:
            #print(self.lst)

self.logFile.write(self.timestampStr_2+"\t"+str(self.lst[0])+"\t"+str(self.lst[1])+"\t"+s
tr(self.lst[2])+"\t"+str(self.lst[3])+"\t"+

str(self.lst[4])+"\t"+str(self.lst[5])+"\t"+str(self.lst[6])+"\t"+str(self.lst[8])+"\t"+str(
self.lst[11])+"\t"+

str(self.lst[12])+"\t"+str(self.lst[9])+"\t"+str(self.lst[10])+"\t"+str(self.lst[7])+"\n")
```

```
            self.lst = self.lst[13:]

log = Log()
buffer = Buffer(log)

#Connect to Roboteq motor controller USB/serial port
def init_serial():

    global serialnotavailable
    global serialnotcreated
    serialPort = "/dev/ttyACM1"
    baudRate = 9600

    try:
        ser = serial.Serial(
            serialPort,
            baudRate,
            parity = serial.PARITY_NONE,
            stopbits = serial.STOPBITS_ONE,
            bytesize = serial.EIGHTBITS,
            timeout = 0.3,
            writeTimeout = 0) #ensure non-blocking

        if ser.isOpen():
            serialnotavailable = 0
            serialnotcreated = 0
            print("HBL2360A connected to: " + ser.portstr + "\n")
            return ser
        return None

    except serial.SerialException:
        serialnotavailable = 1
        serialnotcreated = 1
        print("Can't open port")
        return None

def serial_setup():

    try:
        ser.open()
    except:
        return

#Receive motor values from Roboteq controller
def ReadFromSerial(commandinput, motorChannel):

    global serialnotavailable
    global serialnotcreated

    if serialnotcreated == 1: #Try to create the serial connection when it was failed at
startup
        init_serial()

    serial_setup() #Set up serial connection

    command = commandinput + str(motorChannel) + ' \r'
    value = 0
```

```
    try:
        ser.write(command.encode())
        serialnotavailable = 0
    except:
        Controller_Error = "No connection!"
        if commandinput == '?FF ':
            value = "No connection!"
        try:
            ser.close()
            serialnotavailable = 1
        except:
            pass

    try:
        data = ser.readline().decode().strip()
        value = int(data.split('=')[1])
        serialnotavailable = 0
    except:
        Controller_Error = "No connection!"
        if commandinput == '?FF ':
            value = "No connection!"
        try:
            ser.close()
            serialnotavailable = 1
        except:
            pass

    return value

def read_motor_values():

    global start_time
    global check_time
    start_time = time.time()

    try:
      while True:
          start_time2 = time.time()

          M1_current = ReadFromSerial('?A ',1)/10 #Motor 1 current, A
          buffer.push(M1_current)
          M2_current = ReadFromSerial('?A ',2)/10 #Motor 2 current, A
          buffer.push(M2_current)

          M1_speed = ReadFromSerial('?BS ',1)/10 #Motor 1 speed, RPM
          buffer.push(M1_speed)
          M2_speed = ReadFromSerial('?BS ',2)/10 #Motor 2 speed, RPM
          buffer.push(M2_speed)

          M1_pwm = ReadFromSerial('?P ',1)/10 #Motor 1 Duty cycle, %
          buffer.push(M1_pwm)
          M2_pwm = ReadFromSerial('?P ',2)/10 #Motor 2 Duty cycle, %
          buffer.push(M2_pwm)

          Input_voltage = ReadFromSerial('?V ',2)/10 #Roboteq controller input voltage,
V
```

```
        buffer.push(Input_voltage)

        Input_current1 = ReadFromSerial('?BA ',1)/10 #Roboteq controller ch1
current, A
        Input_current2 = ReadFromSerial('?BA ',2)/10 #Roboteq controller ch2
current, A

        Controller_Error = ReadFromSerial('?FF ',1) #Roboteq controller error
        buffer.push(Controller_Error)

        start_time3 = time.time()
        check_time = round((start_time3 - start_time2), 2)
        print(check_time)

        Input_current = round((Input_current1 + Input_current2)/10, 2) #Roboteq
controller current, A
        buffer.push(Input_current)

        M1_consumption = abs(round((M1_current*Input_voltage*check_time), 2))
#Motor 1 Energy consumption, Ws
        buffer.push(M1_consumption)
        M2_consumption = abs(round((M2_current*Input_voltage*check_time), 2))
#Motor 2 Energy consumption, Ws
        buffer.push(M2_consumption)

        M1_temperature = ReadFromSerial('?T ',2) #Motor 1 temperature, C
        buffer.push(M1_temperature)
        M2_temperature = ReadFromSerial('?T ',3) #Motor 2 temperature, C
        buffer.push(M2_temperature)


        Roboteq_msg =
{'Motor1_current':M1_current,'Motor2_current':M2_current,'Motor1_speed':M1_speed,
'Motor2_speed':M2_speed,

'Motor1_pwm':M1_pwm,'Motor2_pwm':M2_pwm,'Input_voltage':Input_voltage,'Input_
current':Input_current,'Controller_Error':Controller_Error,
                    'Motor1_temperature':M1_temperature,
'Motor2_temperature':M2_temperature, 'Motor1_consumption':M1_consumption,
'Motor2_consumption':M2_consumption}

        #print(str(Roboteq_msg))
        print("M1 current=" + str(M1_current) + " M2 current=" + str(M2_current) +
" M1 speed=" + str(M1_speed) + " M2 speed=" + str(M2_speed))
        print("M1 duty cycle=" + str(M1_pwm) + " M2 duty cycle=" + str(M2_pwm) +
" Input voltage=" + str(Input_voltage) + " Input current=" + str(Input_current))
        print("Controller Error: " + str(Controller_Error))
        print("Energy consumption of motor 1=" + str(M1_consumption) + " Energy
consumption of motor 2=" + str(M2_consumption) + "\n")


        time.sleep(0.1)

    except KeyboardInterrupt:
        ser.close()
        log.close()
        print("Program Stopped!")
```

```
ser = init_serial()

if ser is not None:
    read_motor_values()
```

Lisa 2 Juhtimiskood 2. MAVlinki kasutades

```
import time

from datetime import datetime

from pymavlink import mavutil




master = mavutil.mavlink_connection('tcp:192.168.100.3:5759')


class Log():


    def __init__(self):

        self.dateTimeObj_1 = datetime.now()

        self.timestampStr_1 = self.dateTimeObj_1.strftime("%d-%b-%H:%M")

        path = "/home/nymo/Nymo python scripts/Alx_Diagn/Logs/mavlink.txt"

        new_path = '%s_%s' % (path, self.timestampStr_1)

        self.log = open(new_path, "w")

        self.log.write("Date"+"\t"+"Battery                    current"+"\t"+"Battery
voltage"+"\t"+"Battery SOC"+"\t"+"Power"+"\n")


    def write(self, s):

        self.log.write(s)


    def close(self):
```

```
        self.log.close()


class Buffer():


    def __init__(self, log):
        self.logFile = log
        self.lst = []


    def push(self, value):
        self.dateTimeObj_2 = datetime.now()
        self.timestampStr_2 = self.dateTimeObj_2.strftime("%d-%b-%Y (%H:%M:%S)")


        self.lst.append(value)




        if len(self.lst) == 4:
            #print(self.lst)

self.logFile.write(self.timestampStr_2+"\t"+str(self.lst[0])+"\t"+str(self.lst[1])+"\t"+s
tr(self.lst[2])+"\t"+str(self.lst[3])+"\n")
            self.lst = self.lst[4:]


log = Log()
buffer = Buffer(log)
```

```
def read_values():


    while True:
        msg = master.recv_msg()
        if msg == None:
            pass
        else:
            try:
                msg2=msg.to_dict()
                if msg2['mavpackettype'] == 'BATTERY_STATUS' and ((msg2['voltages'])[0]) < 65535:
                    #BattInfo = msg2
                    #print(BattInfo)
                    Battery_current = round((msg2['current_battery'])/100,1)
                    buffer.push(Battery_current)
                    Battery_voltage = round(((msg2['voltages'])[0])/1000,1)
                    buffer.push(Battery_voltage)
                    Battery_SOC = round(((msg2['voltages'])[9])/10,1)
                    buffer.push(Battery_SOC)
                    Power = abs(round((Battery_current*Battery_voltage),1))
                    buffer.push(Power)
                    print("Battery current = "+str(Battery_current)+" Battery voltage = "+str(Battery_voltage)+" SOC = "+str(Battery_SOC))
```

```
        except KeyboardInterrupt:
            print("Program stopped")
            log.close()

read_values()
```