TALLINNA TEHNIKAÜLIKOOL
Infotehnoloogia teaduskond

Magnus Teekivi 155705IAPB

LAIENDATAVA ROBOTSÕIDUKITE KASUTAJALIIDESE ARENDAMINE VUE.JS BAASIL

Bakalaureusetöö

Juhendaja: Gert Kanter
magister

Tallinn 2019

Autorideklaratsioon

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

Autor: Magnus Teekivi

27.05.2019

Annotatsioon

Lõputöö esmaseks eesmärgiks on arendada ettevõtte Milrem Robotics robotsõidukiga THeMIS kasutatavat kasutajaliidest. Kasutajaliidese realisatsioon rajaneb veebitehnoloogiatele ning on laias laastus kahekihiline – eesrakendus ja tagarakendus (ingl. k. *frontend* ja *backend*). Lõputöö üheks tulemuseks on olemasoleva tagarakendusega ühilduv täiesti uus kasutajaliidese eesrakendus. Esialgses arenduses on tagarakenduse osatähtsus väiksem.

Kasutajaliidese eesrakenduse põhilisteks funktsionaalsusteks on sõiduki olekute kuvamine, erinevate režiimide muutmise võimaldamine, kaamerapiltide näitamine ning vigade kuvamine.

Lõputöö teiseks eesmärgiks on analüüsida, kuidas on võimalik muuta arendatud kasutajaliides universaalsemaks – saavutada ühilduvus teist tüüpi sõidukitega.

Lõputöö on kirjutatud eesti keeles ning sisaldab teksti 30 leheküljel, 9 peatükki, 5 joonist, 0 tabelit.

Abstract

Development of Extensible Vue.js-based User Interface for Robotic Vehicles

The first goal of this thesis is to develop a user interface for a robotic vehicle manufactured by an Estonian company, Milrem Robotics. It has two main layers – frontend and backend. The user interface's frontend is developed mainly using web technologies (including the Vue.js framework). The frontend and backend exchange messages over Websocket using JSON RPC.

One of the results of this thesis is a new frontend for an already-existing user interface backend. During initial development, the focus is more on the frontend, but backend gets some development work by the author too. During this phase of development, the UI's structure (like buttons, telemetry fields and settings) is fixed.

The frontend's main responsibilities include displaying vehicle's statuses, allowing the changing of different modes, displaying camera output(s) (if available) and showing fault messages received from the active vehicle.

The second goal of this thesis is to analyze what would it take to make the user interface more universal. That is, the goal would be for the user interface to be able to connect with different types of robotic vehicles. This is to be achieved via an abstraction of the interfacing with vehicles. That abstraction would be implemented in the backend using Python's class inheritance capabilities.

Because the user interface would get compatible with more vehicles than THeMIS alone, it could no longer be fixed on the frontend side. Now the frontend would query the backend for which buttons to put into the main view, what view modes are available, what settings can be changed, and so on.

The thesis is in Estonian and contains 30 pages of text, 9 chapters, 5 figures, 0 tables.

Lühendite ja mõistete sõnastik

API	*Application Programming Interface*, programmiliides
CSS	*Cascading Style Sheets*, kaskaadlaadistik, kasutatakse HTML lehtede stiilimiseks
eesrakendus	Ingl. k. *frontend*, rakenduse osa, mis käivitatakse veebibrauseris
EMG	*Emergency*, kasutajaliideses kasutatav lühend sõnast eriolukord
GPS	*Global Positioning System*, globaalne positsioneerimise süsteem
GUI	*Graphical User Interface*, graafiline kasutajaliides
HTML	*HyperText Markup Language*, hüpertekst-märgistuskeel
HTTP	*HyperText Transfer Protocol*, veebis kasutatav põhiline rakendustaseme protokoll
I/O	*Input/Output*, sisend/väljund
Javascript	Eesrakendustes kasutatav programmeerimiskeel
JSON	*JavaScript Object Notation*, lihtsustatud andmevahetusvorming
LGPL	*GNU Lesser General Public License*, vabatarkvara litsents
mock	Mingit komponenti/liidestust teesklev programmikood
NPM	Pakihaldur programmeerimiskeelele Javascript
operaatorseade	Seade (enamasti tahvelarvuti), millel käivitatakse kasutajaliidese ees- ja tagarakendus
Python	Üldkasutatav kõrgtaseme programmeerimiskeel
RPC	*Remote Procedure Call*, kaugprotseduurikutse
tagarakendus	Ingl. k. *backend*, rakenduse osa, mis töötab tagataustal
THeMIS	*Tracked Hybrid Modular Infantry System*, ettevõtte Milrem Robotics poolt arendatav robotsõiduk
UI	*User Interface*, kasutajaliides
Websocket	Veebi eesrakenduses kasutatav sokkel

Sisukord

Jooniste loetelu

1 Sissejuhatus

Antud lõputöö eesmärk on arendada robotsõidukite kasutajaliides. Kasutajaliides on loodud ettevõttes Milrem Robotics[1]. Kasutajaliidese eesrakenduse arendamisel kasutatakse veebitehnoloogiaid, milledest põhiline on *Javascripti*[2] raamistik *Vue.js*[3].

Kasutajaliidese eesrakenduse põhilisteks funktsionaalsusteks on sõiduki olekute kuvamine, erinevate režiimide muutmise võimaldamine, kaamerapiltide näitamine ning vigade kuvamine. Eesrakenduse funktsionaalsuste hulka sõiduki liigutamine ei kuulu – sõidukit saab liigutada juhtkangiga ning sellega liidestumise eest vastutab tagarakendus.

Kasutajaliidese tagarakendus vastutab juhtkangiga liidestumise kõrval põhiliselt robotsõidukitega suhtlemise eest. Antud lõputöös kasutatav tagarakendus on juba eelnevalt olnud arenduses. See loodi varasema kasutajaliidese eesrakenduse arendamisega paralleelselt.

Varasem kasutajaliidese eesrakendus oli loodud prototüübiks. Selle arendamisel ei kasutatud sisuliselt ühtegi Javascripti eesrakendus ega ka CSS[4] (*Cascading Style Sheets*) stiili raamistikku. Ühest küljest tagas see optimaalse ressursside kasutuse. Teisalt oli selle algne (ja ka edasine) arendamine aeganõudvam ning programmikoodi haldamine tülikam.

Uus kasutajaliidese eesrakendus on loodud sisuliselt nullist ning kasutab mitmeid teeke, raamistikke ja arendusvahendeid. Nende rakendamisel on potentsiaal muuta arendus ja koodihaldus tõhusamaks, aidates lühendada arendusaega ning vähendada programmivigade hulka. Samas võivad kasutatavad teegid ja raamistikud suurendada eesrakenduse ressursinõudlikkust.

---

[1] https://milremrobotics.com/

[2] https://developer.mozilla.org/en-US/docs/Web/JavaScript

[3] https://vuejs.org/

[4] https://developer.mozilla.org/en-US/docs/Web/CSS

Esialgne arendus toimub spetsiifiliselt Milrem Robotics-i poolt arendatud robotsõiduki THeMIS[1] jaoks. Seejärel analüüsitakse tööd, mida oleks vaja teha, et muuta kasutajaliides universaalsemaks.

Lõputöö dokument toob välja ka kasutajaliidese ja selle arendusega seonduvad tähelepanekud ja järeldused ning edasised arendusvõimalused.

[1] https://milremrobotics.com/themis/

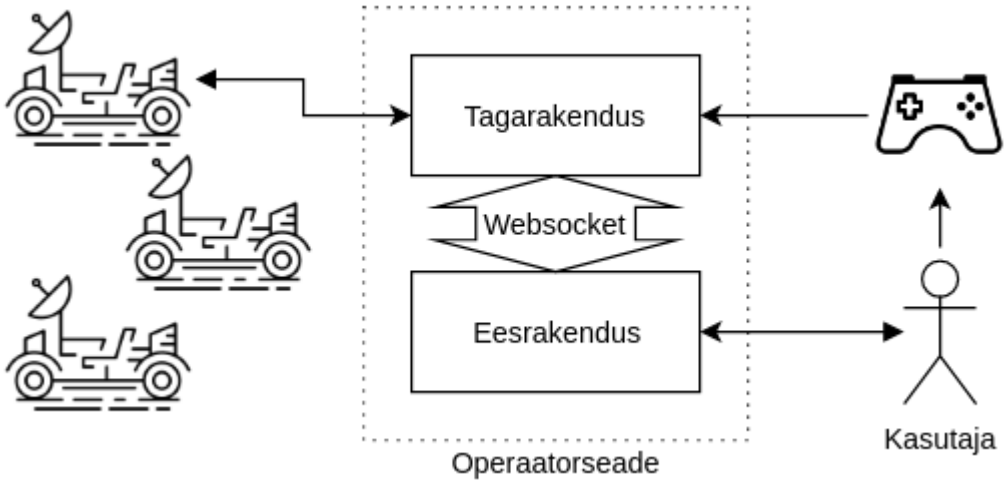
2 Probleemi tutvustus

Antud peatüki eesmärk on anda ülevaade lõputöö poolt lahendatava probleemi olemusest. Esimene alapeatükk kirjeldab üldisemalt keskkonda, kuhu uus kasutajaliidese eesrakendus asetub. Seejärel on käsitletud täpsemalt kasutajaliidese tagarakendust, eesrakendust ja nende suhestumist. Peatükk on kokku võetud argumentatsiooniga uue eesrakenduse vajalikkuse ja aktuaalsuse osas.

2.1 Suurem pilt

Lõputöö esimese arendusetapi põhiskoobiks on uue eesrakenduse arendus. Kasutajaliides ja selle kasutamine hõlmab aga enamat kui eesrakendust. Selle alapeatüki eesmärk on välja tuua, kuidas kasutajaliidese eesrakendus sobitub suuremasse pilti.

Joonis 1 illustreerib kasutajaliidese ja sellega seonduvate komponentide/asjaosaliste vahelisi seoseid.



Joonis 1. Operaatorseade ja sellega seonduvad osad/osalised

2.2 Kasutajaliidese komponendid

Järgnevas on välja toodud kasutajaliidese tagarakenduse ja eesrakenduse olemus, ülesanded ja suhestumine teiste komponentidega. Lisaks on kirjeldatud, mida kujutab endast operaatorseade.

2.2.1 Tagarakendus

Nagu sissejuhatuses sai märgitud, kasutajaliidese tagarakenduse ülesandeks on põhiliselt liidestumine robotsõidukite ja juhtkangiga. Tagarakendus peab arvet robotsõidukite üle, mis on saadaval ühendumiseks. Kasutaja saab eesrakenduse kaudu valida soovitud sõiduki ning algatada sellega ühenduse. Aktiivset sõidukit on võimalik hiljem sama sessiooni jooksul vahetada.

Esiteks võtab tagarakendus aktiivselt robotsõidukilt vastu andmeid ehk telemeetriat. Kättesaadav telemeetria oleneb robotsõiduki tüübist ja selle tarkvarast. See võib sisaldada näiteks sõiduki liikumiskiiruse väärtust, aku taset ja/või GPS-koordinaate (*Global Positioning System*). Samuti peab tagarakendus arvet sõiduki poolt saadetud veateadete üle.

Teiseks saadab tagarakendus aktiivsele robotsõidukile käsklusi. Nende hulka kuuluvad režiimide ja sätete muutmise käsud, nagu näiteks lülitumine erinevate kiirusrežiimide vahel või tulede sisse/välja lülitamine. Mis veel tähtsam, aktiivsele robotsõidukile saadetakse liikumiskäsklusi.

Liikumiskäsklusi ja ka mõnda režiimi muutmise käsku saab sõidukile anda juhtkangi abil. Juhtkangiga liidestumine on samuti üks tagarakenduse ülesannetest. Tagarakendus toetab erinevaid juhtkange ja nende vastendusi, juhindudes konfiguratsioonifailidest. Aktiivse juhtkangi mappingu saab valida eesrakendusest.

Tagarakenduse ja eesrakenduse vaheline liidestumine on realiseeritud üle *Websocket*[1]-i.

[1] https://tools.ietf.org/html/rfc6455

Antud lõputöös rakendatav tagarakendus on juba varasemalt olnud arenduses. See on kirjutatud programmeerimiskeeles *Python*[1].

2.2.2 Eesrakendus

Kasutajaliidese eesrakenduse teostamiseks on valitud veebitehnoloogiad, seega avatakse see veebibrauseris. Et lihtsustada arendust ja testimist, on esialgse arenduse vältel ühilduvuse suhtes pandud rõhku ainult veebibrauseritele *Google Chrome*[2]-le ja *Chromium*[3]-ile.

Kasutajaliidese eesrakendus on mõeldud puuteekraanidel kasutamiseks. Esimese sammuna võimaldab see valida robotsõiduki, mille külge ühenduda. Kui see samm on läbitud, siis kuvatakse sõiduki kohta erinevaid andmeid ja võimaldatakse valida mitmesuguste sõiduki režiimide ning sätete vahel (sõltuvalt aktiivse sõiduki tüübist). Samuti kuvatakse sõidukilt saadud veateateid. Kui sõidukil on kaamerad ning nende pilt on kättesaadav, võimaldatakse näha seda eesrakenduses.

Eesrakenduse poolt antavate käskluste hulka ei kuulu sõiduki liikumiskäsklused. Selle funktsionaalsuse eesrakendusest välja jätmine põhjuse taga on põhiliselt eesmärk saavutada juhtimise ohutus ja töökindlus.

Et saada kuvatavaid andmeid ning anda käsklusi, peab eesrakendus üle *Websocket*-i sõnumivahetust tagarakendusega. Sõnumivahetus toimub *JSON-RPC 2.0*[4] vormis.

Nagu eelnevalt mainitud, on eesrakenduse realiseerimiseks valitud veebitehnoloogiad. Antud valik muudab eesrakenduse väliskeskkonnaga suhestumise küllaltki piiratuks. Sestap osutub tagarakendus vajalikuks. Ilma selleta peaks suhtlus robotiga toimuma otse üle *Websocket*-i või läbi HTTP[5] (*Hypertext Transfer Protocol*) päringute. Viimane ei tundu kuigi mõistlik. Juhtkangidega liidestumiseks oleks võimalik kasutada *Javascripti* API-liidest (programmiliidest). Sellegipoolest on tagarakendusepoolne juhtkangiga

[1] https://www.python.org/

[2] https://www.google.com/chrome/

[3] https://www.chromium.org/

[4] https://www.jsonrpc.org/specification

[5] https://developer.mozilla.org/en-US/docs/Web/HTTP

liidestumine mõistlikum, kuna nii saab sõidukile anda liikumis- ja valitud režiimikäsklusi ka siis, kui veebibrauser peaks olema suletud.

2.2.3 Operaatorseade

Tagarakendus ja eesrakendus on mõeldud käivitatama üheaegselt samal seadmel. Antud lõputöö kontekstis nimetatakse seda seadet operaatorseadmeks. Käesolevas lõputöös arendatava kasutajaliidese kõige tüüpilisemaks operaatorseadmeks on tahvelarvuti.

Kuna operaatori eesrakendus kasutab veebitehnoloogiaid, siis on see teoorias kasutatav paljude erinevate tahvelarvutite ja veebibrauseritega. Aga kuna praegune operaatori tagarakendus on kirjutatud *Pythonis* ja on küllaltki spetsiifiline, siis nõuab see tahvelarvutilt kindlatele nõuete vastavat operatsioonisüsteemi. Arenduse käigus oli kasutatavaks operatsioonisüsteemiks valdavalt *GNU/Linux*, täpsemalt distrod *Debian*[1] 9 ja *Ubuntu*[2] 18.04. Varasemalt oli tehtud arendustööd selle nimel, et tagarakendus jookseks ka *Windowsi*[3] peal. Siiski polnud *Windowsiga* ühilduvus arendamise jooksul prioriteet ning on sestap küllalt vähe testitud.

2.3 Uue eesrakenduse vajalikkus ja aktuaalsus

Varasemalt loodud eesrakendus kasutas küllalt vähest hulka teeke. Javascripti poole peal piirduti teekidega *sass-color-helpers*[4], *lodash*[5] ja *simple-jsonrpc-js*[6] (viimast kahte kasutab ka uus eesrakendus). Eelmise eesrakenduse ehitamiseks kasutati *Webpack*[7] 2-e arendustööriistu. CSS raamistikku ei kasutanud eelmine eesrakendus üldse. Need asjaolud tingisid selle eesrakenduse madala ressursside nõudlikkuse. Aga kuna eelmist eesrakendust realiseeriva *Javascripti* kood oli suuresti nullist ja ilma raamistikuta

[1] https://www.debian.org/

[2] https://www.ubuntu.com/

[3] https://www.microsoft.com/en-us/windows

[4] https://github.com/voxpelli/sass-color-helpers

[5] https://lodash.com/

[6] https://github.com/jershell/simple-jsonrpc-js

[7] https://webpack.js.org/

kirjutatud, siis võib selle edasiarendus ja haldamine osutuda tülikaks ja aeganõudevaks. Lõputöö autori hinnangul ei skaleeruks eelmine eesrakendus piisavalt suurel määral.

Uus eesrakendus erineb eelmisest põhiliselt just selle poolest, et selles kasutatakse nii Javascripti kui ka CSS raamistikku. Javascripti raamistikuks on valitud *Vue.js* ning CSS raamistikuks *Semantic UI*[1]. Tänu raamistike kasutuselevõtule väheneb potentsiaalselt koodi hulk, mida on vaja kasutajaliidese realiseerimiseks. Lisaks kasutab uus eesrakendus veel mõnda teeki (täpsemalt jaotises 3.3.4). Märkimisväärne osa funktsionaalsusest ja kujundusest, mis eelmise eesrakenduse puhul tuli ise realiseerida, on nüüd raamistike ja teekide abil juba eelnevalt olemas, dokumenteeritud, testitud ning pidevas täiustamises.

Raamistike ja teekide kasutamine võib küll nõuda arendajatelt sisseelamisaega, kuid lõputöö autor leiab enda kogemusele põhinedes, et lisa ajakulu on seda väärt. Nii *Vue.js-i* kui ka *Semantic UI* kasutajaskond on küllaltki suur. Tänu Internetist leitavate abimaterjalide suurele hulgale on sisse elamine kergem.

[1] https://semantic-ui.com/

3 Eeltöö

Selle peatüki eesmärk on tuua välja sammud ja eeltöö, mis teostati enne arenduskeskkonna üles seadmist ja tegelikku arendust.

3.1 Projektihaldus ja seonduvad keskkonnad

Projektihaldus toimus *Atlassian Jira*[1]-s. Kuna tagarakenduse ja eelmise eesrakenduse arendus kajastus ühe ja sama Jira projekti all, siis võeti vastu otsus ka uue eesrakenduse arendus seal samas hallata.

Sarnane olukord oli dokumentatsiooniga, mida hallati *Atlassian Confluence*[2]-s, ning versioonihaldusega *Git*[3]-i repositooriumis (sellest täpsemalt peatükis 5.1).

3.2 Nõuete defineerimine

Enne arendust töötas ettevõtte tarkvarameeskoond koos selle lõputöö autoriga välja esialgsed nõuded kasutajaliidese uuele eesrakendusele. Väiksemal määral uuendati/lisati nõudeid ka varasemalt olemas olnud tagarakendusele.

Kuna tegemist polnud täiesti esimese eesrakenduse generatsiooniga ning tagarakendusel oli enne uue arendust suurem osa vaja minevast funktsionaalsusest olemas, siis ei olnud esialgsete nõuete paika panemine kuigi keeruline. Vana eesrakendus oli juba saanud reaalset rakendamist ning sellest saadud tagasiside oli suureks abiks.

[1] https://www.atlassian.com/software/jira

[2] https://www.atlassian.com/software/confluence

[3] https://git-scm.com/

Nõuded said dokumenteeritud ettevõtte *Confluence*-is.. Lõplikud nõuded on toodud peatükis 4.

3.3 Taustauuring ja valik tehnoloogiate osas

Selles alapeatükis on kirjeldatud ja põhjendatud mitmed kasutatavaid tehnoloogiaid puudatavad otsused.

3.3.1 Rakenduse platvorm

Rakenduse platvormi all on siinkohal silmas peetud keskkonda, milles kasutajaliides on mõeldud käivituma. Aruteludel oli kaalutud peamiselt veebiplatvormi ja arendamist töölauarakendusena (GUI, *Graphical User Interface*). Mobiilirakendustes (*Android*[1] ja/või *iOS*[2]) ei nähtud antud lõputöö kasutusjuhule kuigi suurt kasu.

Veebiplatvorm oli saanud end eelmise eesrakenduse arendamise vältel ja hiljem testimise käigus tõestada. Veebiplatvorm on osalt tänu oma populaarsusele küllaltki laialdaselt rakendatav. Paljudel juhtudel võib HTML[3]-i (*HyperText Markup Language*), CSS-i ja Javascript-i rakendamine arendusprotsessis osutuda tõhusamaks, kui töölauarakenduste kirjutamine [1]. Välja tulid ka omapärad, milledest peamine on vajadus eraldi eesrakenduse ja tagarakenduse kihtide järele, mis peavad teineteisega mingil moel suhtlema (antud lõputöö puhul tehakse seda üle *Websocket*-i). See ei ole tingimata puudus. Ent töölauarakenduse korral on kihtideks jaotamise osas suurem vabadus.

Töölauaplatvormi puhul sai uuritud erinevaid GUI teeke. GUI teegil põhinev kasutajaliides võib võtta potentsiaalselt vähem ressursse. Üheks prioriteediks oli ühilduvus Python-iga, et oleks vähem lisatööd tagarakendusega integreerimisel. Teiseks prioriteediks oli kasutatavus GNU/Linux-il, kuna tagarakendusel on Linuxispetsiifilisi mooduleid. Sõelale jäid GTK[4] ja Qt[5]. GTK litsents (LGPLv2.1+[6], *GNU Lesser General*

[1] https://www.android.com/

[2] https://developer.apple.com/ios/

[3] https://developer.mozilla.org/en-US/docs/Web/HTML

[4] https://www.gtk.org/

[5] https://www.qt.io/

[6] https://www.gnu.org/licenses/old-licenses/lgpl-2.1.html

Public License) lubab kommertsiaalset kasutust ilma lisatasu nõudeta. Aga Qt paraku seab kommertsiaalsele kasutusele küllaltki ranged piirangud [2].

Lõppkokkuvõttes otsustas lõputöö autor koos kolleegidega veebiplatvormi kasuks. Seda esiteks varasema kogemuse põhjal ning teiseks potentsiaalselt tõhusama arenduse nimel. Mobiilirakenduse loomine oleks võtnud liiga suure ressursi – lõputöö autori kolleegide seas polnud arendajat, kes oleks selle platvormiga piisaval määral kokku puutunud.

3.3.2 Ehitamisvahend

Arendusvahend, millega eesrakenduse pakki ehitama hakatakse, jäi samaks, mida kasutatati eelmise eesrakenduse puhul – *Webpack*, seekord versioon 4. Selle sama vahendi juurde jäämine võimaldas lõputöö autoril saada paremat abi kolleegidelt.

3.3.3 Javascript-i raamistik

Javascript-i maailmas on raamistike turg peamiselt kolme raamistiku käes - *Angular*[1], *React*[2] ja *Vue.js*. *Angulari* arendab ja toetab *Google*[3], *React*-i *Facebook*[4]. *Vue.js* on siinkohal erand – sellel pole lõputöö kirjutamise hetkel ühegi suurkorporatsiooni otsest tuge taga.

Nende kolme seast sai otsustatud *Vue.js*-i kasuks. Üheks mõjuvamaks argumendiks oli selle kerge omandatavus (võrreldes ülejäänud kahega). Lõputöö autori subjektiivsel kogemusel on Vue.js-i näol tegemist lihtsamini kasutatava raamistikuga kui React. "React ei ole täielik raamistik ning keerulisemad võimalused nõuavad kolmandate osapoolte teeke." Angulari kohta on öeldud, et "selle õppimiskõver on järsk". Samas "Vue pakub kõrgemat kohandatavust ja on seeläbi lihtsamini õpitav kui Angular või React. [...] Siiski, Vue lihtsus ja paindlikkus on kui kaheteraline mõõk - see võimaldab kehva koodi, mida on raske siluda ja testida" [3]. Viimases tsitaadis toodud *Vue.js*-iga seonduv oht on lõputöö autori arvates välditav.

---

[1] https://angular.io/

[2] https://reactjs.org/

[3] https://www.google.com/

[4] https://www.facebook.com/

Lõputöös kasutatakse ka *Vue.js*-iga ühilduvaid ametlikke lisasid *Vue Router*[1] ja *Vuex*[2]. *Vue Router* võimaldab erinevate vaadete vahel liikumist. *Vuex* vastutab keskse (komponentide-ülese) andmete salvestamise eesrakenduses.

3.3.4 CSS raamistik

CSS raamistikud pakuvad eeldefineeritud stiilimist veebilehe/-rakenduse HTML-ile. Nagu enamike raamistike puhul, nõuavad need esialgu dokumentatsiooniga tutvumist ning mõnedel juhtudel ka eelseadistamist. Aga vastukaaluks võivad need lõppkokkuvõttes vähendada disainimisele kuluvat arendusaega.

CSS raamistiku valimine osutus keerulisemaks kui *Javascript*-i raamistiku valimine. Peamiselt selle tõttu, et CSS raamistikud kipuvad olema kindlama fookusega. Enamik neist on mõeldud tavapärasemate veebilehtede loomiseks. Kasutajaliidese eesrakenduse juures on oluline, et raamistik oleks paindlik. Vastasel juhul võib raamistik antud lõputöö kasutusjuhule vastu töötada.

Uuritud sai mitmeid raamistikke. Nende seas oli mitmeid spetsiaalselt kasutajaliideste loomiseks mõeldud lahendusi. Ent nende probleemiks oli vähene kasutajaskond ja/või väike arendusaktiivsus. Lõpuks jäid valikusse kaks – *Bootstrap*[3] ja *Semantic UI*.

Otsustatud sai *Semantic UI* kasuks, sest see tundus lõputöö kasutusjuhtu arvestades sobivam ning võimekam. *Bootstrap*-ist jäi mulje, et see järgib suuremal määral kindlat stiili, mis ei pruugi selles lõputöös arendatava kasutajaliidese jaoks sobida.

3.3.5 Ülejäänud pakid

Semantic UI interaktiivse funktsionaalsuse toimimiseks on eesrakenduse poolt kasutatud pakkide seas *jQuery*[4]. *Semantic UI*-st lahus kasutab eesrakendus *jQuery*-t küllaltki vähe.

Et hõlbustada andmestruktuuride peal tehtavaid operatsioone, kasutatakse eesrakenduses Javascripti teeki *lodash*.

---

[1] https://router.vuejs.org/

[2] https://vuex.vuejs.org/

[3] https://getbootstrap.com/

[4] https://jquery.com/

Eesrakenduse suhtlus tagarakendusega toimub üle *Websocket*-i teostavate kaugprotseduurikutsete (RPC) kaudu. Et seda hõlbustada, kasutab eesrakendus teeki *simple-jsonrpc-js*. Eesrakenduse komponendid saavad kuulata tagarakenduse poolt saadetavat infot. Selle hõlbustamiseks on kasutusel lõputöö autori poolt modifitseeritud teek *PubSubJS*[1].

Lisaks eelmainitule on kasutusel mitmed *Webpack*-iga kasutamiseks mõeldud pakid ja teegid, mida pole mõtet eraldi välja tuua.

[1] https://github.com/Teekivi/PubSubJS

4 Lõplikud nõuded

Alljärgnevalt on toodud funktsionaalsed ja mittefunktsionaalsed nõuded kasutajaliidese eesrakendusele ja üldisemalt. Nende kujunemine on toimunud nii eeltöö, arendamise kui ka testimise faasis. Siinkohal on toodud lõplik nõuete kogum, millele kasutajaliides peab vastama (või mõnede juures võiks vastata). Viimaks on kirjeldatud nõuded arendusprotsessile.

4.1 Funktsionaalsed nõuded

Funktsionaalsete nõuete kohta on kirjutatud, et "[need] vastavad küsimusele "Mida peab tarkvara tegema?" (näiteks, süsteem peab võimaldama kauba tellimist)" [4]. See jaotis loetleb kasutajaliidest puudutavad olulisemad funktsionaalsed nõuded.

Kasutajaliides:

1. Peab võimaldama näha loendit samasse võrku ühendatud sõidukitest.
2. Peab võimaldama valida sõiduki, mille külge ühenduda ning võimaldama hiljem seda vahetada.
3. Peab kuvama kaamera(te) pilti, juhul sõidukile on see/need paigaldatud.
4. Peab kuvama sõiduki olekuga seotud parameetreid.
5. Peab võimaldama näha ja muuta sõiduki mitmesuguseid sätteid.
6. Peab erinevate tulede olemasolul võimaldab nende konfigureerimist.
7. Peab erinevate I/O (*input/output* ehk sisend/väljund) väljaviikude olemasolul võimaldama nende sisse ja välja lülitamist.
8. Peab võimaldama valida erinevate juhtkangi vastenduste vahel.
9. Peab kuvama sõiduki poolt saadetud veateateid (vastavalt tõsidusele dialoogiakna või sõnumiribaga).
10. Peab koguma sõiduki poolt saadetud veateated mällu ning võimaldab näha nende loendit ning iga veateate kohta täpsustavaid andmeid.
11. Peab pakkuma võimalust tagarakenduse poolt kogutud logisid logiserverisse üles laadida.

12. Peab võimaldama vahetada kahe erineva disainilahenduse vahel - tavaline ja öörežiim.
13. Peab võimaldama muuta latentsusaja liimiti, mida ületades keelatakse sõiduki liigutamine.
14. Peab võimaldama kasutajaliidese animatsioonid keelata või lubada.
15. Peab võimaldama laiendamist lisavaadete näol.

4.2 Mittefunktsionaalsed nõuded

Kasutajaliidese olulisemad mittefunktsionaalsed nõuded on toodud järgnevalt.

1. Kasutajaliides peab jooksma modernsel tahvelarvutil sujuvalt. Kasutajaliides ei tohi jätta aeglast muljet, eriti kui animatsioonid on keelatud.
2. Kasutajaliidese poolt kuvatav sõidukiühenduse latentsusaeg peab kasutamisel olema 90 protsendil ajast alla 30 ms-i, välja arvatud juhul, kui kõrge latentsusaja põhjustab kehv ühendus sõidukiga või sõiduki (riistvara ja/või tarkvara) ebaoptimaalsus.
3. Automaattestid. Testide katvus võiks olla vähemalt 70%.
4. Kasutajaliides peab olema intuitiivne. Seda kasutama õppimine ei tohi olla liialt keeruline.

4.3 Arendusprotsessi nõuded

- Tarkvara koodi versioneerimine toimub *Git*-i repositooriumis.
- Arendusülesandeid hallatakse *Atlassian Jira* keskkonnas.
- Arendust toetavad regulaarsed koosolekud.
- Koosolekud ja nõuded dokumenteeritakse *Confluence*-is.
- Kasutajaliides läbib funktsionaaltestid sõidukitega.
- Kasutajaliidese arendusprotsessis arvestatakse testijate ja kasutajate tagasisidega.
- Keerulisemate muudatuste/täienduste puhul peab lähtekood saama läbivaatusel heakskiidu.

5 Arendus robotsõiduki THeMIS jaoks

THeMIS on ettevõtte Milrem Robotics poolt arendatav robotsõiduk (vt Joonis 2). Esmane kasutajaliidese arendus toimus ainult THeMIS-t silmas pidades. Selles arendusfaasis eeldas eesrakendus kasutajaliidese tagarakenduselt kindlat funktsionaalsust. Näiteks uue kuvatava telemeetriavälja lisamiseks tuli vastavad koodimuudatused viia läbi nii tagarakenduses kui ka eesrakenduses.

Joonis 2. Foto robotsõidukist THeMIS

Järgnevalt on välja toodud põhilised etapid robotsõiduki THeMIS spetsiifilise kasutajaliidese arenduses.

5.1 Esimesed sammud

Kasutajaliidese arenduses kasutati koodi versioneerimiseks *Git*-i koos *git-flow-avh*[1] branchimise metoodikaga. Uue eesrakenduse arenduse baas-branchiks oli esialgu develop-uing (*uing* tähendab siin *user interface next generation*).

Uue kasutajaliidese eesrakenduse failide jaoks loodi projekti juurkausta uus kaust *mildef-uing*, mis eksisteeris paralleelselt eelmise eesrakenduse kaustaga *mildef-ui*. Autor lõi

[1] https://github.com/petervanderdoes/gitflow-avh

esmalt esialgse konfiguratsiooni NPM-i, *Semantic-UI* ning *Webpack*-i jaoks. Kulus ligikaudu nädal arendusaega, et saada *Webpack* ja *Semantic-UI* sobival viisil tööle.

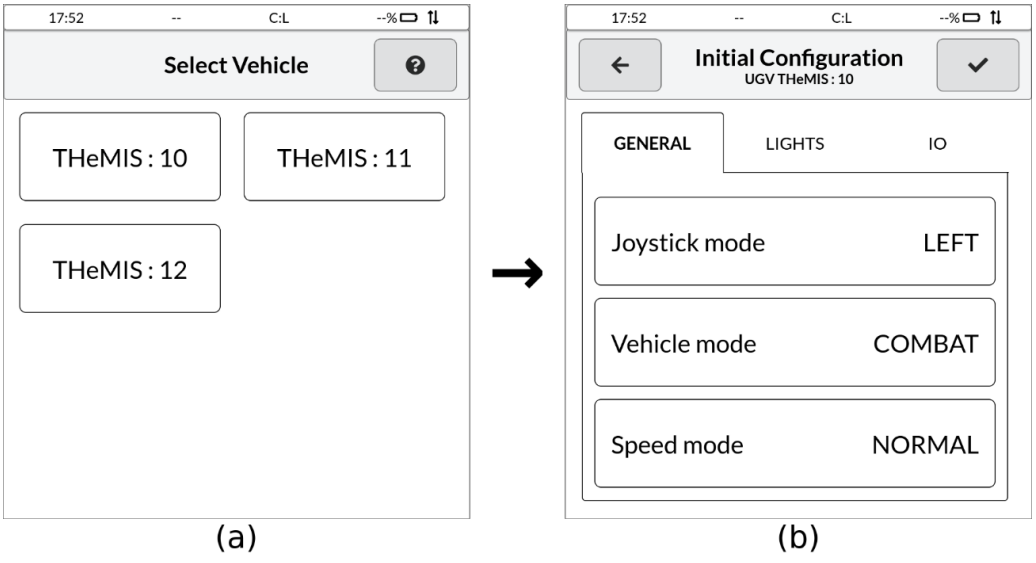
Enne vaadete juurde asumist lõi lõputöö autor kõigepealt komponendi *App*, mille sees kuvatakse erinevaid vaateid. Komponendi *App* sees paikneb globaalne *StatusBar* komponent, mida kuvatakse ekraani ülaosas, ning mis kuvab kellaaega, aktiivse sõiduki nime, juhtkangi mappingut ning võrguühenduse staatust. Enamik vaateid kasutavad üldist päiseriba komponenti *HeaderBar*, millel on nupp eelmisesse vaatesse tagasi minemiseks ja vaate pealkiri. Vaade saab seda komponenti ka laiendada, lisades paremale poole nupu.

Veel enne esimese vaate loomist seadis lõputöö autor üles *Vue.js*-iga koos käiva teegi *Vue Router*. See teek hõlbustab erinevate vaadete vahel liikumist. Vaated on programmikoodis realiseeritud *Vue.js* komponentidena.

Lõputöö autor võttis uue eesrakenduse arendamisel eelmise eesrakenduse *Websocket RPC*-ga liidestumise programmikoodi aluseks. See aitas arendusaega kokku hoida.

5.2 Sõiduki valimise realiseerimine

Esimesena lõi lõputöö autor sõiduki valimise vaate (*VehicleSelectionView*, vt Joonis 3 (a)). Seda vaadet näeb kasutaja, kui ta pole veel ühegi sõidukiga ühendust algatanud. Sõidukite olemasolu korral näidatakse siin iga ühe kohta nuppu. Kättesaadavate sõidukite nimekirja päritakse tagarakenduselt iga sekundi tagant.

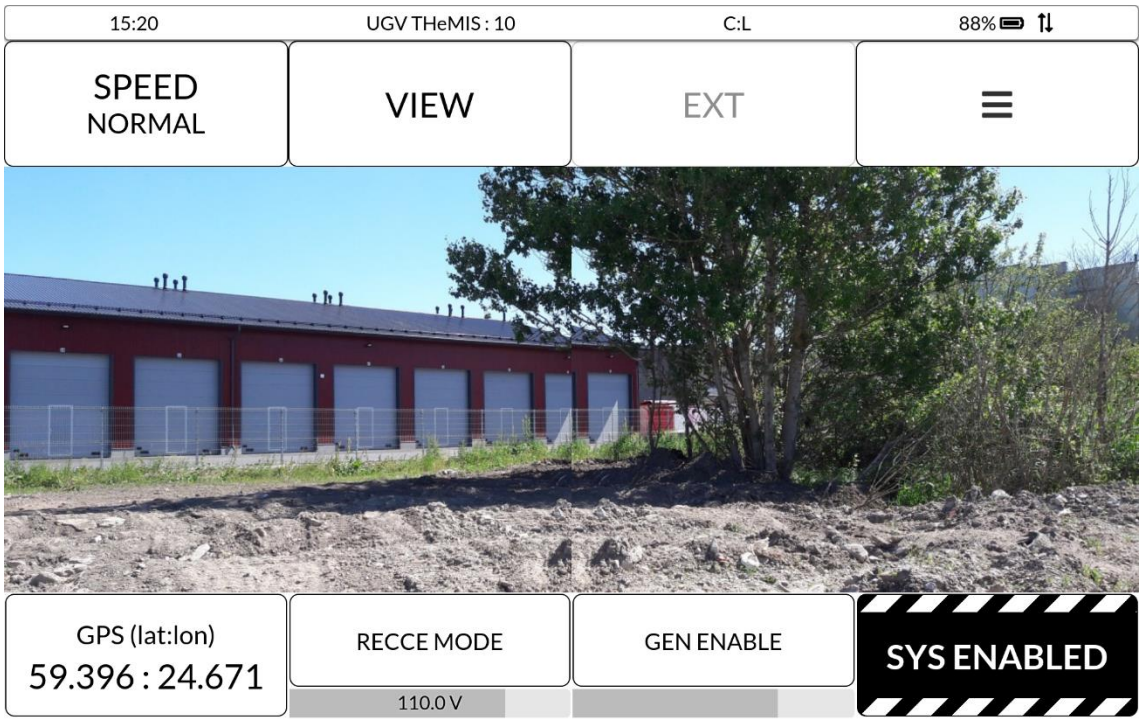


Joonis 3. Kasutajaliidese eesrakenduse vaated: (a) VehicleSelectionView, (b) InitialConfigurationView

Kui kasutaja valib siin vaates sõiduki, siis avaneb esialgse konfiguratsiooni vaade (*InitialConfigurationView*, vt Joonis 3 (b)). Kasutajaliidese tagarakendus jätab mitmed sätted sõidukipõhiselt meelde ning võimaldab selles vaates neid näha ja soovi korral muuta. Vastava tagarakenduse funktsionaalsuse arendas lõputöö autori kolleeg. Esialgse konfiguratsiooni vaatest on võimalik sõiduki valimise vaatesse tagasi minna või otsustada ühenduda sõidukiga. Sõidukiga ühendumisel avaneb põhivaade.

5.3 Põhivaate loomine

Põhivaate jaoks lõi lõputöö autor komponendi *MainView*. Põhivaade koosneb kolmest osast – ülemine nuppude riba, sisuala ja alumine nuppude riba. Joonis 4 kujutab põhivaadet, kui aktiivne on kaamerapilti kuvav režiim.



Joonis 4. Põhivaade kaamerapilti kuvavas režiimis

Ülemine nuppude riba on realiseeritud komponendis *MainViewHeader*. Sellel on nupp robotsõiduki THeMIS kiirusrežiimi muutmiseks. Ülejäänud kolm nuppu puudutavad kasutajaliidest – esimene nendest võimaldab valida sisualal kuvatavat, teine on reserveeritud lisafunktsionaalsuse ja/või robotsõiduki pealisehitise jaoks, ning viimaks on sellel nupp sätete vaate avamiseks.

Sisualal näidatakse vastavalt kasutaja soovile (põhivaate režiimile) kas kaamerapilti (kui see on saadaval; komponent *CameraModeContent*), täielikku või ainult temperatuuride

telemeetriat (komponent *DataContent*) või lihtsalt logo (komponent *LogoModeContent*). Täielikku sõiduki telemeetriat kuvatakse sakkidesse organiseeritult. Temperatuuride telemeetria kujutab endast ühe saki (*"Temps"*) sisu. *DataContent* komponent on andmetele orienteeritud – selle HTML ehitub telemeetria välju kirjeldava andmestruktuuri põhjal, kasutades põhiliselt Vue.js-i itereerimise võimekust. Põhivaate režiime on täpsemalt käsitletud Lisas 1.

Vaate allosas paiknev riba (komponent *MainViewFooter*) sisaldab esiteks nuppu sõiduki GPS koordinaatide väärtustega. Sellel vajutamine vahetab kuvamisühikuid – MGRS või pikkus- ja laiuskraad. Veel on sellel ribal nupp sõiduki režiimi muutmiseks (selle all olev riba näitab kõrgepinge väärtust), nupp generaatori sisse või välja lülitamiseks (selle all olev riba näitab kütuse taset) ning nupp kõrgepinge sisse või välja lülitamiseks.

5.4 Parendused eesrakenduse ja tagarakenduse vahelise suhtluse osas

Põhivaates ja teistes vaadetes kuulatakse üle *Websocket*-i tagarakenduse poolt saadetavaid vaja minevaid väärtusi. Lõputöö autor on näinud vaeva, et programmikood registreeriks end kuulama ainult neid väärtusi, mida läheb aktiivselt vaja.

Nagu eelnevalt välja toodud, toimub eesrakenduse ja tagarakenduse vaheline suhtlus üle *Websocket JSON RPC*. Kuulamaks tagarakenduse poolt algatatud sõnumeid, saavad erinevad eesrakenduse komponendid n.ö. "tellida" erinevate sõnumite teemasid. Et ressursikasutus (nii protsessori kui ka suhtluskanali osas) oleks efektiivsem, tegi lõputöö autor tihti saadetavad sõnumid väljade järgi granulaarseks. See tähendab näiteks telemeetria puhul iga välja eraldi tellimise võimaldamist. Tänu sellele ei saadeta tagarakendusest väljade väärtusi, mida antud hetkel ei näidata ega muul moel kasutata. Sarnasel moel sai granulaarsuse ka juhtkangi sisendi teema – varem saatis tagarakendus juhtkangi iga olekumuutuse peale eesrakendusele sõnumeid, isegi kui enamik nendest olumuutustest ei olnud eesrakendusele olulised. Eeltoodud muudatused puudutasid nii eesrakendust kui ka tagarakendust.

5.5 Sätete vaate loomine

Sõiduki ja kasutajaliidese erinevaid sätteid võimaldab näha ja muuta vaade *SettingsView*. Selles vaates on sätted jaotatud kolme sakki – "*General*" (üldised sõidukit puudutavad

sätted), "*Modes*" (sõiduki erinevaid režiime puudutavad sätted) ning "*Help*" (abistav info, versioonid, mõned kasutajaliidest puudutavad sätted ja toimingud.)

Sätete vaatest on võimalik minna täielikku telemeetriat kuvavasse vaatesse. See vaade on kasulik, kui põhivaate sisurežiimiks on kaamerapildi kuvamine. Samuti saab sätete vaates vahetada aktiivset sõidukit, muuta sõidukil olevate tulede ja erinevate väljundite sisselülitatust.

"*Help*" saki alt on leitav võimalus muuta sõidukiühenduse latentsusaja piirangut. See piirang keelab ohutuse tagamiseks sõidukile käskude andmise, kui ühenduse latentsuaeg ületab teatud väärtuse. Sama saki alt saab lülitada eesrakenduse animatsioonid sisse või välja, mis tuleb erineva jõudlusega operaatorseadmete puhul kasuks. Viimaks väärib välja toomist nupp "*Upload Logs*", mille valimisel avaneb vaade kasutajaliidese tagarakenduse poolt kogutud sõiduki- ja operaatori logide üles laadimiseks.

Nagu *DataContent*, on ka sätete vaate komponendi HTMLi konstrueerimine suuresti andmetele orienteeritud. SettingsView-i üheks andmeatribuudiks on järjend objektidest, millest igaüks vastab kindlale sakile. Sakile vastav objekt sisaldab saki pealkirja ning järjendit erinevatest sisuelementidest. Iga saki sisu kuvamiseks kasutatakse abikomponenti *SettingsPanel*. Selle abikomponendi HTMLis käiakse saki andmeobjekt *Vue.js*-iga läbi, mille läbi ehitubki sätete vaate sisu.

Sätete vaate "*Help*" saki sisule pääseb ligi ka esialgsest sõiduki valimise vaatest. Tänu sellele on võimalik näha abiinfot, versioone ja muuta kasutajaliidest puudutavaid sätteid ilma, et ollakse sõidukiga juba ühenduse algatanud.

Lisas 2 on toodud kuvatõmmised sätete vaatest.

5.6 Erinevate teavituste kuvamine

Robotsõiduki THeMIS jaoks on defineeritud teatud vead, mis sellel sõidukil võivad esineda. Kasutajaliidese eesrakendus näitab vähem tõsiseid vigu sõnumiriba abil. Kriitilisemate vigade puhul kasutatakse dialoogiakent, mille jaoks lõi lõputöö autor komponendi *FaultModal*. See komponent kuvab vea koodi, sõnumi ning soovitatava kasutajapoolse tegevuse. *FaultModal*-il on nupp, mis viib veale vastavate detailide vaatesse, ning nupp dialoogiakna sulgemiseks. Kui ootel on veel vigu, siis lisandub

dialoogiakna allossa riba, mis näitab ootel olevate vigade arvu ning mis võimaldab minna vigade loendi vaatesse või ignoreerida ülejäänud vigu.

Kasutajaliides hoiab puhvris viimast 50-t viga. Loendile neist saab ligi sätete alt. Loendi jaoks lõi lõputöö autor vaate *FaultListView*. Vigade loendi vaade viitab omakorda vea detailvaatele *FaultDetailView*.

THeMIS-e puhul on olekuid, mida tuleb näidata iga vaate üleselt ja püsivalt. Üks nendest on niiöelda *EMG error* (ingliskeelsest sõnast *emergency*). Teine nendest on ohulüliti aktiivsus. Selliste olukordade puhul kuvab eesrakendus globaalset raami vastava tekstiga. Kasutajaliides võimaldab *EMG error*-it ignoreerida, kui kasutaja lubab *EMG mode* režiimi. Ohulülitit ignoreerida ei saa.

Kasutajaliides kuvab dialoogiakna komponendi *InfoModal* abil teate, kui aktiivne sõiduk ei ole enam kättesaadav. Sellisel juhul saab aktiivseks esialgne sõiduki valimise vaade.

Kui sõidukiga ühenduse latentsusaeg ületab sellele seatud piirangut, siis hakkab olekuribal (*StatusBar*) ühenduse staatuse ikoon vaheldumisi vilkuma tavapärase ikooni ja hüüumärgiga ikooni vahel.

Kui operaatorseadmega on ühendatud või ühendatakse mitteühilduv juhtkang, siis kuvatakse eesrakenduses vastav teade. Samuti kuvatakse teade, kui juhtkang ühendatatakse operaatorseadmest lahti või kui kasutajaliides käivitatakse nii, et juhtkang pole järele ühendatud. Nende teadete kuvamiseks kasutatakse komponenti *InfoModal*.

Lisas 3 on toodud kuvatõmmised selles jaotises mainitud teavitustest.

5.7 Dialoogiakende komponendid

Eelnevas tekstis on mainitud erinevaid dialoogiaknaid. Eri tüüpi dialoogiakende jaoks on loodud eraldi Vue.js komponendid. Nendeks on:

- *FaultModal* – kuvab veateate, mis sisaldab vea koodi, esinemise kellaaega, kirjeldust ja soovitatud kasutajapoolset tegevust. Võimaldab minna vea detailvaatesse, vigade ootel olemise korral vigade loendivaatesse.
- *InfoModal* – dialoogiaken lihtsal kujul info kuvamiseks. Omab pealkirja ja tekstilist sisu.

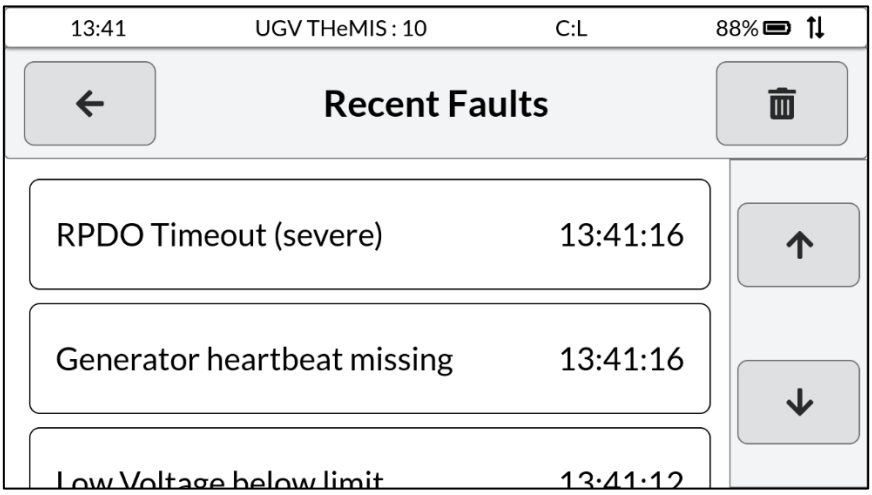
- *OptionModal* – dialoogiaken, mille abil saab kuvada kasutajale mitut valikut.
- *JoystickModeModal* – *OptionModal*-it kasutav dialoogiaken, mis kuvab aktiivse juhtkangi *mapping*-ud ning võimaldab nende vahel valida.

Eelkirjeldatud dialoogiakna komponendid võtavad aluseks komponendi *BaseModal* (*JoystickModeModal* küll kaudselt). *BaseModal* kasutab Semantic UI dialoogiakende moodulit[1]. Lisaks vastutab see komponent dialoogiakende ootejärjendi eest. Ootejärjendisse lähevad dialoogiaknad siis, kui üks dialoogiaken on juba aktiivne ja tahetakse kuvada uut dialoogiakent.

5.8 Ekraaninuppude abil kerimine

Antud kasutajaliides on mõeldud olema kasutatav operaatorseadmetel, mille ekraani puutetundlikus ei pruugi olla piisavalt hea mugavaks libistamiseks (ingl. k. *swipe*). Samuti võib esineda olukordi, kus isegi hea puutetundlikkusega operaatorseadme korral oleks kindlam kerida kuvatavat sisu nuppude abil.

Sellise olukorra lahenduseks lõi lõputöö autor komponendi *ButtonScrollableContainer*. Implementatsiooni poole pealt kujutab komponent endast pakendit (ingl. k. *wrapper*), mis sisaldab kasutavate komponentide poolt määratud keritavat sisu ning selle kõrval (või all) paiknevat üles ja alla kerimise nuppude riba. Kerimisnuppude riba on sõltuvalt tahvelarvuti ekraani orientatsioonist kas vertikaalne või horisontaalne.



Joonis 5. Komponendi *ButtonScrollableContainer* kasutatuna vaates *FaultListView*

[1] https://semantic-ui.com/modules/modal.html

5.9 Automaattestide hõlbustamine

Loodud koodi arendusarvutis testimiseks oli esialgu sisuliselt ainus võimalus kõigepealt arvutis käivitada THeMIS-e tarkvara ning kasutajaliidese tagarakendus. Seejärel sai ühenduda kasutajaliidese eesrakenduse kaudu THeMIS-e tarkvaraga.

Sellisel lähenemisel olid aga omad puudused. Esiteks THeMIS-e tarkvara eeldas juurdepääsu mitmesugustele riistvarakomponentidele, mis olid päris sõiduki peal olemas, aga millele polnud tarkvaralist mock-i. THeMIS-e tarkvara küll käivitus, kuid osa funktsionaalsust (näiteks realistlike telemeetriasõnumite saatmine) oli puudu. Teiseks oleks sellise käivitamisprotsessi läbimine automaatsete integratsioonitestide vältel pigem keeruline.

Et lihtsustada eesrakenduse automaatteste, kirjutas lõputöö autor *Websocket RPC mock*-i. Tänu sellisele mock-ile sai eesrakendus töötada ilma, et tagarakendus oleks käivitatud. Antud mock hõlmas sisuliselt kogu *RPC* funktsionaalsust, mis oli tarvis kasutajaliidese eesrakenduse töötamiseks. See simuleeris kolme sõiduki olemasolu, võimaldas nende vahel valida, saata neile käsklusi ning andis teatud (enamasti realistlikus) vahemikus juhuslikke arvuliste telemeetria väljade väärtusi.

Tagarakendusega suhtleva ja *mock*-itud *Websocket RPC* vahel saab valida *Webpack*-i arendusserveri käivitamisel. Kui keskkonnamuutuja MOCK_RPC väärtus on mittetühi, siis kasutatakse mock-itud *RPC*-d, vastasel juhul tegelikku *RPC* implementatsiooni.

Algses variandis võimaldas mock simuleerida erinevaid olukordi (näiteks vigade esinemisi) läbi Javascript-i liidese, näiteks brauseri arenduskonsooli abil. Hiljem lisas lõputöö autor mock-ile reageerimise mitmetele klaviatuuri klahvidele:

- Klahv A simuleerib ühte sõnumiribana kuvatavat viga.
- Klahvid S, D ja F simuleerivad erinevaid dialoogiaknaga kuvatavaid vigu.
- Klahv G simuleerib tundmatut viga, mis kuvatakse sõnumiribana.
- Klahv E simuleerib *EMG error*-it.
- Klahv J simuleerib juhtkangi eemaldamist/taasühendamist.
- Klahv K simuleerib sõidukiga ühendumise kadumist.
- Klahv L simuleerib kõrget sõidukiga ühenduse latentsusaega.

Selline funktsionaalsus tuleb kasuks lisaks automaattestidele ka kasutajaliidese demonstreerimise juures. *Mock*-itud kasutajaliidest on võimalik ehitada ning paigaldada operaatorseadme (enamasti tahvelarvuti) peale. Operaatorseadmele saab ühendada järele klaviatuuri ning selle abil demonstreerida kasutajaliidese testijatele, klientidele ja teistele huvilistele eesrakenduse olukordi, mida ei saa puuteekraanilt esile kutsuda.

5.10 Öörežiimi implementeerimine

Esialgses arenduses oli kasutajaliidese eesrakenduse komponentidel suuresti must-valge värviskeem. Öises ja muidu pimedamas keskkonnas on otstarbekam rakendada tumedamat värvilahendust. Tumedama värvilahenduse (nö. öörežiimi) realiseerimiseks tuli lõputöö autoril modifitseerida Semantic UI poolt pakutud värvilahendust. Värvilahenduse muutmine ning samas tava- ja öörežiimi vahel lülitumise võimaldamine osutus tehnilise poole pealt küllaltki kohmakaks. Et mitte kulutada ülemäära palju aega öörežiimi realiseerimisele, otsustas lõputöö autor mitmed Semantic UI värvide väärtused sisuliselt üle kirjutada (ingl.k. *override*) CSS attribuutidega / muutujatega[1]. Tava- ja öörežiimi vahel saab lülitada sätete vaate *HeaderBar*-i parempoolse nupuga. Mõned kuvatõmmised öörežiimist on toodud Lisas 4.

5.11 Eesrakenduse elementide paigutamise parendamine

Algselt kasutas kasutajaliidese eesrakendus põhiliste HTML elementide (nagu *StatusBar*, *HeaderBar* ja vaate sisu) paigutamiseks CSS-i attribuuti *position: fixed* või *position: absolute* koos eelnevalt arvutatud CSS-i positsioneerimise (*top, right, bottom, left)* ja/või *margin*, *padding* väärtustega. Selline lähenemisviis osutus küllaltki tülikaks, eriti kui hiljem osutus vajalikuks paigutuses muudatusi teha. Lisaks kaasnes sellise paigutusega vaadete vahel liikumise animatsiooni juures visuaalne anomaalia.

Lõputöö autor vahetas eelnevalt kirjeldatu CSS3 *flexbox*[2]-i vastu. Tänu sellele ei pea eesrakenduse erinevate komponentide positsioone enam käsitsi arvutama. Nüüd on ka erinevate vaadete vahel liikumise animatsioon korras.

[1] https://developer.mozilla.org/en-US/docs/Web/CSS/Using_CSS_custom_properties

[2] https://www.w3.org/TR/css-flexbox-1/

6 Ühildamine teiste robotsõidukitega

Eelnevalt sai analüüsitud kasutajaliidese arendusetappi, mis toimus spetsiifiliselt Milrem Robotics-i robotsõidukile THeMIS. Antud peatükk analüüsib arendustööd, mida oleks vaja läbi viia robotsõidukitega liidestumise abstraheerimiseks. Eesrakenduse laiemalt kasutatavaks tegemine on osaliselt implementeeritud. Kui sõidukiga liidestumise abstraktsioon lõpuni implementeerida, oleks erinevat tüüpi robotsõidukitega ühilduvus lihtsamini saavutatav.

6.1 Kasutajaliidese definitsiooni üldistamine

Eelnevas arenduses oli kasutajaliidese eesrakendusel alati kindel vorm ja funktsionaalsus. Näiteks oli alati teada, mis nupud peaksid olema põhivaatel, millised on telemeetriaväljad ning mida saab sätete vaates konfigureerida. Et liidestuda erinevat tüüpi robotsõidukitega, tuleb siinkohal tagada kasutajaliidese kohanduvus vastavalt sõiduki võimalustele.

Seetõttu peab nüüd eesrakendus pärima kasutajaliidese tagarakenduselt kasutajaliidese ülesehituse definitsiooni iga kord, kui toimub uue sõidukiga ühendumine. See definitsioon võib olla tagarakenduse poolt saadetud *JSON*[1]-i kujul üle *Websocket RPC*. Kasutajaliidese definitsioon peaks kirjeldama:

- põhivaate ülemise ja alumise riba sisu definitsiooni;
- põhivaate erinevate režiimide (kaamerad, telemeetria, logo) definitsioonid;
- sõidukispetsiifiliste sätete definitsiooni.

Põhivaate ülemise ja alumise riba ning sätete definitsioonid koosnevad erinevat tüüpi elementidest. Analüüsi käigus leidis lõputöö autor, et nende seas võiksid olla:

- *Group* – element, mis grupeerib alamelemente;

[1] https://www.json.org/

- *OptionModalButton* – nupp, mille vajutamine avab valikutega dialoogiakna;
- *ProgressBar* – kindla väärtuse täituvust kuvav element;
- *RouteButton* – nupp, mille vajutamine viib kindlasse vaatesse;
- *SwitchableDisplayButton* – nupp, mis kuvab erinevaid väärtusi, mille vahel saab valida nupule korduvalt vajutades;
- *ToggleButton* – nupp, mis lülitab mingit atribuuti kahe erineva väärtuse vahel.

Hetkel realiseeritud üldistavas koodis kasutab eesrakendus *Vue.js*-i võimalusi, et tuvastada põhivaate mõlema riba igale elemendile vastav *Vue.js*-i komponendi tüüp ning anda komponendile nõutud sisu ja funktsionaalsus. Group tüüpi elemendi peal toimub itereerimine ning seda elementi saab kasutada ka rekursiivselt.

Eesrakendus puhverdab tagarakenduselt saadud kasutajaliidese definitsiooni seni kaua, kuni toimub ühendumine uue sõidukiga. Uus definitsiooni päring toimub ka siis, kui uueks sõidukiks on sama tüüpi sõiduk. Seda seetõttu, et osa kasutajaliidese definitsioonist – näiteks kaameratega seonduv info – võib sama tüüpi sõidukitel olla erinev.

6.2 Websocket RPC API üldistamine

Varasemalt olid tagarakenduse *Websocket RPC API*-l kindlad meetodid, millega sai erinevaid režiime (de)aktiveerida. Kuid nüüd ei ole iga erineva sõiduki tüübi jaoks otstarbekas luua enda *RPC API*-t. Lõputöö autori hinnangul on mõttekam üldistada kõne all olev programmiliides selle teel, et erinevate režiimide atribuutide pärimiseks ja muutmiseks on ühtsed meetodid. Nii võiks tagarakenduses olla RPC meetod *get_property_value*, mis võtab argumendina atribuudi nime ja tagastab selle hetkväärtuse. Teiseks meetodiks oleks *set_property_value*, mis lisaks atribuudi nimele võtab argumendina selle uue väärtuse ning sobimise korral uuendab atribuuti.

6.3 Juhtkangiga liidestumise üldistamine

Spetsiifiliselt robotsõidukiga THeMIS ühenduvas kasutajaliidese tagarakenduses oli juhtkangi vastendustes võimalik otse viidata erinevatele sõidukiga seonduvatele režiimidele ja tegevustele. Nüüd aga peaksid vastendused olema üldistatud – need võiksid viidata üldlevinud tegevustele otse (nagu sõiduki sisse/välja lülitamine, liigutamine). Vastendustes tuleks ilmselt kasutada ka koodnimetusi, et katta robotsõidukite

spetsiifilisemaid režiime/tegevusi (nagu näiteks *mode1*, *mode2*, *action1*, *action2*). Erinevate sõiduki tüübid võivad sama koodnimega juhtkangi sõnumit kasutada erineva režiimi või tegevuse jaoks.

6.4 Sõidukiga liidestumise üldistamine

Uues tagarakenduse programmikoodis on vaja luua abstraktsioon iga toetatud sõiduki tüübi jaoks. Siinkohal tasuks rakendada objektorienteeritud programmeerimist. Igale sõidukitüübile võiks vastata kindel Pythoni klass, mis (otse või kaudselt) laiendab abstraktset *BaseVehicle* klassi. *BaseVehicle* klassil oleksid üldised abstraktsed meetodid kasutajaliidese definitsiooni pärimiseks, atribuutide väärtuste küsimiseks ja muutmiseks ning sõidukiga ühendumiseks ja ühenduse katkestamiseks.

Sõidukite ilmumise ja kadumise tuvastamiseks vastutaksid samuti sõiduki tüübile vastavad (sõidukeid haldavad) klassid. Need laiendaksid *BaseVehicleManager* klassi. Iga ilmunud sõiduki jaoks loodaks tüübile vastavalt sõiduki (*BaseVehicle* klassi laiendav) objekt ning lisataks üldisesse sõidukiobjektide loendisse. Sõiduki kadumisel eemaldaks sõiduki haldaja vastava objekti eelmainitud loendist.

Kui eelnevalt kirjeldatu on realiseeritud, siis tuleks teha arendustööd, et tuua robotsõidukiga THeMIS liidestuv kood üle uuele süsteemile. Sisuliselt tähendaks see sõidukile vastava klassi (*ThemisVehicle*) ja seda tüüpi sõidukite haldava klassi (*ThemisVehicleManager*) loomist. Klassi *ThemisVehicle* loomine eeldaks muuhulgas režiimide väärtuste ühtlustamist, et eesrakenduses ei peaks senisel määral arvestatama THeMIS-e spetsiifikaga.

6.5 Kasutajaliidese ühildamine teiste robotsõidukitega

Nagu THeMIS-e puhul, tuleks ka iga uut tüüpi robotsõiduki jaoks luua sõidukit kirjeldav klass ja sõidukit haldav klass. Kui THeMIS-ega on liidestuvus on uue süsteemi kaudu tagatud ja testitud, siis ei tohiks teiste robotsõidukitega ühildamine olla enam suur töö. Sarnaste robotsõidukite kohta saaks luua ühiseid ülemklasse.

Kui sõiduki abstraheerimine on teostatud, on võimalik luua ka *mock*-itud sõidukeid. Sellised virtuaalsed sõidukid võimaldaksid luua tagarakendust hõlmavaid integratsiooniteste.

7 Tähelepanekud ja järeldused

Suurem osa arendustööst kulus robotsõidukile THeMIS spetsiifilise kasutajaliidese variandi arendamiseks. Kuna antud lõputöö kirjutamise hetkel on olnud see robotsõiduk enamasti militaarses rakenduses, siis on ka kasutajaliidese kasutuskogemuse kujundamisel püütud antud valdkonna iseärasusi silmas pidada.

Esiteks väärib välja toomist lihtsuse aspekt. Kasutajaliidese stiilimisel on piirdutud üsna väheste värvidega. Värvide juures on taotletud võimalikult kõrget kontrasti. Kujunduselemendid on tehtud piisavalt suured, et ekraanil olevaid nuppe oleks kergem kasutada ning tekst oleks hästi nähtav.

On oluline, et kasutajaliides reageeriks kasutaja käsklustele võimalikult kiiresti. Esialgse arenduse käigus lõi lõputöö autor animatsioonid, mis rakenduvad erinevate vaadete vahel liikumisel. Paraku ei olnud ettevõttes kasutusel olevad operaatorseadmed tehniliste näitajate poolest ja/või riistvaraprobleemide tõttu kuigi võimekad. Seetõttu tuli teha otsus kasutajaliidese animatsioonid vaikimisi keelata. Kasutaja võib animatsioonid soovi korral sisse lülitada, kuid ettevõtte käsutuses olnud operaatorseadmete puhul ei olnud see kuigi mõistlik. Õnneks ei muuda nende keelamine kasutuskogemust märkimisväärselt ebaintuitiivsemaks. Animatsioonide keelamine haakub hästi eelmisel lõigus toodud lihtsuse aspektiga.

Suurte robotsõidukite kasutajaliidese puhul on üks olulisemaid aspekte ohutus. Eesrakenduse juures tuleb see kõige otsesemalt välja ilmselt põhivaate nuppude juures. Esialgu olid põhivaate alumise riba nupud lihtkujul lülitavad nupud – kui kasutaja vajutas selle peal, siis lülitus vastav säte (näiteks generaatori sisselülitatus) teise olekusse. Kuid sellise lähenemise korral oli oht, et kasutaja võib kogemata operaatorseadme ekraani puudutades teha soovimatuid toiminguid. Seetõttu sai lisatud alumise riba nuppudele dialoogiakna avamine – nüüd peab kasutaja tegema kaks vajutust, et mingit režiimi muuta. Ohutust aitab tagada ka sõidukiühendusele kehtestatud lubatud latentsusaja piirang.

8 Edasised arendusvõimalused

Kui lõputöö autor alustas uue kasutajaliidese eesrakenduse arendamist, siis oli eesmärgiks saada mõistliku ajaga valmis esialgne funktsionaalne prototüüp. Seetõttu sai otsustatud võtta kasutusele CSS raamistik *Semantic UI*. Antud raamistik tundus oma olemuselt ja selle küllaltki kõrget populaarsust arvestades tol etapis kasutajaliidese eesrakenduse jaoks parima valikuna. Nüüd, kus kasutajaliidesest valmis stabiilne ja täielikult funktsionaalne versioon, oleks aga mõttekas saada *Semantic UI* kasutamisest lahti. Stiilimise osas tähendaks see nullist peale kirjutatud CSS / SCSS[1] faile. Seni *Semantic UI*-d kasutanud funktsionaalsus tuleks tuua üle *Vue.js*-ile. Selliste muudatuste põhjenduseks on *Semantic UI* liigne keerukus antud kasutajaliidese jaoks. Lõputöö autor töötas lõppkokkuvõttes suuresti vastu raamistiku poolt pakutud kujundusele. Seda eriti öörežiimi realiseerimisel. Raamistiku kasutus nõuab kindlat ja lõputöö autori arvates küllaltki kohmakat HTMLi kasutust. Samuti tõstab *Semantic UI* oluliselt ehitamisaega ja suurendab oluliselt ehitatud kasutajaliidese paki suurust. Ilmselt langetab raamistik ka eesrakenduse jõudlust tunduvalt (hetkel pole veel täpsemaid katsetusi selle osas tehtud).

Vue.js-i peaarendaja Evan You kirjutas ühes artiklis, et *Vue.js*-i uus versioon – 3.0 – saab tõenäoliselt valmis 2019. aasta jooksul. On lubatud, et see versioon saab olema märgatavalt parema jõudlusega, samas olles suuremalt jaolt ühilduv praeguse versiooniga (2.x). Sellel saab olema ka mitmeid uusi võimalusi – näiteks *Javascripti* klasside põhised *Vue.js* komponentide definitsioonid ja *TypeScript*-i tugi [5]. Kui uus versioon välja tuleb, siis tasub kasutajaliidese eesrakenduse refaktoreerimist kaaluda.

Selles lõputöös kasutati kasutajaliidese eesrakenduse ehitamiseks arendustööriista *Webpack* 4. Kaasaegses Javascript-is on võimalik selle vahendi kasutus asendada veebibrauseri poolt täidetavate *import*[2]-idega. Kui jätta Webpack 4 kasutus ära, siis muutub kasutajaliidese ehitamine märksa lihtsamaks – triviaalsemal juhul piisab ainult

[1] https://sass-lang.com/

[2] https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Statements/import

failide kopeerimisest vastavasse kausta. *Webpack*-i puhul loodi ehitamisel üldjuhul üks suurem väljundfail (*Javascript* koodi kujul). Ent see fail võib (hetkel näiteks *Semantic UI*-st tingitult) kasvada ülemäära suureks. Webpacki kasutamine tasub mitmetel juhtudel ära [6]. Ent Webpack-i enda arendusprotsessi kohta on toodud välja üsnagi fundamentaalset kriitikat [7].

Kasutajaliidese arendusprotsessis ei olnud automaattestidel paraku kuigi suurt osakaalu. Lõputöö autor küll kirjutas eesrakenduse jaoks mocki ning testimise eest vastutav kolleeg lõi ka seda rakendavad veebibrauseripõhised automaattestid, kuid arenduse vältel neid teste regulaarselt ei jooksutatud. See tähendas, et lõputöö autor pidi enne enne mingi mittetriviaalse muudatuse commit-imist proovima manuaalselt, kas kasutajaliidese funktsionaalsus säilis. Eesrakendusele selle lõputöö raames ühikteste ei loodud, ehkki *Vue.js*-il on vajalik võimekus olemas[1]. Seega tasub tulevases arenduses automaatteste kõrgemalt prioritiseerida.

[1] https://vuejs.org/v2/guide/unit-testing.html

9 Kokkuvõte

Antud lõputöö raames tegi lõputöö autor arendustööd esmalt robotsõiduki THeMIS spetsiifilise kasutajaliidese arenduseks. Esialgne põhirõhk oli uue eesrakenduse arendus, tagarakendus oli eelnevalt juba olemas. Kasutajaliidese eesrakenduse põhifunktsionaalsusteks on erinevate režiimide muutmise võimaldamine, telemeetria kuvamine, kaamerapiltide näitamine ning veateadete kogumine ja kuvamine. Selles arendusetapis oli eesrakendusel kindel struktuur (nuppude paigutus, nende olemus/funktsionaalsus, kaamera-/telemeetriarežiimid, sätted jms).

Teises arendusetapis analüüsis lõputöö autor arendustööd, mida oleks vaja teha robotsõidukitega liidestumise abstraheerimiseks. Ühelt poolt tähendas see seda, et eesrakendus peaks tagarakenduselt pärima sõiduki vahetumisel kasutajaliidese definitsiooni, mille abil ehituks eesrakendus vastavalt sõiduki omadustele ja võimalustele. Teisalt tooks see kaasa tagarakenduses objektorienteeritud lähenemise, et saavutada ühilduvus erinevat tüüpi robotsõidukitega.

Lõpuosa kahes peatükis on analüüsitud kasutajaliidese aspekte ning toodud välja arenguvõimalused.

Kasutatud kirjandus

[1] S. Jones, „Why Web Apps are Faster to Develop than Desktop Apps,“ Black Pepper Software, 21 March 2014. [Võrgumaterjal]. Available: https://www.blackpepper.co.uk/blog/web-apps-faster-develop-desktop-apps. [Kasutatud 27 May 2019].

[2] S. Smith, „Can I use the free QT for c++ commercially?,“ Quora, 29 April 2018. [Võrgumaterjal]. Available: https://www.quora.com/Can-I-use-the-free-QT-for-c++-commercially/answer/Sarah-Smith-91. [Kasutatud 27 May 2019].

[3] S. Daityari, „Angular vs React vs Vue: Which Framework to Choose in 2019,“ CodeinWP, 27 April 2019. [Võrgumaterjal]. Available: https://www.codeinwp.com/blog/angular-vs-vue-vs-react/. [Kasutatud 27 May 2019].

[4] J. Tepandi, „Tarkvara protsessid, kvaliteet ja standardid,“ 3 October 2018. [Võrgumaterjal]. Available: https://tepandi.ee/tks-loeng.pdf. [Kasutatud 27 May 2019].

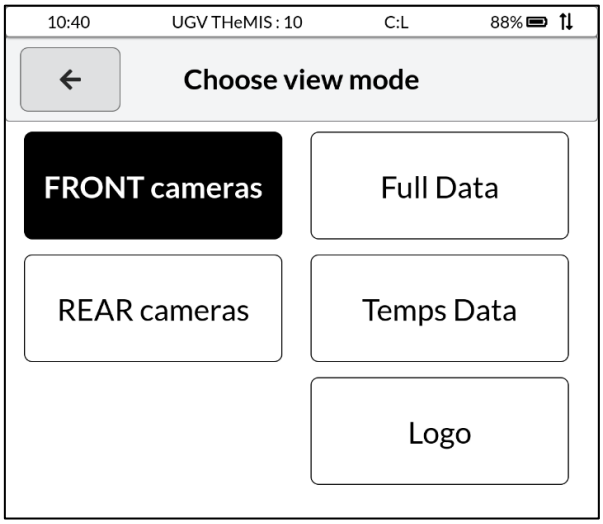
[5] E. You, „Plans for the Next Iteration of Vue.js,“ 30 September 2018. [Võrgumaterjal]. Available: https://medium.com/the-vue-point/plans-for-the-next-iteration-of-vue-js-777ffea6fabf. [Kasutatud 27 May 2019].

[6] Webpack, „Why webpack,“ 24 December 2018. [Võrgumaterjal]. Available: https://webpack.js.org/concepts/why-webpack/. [Kasutatud 27 April 2019].

[7] A. M. Baptista, „The Problem with Webpack and Why It Is (Kind of) Our Fault,“ A Medium Corporation, 12 March 2018. [Võrgumaterjal]. Available: https://medium.com/@allanbaptista/the-problem-with-webpack-8a025268a761. [Kasutatud 27 May 2019].

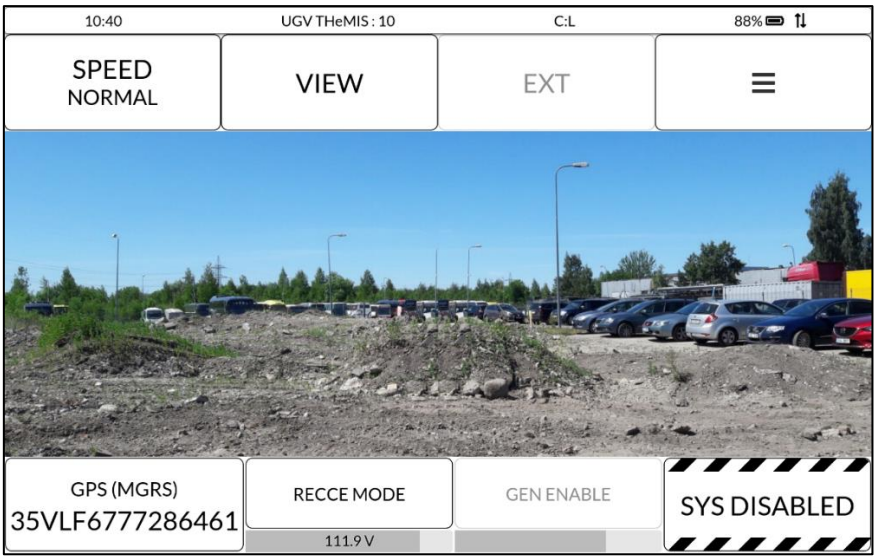
Lisa 1 – Põhivaate režiimid

Põhivaate keskmise osa sisu määrab põhivaate režiim. Nende vahel saab valida, kui vajutada põhivaates nuppu View. Seda nuppu vajutades avaneb vaade *MainModeSelectionView.*

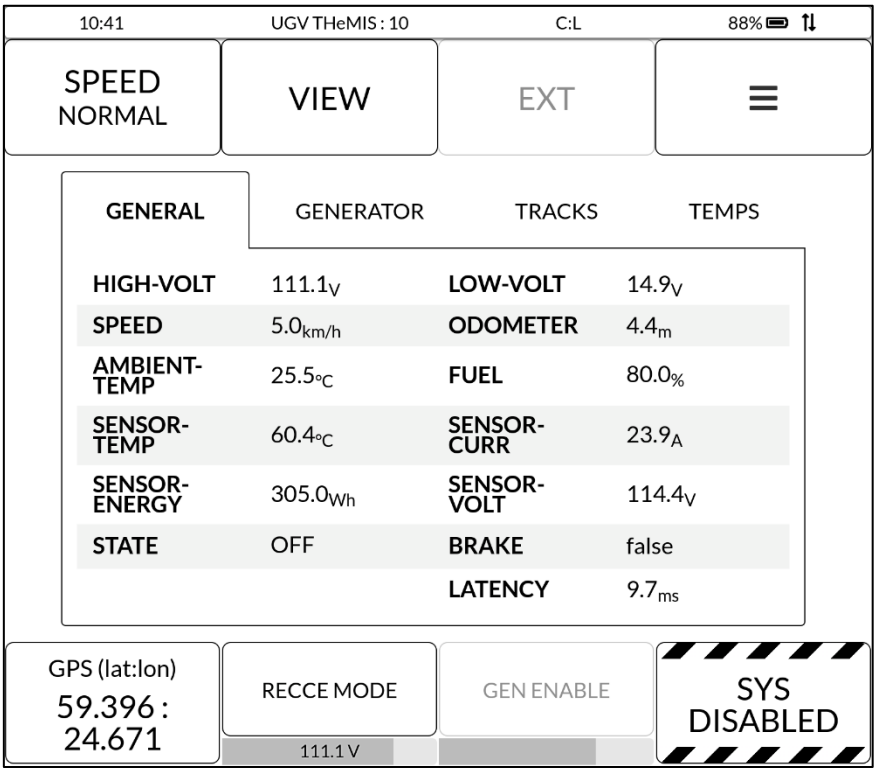


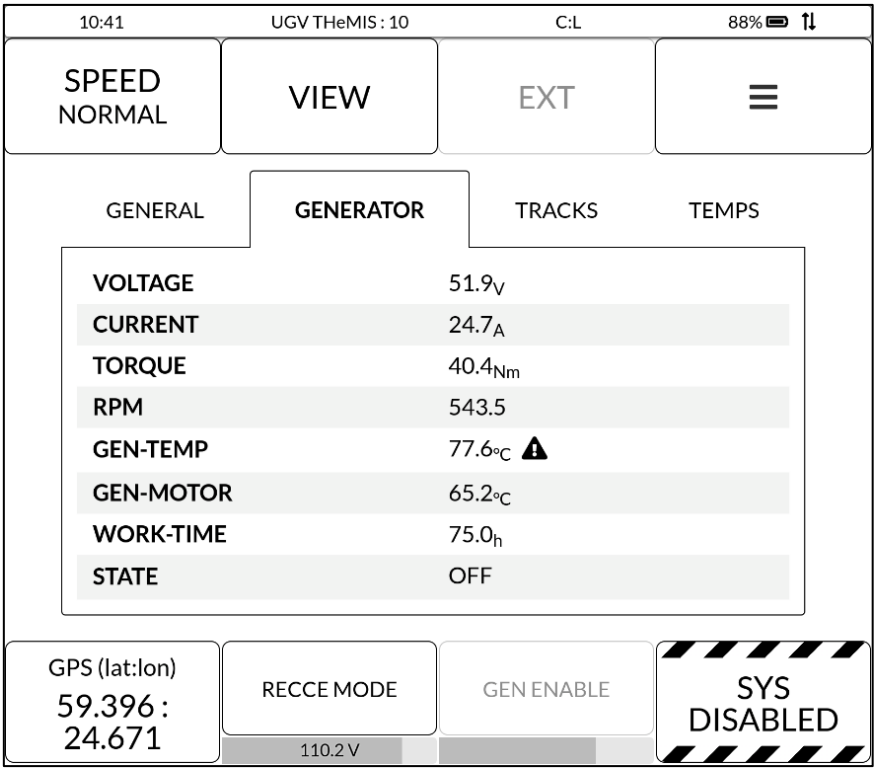
Robotsõidukile THeMIS spetsialiseeritud kasutajaliidese variandil on viis põhivaate režiimi:

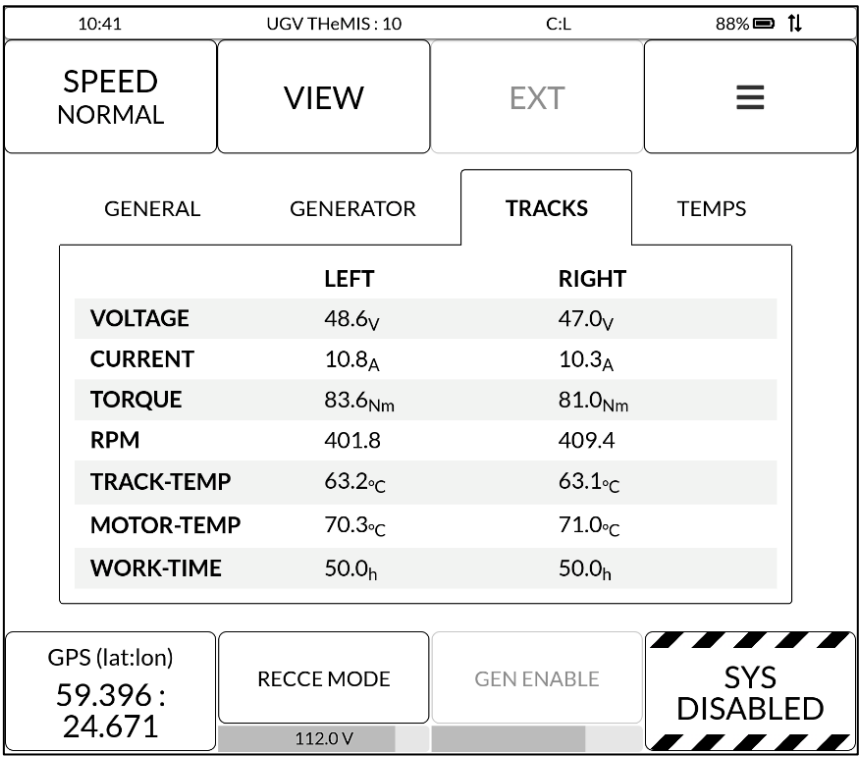
Kaamerarežiim *FRONT cameras* (*REAR cameras* on analoogiline):

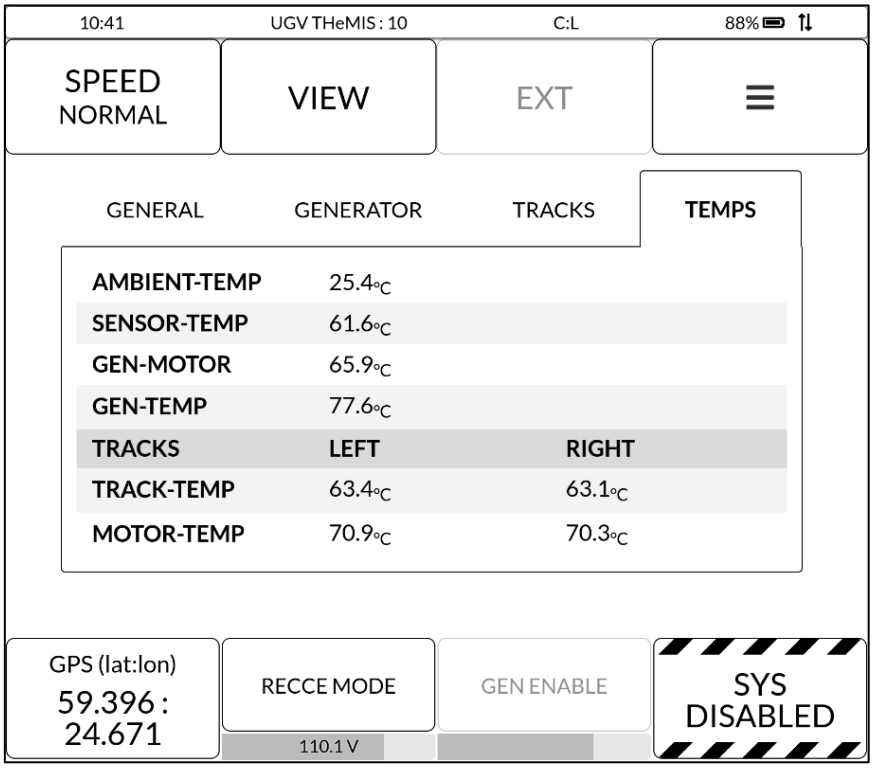


Telemeetriarežiim *Full Data*:

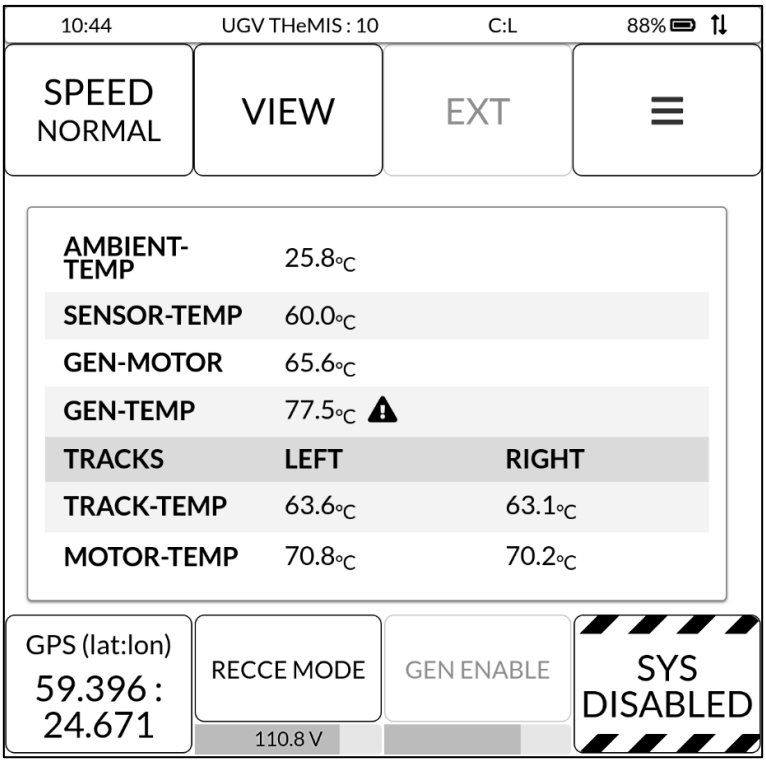




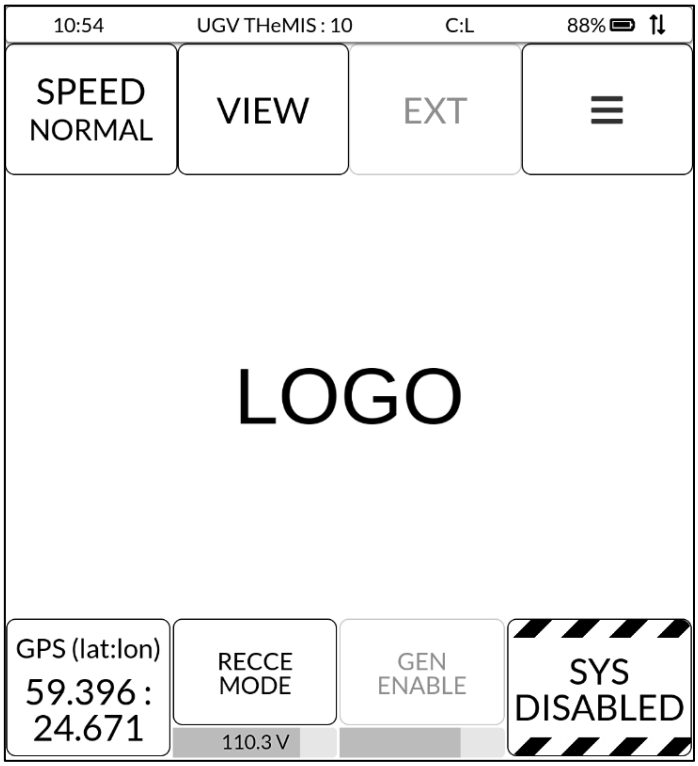

10:41
UGV THeMIS : 10
C:L
88%
SPEED
NORMAL
VIEW
EXT
GENERAL
GENERATOR
TRACKS
TEMPS
LEFT
RIGHT
VOLTAGE
48.6V
47.0V
CURRENT
10.8A
10.3A
TORQUE
83.6Nm
81.0Nm
RPM
401.8
409.4
TRACK-TEMP
63.2°C
63.1°C
MOTOR-TEMP
70.3°C
71.0°C
WORK-TIME
50.0h
50.0h
GPS (lat:lon)
59.396 :
24.671
RECCE MODE
GEN ENABLE
SYS
DISABLED
112.0 V


10:41
UGV THeMIS : 10
C:L
88%
SPEED
NORMAL
VIEW
EXT
GENERAL
GENERATOR
TRACKS
TEMPS
AMBIENT-TEMP
25.4°C
SENSOR-TEMP
61.6°C
GEN-MOTOR
65.9°C
GEN-TEMP
77.6°C
TRACKS
LEFT
RIGHT
TRACK-TEMP
63.4°C
63.1°C
MOTOR-TEMP
70.9°C
70.3°C
GPS (lat:lon)
59.396 :
24.671
RECCE MODE
GEN ENABLE
SYS
DISABLED
110.1 V

Ainult temperatuuride telemeetriat kuvav režiim *Temps Data*;



Staatilist kujutist kuvav režiim *Logo*:

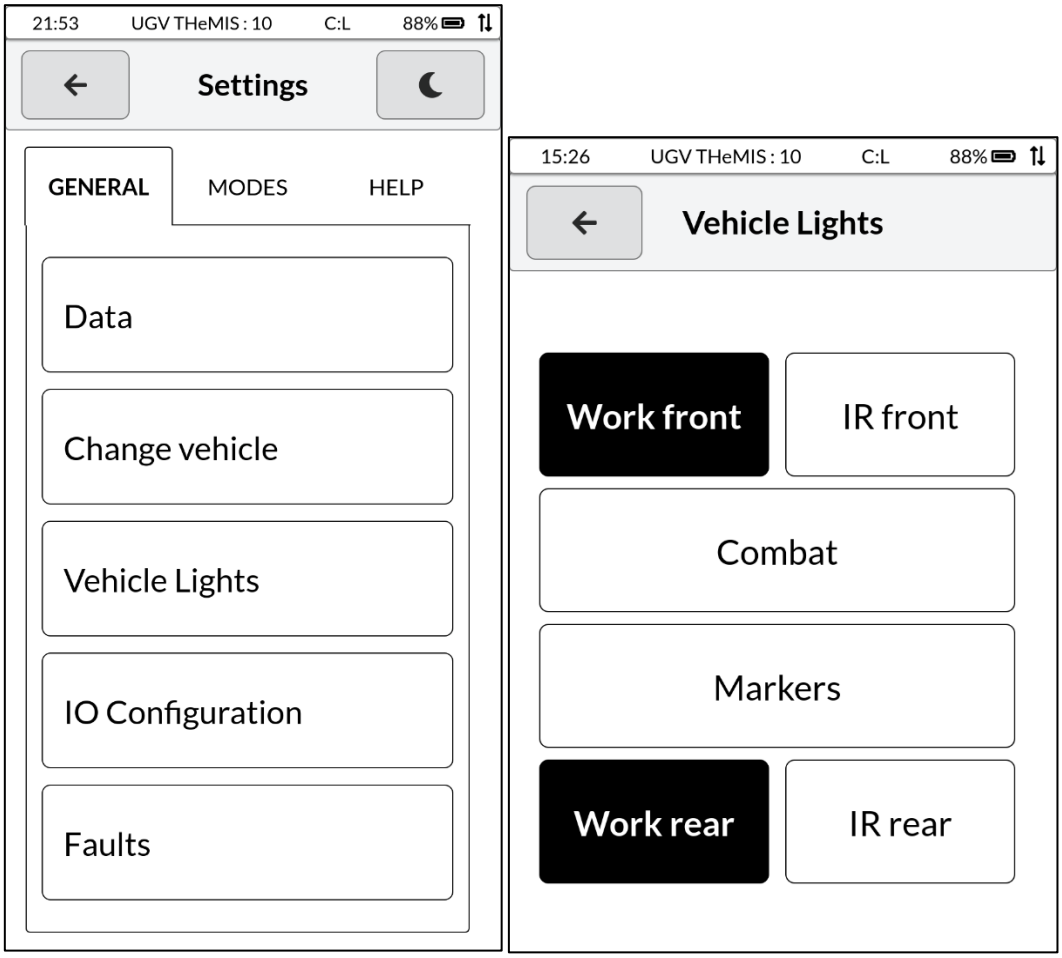


Lisa 2 – Sätete vaade

Sätete vaade (*SettingsView*) võimaldab näha ja muuta erinevaid sõiduki ja kasutajaliidese sätteid. Lisaks väärib välja toomist, et sätete vaates saab vahetada aktiivset sõidukit, näha loendit sõidukil esinenud vigadest ja laadida kogutud logid logiserverisse.

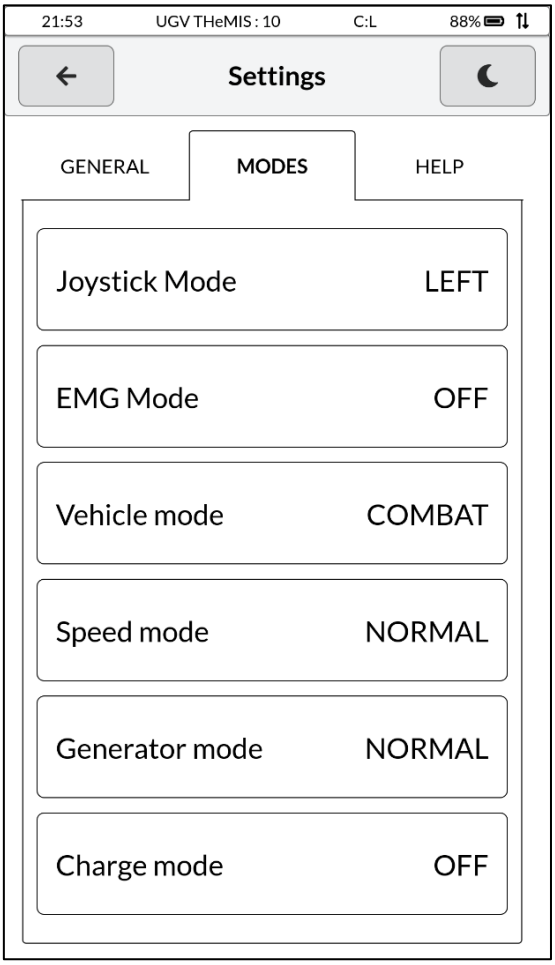
Järgnevalt on toodud kuvatõmmised kõigist kolmest sätete vaate sakist.

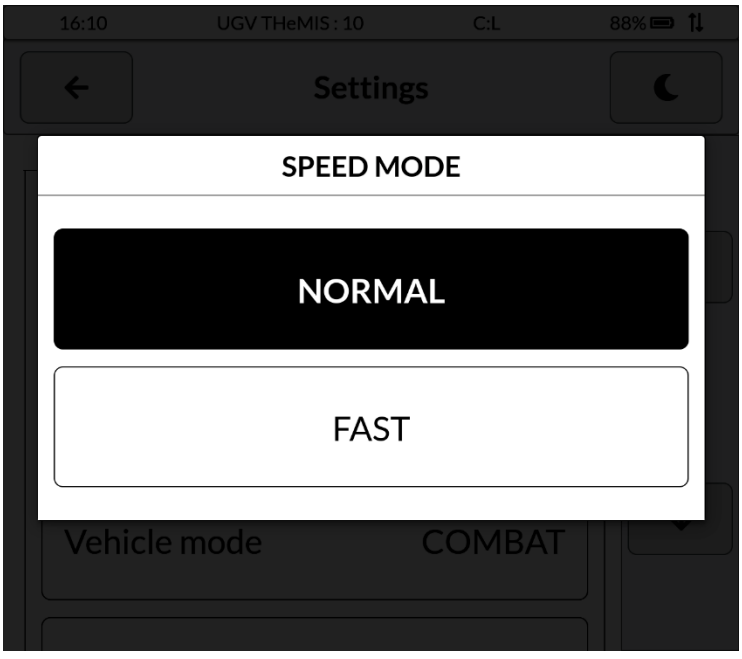
General – üldised sätted ja toimingud



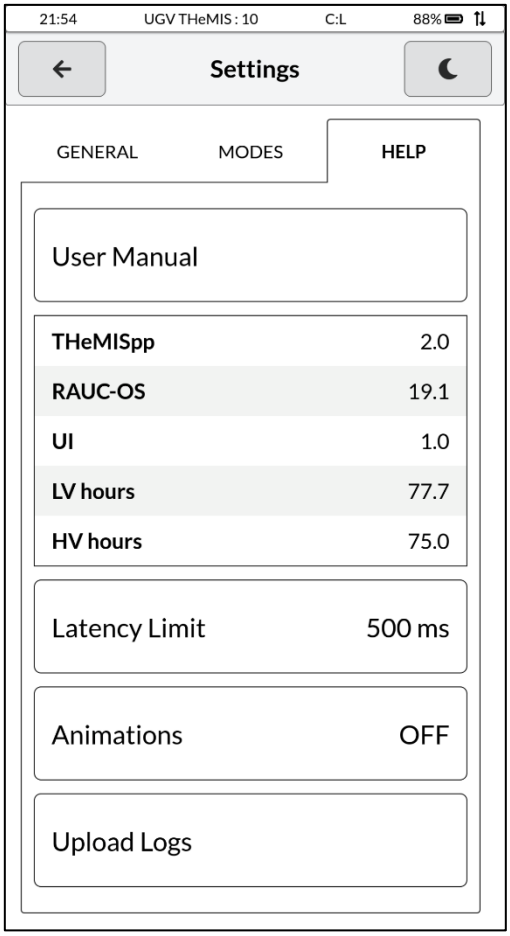
(*IO Configuration* vaade on *Vehicle Lights* vaatega analoogiline.)

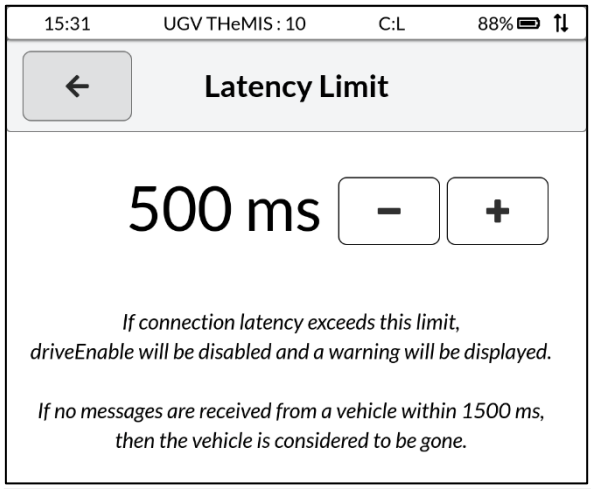
Modes – juhtkangi ja sõiduki režiimid





Help – abistav informatsioon, kasutajaliidest puudutavad sätted ja toimingud



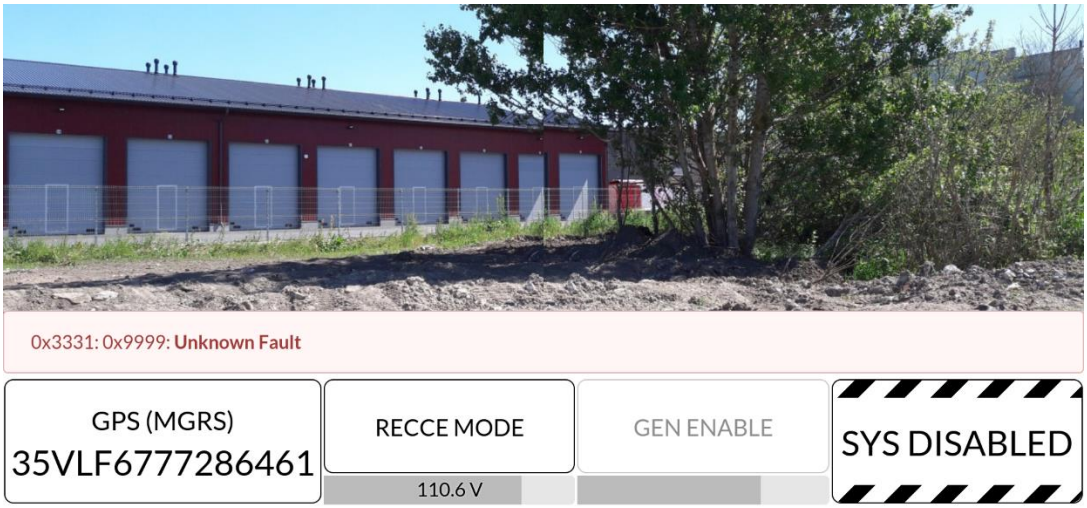


Lisa 3 – Teavitused

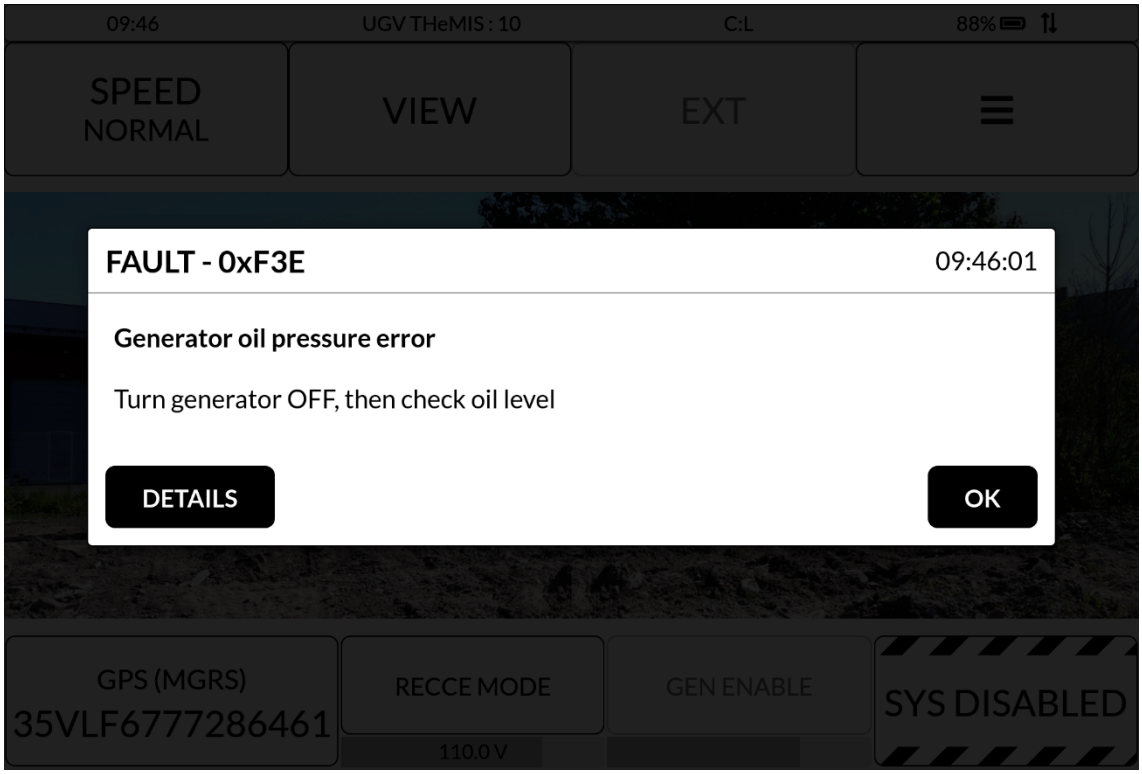
Lõputöös arendataval kasutajaliidese eesrakendusel on erinevad viisid, kuidas see kuvab teavitusi.

Väiksema tõsidusega sõiduki vigu kuvatakse sõnumiriba abil:

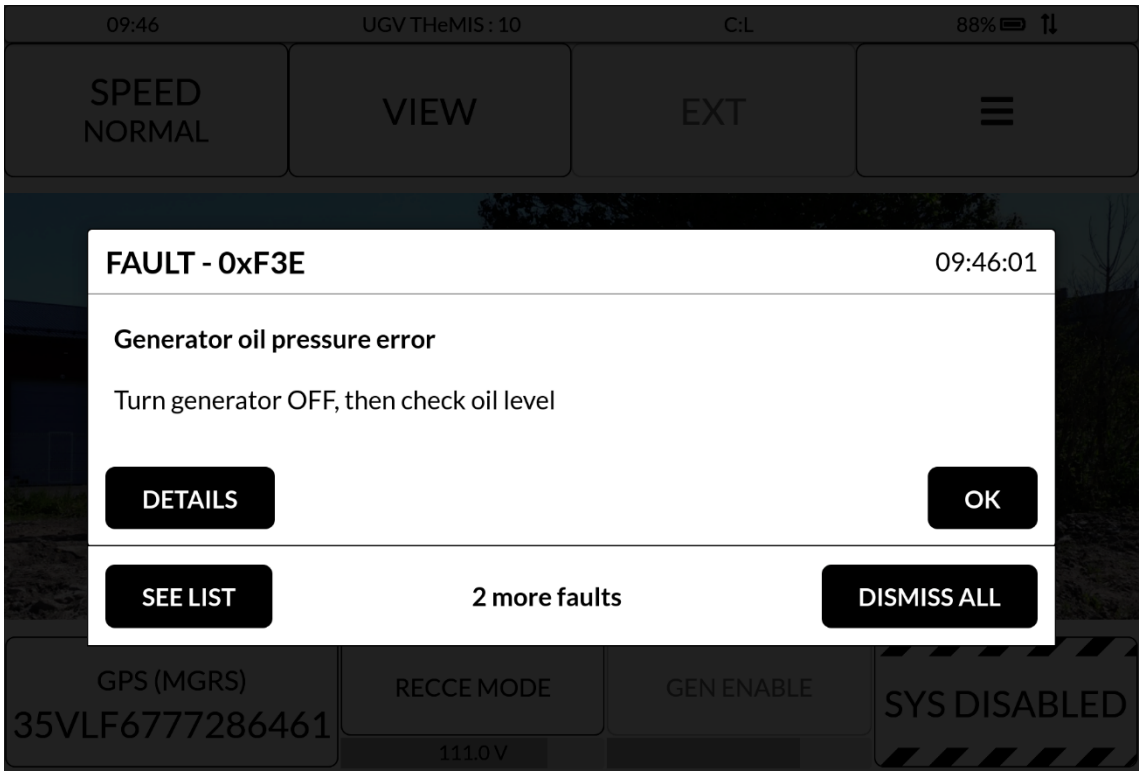
Tundmatuid sõiduki vigu kuvatakse samuti sõnumiriba abil:



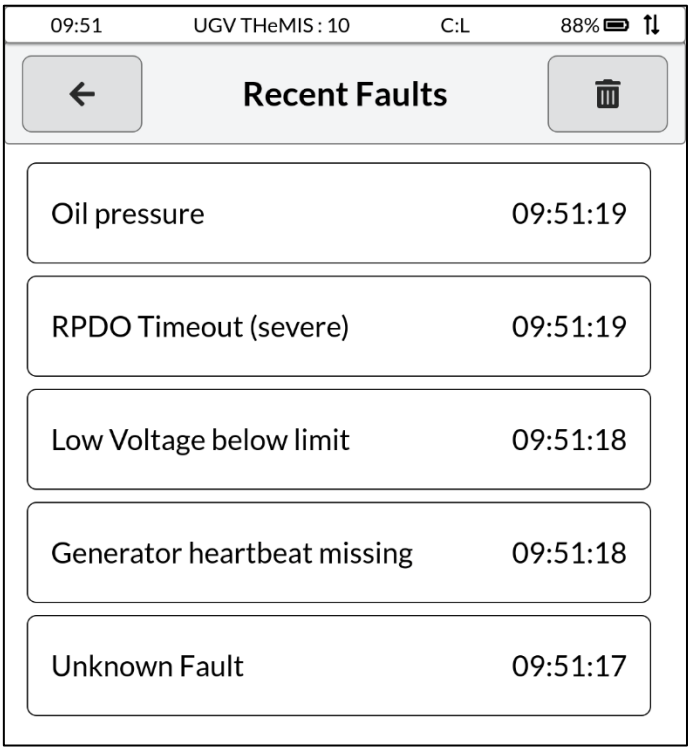
Kriitilisemaid sõiduki vigu kuvatakse dialoogiakna abil (component *FaultModal*):



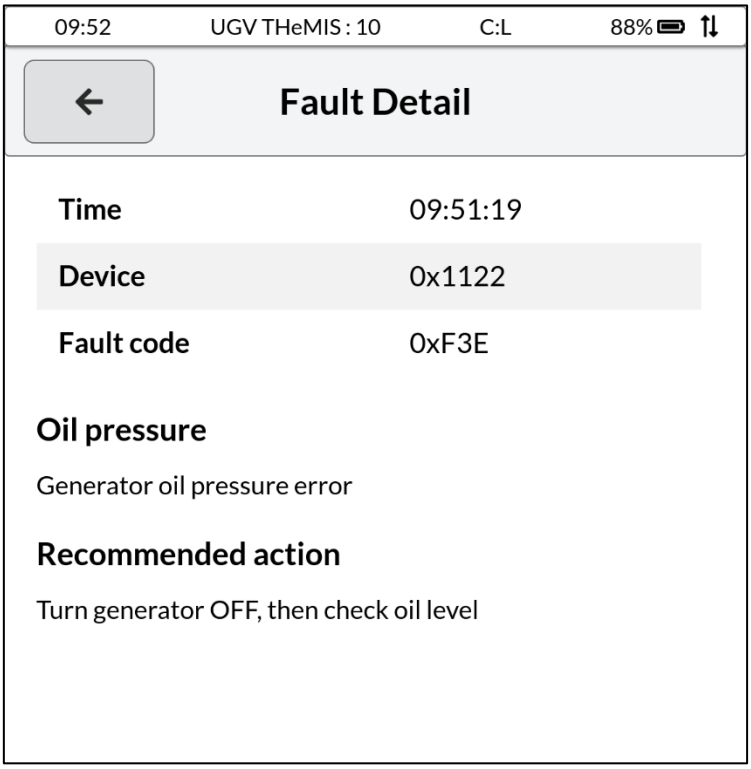
Kui ootel on veel vähemalt üks (unikaalne) viga, siis ilmub dialoogiakna alla lisariba:



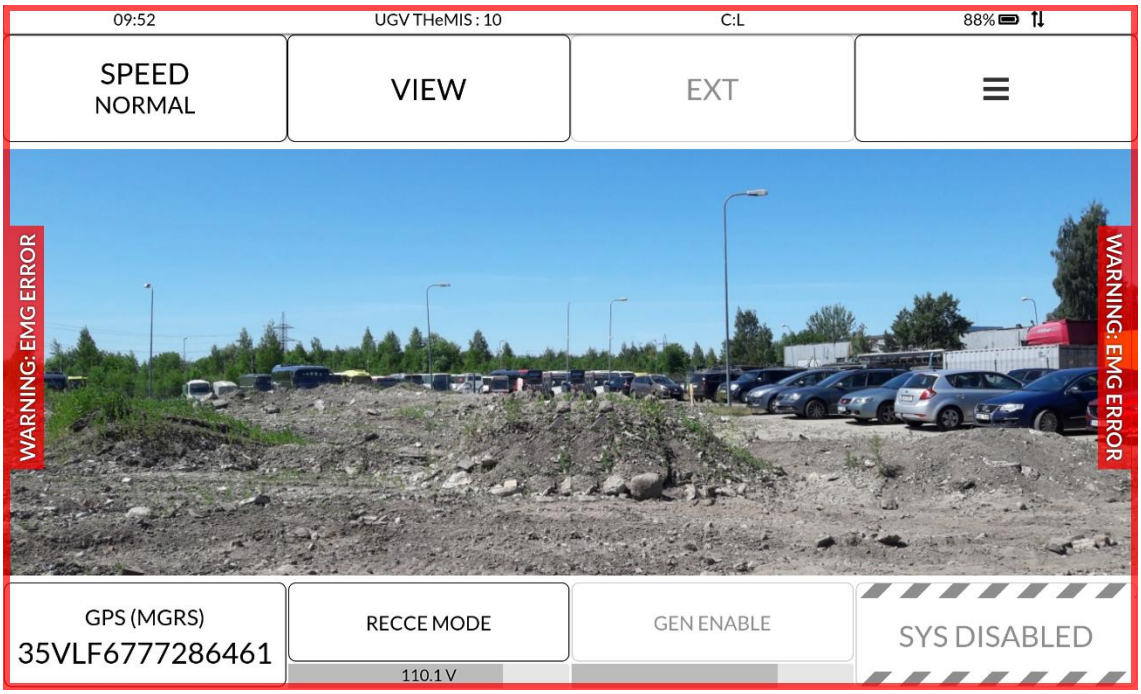
Eesrakenduseni jõudnud veateated puhverdatakse. Loend viiekümnest viimasest veatest on nähtav vaates *FaultListView:*



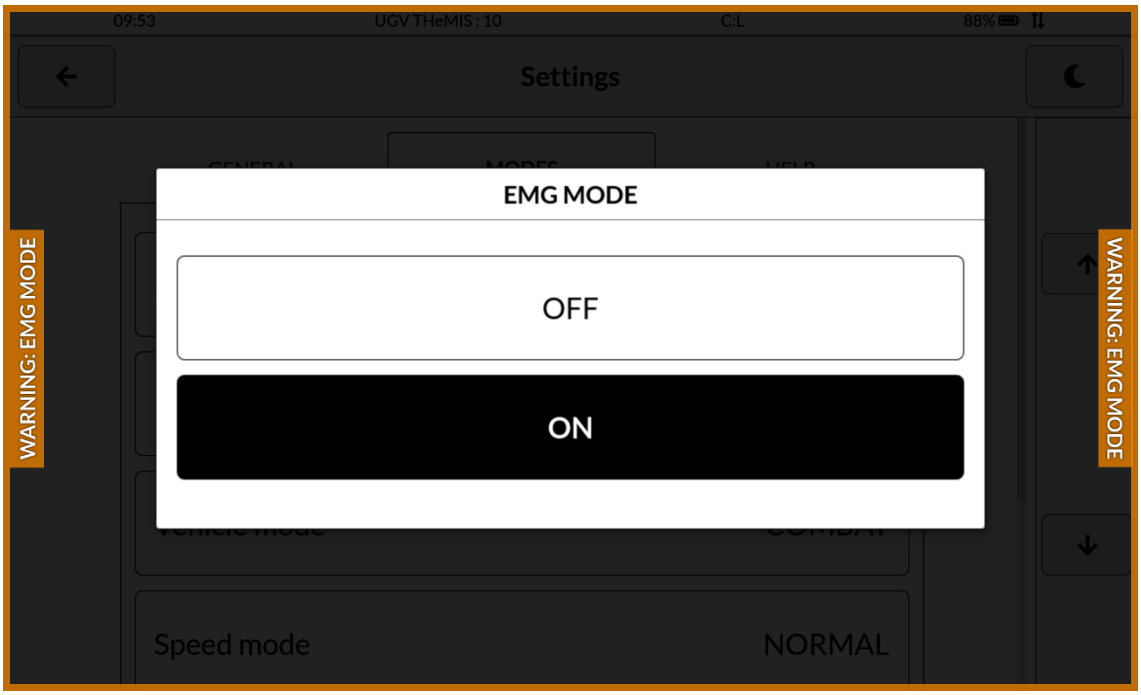
Konkreetse veateate detaile näitab vaade *FaultDetailView*:



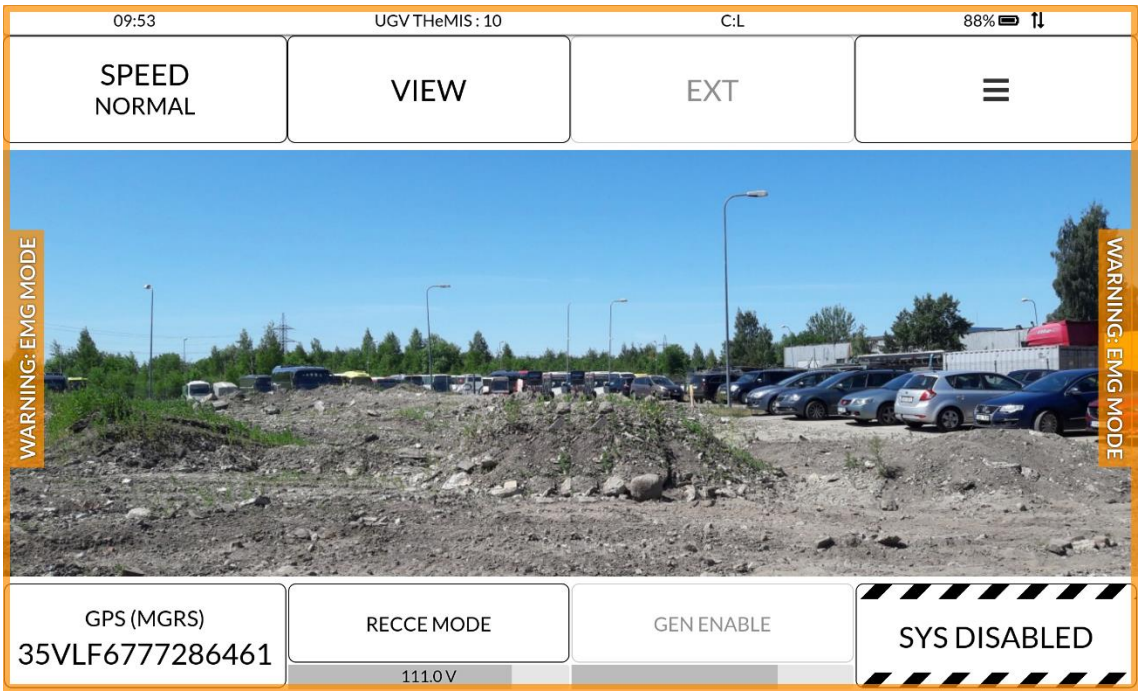
Püsivamat viga, *EMG Error* näidatakse punast värvi vastavat teksti kuvava raamiga:



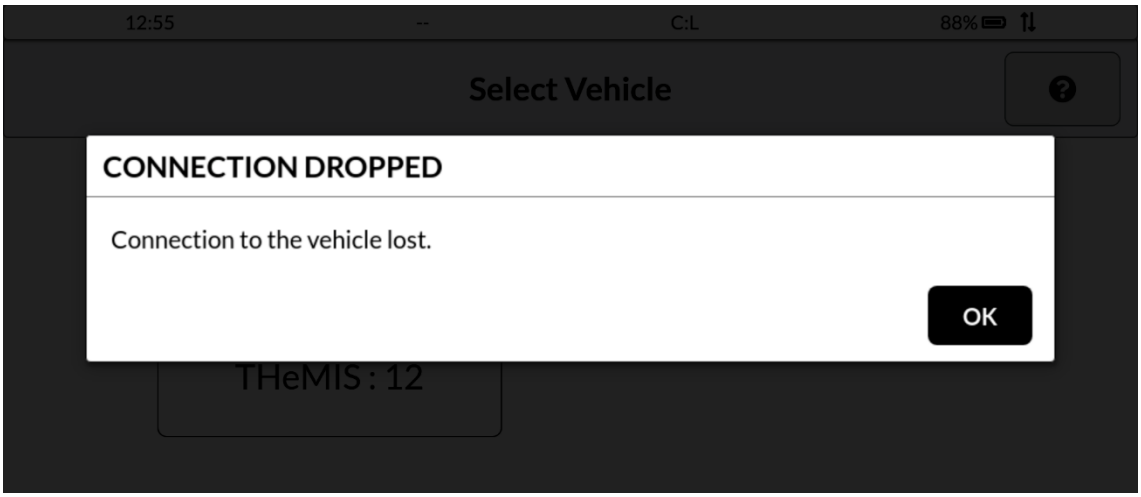
EMG Error-it on võimalik ignoreerida, aktiveerides sätetes *Modes* alt *EMG Mode*:



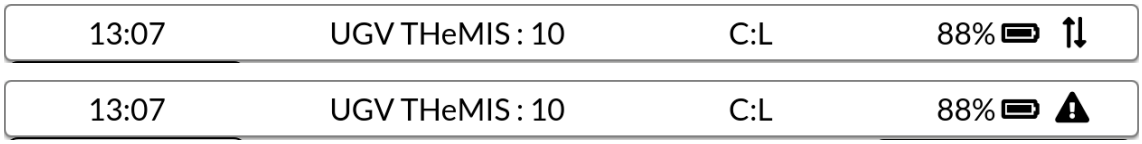
EMG Mode raam on globaalselt nähtav.
(Ohulüliti aktiivsust näidatakse analoogiliselt. Sel juhul on raami tekstiks *EMG Stop*.)



Kui aktiivse sõidukiga katkeb ühendus, kuvatakse vastav teade ning aktiivseks saab esialgne sõiduki valimise vaade.

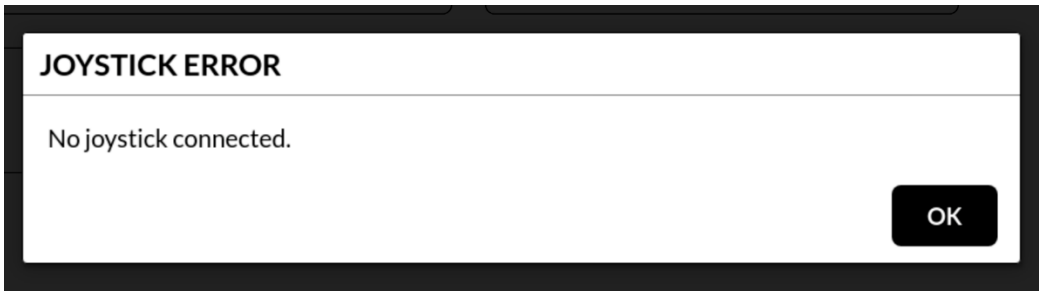


Juhul, kui sõiduki ja kasutajaliidese vahelise ühenduse latentsusaeg ületab kasutaja poolt muudetavat piirangut, kuvatakse olekuribal (*StatusBar*) vilkuv hüüumärgi ikoon.

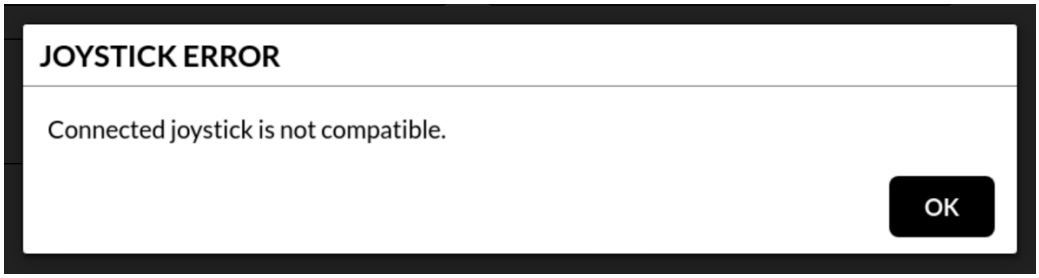


Lisaks sõidukiga seonduvatele teavitustele kuvab kasutajaliides ka juhtkangiga liidestumisel esinevaid vigu. Juhtkangi ühendatus on sõiduki liigutamise eelduseks.

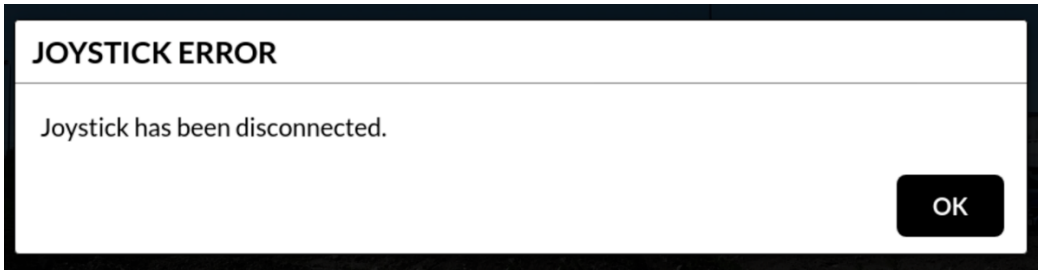
Kui kasutajaliidese eesrakenduse käivitumise hetkel pole juhtkang operaatorseadmega ühendatud, näidatakse kasutajaliidese sellist veateadet:



Kui operaatorseadmega ühendatatakse juhtkang, mille kohta pole vastenduse konfiguratsioonifaili, kuvatakse järgnev viga:



Kui aktiivse juhtkangiga kaob ühendus, siis kuvatakse järgnevat viga:



Lisa 4 – Kuvatõmmised öörežiimist

Kasutajaliidese öörežiim on aktiveeritav sätete vaate päiseriba parempoolse nupu kaudu. Selles režiimis vahetub eesrakenduse must ja valge värviskeem rohelise ja musta vastu.