TALLINNA TEHNIKAÜLIKOOL
Infotehnoloogia teaduskond
Tarkvarateaduse instituut

Karmo Kuurberg 153389IAPM

KASUTAJALIIDESE RAAMISTIK JUHTSÜSTEEMIDELE

Magistritöö

Juhendaja: Jaagup Irve
Tehnikateaduste magister

Tallinn 2017

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: Karmo Kuurberg

08.05.2017

Annotatsioon

Käesoleva töö eesmärk on luua alternatiiv olemasolevale C++ ja Qt põhisele raamistikule, mille abil luuakse kasutajaliideseid CDP-nimelise tarkvaraga loodud juhtsüsteemidele, kuid mitmed töö tulemused on tõenäoliselt kohaldatavad ka teistele juhtsüsteemidele. Loodav raamistik oleks modernsema välimusega ja toetaks laia valikut seadmeid. Selleks kasutatakse veebitehnoloogiaid.

Töös otsitakse võimalusi, kuidas raamistik oleks lihtne ja kasutatav ka vähese veebiarenduskogemusega inimesele. Selle käigus üritatakse võimalikult palju kasutada kolmanda osapoole loodud raamistikke ja teeke, et vähendada arendusele ja eriti hilisemale hooldusele kuluvat aega. Lisaks uuritakse, kuidas võimalikult hästi hoida lahus välimus funktsionaalsusest, et disainitrendide muutudes ei peaks mõne aasta pärast otsast peale alustama.

Töö tulemusena valmib uus raamistik, mille aluseks valitakse Javascripti raamistik Angular. Sellele luuakse hulk direktiive, mis pakendavad kasutajaliidese elemente – nagu nupp, tekstiväli, mõõdik – nii, et juhtsüsteemiga suhtlemiseks piisaks vaid deklaratiivsest koodist ja et elemendi välimus oleks lahus funktsionaalsusest.

Valminud töö lähtekood on kättesaadav lehelt https://gitlab.com/karmo/angular-cdp.

Lõputöö on kirjutatud eesti keeles ning sisaldab teksti 55 leheküljel, 8 peatükki, 27 joonist, 7 tabelit.

Abstract

User Interface Framework for Control Systems

The aim of this work is to create an alternative to existing framework based on C++ and Qt that is used to create user interfaces for control systems developed with software called CDP. However, some of the results of this work are likely to be relevant to other control systems as well. The new UI framework would have a more modern look and would support a wide variety of devices. Web technologies are used for this purpose.

It is searched how to create a framework which would be simple and usable even to a person with little experience of web development. One of the goals is to use as many third-party, frameworks and libraries as possible to reduce the amount of time spent on development and especially later on maintenance. In addition, it is researched how to keep UI design separate from functionality to avoid big changes should design trends change in coming years.

One of the results of this study will be a new framework based on a Javascript framework called Angular. Several new directives are created to wrap common UI elements like button, input, gauge to enable communication with control system using only declarative code. Where possible, the design and functionality of the element will be separated.

The source code of this work is available at https://gitlab.com/karmo/angular-cdp.

The thesis is in Estonian and contains 55 pages of text, 8 chapters, 27 figures, 7 tables.

Lühendite ja mõistete sõnastik

CDP	*Control Design Platform* Ettevõtte CDP Technologies toodetud raamistik juhtsüsteemide loomiseks [1].
CDP signaal	CDP rakendusraamistiku kaudu teisele komponendile edastatud numbriline väärtus. Tavaliselt pärineb info mõnelt andurilt, kuid tegu võib olla ka näiteks süsteemi mälukasutuse või protsessori hõivatusega.
CDP süsteem	Antud töös on selle all mõeldud juhtsüsteemi, mis koosneb mitmest CDP raamistikuga loodud rakendusprogrammist, mis asuvad füüsiliselt eraldi kontrolleritel ja suhtlevad omavahel üle lokaalvõrgu.
StudioAPI	Protokoll, mille abil saab pärida andmeid CDP-ga tehtud juhtsüsteemilt. Selle alla kuulub näiteks juhtsüsteemi struktuur ja süsteemis liikuvate signaalide väärtused ehk süsteemi olek.
Vidin	Graafilise kasutajaliidese element, mis kuvab informatsiooni või pakub kasutajale võimalust suhelda rakendusprogrammiga [2].
Qt	Raamistik, mille antud töö kontekstis olulisim osa on graafiliste rakenduste loomine [3]. Toetatud on teiste seas nii Linux kui Windows operatsioonisüsteemid.

Sisukord

Jooniste loetelu

Tabelite loetelu

1 Sissejuhatus

CDP on tarkvara juhtsüsteemide loomiseks. Tegemist on platvormiga, mis ühendab ühes alamvõrgus olevad kontrollerid ühtseks süsteemiks ja toetab mitmeid vajalikke protokolle tööstusseadmetega suhtlemiseks (nt Modbus, CANopen, I2C) [1].

Olemas on ka võimalus luua C++ ja Qt abil kasutajaliidest, et jälgida ja juhtida loodud süsteemi. See aga vajab, et lõppkasutaja paigaldaks oma arvutisse valmis kompileeritud programmi ning seetõttu:

- Uuenduste sisse viimine on tülikas, sest igasse arvutisse tuleb eraldi uus versioon paigaldada.
- Toetatud on ainult Windows ja Linux operatsioonisüsteemidega seadmed. Viimastel aastatel aga on kerkinud esile uusi olulisi platvorme, nt Androidi ja iOS põhised tahvelarvutid ning nutitelefonid.

Teine probleem on see, et kasutajaliidese elemendid on loodud aastaid tagasi ning tänasel päeval näeb nende disain vananenud välja.

Antud töö käigus luuakse ettevõtte CDP Technologies tellimusel uus kasutajaliidese loomise raamistik laienduseks tootele, mis on kodukasutajale tasuta kasutada. Laienduse lähtekood tehakse avalikuks ning kuna kontrolleritelt andmete pärimiseks kasutatakse CDP avalikku API-t, siis teoreetiliselt saaks kogukond pärast magistritöö valmimist selle arenduse üle võtta.

1.1 Ülesande püstitus

Lähteülesandeks on lahendada eeltoodud probleemid luues uue raamistiku, mille abil kasutaja saaks kiirelt ja lihtsalt luua veebipõhise kasutajaliidese oma süsteemile. Sellisel

juhul toetaks see kõiki seadmeid, millel on olemas veebilehitseja. Siinkohal on oluline märkida paar nõuet:

- Veebilehe peaks suutma valmis komponentidest kokku panna inimene, kellel pole suuremat veebiarenduse kogemust, kuna paljud olemasolevad kasutajad on mehaanika või riistvaralähedase programmeerimise taustaga. Võib eeldada baasteadmisi HTML-ist, aga mitte kursis olekut kõikide tänapäevaste veebiarenduse raamistike ja tehnoloogiatega.

- Rõhku tuleb panna andmete visualiseerimisele. Andurite näite (nt mootori kiirus) ja süsteemi olekuid (nt kas uks on kinni või lahti) tuleb kujutada kasutajale arusaadavalt. Kõiki näite peaks saama ka graafiliselt esitada.

- Tulemus peab olema kasutatav nii puuteekaani kui hiirega.

1.2 Metoodika

Et leida parim lahendus antud probleemile, tehakse esmalt ülevaade teistest sarnasel teemal tehtud töödest ning uuritakse konkurentide tooteid. Seejärel täpsustatakse nõudeid vastavalt antud töö eesmärgile ning nendest lähtuvalt valitakse sobivad tööriistad ja teegid, et realiseerida töötav laiendus tootele. Tulemust hinnatakse vastavalt nõuetele ja püstitatud eesmärkidele.

1.3 Ülevaade tööst

Seotud tööde peatükis tehakse ülevaade sarnasel teemal tehtud töödest. Vaadatakse ka juhtsüsteemide vallas tegutsevate konkurentide lahendusi.

Analüüsi peatükis kirjeldatakse pikemalt probleemi tausta ning olemasolevaid võimalusi ja piiranguid. Põhjendatakse, miks käesolev töö vajalik on ning pannakse nõuded uuele süsteemile.

Tehnilise lahenduse peatükis tuleb põhjalikumalt juttu, mille põhjal valiti kasutatavad kolmanda osapoole teegid ja alusraamistik. Selleks võrreldakse erinevaid samalaadseid tooteid ja tehakse valik.

Seejärel kirjeldatakse, kuidas loodi alusraamistikule pakendid, mis panevad erinevad kolmanda osapoole kasutajaliidese teegid suhtlema CDP juhtsüsteemiga, ning tehakse ülevaade, kuidas paistab loodud raamistik kasutajale.

Töö lõpus vaadatakse loodud lahenduse vastavust nõuetele ja püstitatud eesmärkidele ning tuuakse välja võimalikud kitsaskohad. Viimaks kirjeldatakse ideid võimalikeks edasiarendusteks.

2 Seotud tööd

Antud töös on plaanis eeltoodud probleemid lahendada, luues veebipõhise kasutajaliidese tegemise raamistiku, mis kasutaks ettevõtte CDP Technologies enda poolt loodud avalikku liidest. Seega otseselt sama asja varem tehtud pole. Küll aga on kasulik uurida valdkonda ja konkurentide tegevust – juhtsüsteemidele kasutajaliideseid on tehtud varemgi.

2.1 Targa maja lahendused

Üks kiiresti arenev valdkond juhtsüsteemide vallas on targa maja lahendused. Tööstusettevõtted on tihti automaatika vallas konservatiivsed ning kasutavad edasi vanu lahendusi, kui pole otsest põhjust neid vahetada. Teine probleem on see, et süsteemid on disainitud inseneridele, kes on läbinud vastava koolituse, ja seega pole need mõnikord kuigi intuitiivsed [4].

Targa maja lahendused on aga suunatud kodutarbijale, kes valiku tegemisel arvestab ka sellega, et kasutajaliides näeks hea välja [5] ja töötaks kõigi moodsate seadmetega, sealhulgas nutiseadmetega. Klient peaks olema võimeline looma tehniliselt võimeka liidese valdkonda süübimata.

2.1.1 „A Critical Review of User Studies on Healthy Smart Homes“ [5]

Artikkel teeb ülevaate targa kodu lahenduste kasutuskogemustest. Huvitavama tulemusena on välja toodud, et eri kasutajatel on liidesele erinevad ootused. Eriti vanematel inimestel on mõnikord probleeme kasutajaliidese käsitlemisega. Seetõttu peaks liides olema mitmemodaalne, kohandatav inimese oskustele ja vajadustele.

Puudusena toob artikkel välja, et paljud selle valdkonna tööd on jäänud prototüübi tasemele ja seega ei tohiks teha järeldusi kasutuskogemuse pikaajalise mõju kohta.

2.1.2 „Web based solution for smart home functionality extension and control“ [6]

Antud artiklis on otsustatud kasutada veebipõhiseid lahendusi. Positiivne on see, et veebitehnoloogiad ei ole tihedalt seotud konkreetse targa maja platvormiga, vaid lahendused on universaalsed ja hästi dokumenteeritud.

Ka on välja toodud, et veebipõhine kasutajaliides on erinevalt töölauarakendustest kättesaadav pea kõikidel platvormidel ning tõenäoliselt toetab ka seadmeid, mis kasutaja tulevikus soetada võib.

Viidatud artiklis on loodud veebipõhine kasutajaliideste loomise rakendus juhtsüsteemidele, mis suudab suhelda konkreetse tootja seadmetega läbi tagarakendi vahekihi. Mitmeski mõttes sarnaneb loodud süsteem CDP põhimõttele ja sellele loodud esirakendi osa antud töö eesmärgile.

Kuna ükski raamistik ei suuda katta kõiki olemasolevate ja tulevaste koduautomaatika seadmete vajadusi, et palju rõhku pani autor sellele, et loodud rakendus oleks kergesti laiendatav.

2.2 SKA raadioteleskoop

Ruutkilomeeter-rida (Square Kilometer Array, SKA) on plaanitav raadioteleskoop, mida kava kohaselt hakatakse ehitama 2018. aastal. Tegemist pole ühe ainsa teleskoobiga, vaid see koosneb mitmetest antennidest suurel maa-alal [7].

Juba praegu tegeletakse juhtsüsteemide kasutajaliideste prototüüpimisega [8]. Kuna liideseid tuleb palju, siis sarnaselt antud tööle on plaanis luua raamistik nende loomiseks. Kasutatakse nii traditsioonilisi töölauarakendusi, mis on tehtud Java abil, kui ka veebipõhiseid lahendusi. Pärast mõningast prototüüpimist toob töörühm välja:

- JavaScripti raamistikud on kasulikud ja praktilised juhtsüsteemide jaoks, kuid nad pole veel testinud, kui hästi need skaleeruvad.
- Moodsatel JavaScripti raamistikel (React, Angular, Angular 2) pole suurt vahet, kõik võimaldavad vastutusi jagada MVC põhimõtete järgi ja luua taaskasutatavaid komponente.
- Töölauarakendustel on paar eelist: võimalus kliendi arvutisse andmeid salvestada ja teha intensiivsemat töötlust lokaalselt.
- Websocket tehnoloogia (alternatiiv HTTP päringutele) võimaldab suure läbilaskevõimega ja peaaegu reaalajalisi andmevooge. Seega ei jää ka andmeedastus enam veebipõhistel lahendustel pudelikaelaks.

2.3 Konkurendid

Et hinnata antud töö ajakohasust ja vajalikkust, on kasulik vaadata konkurentide sarnaseid lahendusi.

2.3.1 Arduino kasutajaliidesed

Arduino on ettevõte, mis toodab nii riistvara kui tarkvara juhtsüsteemide loomiseks [9]. Seega on konkurent CDP platvormile. Samas pole neil ühtset raamistiku, millega luua graafilisi kasutajaliideseid. Kuna platvorm suudab suhelda väliste seadmetega üle järjestikpordi, kasutatakse erinevaid kolmanda osapoole lahendusi [10]. Paar näidet:

- Jubito – toetab mitmeid riistvaralisi seadmeid, sealhulgas Arduino ja Raspberry Pi. Sisaldab sisseehitatud veebiserverit ja võimaldab programmeerimata lihtsamaid HTML5 põhiseid kasutajaliideseid luua [11].
- MegunoLink – töölaua programm kasutajaliideste loomiseks. Hea monitoorimiseks ja graafikute loomiseks [12].
- Qt – mitmeplatvormiline raamistik kasutajaliideste loomiseks, mis on praegu ka peamine kasutajaliideste loomise vahend CDP süsteemidele. Seda kasutavad näiteks paljud Windowsi ja Linuxi töölauarakendused. On võimalik panna suhtlema ka Arduinoga [13].

Arduino kogemus näitab, et tugev kogukond võib luua palju kasulikku. Seetõttu on ka antud töös plaanis luua avatud lähtekoodiga kasutajaliideste raamistik, mida kogukond saaks vajadusel täiendada ja parendada.

2.3.2 LabVIEW

LabVIEW on süsteemidisaini platvorm, mis sarnaselt CDP-le toetab mitmeid tööstusseadmete protokolle ja võimaldab juhtsüsteemide loomist [14]. Põhiline viis kasutajaliideste loomiseks on endiselt töölauarakendused.

Samas on ka LabVIEW loonud võimaluse luua veebipõhiseid kõhna kliendiga kasutajaliideseid. Küll aga kasutava nad Microsoft Silverlight tehnoloogiat [15], mis tänaseks on hakanud populaarsust kaotama [16], kuna ei toeta nutiseadmeid.

Nutiseadmete abil süsteemi juhtimiseks on loodud Data Dashboard, mis võimaldab graafiliselt luua olemasolevale LabVIEW süsteemile mobiilse kasutajaliidese [17].

2.4 Järeldused

Tundub, et paljudes seotud töödes ja uuemates konkurentide lahendustes on kasutusele võetud kergesti laiendatav veebipõhine kasutajaliides, kuna see on kättesaadav kõigist kasutaja seadmetest ning on kergem sisse viia tarkvarauuendusi võrreldes töölauarakendustega. Tundub ka, et seotud töödes on kogemused veebipõhiste liidestega üldiselt olnud positiivsed: tehnoloogiad on levinud ja hästi dokumenteeritud ning pole probleemi ka jõudlusega. Seega tundub mõistlik ka antud töö teema viia üle CDP platvormi kasutajaliidesed veebipõhistele tehnoloogiatele. Vastasel juhul on oht, et ettevõte „jääb rongist maha“ ning hakkab kliente kaotama.

Ainus negatiivne näide oli LabVIEW veebipõhine liides, mis oli ehitatud kasutades Silverlight platvormi, mis kahjuks ei kogunud piisavalt populaarsust ning mille arendamine on lõpetatud. See näitab, et ka antud töös loodavas lahenduses tuleb analüüsida, kui tulevikukindlad on valitud kolmanda osapoole komponendid.

3 Analüüs

Selles peatükis kirjeldatakse pikemalt probleemi tausta, tehakse ülevaade olemasolevatest võimalustest ning pannakse kirja nõuded uuele süsteemile.

3.1 Taust

CDP on platvorm juhtsüsteemide arendamiseks, mille olulisemad omadused on:

- CDP rakendusprogramm töötab seadmetel, millel jookseb Windows või Linux operatsioonisüsteem. Selleks võib olla tavaline personaalarvuti, kuid ka näiteks miniarvuti Raspberry Pi või mõni tööstuslik kontroller, mis käitab Linux operatsioonisüsteemi.

- Tugi mitmetele tööstusseadmete protokollidele nagu Modbus ja CANOpen. Kodukasutajatele on ilmselt olulisem tugi Raspberry Pi GPIO portidele, mille kaudu saab ühenduda näiteks Sense HAT lisaseadmega, mis sisaldab muuhulgas sensoreid temperatuuri, õhurõhu ja niiskuse mõõtmiseks [18].

- Võime leida üles kõik samas lokaalvõrgus tegutsevad seadmed, kus töötab CDP rakendusprogramm, ning vahetada omavahel andmeid ja saata sõnumeid. Näiteks kui ühe seadme küljes on temperatuurisensor, pääseb iga teine CDP rakendusprogrammiga seade näidule ligi sama kergelt, kui sensoriga seade ise.

- Lisaks andmete lugemisele ja edastamisele saab kasutaja luua oma komponendi, mis töötleb mõnelt sensorilt või tööstusseadmelt loetud andmeid ning reageerib nendele. Näiteks kui temperatuurianduri näit on liiga kõrge, võib komponent saata sõnumi käsuga lülitada küttekeha välja. Lihtsamaid asju saab teha programmeerimata, kuid on ka võimalus kirjutada koodi C++ keeles.

- Kui seadme küljes on ekraan, siis võib sinna teha Qt raamistiku abil kasutajaliidese, millega jälgida ja juhtida süsteemi. Selle kaudu saaks näiteks lõppkasutaja sättida, millist temperatuuri juhtsüsteem ruumis hoidma peaks.

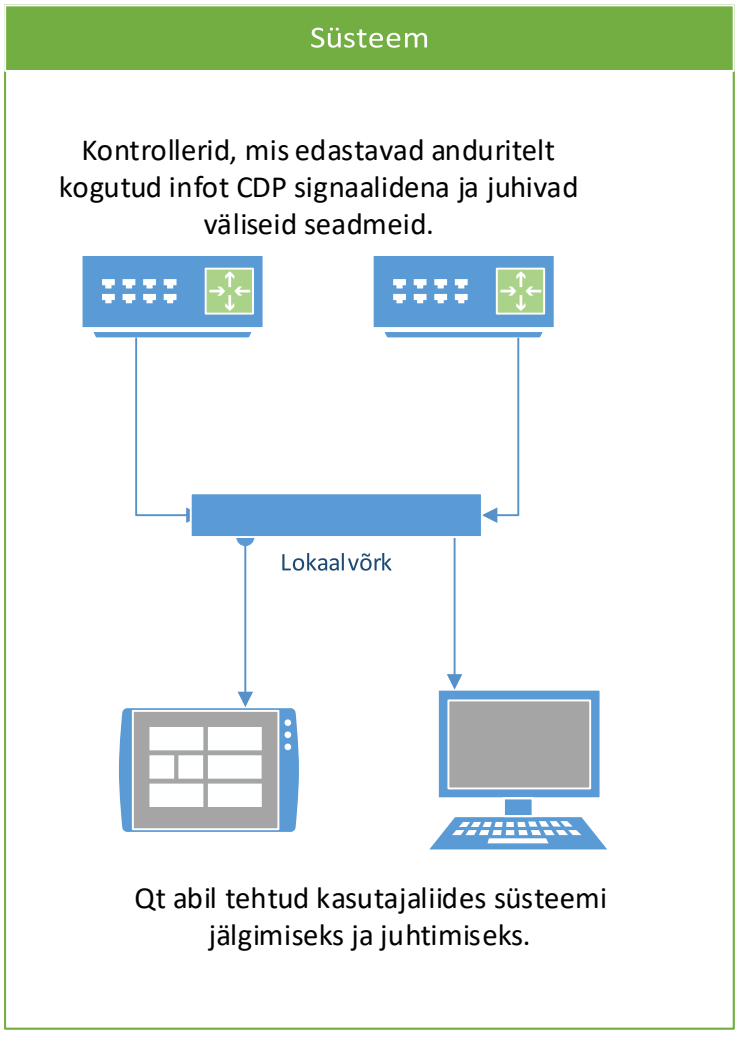
3.2 Olemasolevad kasutajaliidese loomise võimalused

Hetkel on CDP süsteemi jaoks kasutajaliidese loomiseks kaks võimalust. Kasutada võib Qt raamistikku või Javascripti teeki.

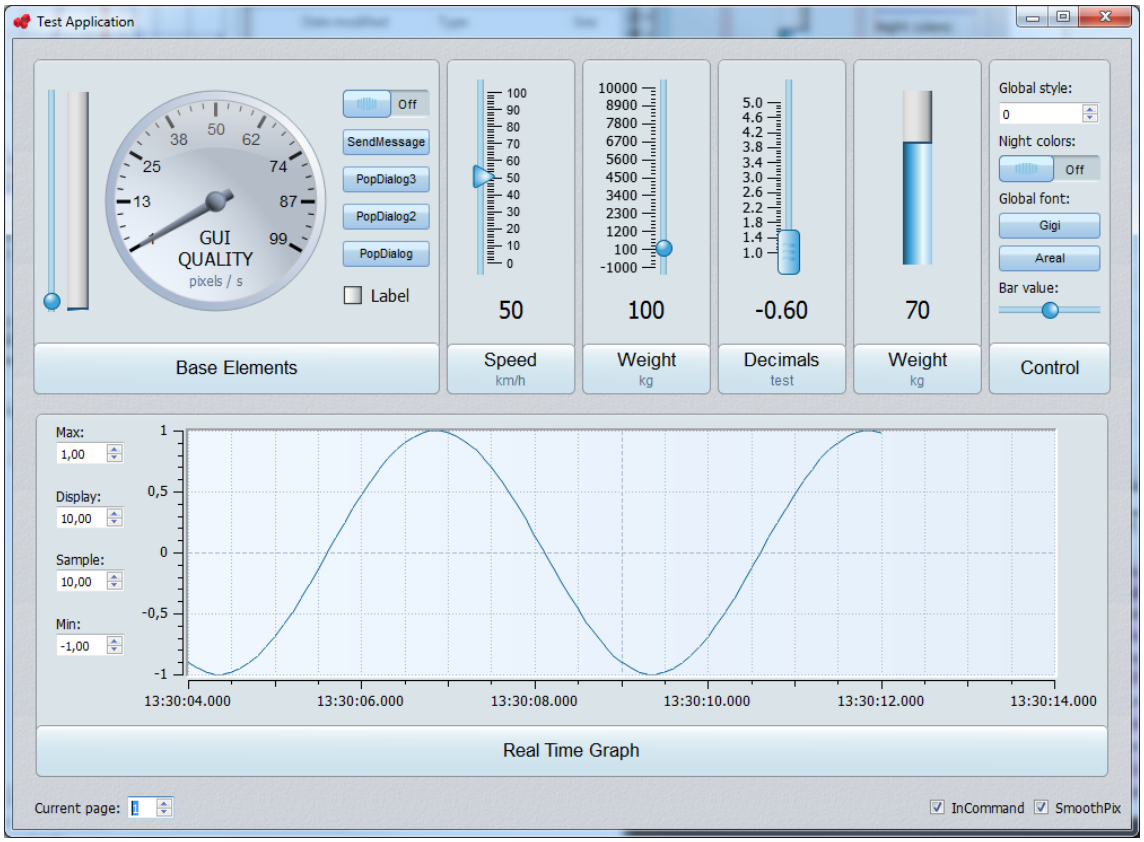
3.2.1 Qt raamistik

Enim kasutatud ja algajale kõige lihtsam on luua kasutajaliides Qt raamistiku abil. CDP Studio arenduskeskkonnas saab liidese teha, lohistades hiirega vormile valmis vidinaid (nt nupud, mõõdikud), mis suudavad suhelda CDP raamistikuga.

Kui vorm on valmis, saab selle CDP Studio arenduskeskkonnas kohe valmis kompileerida ning seejärel võib selle paigaldada mõnesse Windows või Linux operatsioonisüsteemiga arvutisse. Kõik vidinad on kasutatavad nii hiire- ja klaviatuuriga arvutil kui puutetundlikul ekraanil.

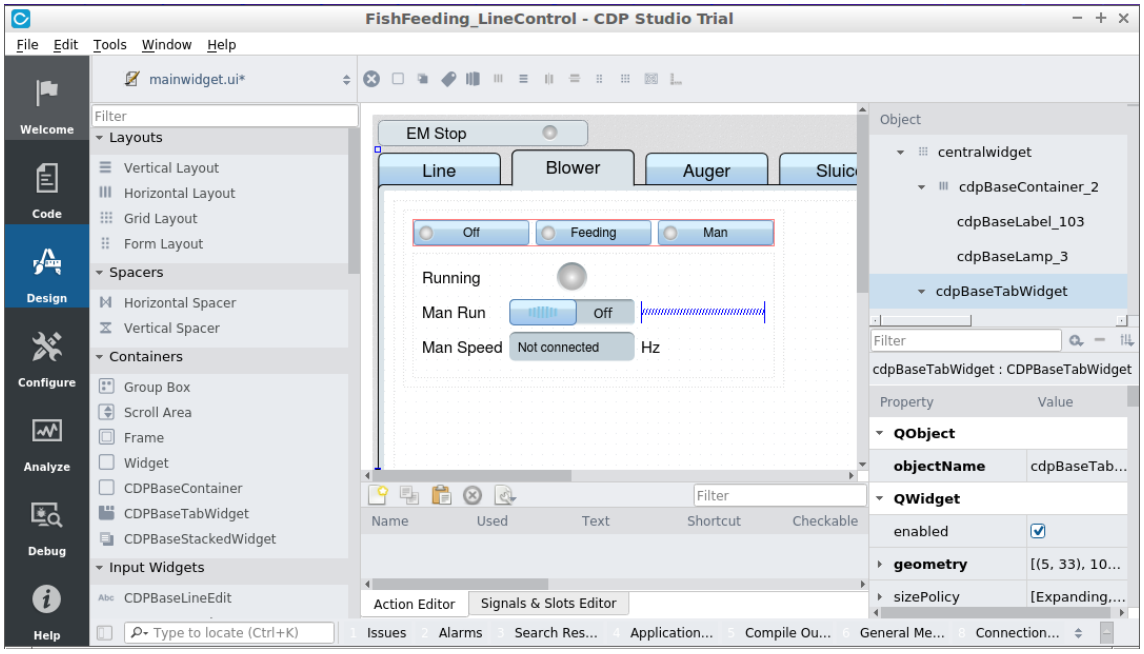


Joonis 1 Juhtsüsteemi arhitektuur



Joonis 2 Näide juhtsüsteemi Qt põhisest kasutajaliidesest

Ülal toodud vormi loomiseks kasutatava tööriista CDP Studio ekraanitõmmis on allpool. Vasakus servas on nimekiri vidinatest: nupud, raamid, lülitid. Neid saab hiirega lohistada vormile ning atribuutide kaudu panna suhtlema CDP süsteemiga.



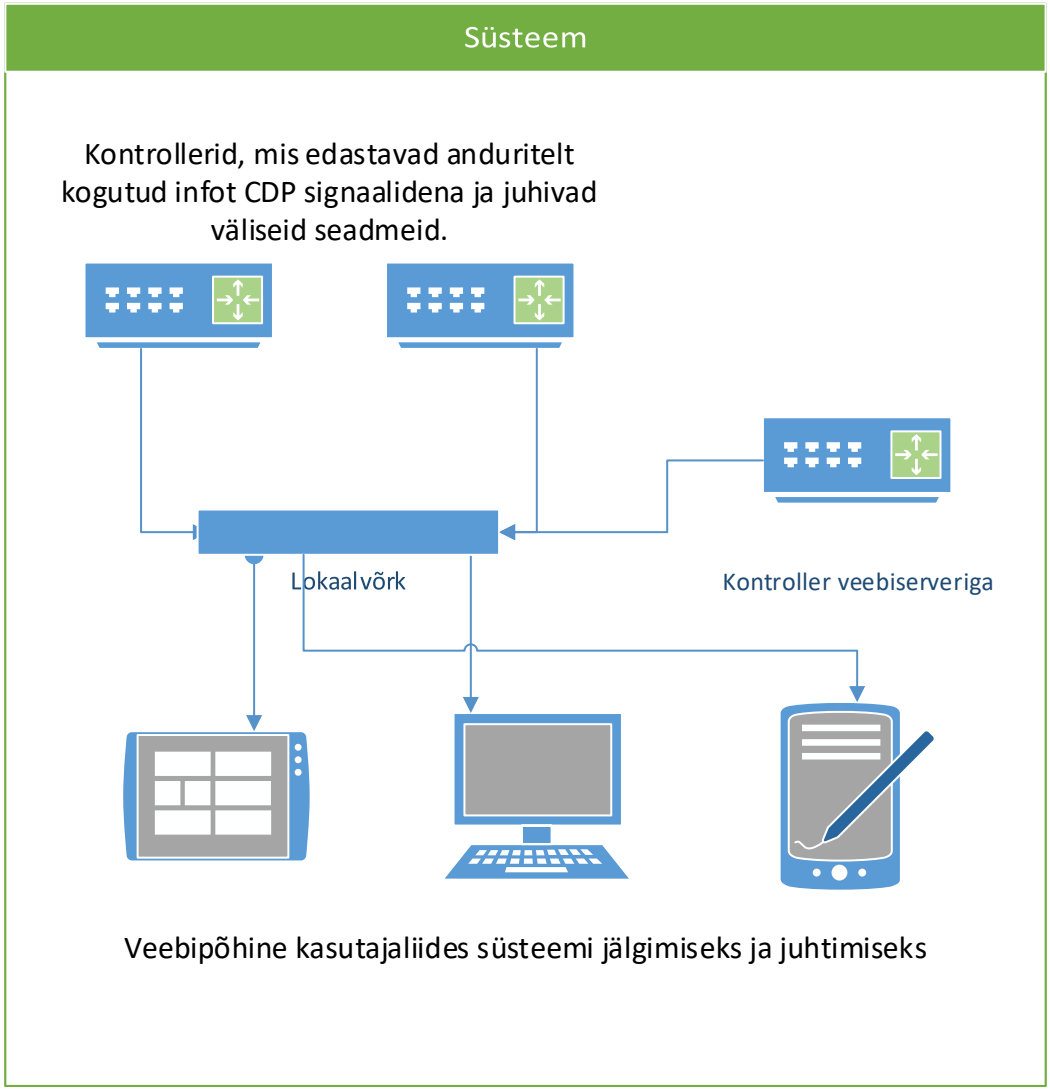
Joonis 3 Qt põhise kasutajaliidese loomine CDP Studio arenduskeskkonnaga

Qt põhine kasutajaliides on küll hästi integreeritud CDP platvormiga, kuid:

- CDP toega vidinad on loodud aastaid tagasi ja välimus on tänaseks veidi vananenud. Disainitööstus on liikunud rohkem lamedamate kasutajaliideste poole (ing k *flat design*).
- Puudub tugi Android ja iOS operatsioonisüsteemiga seadmetele, mis on tänaseks üpriski laia levikuga.

3.2.2 Javascripti teek

CDP rakendusprogrammidega saavad välised programmid suhelda StudioAPI-nimelise liidesele kaudu. Selle kaudu andmete vahetamiseks on tehtud Javascripti teek, mis võimaldab teha veebipõhiseid kasutajaliideseid. Arhitektuur on järgnev:



Joonis 4 Veebipõhise kasutajaliidesega juhtsüsteemi arhitektuur

Veebipõhise kasutajaliidese eeliseks on see, et kliendi seadmesse ei pea paigaldama valmis kompileeritud programmi. Piisab veebilehitseja olemasolust ning hästi loodud veebileht töötab nii Windows ja Linux personaalarvutites kui ka näiteks Android ja iOS nutiseadmetes.

Javascripti teek võimaldab nime järgi otsida kõiki CDP-ga loodud süsteemis liikuvaid signaale ja reageerida nende väärtuse muutusele. Kuid erinevalt Qt põhisest

kasutajaliidesest, peab Javascripti teegi kasutaja ise protseduurilist koodi kirjutama, puudub võimalus hiirega komponente kokku lohistada. Välja näeb see järgmine:

```
<h1 id="myValue">Value goes here</h1>
<script>
    var client = new studio.api.Client(window.location.host);
    client.find("DemoApp.CPULoad").then(function(cpuLoad) {
        cpuLoad.subscribeToChildValues("Value", function(value) {
            document.getElementById("myValue").innerHTML = value;
        })
    });
</script>
```

Joonis 5 StudioAPI Javascripti klient

Ülal olev kood uuendab pealkirjaelemendi h1 väärtust iga kord, kui signaal nimega CPULoad muudab oma väärtust. Tulemus on järgnevalt:

Joonis 6 Pealkirjaelemendile väärtuse andmine

3.3 Motivatsioon

Nagu peatükis 3.2 mainitud, on hetkel kasutajaliidese loomiseks kaks võimalust, millel mõlemal on omad puudused.

Qt põhine vajab igasse seadmesse valmis kompileeritud rakenduse paigaldamist ning puudub iOS ja Android seadmete tugi. Ka on vidinate välimus pisut vananenud.

Veebipõhine lahendus on vaba eeltoodud probleemidest aga annab kasutajale ainult võimaluse küsida CDP rakendustelt signaalide väärtusi, kirjutades protseduurilist koodi. Puuduvad Qt süsteemile sarnased valmis vidinad (nupud, mõõdikud, graafikud), mis teeks kasutajale liidese loomise lihtsaks ja mugavaks.

3.3.1 Kasutajad

Olemasolevad toote kliendid on tihti mehaanika või riistvaralähedase programmeerimise taustaga. Veebitehnoloogiatega on mitmetel neist vähe kogemusi. Hea oleks vähendada barjääri veebipõhise liidese loomiseks. Olemas on Javascripti API, aga selle kasutamiseks:

- Peab hakkama otsima sobivaid vidinaid.

- Paljud Javascripti teegid graafikute ja mõõdikutega ei ole mõeldud dünaamiliste andmete esitamiseks. Isegi näiteks 10 Hz signaali puhul on protsessori kasutus 100% (lähemalt peatükis 4.2).

- Peab olema tuttav *callback* funktsioonidega ja *Promise* objektidega ning oskama kirjutada protseduurilist koodi (näide Joonis 5, lk 23).

Et saada ülevaade tüüpilisest kasutajast, küsitleti kuute olemasolevat klienti:

- 83% inimestest väitis, et saab aru lihtsamast HTML-st, ning 100% saab aru lihtsamast Javascript koodist.

- *Callback* funktsioonidega on tuttav kaks kolmandikku, *Promise* objektidest on kuulnud kolmandik, aga kasutanud pole keegi.

- Pool vastanutest on kasutanud JQuery teeki. Kolmandik ka vähemalt ühte uuematest MVC-laadsetest Javascripti raamistikest (valikus oli Angular, React ning Vue.js)

Küsitlus näitas, et keskmine CDP kasutaja on tuttav HTML ja Javascripti põhitõdedega, aga vaid vähemus on kursis viimastel aastatel populaarsust kogunud raamistikega ja hiljuti keelde lisatud *Promise* objektidega. Sellest võib järeldada, et veebitehnoloogiaid on kunagi õpitud, aga pole enam mõnda aega rakendatud. See võib olla tõsine barjäär peatükis 3.2.2 mainitud Javascripti teegi abil veebipõhise kasutajaliidese loomiseks.

Ka kodukasutaja võib olla pigem algaja veebidisainis, aga tahaks kiiresti ja mugavalt midagi lihtsat kokku panna, nt koduilmajaam Raspberry Pi miniarvuti ja Sense HAT lisamooduli abil. Aitaks valmis komponendid, kus kasutaja saaks deklaratiivselt väljendada, mida tal vaja on (nupp, graafik vms) ja mille külge CDP süsteemis see ühendub. Ehkki kodukasutajad otseselt raha ettevõttele ei maksa, suurendab nende kaasamine toote tuntust ja populaarsust. Toimiv kogukond võib hakata lahendust edasi arendama.

Ärikasutajad aga näeks antud töö käigus valmis tehtud komponentide abil, et on lihtne alustada veebipõhise liidese loomisega ning arendajad saavad need näiteks võtta, kui on vaja keerulisemaid asju tegema hakata.

3.4 Nõuded uuele süsteemile

Antud töö eesmärgiks on lihtsustada veebipõhise kasutajaliidese loomist. Selleks on plaanis luua valmis komponendid, mille abil kasutaja saaks deklaratiivselt kirjeldada juhtsüsteemi elemente ning pääseks protseduurilise koodi kirjutamisest CDP süsteemi juhtimisel.

3.4.1 Funktsionaalsed

1. Raamistik peab sisaldama vasteid järgmistele Qt põhise liidese vidinatele:
    - Nupp. Vajutades saadetakse sõnum mõnele CDP rakendusprogrammile.
    - Lüliti. Näitab tõeväärtussignaali olekut. Vajutuse peale muudab olekut.
    - Mõõdik. Osutiga näidik, mis kuvab numbrilist väärtust.
    - Graafik. Joondiagramm, mis näitab reaalajas signaali väärtuse muutumist.
    - Tekstiväli. Sõne või numbri sisestamiseks ja kuvamiseks.
    - Silt. Sõne kuvamiseks.
2. Kõik ülal loetletud komponendid peavad suutma suhelda CDP raamistikuga.
3. Vidinate värvi ja suurust peab saama muuta.

3.4.2 Mittefunktsionaalsed

1. Kõik komponendid on kasutatavad nii hiire kui sõrmega ja töötavad nii arvutites kui nutiseadmetes.
2. Klientide jaoks tasuta kasutada, ka ärilistel eesmärkidel. Seetõttu ei saa kasutada kolmanda osapoole teeke, mis on piirava litsentsiga või tasulised. Probleem on selles, et tasuliste teekide jaoks ei pea litsentsi ostma mitte raamistiku looja ehk antud töö autor, vaid lõppkasutaja, kes üritab mõnda lihtsamat süsteemi selle abil

teha. Sellisel juhul oleks loodaval raamistikul oluliselt raskem konkureerida olemasoleva Qt põhise lahendusega, mida saab tasuta kasutada.

3. Kui 10 vidinat on seotud 10 Hz signaaliga, siis protsessori kasutus ei tohi ületada 10%.

    a. Antud töös mõõdetakse protsessorikasutust arvutiga, kus on 1,7 GHz Intel 3317U protsessor (toodetud 2012. aastal, maksimaalne energiatarve 17 W) [19].

    b. Piirangut on vaja, sest liigne protsessorikasutus vähendab masina jõudlust ning mobiilsete seadmete puhul ka akukestvust. Piirang tagab, et loodud juhtsüsteemi kasutajaliides oleks sujuv ka nõrgema jõudlusega seadmetes, nt nutitelefonis.

    c. Lihtsamate vidinate puhul, nt silt või nupp, võiks protsessorikasutus samadel tingimustel olla umbes 1%. Üldine piirang on valitud leebem, et mitte liigselt piirata keerulisemate vidinate valikut, näiteks reaalaja andmeid näitav graafikuteek.

3.4.3 Muud eesmärgid

1. Luua deklaratiivne viis veebipõhise kasutajaliidese loomiseks. Komponente peab saama kasutada kirjutamata protseduurilist koodi.

2. Hoida uut raamistikku võimalikult õhukesena ja maksimaalselt kasutada ära kolmanda osapoole teekide võimalusi. Viimane on vajalik, et raamistiku ajakohasena hoidmiseks kuluks võimalikult vähe ressurssi. Veebidisaini maailm areneb kiirest ja hea on muutustega kaasas käia kulutamata liigselt aega. Tähtis on hoida stiil ja funktsionaalsus eraldi.

3. Kasutaja peaks olema suuteline liidese valmis tegema nii, et ta ei pea teadma midagi modernsetest Javascripti raamistikest nagu Angular, React või Vue.js. Piisaks baasteadmistest HTML kohta.

3.4.4 Turvalisus

Turvalisuse tagamine ei ole osa antud töö skoobist, sest piirdutakse kliendipoolse kasutajaliidesega. Autentimine vajaks suletud lähtekoodiga CDP serveripoolse osa täiendamist.

Et CDP süsteemi ning antud töö tulemust turvaliselt kasutada, seadistatakse juhtsüsteem järgnevalt:

- CDP suhtleb ainult lokaalvõrgus olevate seadmetega. Sama piirang laieneb ka käesoleva töö raames loodavale veebipõhisele kasutajaliidesele. Üldiselt CDP juhtsüsteemi avalikku Internetti ei ühendata.

- Kui on vaja kaugligipääsu CDP süsteemile, siis tehakse seda läbi VPN või SSH tunneli. Sel juhul tagavad nimetatud lahendused turvalisuse.

- Kui tulevikus laiendatakse CDP-ga suhtlemiseks mõeldud protokolli autentimisega, siis on võimalik ka käesolevas töös loodud vidinaid kasutada turvaliselt üle avaliku Interneti.

4 Tehniline lahendus

Selles peatükis on kirjeldatud, mille põhjal valiti kasutatavad kolmanda osapoole teegid ja alusraamistik. Selleks võrreldakse erinevaid samalaadseid tooteid ja tehakse valik.

4.1 Lahenduse idee

Teha Angular või muu sarnase raamistiku (nt Vue.js) abil valmis direktiivid, mille abil kasutaja saaks palju lihtsamalt oma juhtsüsteemile kasutajaliidese teha.

Näiteks pealkirjaelemendile *h1* väärtuse andmine võiks välja näha järgmiselt.

```
<h1><cdp-label routing="webApp.CPULoad"/></h1>
```

Joonis 7 Pealkirjaelement, mis näitab signaali väärtust

Ülal toodud märgend teeb täpselt sama, mis peatükis 3.2.2 toodud kood (Joonis 5, lk 23). Elemendi *h1* väärtus uueneb dünaamiliselt iga kord, kui signaali CPULoad väärtus muutub.

Selline tulemus tähendaks, et veebiliidese tegemisega saaks hakkama iga inimene, kes on kokku puutunud HTML-iga. Puudub vajadus teada midagi Javascript keele *callback* funktsioonidest ja *Promise* objektidest. Piisab sellest, kui deklareerida, mida vaja on.

Et loodava raamistiku hooldusele kuluks võimalikult vähe vaeva, on eesmärk ära kasutada võimalukult palju kolmanda osapoole tehnoloogiad (eesmärk 2 peatükis 3.4.3).

4.2 Graafikute teegi valimine

Nagu mainitud nõuete peatükis 3.4, peaks saama kõiki CDP süsteemis liikuvaid numbrilisi väärtusi esitada graafiliselt reaalajas muutuva joondiagrammina. Teegi valikul tuleb silmas pidada, et litsents lubaks seda ka ärilistel eesmärkidel tasuta kasutada ning et graafiku dünaamilisel uuendamisel ei oleks protsessori kasutus liiga suur.

Järgnevalt on tehtud ülevaade populaarsematest turul leiduvatest graafiteekidest, mille litsents on antud töö jaoks sobiv.

4.2.1 Google Charts

Litsents: Creative Commons Attribution 3.0, mis lubab toodet levitada ja kasutada, ka ärilistel eesmärkidel [20].

Tegelikult pakub Google mitut erinevat tüüpi graafikut. Antud töö kontekstis on huvitavamad neist Annotation Chart ja Line Chart [21].

Annotation Chart

Joondiagramm, mis on mõeldud ajalooliste aegandmete kuvamiseks. All servas on abiriba RangeSelector, kus on väikselt kuvatud kogu graafik ning mille abil saab valida peagraafikul nähtavate andmete alg- ja lõppaja.

Joonis 8 Google Annotation Chart. Alumises servas on näha abiriba RangeSelector.

Plussid:

- Hea andmete analüüsimiseks. Võimaldab kuvada andmeid konkreetsest ajaperioodist.
- Hiirega peale liikudes näitab väärtust konkreetses asukohas.

Miinused:

- Abiriba RangeSelector ei ole kasutatav puutetundliku ekraaniga (ei vasta mittefunktsionaalsele nõudele 1).

- Ei sobi reaalajaandmete kuvamiseks. Iga kord, kui lisada graafikule üks punkt, joonistatakse kogu graafik uuesti ning 10 Hz signaali puhul on ühe lõime protsessori kasutus maksimaalne (ei vasta mittefunktsionaalsele nõudele 3).

Line Charts

Joondiagramm andmete kuvamiseks.

Plussid:

- Hiirega peale liikudes näitab väärtust konkreetses asukohas.
- Võib lisada eraldi joone, mis näitab andmete trendi.

Miinused:

- Ei vasta 3. mittefunktsionaalse nõudele (protsessorikasutus). Parima tulemuse saab, kui animatsioonid on välja lülitatud (1 graafik – 15% protsessorikasutust). Sel juhul aga on graafik, kuhu reaalajas andmed lisanduvad, üpriski hüplik.

4.2.2 D3 ja C3 teegid

D3 on Javascripti teek andmete visualiseerimiseks [22]. C3 on loodud, et lihtsustada D3 teegi kasutamist [23]. Üks osa C3 teegist on ka joondiagramm.

Litsents: D3 teegil BSD 3-clause ja C3 teegil MIT. Mõlemad on üpriski lubavad litsentsid ja sobivad antud töö jaoks [24] [25].

Plussid:

- Kuna C3 on üpriski populaarne teek, on sellele olemas pakendid, et seda paremini integreerida teiste Javascript raamistike ja teekidega. Näiteks Angular C3 Simple, mis on loonud direktiivid, et lihtsustada teegi kasutamist koos Angular raamistikuga [26].

Miinused:

- Sujuv kerimine uute andmete peale tulles on võimalik animatsioonidega [27]. Kahjuks eeldab see meetod, et uued andmepunktid tulevad fikseeritud intervalliga, vastasel juhul ei ole võimalik määrata õiget animatsiooni kestvust.

CDP raamistik aga saadab uue infopunkti ainult siis, kui väärtus on muutunud, seega nõuaks CDP raamistikuga integreerimine vahekihti, mis tekitaks aeg-ajalt andmepunkte ka siis, kui väärtus ei ole muutunud.

- Protsessorikasutus ei vasta 3. mittefunktsionaalsele nõudele. Vaid 1 graafiku ja ühe signaaliga on protsessorikasutus 9%, kahe graafikuga juba 21%.

4.2.3 RGraph

SVG põhine graafikuteek, mis eraldi toetab ka reaalajaandmete kuvamist [28]. Litsentsiks on MIT, mis on antud töö jaoks sobiv [25].

Plussid:

- Lihtne kasutada, kodulehel hea näide.

Miinused:

- Protsessorikasutus ei vasta 3. mittefunktsionaalsele nõudele. Vaid 2 graafikuga on juba protsessorikasutus 20%.

4.2.4 Smoothie Charts

Graafikuteek, mis on mõeldud sujuvaks reaalajaandmete kuvamiseks [29]. Litsentsiks on MIT, mis on antud töö jaoks sobiv [25].

Plussid:

- Väike protsessorikasutus.
    - 10 graafikut 1 signaaliga: 8%
    - 1 graafik 10 signaaliga: 4%
- Sujuv kerimine.
- Saab lisada viite, et graafiku paremas servas ei oleks näha, kui andmepunktid võrgu latentsuse tõttu viivitusega saabuvad.
- Kuna on mõeldud reaalaja andmete jaoks, on uue andmepunkti lisamine väga lihtne. Kui teistel antud töös kirjeldatud graafikuteekidel tuleb selle jaoks oma

puhvrit hallata ning andmepunkte sealt ka õigel ajal eemaldada, siis Smoothie Charts teek haldab seda ise.

Miinused:

- Puudub tugi selleks, et hiirega peale liikudes näitaks konkreetse andmepunkti väärtust.

- Komponendi laius tuleb määrata pikslites ning see ei kohandu automaatselt seadme ekraani laiusega (seotud mittefunktsionaalse nõudega 1). Samas on probleemist kerge üle saada, kui ise lisada sündmusevaatur, mis ekraani suuruse muutudes Smoothie Charts graafiku laiust muudab.

- Puudub legend, mis paneks vastavusse joone värvuse ja kuvatava väärtuse nime. Kuid ka seda oleks kerge ise juurde teha.

4.2.5 Järeldused

Tundub, et Javascripti maailmas pole ühte ideaalset lahendust reaalaja andmete graafiliseks kuvamiseks. Tuleb valida, kas eelistada paljude võimalustega graafikuid, mis toetavad näiteks andmete trendi näitavat lisajoont ja hiirega peale liikudes näitavad signaali täpset väärtust, kuid dünaamiliselt uuenedes kasutavad liigselt protsessorit. Teine võimalus on võtta ainult reaalajaandmete jaoks mõeldud graafikuteek, mis on sujuv ja saab suurepäraselt hakkama pidevalt muutuvate andmetega, kuid toetab vähem võimalusi.

Antud töös on otsustatud kasutada viimast varianti, sest:

- Madal protsessorikasutus tagab parema jõudluse. Mobiilsetel seadmetel parandab ka akukestvust.

- Sujuv graafik näeb subjektiivselt parem välja, kui hüplik animatsioonideta graafik.

- Reaalaja andmete jaoks mõeldud graafikuteeki on lihtsam kasutada CDP süsteemist tulevate andmetega. Pole vaja luua keerukat vahekihti, mis haldaks graafiku puhvrit ja looks ise võrdse intervalliga andmepunkte sujuvuse tagamiseks.

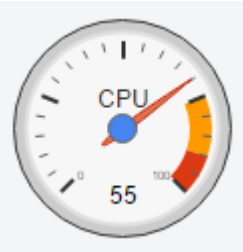
Seepärast kasutatakse antud töös Smoothie Charts teeki.

4.3 Mõõdikute teegi valimine

Sarnaselt graafikute teegile ei kannata ka osa mõõdikuid 10 korda sekundis näidu uuendamist. Järgnevalt on võrreldud erinevaid teeke jõudluse, võimaluste ja välimuse seisukohalt.

4.3.1 Google Charts Gauge

Google Charts teek sisaldab ka mõõdikut [21]. Litsents Creative Commons Attribution 3.0 on sobilik [20]. Välja näeb see järgmiselt:

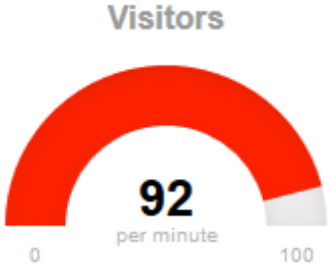


Joonis 9 Google Charts Gauge mõõdik

Testide järgi töötab hästi, aga protsessorikasutus kümne 10 Hz sagedusega uueneva graafikuga on umbes 20%. Kui animatsioonid välja lülitada, langeb see umbes 14% juurde.

4.3.2 JustGage

Modernse välimusega mõõdikuteek. Kasutab vektorgraafikat, nii et näeb hea välja iga resolutsiooniga [30]. Litsents MIT on sobilik [25]. Välimus on järgnev:

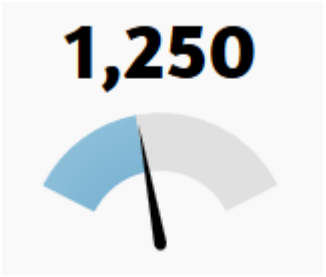


Joonis 10 JustGage mõõdik

Animatsioonidega on kahjuks protsessorikasutus üpriski kõrge, isegi üks mõõdik 10 Hz uuenemissagedusega võtab 30% protsessorist. API kaudu animatsioone kahjuks välja lülitada ei saa, ainult muutes teegi lähtekoodi [31].

4.3.3 Gauge.js

Samuti modernse välimusega ja resolutsioonist sõltumatu mõõdikuteek. Erinevalt JustGage teegist on sellel võimalus animatsioonid välja lülitada [32]. Litsents MIT on sobilik [25]. Välja näeb see järgnevalt:



Joonis 11 Gauge.js mõõdik

Juhul kui animatsioonid on välja lülitatud, on protsessorikasutus üpriski madal, ka kümne mõõdiku puhul umbes 1,5%.

4.3.4 Järeldused

Sarnaselt Javascripti graafikuteekidega on ka mitmetel mõõdikutel probleeme suure sagedusega dünaamiliselt uuenevate andmetega. Testitud teekidest ainsana vastas 3. mittefunktsionaalsele nõudele protsessorikasutuse osas Gauge.js, seda ka ainult siis kui animatsioonid olid välja lülitatud.

Seepärast kasutatakse antud töös Gauge.js teeki.

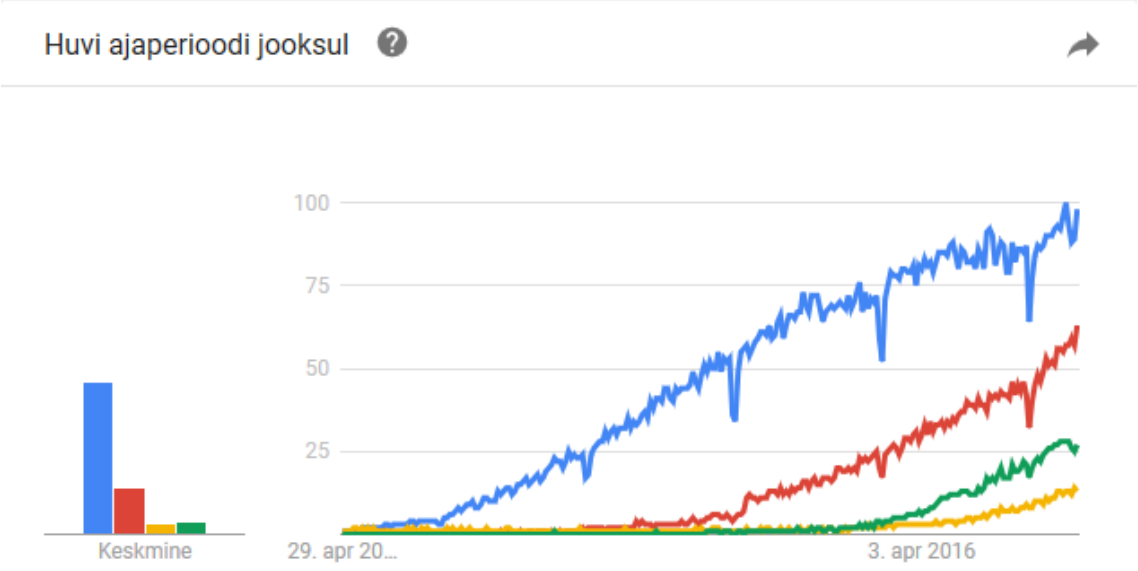
4.4 Alusraamistiku valimine

Et saada tulemust, mida kirjeldab Joonis 7 peatükis 4.1 „Lahenduse idee“, on vaja kasutada alusraamistikku, mis võimaldaks luua omaenda HTML elemente, mida saaks kasutada koos tavaliste HTML5 elementidega. Sellist võimalust pakuvad mitmed raamistikud, näiteks Angular, React ja Vue.js.

Nagu peatükis 2.2 viidatud SKA raadioteleskoobi projekti kasutajaliidese prototüüpimist kirjeldavas artiklis öeldud, siis raamistikelt suurt vahet pole – kõik nad võimaldavad

komponentide põhist arhitektuuri, mis on antud töö jaoks vajalik. Samas üks neist tuleb valida.

Hea kriteerium on populaarsus. Üks võimalus seda mõõta on tööriista Google Trends abil, mis näitab kui palju valitud termineid otsitakse. Valides valdkonnaks Programmeerimine ning ajaperioodiks 5 aastat on tulemus järgnev.



Joonis 12 Google Trends. Sinine – Angular, punane – React, roheline – Angular 2, kollane – Vue.

Nagu näha, on selgelt kõige populaarsem sinisega märgitud Angular, kuid ka punase joonega React kasvab päris kiiresti.

Antud töös on kasutatud Angular raamistiku esimest versiooni, sest:

- See on kõige populaarsem, mis tähendab, et:
    - On rohkem inimesi, kes on sellega tuttavad.
    - On saada väga palju just Angular raamistiku jaoks kohandatud teeke.
    - Väiksem oht, et projekt hüljatakse lähiajal kasutajate vähesuse tõttu.
- Kandidaadiks oli ka Angular 2, mis on küll uuem, aga erinev esimest versioonist. Põhiline probleem on see, et enamus näited ei ole mitte Javascriptis, vaid kasutatud on vähem tuntud Typescript keelt, mille õppimine võib olla liigne barjäär peatükis 3.3.1 mainitud sihtrühmale.

Valitud raamistik Angular on kaitstud MIT litsentsiga, mis lubab seda tasuta kasutada, ka ärilistel eesmärkidel [25].

4.5 Kasutajaliidese välimus

Et anda HTML põhielementidele – tabelitele, nuppudele jne – moodsam välimus, kasutatakse selleks eraldi teeke. Kuna käesoleva töö peamine eesmärk on pakkuda kasutajaliidese elemente uue funktsionaalsusega, mis lubaks neil suhelda CDP süsteemiga, siis välimuse teegi valimisele suurt rõhku ei panda. Rohkem keskendutakse sellele, et disainitrendide muutudes oleks välimuse teek kergelt asendatav.

4.5.1 Angular raamistiku jaoks kohandatud teegid

Kuna peatükis 4.4 valitud Angular alusraamistik on üpriski populaarne, on sellele kohandatud mitmeid välimust muutvaid teeke. Üks näide on AngularJS Material [33].

Nupu deklareerimine selle teegi abil käib järgnevalt.

```
<md-button class="md-raised"> Raised Button </md-button>
```

Joonis 13 AngularJS Material nupp

Kui peaks olema vaja asendada seda tüüpi nupp mõne teisega, tuleb muuta kahte asja:

- Elemendi nime, sest *md-button* on AngularJS Material spetsiifiline.
- Atribuuti *class*, kus tuleb asendada *md-raised*.

4.5.2 Angular raamistikust sõltumatud teegid

Teine võimalus on kasutada üldiseid kasutajaliidese välimuse teeke, mis pole seotud ühegi kindla Javascript alusraamistikuga. Heaks näiteks on Bootstrap ja Semantic UI, mis mõlemad kasutavad CSS-i, et anda olemasolevatele elementidele uus välimus [34] [35].

Näiteks Semantic UI ja Bootstrap nupud on kirjeldatud järgnevalt.

```
<button class="ui primary button"> Save </button>
<button class="btn btn-primary">   Save </button>
```

Joonis 14 Nupu välimuse deklareerimine CSS abil

Seega, et asendada üks CSS-põhine raamistik teisega, tuleb vaid muuta *class* atribuuti.

4.5.3 Välimuse teegi valik

Üks antud töö eesmärkidest on hoida välimus lahus funktsionaalsusest. Eelneva põhjal võib öelda, et parem on seda teha, kui kasutada Angular raamistikust sõltumatuid teeke. Veelgi enam, kui lihtsamad käesoleva töö käigus loodavad komponendid vaid pakendavad tavalisi HTML elemente, siis võib *class* atribuudi abil kasutaja ise määrata, millist CSS-põhist teeki ta eelistab.

Idee on järgnev:

```
<cdp-button routing="..." class="ui primary button"> Save </button>
<cdp-button routing="..." class="btn btn-primary">   Save </button>
```

Joonis 15 CDP raamistikuga seotud nupu välimuse muutmine CSS abil

Ülemisel real on kasutatud Semantic UI teeki, alumisel Bootstrap teeki. Kui jätta teegi valik kasutaja ülesandeks, siis antud töö käigus loodavad komponendid isegi ei pea teadma, millise välimusega nad on.

Järgnevate peatükkide näidetes kasutatakse Semantic UI teeki.

4.6 Kasutatavad tehnoloogiad

Kokkuvõtvalt kasutatakse antud töös järgnevaid tehnoloogiad:

- Alusraamistik – Angular
- Graafikuteek – Smoothie Charts
- Mõõdikuteek – Gauge.js
- Välimuse teek – Semantic UI (kergelt asendatav)

5 Raamistik kasutaja vaatest

Et kasutajal oleks lihtne alustada, tuleb raamistikuga kaasa mall, mis sisaldab vajalikke teeke ja index.html faili, kus on deklareeritud kõik sõltuvused. Sellisel juhul võib kasutaja malli HTML *body* elemendi alla hakata Angular raamistikust midagi teadmata lisama järgnevalt kirjeldatud märgendeid (nt nupud, lülitid).

Raamistikku saab kasutada kahte moodi:

1. Kui kogemus piirdub HTML-iga, siis võib lihtsalt kasutada antud töö käigus loodud elemente.

2. Kogemusega veebiarendaja võib elemente lisada ja täiendada. Vajadusel pöörduda ka otse StudioAPI teenuse poole, et suhelda CDP süsteemiga. Abiks on talle antud töös realiseeritud raamistiku avatud lähtekood.

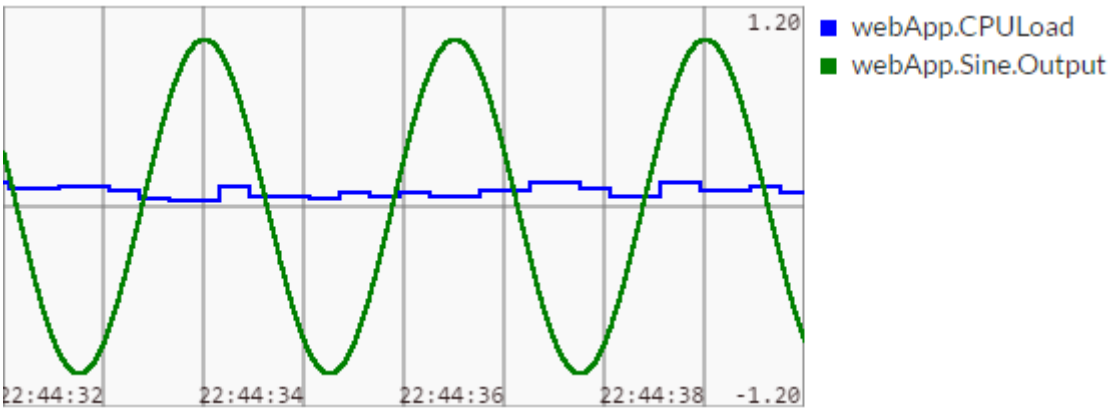
5.1 Graafik

Et kuvada graafikut, mis näitab kahte reaalajas uuenevat signaali, piisab, kui kasutaja deklareerib graafiku elemendi, määrab selle suuruse ja lisab väärtused, mida kuvama hakatakse.

```
<cdp-chart width="400" height="200">
    <cdp-chart-line routing="webApp.CPULoad"></cdp-chart-line>
    <cdp-chart-line routing="webApp.Sine.Output"></cdp-chart-line>
</cdp-chart>
```

Joonis 16 Graafik kahe reaalajas muutuva signaaliga

Need read tekitavad järgneva graafiku.



Joonis 17 Graafik kahe signaaliga

5.1.1 Täiendused

Graafiku komponent baseerub Smoothie Charts teegil ning *options* atribuudi kaudu saab kasutaja muuta kõiki selle teegi valikuid. Kuid et lihtsustada selle kasutamist, on antud töö käigus lisaks teegi pakendamisele lisatud järgnev funktsionaalsus:

- Kohanduv laius. On lisatud sündmusevaatur, mis muudab graafiku ala kitsamaks või laiemaks, kui veebilehitseja suurus muutub. Nii näeb graafik hea välja nii väiksel telefoniekraanil kui ka lauaarvutis.

- Legend (nähtav Joonis 17 paremas servas). Paneb vastavusse joone värvi ja kuvatava väärtuse nime.

- Graafikule väärtusi lisades antakse igale joonele automaatselt eri värvus, kui kasutaja teisiti ei määra. Otse Smoothie Charts teeki kasutades tuleb värvus alati käsitsi määrata.

- Y-telje vahemiku määramise funktsioon. Smoothie Charts võimaldab kasutajal ette anda oma funktsiooni, mis dünaamiliselt määrab y-telje vahemiku vastavalt graafikul olevatele väärtustele. Et juhtsüsteemi jälgimine liigselt segadust ei tekitaks, kasutatakse antud töös funktsiooni, mis lubab y-telje ülemisel ja alumisel piiril kasvada, aga mitte kahaneda. Nii paistavad signaali ekstreemumid hästi välja ning kasutaja ei ehmata pisikese kõikumise peale (vaikimisi funktsiooniga võib vahemik ka kahaneda).

5.1.2 Atribuudid

Elementi *cdp-chart* saab muuta järgmiste atribuutidega.

Tabel 1 Elemendi cdp-chart atribuudid

Atribuut	Kirjeldus
width (kohustuslik)	Graafiku maksimaalne laius
height (kohustuslik)	Graafiku kõrgus
options	Objekt, mis sisaldab Smoothie Charts spetsiifilisi valikuid, näiteks tausta värv, teksti värv jne.
display-legend	Vaikimisi tõene. Kui väär, siis ei kuvata paremas servas legendi, mis näitab milline väärtus millise värviga seotud on.

Elemendi *cdp-chart* sisse tuleb lisada üks või mitu *cdp-chart-line* elementi, mis kirjeldavad graafikul kuvatavaid väärtusi.

Tabel 2 Elemendi cdp-chart-line atribuudid

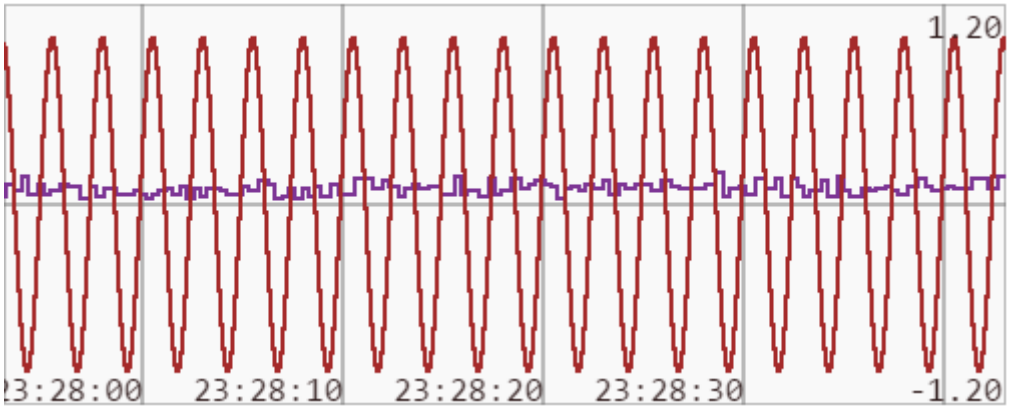
Atribuut	Kirjeldus
routing (kohustuslik)	CDP süsteemis liikuv väärtus, mida kuvada.
color	Võimaldab määrata kuvatavale väärtusele käsitsi värvuse. Atribuudi puudumisel valitakse väärtustele automaatselt erinevad värvused.

Näide atribuutide kasutamisest.

```
<cdp-chart width="500" height="200" display-legend="false" options='{
                "labels": { "fontSize": 17 },
                "millisPerPixel": 100,
                "grid": { "millisPerLine": 10000 }
            }'>
    <cdp-chart-line routing="webApp.CPULoad" color="#7d3595"></cdp-chart-line>
    <cdp-chart-line routing="webApp.Sine.Output" color="brown"></cdp-chart-line>
</cdp-chart>
```

Joonis 18 Graafikukomponendi kohandamine

Ülal on graafiku legend välja lülitatud, tekst muudetud suuremaks, kuvatav ajaperiood pikemaks ja vertikaalsete joonte vahe suuremaks. Mõlemale joonele on antud värvus käsitsi. Kuvatud on täpselt samu signaale, mis Joonis 17 peal.



Joonis 19 Kohandatud graafik

Kõiki võimalusi graafiku kohandamiseks võib lugeda Smoothie Charts kodulehelt [29].

5.2 Mõõdik

Mõõdiku direktiivil peab kasutaja ainult deklareerima, millist signaali näidata.

```
<cdp-gauge routing="webApp.CPULoad"></cdp-gauge>
```

Joonis 20 Märgend mõõdiku kuvamiseks

Tulemus on järgnev:

Joonis 21 Mõõdik

5.2.1 Täiendused

Et näidu väärtus paremini välja paistaks, on lisatud mõõdiku alumisse serva väärtuse numbriline näit.

5.2.2 Atribuudid

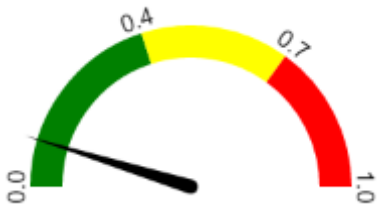
Elemendi *cdp-gauge* atribuudid on järgmised.

Tabel 3 Elemendi cdp-gauge atribuudid

Atribuut	Kirjeldus
routing (kohustuslik)	CDP süsteemis liikuv väärtus, mida kuvada.
width	Mõõdiku laius. Kõrgus tuletatakse laiuse põhjal

options	Objekt, mis sisaldab kasutatud teegi Gauge.js spetsiifilisi valikuid. Nende abil saab muuta värvusi, nurka ja palju muud [32].
min-value	Skaala vasak serv, sellest madalamaid väärtusi kuvada ei saa.
max-value	Skaala parem serv, sellest suuremaid väärtusi kuvada ei saa.
display-text-field	Vaikimisi tõene. Kui väär, siis ei kuvata mõõdiku alumises servas numbrilist väärtust.

Atribuutide abil saab näiteks muuta mõõdiku kaare nurka, luua eri värvidega sektsioone, lisada numbrilisi näite kaare peale ja palju muud. Lõplik nimekiri on kättesaadav Gauge.js dokumentatsioonist [32].



Joonis 22 Atribuutidega kohandatud mõõdik

Kood ülal nähtava mõõdiku deklareerimiseks on välja toodud peatükis Lisa 1.

5.3 Nupp

Nupp cdp-button võimaldab saata sõnumi, mille tüüp on täpsustatud atribuudiga *routing*.

```
<cdp-button routing="webApp.CM_SUSPEND" class="ui primary button">
    CM_SUSPEND</cdp-button>
```

Joonis 23 Nupp, mis saadab CDP süsteemi sõnumi

Ülal kirjeldatud nupp saadab CDP rakendusprogrammile nimega webApp sõnumi nimega CM_SUSPEND, mis käsib rakendusel ennast sulgeda.

Kõik *cdp-button* elemendile antud atribuudid antakse edasi sisemisele standardsele HTML elemendile *button*. Seega saab seda pakendit kasutada igal pool, kus eeldatakse tavalist *button* elementi. Ülal joonisel on näiteks kasutatud atribuuti *class*, et muuta nupu stiili kasutades Semantic UI teeki, kuid selle asemel oleks saanud kasutada ka mõnda muud CSS-põhist teeki.

Tabel 4 Elemendi cdp-button atribuudid

Atribuut	Kirjeldus
routing (kohustuslik)	CDP süsteemi komponent, millele saata sõnum, ja sõnumi tüüp.
[kõik muud atribuudid]	Antakse edasi sisemiselt kasutatavale HTML elemendile *button*. Näiteks atribuut *class* võimaldab määrata nupu stiili erinevate CSS-põhiste teekide abil.

5.4 Tekstiväli

Tekstivälja elementi saab kasutada väärtuse kuvamiseks ja muutmiseks. Kui miski muu CDP süsteemis uuendab väärtust, kuvab tekstiväli uue väärtuse. Kui kasutaja sisestab tekstivälja midagi, siis teavitatakse sellest ülejäänud CDP süsteemi.

```
<cdp-input class="ui input" routing="webApp.Description"></cdp-input>
```

Joonis 24 Märgend tekstivälja deklareerimiseks

Sarnaselt nupu elemendile annab ka tekstiväli kõik atribuudid edasi sisemisele standardsele HTML elemendile, milleks antud juhul on *input*.

Näiteks atribuudi *class* abil saab määrata tekstivälja stiili ning atribuudi *type* abil anda veebilehitsejale vihje, mis tüüpi väärtust oodatakse. Numbrilise välja puhul võidakse kuvada noolekesed väärtuse muutmiseks, tõeväärtuse puhul märkeruut jne.

Tabel 5 Elemendi cdp-input atribuudid

Atribuut	Kirjeldus
routing (kohustuslik)	CDP süsteemis liikuv väärtus, mida kuvada või muuta.
[kõik muud atribuudid]	Antakse edasi sisemiselt kasutatavale HTML elemendile *input*. Näiteks atribuut *class* võimaldab määrata nupu stiili erinevate CSS-põhiste teekide abil ja atribuut *type* luua eri tüüpi välju, näiteks tekst, number, märkeruut.

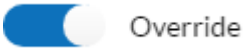
5.5 Lüliti

Lüliti on realiseeritud tekstivälja elemendi baasil. Lihtsalt CSS annab sellele teise kuju. Kõik atribuudid on samad, mis tekstiväljal (peatükk 5.4).

```
<div class="ui toggle checkbox">
    <cdp-input type="checkbox"
               routing="webApp.OverrideStateMachine" />
    <label>Override</label>
</div>
```

Joonis 25 Lüliti on tekstiväli teise välimusega

Tulemus on järgnev:



Joonis 26 Lüliti

5.6 Silt

Sildi direktiiv oskab kuvada andmeid CDP süsteemist tekstina. Tegu võib olla nii numbrilise väärtuse kui sõnega.

```
<h2>CPU Load: <cdp-label routing="webApp.CPULoad"></cdp-label></h2>
```

Joonis 27 Märgend, et kuvada väärtus tekstina

Nagu ülal olevast näitest näha, saab sildi elementi *cdp-label* vabalt paigutada mõne teise elemendi sisse. Antud juhul *h2* element annab tekstile pealkirjale iseloomuliku suurema ja paksema välimuse.

5.6.1 Täiendused

Kuna *double* tüüpi ujukomaarvude kuvamisel võib tekkida ebamugavalt palju komakohti, mis kasutajat teatud olukordades tegelikult ei huvita, siis antud töös lisati atribuut *precision*, millega saab määrata maksimaalse komakohtade arvu.

5.6.2 Atribuudid

Silti saab seadistada järgnevate valikutega.

Tabel 6 Elemendi cdp-label atribuudid

Atribuut	Kirjeldus
routing (kohustuslik)	CDP süsteemis liikuv väärtus, mida kuvada.
precision	Kui kuvatakse numbrilisi väärtusi, siis see atribuut määrab komakohtade arvu.

5.7 Litsents

Käesoleva töö käigus loodud komponendid on kaitstud MIT litsentsiga, mis lubab toodet täiendada ja tasuta kasutada, ka ärilistel eesmärkidel [25].

6 Hinnang tulemusele

Selles peatükis hinnataks antud töö käigus realiseeritud raamistiku vastavust püstitatud eesmärkidele.

6.1 Vastavus nõuetele

Kõige lihtsam viis antud töö tulemust hinnata on võrrelda seda peatükis 3.4 kirjeldatud nõuete ja eesmärkidega.

6.1.1 Funktsionaalsed nõuded

Vasted kõigile loetletud Qt põhistele vidinatele (nupp, lüliti jne) on loodud ning need oskavad suhelda CDP raamistikuga. Igaühe värvi ja suurust saab muuta. Seega kõik funktsionaalsed nõuded on täidetud.

6.1.2 Mittefunktsionaalsed nõuded

Kõik antud töös realiseeritud komponendid on kasutatavad nii hiire kui sõrmega. Kolmanda osapoole teekide puhul on jälgitud, et kõik neist oleks litsentsi järgi tasuta kasutada, ka ärilistel eesmärkidel.

Et hinnata protsessorikasutuse nõuet, testiti kõiki komponente. Kui 10 koopiat samast elemendist on igaüks seotud 10 Hz sagedusega muutuva signaaliga, peaks protsessorikasutus olema alla 10%. Testid tehti arvutiga, kus on 1,7 GHz Intel 3317U protsessor (toodetud 2012. aastal, maksimaalne energiatarve 17 W) [19].

Tabel 7 Elementide protsessorikasutus

Element	Protsessorikasutus
Nupp	Nupp reageerib ainult hiirega vajutusele, pole väärtust, mis uueneks 10 Hz sagedusega

Lüliti	5%, aga tuleb arvestada, et tavakasutuses lüliti väärtus ilmselt 10 korda sekundis ei muutu. Konstantse väärtuse puhul on protsessorikasutus alla 1%.
Mõõdik	1,5%
Graafik	• 10 graafikut 1 signaaliga – 8% • 1 graafik 10 signaaliga – 4%
Tekstiväli	1,5%
Silt	Alla 1%

Nagu näha, siis enamike elementide puhul jääb protsessorikasutus isegi alla 2%. Erandiks on vaid graafik, mis on ka keerulisem element ja vajab 8% protsessorist. Seega on protsessorikasutuse mittefunktsionaalne nõue täidetud.

6.1.3 Muud püstitatud eesmärgid

Järgnevalt on hinnatud kolme peatükis 3.4.3 välja toodud eesmärki:

1. Loodud komponendid võimaldavad luua deklaratiivselt kasutajaliidese elemente. Ei väärtuste kuvamiseks ega muutmiseks pole vaja kirjutada protseduurilist koodi.

2. Loodud komponendid on õhukesed pakendid, mis toetuvad tugevalt kolmanda osapoole teekidele. Täpsemalt on seda analüüsitud peatükis 6.2.

3. Et veebilehe suudaks valmis teha inimene, kellel on olemas baasteadmised HTML-ist, aga ei ole tuttav modernsete Javascript teekidega nagu Angular, on loodud lehe mall, mis kirjeldab kõik sõltuvused, sealhulgas Angular raamistik ning antud töö käigus loodud Angular direktiivid ja komponendid. Seejärel saab kasutaja *body* elemendi sisse kirjutada talle tuttavaid HTML märgendeid, tal on lihtsalt võimalus deklareerida ka antud töö käigus loodud elemente (nt nupp, mis saadab sõnumi CDP süsteemi).

6.2 Välimuse uuendamine

Selleks, et tarkvara püsiks konkurentsis, tuleb seda pidevalt uuendada. Eriti kasutajaliides peab kaasas käima üldiste disainitrendidega. Üks antud töö eesmärkidest oli, et selle integreerimine mõne uue kasutajaliidese teegiga oleks võimalikult lihtne.

6.2.1 Põhielemendid

Kui rääkida töö käigus loodud lihtsamatest elementidest nagu nupp, lüliti, silt, tekstiväli, siis neid saab juba praegu siduda atribuudi *class* abil erinevate CSS-põhiste teekidega nagu Semantic UI või Bootstrap. Seega on funktsionaalsus lahti seotud välimusest. Esimene on realiseeritud Javascript abil, teine CSS abil. Seetõttu võib eeldada, et disainimuutuste mõttes on neid üpriski kerge ajakohasena hoida.

6.2.2 Graafik ja mõõdik

Kaks keerulisemat elementi, mis on antud töös realiseeritud teegi osa, on graafik ja mõõdik. Nende probleem on see, et suur osa välimust mõjutavaid tegureid tuleb ette anda Javascript koodi abil. Sellegipoolest on antud töös kasutatud kolmanda osapoole komponentidele üritatud luua võimalikult õhuke pakend. Spetsiifilised valikud saab ette anda atribuudi *options* abil, mis võtab vastu Javascript objekti. Kui muuta kasutatavat kolmanda osapoole teeki, siis muutuks lihtsalt *options* atribuudi sisu.

Kasutajale aga oleks see oluliselt suurem muudatus, kui põhielementide uuendamine (peatükk 6.2.1). Viimasel juhul piisaks sellest, kui lihtsalt asendada üks CSS-põhine raamistik teisega, näiteks eemaldada Bootstrap ja lisada Semantic UI. Seejärel saaks otsi ja asenda põhimõttel muuta Bootstrap spetsiifilised *class* atribuudi sisud Semantic UI teegile vajalikeks. Graafiku ja mõõdiku puhul aga *options* atribuudi sisuks on suur ja keeruline Javascript keele objekt, mis on tõenäoliselt eri teekides liiga erinev otsi ja asenda meetodi kasutamiseks.

6.2.3 Järeldused

Antud töös realiseeritud põhielemente on üpriski kerge hoida maailma disainitrendidega ajakohasena, piisab vaid ühe CSS-põhise raamistiku asendamisest teisega.

Keerulisemad elemendid, nagu graafik ja mõõdik, vajavad kasutajalt suuremat vaeva, sest nendel teekidel on väga palju erinevaid valikuid, mis tõenäoliselt eri teekide vahel ei ühti.

6.3 Probleemid

Tarkvaraarenduses tuleb tihti teha kompromisse, üks lahendus ei sobi kõigiks juhtudeks. Järgnevalt on välja toodud antud töös kasutatud lahenduste võimalikud puudused.

6.3.1 Graafikuteek

Nagu peatükis 4.2 mainitud, oli valida erinevate kolmanda osapoole loodud graafikuteekide vahel. Antud töös otsustati reaalajaandmete kuvamiseks mõeldud Smoothie Charts teegi kasuks, mis on sobilik põhilise kasutusjuhu jaoks, milleks on reaalajas uuenevate andmete jälgimine. Samas ei ole see hea, kui on vaja analüüsida staatilisi andmeid, näiteks peatades korraks graafiku uuenemise, et uurida ühte lõiku. Põhjused on järgnevad.

- Hiirega peale liikudes ei ilmu kohtspikker täpse väärtusega antud ajahetkel.
- Pole võimalik kerida ajas edasi ja tagasi.
- Pole võimalik dünaamiliselt suurendada või vähendada graafiku telgede vahemikke. Vahel võib kasutaja soovida näha ülevaatliku pilti üle pika ajaperioodi või vastupidi sisse suurendada ühele konkreetsele hetkele.

6.3.2 Alusraamistik

Ehkki loodud lahendus on võimaldab kergesti kaasas käia disainitrendidega nagu kirjeldatud peatükis 6.2, on see siiski väga tugevalt seotud alusraamistikuga, milleks on Angular. Ehkki see on praegusel hetkel üpriski populaarne, muutub Javascripti maailm kiiresti ja on oht, et aastate pärast on see oma tuntuse kaotanud.

Probleem pole niivõrd oluline sellele sihtrühmale, kellel on vaid baasteadmised HTML-ist. Nemad saavad endiselt kasutada ette antud malli, kuhu võib kirjutada tavalisi HMTL märgendeid. Lihtsalt lisaks standardsetele elementidele võib kasutada ka antud töös realiseeritud elemente. Ka kogenud veebiarendajad ilmselt suudaks näidete põhjal Angular põhitõed kiiresti omandada.

Suurem oht on see, et aegunud raamistiku baasil mõne aja pärast kogukond enam ei aita seda täiendada. Üks antud töö eesmärke on luua avatud lähtekoodiga lahendus, mida kogukond saaks hiljem edasi arendada.

Samas positiivne on see, et kui millalgi minna üle uuele alusraamistikule, siis tänu antud töös kirjeldatud kogemusele, läheks sellise raamistiku loomine järgmine kord palju kiiremini.

6.4 Järeldused

Loodud raamistik vastab praegustele nõuetele ja on üpriski vastupidav disainitrendide muudatustele. Suurim oht on see, et alusraamistik Angular kaotab oma tuntuse aastate jooksul ning võimalik kogukond enam ei pruugi olla huvitatud antud töös realiseeritud komponentide täiendamisest ja edasi arendamisest.

7 Võimalikud edasiarendused

Kõik antud töö käigus loodud komponendid on avatud lähtekoodiga, et kogukond saaks vajadusel neid täiendada ja parendada.

7.1 Komponentide lisamine

Antud töö käigus selgus näiteks, et erinevatel graafikutel ja mõõdikutel on kõigil omad head ja halvad küljed. Pole ühte lahendust, mis sobiks kõigile kasutusjuhtudele ideaalselt. Üks võimalik edasiarendus oleks pakkuda raamistikuga rohkem valmis komponente, näiteks mitut eri tüüpi graafikuteeke, et kasutajal oleks rohkem valikut. Vajaduse korral võib ka lisada käesolevas töös nimetamata kasutajaliidese elemente.

7.2 Kasutajaliidese loomine koodi kirjutamata

Antud töö tulemusena valmisid direktiivid, mida oma HTML failis kokku ladudes võib kiirelt ja lihtsalt luua juhtsüsteemi.

Algaja kasutaja jaoks saaks asja veelgi lihtsamaks teha, kui tal oleks võimalik kasutajaliidese elemente kokku pukseerida, ilma et ta peaks kordagi avama mõne HTML või Javascript faili.

Kuna baasraamistik on olemas, oleks seda üpriski lihtne realiseerida. Vaja on ainult tuvastada asukoht, kuhu element pukseeriti, ja lisada antud töö käigus loodud sobiv element HTML failis õigesse kohta.

8 Kokkuvõte

Antud töö eesmärgiks oli luua uus juhtsüsteemile kasutajaliidese loomise raamistik, mis toetaks laia valikut seadmeid ja mille välimus oleks võimalikult palju lahus funktsionaalsuseks. Seotud töödest saadi kinnitust, et hea lähenemine on kasutada veebipõhiseid tehnoloogiaid.

Töö käigus jõuti järeldusele, et:

- Mitmed veebitehnoloogiaid kasutatavad mõõdikute ja graafikute teegid on mõeldud staatiliste andmete kuvamiseks ning reaalajas uuenevate andmetega need koormavad liigselt protsessorit. Vaid vähesed teegid on sobilikud sedalaadi kasutusjuhule.

- Modernsed Javascripti raamistikud – nagu Angular, React, Vue.js – kõik võimaldavad luua pakendeid kasutajaliidese elementidele nii, et juhtsüsteemiga suhtlemiseks piisab deklaratiivsest koodist.

- CSS-põhiseid kasutajaliidese teeke on võimalik Angular raamistikuga integreerida nii, et välimus ja funktsionaalsus on täielikult lahus. Käesoleva töö käigus loodud Angular direktiivid ei sõltu CSS-põhisest raamistikust, seega on viimane kergesti asendatav disainitrendide muutudes.

Töö tulemusena loodi valmis komponendid, mis pakendades kasutajaliidese elemente, võimaldavad protseduurilist koodi kirjutamata suhelda CDP abil loodud juhtsüsteemiga.

Kasutatud kirjandus

[1] "CDP Studio," CDP Technologies, [Online]. Available: http://cdpstudio.com/. [Accessed 12 03 2017].

[2] "Vallaste e-teatmik," [Online]. Available: http://vallaste.ee/. [Accessed 12 03 2017].

[3] "Qt | Cross-platform software development for embedded & desktop," Qt, [Online]. Available: https://www.qt.io/. [Accessed 21 04 2017].

[4] K. Viikki and J. Palviainen, "Integrating Human-Centered Design into Software Development: An Action Research Study in the Automation Industry," in *2011 37th EUROMICRO Conference on Software Engineering and Advanced Applications* , 2011.

[5] M. J. Kim, M. W. Oh, M. E. Cho, H. Lee and J. T. Kim, "A Critical Review of User Studies on Healthy Smart Homes," *Indoor and Built Environment,* 7 12 2012.

[6] B. Petelj, M. Pandurov, D. Stefanović and I. Papp, "Web based solution for smart home functionality extension and control," in *2015 IEEE 5th International Conference on Consumer Electronics - Berlin (ICCE-Berlin)* , Berlin, 2015.

[7] "Square Kilometre Array," [Online]. Available: http://skatelescope.org/. [Accessed 19 03 2017].

[8] A. Marassi, G. Brajnik, M. Nico, V. Alberti and G. L. Roux, "A user interface framework for the Square Kilometre Array: concepts and responsibilities," in *Software and Cyberinfrastructure for Astronomy IV*, Edinburgh, 2016.

[9] "Arduino - Introduction," Arduino AG, [Online]. Available: https://www.arduino.cc/en/Guide/Introduction. [Accessed 20 03 2017].

[10] "Arduino Playground," Arduino AG, [Online]. Available: http://playground.arduino.cc/Main/InterfacingWithSoftware. [Accessed 20 03 2017].

[11] "Jubito," [Online]. Available: http://jubito.org/. [Accessed 20 03 2017].

[12] "MegunoLink," [Online]. Available: http://www.megunolink.com/. [Accessed 20 03 2017].

[13] "Control Your Arduino From Your PC With the Qt Gui," [Online]. Available: http://www.instructables.com/id/Control-your-arduino-from-your-PC-with-the-Qt-Gui/. [Accessed 20 03 2017].

[14] "LabVIEW System Design Software," National Instruments, [Online]. Available: http://www.ni.com/labview/. [Accessed 21 03 2017].

[15] "LabVIEW Web UI Builder," National Instruments, [Online]. Available: http://www.ni.com/uibuilder/. [Accessed 20 03 2017].

[16] "Usage of Silverlight for websites," W3Techs, [Online]. Available: https://w3techs.com/technologies/details/cp-silverlight/all/all. [Accessed 20 03 2017].

[17] "Integrating NI Data Dashboard for LabVIEW into your LabVIEW Applications," National Instruments, [Online]. Available: http://www.ni.com/white-paper/14689/en/. [Accessed 21 03 2017].

[18] "Sense HAT - Raspberry Pi," [Online]. Available: https://www.raspberrypi.org/products/sense-hat/. [Accessed 20 04 2017].

[19] "Intel Product Specifications," [Online]. Available: https://ark.intel.com/products/65707/Intel-Core-i5-3317U-Processor-3M-Cache-up-to-2_60-GHz. [Accessed 30 04 2017].

[20] "Creative Commons Legal Code," [Online]. Available: https://creativecommons.org/licenses/by/3.0/legalcode. [Accessed 25 04 2017].

[21] "Using Google Charts | Charts | Google Developers," [Online]. Available: https://developers.google.com/chart/interactive/docs/. [Accessed 28 04 2017].

[22] "D3.js - Data-Driven Documents," [Online]. Available: https://d3js.org/. [Accessed 25 04 2017].

[23] "C3.js | D3-based reusable chart library," [Online]. Available: http://c3js.org/. [Accessed 25 04 2017].

[24] "The 3-Clause BSD License | Open Source Initiative," [Online]. Available: https://opensource.org/licenses/BSD-3-Clause. [Accessed 25 04 2017].

[25] "The MIT License | Open Source Initiative," [Online]. Available: https://opensource.org/licenses/MIT. [Accessed 25 04 2017].

[26] "Angular C3 Simple," [Online]. Available: https://github.com/wasilak/angular-c3-simple. [Accessed 25 04 2017].

[27] M. Bostock, "Path Transitions," [Online]. Available: https://bost.ocks.org/mike/path/. [Accessed 25 04 2017].

[28] "RGraph demo: A scrolling SVG Line chart," [Online]. Available: https://www.rgraph.net/demos/svg-line-dynamic.html. [Accessed 25 04 2017].

[29] "Smoothie Charts: A JavaScript Charting Library for Streaming Data," [Online]. Available: http://smoothiecharts.org/. [Accessed 26 04 2017].

[30] "justGage.com," [Online]. Available: http://justgage.com/. [Accessed 27 04 2017].

[31] "JustGage Issue #237 Can I disable any animations?," [Online]. Available: https://github.com/toorshia/justgage/issues/237. [Accessed 27 04 2017].

[32] "gauge.js," [Online]. Available: http://bernii.github.io/gauge.js/. [Accessed 28 04 2017].

[33] "AngularJS Material - Introduction," [Online]. Available: https://material.angularjs.org/latest/. [Accessed 07 05 2017].

[34] "Bootstrap · The world's most popular mobile-first and responsive front-end framework.," [Online]. Available: http://getbootstrap.com/. [Accessed 07 05 2017].

[35] "Semantic UI," [Online]. Available: https://semantic-ui.com/. [Accessed 07 05 2017].

Lisa 1 – Atribuutidega kohandatud mõõdik

Peatükis 5.2.2 (lk 43) joonisel nähtava mõõdiku saab deklareerida järgnevate ridadega.

```
<cdp-gauge width="200px" min-value="0" max-value="1"
           display-text-field="false" routing="webApp.CPULoad"
           options='{
               "angle": 0,
               "staticLabels": {
                    "font": "18px sans-serif",
                    "labels": [0, 0.4, 0.7, 1],
                    "fractionDigits": 1
               },
               "staticZones": [
                   {"strokeStyle": "green", "min": 0, "max": 0.4},
                   {"strokeStyle": "yellow", "min": 0.4, "max": 0.7},
                   {"strokeStyle": "red", "min": 0.7, "max": 1}
                ]
           }'
></cdp-gauge>
```